Joni Korkalainen

Rikkaiden WWW-sovellusten luominen

Tietotekniikan pro gradu -tutkielma

15. joulukuuta 2014

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Joni Korkalainen

Yhteystiedot: joni.p.korkalainen@student.jyu.fi

Ohjaaja: Vesa Lappalainen

Työn nimi: Rikkaiden WWW-sovellusten luominen

Title in English: Rich Internet Application development

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Sivumäärä: 125+191

Tiivistelmä: WWW-sovellusten luominen on nykyään entistä monimutkaisempaa rikkaiden WWW-sovellusten yleistymisen myötä. Tällaisten WWW-sovellusten käytettävyys vastaa perinteisiä työpöytäsovelluksia. Erilaisia WWW-sovelluskehyksiä näiden sovellusten luomisen helpottamiseksi on tarjolla runsaasti ja niiden väliltä valitseminen voi olla hankalaa. Tässä tutkielmassa vertailtiin neljää erilaista sovelluskehystä (Vaadin, ASP.NET, Ruby on Rails ja Seaside) keskenään toteuttamalla niillä Ohjelmointi 2 -kurssin malliharjoitustyön (Kerho-sovelluksen) WWW-versio. Kehyksiä verrattiin erityisesti rikkaiden WWW-sovellusten luomisen perusteella. Huomattiin, että Vaadin, joka on tarkoitettu erityisesti rikkaiden WWW-sovellusten luomiseen, pärjäsi vertailussa parhaiten. Muilla sovelluskehyksillä tuli vastaan erilaisia ongelmia. Ruby on Rails pärjäsi kaikkein huonoimmin, vaikka se olikin ainoa kehys Vaadin-kehyksen lisäksi, jolla saatiin luotua täysin suunnitelman mukainen sovellus.

Avainsanat: Rikas WWW-sovellus, WWW-sovelluskehys, Ajax, Vaadin, ASP.NET, Ruby on Rails, Seaside

Abstract: Developing web-applications is getting more challenging due to popularity of rich internet applications (RIAs). Usability of these applications is similar to traditional desktop applications. A lot of different web-application frameworks

exists, that makes it easier to develop these kind of applications. However, deciding between these frameworks can be difficult. Four different kind of web-frameworks were compared in this thesis (Vaadin, ASP.NET, Ruby on Rails and Seaside). A web-application for a practical work -example of a programming course were build with each of them. The frameworks were compared especially by their ability to build RIAs. Vaadin, which is designed especially for developing RIAs, was noticed to be the best framework of the comparison. Ruby on Rails was the worst, even though it was the only framework besides Vaadin, that made it possible to build an application that worked entirely according to the plan.

Keywords: Rich Internet Application, Web application framework, Ajax, Vaadin, ASP.NET, Ruby on Rails, Seaside

Termiluettelo

.NET	Microsoftin kehittämä sovelluskehys.
Ajax	*Asynchronous JavaScript And XML*: Joukko WWW-sovelluskehityksen tekniikoita, joiden avulla WWW-sovelluksista voi tehdä vuorovaikutteisempia.
API	*Application Programming Interface* : Ohjelmointirajapinta, jonka avulla voidaan käyttää tiettyä ohjelmistokomponenttia.
ASP	*Active Server Pages* : Microsoftin kehittämä tapa dynaamisten WWW-sivujen luomiseen.
C#	Microsoftin kehittämä ohjelmointikieli.
CSS	*Cascading Style Sheets*: Kieli, joka määrittelee WWW-sivujen visuaalisen tyylin.
CRUD	*Create, Read, Update and Delete*: Neljä perustoimintoa tietokantojen tiedoille.
DLL	*Dynamic Link Library*: Windows-sovelluksen käyttämä kirjasto, joka sisältää suoritettavia funktioita tai dataa.
DOM	*Document Object Model*: Ohjelmointirajapinta, joka mahdollistaa (X)HTML-dokumenttien sisällön muokkauksen.
Gem	Ohjelmakirjasto, joka laajentaa Ruby-kieltä.
GWT	*Google Web Toolkit*: WWW-sovelluskehys monimutkaisten selainpuolen sovellusten luomiseen Java-koodilla.
HTML	*Hypertext Markup Language*: WWW-sivujen luomiseen käytettävä merkintäkieli.
HTTP	*HyperText Transfer Protocol*: WWW:n käyttämä protokolla, joka määrittää kuinka viestit lähetetään.
JAR	*Java Archive*. Javan pakkausformaatti. JAR-tiedostoilla on `.jar`-tiedostopääte
Java	Olio-ohjelmointikieli.
Java Servletti	Java-luokka, joka laajentaa palvelimen ominaisuuksia.

Javascript	Oliopohjainen skriptikieli.
JSON	*JavaScript Object Notation*: Kevyt dataformaatti, joka perustuu Javascript-kielen osajoukkoon.
JSP	*Java Server Pages*: Tapa luoda dynaamisia WWW-sivuja sisällyttämällä Java-koodia HTML-koodin sekaan.
MVC	*Model–View–Controller*: WWW-sovelluksissa käytetty arkkitehtuurimalli, jossa sovellus jaetaan kolmeen osaan: Malli (data), näkymä (käyttöliittymä) ja ohjain (sovelluslogiikka).
ORM	*Object-Relational Mapping*: Mekanismi, jolla voi käyttää ja muokata olioita ilman että tarvitsee huolehtia kuinka ne on yhteydessä niiden dataan.
PHP	*PHP: Hypertext Preprocessor*: Palvelinpuolen skriptikieli.
Python	Olio-ohjelmointikieli.
REST	*Representational State Transfer*: Arkkitehtuurityyli, jossa resursseja pyydetään ja palautetaan neljän HTTP-operaation kautta: GET, POST, PUT ja DELETE.
RPC	*Remote Procedure Call*: Mekanismi asiakkaan ja palvelimen väliseen vuorovaikutukseen verkon kautta.
Ruby	Olio-ohjelmointikieli.
SCSS	Tyylitiedostoformaatti, joka on ylijoukko CSS3-formaatista.
Sessio	Mahdollistaa tiedon liittämisen yksittäiseen WWW-sovelluksen käyttäjään.
Smalltalk	Olio-ohjelmointikieli.
SQL	*Structured Query Language*: Tietokannan kyselykieli.
URL	*Uniform Resource Locator*: Tietyn WWW-sivun tai tiedoston osoite Internet-verkossa.
Visual Basic	Microsoftin kehittämä ohjelmointikieli.
XHTML	*Extensible Hypertext Markup Language*: WWW-sivujen luomiseen käytettävä merkintäkieli, joka täyttää XML:n muo-

	tovaatimukset.
XML	*Extensible Markup Language*: Kieli, jolla voi kuvata tietoa ja määritellä tiedolle rakenteen.
XUL	*XML User Interface Markup Language*: Mozillan kehittämä käyttöliittymän kuvauskieli.
YAML	*YAML Ain't Markup Language*: Datan sarjallistamis-standardi kaikille ohjelmointikielille.
ZUML	*ZK User Interface Markup Language*: Käyttöliittymän kuvauskieli, joka perustuu XUL-kieleen.
WSDL	*Web Services Description Language*: XML-muotoiltu kieli WWW-palveluiden kuvaamiseen.
WWW	*World Wide Web*: Internet palvelinten järjestelmä, joka tukee HTML-dokumentteja.

Kuviot

Taulukot

Sisältö

1 Johdanto

WWW (*World Wide Web*) [12] -sovellukset on perinteisesti mielletty erilaisiksi verrattuna työpöytäsovelluksiin, joissa käyttöliittymää päivitetään vain tietty osa kerrallaan ja sovelluksen vasteet käyttäjille ovat nopeita. Perinteiset WWW-sovellukset noudattavat pyyntö-vastaus -mallia, jossa käyttäjä joutuu aina odottamaan palvelimen vastausta jokaisen toiminnon jälkeen (koko sivu ladataan uudelleen). Tästä syystä käyttäjät kokevat ne erilaisiksi verrattuna työpöytäsovelluksiin eikä tällaisten sovellusten käyttäminen aina ole kovin käyttäjäystävällistä. Tällaiset WWW-sovellukset ovat jatkuvasti vähenemässä niin sanottujen rikkaiden WWW-sovellusten yleistymisen myötä. [72]

Internet onkin nykyään suosittu paikka julkaista uusia sovelluksia. Tulevaisuudessa (ja miksei jo nykyäänkin) suurin osa sovelluksista julkaistaan tietyn käyttöjärjestelmän sijasta Internettiin käytettäväksi WWW-selaimen kautta. Monia työpöytätyylisiä WWW-sovelluksia on jo olemassa, kuten tekstinkäsittelyä ja taulukkolaskentaa, ja työpöytäsovelluksista tuttuja rikkaita käyttöliittymiä näkee useimmissa WWW-sovelluksissa. Näiden luomisen on mahdollistanut erilaiset keinot tehokkaamman kommunikaation luomiseen asiakkaan (WWW-selain) ja palvelimen välille. Niiden myötä WWW-sovellusten luominen on muuttunut entistä monimutkaisemmaksi ja niiden luomisessa on käytettävä monia erilaisia tekniikoita. [26] [63]

WWW-selainten perinteinen käyttöliittymä ei ole juurikaan kehittynyt WWW-sovellusten kehittymisen myötä. Esimerkiksi niiden navigointitoiminnot (takaisin- ja eteenpäin-napit) ovat olleet mukana (ja toimineet samalla tavalla) ensimmäisistä selaimista asti. Tämä asettaa myös haasteita rikkaiden WWW-sovellusten kehittämiselle. [63]

WWW-sivujen kehittämisen monimutkaistuessa on kehitetty erilaisia WWW-sovelluskehyksiä WWW-sovellusten luomista helpottamaan. Niitä onkin nykyään tarjolla runsaasti eri ohjelmointikielille ja niistä valitseminen voi olla hankalaa. Nykyään on olemassa sovelluskehyksiä erityisesti rikkaiden sovellusten luomiseen ja vanhoja

sovelluskehyksiä on päivitetty mahdollistamaan myös rikkaiden sovellusten luominen. Tässä tutkielmassa on tarkoitus tutkia rikkaiden WWW-sovellusten luomista erilaisilla sovelluskehyksillä, ja vertailla tiettyjä sovelluskehyksiä keskenään.

Tässä tutkielmassa vertaillaan neljää erilaista WWW-sovelluskehystä keskenään eri ominaisuuksien suhteen. Vertailu tehdään toteuttamalla sovelluskehyksillä Ohjelmointi 2 -kurssin malliharjoitustyön (Kerho-sovelluksen) WWW-versio. Vertailussa tarkastellaan kuinka sovelluskehyksillä onnistuu rikkaille WWW-sovelluksille ominaisten ja tärkeiden toimintojen luominen. Tämän lisäksi sovelluskehyksiä vertaillaan myös muiden WWW-sovelluskehyksille yleisesti tärkeiden ominaisuuksien perusteella (tietokannat jätetään kuitenkin suosiolla vertailusta pois). Yksi vertailukehyksistä (Vaadin) on suunniteltu juuri rikkaiden WWW-sovellusten kehittämiseen, kun muihin sovelluskehyksiin kyseiset ominaisuudet on lisätty jälkeenpäin.

Tämän tutkielman alussa kerrotaan rikkaista WWW-sovelluksista: Niiden toteutustavoista sekä niistä tulevista hyödyistä ja ongelmista. Tämän jälkeen annetaan esimerkki, kuinka rikas WWW-sovellus saadaan luotua ilman minkäänlaisten sovelluskehysten apua. Tästä saadaan hyvä vertailukohta sille, miten eri sovelluskehykset auttavat näiden luomisessa. Seuraavissa luvuissa esitellään vertailuun tulevat WWW-sovelluskehykset: Vaadin, ASP.NET, Ruby on Rails ja Seaside. Näiden valinnassa pyrittiin saamaan erilaisia tapoja rikkaiden WWW-sovellusten luomiseen, mutta myös valitsemaan jonkin verran suosittuja sovelluskehyksiä. Näiden jälkeen esitellään vielä muutamia muita WWW-sovelluskehyksiä, jotka tarjoavat erilaisia keinoja WWW-sovellusten luomiseen. Tämän tarkoituksena on muistuttaa myös muiden sovelluskehysten olemassaolosta, ja esitellä erilaisia lähestymistapoja WWW-sovellusten luomiselle. Lopuksi kuvataan eri sovelluskehyksillä luodut kerhosovellukset, esitetään esimerkki sovelluskehysten käytöstä ja vertaillaan sovelluskehyksiä eri ominaisuuksien suhteen. Liitteissä A, B, C ja D kuvataan kerhosovellusten luominen eri sovelluskehyksillä ja liitteistä E, F, G ja H löytyy kerhosovellusten lähdekoodit. Lisäksi liitteistä I, J, K, L ja M löytyy tutkielmassa esitettävien valuuttamuunnin-esimerkkien lähdekoodit.

2 Rikkaat WWW-sovellukset

Perinteiset WWW-sovellukset käyttävät asiakas-palvelin-arkkitehtuuria, joissa koko sivu päivitetään uudelleen jokaisen pyynnön jälkeen: Palvelin palauttaa asiakkaalle aina koko WWW-sivun. [49] [26, s. 413] Rikkaat WWW-sovellukset mahdollistavat sivun osittaisen päivittämisen: Tietyn elementin (tai komponentin) sisältö päivitetään tai koko elementti (tai komponentti) korvataan toisella. Rikkaat WWW-sovellukset tarjoavat paremman käytettävyyden ja vastaavat toiminnallisuudeltaan vastaavia työpöytäsovelluksia. Nämä sovellukset eivät käytä WWW-sivuille tyypillistä linkki-navigaatiota vaan sivut päivitetään esimerkiksi tapahtumien kautta, aivan kuten työpöytäsovelluksissa. [63, s. 294]

2.1 Eri tapoja luoda rikkaita WWW-sovelluksia

Rikkaat WWW-sovellukset voidaan jakaa kolmeen kategoriaan sen mukaan kuinka ne on luodaan.

Ensimmäinen tyyppi on liitännäisiin (engl. *plug-in*) perustuvat sovellukset, joissa sovellus luodaan tietylle alustalle kuten Adobe AIR, Microsoft Silverlight, JavaFX tai Java Applet. Adobe AIR -sovellukset ohjelmoidaan ActionScript-kielellä, Silverlight on .NET [47, s. 3-4] -perustainen ja kaksi viimeistä tarjoavat nimensä mukaisesti alustan Java [6, s. 4] -sovelluksille. Alustojen etuna on, ettei palvelin- ja asiakaspuolia tarvitse ohjelmoida erikseen. Niissä sovellus ohjelmoidaan palvelimelle, ja sama sovellus voidaan ajaa asiakkaan puolella ilman erillistä asiakaspuolen koodausta. Ne toimivat selaimessa liitännäisen avulla, joka siis pitää olla selaimessa asennettuna. [26, s. 415] [51, s. 1367]

Toinen tyyppi (ja vähiten käytetty) on selainpohjaiset sovellukset. Ne käyttävät yleensä tiettyä käyttöliittymäkieltä (luotu XML-kielellä (*Extensible Markup Language*) [73, s. 2]), joka antaa kehittäjän määritellä elementit ja niiden vuorovaikutuksen. Esimerkiksi Mozilla-säätiön XUL (*XML User Interface Markup Language*) [57, s. 7] on tällainen kieli. [26, s. 415]

Kolmas tyyppi on uusin, ja myös se, johon tässä tutkielmassa keskitytään. Ajax-tekniikalla (*Asynchronous JavaScript And XML*) [34, s. 67] luotuja sovelluksia sanotaan myös skripti-perustaisiksi. Ne yhdistävät eri tekniikoita, joihin yleensä kuuluu XHTML/HTML (*(Extensible) Hypertext Markup Language*) [34, s. 66], CSS (*Cascading Style Sheets*) [34, s. 66], DOM (*Document Object Model*) [34, s. 66] ja Javascript [63]. HTML- ja CSS-kielillä luodaan ulkoasu, Javascriptillä asynkroniset pyynnöt palvelimelle ja DOM:in avulla hoidetaan käyttöliittymän päivittäminen. Tämä tekniikka ei vaadi selaimeen mitään liitännäisiä, mutta Javascript pitää olla käytössä. Kehittäjän ei kuitenkaan enää välttämättä tarvitse osata kaikkia edellä mainittuja tekniikoita sillä Ajax-sovellusten luomiseen on kehitetty erilaisia työkaluja. Useimmat WWW-sovelluskehykset tarjoavat keinoja Ajax-toimintojen luomiseen ja osa sovelluskehyksistä myös tekee kaiken täysin automaattisesti. Lisäksi erilaiset Javascript-kirjastot helpottavat Ajaxin käyttöä. [26, s. 415]

Tässä tutkielmassa keskitytään jatkossa ainoastaan Ajax-sovelluksiin. Taulukossa 1 esitetään vertailuun valittujen sovelluskehysten julkaisu ja päivitystietoja sekä niillä luotuja sovelluksia. Nämä sovelluskehykset siis tarjoavat erilaisia keinoja Ajax-toiminnallisuuksien luomiseen. Taulukossa mainittu nykyinen versio sovelluskehyksestä, on myös se versio, joka tässä tutkielmassa esitellään, ja jolla vertailusovellus luodaan.

2.2 Rikkaiden WWW-sovellusten tuomat hyödyt

Rikkaat WWW-sovellukset mahdollistavat sivun osittaisen päivittämisen, ilman että koko sivu ladataan uudelleen. Tämä parantaa sivun reagoivuutta, interaktiivisuutta ja käytettävyyttä. [49] Käyttäjälle ei tarvitse esittää uutta sivua aina napin painalluksen jälkeen, vaan muutokset voidaan päivittää nykyiseen sivuun. Käyttäjän ei myöskään tarvitse painaa nappia tai klikata linkkiä, jotta tiedot lähetetään palvelimelle. Näin esimerkiksi käyttäjän syötteet tekstikenttään voidaan tarkistaa jo ennen niiden lähettämistä (esimerkiksi kun käyttäjä poistuu tekstikentästä), ja näyttää ilmoitus, jos niissä on virheitä. [72] Rikkaat WWW-sovellukset voivat myös vähentää verkkoliikenteen määrää, koska dataa lähetetään vain tarpeen mukaan, eikä palvelimen

Taulukko 1: Vertailtavien sovelluskehysten perustiedot [55] [70] [58] [59] [60] [44] [41] [61] [34, s. 26-27] [47, s. 12] [25, luku 1.2]

Kehys	**1. versio ja julkaisuvuosi**	**Nykyinen versio**	**Viimeisin päivitys**	**Tunnetuimpia sovelluksia**
Vaadin	IT Mill Toolkit Release 4, 2006	Vaadin 7	Vaadin 7.3.5, 18.11.2014	- Puma Range Toolbox - Aktia Forms Generator - ViLLE - SentiOne - Motonet Webstore
ASP.NET	ASP.NET 1.0, 2000	ASP.NET 4.5	VS2013 Update 4 ja VS2015 Preview, 12.11.2014	- Ei tietoa.
Ruby on Rails	Rails 1.0, 2005	Rails 4.1	Rails 4.1.8, 16.11.2014	- Twitter (ei enää käytä) - Shopify - Bleacher Report - Heroku - Github
Seaside	Seaside 0.9	Seaside 3.1	Seaside 3.1, 27.12.2013	- Reserve Travel - Cmsbox - Yesplan

tarvitse aina palauttaa koko sivua [26].

2.3 Rikkaiden WWW-sovellusten tuomat ongelmat (ja mahdollisia ratkaisuja)

Ajax-sovellusten luomisessa ongelmana on eri selainten Javascript-toteutuksien erot: Toiset selaimet tarjoavat täydellisempiä toteutuksia suuremmalla määrällä ominaisuuksia, kun taas toissa selaimissa toteutukset ovat epätäydellisiä eivätkä noudata standardeja. Kehittäjien tuleekin olla tarkkana, että sovellus toimii suosituimmilla selaimilla. Vaikka koodi toimisikin, voi sen käyttäytymisessä olla eroja eri selaimilla, minkä vuoksi täysin samanlaisen toiminnallisuuden luominen eri selaimille on hankalaa. Javascript-kirjastojen avulla nämä erot selaimissa saadaan kuitenkin piilotettua. Suosituimpiin Javascript-kirjastoihin kuuluu nykyään esimerkiksi Prototype, Scriptaculous (Prototypen lisäosa), Dojo ja jQuery. [62, s. 367-368] [26, s. 415] [40, s. 35, 95]

Taivalsaaren ym. [63, s. 297] mukaan WWW-selaimilla on myös ominaisuuksia, jotka eivät välttämättä ole käytettäviä rikkaissa WWW-sovelluksissa. Esimerkiksi “takaisin-” ja “eteenpäin-” nappien käyttö ei ole niin selvää sovelluksille, joilla on monimutkainen sisäinen tila, ja jotka kommunikoivat dynaamisesti palvelimen kanssa.

Vaikka kyseiset napit olisivatkin käytettäviä, eivät ne välttämättä toimi oikein, koska Ajax-päivitykset eivät muuta WWW-sivun osoitetta. Vaikka käyttäjä olisi painanut sovelluksessa useita painikkeita ja sivun käyttöliittymä olisi muuttunut, selaimen takaisin napin painaminen voi viedä käyttäjän kuitenkin kokonaan pois sivulta. Myöskään kirjanmerkit eivät osoita sovelluksen oikeaan tilaan. Käyttäjä voi haluta lisätä kirjanmerkkiin tietyn sovelluksen näkymän, mutta kirjanmerkkiin tuleekin sovelluksen alkutilanne. Nämä ongelmat voidaan korjata esimerkiksi käyttämällä sovelluksen eri tiloille erilaista URL:n (*Uniform Resource Locator*) [64] `#ankkuria`, jonka kautta sovelluksen tila saadaan selville.

Farrellin ym. [26, s. 416-417] mukaan rikkaiden WWW-sovellusten käyttäjät eivät

välttämättä huomaa sivulle tulleita päivityksiä: He eivät ehkä ymmärrä kuinka päivitykset näytetään, joten he eivät osaa paikantaa niitä. Varsinkin jos päivitykset ilmaantuvat eri paikkaan kuin missä käyttäjä on viimeksi toiminut. Tämän ongelman välttämiseksi on hyvä näyttää jonkinlainen ilmoitus onnistuneen prosessin/toiminnon jälkeen.

Toimintojen suoritusajat palvelimella saattavat joskus olla liian pitkiä rikkaille WWW-sovelluksille: Perinteisissä WWW-sovelluksissa toiminnon suorittamisen näkee selaimen ikkunan latautumisesta, kun taas rikkaissa WWW-sovelluksissa toiminnon suorittamista ei huomaa mistään, jos sitä ei erikseen näytä. Tästä syystä sovelluksen käyttäjä saattaa luulla sovelluksen jumiutuneen tai ettei toiminto jostain syystä toimi. Sovelluksen onkin hyvä näyttää käyttäjälle, että toimintoa suoritetaan ja nyt pitää odottaa. Pitkään kestävissä toiminnoissa olisi hyvä myös näyttää toimintojen eteneminen ja mahdollisesti myös kuinka paljon kauemmin niiden suorittaminen vielä kestää. [26, s. 416-417]

3 Rikas WWW-sovellus ilman sovelluskehysten käyttämistä

Tässä kappaleessa kerrotaan kuinka rikas WWW-sovellus toteutetaan ilman minkäänlaisten sovelluskehysten käyttöä. Esimerkissä käytetään kuitenkin apuna jQuery-Javascript-kirjastoa, koska eri WWW-selainten Javascript- ja Ajax-toteutukset eroavat toisistaan. jQuery piilottaa nämä eroavuudet, jolloin joka selaimelle ei tarvitse itse tehdä erikseen koodia.

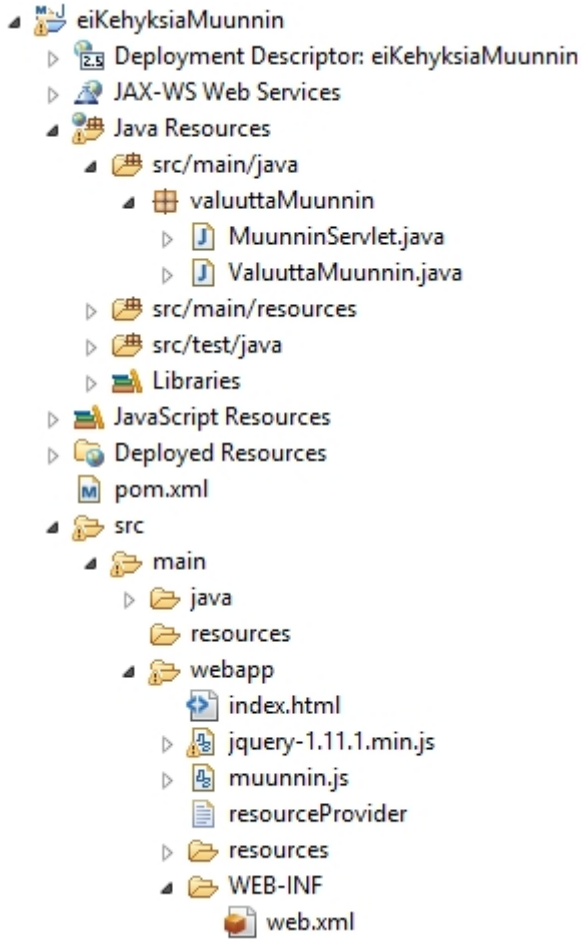
Esimerkkinä luodaan kuvion 1 mukainen valuuttamuunnin. Kuvio 2 esittää sovelluksen projektirakenteen.

Kuvio 1: Valuuttamuunnin-sovelluksen käyttöliittymä

3.1 Käyttöliittymä (HTML)

Sivun käyttöliittymä luodaan HTML-koodilla. Esimerkissä HTML-sivulle luodaan lomakekomponentit ja annetaan niille sopivat ID:t, jotta niihin päästään käsiksi DOM-rajapinnan kautta:

```
<head>
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript" src="muunnin.js"></script>
<title>Valuuttamuunnin</title>
</head>
<body>
<form>
   <p>
      <input type="text" id="textSumma" />
      <select id="selectValuutta"><option value=""> </option></
         select>
      <input type="button" id="buttonMuunna" value="Muunna" />
      <strong id="labelTulos">0 &#8364;</strong>
   </p>
</form>
</body>
```



Kuvio 2: Valuuttamuunnin ilman sovelluskehyksiä -projektirakenne

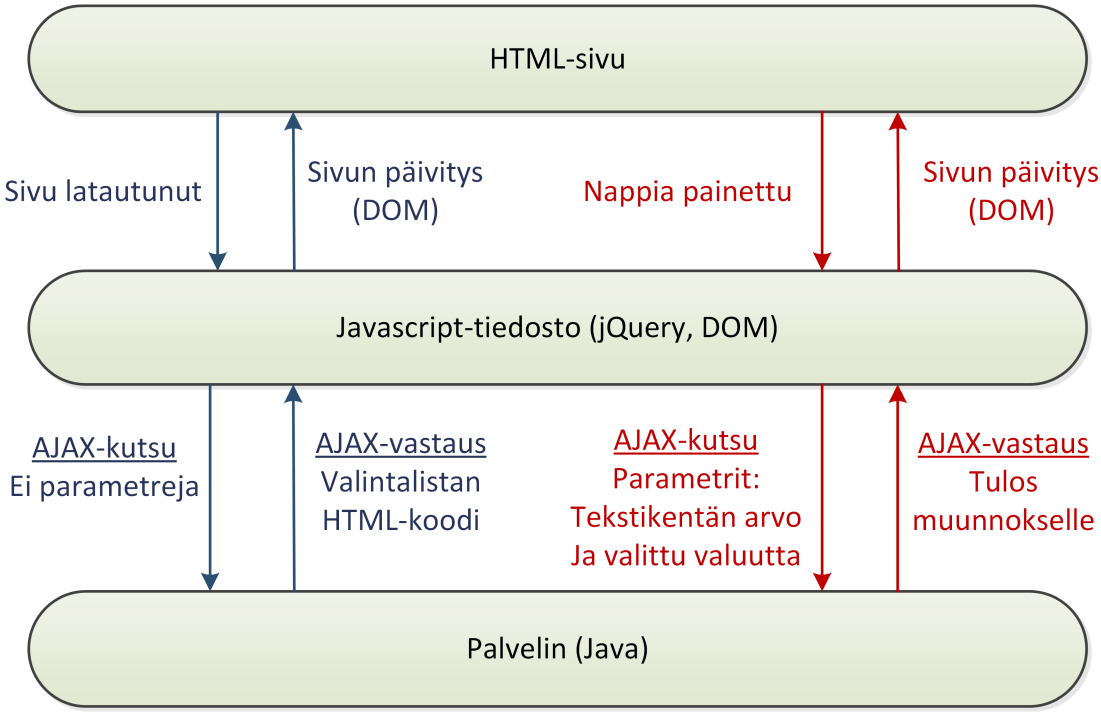
3.2 Ajax-kommunikaatio ja HTML-sivun päivitys

Ajaxin avulla otetaan yhteys palvelimelle, joka palauttaa vastauksen. Tämän jälkeen sivun käyttöliittymä päivitetään DOM-rajapinnan kautta. Esimerkeissä Ajax-pyynnöt ja käyttöliittymän päivittäminen tehdään jQueryn avulla. Kuvio 3 kuvaa sovelluksen osat ja niiden välisen Ajax-kommunikaation.

Sivun latauduttua palvelimelta haetaan Ajaxin avulla valintalaatikon alkiot, joihin palvelin asettaa oikeat valuuttojen nimet. Palvelin palauttaa valintalaatikon sisällön HTML-koodin, joka sitten lisätään DOM-rajapinnan kautta (jQueryn avulla) valintalaatikon sisällöksi (valintalaatikko-elementin HTML-koodi korvataan uudella):

```
$.ajax({
   async: true,
```

```
    url: "valuuttaMuunnin",
    data: "",
    processData: false,
    type: "POST",
    success: function(data, textStatus, request){
       $("#selectValuutta").html(request.responseText);
    }}
);
```



Kuvio 3: Ajax-kommunikaation toteutus ilman sovelluskehysten käyttämistä

Napin painamisen jälkeen lähetetään tekstikentän arvo ja valintalistan valinta Ajax-kutsuna ja saadaan vastauksena tulos muunnetulle valuutalle. Tämä päivitetään sitten HTML-sivulle DOM-rajapinnan kautta (jQueryn avulla). Data lähetetään kutsussa *querystring*-muodossa: `parametri=arvo&parametri2=arvo2...`

```
$('#buttonMuunna').click(laheta);

function laheta(e) {
   var summa = $("#textSumma").val();
   var valuutta = $("#selectValuutta").val();
   var tulos;

   $.ajax({
      async: true,
      url: "valuuttaMuunnin",
      data: "summa=" + summa + "&valuutta=" + valuutta,
      processData: false,
      type: "POST",
```

```
      success: function(data, textStatus, request){
         if (request.responseText === "virhe") {
            $("#labelTulos").text("Virheellinen syöte!");
         }
         else {
            tulos = request.responseText + " euroa";
            $("#labelTulos").text(tulos);
         }
      }
   });
}
```

3.3 Ajax-kutsun vastaanotto palvelimella

Palvelin ottaa vastaan Ajax-kutsun ja palauttaa vastauksen asiakkaalle. Lähettäjän lähettämä data saadaan parametrien nimien avulla. Esimerkissä pyyntö otetaan vastaan Javalla *Java Servletin* [34, s. 66-67] avulla. Palvelinpuolen voisi tietysti tehdä muillakin ohjelmointikielillä, ja itse asiassa se voisi ollakin yksinkertaisempi toteuttaa esimerkiksi PHP:llä [10] [48, s. xviii] tai Pythonilla [40, s. 6].

Jos pyynnössä ei ole mukana dataa, palautetaan valintalaatikon alkiot HTML-koodina, johon haetaan valuuttojen nimet `ValuuttaMuunnin`-luokasta metodilla `muunnin.getValuuttaNimet()`. Muuten lasketaan muunnos pyynnössä olevalle datalle (parametrit "`summa`" eli muutettava määrä ja "`valuutta`" eli valuutta josta muutetaan) ja palautetaan saatu tulos:

```
public void doPost(HttpServletRequest request, HttpServletResponse
    response)
      throws ServletException, IOException {

   response.setContentType("text/plain");

   if (request.getParameter("summa") == null) {
      List<String> nimet = muunnin.getValuuttaNimet();
      String selectHTML = "";

      for (String nimi : nimet) {
         selectHTML = selectHTML + "<option value=\"" + nimi + "\">
             " + nimi + "</option>";
      }

      response.getWriter().write(selectHTML);
   }
   else {
      String maara = request.getParameter("summa");
      String valuutta = request.getParameter("valuutta");

      String tulos = muunnin.muunnaValuutta(valuutta, maara);
      response.getWriter().write(tulos);
```

```
    }
}
```

`web.xml`-tiedostossa kerrotaan edellisen *Java Servletin* osoite (`/valuuttaMuunnin`):

```
<servlet>
   <servlet-name>Valuuttamuunnin</servlet-name>
   <servlet-class>valuuttaMuunnin.MuunninServlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>Valuuttamuunnin</servlet-name>
   <url-pattern>/valuuttaMuunnin</url-pattern>
</servlet-mapping>
```

4 Vaadin

Vaadin on WWW-sovelluskehys Java-kielelle, joka on suunniteltu erityisesti rikkaiden WWW-käyttöliittymien luomiseen. Vaadin-kehyksen palvelinpohjaisen ohjelmointimallin avulla WWW-käyttöliittymiä voi luoda kuten perinteisiä Java-työpöytäsovelluksia. Vaadin hoitaa käyttöliittymän hallitsemisen selaimessa ja Ajax-kommunikaation asiakkaan ja palvelimen välillä automaattisesti. Vaadin-kehyksen kanssa ei välttämättä tarvitse opetella eikä luoda normaaleja selainteknologioita, kuten HTML ja Javascript (CSS-kieltä joutuu kuitenkin käyttämään tyylien määrittämisessä). [34, s. 21]

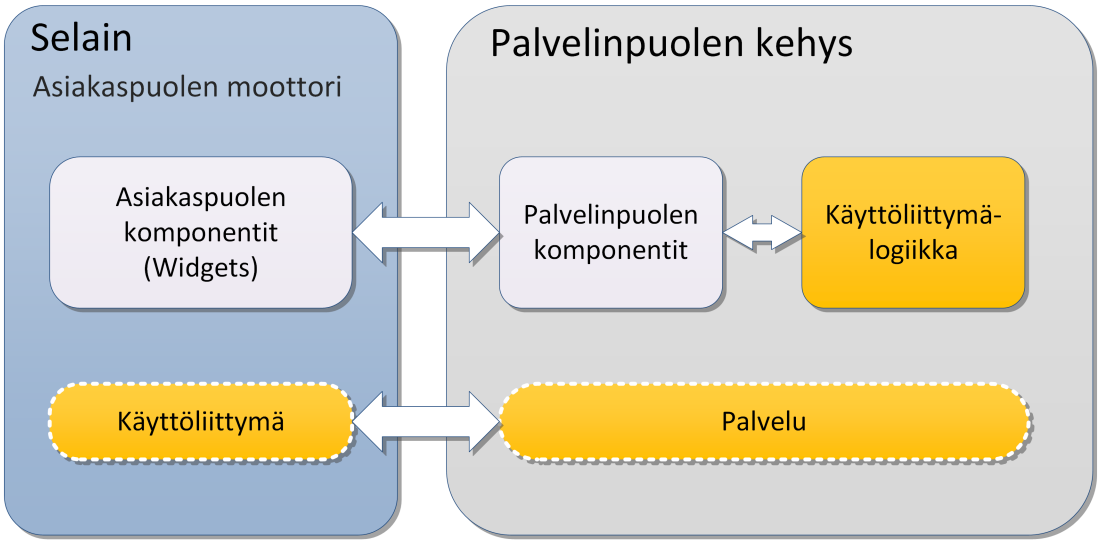
4.1 Vaadin-sovelluskehityksen mallit

Grönroos [34, s. 22, 63] esittelee kaksi mallia Vaadin-sovellusten kehittämiseen: Asiakaspuolen kehitys ja palvelinpuolen kehitys. Palvelinpuolen kehitys mahdollistaa sovellusten luomisen kokonaan palvelimen puolella Vaadin-kehyksen *asiakaspuolen moottorin* avulla. Asiakaspuolen kehitys mahdollistaa asiakaspuolen komponenttien ja toiminnallisuuksien toteuttamisen Java-koodilla, joka käännetään Javascript-koodiksi ja suoritetaan selaimessa. Myös kokonaan asiakaspuolella (WWW-selaimessa) suoritettavien sovellusten luominen on mahdollista. Sovellusten luomisessa voidaan käyttää näitä kahta mallia yhtä aikaa. Vaadin-kehys sisältää palvelinpuolen ja asiakaspuolen APIt (*Application Programming Interface*) [42].

Grönroosin [34, s. 67] mukaan asiakaspuolen kehys perustuu *GWT*-kehykseen (*Google Web Toolkit*) [36] [69] ja se sisältää *Vaadin Compiler* -kääntäjän (laajennus *GWT Compiler* -kääntäjästä), joka mahdollistaa Java-koodin kääntämisen Javascript-koodiksi.

Palvelinpuolen Vaadin-sovelluksen arkkitehtuuri koostuu palvelinpuolen kehyksestä ja asiakaspuolen moottorista (kuvio 4). Sovelluksen käyttöliittymälogiikka suoritetaan *Java Servlettinä* Java-sovelluspalvelimella. [34, s. 22] Palvelinpuolen sovelluksen asiakaspuoli on Javascript-koodia, joka luodaan asiakaspuolen moottorin

avulla palvelinpuolen Java-koodista [28, s. 74]. Asiakaspuoli hallitsee siis sovelluksen renderöinnin selaimessa ja käyttäjän vuorovaikutuksen sovelluksen kanssa. Palvelinpuoli hallitsee tapahtumat, jotka tulevat asiakkaalta, ja välittää tehdyt muutokset käyttöliittymässä asiakkaalle. [29, s. 55]



Kuvio 4: Vaadin palvelinpuolen kehyksen arkkitehtuuri [34, s. 22]

Seuraavat alaluvut kuvaavat Vaadin-kehyksen palvelinpuolen kehitystä.

4.2 Asiakaspuolen moottori

Asiakaspuolen moottori (engl. *Client-Side Engine*) esittää palvelinpuolen sovelluksen käyttöliittymän selaimessa: Palvelinpuolen komponentit esitetään asiakaspuolen komponentteina (engl. *client-side widgets*) HTML-muodossa. *ApplicationConnection*-luokka hoitaa komponenttien esittämisen ja niiden kommunikoinnin palvelimelle. Asiakaspuolen moottori ja sen asiakaspuolen komponentit myös manipuloivat sivun DOM-puuta ja hoitavat tapahtumien synnyttämisen ja välittämisen palvelimelle. Moottori suoritetaan selaimessa Javascript-koodina, joten selaimeen ei tarvitse asentaa mitään lisäosia (engl. *plugin*) Vaadin sovelluksen käyttämiseen. [34, s. 22, 66-69]

4.3 Palvelimen ja asiakkaan välinen kommunikointi

Vaadin käyttää Ajax-tekniikoita, joiden avulla rikkaat käyttöliittymät saadaan aikaiseksi [34, s. 22]. Yhteydenpito asiakkaan ja palvelimen välillä hoidetaan JSON (*JavaScript Object Notation*) [45] -viesteillä HTTP (*HyperText Transfer Protocol*) [8] -protokollan yli (Ajaxin avulla) [29, s. 56].

Vaadin-komponentit koostuvat kahdesta osasta: Palvelinpuolen ja asiakaspuolen komponentista, jotka yhdistetään toisiinsa *liittimillä* (engl. *connectors*). Palvelinpuolen komponentille luodaan asiakaspuolelle liitin ja komponentti (*widget*), kun sivu esitetään selaimessa. [34, s. 442-444]

Grönroosin [34, s. 444] mukaan komponentin tila synkronoidaan automaattisesti asiakaspuolen komponentin kanssa niin sanotun *jaetun tilan olion* (engl. *shared state object*) avulla, joka toteuttaa `ComponentState`-rajapinnan. Molempien puolien komponentit käyttävät tätä oliota. Lisäksi komponentit voivat käyttää RPC (*Remote Procedure Call*) [35] -mekanismia kommunikoimiseen. Sitä käytetään enimmäkseen tapahtumista ilmoittamiseen: Esimerkiksi kun asiakaspuolen napin liitin vastaanottaa painalluksen, se lähettää tapahtuman palvelimelle RPC:n avulla.

4.4 Sovelluksen peruselementit

Palvelinpuolen Vaadin-sovelluksen käyttöliittymä toteutetaan UI-luokkana, joka luo ja hallitsee käyttöliittymäkomponentteja. Sovelluksella pitää olla ainakin yksi UI-luokka, joka laajentaa abstraktin `com.vaadin.ui.UI`-luokan ja toteuttaa `init()`-metodin. Tämä UI edustaa HTML-osiota, jossa Vaadin-sovellus ajetaan selaimessa. Yleensä se täyttää koko sivun, mutta se voi myös olla vain sivun osa. Kun käyttäjä avaa Vaadin UI:n osoitteen, sille luodaan automaattisesti uudet *UI*- ja *Sivu*-oliot. [34, s. 73-75]

Vaadin-kehyksen UI:t ajetaan *servletteinä*, jotka on pakattu `VaadinServlet`-luokkaan. UI määritellään servletille parametrina `web.xml`-tiedostossa. [34, s. 68] UI-luokka tarjoaa sovelluksen lähtöpisteen (engl. *entry point*) `init()`-metodin kautta.

Kyseinen metodi suoritetaan, kun UI-luokkaan liitettyyn servlettiin otetaan yhteys. Lisäksi UI-luokan kautta saa asetettua sovelluksen HTML-otsikon ja sovelluksen teeman. [29, s. 71]

Grönroos [34, s. 75-76] kuvaa myös muita palvelinpuolen Vaadin-sovelluksen perus-elementtejä, joita ovat:

- **Sivu (engl. *page*)**: Sivu-olio edustaa sitä WWW-sivua ja selaimen ikkunaa, jossa siihen liittyvä UI ajetaan.
- **VaadinSessio:** `VaadinSession`-olio edustaa käyttäjän sessiota [25, luku 18], jossa on avattuna yksi tai useampi UI. Sessio käynnistyy, kun UI avataan.
- **Käyttöliittymäkomponentit:** Käyttöliittymä koostuu käyttöliittymäkomponenteista, jotka asetellaan sivulle ulkoasukomponenttien avulla.
- **Tapahtumat:** Vaadin noudattaa tapahtumapohjaista mallia, jossa käyttäjän vuorovaikutus sovelluksen kanssa perustuu komponenttien aiheuttamiin tapahtumiin ja tapahtumia käsitteleviin kuuntelijoihin.
- **Resurssit:** Vaadin-sovellus voi sisältää myös resursseja, kuten kuvia.
- **Teemat:** Sovelluksen ulkoasu erotetaan logiikasta teemojen avulla. Teemat ovat CSS- tai SCSS [34, s. 268] -tyylitiedostoja, mutta käyttäjän luomat teemat voivat sisältää myös HTML-mallineita ja muita resursseja (kuten kuvia).
- **Datamalli:** Vaadin-kehyksen datamalli mahdollistaa komponenttien liittämiseen suoraan dataan: Komponentit voivat saada sisältönsä ja päivittää sen suoraan datamallin kautta, ilman erillistä kontrollikoodia.

4.5 Käyttöliittymäkomponentit ja ulkoasu

Vaadin sisältää kahdenlaisia komponentteja: Toisten kanssa käyttäjä vuorovaikuttaa (käyttöliittymäkomponentit) ja toisilla edelliset komponentit sijoitetaan tiettyyn paikkaan sivulla (ulkoasukomponentit). Ulkoasukomponenttien avulla komponentit saa sijoiteltua sivulle pelkän Java-koodin avulla. [34, s. 106, 220-221] Taulukossa 2 kuvataan tärkeimmät ulkoasukomponentit ja kuinka ne toimii. Käyttöliittymäkomponentit ovat muokattavia ja uudelleenkäytettäviä elementtejä, jotka muodostavat

sovelluksen käyttöliittymän [39, s. 57]. Vaadin sisältää valikoiman valmiita käyttöliittymäkomponentteja ja niitä on tarjolla myös lisäosina. Lisäksi Vaadin tarjoaa myös mahdollisuuden luoda omia komponentteja. [29, s. 77]

Taulukko 2: Vaadin-ulkoasukomponentit [34, s. 222-248]

Komponentti	**Kuvaus**
AbsoluteLayout	Asettaa komponentit tiettyihin koordinaatteihin.
VerticalLayout	Asettaa komponentit allekkain.
HorizontalLayout	Asettaa komponentit vierekkäin.
GridLayout	Asettaa komponentit ruudukkoon, joka määritellään sarakkeiden ja rivien lukumäärällä.
FormLayout	Asettaa komponentit ja niiden otsikot kahteen sarakkeeseen. Näyttää ilmaisimella komponentin virheen tai pakollisuuden.
CssLayout	Komponentit sisältyvät yksinkertaiseen DOM-rakenteeseen div-elementtien sisään. Oletuksena komponentit järjestetään vierekkäin.

Grönroosin [34, s. 221] mukaan Vaadin-sovelluksen ulkoasun voi luoda myös HTML-mallineen avulla, joka sisältää paikat Vaadin-kehyksen käyttöliittymäkomponenteille. Tämä tehdään `CustomLayout`-komponentin avulla. Tässä kuitenkin menettää mahdollisuuden hallita ulkoasua dynaamisesti.

Vaadin-kehyksen Eclipse-liitännäinen sisältää myös visuaalisen Vaadin editorin, jolla käyttöliittymät voi luoda. Editori luo koodin, joka luo käyttöliittymän. [34, s. 222]

Vaadin-sovelluksen visuaalinen tyyli määritellään teeman avulla. Vaadin tarjoaa valikoiman valmiita teemoja, joita voi muokata erilaisten asetusten avulla. Myös omia teemoja voi luoda laajentamalla perusteeman (`base`). [34, s. 259-260, 278-279]

5 ASP.NET

ASP.NET on WWW-sovelluskehys, joka on luotu .NET-kehyksen päälle [31, s. 4]. Sen avulla WWW-sovelluksia luodaan käyttäen HTML:ää, CSS:ää, Javascriptiä ja palvelinpuolen koodausta [4]. ASP.NET tukee kolmea eri tapaa luoda WWW-sovelluksia:

- **WWW-sivut (engl. *Web Pages*)**: Yksittäisten sivujen malli ja yksinkertaisin ASP.NET-malli, joka sekoittaa sovelluslogiikkakoodin HTML-koodin sekaan. Samankaltainen kuin JSP (*Java Server Pages*) [6, s. 3] ja perinteinen ASP (*Active Server Pages*) [31, s. 4]. [4] [3]
- **WWW-lomakkeet (engl. *Web Forms*)**: Tapahtumapohjainen, perinteinen ASP.NET-malli. WWW-sivuihin lisätään palvelinkontrolleja ja palvelinpuolen koodia. [4] Palvelinkontrollien aiheuttamat tapahtumat käsitellään palvelinpuolen koodissa tapahtumankäsittelijöiden avulla. [3]
- **MVC:** Lisäosa, jolla saadaan luotua ASP.NET-sovelluksia suositun Malli-Näkymä-Ohjain-arkkitehtuurin (engl. *Model-View-Controller, MVC*) [31, s. 1188] mukaisesti. Julkaistiin ASP.NET 3.5:n (2008) jälkeen. [4] [31, s. 4]

ASP.NET-sovelluksista puhuttaessa tarkoitetaan yleensä WWW-lomakkeet -mallia. Tässä luvussa keskitytään jatkossa tähän malliin.

5.1 WWW-lomakkeet

ASP.NET WWW-lomakkeet oli alkujaan täysin uudenlainen tapa tehdä WWW-sovelluksia: Sovellusten luominen muistuttaa normaalien Windows-sovellusten luomista (niissä on samanlainen kontrollipohjainen käyttöliittymä). Käyttäjän toimiin sivulla pystytään reagoimaan tapahtumien avulla (kuten `Button.Click`) sen sijaan että napin painallus lähetettäisiin esimerkiksi `HTTP POST` -metodin kautta. WWW-lomakkeet tarjoaa runsaasti erilaisia kontrolleja ja komponentteja WWW-sovelluksien kehittämisen avuksi. [5] [47, s. 77]

ASP.NET WWW-lomakkeet -sovellus luodaan HTML-, CSS- ja Javascript-koodeilla sekä ASP.NET-palvelinkontrolleilla ja palvelinpuolen koodilla. Palvelinpuolen koodi voi olla toteutettu millä tahansa .NET-tuetulla kielellä (kuten C# [47, s. 4] tai Visual Basic [47, s. 4]). Koodi voidaan upottaa suoraan HTML-sivulle kuten esimerkiksi JSP:ssä tai sen voi sijoittaa erilliseen kooditiedostoon. Koodi voidaan ajaa sovelluksen latauduttua tai käyttäjän syötteiden kautta. [5]

WWW-lomakkeet käännetään ja suoritetaan palvelimella, ja kun kaikki prosessointi on saatu valmiiksi (mukaan lukien palvelinkontrollien prosessointi .aspx-tiedostoista ja niistä saadun HTML-koodin lisääminen sivulle) ja sivun HTML-koodi kasattua, se lähetetään takaisin selaimeen. [5] [62, s. 10]

5.2 Sovellus- ja kansiorakenne

ASP.NET-sovelluksen saa luotua yhdellä WWW-lomaketiedostolla (.aspx-tiedosto). Sovellus voi kuitenkin sisältää paljon muutakin. Taulukossa 3 kuvataan, mitä osia ASP.NET-sovellukseen voi kuulua.

ASP.NET 1.0/1.1 -versiot käänsivät koko WWW-sovelluksen DLL (*Dynamic Link Library*) [7] -tiedostoon. Nykyään ASP.NET sisältää määritellyn kansiorakenteen, jota käyttämällä koodi käännetään automaattisesti ja sovelluksen teemat ovat saavutettavissa kaikkialta sovelluksesta. [31, s. 78] Seuraavassa selitetään lyhyesti ASP.NET-määritellyt kansiot:

- **Bin:** Sisältää valmiiksi käännetyt .NET-komponentit (yleensä .dll-tiedostot), joita sovellus käyttää. [47, s. 187]
- **App_Code:** Sisältää kooditiedostot, jotka käännetään dynaamisesti sovelluksen käytettäväksi [47, s. 187]. ASP.NET-palvelin kääntää tämän kansion luokat automaattisesti, jolloin koodi on saavutettavissa sovelluksen jokaiselta sivulta. Tämän kansion käyttämisen sijaan kehittäjä voi lisätä myös erillisen, valmiiksi käännetyn koodin. [31, s. 78]
- **App_Data:** Oletushakemisto ASP.NET-sovelluksen käyttämälle datalle kuten tietokannoille ja XML-tiedostoille. [31, s. 82] [47, s. 187]

Taulukko 3: ASP.NET-sovelluksen osat [47, s. 184-186]

Osa	**Tiedosto**	**Kuvaus**
WWW-lomakkeet	.aspx-tiedostot	ASP.NET-sovelluksen perusosat.
Isäntäsivut	.master-tiedostot	Mallineet, joiden avulla saa luotua useita WWW-lomakkeita samalla rakenteella.
WWW-palvelut	.asmx-tiedosto	Mahdollistaa funktioiden jakamisen toisille koneille ja sovellusalustoille.
Palvelinkoodit	Esim. .cs-tiedostot	Erilliset palvelinpuolen koodit.
Konfiguraatio-tiedosto	web.config	Sovellustason asetukset (esim. turvallisuuteen ja debuggaukseen liittyvät).
Sovellustapahtumat	global.asax	Sovellustason tapahtumankäsittelijät (esim. kun sovellus käynnistyy).
Komponentit	Yleensä .dll-tiedostot	Erilliset käännetyt komponentit, joita sovellus käyttää.
Resurssit	Esim. .css ja .jpg	Lisätiedostot, joita sovellus käyttää (esim. tyylitiedosto tai kuva).

- **App_GlobalResource:** Sisältää resurssit, jotka ovat saavutettavissa jokaiselta sovelluksen sivulta. [47, s. 187]
- **App_LocalResources:** Sisältää resurssit, jotka ovat saavutettavissa vain yhdeltä sivulta. [47, s. 187]
- **App_Themes:** Sisältää sovelluksen käyttämät teemat, joiden avulla saavutetaan sivuston yhtenäinen ulkoasu ja ulkoasun helppo muunnettavuus. [47, s. 187]
- **App_WebReferences:** Tarjoaa automaattisen pääsyn ulkopuolisiin WWW-palveluihin, joihin sovelluksesta viitataan [31, s. 83]. Sisältää viittaukset (esimerkiksi WSDL (*Web Services Description Language*) [13] -tiedostot) sovelluksen käyttämiin palveluihin [47, s. 187].
- **App_Browsers:** Sisältää `.Browser`-tiedostot (eli XML-tiedostot), jotka kuvaavat asiakaspuolen selainten ominaisuuksia. [31, s. 83] [47, s. 187]

5.3 Palvelinkontrollit

Palvelinkontrollit ovat .NET-sovelluskehyksen luokkia, jotka edustavat tiettyjä WWW-sovelluksen elementtejä. ASP.NET tarjoaa kahdentyyppisiä palvelinkontrolleja: Web- ja HTML-palvelinkontrollit. HTML-palvelinkontrollit ovat luokkia, jotka edustavat perinteisiä HTML-elementtejä, kuten `HtmlAnchor` (`<a>`-elementti) ja `HtmlSelect` (`<select>`-elementti). [47, s. 129]

Myös Web-palvelinkontrollit sisältävät perinteiset HTML-elementit, mutta niillä on ominaisuuksia ja metodeja, jotka tekevät niiden luomisesta ja käyttämisestä helpompaa. Näiden lisäksi Web-palvelinkontrollit sisältävät monia erikoisominaisuuksia ja -toiminnallisuuksia (esimerkiksi kalenterin). Nämä kontrollit voivat luoda suuren määrän HTML-koodia, ja jopa asiakaspuolen Javascript-koodia kontrollin toiminnallisuuden mahdollistamiseksi. [47, s. 129-130] Web-palvelinkontrollit valitsevat esitysmuotonsa mukautuvasti ottaen asiakkaan ominaisuudet huomioon: Esitysmuoto riippuu esimerkiksi selaimesta tai sen versiosta [47, s. 11]. ASP.NET hoitaa selaimen havaitsemisen automaattisesti [31, s. 124].

MacDonaldin ym. [47, s. 129] mukaan ASP.NET tarjoaa myös mahdollisuuden muut-

taa minkä tahansa HTML-elementin palvelinkontrolliksi. Tämä tapahtuu lisäämällä elementille attribuutti ja arvo `runat="server"`:

```
<input type="button" id="nappi" value="paina" runat="server">
```

Tämän jälkeen sitä voi käyttää palvelimella kuten mitä tahansa Web-palvelinkontrollia.

Palvelinkontrollit noudattavat tapahtumapohjaista mallia: Käyttäjän toiminto kontrollissa aiheuttaa tapahtuman, joka vastaanotetaan palvelinpuolen koodissa tapahtumankäsittelijän avulla. Palvelinkontrollien tyyliä voi muuttaa muokkaamalla kontrollin ominaisuuksia (engl. *properties*). [31, s. 107-108]

5.4 ASP.NET AJAX

Gaylord ym. [31, s. 860] esittelevät ASP.NET AJAX -lisäosan, joka sisältää sekä asiakas- että palvelinpuolen komponentteja Ajax-toiminnallisuuden lisäämiseksi ASP.NET-sovellukseen. Se tarjoaa skriptejä, jotka automatisoivat suurimman osan Ajax-kommunikaatioista. .NET versiosta 3.5 lähtien on ASP.NET AJAX valmiiksi asennettuna. ASP.NET AJAX -sovellukset toimivat kaikissa suosituimmissa selaimissa, koska selainten yhteensopivuusongelmat on otettu huomioon.

ASP.NET AJAX on jaettu kahteen osaan: Asiakaspuolen kehykseen ja palvelinpuolen kehykseen. Asiakaspuoli tarjoaa Javascript-kirjaston, joka tarjoaa skriptejä asiakkaan kommunikaatioon palvelimelle. Kirjasto hoitaa myös selainten yhteensopivuusongelmat. Asiakaspuolen teknologiat ovat täysin riippumattomia ASP.NETistä, joten sitä voi käyttää myös muiden teknologioiden (kuten PHP tai JSP) kanssa. ASP.NET tukee myös jQuery Javascript-kirjastoa, mutta ASP.NET AJAX tarjoaa ominaisuuksia, joita ei siitä löydy. [31, s. 861-862]

Gaylordin ym. [31, s. 871] mukaan palvelinpuolen kehyksen toiminta perustuu `ScriptManager`-kontrolliin. Kyseinen kontrolli lisätään jokaiselle sivulle, joilla haluaa käyttää ASP.NET AJAX -ominaisuuksia. Se hallitsee sivuilla käytettyjä Javascript-kirjastoja sekä välittää viestejä palvelimen ja asiakkaan välillä, kun vain osa

sivusta päivitetään.

Palvelinpuolen kehyksessä sivun osittainen päivittäminen saadaan aikaan `UpdatePanel`-kontrollin avulla. Kun tapahtuma syntyy `UpdatePanel`-kontrollin sisällä, ASP.NET AJAXin luoma Javascript-koodi keskeyttää koko sivun uudelleen päivityksen ja synnyttää asynkronisen kutsun sen sijasta. Palvelimella suoritetaan normaalit sivutapahtumat (esimerkiksi `Page_Init` ja `Page_Load`) ja viimeisenä varsinainen kutsun aiheuttanut tapahtuma. Tämän jälkeen sivu renderöidään HTML-koodiksi ja palautetaan selaimelle. Selaimen Javascript-koodi päivittää sivun kaikki `UpdatePanel`-kontrollit korvaamalla niiden HTML-sisällöt uudella. [47, s. 1260]

`UpdatePanel`-kontrollin `<ContentTemplate>`-elementti sisältää sisällön, joka päivitetään asynkronisesti ilman koko sivun uudelleenlatausta. Mikä tahansa kontrolli (esimerkiksi `Button`), joka normaalisti aiheuttaa koko sivun päivityksen ja joka on sijoitettu `<ContentTemplate>`-elementin sisälle, aiheuttaa asynkronisen kutsun koko sivun päivittämisen sijaan. Tapahtuman aiheuttavien kontrollien lisäämistä `<ContentTemplate>`-elementin sisään kannattaa kuitenkin välttää (jos niitä ei tarvitse päivittää), koska tällöin myös ne päivitetään sivulle uudestaan, mikä tietysti hidastaa sovelluksen toimintaa. [31, s. 875-877]

`<Triggers>`-elementin avulla voidaan määritellä `<ContentTemplate>`-elementin ulkopuolisia kontrolleja aiheuttamaan `UpdatePanel`-kontrollin päivittäminen ja asynkronisen kutsu. `<Triggers>`-elementti voi sisältää kahdentyyppisiä kontrolleja: `AsyncPostBackTrigger` aiheuttaa asynkronisen kutsun ja `PostBackTrigger` koko sivun päivittävän kutsun. Kyseiset kontrollit määritellään antamalla niille kutsun aiheuttavan kontrollin ID ja tapahtuma, joka kutsun aiheuttaa. `AsyncPostBackTrigger`-kontrollille määritelty kontrolli ja tapahtuma aiheuttaa siis aina asynkronisen kutsun. [47, s. 1266-1267]

MacDonaldin ym. [47, s. 1264] mukaan `UpdatePanel`-kontrollin attribuutilla `UpdateMode="Conditional"` saa määriteltyä sisällön päivittymään vain kun asetetut triggeriehdot toteutuvat. Oletusarvo `UpdateMode="Always"` päivittää elementin jokaisella asynkronisella takaisinkutsulla (myös muista `UpdatePanel`-

kontrolleista tulevilla).

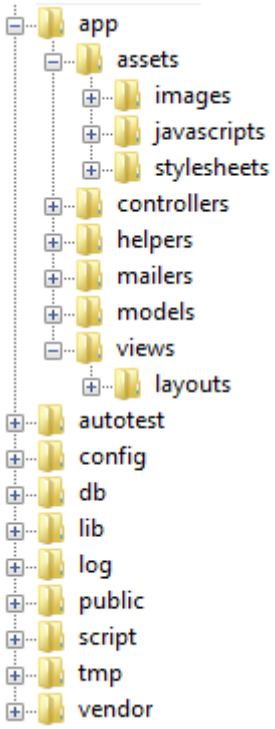
`UpdatePanel`-kontrolleja voi lisätä sivulle useita, jolloin voi itse kontrolloida, mitä alueita milloinkin päivitetään. Kaikille kontrolleille saa tietysti määriteltyä omat `<ContentTemplate>`- ja `<Triggers>`-elementit. [31, s. 882-883]

6 Ruby on Rails

Ruby on Rails (lyhyesti Rails) on WWW-sovelluskehys Ruby [16] -kielelle. Railsin käyttö kuvataan Rails APIssa, joka on dokumentoitu verkossa. Railsilla on myös laaja yhteisö, joka tuottaa ohjelmakirjastoja (RubyGem tai "gem" [42]) helpottamaan WWW-sovellusten luomista. Rails WWW-sovellus luodaan HTML-, CSS- ja Javascript-kielillä yhdessä Ruby-ohjelmointikielen kanssa. Rails noudattaa Malli-Näkymä-Ohjain-arkkitehtuuria. [42]

6.1 Kansiorakenne

Kehoen [42] mukaan kaikki Rails-sovellukset noudattavat samaa kansiorakennetta: Mallille, näkymälle ja ohjaimelle on omat kansiot kuten myös esimerkiksi kuville, CSS-tyylitiedostoille ja Javascript-tiedostoille. Kuva 5 esittää tämän kansiorakenteen.



Kuvio 5: Railsin kansiorakenne

`app`-kansioon sijoitetaan varsinaiset sovelluksen koodit (esimerkiksi ohjaimet, näky-

mät ja mallit). `config`-kansioon sijoitetaan Railsin konfiguraatiotiedostot: Esimerkiksi tiedosto reiteille (engl. *routes*) löytyy täältä. `vendor`-kansio sisältää kolmannen osapuolen koodit kuten Railsin liitännäiset (engl. *plugins*) ja riippuvuudet (engl. *dependencies*). `lib`-kansioon voi laittaa koodit, jotka eivät suoraan sisälly mallille, näkymälle tai ohjaimelle sekä koodit, jotka jaetaan näiden kesken. [56, s. 256-260, 263]

6.2 Railsin filosofia

Railsin filosofia perustuu kahteen periaatteeseen: "Älä toista itseäsi" (engl. *Don't Repeat Yourself, DRY*) ja "käytäntö ennen konfigurointia". (engl. *Convention-over-Configuration, CoC*) [54, s. 7]. Esimerkiksi seuraavat Railsin ominaisuudet seuraavat näitä periaatteita: [54, s. 8]

- **Sovelluksen rakenteen automaattinen asennus:** Koko sovelluksen rakenteen saa luotua yhdellä komennolla. Se luo kansiot ja perus-WWW-tiedostot.
- **Luokkien ja tietokannan taulujen dynaaminen yhdistäminen:** Rails käyttää ORMia (engl. *Object-Relational Mapping*) [54, s. 40-41] tietokannan ja sovelluksen väliseen yhteydenpitoon. Yleensä ORM kuitenkin vaatii erillisen konfiguroinnin, mutta ei Railsin tapauksessa: Taulut yhdistetään luokkiin sovelluksen ajoaikana automaattisesti.
- **Samanlaisen koodin toiston rajoittaminen:** Sovelluksilla on yleensä koodipaloja, jotka ovat olennaisesti samanlaiset muiden samanlaisten sovellusten kanssa. Sama koodi voi olla monen sovelluksen käytettävissä vain pienillä muutoksilla. Rails yrittää välttää tällaisen koodin toistamisen tarjoamalla tietyt toiminnot (kuten tietokantaan yhdistäminen) mahdollisimman yksinkertaisella koodilla.
- **Sisäänrakennettu Ajax-tuki:** Railsissa Ajax on osa sen peruskirjastoja (engl. *core libraries*): Kun asentaa Railsin, myös Ajax-tuki on suoraan käytettävissä. Ajaxin käyttö on yhtä helppoa kuin minkä tahansa muun Railsin tarjoaman kirjaston.

6.3 Malli-Näkymä-Ohjain-rakenne

Rails toteuttaa MVC-arkkitehtuurin, joka erottaa datan, käyttöliittymän ja sovelluslogiikan toisistaan. Ohjaimet toimivat mallin ja näkymän välillä: Ne vastaavat käyttäjän toimintoihin, keräävät tarvittavan datan mallin olioista ja palauttavat tulokset näkymälle näytettäväksi. [71]

Viswanathanin [71] mukaan MVC parantaa sovelluksen modulaarisuutta ja uudelleenkäyttöä: Jokainen MVC:n komponentti on kehiteltävissä ja muunneltavissa täysin itsenäisesti, erillään muista komponenteista. Näin esimerkiksi käyttöliittymän muokkaaminen tai rakentaminen kokonaan uudelleen on mahdollista koskematta ollenkaan dataan tai sovelluslogiikkaan.

Railsin kansiorakenne noudattaa siis MVC-arkkitehtuuria ja lisäksi MVC-erottelun apuna käytetään "alikehyksiä": `ActionController` (Ohjain), `ActionView` (Näkymä) ja `ActiveRecord` (Malli). [71]

`ActionDispatch`, `ActionController` ja `ActionView` kuuluvat yhteen komponenttiin nimeltään `Action Pack` . Tämä siksi että niillä on hyvin läheinen vuorovaikutus: Ohjain välittää dataa näkymälle ja vastaanottaa näkymän luomalta sivulta tapahtumia. `ActionDispatch` reitittää pyynnöt ohjaimelle, `ActionController` muuntaa pyynnöt vastauksiksi ja pyytää `ActionView`-alikehystä muotoilemaan vastaukset. [56, s. 34, 307]

6.3.1 ActionDispatch

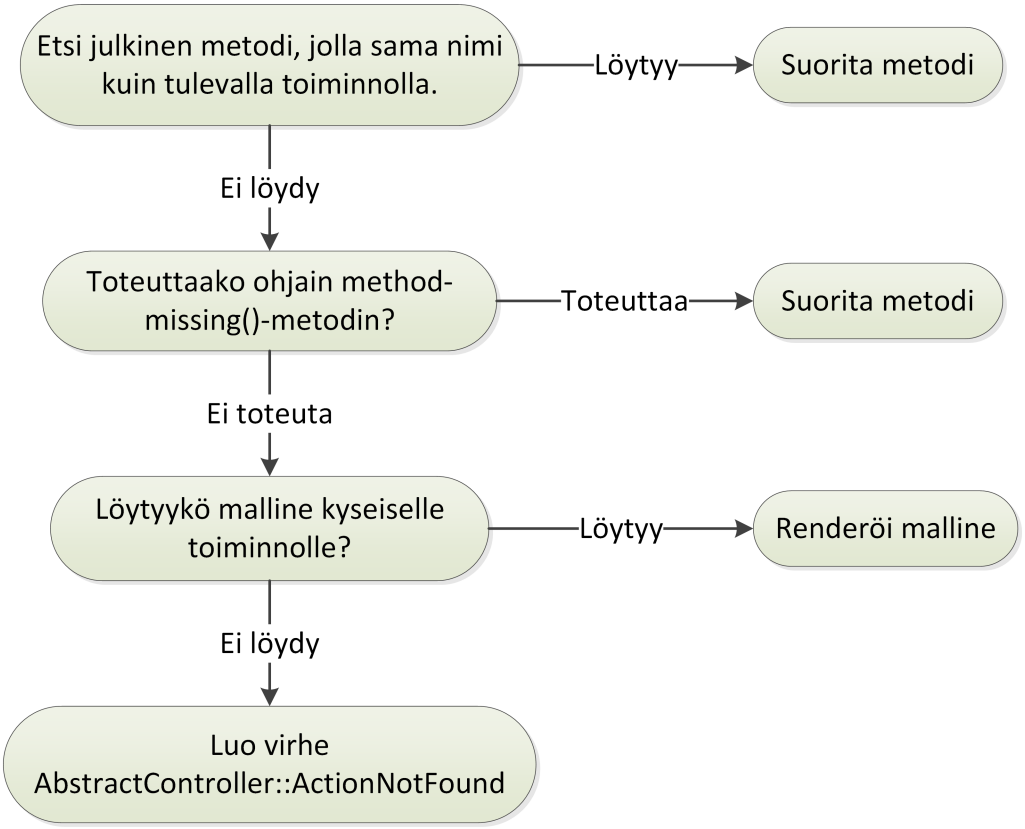
Rubyn ym. [56, s. 314] mukaan Rails koodaa pyynnön URL:n tiedot ja käyttää `ActionDispatch`-alijärjestelmää määrittämään, mitä pyynnöllä tehdään. Lopulta on saatu selville ohjain, joka pyynnön käsittelee, sekä lista *request*-parametreista. Suoritettava ohjaimen toiminto saadaan selville joko request-parametreista tai itse HTTP-metodista.

Railsissa pyynnöt ohjataan tietylle ohjaimelle ja toiminnolle *reittien* avulla [14, s. 24]. Jokaiselle pyynnölle pitää määritellä reitti `routes.rb`-tiedostoon. Rails tarjoaa kaksi

tapaa määritellä reittejä: Toisessa määritellään suora reitti URL:stä toimintoon ja toisessa määritellään reitti resurssien (kuten määriteltyjen mallien) avulla [56, s. 308].

6.3.2 ActionController

`ActionController` vastaanottaa pyyntöjä URL:stä, prosessoi ne toimintoina tai olioina ja palauttaa näkymän selaimelle. Kehittäjä luo `ActionController`-alikehykseen toimintoja (engl. *Actions*), jotka määrittävät kuinka ohjain reagoi tiettyihin HTTP-pyyntöihin. Nämä toiminnot toteutetaan julkisina metodeina. Ohjain luo toiminnoille ympäristön, jonka metodeilla pääsee käsiksi kyseisen URL:n tai pyynnön tietoon. Kuvio 6 esittää, kuinka ohjain vastaanottaa pyynnön. [16]



Kuvio 6: Railsin ohjaimen pyynnön vastaanotto [56, s. 319]

Bächlen ym. [16] mukaan `ActionController`-alikehys luo myös yhteyden `ActiveRecord`-alikehykseen ja välittää dataa tietokannasta `ActionView`-alikehykselle. Muita `ActionController`-alikehyksen toimintoja on esimerkiksi sessioiden

käsittely.

6.3.3 ActionView

Railsissa näkymä luo sisällön, joka näytetään selaimessa, prosessoidaan sovelluksessa tai lähetetään sähköpostilla. Yksinkertaisimmillaan näkymä on HTML-koodia, joka näyttää määrättyä tekstiä. Näkymä voi myös näyttää dynaamista sisältöä, joka luodaan ohjaimen toiminnossa. Tämä dynaaminen sisältö näytetään mallineen avulla. Rails määrittelee erilaiset mallineet (engl. *templates*) (taulukko 4), joiden avulla esityskerros eritellään muusta sovelluksesta. `ActionView`-alikehys kapseloi kaiken toiminnon, joka tarvitaan mallineiden renderöintiin (eli useimmiten HTML-, XML- tai Javascript-koodin luomiseen asiakkaalle). [16] [56, s. 34, 341]

Taulukko 4: Railsin mallinetyypit [56, s. 35, 342-343]

Malline	**Sisältö**	**Tehtävä**
html.erb	HTML ja Ruby	Luo HTML-sivut.
js.erb	Javascript ja Ruby	Luo Javascript-osia palvelimella, jotka suoritetaan selaimessa.
xml.builder	Ruby	Rakentaa XML-dokumentteja Ruby-koodista.
SCSS	CSS	Kontrolloi sisällön ulkoasua selaimessa.

ERb-tiedosto (*Embedded Ruby*) sisällyttää Ruby-koodia näkymä-dokumenttiin [56, s. 34-35]. Ajoaikana Ruby-koodin prosessointi täyttää mallineen oikealla sisällöllä. Ruby-koodi sijoitetaan mallineeseen `<%...%>` -tageilla. [16]

Ruby-koodi siis lisää dynaamista sisältöä mallineeseen ja sillä pääsee käsiksi ohjaimen muodostamaan tietoon: [56, s. 342]

- Kaikki ohjaimen attribuutit ovat saatavilla mallineessa, näin toiminnot välittä-

vät dataa mallineelle.

- Ohjaimen oliot `flash`, `headers`, `logger`, `params`, `request`, `response`, ja `session` ovat saatavilla metodeina näkymässä. `Flash`-oliota lukuun ottamatta näitä ei tulisi kuitenkaan käyttää näkymässä suoraan, sillä niiden käsittely kuuluu ohjaimelle.
- Nykyinen ohjain-olio on saatavilla `controller`-attribuutilla. Tämän avulla malline voi kutsua mitä tahansa ohjaimen julkista metodia.
- Polku mallineen kotihakemistoon löytyy attribuutista `base_path`.

Osittainen malline (engl. *partial-page template*), jota kutsutaan usein nimellä *partial*, mahdollistaa uudelleenkäytettävän sivun osan luomisen. Se on sisällöltään samanlainen kuin tavallinen malline, mutta sen tiedostonimen alussa on alaviiva. Muut mallineet kutsuvat sitä `render(partial:)`-metodin avulla:

```
<\%= render(partial: "article", object: @an\_article) \%>
```

Tässä `partial`-parametri on osittaisen mallineen nimi ilman alaviivaa. `object`-parametrin avulla saa välitettyä olion osittaiselle mallineelle. Kun osittainen malline on renderöinyt itsensä, se palauttaa kontrollin sitä kutsuneelle mallineelle. [56, s. 363-364]

6.3.4 ActiveRecord

`ActiveRecord`-alikehys muodostaa yhteyden olioiden ja tietokannan välille ORMin avulla. Luokat yhdistyvät suoraan tietokannan tauluihin: Luokan oliot edustavat tietokannan tauluja ja attribuutit sarakkeita. [16] Näin siis Railsissa oliomalli synkronoidaan automaattisesti tietokantamallin kanssa [71]. Railsin ORM ei vaadi konfigurointia XML-tiedoston kautta: `ActiveRecord::Base` aktivoi taulut automaattisesti, kun niitä kutsutaan tietokannasta [16].

Bächlen ym. [16] mukaan `ActiveRecord`-alikehys muuntaa `ActionController`-alikehykseltä tulevat CRUD (*Create, Read, Update and Delete*) [56, s. 277] [54, s. 41] -toiminnot SQL (*Structured Query Language*) [11] -pyynnöiksi, lähettää ne tietokannalle

ja palauttaa tulokset takaisin `ActionController`-alikehykselle. `ActiveRecord`-alikehys myös tarkistaa onko käyttäjällä oikeus päästä käsiksi tai muuntaa tiettyä tietokannan taulua.

6.4 Ajax

Gadboisin [30] mukaan Rails tarjoaa Rails 3 -versiosta ylöspäin niin sanotun *"huomaamattoman"* Javascript-tuen (engl. *Unobtrusive JavaScript*). Se auttaa erottamaan Javascript-koodin HTML-koodista, ja tarjoaa mahdollisuuden valita mikä tahansa Javascript-kirjasto (jolle löytyy Rails-toteutus), käytettäväksi avustaja-metodien kanssa. Aiemmin Rails oli tiukasti kytköksissä Prototype-kirjastoon.

Rails toteuttaa kaikki Javascript-avustaja -toiminnot huomaamattomasti lisäämällä seuraavat HTML 5 -attribuutit HTML-elementeille: [30]

- **data-method:** REST (*Representational State Transfer*) [27] -metodi, jota käytetään lomakkeen lähettämisessä.
- **data-confirm:** Varmistusviesti, joka kysytään ennen tietyn toiminnon suorittamista.
- **data-remote:** Jos tosi, lähetetään Ajax-kutsuna.
- **data-disable-with:** Poistaa lomake-elementit käytöstä lomakkeen lähetyksen aikana.

Gadboisin [30] mukaan esimerkiksi napin painalluksen välittäminen Ajax-kutsuna toteutetaan lisäämällä napille attribuutti `:remote => true`. Lopulliseen HTML-elementtiin tulee tällöin lisää vain attribuutti `data-remote="true"`. Ennen Rails 3:sta tämä loi Javascript-koodia elementin sisään.

Ohjaimelle pitää myös kertoa, että sen pitää reagoida Ajax-kutsuun. Tämä onnistuu lisäämällä kutsuttavan toiminnon `respond_to`-lohkoon `format.js`. Rails reagoi tähän Javascript-pyyntöön renderöimällä `.js.erb`-mallineen, jolla on sama nimi kuin ohjaimen toiminnolla jota kutsutaan, ja joka sijaitsee ohjainta vastaavan näkymän kansiossa. Tässä `.js.erb`-mallineessa voi siis käyttää haluamaansa Javascript-

kirjastoa apuna. [30]

7 Seaside

Seaside on WWW-sovelluskehys Smalltalk [25, luku 1.3] -kielelle. Seasidessa XHTML luodaan kokonaan Smalltalk-koodilla: Siinä ei ole erillisiä XHTML-tiedostoja tai -mallineita [25, luku 1].

Seaside-sovellukset perustuvat itsenäisten komponenttien yhdistämiseen. Yksi komponentti määrittelee tietyn sivun osan ulkoasun ja käyttäytymisen [23, s. 237]. Jokainen komponentti on vastuussa itsensä esittämisestä sivulla (engl. *rendering*), omasta tilastaan ja kontrollivirrastaan (engl. *control flow*) [25, luku: 1]. Sivulla näkyvät komponentit renderöivät itsensä `renderContentOn:`-metodin avulla, jolloin ne luovat XHTML-esityksen itsestään [23, s. 237].

Seaside sisältää myös WWW-palvelimen, jonka kautta luotavia sivuja voi testata, sekä WWW-käyttöliittymän sovellusten hallitsemista varten. WWW-käyttöliittymän kautta voi lisätä, muokata ja poistaa sovelluksia. Sovelluksen konfiguraatiosivulla voi esimerkiksi määrätä sovelluksen juuriluokan ja lisätä kirjastoja. [15, s. 240, 244]

7.1 Komponenttien renderöinti

Ducassen ym. [25, luku 7] mukaan Seasidessa kehittäjän ei tarvitse itse kirjoittaa XHTML-koodia, eikä huolehtia, onko sivun XHTML validia: XHTML luodaan piirtoalueiden (engl. *canvas*) ja siveltimien (engl. *brush*) avulla.

Seaside-sovellukset luodaan `WAComponent`-luokan aliluokkina. Kun komponentti pitää esittää, Seaside lähettää sille viestin

```
WAComponent>>renderContentOn:
```

joka sisältää parametrina `WARenderCanvas`-luokan ilmentymän (nimetään yleensä `html`). Tämä ilmentymä eli *piirtoalue* on se paikka johon komponentit "piirretään". Se tarjoaa rajapinnan XHTML:lle, jonka kautta on helppo luoda tekstiä, linkkejä, kuvia ja niin edelleen. [25, luku 7] Piirtoalue tarjoaa erilaisia *siveltimiä*, joita voidaan

käyttää sisällön renderöimiseen piirtoalueelle. Kaikille XHTML-elementeille on oma sivellin. [15, s. 249]

Piirtoalueelle välitetty viesti pyytää sitä ottamaan käyttöön uuden siveltimen. Esimerkiksi viestillä `break` eli koodilla `html break` voi lisätä rivivaihdon. Jokainen sivellin liittyy tietyn tyyppiseen HTML-tagiin, ja sille voi välittää parametreja, joilla renderöintiä määritellään tarkemmin. Monille siveltimille voi välittää useita viestejä erottamalla ne `;`-merkillä. Esimerkiksi otsikko-siveltimelle voi määrätä tason ja tekstin seuraavasti [25, luku 7]:

```
html heading
   level: 2;
   with: 'Toisen tason otsikko'.
```

Tämä luo seuraavan HTML-koodin:

```
<h2>Toisen tason otsikko</h2>
```

Siveltimen sisältö siis määritellään `with:`-viestillä. Tämä viesti tulee lähettää viimeiseksi, sillä se aiheuttaa XHTML-tagin renderöinnin. `with:`-viestiä ei tarvitse käyttää, jos siveltimelle ei määritä muita ominaisuuksia, esimerkiksi [15, s. 250-251]

```
html paragraph: 'Sisältö ilman with:-viestiä.'
```

luo seuraavan HTML-koodin:

```
<p>Sisältö ilman with:-viestiä.</p>
```

7.2 Takaisinkutsut

Komponentit reagoivat käyttäjän toimiin *takaisinkutsujen* (engl. *action callbacks*) avulla. Takaisinkutsut on määritelty esimerkiksi napeille, valintalaatikoille ja tekstikentille. [23, s. 238]

Ducassen ym. [25, luku 9] mukaan takaisinkutsu on Smalltalk-lohko, joka suoritetaan kun selain lähettää pyynnön takaisin Seasidelle (esimerkiksi kun käyttäjä on

painanut nappia). Kun takaisinkutsu on suoritettu, Seaside renderöi komponentin. Ennen renderöintiä suoritetaan kaikki aktiiviset takaisinkutsut. Takaisinkutsut ovat niitä paikkoja, joissa sovelluksen tilaa muutetaan: Sovelluksen tilaa ei saa muuttaa renderöinnin aikana.

Takaisinkutsu voi sisältää mitä tahansa Smalltalk-koodia. Siinä ei kuitenkaan saa renderöidä piirtoaluetta, koska tällöin piirtoalue on mitätön. Ajax-kutsuissa piirtoalue kuitenkin yleensä renderöidään: Tässä tapauksessa Ajax välittää uuden "renderöijän" takaisinkutsuun. Vanhaa piirtoaletta (yleensä nimellä `html`) ei tässäkään tule käyttää. [25, luku 9]

7.3 Kontrollivirrat

Takaisinkutsussa komponentti voi määritellä kontrollivirran, joka kuvaa sarjan, jossa komponentti on väliaikaisesti korvattu toisilla komponenteilla [24, s. 5-6]. Samalla sivulla voi olla kerrallaan useita kontrollivirtoja, yksi jokaiselle komponentille [23, s. 231].

Komponentti voi siirtää kontrollin toiselle komponentille `call`-metodin avulla. Tällöin komponentti korvataan väliaikaisesti toisella. Muut sivun komponentit pysyvät toiminnallisina ja niitä voidaan käyttää normaalisti. Komponentti palauttaa kontrollin sitä kutsuneelle komponentille `answer`-metodin avulla. Palautuksessa komponentti voi myös palauttaa olion vastauksen mukana. [24, s. 5-6]

Ducassen ym. [24, s. 5-6] mukaan Seaside toteuttaa tämän kutsu-vastaus mekanismin jatkumoiden avulla. Jatkumot ovat suorituspinon esityksiä tietyltä ajalta. Jatkumo on periaatteessa lopetettu prosessi, johon voidaan palata uudelleen useita kertoja tallennetun tilan kautta.

7.4 Ajax

Seasideen on integroituna Prototype-, Script.aculo.us-, jQuery- ja jQueryUI-kirjastot (jQuery ja jQueryUI Seasiden versiosta 3 lähtien). Kaikkiin näiden kirjastojen toi-

mintoihin pääsee käsiksi Smalltalk-koodin kautta. Käytettävä kirjasto pitää lisätä sovellukseen esimerkiksi Seasiden konfiguraatio sivulla. Lisäksi kirjaston paketti pitää ladata, jos sitä ei ole. [25, luvut 20, 21]

Ducassen ym. [25, luku 21] mukaan Smalltalk-puoli jQueryn integraatiosta luodaan automaattisesti jQuery-dokumentaatiosta, joten se on aina ajan tasalla. Koodilla `html jQuery` saa luotua `JQueryClass`-ilmentymän, jonka kautta jQueryn toimintoihin pääsee käsiksi. Esimerkiksi koodilla

```
html jQuery id: 'labelOtsikko'
```

pääsee käsiksi elementtiin, jonka id on `labelOtsikko`.

8 Muita WWW-sovelluskehyksiä

Tässä tutkielmassa vertailtavien sovelluskehysten lisäksi on tarjolla myös paljon muitakin WWW-sovelluskehyksiä, jotka tarjoavat myös erilaisia tapoja WWW-sovellusten luomiseen. Tässä luvussa esitellään sovelluskehyksiä, jotka eroavat jotenkin vertailuun otetuista. Taulukossa 5 esitetään sovelluskehysten ominaisuuksia. Taulukon neljä ensimmäistä sovelluskehystä kuvataan tarkemmin seuraavissa alaluvuissa.

Taulukko 5: Muita WWW-sovelluskehyksiä [17, luvut 1, 2, 11] [52, s. 15, 25]

Kehys	**Ohjelmointikieli**	**Ulkoasukieli**	**Ajax**
Django	Python	Django-malline	Django-dajax-lisäosa
JavaServer Faces	Java	JSP tai Facelet	Sisäänrakennettu Ajax-tuki
Apache Wicket	Java	HTML	Ajax-komponentit
ZK	Java	ZUML	Automaattinen
Lift	Scala	Lift-malline (HTML ja Lift-tagit)	Liftin SHtml-olion Ajax-metodit ja -komponentit
Iliad	Smalltalk	Smalltalk (luo XHTML:n automaattisesti Iliad-elementeistä)	Automaattinen (lähettää normaalin pyynnön jos Javascript ei ole käytettävissä)

8.1 Django

Django on WWW-sovelluskehys Python-kielelle. Se seuraa löyhästi Malli-Näkymä-Ohjain-mallia (MVC) ja voidaan kuvata niin sanotulla Malli-Malline-Näkymä-mallilla (engl. *Model-Template-View, MTV*) [40, s. 5] [37, s. 21].

Djangon ominaisuuksia [40, s. 8, 9]:

- **Komponentit:** Django tarjoaa valikoiman integroituja komponentteja.
- **Lomake-API:** Lomakkeiden muokkaamiseen ja validoimiseen.
- **Todentaminen (engl. *authentication*)**: Laajennettava todentamisjärjestelmä.
- **Välimuistijärjestelmä (engl. *caching system*)**: Sovellusten suorituskyvyn nopeuttamiseen.
- **RSS-kehys:** RSS-syötteiden luomiseen.
- **URL-suunnittelu:** Django antaa kehittäjän määritellä kaavoja URLia varten. Kaavoja hallitaan Python-funktioilla. Näin voidaan saada aikaan sekä käyttäjäettä hakukoneystävällisiä osoitteita.
- **Kehitysympäristö:** Djangon kehitysympäristö sisältää kevyen WWW-palvelimen kehitystä ja testausta varten. Testaustilassa (engl. *debugging mode*) se tarjoaa läpikotaiset ja yksityiskohtaiset virheviestit.
- **Tuki monikielisyydelle:** Djangossa on sisäänrakennettu kansainvälistämisjärjestelmä (engl. *internationalization system*).
- **Ylläpitokäyttöliittymä (engl. *administration interface*)**: Djangon ylläpitokäyttöliittymä tekee sovelluksen datan hallitsemisesta helpompaa. Django voi luoda automaattisesti WWW-sivun, joka antaa todennettujen käyttäjien lisätä, poistaa ja muokata olioita [22].

8.1.1 Malli

Djangon malli käyttää Ruby on Railsin tavoin ORMia tietokannan ja Python-olioiden yhdistämiseen [37, s. 21]. Se tukee monia tietokantajärjestelmiä, joiden vaihtaminen tapahtuu konfigurointitiedostoa muuttamalla [40, s. 8]. Siis toisin kuin Railsissa, Djangossa ORM vaatii konfigurointitiedoston.

8.1.2 Näkymä

Djangossa näkymä tyypillisesti palvelee tiettyä toimintoa ja sillä on tietty malline [22]. Näkymä hakee datan, lataa mallineen ja muuntaa sen koodin datan avulla selai-

men tulkittavaan muotoon. Näkymä on Djangossa Python-funktio, jota kutsutaan suorittamaan pyyntöjä (ja palauttamaan HTTP-vastaus). [50, s. 9] [22]

8.1.3 Malline

Newmanin [50, s. 8] mukaan Djangon malline erottaa esityksen sovelluslogiikasta. Se on suunniteltu niin, että se soveltuu sekä ulkoasusuunnittelijoille että ohjelmoijille: Se on helppo oppia myös "ei-ohjelmoijille".

Mallineessa on dataa varten "paikanpitäjiä" (engl. *placeholders*), jotka korvataan oikeilla arvoilla kun malline käännetään. Kun lopputulos lopulta lähetetään selaimelle, ei siinä ole enää mallinetta näkyvissä vaan se on puhdasta HTML-koodia. Myös silmukoiden ja ehtolauseiden käyttö on mahdollista mallineessa: Tämä logiikka ajetaan käännöksen yhteydessä ja niistä saatu data lisätään oikeille paikoille. [50, s. 8-9]

8.1.4 Ajax

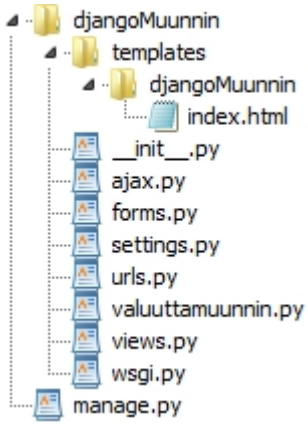
Django-dajax on työkalu asynkronisen käyttöliittymälogiikan tekemiseen Pythonilla. Se käyttää *django-dajaxice*-työkalua kommunikaation luomiseen palvelimen ja asiakkaan välille. Dajaxin avulla voi muokata sivun DOM-rakennetta suoraan Python-koodilla. Dajax tukee neljää suosituinta Javascript-kirjastoa, joiden avulla tarvittavat Javascript-toiminnot saa toteutettua: Prototype, jQuery, Dojo ja mootols. [20]

Dajaxice on Ajax-kirjasto (Javascript-kehys) Djangolle. Se luo asynkronisen yhteyden Django-palvelimen ja Javascript-koodin välille. Se pyrkii erottamaan esityslogiikan ja palvelimen logiikan toisistaan: Ajax-funktioissa ei ole esityslogiikan koodia. Dajaxice ei vaadi muiden Javascript-kirjastojen käyttöä, se sisältää itse kaiken oleellisen. [21]

8.1.5 Esimerkki

Kuvio 7 esittää sovelluksen kansiorakenteen.

index.html: Malline valuuttamuuntimelle. Luodaan lomake, johon otetaan näky-



Kuvio 7: Django valuuttamuunnin-sovelluksen kansiorakenne

mältä vastaan `csrf_token` , jota tarvitaan jotta voidaan luoda Ajax-yhteys sekä `valuuttaField` , joka on select-kenttä sisältäen valuuttojen nimet. Javascriptin avulla otetaan Dajax-yhteys palvelimelle, kun nappia painetaan:

```
<body>
<form action="" method="post">
   {% csrf_token %}
   <p>
      <input type="text" value="" id="textSumma">
      {{ valuuttaField }}
      <input type="button" value="Muunna" onclick="muunna();"> =
      <label id="labelTulos">0 &euro;</label>
   </p>
</form>

<script type="text/javascript">
function muunna(){
   Dajaxice.djangoMuunnin.laskeTulos(Dajax.process, {'valuutta':$('
      #id_selectValuutta').val(), 'summa':$('#textSumma').val()});
}
</script>
</body>
```

forms.py: Luodaan select-kenttä, johon haetaan valuuttojen nimet. Kenttä luodaan *Django forms*-komponentin avulla:

```
class ValuuttaSelectField(forms.Form):
   muunnin = ValuuttaMuunnin()
   valuutat = muunnin.getValuuttaNimet()

   VALUUTTA_VALINNAT = [(valuutta, valuutta) for valuutta in
      valuutat]

   selectValuutta = forms.ChoiceField(choices=VALUUTTA_VALINNAT,
      widget=forms.Select(), label='')
```

ajax.py: Otetaan Ajax-kutsu vastaan. Lasketaan ja palautetaan muunnos valuutalle:

```
@dajaxice_register
def laskeTulos(request, valuutta, summa):
   dajax = Dajax()

   muunnin = ValuuttaMuunnin()
   tulos = muunnin.muunnaValuutta(valuutta, summa)

   if tulos == "virhe":
      dajax.assign('#labelTulos','innerHTML','Virheellinen syöte!')
   else:
      dajax.assign('#labelTulos','innerHTML',tulos + ' &euro;')

   return dajax.json()
```

views.py: Näkymä valuuttamuuntimelle. Näytetään `index.html` ja välitetään sille `csrf_token` ja select-kenttä `valuuttaField`:

```
def djangoMuunninLomake(request):
   c = {}

   valuuttaField = ValuuttaSelectField()

   c.update(csrf(request))
   c.update({'valuuttaField': valuuttaField})

   return render_to_response('djangoMuunnin/index.html', c)
```

urls.py: Määritellään `djangoMuunninLomake`-näkymälle URL-osoitteeksi `valuuttamuunnin` ja mahdollistetaan, että Ajax-kutsu löytää tarvittavan metodin:

```
dajaxice_autodiscover()

urlpatterns = patterns('',
   url(r'^valuuttamuunnin/$', djangoMuunninLomake),
   url(dajaxice_config.dajaxice_url, include('dajaxice.urls')),
)

urlpatterns += staticfiles_urlpatterns()
```

8.2 JavaServer Faces

JavaServer Faces (JSF) on palvelinpuolen WWW-sovelluskehys Java-kielelle. Se sisältää APIn esimerkiksi käyttöliittymäkomponenttien ja tapahtumien hallitsemiseksi. [18]

JSF:n **hallitut pavut** (engl. *managed beans*) käsittelevät tapahtumat ja päivittävät *Mallin* käyttäjän syötteiden mukaisesti. Ne ovat Java-luokkia, joiden attribuuteilla ja metodeilla yhdistetään sovellus JSF:n tapahtumankäsittelijä APIin. [38, s. 8]

JSF Expression Language -kielen (JSF EL) avulla käyttöliittymäkomponentit yhdistetään hallittujen papujen metodeihin ja propertyihin [38, s. 8]. Esimerkiksi *value="#muunnin.arvo"* liittää elementin sisällön *Muunnin*-pavun *arvo*-propertyyn:

```
<h:inputText id="arvo" value="#{muunnin.arvo}"></h:inputText>
```

Hlavatsin [38, s. 8] mukaan JSF sisältää muuntimet tavallisimmille datatyypeille kuten numeroille, Boolean-arvoille ja päivämäärille sekä tarjoaa mahdollisuuden luoda omia muuntimia. Lisäksi JSF sisältää validaattorit tavallisimmille tarkistuksille ja tarjoaa mahdollisuuden luoda omia validaattoreita.

JSF-näkymän voi luoda kahdella tavalla: Joko JSP:llä tai Facelettien (engl. *Facelet*) avulla. JSP on oletus JSF versioissa 1.x. Faceletit ovat osa JSF 2.0 standardia ja myös suositeltu näkymä-teknologia. [43, s. 43]

8.2.1 Ajax

JSF versiosta 2.0 ylöspäin on Ajax-tuki sisäänrakennettuna. Aiemmissa versioissa Ajax-toiminnallisuudet saa luotua vaikka Apache Myfaces- tai ICEFaces-kehyksen avulla. JSF:n (2.0+) Ajaxissa Ajax-toiminto luodaan `f:ajax`-tagin avulla HTML-koodissa:

```
<h:commandButton value="Muunna">
   <f:ajax execute="arvo selectValuutta" render="tulos" />
</h:commandButton>
```

`execute`-attribuuttiin laitetaan niiden kenttien id:t, joiden arvoja tarvitaan palvelimella ja `render`-attribuutilla kerrotaan mikä elementti päivitetään Ajax-kutsulla. Yllä olevassa koodissa siis palvelimella käytetään `arvo`- ja `selectValuutta`-kenttien arvoja ja kutsulla päivitetään `tulos`-elementti:

```
<h:outputText id="tulos" value="#{muunnin.muunna}" />
```

`tulos`-elementissä `value="#muunnin.muunna"` tarkoittaa, että kun elementti päivitetään, niin suoritetaan `Muunnin`-pavun `Muunna`-metodi ja laitetaan sen paluuarvo `tulos`-elementin sisällöksi.

8.2.2 Apache MyFaces

Apache MyFaces on sovelluskehys, joka luo JavaServer Faces toteutuksen. Se koostuu useista aliprojekteista. **Core**-projekti toteuttaa JSF-standardin. Siinä on osiot eri JSF-versioille. [43, s. 11-12]

Kummelin [43, s. 12-14, 119] mukaan Tomahawk-, Trinidad- ja Tobago-projektit tarjoavat uusia komponenttia JSF-kehykselle:

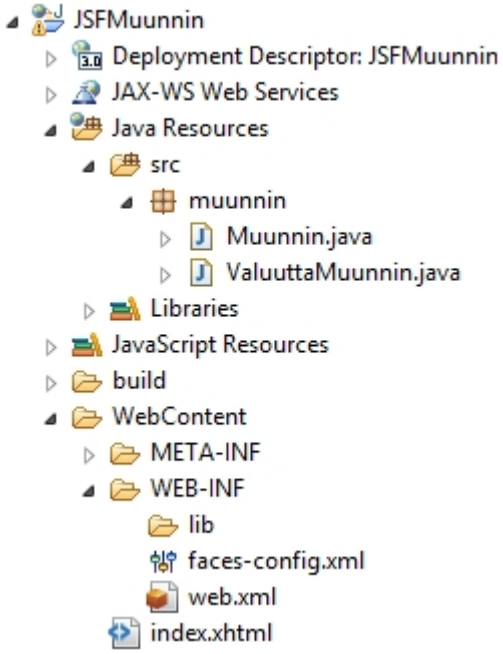
- **Tomahawk** on ollut mukana MyFaces-projektin alusta asti. Se sisältää sekä laajennettuja versioita JSF-komponenteista että täysin uusia komponentteja (esimerkiksi CAPTCHA-komponentti).
- **Trinidad** tarjoaa suuren määrän lisäkomponentteja erityisesti rikkaiden WWW-sovellusten kehittämisen avuksi. Se sisältää Ajax-toiminnallisuuksia kuten sivun osittaisen päivittämisen sekä esimerkiksi dialogin. Komponenttien ulkoasua voi muokata skinien (engl. *skins*) avulla.
- **Tobago** keskittyy erottamaan sovelluksen rakenteen ulkoasusta. Myös se tarjoaa Trinidadin tavoin suuren määrän lisäkomponentteja. Tobago sisältää neljä erilaista teemaa, joista sivun tyylin voi valita.

8.2.3 ICEfaces

ICEFaces on Ajax-lisäosa JavaServer Facesille. Se on komponenttikirjasto, joka sisältää AJAX-käytettäviä versioita JSF-komponenteista. Näin saa luotua Ajax-sovelluksia JSF:llä ilman Javascript-koodia. [38, s. 195]

8.2.4 Esimerkki

Esimerkkisovellus luodaan JSF:n Ajax-tuen avulla (ilman Apache MyFaces- tai ICEfaces-lisäosia). Kuvio 8 esittää sovelluksen projektirakenteen.



Kuvio 8: JavaServer Faces valuuttamuunnin-sovelluksen projektirakenne

index.xhtml: Luodaan JSF-sivu XHTML:llä. Aluksi määritellään nimiavaruudet, joiden avulla JSF-tageja saadaan käytettyä:

```
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html">
```

Luodaan sivun sisältö ja Ajax-toiminnallisuus. `h:panelGrid`-tagin avulla saadaan sopivat välit lomake-elementeille `cellspacing`-attribuutilla:

```
<h:body>
   <h3>ValuuttaMuunnin JSF:llä</h3>

   <h:form>
      <h:panelGrid id="panel" columns="4" border="0" cellpadding="0
          " cellspacing="6">

         <h:inputText id="arvo" value="#{muunnin.arvo}"></h:
             inputText>

         <h:selectOneMenu id="selectValuutta" value="#{muunnin.
             selectValuutta}">
```

```
            <f:selectItems value="#{muunnin.valuutat}" />
         </h:selectOneMenu>

         <h:commandButton value="Muunna">
            <f:ajax execute="arvo selectValuutta" render="tulos" />
         </h:commandButton>

         <h:outputText id="tulos" value="#{muunnin.muunna}" />
      </h:panelGrid>
   </h:form>
</h:body>
```

Muunnin.java: Luodaan hallittu papu. Ensin kerrotaan annotaatioden avulla, että kyseessä on hallittu papu (onnistuu vain JSF 2.0+ -versioissa, aiemmissa pitää käyttää `faces-config.xml`-tiedostoa) ja lisätään tarvittavat importit:

```
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean
@SessionScoped
public class Muunnin implements Serializable {
...
```

Luodaan getterit ja setterit luokan propertyille, joihin JSF-sivun elementit yhdistetään:

```
public String getArvo() {
   return arvo;
}
public void setArvo(String value) {
   this.arvo = value;
}

public String getSelectValuutta() {
   return selectValuutta;
}
public void setSelectValuutta(String value) {
   this.selectValuutta = value;
}
```

Papu hakee valuuttojen nimet `ValuuttaMuunnin`-luokalta ja laittaa ne `valuutat`-listaan, johon JSF-sivun valintalista on yhdistetty. Nimet ovat listassa `SelectItem`-olioina, jotka voi suoraan lisätä JSF-valintalistaan:

```
public Muunnin() {
   muunnin = new ValuuttaMuunnin();
   String[] nimet = muunnin.getValuuttaNimet();

   valuutat = new ArrayList<SelectItem>();
   for (String nimi : nimet) {
```

```
        valuutat.add(new SelectItem(nimi));
    }
}
```

Pavun metodi `Muunna` muuntaa valuutan ja palauttaa tuloksen tai virheen, jos syöte on ollut vääränlainen. Huomaa, että metodin nimi alkaa sanalla `get` ja loppuosa alkaa isolla kirjaimella. JSF-sivulla tätä metodia kutsutaan nimellä `muunna`:

```
public String getMuunna() {
    if (arvo.equals("")) return "0,00 euroa";

    String tulos = muunnin.muunnaValuutta(selectValuutta, arvo);

    if (tulos.equals("virhe")) return "Virheellinen luku!";

    return tulos + " euroa";
}
```

8.3 Apache Wicket

Wicket-sovellus luodaan HTML- ja Java-koodeilla. Wicket-komponentit yhdistetään HTML-puolella Java-koodiin *Wicket id* -attribuuttien avulla. Wicketissä HTML on puhdasta HTML-koodia ja Java puhdasta Java-koodia. [1] Wicket tähtää yksinkertaisuuteen: Siinä ei ole konfigurointitiedostoja vaan se on yksinkertaisesti luokkakirjasto [46].

Wicket ei lisää mitään omaa syntaksia HTML-kieleen vaan se laajentaa sitä omalla nimiavaruudella, joka on täysin XHTML-standardin mukainen. Eli sivun suunnittelija voi käyttää mitä tahansa HTML-editoria sivujen ja Wicket-komponenttien luomiseen. [46]

Wicketissä myös HTML-tiedosto tulee Java-tiedostojen kanssa samaan pakettiin, eikä `webapp`- tai `webcontent`-pakettiin kuten Javan dynaamisissa WWW-projekteissa yleensä.

8.3.1 Komponentit ja niiden käyttäminen

Wicket käyttää id-attribuuttia Wicket-nimiavaruudessa ("`wicket:id`") merkkaamaan HTML-tagit, joita kuuluu kohdella Wicket-komponentteina [46]. Esimerkiksi HTML-koodissa annetaan elementille id seuraavasti:

```
<span wicket:id="message">0</span>
```

ja siihen viitataan Java-koodissa seuraavasti:

```
Label label = new Label("message", "0");
```

Locken [46] mukaan näitä HTML-koodin wicket-attribuutteja ei ole pakko näyttää loppukäyttäjälle, sillä Wicket sisältää asetuksen, jolla ne siivotaan pois koodista, ja jäljelle jää puhdas HTML-koodi.

Jokainen Wicket-komponentti, joka HTML-sivulle lisätään, pitää lisätä sivulle myös Java-tiedoston puolella (esimerkiksi `form.add(buttonMuunna)`).

Wicketissä uusien komponenttien luominen on tehty mahdollisimman helpoksi. Sillä voi laajentaa tai yhdistää olemassa olevia komponentteja tai luoda täysin uusia komponentteja. Nämä komponentit pakataan JAR (*Java Archive*) [9] -tiedostoon ja niiden käyttäminen onnistuu lisäämällä JAR-tiedosto sovelluksen `lib`-kansioon. [46]

8.3.2 Ajax

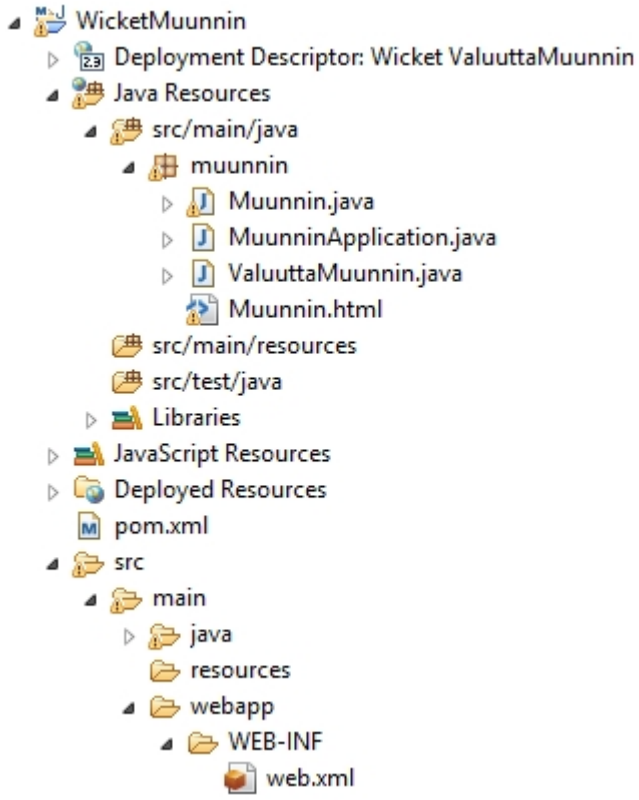
Wicket tarjoaa Ajax-versioita tavallisista komponenteista, kuten linkistä ja napista, mutta myös täysin Ajax-toiminnoille suunniteltuja komponentteja. Tavallisen komponentin Ajax-version saa lisäämällä sen nimen eteen "Ajax": Esimerkiksi `AjaxButton` ja `AjaxLink`. `AjaxLink`-komponenttia käytetään `OnClick`-metodin kautta ja `AjaxButton`-komponenttia `OnSubmit`-metodin kautta. `AjaxButton`-komponentin (kuten myös `Button`-komponentin) pitää olla `Form`-komponentin sisällä, jotta OnSubmit toimii. Ajax-komponenttien tapahtumankäsittelijät vastaanottavat `AjaxRequestTarget`-olion, jonka `add`-metodilla voidaan kertoa Wicketil-

le, mitkä komponentit kyseisellä Ajax-kutsulla päivitetään uudelleen (esimerkiksi `target.add(labelTulos);`). `OnSubmit`-metodi vastaanottaa lisäksi `Form`-parametrina lomakkeen, johon nappi kuuluu. [19, s. 136-137]

Toisin kuin esimerkiksi JSF-kehyksessä, Wicketissä Ajax-toiminnot hoidetaan kokonaan palvelimen puolella, eikä HTML-sivuun tarvitse sen takia tehdä mitään muutoksia.

8.3.3 Esimerkki

Kuvio 9 esittää sovelluksen projektirakenteen.



Kuvio 9: Wicket valuuttamuunnin-sovelluksen projektirakenne

Muunnin.html: HTML-elementeille annetaan `wicket:id`-arvot, jolloin niistä saadaan Wicket-komponentteja:

```
<body>
<form wicket:id="form">
   <input type="text" wicket:id="textSumma"/>
```

```
   <select wicket:id="selectValuutta"></select>
   <input type="button" value="Muunna" wicket:id="buttonMuunna"/>
   <span wicket:id="labelTulos">0</span>
</form>
</body>
```

Muunnin.java: Kaikki alla oleva sisältö tulee `public Muunnin()` -konstruktoriin. Ensin luodaan lomake, tekstikenttä ja label. Annetaan niille samat id:t kuin HTML-puolella:

```
Form form = new Form("form");
textSumma = new TextField<String>("textSumma", Model.of(""));
labelTulos = new Label("labelTulos", "0 euroa");
// Mahdollistaa tuloksen päivittämisen AJAX:illa.
labelTulos.setOutputMarkupId(true);
```

Luodaan valuuttavalintalista. `new PropertyModel`-oliolla asetetaan valintalistan alkioiksi `valuutat`-listan alkiot ja oletusvaluutaksi `valittuValuutta`-attribuutin arvo:

```
valuutat = muunnin.getValuuttaNimet();
// Attribuutti, johon laitetaan ensimmäinen valuutta.
valittuValuutta = valuutat.get(0);

selectValuutta = new DropDownChoice<String>(
      "selectValuutta", new PropertyModel<String>(this, "
         valittuValuutta"), valuutat);
```

Luodaan nappi ja sille AJAX-tapahtumankäsittelijä:

```
Button buttonMuunna = new AjaxButton("buttonMuunna"){
   @Override
   public void onSubmit(AjaxRequestTarget target, Form form) {
      String tulos = muunnin.muunnaValuutta(selectValuutta.
         getDefaultModelObjectAsString(), textSumma.
         getDefaultModelObjectAsString());

      if (tulos.equals("virhe")) labelTulos.setDefaultModelObject("
         Virheellinen syöte!");
      else labelTulos.setDefaultModelObject(tulos + " euroa");
      target.add(labelTulos);
   }
};
```

Lisätään Wicket-komponentit sivulle:

```
form.add(textSumma);
form.add(selectValuutta);
form.add(buttonMuunna);
form.add(labelTulos);
```

```
add(form);
```

MuunninApplication.java: `getHomePage()`-metodilla kerrotaan oletussivuksi `Muunnin.class`. Kun sovelluksen URL:ään otetaan yhteys (määritellään `web.xml`-tiedostossa), Wicket palauttaa `Muunnin`-sivun renderöimän koodin:

```
public class MuunninApplication extends WebApplication
{
   @Override
   public Class<? extends WebPage> getHomePage()
   {
      return Muunnin.class;
   }
   @Override
   public void init()
   {
      super.init();
      // add your configuration here
   }
}
```

8.4 ZK

Schumacherin ym. [57, s. 6] mukaan ZK on komponenttipohjainen Ajax-WWW-sovelluskehys Java-kielelle, joka keskittyy erityisesti käyttöliittymäpuoleen. Se sisältää laajan valikoiman XUL- ja XHTML-komponentteja, joita hallitaan tapahtumien avulla. Se tarjoaa myös oman käyttöliittymäkielen ZUML (*ZK User Interface Markup Language*) [57, s. 8]. ZUML-sivun elementteihin päästään käsiksi palvelimella niiden `id`-arvojen avulla.

ZK käyttää siis tapahtumapohjaista mallia, joka hoitaa WWW-sovelluksen pyyntö-vastaus-toiminnot. Kehys itse hoitaa tämän teknisen toteutuksen, joten selaimen puolelle ei tarvitse itse kirjoittaa yhtään Javascript-koodia. Kehys sisällyttää joitakin Javascript-kirjastoja sivulle, ja luo tarvittavan Javascript-koodin automaattisesti. [57, s. 8]

8.4.1 Sivun ulkoasun luominen

ZUML on XML-muotoiltu kieli (muunnos XUL-kielestä) käyttöliittymien suunnitteluun. Sen avulla voidaan sisällyttää sekä XUL- että HTML-koodia samalle sivulle. ZUML-kieleen voidaan myös sisällyttää ohjelmakoodia (toisin kuin XUL-kieleen). [57, s. 8]

Schumacher ym. [57, s. 34-35] esittelevät ZK:n tarjoamia komponentteja myös ulkoasun luomiseen. Tärkeimpiä ovat `vbox`, `hbox`, `box` ja `grid`. Näitä ei kuitenkaan ole pakko käyttää, sillä ulkoasun voi halutessaan luoda pelkästään XHTML- ja CSS-koodeilla.

8.4.2 Kuinka ZK toimii?

ZK-kehyksen pääelementtejä ovat **ZK Loader**, **ZK Client Engine** ja **ZK AU (Asynchronous Update) Engine**: [57, s. 15]

- **ZK Loader:** On vastuussa sivun lataamisesta ja tulkitsemisesta. Se valitsee sivun selaimelta tulevan URL-pyynnön perusteella, jonka se sitten muuntaa HTML-sivuksi ja palauttaa selaimelle.
- **ZK Client Engine:** Lähettää "ZK-pyyntöjä" palvelimelle ja vastaanottaa palvelimelta "ZK-vastauksia". Näiden vastausten perusteella päivitetään sivun DOM-puu, eli tätä voidaan kutsua ZK:n Ajaxin asiakkaan puoleksi.
- **ZK AU (Asynchronous Update) Engine:** ZK:n Ajaxin palvelimen puoli.

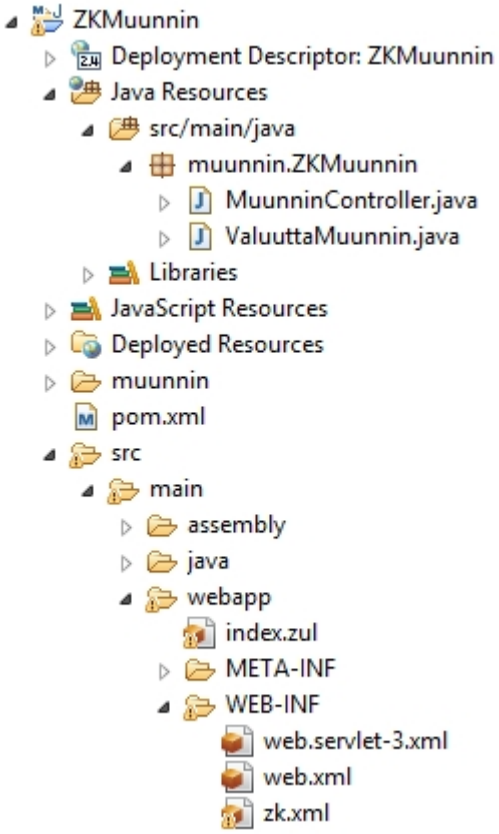
Sovelluksen käyttäjän puoli koostuu HTML-sivuista. ZK-kehyksessä nämä sivut koostuvat **työpöydistä**, **sivuista** ja **komponenteista**: [57, s. 16]

- **Komponentti:** Käyttöliittymäolio (kuten `Button`, `Textbox` tai `Label`). Komponentti on palvelimella Java-olio, jolla on myös visuaalinen esitys selaimessa. Visuaalinen esitys luodaan, kun komponentti kiinnitetään sivuun ja poistetaan, kun kiinnitys sivulta poistetaan.
- **Sivu:** Sisältää komponentteja, eli se on säiliö komponenteille.
- **Työpöytä:** Säiliö sivuille. Sivut, jotka ovat saavutettavissa samasta URL:stä,

liitetään samaan työpöytään.

8.4.3 Esimerkki

Kuvio 10 esittää sovelluksen projektirakenteen.



Kuvio 10: ZK valuuttamuunnin-sovelluksen projektirakenne

index.zul: HTML-elementeille annetaan sopivat id:t. Elementit laitetaan `<hlayout>`-elementin sisään, joka järjestää ne riviin. `<window apply=...>`-tagi kertoo elementtejä käyttävän kontrollerin:

```
<zk>
   <window apply="muunnin.ZKMuunnin.MuunninController">
      <hlayout>
         <textbox id="textSumma" />
         <selectbox id ="selectValuutta"></selectbox>
         <button id ="buttonMuunna" label="Muunna" />
         <label id="labelTulos" value ="0 euroa"/>
      </hlayout>
   </window>
</zk>
```

MuunninController.java: Laajentaa `SelectorComposer<Component>`-luokan, jolloin sillä päästään käsiksi `.zul`-tiedoston elementteihin `@Wire`-annotaation avulla (komponenttien nimet ovat samat kuin `.zul`-elementtien id:t):

```
public class MuunninController extends SelectorComposer<Component>
    {

    @Wire
    private Textbox textSumma;
    @Wire
    private Label labelTulos;
    @Wire
    private Button buttonMuunna;
    @Wire
    private Selectbox selectValuutta;
```

`.zul`-sivun latautumisen jälkeen asetetaan valintalistan alkiot:

```
@Override
public void doAfterCompose(Component comp) throws Exception{
    // Pitää tehdä!
    super.doAfterCompose(comp);

    muunnin = new ValuuttaMuunnin();
    valuutat = muunnin.getValuuttaNimet();

    ListModelList<String> listModel = new ListModelList<String>(
        valuutat);
    // Laitetaan valituksi ensimmäinen valuutta,
    // muuten Selectboxiin tulee tyhjä valinta.
    listModel.addToSelection(valuutat.get(0));

    selectValuutta.setModel(listModel);
}
```

Luodaan napille tapahtumankäsittelijä. `@Listen`-annotaatiolla kerrotaan, että kuunnellaan `buttonMuunna`-napin `onClick`-tapahtumia:

```
@Listen("onClick=#buttonMuunna")
public void doMuunna() {
    String tulos = muunnin.muunnaValuutta(valuutat.get(
        selectValuutta.getSelectedIndex()), textSumma.getValue());

    if (tulos.equals("virhe")) labelTulos.setValue("Virheellinen syö
        te!");
    else labelTulos.setValue(tulos + " euroa");
}
```

9 Vertailu: Kerhon WWW-sovellus eri menetelmillä

Tässä luvussa tehdään vertailu sovelluskehyksistä Ohjelmointi 2 -kurssin mallihar-joitustyön (Kerho-sovelluksen) WWW-version luomisen perusteella. Ensin kerrotaan suunnitelma toteutettavalle kerhosovellukselle ja kerrotaan mitä asioita sovelluske-hyksistä on tarkoitus verrata. Tämän jälkeen kuvataan eri sovelluskehyksillä luotujen kerhosovellusten rakenne. Esimerkkinä sovelluskehysten käyttämisestä esitetään kerhosovellusten jäsenhakulomakkeen luominen ja sen toiminnallisuuden toteuttaminen. Koko kerhosovellusten luominen eri sovelluskehyksillä kuvataan liitteissä A, B, C ja D. Toteutetuissa sovelluksissa voi olla pieniä puutteita esimerkiksi joidenkin tarkistusten kohdalla, mutta ne eivät silloin ole olennaisia itse vertailun kannalta.

Vaadin-kehyksellä käytetään Vesa Lappalaisen tekemän Kerho-sovelluksen koodia. Muille sovelluskehyksille tehtiin oma versio Kerho-sovelluksesta, jotka yritettiin saada muistuttamaan toiminnaltaan Lappalaisen versiota. Myös sovelluskehyksistä kutsuttavien Kerho-sovelluksen metodien nimien pitäisi olla samanlaisia (ellei jopa ihan samoja). Joitakin epäolennaisuuksia jätettiin noista Kerho-sovelluksista kuiten-kin pois (kuten jäsenten lajittelu), kuitenkin niin että kaikki sivujen toiminnallisuudet pystyy testaamaan.

9.1 Suunnitelma

Kerho-sovelluksen WWW-versiossa on kaksi sivua: Kerhosivu ja jäsenen muokkaus-sivu. Kerhosivulla voi hakea jäseniä ja se näyttää jäsenet ja valitun jäsenen tiedot ja harrastukset. Sivulla voi myös lisätä ja poistaa jäseniä. Jäsensivulla voi muoka-ta jäsenen tietoja tai harrastuksia sekä lisätä ja poistaa harrastuksia. Kaikki sivujen toiminnallisuudet toteutetaan Ajaxin avulla (paitsi tietysti linkki toiselle sivulle).

Jokaisella jäsenellä on linkki muokkaussivulle, minne välitetään parametrina jäsenen id. Jäsenen sivulla on myös linkki takaisin kerhoon, minne välitetään myös id. Jä-sensivulta siirryttäessä takaisin kerhosivulle, tulisi sivun näyttää samalta kuin sieltä poistuttaessa. Eli valitun jäsenen lisäksi myös hakutulos on sama.

Kerhosivulla tulisi takaisin-napin ja kirjanmerkkien toimia oikein: Kun on valittu listasta toinen jäsen, vie takaisin-napin painaminen edelliseen jäseneen eikä pois sivulta.

Jäsenen tietojen syöttämislomake pyritään tekemään uudelleenkäytettävänä komponenttia, jota voidaan käyttää sekä uuden jäsenen luomisessa että jäsenen muokkaamisessa. Komponentti vastaanottaa jäsenen, jolta kenttiin otetaan oletusarvot. Komponentilta voi pyytää esimerkiksi muokatun jäsenen ja tiedon onko syöttökentissä virheitä.

Kuvioissa 11, 12, 13 ja 14 näytetään esimerkit valmiista käyttöliittymistä. Kuvat esittävät sovelluksen eri tiloja ja ne on otettu eri sovelluskehyksillä luoduista sovelluksista.

Taulukoissa 6 ja 7 kuvataan tarkemmin sivujen toiminnallisuudet ja niiden vaatimukset. Vertailun kohteet ja vertailtavat asiat kuvataan taulukossa 8.

Taulukko 6: Kerhosivun vaatimukset

Toiminto	**Vaatimukset**
Jäsenten haku	- Haku tehdään annetun hakutekstin ja valitun hakukentän perusteella. - Haun tulokset päivitetään jäsenvalintalistaan. - Näytetään listan ensimmäisen jäsenen tiedot. - Hakuun lisätään viive, jonka aikana näytetään tieto, että tietoja haetaan ja käyttäjän pitää odottaa (mielellään pyörivällä ilmaisimella). - Haun jälkeen ilmaistaan, että jäsenlista on muuttunut.
Jäsenen valinta	- Valitun jäsenen tiedot ja harrastukset näytetään listan vierellä. - Tietojen otsikkorivillä on linkki jäsenen muokkaussivulle.

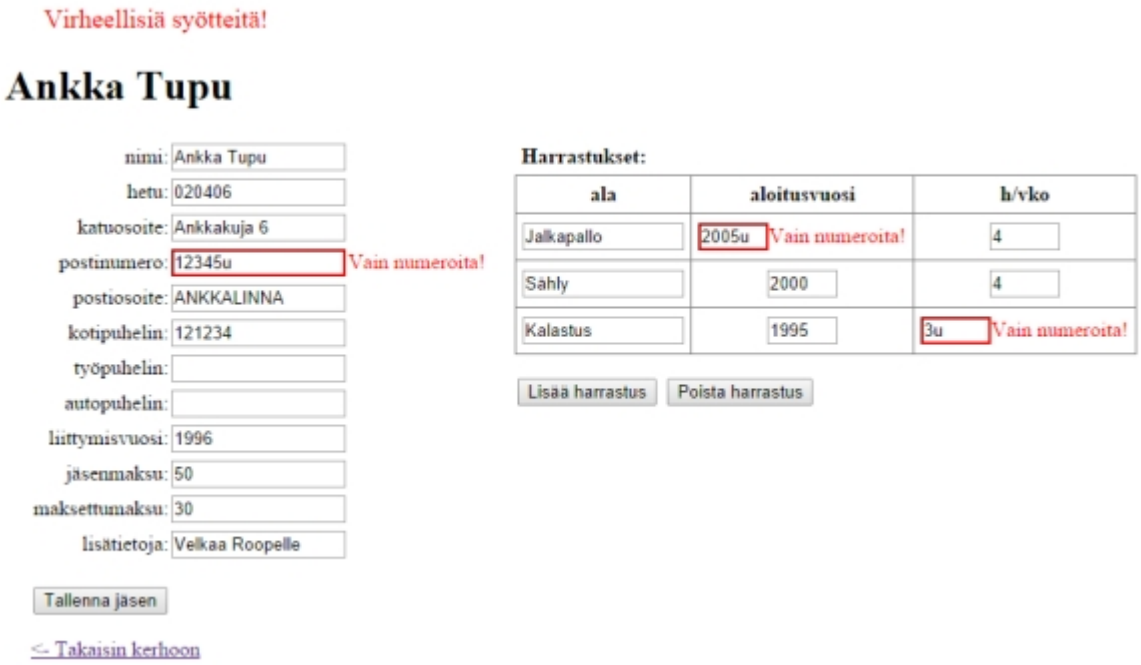
Jäsenen lisäys	- Jäsen lisätään dialogin avulla joka avautuu napista. - Käyttäjän syötteet tarkistetaan ja virheistä ilmoitetaan välittömästi. - Jos syötteissä on virheitä ei lisääminen onnistu. - Jäsenen nimi ei saa olla tyhjä. - Tallennusnappia painettaessa ilmoitetaan jos syötteissä on virheitä tai nimi on tyhjä. - Lisäyksen jälkeen ilmaistaan, että jäsenlista on muuttunut.
Jäsenen poisto	- Valittu jäsen poistetaan napilla. - Poisto varmistetaan käyttäjältä. - Poiston jälkeen ilmaistaan, että jäsenlista on muuttunut.

Taulukko 7: Jäsensivun vaatimukset

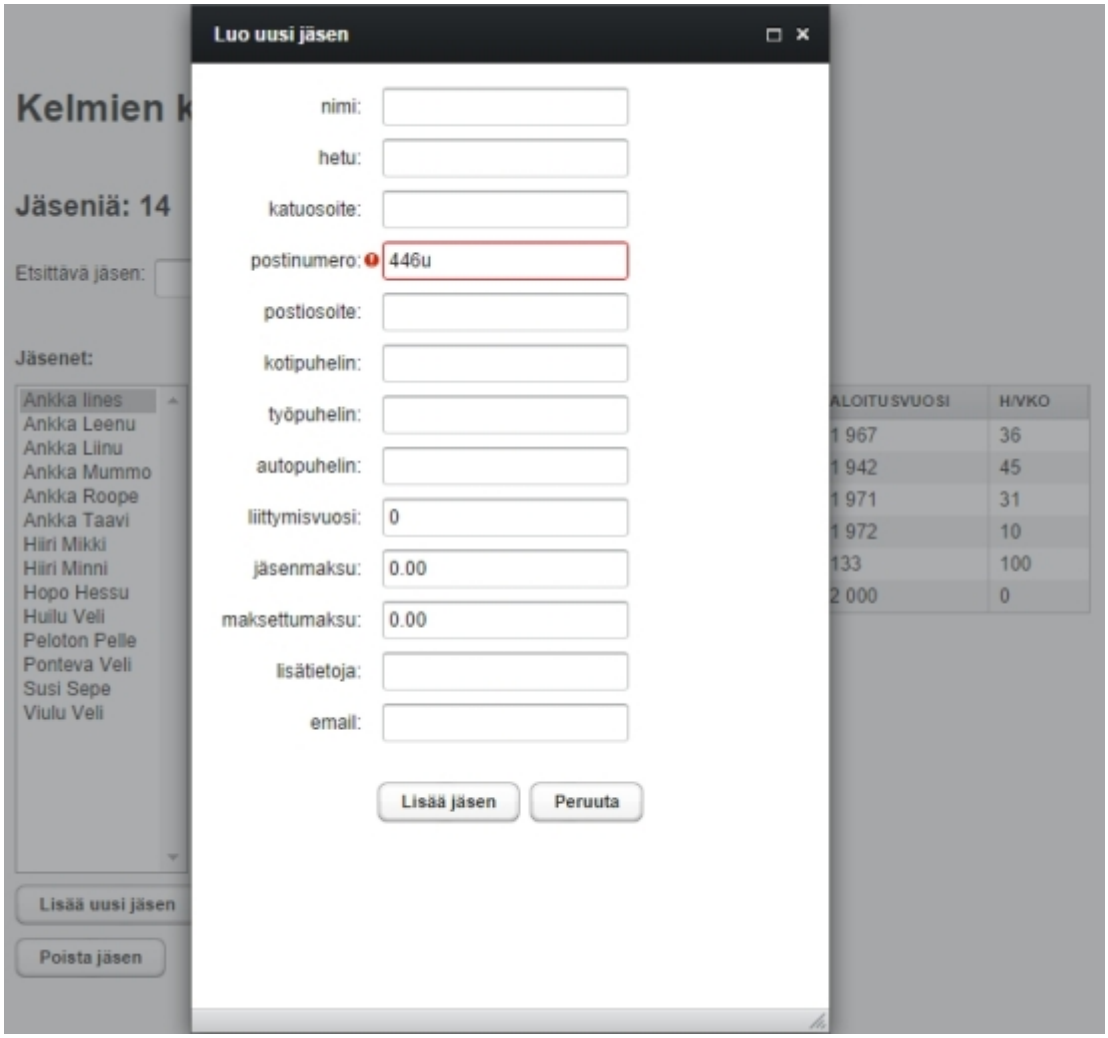
Toiminto	**Vaatimukset**
Jäsentietojen muokkaus	- Jäsenen tietoja voi muokata tekstikentillä, joissa näytetään vanha arvo.
Harrastusten muokkaus	- Jäsenen harrastuksia voi muokata taulukossa olevilla tekstikentillä, joissa näytetään vanha arvo.

Syötteiden tarkistus ja tallentaminen	- Jäsenen tiedot ja harrastukset tallennetaan samalla napilla. - Myös uudet ja poistetut harrastukset tallennetaan kerhoon vasta tallenna-napilla. - Käyttäjän syötteet tarkistetaan ja virheistä ilmoitetaan välittömästi. - Jos syötteissä on virheitä ei tallentaminen onnistu. - Jäsenen nimi ei saa olla tyhjä. - Tallennusnappia painettaessa ilmoitetaan, jos syötteissä on virheitä tai nimi on tyhjä. - Tallennuksen onnistumisesta (tai epäonnistumisesta) näytetään ilmoitus.
Harrastuksen poisto	- Aktiivisena olevan (joku harrastuksen kenttä klikattuna) harrastuksen voi poistaa. - Poisto varmistetaan käyttäjältä. - Poiston jälkeen ilmaistaan, että harrastustaulukko on muuttunut.
Harrastuksen lisäys	- Jäsen lisätään dialogin avulla joka avautuu napista. - Syötteet tarkistetaan kuten uuden jäsenen tapauksessa, paitsi että ala ei saa olla tyhjä. - Lisäyksen jälkeen ilmaistaan, että harrastustaulukko on muuttunut.

Kuvio 11: Kerhosivun käyttöliittymä Seaside-kehyksellä



Kuvio 12: Jäsensivun käyttöliittymä ASP.NET-kehyksellä



Kuvio 13: Kerhosivun jäsenenlisäysdialogi Vaadin-kehyksellä

Kuvio 14: Jäsensivun harrastuksenlisäysdialogi Ruby on Rails -kehyksellä

Taulukko 8: Vertailukohteet

Kohde	**Lisätietoja**
Ajax-toiminnallisuus	- Tarvittavat lisäosat/konfiguraatiot. - Kutsun tekeminen palvelimelle. - Käyttöliittymän päivittäminen.
Dialogit	- Oman dialogin luominen. - Dialogin ulkoasun muokkaaminen. - Varmistusdialogin näyttäminen.
Odotusilmaisimen näyttäminen	- Ilmaisimen päivittäminen sivulle - Pyörivän ilmaisimen lisääminen.
Kirjanmerkkien ja takaisinnapin toimimisen toteuttaminen	- Uri-fragmentin asettaminen ja sen muutoksien käsittely.
Ilmoitukset käyttäjälle	- Käyttöliittymän muutosten näyttäminen (animaatiot). - Ilmoitukset esimerkiksi tallennuksen onnistumisesta.
Komponenttien käyttäminen	- Tapahtumat. - Arvojen saaminen. - Komponenttien muokkaaminen.
Uudelleenkäytettävän komponentin luominen	- Oman jäsentietolomake-komponentin luominen.
Ulkoasun luominen	- Ulkoasukieli (HTML vai joku muu?) - Tyylien käyttäminen. - Visuaalinen editori?
Toiselle sivulle siirtyminen	- Uuden sivun luominen ja sen osoite. - Sessioiden käyttö. - Request-parametrien saaminen.
Tarvittavat kielet	- Mitä kieliä sovelluksen tekemiseen pitää osata.
Suorituskyky	- Sovelluksen eri toimintojen suoritusajat.

9.2 Vaadin-kerhosovelluksen rakenne ja esimerkki

Vaadin-kehyksen kerhosovellus koostuu kahdesta UI-luokasta: `Vaadinkerho.java` on kerhosivun luokka ja `Vaadinjasen.java` jäsensivun luokka. Sivujen sisällöt luotiin *Vaadin composite* -komponentteina (`Kerhosivu.java` ja `Jasensivu.java`), joiden avulla ulkoasu saatiin luotua Vaadin-kehyksen visuaalisella editorilla. Jäsentietolomake luotiin uudelleenkäytettävänä komponenttina (`JasenTietoLomake.java`), joka on myös *Vaadin composite* -komponentti. Kuvio 15 näyttää sovelluksen projektirakenteen.

Vaadinkerho.java-tiedostossa luodaan UI-luokka kerhosivulle. `init`-metodissa luodaan `VerticalLayout`-ulkoasukomponentti (`layout`), joka asetetaan sivun sisällöksi. Ulkoasukomponentille lisätään uusi `Kerhosivu`-komponentti, jolle välitetään *request*-parametri `id`:

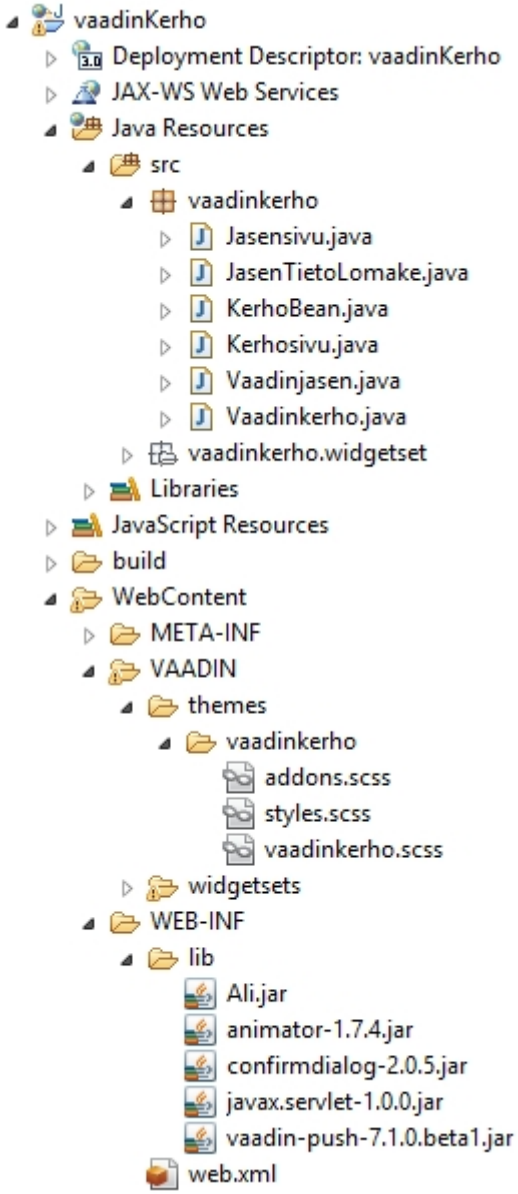
```
protected void init(VaadinRequest request) {
    final VerticalLayout layout = new VerticalLayout();
    layout.setMargin(true);
    setContent(layout);
    layout.setSizeFull();

    String id = request.getParameter("id");

    Kerhosivu sivu = new Kerhosivu(id);
    layout.addComponent(sivu);
}
```

Kerhosivu.java-tiedostossa luodaan *Vaadin composite* -komponentti, jossa luodaan kerhosivun sisältö. Sivun perusulkoasu luotiin visuaalisella Vaadin editorilla (kuvio 16).

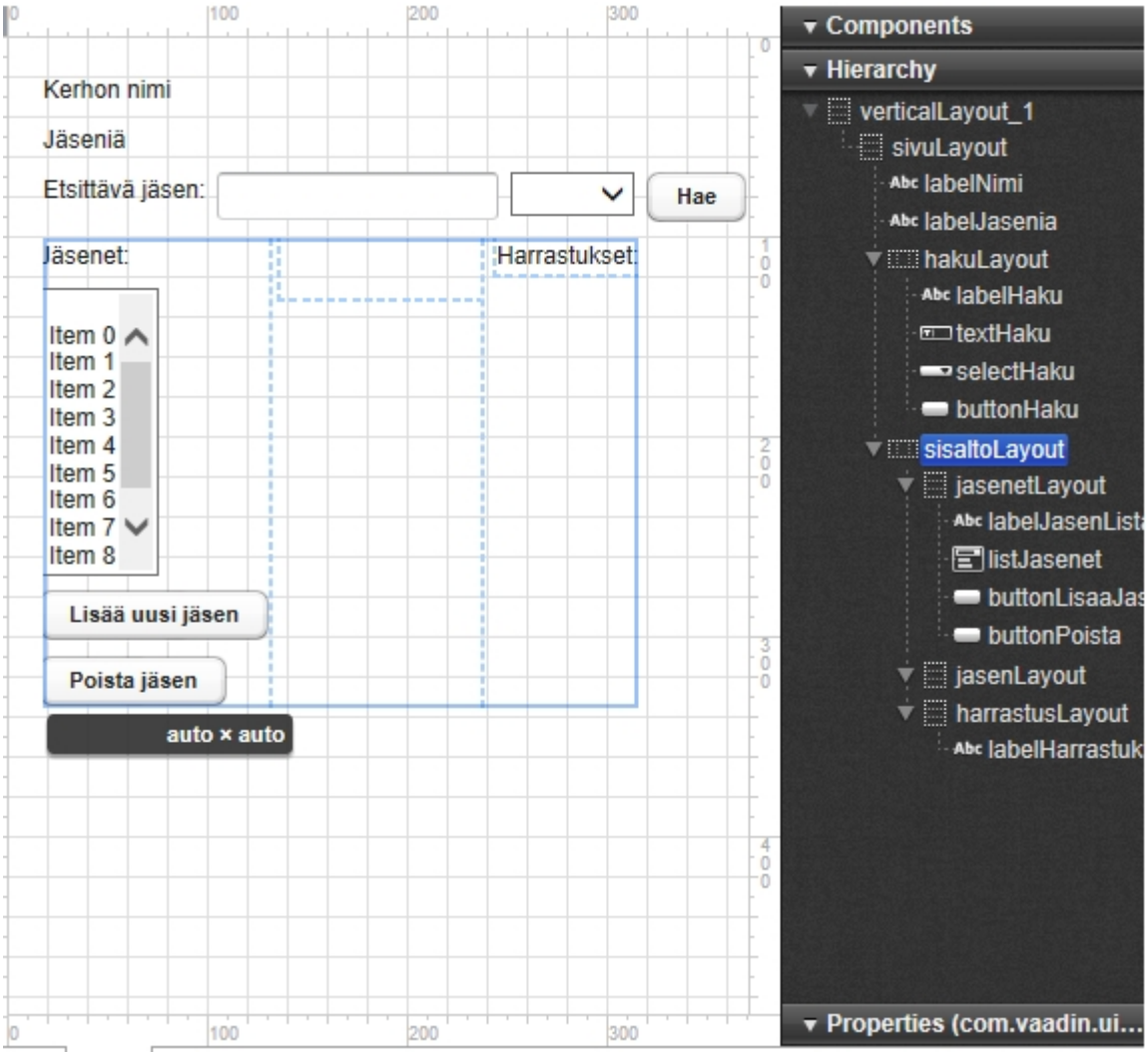
Editorilla siis luotiin hakulomaketta varten `HorizonalLayout`-ulkoasukomponentti (`hakuLayout`), jonka sisään hakulomakkeen elementit (*Label* `labelHaku`, *TextField* `textHaku`, *NativeSelect* `selectHaku` ja *Button* `buttonHaku`) sijoitettiin.

Konstruktorissa luodaan `selectHaku`-komponentille alkiot ja lisätään hakunapille klikkaustenkuuntelija. Luodaan myös uusi `odotusilmaisin`-olio. Luodaan myös `kerho`-olio ja luetaan sen tiedot tiedostosta:

```
public Kerhosivu(String id) {
    buildMainLayout();
```



Kuvio 15: Vaadin-kerhoprojektin kansiorakenne



Kuvio 16: Vaadin-kerhosivun visuaalinen editori -näkymä

```
    setCompositionRoot(mainLayout);

    ...

    odotusIlmaisin = new ProgressBar();
    odotusIlmaisin.setIndeterminate(true);
    odotusIlmaisin.setEnabled(true);

    kerho = KerhoBean.getKerho();
    try {
       kerho.lueTiedostosta("kelmit");
    } catch (SailoException e) {
       e.printStackTrace();
    }

    ...

    apuJasen = new Jasen();
    luoHakukentat();
    buttonHaku.addClickListener(this);

    ...
}
```

Lisätään napeille kuuntelijat. Painettu nappi saadaan parametrina vastaanotetun tapahtuman metodilla `event.getButton()`. Hakunapin tapauksessa otetaan hakukenttä `selectHaku`-komponentilta ja hakuehto `textHaku`-komponentilta ja suoritetaan haku:

```
@Override
public void buttonClick(ClickEvent event) {
    // Hakunappi
    if (event.getButton() == buttonHaku) {
       int hakukentta = Integer.parseInt(selectHaku.getValue().
          toString());
       String hakuehto = textHaku.getValue();
       haeJasenet(hakuehto, hakukentta);
    }

    ...
}
```

Luodaan alkiot `selectHaku`-komponentille. Metodi `setNullSelection Allowed(false)` poistaa valintalistan alusta tyhjän rivin, `setItemCaption(i, apuJasen.getKysymys(i))` asettaa alkiolle näkyväksi tekstiksi kentän kysymyksen, kun varsinainen arvo on kentän numero ja `setValue(apuJasen.ekaKent ta())` asettaa valituksi alkioksi ensimmäisen kentän:

```
public void luoHakukentat() {
    selectHaku.setNullSelectionAllowed(false);
    for (int i = apuJasen.ekaKentta(); i < apuJasen.getKenttia(); i
       ++) {
```

```
        selectHaku.addItem(i);
        selectHaku.setItemCaption(i, apuJasen.getKysymys(i));
    }
    selectHaku.setValue(apuJasen.ekaKentta());
}
```

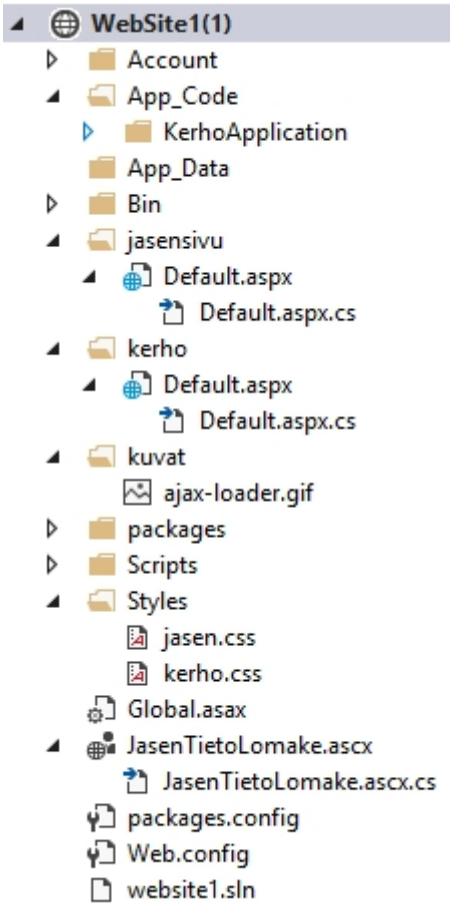
Haetaan jäsenet hakuehdon ja hakukentän mukaan. Hakuun lisätään viive jonka aikana näytetään odotusilmaisin. Odotusilmaisin päivitetään käyttöliittymään *Vaadin Push* -kirjaston avulla [34, luku 11.16]. Odotusilmaisin poistetaan sivulta haun jälkeen ja poisto päivittyy sivulle automaattisesti Ajax-kutsun suorituksen jälkeen. Päivitetään jäsenlista uudelleen sivulle ja asetetaan sessioon *hakuehto*- ja *hakukentta*-muuttujat:

```
public void haeJasenet(String hakuehto, int hakukentta) {
    Label labelOdota = new Label("Jäseniä haetaan...");
    hakuLayout.addComponent(labelOdota);
    hakuLayout.addComponent(odotusIlmaisin);
    UI.getCurrent().push();

    try {
        Thread.sleep(3000);
    } catch(InterruptedException ex) {
        Thread.currentThread().interrupt();
    }

    jasenLista = (List<Jasen>) kerho.etsi("*" + hakuehto + "*",
        hakukentta);

    hakuLayout.removeComponent(labelOdota);
    hakuLayout.removeComponent(odotusIlmaisin);

    if (jasenLista.size() > 0) {
        paivitaJasenlista(jasenLista.get(0).getTunnusnro());
    }
    else paivitaJasenlista(0);

    VaadinSession.getCurrent().setAttribute("hakuehto", hakuehto);
    VaadinSession.getCurrent().setAttribute("hakukentta", Integer.
        toString(hakukentta));
}
```

9.3 ASP.NET-kerhosovelluksen rakenne ja esimerkki

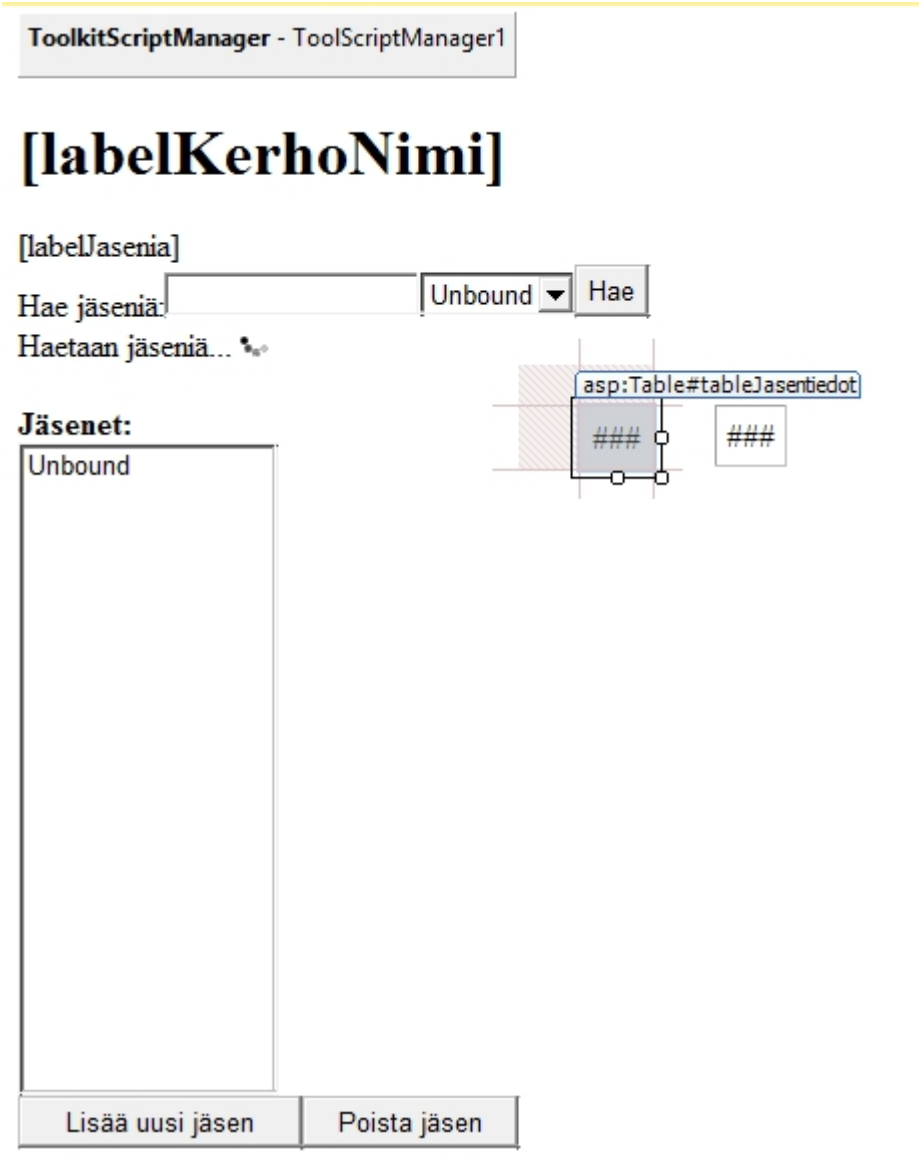
ASP.NET-kerhosovellus koostuu kahdesta WWW-lomakesivusta (`kerho` ja `jasensivu`) ja koostetusta komponentista (*Web User Control*) `JasenTietoLomake`. Kuvio 17 esittää sovelluksen kansiorakenteen.

Kerhosivu on ASP.NET WWW-lomakesivu, joka koostuu **Default.aspx**-ulkoasutie-



Kuvio 17: ASP.NET-kerhosovelluksen kansiorakenne

dostosta ja **Default.aspx.cs**-kooditiedostosta. Kuvio 18 esittää Visual Studion visuaalisen editorin näkymän kerhosivusta (*Default.aspx*).



Kuvio 18: ASP.NET-kerhosivun Visual Studion editorinäkymä

Default.aspx-tiedostossa luodaan sivun perusulkoasu ja Ajax-toiminnallisuus. Myös animaatiot luodaan tässä tiedostossa. Komponentit voi lisätä sivulle myös visuaalisella editorilla, mutta sillä ei voi luoda HTML-elementtejä (paitsi `div`-elementin).

Luodaan hakulomake, joka laitetaan `UpdatePanel`-kontrollin sisään, jolloin napin painallus välitetään asynkronisesti. Erillistä `trigger`-elementtiä ei siis tarvitse mää-

ritellä. Kyseinen `UpdatePanel` liitetään `UpdateProgress`-kontrolliin, jolloin odotusilmaisin näkyy automaattisesti kyseisen `UpdatePanel`-kontrollin päivittymisen aikana. Kaikki tämän hakulomakkeen elementit ovat `UpdatePanel`-kontrollin sisällä, koska se tekee automaattisesti rivinvaihdon, ja jonkun elementin sijoittaminen sen ulkopuolelle rikkoisi ulkoasun. Kaikki lomakkeen elementit ovat ASP.NET-kehyksen WWW-palvelinkontrolleja. Napille määritellään tapahtumankuuntelija klikkauksille (`OnClick="buttonHae_Click"`), tämän pystyy määrittelemään myös visuaalisen editorin kautta:

```
<asp:UpdatePanel ID="updatePanelHaku" UpdateMode="Conditional"
    runat="server">
    <ContentTemplate>
        <asp:Label ID="labelHae" runat="server" Text="Hae jäseniä:"
            ></asp:Label>
        <asp:TextBox ID="textHae" runat="server"></asp:TextBox>
        <asp:DropDownList ID="selectKentta" runat="server">
        </asp:DropDownList>
        <asp:Button ID="buttonHae" runat="server" Text="Hae" OnClick=
            "buttonHae_Click" />
    </ContentTemplate>
</asp:UpdatePanel>
```

Luodaan `UpdateProgress`-kontrolli, joka liitetään edellisen `UpdatePanel`-kontrollin päivittymiseen `AssociatedUpdatePanelID`-attribuutin avulla. `ProgressTemplate`-elementtiin annetaan sisältö, joka näytetään päivittymisen aikana. Tässä siihen lisätään teksti ja pyörivä latauskuva:

```
<asp:UpdateProgress ID="UpdateProgressHaku" AssociatedUpdatePanelID
    ="updatePanelHaku"
        runat="server">
    <ProgressTemplate>
        Haetaan jäseniä...
        <asp:Image ID="imageLataus" runat="server" ImageUrl="~/kuvat/
            ajax-loader.gif" />
    </ProgressTemplate>
</asp:UpdateProgress>
```

Default.aspx.cs-tiedostossa luodaan sivun toiminnallisuus ja dynaamisesti luotavat sivun elementit.

`Page_Load`-metodi suoritetaan sivun latauduttua ja jokaisella takaisinkutsulla (myös Ajax). `IsPostBack` kertoo onko kyseessä takaisinkutsu, joten sen avulla voidaan jättää toimintoja suorittamatta, kun kyseessä on takaisinkutsu. Hakulomak-

keen `DropDownList`-elementin alkioita ei tarvitse luoda takaisinkutsuissa, koska sitä ei luoda dynaamisesti. `kerho`-olio luodaan ja sen tiedot haetaan tiedostosta jokaisella takaisinkutsulla, koska sitä tarvitaan niissä kaikissa:

```
protected void Page_Load(object sender, EventArgs e)
{
   kerho = new Kerho();
   kerho.lueTiedostosta("kelmit");

   ...

   // Ettei suoriteta jokaisella takaisinkutsulla.
   if (!IsPostBack)
   {
      ...

      apuJasen = new Jasen();
      asetaHakukentat();

      ...
   }
}
```

Luodaan alkiot hakulomakkeen `DropDownList`-elementille. Luodaan ne `ListItem`-olioina, joille annetaan tekstiksi kentän kysymys ja arvoksi kentän numero:

```
public void asetaHakukentat()
{
   for (int i = apuJasen.ekaKentta(); i < apuJasen.getKenttia(); i
      ++)
   {
      ListItem item = new ListItem(apuJasen.getKysymys(i), i.
         ToString());
      selectKentta.Items.Add(item);
   }
}
```

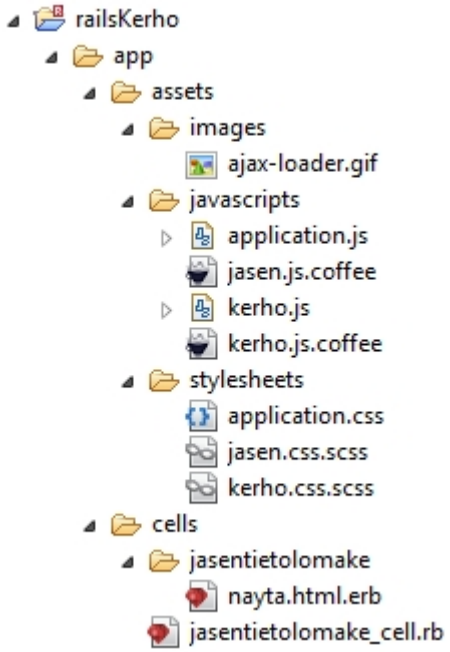
Luodaan kuuntelija hakunapin klikkauksille. Tämä metodi luodaan automaattisesti, kun tapahtumankuuntelijan määrittelee visuaalisen editorin kautta. Asetetaan hakuun viive. Asetetaan hakuteksti ja hakukenttä sessiomuuttujiin, päivitetään jäsenvalintakomponentin alkiot, näytetään animaatio päivittämällä siihen liitetty `UpdatePanel` ja lisätään historiapiste uudelleen valitun jäsenen id:lle:

```
protected void buttonHae_Click(object sender, EventArgs e)
{
   System.Threading.Thread.Sleep(3000);
   loydetyt = kerho.etsi(textHae.Text, Int32.Parse(selectKentta.
      SelectedValue));
   Session["hakuteksti"] = textHae.Text;
   Session["hakukentta"] = selectKentta.SelectedValue;
```

```
    if (loydetyt.Count > 0) asetaJasenLista(loydetyt[0].getTunnusNro
        ());
    else asetaJasenLista(-1);
    updatePanelAnimaatio.Update();
    ToolScriptManager1.AddHistoryPoint("jasen", listBoxJasenet.
        SelectedValue, "Kerho: Jäsen " + listBoxJasenet.SelectedValue
        );
}
```

9.4 Ruby on Rails -kerhosovelluksen rakenne ja esimerkki

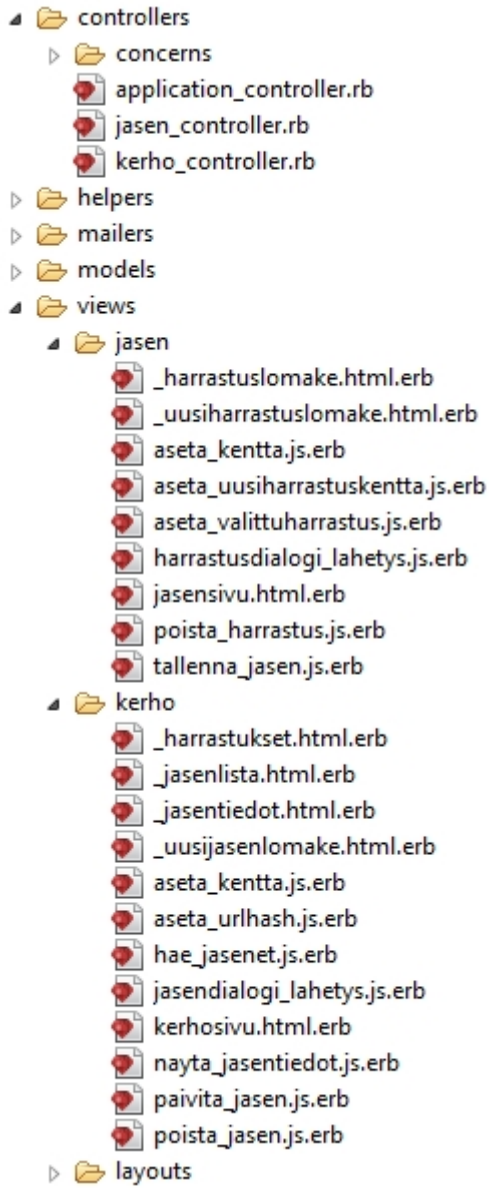
Rails-kerhosovellus koostuu kahdesta ohjaimesta (`kerho` ja `jasen`) ja niiden toiminnoista (`kerhosivu` ja `jasensivu`). Sivujen eri osat luotiin partialien avulla. Jäsentietolomake luotiin *cells*-lisäosan [32] avulla uudelleenkäytettävänä näkymänä. Kuviot 19 ja 20 esittävät sovelluksen rakenteen.



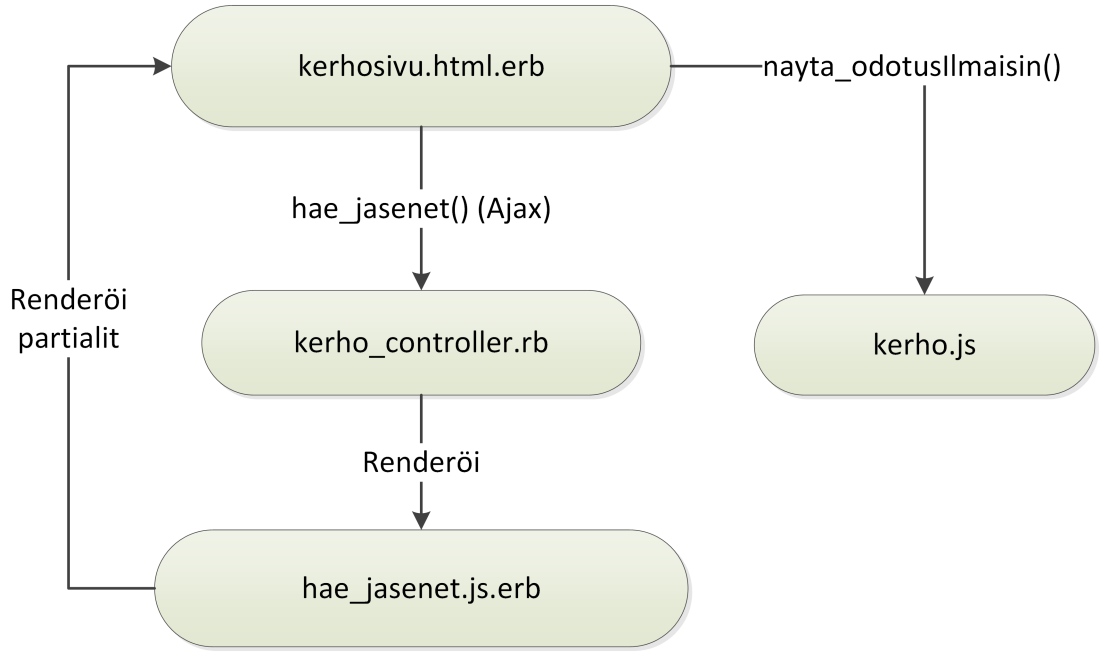
Kuvio 19: Rails-kerhoprojektin kansiorakenne (kuva1)

Kuvio 21 kuvaa hakunapin klikkauksilla suoritettavien Javascript- ja Ajax-kutsujen toiminnan.

kerhosivu.html.erb-mallineessa luodaan kerhosivu. Asetetaan sivun Javascript-tiedosto:



Kuvio 20: Rails-kerhoprojektin kansiorakenne (kuva2)



Kuvio 21: Rails-kerhosovelluksen hakunapin Javascript- ja Ajax-kutsut

```
<%= javascript_include_tag "kerho" %>
```

Luodaan jäsenten hakulomake. Lomake luodaan Railsin `form_tag`-avustajan avulla. `kerho_hae_jasenet_path`-reitti (määritelty `routes.rb`-tiedostossa) kertoo lomakkeen lähettämisessä suoritettavan toiminnon (`kerho`-ohjaimen `hae_jasenet`-toiminto). `remote: true`-parametrilla kerrotaan, että lomake lähetetään asynkronisesti. Myös lomakkeen kentät luodaan Rails-avustajien avulla. Tekstikentälle asetetaan valittu arvo `@hakuehto`-attribuutin avulla, joka asetetaan ohjaimessa. Hakukenttien valintalistalle asetetaan alkiot `options_for_select`-parametrin avulla, jolle määritellään alkiot `@kentat`-hashmapilla ja valittu alkio `@valittukentta`-attribuutilla (myös nämä asetetaan ohjaimessa). `submit_tag` -avustaja luo napin lomakkeen lähettämiselle. Määritellään napin klikkaukselle Javascript-funktio (`:onclick => "nayta_odotusIlmaisin()"`), jolla odotusilmaisin asetetaan näkyviin:

```
<%= form_tag(kerho_hae_jasenet_path, remote: true) do %>
   <%= text_field_tag(:hakuehto, @hakuehto) %>
   <%= select_tag(:hakukentta, options_for_select(@kentat,
      @valittukentta)) %>
```

```
   <%= submit_tag 'Hae', :onclick => "nayta_odotusIlmaisin()" %>
<% end %>
```

Odotusilmaisin luodaan normaalina html-elementtinä, joka asetetaan tyylin avulla piiloon. Sille asetetaan teksti ja kuva. `asset_path`-reitti hakee kuvan suoraan `assets/images`-kansiosta:

```
<p id="odotusIlmaisin" style="display:none">
   Haetaan jasenia... <img src="<%= asset_path( 'ajax-loader.gif' )
        %>" />
</p>
```

kerho.js-tiedosto sisältää kerhosivun Javascriptin. Näytetään siinä odotusilmaisin jQueryn avulla:

```
function nayta_odotusIlmaisin() {
   $('#odotusIlmaisin').show();
}
```

kerho_controller.rb-ohjaimen `kerhosivu`-toiminnossa luodaan `kerhosivu`-näkymän tarvitsemat attribuutit. Sivua avatessa siis näkymä renderöidään aina tämän toiminnon kautta. Hakulomaketta varten luodaan sen `select_tag`-avustajalle alkiot:

```
def kerhosivu

   ...

   @apujasen = Jasen.new
   @kerho = Kerho.new
   @kerho.lue_tiedostosta("kelmit")

   ...

   luo_hakulista_alkiot
end
```

Luodaan alkiot hakulomakkeen `select_tag`-avustajalle. Alkiot luodaan hashmappiin, johon laitetaan kentän kysymys ja sen numero:

```
def luo_hakulista_alkiot
   @kentat = Hash.new

   (@apujasen.eka_kentta..@apujasen.get_kenttia-1).each do |i|
      @kentat[@apujasen.get_kysymys(i)] = i
   end
```

```
end
```

Otetaan vastaan näkymältä lähetetty hakunapin painallus. Asetetaan hakuun viive. Hakuehto ja hakukenttä saadaan parametreista, joiden nimi tulee suoraan lomakkeen kenttien nimistä. Tässä pitää luoda aina myös `kerho`-olio uudelleen ja lukea sen tiedot tiedostosta, koska kaikki attribuutit nollautuvat jokaisella pyynnöllä. Päivitetään haun jälkeen jäsenvalintakomponentin alkiot ja valitaan siitä ensimmäinen jäsen. `respond_to`-lohkossa kerrotaan, että sivu päivitetään `.js.erb`-mallineella (`format.js`), eikä renderöidä koko sivua. Tässä siis Rails renderöi `.js.erb`-mallineen, jolla on sama nimi kuin tällä metodilla (`hae_jasenet.js.erb`). Tässä metodissa pitää asettaa kaikki attribuutit, joita kyseisen mallineen renderöinnissä tarvitaan:

```
def hae_jasenet
   sleep 3
   session[:toiminto] = "hae_jasenet"
   session[:hakuehto] = params[:hakuehto]
   session[:hakukentta] = params[:hakukentta]

   @kerho = Kerho.new
   @kerho.lue_tiedostosta("kelmit")
   loydetyt = @kerho.etsi(session[:hakuehto], session[:hakukentta])

   if loydetyt.count > 0
      @hashid = valitse_jasen(loydetyt[0])
   else
      @hashid = "n"
   end

   luo_jasenlista_alkiot(loydetyt)

   respond_to do |format|
      format.js
   end
end
```

hae_jasenet.js.erb-mallineessa asetetaan uusi url-hash ja renderöidään muutettavat partialit uudelleen sivulle. Animoidaan myös muuttuneet elementit sekä piilotetaan odotusilmaisin:

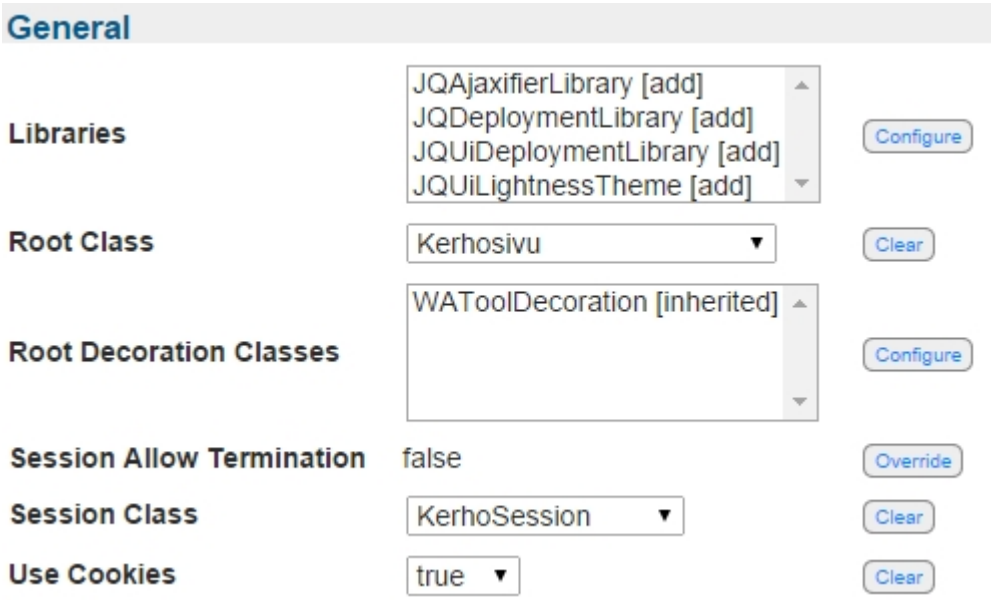
```
window.location.hash = '<%= @hashid %>';

$("#jasenValinnat").html("<%= escape_javascript( render(:partial =>
     "jasenlista")) %>");
$("#jasentiedot").html("<%= escape_javascript( render(:partial => "
    jasentiedot")) %>");
$("#harrastukset").html("<%= escape_javascript( render(:partial =>
    "harrastukset")) %>");
```

```
$("#jasenValinnat").fadeOut(500);
$("#jasenValinnat").fadeIn(500);

$("#jasentiedot").fadeOut(500);
$("#jasentiedot").fadeIn(500);

$("#harrastukset").fadeOut(500);
$("#harrastukset").fadeIn(500);

$('#odotusIlmaisin').hide();
```

9.5 Seaside-kerhosovelluksen rakenne ja esimerkki

Seaside-kerhosovellus koostuu kolmesta `WAComponent`-luokan perivästä Seaside-komponentista `Kerhosivu`, `Jasensivu` ja `JasenTietoLomake`. Lisäksi siihen kuuluu sessioluokka `KerhoSession` (perii `WASession`-luokan) ja tiedostokirjastoluokka `KerhoFileLibrary` (perii `WAFileLibrary`-luokan). Ajax-toiminnallisuudet luodaan Seasiden jQuery-integraation avulla.

Seasiden konfiguraatiosivulla on luotu *kerho*- ja *jasen*-sovellukset, joiden juurikomponenteiksi `Kerhosivu`- ja `Jasensivu`-komponentit asetettiin. Kuvio 22 esittää *kerho*-sovelluksen konfiguraatiosivulle tehdyt muutokset.



Kuvio 22: Seasiden kerho-sovelluksen konfiguraatiosivun muutokset

Luodaan `Kerhosivu`-luokka ja määritellään sille attribuutit (`instanceVariableNames`):

```
WAComponent subclass: #Kerhosivu
   instanceVariableNames: 'kerho tallennettavaJasen jasenLista
      tietoLomake'
   classVariableNames: ''
   poolDictionaries: ''
   category: 'WWWKerho'
```

Luokkapuolen metodissa (`Kerhosivu class»#canBeRoot`) kerrotaan Seasidelle, että tämä komponentti haluaa toimia omana sovelluksena [15, s. 245]:

```
canBeRoot

^true
```

Alustetaan luokka. Luodaan uusi `kerho`-olio ja luetaan sen tiedot tiedostosta:

```
initialize

super initialize.
kerho := Kerho new.
kerho lueTiedostosta: 'kelmit'.

...
```

Näytetään sivun sisältö `renderContentOn:`-metodin avulla. Hakulomake luodaan `div`-elementin sisään oman metodin kautta (`html`-piirtoalue välitetään sille parametrina). Odotusilmaisin luodaan `p`-elementin sisään ja se asetetaan tyylin avulla piilotetuksi. Odotusilmaisimen kuva on lisätty `KerhoFileLibrary`-luokkaan Seasiden konfiguraatiosivulla:

```
renderContentOn: html
| kentat apuJasen jasenId hakuehto hakukentta |

...

html div
   class: 'hakuLomake';
   with: [ self luoHakuLomake: html and: hakuehto and: hakukentta
      ].

html paragraph
   class: 'piilotettu';
   id: 'odotusilmaisin';
   with: [
      html text: 'Haetaan jäseniä...'.
      html image url: KerhoFileLibrary / #ajaxloaderGif. ].
```

```
...
```

Luodaan hakulomake. Tekstikentän arvoksi laitetaan parametrina saatu arvo ja `select`-elementin alkiot sekä hakunappi luodaan oman metodin kautta:

```
luoHakuLomake: html and: hakuehto and: hakukentta

html label with: 'Hae jäseniä: '.
html textInput
   id: #textHae;
   with: hakuehto.
html select
   id: #selectHae;
   with: [ self hakuValinnat: html and: hakukentta ].
self hakuNappi: html.
```

Luodaan alkiot hakulomakkeen `select`-elementille `option`-siveltimen avulla. Alkiolle asetetaan arvoksi kentän numero (`value: k;`) ja näkyväksi tekstiksi kentän kysymys (`with: (apuJasen getKysymys: k)`). Otetaan kentät jäseneltä ja asetetaan parametrina saatu kentän numero valituksi (`selected: true;`):

```
hakuValinnat: html and: hakukentta
| apuJasen |

apuJasen := Jasen new.

apuJasen ekaKentta to: apuJasen getKenttia - 1 do: [ :k |
   k asString = hakukentta
      ifTrue: [
         html option
            selected: true;
            value: k;
            with: (apuJasen getKysymys: k) ]
      ifFalse: [
         html option
            value: k;
            with: (apuJasen getKysymys: k) ]]
```

Luodaan hakunappi. Asetetaan sille kaksi klikkaus-tapahtumaa. Toisessa asetetaan vain odotusilmaisin näkyviin. Toisen `callback`-lohkoissa otetaan syötetty hakuehto `textInput`-elementiltä ja valittu hakukenttä `select`-elementiltä jQuery-integraation avulla (`(html jQuery id: #textHae) value`). `script`-lohkossa asetetaan haulle viive ja suoritetaan haku. Tallennetaan hakuehto ja hakukenttä myös sessioon. Lopuksi päivitetään jäsenlista ja piilotetaan odotusilmaisin (`(s jQuery id: #odotusilmaisin) hide`):

```
hakuNappi: html
| hakuEhto hakuKentta viive |

html submitButton
   onClick: (html jQuery: #odotusilmaisin) show;
   onClick: (html jQuery ajax
      callback: [ :value | hakuEhto := value ] value: (html jQuery
         id: #textHae) value;
      callback: [ :value | hakuKentta := value ] value: (html
         jQuery id: #selectHae) value;

      script: [ :s |
         viive := Delay forSeconds: 3.
         viive wait.

         self session tallenna: hakuEhto and: hakuKentta.
         jasenLista := kerho etsi: hakuEhto and: hakuKentta
            asInteger.
         jasenLista size > 0
            ifTrue: [ self paivitaJasenLista: s and: 1. ]
            ifFalse: [ self paivitaJasenLista: s and: 0. ].

         s << (s jQuery id: #odotusilmaisin) hide ]);
   with: 'Hae'.
```

9.6 Vertailu

9.6.1 Ajax-toiminnallisuus

Vaadin-kehyksellä Ajax-toiminnot tulee automaattisesti, eikä normaalien sivun päivitysten tekeminen ole edes mahdollista (paitsi siirtymällä uudelle sivulle). Komponenteille loi vain haluamansa tapahtuman samaan tyyliin kuin esimerkiksi Javan *Swing*-kirjastossa, ja se suoritettiin automaattisesti Ajax-kutsuna. Muokatut elementit päivitettiin myös sivulle automaattisesti kutsun suoritettua. Tekstikentät ja valintalistat sai lähettämään tapahtuman palvelimelle heti tapahtuman synnyttyä `setImmediate = "true"` -attribuutilla. Oletuksena niiden tapahtumat välitetään kun seuraava asynkroninen kutsu tulee (esimerkiksi kun nappia on klikattu).

ASP.NETissä Ajax-toiminnallisuudet luotiin luvussa 5.4 kuvatulla ASP.NET-AJAX-lisäosan palvelinpuolen kehyksellä `ScriptManager-` ja `UpdatePanel`-kontrollien avulla. Tämä lisäosa oli siis ASP.NET 4.5 -versiossa valmiiksi asennettuna. Dialogia ja animaatioita varten piti asentaa erikseen *Ajax Control Toolkit*-lisäosa. Sovelluslogiikkakoodiin (`.aspx.cs`-tiedosto) ei tarvinnut tehdä Ajaxia varten mitään muutoksia.

Ainoastaan tekstikentät piti asettaa `UpdatePanel`-kontrollien *Triggereiksi* koodin puolella, kun kentät luotiin dynaamisesti. `ScriptManager`- ja `UpdatePanel`-kontrollit sai luotua myös Visual Studion ulkoasueditorilla eli sitäkään koodia ei tarvinnut itse kirjoittaa. Napit sai helposti muokattua lähettämään Ajax-kutsun laittamalla ne `UpdatePanel`-kontrollin `ContentTemplate`-elementin sisälle, jolloin ne eivät aiheuta koko sivun päivitystä. Tietysti jos nappi on määritelty jonkun muun elementin *triggeriksi*, niin silloin se aiheuttaa Ajax-kutsun automaattisesti. `UpdatePanel`-kontrollin sai myös päivitettyä ohjelmallisesti koodin puolelta, mikä oli tarpeen esimerkiksi animaatioiden kanssa.

Seasidessa Ajax-toiminnallisuudet luotiin luvussa 7.4 kuvatun Seasiden jQuery-integraation avulla. jQuery-integraatio otettiin käyttöön lisäämällä tarvittavat kirjastot sovelluksen konfiguraatiosivulla (ks. kuvio 22), jotka olivat siellä valmiiksi asennettuina (Seaside 3.1 -versiossa). jQuery-toimintoja voitiin siis käyttää Smalltalk-koodin kautta, eikä varsinaista jQuery-koodia tarvinnut kirjoittaa (eikä itse jQueryn osaamistakaan välttämättä vaadittu). Esimerkkejä löytyi sen verran, että itse jQuery-dokumentaatiota ei hirveästi tarvinnut käyttää, ainoastaan animaatioiden kohdalla piti sieltä katsoa. Aluksi hieman pohdittavaa aiheutti, kun oli monta tapaa tehdä Ajax-kutsu jQueryn avulla: `html jQuery ajax` ja `(html jQuery id: 'id') load`. Mutta lopulta tuota `load`-tapaa ei tainnut jäädä enää mihinkään kohtaan, koska sitä käyttämällä sai päivitettyä sivulle vain yhden elementin (sen mikä siinä ladataan). Nämä Ajax-kutsut välittivät käytettäväksi uuden renderöijän, jonka kautta muutokset päivitettiin jQueryn avulla, eli vanhaa `html`-piirtoaluetta ei tässä enää käytetty (tämän käyttö siis kiellettiin luvussa 7.2). Tosin myös `html`-piirtoaluetta käyttämällä tuo sovellus toimi ihan oikein, koska aluksi päivitin muutokset (vahingossa) sillä.

Ruby on Railsissa käytettiin luvussa 6.4 kuvattua Railsin huomaamatonta Javascript tukea ja jQuery-kirjastoa. jQuery-kirjasto oli valmiina käytettäväksi (Rails 4.1), mutta jQuery UI piti lisätä *gem*-paketilla (esimerkiksi lisäämällä *gem*-määritys `Gemfile`-tiedostoon ja suorittamalla `bundle install`). Oikeastaan tuota jQuery UI -kirjastoa en tainnut kuitenkaan edes tarvita, kun dialoginkin tein lopulta erillisellä lisäosalla.

Ajax-toiminnallisuuden luomiseksi tarvittiin aika paljon eri osia (`.js.erb`-tiedosto, muutokset näkymään ja ohjaimeen ja lisäksi `.js-tiedosto`) ja piti myös kirjoittaa oikeaa jQuery-koodia. `.js.erb`-mallineisiin joutui myös kirjoittamaan Ruby-koodia Javascript-koodin sekaan, mikä teki koodista aika sekavaa. Nappien painallukset sai välitettyä ohjaimelle Ajax-kutsuna pelkän `remote: true` -attribuutin avulla. Kyseisen parametrin sai lisättyä myös lomake-avustajalle, jolloin kaikki sen napit luovat Ajax-kutsun. Muut tapahtumat (kuten valintalistan arvon muutos) piti välittää ohjaimelle Javascript-tiedoston kautta Ajax-kutsuna (esimerkiksi jQueryn avulla). Ajax-kutsuissa piti aina myös muistaa luoda reitti `routes.rb`-tiedostoon. Railsissa Ajax-toimintojen luominen vaati kaikkein eniten työtä, ja myös virheiden mahdollisuus kasvoi, kun piti käyttää niin montaa eri tiedostoa. Lisäksi Ajax-kutsussa pitää olla `.js.erb`-malline Ajax-kutsun vastaanottavalle metodille, vaikka siinä ei mitään tekisikään. Lisäksi täysin samanlainen (ja samanniminen) `.js.erb`-malline piti luoda sekä `jasen`- että `kerho`-näkymiin.

Railsissa lisäksi Javascript-tiedostoa (`kerho.js`) varten piti lisätä `config/initializers/assets.rb`-tiedostoon rivi:

```
Rails.application.config.assets.precompile += \%w( kerho.js )
```

Ja `html.erb`-sivulle piti lisätä:

```
<\%= javascript_include_tag "kerho" \%>
```

Railsissa *Partialien* (ks. luku 6.3.3 ja osittainen malline) käyttö osoittautui käyttöliittymän päivittämisessä varsin tehokkaaksi: Loi vaan `partial`-tiedoston, johon loi HTML-koodin, joka tiettyyn paikkaan sivulle tulee. Sitten `.js.erb`-tiedostossa vaan kertoi (jQueryn avulla), että tietyn *partialin* haluaa päivitettävän tietyn elementin sisään.

ASP.NETissä ongelmia aiheutti se, että `Page_Init`- ja `Page_Load`-metodit suoritettiin jokaisella takaisinkutsulla. Siis aina ensin suoritettiin kyseiset metodit ja vasta niiden jälkeen itse kontrollin aiheuttama tapahtuma. Tämän pystyi kuitenkin kiertämään `if (!IsPostBack)` -lohkon avulla. `IsPostBack` siis kertoo onko

kyseessä takaisinkutsu. Kuitenkin kaikki dynaamisesti luodut sivun osat piti luoda uudelleen myös takaisinkutsussa tai muuten ne hävisivät, joten myös niiden tarvitsemat oliot piti luoda aina uudelleen.

ASP.NETissä ja Railsissa oli sama ongelma, että kaikki attribuutit nollautuvat jokaisella takaisinkutsulla. Eli tarvittavat oliot (esimerkiksi kerho) pitää luoda aina uudelleen, kun niitä tarvitaan. ASP.NETissä nämä pystyi luomaan esimerkiksi `Page_Load`-metodissa, jolloin ne luotiin automaattisesti jokaisella takaisinkutsulla. Tällöin ne kuitenkin luodaan myös turhaan, vaikka niitä ei tarvittaisi. Kannattikin sijoittaa `if (!IsPostBack)` -lohkoon kaikki oliot, joita ei tarvita luoda aina uudelleen, ja luoda ne sitten aina myös niissä metodeissa, joissa niitä käytetään. Railsissa luotiin myös kaikki tarvittavat oliot uudelleen kaikissa toiminnoissa, missä niitä tarvittiin (tai sitten otettiin ne sessiomuuttujista).

ASP.NETissä ja Railsissa jäsenen tiedot asetettiin ja tarkistettiin lomakkeen kentistä uudelleen ennen tallentamista, jolloin muokattua jäsentä ja virheiden määrää ei tarvinnut säilyttää sessiomuuttujissa. ASP.NETissä tekstikenttien arvot saatiin hakemalla elementti sen id-arvon perusteella ja kysymällä arvo siltä. Railsissa lomake sijoitettiin `form_tag`-avustajan sisään, jolloin tekstikenttien arvot välitettiin `submit`-painikkeella ohjaimelle parametreina (joiden nimet olivat tekstikenttien nimet). Molemmissa kehyksissä piti tallentaa muutetut harrastukset sessioon ja tallennusvaiheessa ensin poistaa kerhosta vanhat jäsenen harrastukset ja sitten lisätä sessiosta uudet harrastukset. Ilman sessioiden käyttöä olisi pitänyt tallentaa harrastukset kerhoon aina heti uuden harrastuksen lisäyksessä ja harrastuksen poistossa, jo ennen tallenna-napin painamista. Tämä olioiden tallentaminen sessioon oli helpompi tehdä ASP.NETissä kuin Railsissa. Railsissa joutui muuttamaan ne sessiota varten YAML (*YAML Ain't Markup Language*) [65] -muotoon (`to_yaml`-metodilla) ja haettaessa sessiosta lataamaan ne siitä (`YAML.load()`-metodilla).

9.6.2 Dialogit

Vaadin-kehyksellä modaalin dialogin luominen onnistuu ali-ikkunoiden avulla luomalla uuden ikkunan ja lisäämällä se UI-luokkaan. Dialogin ulkoasu tulee sovelluksen teeman mukaan, eli sen muokkaaminen onnistuu vaihtamalla tai muokkaamalla teemaa. Ulkoasun muuttaminen onnistui myös vaihtamalla dialogin tyylinimeksi joku teeman vaihtoehtoisista dialogityyleistä (kuten `Reindeer.WINDOW_LIGHT`, missä `Reindeer` on teeman nimi). Dialogi sisältää automaattisesti oikeassa yläkulmassa napit dialogin sulkemiselle ja dialogin suurentamiseksi koko selaimen ikkunan kokoiseksi. Dialogille pitää luoda oma ulkoasukomponentti (layout), johon dialogin sisältö luodaan. Sisällön voi myös luoda visuaalisella editorilla, jos dialogin luo erillisenä komponenttina. Tässä kuitenkin dialogin sisältö piti luoda melkein kokonaan dynaamisesti niin luonti koodin kautta tuntui yksinkertaisemmalta. Näin pystyi myös käsittelemään tapahtumat suoraan sivun koodissa, eikä tietoja tarvinnut välittää dialogin ja sivun välillä.

ASP.NETissä dialogi luotiin *Ajax Control Toolkit*-lisäosan dialogilla. Dialogin sisällön ulkoasu piti luoda kokonaan itse ja myös taustasivun tyyli pitää määritellä. Myöskään sulkemisruksia ei tule automaattisesti dialogin yläkulmaan. Tällä siis dialogin ulkoasun luominen vaatii vähän enemmän työtä, mutta persoonallisen näköisen dialogin luominen on helpompaa. Dialogin kanssa ongelmia aiheutti se, että senkin sisältö pitää luoda uudelleen jokaisella takaisinkutsulla. Sainkin sen lopulta toimimaan, kun loin dialogin sisällön omalla komponentilla, ja tuon komponentin sisältö luotiin aina automaattisesti `Page_Load`-metodissa. Jäsensivulla piti sitten myös lisätä dialogin sisällön luonti kyseiseen metodiin. Aiemmin loin sisällön vain kun dialogi avataan, jolloin takaisinkutsuissa dialogin sisältö hävisi.

Railsissa en saanut dialogia toimimaan normaalilla jQuery UI:n dialogilla (jostain syystä dialogi ei avautunut). Se kyllä toimi muutaman kerran, mutta sitten siihen (tai sovellukseen) tehtiin jotain muutoksia ja sen jälkeen sitä ei saatu enää millään toimimaan. Dialogi luotiin sitten *jquery-modal-rails*-kirjaston [33] avulla. Tämäkin valitti, ettei löydä kuvaa *"close"*, joten piti poistaa viittaus tuohon kuvaan tuon kirjaston tyylitiedostosta. Nyt ei sitten dialogissa näkynyt dialogin sulkemisruksia yläkulmas-

sa, vaikka sieltä painamalla dialogi kyllä sulkeutui. Dialogin sulkeminen napilla piti toteuttaa `.js.erb`-tiedostossa jQueryn avulla (`$.modal.close();`). Dialogin avaaminen onnistui avustajalla *link_to_modal*, jolle sai annettua sen elementin id:n, jonka se näyttää dialogin sisältönä. Sitten loi `div`-elementin dialogin sisällöksi, jonka asetti tyylillä piilotetuksi (`style="display:none"`). Dialogin avaamista napilla ei tässä ilmeisesti voi tehdä, *link_to_modal* on siis linkki. Normaalia jQuery UI:n dialogia käytettäessä piti jQuery UI:n teema lisätä sovellukseen itse, mutta *jquery-modal-rails*-kirjastossa teema tuli kirjaston mukana. Tämän kirjaston teema on varsin pelkistetty, mikä helpottaa omanlaisen dialogin luomista. Itse asiassa sen sisällön ulkoasu pitää kokonaan luoda itse, kuten ASP.NET-kehyksenkin dialogissa. jQuery UI:n normaalilla dialogilla saisi helposti näyttävämmän ulkoasun valmiiden teemojen avulla sekä muutettua dialogin ulkoasun vaihtamalla teemaa. Sille siis saisi lisättyä minkä tahansa teeman jQuery UI:n valikoimasta.

Seasidessa oman dialogin sai luotua jQuery-integraation avulla: Dialogi luotiin vastaavasti kuten normaali jQuery-dialogi, paitsi Smalltalk-koodilla. Dialogin toimimiseksi piti lisätä *JQUiLightnessTheme* (tai *JQUiDarknessTheme*) sovelluksen kirjastoihin. Nämä kaksi teemaa olivat valmiina konfiguraatiosivun kirjastoissa. En kokeillut lisätä kirjastoihin täysin uutta jQuery UI:n teemaa (en helposti keksinyt kuinka se olisi tehty, tai onko se edes mahdollista), joten dialogin ulkoasun muunneltavuuteen en voi ottaa kantaa. Jostain syystä dialogi ei toiminut kunnolla kun käytössä oli Pharo 2.0 ja Seaside 3.1: Kun sivulle tuli linkin kautta (kerhosivulta jäsensivulle tai jäsensivulta kerhosivulle), näkyi dialogin sisältö sivulla ilman dialogin avausnappia (sivun päivittäminen uudelleen korjasi asian). Kun lopulta kokeilin Pharo 1.3 ja Seaside 3.0 yhdistelmällä niin tuo toimikin yllättäen ihan oikein.

Vaadin-kehyksessä varmistusdialogi luotiin *ConfirmDialog*-lisäosan [68] avulla. Tälle määriteltiin otsikko, varmistusteksti ja nappien tekstit. Dialogin sulkemistapahtumassa tarkistettiin onko painettu "Kyllä"-nappia. Varmistustekstissä sai helposti näytettyä myös valitun jäsenen nimen tai harrastuksen alan, sillä dialogi luodaan dynaamisesti aina kun nappia painetaan.

ASP.NET-kehyksessä varmistusdialogi luotiin *Ajax Control Toolkit*-lisäosan

`ConfirmButtonExtender`-kontrollilla. Tämä luotiin `.aspx`-ulkoasutiedostossa ja sille tarvitsi määrittää vain kontrolli, johon varmistus liitetään ja varmistusteksti. Jäsenen nimeä en saanut tähän näkyviin: Vaikka varmistustekstin muutti koodin puolella, ei se silti päivittynyt kontrolliin, eikä tuota kontrollia saanut lisättyä `UpdatePanel`-kontrollinkaan sisälle.

Railsissa varmistusdialogin sai luotua lisäämällä sen luova attribuutti linkkikomponenttiin ja antamalla sille varmistusteksti. Tässä saisi valitun jäsenen nimen näkyviin, kun kyseisen linkin renderöisi uudelleen aina kun valitaan uusi jäsen (koska varmistusteksti luodaan kun linkki luodaan). Seasidessa varmistus saatiin luotua lisäämällä napin Ajax-kutsuun `confirm:` -attribuutti, jolle annettiin varmistusteksti. Tässäkin saisi valitun jäsenen nimen näkyviin, kun nappi luotaisiin uudelleen aina uuden jäsenen valinnassa. Molemmissa kehyksissä kaikki hoitui automaattisesti näiden attribuuttien avulla: Ne avaa dialogin *"kyllä"*- ja *"ei"*-napeilla ja *"kyllä"*-nappia painettaessa Ajax-kutsu suoritetaan ja *"ei"*-napilla ei. Kyseinen dialogi on selaimen oma dialogi ja sen ulkoasu siis riippuu selaimesta.

Vain Vaadin-kehyksen varmistusdialogissa sai erikseen määriteltyä myös dialogin nappien tekstit. Muissa varmistusdialogeissa ne tuli automaattisesti, enkä tiedä saako niitä edes muutettua.

9.6.3 Komponenttien käyttäminen

Vaadin- ja ASP.NET-kehyksissä komponenttien käyttäminen oli varsin helppoa. Komponenttien arvojen kysyminen ja muuttaminen sekä muut muutokset komponentteihin onnistui Vaadin-kehyksessä suoraan Java-koodilla ja ASP.NETissä C#-koodilla, ja myös Eclipsen sekä Visual Studion automaattinen täydennys tarjosi apuja. Vaadin- ja ASP.NET-kehyksissä komponentin sai haettua koodin puolella sen id-arvon perusteella. Esimerkiksi komponenttien tyylinimen sai haettua (ja myös muutettua) missä tahansa kohtaa koodia ja tekstikentän pituuden määriteltyä dynaamisesti suoraan koodin puolelta.

Railsissa komponenttien arvot (siis vain arvot) sai välitettyä ohjaimelle vain kun

ne on lomake-avustajan sisällä (ja kun lomake lähetetään `submit`-napilla). Muuten komponentteihin pääsi käsiksi vain Javascriptin avulla joko Javascript- tai `.js.erb`-tiedostossa. Eli esimerkiksi komponentin tyylinimi piti välittää ohjaimelle Javascript-tiedoston kautta ja uusi tyylinimi asettaa `.js.erb`-tiedostossa. Tekstikentän pituus piti muuttaa tyylien avulla. Railsin komponentit eivät nappia lukuun ottamatta myöskään sisällä mitään tapahtumia, eli kaikki muut komponenttien tapahtumat pitää tehdä Javascriptin kautta.

Seasidessa komponenttien arvot sai kysyttyä `callback`-lohkossa, jossa komponentti voitiin hakea sen id:n perusteella jQuery-integraation avulla. Myös komponenttien tyylinimen sai haettua callback-lohkossa jQuery-haulla ja uuden tyylinimen asetettua Ajaxin `script`-lohkossa myös jQueryn avulla. Tekstikentän pituus piti muuttaa tyylien kautta kuten Railsissa.

ASP.NET-tekstikenttä sisältää tapahtuman vain tekstinvaihtumiselle. Eli harrastusten poistossa pitää tekstikenttään kirjoittaa jotain, että harrastus tulee valituksi. Seaside ja Ruby on Rails -tekstikentille sai klikkaustapahtuman Javascriptin avulla. Vaadin-kehyksessä sai asetettua klikkaustapahtuman tekstikentälle ja myös tekstinvaihtumistapahtumalle sai määriteltyä kuinka usein tapahtuma synnytetään: Heti kirjaimen syöttämisen jälkeen, sitten kun tekstin syötössä on tauko tai kentästä poistuttaessa. Seasiden ja Ruby on Railsin -teksitentille sai tekstinvaihtumistapahtuman (`onChange`) lisäksi myös napin ylösnousemistapahtuman (`onKeyUp`), jossa tapahtuma välitetään jokaisen syötetyn arvon jälkeen (nämä tapahtumat tehtiin myös Javascriptin avulla). ASP.NET-tekstikentän tekstinvaihtumistapahtuma välitetään vasta kun kentästä poistutaan.

Seasiden valintalistakomponentin alkiot piti luoda `option`-siveltimellä, jos halusi alkiolle eri arvon ja näkyvän tekstin. Tällöin piti asettaa suoraan joku `option`-elementeistä valituksi koodin puolelta asettamalla sille attribuutti `selected="true"`, eikä valittua arvoa saanut asetettua itse valintalistakomponentista. Valintalistan alkiot saa myös luotua suoraan listasta, mutta tällöin alkion arvo ja näkyvä teksti ovat samat (valitun alkion saa tällöin asetettua suoraan valintalistakomponentin attribuutilla). Railsissa sai valintalistan alkiot luotua hashmapista, jossa oli alkioiden

arvot ja näkyvä teksti. Valitun alkion sai määriteltyä näkymässä attribuutilla, jossa oli alkion arvo. Attribuutti siis asetettiin aina ohjaimessa, kun valintalistakomponentti renderöitiin. Vaadin-kehyksessä alkiot luotiin valintalistakomponentille sen metodeilla: `addItem()` lisäsi listaan uuden alkion (arvo annettiin parametrina), ja `setItemCaption()` loi alkiolle näkyvän tekstin (parametreina alkion arvo ja näkyvä teksti). Valitun arvon sai asetettua metodilla `setValue()` antamalla sille parametriksi valittavan alkion arvo. ASP.NET-kehyksessä alkiot luotiin `ListItem`-olioina, joille annettiin parametreiksi alkion teksti ja arvo. Valittu alkio saatiin asetettua valintakomponentin metodilla `SelectedValue()`, jolle annettiin parametriksi valittava arvo.

Vaadin sisälsi erillisen taulukko-komponentin, joka loi automaattisesti sarakkeiden lajittelun otsikkoa klikkaamalla. Sarakkeelle sai määriteltyä tyypin sen solujen arvoille ja lajittelu toimi sen mukaisesti. Sarakkeiden solujen arvojen piti myös olla oikeaa tyyppiä, eli `Integer`-arvot piti lisätä `Integer`-olioina. Muissa kehyksissä taulukko oli normaali HTML-taulukko, vaikkakin ASP.NETissä se luotiin WWW-palvelinkontrollitaulukolla.

ASP.NET ja Vaadin sisältävät visuaalisen editorin, joilla komponentit voi luoda ja lisätä sivulle. Vaadin-kehyksen editorilla komponenttien sijoittelu oli helpompaa, koska sillä sai lisättyä myös ulkoasukomponentit. Vaadin-kehyksen editorissa komponentit sai lisättyä myös varsinaisen ikkunan vieressä olevaan puurakenteeseen sivun komponenteista, jossa niiden asettelu oikeaan paikkaan olikin helpompaa. ASP.NETissä komponenttien asettelu editorilla oli hankalampaa, joten yleensä komponentti luotiinkin ensin editorilla ja vaihdettiin sitten oikeaan paikkaan koodin puolella. ASP.NETin editorilla sai komponenteille luotua myös tapahtumienkäsittelijät. Tämä loi tarvittavan koodin sekä ulkoasusivuun että koodin puolelle. Tämä onkin varsin hyvä ominaisuus, sillä sieltä myös helposti näkee, mitä tapahtumia komponenteille on olemassa, eikä niiden luomisen syntaksista tarvitse huolehtia itse. Vaadin-kehyksen editorilla tapahtumankäsittelijöiden luominen ei onnistu, ja aina sitä oikeaa tapahtumanluomiskoodia (ja tapahtuman oikeaa nimeä) joutui jostain hakemaan.

9.6.4 Uudelleenkäytettävän komponentin luominen

Vaadin-kehyksessä oman jäsentietolomake-komponentin sai luotua perimällä `CustomComponent`-luokan, ja luomalla lomakkeen siihen. Tämä on siis täysin sama komponentti, jolla varsinaiset sivut luotiin UI-luokkien sisälle. Olisi voinut myös suoraan periä esimerkiksi `FormLayout`-luokan, mutta Vaadin-kirjassa suositeltiin tuota toista tapaa. `CustomComponent`-luokan perimällä saa myös luotua komponentin ulkoasun Vaadin-editorilla.

ASP.NETissä sai luotua *Web Custom Component*-komponentin, jonka sisällön luominen onnistui kuten normaalin ASP.NET-sivun luominen.

Seasidessa oman komponentin luominen onnistui kuten normaalin Seaside-sivun luominen. Loi vaan `RenderContentOn: html` -metodiin sen jäsenen lomakkeen. Sivun `children`-metodissa pitää kertoa kaikki omat komponentit, jotka meinaa sivulla näyttää, vaikka tuo kyllä toimi ilmankin [25, luku 12.1]. Komponentin sisällön päivittäminen ei jostain syystä onnistunut, kun se oli jo kerran renderöity sivulle. Tästä syystä dialogin lomake piti päivittää Ajax-kutsussa käyttämällä dialogin elementin id:tä.

Railsissa omaa komponenttia ei voinut suoraan luoda. *Cells*-lisäosan avulla sai luotua uudelleenkäytettävän näkymäkomponentin, jolle sai välitettyä jäsenen, jota lomakkeen luomisessa käytetään. Tällä ei kuitenkaan pystynyt tarkistamaan lomakkeen virheitä eikä palauttamaan jäsentä tai tietoa virheistä. Kaikki tämän näkymäkomponentin logiikka siis suoritettiin erikseen varsinaisen sivun ohjaimessa. Täysin oman komponentin olisi ilmeisesti voinut luoda *Apotomo*-lisäosalla [2], mutta sen käyttäminen erosi normaalista Railsin käytöstä jonkin verran, eikä sitä nyt tähän ehditty opetella.

9.6.5 Ulkoasun luominen

Vaadin-kehyksellä ulkoasun luominen onnistui suureksi osaksi kehyksen visuaalisella editorilla. Dynaamisesti luotavat osat piti tietysti luoda koodin puolella. Editori loi siis koodin, jonka kautta käyttöliittymä luotiin, kuten luvussa 4.5 mainittiin. Tämä

koodi on varsin hyödyllinen varsinkin Vaadin-kehyksen käyttöä opetellessa, koska siitä voi katsoa mallia ulkoasun ja komponenttien luomiseen. Ulkoasukomponenteista (taulukko 2) eniten käytettiin `VerticalLayout`- ja `HorizontalLayout`-komponentteja. `GridLayout`-komponenttia käytettiin ainoastaan kerhosivulla jäsenen tietojen näyttämiseen, jolloin kysymykset ja tiedot sai nätisti sarakkeisiin. Lomakkeet sai luotua `FormLayout`-komponentin avulla. Tyylien lisääminen onnistui varsin helposti: Vaadin loi teeman (ks. luku 4.5) ja sinne tyylitiedoston automaattisesti, eli tarvitsi vain lisätä omat tyylit tyylitiedostoon. Komponenteille sitten annettiin tyylinimet joko koodin puolella tai visuaalisessa editorissa.

ASP.NET- ja Ruby on Rails- kehyksillä ulkoasu luotiin normaalilla HTML-koodilla, johon kehysten omat komponentit sijoitettiin. ASP.NET sisälsi kuitenkin myös visuaalisen editorin, mutta tätä käytettiin ainoastaan ASP.NET-komponenttien koodin kopioimiseen. Komponenttien sijoittaminen editorilla oli hieman hankalaa, joten ne laitettiin aina koodin puolella oikeaan paikkaan. Editorilla ei myöskään saanut kaikkia HTML-elementtejä lisättyä. Railsin ulkoasukoodista tuli aika sekavaa, kun sinne piti sekaan sijoittaa Ruby-koodia (varsinkin joidenkin partialien kohdalla). Sivun eri osat luotiin siis partialien avulla (luku 6.3.3), mikä selvensi ulkoasukoodia jonkin verran. ASP.NETissä taas ulkoasukoodia sekoitti aika paljon `UpdatePanel`- ja muut Ajax-kontrollit. Rails loi tyylitiedostot automaattisesti `stylesheets`-kansioon jokaiselle ohjaimelle. Eli ohjaimeen liittyvät tyylit sijoitetaan aina sen omaan tyylitiedostoon. ASP.NETissä tyylitiedoston sai itse luotua haluamaansa paikkaan, mutta tyyleille oli myös varattu oma `Styles`-kansio. `.aspx`-tiedostossa sitten kerrottiin sivun tyylitiedoston osoite. Luvussa 5.2 mainittua *App_Themes*-kansiota ei kerhosovelluksessa käytetty, eikä vielä opeteltu ollenkaan noiden teemojen käyttöä.

Seasidessa ulkoasu luotiin HTML-elementeillä Seasiden piirtoalueiden ja siveltimien (ks. luku 7.1) avulla. Varsinaista HTML-koodia ei siis tarvinnut kirjoittaa, mutta HTML-elementtien käyttäminen piti kuitenkin olla hallussa. Sivun eri osat saatiin luotua omissa metodeissaan välittämällä niille parametrina `renderContentOn:`-metodin vastaanottama `html`-piirtoalue, jolloin koodia saatiin selkeämmäksi. Seasidessa tyylien käyttöön ottaminen oli vähän hankalampaa: Piti luoda erillinen

`WAFileLibrary`-luokan perivä luokka, jonne tyylitiedostot sijoitetaan (tänne sijoitetaan myös esimerkiksi kuvat). Sitten sivun `updateRoot`:-metodissa piti kertoa se tyylitiedosto, jota sivulla käytetään.

ASP.NETissä ja Seasidessa piti tyylitiedosto päivittää itse selaimessa, aina kun siihen oli tehnyt muutoksia. Seasidessa tämän tyylitiedoston löysi konfiguraatiosivun kautta, kun sen osoitetta ei muuten keksitty:

```
http://localhost:8080/files/KerhoFileLibrary/kerho.css
```

ASP.NETissä tyylitiedoston löysi suoraan kansiorakenteen avulla. Vaadin-kehyksessä ja Railsissa tyylitiedoston muutokset näkyivät automaattisesti, Railsissa myös ilman palvelimen uudelleenkäynnistystä.

Vaadin-kehyksellä lomakkeen luominen oli kaikkein yksinkertaisinta. Sen `FormLayout`-ulkoasukomponentti asetti tekstikentät nätisti riviin ja näytti otsikon tekstikentän edessä. Lisäksi `FormLayout` näytti virheilmaisimen tekstikentän edessä, jos tekstikentälle oli asetettu virhe. Muilla kehyksillä lomakkeiden ulkoasu piti luoda taulukoilla, jos ne halusi nätisti riviin. Ja virheille piti luoda oma sarake taulukkoon.

9.6.6 Odotusilmaisimen näyttäminen

Pitkään kestävissä Ajax-toiminnoissa kannattaa näyttää käyttäjälle ilmaisin, että toimintoa suoritetaan (luku 2.3). Ajax-toiminnot päivittävät sivun kuitenkin vasta kutsun suorituksen jälkeen. Siksi odotusilmaisimen päivittämisessä sivulle joutui käyttämään erilaisia keinoja.

Vaadin-kehyksessä odotusilmaisimen sai päivitettyä sivulle *Vaadin Push* -kirjaston avulla. Tämän sai *web.xml*-tiedostossa määrittelemällä toimimaan niin, että sille voidaan kertoa koska sivun muutokset päivitetään. Ilmaisimen poistamista näkyvistä ei tarvinnut erikseen päivittää lisäosalla, koska se päivittyi automaattisesti, kun Ajax-kutsu oli valmis. Pyörivän ilmaisimen näyttäminen onnistui `progressbar`-komponentin avulla eli sitä kuvaa ei tarvinnut itse etsiä. Vaadin-kehyksessä en

onnistunut luomaan samalle napille kahta `onClick`-tapahtumankäsittelijää, jolloin toisessa olisi näytetty ilmaisin ja toisessa tehty haku ja poistettu ilmaisin, kuten Seasidessa tehtiin. Siksi piti päivittää ilmaisin sivulle kutsun suorituksen aikana tuon kirjaston avulla.

ASP.NETissä odotusilmaisimen näyttäminen onnistui `UpdateProgress`-kontrollin avulla. Sen `<ProgressTemplate>`-elementtiin sijoitettiin näytettävä teksti ja kuva. `UpdateProgress`-kontrolli liitettiin sitten siihen `UpdatePanel`-kontrolliin, jonka päivityksessä odotusilmaisin haluttiin näyttää, ja ASP.NET AJAX hoiti ilmaisimen näyttämisen ja poistamisen automaattisesti.

Railsissa odotusilmaisimen näyttämistä varten piti luoda painikkeelle Javascript-funktio, jossa ilmaisin asetetaan näkyviin Javascriptin avulla. Ilmaisin poistettiin näkyvistä samalla kun muut käyttöliittymän päivitykset näytettiin `.js.erb`-tiedossa.

Seasidessa odotusilmaisin näytettiin luomalla hakunapille toinen `onClick:`-metodi, jossa ilmaisin näytettiin jQuery-integraation avulla. Ilmaisin piilotettiin näkyvistä Ajax-kutsun luovassa `onClick:`-metodissa, kun haku on suoritettu ja tulokset näytetty.

Vaadin oli siis ainoa kehys, jossa varsinaista pyörivää odotusilmaisimen kuvaa ei tarvinnut erikseen etsiä ja lisätä projektiin. Tietysti jos haluaa erilaisen ilmaisimen kuin Vaadin-kehyksen ilmaisin niin pitää käyttää erillistä kuvaa. Odotusilmaisimen näyttäminen oli selkeästi helpoin luoda ASP.NET-kehyksessä. Siinä ilmaisimen näyttäminen onnistui helposti ASP.NET AJAXin omalla kontrollilla eikä itse tarvinnut huolehtia ilmaisimen näyttämisestä tai poistamisesta. Vaadin-kehyksessä joutui lisäämään tätä varten tuon uuden lisäosan, ja sitä joutui vielä konfiguroimaan `web.xml`-tiedostossa. Rails oli ainoa kehys, jossa ilmaisimen näyttäminen ja poistaminen piti tehdä aivan eri paikoissa.

9.6.7 Kirjanmerkkien ja takaisin-napin toiminta

Ajax-päivitykset eivät normaalisti muuta sivun osoitetta, jolloin selaimen kirjanmerkit ja takaisin-nappi eivät toimi oikein (luku 2.3). Tämä ongelma ratkaistiin kaikilla kehyksillä (paitsi Seasidella, jossa sitä ei saatu toimimaan) asettamalla Ajax-päivityksessä sivun osoitteeseen `#ankkuri`, josta sovelluksen tila saadaan selville. Seuraavissa kappaleissa url-fragmentilla ja url-hashilla tarkoitetaan tuota osoitteen `#ankkuria`.

Vaadin-kehyksessä *url-fragmenttiin* lisättiin jäsenen id kun jäsenvalinta muuttui. Sivun latautuessa tarkistetaan *url-fragmentti* ja valitaan jäsen sen mukaan. Takaisin-napin sai toimimaan *fragmetinvaihtumistapahtuman* avulla: Kun arvo muuttuu, päivitetään valintalista ja jäsentiedot fragmentin arvon mukaan. *url-fragmentti* lisätään osoitteeseen myös sivun ensimmäisellä latauksella (tällöinkin listan valinta muuttuu kun asetetaan ensimmäinen valinta), eli takaisin-nappi toimii myös sivun latauduttua valittuun jäseneen. Tämän toiminnallisuuden voisi tehdä myös Vaadin-kehyksen *Navigator*-komponentin avulla, mutta tämä ratkaisu oli tähän kohtaan toimivampi, varsinkin kun sivu oli tehty ensin ilman navigointia.

ASP.NETissä toiminnallisuus luotiin *ScriptManager*-kontrollin avulla asettamalla sille attribuutti `EnableHistory="true"` ja luomalla historiatietojen muuttuessa suoritettava metodi `OnNavigate="ToolScriptManager1_Navigate"`. Historiatiedot sai näytettyä selkeäkielisenä `EnableSecureHistoryState="false"` attribuutilla, oletuksena ne näytetään salattuna. Jäsenlistan valinnan muuttuessa asetetaan historiapiste jos kyseessä on asynkroninen kutsu eikä olla navigoimassa. Historiapisteeseen laitetaan pisteen nimi, arvo ja teksti joka näytetään selaimen otsikossa ja takaisin-napin historiatiedoissa. `ToolScriptManager1_Navigate`-metodissa valitaan jäsenlistasta historiapisteen arvo jos se löytyy, jos ei niin valitaan listan ensimmäinen jäsen (tämä piti tehdä koska historiapistettä ei luoda kuin asynkronisilla kutsuilla). Tässä piti päivittää `UpdatePanel`-kontrollit erikseen koodin puolella, jotta ne muuttuivat takaisin-napilla. Historiapistettä ei voinut luoda sivun ensimmäisellä latauksella, eli takaisin-nappi ei toimi ensimmäisenä valittuun jäseneen. Kirjanmerkit kuitenkin toimivat, koska jäsensivultakin tullessa asetetaan id-

parametri, jonka avulla jäsen haetaan. Historiapistettä ei tarvinnut tarkistaa erikseen sivun latautuessa vaan kirjanmerkit toimivat tuon metodin avulla oikein. ASP.NET oli ainoa kehys, jossa sai helposti muutettua myös sivun otsikon url-fragmentin vaihtuessa.

Railsissa asetettiin `.js.erb`-tiedostossa `window.location.hash`-arvo. Javascript-tiedostossa toteutettiin funktio hashin vaihtumiselle, jossa otettiin Ajax-yhteys palvelimelle ja päivitettiin jäsenen tiedot. Myös sovelluksen latauduttua tarkitetaan hash ja näytetään kyseisen jäsenen tiedot tai asetetaan hashiksi ensimmäisen jäsenen id. Tässä siis homma hoidettiin Javascriptin ja jQueryn avulla. Samalla piti Ajax-yhteyden kautta hakea jäsen kyseiselle id:lle ja päivittää käyttöliittymä `.js.erb`-tiedoston avulla. Pieni ongelma oli, että hashin muuttumismetodi suoritetaan aina myös kun hash asetetaan. Tämän sai kuitenkin aika helposti ratkaistua sessiomuuttujan avulla.

Seasidessa ei saatu url-hashia asetettua ollenkaan. Sen kerhosovelluksessa siis takaisin-nappi ja kirjanmerkit eivät toimi Ajax-toiminnoissa.

9.6.8 Animaatiot

Animaatioiden avulla saadaan näytettyä käyttäjälle Ajax-päivityksessä tehdyt sivun muutokset, koska käyttäjät eivät muuten välttämättä huomaa niitä (luku 2.3).

Vaadin-kehyksessä animaatiot sai luotua *Animator*-lisäosan [67] avulla. Sillä ei saanut kuitenkaan CSS-tyylejä animoitua, eli esimerkiksi pelkästään elementin taustavärin animointi ei onnistunut. Animator-lisäosasta on kuitenkin myös uusi versio, jossa CSS-animointi onnistuu, mutta se ei toimi Internet Explorer -selaimessa [67]. Animator-lisäosan toimimiseksi piti widgetset kääntää ja lisätä se *web.xml*-tiedostossa jokaiseen servlettiin jossa sitä käytetään. Vaadin-kehyksen *Notification*-komponentilla sai helposti näytettyä tallentamisen ilmoitukset. *Notification*-ilmoituksen saa myös muokattua haluamansa näköiseksi CSS:n avulla. Ilmoituksen näkymisajan saa määriteltyä tai sen voi näyttää niin kauan kunnes sitä klikataan. Tällä komponentilla saisi tietysti näytettyä kaikki muutkin päivitys-ilmoitukset, eli varsinaisia animaatioita ei

olisi pakko käyttää.

ASP.NETissä animaatiot saatiin luotua *Ajax Control Toolkit*-lisäosan animaatioilla. Animaatio saatiin liitettyä tiettyyn `UpdatePanel`-kontrolliin, jonka päivittyessä animaatio näytetään. Kuitenkin kun se liitettiin jäsenlistan `UpdatePanel`-kontrolliin, niin animaatio näkyi myös kun valitsi uuden jäsenen listasta. Luotiin sitten tyhjä `UpdatePanel`-kontrolli, jonka päivittymiseen kyseinen animaatio liitettiin. Sitten koodin puolella päivitettiin tämä tyhjä `UpdatePanel`-kontrolli, kun animaatio haluttiin näyttää.

Railsissa animaatiot luotiin puhtaalla jQuery-koodilla. Seasidessa käytettiin Seasiden jQuery-integraatiota, eli käytettiin jQuery-animaatioita, mutta varsinaista jQuery-koodia ei tarvinnut kirjoittaa.

Vaadin-kehyksessä sai tietyn elementin animaatiot lisättyä vain yhteen paikkaan (esimerkiksi kun elementin tiedot muuttuu). Railsissa piti animoida elementit erikseen jokaisessa Ajax-kutsun metodissa, jolla animaation halusi näyttää. Siis animaatiokoodi piti kirjoittaa jokaisen metodin `.js.erb`-mallineeseen. Seasidessa piti myös luoda tietyn elementin animaatio jokaisessa metodissa, jossa se haluttiin näyttää. ASP.NETissä oli helppo näyttää tietty animaatio päivittämällä animaatioon liitetty `UpdatePanel`-kontrolli koodin puolella: Näin siis animaatio tarvittiin luoda vain kerran, vaikka se näytettiinkin monessa toiminnossa.

9.6.9 Toiselle sivulle siirtyminen

Vaadin-kehyksessä uusi sivu luotiin uudella UI-luokalla (ks. luku 4.4). Sovelluksella voi siis olla myös useampi UI-luokka. Uudelle sivulle piti luoda uusi *VaadinServlet*, johon kyseinen luokka liitetään *web.xml*-tiedostossa, jossa myös määritellään sivun osoite. Jos projekti sisältää useita UI-luokkia niin kaikille luokille pitää määritellä osoitteelle alikansio (oletuksena ensimmäiselle UI:lle tulee `/*`). *Request*-parametrit sai vastaan UI-luokan `init`-metodissa, joka vastaanottaa *Vaadin request* -olion, jolta parametrin saa pyydettyä. *Request*-parametri välitettiin sitten parametrina sivukomponentille. Sessiomuuttujien käyttö oli helppoa: Käytettiin `VaadinSession`-oliota

(luku 4.4 ja lista sovelluksen peruselementeistä), johon sai määritellä omat muuttujat ja pyydettyä muuttujien arvoja.

ASP.NETissä uusi sivu luotiin uudella `.aspx`-tiedostolla (WWW-lomakkeella). Sivun osoitteen sai muokattua laittamalla kyseinen tiedosto tiettyyn kansioon: ASP.NET avaa oletuksena kansiosta default.aspx-tiedoston. *Request*-parametreja varten piti sivulle siirtävän linkin osoitteeseen lisätä myös tuo `default.aspx` tai *request*-parametrit eivät toimineet. Muuten *request*-parametrien ja sessioiden käyttö oli hyvin yksinkertaista.

Railsissa uusi sivu luodaan uudella ohjaimella ja sen toiminnolla. Sivun osoitteen saa määriteltyä *routes.rb*-tiedostossa, mutta se määritellään myös automaattisesti osoitteeseen *ohjaimennimi/toiminnonnimi*. *Request*-parametrien ja sessioiden käyttö oli yksinkertaista. Sessiomuuttujia joutuikin käyttämään aika paljon muuttujien tallentamiseen pyyntöjen välillä.

Seasidessa uusi sivu luodaan uutena `WAComponent`-luokan perivänä luokkana, joka liitetään tiettyyn Seaside-sovellukseen esimerkiksi Seasiden konfiguraatiosivulla. Uuden sovelluksen saa myös luotua konfiguraatiosivulla. Osoite määrätään sovelluksen luomisessa (sovelluksen nimi) eikä luokan nimellä. *Request*-parametrit sai käyttöön ilman mitään lisätoimia. Sessioita varten piti luoda oma sessiokomponentti, joka liitetään sivulle esimerkiksi sovelluksen konfiguraatiosivulla. Lisäksi sovelluksen konfiguraatiosivulla piti muuttaa session tallennusmuoto evästeisiin tai sessiot eivät toimineet. Sessiomuuttujat tallennettiin ja haettiin sessiokomponentin metodien kautta. Komponenttiin luotiin myös metodi, joka tarkisti, onko tiettyä sessiomuuttujaa määritelty.

9.6.10 Tarvittavat kielet

Taulukossa 9 esitetään kerhosovellusten luomisessa käytetyt kielet eri kehyksillä. Huomataan, että Seasidessa päästiin kaikkein vähimmällä. Seasidessa kylläkin jQueryn osaamisesta on hyötyä, ja varsinkin animaatioiden kohdalla piti jQueryn APIa tutkia. Myös HTML-elementit piti Seasidea käytettäessä olla hallussa. Ruby on

Railsissa eri kieliä pitää osata eniten: Se oli ainoa kehys, jossa piti kirjoittaa oikeaa Javascript- ja jQuery-koodia. Itse asiassa tuota Javascript-koodia tuli vieläpä aika paljon, sekä omaan `.js`-tiedostoon että myöskin `.js.erb`-mallineisiin.

Vaadin oli ainoa kehys, jossa HTML-koodia tai -elementtejä ei tarvinnut käyttää ollenkaan, eikä niiden osaamistakaan vaadittu. Eikä myöskään Javascriptin tai minkään Javascript-kirjaston käyttöä tarvinnut ymmärtää, kuten ei myöskään ASP.NET-kehyksessä. Käytännössä siis Vaadin- ja ASP.NET-kehyksissä selvittiin vähimmällä määrällä kieliä. Vaadin-kehyksessä kaikki varsinainen sovellus- ja ulkoasukoodi tehtiin Javalla. XML:ää tarvittiin ainoastaan *web.xml*-tiedostossa uuden sivun servletin luomisessa ja sivun osoitteiden määrittämisessä sekä *Vaadin Push* -lisäosan konfiguroinnissa. ASP.NETissä taas HTML-koodia joutui käyttämään paljon sivun ulkoasun luomisessa, joten se piti olla hallussa. Kaikissa kehyksissä tyylejä muokattiin CSS:n avulla, eli sen osaaminen näyttää nykyään olevan aika hyödyllistä.

Taulukko 9: Kerhosovelluksissa käytetyt kielet kehyksittäin

Kehys	**Käytetyt kielet**
Vaadin	- Java - CSS - XML
ASP.NET	- C# - HTML - CSS
Ruby on Rails	- Ruby - HTML - CSS - jQuery - Javascript
Seaside	- Smalltalk - CSS

9.6.11 Suorituskyky

Taulukossa 10 esitetään eri toimintojen suoritusaikojen keskiarvoja eri sovelluskehyksillä. Kukin toiminto suoritettiin viisi kertaa ja niistä laskettiin keskiarvo.

Jäsenen valinnassa Vaadin-kehyksellä ei tullut mitään tuloksia. Verkkoliikenne näytti vaan tyhjää tuossa toiminnossa. Huomataan kuitenkin, että muissa Ajax-toiminnoissa Vaadin oli kaikkein nopein, joten luultavasti se pärjäisi vertailussa myös tuossa jäsenen valinnassa. Vaadin-kehyksellä kuitenkin toiselle sivulle siirtyminen oli selvästi hitainta, siinä `styles.css`-tiedoston lataaminen vei aina eniten aikaa (useita sekunteja).

Ruby on Rails -kehyksellä Ajax-toimintojen suorittaminen vei kaikkein eniten aikaa. Se oli reilusti hitaampi muihin kehyksiin verrattuna. Toiselle sivulle siirtyminen oli kuitenkin nopeaa.

Seasidella uudelle sivulle siirtyminen oli todella nopeaa, ja Ajax-toiminnot suoritettiin Vaadin-kehyksen tasoisesti. Tämä on siis suorituskyvyltään Vaadin-kehystä parempi vaihtoehto, jos rikkaiden toimintojen lisäksi sovellus sisältää useita sivuja.

ASP.NET pärjäsi myös ihan hyvin Ajax-toimintojen suoritusnopeudessa, mutta oli kuitenkin aina Vaadin-kehystä ja Seasidea jonkin verran hitaampi. Toiselle sivulle siirtyminen oli sillä selvästi Vaadin-kehystä nopeampaa, mutta silti reilusti kahta muuta kehystä hitaampaa.

9.7 Yhteenveto ja pohdintaa

Vaadin- ja Ruby on Rails -kehyksillä saatiin luotua täysin suunnitelman mukainen sovellus. ASP.NET-sovelluksessa harrastuksen valitsemisessa poistoa varten pitää tekstikenttään kirjoittaa jotain ennen kuin harrastus valitaan. Seasidessa taas ei saatu kirjanmerkkejä ja takaisin-nappia toimimaan Ajax-päivityksissä.

Vaadin oli ainoa kehys, jolla ei tullut vastaan mitään täysin odottamattomia ongelmia (taulukko 11). Vaikka tietysti siinäkin piti joissain kohdissa vähän tuskailla, mutta on-

Taulukko 10: Kerhosovellusten toimintojen suoritusaikojen keskiarvot eri sovelluskehyksillä

Kehys	**Jäsenen valinta**	**Jäsenkentän asetus**	**Harrastuksen valinta**	**Jäsenen tallennus**	**Toiselle sivulle siirtyminen**
Vaadin	Ei tuloksia.	34 ms	36 ms	35 ms	3.1 s
ASP.NET	49 ms	63 ms	67 ms	46 ms	937 ms
Ruby on Rails	340 ms	87 ms	220 ms	107 ms	458 ms
Seaside	27 ms	41 ms	39 ms	32 ms	136 ms

gelmat johtuivat kuitenkin aina omasta koodista tai siitä ettei jotain toimintaa täysin ymmärretty. Muilla sovelluskehyksillä tuli vastaan täysin kehyksistä johtuvia ongelmia: Joko kehysten ominaisuuksista johtuvia tai sitten ihan vain bugeja. Varsinkin dialogien kanssa oli muilla kehyksillä hankaluuksia. Vaadin onkin suunniteltu juuri rikkaiden sovellusten luomiseen, kun taas muissa kehyksissä Ajax-toiminnallisuudet on lisätty niihin jälkeenpäin.

Taulukko 11: Ongelmat kerhosovelluksen luomisessa eri kehyksillä

Kehys	**Ongelmat**
Vaadin	- Ei ongelmia.
ASP.NET	- `Page_Init`- ja `Page_Load`-metodien suoritus jokaisella Ajax-pyynnöllä. - Attribuutit nollautuvat jokaisella pyynnöllä. - Jokainen dynaamisesti luotu elementti piti luoda jokaisella takaisinkutsulla (myös dialogin sisältö). - Tekstikentällä ei ollut muita tapahtumia kuin tekstin vaihtuminen, ja sekin lähetettiin vasta kentästä poistuttaesta.

Ruby on Rails	- Attribuuttien arvoja ei säilytetä pyyntöjen välillä. - jQueryn normaali dialogi ei suostunut toimimaan (piti käyttää lisäosaa). - Ei voi luoda omaa komponenttia (paitsi ehkä lisäosan avulla).
Seaside	- Valintalistan arvon ja näkyvän tekstin erottaminen. - Url-hashia ei saanut asetettua. - Dialogin ongelmat Seaside 3.1 -versiossa.

Railsissa oli vielä ongelmia ääkkösten kanssa. Ääkkösiä ei saatu toimimaan ohjaimelta tulevalla tekstillä. Koko ohjelma kaatui, jos yritti ääkkösiä syöttää ohjaimelta näkymälle. Näkymässäkin piti jokaiseen partialiin erikseen määritellä koodaus, joka sitten toistui varsinaisessa HTML-koodissa useaan kertaan.

Seasidessa oli eroja sovelluksen toiminnassa eri Smalltalk-ympäristöissä ja Seasiden versioissa. Pharo 2.0 ja Seaside 3.1 -versiossa oli ongelmia dialogin kanssa: Kun sivulle tuli linkin kautta, piti sivu päivittää uudelleen ennen kuin dialogi toimi (muuten näkyi vaan dialogin lomake sivulla ilman dialogin aukaisunappia). Kuitenkin Pharo 1.4 ja Seaside 3.0 -versiossa tuo toimi sitten ihan oikein. Jäsenen tallentaminen ei sitten onnistunut Pharo 1.4 ja Seaside 3.0 -versiolla. Tässä kuitenkin virhe tuli varsinaisesta kerho-koodista, eikä itse Seaside-kehyksestä. Tuo koodi siis tehtiin ensin Pharo 2.0 -versiolla, eli ilmeisesti tähän versioon on tullut jotain päivityksiä, jotka ei toimi tuossa vanhassa.

Vaadin-kehyksessä aloittelijan kynnys on aika pieni, varsinkin jos Java ja vaikka Swing-kirjasto ovat entuudestaan tuttuja. Tällöin opeteltavaksi jää oikeastaan vain Vaadin API. Ulkoasunkin saa helposti luotua Vaadin editorilla, ja siitä saa myös otettua mallia, kuinka elementtejä luodaan koodin puolella. Myöskään luovuutta ei juurikaan rajoiteta. Ulkoasun saa muokattua mieleisekseen luomalla oman teeman, jos valmiit eivät kelpaa. Vaadin-kehyksessä saa myös luotua täysin omia asiakaspuolen käyttöliittymäkomponentteja ja myös Javascriptin lisääminen sivulle onnistuu.

Myös ASP.NET-kehyksessä aloittelijan kynnys on melko pieni, jos Visual Studio ja vaikkapa C#-kieli ovat tuttuja, ja varsinkin jos on tehnyt aiemmin Windows Forms- tai WPF-sovelluksia. Visual Studion visuaalinen editori auttaa myös komponenttien luomisessa ja muokkaamisessa, ja myös niiden tapahtumankäsittelijöiden luomisessa. Tässä pitää kuitenkin myös HTML-koodin olla hallussa. ASP.NET-kehyksellä saa luotua täysin omia palvelinkontrolleja (jos valmiit eivät riitä) ja omille komponenteille saa myös luotua omia tapahtumia.

Ruby on Railsissa aloittelijan kynnys on hieman isompi, vaikka Ruby-kieli olisikin hallussa. Siinä on kuitenkin aika paljon opeteltavaa eri osien luomisessa ja niiden välisessä vuorovaikutuksessa, varsinkin jos aikoo tehdä Ajax-sovelluksia. Myös reittien käyttäminen ja ymmärtäminen voi aluksi tuntua hankalalta. Rails myös mielestäni rajoittaa jonkin verran luovuutta. Varsinkin komponenttien tietojen välittäminen palvelimelle on aika rajoittunutta, mutta sitä voi onneksi kiertää Javascript-tiedoston avulla.

Myös Seasidessa aloittelijan kynnys on hieman isompi. Mutta auttaa varmasti, jos Smalltalk on entuudestaan hallussa. Sillä on kuitenkin aika omanlainen toimintatapa (esimerkiksi takaisinkutsut (luku 7.2) ja kontrollivirrat (luku 7.3)). Tosin noita kontrollivirtoja (jotka oikeastaan ovat yksi Seasiden ominaispiirre) ei tarvitse Ajax-toiminnoissa käyttää tai osata, eikä niitä käytetty kertaakaan koko kerhosovelluksen luomisessakaan. Seasidessa käyttöliittymäkomponentit rajoittuvat HTML-komponentteihin (jQuery UI:n ja Scriptaculouksen komponentit ovat tietysti käytettävissä), eikä omien käyttöliittymäkomponenttien luominen taida olla mahdollista.

Kaikissa kehyksissä piti käyttää lisäosia sovelluksen luomisessa. Vaadin-kehyksen lisäosat löytyi Vaadin-kehyksen kotisivuilta ja ne lisättiin sovellukseen JAR-tiedostoina (onnistuisi myös Mavenin tai Ivyn avulla) [66]. ASP.NETissä jouduttiin lisäämään *Ajax Control Toolkit* -lisäosa, jonka lisääminen onnistui kokonaan Visual Studion kautta. Railsissa kirjastojen lisääminen onnistui lisäämällä uusi *gem*-määritys *Gemfile*-tiedostoon. Seasidessa ei varsinaisesti tarvinnut hakea mitää uusia kirjastoja, kun vaan lisäsi ne käyttöön sivun konfiguraatiosivulla (Seaside 3.0-versiosta ylöspäin on jQuery oletuksena asennettuna).

Taulukossa 12 kuvataan eri kehyksien plussia ja miinuksia, joita kerhosovellusten luomisessa havaittiin. Miinukset siis noiden taulukon 11 ongelmien lisäksi.

Yleisesti voidaan sanoa, että tämän tutkimuksen perusteella Vaadin on näistä sovelluskehyksistä paras vaihtoehto rikkaiden WWW-sovellusten luomiseen. Ruby on Rails taas vaikutti kaikkein heikoimmalta. Tosin Seasidekin voidaan laskea muita heikommaksi, kun otetaan huomioon dialogin ongelmat Seaside 3.1 -versiossa sekä kirjanmerkkien ja takaisin-napin toimimattomuus. Muuten Seaside pärjäsi sovelluksen luomisessa Railsia paremmin. ASP.NET voidaan sijoittaa Vaadin-kehyksen jälkeen toiseksi. Siinä positiivista oli komponenttien käyttämisen helppous ja Ajax-tominnallisuuksien luominen ilman Javascriptin tai jQueryn ymmärtämistä. Heikkoutena siinä oli tekstikentän tapahtumat (vain arvon vaihtumiselle, kun tekstikentästä poistutaan) ja se, että kaikki attribuutit nollautuvat jokaisella pyynnöllä.

Lopuksi pitää kuitenkin mainita, että Vaadin-kehys oli tuttu jo ennen vertailua (sillä oli jo tehty erilaisia sovelluksia), mikä saattoi tietysti vaikuttaa arvioon jonkin verran. Myös Ruby on Railsia oli tosin käytetty hieman jo aiemmin, sen verran että sen perustoiminnallisuus oli jotenkin hallussa. Tämä nyt ei ainakaan Railsin arviossa näkynyt. ASP.NET- ja Seaside- kehyksiin tutustuttiin vasta vertailun aikana. Tämä vertailu ei tietysti muutenkaan ollut aivan täydellisen kattava, esimerkiksi tietokantojen käyttäminen jätettiin tästä kokonaan pois (tämä olisi voinut parantaa ainakin Railsin arviota). Myöskään sovelluksen julkaisemista (ja julkaistun sovelluksen päivittämistä) ei tässä otettu huomioon, eikä sovelluksen muokattavuutta (ulkoasun/toiminnallisuuden muuttamista).

Taulukko 12: Vertailukehysten plussat ja miinukset

Kehys	**Plussat ja miinukset**
Vaadin	+ Ei tarvitse huolehtia Ajax-toiminnallisuuksien luomisesta. + Visuaalinen editori. + Ulkoasukomponentit. + Kaikki koodi puhdasta Java-koodia (lukuunottamatta web.xml-konfiguraatiotiedostoa ja tyylitiedostoa). + Ulkoasu onnistuu ilman HTML-koodia. + Taulukko-komponentti, joka luo automaattisesti lajittelun sarakkeittain otsikkoa klikkaamalla. + Komponenttien käyttäminen ja muokkaaminen koodin puolelta helppoa. + Paljon lisäosia helpottamaan sovellusten luomista. + Varmistusdialogiin sai helposti valitun jäsenen nimen tai harrastuken alan näkyviin. + Ajax-toimintojen suorituskyky. - Tapahtumankäsittelijöiden luominen ei onnistu visuaalisen editorin kautta. - Odotusilmaisimen näyttämistä varten piti lisätä erillinen kirjasto. - Toiselle sivulle siirtyminen hidasta.

ASP.NET	+ Ajax-toiminnallisuudet kokonaan ulkoasukoodissa. + Visuaalinen editori. + Tapahtumankäsittelijöiden luominen onnistuu visuaalisen editorin kautta. + Komponenttien käyttäminen ja muokkaaminen koodin puolelta helppoa. + Ajax Control Toolkit -lisäosa. - UpdatePanelit (ja muu Ajax-toiminnallisuus) sekoittaa ulkoasukoodia. - Tyylitiedosto pitää päivittää itse selaimessa muutosten jälkeen.
Seaside	+ Kaikki koodi puhdasta Smalltalk-koodia (lukuunottamatta tyylitiedostoja). + Toimiva jQuery-integraatio. + Varmistusdialogin helppo luominen. + Suorituskyky hyvä kaikissa toiminnoissa. - Pitää osata HTML-tekniikka (ja hieman myös ymmärtää jQueryä). - Omien sessiomuuttujien käyttöönotto turhan hankalaa. - Tyylitiedoston käyttöönotto turhan hankalaa. - Tyylitiedosto pitää päivittää itse selaimessa muutosten jälkeen.

Ruby on Rails	+ Ulkoasu ja sovelluslogiikka erillään. + jQuery puhdasta jQuery-koodia (jos osaa sitä). + Varmistusdialogin helppo luominen. + Toiselle sivulle siirtyminen nopeaa. - jQuery puhdasta jQuery-koodia (jos ei osaa sitä). - Sovellus jakautuu moneen osaan (eri tiedostojen suuri määrä). - Komponenttien käyttäminen (esim. niiden arvojen tai tyylinimien välittäminen ohjaimelle). - Javascript-koodin suuri määrä. - Javascript- ja Ruby-koodit sekaisin (.js.erb). - HTML- ja Ruby-koodit sekaisin (.html.erb). - Ruby-koodi sekoittaa ulkoasukoodia. - Kaikille Ajax-kutsuille piti luoda reitti routes.rb-tiedostoon. - Ongelmat ääkkösten kanssa. - Ajax-toimintojen suorituskyky.

10 Yhteenveto

Ajaxin käyttäminen WWW-sovelluskehityksessä on nykyään runsasta. Käyttäjät ovat jo tottuneet työpöytätyylisiin käyttöliittymiin ja osaavat jo odottaa sellaisia. Erilaisia WWW-sovelluskehyksiä, jotka auttavat myös Ajax-toiminnallisuuksien luomisessa, on nykyään tarjolla runsaasti ja niiden väliltä valitseminen voi olla hankalaa.

Tässä tutkielmassa esiteltiin ensin rikkaat WWW-sovellukset sekä eri tapoja luoda niitä. Esiteltiin myös niistä saatavia hyötyjä ja niiden aiheuttamia ongelmia. Ajax-sovellusten luomiseen perehdyttiin luomalla esimerkkinä *valuuttamuunnin*-sovellus, jossa Ajax-toiminot toteutettiin ilman sovelluskehysten apua. Sitten esiteltiin neljä vertailuun otettua sovelluskehystä ja niiden lisäksi myös muita hieman erilaisia lähestymistapoja tarjoavia sovelluskehyksiä. Sovelluskehyksiä vertailtiin luomalla niillä Ohjelmointi 2 -kurssin malliharjoitustyön (Kerho-sovelluksen) WWW-versio.

Vertailussa huomattiin, että Vaadin, joka on suunniteltu juuri rikkaiden käyttöliittymien toteuttamiseen, suoriutui tehtävästä varsin hyvin. Sillä ei tarvinnut Ajax-toiminnallisuuksien toteuttamista erikseen miettiä: Tapahtumankäsittelijöissä teki vaan käyttöliittymään tarvittavat muutokset, ja ne päivittyivät sivulle automaattisesti. Eikä Vaadin-kehyksestä paljoakaan huonoja puolia löytynyt. Suorituskykykin oli Ajax-toiminnoissa kaikkein parasta. Yksi huono puoli oli kuitenkin toiselle sivulle siirtymisen hitaus. Muilla sovelluskehyksillä, joihin Ajax-ominaisuudet on lisätty jälkeenpäin, tuli vastaan erilaisia ongelmia.

Ruby on Rails pärjäsi vertailussa kaikkein huonoimmin. Siinä Ajax-toiminnallisuuksien luomiseen joutui luomaan monia eri osia, ja myös käyttämään oikeaa Javascript-koodia. Railsin suorituskyky Ajax-toiminnoissa oli myös kaikkein heikointa. Railsissa joutui myös sekoittamaan Ruby-koodia HTML- ja Javascript-koodien sekaan, mikä teki koodista aika sekavaa. Toisaalta Ruby on Rails oli ainoa sovelluskehys Vaadin-kehyksen lisäksi, jolla täysin suunnitelman mukainen sovellus saatiin toteutettua.

ASP.NET tarjosi aivan omanlaisen keinon Ajax-toiminnallisuuksien tekemiseen: `UpdatePanel`-kontrollien avulla Ajax-toiminnot sai luotua kokonaan ulkoasusi-

vun puolella. Huono puoli tässä oli se, että tuo sekoitti ulkoasukoodia aika paljon. ASP.NET-kehyksellä ei onnistunut tekstikentän klikkaustapahtuman luominen, joten poistettavan harrastuksen valinta kenttää klikkaamalla ei onnistunut.

Seasidella ei saatu takaisin-nappia ja kirjanmerkkejä toimimaan Ajax-toiminnoissa, koska Url-hashia ei saanut asetettua. Seasiden suorituskyky oli todella hyvä, myös Ajax-toiminnoissa, mutta varsinkin uudelle sivulle siirtymisessä. Seasiden vahvuus oli myös se, että kaikki koodi oli puhdasta Smalltalk-koodia. Kuitenkin HTML-elementit piti olla hallussa, ja hieman piti myös ymmärtää jQueryä. Sessioiden käyttö oli Seasidessa turhan hankalaa, kun omien sessiomuuttujien käyttöä varten piti tehdä oma sessioluokka, jonka kautta muuttujat asetetaan ja palautetaan. Muissa sovelluskehyksissä omien sessiomuuttujien käyttäminen onnistui suoraan määrittelemällä uusi sessiomuuttuja.

Vaadin ja ASP.NET erottuivat muista sovelluskehyksistä niiden komponenttien käytön ja muokkaamisen helppoudella. Komponentteihin pääsi käsiksi suoraan koodin puolelta ja niiltä sai pyydettyä arvoja ja määriteltyä eri ominaisuuksia. Seasidessa ja Ruby on Railsissa komponenttien dynaamiset muutokset piti tehdä jQueryn avulla Ajax-kutsuissa. Railsissa komponenttien arvot sai välitettyä ohjaimelle vain, kun ne oli lomakkeen sisällä, muuten arvot piti välittää Javascript-tiedoston kautta.

Vaadin-kehyksen kaltaisille sovelluskehyksille on varmasti tulevaisuudessa vielä enemmän tarvetta, kun normaaleja työpöytäsovelluksia aletaan tarjoamaan entistä enemmän Internetin kautta. Jatkotutkimuksen aihetta varmasti olisi ottaa vertailuun myös muita juuri rikkaiden sovellusten luomiseen kehitettyjä kehyksiä. Myös tietokantojen käyttö jäi tästä vertailusta kokonaan pois, vaikka se onkin nykyään tärkeä osa WWW-sovelluksia.

Kirjallisuutta

[1] *Apache Wicket: Features*, saatavilla WWW-muodossa <URL: http://wicket.apache.org/meet/features.html>, viitattu 28.06.2014.

[2] *apotomo.de: Peter's Guide*, saatavilla WWW-muodossa <URL: http://apotomo.de/peters-guide-1.1/introduction.html>, viitattu 17.11.2014.

[3] *asp.net: Get Started with ASP.NET Web Sites*, saatavilla WWW-muodossa <URL: http://www.asp.net/get-started/websites>, viitattu 26.11.2014.

[4] *ASP.NET Tutorial*, saatavilla WWW-muodossa <URL: http://www.w3schools.com/ASPNET/>, viitattu 08.05.2014.

[5] *ASP.NET Web Forms - Tutorial*, saatavilla WWW-muodossa <URL: http://www.w3schools.com/ASPNET/aspnet_intro.asp>, viitattu 09.05.2014.

[6] Aneesha Bakharia, *JavaServer Pages Fast & Easy Web Development*, Prima Publishing, 2002.

[7] Vangie Beal, *webopedia.com: DLL*, saatavilla WWW-muodossa <URL: http://www.webopedia.com/TERM/D/DLL.html>, viitattu 03.12.2014.

[8] Vangie Beal, *webopedia.com: HTTP - HyperText Transfer Protocol*, saatavilla WWW-muodossa <URL: http://www.webopedia.com/TERM/H/HTTP.html>, viitattu 03.12.2014.

[9] Vangie Beal, *webopedia.com: JAR*, saatavilla WWW-muodossa <URL: http://www.webopedia.com/TERM/J/JAR.html>, viitattu 07.12.2014.

[10] Vangie Beal, *webopedia.com: PHP*, saatavilla WWW-muodossa <URL: http://www.webopedia.com/TERM/P/PHP.html>, viitattu 07.12.2014.

[11] Vangie Beal, *webopedia.com: SQL - structured query language*, saatavilla WWW-muodossa <URL: http://www.webopedia.com/TERM/S/SQL.html>, viitattu 02.12.2014.

[12] Vangie Beal, *webopedia.com: World Wide Web*, saatavilla WWW-muodossa <URL: http://www.webopedia.com/TERM/W/World_Wide_Web.html>, viitattu 02.12.2014.

[13] Vangie Beal, *webopedia.com: WSDL*, saatavilla WWW-muodossa <URL: http://www.webopedia.com/TERM/W/WSDL.html>, viitattu 03.12.2014.

[14] Edward Benson, *The Art of Rails*, Wiley Publishing, 2008.

[15] Andrew P. Black, Stéphane Ducasse, Oscar Nierstrasz ja Damien Pollet, *Pharo by Example*, Square Bracket Associates, 2009.

[16] Michael Bächle ja Paul Kirchberg, *software technology: Ruby on Rails*, IEEE Computer Society, 2007.

[17] Derek Chen-Becker, Marius Danciu ja Tyler Weir, *Exploring Lift Lift 2.0 Edition*, 2012.

[18] Karolina Czekalska, Bartosz Sakowicz, Jan Murlewski ja Andrzej Napieralski, *Hotel Reservation System Based on the JavaServer Faces Technology*, Proceedings of the International Conference TCSET'2008, 2008.

[19] Andrea Del Bene, Martin Grigorov, Carsten Hufe, Christian Kroemer, Daniel Bartl ja Paul Bor, *Apache Wicket User Guide - Reference Documentation*, The Apache Software Foundation, 2014.

[20] *django-dajax 0.9 documentation*, saatavilla WWW-muodossa <URL: http://django-dajax.readthedocs.org/en/latest/>, viitattu 29.05.2014.

[21] *django-dajaxice 0.6 documentation*, saatavilla WWW-muodossa <URL: http://django-dajaxice.readthedocs.org/en/latest/>, viitattu 29.05.2014.

[22] *Django documentation: Django at a glance*, saatavilla WWW-muodossa <URL: https://docs.djangoproject.com/en/1.6/intro/overview/>, viitattu 14.05.2014.

[23] Stéphane Ducasse, Adrian Lienhard ja Lukas Renggli, *Seaside - A Multiple Control Flow Web Application Framework*, ESUG Conference 2004 Research Track, 2004.

[24] Stéphane Ducasse ja Lukas Renggli, *Seaside: A Flexible Environment for Building Dynamic Web Applications*, IEEE Computer Society, 2007.

[25] Stéphane Ducasse, Lukas Renggli, C. David Shaffer, Rick Zaccone ja Michael Davies, *Dynamic Web Development with Seaside*, 2014, saatavilla WWW-muodossa <URL: http://book.seaside.st/book>.

[26] Jason Farrell ja George S. Nezlek, *Rich Internet Applications The Next Stage of Application Development*, Proceedings of the ITI 2007 29th Int. Conf. on Information Technology Interfaces, 2007.

[27] Xinyang Feng, Jianjing Shen, Ying Fan, *REST: An Alternative to RPC for Web Services Architecture*, 2009 First International Conference on Future Information Networks, 2009.

[28] Nicolas Fränkel, *Learning Vaadin*, Packt Publishing, 2011.

[29] Nicolas Fränkel, *Learning Vaadin 7 (2nd Edition)*, Packt Publishing, 2013.

[30] John Gadbois, *Using Unobtrusive JavaScript and AJAX with Rails 3*, 19.10.2010, saatavilla WWW-muodossa <URL: http://code.tutsplus.com/tutorials/using-unobtrusive-javascript-and-ajax-with-rails-3-net-15243>.

[31] Gaylord, Jason N. Wenz, Christian Rastogi ja Pranav , *Professional ASP.NET 4.5 in C# and VB*, Wiley Publishing, 2013.

[32] *GitHub.com: apotonick/cells*, saatavilla WWW-muodossa <URL: https://github.com/apotonick/cells>, viitattu 17.11.2014.

[33] *GitHub.com: dei79/jquery-modal-rails*, saatavilla WWW-muodossa <URL: https://github.com/dei79/jquery-modal-rails>, viitattu 20.11.2014.

[34] Marko Grönroos, *Book of Vaadin:Vaadin 7 Edition - 4th Revision*, Vaadin Ltd, 2014.

[35] *GWT Documentation: Ajax Communication*, saatavilla WWW-muodossa <URL: http://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html>, viitattu 02.12.2014.

[36] *GWT Overview*, saatavilla WWW-muodossa <URL: http://www.gwtproject.org/overview.html>, viitattu 02.12.2014.

[37] Jonathan Hayward, *Django JavaScript Integration : AJAX and JQuery*, Packt Publishing Ltd, 2011.

[38] Ian Hlavats, *JSF 1.2 Components : Develop Advanced Ajax-enabled JSF Applications*, Packt Publishing Ltd , 2010.

[39] Jaroslav Holaň ja Ondřej Kvasnovský, *Vaadin 7 Cookbook*, Packt Publishing, 2013.

[40] Ayman Hourieh, *Web 2.0 Website Programming with Django : Step Through the Development Of A Complete Social Bookmarking Application with the Python Web*

Framework That Encourages Clean And Rapid Development, Packt Publishing Ltd, 2008.

[41] Daniel Kehoe, *Ruby on Rails Version History*, saatavilla WWW-muodossa <URL: http://railsapps.github.io/rails-release-history.html>, 17.11.2014.

[42] Daniel Kehoe, *What is Ruby on Rails?*, saatavilla WWW-muodossa <URL: http://railsapps.github.io/what-is-ruby-rails.html>, 11.10.2013.

[43] Bart Kummel, *Apache Myfaces 1.2 Web Application Development : Building Next-Generation Web Applications With Jsf And Facelets*, Packt Publishing Ltd , 2009.

[44] Wei-Meng Lee, *What Is ASP.NET*, saatavilla WWW-muodossa <URL: http://www.windowsdevcenter.com/pub/a/dotnet/2005/09/19/what-is-asp-net.html>, 19.09.2005.

[45] Boci Lin, Yan Chen, Xu Chen ja Yingying Yu, *COMPARISION BETWEEN JSON AND XML IN APPLICATIONS ON AJAX*, 2012 International Conference on Computer Science and Service System, 2012.

[46] Jonathan Locke, *Meet Apache Wicket*, saatavilla WWW-muodossa <URL: http://wicket.apache.org/meet/introduction.html>, viitattu 28.06.2014.

[47] Matthew MacDonald, Adam Freeman ja Mario Szpuszta, *Pro ASP.NET in C# 2010, Fourth Edition*, Apress , 2010.

[48] Julie Meloni, *PHP Essentials*, Premier Press , 2003.

[49] Ali Mesbah ja Arie van Deursen, *Migrating Multi-page Web Applications to Single-page AJAX Interfaces*, 11th European Conference on Software Maintenance and Reengineering (CSMR'07), 2007

[50] Scott Newman, *Django 1.0 Template Development*, Packt Publishing Ltd, 2008.

[51] Daniel Pavlic, Mile Pavlic ja Vladan Jovanovic, *Future of Internet Technologies*, MIPRO 2012, 2012.

[52] Nicolas Petton ja Steven Costiou, *Iliad developer's guide*, 2010.

[53] Xinyang Qiu, *.NET Web Development and Tools Blog*, saatavilla WWW-muodossa <URL: http://blogs.msdn.com/b/webdev/archive/2014/11/12/an

nouncing-asp-net-features-in-visual-studio-2015-preview-and-vs2013-update-4.aspx>, 12.11.2014.

[54] A. P Rajshekhar, *Building Dynamic Web 2.0 Websites with Ruby on Rails*, Packt Publishing Ltd, 2008.

[55] *Release Notes for Vaadin Framework 7.3.5*, saatavilla WWW-muodossa <URL: http://vaadin.com/download/release/7.3/7.3.5/release-notes.html>, viitattu 26.11.2014.

[56] Sam Ruby, Dave Thomas ja David Heinemeier Hansson, *Agile Web Development with Rails, Fourth Edition*, Pragmatic Programmers, 2011.

[57] Juergen Schumacher ja Markus Stäuble, *ZK Developer's Guide : Developing Responsive User Interfaces For Web Applications Using Ajax, XUL, And The Open Source ZK Rich Web Client Development Framework*, Packt Publishing, 2008.

[58] *seaside.st: History*, saatavilla WWW-muodossa <URL: http://www.seaside.st/about/history>, viitattu 26.11.2014.

[59] *seaside.st: New in Seaside 3.1*, saatavilla WWW-muodossa <URL: http://seaside.st/community/development/seaside31>, viitattu 26.11.2014.

[60] *seaside.st: Success Stories*, saatavilla WWW-muodossa <URL: http://seaside.st/about/users>, viitattu 26.11.2014.

[61] Dan Siepen, *Top 15 sites built with Ruby on Rails*, saatavilla WWW-muodossa <URL: https://thecoderfactory.com/posts/top-15-sites-built-with-ruby-on-rails>, 26.02.2014.

[62] Imar Spaanjaars, *Beginning ASP.NET 4 in C# and VB*, Wrox Press, 2010.

[63] Antero Taivalsaari, Tommi Mikkonen, Dan Ingalls ja Krzysztof Palacz, *Web Browser as an Application Platform*, 34th Euromicro Conference Software Engineering and Advanced Applications, 2008.

[64] *techterms.com: URL*, saatavilla WWW-muodossa <URL: http://www.techterms.com/definition/url>, 01.12.2006.

[65] *The Official YAML Web Site*, saatavilla WWW-muodossa <URL: http://www.yaml.org/>, viitattu 07.12.2014.

[66] *Vaadin Directory*, saatavilla WWW-muodossa <URL:

https://vaadin.com/directory>, viitattu 20.11.2014.

[67] *Vaadin Directory: Animator*, saatavilla WWW-muodossa <URL: https://vaadin.com/directory#addon/animator:vaadin>, viitattu 19.11.2014.

[68] *Vaadin Directory: ConfirmDialog*, saatavilla WWW-muodossa <URL: https://vaadin.com/directory#addon/confirmdialog:vaadin>, viitattu 17.11.2014.

[69] *vaadin.com: GWT*, saatavilla WWW-muodossa <URL: https://vaadin.com/gwt>, viitattu 02.12.2014.

[70] *vaadin.com: Who is Using Vaadin?*, saatavilla WWW-muodossa <URL: https://vaadin.com/who-is-using-vaadin>, viitattu 26.11.2014.

[71] Viswa Viswanathan, *Rapid Web Application Development: A Ruby on Rails Tutorial*, IEEE Computer Society, 2008.

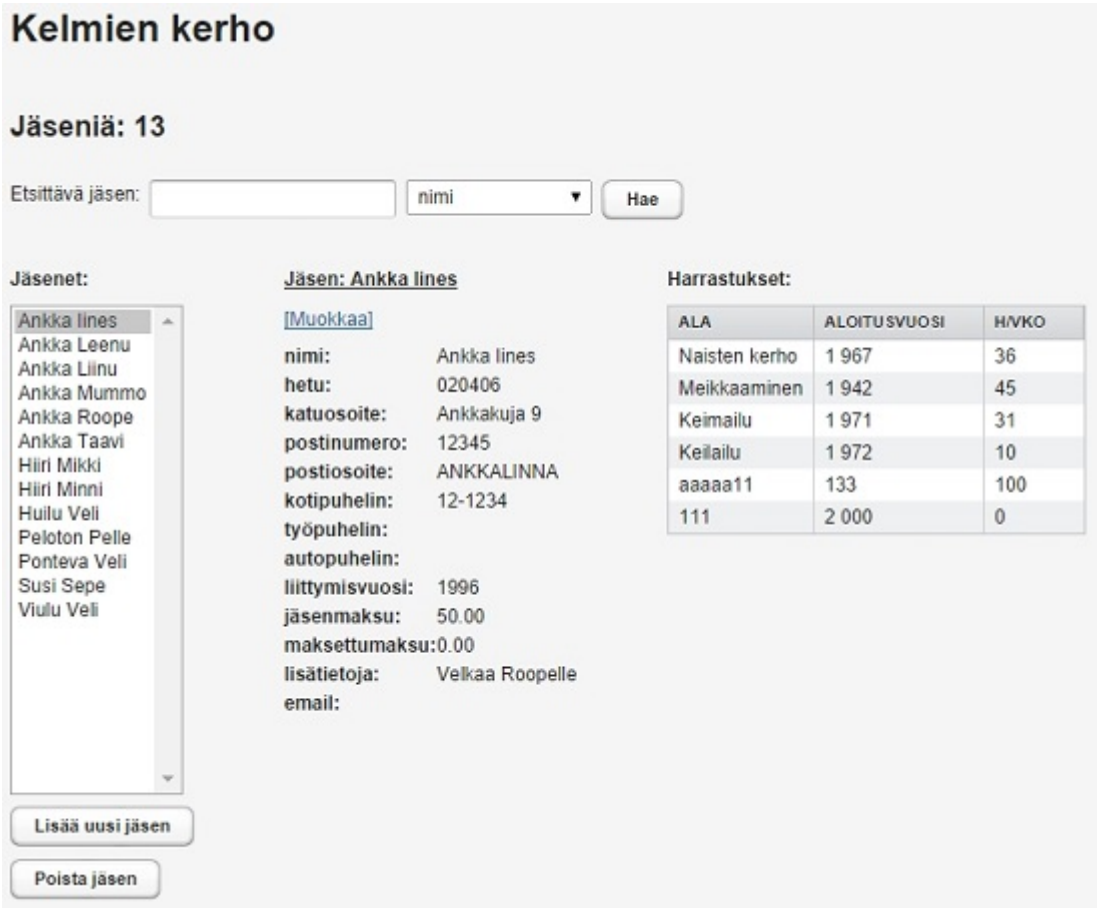
[72] H. Wang ja J. Yang, *Research and Application of Web Development Based on ASP.NET 2.0+Ajax*, IEEE Computer Society, 2008.

[73] Erik Westermann, *Learn XML In a Weekend*, Premier Press, 2002.

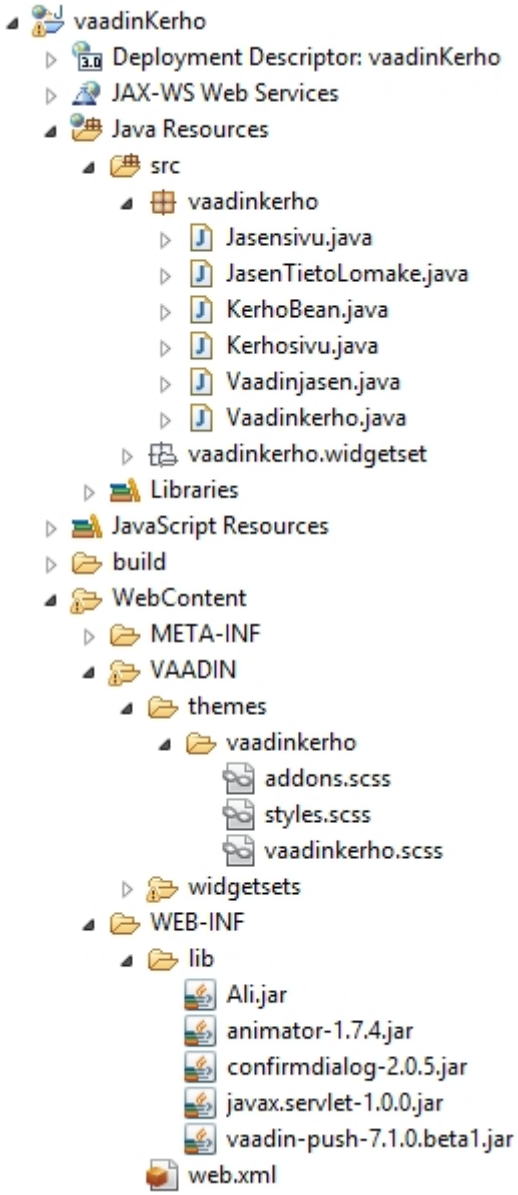
Liitteet

A Vaadin-kerhosovelluksen luominen

Vaadin-kehyksen kerhosovellus koostuu kahdesta UI-luokasta: `Vaadinkerho.java` on kerhosivun luokka ja `Vaadinjasen.java` jäsensivun luokka. Sivujen sisällöt luotiin *Vaadin composite* -komponentteina (`Kerhosivu.java` ja `Jasensivu.java`), joiden avulla ulkoasu saatiin luotua Vaadin-kehyksen visuaalisella editorilla. Jäsentietolomake luotiin uudelleenkäytettävänä komponenttina (`JasenTietoLomake.java`), joka on myös *Vaadin composite* -komponentti. Kuvio 23 näyttää kerhosivun käyttöliittymän ja kuvio 24 näyttää sovelluksen projektirakenteen.



Kuvio 23: Vaadin-kerhosovelluksen kerhosivun käyttöliittymä



Kuvio 24: Vaadin-kerhoprojektin kansiorakenne

A.1 JasenTietoLomake.java

Uudelleenkäytettävä komponentti jäsenen tietojen syöttämiseen. Komponentti on *Vaadin composite* -komponentti, joka laajentaa `CustomComponent`-luokan, aivan kuten `Kerhosivu`- ja `Jasensivu`-komponentitkin. Konstruktorissa luodaan ja asetetaan sivun perusasettelu (`VerticalLayout`) ja luodaan lomakeasettelu (`FormLayout`), johon kentät sijoitetaan:

```
public JasenTietoLomake(Jasen muokattavaJasen) {
   VerticalLayout layout = new VerticalLayout();
   setCompositionRoot(layout);
   setSizeFull();

   lomakeLayout = new FormLayout();
   lomakeLayout.setMargin(true);
   layout.addComponent(lomakeLayout);

   jasen = muokattavaJasen;
   lisaaKentat();
}
```

Komponentilta voi pyytää muokatun jäsenen, tiedon onko kentissä virheitä ja onko nimikenttä tyhjä:

```
public Jasen getJasen() {
   return jasen;
}

public boolean onkoVirheita() {
   return virheita > 0;
}

public boolean nimiTyhja() {
   return nimiKentta.getValue().equals("");
}
```

Lisätään lomakkeeseen kentät jäsenen tiedoille:

```
public void lisaaKentat() {
   for (int i = jasen.ekaKentta(); i < jasen.getKenttia(); i++) {
      fieldJasentieto = luoKentta(i);
      luoKuuntelija();

      lomakeLayout.addComponent(fieldJasentieto);
   }
}
```

Luodaan kenttä tietylle jäsenen kysymykselle. `setImmediate(true)` asettaa kentän aiheuttamaan tapahtuman itse ja `setTextChangeEventMode(TextChangeEv`

`entMode.EAGER)` asettaa tapahtuman tapahtuvaksi heti kun kenttään on syötetty uusi arvo:

```
public TextField luoKentta(int i) {
   fieldJasentieto = new TextField(jasen.getKysymys(i) + ": ");
   fieldJasentieto.setImmediate(true);
   fieldJasentieto.setTextChangeEventMode(TextChangeEventMode.EAGER
      );
   fieldJasentieto.setId(Integer.toString(i));
   fieldJasentieto.setValue(jasen.anna(i));
   fieldJasentieto.setWidth("150px");
   if (i == jasen.ekaKentta()) nimiKentta = fieldJasentieto;

   return fieldJasentieto;
}
```

Luodaan kentälle kuuntelija tekstin vaihtumiselle. Muokattu kenttä saadaan metodilla `event.getComponent()`:

```
public void luoKuuntelija() {
   fieldJasentieto.addTextChangeListener(new TextChangeListener() {
      public void textChange(TextChangeEvent event) {
         TextField field = (TextField) event.getComponent();
         int kentta = Integer.parseInt(field.getId());

         String virhe = jasen.aseta(kentta, event.getText());
         tarkistaVirhe(virhe, field);
      }
   });
}
```

Tarkistetaan tiedon asetuksessa saatu virhe ja näytetään virhe, jos sellainen löytyy. Virhe näytetään tekstikentän `setComponentError`-metodin avulla, jolle välitetään uusi virhe (`new UserError(virhe)`). Lisäksi kentän tyyli muutetaan metodilla `field.setStyleName("virheKentta")`:

```
public void tarkistaVirhe(String virhe, TextField field) {
   if (virhe != null && field.getComponentError() == null) {
      field.setStyleName("virheKentta");
      virheita++;
   }
   else if (virhe == null && field.getComponentError() != null) {
      field.setStyleName("");
      virheita--;
   }

   if (virhe != null) field.setComponentError(new UserError(virhe))
      ;
   else field.setComponentError(null);
}
```

A.2 Vaadinkerho.java

UI-luokka kerhosivulle. `init`-metodissa luodaan ulkoasukomponentti (`layout`), joka asetetetaan sivun sisällöksi. Ulkoasukomponenttiin lisätään uusi `Kerhosivu`-komponentti, jolle välitetään *request*-parametri `id`:

```
protected void init(VaadinRequest request) {
    final VerticalLayout layout = new VerticalLayout();
    layout.setMargin(true);
    setContent(layout);
    layout.setSizeFull();

    String id = request.getParameter("id");

    Kerhosivu sivu = new Kerhosivu(id);
    layout.addComponent(sivu);
}
```

A.3 Vaadinjasen.java

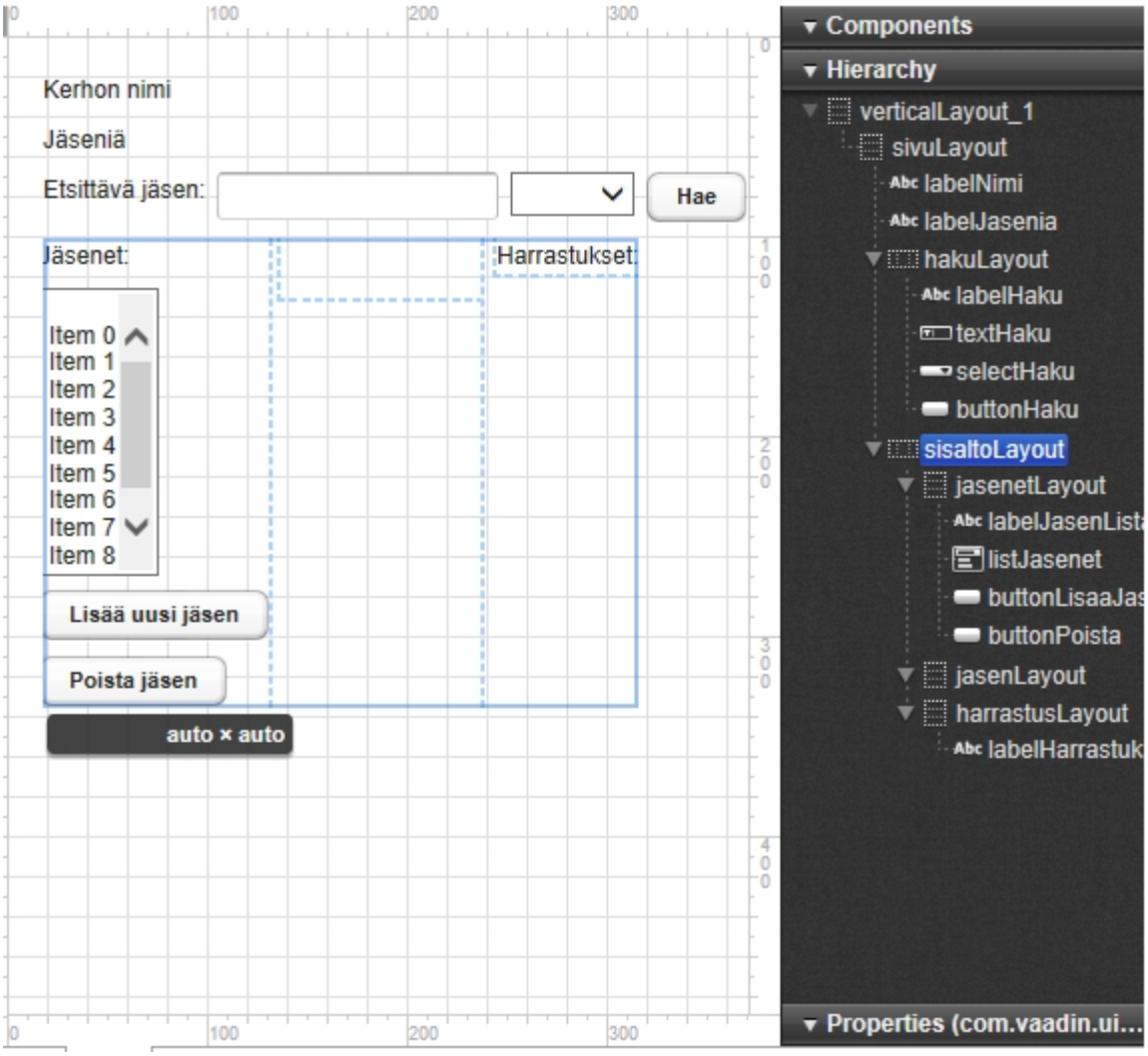
Toinen UI-luokka, jossa tehdään täysin sama kuin `Vaadinkerho.java`-luokassa, paitsi ulkoasukomponenttiin lisätään uusi `Jasensivu`-komponentti.

A.4 Kerhosivu.java

Vaadin composite -komponentti, jossa luodaan kerhosivun sisältö. Sivun perusulkoasu luotiin visuaalisella Vaadin editorilla (kuvio 25).

Konstruktorissa luodaan sivun ulkoasu, lisätään editorissa luoduille napeille ja jäsenvalintakomponentille tapahtumienkuuntelijat (kuuntelija luodaan myös uriFragmentin vaihtumiselle). Luodaan alkiot hakukenttävalintakomponentille ja jäsenvalintakomponentille. Luodaan myös animaatio- ja odotusilmaisin-komponentit. Parametrina saadaan valittavan jäsenen id:

```
public Kerhosivu(String id) {
    buildMainLayout();
    setCompositionRoot(mainLayout);

    proxy = new AnimatorProxy();
    mainLayout.addComponent(proxy);

    odotusIlmaisin = new ProgressBar();
    odotusIlmaisin.setIndeterminate(true);
    odotusIlmaisin.setEnabled(true);
```



Kuvio 25: Vaadin-kerhosivun visuaalinen editori -näkymä

```
    kerho = KerhoBean.getKerho();
    try {
       kerho.lueTiedostosta("kelmit");
    } catch (SailoException e) {
       e.printStackTrace();
    }

    apuJasen = new Jasen();
    apuHarrastus = new Harrastus();

    asetaNimiJaJasenmaara();
    luoHakukentat();

    listJasenet.setNullSelectionAllowed(false);
    listJasenet.setImmediate(true);
    listJasenet.setRows(20);

    buttonHaku.addClickListener(this);
    buttonLisaaJasen.addClickListener(this);
    buttonPoista.addClickListener(this);

    lisaaKuuntelijat();
    haeJasenetJaPaivitaLista(id);
}
```

Asetetaan sivun otsikko ja kerhon jäsenien lukumäärä. Käytetään `Label`-komponentin HTML-sisältömuotoa:

```
public void asetaNimiJaJasenmaara() {
    labelNimi.setContentMode(ContentMode.HTML);
    labelJasenia.setContentMode(ContentMode.HTML);
    labelNimi.setValue("<h1>" + kerho.getNimi() + "</h1>");
    labelJasenia.setValue("<h2>Jäseniä: " + Integer.toString(kerho.
       getJasenia()) + "</h2>");
}
```

Luodaan hakukenttävalintakomponentin (`selectHaku`) alkiot. Metodi `setNullSelectionAllowed(false)` poistaa valintalistan alusta tyhjän rivin, `setItemCaption(i, apuJasen.getKysymys(i))` asettaa valintalistaan näkyväksi tekstiksi kentän kysymyksen, kun varsinainen arvo on kentän numero ja `setValue(apuJasen.ekaKentta())` asettaa valituksi ensimmäisen kentän:

```
public void luoHakukentat() {
    selectHaku.setNullSelectionAllowed(false);
    for (int i = apuJasen.ekaKentta(); i < apuJasen.getKenttia(); i
       ++) {
       selectHaku.addItem(i);
       selectHaku.setItemCaption(i, apuJasen.getKysymys(i));
    }
    selectHaku.setValue(apuJasen.ekaKentta());
}
```

Lisätään napeille kuuntelijat. Painettu nappi saadaan parametrina vastaanotetun tapahtuman metodilla `event.getButton()`:

```
@Override
public void buttonClick(ClickEvent event) {
   // Hakunappi
   if (event.getButton() == buttonHaku) {
      int hakukentta = Integer.parseInt(selectHaku.getValue().
         toString());
      String hakuehto = textHaku.getValue();
      haeJasenet(hakuehto, hakukentta);
   }
   // Jäsenenlisäysnappi
   else if (event.getButton() == buttonLisaaJasen) {
      avaaUusiJasenIkkuna();
   }
   // Jäsenenpoistonappi
   else if (event.getButton() == buttonPoista) {
      poistaJasen();
   }
}
```

Lisätään jäsenvalintakomponentin valinnanvaihtumiselle ja uriFragmentin vaihtumiselle kuuntelijat. Kun uriFragment vaihtuu, valitaan jäsenvalintakomponentista kyseisen id:n omaava jäsen ja näytetään jäsenen tiedot. Jäsenvalinnan vaihtuessa näytetään valitun jäsenen tiedot ja päivitetään uriFragmentti:

```
public void lisaaKuuntelijat() {
   Page.getCurrent().addUriFragmentChangedListener(
         new UriFragmentChangedListener() {
            public void uriFragmentChanged(
                  UriFragmentChangedEvent source) {
               int id = Integer.parseInt(source.getUriFragment());
               listJasenet.setValue(id);
               naytaJasenTiedot(id);
            }
         });


   listJasenet.addValueChangeListener(new ValueChangeListener() {
      private static final long serialVersionUID = 1L;
      @Override
      public void valueChange(ValueChangeEvent event) {
         if (listJasenet.getValue() != null) {
            Page.getCurrent().setUriFragment(listJasenet.getValue()
               .toString());
            naytaJasenTiedot(Integer.parseInt(listJasenet.getValue
               ().toString()));
         }
      }
   });
}
```

Haetaan kerhon jäsenet ja päivitetään jäsenlista. Otetaan hakuehto ja hakukenttä

sessiosta, jos ne on asetettu. Jos jäsenen id saadaan uriFragmentista, niin valitaan listasta se jäsen. Muuten käytetään parametria `id`, paitsi jos se on `null` niin valitaan listasta ensimmäinen jäsen.

```
public void haeJasenetJaPaivitaLista(String id) {
   String haku = "";
   int kentta = 0;
   String uriFragment = Page.getCurrent().getUriFragment();

   if (VaadinSession.getCurrent().getAttribute("hakuehto") != null)
       {
      haku = VaadinSession.getCurrent().getAttribute("hakuehto").
         toString();
      kentta = Integer.parseInt( VaadinSession.getCurrent().
         getAttribute("hakukentta").toString() );
      jasenLista = (List<Jasen>) kerho.etsi("*" + haku + "*",
         kentta);
      textHaku.setValue(haku);
      selectHaku.select(kentta);
   }
   else jasenLista = (List<Jasen>) kerho.etsi("*", apuJasen.
      ekaKentta());

   if (jasenLista.size() > 0) {
      if (uriFragment != null) {
         try {
            int jasenId = Integer.parseInt(uriFragment);
            paivitaJasenlista(jasenId);
         } catch (NumberFormatException e) {
            paivitaJasenlista(jasenLista.get(0).getTunnusnro());
         }
      }
      else if (id != null) {
         try {
            int jasenId = Integer.parseInt(id);
            paivitaJasenlista(jasenId);
         } catch (NumberFormatException e) {
            paivitaJasenlista(jasenLista.get(0).getTunnusnro());
         }
      }
      else paivitaJasenlista(jasenLista.get(0).getTunnusnro());
   }
   else paivitaJasenlista(0);
}
```

Kun jäsenlista muuttuu, päivitetään jäsenvalintakomponentin alkiot seuraavan metodin avulla. Parametrina vastaanotetaan listasta valittavan jäsenen id. Animoidaan valintakomponentti sekä jäsen- ja harrastustiedot kun lista muuttuu. Animaatiot tehdään *Animator*-lisäosan [67] avulla:

```
public void paivitaJasenlista(int jasenId) {
   listJasenet.removeAllItems();

   if (jasenLista.size() > 0) {
      for (Jasen jasen : jasenLista) {
         listJasenet.addItem(jasen.getTunnusnro());
```

```
            listJasenet.setItemCaption(jasen.getTunnusnro(), jasen.
                getNimi());
        }
        listJasenet.setValue(jasenId);
    }
    else {
        jasenLayout.removeAllComponents();
        harrastusLayout.removeAllComponents();
        Label labelHuomautus = new Label("<b>Ei löytynyt yhtään jä
            sentä!</b>");
        labelHuomautus.setContentMode(ContentMode.HTML);
        jasenLayout.addComponent(labelHuomautus);
    }

    proxy.animate(listJasenet, AnimType.FADE_IN).setDuration(500).
        setDelay(350);
    proxy.animate(jasenLayout, AnimType.FADE_IN).setDuration(500).
        setDelay(350);
    proxy.animate(harrastusLayout, AnimType.FADE_IN).setDuration
        (500).setDelay(350);
}
```

Näytetään valitun jäsenen tiedot. Jäsenen tiedot näytetään `GridLayout`-ulkoasukomponentissa, jolloin ne saadaan nätisti sarakkeisiin:

```
public void naytaJasenTiedot(int id) {
    jasenLayout.removeAllComponents();
    Jasen jasen = kerho.annaJasenId(id);

    Label labelJasenOtsikko = new Label("Jäsen: " + jasen.getNimi())
        ;
    labelJasenOtsikko.setStyleName("labelJasenOtsikko");
    jasenLayout.addComponent(labelJasenOtsikko);

    Link linkJasen = new Link("[Muokkaa]",
          new ExternalResource("/vaadinKerho/jasensivu/?id=" + id));
    jasenLayout.addComponent(linkJasen);

    GridLayout tietoLayout = new GridLayout(2, apuJasen.getKenttia()
        );
    jasenLayout.addComponent(tietoLayout);

    for (int i = apuJasen.ekaKentta(); i < apuJasen.getKenttia(); i
        ++) {
        Label labelKysymys = new Label("<b>" + jasen.getKysymys(i) +
            ":</b> ");
        Label labelArvo = new Label(jasen.anna(i));
        labelKysymys.setContentMode(ContentMode.HTML);

        tietoLayout.addComponent(labelKysymys, 0, i);
        tietoLayout.addComponent(labelArvo, 1, i);
    }
    naytaHarrastukset(jasen);
}
```

Valitun jäsenen harrastukset näytetään taulukossa. Poistetaan aina ensin vanha taulukko ja luodaan sitten uusi:

```
public void naytaHarrastukset(Jasen jasen) {
   if (harrastusLayout.getComponentIndex(tableHarrastukset) != -1)
      {
      harrastusLayout.removeComponent(tableHarrastukset);
   }

   harrastusLista = kerho.annaHarrastukset(jasen);
   tauluObjektiLista = new ArrayList<Object>();

   tableHarrastukset = new Table();
   tableHarrastukset.setPageLength(harrastusLista.size());
   harrastusLayout.addComponent(tableHarrastukset);

   luoTaulukkoOtsikot();

   if (harrastusLista.size() > 0) luoTaulukkoTietorivit();
}
```

Luodaan otsikkorivi taulukolle. Otsikot luodaan `addContainerProperty`-metodilla, jolle annetaan otsikon nimi ja kyseisen sarakkeen sisältämien arvojen tyyppi. Kun asetetaan lukumuodossa olevat arvot `Integer`-muodossa, saadaan niiden lajittelu toimimaan oikein (Vaadin siis luo automaattisesti lajittelun sarakkeittain otsikosta painamalla):

```
public void luoTaulukkoOtsikot() {
   // Määritellään taulukon sarakkeet ja annetaan niille otsikot.
   if (harrastusLista.size() > 0) {
      for (int i = apuHarrastus.ekaKentta(); i < apuHarrastus.
         getKenttia(); i++) {
         if (apuHarrastus.getKysymys(i).equals("h/vko") ||
            apuHarrastus.getKysymys(i).equals("aloitusvuosi")) {
            tableHarrastukset.addContainerProperty(apuHarrastus.
               getKysymys(i), Integer.class,  null);
         }
         else {
            tableHarrastukset.addContainerProperty(apuHarrastus.
               getKysymys(i), String.class,  null);
         }
      }
   }
   else {
      tableHarrastukset.addContainerProperty("Ei harrastuksia!",
         FormLayout.class,  null);
   }
}
```

Luodaan taulukkoon varsinaiset harrastustietorivit. Rivit luodaan oliolistana (`List<Object>`), johon rivin arvot laitetaan. Jos arvo on numeromuotoista, lisätään se listaan `Integer`-oliona, muuten `String`-muodossa. Kun rivi lisätään taulukkoon, annetaan sille myös rivinumero:

```
public void luoTaulukkoTietorivit() {
   int riviNro = 1;

   // Lisätään harrastukset taulukkoon.
   for (Harrastus harrastus : harrastusLista) {
      tauluObjektiLista.clear();
      for (int i = harrastus.ekaKentta(); i < harrastus.getKenttia
          (); i++) {
         if (apuHarrastus.getKysymys(i).equals("h/vko") ||
             apuHarrastus.getKysymys(i).equals("aloitusvuosi")) {
            tauluObjektiLista.add(new Integer(Integer.parseInt(
               harrastus.anna(i))));
         }
         else {
            tauluObjektiLista.add(harrastus.anna(i).toString());
                           }
      }
      tableHarrastukset.addItem(tauluObjektiLista.toArray(), new
          Integer(riviNro));
      riviNro++;
   }
}
```

Haetaan jäsenet hakuehdon ja hakukentän mukaan. Hakuun lisätään viive jonka aikana näytetään odotusilmaisin. Odotusilmaisin päivitetään käyttöliittymään *Vaadin Push* -kirjaston [34, luku 11.16] avulla. Odotusilmaisin poistetaan sivulta haun jälkeen ja poisto päivittyy sivulle automaattisesti Ajax-kutsun suorituksen jälkeen. Päivitetään jäsenlista uudelleen sivulle ja asetetaan sessioon *hakuehto*- ja *hakukentta*-muuttujat:

```
public void haeJasenet(String hakuehto, int hakukentta) {
   Label labelOdota = new Label("Jäseniä haetaan...");
   hakuLayout.addComponent(labelOdota);
   hakuLayout.addComponent(odotusIlmaisin);
   UI.getCurrent().push();

   try {
      Thread.sleep(3000);
   } catch(InterruptedException ex) {
      Thread.currentThread().interrupt();
   }

   jasenLista = (List<Jasen>) kerho.etsi("*" + hakuehto + "*",
       hakukentta);

   hakuLayout.removeComponent(labelOdota);
   hakuLayout.removeComponent(odotusIlmaisin);

   if (jasenLista.size() > 0) {
      paivitaJasenlista(jasenLista.get(0).getTunnusnro());
   }
   else paivitaJasenlista(0);

   VaadinSession.getCurrent().setAttribute("hakuehto", hakuehto);
   VaadinSession.getCurrent().setAttribute("hakukentta", Integer.
       toString(hakukentta));
}
```

Seuraava metodi poistaa valitun jäsenen kerhosta. Poisto varmistetaan *ConfirmDialog*-lisäosan [68] avulla: Jos poisto hyväksytään (`dialog.isConfirmed()`), niin poistetaan jäsen ja päivitetään jäsenvalintalista, muuten ei tehdä mitään :

```
public void poistaJasen() {
   final int jasenId = Integer.parseInt(listJasenet.getValue().
      toString());
   final Jasen jasen = kerho.annaJasenId(jasenId);

   ConfirmDialog.show(UI.getCurrent(), "Jäsenen poisto", "
      Poistetaanko jäsen: " + jasen.getNimi() + "?",
         "Kyllä", "Ei", new ConfirmDialog.Listener() {

      public void onClose(ConfirmDialog dialog) {
         if (dialog.isConfirmed()) {
            kerho.poista(jasenId);
            try {
               kerho.talleta();
            } catch (SailoException e) {
               e.printStackTrace();
            }
            jasenLista.remove(jasen);

            labelJasenia.setValue("<h2>Jäseniä: " + Integer.
               toString(kerho.getJasenia()) + "</h2>");

            if (jasenLista.size() > 0) {
               paivitaJasenlista(jasenLista.get(0).getTunnusnro());
            }
            else paivitaJasenlista(0);
         }
      }
   });
}
```

Jäsenenlisäysdialogi luodaan ja avataan seuraavalla metodilla. Jäsenenlisäyslomake luodaan `JasenTietoLomake`-komponenttina:

```
public void avaaUusiJasenIkkuna() {
   dialogiLayout = new VerticalLayout();

   apuJasen = new Jasen();
   uusiJasenWindow = new Window("Luo uusi jäsen");
   uusiJasenWindow.setWidth("400px");
   uusiJasenWindow.setHeight("100\%");
   uusiJasenWindow.setModal(true);

   uusiJasenWindow.setContent(dialogiLayout);

   tietoLomake = new JasenTietoLomake(apuJasen);
   dialogiLayout.addComponent(tietoLomake);

   labelVirhe = new Label("");
   labelVirhe.setStyleName("labelVirhe");
   labelVirhe.setWidth("100%");
```

```
    dialogiLayout.addComponent(labelVirhe);

    lisaaDialogiNapit();

    uusiJasenWindow.center();
    UI.getCurrent().addWindow(uusiJasenWindow);
}
```

Lisätään dialogiin *Lisää jäsen*- ja *Peruuta*-napit ja niille kuuntelijat. Luodaan napit `HorizontalLayout`-ulkoasukomponentin sisään, jolloin ne saadaan vierekkäin:

```
public void lisaaDialogiNapit() {
    Button buttonLisaa = new Button("Lisää jäsen");
    buttonLisaa.addClickListener(new Button.ClickListener() {
        public void buttonClick(ClickEvent event) {
            tallennaUusiJasen();
        }
    });

    Button buttonPeruuta = new Button("Peruuta");
    buttonPeruuta.addClickListener(new Button.ClickListener() {
        public void buttonClick(ClickEvent event) {
            uusiJasenWindow.close();
        }
    });

    HorizontalLayout buttonLayout = new HorizontalLayout();
    buttonLayout.setSpacing(true);
    dialogiLayout.addComponent(buttonLayout);
    dialogiLayout.setComponentAlignment(buttonLayout, Alignment.
        MIDDLE_CENTER);

    buttonLayout.addComponent(buttonLisaa);
    buttonLayout.addComponent(buttonPeruuta);
}
```

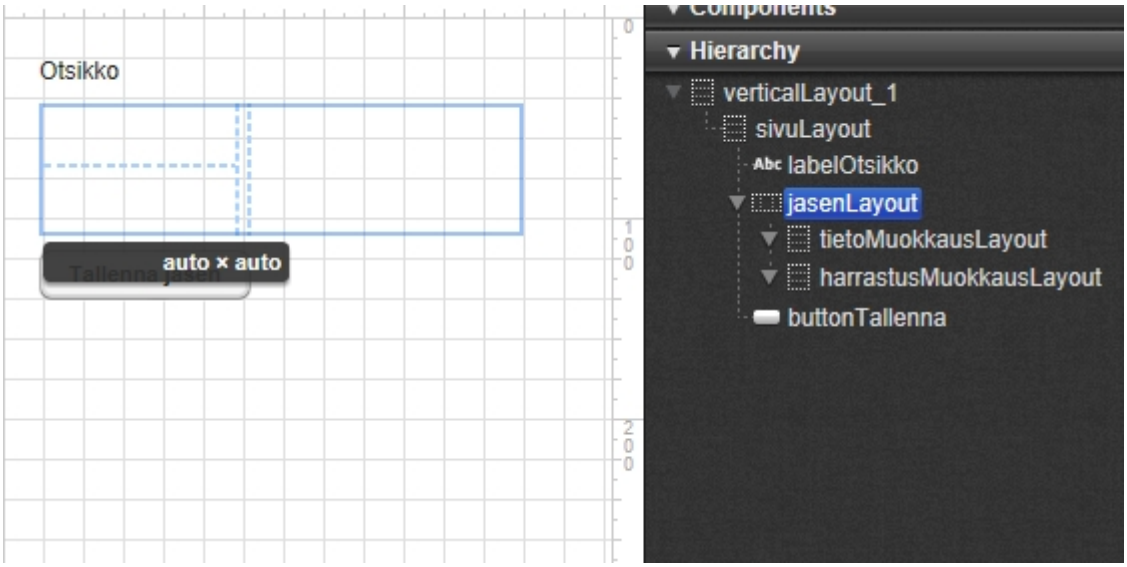
Uusi jäsen lisätään ja tallennetaan seuraavalla metodilla. `JasenTietoLomake`-komponentilta otetaan lisättävä jäsen ja tarkistetaan ettei kentissä ole virheitä eikä nimikenttä ole tyhjä. Suljetaan dialogi ja päivitetään jäsenvalintalista, kun lisäys onnistuu. Jos kentissä on virheitä näytetään ilmoitus, joka animoidaan näkymään vain vähän aikaa:

```
public void tallennaUusiJasen() {
    if (!tietoLomake.onkoVirheita() && !tietoLomake.nimiTyhja()) {
        apuJasen = tietoLomake.getJasen();
        try {
            apuJasen.rekisteroi();
            kerho.korvaaTaiLisaa(apuJasen);
            kerho.talleta();
        } catch (SailoException e) {
            e.printStackTrace();
        }
        uusiJasenWindow.close();
```

```
            labelJasenia.setValue("<h2>Jäseniä: " + Integer.toString(
                kerho.getJasenia()) + "</h2>");
            jasenLista = (List<Jasen>) kerho.etsi("*", 1);
            paivitaJasenlista(apuJasen.getTunnusnro());
        }
        else if (tietoLomake.nimiTyhja()) {
            labelVirhe.setValue("Nimi ei saa olla tyhjä!");
            proxy.animate(labelVirhe, AnimType.FADE_IN).setDuration(4000)
                .setDelay(0);
            proxy.animate(labelVirhe, AnimType.FADE_OUT).setDuration
                (2000).setDelay(0);
        }
        else {
            labelVirhe.setValue("Virheellisiä syötteitä!");
            proxy.animate(labelVirhe, AnimType.FADE_IN).setDuration(4000)
                .setDelay(0);
            proxy.animate(labelVirhe, AnimType.FADE_OUT).setDuration
                (2000).setDelay(0);
        }
    }
```

A.5 Jasensivu.java

Vaadin composite -komponentti, jossa luodaan jäsensivun sisältö. Sivun perusulkoasu luotiin visuaalisella Vaadin editorilla (kuvio 26).



Kuvio 26: Vaadin-jäsensivun visuaalinen editori -näkymä

Konstruktorissa luodaan sivun sisältö. Luodaan uusi `JasenTietoLomake`-komponentti ja lisätään se sivulle sekä luodaan harrastustenmuokkaustaulukko:

```
public Jasensivu(String id) {
```

```
    buildMainLayout();
    setCompositionRoot(mainLayout);

    proxy = new AnimatorProxy();
    mainLayout.addComponent(proxy);

    virheita = 0;
    kerho = KerhoBean.getKerho();

    try {
       kerho.lueTiedostosta("kelmit");
    } catch (SailoException e) {
       e.printStackTrace();
    }

    jasenId = Integer.parseInt(id);
    jasen = kerho.annaJasenId(jasenId);
    apuHarrastus = new Harrastus();

    buttonTallenna.addClickListener(this);
    labelOtsikko.setContentMode(ContentMode.HTML);
    labelOtsikko.setValue("<h1>" + jasen.getNimi() + "</h1>");

    Link linkKerho = new Link("<- Takaisin kerhoon",
          new ExternalResource("/vaadinKerho/kerhosivu/?id=" + id));
    sivuLayout.addComponent(linkKerho);

    tietoLomake = new JasenTietoLomake(jasen);
    tietoMuokkausLayout.addComponent(tietoLomake);
    luoHarrastusMuokkausAlue();
}
```

Luodaan harrastustenmuokkaustaulukko ja napit harrastusten lisäykselle ja poistolle. Animoidaan taulukko kun se luodaan:

```
public void luoHarrastusMuokkausAlue() {
    harrastusLista = kerho.annaHarrastukset(jasen);
    luoHarrastusMuokkausTaulukko();
    luoHarrastusMuokkausNapit();

    proxy.animate(tableHarrastukset, AnimType.FADE_IN).setDuration
        (500).setDelay(200);
}
```

Harrastustaulukko ja sen otsikot luodaan samaan tapaan kuin kerhosivulla, joten sitä ei tässä käydä läpi muuten kuin varsinaisten tekstikenttien lisäyksen osalta. Kentät lisätään `FormLayout`-ulkoasukomponentin sisään niin saadaan tekstikentän virhe näkyviin `setComponentError()`-metodilla (sarakkeiden sisällöksi on otsikkorivillä asetettu `FormLayout.class`):

```
public void luoTaulukkoKentat() {
    int riviNro = 1;

    // Lisätään harrastukset taulukkoon.
```

```
    for (Harrastus harrastus : harrastusLista) {
        tauluObjektiLista.clear();
        for (int i = harrastus.ekaKentta(); i < harrastus.getKenttia
            (); i++) {
            // Lisätään FormLayout, jonka sisään tulee tekstikenttä.
            // Näin saadaan myös virhe näkyviin.
            FormLayout harrastusKenttaLomake = new FormLayout();
            harrastusKenttaLomake.setSpacing(false);
            harrastusKenttaLomake.setMargin(false);
            TextField fieldHarrastus = luoKentta(harrastus, i);

            harrastusKenttaLomake.addComponent(fieldHarrastus);
            tauluObjektiLista.add(harrastusKenttaLomake);
        }
        tableHarrastukset.addItem(tauluObjektiLista.toArray(), new
            Integer(riviNro));
        riviNro++;
    }
}
```

Tekstikenttä luodaan `luoKentta(harrastus, i)`-metodilla. Kentän luonti tapahtuu samaan tapaan kuin `JasenTietoLomake`-komponentin kentän, kuten myös tekstinvaihtumiskuuntelijan luonti ja kentän virheen näyttäminen, joten niitä ei tässä käydä enää läpi. Kentälle lisätään myös `FocusListener`, joka kuuntelee kentän aktivoitumista. Tällöin asetetaan kyseinen harrastus valituksi harrastuksen poistoa varten:

```
fieldHarrastus.addFocusListener(new FocusListener() {
    @Override
    public void focus(FocusEvent event) {
        TextField field = (TextField) event.getComponent();
        String[] osat = field.getId().split(",");

        for (Harrastus harrastus : harrastusLista) {
            if (harrastus.getTunnusnro() == Integer.parseInt(osat[0]))
                {
                valittuHarrastus = harrastus;
                break;
            }
        }
    }
});
```

Lisätään napit harrastusten lisäystä ja poistoa varten. Poistamisnappi luodaan vain, jos taulukossa on harrastuksia:

```
public void luoHarrastusMuokkausNapit() {
    HorizontalLayout harrastusNappiLayout = new HorizontalLayout();
    harrastusNappiLayout.setMargin(true);
    harrastusNappiLayout.setSpacing(true);
    harrastusMuokkausLayout.addComponent(harrastusNappiLayout);
```

```
    Button buttonLisaaHarrastus = new Button("Lisää uusi harrastus")
        ;
    harrastusNappiLayout.addComponent(buttonLisaaHarrastus);

    buttonLisaaHarrastus.addClickListener(new Button.ClickListener()
         {
       public void buttonClick(ClickEvent event) {
          lisaaHarrastus();
       }
    });

    if (harrastusLista.size() > 0) {
       Button buttonPoistaHarrastus = new Button("Poista harrastus")
           ;
       harrastusNappiLayout.addComponent(buttonPoistaHarrastus);

       buttonPoistaHarrastus.addClickListener(new Button.
           ClickListener() {
          public void buttonClick(ClickEvent event) {
             poistaHarrastus();
          }
       });
    }
}
```

`lisaaHarrastus()`-metodi avaa dialogin uuden harrastuksen luomista varten. Dialogi luodaan samaan tapaan kuin kerhosivulla jäsenenlisäysdialogi, paitsi sen sisältö luodaan itse ilman omaa komponenttia.

`poistaHarrastus()`-metodi poistaa valitun harrastuksen ja varmistaa poiston käyttäjältä. Poisto varmistetaan *ConfirmDialog*-lisäosan avulla samoin kuin kerhosivulla jäsenen poisto. Jos mitään harrastusta ei ole valittuna niin näytetään ilmoitus *Notification*-komponentilla.

Harrastuksen lisäyksen ja poiston jälkeen harrastustaulukko ja napit luodaan uudelleen ensin poistamalla vanhat (`harrastusMuokkausLayout.removeAllComponents()`) ja lisäämällä sitten uudet (`luoHarrastusMuokkausAlue()`), jolloin harrastustaulukko animoidaan.

Seuraava metodi tallentaa jäsenen tiedot (ja harrastukset) ja näyttää ilmoituksen tallennuksen onnistumisesta. Ilmoitus näytetään *Notification*-komponentilla, jonka tyyppi asetetaan sen mukaan tallennettiinko jäsen (joko *HUMANIZED_MESSAGE* tai *WARNING_MESSAGE*):

```
public void tallennaJasen() {
    if (virheita == 0 && !tietoLomake.onkoVirheita()) {
       jasen = tietoLomake.getJasen();
```

```
      try {
         kerho.korvaaTaiLisaa(jasen);
         kerho.talleta();
      } catch (SailoException e) {
         e.printStackTrace();
      }
      Notification onnistumisIlmoitus = new Notification("Jäsen
         tallennettu","",
            Notification.Type.HUMANIZED_MESSAGE);
      onnistumisIlmoitus.setDelayMsec(3000);
      onnistumisIlmoitus.setPosition(Position.MIDDLE_LEFT);
      onnistumisIlmoitus.show(Page.getCurrent());
   }
   else {
      Notification virheIlmoitus= new Notification("Virheellisiä sy
         ötteitä","Muutoksia ei tallennettu",
            Notification.Type.WARNING_MESSAGE);
      virheIlmoitus.setDelayMsec(3000);
      virheIlmoitus.setPosition(Position.MIDDLE_LEFT);
      virheIlmoitus.show(Page.getCurrent());
   }
}
```

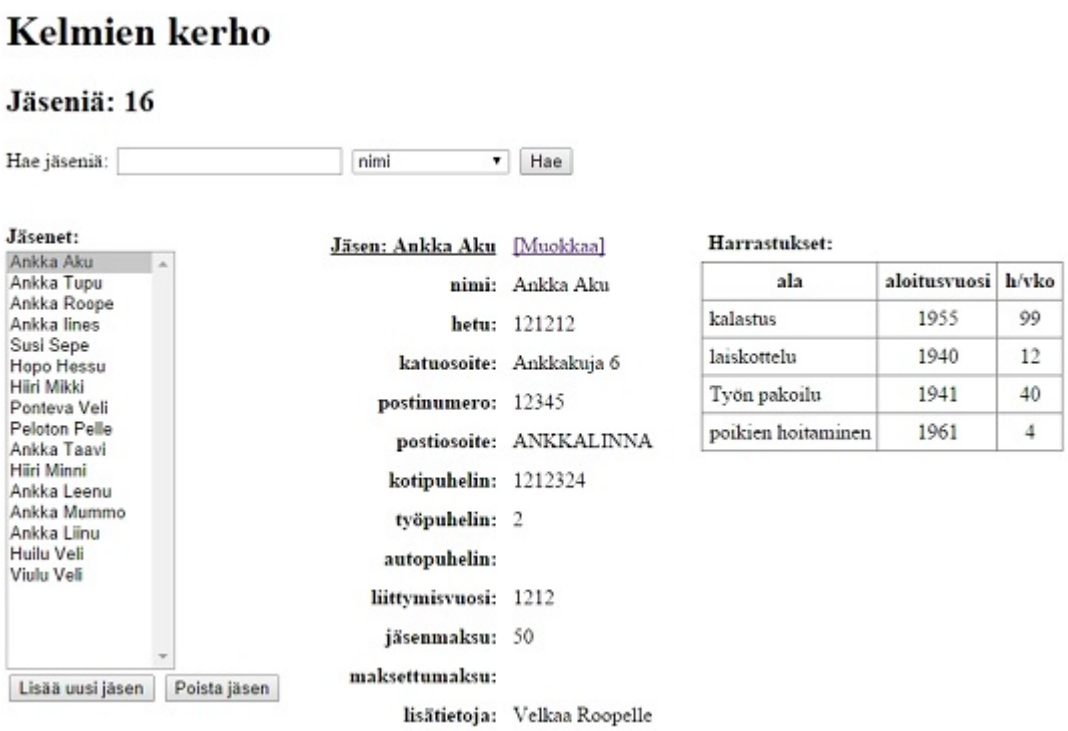
B ASP.NET-kerhosovelluksen luominen

ASP.NET-kerhosovellus koostuu kahdesta WWW-lomakesivusta (`kerho` ja `jasensivu`) ja koostetusta komponentista (*Web User Control*) `JasenTietoLomake`. Kuvio 27 näyttää sovelluksen kerhosivun käyttöliittymän ja kuvio 28 esittää sovelluksen kansiorakenteen.

B.1 JasenTietoLomake

JasenTietoLomake on ASP.NET *Web User Control* eli koostettu komponentti. **JasenTietoLomake.ascx**-tiedostossa luodaan komponentin ulkoasu ja Ajax-toiminnallisuus `UpdatePanel`-kontrollin avulla. Luodaan `Table`-kontrolli, johon lomake luodaan koodin puolella:

```
<asp:UpdatePanel ID="updatePanelJasenlomake" UpdateMode="
   Conditional" runat="server">
   <ContentTemplate>
      <asp:Table ID="tableJasenLomake" runat="server" BorderStyle="
         None" Caption="" CaptionAlign="Top"
         CellPadding="0" CssClass="tableJasentietoLomake">
      </asp:Table>
   </ContentTemplate>
</asp:UpdatePanel>
```



Kuvio 27: ASP.NET-kerhosovelluksen kerhosivun käyttöliittymä

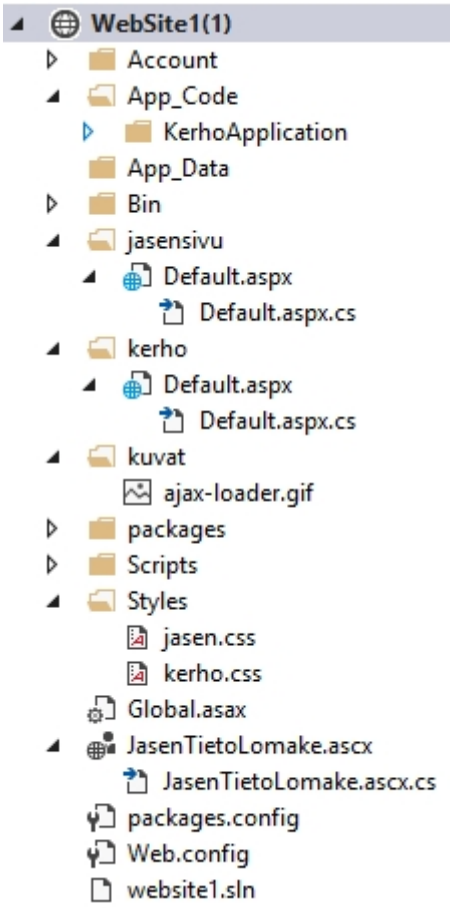
JasenTietoLomake.ascx.cs-tiedostossa luodaan komponentin toiminnallisuus ja dynaamisesti luotavat komponentin osat.

Komponentille voi asettaa jäsenen ja siltä voi pyytää jäsenen ja tiedon onko nimikenttä tyhjä:

```
public Jasen Jasen
{
   get { return jasen; }
   set
   {
      jasen = value;
   }
}

public Boolean Nimityhja
{
   get { return nimiKentta.Text.Equals(""); }
}
```

`Page_Init`-metodissa luodaan vain uusi jäsen. `Page_Load`-metodissa luodaan jäsenenmuokkauslomake. Jokaiselle jäsenen kentälle luodaan taulukkoon oma rivi:

```
protected void Page_Load(object sender, EventArgs e)
{
```



Kuvio 28: ASP.NET-Kerhon kansiorakenne

```
    luoJasentietoLomake();
}

public void luoJasentietoLomake()
{
    for (int i = jasen.ekaKentta(); i < jasen.getKenttia(); i++)
    {
        luoLomakerivi(i);
    }
}
```

Luodaan taulukkoon yksi rivi (taulukko, jonka id on *tableJasenLomake* luotiin siis `.ascx`-tiedostossa). Luodaan riville solut kysymykselle, tekstikentälle ja virheilmoitukselle:

```
public void luoLomakerivi(int i)
{
    TableRow tietoRivi = new TableRow();
    tableJasenLomake.Rows.Add(tietoRivi);

    TableCell kysymysSolu = luoKysymysSolu(i);
    TableCell tietoSolu = new TableCell();
    TableCell virheSolu = luoVirheSolu(i);
    TextBox textTieto = luoTietoKentta(i);

    tietoSolu.Controls.Add(textTieto);
    tietoRivi.Cells.Add(kysymysSolu);
    tietoRivi.Cells.Add(tietoSolu);
    tietoRivi.Cells.Add(virheSolu);

    luoJasentietoTriggeri(i);
}
```

Luodaan taulukkoon solu jäsenen kentän kysymykselle:

```
public TableCell luoKysymysSolu(int i)
{
    TableCell kysymysSolu = new TableCell();
    kysymysSolu.Text = jasen.getKysymys(i) + ":";
    kysymysSolu.Style.Add("text-align", "right");
    return kysymysSolu;
}
```

Luodaan taulukkoon tekstikenttä jäsenen kentälle. Annetaan sille id:ksi kentän numero ja arvoksi kentän arvo. Attribuutti `AutoPostBack = true` asettaa kentän välittämään tapahtuman itse aina tekstin vaihtuessa. Lisätään kentälle myös tapahtumankäsittelijä (`new EventHandler`) tekstinvaihtumiselle (`TextChanged`):

```
public TextBox luoTietoKentta(int i)
{
    TextBox textTieto = new TextBox();
```

```
    textTieto.ID = i.ToString();
    textTieto.AutoPostBack = true;
    textTieto.TextChanged += new EventHandler(textTieto_TextChanged)
        ;
    textTieto.Text = jasen.anna(i);
    textTieto.Width = (Unit.Pixel(150));

    if (i == jasen.ekaKentta()) nimiKentta = textTieto;
    return textTieto;
}
```

Luodaan taulukkoon virhesolu jäsenen kentälle ja annetaan sille yksilöllinen id:

```
public TableCell luoVirheSolu(int i)
{
    TableCell virheSolu = new TableCell();
    virheSolu.ID = "virhe" + i.ToString();
    virheSolu.CssClass = "virheTeksti";
    return virheSolu;
}
```

Luodaan `UpdatePanel`-kontrolliin triggeri tekstikentän (`trig.ControlID`) tekstinvaihtumiselle (`trig.EventName`). `updatePanelJasenlomake` luotiin `.ascx`-tiedostossa:

```
public void luoJasentietoTriggeri(int i)
{
    AsyncPostBackTrigger trig = new AsyncPostBackTrigger();
    trig.ControlID = i.ToString();
    trig.EventName = "TextChanged";
    updatePanelJasenlomake.Triggers.Add(trig);
}
```

Luodaan tekstikenttien tekstinvaihtumiskuuntelija. Tapahtuman aiheuttaja saadaan `sender`-olion kautta:

```
protected void textTieto_TextChanged(object sender, EventArgs e)
{
    TextBox box = (TextBox)sender;
    asetaJaTarkistaTieto(box);
}
```

Asetetaan tekstikentän arvo jäsenelle ja tarkistetaan ja näytetään mahdollinen virhe. Muutetaan myös tekstikentän tyyli virheen mukaan. Solu, jossa virhe pitää näyttää, saadaan `FindControl`-metodilla, jolle annetaan parametrina haettavan elementin id:

```
public void asetaJaTarkistaTieto(TextBox box)
{
   int kentta = Int32.Parse(box.ID);
   String virhe = jasen.aseta(kentta, box.Text);

   if (virhe != null && box.CssClass.Equals(""))
   {
      box.CssClass = "virheTieto";
   }

   else if (virhe == null && box.CssClass.Equals("virheTieto"))
   {
      box.CssClass = "";
   }

   TableCell cell = (TableCell)FindControl("virhe" + box.ID);
   if (virhe != null) cell.Text = virhe;
   else cell.Text = "";
}
```

Asetetaan jäsenen kenttiin arvot lomakkeen tekstikentistä ja tarkistetaan niiden virheet. Tämä pitää tehdä näin, koska virhetietoja ei voida säilyttää attribuutissa, sillä ne nollautuvat takaisinpyynnöillä. Haetaan jäsenen kenttää vastaava tekstikenttä `FindControl`-metodilla:

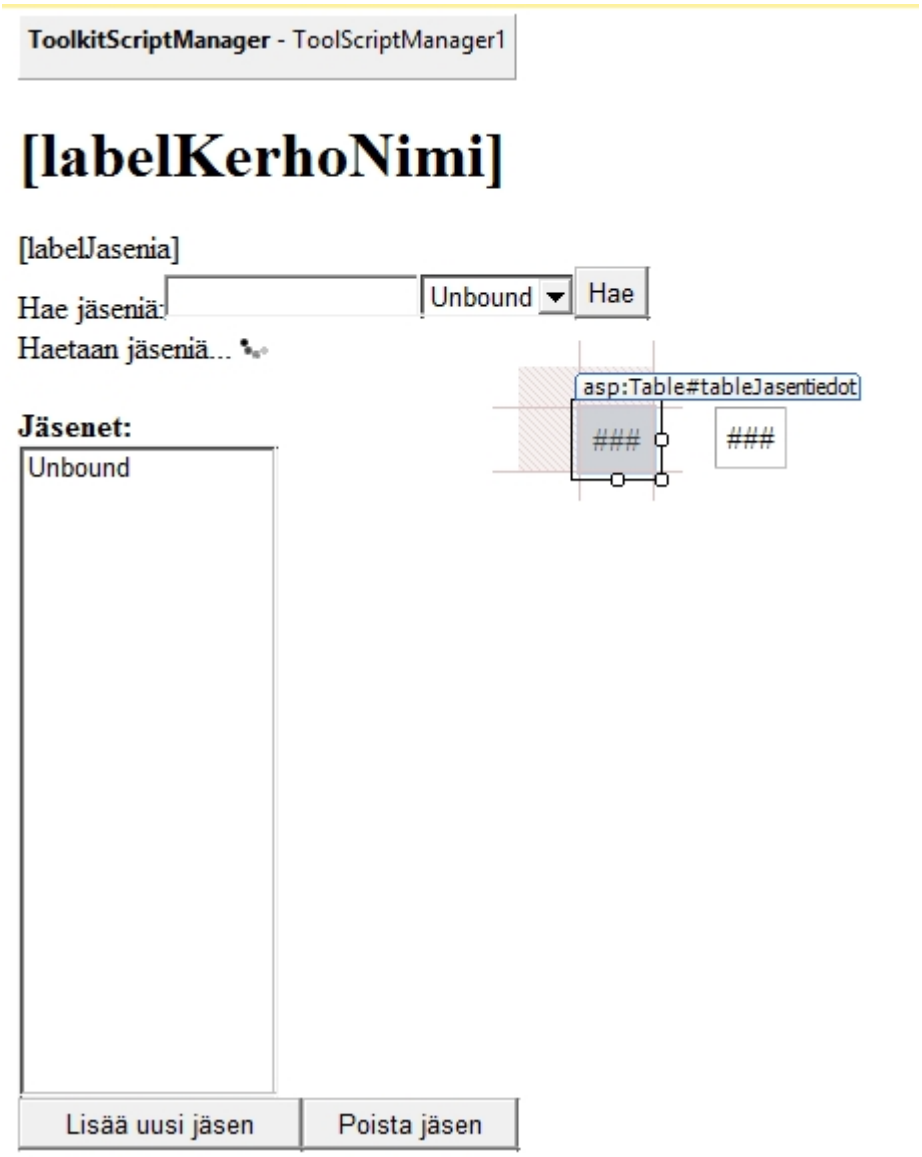
```
public int tarkistaJasenKentat()
{
   int tallennusVirheita = 0;

   for (int i = jasen.ekaKentta(); i < jasen.getKenttia(); i++)
   {
      TextBox box = (TextBox)FindControl(i.ToString());
      String virhe = jasen.aseta(i, box.Text);

      if (virhe != null) tallennusVirheita++;
   }

   return tallennusVirheita;
}
```

Päivitetään lomake (poistetaan vanha ja luodaan uusi). Tätä tarvitaan, jotta lomakkeeseen saadaan muutettua tyhjän jäsenen tiedot:

```
public void paivitaLomake()
{
   tableJasenLomake.Controls.Clear();
   luoJasentietoLomake();
   updatePanelJasenlomake.Update();
}
```

B.2 Kerhosivu

Kerhosivu on ASP.NET WWW-lomakesivu, joka koostuu **Default.aspx**-ulkoasutiedostosta ja **Default.aspx.cs**-kooditiedostosta. Kuvio 29 esittää Visual Studion visuaalisen editorin näkymän kerhosivusta (*Default.aspx*).



Kuvio 29: ASP.NET-Kerhosivun Visual Studion editorinäkymä

Default.aspx-tiedostossa luodaan sivun perusulkoasu ja Ajax-toiminnallisuus. Myös animaatiot luodaan tässä tiedostossa. Kontrollit voi lisätä sivulle myös visuaalisella editorilla, mutta sillä ei voi luoda HTML-elementtejä (paitsi `div`-elementin).

Rekisteröidään *Ajax Control Toolkit*-lisäosa ja oma `JasenTietoLomake`-komponentti sivulle. Nämä voi myös luoda visuaalisen editorin kautta:

```
<%@ Register TagPrefix="ajaxToolkit" Namespace="AjaxControlToolkit"
    Assembly="AjaxControlToolkit" %>
<%@ Register Src="../JasenTietoLomake.ascx" TagName="
   JasenTietoLomake" TagPrefix="uc1" %>
```

Määritellään sivun tyylitiedosto ja selaimessa näkyvä otsikko:

```
<head id="Head1" runat="server">
   <link href="../Styles/kerho.css" rel="stylesheet" type="text/css
       " />
   <title>Kerho</title>
</head>
```

Kaikki sivun sisältö tulee `body`-elementtiin lisätyn `form`-elementin sisään. Otetaan `ToolkitScriptManager`-kontrolli käyttöön ja asetetaan sille historiatiedot käyttöön (`EnableHistory="true"`). Historiatiedot asetetaan näkymään osoitteessa selkeäkielisenä (`EnableSecureHistoryState="false"`). Asetetaan myös kuuntelija navigoinnille (`OnNavigate`), joka kuuntelee selaimen takaisin- tai eteenpäin-nappien painalluksia:

```
<form id="form1" runat="server">
   <ajaxToolkit:ToolkitScriptManager ID="ToolScriptManager1" runat=
       "Server" EnableHistory="true"
      EnableSecureHistoryState="false" OnNavigate="
         ToolScriptManager1_Navigate" />
```

Luodaan sivun otsikoihin kerhon nimi ja kerhon jäsenmäärä. Jäsenmäärä laitetaan `UpdatePanel`-kontrollin `ContentTemplate`-elementin sisään, jolloin se voidaan päivittää asynkronisesti. Triggereiksi asetetaan jäsenen lisäys- ja poistonappien `Click`-tapahtumat, jolloin jäsenmäärä päivittyy automaattisesti niitä painettaessa (tietysti uusi sisältö `Label`-kontrollille pitää luoda koodin puolella):

```
<h1>
   <asp:Label runat="server" ID="labelKerhoNimi"></asp:Label>
</h1>
<h2>
   <asp:UpdatePanel ID="updatePanel1" UpdateMode="Conditional"
       runat="server">
      <ContentTemplate>
         <asp:Label runat="server" ID="labelJasenia"></asp:Label>
      </ContentTemplate>
      <Triggers>
```

```
            <asp:AsyncPostBackTrigger ControlID="buttonLisaa"
                EventName="Click" />
            <asp:AsyncPostBackTrigger ControlID="buttonPoista"
                EventName="Click" />
        </Triggers>
    </asp:UpdatePanel>
</h2>
```

Luodaan hakulomake, joka laitetaan myös `UpdatePanel`-kontrollin sisään. Tässä myös napin painallus välitetään asynkronisesti, koska se on myös `UpdatePanel`-kontrollin sisällä, joten erillistä triggeriä ei tarvitse asettaa. Kaikki nämä hakulomakkeen kontrollit ovat `UpdatePanel`-kontrollin sisällä, koska se tekee automaattisesti rivinvaihdon, ja jonkun elementin sijoittaminen sen ulkopuolelle rikkoisi ulkoasun (muutenhan riittäisi kun nappi olisi sen sisällä). Nämä kaikki kontrollit ovat siis ASP.NET-kehyksen WWW-palvelinkontrolleja:

```
<asp:UpdatePanel ID="updatePanelHaku" UpdateMode="Conditional"
    runat="server">
    <ContentTemplate>
        <asp:Label ID="labelHae" runat="server" Text="Hae jäseniä:"
            ></asp:Label>
        <asp:TextBox ID="textHae" runat="server"></asp:TextBox>
        <asp:DropDownList ID="selectKentta" runat="server">
        </asp:DropDownList>
        <asp:Button ID="buttonHae" runat="server" Text="Hae" OnClick=
            "buttonHae_Click" />
    </ContentTemplate>
</asp:UpdatePanel>
```

Luodaan `UpdateProgress`-kontrolli, joka liitetään edellisen `UpdatePanel`-kontrollin päivittymiseen. `ProgressTemplate`-elementtiin annetaan sisältö, joka näytetään päivittymisen aikana. Tässä siihen lisätään teksti ja pyörivä latauskuva:

```
<asp:UpdateProgress ID="UpdateProgressHaku" AssociatedUpdatePanelID
    ="updatePanelHaku"
        runat="server">
    <ProgressTemplate>
        Haetaan jäseniä...
        <asp:Image ID="imageLataus" runat="server" ImageUrl="~/kuvat/
            ajax-loader.gif" />
    </ProgressTemplate>
</asp:UpdateProgress>
```

Luodaan jäsenvalintakomponentti. Sille asetetaan valinnanvaihtumistapahtuma ja asetetaan kontrolli välittämään tapahtuma aina valinnan vaihtuessa (`AutoPostBack` =“`True`”). Triggereiksi asetetaan haku-, poisto- ja lisäysnappien klikkaukset:

```
<asp:UpdatePanel ID="updatePanelListJasenet" UpdateMode="
    Conditional" runat="server">
   <ContentTemplate>
      <asp:ListBox ID="listBoxJasenet" runat="server" Rows="20"
         Width="130px"
         OnSelectedIndexChanged="
            listBoxJasenet_SelectedIndexChanged" AutoPostBack="True
            ">
      </asp:ListBox>
   </ContentTemplate>
   <Triggers>
      <asp:AsyncPostBackTrigger ControlID="buttonHae" EventName="
         Click" />
      <asp:AsyncPostBackTrigger ControlID="buttonPoista" EventName=
         "Click" />
      <asp:AsyncPostBackTrigger ControlID="buttonLisaa" EventName="
         Click" />
   </Triggers>
</asp:UpdatePanel>
```

Luodaan nappi, jolla avataan uuden jäsenen luomisdialogi, sekä jäsenen poistonappi. Luodaan *Ajax Control Toolkit*-lisäosan `ConfirmButtonExtender`-kontrolli, jonka avulla varmistetaan poisto. Sille annetaan parametreina kontrollin id, johon varmistus liitetään, ja varmistuksessa näytettävä teksti:

```
<asp:Button ID="buttonUusiJasen" runat="server" Text="Lisää uusi jä
    sen" />
<asp:Button ID="buttonPoista" runat="server" OnClick="
   buttonPoista_Click" Text="Poista jäsen" />
<ajaxToolkit:ConfirmButtonExtender ID="poistoVarmistus" runat="
    server"
   TargetControlID="buttonPoista"
   ConfirmText="Poistetaanko valittu jäsen?" />
```

Luodaan taulukot jäsenen tiedoille ja harrastuksille. `UpdatePanel`-kontrollin triggereiksi asetetaan jäsenen lisäys-, haku- ja poistonappien klikkaukset sekä jäsenvalintalistan valinnanvaihtuminen. Taulukot luodaan ASP.NET WWW-palvelinkontrolleina:

```
<asp:UpdatePanel ID="UpdatePanelJasentiedot" UpdateMode="
    Conditional" runat="server">
   <ContentTemplate>
      <asp:Table ID="tableJasentiedot" runat="server" BorderStyle="
         None" Caption="" CaptionAlign="Top"
         CellPadding="5" CssClass="tableJasentiedot">
      </asp:Table>
      <asp:Table ID="tableHarrastukset" runat="server" BorderStyle=
         "None" Caption="" CaptionAlign="Top"
         CellPadding="5" CssClass="tableHarrastukset" GridLines="
            Both">
      </asp:Table>
   </ContentTemplate>
   <Triggers>
```

```
      <asp:AsyncPostBackTrigger ControlID="buttonLisaa" EventName="
         Click" />
      <asp:AsyncPostBackTrigger ControlID="buttonHae" EventName="
         Click" />
      <asp:AsyncPostBackTrigger ControlID="buttonPoista" EventName=
         "Click" />
      <asp:AsyncPostBackTrigger ControlID="listBoxJasenet"
         EventName="SelectedIndexChanged" />
   </Triggers>
</asp:UpdatePanel>
```

Luodaan dialogi uuden jäsenen lisäykselle *Ajax Control Toolkit*-lisäosan `ModalPopupExtender`-kontrollin avulla. Dialogi avataan `buttonUusiJasen`-napilla ja dialogin sisältö luodaan `panelUusiJasen`-elementissä. `dialogOtsikko`-elementistä dialogia voi raahata. `BackgroundCssClass`-attribuutissa määritellään dialogin taustalle jäävän sivun tyyli:

```
<ajaxToolkit:ModalPopupExtender ID="uusiJasenDialog" runat="server"
   TargetControlID="buttonUusiJasen"
   PopupControlID="panelUusiJasen"
   PopupDragHandleControlID="dialogOtsikko"
   BackgroundCssClass="modalDialogTausta">
</ajaxToolkit:ModalPopupExtender>
```

Luodaan dialogin sisältö. Lisätään siihen oma `JasenTietoLomake`-komponentti. Luodaan myös paikka virheilmoitukselle sekä dialogin napit. Nappien klikkaukset lisätään virheilmoituksen triggereiksi:

```
<asp:Panel ID="panelUusiJasen" Style="display: none" runat="server"
   >
   <div id="panelUusiJasenSisalto" class="panelUusiJasenSisalto"
      runat="server">
      <div class="dialogOtsikko" id="dialogOtsikko">
         <p>Lisää uusi jäsen</p>
      </div>
      <uc1:JasenTietoLomake ID="JasenTietoLomake1" runat="server"
         />
      <div class="dialogButtons">
         <asp:UpdatePanel ID="updatePanelVirheIlmoitus" UpdateMode=
            "Conditional" runat="server">
            <ContentTemplate>
               <p id="uusiJasenVirheIlmoitus" class="
                  uusiJasenVirheIlmoitus" runat="server"></p>
            </ContentTemplate>
            <Triggers>
               <asp:AsyncPostBackTrigger ControlID="buttonLisaa"
                  EventName="Click" />
               <asp:AsyncPostBackTrigger ControlID="buttonPeruuta"
                  EventName="Click" />
            </Triggers>
         </asp:UpdatePanel>
         <asp:Button ID="buttonLisaa" OnClick="buttonLisaa_Click"
            runat="server" Text="Lisää jäsen" />
```

```
         <asp:Button ID="buttonPeruuta" OnClick="
            buttonPeruuta_Click" runat="server" Text="Peruuta" />
      </div>
   </div>
</asp:Panel>
```

Luodaan animaatio *Ajax Control Toolkit*-lisäosan `UpdatePanelAnimationExtender`-kontrollin avulla. Liitetään se sitä varten luodun tyhjän `UpdatePanel`-kontrollin päivittymiseen (`TargetControlID="updatePanelAnimaatio"`), jolloin animaatio saadaan näytettyä koodin puolelta päivittämällä kyseinen `UpdatePanel`:

```
<ajaxToolkit:UpdatePanelAnimationExtender ID="upae" runat="server"
   TargetControlID="updatePanelAnimaatio">
   <Animations>
      <OnUpdated>
         <Sequence>
            <Parallel duration="1">
               <Color
                  AnimationTarget="listBoxJasenet"
                  Property="style"
                  PropertyKey="backgroundColor"
                  StartValue="#FFFF66"
                  EndValue="#FFFFFF" />
               <Color
                  AnimationTarget="tableJasentiedot"
                  Property="style"
                  PropertyKey="backgroundColor"
                  StartValue="#FFFF66"
                  EndValue="#FFFFFF" />
               <Color
                  AnimationTarget="tableHarrastukset"
                  Property="style"
                  PropertyKey="backgroundColor"
                  StartValue="#FFFF66"
                  EndValue="#FFFFFF" />
            </Parallel>
         </Sequence>
      </OnUpdated>
   </Animations>
</ajaxToolkit:UpdatePanelAnimationExtender>
```

Toinen animaatio liitetään suoraan `updatePanelVirheIlmoitus`-kontrolliin (joka on dialogissa), jolloin animaatio näytetään kyseisen `UpdatePanel`-kontrollin päivittyessä:

```
<ajaxToolkit:UpdatePanelAnimationExtender ID="upae2" runat="server"
    TargetControlID="updatePanelVirheIlmoitus">
   <Animations>
      <OnUpdated>
         <Sequence>
            <FadeIn AnimationTarget="uusiJasenVirheIlmoitus" Fps="
               20" />
            <FadeOut AnimationTarget="uusiJasenVirheIlmoitus" Fps="
               20" />
```

```
            </Sequence>
        </OnUpdated>
    </Animations>
</ajaxToolkit:UpdatePanelAnimationExtender>
```

Default.aspx.cs-tiedostossa luodaan sivun toiminnallisuus ja dynaamisesti luotavat sivun elementit.

`Page_Load`-metodi suoritetaan sivun latauduttua ja jokaisella takaisinkutsulla (myös Ajax). `IsPostBack` kertoo onko kyseessä takaisinkutsu, joten sen avulla voidaan jättää toimintoja suorittamatta kun kyseessä on takaisinkutsu. `kerho`-olio luodaan ja sen tiedot haetaan tiedostosta jokaisella takaisinkutsulla, koska sitä tarvitaan niissä kaikissa (kaikki attribuutit siis nollautuvat aina takaisinkutsuissa). Sivun ensimmäisellä latauksella otetaan hakuteksti ja hakukenttä sessiosta, jos ne on asetettu. Valittavan jäsenen id otetaan *request*-parametrista, jos se löytyy:

```
protected void Page_Load(object sender, EventArgs e)
{
    kerho = new Kerho();
    kerho.lueTiedostosta("kelmit");

    // Ettei suoriteta jokaisella takaisinkutsulla.
    if (!IsPostBack)
    {
        apuJasen = new Jasen();
        labelKerhoNimi.Text = kerho.getNimi();
        labelJasenia.Text = "Jäseniä: " + kerho.getJasenia().ToString
            ();
        asetaHakukentat();

        if (Session["hakuteksti"] == null) loydetyt = kerho.etsi("",
            apuJasen.ekaKentta());
        else
        {
            loydetyt = kerho.etsi((String)Session["hakuteksti"], Int32
                .Parse((String)Session["hakukentta"]));
            textHae.Text = (String)Session["hakuteksti"];
            selectKentta.SelectedValue = (String)Session["hakukentta"
                ];
        }

        int valittuJasenId = 0;
        String query = Request.QueryString["id"];
        if (query != null) valittuJasenId = Int32.Parse(query);
        else if (loydetyt.Count > 0) valittuJasenId = loydetyt[0].
            getTunnusNro();

        asetaJasenLista(valittuJasenId);
    }
}
```

Luodaan kuuntelija hakunapin klikkauksille. Asetetaan hakuun viive. Asetetaan hakuteksti ja hakukenttä sessiomuuttujiin, päivitetään jäsenvalintakomponentin alkiot, näytetään animaatio päivittämällä siihen liitetty `UpdatePanel`-kontrolli (`updatePanelAnimaatio.Update()`) ja lisätään historiapiste valitun jäsenen id-arvolle. Historiapisteelle annetaan nimi, arvo ja selaimen otsikko:

```
protected void buttonHae_Click(object sender, EventArgs e)
{
   System.Threading.Thread.Sleep(3000);
   loydetyt = kerho.etsi(textHae.Text, Int32.Parse(selectKentta.
      SelectedValue));
   Session["hakuteksti"] = textHae.Text;
   Session["hakukentta"] = selectKentta.SelectedValue;

   if (loydetyt.Count > 0) asetaJasenLista(loydetyt[0].getTunnusNro
      ());
   else asetaJasenLista(-1);
   updatePanelAnimaatio.Update();
   ToolScriptManager1.AddHistoryPoint("jasen", listBoxJasenet.
      SelectedValue, "Kerho: Jäsen " + listBoxJasenet.SelectedValue
      );
}
```

Luodaan jäsenvalintakomponentin valinnanvaihtumiskuuntelija. Jos kyseessä on asynkroninen kutsu, ja jos ei olla navigoimassa (painettu esimerkiksi selaimen takaisin-nappia) niin lisätään valitun jäsenen id:lle historiapiste. Näytetään valitun jäsenen tiedot sivulla:

```
protected void listBoxJasenet_SelectedIndexChanged(object sender,
   EventArgs e)
{
   // Lisätään jäsenen id:lle historiapiste.
   if (ToolScriptManager1.IsInAsyncPostBack && !ToolScriptManager1.
      IsNavigating)
   {
      ToolScriptManager1.AddHistoryPoint("jasen", listBoxJasenet.
         SelectedValue, "Kerho: Jäsen " + listBoxJasenet.
         SelectedValue);
   }
   naytaJasenTiedot();
}
```

Luodaan kuuntelija navigoinnille. Asetetaan jäsenvalintakomponentin valituksi jäseneksi historiapisteestä löytyvä id ja päivitetään jäsenen tiedot. Tässä pitää myös päivittää `UpdatePanel`-kontrollit täältä koodin puolelta, että niiden sisältö päivittyy:

```
protected void ToolScriptManager1_Navigate(object sender,
   HistoryEventArgs e)
```

```
{
   if (e.State["jasen"] == null) listBoxJasenet.SelectedValue =
       loydetyt[0].getTunnusNro().ToString();
   else listBoxJasenet.SelectedValue = e.State["jasen"].ToString();

   naytaJasenTiedot();
   updatePanelListJasenet.Update();
   UpdatePanelJasentiedot.Update();
}
```

Luodaan alkiot hakukenttävalintakomponentille (`selectKentta`). Luodaan ne `ListItem`-olioina, joille annetaan tekstiksi kentän kysymys ja arvoksi kentän numero:

```
public void asetaHakukentat()
{
   for (int i = apuJasen.ekaKentta(); i < apuJasen.getKenttia(); i
       ++)
   {
      ListItem item = new ListItem(apuJasen.getKysymys(i), i.
          ToString());
      selectKentta.Items.Add(item);
   }
}
```

Asetetaan löydetyt jäsenet jäsenvalintakomponentin (`listBoxJasenet`) alkioiksi. Parametrina vastaanotetaan valittavan jäsenen id. Valintakomponentin alkiot luodaan myös `ListItem`-olioina. Näytetään valitun jäsenen tiedot tai tieto, ettei jäseniä ole löytynyt:

```
public void asetaJasenLista(int jasenId)
{
   listBoxJasenet.Items.Clear();

   if (loydetyt.Count > 0)
   {
      foreach (Jasen jasen in loydetyt)
      {
      ListItem item = new ListItem(jasen.getNimi(), jasen.
          getTunnusNro().ToString());
          listBoxJasenet.Items.Add(item);
      }
      listBoxJasenet.SelectedValue = jasenId.ToString();
      naytaJasenTiedot();
   }
   else
   {
      naytaVirheIlmoitus();
   }
}
```

Näytetään jäsenen tiedot taulukossa. Luodaan taulukkoon otsikkorivi

(`luoTietoOtsikkorivi(jasen)`) ja jokaiselle jäsenen kentälle oma rivi (`luoTietorivi(jasen, i)`). Rivit luodaan kuten `JasenTietoLomake`-komponentin taulukolle, joten niitä ei enää tarkemmin esitellä:

```
public void naytaJasenTiedot()
{
   tableJasentiedot.Controls.Clear();
   tableHarrastukset.Controls.Clear();
   Jasen jasen = kerho.annaJasenId(Convert.ToInt32(listBoxJasenet.
       SelectedValue));
   luoTietoOtsikkorivi(jasen);

   for (int i = jasen.ekaKentta(); i < jasen.getKenttia(); i++)
   {
      luoTietorivi(jasen, i);
   }
   naytaJasenenHarrastukset(jasen);
}
```

Taulukon otsikkorivi sisältää linkin jäsenen muokkaussivulle:

```
HyperLink link = new HyperLink();
link.Text = "[Muokkaa]";
link.NavigateUrl = "/jasensivu/default.aspx?id=" + jasen.
    getTunnusNro();
```

Näytetään jäsenen harrastukset taulukossa. Taulukolle luodaan otsikko, rivi harrastusten kysymyksille ja rivit varsinaisille harrastusten tiedoille:

```
public void naytaJasenenHarrastukset(Jasen jasen)
{
   tableHarrastukset.Visible = true;
   List<Harrastus> harrastukset = kerho.annaHarrastukset(jasen);

   luoHarrastusOtsikkorivi();
   luoHarrastusKysymysRivi(harrastukset);
   luoHarrastusTietorivit(harrastukset);
}
```

Luodaan rivit harrastusten tiedoille. Jokaiselle harrastuksen kentälle luodaan taulukkoon oma solu, jonka tekstiksi asetetaan kentän arvo. Solun sisältö keskitetään, jos kyseessä ei ole ensimmäinen kenttä:

```
public void luoHarrastusTietorivit(List<Harrastus> harrastukset)
{
   foreach (Harrastus harrastus in harrastukset)
   {
      TableRow harrastusRivi = new TableRow();
      tableHarrastukset.Rows.Add(harrastusRivi);

      for (int i = harrastus.ekaKentta(); i < harrastus.getKenttia
          (); i++)
```

```
        {
            TableCell harrastusSolu = new TableCell();
            harrastusSolu.Text = harrastus.anna(i);
            if (i != harrastus.ekaKentta()) harrastusSolu.
                HorizontalAlign = HorizontalAlign.Center;
            harrastusSolu.BorderStyle = BorderStyle.Solid;
            harrastusRivi.Cells.Add(harrastusSolu);
        }
    }
}
```

Luodaan kuuntelija dialogin jäsenenlisäysnapille. Tarkistetaan `JasenTietoLomake`-komponentilta kenttien virheet ja tallennetaan jäsen, jos virheitä ei ole. Näytetään myös animaatio ja päivitetään historiapiste. Jos kentissä on virheitä, niin näytetään siitä virheilmoitus:

```
public void buttonLisaa_Click(object sender, EventArgs e)
{
    apuJasen = new Jasen();
    loydetyt = kerho.etsi("", apuJasen.ekaKentta());

    if (JasenTietoLomake1.tarkistaJasenKentat() == 0 && !
        JasenTietoLomake1.Nimityhja)
    {
        apuJasen = JasenTietoLomake1.Jasen;
        apuJasen.rekisteroi();
        kerho.korvaaTaiLisaa(apuJasen);
        kerho.tallenna();
        loydetyt.Add(apuJasen);
        asetaJasenLista(apuJasen.getTunnusNro());
        // Näytetään animaatio
        updatePanelAnimaatio.Update();
        Session["hakuteksti"] = null;
        labelJasenia.Text = "Jäseniä: " + kerho.getJasenia().ToString
            ();
        ToolScriptManager1.AddHistoryPoint("jasen", listBoxJasenet.
            SelectedValue, "Kerho: Jäsen " + listBoxJasenet.
            SelectedValue);

        suljeDialogi();
    }
    else if (JasenTietoLomake1.Nimityhja) uusiJasenVirheIlmoitus.
        InnerHtml = "Nimi ei saa olla tyhjä!";
    else uusiJasenVirheIlmoitus.InnerHtml = "Virheellisiä syötteitä!
        ";
}
```

Suljetaan dialogi, jolloin asetetaan myös `JasenTietoLomake`-komponentille tyhjä jäsen ja päivitetään komponentin sisältö. Tyhjennetään myös mahdollinen virheilmoitus:

```
public void suljeDialogi()
{
    apuJasen = new Jasen();
```

```
    JasenTietoLomake1.Jasen = apuJasen;
    JasenTietoLomake1.paivitaLomake();
    uusiJasenVirheIlmoitus.InnerHtml = "";

    uusiJasenDialog.Hide();
}
```

Luodaan kuuntelija jäsenen poistamisnapille. Poistettavan jäsenen id otetaan jäsenvalintakomponentin valinnasta (`listBoxJasenet.SelectedValue`). Näytetään animaatio ja päivitetään historiapiste:

```
protected void buttonPoista_Click(object sender, EventArgs e)
{
    apuJasen = new Jasen();
    loydetyt = kerho.etsi("", apuJasen.ekaKentta());

    int id = Convert.ToInt32(listBoxJasenet.SelectedValue);

    loydetyt.Remove(kerho.annaJasenId(id));
    kerho.poista(id);
    kerho.tallenna();
    asetaJasenLista(loydetyt[0].getTunnusNro());
    labelJasenia.Text = "Jäseniä: " + kerho.getJasenia().ToString();

    // Näytetään animaatio
    updatePanelAnimaatio.Update();

    ToolScriptManager1.AddHistoryPoint("jasen", listBoxJasenet.
        SelectedValue, "Kerho: Jäsen " + listBoxJasenet.SelectedValue
        );
    Session["hakuteksti"] = null;
}
```

B.3 Jäsensivu

Jäsensivu on myös on ASP.NET WWW-lomakesivu, joka koostuu **Default.aspx**-ulkoasutiedostosta ja **Default.aspx.cs**-kooditiedostosta.

Default.aspx-tiedosto on hyvin samankaltainen Kerhosivun vastaavan kanssa (ei ole mitään uutta), joten sitä ei tässä turhaan käydä läpi.

Default.aspx.cs-tiedoston `Page_Init`-metodi suoritetaan ennen `Page_Load`-metodia, mutta myös se suoritaan jokaisella takaisinkutsulla. Otetaan jäsenen id `request`-parametrista ja asetetaan kyseinen jäsen `JasenTietoLomake`-komponentin jäseneksi:

```
protected void Page_Init(object sender, System.EventArgs e)
```

```
{
   String query = Request.QueryString["id"];
   if (query != null) jasenId = Int32.Parse(query);

   kerho = new Kerho();
   kerho.lueTiedostosta("kelmit");
   jasen = kerho.annaJasenId(jasenId);
   JasenTietoLomakeJasensivu.Jasen = jasen;
}
```

Sivun ensimmäisellä latauksella otetaan jäsenen harrastukset kerhosta ja asetetaan ne sessiomuuttujaan. Takaisinkutsuissa harrastukset sitten otetaan kyseisestä sessiomuuttujasta, johon muutetut harrastukset aina tallennetaan. Harrastustenmuokkauslomake ja dialogin sisältö pitää luoda uudelleen myös jokaisessa takaisinkutsussa, tai ne häviävät. Dialogin tapauksessa voisi tietysti tallentaa tiedon dialogin auki olemisesta sessiomuuttujaan, ja luoda sen sisältö tässä sen mukaan, niin sitä ei luotaisi turhaan kun dialogi ei ole auki:

```
protected void Page_Load(object sender, EventArgs e)
{
   // Ettei suoriteta jokaisella takaisinkutsulla.
   if (!IsPostBack)
   {
      harrastukset = kerho.annaHarrastukset(jasen);
      Session["harrastukset"] = harrastukset;

      labelJasenNimi.Text = jasen.getNimi();
      linkTakaisin.NavigateUrl = "../kerho/default.aspx?id=" +
         jasen.getTunnusNro();
   }
   // Suoritetaan vain takaisinkutsuilla.
   else
   {
      harrastukset = Session["harrastukset"] as List<Harrastus>;
   }

   naytaHarrastusMuokkausLomake();
   // Myös dialogin sisältö pitää luoda uudelleen jokaisella
      takaisinkutsulla.
   luoUusiHarrastusLomake();
}
```

Luodaan harrastustenmuokkauslomake. Luodaan sille rivit kuten Kerhosivun taulukolle. Tyhjennetään taulukko aina aluksi:

```
public void naytaHarrastusMuokkausLomake()
{
   tableHarrastusLomake.Controls.Clear();

   luoHarrastusMuokkausOtsikkorivi();
   luoHarrastusMuokkausKysymysRivi();
   luoHarrastusMuokkausKenttarivit();
```

```
}
```

Luodaan rivit harrastustenmuokkauskentille. Jokaiselle harrastuksen kentälle luodaan oma solu, johon lisätään tekstikenttä ja paikka virheilmoitukselle. Tekstikentän tekstinvaihtuminen asetetaan triggeriksi `UpdatePanel`-kontrollille:

```
public void luoHarrastusMuokkausKenttarivit()
{
   foreach (Harrastus harrastus in harrastukset)
   {
      TableRow harrastusRivi = new TableRow();
      tableHarrastusLomake.Rows.Add(harrastusRivi);

      for (int i = harrastus.ekaKentta(); i < harrastus.getKenttia
          (); i++)
      {
         TableCell harrastusSolu = new TableCell();
         if (i != harrastus.ekaKentta()) harrastusSolu.
            HorizontalAlign = HorizontalAlign.Center;
         harrastusSolu.BorderStyle = BorderStyle.Solid;

         TextBox textTieto = luoHarrastusMuokkausKentta(harrastus,
            i);
         Label labelVirhe = luoHarrastusMuokkausVirheLabel(
            harrastus, i);
         harrastusSolu.Controls.Add(textTieto);
         harrastusSolu.Controls.Add(labelVirhe);
         harrastusRivi.Cells.Add(harrastusSolu);

         luoHarrastusMuokkausTriggeri(harrastus, i);
      }
   }
}
```

Luodaan tekstikenttä tietylle harrastuksen kentälle metodilla `luoHarrastusMuokkausKentta(harrastus, i)`. Tämä luodaan kuten jäsentietolomakkeen kenttä. Annetaan sille yksilöllinen id (`harrastus.getId() + ","+ i.ToString()`). Myös tekstinvaihtumiskuuntelija, triggeri ja virheilmoitus luodaan kuten jäsentietolomakkeessa.

Tekstikentän tekstinvaihtumiskuuntelijassa asetetaan tekstikentän sisältö harrastukselle sekä tarkistetaan virhe seuraavalla metodilla. Harrastuksen id ja kentän numero saadaan tekstikentän id:stä. Asetetaan muokattu harrastus valituksi (sessiomuuttujaan) harrastuksen poistoa varten. Asetuksen jälkeen harrastukset tallennetaan sessiomuuttujaan. Näytetään mahdollinen virheilmoitus (paikka haetaan `FindControl`-metodilla) ja muutetaan tekstikentän tyyli virheen mukaan:

```
public void asetaJaTarkistaHarrastusMuokkausKentta(TextBox box)
{
   string[] osat = box.ID.Split(',');
   String virhe = "";

   foreach (Harrastus har in harrastukset)
   {
      if (har.getId() == Int32.Parse(osat[0]))
      {
         virhe = har.aseta(Int32.Parse(osat[1]), box.Text);
         Session["valittuHarrastusID"] = har.getId();
         break;
      }
   }
   Session["harrastukset"] = harrastukset;

   if (virhe != null && box.CssClass.Equals(""))
   {
      box.CssClass = "virheTieto";
   }
   else if (virhe == null && box.CssClass.Equals("virheTieto"))
   {
      box.CssClass = "";
   }

   Label label = (Label)FindControl("virhe" + box.ID);
   if (virhe != null) label.Text = virhe;
   else label.Text = "";
}
```

Tallennetaan jäsen kerhoon, jos kentissä ei ole virheitä. Tämä metodi suoritetaan siis tallennusnapin klikkauksilla. Kaikki kentät asetetaan jäsenelle ja harrastuksille tässä uudestaan (ja tarkistetaan virheet), koska attribuutit nollautuvat aina takaisinkutsuissa. Näytetään ilmoitus tallennuksen onnistumisesta:

```
public void tallennaJasen()
{
   tallennusVirheita = 0;
   asetaKentat();

   if (tallennusVirheita == 0)
   {
      labelIlmoitus.Text = "Jäsen tallennettu!";
      labelIlmoitus.CssClass = "onnistumisIlmoitus";
      jasen = JasenTietoLomakeJasensivu.Jasen;
      kerho.korvaaTaiLisaa(jasen);
      kerho.tallenna();
   }
   else
   {
      labelIlmoitus.Text = "Virheellisiä syötteitä!";
      labelIlmoitus.CssClass = "virheIlmoitus";
   }
}
```

Asetetaan tekstikenttien arvot jäsenelle ja harrastuksille ja tarkistetaan niistä virheet.

Jäsenen kentät asetetaan `JasenTietoLomake`-komponentin metodilla `tarkistaJasenKentat()`:

```
public void asetaKentat()
{
   tallennusVirheita = JasenTietoLomakeJasensivu.
      tarkistaJasenKentat();
   asetaHarrastusKentat();
}
```

Asetetaan harrastuskentät. Poistetaan ensin kerhosta vanhat harrastukset ja lisätään sitten uudet harrastukset, jotka on otettu sessiosta:

```
public void asetaHarrastusKentat()
{
   List<Harrastus> kerhonHarrastusLista = kerho.annaHarrastukset(
      jasen);

   // Poistetaan vanhat harrastukset kerhosta.
   foreach (Harrastus har in kerhonHarrastusLista)
   {
      kerho.poistaHarrastus(har);
   }

   // Tarkistetaan ja lisätään uudet harrastukset kerhoon.
   foreach (Harrastus har in harrastukset)
   {
      for (int i = har.ekaKentta(); i < har.getKenttia(); i++)
      {
         TextBox box = (TextBox)FindControl(har.getId() + "," + i.
            ToString());
         String virhe = har.aseta(i, box.Text);

         if (virhe != null) tallennusVirheita++;
      }
      kerho.lisaa(har);
   }
}
```

Luodaan dialogiin harrastuksenlisäyslomake. Tämä tehdään samalla tavalla kuin `JasenTietoLomake`-komponentin lomake, joten tätä ei tämän tarkemmin enää käsitellä:

```
public void luoUusiHarrastusLomake()
{
   apuHarrastus = new Harrastus(jasen.getTunnusNro().ToString());
   for (int i = apuHarrastus.ekaKentta(); i < apuHarrastus.
      getKenttia(); i++)
   {
      luoUusiHarrastusLomakerivi(i);
   }
}
```

Luodaan kuuntelija dialogin harrastuksenlisäysnapille. Tarkistetaan virheet lomakkeelta ja lisätään harrastus, jos virheitä ei ole. Lisätään uusi harrastus listaan ja asetetaan uusi lista sessiomuuttujaan. Luodaan harrastustenmuokkauslomake uudelleen ja näytetään animaatio. Jos on virheitä niin näytetään virheilmoitus:

```
public void buttonLisaa_Click(object sender, EventArgs e)
{
   if (tarkistaUusiharrastusKentat() == 0 && !alaKentta.Text.Equals
       (""))
   {
      apuHarrastus.rekisteroi();
      harrastukset.Add(apuHarrastus);
      Session["harrastukset"] = harrastukset;
      naytaHarrastusMuokkausLomake();
      updatePanelAnimaatio.Update();
      suljeDialogi();
   }
   else if (alaKentta.Text.Equals("")) uusiharrastusVirheIlmoitus.
       InnerHtml = "Ala ei saa olla tyhjä!";
   else uusiharrastusVirheIlmoitus.InnerHtml = "Virheellisiä syö
       tteitä!";
}
```

Dialogin sulkemisessa luodaan uusi tyhjä harrastus jäsenelle ja päivitetään dialogin lomake sen tiedoilla:

```
public void suljeDialogi()
{
   apuHarrastus = new Harrastus(jasen.getTunnusNro().ToString());
   uusiharrastusVirheIlmoitus.InnerHtml = "";
   paivitaUusiharrastusLomake();

   uusiHarrastusDialog.Hide();
}
```

Luodaan dialogin lomake uudelleen:

```
public void paivitaUusiharrastusLomake()
{
   tableUusiHarrastusLomake.Controls.Clear();
   luoUusiHarrastusLomake();
   updatePanelUusiHarrastuslomake.Update();
}
```

Luodaan kuuntelija harrastuksenpoistonapille. Poistettavan harrastuksen id otetaan sessiomuuttujasta. Poistetaan harrastus harrastuslistasta ja päivitetään uusi harrastuslista sessiomuuttujaan. Päivitetään harrastustenmuokkauslomake ja näytetään animaatio:

```
protected void buttonPoistaHarrastus_Click(object sender, EventArgs
    e)
{
   foreach (Harrastus har in harrastukset)
   {
      if (har.getId() == (int)Session["valittuHarrastusID"])
      {
         harrastukset.Remove(har);
         Session["harrastukset"] = harrastukset;

         naytaHarrastusMuokkausLomake();
         updatePanelAnimaatio.Update();
         return;
      }
   }
}
```

C Ruby on Rails -kerhosovelluksen luominen

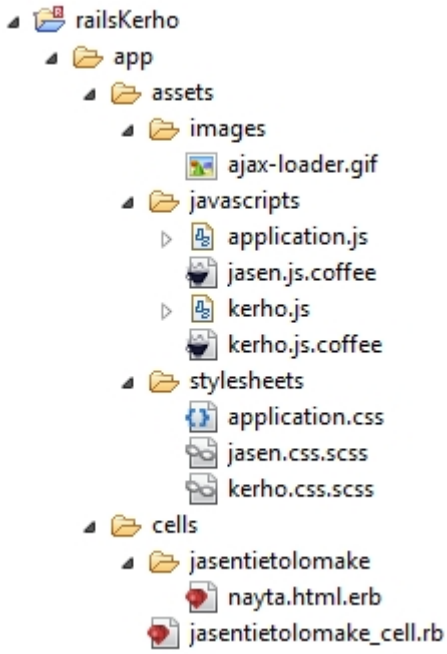
Rails-sovellus koostuu kahdesta ohjaimesta (`kerho` ja `jasen`) ja niiden toiminnoista (`kerhosivu` ja `jasensivu`). Sivujen eri osat luotiin partialien avulla. Jäsentietolomake luotiin *cells*-lisäosan avulla uudelleenkäytettävänä näkymänä. Kuvio 30 näyttää sovelluksen kerhosivun käyttöliittymän. Kuviot 31 ja 32 esittävät sovelluksen rakenteen. Kuvio 33 kuvaa Ajax-kutsun toiminnan, kun kutsu tehdään Javascript-tiedoston kautta.

C.1 Jäsentietolomake-cell

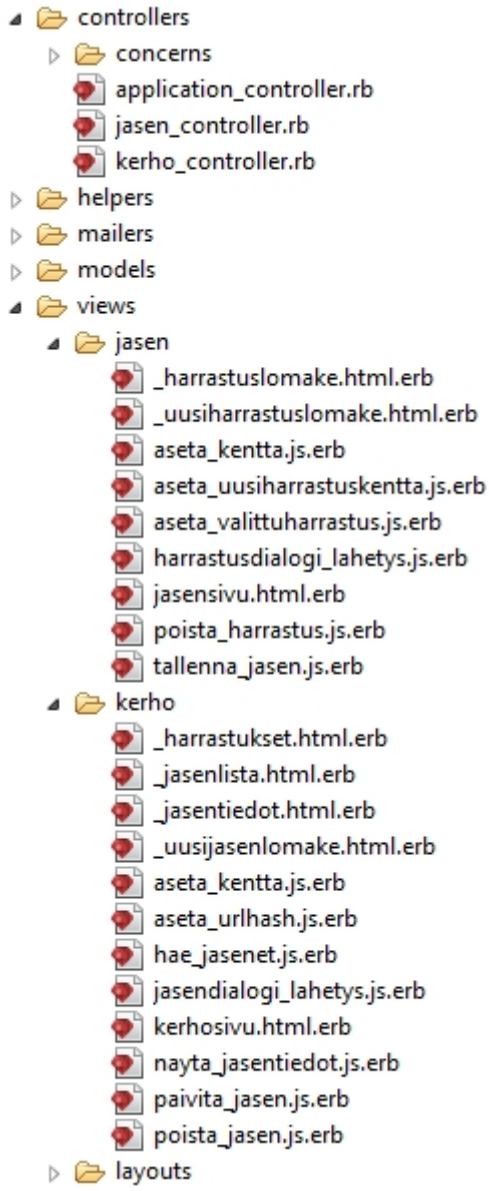
Jäsentietolomake luotiin uudelleenkäytettävänä näkymänä *cells*-lisäosan avulla.

nayta.html.erb-mallineessa luodaan jäsenenmuokkauslomake normaalina html-taulukkona. Riveille asetetaan kentän kysymys ja tekstikenttä, jonka sisällöksi asetetaan jäsenen kentän arvo. Tekstikenttä luodaan `text_field_tag`-avustajan avulla. Sille annetaan nimi kentän numeron mukaan, jolloin sen arvo saadaan helposti ohjaimella parametrista kentän numeron perusteella (kun tekstikenttä sijotetaan `form_tag`-avustajan sisään ja lomake lähetetään `submit_tag`"-napilla). Kentälle asetetaan Javascript-funktio napin ylösnousemiselle (`:onkeyup => “aseta_arvo('jasen', this.value, this.name)”`), jolle välitetään parametrina tieto, että kyseessä on jäsenkenttä, kentän arvo ja kentän nimi:

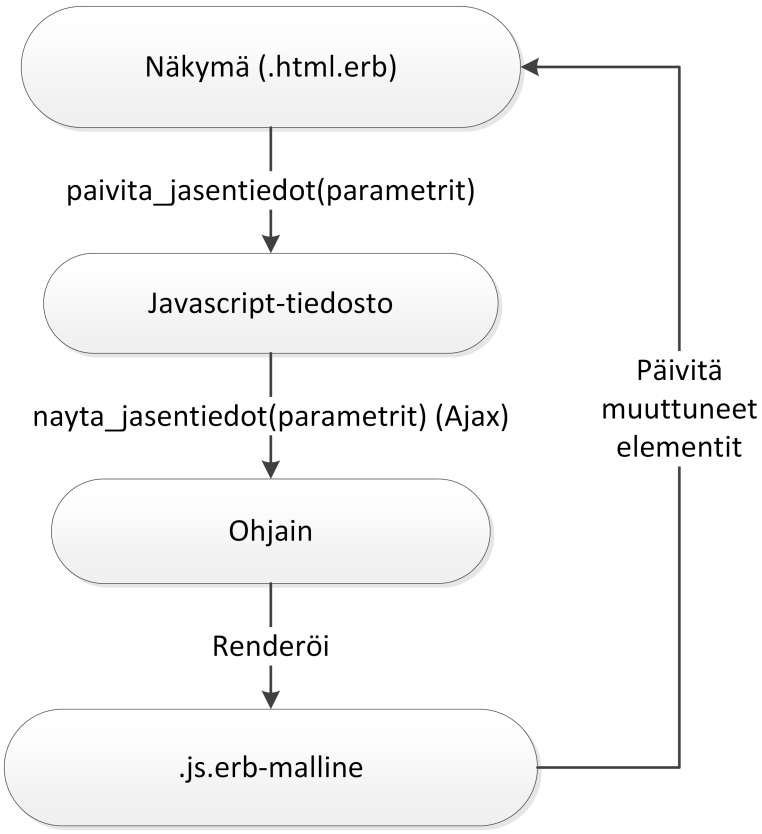
Kuvio 30: Rails-kerhosovelluksen kerhosivun käyttöliittymä



Kuvio 31: Rails-kerhoprojektin kansiorakenne (kuva1)



Kuvio 32: Rails-kerhoprojektin kansiorakenne (kuva2)



Kuvio 33: Railsin kerho-sovelluksen Ajax-kutsu Javascript-tiedoston kautta

```
<table class="tableJasenLomake">
   <% (@apujasen.eka_kentta..@apujasen.get_kenttia-1).each do |i|
      %>
      <tr>
         <td><b><%= @apujasen.get_kysymys(i)%>:</b></td>

         <td><%= text_field_tag i.to_s, @apujasen.anna(i), :class
            => "jasenkentta", :onkeyup => "aseta_arvo('jasen', this
            .value, this.name)" %></td>

         <td id=<%= "virhe" + i.to_s %> class="virhe"></td>
      </tr>
   <% end %>
</table>
```

jasentietolomake_cell.rb-tiedostossa luodaan toiminto, jolla edellinen malline renderöidään. `jasen`-parametrina vastaanotetaan lomakkeen luomisessa käytettävä jäsen:

```
def nayta(parametrit)
   @apujasen = parametrit[:jasen]
   render
end
```

kerho.js-tiedostossa luodaan Javascript-funktio kentän asettamiselle. jQueryn avulla selvitetään onko kentälle asetettu virhetyyli ja välitetään siitä tieto ohjaimelle. Ohjaimelle välitetään myös kentän tyyppi (jäsen vai harrastus), kentän arvo ja kentän numero. Ajax-kutsun osoiteelle (`aseta_kentta`) pitää lisätä myös reitti `routes.rb`-tiedostoon (liite G.28):

```
function aseta_arvo(tyyppi,teksti, i) {
   var onkoVirhetyyli = $("#" + i).hasClass("virheKentta");
   jQuery.ajax({
      url: "aseta_kentta",
      type: "GET",
      data: {"tyyppi": tyyppi, "teksti": teksti, "i": i, "onkovirhe
         ": onkoVirhetyyli},
   });
}
```

C.2 Kerhosivun html-mallineet

kerhosivu.html.erb-mallineessa luodaan sivu. Ensin asetetaan sivun Javascript-tiedosto ja sitten sivun otsikot. Otsikot luodaan normaalisti html-elementeillä, joiden sisään sijoitetaan Ruby-koodia `<%= %>` -tagien sisään (=-merkki kertoo, että koodin

tulos näytetään sivulla). Otsikoihin siis sijoitetaan ohjaimen `kerhosivu`-toiminnossa asestetut attribuutit:

```
<%= javascript_include_tag "kerho" %>

<h1> <%= @kerhonimi %> </h1>
<h3 id="jasenmaara"> Jäseniä: <%= @jasenia %> </h3>
```

Luodaan jäsenten hakulomake. Lomake luodaan Railsin `form_tag`-avustajan avulla. `kerho_hae_jasenet_path`-reitti (määritelty `routes.rb`-tiedostossa) kertoo lomakkeen lähettämisessä suoritettavan toiminnon (`kerho`-ohjaimen `hae_jasenet`-toiminto). `remote: true`-parametrilla kerrotaan, että lomake lähetetään asynkronisesti. Myös lomakkeen kentät luodaan Rails-avustajien avulla. Tekstikentälle asetetaan valittu arvo `@hakuehto`-attribuutin avulla. Hakukentän valintakomponentille asetetaan alkiot `options_for_select`-parametrin avulla, jolle määritellään alkiot `@kentat`-hashmapilla ja valittu alkio `@valittukentta`-attribuutilla. `submit_tag`-avustaja luo napin lomakkeen lähettämiselle. `:onclick => "nayta_odotusIlmaisin()"` parametrilla asetetaan napin klikkaukseen myös Javascript-funktio, jolla odotusilmaisin asetetaan näkyviin:

```
<%= form_tag(kerho_hae_jasenet_path, remote: true) do %>
   <%= text_field_tag(:hakuehto, @hakuehto) %>
   <%= select_tag(:hakukentta, options_for_select(@kentat,
      @valittukentta)) %>
   <%= submit_tag 'Hae', :onclick => "nayta_odotusIlmaisin()" %>
<% end %>
```

Odotusilmaisin luodaan normaalina html-elementtinä, joka asetetaan tyylin avulla piiloon. Sille asetetaan teksti ja kuva. `asset_path`-reitti hakee kuvan suoraan `assets/images`-kansiosta:

```
<p id="odotusIlmaisin" style="display:none">
   Haetaan jasenia... <img src="<%= asset_path( 'ajax-loader.gif' )
       %>" />
</p>
```

Jäsenvalintakomponentti luodaan partialin (`_jasenlista.html.erb`) avulla. Dialogi luodaan `link_to_modal`-avustajalla (*jquery-modal-rails*-lisäosa), jolle annetaan linkin nimi ja dialogin sisällön id (`#uusijasendialog`). Jäsenen poistolinkkiin asetetaan varmistus (`:data => { confirm: "Poistetaanko valittu`

jäsen?"}):

```
<p id="jasenValinnat">
   <%= render partial: "jasenlista" %>
</p>
<%= link_to_modal "Lisää uusi jäsen", "#uusijasendialog" %>
</br>
<%= link_to 'Poista jäsen', kerho_poista_jasen_path, :remote =>
   true, :data => { confirm: "Poistetaanko valittu jäsen?" } %>
```

Dialogin sisältö, jäsentietotaulukko ja harrastustaulukko luodaan partialien avulla. Dialogin sisältö asetetaan lisäksi piiloon ja annetaan sille otsikko:

```
<div id="uusijasendialog" title="Uusi jasen" style="display:none">
   <%= render partial: "uusijasenlomake" %>
</div>

<div id="jasentiedot" class="jasentiedot">
   <%= render partial: "jasentiedot" %>
</div>

<div id="harrastukset" class="harrastukset">
   <%= render partial: "harrastukset" %>
</div>
```

_jasenlista.html.erb-partialissa luodaan jäsenvalintakomponentti, joka luodaan kuten hakukenttävalintakomponentti, paitsi tälle asetetaan koko (`size: 20`), jottei tule alasvetovalikkoa. Asetetaan arvon muuttumiselle Javascript-funktio (`:onchange => "paivita_jasentiedot(this.value)"`), jonka kautta valitun jäsenen tiedot näytetään sivulla:

```
<%= select_tag(:select_jasen, options_for_select(@jasenlista_alkiot
   , @jasenid),
      {size: 20, :class => 'selectJasen', :onchange => "
         paivita_jasentiedot(this.value)"}) %>
```

_uusijasenlomake.html.erb-partialissa luodaan dialogin sisältö. Luodaan lomake, johon luodaan uudelleenkäytettävä *cells*-näkymä. Se renderöidään suorittamalla `jasentietolomake`-cellin `nayta`-toiminto ja viemällä sille parametrina jäsen (`@lomakejasen`). `kerho`-ohjaimen `jasendialogi_lahetys`-toiminto vastaanottaa lomakkeen lähetyksen molemmista napeista (painettu nappi saadaan ohjaimessa `commit`-parametrilla):

```
<%= form_tag(kerho_jasendialogi_lahetys_path, remote: true) do %>
```

```
   <%= render_cell :jasentietolomake, :nayta, :jasen =>
      @lomakejasen %>

   <p id="uusijasenilmoitus" class="virheilmoitus"></p>
   <%= submit_tag 'Peruuta' %>
   <%= submit_tag 'Lisää jäsen' %>

<% end %>
```

_jasentiedot.html.erb-partialissa luodaan taulukko jäsenen tiedoille. Taulukko luodaan normaalina html-taulukkona. Jäsenelle luodaan linkki sen muokkaussivulle (`jasen_jasensivu_path`), jonne viedään parametrina jäsenen id (`id: @valittujasen.get_tunnusnro`):

```
<% if @valittujasen != "" %>
   <p>
      <b>Jasen: <%= @valittujasen.get_nimi %> </b>
       <%= link_to "[Muokkaa]", jasen_jasensivu_path(id:
          @valittujasen.get_tunnusnro) %>
   </p>

   <table class="tableJasenLomake">
      <% (@valittujasen.eka_kentta..@valittujasen.get_kenttia-1).
         each do |i| %>
         <tr>
               <td><b><%= @valittujasen.get_kysymys(i)%>:</b></td>
               <td><%= @valittujasen.anna(i) %></td>
         </tr>
      <% end %>
   </table>
<% else %>
   <p>
      <b>Hakuehdoilla ei löytynyt yhtään jäsenta!</b>
   </p>
<% end %>
```

_harrastukset.html.erb-partialissa luodaan taulukko jäsenen harrastuksille. Taulukkoon luodaan ensin kysymysrivi ja sitten rivit jokaiselle harrastukselle:

```
<% if @valittujasen != "" %>
   <% if @harrastukset.size > 0 %>
      <table class="tableHarrastukset">
         <caption><b>Harrastukset:</b></caption>
         <tr>
            <% (@harrastukset[0].eka_kentta..@harrastukset[0].
               get_kenttia-1).each do |i| %>
               <td><b><%= @harrastukset[0].get_kysymys(i) %></b></
                  td>
            <% end %>
         </tr>

         <% @harrastukset.each do |harrastus| %>
            <tr>
               <% (harrastus.eka_kentta..harrastus.get_kenttia-1).
                  each do |i| %>
```

```
                    <% if i == harrastus.eka_kentta %>
                       <td class="tableEka"> <%= harrastus.anna(i) %>
                           </td>
                    <% else %>
                       <td> <%= harrastus.anna(i) %> </td>
                    <% end %>
                 <% end %>
              </tr>
           <% end %>
        </table>
     <% end %>
<% end %>
```

C.3 Kerhosivun Javascript-funktiot

`kerho.js`-tiedosto sisältää kerhosivun Javascriptin.

Näytetään odotusilmaisin jQueryn avulla:

```
function nayta_odotusIlmaisin() {
   $('#odotusIlmaisin').show();
}
```

Jäsenen tietojen näyttäminen. Parametrina saadaan valitun jäsenen id, joka välitetään ohjaimelle. Ajax-kutsu tehdään jQueryn avulla:

```
function paivita_jasentiedot(jasenid) {
   jQuery.ajax({
      url: "nayta_jasentiedot",
      type: "GET",
      data: {"jasenid": jasenid},
   });
}
```

Url-hashin muuttumiselle asetetaan suoritettava funktio. Otetaan tällöin hash url:stä ja välitetään se ohjaimelle, jossa päivitetään jäsenen tiedot:

```
window.onhashchange = urlHashMuuttunut;

function urlHashMuuttunut() {
   var hashid = window.location.hash.substring(1);

   jQuery.ajax({
      url: "paivita_jasen",
      type: "GET",
      data: {"hashid": hashid},
   });
};
```

Sivun latauduttua tarkistetaan onko url-hashia asetettu ja asetetaan se jos ei ole. Jos on asetettu, niin suoritetaan hashin muuttumismetodi, jossa päivitetään kyseisen jäsenen tiedot:

```
$( window ).load(function() {
   if (window.location.hash == "")
   {
      jQuery.ajax({
         url: "aseta_urlhash",
         type: "GET",
         data: {},
      });
   }
   else urlHashMuuttunut();
});
```

C.4 Kerhosivun ohjain

Ohjaimen (`kerho_controller.rb`) `kerhosivu`-toiminnossa luodaan `kerhosivu`-näkymän tarvitsemat attribuutit. Sivua avatessa siis näkymä renderöidään aina tämän toiminnon kautta:

```
def kerhosivu
   session[:toiminto] = ""
   session[:edellinenhash]  = ""
   @apujasen = Jasen.new
   @lomakejasen = @apujasen

   @kerho = Kerho.new
   @kerho.lue_tiedostosta("kelmit")
   @jasenia = @kerho.get_jasenia
   @kerhonimi = @kerho.get_nimi

   haejasenet_ja_luojasenlista

   if @valittujasen != ""
      @harrastukset = @kerho.anna_harrastukset(@valittujasen)
   end

   luo_hakulista_alkiot
end
```

Haetaan kerhon jäsenet ja luodaan alkiot jäsenvalintakomponentille. Hakuehto ja hakukenttä otetaan sessiosta, jos ne on asetettu, muuten haetaan kaikki jäsenet:

```
def haejasenet_ja_luojasenlista
   if session[:hakuehto] == nil
      jasenet = @kerho.etsi("", @apujasen.eka_kentta)
      session[:hakuehto] = ""
      session[:hakukentta] = 1
      @valittukentta = @apujasen.eka_kentta
```

```
   else
      jasenet = @kerho.etsi(session[:hakuehto], session[:hakukentta
         ])
      @valittukentta = session[:hakukentta]
   end

   @hakuehto = session[:hakuehto]
   luo_jasenlista_alkiot(jasenet)
   id = params[:id]

   # Pitaa tallentaa sessioon, jotta
   # saadaan id asetettua hashiin
   # aseta_urlhash-metodissa.
   session[:paramid] = id

   if id == nil && jasenet.count > 0
      @jasenid = jasenet[0].get_tunnusnro
      @valittujasen = jasenet[0]
   elsif id != nil
      @valittujasen = @kerho.anna_jasen_id(id)
      @jasenid = @valittujasen.get_tunnusnro
   end
   session[:valittuid] = @jasenid
end
```

Luodaan alkiot hakukenttävalintakomponentille. Alkiot luodaan hashmappiin, johon laitetaan kentän kysymys ja sen numero:

```
def luo_hakulista_alkiot
   @kentat = Hash.new

   (@apujasen.eka_kentta..@apujasen.get_kenttia-1).each do |i|
      @kentat[@apujasen.get_kysymys(i)] = i
   end
end
```

Luodaan alkiot jäsenvalintakomponentille. Jos ei ole löytynyt yhtään jäsentä, tehdään yksi tyhjä alkio. Muuten asetetaan alkioon jäsenen nimi ja tunnusnumero:

```
def luo_jasenlista_alkiot(jasenet)
   @jasenlista_alkiot = Hash.new

   if jasenet.count == 0
      @jasenlista_alkiot = {"" => ""}
      @jasenid = ""
      @valittujasen = ""
   else
      jasenet.each {|jasen|
         @jasenlista_alkiot[jasen.get_nimi] = jasen.get_tunnusnro
      }
   end
end
```

Otetaan vastaan Javascriptillä tehty Ajax-kutsu jäsenen tietojen näyttämiseksi. Jäsenen id välitettiin siis Ajax-kutsussa, joka nyt saadaan `params`-listasta. `respond_to`-lohkossa kerrotaan, että sivu päivitetään `.js.erb`-mallineella (`format.js`), eikä renderöidä koko sivua. Tässä siis Rails renderöi `.js.erb`-mallineen, jolla on sama nimi kuin tällä metodilla (`nayta_jasentiedot.js.erb`). Tässä pitää `kerho`-olio luoda aina uudelleen ja lukea sen tiedot tiedostosta, koska attribuutit nollautuvat jokaisella pyynnöllä:

```
def nayta_jasentiedot
   session[:toiminto] = "nayta_jasentiedot"
   @id = params[:jasenid]

   @kerho = Kerho.new
   @kerho.lue_tiedostosta("kelmit")
   jasen = @kerho.anna_jasen_id(@id)
   @hashid = valitse_jasen(jasen)

   respond_to do |format|
      format.js
   end
end
```

Asetetaan valitun jäsenen attribuutit ja palautetaan sen tunnusnumero:

```
def valitse_jasen(jasen)
   @valittujasen = jasen
   @jasenid = @valittujasen.get_tunnusnro
   session[:valittuid] = @jasenid
   @harrastukset = @kerho.anna_harrastukset(@valittujasen)
   return @jasenid
end
```

Otetaan vastaan Ajax-kutsu jäsenen kentän asettamiseksi, jossa tarkistetaan kentän virhe. Kentän arvo ja numero saadaan kutsun parametreista. Tässäkin renderöidään `.js.erb`-malline, jolla näytetään kentän mahdollinen virhe:

```
def aseta_kentta
   apujasen = Jasen.new
   arvo = params[:teksti]
   @i = params[:i]

   @virhe = apujasen.aseta(@i.to_i, arvo)

   respond_to do |format|
      format.js
   end
end
```

Otetaan vastaan näkymältä lähetetty dialogin napinpainallus. Näkymällä asetettu reitti siis osoittaa tähän metodiin. Painettu nappi saadaa `commit`-parametrista:

```
def jasendialogi_lahetys
   painettu_nappi = params[:commit]

   if painettu_nappi == "Peruuta"
      peruuta_jasenlisays
      @toiminto = "sulje"
   else
      @toiminto = "tallenna"
      lisaa_jasen
      session[:toiminto] = "lisaa_jasen"
   end

   respond_to do |format|
      format.js
   end
end
```

Jos on painettu *peruuta*-nappia, niin luodaan vaan uusi tyhjä jäsen, joka asetetaan lomakkeen jäseneksi, jotta saadaan luotua tyhjä lomake.

```
def peruuta_jasenlisays
   @apujasen = Jasen.new
   @lomakejasen = @apujasen
end
```

Jos on painettu *lisäys*-nappia, niin lisätään ja tallennetaan uusi jäsen kerhoon, jos ei ole virheellisiä syötteitä. Tässä ensin asetetaan kaikki lomakkeen kentät uuteen jäseneen, jolloin myös tarkistetaan kaikki virheet. Näin ei tarvitse säilyttää jäsentä eikä virhetietoja sessiomuuttujissa. Asetetaan myös kaikki renderöinnissä tarvittavat attribuutit. Virhetilanteessa asetetaan vaan tarvittavat *ilmoitus*- ja *virhe*-attribuutit:

```
def lisaa_jasen
   aseta_kentat_jaseneen
   nimikentta_arvo = params[@apujasen.eka_kentta.to_s]

   if @virheita == 0 && nimikentta_arvo != ""
      @kerho = Kerho.new
      @kerho.lue_tiedostosta("kelmit")
      @apujasen.rekisteroi
      lisaa_jasen_kerhoon
      @hashid =   @apujasen.get_tunnusnro
      @ilmoitus = "Jasen tallennettu!"
      @virheita = false
      @jasenmaara = @kerho.get_jasenia

      loydetyt = @kerho.etsi("", @apujasen.eka_kentta)
      luo_jasenlista_alkiot(loydetyt)

      @apujasen = Jasen.new
      @lomakejasen = @apujasen
```

```
      session[:hakuehto] = nil
      session[:toiminto] = "lisaa_jasen"
   elsif nimikentta_arvo == ""
      @ilmoitus = "Nimi ei saa olla tyhja!"
      @virheita = true
   else
      @ilmoitus = "Virheellisia syotteita!"
      @virheita = true
   end
end
```

Asetetaan lomakkeen kentät uuteen jäseneen ja tarkistetaan virheet. Lomakkeen tekstikentän arvo saadaan parametrista kentän numeron perusteella (`params[i.to_s]`), koska lomake on lähetetty `form_tag`-avustajan sisältä `submit`-painikkeella (ja koska tekstikentille asetettiin nimet niiden numeron mukaan):

```
def aseta_kentat_jaseneen
   @virheita = 0
   @apujasen = Jasen.new

   (@apujasen.eka_kentta..@apujasen.get_kenttia-1).each do |i|
         @virhe = @apujasen.aseta(i, params[i.to_s])
         tarkista_virhe
   end
end
```

Lisätään virheiden määrää, jos kentän syöte on virheellinen:

```
def tarkista_virhe
   if @virhe != nil
      @virheita = @virheita + 1
   end
end
```

Lisätään ja tallennetaan uusi jäsen kerhoon ja valitaan kyseinen jäsen:

```
def lisaa_jasen_kerhoon
   @kerho.korvaa_tai_lisaa(@apujasen)
   @kerho.tallenna
   valitse_jasen(@apujasen)
end
```

Otetaan vastaan näkymältä lähetetty hakunapin painallus. Asetetaan hakuun viive. Hakuehto ja hakukenttä saadaan parametreista, joiden nimi tulee suoraan lomakkeen kenttien nimistä:

```
def hae_jasenet
   sleep 3
   session[:toiminto] = "hae_jasenet"
   session[:hakuehto] = params[:hakuehto]
   session[:hakukentta] = params[:hakukentta]

   @kerho = Kerho.new
   @kerho.lue_tiedostosta("kelmit")
   loydetyt = @kerho.etsi(session[:hakuehto], session[:hakukentta])

   if loydetyt.count > 0
      @hashid = valitse_jasen(loydetyt[0])
   else
      @hashid = "n"
   end

   luo_jasenlista_alkiot(loydetyt)

   respond_to do |format|
      format.js
   end
end
```

Otetaan vastaan jäsenenpoistamislinkin painallus. Poistettavan jäsenen id otetaan sessiomuuttujasta.

```
def poista_jasen
   session[:toiminto] = "poista_jasen"

   @kerho = Kerho.new
   @kerho.lue_tiedostosta("kelmit")
   @kerho.poista(session[:valittuid])
   @kerho.tallenna
   loydetyt = @kerho.etsi("", 1)

   if loydetyt.count > 0
      @hashid = loydetyt[0].get_tunnusnro
   else
      @hashid = "n"
   end

   luo_jasenlista_alkiot(loydetyt)
   valitse_jasen(loydetyt[0])

   @jasenmaara = @kerho.get_jasenia
   session[:hakuehto] = nil

   respond_to do |format|
      format.js
   end
end
```

Otetaan vastaan Javascript-tiedostosta tullut hashin-muuttuminen. Jos ei ole juuri asetettu hashia jossakin toiminnossa (`session[:toiminto] == ""`), niin päivitetään jäsenen tiedot. Muuten ei tehdä mitään. `@paivita_jasen`-attribuutilla kerrotaan

.js.erb-mallineelle, pitääkö käyttöliittymä päivittää:

```
def paivita_jasen
   if session[:toiminto] == ""
      @paivita_jasen = true
      toiminto_paivita_jasen
   else
      @paivita_jasen = false
   end

   session[:toiminto] = ""

   respond_to do |format|
      format.js
   end
end
```

Asetetaan hashin muuttumisen jälkeen käyttöliittymän päivittämisessä tarvittavat attribuutit:

```
def toiminto_paivita_jasen
   id = params[:hashid]
   @kerho = Kerho.new
   @kerho.lue_tiedostosta("kelmit")
   loydetyt = @kerho.etsi(session[:hakuehto], session[:hakukentta])

   @valittujasen = @kerho.anna_jasen_id(id)
   valitse_jasen(@valittujasen)

   luo_jasenlista_alkiot(loydetyt)
end
```

Otetaan vastaan Javascript-tiedostolta tullut url-hashin asettamiskutsu. Asetetaan tarvittava @hashid-attribuutti, jolla url-hash asetetaan .js.erb-mallineessa. Tämänkään jälkeen ei turhaan enää päivitetä käyttöliittymää hashin muuttumisen takia (session[:toiminto] = "aseta_urlhash"):

```
def aseta_urlhash
   session[:toiminto] = "aseta_urlhash"

   if session[:paramid] != nil
      @hashid = session[:paramid]
   else
      kerho = Kerho.new
      kerho.lue_tiedostosta("kelmit")
      jasenet = kerho.etsi("", "")
      @hashid = jasenet[0].get_tunnusnro
   end

   respond_to do |format|
      format.js
   end
end
```

C.5 Kerhosivun js-mallineet

aseta_kentta.js.erb-mallineessa näytetään virheilmoitus oikean kentän kohdalla (`#virhe<%= escape_javascript(@i) %>`). Virheteksti saadaan ohjaimessa asetetusta attribuutista (`@virhe`). jQueryn avulla otetaan tieto, onko kentälle asetettu virhetyyli, ja muutetaan kentän tyyli virhetekstin ja kentän edellisen tyylin mukaan:

```
$("#virhe<%= escape_javascript(@i) %>").html("<%= escape_javascript
    ( @virhe ) %>");

var onkoVirhetyyli = $("#<%= escape_javascript( @i ) %>").hasClass(
    "virheKentta");

<% if @virhe != nil %>

   if (!onkoVirhetyyli) {
      $("#<%= escape_javascript( @i ) %>").addClass("virheKentta");
   }

<% else %>

   if (onkoVirhetyyli) {
      $("#<%= escape_javascript( @i ) %>").removeClass("virheKentta
          ");
   }

<% end %>
```

nayta_jasentiedot.js.erb-mallineessa asetetaan uusi url-hash (`@hashid`) ja renderöidään uudet jäsentiedot sivulle partialien avulla:

```
window.location.hash = '<%= @hashid %>';

$("#jasentiedot").html("<%= escape_javascript( render(:partial => "
    jasentiedot")) %>");
$("#harrastukset").html("<%= escape_javascript( render(:partial =>
    "harrastukset")) %>");
```

hae_jasenet.js.erb-mallineessa asetetaan myös uusi hash ja renderöidään tarvittavat partialit. Animoidaan myös muuttuneet elementit sekä piilotetaan odotusilmaisin:

```
window.location.hash = '<%= @hashid %>';

$("#jasenValinnat").html("<%= escape_javascript( render(:partial =>
     "jasenlista")) %>");
$("#jasentiedot").html("<%= escape_javascript( render(:partial => "
    jasentiedot")) %>");
$("#harrastukset").html("<%= escape_javascript( render(:partial =>
    "harrastukset")) %>");

$("#jasenValinnat").fadeOut(500);
$("#jasenValinnat").fadeIn(500);
```

```
$("#jasentiedot").fadeOut(500);
$("#jasentiedot").fadeIn(500);

$("#harrastukset").fadeOut(500);
$("#harrastukset").fadeIn(500);

$('#odotusIlmaisin').hide();
```

jasendialogi_lahetys.js.erb-mallineessa toimitaan painetun napin mukaan. Jos on painettu *peruuta*-nappia (`@toiminto == "sulje"`), niin suljetaan dialogi ja renderöidään dialogin sisältö uudelleen, jotta siihen saadaan tyhjät kentät uudella aukaisulla. Jos on painettu *lisäys*-nappia, niin toimitaan virheiden mukaan. Jos on virheitä niin animoidaan virheilmoitus. Jos ei ole virheitä, niin suljetaan dialogi, asetetaan uusi url-hash, renderöidään uusi dialogin sisältö sekä renderöidään ja animoidaan muuttuneet elementit. Päivitetään myös uusi jäsenmäärä, joka saadaan `@jasenmaara`-attribuutista:

```
<% if @toiminto == "sulje" %>
   $.modal.close();
   <!-- Luodaan lomake uudelleen, jotta saadaan uudella aukaisulla
      tyhjat tekstikentat. -->
   $("#uusijasendialog").html("<%= escape_javascript( render(:
      partial => "uusijasenlomake")) %>");
<% else %>

   $("#uusijasenilmoitus").html("<%= escape_javascript( @ilmoitus )
       %>");

   <% if @virheita == true %>

      $("#uusijasenilmoitus").animate({
         backgroundColor: "white",
         color: "#fff"
      }, 500 );

      $("#uusijasenilmoitus").animate({
         backgroundColor: "#aa0000",
         color: "#fff"
      }, 1000 );

   <% else %>

      $.modal.close();
      window.location.hash = '<%= @hashid %>';

      $("#uusijasendialog").html("<%= escape_javascript( render(:
         partial => "uusijasenlomake")) %>");
      <!-- Paivitetaan jasenlista ja jasentiedot sivulle -->
      $("#jasenValinnat").html("<%= escape_javascript( render(:
         partial => "jasenlista")) %>");
      $("#jasentiedot").html("<%= escape_javascript( render(:
         partial => "jasentiedot")) %>");
      $("#harrastukset").html("<%= escape_javascript( render(:
         partial => "harrastukset")) %>");
```

```
      $("#jasenmaara").html("Jasenia: <%= escape_javascript(
         @jasenmaara.to_s ) %>");


      $("#jasenValinnat").fadeOut(500);
      $("#jasenValinnat").fadeIn(500);

      $("#jasentiedot").fadeOut(500);
      $("#jasentiedot").fadeIn(500);

      $("#harrastukset").fadeOut(500);
      $("#harrastukset").fadeIn(500);

   <% end %>
<% end %>
```

poista_jasen.js.erb-mallineessa renderöidään ja animoidaan muuttuneet elementit ja asetetaan uusi url-hash:

```
window.location.hash = '<%= @hashid %>';

<!-- Paivitetaan jasenlista ja jasentiedot sivulle -->
$("#jasenValinnat").html("<%= escape_javascript( render(:partial =>
    "jasenlista")) %>");
$("#jasentiedot").html("<%= escape_javascript( render(:partial => "
   jasentiedot")) %>");
$("#harrastukset").html("<%= escape_javascript( render(:partial =>
   "harrastukset")) %>");
$("#jasenmaara").html("Jasenia: <%= escape_javascript( @jasenmaara.
   to_s ) %>");

$("#jasenValinnat").fadeOut(500);
$("#jasenValinnat").fadeIn(500);

$("#jasentiedot").fadeOut(500);
$("#jasentiedot").fadeIn(500);

$("#harrastukset").fadeOut(500);
$("#harrastukset").fadeIn(500);
```

paivita_jasen.js.erb-mallineessa renderöidään elementit, jos ohjaimella on niin asetettu (`@paivita_jasen == true`):

```
<% if @paivita_jasen == true %>
   $("#jasenValinnat").html("<%= escape_javascript( render(:partial
       => "jasenlista")) %>");
   $("#jasentiedot").html("<%= escape_javascript( render(:partial
      => "jasentiedot")) %>");
   $("#harrastukset").html("<%= escape_javascript( render(:partial
      => "harrastukset")) %>");
<% end %>
```

aseta_urlhash.js.erb-mallineessa asetetaan uusi url-hash:

```
window.location.hash = '<%= @hashid %>';
```

C.6 Jäsensivun html-mallineet

jasensivu.html.erb-mallineessa luodaan jäsensivu. Jäsentietolomake luodaan *cell*-komponentin avulla ja harrastuslomake sekä dialogin sisältö partialien avulla. Tämä sivu käyttää samaa Javascript-tiedostoa kuin kerhosivu.

_harrastuslomake.html.erb-mallineessa luodaan taulukko harrastusten muokkausta varten. Ensin luodaan otsikkorivi harrastuksen kysymyksistä ja sitten rivit harrastusten tiedoille. Tietoriveille luodaan tekstikentät joiden arvoksi laitetaan kyseisen kentän arvo. Tekstikentille luodaan yksilölliset nimet (`id.to_s + -"+ i.to_s`), jotta harrastukset ja niiden kentät tunnistetaan ohjaimessa. Kentälle luodaan javascript-funktiokutsu napin ylösnousemiselle (`:onkeyup => "aseta_arvo ('harrastus', this.value, this.name)"`), jonne välitetään parametreina tieto, että kyseessä on harrastuskenttä, kentän arvo ja kentän nimi. Lisäksi luodaan funktiokutsu kentän aktivoitumiselle (`:onfocus => "aseta_harrastus (this.name)"`), jonne välitetään parametrina kentän nimi. Tekstikentän viereen luodaan paikka virheilmoitukselle, jolle annetaan myös yksilöllinen id (`"virhe" + kentta`):

```
<% if @harrastukset.size > 0 %>
      <caption><b>Harrastukset:</b></caption>
      <tr>
         <% (@harrastukset[0].eka_kentta..@harrastukset[0].
            get_kenttia-1).each do |i| %>
            <td><b><%= @harrastukset[0].get_kysymys(i) %></b></td>
         <% end %>
      </tr>

      <% @harrastukset.each do |harrastus| %>
         <tr>
            <% id = harrastus.get_tunnusnro %>
            <% (harrastus.eka_kentta..harrastus.get_kenttia-1).each
                 do |i| %>

               <% kentta = id.to_s + "-" + i.to_s %>
               <% if i ==  harrastus.eka_kentta %>
                  <% tyyli = "harrastusekakentta" %>
               <% else %>
                  <% tyyli = "harrastuskentta" %>
               <% end %>

               <td>
```

```
                    <%= text_field_tag kentta, harrastus.anna(i),
                        :class => tyyli,:onkeyup => "aseta_arvo('
                        harrastus', this.value, this.name)",
                        :onfocus => "aseta_harrastus(this.name)" %>
                    <span id=<%= "virhe" + kentta %> class="virhe"
                        ></span>
                </td>
            <% end %>
        </tr>
    <% end %>
<% else %>
   <caption><b>Harrastukset:</b></caption>
   <tr><td>Ei harrastuksia!</td></tr>
<% end %>
```

_uusiharrastuslomake.html.erb-mallineessa luodaan harrastusdialogin sisältö samaan tapaan kuin kerhosivun dialogissa. Tässä tekstikentille asetetaan napin ylösnousemiselle Javascript-funktio `aseta_uusiharrastuskentta(this.value, this.name)` ja tekstikentän nimeksi annetaan `"uusiharrastus" + "-" + i.to_s`.

C.7 Jäsensivun Javascript-funktiot

Luodaan Ajax-kutsu valitun harrastuksen asettamiselle. Parametrina saadaan harrastuskentän nimi, joka välitetään ohjaimelle:

```
function aseta_harrastus(valittuKentta) {
   jQuery.ajax({
      url: "aseta_valittuharrastus",
      type: "GET",
      data: {"valittukentta": valittuKentta},
   });
}
```

Luodaan Ajax-kutsu dialogin kentän sisällön asettamiseksi harrastukselle. Tämä funktio toimii samoin kuin `aseta_arvo`-funktio, mutta ohjaimen puolella toimitaan eri tavalla. Tietysti tämänkin toiminnallisuuden olisi voinut toteuttaa tuossa toisessa funtiossa vain antamalla funktiokutsulle erilaisen *"tyyppi"*-parametrin, ja sitten ohjaimen puolella siirtämään suoritus oikealle metodille, mutta tämä nyt tuli tehtyä näin erikseen:

```
function aseta_uusiharrastuskentta(arvo, kentta) {
   var onkoVirhetyyli = $("#" + kentta).hasClass("virheKentta");

   jQuery.ajax({
```

```
      url: "aseta_uusiharrastuskentta",
      type: "GET",
      data: {"arvo": arvo, "kentta": kentta, "onkovirhe":
         onkoVirhetyyli},
   });
}
```

`aseta_arvo`-funktiona käytetään samaa funktiota kuin jäsentietolomakkeessa. Sille viedään siis parametrina tieto, onko kyseessä harrastus- vai jäsentietokenttä.

C.8 Jäsensivun ohjain

Ohjaimen (`jasen_controller.rb`) `jasensivu`-toiminnossa asetetaan sivun latautuessa tarvittavat attribuutit sekä myöhemmin tarvittavat muuttujat sessioon:

```
def jasensivu
   kerho = Kerho.new
   kerho.lue_tiedostosta("kelmit")

   @id = params[:id]
   @jasen = kerho.anna_jasen_id(@id)
   @jasennimi = @jasen.get_nimi
   @lomakejasen = @jasen
   @apuharrastus = Harrastus.new(@id)
   @harrastukset = kerho.anna_harrastukset(@jasen)

   session[:jasenid] = @id
   session[:apuharrastus] = @apuharrastus.to_yaml
   session[:harrastukset] = @harrastukset.to_yaml
   session[:harrastusvirheita] = 0
   session[:uusiharrastusvirheet] = 0
end
```

Otetaan vastaan Ajax-kutsu kentän asettamiselle. Parametrista saadaan kentän tyyppi ja toimitaan sen mukaan. Lopuksi renderöidään `.js.erb`-malline:

```
def aseta_kentta
   if params[:tyyppi] == "jasen"
      aseta_jasenkentta
   else
      aseta_harrastuskentta
   end
   respond_to do |format|
      format.js
   end
end
```

Asetetaan kentän arvo jäsenelle. Kentän arvo ja numero saadaan Ajax-kutsun parametreista:

```
def aseta_jasenkentta
   apujasen = Jasen.new
   arvo = params[:teksti]
   @i = params[:i]

   @virhe = apujasen.aseta(@i.to_i, arvo)
end
```

Asetetaan kentän arvo harrastukselle. Harrastuksen id ja kentän numero saadaan parametrina saadusta kentän nimestä. Sessioon tallennetusta jäsenen harrastuslistasta etsitään oikea harrastus ja tallennetaan asetuksen jälkeen uusi harrastuslista sessioon:

```
def aseta_harrastuskentta
   harrastukset = YAML.load(session[:harrastukset])

   @i = params[:i]
   osat = @i.split('-')
   id = osat[0]
   i = osat[1]

   harrastukset.each do |harrastus|
      if harrastus.get_tunnusnro.to_s == id
         @virhe = harrastus.aseta(i.to_i, params[:teksti])
         tarkista_harrastusvirhe
      break
      end
   end
   session[:harrastukset] = harrastukset.to_yaml
end
```

Tarkistetaan harrastuskentän asetuksessa saatu virhe. Parametrista saadaan tieto, onko kentälle asetettu virhetyyli, ja muutetaan virhemäärää virhetyylin ja virhetekstin mukaan:

```
def tarkista_harrastusvirhe
   onkovirhe = params[:onkovirhe]

   if @virhe == nil && onkovirhe == "true"
      session[:virheita] = session[:virheita] - 1
   elsif @virhe != nil && onkovirhe == "false"
      session[:virheita] = session[:virheita] + 1
   end
end
```

Tallennetaan jäsenen tiedot ja harrastukset kerhoon, jos lomakkeiden kentissä ei ole virheitä. Asetetaan ensin jäsentietolomakkeen kentät uudelleen jäseneen ja tarkistetaan virheet. Kerhon harrastukset korvataan myös uusilla:

```
def tallenna_jasen
```

```
    @kerho = Kerho.new
    @kerho.lue_tiedostosta("kelmit")
    aseta_kentat_jaseneen
    nimikentta_arvo = params[@jasen.eka_kentta.to_s]

    if @virheita == 0 && session[:harrastusvirheita] == 0 &&
       nimikentta_arvo != ""
       @kerho.korvaa_tai_lisaa(@jasen)
       korvaa_kerhon_harrastukset
       @kerho.tallenna
       @ilmoitus = "Jasen tallennettu!"
       @virheita = false
    elsif nimikentta_arvo == ""
       @ilmoitus = "Nimi ei saa olla tyhja!"
       @virheita = true
    else
       @ilmoitus = "Virheellisia syotteita!"
       @virheita = true
    end
    respond_to do |format|
       format.js
    end
end
```

Asetetaan jäsentietolomakkeen kentät jäseneen ja tarkistetaan virheet. Jäsen haetaan kerhosta sen id:n perusteella, joka saadaan sessiomuuttujasta. Virheentarkistuksessa lisätään virheiden määrää kuten kerhosivulla:

```
def aseta_kentat_jaseneen
    @virheita = 0
    @jasen = @kerho.anna_jasen_id(session[:jasenid])

    (@jasen.eka_kentta..@jasen.get_kenttia-1).each do |i|
          @virhe = @jasen.aseta(i, params[i.to_s])
          tarkista_virhe
    end
end
```

Korvataan kerhon harrastukset uusilla. Ensin poistetaan kerhosta vanhat harrastukset ja sitten lisätään sessioon tallennetusta harrastuslistasta uudet harrastukset:

```
def korvaa_kerhon_harrastukset
    harrastukset = @kerho.anna_harrastukset(@jasen)
    uudet_harrastukset = YAML.load(session[:harrastukset])

    harrastukset.each do |harrastus|
       @kerho.poista_harrastus(harrastus)
    end
    uudet_harrastukset.each do |uusi_harrastus|
       @kerho.lisaa(uusi_harrastus)
    end
end
```

Otetaan vastaan näkymältä lähetetty harrastusdialogin napinpainallus ja toimitaan painetun napin mukaan:

```
def harrastusdialogi_lahetys
   painettu_nappi = params[:commit]
   if painettu_nappi == "Peruuta"
      peruuta_harrastuslisays
      @toiminto = "sulje"
   else
      @toiminto = "tallenna"
      lisaa_harrastus
   end
   respond_to do |format|
      format.js
   end
end
```

Peruutusnapilla nollataan tarvittavat muuttujat:

```
def peruuta_harrastuslisays
   @apuharrastus = Harrastus.new(session[:jasenid])
   session[:apuharrastus] = @apuharrastus.to_yaml
   session[:uusiharrastusvirheet] = 0
end
```

Lisäysnapilla lisätään uusi harrastus, jos kentissä ei ole virheitä. Asetetaan `.js.erb`-mallineessa tarvittavat attribuutit:

```
def lisaa_harrastus
   harrastus = Harrastus.new(0)
   alakentta_arvo = params["uusiharrastus" + "-" + harrastus.
      eka_kentta.to_s]

   if session[:uusiharrastusvirheet] == 0 && alakentta_arvo != ""
      lisaa_harrastus_listaan
      @uusiharrastusilmoitus = ""
      @uusiharrastusvirheita = false
      session[:uusiharrastusvirheet] = 0
   elsif alakentta_arvo == ""
      @uusiharrastusilmoitus = "Ala ei saa olla tyhja!"
      @uusiharrastusvirheita = true
   else
      @uusiharrastusilmoitus = "Virheellisia syotteita!"
      @uusiharrastusvirheita = true
   end
end
```

Rekisteröidään uusi harrastus ja lisätään se harrastuslistaan sessiomuuttujaan:

```
def lisaa_harrastus_listaan
   @apuharrastus = YAML.load(session[:apuharrastus])
   @apuharrastus.rekisteroi

   @harrastukset = YAML.load(session[:harrastukset])
```

```
    @harrastukset.push(@apuharrastus)

    session[:harrastukset] = @harrastukset.to_yaml
    @apuharrastus = Harrastus.new(YAML.load(session[:jasen]).
        get_tunnusnro)
    session[:apuharrastus] = @apuharrastus.to_yaml
end
```

Asetetaan harrastusdialogin tekstikentän arvo uudelle harrastukselle. Kenttä ja sen arvo saadaan parametrista:

```
def aseta_uusiharrastuskentta
    @apuharrastus = YAML.load(session[:apuharrastus])
    @kentta = params[:kentta]

    osat = @kentta.split('-')
    @i = osat[1]

    @uusiharrastusvirhe = @apuharrastus.aseta(@i.to_i, params[:arvo
        ])
    session[:apuharrastus] = @apuharrastus.to_yaml
    tarkista_uusiharrastusvirhe
end
```

Tarkistetaan uuden harrastuskentän asetuksessa saatu virhe ja muutetaan virhemäärää kentän tyylin ja virhetekstin mukaan:

```
def tarkista_uusiharrastusvirhe
    onkovirhe = params[:onkovirhe]

    if @uusiharrastusvirhe == nil && onkovirhe == "true"
        session[:uusiharrastusvirheet] = session[:
            uusiharrastusvirheet] - 1
    elsif @uusiharrastusvirhe != nil && onkovirhe == "false"
        session[:uusiharrastusvirheet] = session[:
            uusiharrastusvirheet] + 1
    end
end
```

Poistetaan valittu harrastus sessioon tallennetusta harrastuslistasta. Valitun harrastuksen id saadaan myös sessiomuuttujasta:

```
def poista_harrastus
    @harrastukset = YAML.load(session[:harrastukset])

    @harrastukset.each do |harrastus|
        if harrastus.get_tunnusnro.to_s == session[:
            valittu_harrastusid]
        @harrastukset.delete(harrastus)
        break
        end
    end
    session[:harrastukset] = @harrastukset.to_yaml
```

```
    respond_to do |format|
       format.js
    end
end
```

Otetaan vastaan Ajax-kutsu valitun harrastuksen asettamiseksi. Valitun harrastuksen id saadaan parametrissa olevasta kentän nimestä. Asetetaan kyseinen id sessiomuuttujaan:

```
def aseta_valittuharrastus
    valittukentta = params[:valittukentta]
    osat = valittukentta.split('-')
    valittuid = osat[0]
    session[:valittu_harrastusid] = valittuid

    respond_to do |format|
       format.js
    end
end
```

C.9 Jäsensivun js-mallineet

aseta_kentta.js.erb-malline on täysin samanlainen kuin kerhosivun vastaava.

aseta_uusiharrastuskentta.js.erb-malline on edellisen kaltainen paitsi attribuutit ja virheilmoituspaikan id ovat erinimisiä.

aseta_valittuharrastus.js.erb-malline on täysin tyhjä. Siinä ei siis tarvitse tehdä mitään, mutta kyseinen malline pitää kuitenkin olla kyseisen Ajax-kutsun takia.

harrastusdialogi_lahetys.js.erb-malline toimii samaan tapaan kuin kerhosivun `jasendialogi_lahetys.js.erb`-malline. Tässäkin peruutuksessa suljetaan dialogi ja renderöidään dialogin sisältö uudelleen. Lisäysnapilla tarkistetaan virheet, ja jos on virheitä niin animoidaan virheilmoitus. Jos ei ole virheitä niin renderöidään dialogin sisältö ja harrastustenmuokkauslomake ja suljetaan dialogi sekä animoidaan harrastustenmuokkauslomake:

```
...
<!-- Jos painettu lisaysnappia ja ei ole virheita -->
$.modal.close();
```

```
<!-- Luodaan lomake uudelleen, jotta saadaan uudella aukaisulla
    tyhjat tekstikentat. -->
$("#uusiharrastusdialog").html("<%= escape_javascript( render(:
    partial => "uusiharrastuslomake")) %>");

<!-- Paivitetaan harrastustaulukko sivulle -->
$("#tableHarrastukset").html("<%= escape_javascript( render(:
    partial => "harrastuslomake")) %>");

$("#tableHarrastukset").animate({
   backgroundColor: "yellow"
   }, 500 );

$("#tableHarrastukset").animate({
   backgroundColor: "white"
   }, 500 );

...
```

poista_harrastus.js.erb-mallineessa tehdään edellisen koodin mukainen harrastustaulukon päivitys ja animointi.

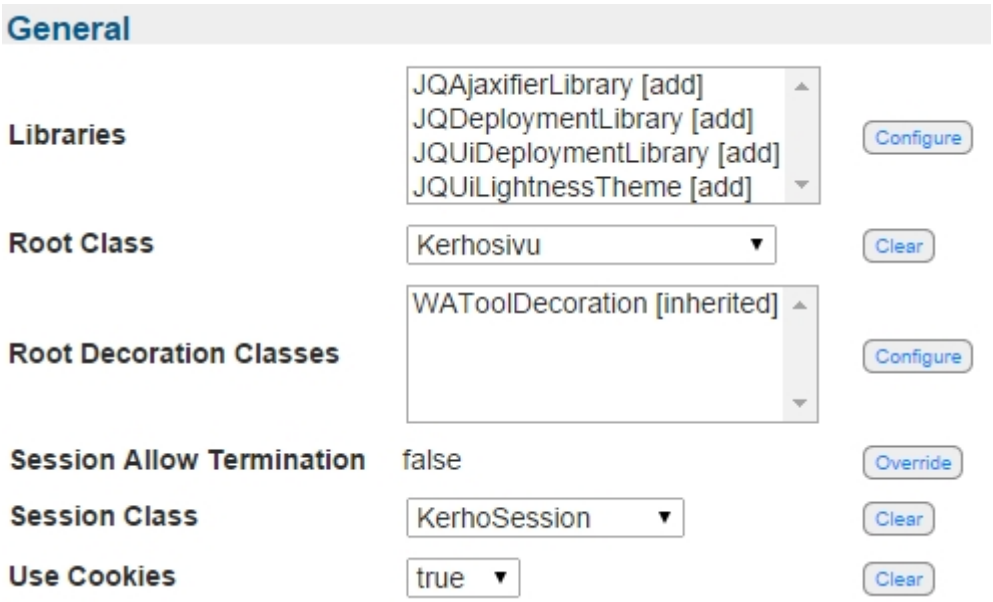
tallenna_jasen.js.erb-mallineessa renderöidään ilmoitus tallennuksen onnistumisesta ja animoidaan ilmoitus sen mukaan onko kentissä ollut virheellisiä syötteitä:

```
$("#tallennusilmoitus").html("<%= escape_javascript( @ilmoitus ) %>
    ");

<% if @virheita == true %>

   $("#tallennusilmoitus").animate({
      backgroundColor: "white",
      color: "#fff"
      }, 500 );

   $("#tallennusilmoitus").animate({
      backgroundColor: "#aa0000",
      color: "#fff"
      }, 1000 );

<% else %>

   $("#tallennusilmoitus").animate({
      backgroundColor: "white",
      color: "#fff"
      }, 500 );

   $("#tallennusilmoitus").animate({
      backgroundColor: "green",
      color: "#fff"
      }, 1000 );

<% end %>
```

D Seaside-kerhosovelluksen luominen

Seasiden Kerho-sovellus koostuu kolmesta `WAComponent`-luokan perivästä Seaside-komponentista: `Kerhosivu`, `Jasensivu` ja `JasenTietoLomake`. Lisäksi siihen kuuluu sessioluokka `KerhoSession` (perii `WASession`-luokan) ja tiedostokirjastoluokka `KerhoFileLibrary` (perii `WAFileLibrary`-luokan). Ajax-toiminnallisuudet luodaan Seasiden jQuery-integraation avulla.

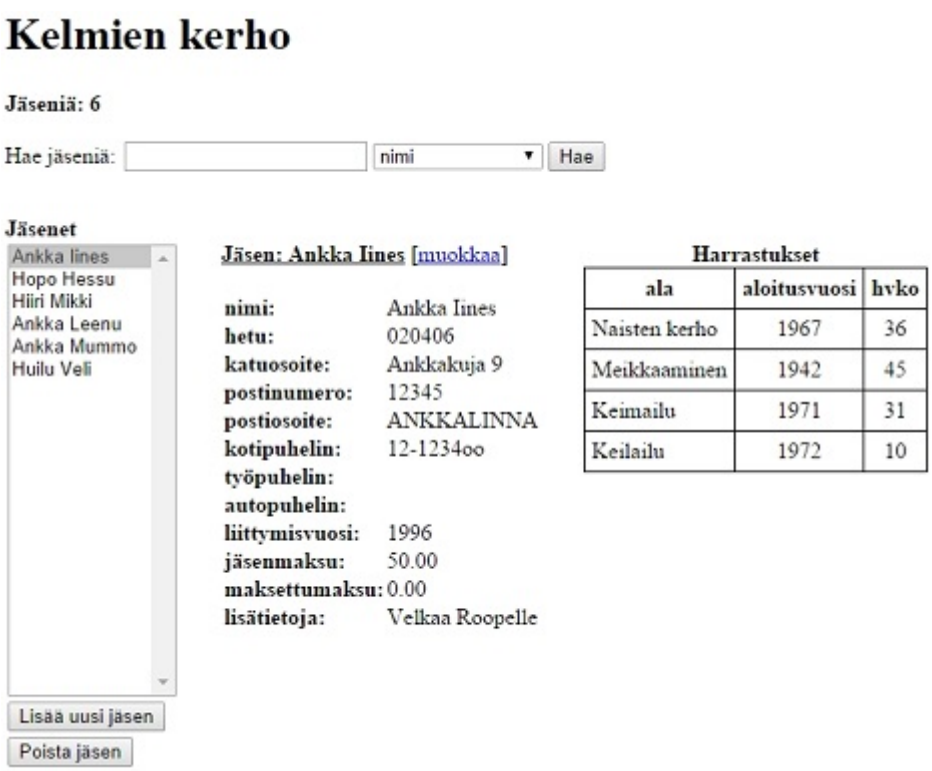
Seasiden konfiguraatiosivulla on luotu *kerho*- ja *jasen*-sovellukset, joiden juurikomponenteiksi `Kerhosivu`- ja `Jasensivu`-komponentit asetettiin. Kuvio 34 esittää *kerho*-sovelluksen konfiguraatiosivulle tehdyt muutokset ja kuvio 35 näyttää kerhosivun käyttöliittymän.



Kuvio 34: Seasiden kerho-sovelluksen konfiguraatiosivun muutokset

D.1 JasenTietoLomake

Tätä luokkaa käytetään sivuilla uudelleenkäytettävänä komponenttina. Se luodaan kuitenkin täysin samalla tavalla kuin normaalit Seaside-komponentit. Alla kuvattujen metodien lisäksi luokkaan kuuluu seuraavat metodit, joita en tässä sen tarkemmin esittele: `asetaJasen:` (asettaa uuden jäsenen luokalle), `getJasen` (palauttaa



Kuvio 35: Seaside-kerhosovelluksen kerhosivun käyttöliittymä

luokan jäsenen), `nimiTyhja` (palauttaa tiedon onko jäsenen nimikenttä tyhjä) ja `onkoVirheita` (palauttaa tiedon onko lomakkeen kentissä virheitä).

Luodaan luokka ja määritellään sille attribuutit (`instanceVariableNames`):

```
WAComponent subclass: #JasenTietoLomake
   instanceVariableNames: 'jasen virheita nimikenttaArvo'
   classVariableNames: ''
   poolDictionaries: ''
   category: 'WWWKerho'
```

Näytetään komponentin sisältö `renderContentOn:`-metodin avulla. `div`-elementin sisältö luodaan omana metodina:

```
renderContentOn: html

html div
   id: #lomakeSisalto;
   with:  [ self luoJasenLomake: html ].
```

Luodaan lomake taulukkona. Taulukon rivit (`tableRow`) luodaan omana metodina:

```
luoJasenLomake: html

html table
   class: 'tableJasenKentat';
   with: [
      jasen ekaKentta to: jasen getKenttia - 1 do: [ :k |
         html tableRow: [ self luoJasenKenttaRivi: html and: k ]]].
```

Luodaan yksi rivi taulukkoon. `tableData`-sivellin määrittää riville sarakkeen. Luodaan sarakkeet kentän kysymykselle, tekstikentälle ja virheilmoitukselle. Virheilmoitukselle annetaan yksilöllinen id, jotta siihen päästään käsiksi jQuery-integraation avulla. Taulukon tekstikentät luodaan omana metodina:

```
luoJasenKenttaRivi: html and: k
| kysymys |

kysymys := jasen getKysymys: k.

html tableData: [ html text: kysymys, ': ' ].

html tableData: [ self luoJasenKentta: html and: kysymys and: k ].

html tableData: [
   html label
      id: 'virhe',kysymys;
      class: 'virheLabel';
      with: ''. ].
```

Luodaan tekstikenttä tietylle jäsenen kysymykselle. Kentälle luodaan kuuntelija näppäimen ylösnousemiselle, jolloin tarkistetaan syöte ja näytetään mahdollinen virhe. `callback`-lohkoissa otetaan kentän arvo ja tieto onko kentälle asetettu virhetyyli. `html jQuery this` hakee kyseisen kentän, `hasClass:` palauttaa tiedon onko kentällä tiettyä tyyliluokkaa määriteltynä ja `value` palauttaa kentän arvon. `script`-lohkossa asetetaan arvo jäsenelle, asetetaan tekstikentän tyyli ja näytetään mahdollinen virhe. Päivitettävät sivun osat saadaan jQuery-integraatiolla elementin id:n avulla (`s jQuery id: 'text',kysymys`). Virheen näyttämisessä päivitetään elementin sisältö (html-koodi) uudella ja elementin renderöinnissä käytetään nyt Ajaxin välittämää renderöijää (`h`) eikä vanhaa html-piirtoaluetta (`html: [ :h | h render: virhe ]`):

```
luoJasenKentta: html and: kysymys and: k
| virhe arvo onkoVirheTyyli |

html textInput
```

```
   id: 'text',kysymys;
   class: 'jasenkentta';
   value: (jasen anna: k);
   onKeyUp: (html jQuery ajax
      callback: [ :value | onkoVirheTyyli := value ] value: ((html
          jQuery this) hasClass: 'virheKentta');
      callback: [ :value | arvo := value ] value: (html jQuery this
          ) value;

      script: [ :s |
         k = jasen ekaKentta
            ifTrue: [ nimikenttaArvo := arvo. ].

         virhe := jasen aseta: k and: arvo.

         virhe = '' & onkoVirheTyyli = 'true'
            ifTrue: [
               virheita := virheita - 1.
               s << (s jQuery id: 'text',kysymys) removeClass: '
                  virheKentta' ].

         virhe ~= '' & onkoVirheTyyli = 'false'
            ifTrue: [
               virheita := virheita + 1.
               s << (s jQuery id: 'text',kysymys) addClass: '
                  virheKentta' ].

         s << (s jQuery id: 'virhe',kysymys) html: [ :h | h render:
             virhe ]]).
```

D.2 Kerhosivu

Luodaan luokka ja määritellään sille attribuutit (`instanceVariableNames`):

```
WAComponent subclass: #Kerhosivu
   instanceVariableNames: 'kerho tallennettavaJasen jasenLista
      tietoLomake'
   classVariableNames: ''
   poolDictionaries: ''
   category: 'WWWKerho'
```

Luokkapuolen metodissa (`Kerhosivu class»#canBeRoot`) kerrotaan Seasidelle, että tämä komponentti haluaa toimia omana sovelluksena [15, s. 245]:

```
canBeRoot

^true
```

Asetetaan sivulle otsikko (selaimeen) ja tyylitiedoston osoite. Tyylitiedosto on siis luotu `KerhoFileLibrary`-luokkaan:

```
updateRoot: anHtmlRoot

super updateRoot: anHtmlRoot.
anHtmlRoot title: kerho getNimi.

anHtmlRoot link
   type: 'text/css';
   beStylesheet;
   addAll;
   url: KerhoFileLibrary  / 'kerho.css';
   yourself.
```

Alustetaan luokka. Luodaan uusi `kerho`-olio ja luetaan sen tiedot tiedostosta. Luodaan myös uusi `JasenTietoLomake`-komponentti:

```
initialize

super initialize.
kerho := Kerho new.
kerho lueTiedostosta: 'kelmit'.
tallennettavaJasen := Jasen new.
tietoLomake := JasenTietoLomake new.
```

Kerrotaan Seasidelle sivulla näytettävät omat komponentit:

```
children

^ Array with: tietoLomake
```

Näytetään sivun sisältö `renderContentOn:`-metodin avulla. Hakuehto ja hakukenttä otetaan sessiosta, jos ne on siihen asetettu ja valittavan jäsenen id *request*-parametrista, jos se löytyy. Sivun eri osien (hakulomake, jäsenvalintalista, jäsentiedot ja harrastukset) luominen on jaettu omiin metodeihin (metodien sisältö luodaan `div`-elementtien sisään):

```
renderContentOn: html
| kentat apuJasen jasenId hakuehto hakukentta |

apuJasen := Jasen new.

self session onkoTallennettu
   ifTrue: [
      hakuehto := self session getHakuehto.
      hakukentta := self session getHakukentta ];
   ifFalse: [
      hakuehto := ''.
      hakukentta := apuJasen ekaKentta ].

jasenLista := kerho etsi: hakuehto and: hakukentta asInteger.
jasenId := self requestContext request at: 'id'.
```

```
jasenId = nil
   ifTrue: [
      jasenLista size > 0
         ifTrue: [
            apuJasen := jasenLista at: 1 ]];
   ifFalse: [ apuJasen := kerho annaJasenId: jasenId ].

html heading with: kerho getNimi.
html paragraph
   id: #labelJasenet;
   class: 'labelJasenet';
   with: 'Jäseniä: ',kerho getJasenia asString.

html div
   class: 'hakuLomake';
   with: [ self luoHakuLomake: html and: hakuehto and: hakukentta
      ].

html paragraph
   class: 'piilotettu';
   id: 'odotusilmaisin';
   with: [
      html text: 'Haetaan jäseniä...'.
      html image url: KerhoFileLibrary / #ajaxloaderGif. ].

html label
   class: 'labelJasenet';
   with: 'Jäsenet'.
html break.

html div
   id: #jasenValintalista;
   class: 'jasenValintalista';
   with: [ self luoJasenValinnat: html and: apuJasen getTunnusNro
      ].

html div
   id: #jasenTiedot;
   class: 'jasenTiedot';
   with: [ self naytaJasenTiedot: html and: apuJasen getTunnusNro.
      ].

html div
   id: #jasenHarrastukset;
   class: 'jasenHarrastukset';
   with: [ self naytaJaseneneHarrastukset: html and: apuJasen
      getTunnusNro. ].
```

Luodaan hakulomake. Tekstikenttään laitetaan parametrina saatu arvo ja hakukenttävalintaelementin (`#selectHae`) alkiot luodaan omana metodina:

```
luoHakuLomake: html and: hakuehto and: hakukentta

html label with: 'Hae jäseniä: '.
html textInput
   id: #textHae;
   with: hakuehto.
html select
   id: #selectHae;
   with: [ self hakuValinnat: html and: hakukentta ].
```

```
self hakuNappi: html.
```

Luodaan alkiot hakukenttävalintaelementille `option`-siveltimen avulla. Alkiolle asetetaan arvoksi kentän numero ja näkyväksi tekstiksi kentän kysymys. Otetaan kentät jäseneltä ja asetetaan parametrina saatu kentän numero valituksi (`selected: true`):

```
hakuValinnat: html and: hakukentta
| apuJasen |

apuJasen := Jasen new.

apuJasen ekaKentta to: apuJasen getKenttia - 1 do: [ :k |
   k asString = hakukentta
      ifTrue: [
         html option
            selected: true;
            value: k;
            with: (apuJasen getKysymys: k) ]
      ifFalse: [
         html option
            value: k;
            with: (apuJasen getKysymys: k) ]]
```

Luodaan hakunappi. Asetetaan sille kaksi klikkaus-tapahtumankäsittelijää. Toisessa asetetaan vain odotusilmaisin näkyviin. Toisen `callback`-lohkoissa otetaan syötetty hakuehto ja valittu hakukenttä. `script`-lohkossa asetetaan haulle viive ja suoritetaan haku. Tallennetaan hakuehto ja hakukenttä myös sessioon. Lopuksi päivitetään jäsenlista ja piilotetaan odotusilmaisin.

```
hakuNappi: html
| hakuEhto hakuKentta viive |

html submitButton
   onClick: (html jQuery: #odotusilmaisin) show;
   onClick: (html jQuery ajax
      callback: [ :value | hakuEhto := value ] value: (html jQuery
         id: #textHae) value;
      callback: [ :value | hakuKentta := value ] value: (html
         jQuery id: #selectHae) value;

      script: [ :s |
         viive := Delay forSeconds: 3.
         viive wait.

         self session tallenna: hakuEhto and: hakuKentta.
         jasenLista := kerho etsi: hakuEhto and: hakuKentta
            asInteger.
         jasenLista size > 0
            ifTrue: [ self paivitaJasenLista: s and: 1. ]
            ifFalse: [ self paivitaJasenLista: s and: 0. ].
```

```
        s << (s jQuery id: #odotusilmaisin) hide ]);
    with: 'Hae'.
```

Päivitetään jäsenvalintalista ja haetaan siitä valittavan jäsenen id. Samalla päivitetään myös valitun jäsenen tiedot ja harrastukset ja kerhon jäsenien määrä sivulle. Elementit myös animoidaan jQuery-integraation avulla:

```
paivitaJasenLista: s and: valittavaJasen
| valittuId |

valittavaJasen = 1
   ifTrue: [ valittuId := (jasenLista at: 1) getTunnusNro. ];
   ifFalse: [
      valittavaJasen = 0
         ifTrue: [ valittuId := '' ] ;
         ifFalse: [ valittuId := (jasenLista at: (jasenLista size))
             getTunnusNro. ]].

s << (s jQuery: #jasenValintalista) html: [ :r | r render: [ self
    luoJasenValinnat: r and: valittuId ]].
s << (s jQuery: #jasenTiedot) html: [ :h | h render: [ self
    naytaJasenTiedot: h and: valittuId ]].
s << (s jQuery: #jasenHarrastukset) html: [ :h | h render: [ self
    naytaJaseneneHarrastukset: h and: valittuId ]].
s << (s jQuery: #labelJasenet) html: 'Jäseniä: ',kerho getJasenia
    asString.

s << (s jQuery: #selectJasen) fadeOut: 400 milliseconds.
s << (s jQuery: #jasenTiedot) fadeOut: 400 milliseconds.
s << (s jQuery: #jasenHarrastukset) fadeOut: 400 milliseconds.
s << (s jQuery: #selectJasen) fadeIn: 400 milliseconds.
s << (s jQuery: #jasenTiedot) fadeIn: 400 milliseconds.
s << (s jQuery: #jasenHarrastukset) fadeIn: 400 milliseconds.
```

Luodaan jäsenvalintaelementti ja jäsenen lisäys- ja poisto-napit. Nämä kaikki luodaan omina metodeina:

```
luoJasenValinnat: html and: valittuId

html div
   with: [
      self jasenValintaLista: html and: valittuId.
      html break.
      self luoUusiJasenDialogi: html.
      html break.
      self poistaNappi: html. ].
```

Luodaan jäsenvalintaelementti. Määritellään sille valinnanvaihtumistapahtuma (`onChange`), jonka `callback`-lohkossa otetaan elementin valittu arvo ja `script`-lohkossa päivitetään jäsenen tiedot ja harrastukset. Valintaelementin alkiot luodaan omana metodina, jolle viedään tieto, mikä jäsen pitää olla valittuna:

```
jasenValintaLista: html and: valittavaJasen
| valittuId |

html select
   id: #selectJasen;
   class: 'selectJasen';
   size: 20;
   onChange: (html jQuery ajax
      callback: [ :value | valittuId := value] value: (html jQuery
         this) value;

      script: [ :s |
         s << (s jQuery: #jasenTiedot) html: [ :r | r render: [
            self naytaJasenTiedot: r and: valittuId ]].
         s << (s jQuery: #jasenHarrastukset) html: [ :h | h render:
              [ self naytaJaseneneHarrastukset: h and: valittuId
             ]]]);

   with: [ self jasenValinnat: html and: valittavaJasen ].
```

Luodaan jäsenvalintaelementin alkiot sen mukaan onko kerhosta löydetty jäseniä: Jos ei ole niin laitetaan vain yksi tyhjä alkio. Asetetaan parametrina saadun id:n omaava jäsen valituksi:

```
jasenValinnat: html and: valittuId

jasenLista size > 0
   ifFalse: [
      html option
         value: '';
         with: '' ];
   ifTrue: [
      jasenLista do: [ :jasen |
         jasen getTunnusNro = valittuId
            ifTrue: [
               html option
                  selected: true;
                  value: jasen getTunnusNro;
                  with: jasen getNimi ];
            ifFalse: [
               html option
                  value: jasen getTunnusNro;
                  with: jasen getNimi ]]]
```

Näytetään valitun jäsenen tiedot taulukkona tai ilmoitus, jos jäseniä ei ole löytynyt. Rivit taulukkoon luodaan omana metodina:

```
naytaJasenTiedot: html and: id
| jasen |

id = ''
   ifFalse: [
      jasen := kerho annaJasenId: id.

      html div
```

```
         with: [
            html label
               class: 'jasenOtsikko';
               with: 'Jäsen: ',jasen getNimi.
            html label with: ' ['.
            html anchor
               url: '/jasen?id=',id;
                  with: 'muokkaa'.
            html label with: ']'.
            html break.
            html break.
            html table
               class: 'tableJasenTiedot';
               with: [ self luoJasenTietoRivit: html and: jasen.
                  ]]];
   ifTrue: [
      html paragraph
         class: 'boldText';
         with: 'Ei löytynyt yhtään jäsentä!'. ].
```

Luodaan rivit jäsentietotaulukkoon. Sarakkeiksi laitetaan kentän kysymys ja sen arvo:

```
luoJasenTietoRivit: html and: jasen
| kysymys arvo |

jasen ekaKentta to: jasen getKenttia - 1 do: [ :k |
   kysymys := jasen getKysymys: k.
   arvo := jasen anna: k.

   html tableRow: [
      html tableData
         class: 'boldText';
         with: kysymys,': '.
      html tableData with: arvo. ]].
```

Näytetään valitun jäsenen harrastukset taulukkona. Annetaan taulukolle otsikko ja luodaan taulukolle harrastusten otsikkorivi (omana metodina). Jokaiselle harrastukselle luodaan taulukkoon oma rivi (omana metodina) tai näytetään tieto, jos harrastuksia ei ole:

```
naytaJaseneneHarrastukset: html and: id
| jasen harrastukset apuHarrastus |

id = ''
   ifFalse: [
      jasen := kerho annaJasenId: id.
      harrastukset :=kerho annaHarrastukset: jasen.

      apuHarrastus := Harrastus new.
      html div: [
         html table
            class: 'tableHarrastukset';
            with: [
               html tableCaption: 'Harrastukset'.
```

```
                self luoHarrastusOtsikkoRivi: html and: apuHarrastus
                    .

                harrastukset size > 0
                   ifTrue: [
                      harrastukset do: [ :harrastus  |
                         self luoHarrastusTietoRivi: html and:
                            harrastus. ]];
                   ifFalse: [
                      html tableRow: [ html tableData colSpan: 3;
                         with: 'Ei harrastuksia!' ]]]]]
```

Luodaan harrastustaulukon otsikkorivi, jonka sarakkeiksi laitetaan harrastuksen kenttien kysymykset:

```
luoHarrastusOtsikkoRivi: html and: apuHarrastus

html tableRow
   class: 'boldText';
   with: [
      apuHarrastus ekaKentta to: apuHarrastus getKenttia - 1 do: [
         :k |
         html tableData: [
            html label with: (apuHarrastus getKysymys: k). ]]].
```

Luodaan harrastustaulukon harrastusrivi, jonka sarakkeisiin laitetaan harrastuksen kenttien arvot. Ensimmäisen kentän tyyli asetetaan erilaiseksi (teksti tasataan vasemmalle):

```
luoHarrastusTietoRivi: html and: harrastus
| tdClass |

html tableRow: [
   harrastus ekaKentta to: harrastus getKenttia - 1 do: [ :k |

      k = harrastus ekaKentta
         ifTrue: [ tdClass := 'tableEka' ];
         ifFalse: [ tdClass := '' ].

      html tableData
         class: tdClass;
         with: [
            html label with: (harrastus anna: k). ]]].
```

Luodaan dialogi uuden jäsenen lisäämiselle sekä sille aukaisunappi (omana metodina). Dialogin sisällöksi laitetaan `JasenTietoLomake`-komponentti ja paikka virheilmoitukselle. `script`-attribuutilla luodaan dialogi jQuery-integraation avulla. Määritetään sille parametrit `autoOpen: false`, jolloin se ei aukea aina sivun latautuessa ja `modal: true`, jolla saadaan dialogi modaaliseksi. Luodaan dialogille

kaksi nappia, joista toinen vain sulkee dialogin. Toiselle napille luodaan Ajax-skripti, jossa tarkistetaan lomake, ja tallennetaan uusi jäsen kerhoon, jos lomakkeessa ei ole virheitä. Tällöin myös päivitetään jäsenlista ja suljetaan dialogi. Jos lomakkeessa on virheitä niin näytetään virheilmoitus, joka animoidaan näkymään vain hetken aikaa:

```
luoUusiJasenDialogi: html

html div
   id: #uusiJasenDialog;
   script:
      (html jQuery new dialog
         title: 'Lisää uusi jäsen';
         autoOpen: false;
         width: 500;
         height: 600;
         modal: true;
         addButton: 'Peruuta' do: (html jQuery this dialog close);
         addButton: 'Lisää uusi jäsen'
            do: (html jQuery ajax
               script: [ :s |
                  tietoLomake nimiTyhja
                     ifTrue: [
                        s << (s jQuery: #virheIlmoitus) html: [ :r
                            | r render: 'Nimi ei saa olla tyhjä!' ].
                        s << (s jQuery: #virheIlmoitus) fadeIn:
                            1000 milliseconds.
                        s << (s jQuery: #virheIlmoitus) fadeOut:
                            1500 milliseconds ];
                     ifFalse: [
                        tietoLomake onkoVirheita
                           ifTrue: [
                              s << (s jQuery: #virheIlmoitus) html:
                                   [ :r | r render: 'Virheellisiä sy
                                  ötteitä!' ].
                              s << (s jQuery: #virheIlmoitus)
                                  fadeIn: 1000 milliseconds.
                              s << (s jQuery: #virheIlmoitus)
                                  fadeOut: 1500 milliseconds ];
                           ifFalse: [
                              tallennettavaJasen := tietoLomake
                                  getJasen.
                              self tallennaUusiJasen.
                              jasenLista := kerho etsi: '' and: ''.
                              self paivitaJasenLista: s and: -1.
                              s << (s jQuery: #labelJasenet) html:
                                  [ :r | r render: 'Jäseniä: ',kerho
                                   getJasenia asString ].
                              s << (s jQuery: #uusiJasenDialog)
                                  dialog close ]]]));
   with: [
      html render: tietoLomake.
      html paragraph
         id: #virheIlmoitus;
         class: 'virheIlmoitus';
         with: 'Virheellisiä syötteitä!' ].

self dialoginAukaisuNappi: html.
```

Luodaan dialoginaukaisunappi. Sen toisessa `onClick`-metodissa suoritetaan Ajax-skripti, jossa asetetaan `JasenTietoLomake`-komponenttiin tyhjä jäsen ja päivitetään kyseisen komponentin sisältö (pitää tehdä näin koska muuten tyhjän jäsenen tiedot eivät päivity lomakkeeseen). Toisessa `onClick`-metodissa avataan dialogi:

```
dialoginAukaisuNappi: html

html submitButton
   onClick: (html jQuery ajax
      script: [ :s |
         tallennettavaJasen := Jasen new.
         tietoLomake asetaJasen: tallennettavaJasen.
         "Tässä pitää päivittää tyhjä jäsen lomakkeeseen"
         s << (s jQuery: #lomakeSisalto) html: [ :r | r render: [
            tietoLomake luoJasenLomake: r ]]]);

   onClick: (html jQuery: #uusiJasenDialog) dialog open;
   with: 'Lisää uusi jäsen'
```

Tallennetaan jäsen kerhoon:

```
tallennaUusiJasen

tallennettavaJasen rekisteroi.
kerho korvaaTaiLisaa: tallennettavaJasen.
kerho tallenna.
```

Luodaan jäsenenpoistonappi. Poisto varmistetaan `confirm`-attribuutin avulla. `callback`-lohkossa otetaan valittu jäsen jäsenvalintaelementiltä. `script`-lohkossa poistetaan valittu jäsen kerhosta ja päivitetään jäsenvalintalista:

```
poistaNappi: html
| valittuId |

html submitButton
   onClick: (html jQuery ajax
      confirm: 'Poistetaanko valittu jasen?';
      callback: [ :value | valittuId := value ] value: (html jQuery
         : #selectJasen) value;

      script: [ :s |
         jasenLista remove: (kerho annaJasenId: valittuId).
         kerho poista: valittuId.
         kerho tallenna.
         s << (s jQuery: #labelJasenet) html: [ :r | r render: 'Jä
            seniä: ',kerho getJasenia asString ].
         self paivitaJasenLista: s and: 1 ]);
   with: 'Poista jäsen'.
```

D.3 Jasensivu

Tämäkin komponentti sisältää `Kerhosivu`-komponentin tapaan `updateRoot:`- ja `children`-metodit ja luokkapuolen `canBeRoot`-metodissa kerrotaan, että sivu toimii omana sovelluksena.

Luodaan luokka ja määritellään sille attribuutit:

```
WAComponent subclass: #Jasensivu
   instanceVariableNames: 'jasenId kerho jasen tallennettavaJasen
      virheita harrastukset tallennettavaHarrastus tietoLomake
      harrastuslomakeVirheita uusiharrastusVirheita'
   classVariableNames: ''
   poolDictionaries: ''
   category: 'WWWKerho'
```

Alustetaan luokka. Luodaan uusi `kerho`-olio ja `JasenTietoLomake`-komponentti:

```
initialize

super initialize.
kerho := Kerho new.
kerho lueTiedostosta: 'kelmit'.
jasenId  := ''.
virheita := 0.
harrastuslomakeVirheita := 0.
tietoLomake := JasenTietoLomake new.
```

Näytetään sivu sisältö. Otetaan jäsenen id *request*-parametrista ja asetetaan sillä saatu jäsen `JasenTietoLomake`-komponenttiin. `JasenTietoLomake` lisätään `div`-elementin sisään, johon tulee myös linkki takaisin kerhoon ja tallennusnappi (luodaan omana metodina). Harrastusten muokkauslomake tulee toiseen `div`-elementtiin ja myös se luodaan omana metodina:

```
renderContentOn: html

jasenId := self requestContext request at: 'id' .
jasen := kerho annaJasenId: jasenId.
tietoLomake asetaJasen: jasen.

html heading with: jasen getNimi.
html div
   id: #ilmoitus;
   class: 'ilmoitus'.
html div
   id: #jasenLomake;
   class: 'jasenTietoLomake';
   with: [
      html render: tietoLomake.
      html break.
```

```
      html anchor
         url: '/kerho?id=',jasenId;
         with: 'Takaisin kerhoon'.
      html space.
      self tallennusNappi: html ].
html div
   id: #harrastusLomake;
   with: [ self naytaHarrastusLomake: html. ].
```

Luodaan tallennusnappi. `script`-lohkossa tarkistetaan virheet `JasenTietoLomake`-komponentilta ja harrastusten muokkauslomakkeelta ja näytetään ilmoitus tallennuksen onnistumisesta animaation avulla. Tallennetaan jäsen kerhoon, jos ei ole virheitä:

```
tallennusNappi: html

html submitButton
   onClick: (html jQuery ajax
      script: [:s  |
         tietoLomake nimiTyhja
            ifTrue: [
               s << (s jQuery: #ilmoitus) html: [ :r | r render: [
                  self naytaIlmoitus: r and: 'Nimi ei saa olla tyhj
                  ä!' ]].
               s << (s jQuery: #ilmoitus) hide: 400 milliseconds.
               s << (s jQuery: #ilmoitus) show: 400 milliseconds.
                  ];
            ifFalse: [
               harrastuslomakeVirheita > 0 | tietoLomake
                  onkoVirheita
                  ifFalse: [
                     kerho korvaaTaiLisaa: jasen.
                     kerho tallenna.
                     s << (s jQuery: #ilmoitus) html: [ :r | r
                        render: [ self naytaIlmoitus: r and: 'Jäsen
                         tallennettu!' ]].
                     s << (s jQuery: #ilmoitus) hide: 400
                        milliseconds.
                     s << (s jQuery: #ilmoitus) show: 400
                        milliseconds. ];
                  ifTrue: [
                     s << (s jQuery: #ilmoitus) html: [ :r | r
                        render: [ self naytaIlmoitus: r and: '
                        Virheellisiä syötteitä' ]].
                     s << (s jQuery: #ilmoitus) hide: 400
                        milliseconds.
                     s << (s jQuery: #ilmoitus) show: 400
                        milliseconds. ]]]);
   with: 'Tallenna jäsen'.
```

Näytetään ilmoitus tallennuksen onnistumisesta. Tässä käytetään jQuery UI:n ilmoitustyylejä:

```
naytaIlmoitus: html and: teksti
```

```
harrastuslomakeVirheita > 0 | tietoLomake onkoVirheita |
    tietoLomake nimiTyhja
    ifFalse: [
       html div
          class: 'ui-corner-all ui-state-highlight';
          with: [
             html span class: 'ui-icon ui-icon-info'.
             html text: teksti. ]];
    ifTrue: [
        html div
          class: 'ui-corner-all ui-state-error';
          with: [
             html span class: 'ui-icon ui-icon-alert'.
             html text: teksti. ]].
```

Luodaan harrastusten muokkauslomake taulukkona ja sen alle harrastusten lisäys- ja poistonappi. Poistonappi luodaan vain, jos taulukossa on harrastuksia:

```
naytaHarrastusLomake: html
| apuHarrastus |

harrastukset := kerho annaHarrastukset: jasen.

harrastukset size > 0
    ifTrue: [
       harrastuslomakeVirheita := 0.
       apuHarrastus := harrastukset at: 1.
       html div
          id: #harrastusLomake;
          class: 'harrastusLomake';
          with: [
             html table
                class: 'tableHarrastukset';
                with: [
                   html tableCaption: 'Harrastukset'.
                   self luoHarrastusOtsikkoRivi: html and:
                      apuHarrastus.

                   harrastukset do: [ :harrastus  |
                       self luoHarrastusKenttaRivit: html and:
                          harrastus. ]].
          html break.
          self luoUusiHarrastusDialogi: html.
          self poistaHarrastusNappi: html. ]];
    ifFalse: [
       html div
          class: 'harrastusLomake';
          with: [
             html paragraph
                class: 'boldText';
                with: 'Harrastukset'.
             html paragraph with: 'Ei harrastuksia!'.
             self luoUusiHarrastusDialogi: html. ]].
```

Harrastustaulukon otsikkorivi luodaan kuten kerhosivulla. Kenttäriveille laitetaan tekstikentät harrastusten arvoilla ja niiden viereen paikat virheilmoituksille. Virheilmoituslabeleille annetaan yksilölliset id:t, jotta niihin päästään käsiksi jQuery-inte-

graation avulla:

```
luoHarrastusKenttaRivit: html and: harrastus
| id |

id := harrastus getId.

html tableRow: [
   harrastus ekaKentta to: harrastus getKenttia - 1 do: [ :k |
      html tableData: [
         self luoHarrastusKentta: html and: harrastus and: k and:
            id.

         html label
            id: 'virhe',id,(harrastus getKysymys: k);
            class: 'virheLabel';
            with: ''. ]]]
```

Luodaan kenttä tietylle harrastuksen kysymykselle kuten jäsentietolomakkeessa. Annetaan sille yksilöllinen id (`id,(harrastus getKysymys: k)`). Määritellään sille tapahtuma myös kentän aktivoitumiselle (`onFocus`). Kentän aktivoituessa (esimerkiksi kun sitä klikataan) asetetaan kyseisen kentän harrastus attribuuttiin harrastuksen poistoa varten:

```
onFocus: (html jQuery ajax
   script: [:s | tallennettavaHarrastus := harrastus. ]).
```

Dialogin luominen tehdään kuten kerhosivulla, paitsi sen sisältämä lomake luodaan itse eikä käytetä valmista komponenttia. Lomake luodaan `JasenTietoLomake`-komponentin lomakkeen tapaan.

Dialogin aukaisunapissa luodaan uusi harrastus jäsenelle. Tässä pitää myös vielä päivittää lomakkeen sisältö, jotta uuden tyhjän harrastuksen kentät näkyvät:

```
dialogiAukaisuNappi: html

html submitButton
   onClick:
      (html jQuery ajax
         script: [ :s |
            uusiharrastusVirheita := 0.
            tallennettavaHarrastus := Harrastus forJasen: jasenId.
            s << (s jQuery: #uusiHarrastusLomake) html: [ :r | r
               render: [ self luoHarrastusLomakeTaulu: r ]]]);
   onClick: (html jQuery: #uusiHarrastusDialog) dialog open;
   with: 'Lisää uusi harrastus'.
```

E Vaadin-kerhon lähdekoodit

E.1 Vaadinkerho.java

```
package vaadinkerho;

import com.vaadin.annotations.Theme;
import com.vaadin.server.VaadinRequest;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;

@SuppressWarnings("serial")
@Theme("vaadinkerho")
public class Vaadinkerho extends UI {

   @Override
   protected void init(VaadinRequest request) {
      final VerticalLayout layout = new VerticalLayout();
      layout.setMargin(true);
      setContent(layout);
      layout.setSizeFull();

      String id = request.getParameter("id");

      Kerhosivu sivu = new Kerhosivu(id);
      layout.addComponent(sivu);
   }
}
```

E.2 Kerhosivu.java

```
package vaadinkerho;

import java.util.ArrayList;
import java.util.List;

import kerho.Harrastus;
import kerho.Jasen;
import kerho.Kerho;
import kerho.SailoException;

import org.vaadin.dialogs.ConfirmDialog;
import org.vaadin.jouni.animator.AnimatorProxy;
import org.vaadin.jouni.animator.shared.AnimType;

import com.vaadin.annotations.AutoGenerated;
import com.vaadin.data.Property.ValueChangeEvent;
import com.vaadin.data.Property.ValueChangeListener;
import com.vaadin.server.ExternalResource;
import com.vaadin.server.Page;
import com.vaadin.server.VaadinSession;
import com.vaadin.server.Page.UriFragmentChangedEvent;
import com.vaadin.server.Page.UriFragmentChangedListener;
import com.vaadin.shared.ui.label.ContentMode;
import com.vaadin.ui.Alignment;
import com.vaadin.ui.Button;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.CustomComponent;
import com.vaadin.ui.FormLayout;
import com.vaadin.ui.GridLayout;
```

```
import com.vaadin.ui.HorizontalLayout;
import com.vaadin.ui.Label;
import com.vaadin.ui.Link;
import com.vaadin.ui.ListSelect;
import com.vaadin.ui.NativeSelect;
import com.vaadin.ui.ProgressBar;
import com.vaadin.ui.Table;
import com.vaadin.ui.TextField;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;
import com.vaadin.ui.Window;

public class Kerhosivu extends CustomComponent implements Button.
    ClickListener {

    /*- VaadinEditorProperties={"grid":"RegularGrid,20","showGrid":
        true,"snapToGrid":true,"snapToObject":true,"movingGuides":
        false,"snappingDistance":10} */

    @AutoGenerated
    private VerticalLayout mainLayout;
    @AutoGenerated
    private VerticalLayout sivuLayout;
    @AutoGenerated
    private HorizontalLayout sisaltoLayout;
    @AutoGenerated
    private VerticalLayout harrastusLayout;
    @AutoGenerated
    private Label labelHarrastukset;
    @AutoGenerated
    private VerticalLayout jasenLayout;
    @AutoGenerated
    private VerticalLayout jasenetLayout;
    @AutoGenerated
    private Button buttonPoista;
    @AutoGenerated
    private Button buttonLisaaJasen;
    @AutoGenerated
    private ListSelect listJasenet;
    @AutoGenerated
    private Label labelJasenLista;
    @AutoGenerated
    private HorizontalLayout hakuLayout;
    @AutoGenerated
    private Button buttonHaku;
    @AutoGenerated
    private NativeSelect selectHaku;
    @AutoGenerated
    private TextField textHaku;
    @AutoGenerated
    private Label labelHaku;
    @AutoGenerated
    private Label labelJasenia;
    @AutoGenerated
    private Label labelNimi;
    private Kerho kerho;
    private List<Jasen> jasenLista;
    private List<Harrastus> harrastusLista;
    private Jasen apuJasen;
    private Harrastus apuHarrastus;
    private Table tableHarrastukset;
    private List<Object> tauluObjektiLista;
    private Window uusiJasenWindow;
    private VerticalLayout dialogiLayout;
    private ProgressBar odotusIlmaisin;
```

```
private AnimatorProxy proxy;
private JasenTietoLomake tietoLomake;
private Label labelVirhe;

private static final long serialVersionUID = 1L;
/**
 * The constructor should first build the main layout, set the
 * composition root and then do any custom initialization.
 *
 * The constructor will not be automatically regenerated by the
 * visual editor.
 */
public Kerhosivu(String id) {
   buildMainLayout();
   setCompositionRoot(mainLayout);

   proxy = new AnimatorProxy();
   mainLayout.addComponent(proxy);

   odotusIlmaisin = new ProgressBar();
   odotusIlmaisin.setIndeterminate(true);
   odotusIlmaisin.setEnabled(true);

   kerho = KerhoBean.getKerho();
   try {
      kerho.lueTiedostosta("kelmit");
   } catch (SailoException e) {
      e.printStackTrace();
   }

   apuJasen = new Jasen();
   apuHarrastus = new Harrastus();

   asetaNimiJaJasenmaara();
   luoHakukentat();

   listJasenet.setNullSelectionAllowed(false);
   listJasenet.setImmediate(true);
   listJasenet.setRows(20);

   buttonHaku.addClickListener(this);
   buttonLisaaJasen.addClickListener(this);
   buttonPoista.addClickListener(this);

   lisaaKuuntelijat();
   haeJasenetJaPaivitaLista(id);
}


/**
 * Hakee jäsenet kerhosta ja lisää ne valintalistaan.
 *
 * Ottaa hakuehdon ja hakukentan sessiosta jos ne on asetettu,
 * muuten hakee kaikki jäsenet.
 *
 * Jos request parametrina on id, asetetaan se valituksi.
 *
 * Jos urliin on asetettu urifragmentti, asetetaan valituksi
 * kyseinen id.
 *
 * Jos request parametri tai urifragmentti ei ole oikeaa muotoa
    id:ksi,
 * tai jos kumpaakaan ei ole asetettu,
 * asetetaan valituksi listan ensimmäinen jäsen.
 *
```

```
     * @param id Mikä jäsen asetetaan valituksi (request parametri).
     */
    public void haeJasenetJaPaivitaLista(String id) {
       String haku = "";
       int kentta = 0;
       String uriFragment = Page.getCurrent().getUriFragment();

       if (VaadinSession.getCurrent().getAttribute("hakuehto") !=
          null) {
          haku = VaadinSession.getCurrent().getAttribute("hakuehto")
             .toString();
          kentta = Integer.parseInt( VaadinSession.getCurrent().
             getAttribute("hakukentta").toString() );
          jasenLista = (List<Jasen>) kerho.etsi("*" + haku + "*",
             kentta);
          textHaku.setValue(haku);
          selectHaku.select(kentta);
       }
       else jasenLista = (List<Jasen>) kerho.etsi("*", apuJasen.
          ekaKentta());

       if (jasenLista.size() > 0) {
          if (uriFragment != null) {
             try {
                int jasenId = Integer.parseInt(uriFragment);
                paivitaJasenlista(jasenId);
             } catch (NumberFormatException e) {
                paivitaJasenlista(jasenLista.get(0).getTunnusnro());
             }
          }
          else if (id != null) {
             try {
                int jasenId = Integer.parseInt(id);
                paivitaJasenlista(jasenId);
             } catch (NumberFormatException e) {
                paivitaJasenlista(jasenLista.get(0).getTunnusnro());
             }
          }
          else paivitaJasenlista(jasenLista.get(0).getTunnusnro());
       }
       else paivitaJasenlista(0);
    }


    /**
     * Kuuntelija sivun nappien klikkauksille.
     */
    @Override
    public void buttonClick(ClickEvent event) {
       // Hakunappi
       if (event.getButton() == buttonHaku) {
          int hakukentta = Integer.parseInt(selectHaku.getValue().
             toString());
          String hakuehto = textHaku.getValue();
          haeJasenet(hakuehto, hakukentta);
       }
       // Jäsenen lisäysnappi
       else if (event.getButton() == buttonLisaaJasen) {
          avaaUusiJasenIkkuna();
       }
       // Jäsenen poistonappi
       else if (event.getButton() == buttonPoista) {
          poistaJasen();
       }
    }
```

```
/**
 * Lisää kuuntelijat jäsenlistalle ja uriFragmentin
    vaihtumiselle.
 */
@SuppressWarnings("serial")
public void lisaaKuuntelijat() {
   Page.getCurrent().addUriFragmentChangedListener(
         new UriFragmentChangedListener() {
            public void uriFragmentChanged(
                  UriFragmentChangedEvent source) {
               int id = Integer.parseInt(source.getUriFragment()
                  );
               listJasenet.setValue(id);
               naytaJasenTiedot(id);
            }
         });


   listJasenet.addValueChangeListener(new ValueChangeListener()
      {
      private static final long serialVersionUID = 1L;
      @Override
      public void valueChange(ValueChangeEvent event) {
         if (listJasenet.getValue() != null) {
            Page.getCurrent().setUriFragment(listJasenet.
               getValue().toString());
            naytaJasenTiedot(Integer.parseInt(listJasenet.
               getValue().toString()));
         }
      }
   });
}


/**
 * Poistaa valitun jäsenen kerhosta.
 * Varmistaa poiston dialogilla.
 */
@SuppressWarnings("serial")
public void poistaJasen() {
   final int jasenId = Integer.parseInt(listJasenet.getValue().
      toString());
   final Jasen jasen = kerho.annaJasenId(jasenId);

   ConfirmDialog.show(UI.getCurrent(), "Jäsenen poisto", "
      Poistetaanko jäsen: " + jasen.getNimi() + "?",
         "Kyllä", "Ei", new ConfirmDialog.Listener() {

      public void onClose(ConfirmDialog dialog) {
         if (dialog.isConfirmed()) {
            kerho.poista(jasenId);
            try {
               kerho.talleta();
            } catch (SailoException e) {
               e.printStackTrace();
            }
            jasenLista.remove(jasen);

            labelJasenia.setValue("<h2>Jäseniä: " + Integer.
               toString(kerho.getJasenia()) + "</h2>");

            if (jasenLista.size() > 0) {
```

```
                paivitaJasenlista(jasenLista.get(0).getTunnusnro
                    ());
            }
            else paivitaJasenlista(0);
         }
      }
   });
}


/**
 * Hakee jäsenet kerhosta ja päivittää jäsenlistan.
 */
public void haeJasenet(String hakuehto, int hakukentta) {
   Label labelOdota = new Label("Jäseniä haetaan...");
   hakuLayout.addComponent(labelOdota);
   hakuLayout.addComponent(odotusIlmaisin);
   UI.getCurrent().push();

   try {
      Thread.sleep(3000);
   } catch(InterruptedException ex) {
      Thread.currentThread().interrupt();
   }

   hakuLayout.removeComponent(labelOdota);
   hakuLayout.removeComponent(odotusIlmaisin);

   jasenLista = (List<Jasen>) kerho.etsi("*" + hakuehto + "*",
      hakukentta);

   if (jasenLista.size() > 0) {
      paivitaJasenlista(jasenLista.get(0).getTunnusnro());
   }
   else paivitaJasenlista(0);

   VaadinSession.getCurrent().setAttribute("hakuehto", hakuehto)
      ;
   VaadinSession.getCurrent().setAttribute("hakukentta", Integer
      .toString(hakukentta));
}


/**
 * Asettaa otsikoihin kerhon nimen ja jäsenmäärän.
 */
public void asetaNimiJaJasenmaara() {
   labelNimi.setContentMode(ContentMode.HTML);
   labelJasenia.setContentMode(ContentMode.HTML);
   labelNimi.setValue("<h1>" + kerho.getNimi() + "</h1>");
   labelJasenia.setValue("<h2>Jäseniä: " + Integer.toString(
      kerho.getJasenia()) + "</h2>");
}


/**
 * Luo hakuvalintakentät.
 */
public void luoHakukentat() {
   selectHaku.setNullSelectionAllowed(false);
   for (int i = apuJasen.ekaKentta(); i < apuJasen.getKenttia();
       i++) {
      selectHaku.addItem(i);
      selectHaku.setItemCaption(i, apuJasen.getKysymys(i));
   }
```

```
        selectHaku.setValue(apuJasen.ekaKentta());
    }


    /**
     * Päivittää jäsenvalintalistan.
     * @param jasenId mikä jäsen valitaan listasta.
     */
    public void paivitaJasenlista(int jasenId) {
        listJasenet.removeAllItems();

        if (jasenLista.size() > 0) {
            for (Jasen jasen : jasenLista) {
                listJasenet.addItem(jasen.getTunnusnro());
                listJasenet.setItemCaption(jasen.getTunnusnro(), jasen.
                    getNimi());
            }
            listJasenet.setValue(jasenId);
        }
        else {
            jasenLayout.removeAllComponents();
            harrastusLayout.removeAllComponents();
            Label labelHuomautus = new Label("<b>Ei löytynyt yhtään
                jäsentä!</b>");
            labelHuomautus.setContentMode(ContentMode.HTML);
            jasenLayout.addComponent(labelHuomautus);
        }

        proxy.animate(listJasenet, AnimType.FADE_IN).setDuration(500)
            .setDelay(350);
        proxy.animate(jasenLayout, AnimType.FADE_IN).setDuration(500)
            .setDelay(350);
        proxy.animate(harrastusLayout, AnimType.FADE_IN).setDuration
            (500).setDelay(350);
    }


    /**
     * Näyttää valitun jäsenen tiedot.
     * @param id näytettävän jäsenen tunnusNro
     */
    public void naytaJasenTiedot(int id) {
        jasenLayout.removeAllComponents();
        Jasen jasen = kerho.annaJasenId(id);

        Label labelJasenOtsikko = new Label("Jäsen: " + jasen.getNimi
            ());
        labelJasenOtsikko.setStyleName("labelJasenOtsikko");
        jasenLayout.addComponent(labelJasenOtsikko);

        Link linkJasen = new Link("[Muokkaa]",
              new ExternalResource("/vaadinKerho/jasensivu/?id=" + id
                  ));
        jasenLayout.addComponent(linkJasen);

        GridLayout tietoLayout = new GridLayout(2, apuJasen.
            getKenttia());
        jasenLayout.addComponent(tietoLayout);

        for (int i = apuJasen.ekaKentta(); i < apuJasen.getKenttia();
             i++) {
            Label labelKysymys = new Label("<b>" + jasen.getKysymys(i)
                + ":</b> ");
            Label labelArvo = new Label(jasen.anna(i));
            labelKysymys.setContentMode(ContentMode.HTML);
```

```
            tietoLayout.addComponent(labelKysymys, 0, i);
            tietoLayout.addComponent(labelArvo, 1, i);
        }
        naytaHarrastukset(jasen);
    }


    /**
     * Näyttää valitun jäsenen harrastukset taulukossa.
     * @param id jäsenen tunnusNro, keneltä harrastukset näytetään.
     */
    public void naytaHarrastukset(Jasen jasen) {
        if (harrastusLayout.getComponentIndex(tableHarrastukset) !=
            -1) {
            harrastusLayout.removeComponent(tableHarrastukset);
        }

        harrastusLista = kerho.annaHarrastukset(jasen);
        tauluObjektiLista = new ArrayList<Object>();

        tableHarrastukset = new Table();
        tableHarrastukset.setPageLength(harrastusLista.size());
        harrastusLayout.addComponent(tableHarrastukset);

        luoTaulukkoOtsikot();

        if (harrastusLista.size() > 0) luoTaulukkoTietorivit();
    }


    /**
     * Luo taulukon otsikkorivin.
     */
    public void luoTaulukkoOtsikot() {
        // Määritellään taulukon sarakkeet ja annetaan niille otsikot
            .
        if (harrastusLista.size() > 0) {
            for (int i = apuHarrastus.ekaKentta(); i < apuHarrastus.
                getKenttia(); i++) {
                if (apuHarrastus.getKysymys(i).equals("h/vko") ||
                    apuHarrastus.getKysymys(i).equals("aloitusvuosi")) {
                    tableHarrastukset.addContainerProperty(apuHarrastus.
                        getKysymys(i), Integer.class,  null);
                }
                else {
                    tableHarrastukset.addContainerProperty(apuHarrastus.
                        getKysymys(i), String.class,  null);
                }
            }
        }
        else {
            tableHarrastukset.addContainerProperty("Ei harrastuksia!",
                FormLayout.class,  null);
        }
    }


    /**
     * Luo taulukkoon harrastustietorivit.
     */
    public void luoTaulukkoTietorivit() {
        int riviNro = 1;

        // Lisätään harrastukset taulukkoon.
```

```
      for (Harrastus harrastus : harrastusLista) {
         tauluObjektiLista.clear();
         for (int i = harrastus.ekaKentta(); i < harrastus.
             getKenttia(); i++) {
            if (apuHarrastus.getKysymys(i).equals("h/vko") ||
                apuHarrastus.getKysymys(i).equals("aloitusvuosi")) {
               tauluObjektiLista.add(new Integer(Integer.parseInt(
                  harrastus.anna(i))));
            }
            else {
               tauluObjektiLista.add(harrastus.anna(i).toString());
                                 }
         }
         tableHarrastukset.addItem(tauluObjektiLista.toArray(), new
             Integer(riviNro));
         riviNro++;
      }
   }


   /**
    * Avaa uuden ikkunan uuden jäsenen lisäämiseksi.
    */
   public void avaaUusiJasenIkkuna() {
      dialogiLayout = new VerticalLayout();

      apuJasen = new Jasen();
      uusiJasenWindow = new Window("Luo uusi jäsen");
      uusiJasenWindow.setWidth("400px");
      uusiJasenWindow.setHeight("100%");
      uusiJasenWindow.setModal(true);

      uusiJasenWindow.setContent(dialogiLayout);

      tietoLomake = new JasenTietoLomake(apuJasen);
      dialogiLayout.addComponent(tietoLomake);

      labelVirhe = new Label("");
      labelVirhe.setStyleName("labelVirhe");
      labelVirhe.setWidth("100%");
      dialogiLayout.addComponent(labelVirhe);

      lisaaDialogiNapit();

      uusiJasenWindow.center();
      UI.getCurrent().addWindow(uusiJasenWindow);
   }


   /**
    * Luo tallenna- ja peruuta-napit dialogiin.
    */
   @SuppressWarnings("serial")
   public void lisaaDialogiNapit() {
      Button buttonLisaa = new Button("Lisää jäsen");
      buttonLisaa.addClickListener(new Button.ClickListener() {
         public void buttonClick(ClickEvent event) {
            tallennaUusiJasen();
         }
      });

      Button buttonPeruuta = new Button("Peruuta");
      buttonPeruuta.addClickListener(new Button.ClickListener() {
         public void buttonClick(ClickEvent event) {
            uusiJasenWindow.close();
```

```
        }
    });

    HorizontalLayout buttonLayout = new HorizontalLayout();
    buttonLayout.setSpacing(true);
    dialogiLayout.addComponent(buttonLayout);
    dialogiLayout.setComponentAlignment(buttonLayout, Alignment.
        MIDDLE_CENTER);

    buttonLayout.addComponent(buttonLisaa);
    buttonLayout.addComponent(buttonPeruuta);
}


/**
 * Lisää ja tallentaa uuden jäsenen kerhoon.
 * Sulkee dialogin ja päivittää jäsenlistan.
 * Näyttää virheilmoituksen jos virheitä.
 */
public void tallennaUusiJasen() {
    if (!tietoLomake.onkoVirheita() && !tietoLomake.nimiTyhja())
        {
        apuJasen = tietoLomake.getJasen();
        try {
            apuJasen.rekisteroi();
            kerho.korvaaTaiLisaa(apuJasen);
            kerho.talleta();
        } catch (SailoException e) {
            e.printStackTrace();
        }
        uusiJasenWindow.close();
        labelJasenia.setValue("<h2>Jäseniä: " + Integer.toString(
            kerho.getJasenia()) + "</h2>");
        jasenLista = (List<Jasen>) kerho.etsi("*", 1);
        paivitaJasenlista(apuJasen.getTunnusnro());
    }
    else if (tietoLomake.nimiTyhja()) {
        labelVirhe.setValue("Nimi ei saa olla tyhjä!");
        proxy.animate(labelVirhe, AnimType.FADE_IN).setDuration
            (4000).setDelay(0);
        proxy.animate(labelVirhe, AnimType.FADE_OUT).setDuration
            (2000).setDelay(0);
    }
    else {
        labelVirhe.setValue("Virheellisiä syötteitä!");
        proxy.animate(labelVirhe, AnimType.FADE_IN).setDuration
            (4000).setDelay(0);
        proxy.animate(labelVirhe, AnimType.FADE_OUT).setDuration
            (2000).setDelay(0);
    }
}


@AutoGenerated
private VerticalLayout buildMainLayout() {
    // common part: create layout
    mainLayout = new VerticalLayout();
    mainLayout.setImmediate(false);
    mainLayout.setWidth("100%");
    mainLayout.setHeight("100%");
    mainLayout.setMargin(false);

    // top-level component properties
    setWidth("100.0%");
    setHeight("100.0%");
```

```
        // sivuLayout
        sivuLayout = buildSivuLayout();
        mainLayout.addComponent(sivuLayout);

        return mainLayout;
    }


    @AutoGenerated
    private VerticalLayout buildSivuLayout() {
        // common part: create layout
        sivuLayout = new VerticalLayout();
        sivuLayout.setImmediate(false);
        sivuLayout.setWidth("-1px");
        sivuLayout.setHeight("-1px");
        sivuLayout.setMargin(true);
        sivuLayout.setSpacing(true);

        // labelNimi
        labelNimi = new Label();
        labelNimi.setImmediate(false);
        labelNimi.setWidth("-1px");
        labelNimi.setHeight("-1px");
        labelNimi.setValue("Kerhon nimi");
        sivuLayout.addComponent(labelNimi);

        // labelJasenia
        labelJasenia = new Label();
        labelJasenia.setImmediate(false);
        labelJasenia.setWidth("-1px");
        labelJasenia.setHeight("-1px");
        labelJasenia.setValue("Jäseniä");
        sivuLayout.addComponent(labelJasenia);

        // hakuLayout
        hakuLayout = buildHakuLayout();
        sivuLayout.addComponent(hakuLayout);

        // sisaltoLayout
        sisaltoLayout = buildSisaltoLayout();
        sivuLayout.addComponent(sisaltoLayout);

        return sivuLayout;
    }


    @AutoGenerated
    private HorizontalLayout buildHakuLayout() {
        // common part: create layout
        hakuLayout = new HorizontalLayout();
        hakuLayout.setImmediate(false);
        hakuLayout.setWidth("-1px");
        hakuLayout.setHeight("-1px");
        hakuLayout.setMargin(false);
        hakuLayout.setSpacing(true);

        // labelHaku
        labelHaku = new Label();
        labelHaku.setImmediate(false);
        labelHaku.setWidth("-1px");
        labelHaku.setHeight("-1px");
        labelHaku.setValue("Etsittävä jäsen:");
        hakuLayout.addComponent(labelHaku);
```

```
        // textHaku
        textHaku = new TextField();
        textHaku.setImmediate(false);
        textHaku.setWidth("-1px");
        textHaku.setHeight("-1px");
        hakuLayout.addComponent(textHaku);

        // selectHaku
        selectHaku = new NativeSelect();
        selectHaku.setImmediate(false);
        selectHaku.setWidth("-1px");
        selectHaku.setHeight("-1px");
        hakuLayout.addComponent(selectHaku);

        // buttonHaku
        buttonHaku = new Button();
        buttonHaku.setCaption("Hae");
        buttonHaku.setImmediate(true);
        buttonHaku.setWidth("-1px");
        buttonHaku.setHeight("-1px");
        hakuLayout.addComponent(buttonHaku);

        return hakuLayout;
    }


    @AutoGenerated
    private HorizontalLayout buildSisaltoLayout() {
        // common part: create layout
        sisaltoLayout = new HorizontalLayout();
        sisaltoLayout.setImmediate(false);
        sisaltoLayout.setWidth("-1px");
        sisaltoLayout.setHeight("-1px");
        sisaltoLayout.setMargin(false);
        sisaltoLayout.setSpacing(true);

        // jasenetLayout
        jasenetLayout = buildJasenetLayout();
        sisaltoLayout.addComponent(jasenetLayout);

        // jasenLayout
        jasenLayout = new VerticalLayout();
        jasenLayout.setStyleName("jasenLayout");
        jasenLayout.setImmediate(false);
        jasenLayout.setWidth("-1px");
        jasenLayout.setHeight("-1px");
        jasenLayout.setMargin(false);
        jasenLayout.setSpacing(true);
        sisaltoLayout.addComponent(jasenLayout);

        // harrastusLayout
        harrastusLayout = buildHarrastusLayout();
        sisaltoLayout.addComponent(harrastusLayout);

        return sisaltoLayout;
    }


    @AutoGenerated
    private VerticalLayout buildJasenetLayout() {
        // common part: create layout
        jasenetLayout = new VerticalLayout();
        jasenetLayout.setStyleName("jasenetLayout");
        jasenetLayout.setImmediate(false);
        jasenetLayout.setWidth("-1px");
```

```
        jasenetLayout.setHeight("-1px");
        jasenetLayout.setMargin(false);
        jasenetLayout.setSpacing(true);

        // labelJasenLista
        labelJasenLista = new Label();
        labelJasenLista.setStyleName("labelJasenLista");
        labelJasenLista.setImmediate(false);
        labelJasenLista.setWidth("-1px");
        labelJasenLista.setHeight("-1px");
        labelJasenLista.setValue("Jäsenet:");
        jasenetLayout.addComponent(labelJasenLista);

        // listJasenet
        listJasenet = new ListSelect();
        listJasenet.setImmediate(false);
        listJasenet.setWidth("-1px");
        listJasenet.setHeight("-1px");
        jasenetLayout.addComponent(listJasenet);

        // buttonLisaaJasen
        buttonLisaaJasen = new Button();
        buttonLisaaJasen.setCaption("Lisää uusi jäsen");
        buttonLisaaJasen.setImmediate(true);
        buttonLisaaJasen.setWidth("-1px");
        buttonLisaaJasen.setHeight("-1px");
        jasenetLayout.addComponent(buttonLisaaJasen);

        // buttonPoista
        buttonPoista = new Button();
        buttonPoista.setCaption("Poista jäsen");
        buttonPoista.setImmediate(true);
        buttonPoista.setWidth("-1px");
        buttonPoista.setHeight("-1px");
        jasenetLayout.addComponent(buttonPoista);

        return jasenetLayout;
    }


    @AutoGenerated
    private VerticalLayout buildHarrastusLayout() {
        // common part: create layout
        harrastusLayout = new VerticalLayout();
        harrastusLayout.setStyleName("harrastusLayout");
        harrastusLayout.setImmediate(false);
        harrastusLayout.setWidth("-1px");
        harrastusLayout.setHeight("-1px");
        harrastusLayout.setMargin(false);
        harrastusLayout.setSpacing(true);

        // labelHarrastukset
        labelHarrastukset = new Label();
        labelHarrastukset.setStyleName("labelHarrastukset");
        labelHarrastukset.setImmediate(false);
        labelHarrastukset.setWidth("-1px");
        labelHarrastukset.setHeight("-1px");
        labelHarrastukset.setValue("Harrastukset:");
        harrastusLayout.addComponent(labelHarrastukset);

        return harrastusLayout;
    }
}
```

E.3 Vaadinjasen.java

```
package vaadinkerho;

import com.vaadin.annotations.Theme;
import com.vaadin.server.VaadinRequest;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;

@SuppressWarnings("serial")
@Theme("vaadinkerho")
public class Vaadinjasen extends UI {

   @Override
   protected void init(VaadinRequest request) {
      final VerticalLayout layout = new VerticalLayout();
      layout.setMargin(true);
      setContent(layout);
      layout.setSizeFull();

      String id = request.getParameter("id");

      Jasensivu sivu = new Jasensivu(id);
      layout.addComponent(sivu);
   }
}
```

E.4 Jasensivu.java

```
package vaadinkerho;

import java.util.ArrayList;
import java.util.List;

import kerho.Harrastus;
import kerho.Jasen;
import kerho.Kerho;
import kerho.SailoException;

import org.vaadin.dialogs.ConfirmDialog;
import org.vaadin.jouni.animator.AnimatorProxy;
import org.vaadin.jouni.animator.shared.AnimType;

import com.vaadin.annotations.AutoGenerated;
import com.vaadin.event.FieldEvents.FocusEvent;
import com.vaadin.event.FieldEvents.FocusListener;
import com.vaadin.event.FieldEvents.TextChangeEvent;
import com.vaadin.event.FieldEvents.TextChangeListener;
import com.vaadin.server.ExternalResource;
import com.vaadin.server.Page;
import com.vaadin.server.UserError;
import com.vaadin.shared.Position;
import com.vaadin.shared.ui.label.ContentMode;
import com.vaadin.ui.AbstractTextField.TextChangeEventMode;
import com.vaadin.ui.Alignment;
import com.vaadin.ui.Button;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.CustomComponent;
import com.vaadin.ui.FormLayout;
import com.vaadin.ui.HorizontalLayout;
import com.vaadin.ui.Label;
```

```
import com.vaadin.ui.Link;
import com.vaadin.ui.Notification;
import com.vaadin.ui.Table;
import com.vaadin.ui.TextField;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;
import com.vaadin.ui.Window;

public class Jasensivu extends CustomComponent implements Button.
    ClickListener{

   /*- VaadinEditorProperties={"grid":"RegularGrid,20","showGrid":
       true,"snapToGrid":true,"snapToObject":true,"movingGuides":
       false,"snappingDistance":10} */

   @AutoGenerated
   private VerticalLayout mainLayout;
   @AutoGenerated
   private VerticalLayout sivuLayout;
   @AutoGenerated
   private Button buttonTallenna;
   @AutoGenerated
   private HorizontalLayout jasenLayout;
   @AutoGenerated
   private VerticalLayout harrastusMuokkausLayout;
   @AutoGenerated
   private VerticalLayout tietoMuokkausLayout;
   @AutoGenerated
   private Label labelOtsikko;
   private Kerho kerho;
   private int jasenId;
   private Jasen jasen;
   private int virheita;
   private int uusiHarrastusvirheita;
   private List<Harrastus> harrastusLista;
   private Table tableHarrastukset;
   private List<Object> tauluObjektiLista;
   private Harrastus apuHarrastus;
   private Window uusiHarrastusWindow;
   private FormLayout uusiHarrastusLayout;
   private VerticalLayout dialogiLayout;
   private Label labelVirhe;
   private Harrastus valittuHarrastus;
   private AnimatorProxy proxy;
   private JasenTietoLomake tietoLomake;
   private TextField alaKentta;

   private static final long serialVersionUID = 1L;
   /**
    * The constructor should first build the main layout, set the
    * composition root and then do any custom initialization.
    *
    * The constructor will not be automatically regenerated by the
    * visual editor.
    */
   public Jasensivu(String id) {
      buildMainLayout();
      setCompositionRoot(mainLayout);

      proxy = new AnimatorProxy();
      mainLayout.addComponent(proxy);

      virheita = 0;
      kerho = KerhoBean.getKerho();
```

```
      try {
         kerho.lueTiedostosta("kelmit");
      } catch (SailoException e) {
         e.printStackTrace();
      }

      jasenId = Integer.parseInt(id);
      jasen = kerho.annaJasenId(jasenId);
      apuHarrastus = new Harrastus();

      buttonTallenna.addClickListener(this);
      labelOtsikko.setContentMode(ContentMode.HTML);
      labelOtsikko.setValue("<h1>" + jasen.getNimi() + "</h1>");

      Link linkKerho = new Link("<- Takaisin kerhoon",
            new ExternalResource("/vaadinKerho/kerhosivu/?id=" + id
               ));
      sivuLayout.addComponent(linkKerho);

      tietoLomake = new JasenTietoLomake(jasen);
      tietoMuokkausLayout.addComponent(tietoLomake);
      luoHarrastusMuokkausAlue();
   }


   /**
    * Kuuntelija tallennusnapille.
    */
   @Override
   public void buttonClick(ClickEvent event) {
      if (event.getButton() == buttonTallenna) {
         tallennaJasen();
      }
   }


   /**
    * Tallentaa jäsenen jos ei virheitä
    * ja näyttää ilmoituksen tallennettiinko jäsen.
    */
   public void tallennaJasen() {
      if (virheita == 0 && !tietoLomake.onkoVirheita()) {
         jasen = tietoLomake.getJasen();
         try {
            kerho.korvaaTaiLisaa(jasen);
            kerho.talleta();
         } catch (SailoException e) {
            e.printStackTrace();
         }
         Notification onnistumisIlmoitus = new Notification("Jäsen
             tallennettu","",
                Notification.Type.HUMANIZED_MESSAGE);
         onnistumisIlmoitus.setDelayMsec(3000);
         onnistumisIlmoitus.setPosition(Position.MIDDLE_LEFT);
         onnistumisIlmoitus.show(Page.getCurrent());
      }
      else {
         Notification virheIlmoitus= new Notification("Virheellisiä
              syötteitä","Muutoksia ei tallennettu",
                Notification.Type.WARNING_MESSAGE);
         virheIlmoitus.setDelayMsec(3000);
         virheIlmoitus.setPosition(Position.MIDDLE_LEFT);
         virheIlmoitus.show(Page.getCurrent());
      }
   }
```

```
/**
 * Luo taulukon ja napit harrastuksien muokkausta
 * poistoa ja lisäystä varten.
 */
public void luoHarrastusMuokkausAlue() {
   harrastusLista = kerho.annaHarrastukset(jasen);
   luoHarrastusMuokkausTaulukko();
   luoHarrastusMuokkausNapit();

   proxy.animate(tableHarrastukset, AnimType.FADE_IN).
      setDuration(500).setDelay(200);
}


/**
 * Luo taulukon tekstikentillä harrastusten
 * muokkausta varten.
 */
public void luoHarrastusMuokkausTaulukko() {
   tauluObjektiLista = new ArrayList<Object>();

   tableHarrastukset = new Table("Harrastukset:");
   tableHarrastukset.setPageLength(harrastusLista.size());
   harrastusMuokkausLayout.addComponent(tableHarrastukset);

   luoTaulukkoOtsikot();

   if (harrastusLista.size() > 0) luoTaulukkoKentat();
}


/**
 * Luo harrastustaulukon otsikkorivin.
 */
public void luoTaulukkoOtsikot() {
   // Määritellään taulukon sarakkeet ja annetaan niille otsikot
      .
   if (harrastusLista.size() > 0) {
      for (int i = apuHarrastus.ekaKentta(); i < apuHarrastus.
         getKenttia(); i++) {
         tableHarrastukset.addContainerProperty(apuHarrastus.
            getKysymys(i), FormLayout.class,  null);
      }
   }
   else {
      tableHarrastukset.addContainerProperty("Ei harrastuksia!",
          FormLayout.class,  null);
   }
}


/**
 * Luo taulukkoon tekstikentät harrastuksille.
 */
public void luoTaulukkoKentat() {
   int riviNro = 1;

   // Lisätään harrastukset taulukkoon.
   for (Harrastus harrastus : harrastusLista) {
      tauluObjektiLista.clear();
      for (int i = harrastus.ekaKentta(); i < harrastus.
         getKenttia(); i++) {
```

```
            // Lisätään FormLayout, jonka sisään tulee tekstikenttä
                .
            // Näin saadaan myös virhe näkyviin.
            FormLayout harrastusKenttaLomake = new FormLayout();
            harrastusKenttaLomake.setSpacing(false);
            harrastusKenttaLomake.setMargin(false);
            TextField fieldHarrastus = luoKentta(harrastus, i);

            harrastusKenttaLomake.addComponent(fieldHarrastus);
            tauluObjektiLista.add(harrastusKenttaLomake);
         }
         tableHarrastukset.addItem(tauluObjektiLista.toArray(), new
             Integer(riviNro));
         riviNro++;
      }
   }


   /**
    * Luo tekstikentän harrastukselle.
    * @param harrastus harrastus jolle tekstikenttä luodaan.
    * @param i monesko harrastuksen kenttä on kyseessä.
    * @return harrastuksen tekstikenttä.
    */
   public TextField luoKentta(Harrastus harrastus, int i) {
      TextField fieldHarrastus = new TextField();
      fieldHarrastus.setImmediate(true);
      fieldHarrastus.setTextChangeEventMode(TextChangeEventMode.
          EAGER);
      fieldHarrastus.setId(Integer.toString(harrastus.getTunnusnro
          ()) + "," + Integer.toString(i));
      fieldHarrastus.setValue(harrastus.anna(i));

      if (i != harrastus.ekaKentta()) fieldHarrastus.setWidth("50px
          ");
      else fieldHarrastus.setWidth("130px");

      lisaaKenttaKuuntelijat(fieldHarrastus);
      return fieldHarrastus;
   }


   /**
    * Lisää tekstikentälle tekstinmuutoskuuntelijan virheiden
        tarkistuksia
    * ja arvojen asettamista varten sekä Focus-kuuntelijan
    * harrastuksen poistoa varten.
    * @param fieldHarrastus kenttä jolle kuuntelija lisätään.
    */
   @SuppressWarnings("serial")
   public void lisaaKenttaKuuntelijat(TextField fieldHarrastus) {
      fieldHarrastus.addTextChangeListener(new TextChangeListener()
          {
         public void textChange(TextChangeEvent event) {
            TextField field = (TextField) event.getComponent();
            Harrastus har = new Harrastus();
            String[] osat = field.getId().split(",");

            for (Harrastus harrastus : harrastusLista) {
               if (harrastus.getTunnusnro() == Integer.parseInt(
                   osat[0])) {
                  har = harrastus;
                  break;
               }
            }
```

```
            kerho.poistaHarrastus(har);
            String virhe = har.aseta(Integer.parseInt(osat[1]),
                event.getText());
            asetaKenttaVirhe(field, har, virhe);
         }
      });


      fieldHarrastus.addFocusListener(new FocusListener() {
         @Override
         public void focus(FocusEvent event) {
            TextField field = (TextField) event.getComponent();
            String[] osat = field.getId().split(",");

            for (Harrastus harrastus : harrastusLista) {
               if (harrastus.getTunnusnro() == Integer.parseInt(
                   osat[0])) {
                  valittuHarrastus = harrastus;
                  break;
               }
            }
         }
      });
   }


   /**
    * Asettaa tai poistaa tekstikentän virheen.
    * @param field kenttä minkä virhettä käsitellään.
    * @param har harrastus jolle virhe kuuluu.
    * @param virhe virheteksti tai null jos ei virhettä.
    */
   public void asetaKenttaVirhe(TextField field, Harrastus har,
       String virhe) {
      if (virhe != null && field.getComponentError() == null) {
         field.setStyleName("virheKentta");
         virheita++;
      }
      else if (virhe == null && field.getComponentError() != null)
          {
         field.setStyleName("");
         virheita--;
      }

      if (virhe != null) field.setComponentError(new UserError(
          virhe));
      else {
         field.setComponentError(null);
         kerho.lisaa(har);
      }
   }


   /**
    * Luo lisää- ja poista-napit harrastuksille.
    */
   @SuppressWarnings("serial")
   public void luoHarrastusMuokkausNapit() {
      HorizontalLayout harrastusNappiLayout = new HorizontalLayout
          ();
      harrastusNappiLayout.setMargin(true);
      harrastusNappiLayout.setSpacing(true);
      harrastusMuokkausLayout.addComponent(harrastusNappiLayout);
```

```
        Button buttonLisaaHarrastus = new Button("Lisää uusi
            harrastus");
        harrastusNappiLayout.addComponent(buttonLisaaHarrastus);

        buttonLisaaHarrastus.addClickListener(new Button.
            ClickListener() {
            public void buttonClick(ClickEvent event) {
                lisaaHarrastus();
            }
        });

        if (harrastusLista.size() > 0) {
            Button buttonPoistaHarrastus = new Button("Poista
                harrastus");
            harrastusNappiLayout.addComponent(buttonPoistaHarrastus);

            buttonPoistaHarrastus.addClickListener(new Button.
                ClickListener() {
                public void buttonClick(ClickEvent event) {
                    poistaHarrastus();
                }
            });
        }
    }


    /**
     * Avaa dialogin uuden harrastuksen luomista varten.
     */
    public void lisaaHarrastus() {
        labelVirhe = new Label("");
        labelVirhe.setValue("");
        labelVirhe.setWidth("100%");
        labelVirhe.setStyleName("labelVirhe");

        apuHarrastus = new Harrastus(jasenId);
        uusiHarrastusvirheita = 0;
        uusiHarrastusWindow = new Window("Luo uusi harrastus");
        uusiHarrastusWindow.setWidth("400px");
        uusiHarrastusWindow.setHeight("300px");
        uusiHarrastusWindow.setModal(true);

        dialogiLayout = new VerticalLayout();
        dialogiLayout.setWidth("100%");
        uusiHarrastusWindow.setContent(dialogiLayout);

        uusiHarrastusLayout = new FormLayout();
        uusiHarrastusLayout.setMargin(true);
        lisaaDialogiKentat();
        dialogiLayout.addComponent(uusiHarrastusLayout);
        dialogiLayout.setComponentAlignment(uusiHarrastusLayout,
            Alignment.TOP_CENTER);

        dialogiLayout.addComponent(labelVirhe);
        lisaaDialogiNapit();

        uusiHarrastusWindow.center();
        UI.getCurrent().addWindow(uusiHarrastusWindow);
    }


    /**
     * Luo kentät uuden harrastuksen lisäämistä varten.
     */
    @SuppressWarnings("serial")
```

```
public void lisaaDialogiKentat() {
   for (int i = apuHarrastus.ekaKentta(); i < apuHarrastus.
      getKenttia(); i++) {
      TextField fieldUusiHarrastus = new TextField(apuHarrastus.
         getKysymys(i) + ": ");
      fieldUusiHarrastus.setImmediate(true);
      fieldUusiHarrastus.setTextChangeEventMode(
         TextChangeEventMode.EAGER);
      fieldUusiHarrastus.setId(Integer.toString(i));
      if (i != apuHarrastus.ekaKentta()) fieldUusiHarrastus.
         setWidth("50px");
      else fieldUusiHarrastus.setWidth("130px");
      if (i == apuHarrastus.ekaKentta()) alaKentta =
         fieldUusiHarrastus;

      fieldUusiHarrastus.addTextChangeListener(new
         TextChangeListener() {
         public void textChange(TextChangeEvent event) {
            TextField field = (TextField) event.getComponent();

            String virhe = apuHarrastus.aseta(Integer.parseInt(
               field.getId()), event.getText());

            if (virhe != null && field.getComponentError() ==
               null) {
               field.setStyleName("virheKentta");
               uusiHarrastusvirheita++;
            }
            else if (virhe == null && field.getComponentError()
               != null) {
               field.setStyleName("");
               uusiHarrastusvirheita--;
               if(uusiHarrastusvirheita == 0) {
                  labelVirhe.setStyleName("labelVirhe");
                  labelVirhe.setValue("");
               }
            }

            if (virhe != null) field.setComponentError(new
               UserError(virhe));
            else field.setComponentError(null);
         }
      });
      uusiHarrastusLayout.addComponent(fieldUusiHarrastus);
   }
}


/**
 * Luo tallenna- ja peruuta-napit dialogiin.
 */
@SuppressWarnings("serial")
public void lisaaDialogiNapit() {
   Button buttonLisaa = new Button("Lisää harrastus");
   buttonLisaa.addClickListener(new Button.ClickListener() {
      public void buttonClick(ClickEvent event) {
         tallennaUusiHarrastus();
      }
   });

   Button buttonPeruuta = new Button("Peruuta");
   buttonPeruuta.addClickListener(new Button.ClickListener() {
      public void buttonClick(ClickEvent event) {
         uusiHarrastusWindow.close();
      }
```

```
        });

        HorizontalLayout buttonLayout = new HorizontalLayout();
        buttonLayout.setSpacing(true);
        dialogiLayout.addComponent(buttonLayout);
        dialogiLayout.setComponentAlignment(buttonLayout, Alignment.
            MIDDLE_CENTER);

        buttonLayout.addComponent(buttonLisaa);
        buttonLayout.addComponent(buttonPeruuta);
    }


    /**
     * Lisää ja tallentaa uuden harrastuksen kerhoon.
     * Jos lisäys onnistuu, sulkee dialogin ja
     * päivittää harrastusmuokkaustaulukon sivulle.
     */
    public void tallennaUusiHarrastus() {
        if (uusiHarrastusvirheita == 0 && !alaKentta.getValue().
            equals("")) {
            apuHarrastus.rekisteroi();
            kerho.lisaa(apuHarrastus);

            harrastusMuokkausLayout.removeAllComponents();
            luoHarrastusMuokkausAlue();
            uusiHarrastusWindow.close();
        }
        else if (alaKentta.getValue().equals("")) {
            labelVirhe.setValue("Nimi ei saa olla tyhjä!");
            proxy.animate(labelVirhe, AnimType.FADE_IN).setDuration
                (4000).setDelay(0);
            proxy.animate(labelVirhe, AnimType.FADE_OUT).setDuration
                (2000).setDelay(0);
        }
        else {
            labelVirhe.setValue("Virheellisiä syötteitä!");
            proxy.animate(labelVirhe, AnimType.FADE_IN).setDuration
                (4000).setDelay(0);
            proxy.animate(labelVirhe, AnimType.FADE_OUT).setDuration
                (2000).setDelay(0);
        }
    }


    /**
     * Poistaa valitun harrastuksen.
     * Varmistaa poiston dialogilla.
     */
    @SuppressWarnings("serial")
    public void poistaHarrastus() {
        if (valittuHarrastus == null) {
            Notification onnistumisIlmoitus = new Notification("Ei
                valittua harrastusta!","",
                   Notification.Type.WARNING_MESSAGE);
            onnistumisIlmoitus.setDelayMsec(3000);
            onnistumisIlmoitus.setPosition(Position.TOP_CENTER);
            onnistumisIlmoitus.show(Page.getCurrent());
            return;
        }

        ConfirmDialog.show(UI.getCurrent(), "Harrastuksen poisto", "
            Poistetaanko harrastus: " + valittuHarrastus.anna(
            valittuHarrastus.ekaKentta()) + "?",
               "Kyllä", "Ei", new ConfirmDialog.Listener() {
```

```
            public void onClose(ConfirmDialog dialog) {
                if (dialog.isConfirmed()) {
                    kerho.poistaHarrastus(valittuHarrastus);
                    harrastusMuokkausLayout.removeAllComponents();
                    luoHarrastusMuokkausAlue();
                }
            }
        });
    }


    @AutoGenerated
    private VerticalLayout buildMainLayout() {
        // common part: create layout
        mainLayout = new VerticalLayout();
        mainLayout.setImmediate(false);
        mainLayout.setWidth("100%");
        mainLayout.setHeight("100%");
        mainLayout.setMargin(false);

        // top-level component properties
        setWidth("100.0%");
        setHeight("100.0%");

        // sivuLayout
        sivuLayout = buildSivuLayout();
        mainLayout.addComponent(sivuLayout);

        return mainLayout;
    }


    @AutoGenerated
    private VerticalLayout buildSivuLayout() {
        // common part: create layout
        sivuLayout = new VerticalLayout();
        sivuLayout.setImmediate(false);
        sivuLayout.setWidth("-1px");
        sivuLayout.setHeight("-1px");
        sivuLayout.setMargin(true);
        sivuLayout.setSpacing(true);

        // labelOtsikko
        labelOtsikko = new Label();
        labelOtsikko.setImmediate(false);
        labelOtsikko.setWidth("-1px");
        labelOtsikko.setHeight("-1px");
        labelOtsikko.setValue("Otsikko");
        sivuLayout.addComponent(labelOtsikko);

        // jasenLayout
        jasenLayout = buildJasenLayout();
        sivuLayout.addComponent(jasenLayout);

        // buttonTallenna
        buttonTallenna = new Button();
        buttonTallenna.setCaption("Tallenna jäsen");
        buttonTallenna.setImmediate(true);
        buttonTallenna.setWidth("-1px");
        buttonTallenna.setHeight("-1px");
        sivuLayout.addComponent(buttonTallenna);

        return sivuLayout;
    }
```

```
    @AutoGenerated
    private HorizontalLayout buildJasenLayout() {
        // common part: create layout
        jasenLayout = new HorizontalLayout();
        jasenLayout.setImmediate(false);
        jasenLayout.setWidth("-1px");
        jasenLayout.setHeight("-1px");
        jasenLayout.setMargin(false);
        jasenLayout.setSpacing(true);

        // tietoMuokkausLayout
        tietoMuokkausLayout = new VerticalLayout();
        tietoMuokkausLayout.setImmediate(false);
        tietoMuokkausLayout.setWidth("-1px");
        tietoMuokkausLayout.setHeight("-1px");
        tietoMuokkausLayout.setMargin(false);
        jasenLayout.addComponent(tietoMuokkausLayout);

        // harrastusMuokkausLayout
        harrastusMuokkausLayout = new VerticalLayout();
        harrastusMuokkausLayout.setStyleName("harrastusMuokkausLayout
            ");
        harrastusMuokkausLayout.setImmediate(false);
        harrastusMuokkausLayout.setWidth("-1px");
        harrastusMuokkausLayout.setHeight("-1px");
        harrastusMuokkausLayout.setMargin(true);
        jasenLayout.addComponent(harrastusMuokkausLayout);

        return jasenLayout;
    }
}
```

E.5 JasenTietoLomake.java

```
package vaadinkerho;

import kerho.Jasen;

import com.vaadin.event.FieldEvents.TextChangeEvent;
import com.vaadin.event.FieldEvents.TextChangeListener;
import com.vaadin.server.UserError;
import com.vaadin.ui.AbstractTextField.TextChangeEventMode;
import com.vaadin.ui.CustomComponent;
import com.vaadin.ui.FormLayout;
import com.vaadin.ui.TextField;
import com.vaadin.ui.VerticalLayout;

public class JasenTietoLomake extends CustomComponent {

    private static final long serialVersionUID = 1L;

    private FormLayout lomakeLayout;
    private int virheita = 0;
    private TextField fieldJasentieto;
    private Jasen jasen;
    private TextField nimiKentta;

    public JasenTietoLomake(Jasen muokattavaJasen) {
        VerticalLayout layout = new VerticalLayout();
```

```
        setCompositionRoot(layout);
        setSizeFull();

        lomakeLayout = new FormLayout();
        lomakeLayout.setMargin(true);
        layout.addComponent(lomakeLayout);

        jasen = muokattavaJasen;
        lisaaKentat();
    }


    public Jasen getJasen() {
        return jasen;
    }

    public boolean onkoVirheita() {
        return virheita > 0;
    }

    public boolean nimiTyhja() {
        return nimiKentta.getValue().equals("");
    }


    /**
     * Luo lomakkeen tekstikentät jäsenen tiedoille.
     */
    public void lisaaKentat() {
        for (int i = jasen.ekaKentta(); i < jasen.getKenttia(); i++)
            {
            fieldJasentieto = luoKentta(i);
            luoKuuntelija();

            lomakeLayout.addComponent(fieldJasentieto);
        }
    }


    /**
     * Luo tekstikentän tietylle jäsenen tiedolle.
     * @param i jäsenen kenttä jolle tekstikenttä luodaan.
     * @return tekstikenttä tietylle jäsenen tiedolle.
     */
    public TextField luoKentta(int i) {
        fieldJasentieto = new TextField(jasen.getKysymys(i) + ": ");
        fieldJasentieto.setImmediate(true);
        fieldJasentieto.setTextChangeEventMode(TextChangeEventMode.
            EAGER);
        fieldJasentieto.setId(Integer.toString(i));
        fieldJasentieto.setValue(jasen.anna(i));
        fieldJasentieto.setWidth("150px");
        if (i == jasen.ekaKentta()) nimiKentta = fieldJasentieto;

        return fieldJasentieto;
    }


    /**
     * Luo kuuntelijan tekstikentän tekstinvaihtumiselle.
     */
    @SuppressWarnings("serial")
    public void luoKuuntelija() {
        fieldJasentieto.addTextChangeListener(new TextChangeListener
            () {
```

```
            public void textChange(TextChangeEvent event) {
                TextField field = (TextField) event.getComponent();
                int kentta = Integer.parseInt(field.getId());

                String virhe = jasen.aseta(kentta, event.getText());
                tarkistaVirhe(virhe, field);
            }
        });
    }


    /**
     * Tarkistaa onko kenttään asetettu oikeanlainen arvo.
     * @param virhe tiedon asetuksessa saatu virhe.
     * @param field tekstikenttä joka tarkistetaan.
     */
    public void tarkistaVirhe(String virhe, TextField field) {
        if (virhe != null && field.getComponentError() == null) {
            field.setStyleName("virheKentta");
            virheita++;
        }
        else if (virhe == null && field.getComponentError() != null)
            {
            field.setStyleName("");
            virheita--;
        }

        if (virhe != null) field.setComponentError(new UserError(
            virhe));
        else field.setComponentError(null);
    }
}
```

E.6 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http
    ://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
   version="3.0">
   <display-name>vaadinKerho</display-name>
   <context-param>
      <description>Vaadin production mode</description>
      <param-name>productionMode</param-name>
      <param-value>false</param-value>
   </context-param>
   <servlet>
      <servlet-name>Vaadinkerho</servlet-name>
      <servlet-class>com.vaadin.server.VaadinServlet</servlet-class
          >
      <init-param>
         <description>Vaadin UI class to use</description>
         <param-name>UI</param-name>
         <param-value>vaadinkerho.Vaadinkerho</param-value>
      </init-param>
      <init-param>
         <description>Application widgetset</description>
         <param-name>widgetset</param-name>
         <param-value>vaadinkerho.widgetset.VaadinkerhoWidgetset</
             param-value>
```

```
      </init-param>
      <init-param>
         <description>Legacy mode to return the value of the
            property as a string from AbstractProperty.toString()</
            description>
         <param-name>legacyPropertyToString</param-name>
         <param-value>false</param-value>
      </init-param>
      <init-param>
         <param-name>pushmode</param-name>
         <param-value>manual</param-value>
      </init-param>
      <async-supported>true</async-supported>
   </servlet>
   <servlet>
      <servlet-name>Jasensivu</servlet-name>
      <servlet-class>com.vaadin.server.VaadinServlet</servlet-class
         >
      <init-param>
         <description>Vaadin UI class to use</description>
         <param-name>UI</param-name>
         <param-value>vaadinkerho.Vaadinjasen</param-value>
      </init-param>
      <init-param>
         <description>Application widgetset</description>
         <param-name>widgetset</param-name>
         <param-value>vaadinkerho.widgetset.VaadinkerhoWidgetset</
            param-value>
      </init-param>
   </servlet>
   <servlet-mapping>
      <servlet-name>Vaadinkerho</servlet-name>
      <url-pattern>/kerhosivu/*</url-pattern>
   </servlet-mapping>
   <servlet-mapping>
      <servlet-name>Vaadinkerho</servlet-name>
      <url-pattern>/VAADIN/*</url-pattern>
   </servlet-mapping>
   <servlet-mapping>
      <servlet-name>Jasensivu</servlet-name>
      <url-pattern>/jasensivu/*</url-pattern>
   </servlet-mapping>
   <welcome-file-list>
      <welcome-file>index.html</welcome-file>
      <welcome-file>index.htm</welcome-file>
      <welcome-file>index.jsp</welcome-file>
      <welcome-file>default.html</welcome-file>
      <welcome-file>default.htm</welcome-file>
      <welcome-file>default.jsp</welcome-file>
   </welcome-file-list>
</web-app>
```

E.7 vaadinkerho.scss

```
/* Import the reindeer theme.*/
/* This only allows us to use the mixins defined in it and does not
    add any styles by itself. */
@import "../reindeer/reindeer.scss";

/* This contains all of your theme.*/
```

```
/* If somebody wants to extend the theme she will include this
   mixin. */
@mixin vaadinkerho {
   /* Include all the styles from the reindeer theme */
   @include reindeer;

   /* Insert your theme rules here */

   .jasenetLayout {
      margin-top: 20px;
   }
   .jasenLayout {
      margin-top: 20px;
      margin-left: 50px;
   }
   .harrastusLayout {
      margin-top: 20px;
      margin-left: 50px;
   }
   .labelJasenLista {
      font-weight: bold;
   }
   .labelJasenOtsikko{
      font-weight: bold;
      text-decoration: underline;
   }
   .labelHarrastukset {
      font-weight: bold;
   }
   .harrastusMuokkausLayout {
      margin-left: 50px;
   }
   .virheKentta {
      border-color: red;
   }
   .labelVirhe {
      background-color: red;
      color: white;
      text-align: center;
      margin-bottom: 10px;
   }
}
```

F ASP.NET-kerhon lähdekoodit

F.1 JasenTietoLomake.ascx

```
ï»¿<%@ Control Language="C#" AutoEventWireup="true" CodeFile="
   JasenTietoLomake.ascx.cs" Inherits="WebUserControl" %>

<asp:UpdatePanel ID="updatePanelJasenlomake" UpdateMode="
   Conditional" runat="server">
   <ContentTemplate>
      <asp:Table ID="tableJasenLomake" runat="server" BorderStyle="
         None" Caption="" CaptionAlign="Top"
         CellPadding="0" CssClass="tableJasentietoLomake">
      </asp:Table>
   </ContentTemplate>
</asp:UpdatePanel>
```

F.2 JasenTietoLomake.ascx.cs

```
ï»¿using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using KerhoApplication;

public partial class WebUserControl : System.Web.UI.UserControl
{
   private Jasen jasen;
   private TextBox nimiKentta;


   public Jasen Jasen
   {
      get { return jasen; }
      set
      {
         jasen = value;
      }
   }


   public Boolean Nimityhja
   {
      get { return nimiKentta.Text.Equals(""); }
   }


   protected void Page_Init(object sender, System.EventArgs e)
   {
      jasen = new Jasen();
   }


   protected void Page_Load(object sender, EventArgs e)
   {
      luoJasentietoLomake();
   }


   /// <summary>
   /// Luo tietojenmuokkauslomakkkeen jäsenelle.
   /// </summary>
   public void luoJasentietoLomake()
   {
      for (int i = jasen.ekaKentta(); i < jasen.getKenttia(); i++)
      {
         luoLomakerivi(i);
      }
   }


   /// <summary>
   /// Luo lomaketaulukkoon rivin.
   /// </summary>
   /// <param name="i">Jäsenen kenttä jonka tieto lisätään</param>
```

```
public void luoLomakerivi(int i)
{
   TableRow tietoRivi = new TableRow();
   tableJasenLomake.Rows.Add(tietoRivi);

   TableCell kysymysSolu = luoKysymysSolu(i);
   TableCell tietoSolu = new TableCell();
   TableCell virheSolu = luoVirheSolu(i);
   TextBox textTieto = luoTietoKentta(i);

   tietoSolu.Controls.Add(textTieto);
   tietoRivi.Cells.Add(kysymysSolu);
   tietoRivi.Cells.Add(tietoSolu);
   tietoRivi.Cells.Add(virheSolu);

   luoJasentietoTriggeri(i);
}


/// <summary>
/// Luo taulukkosolun jäsenen kysymyksellä.
/// </summary>
/// <param name="i">Jäsenen kenttä jolle solu luodaan</param>
/// <returns>Taulukkosolu jäsenen kysymyksellä</returns>
public TableCell luoKysymysSolu(int i)
{
   TableCell kysymysSolu = new TableCell();
   kysymysSolu.Text = jasen.getKysymys(i) + ":";
   kysymysSolu.Style.Add("text-align", "right");
   return kysymysSolu;
}


/// <summary>
/// Luo jäsentietokentän jäsenen kysymykselle.
/// </summary>
/// <param name="i">Jäsenen kenttä jolle tekstikenttä luodaan</
    param>
/// <returns>Tekstikenttä jäsenen kysymykselle</returns>
public TextBox luoTietoKentta(int i)
{
   TextBox textTieto = new TextBox();
   textTieto.ID = i.ToString();
   textTieto.AutoPostBack = true;
   textTieto.TextChanged += new EventHandler(
      textTieto_TextChanged);
   textTieto.Text = jasen.anna(i);
   textTieto.Width = (Unit.Pixel(150));

   if (i == jasen.ekaKentta()) nimiKentta = textTieto;
   return textTieto;
}


/// <summary>
/// Luo virheilmoitussolun.
/// </summary>
/// <param name="i">Jäsenen kenttä jolle solu luodaan</param>
/// <returns>Solu virheelle</returns>
public TableCell luoVirheSolu(int i)
{
   TableCell virheSolu = new TableCell();
   virheSolu.ID = "virhe" + i.ToString();
   virheSolu.CssClass = "virheTeksti";
   return virheSolu;
```

```
    }


    /// <summary>
    /// Luo updatepaneliin triggerin tietomuokkauskentälle.
    /// </summary>
    /// <param name="i">Jäsenen kenttä jolle trigger luodaan</param>
    public void luoJasentietoTriggeri(int i)
    {
       AsyncPostBackTrigger trig = new AsyncPostBackTrigger();
       trig.ControlID = i.ToString();
       trig.EventName = "TextChanged";
       updatePanelJasenlomake.Triggers.Add(trig);
    }


    /// <summary>
    /// Kuuntelija jäsentietokenttien tekstin vaihtumiselle.
    /// </summary>
    /// <param name="sender">Tekstikenttä, joka tapahtuman synnyttää
        </param>
    /// <param name="e"></param>
    protected void textTieto_TextChanged(object sender, EventArgs e)
    {
       TextBox box = (TextBox)sender;
       asetaJaTarkistaTieto(box);
    }


    /// <summary>
    /// Asettaa tekstikentän sisällön jäsenelle ja tarkistaa ja nä
        yttää virheen.
    /// </summary>
    /// <param name="box">Tekstikenttä jonka sisältö asetetaan</
        param>
    public void asetaJaTarkistaTieto(TextBox box)
    {
       int kentta = Int32.Parse(box.ID);
       String virhe = jasen.aseta(kentta, box.Text);

       if (virhe != null && box.CssClass.Equals(""))
       {
          box.CssClass = "virheTieto";
       }

       else if (virhe == null && box.CssClass.Equals("virheTieto"))
       {
          box.CssClass = "";
       }

       TableCell cell = (TableCell)FindControl("virhe" + box.ID);
       if (virhe != null) cell.Text = virhe;
       else cell.Text = "";
    }


    /// <summary>
    /// Asettaa jäsenen kenttiin tekstikenttien arvot ja tarkistaa
        virheet.
    /// </summary>
    public int tarkistaJasenKentat()
    {
       int tallennusVirheita = 0;

       for (int i = jasen.ekaKentta(); i < jasen.getKenttia(); i++)
```

```
      {
         TextBox box = (TextBox)FindControl(i.ToString());
         String virhe = jasen.aseta(i, box.Text);

         if (virhe != null) tallennusVirheita++;
      }

      return tallennusVirheita;
   }


   /// <summary>
   /// Luo tietojenmuokkauslomakkeen uudelleen.
   /// </summary>
   public void paivitaLomake()
   {
      tableJasenLomake.Controls.Clear();
      luoJasentietoLomake();
      updatePanelJasenlomake.Update();
   }
}
```

F.3 kerho/Default.aspx

```
ï»¿<%@ Page Title="Kerho" Language="C#" AutoEventWireup="true"
    CodeFile="Default.aspx.cs"
   Inherits="_Default" %>

<%@ Register TagPrefix="ajaxToolkit" Namespace="AjaxControlToolkit"
     Assembly="AjaxControlToolkit" %>
<%@ Register Src="../JasenTietoLomake.ascx" TagName="
    JasenTietoLomake" TagPrefix="uc1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
    http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
   <link href="../Styles/kerho.css" rel="stylesheet" type="text/css
       " />
   <title>Kerho</title>
</head>
<body>
   <form id="form1" runat="server">
      <ajaxToolkit:ToolkitScriptManager ID="ToolScriptManager1"
          runat="Server" EnableHistory="true"
         EnableSecureHistoryState="false" OnNavigate="
             ToolScriptManager1_Navigate" />

      <h1>
         <asp:Label runat="server" ID="labelKerhoNimi"></asp:Label>
      </h1>

      <h2>
         <asp:UpdatePanel ID="updatePanel1" UpdateMode="Conditional
             " runat="server">
            <ContentTemplate>
               <asp:Label runat="server" ID="labelJasenia"></asp:
                   Label>
            </ContentTemplate>
            <Triggers>
               <asp:AsyncPostBackTrigger ControlID="buttonLisaa"
                   EventName="Click" />
```

```
            <asp:AsyncPostBackTrigger ControlID="buttonPoista"
               EventName="Click" />
         </Triggers>
      </asp:UpdatePanel>
   </h2>

   <asp:UpdatePanel ID="updatePanelHaku" UpdateMode="Conditional
      " runat="server">
      <ContentTemplate>
         <asp:Label ID="labelHae" runat="server" Text="Hae jä
            seniä:"></asp:Label>
         <asp:TextBox ID="textHae" runat="server"></asp:TextBox>
         <asp:DropDownList ID="selectKentta" runat="server">
         </asp:DropDownList>
         <asp:Button ID="buttonHae" runat="server" Text="Hae"
            OnClick="buttonHae_Click" />
      </ContentTemplate>
   </asp:UpdatePanel>

   <p>
      <asp:UpdateProgress ID="UpdateProgressHaku"
         AssociatedUpdatePanelID="updatePanelHaku"
         runat="server">
         <ProgressTemplate>
            Haetaan jäseniä...
            <asp:Image ID="imageLataus" runat="server" ImageUrl=
               "~/kuvat/ajax-loader.gif" />
         </ProgressTemplate>
      </asp:UpdateProgress>
   </p>

   <div id="jasenlista" class="jasenlista" runat="server">
      <p>
         Jäsenet:
      </p>
      <asp:UpdatePanel ID="updatePanelListJasenet" UpdateMode="
         Conditional" runat="server">
         <ContentTemplate>
            <asp:ListBox ID="listBoxJasenet" runat="server" Rows
               ="20" Width="130px"
               OnSelectedIndexChanged="
                  listBoxJasenet_SelectedIndexChanged"
                  AutoPostBack="True">
            </asp:ListBox>
         </ContentTemplate>
         <Triggers>
            <asp:AsyncPostBackTrigger ControlID="buttonHae"
               EventName="Click" />
            <asp:AsyncPostBackTrigger ControlID="buttonPoista"
               EventName="Click" />
            <asp:AsyncPostBackTrigger ControlID="buttonLisaa"
               EventName="Click" />
         </Triggers>
      </asp:UpdatePanel>
      <asp:Button ID="buttonUusiJasen" runat="server" Text="Lisä
         ä uusi jäsen" />
      <asp:Button ID="buttonPoista" runat="server" OnClick="
         buttonPoista_Click" Text="Poista jäsen" />
      <ajaxToolkit:ConfirmButtonExtender ID="poistoVarmistus"
         runat="server"
         TargetControlID="buttonPoista"
         ConfirmText="Poistetaanko valittu jäsen?" />
   </div>
```

```
<asp:UpdatePanel ID="UpdatePanelJasentiedot" UpdateMode="
    Conditional" runat="server">
    <ContentTemplate>
        <asp:Table ID="tableJasentiedot" runat="server"
            BorderStyle="None" Caption="" CaptionAlign="Top"
            CellPadding="5" CssClass="tableJasentiedot">
        </asp:Table>
        <asp:Table ID="tableHarrastukset" runat="server"
            BorderStyle="None" Caption="" CaptionAlign="Top"
            CellPadding="5" CssClass="tableHarrastukset"
                GridLines="Both">
        </asp:Table>
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID="buttonLisaa"
            EventName="Click" />
        <asp:AsyncPostBackTrigger ControlID="buttonHae"
            EventName="Click" />
        <asp:AsyncPostBackTrigger ControlID="buttonPoista"
            EventName="Click" />
        <asp:AsyncPostBackTrigger ControlID="listBoxJasenet"
            EventName="SelectedIndexChanged" />
    </Triggers>
</asp:UpdatePanel>

<ajaxToolkit:ModalPopupExtender ID="uusiJasenDialog" runat="
    server"
    TargetControlID="buttonUusiJasen"
    PopupControlID="panelUusiJasen"
    PopupDragHandleControlID="dialogOtsikko"
    BackgroundCssClass="modalDialogTausta">
</ajaxToolkit:ModalPopupExtender>

<asp:Panel ID="panelUusiJasen" Style="display: none" runat="
    server">
    <div id="panelUusiJasenSisalto" class="
        panelUusiJasenSisalto" runat="server">
        <div class="dialogOtsikko" id="dialogOtsikko">
            <p>
                Lisää uusi jäsen
            </p>
        </div>
        <uc1:JasenTietoLomake ID="JasenTietoLomake1" runat="
            server" />
        <div class="dialogButtons">
            <asp:UpdatePanel ID="updatePanelVirheIlmoitus"
                UpdateMode="Conditional" runat="server">
                <ContentTemplate>
                    <p id="uusiJasenVirheIlmoitus" class="
                        uusiJasenVirheIlmoitus" runat="server"></p>
                </ContentTemplate>
                <Triggers>
                    <asp:AsyncPostBackTrigger ControlID="
                        buttonLisaa" EventName="Click" />
                    <asp:AsyncPostBackTrigger ControlID="
                        buttonPeruuta" EventName="Click" />
                </Triggers>
            </asp:UpdatePanel>
            <asp:Button ID="buttonLisaa" OnClick="
                buttonLisaa_Click" runat="server" Text="Lisää jä
                sen" />
            <asp:Button ID="buttonPeruuta" OnClick="
                buttonPeruuta_Click" runat="server" Text="Peruuta
                " />
        </div>
```

```
         </div>
      </asp:Panel>

      <asp:UpdatePanel ID="updatePanelAnimaatio" UpdateMode="
         Conditional" runat="server">
      </asp:UpdatePanel>

      <ajaxToolkit:UpdatePanelAnimationExtender ID="upae" runat="
         server" TargetControlID="updatePanelAnimaatio">
         <Animations>
            <OnUpdated>
               <Sequence>
                  <Parallel duration="1">
                     <Color
                        AnimationTarget="listBoxJasenet"
                        Property="style"
                        PropertyKey="backgroundColor"
                        StartValue="#FFFF66"
                        EndValue="#FFFFFF" />
                     <Color
                        AnimationTarget="tableJasentiedot"
                        Property="style"
                        PropertyKey="backgroundColor"
                        StartValue="#FFFF66"
                        EndValue="#FFFFFF" />
                     <Color
                        AnimationTarget="tableHarrastukset"
                        Property="style"
                        PropertyKey="backgroundColor"
                        StartValue="#FFFF66"
                        EndValue="#FFFFFF" />
                  </Parallel>
               </Sequence>
            </OnUpdated>
         </Animations>
      </ajaxToolkit:UpdatePanelAnimationExtender>

      <ajaxToolkit:UpdatePanelAnimationExtender ID="upae2" runat="
         server" TargetControlID="updatePanelVirheIlmoitus">
         <Animations>
            <OnUpdated>
               <Sequence>
                  <FadeIn AnimationTarget="uusiJasenVirheIlmoitus"
                     Fps="20" />
                  <FadeOut AnimationTarget="uusiJasenVirheIlmoitus"
                      Fps="20" />
               </Sequence>
            </OnUpdated>
         </Animations>
      </ajaxToolkit:UpdatePanelAnimationExtender>
   </form>
</body>
</html>
```

F.4 kerho/Default.aspx.cs

```
ï»¿using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using KerhoApplication;

public partial class _Default : System.Web.UI.Page
{
   private Kerho kerho;
   private List<Jasen> loydetyt;
   private Jasen apuJasen;


   /// <summary>
   /// Suoritetaan sivun latauduttua ja jokaisella takaisinkutsulla
      .
   /// </summary>
   /// <param name="sender"></param>
   /// <param name="e"></param>
   protected void Page_Load(object sender, EventArgs e)
   {
      kerho = new Kerho();
      kerho.lueTiedostosta("kelmit");

      // Ettei suoriteta jokaisella takaisinkutsulla.
      if (!IsPostBack)
      {
         apuJasen = new Jasen();
         labelKerhoNimi.Text = kerho.getNimi();
         labelJasenia.Text = "Jäseniä: " + kerho.getJasenia().
            ToString();
         asetaHakukentat();

         if (Session["hakuteksti"] == null) loydetyt = kerho.etsi("
            ", apuJasen.ekaKentta());
         else
         {
            loydetyt = kerho.etsi((String)Session["hakuteksti"],
               Int32.Parse((String)Session["hakukentta"]));
            textHae.Text = (String)Session["hakuteksti"];
            selectKentta.SelectedValue = (String)Session["
               hakukentta"];
         }

         int valittuJasenId = 0;
         String query = Request.QueryString["id"];
         if (query != null) valittuJasenId = Int32.Parse(query);
         else if (loydetyt.Count > 0) valittuJasenId = loydetyt[0].
            getTunnusNro();

         asetaJasenLista(valittuJasenId);
      }
   }


   /// <summary>
   /// Kuuntelija hakunapin painalluksille.
   /// </summary>
   /// <param name="sender"></param>
   /// <param name="e"></param>
   protected void buttonHae_Click(object sender, EventArgs e)
   {
      System.Threading.Thread.Sleep(3000);
      loydetyt = kerho.etsi(textHae.Text, Int32.Parse(selectKentta.
         SelectedValue));
      Session["hakuteksti"] = textHae.Text;
      Session["hakukentta"] = selectKentta.SelectedValue;
```

```
        if (loydetyt.Count > 0) asetaJasenLista(loydetyt[0].
            getTunnusNro());
        else asetaJasenLista(-1);
        updatePanelAnimaatio.Update();
        ToolScriptManager1.AddHistoryPoint("jasen", listBoxJasenet.
            SelectedValue, "Kerho: Jäsen " + listBoxJasenet.
            SelectedValue);
    }


    /// <summary>
    /// Kuuntelija jäsenlistan valinnan muutoksille.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void listBoxJasenet_SelectedIndexChanged(object sender
        , EventArgs e)
    {
        // Lisätään jäsenen id sivuhistoriaan.
        if (ToolScriptManager1.IsInAsyncPostBack && !
            ToolScriptManager1.IsNavigating)
        {
            ToolScriptManager1.AddHistoryPoint("jasen", listBoxJasenet
                .SelectedValue, "Kerho: Jäsen " + listBoxJasenet.
                SelectedValue);
        }
        naytaJasenTiedot();
    }


    /// <summary>
    /// Kuuntelija historiatietojen muuttumiselle.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void ToolScriptManager1_Navigate(object sender,
        HistoryEventArgs e)
    {
        if (e.State["jasen"] == null) listBoxJasenet.SelectedValue =
            loydetyt[0].getTunnusNro().ToString();
        else listBoxJasenet.SelectedValue = e.State["jasen"].ToString
            ();

        naytaJasenTiedot();
        updatePanelListJasenet.Update();
        UpdatePanelJasentiedot.Update();
    }


    /// <summary>
    /// Asettaa hakukentät hakuvalintalistaan.
    /// </summary>
    public void asetaHakukentat()
    {
        for (int i = apuJasen.ekaKentta(); i < apuJasen.getKenttia();
             i++)
        {
            ListItem item = new ListItem(apuJasen.getKysymys(i), i.
                ToString());
            selectKentta.Items.Add(item);
        }
    }


    /// <summary>
```

```
/// Asettaa löydetyt jäsenet valintalistaan ja näyttää
/// valitun jäsenen tiedot tai tiedon jos yhtään jäsentä
/// ei löytynyt.
/// </summary>
/// <param name="jasenId">Mikä jäsen asetetaan valituksi</param>
public void asetaJasenLista(int jasenId)
{
   listBoxJasenet.Items.Clear();

   if (loydetyt.Count > 0)
   {
      foreach (Jasen jasen in loydetyt)
      {
         ListItem item = new ListItem(jasen.getNimi(), jasen.
            getTunnusNro().ToString());
         listBoxJasenet.Items.Add(item);
      }
      listBoxJasenet.SelectedValue = jasenId.ToString();
      naytaJasenTiedot();
   }
   else
   {
      naytaVirheIlmoitus();
   }
}


/// <summary>
/// Näyttää ilmoituksen jäsentietojen kohdalla ja piilottaa
   harrastustaulukon.
/// </summary>
public void naytaVirheIlmoitus()
{
   TableRow ekaRivi = new TableRow();
   ekaRivi.BorderStyle = BorderStyle.None;
   tableJasentiedot.Rows.Add(ekaRivi);

   TableCell ekaSolu = new TableCell();
   ekaSolu.Text = "Ei löytynyt yhtään jäsentä!";
   ekaSolu.Style.Add("font-weight", "bold");
   ekaSolu.BorderStyle = BorderStyle.None;
   ekaRivi.Cells.Add(ekaSolu);

   tableHarrastukset.Visible = false;
}


/// <summary>
/// Näyttää valitun jäsenen tiedot sivulla.
/// </summary>
public void naytaJasenTiedot()
{
   tableJasentiedot.Controls.Clear();
   tableHarrastukset.Controls.Clear();
   Jasen jasen = kerho.annaJasenId(Convert.ToInt32(
      listBoxJasenet.SelectedValue));
   luoTietoOtsikkorivi(jasen);

   for (int i = jasen.ekaKentta(); i < jasen.getKenttia(); i++)
   {
      luoTietorivi(jasen, i);
   }
   naytaJasenenHarrastukset(jasen);
}
```

```
/// <summary>
/// Luo otsikon jäsentietotaulukolle.
/// </summary>
/// <param name="jasen">Jäsen jolle taulukko luodaan</param>
public void luoTietoOtsikkorivi(Jasen jasen)
{
   TableRow otsikkoRivi = new TableRow();
   tableJasentiedot.Rows.Add(otsikkoRivi);

   TableCell otsikkoSolu = new TableCell();
   otsikkoSolu.Text = "Jäsen: " + jasen.getNimi();
   otsikkoSolu.Style.Add("font-weight", "bold");
   otsikkoSolu.Style.Add("text-decoration", "underline");
   otsikkoRivi.Cells.Add(otsikkoSolu);

   HyperLink link = new HyperLink();
   link.Text = "[Muokkaa]";
   link.NavigateUrl = "/jasensivu/default.aspx?id=" + jasen.
      getTunnusNro();
   TableCell linkkisolu = new TableCell();
   linkkisolu.Controls.Add(link);
   otsikkoRivi.Cells.Add(linkkisolu);
}


/// <summary>
/// Luo jäsentietotaulukkoon rivin jäsenen tiedolle.
/// </summary>
/// <param name="jasen">Jäsen jonka tieto lisätään</param>
/// <param name="i">Kenttä jonka tieto lisätään</param>
public void luoTietorivi(Jasen jasen, int i)
{
   TableRow tietoRivi = new TableRow();
   tableJasentiedot.Rows.Add(tietoRivi);

   TableCell kysymysSolu = new TableCell();
   kysymysSolu.Text = jasen.getKysymys(i) + ":";
   kysymysSolu.Style.Add("font-weight", "bold");
   kysymysSolu.Style.Add("text-align", "right");
   tietoRivi.Cells.Add(kysymysSolu);

   TableCell tietoSolu = new TableCell();
   tietoSolu.Text = jasen.anna(i);
   tietoRivi.Cells.Add(tietoSolu);
}


/// <summary>
/// Näyttää jäsenen harrastukset sivulla taulukossa.
/// </summary>
/// <param name="jasen">Jäsen jonka harrastukset näytetään.</
   param>
public void naytaJasenenHarrastukset(Jasen jasen)
{
   tableHarrastukset.Visible = true;
   List<Harrastus> harrastukset = kerho.annaHarrastukset(jasen);

   luoHarrastusOtsikkorivi();
   luoHarrastusKysymysRivi(harrastukset);
   luoHarrastusTietorivit(harrastukset);
}


/// <summary>
```

```
/// Luo otsikon harrastustaulukolle.
/// </summary>
public void luoHarrastusOtsikkorivi()
{
   TableRow ekaRivi = new TableRow();
   ekaRivi.BorderStyle = BorderStyle.None;
   tableHarrastukset.Rows.Add(ekaRivi);

   TableCell ekaSolu = new TableCell();
   ekaSolu.Text = "Harrastukset:";
   ekaSolu.Style.Add("font-weight", "bold");
   ekaSolu.BorderStyle = BorderStyle.None;
   ekaRivi.Cells.Add(ekaSolu);
}


/// <summary>
/// Luo harrastustaulukkoon rivin jossa harrastusten kysymykset.
/// </summary>
/// <param name="harrastukset">Lista harrastuksista</param>
public void luoHarrastusKysymysRivi(List<Harrastus> harrastukset
    )
{
   TableRow otsikkoRivi = new TableRow();
   tableHarrastukset.Rows.Add(otsikkoRivi);

   if (harrastukset.Count == 0)
   {
      TableCell tyhjaSolu = new TableCell();
      tyhjaSolu.HorizontalAlign = HorizontalAlign.Center;
      tyhjaSolu.Text = "Ei harrastuksia!";
      tyhjaSolu.BorderStyle = BorderStyle.Solid;
      otsikkoRivi.Cells.Add(tyhjaSolu);
      return;
   }

   for (int i = harrastukset[0].ekaKentta(); i < harrastukset
       [0].getKenttia(); i++)
   {
      TableCell otsikkoSolu = new TableCell();
      otsikkoSolu.HorizontalAlign = HorizontalAlign.Center;
      otsikkoSolu.Text = "<b>" + harrastukset[0].getKysymys(i) +
          "</b>";
      otsikkoSolu.BorderStyle = BorderStyle.Solid;
      otsikkoRivi.Cells.Add(otsikkoSolu);
   }
}


/// <summary>
/// Näyttää harrastustaulukossa harrastusten tiedot.
/// </summary>
/// <param name="harrastukset">Lista harrastuksista</param>
public void luoHarrastusTietorivit(List<Harrastus> harrastukset)
{
   foreach (Harrastus harrastus in harrastukset)
   {
      TableRow harrastusRivi = new TableRow();
      tableHarrastukset.Rows.Add(harrastusRivi);

      for (int i = harrastus.ekaKentta(); i < harrastus.
          getKenttia(); i++)
      {
         TableCell harrastusSolu = new TableCell();
         harrastusSolu.Text = harrastus.anna(i);
```

```
            if (i != harrastus.ekaKentta()) harrastusSolu.
               HorizontalAlign = HorizontalAlign.Center;
            harrastusSolu.BorderStyle = BorderStyle.Solid;
            harrastusRivi.Cells.Add(harrastusSolu);
         }
      }
   }


   /// <summary>
   /// Kuuntelija dialogin jäsenenlisäysnapille.
   /// </summary>
   /// <param name="sender"></param>
   /// <param name="e"></param>
   public void buttonLisaa_Click(object sender, EventArgs e)
   {
      apuJasen = new Jasen();
      loydetyt = kerho.etsi("", apuJasen.ekaKentta());

      if (JasenTietoLomake1.tarkistaJasenKentat() == 0 && !
         JasenTietoLomake1.Nimityhja)
      {
         apuJasen = JasenTietoLomake1.Jasen;
         apuJasen.rekisteroi();
         kerho.korvaaTaiLisaa(apuJasen);
         kerho.tallenna();
         loydetyt.Add(apuJasen);
         asetaJasenLista(apuJasen.getTunnusNro());
         // Näytetään animaatio
         updatePanelAnimaatio.Update();
         Session["hakuteksti"] = null;
         labelJasenia.Text = "Jäseniä: " + kerho.getJasenia().
            ToString();
         ToolScriptManager1.AddHistoryPoint("jasen", listBoxJasenet
            .SelectedValue, "Kerho: Jäsen " + listBoxJasenet.
            SelectedValue);

         suljeDialogi();
      }
      else if (JasenTietoLomake1.Nimityhja) uusiJasenVirheIlmoitus.
         InnerHtml = "Nimi ei saa olla tyhjä!";
      else uusiJasenVirheIlmoitus.InnerHtml = "Virheellisiä syö
         tteitä!";
   }


   /// <summary>
   /// Kuuntelija dialogin peruuta-napille.
   /// </summary>
   /// <param name="sender"></param>
   /// <param name="e"></param>
   public void buttonPeruuta_Click(object sender, EventArgs e)
   {
      suljeDialogi();
   }


   /// <summary>
   /// Sulkee dialogin ja tyhjentää dialogin lomakkeen kentät.
   /// </summary>
   public void suljeDialogi()
   {
      apuJasen = new Jasen();
      JasenTietoLomake1.Jasen = apuJasen;
      JasenTietoLomake1.paivitaLomake();
```

```
      uusiJasenVirheIlmoitus.InnerHtml = "";

      uusiJasenDialog.Hide();
   }


   /// <summary>
   /// Kuuntelija jäsenen poistamisnapille.
   /// </summary>
   /// <param name="sender"></param>
   /// <param name="e"></param>
   protected void buttonPoista_Click(object sender, EventArgs e)
   {
      apuJasen = new Jasen();
      loydetyt = kerho.etsi("", apuJasen.ekaKentta());

      int id = Convert.ToInt32(listBoxJasenet.SelectedValue);

      loydetyt.Remove(kerho.annaJasenId(id));
      kerho.poista(id);
      kerho.tallenna();
      asetaJasenLista(loydetyt[0].getTunnusNro());
      labelJasenia.Text = "Jäseniä: " + kerho.getJasenia().ToString
          ();

      // Näytetään animaatio
      updatePanelAnimaatio.Update();

      ToolScriptManager1.AddHistoryPoint("jasen", listBoxJasenet.
          SelectedValue, "Kerho: Jäsen " + listBoxJasenet.
          SelectedValue);
      Session["hakuteksti"] = null;
   }
}
```

F.5 jasensivu/Default.aspx

```
ï»¿<%@ Page Title="Kerho" Language="C#" AutoEventWireup="true"
    CodeFile="Default.aspx.cs"
   Inherits="_Default" %>

<%@ Register TagPrefix="ajaxToolkit" Namespace="AjaxControlToolkit"
     Assembly="AjaxControlToolkit" %>
<%@ Register Src="../JasenTietoLomake.ascx" TagName="
    JasenTietoLomake" TagPrefix="uc1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
    http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
   <link href="../Styles/jasen.css" rel="stylesheet" type="text/css
       " />
   <title>Kerho</title>
</head>
<body>
   <form id="form1" runat="server">
      <ajaxToolkit:ToolkitScriptManager runat="Server" />

      <asp:UpdatePanel ID="updatePanelIlmoitus" UpdateMode="
          Conditional" runat="server">
         <ContentTemplate>
```

```
            <asp:Label runat="server" ID="labelIlmoitus" CssClass="
                labelIlmoitus"></asp:Label>
        </ContentTemplate>
        <Triggers>
            <asp:AsyncPostBackTrigger ControlID="buttonTallenna"
                EventName="Click" />
        </Triggers>
    </asp:UpdatePanel>

    <h1>
        <asp:Label runat="server" ID="labelJasenNimi"></asp:Label>
    </h1>

    <div id="jasenLomake" class="jasenTietolomake" runat="server"
        >
        <uc1:JasenTietoLomake ID="JasenTietoLomakeJasensivu" runat
            ="server" />
        <p>
            <asp:Button ID="buttonTallenna" runat="server" OnClick=
                "buttonTallenna_Click" Text="Tallenna jäsen" />
        </p>
        <p>
            <asp:HyperLink ID="linkTakaisin" runat="server">&lt;-
                Takaisin kerhoon</asp:HyperLink>
        </p>
    </div>

    <div id="harrastusLomake" class="harrastusLomake" runat="
        server">
        <asp:UpdatePanel ID="updatePanelharrastusLomake"
            UpdateMode="Conditional" runat="server">
            <ContentTemplate>
                <asp:Table ID="tableHarrastusLomake" runat="server"
                    BorderStyle="None" Caption=""
                    CaptionAlign="Top" CellPadding="5" CssClass="
                        tableHarrastusLomake" GridLines="Both">
                </asp:Table>
            </ContentTemplate>
            <Triggers>
                <asp:AsyncPostBackTrigger ControlID="buttonLisaa"
                    EventName="Click" />
                <asp:AsyncPostBackTrigger ControlID="
                    buttonPoistaHarrastus" EventName="Click" />
            </Triggers>
        </asp:UpdatePanel>
        <p>
            <asp:Button ID="buttonUusiHarrastus" runat="server"
                Text="Lisää harrastus" />
            <asp:Button ID="buttonPoistaHarrastus" runat="server"
                OnClick="buttonPoistaHarrastus_Click"
                Text="Poista harrastus" />
            <ajaxToolkit:ConfirmButtonExtender ID="
                harrastusPoistoVarmistus" runat="server"
                TargetControlID="buttonPoistaHarrastus"
                ConfirmText="Poistetaanko valittu harrastus?" />
        </p>
    </div>

    <ajaxToolkit:ModalPopupExtender ID="uusiHarrastusDialog"
        runat="server"
        TargetControlID="buttonUusiHarrastus"
        PopupControlID="panelUusiHarrastus"
        PopupDragHandleControlID="dialogOtsikko"
        BackgroundCssClass="modalDialogTausta">
    </ajaxToolkit:ModalPopupExtender>
```

```
<asp:Panel ID="panelUusiHarrastus" Style="display: none"
    runat="server">
    <div id="panelUusiHarrastusSisalto" class="
        panelUusiHarrastusSisalto" runat="server">
        <div class="dialogOtsikko" id="dialogOtsikko">
            <p>
                Lisää uusi harrastus
            </p>
        </div>
        <asp:UpdatePanel ID="updatePanelUusiHarrastuslomake"
            UpdateMode="Conditional" runat="server">
            <ContentTemplate>
                <asp:Table ID="tableUusiHarrastusLomake" runat="
                    server" BorderStyle="None" Caption=""
                     CaptionAlign="Top" CellPadding="0" CssClass="
                        tableUusiHarrastusLomake">
                </asp:Table>
            </ContentTemplate>
        </asp:UpdatePanel>
        <div class="dialogButtons">
            <asp:UpdatePanel ID="updatePanelVirheIlmoitus"
                UpdateMode="Conditional" runat="server">
                <ContentTemplate>
                    <p id="uusiharrastusVirheIlmoitus" class="
                        uusiHarrastusVirheIlmoitus" runat="server"
                        ></p>
                </ContentTemplate>
                <Triggers>
                    <asp:AsyncPostBackTrigger ControlID="
                        buttonLisaa" EventName="Click" />
                    <asp:AsyncPostBackTrigger ControlID="
                        buttonPeruuta" EventName="Click" />
                </Triggers>
            </asp:UpdatePanel>
            <asp:Button ID="buttonLisaa" runat="server" OnClick=
                "buttonLisaa_Click" Text="Lisää Harrastus" />
            <asp:Button ID="buttonPeruuta" runat="server"
                OnClick="buttonPeruuta_Click" Text="Peruuta" />
        </div>
    </div>
</asp:Panel>

<asp:UpdatePanel ID="updatePanelAnimaatio" UpdateMode="
    Conditional" runat="server">
</asp:UpdatePanel>

<ajaxToolkit:UpdatePanelAnimationExtender ID="upae1" runat="
    server" TargetControlID="updatePanelAnimaatio">
    <Animations>
        <OnUpdated>
            <Sequence>
                <Parallel duration="1">
                    <Color
                        AnimationTarget="tableHarrastusLomake"
                        Property="style"
                        PropertyKey="backgroundColor"
                        StartValue="#FFFF66"
                        EndValue="#FFFFFF" />
                </Parallel>
            </Sequence>
        </OnUpdated>
    </Animations>
</ajaxToolkit:UpdatePanelAnimationExtender>
```

```
      <ajaxToolkit:UpdatePanelAnimationExtender ID="upae2" runat="
         server" TargetControlID="updatePanelIlmoitus">
         <Animations>
            <OnUpdated>
               <Sequence>
                  <FadeIn AnimationTarget="labelIlmoitus" Fps="20"
                     />
                  <Color
                     AnimationTarget="labelIlmoitus"
                     Property="style"
                     PropertyKey="backgroundColor"
                     StartValue="#FFFF66"
                     EndValue="#FFFFFF" />
               </Sequence>
            </OnUpdated>
         </Animations>
      </ajaxToolkit:UpdatePanelAnimationExtender>

      <ajaxToolkit:UpdatePanelAnimationExtender ID="upae3" runat="
         server" TargetControlID="updatePanelVirheIlmoitus">
         <Animations>
            <OnUpdated>
               <Sequence>
                  <FadeIn AnimationTarget="
                     uusiharrastusVirheIlmoitus" Fps="20" />
                  <FadeOut AnimationTarget="
                     uusiharrastusVirheIlmoitus" Fps="20" />
               </Sequence>
            </OnUpdated>
         </Animations>
      </ajaxToolkit:UpdatePanelAnimationExtender>
   </form>
</body>
</html>
```

F.6 jasensivu/Default.aspx.cs

```
ï»¿using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using KerhoApplication;

public partial class _Default : System.Web.UI.Page
{
   private Kerho kerho;
   private int jasenId;
   private Jasen jasen;
   private List<Harrastus> harrastukset;
   private Harrastus apuHarrastus;
   private int tallennusVirheita;
   private TextBox alaKentta;


   /// <summary>
   /// Suoritetaan kun sivu luodaan ja jokaisella takaisinkutsulla.
   /// </summary>
   /// <param name="sender"></param>
   /// <param name="e"></param>
```

```
protected void Page_Init(object sender, System.EventArgs e)
{
   String query = Request.QueryString["id"];
   if (query != null) jasenId = Int32.Parse(query);

   kerho = new Kerho();
   kerho.lueTiedostosta("kelmit");
   jasen = kerho.annaJasenId(jasenId);
   JasenTietoLomakeJasensivu.Jasen = jasen;
}


/// <summary>
/// Suoritetaan sivun latauduttua ja jokaisella takaisinkutsulla
   .
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void Page_Load(object sender, EventArgs e)
{
   // Ettei suoriteta jokaisella takaisinkutsulla.
   if (!IsPostBack)
   {
      harrastukset = kerho.annaHarrastukset(jasen);
      Session["harrastukset"] = harrastukset;

      labelJasenNimi.Text = jasen.getNimi();
      linkTakaisin.NavigateUrl = "../kerho/default.aspx?id=" +
         jasen.getTunnusNro();
   }
   // Suoritetaan vain takaisinkutsuissa.
   else
   {
      harrastukset = Session["harrastukset"] as List<Harrastus>;
   }

   naytaHarrastusMuokkausLomake();
   // Myös dialogin sisältö pitää luoda uudelleen jokaisella
      takaisinkutsulla.
   luoUusiHarrastusLomake();
}


/// <summary>
/// Näyttää harrastustenmuokkauslomakkeen.
/// </summary>
public void naytaHarrastusMuokkausLomake()
{
   tableHarrastusLomake.Controls.Clear();

   luoHarrastusMuokkausOtsikkorivi();
   luoHarrastusMuokkausKysymysRivi();
   luoHarrastusMuokkausKenttarivit();
}


/// <summary>
/// Luo otsikon harrastustaulukolle.
/// </summary>
public void luoHarrastusMuokkausOtsikkorivi()
{
   TableRow ekaRivi = new TableRow();
   ekaRivi.BorderStyle = BorderStyle.None;
   tableHarrastusLomake.Rows.Add(ekaRivi);
```

```
        TableCell ekaSolu = new TableCell();
        ekaSolu.Text = "Harrastukset:";
        ekaSolu.Style.Add("font-weight", "bold");
        ekaSolu.BorderStyle = BorderStyle.None;
        ekaRivi.Cells.Add(ekaSolu);
    }


    /// <summary>
    /// Luo harrastustaulukkoon rivin jossa harrastusten kysymykset.
    /// </summary>
    public void luoHarrastusMuokkausKysymysRivi()
    {
        TableRow otsikkoRivi = new TableRow();
        tableHarrastusLomake.Rows.Add(otsikkoRivi);

        if (harrastukset.Count == 0)
        {
            TableCell tyhjaSolu = new TableCell();
            tyhjaSolu.HorizontalAlign = HorizontalAlign.Center;
            tyhjaSolu.Text = "Ei harrastuksia!";
            tyhjaSolu.BorderStyle = BorderStyle.Solid;
            otsikkoRivi.Cells.Add(tyhjaSolu);
            return;
        }

        for (int i = harrastukset[0].ekaKentta(); i < harrastukset
            [0].getKenttia(); i++)
        {
            TableCell otsikkoSolu = new TableCell();
            otsikkoSolu.HorizontalAlign = HorizontalAlign.Center;
            otsikkoSolu.Text = "<b>" + harrastukset[0].getKysymys(i) +
                 "</b>";
            otsikkoSolu.BorderStyle = BorderStyle.Solid;
            otsikkoRivi.Cells.Add(otsikkoSolu);
        }
    }


    /// <summary>
    /// Näyttää harrastustaulukossa harrastustenmuokkauskentat.
    /// </summary>
    public void luoHarrastusMuokkausKenttarivit()
    {
        foreach (Harrastus harrastus in harrastukset)
        {
            TableRow harrastusRivi = new TableRow();
            tableHarrastusLomake.Rows.Add(harrastusRivi);

            for (int i = harrastus.ekaKentta(); i < harrastus.
                getKenttia(); i++)
            {
                TableCell harrastusSolu = new TableCell();
                if (i != harrastus.ekaKentta()) harrastusSolu.
                    HorizontalAlign = HorizontalAlign.Center;
                harrastusSolu.BorderStyle = BorderStyle.Solid;

                TextBox textTieto = luoHarrastusMuokkausKentta(
                    harrastus, i);
                Label labelVirhe = luoHarrastusMuokkausVirheLabel(
                    harrastus, i);
                harrastusSolu.Controls.Add(textTieto);
                harrastusSolu.Controls.Add(labelVirhe);
                harrastusRivi.Cells.Add(harrastusSolu);
```

```
            luoHarrastusMuokkausTriggeri(harrastus, i);
        }
    }
}


/// <summary>
/// Luo harrastusmuokkauskentän.
/// </summary>
/// <param name="harrastus">Harrastus jolle kenttä luodaan</
    param>
/// <param name="i">Kysymys jolle kenttä luodaan</param>
/// <returns>Tekstikenttä harrastuksen kysymykselle</returns>
public TextBox luoHarrastusMuokkausKentta(Harrastus harrastus,
    int i)
{
    TextBox textTieto = new TextBox();
    textTieto.ID = harrastus.getId() + "," + i.ToString();
    textTieto.AutoPostBack = true;
    textTieto.TextChanged += new EventHandler(
        textHarrastusMuokkaus_TextChanged);
    textTieto.Text = harrastus.anna(i);
    if (i != harrastus.ekaKentta()) textTieto.Width = (Unit.Pixel
        (50));
    else textTieto.Width = (Unit.Pixel(120));
    return textTieto;
}


/// <summary>
/// Luo harrastuskentälle virheilmoituslabelin.
/// </summary>
/// <param name="harrastus">Harrastus jolle label luodaan</param
    >
/// <param name="i">Kysymys jolle label luodaan</param>
/// <returns>Label harrastuskentän virheelle</returns>
public Label luoHarrastusMuokkausVirheLabel(Harrastus harrastus,
     int i)
{
    Label labelVirhe = new Label();
    labelVirhe.ID = "virhe" + harrastus.getId() + "," + i.
        ToString();
    labelVirhe.CssClass = "virheTeksti";
    return labelVirhe;
}


/// <summary>
/// Luo updatepaneliin triggerin harrastusmuokkauskentälle.
/// </summary>
/// <param name="harrastus">Harrastus jolle trigger luodaan</
    param>
/// <param name="i">Harrastuksen kenttä jolle trigger luodaan</
    param>
public void luoHarrastusMuokkausTriggeri(Harrastus harrastus,
    int i)
{
    AsyncPostBackTrigger trig = new AsyncPostBackTrigger();
    trig.ControlID = harrastus.getId() + "," + i.ToString();
    trig.EventName = "TextChanged";
    updatePanelharrastusLomake.Triggers.Add(trig);
}


/// <summary>
```

```
/// Kuuntelija harrastuskenttien tekstin vaihtumiselle.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void textHarrastusMuokkaus_TextChanged(object sender,
    EventArgs e)
{
   TextBox box = (TextBox)sender;
   asetaJaTarkistaHarrastusMuokkausKentta(box);
}


/// <summary>
/// Asettaa tekstikentän sisällön harrastukseen ja tarkistaa
    virheen.
/// </summary>
/// <param name="box">Tekstikenttä jonka sisältö asetetaan</
    param>
public void asetaJaTarkistaHarrastusMuokkausKentta(TextBox box)
{
   string[] osat = box.ID.Split(',');
   String virhe = "";

   foreach (Harrastus har in harrastukset)
   {
      if (har.getId() == Int32.Parse(osat[0]))
      {
         virhe = har.aseta(Int32.Parse(osat[1]), box.Text);
         Session["valittuHarrastusID"] = har.getId();
         break;
      }
   }
   Session["harrastukset"] = harrastukset;

   if (virhe != null && box.CssClass.Equals(""))
   {
      box.CssClass = "virheTieto";
   }
   else if (virhe == null && box.CssClass.Equals("virheTieto"))
   {
      box.CssClass = "";
   }

   Label label = (Label)FindControl("virhe" + box.ID);
   if (virhe != null) label.Text = virhe;
   else label.Text = "";
}


/// <summary>
/// Tallentaa jäsenen ja harrastukset jos kentissä ei ole
    virheitä.
/// </summary>
public void tallennaJasen()
{
   tallennusVirheita = 0;
   asetaKentat();

   if (tallennusVirheita == 0)
   {
      labelIlmoitus.Text = "Jäsen tallennettu!";
      labelIlmoitus.CssClass = "onnistumisIlmoitus";
      jasen = JasenTietoLomakeJasensivu.Jasen;
      kerho.korvaaTaiLisaa(jasen);
      kerho.tallenna();
```

```
    }
    else
    {
        labelIlmoitus.Text = "Virheellisiä syötteitä!";
        labelIlmoitus.CssClass = "virheIlmoitus";
    }
}


/// <summary>
/// Asettaa jasenen ja harrastusten kenttiin tekstikenttien
    arvot ja tarkistaa virheet.
/// Korvaa harrastukset kerhoon.
/// </summary>
public void asetaKentat()
{
    tallennusVirheita = JasenTietoLomakeJasensivu.
        tarkistaJasenKentat();
    asetaHarrastusKentat();
}


/// <summary>
/// Asettaa harrastusten kenttiin tekstikenttien arvot ja
    tarkistaa virheet.
/// Korvaa harrastukset kerhoon.
/// </summary>
public void asetaHarrastusKentat()
{
    List<Harrastus> kerhonHarrastusLista = kerho.annaHarrastukset
        (jasen);

    // Poistetaan vanhat harrastukset kerhosta.
    foreach (Harrastus har in kerhonHarrastusLista)
    {
        kerho.poistaHarrastus(har);
    }

    // Tarkistetaan ja lisätään uudet harrastukset kerhoon.
    foreach (Harrastus har in harrastukset)
    {
        for (int i = har.ekaKentta(); i < har.getKenttia(); i++)
        {
            TextBox box = (TextBox)FindControl(har.getId() + "," +
                i.ToString());
            String virhe = har.aseta(i, box.Text);

            if (virhe != null) tallennusVirheita++;
        }
        kerho.lisaa(har);
    }
}


/// <summary>
/// Tallentaa jäsenen kerhoon.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
public void buttonTallenna_Click(object sender, EventArgs e)
{
    tallennaJasen();
}
```

```
/// <summary>
/// Luo uuden harrastuksen luomislomakkeen.
/// </summary>
public void luoUusiHarrastusLomake()
{
   apuHarrastus = new Harrastus(jasen.getTunnusNro().ToString())
      ;
   for (int i = apuHarrastus.ekaKentta(); i < apuHarrastus.
      getKenttia(); i++)
   {
      luoUusiHarrastusLomakerivi(i);
   }
}


/// <summary>
/// Luo harrastuslomaketaulukkoon rivin.
/// </summary>
/// <param name="i">Kenttä jonka tieto lisätään</param>
public void luoUusiHarrastusLomakerivi(int i)
{
   TableRow tietoRivi = new TableRow();
   tableUusiHarrastusLomake.Rows.Add(tietoRivi);

   TableCell kysymysSolu = luoUusiHarrastusKysymysSolu(i);
   TableCell tietoSolu = new TableCell();
   TableCell virheSolu = luoUusiHarrastusVirheSolu(i);
   TextBox textTieto = luoUusiHarrastusKentta(i);

   tietoSolu.Controls.Add(textTieto);
   tietoRivi.Cells.Add(kysymysSolu);
   tietoRivi.Cells.Add(tietoSolu);
   tietoRivi.Cells.Add(virheSolu);

   luoUusiHarrastustietoTriggeri(i);
}


/// <summary>
/// Luo taulukkosolun harrastuksen kysymyksellä.
/// </summary>
/// <param name="i">Kysymys jolle solu luodaan</param>
/// <returns>Taulukkosolu harrastuksen kysymyksellä</returns>
public TableCell luoUusiHarrastusKysymysSolu(int i)
{
   TableCell kysymysSolu = new TableCell();
   kysymysSolu.Text = apuHarrastus.getKysymys(i) + ":";
   kysymysSolu.Style.Add("text-align", "right");
   return kysymysSolu;
}


/// <summary>
/// Luo harrastustietokentän.
/// </summary>
/// <param name="i">Kysymys jolle kenttä luodaan</param>
/// <returns>Tekstikenttä harrastuksen kysymykselle</returns>
public TextBox luoUusiHarrastusKentta(int i)
{
   TextBox textTieto = new TextBox();
   textTieto.ID = "uusiharrastus," + i.ToString();
   textTieto.AutoPostBack = true;
   textTieto.TextChanged += new EventHandler(
      textUusiharrastus_TextChanged);
   textTieto.Text = apuHarrastus.anna(i);
```

```
        if (i != apuHarrastus.ekaKentta()) textTieto.Width = (Unit.
            Pixel(50));
        else textTieto.Width = (Unit.Pixel(120));
        if (i == apuHarrastus.ekaKentta()) alaKentta = textTieto;
        return textTieto;
    }


    /// <summary>
    /// Luo harrastuskentälle virheilmoituslabelin.
    /// </summary>
    /// <param name="harrastus">Harrastus jolle label luodaan</param
        >
    /// <param name="i">Kysymys jolle label luodaan</param>
    /// <returns>Label harrastuskentän virheelle</returns>
    public TableCell luoUusiHarrastusVirheSolu(int i)
    {
        TableCell virheSolu = new TableCell();
        virheSolu.ID = "uusiharrastusvirhe" + i.ToString();
        virheSolu.CssClass = "virheTeksti";
        return virheSolu;
    }


    /// <summary>
    /// Luo updatepaneliin triggerin harrastuksen tietomuokkauskentä
        lle.
    /// </summary>
    /// <param name="i">Kysymys jolle trigger luodaan</param>
    public void luoUusiHarrastustietoTriggeri(int i)
    {
        AsyncPostBackTrigger trig = new AsyncPostBackTrigger();
        trig.ControlID = "uusiharrastus," + i.ToString();
        trig.EventName = "TextChanged";
        updatePanelUusiHarrastuslomake.Triggers.Add(trig);
    }


    /// <summary>
    /// Kuuntelija harrastustietokenttien tekstin vaihtumiselle.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void textUusiharrastus_TextChanged(object sender,
        EventArgs e)
    {
        TextBox box = (TextBox)sender;
        asetaJaTarkistaUusiharrastus(box);
    }


    /// <summary>
    /// Asettaa tekstikentän sisällön harrastuksen kenttään ja
        tarkistaa ja näyttää virheen.
    /// </summary>
    /// <param name="box">Tekstikenttä jonka sisältö asetetaan</
        param>
    public void asetaJaTarkistaUusiharrastus(TextBox box)
    {
        string[] osat = box.ID.Split(',');
        int kentta = Int32.Parse(osat[1]);
        apuHarrastus = new Harrastus(jasen.getTunnusNro().ToString())
            ;
        String virhe = apuHarrastus.aseta(kentta, box.Text);
```

```
      if (virhe != null && box.CssClass.Equals(""))
      {
         box.CssClass = "virheTieto";
      }

      else if (virhe == null && box.CssClass.Equals("virheTieto"))
      {
         box.CssClass = "";
      }

      TableCell cell = (TableCell)FindControl("uusiharrastusvirhe"
         + kentta.ToString());
      if (virhe != null) cell.Text = virhe;
      else cell.Text = "";
   }


   /// <summary>
   /// Asettaa harrastuksen kenttiin tekstikenttien arvot ja
      tarkistaa virheet.
   /// </summary>
   public int tarkistaUusiharrastusKentat()
   {
      int uusiharrastusVirheita = 0;
      apuHarrastus = new Harrastus(jasen.getTunnusNro().ToString())
         ;

      for (int i = apuHarrastus.ekaKentta(); i < apuHarrastus.
         getKenttia(); i++)
      {
         TextBox box = (TextBox)FindControl("uusiharrastus," + i.
            ToString());
         String virhe = apuHarrastus.aseta(i, box.Text);

         if (virhe != null) uusiharrastusVirheita++;
      }

      return uusiharrastusVirheita;
   }


   /// <summary>
   /// Kuuntelija dialogin harrastuksen lisäysnapille.
   /// </summary>
   /// <param name="sender"></param>
   /// <param name="e"></param>
   public void buttonLisaa_Click(object sender, EventArgs e)
   {
      if (tarkistaUusiharrastusKentat() == 0 && !alaKentta.Text.
         Equals(""))
      {
         apuHarrastus.rekisteroi();
         harrastukset.Add(apuHarrastus);
         Session["harrastukset"] = harrastukset;
         naytaHarrastusMuokkausLomake();
         updatePanelAnimaatio.Update();
         suljeDialogi();
      }
      else if (alaKentta.Text.Equals(""))
         uusiharrastusVirheIlmoitus.InnerHtml = "Ala ei saa olla
         tyhjä!";
      else uusiharrastusVirheIlmoitus.InnerHtml = "Virheellisiä syö
         tteitä!";
   }
```

```
    /// <summary>
    /// Kuuntelija dialogin peruuta-napille.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    public void buttonPeruuta_Click(object sender, EventArgs e)
    {
        suljeDialogi();
    }


    /// <summary>
    /// Sulkee dialogin ja tyhjentää dialogin lomakkeen kentät.
    /// </summary>
    public void suljeDialogi()
    {
        apuHarrastus = new Harrastus(jasen.getTunnusNro().ToString())
            ;
        uusiharrastusVirheIlmoitus.InnerHtml = "";
        paivitaUusiharrastusLomake();

        uusiHarrastusDialog.Hide();
    }


    /// <summary>
    /// Luo uuden harrastuksen lomakkeen uudelleen.
    /// </summary>
    public void paivitaUusiharrastusLomake()
    {
        tableUusiHarrastusLomake.Controls.Clear();
        luoUusiHarrastusLomake();
        updatePanelUusiHarrastuslomake.Update();
    }


    /// <summary>
    /// Poistaa valitun harrastuksen kerhosta.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void buttonPoistaHarrastus_Click(object sender,
        EventArgs e)
    {
        foreach (Harrastus har in harrastukset)
        {
            if (har.getId() == (int)Session["valittuHarrastusID"])
            {
                harrastukset.Remove(har);
                Session["harrastukset"] = harrastukset;

                naytaHarrastusMuokkausLomake();
                updatePanelAnimaatio.Update();
                return;
            }
        }
    }
}
```

G Ruby on Rails -kerhon lähdekoodit

G.1 views/kerho/kerhosivu.html.erb

```
<?xml version="1.0" encoding="UTF-8"?>

<%= javascript_include_tag "kerho" %>

<h1> <%= @kerhonimi %> </h1>
<h3 id="jasenmaara"> Jäseniä: <%= @jasenia %> </h3>

<p>
   Hae jäseniä:
   <%= form_tag(kerho_hae_jasenet_path, remote: true) do %>
      <%= text_field_tag(:hakuehto, @hakuehto) %>
      <%= select_tag(:hakukentta, options_for_select(@kentat,
         @valittukentta)) %>
      <%= submit_tag 'Hae', :onclick => "nayta_odotusIlmaisin()" %>
   <% end %>
</p>

<p id="odotusIlmaisin" style="display:none">
   Haetaan jasenia... <img src="<%= asset_path( 'ajax-loader.gif' )
       %>" />
</p>

<div id="jasenlista" class="jasenlista">
   <p id="jasenValinnat">
      <%= render partial: "jasenlista" %>
   </p>
   <%= link_to_modal "Lisää uusi jäsen", "#uusijasendialog" %>
   </br>
   <%= link_to 'Poista jäsen', kerho_poista_jasen_path, :remote =>
      true, :data => { confirm: "Poistetaanko valittu jäsen?" } %>
</div>

<div id="uusijasendialog" title="Uusi jasen" style="display:none">
   <%= render partial: "uusijasenlomake" %>
</div>

<div id="jasentiedot" class="jasentiedot">
   <%= render partial: "jasentiedot" %>
</div>

<div id="harrastukset" class="harrastukset">
   <%= render partial: "harrastukset" %>
</div>
```

G.2 views/kerho/_harrastukset.html.erb

```
<?xml version="1.0" encoding="UTF-8"?>

<% if @valittujasen != "" %>
   <% if @harrastukset.size > 0 %>
      <table class="tableHarrastukset">
         <caption><b>Harrastukset:</b></caption>
         <tr>
            <% (@harrastukset[0].eka_kentta..@harrastukset[0].
               get_kenttia-1).each do |i| %>
```

```
                <td><b><%= @harrastukset[0].get_kysymys(i) %></b></
                    td>
             <% end %>
          </tr>

          <% @harrastukset.each do |harrastus| %>
             <tr>
                <% (harrastus.eka_kentta..harrastus.get_kenttia-1).
                    each do |i| %>
                   <% if i == harrastus.eka_kentta %>
                      <td class="tableEka"> <%= harrastus.anna(i) %>
                          </td>
                   <% else %>
                      <td> <%= harrastus.anna(i) %> </td>
                   <% end %>
                <% end %>
             </tr>
          <% end %>
       </table>
    <% end %>
<% end %>
```

G.3 views/kerho/_jasenlista.html.erb

```
<?xml version="1.0" encoding="UTF-8"?>

<b>Jäsenet:</b></br>
<%= select_tag(:select_jasen, options_for_select(@jasenlista_alkiot
    , @jasenid),
       {size: 20, :class => 'selectJasen', :onchange => "
          paivita_jasentiedot(this.value)"}) %>
```

G.4 views/kerho/_jasentiedot.html.erb

```
<?xml version="1.0" encoding="UTF-8"?>

<% if @valittujasen != "" %>
   <p>
      <b>Jasen: <%= @valittujasen.get_nimi %> </b>
       <%= link_to "[Muokkaa]", jasen_jasensivu_path(id:
           @valittujasen.get_tunnusnro) %>
   </p>

   <table class="tableJasenLomake">
      <% (@valittujasen.eka_kentta..@valittujasen.get_kenttia-1).
          each do |i| %>
          <tr>
                <td><b><%= @valittujasen.get_kysymys(i)%>:</b></td>
                <td><%= @valittujasen.anna(i) %></td>
          </tr>
      <% end %>
   </table>
<% else %>
   <p>
      <b>Hakuehdoilla ei löytynyt yhtään jäsenta!</b>
   </p>
<% end %>
```

G.5 views/kerho/_uusijasenlomake.html.erb

```
<?xml version="1.0" encoding="UTF-8"?>

<h1>Lisää uusi jäsen</h1>

<%= form_tag(kerho_jasendialogi_lahetys_path, remote: true) do %>

   <%= render_cell :jasentietolomake, :nayta, :jasen =>
      @lomakejasen %>

   <p id="uusijasenilmoitus" class="virheilmoitus"></p>
   <%= submit_tag 'Peruuta' %>
   <%= submit_tag 'Lisää jäsen' %>

<% end %>
```

G.6 views/kerho/aseta_kentta.js.erb

```
$("#virhe<%= escape_javascript(@i) %>").html("<%= escape_javascript
    ( @virhe ) %>");

var onkoVirhetyyli = $("#<%= escape_javascript( @i ) %>").hasClass(
    "virheKentta");

<% if @virhe != nil %>

   if (!onkoVirhetyyli) {
      $("#<%= escape_javascript( @i ) %>").addClass("virheKentta");
   }

<% else %>

   if (onkoVirhetyyli) {
      $("#<%= escape_javascript( @i ) %>").removeClass("virheKentta
          ");
   }

<% end %>
```

G.7 views/kerho/aseta_urlhash.js.erb

```
window.location.hash = '<%= @hashid %>';
```

G.8 views/kerho/hae_jasenet.js.erb

```
window.location.hash = '<%= @hashid %>';
```

```
$("#jasenValinnat").html("<%= escape_javascript( render(:partial =>
     "jasenlista")) %>");
$("#jasentiedot").html("<%= escape_javascript( render(:partial => "
    jasentiedot")) %>");
$("#harrastukset").html("<%= escape_javascript( render(:partial =>
    "harrastukset")) %>");

$("#jasenValinnat").fadeOut(500);
$("#jasenValinnat").fadeIn(500);

$("#jasentiedot").fadeOut(500);
$("#jasentiedot").fadeIn(500);

$("#harrastukset").fadeOut(500);
$("#harrastukset").fadeIn(500);

$('#odotusIlmaisin').hide();
```

G.9 views/kerho/jasendialogi_lahetys.js.erb

```
<% if @toiminto == "sulje" %>
   $.modal.close();
   <!-- Luodaan lomake uudelleen, jotta saadaan uudella aukaisulla
      tyhjat tekstikentat. -->
   $("#uusijasendialog").html("<%= escape_javascript( render(:
      partial => "uusijasenlomake")) %>");
<% else %>

   $("#uusijasenilmoitus").html("<%= escape_javascript( @ilmoitus )
       %>");

   <% if @virheita == true %>

      $("#uusijasenilmoitus").animate({
         backgroundColor: "white",
         color: "#fff"
      }, 500 );

      $("#uusijasenilmoitus").animate({
         backgroundColor: "#aa0000",
         color: "#fff"
      }, 1000 );

   <% else %>

      $.modal.close();
      window.location.hash = '<%= @hashid %>';

      $("#uusijasendialog").html("<%= escape_javascript( render(:
         partial => "uusijasenlomake")) %>");
      <!-- Paivitetaan jasenlista ja jasentiedot sivulle -->
      $("#jasenValinnat").html("<%= escape_javascript( render(:
         partial => "jasenlista")) %>");
      $("#jasentiedot").html("<%= escape_javascript( render(:
         partial => "jasentiedot")) %>");
      $("#harrastukset").html("<%= escape_javascript( render(:
         partial => "harrastukset")) %>");
      $("#jasenmaara").html("Jasenia: <%= escape_javascript(
         @jasenmaara.to_s ) %>");
```

```
      $("#jasenValinnat").fadeOut(500);
      $("#jasenValinnat").fadeIn(500);

      $("#jasentiedot").fadeOut(500);
      $("#jasentiedot").fadeIn(500);

      $("#harrastukset").fadeOut(500);
      $("#harrastukset").fadeIn(500);

   <% end %>
<% end %>
```

G.10 views/kerho/nayta_jasentiedot.js.erb

```
window.location.hash = '<%= @hashid %>';

$("#jasentiedot").html("<%= escape_javascript( render(:partial => "
    jasentiedot")) %>");
$("#harrastukset").html("<%= escape_javascript( render(:partial =>
    "harrastukset")) %>");
```

G.11 views/kerho/paivita_jasen.js.erb

```
<% if @paivita_jasen == true %>
   $("#jasenValinnat").html("<%= escape_javascript( render(:partial
       => "jasenlista")) %>");
   $("#jasentiedot").html("<%= escape_javascript( render(:partial
       => "jasentiedot")) %>");
   $("#harrastukset").html("<%= escape_javascript( render(:partial
       => "harrastukset")) %>");
<% end %>
```

G.12 views/kerho/poista_jasen.js.erb

```
window.location.hash = '<%= @hashid %>';

<!-- Paivitetaan jasenlista ja jasentiedot sivulle -->
$("#jasenValinnat").html("<%= escape_javascript( render(:partial =>
     "jasenlista")) %>");
$("#jasentiedot").html("<%= escape_javascript( render(:partial => "
    jasentiedot")) %>");
$("#harrastukset").html("<%= escape_javascript( render(:partial =>
    "harrastukset")) %>");
$("#jasenmaara").html("Jasenia: <%= escape_javascript( @jasenmaara.
    to_s ) %>");

$("#jasenValinnat").fadeOut(500);
$("#jasenValinnat").fadeIn(500);

$("#jasentiedot").fadeOut(500);
$("#jasentiedot").fadeIn(500);
```

```
$("#harrastukset").fadeOut(500);
$("#harrastukset").fadeIn(500);
```

G.13 views/jasen/jasensivu.html.erb

```
<?xml version="1.0" encoding="UTF-8"?>

<%= javascript_include_tag "kerho" %>

<p id="tallennusilmoitus" class="tallennusilmoitus"></p>
<h1><%= @jasennimi %></h1>

<div id="jasenlomake" class="jasenlomake">
   <%= form_tag(jasen_tallenna_jasen_path, remote: true) do %>
      <%= render_cell :jasentietolomake, :nayta, :jasen =>
         @lomakejasen %>
      <p>
         <%= submit_tag 'Tallenna jäsen' %>
      </p>
   <% end %>
   <p>
      <%= link_to "Takaisin kerhoon", kerho_kerhosivu_path(id: @id)
          %>
   </p>
</div>

<div id="harrastuslomake" class="harrastuslomake">
   <table id="tableHarrastukset" class="tableHarrastukset">
      <%= render partial: "harrastuslomake" %>
   </table>
   <%= link_to_modal "Lisää uusi harrastus", "#uusiharrastusdialog"
       %> |

   <%= link_to 'Poista harrastus', jasen_poista_harrastus_path, :
      remote => true, :data => { confirm: "Poistetaanko valittu
      harrastus" } %>
</div>

<div id="uusiharrastusdialog" title="Uusi harrastus" style="display
   :none">
   <%= render partial: "uusiharrastuslomake" %>
</div>
```

G.14 views/jasen/_uusiharrastuslomake.html.erb

```
<?xml version="1.0" encoding="UTF-8"?>
<h1>Lisää uusi harrastus</h1>

<%= form_tag(jasen_harrastusdialogi_lahetys_path, remote: true) do
   %>
   <table class="tableJasenLomake">
      <% (@apuharrastus.eka_kentta..@apuharrastus.get_kenttia-1).
         each do |i| %>
         <tr>
            <td><b><%= @apuharrastus.get_kysymys(i)%>:</b></td>

            <% if i == @apuharrastus.eka_kentta %>
```

```
            <% kentta = "alakentta" %>
         <% else %>
            <% kentta = "uusiharrastus" + "-" + i.to_s %>
         <% end %>

         <td><%= text_field_tag kentta, @apuharrastus.anna(i), :
            onkeyup => "aseta_uusiharrastuskentta(this.value,
            this.name)" %></td>

         <td id=<%= "uusiharrastusvirhe" + i.to_s %> class="
            virhe"></td>
      </tr>

   <% end %>
</table>
<p id="uusiharrastusilmoitus" class="virheilmoitus"></p>
<!-- Piti laittaa molemmat napit näin koska button_to ei
   toiminut formin kanssa. -->
<!-- Formin avulla saadaan kenttien arvot välitettyä ohjaimelle
   ilman Javascript-koodia. -->
<%= submit_tag 'Peruuta' %>
<%= submit_tag 'Lisää harrastus' %>
<% end %>
```

G.15 views/jasen/aseta_kentta.js.erb

```
$("#virhe<%= escape_javascript(@i) %>").html("<%= escape_javascript
    ( @virhe ) %>");

var onkoVirhetyyli = $("#<%= escape_javascript( @i ) %>").hasClass(
    "virheKentta");

<% if @virhe != nil %>

   if (!onkoVirhetyyli) {
      $("#<%= escape_javascript( @i ) %>").addClass("virheKentta");
   }

<% else %>

   if (onkoVirhetyyli) {
      $("#<%= escape_javascript( @i ) %>").removeClass("virheKentta
          ");
   }

<% end %>
```

G.16 views/jasen/aseta_uusiharrastuskentta.js.erb

```
$("#uusiharrastusvirhe<%= escape_javascript(@i) %>").html("<%=
    escape_javascript( @uusiharrastusvirhe ) %>");

var onkoVirhetyyli = $("#<%= escape_javascript( @kentta ) %>").
    hasClass("virheKentta");

<% if @uusiharrastusvirhe != nil %>
```

```
   if (!onkoVirhetyyli) {
      $("#<%= escape_javascript( @kentta ) %>").addClass("
         virheKentta");
   }

<% else %>

   if (onkoVirhetyyli) {
      $("#<%= escape_javascript( @kentta ) %>").removeClass("
         virheKentta");
   }

<% end %>
```

G.17 views/jasen/aseta_valittuharrastus.js.erb

```
<!-- Tama tiedosto on tyhja. Tama pitaa kuitenkin olla vaikka tassa
     ei mitaan tekisikaan. -->
```

G.18 views/jasen/harrastusdialogi_lahetys.js.erb

```
<% if @toiminto == "sulje" %>
   $.modal.close();
   <!-- Luodaan lomake uudelleen, jotta saadaan uudella aukaisulla
      tyhjat tekstikentat. -->
   $("#uusiharrastusdialog").html("<%= escape_javascript( render(:
      partial => "uusiharrastuslomake")) %>");
<% else %>

   $("#uusiharrastusilmoitus").html("<%= escape_javascript(
      @uusiharrastusilmoitus ) %>");

   <% if @uusiharrastusvirheita == true %>

      $("#uusiharrastusilmoitus").animate({
         backgroundColor: "white",
         color: "#fff"
         }, 500 );

      $("#uusiharrastusilmoitus").animate({
         backgroundColor: "#aa0000",
         color: "#fff"
         }, 1000 );

   <% else %>
      $.modal.close();

      <!-- Luodaan lomake uudelleen, jotta saadaan uudella
         aukaisulla tyhjat tekstikentat. -->
      $("#uusiharrastusdialog").html("<%= escape_javascript( render
         (:partial => "uusiharrastuslomake")) %>");
      <!-- Paivitetaan harrastustaulukko sivulle -->
      $("#tableHarrastukset").html("<%= escape_javascript( render(:
         partial => "harrastuslomake")) %>");

      $("#tableHarrastukset").animate({
         backgroundColor: "yellow"
```

```
         }, 500 );

      $("#tableHarrastukset").animate({
         backgroundColor: "white"
         }, 500 );
   <% end %>
<% end %>
```

G.19 views/jasen/poista_harrastus.js.erb

```
<!-- Paivitetaan harrastustaulukko sivulle -->
$("#tableHarrastukset").html("<%= escape_javascript( render(:
    partial => "harrastuslomake")) %>");

$("#tableHarrastukset").animate({
   backgroundColor: "yellow"
   }, 500 );

$("#tableHarrastukset").animate({
   backgroundColor: "white"
   }, 500 );
```

G.20 views/jasen/tallenna_jasen.js.erb

```
$("#tallennusilmoitus").html("<%= escape_javascript( @ilmoitus ) %>
    ");

<% if @virheita == true %>

   $("#tallennusilmoitus").animate({
      backgroundColor: "white",
      color: "#fff"
      }, 500 );

   $("#tallennusilmoitus").animate({
      backgroundColor: "#aa0000",
      color: "#fff"
      }, 1000 );

<% else %>

   $("#tallennusilmoitus").animate({
      backgroundColor: "white",
      color: "#fff"
      }, 500 );

   $("#tallennusilmoitus").animate({
      backgroundColor: "green",
      color: "#fff"
      }, 1000 );

<% end %>
```

G.21 controllers/kerho_controller.rb

```
# coding: utf-8
require '../rubyKerho/Kerho'
class KerhoController < ApplicationController
   def kerhosivu
      session[:toiminto] = ""
      session[:edellinenhash]  = ""
      @apujasen = Jasen.new
      @lomakejasen = @apujasen
      session[:apujasen] = @apujasen.to_yaml
      session[:virheita] = 0

      @kerho = Kerho.new
      @kerho.lue_tiedostosta("kelmit")

      haejasenet_ja_luojasenlista

      @jasenia = @kerho.get_jasenia
      @kerhonimi = @kerho.get_nimi

      if @valittujasen != ""
         @harrastukset = @kerho.anna_harrastukset(@valittujasen)
      end

      luo_hakulista_alkiot
   end


   def haejasenet_ja_luojasenlista
      if session[:hakuehto] == nil
         jasenet = @kerho.etsi("", @apujasen.eka_kentta)
         session[:hakuehto] = ""
         session[:hakukentta] = 1
         @valittukentta = @apujasen.eka_kentta
      else
         jasenet = @kerho.etsi(session[:hakuehto], session[:
            hakukentta])
         @valittukentta = session[:hakukentta]
      end

      @hakuehto = session[:hakuehto]
      luo_jasenlista_alkiot(jasenet)
      id = params[:id]

      # Pitaa tallentaa sessioon, jotta
      # saadaan id asetettua hashiin
      # aseta_urlhash-metodissa.
      session[:paramid] = id

      if id == nil && jasenet.count > 0
         @jasenid = jasenet[0].get_tunnusnro
         @valittujasen = jasenet[0]
      elsif id != nil
         @valittujasen = @kerho.anna_jasen_id(id)
         @jasenid = @valittujasen.get_tunnusnro
      end
      session[:valittuid] = @jasenid
   end


   def luo_jasenlista_alkiot(jasenet)
      @jasenlista_alkiot = Hash.new

      if jasenet.count == 0
```

```
      @jasenlista_alkiot = {"" => ""}
      @jasenid = ""
      @valittujasen = ""
   else
      jasenet.each {|jasen|
         @jasenlista_alkiot[jasen.get_nimi] = jasen.
            get_tunnusnro
      }
   end
end


def luo_hakulista_alkiot
   @kentat = Hash.new

   (@apujasen.eka_kentta..@apujasen.get_kenttia-1).each do |i|
      @kentat[@apujasen.get_kysymys(i)] = i
   end
end


def nayta_jasentiedot
   session[:toiminto] = "nayta_jasentiedot"
   @id = params[:jasenid]

   @kerho = Kerho.new
   @kerho.lue_tiedostosta("kelmit")
   jasen = @kerho.anna_jasen_id(@id)
   @hashid = valitse_jasen(jasen)

   respond_to do |format|
      format.js
   end
end


def valitse_jasen(jasen)
   @valittujasen = jasen
   @jasenid = @valittujasen.get_tunnusnro
   session[:valittuid] = @jasenid
   @harrastukset = @kerho.anna_harrastukset(@valittujasen)
   return @jasenid
end


def aseta_kentta
   @apujasen = YAML.load(session[:apujasen])
   arvo = params[:teksti]

   @i = params[:i]
   if @i == "nimikentta"
      @i = @apujasen.eka_kentta.to_s
   end

   @virhe = @apujasen.aseta(@i.to_i, arvo)
   session[:apujasen] = @apujasen.to_yaml
   tarkista_virhe

   respond_to do |format|
      format.js
   end
end
```

```
    def tarkista_virhe
       onkovirhe = params[:onkovirhe]

       if @virhe == nil && onkovirhe == "true"
          session[:virheita] = session[:virheita] - 1
       elsif @virhe != nil && onkovirhe == "false"
          session[:virheita] = session[:virheita] + 1
       end
    end


    def jasendialogi_lahetys
       painettu_nappi = params[:commit]

       if painettu_nappi == "Peruuta"
          peruuta_jasenlisays
          @toiminto = "sulje"
       else
          @toiminto = "tallenna"
          lisaa_jasen
          session[:toiminto] = "lisaa_jasen"
       end

       respond_to do |format|
          format.js
       end
    end


    def peruuta_jasenlisays
       @apujasen = Jasen.new
       @lomakejasen = @apujasen
       session[:apujasen] = @apujasen.to_yaml
       session[:virheita] = 0
    end


    def lisaa_jasen
       nimikentta_arvo = params[:nimikentta]

       if session[:virheita] == 0 && nimikentta_arvo != ""
          @kerho = Kerho.new
          @kerho.lue_tiedostosta("kelmit")
          @apujasen = YAML.load(session[:apujasen])
          @apujasen.rekisteroi
          lisaa_jasen_kerhoon
          @hashid =   @apujasen.get_tunnusnro
          @ilmoitus = "Jasen tallennettu!"
          @virheita = false
          @jasenmaara = @kerho.get_jasenia

          loydetyt = @kerho.etsi("", @apujasen.eka_kentta)
          luo_jasenlista_alkiot(loydetyt)

          @apujasen = Jasen.new
          @lomakejasen = @apujasen
          session[:hakuehto] = nil
          session[:toiminto] = "lisaa_jasen"
       elsif nimikentta_arvo == ""
          @ilmoitus = "Nimi ei saa olla tyhja!"
          @virheita = true
       else
          @ilmoitus = "Virheellisia syotteita!"
          @virheita = true
       end
```

```
    end


    def lisaa_jasen_kerhoon
       @apujasen = YAML.load(session[:apujasen])
       @kerho.korvaa_tai_lisaa(@apujasen)
       @kerho.tallenna
       valitse_jasen(@apujasen)
    end


    def hae_jasenet
       sleep 3
       session[:toiminto] = "hae_jasenet"
       session[:hakuehto] = params[:hakuehto]
       session[:hakukentta] = params[:hakukentta]

       @kerho = Kerho.new
       @kerho.lue_tiedostosta("kelmit")
       loydetyt = @kerho.etsi(session[:hakuehto], session[:
          hakukentta])

       if loydetyt.count > 0
          @hashid = valitse_jasen(loydetyt[0])
       else
          @hashid = "n"
       end

       luo_jasenlista_alkiot(loydetyt)

       respond_to do |format|
          format.js
       end
    end


    def poista_jasen
       session[:toiminto] = "poista_jasen"

       @kerho = Kerho.new
       @kerho.lue_tiedostosta("kelmit")
       @kerho.poista(session[:valittuid])
       @kerho.tallenna
       loydetyt = @kerho.etsi("", 1)

       if loydetyt.count > 0
          @hashid = loydetyt[0].get_tunnusnro
       else
          @hashid = "n"
       end

       luo_jasenlista_alkiot(loydetyt)
       valitse_jasen(loydetyt[0])

       @jasenmaara = @kerho.get_jasenia
       session[:hakuehto] = nil

       respond_to do |format|
          format.js
       end
    end


    def paivita_jasen
       if session[:toiminto] == ""
```

```
         @paivita_jasen = true
         toiminto_paivita_jasen
      else
         @paivita_jasen = false
      end

      session[:toiminto] = ""

      respond_to do |format|
         format.js
      end
   end


   def toiminto_paivita_jasen
      id = params[:hashid]
      @kerho = Kerho.new
      @kerho.lue_tiedostosta("kelmit")
      loydetyt = @kerho.etsi(session[:hakuehto], session[:
         hakukentta])

      @valittujasen = @kerho.anna_jasen_id(id)
      valitse_jasen(@valittujasen)

      luo_jasenlista_alkiot(loydetyt)
   end


   def aseta_urlhash
      session[:toiminto] = "aseta_urlhash"

      if session[:paramid] != nil
         @hashid = session[:paramid]
      else
         kerho = Kerho.new
         kerho.lue_tiedostosta("kelmit")
         jasenet = kerho.etsi("", "")
         @hashid = jasenet[0].get_tunnusnro
      end

      respond_to do |format|
         format.js
      end
   end
end
```

G.22 controllers/jasen_controller.rb

```
#encoding: utf-8
class JasenController < ApplicationController
   def jasensivu
      kerho = Kerho.new
      kerho.lue_tiedostosta("kelmit")

      @id = params[:id]
      @jasen = kerho.anna_jasen_id(@id)
      @jasennimi = @jasen.get_nimi
      @lomakejasen = @jasen
      @apuharrastus = Harrastus.new(@id)
      @harrastukset = kerho.anna_harrastukset(@jasen)
```

```
      session[:apuharrastus] = @apuharrastus.to_yaml
      session[:harrastukset] = @harrastukset.to_yaml
      session[:jasen] = @jasen.to_yaml
      session[:virheita] = 0
      session[:uusiharrastusvirheet] = 0
   end


   def aseta_kentta
      if params[:tyyppi] == "jasen"
         aseta_jasenkentta
      else
         aseta_harrastuskentta
      end
      respond_to do |format|
         format.js
      end
   end


   def aseta_jasenkentta
      jasen = YAML.load(session[:jasen])
      arvo = params[:teksti]
      @i = params[:i]

      if @i == "nimikentta"
         @i = jasen.eka_kentta.to_s
      end

      @virhe = jasen.aseta(@i.to_i, arvo)
      session[:jasen] = jasen.to_yaml

      if @i != jasen.eka_kentta.to_s
         tarkista_virhe
      end
   end


   def aseta_harrastuskentta
      harrastukset = YAML.load(session[:harrastukset])

      @i = params[:i]
      osat = @i.split('-')
      id = osat[0]
      i = osat[1]

      harrastukset.each do |harrastus|
         if harrastus.get_tunnusnro.to_s == id
            @virhe = harrastus.aseta(i.to_i, params[:teksti])
            tarkista_virhe
         break
         end
      end
      session[:harrastukset] = harrastukset.to_yaml
   end


   def tarkista_virhe
      onkovirhe = params[:onkovirhe]

      if @virhe == nil && onkovirhe == "true"
         session[:virheita] = session[:virheita] - 1
      elsif @virhe != nil && onkovirhe == "false"
         session[:virheita] = session[:virheita] + 1
      end
```

```
    end


    def tallenna_jasen
       nimikentta_arvo = params[:nimikentta]

       if session[:virheita] == 0 && nimikentta_arvo != ""
          @jasen = YAML.load(session[:jasen])
          @kerho = Kerho.new
          @kerho.lue_tiedostosta("kelmit")
          @kerho.korvaa_tai_lisaa(@jasen)

          korvaa_kerhon_harrastukset

          @kerho.tallenna
          @ilmoitus = "Jasen tallennettu!"
          @virheita = false
       elsif nimikentta_arvo == ""
          @ilmoitus = "Nimi ei saa olla tyhja!"
          @virheita = true
       else
          @ilmoitus = "Virheellisia syotteita!"
          @virheita = true
       end
       respond_to do |format|
          format.js
       end
    end


    def korvaa_kerhon_harrastukset
       harrastukset = @kerho.anna_harrastukset(@jasen)
       uudet_harrastukset = YAML.load(session[:harrastukset])

       harrastukset.each do |harrastus|
          @kerho.poista_harrastus(harrastus)
       end
       uudet_harrastukset.each do |uusi_harrastus|
          @kerho.lisaa(uusi_harrastus)
       end
    end


    def harrastusdialogi_lahetys
       painettu_nappi = params[:commit]
       if painettu_nappi == "Peruuta"
          peruuta_harrastuslisays
          @toiminto = "sulje"
       else
          @toiminto = "tallenna"
          lisaa_harrastus
       end
       respond_to do |format|
          format.js
       end
    end


    def peruuta_harrastuslisays
       @apuharrastus = Harrastus.new(YAML.load(session[:jasen]).
          get_tunnusnro)
       session[:apuharrastus] = @apuharrastus.to_yaml
       session[:uusiharrastusvirheet] = 0
    end
```

```
def lisaa_harrastus
   alakentta_arvo = params[:alakentta]

   if session[:uusiharrastusvirheet] == 0 && alakentta_arvo != "
      "
      lisaa_harrastus_listaan
      @uusiharrastusilmoitus = ""
      @uusiharrastusvirheita = false
      session[:uusiharrastusvirheet] = 0
   elsif alakentta_arvo == ""
      @uusiharrastusilmoitus = "Ala ei saa olla tyhja!"
      @uusiharrastusvirheita = true
   else
      @uusiharrastusilmoitus = "Virheellisia syotteita!"
      @uusiharrastusvirheita = true
   end
end


def lisaa_harrastus_listaan
   @apuharrastus = YAML.load(session[:apuharrastus])
   @apuharrastus.rekisteroi

   @harrastukset = YAML.load(session[:harrastukset])
   @harrastukset.push(@apuharrastus)

   session[:harrastukset] = @harrastukset.to_yaml
   @apuharrastus = Harrastus.new(YAML.load(session[:jasen]).
      get_tunnusnro)
   session[:apuharrastus] = @apuharrastus.to_yaml
end


def aseta_uusiharrastuskentta
   @apuharrastus = YAML.load(session[:apuharrastus])
   @kentta = params[:kentta]

   if @kentta == "alakentta"
      @i = @apuharrastus.eka_kentta.to_s
   else
      osat = @kentta.split('-')
      @i = osat[1]
   end

   @uusiharrastusvirhe = @apuharrastus.aseta(@i.to_i, params[:
      arvo])
   session[:apuharrastus] = @apuharrastus.to_yaml
   tarkista_uusiharrastusvirhe
end


def tarkista_uusiharrastusvirhe
   onkovirhe = params[:onkovirhe]

   if @uusiharrastusvirhe == nil && onkovirhe == "true"
      session[:uusiharrastusvirheet] = session[:
         uusiharrastusvirheet] - 1
   elsif @uusiharrastusvirhe != nil && onkovirhe == "false"
      session[:uusiharrastusvirheet] = session[:
         uusiharrastusvirheet] + 1
   end
end
```

```
   def poista_harrastus
      @harrastukset = YAML.load(session[:harrastukset])

      @harrastukset.each do |harrastus|
         if harrastus.get_tunnusnro.to_s == session[:
            valittu_harrastusid]
         @harrastukset.delete(harrastus)
         break
         end
      end
      session[:harrastukset] = @harrastukset.to_yaml

      respond_to do |format|
         format.js
      end
   end


   def aseta_valittuharrastus
      valittukentta = params[:valittukentta]
      osat = valittukentta.split('-')
      valittuid = osat[0]
      session[:valittu_harrastusid] = valittuid

      respond_to do |format|
         format.js
      end
   end
end
```

G.23 cells/jasentietolomake_cell.rb

```
class JasentietolomakeCell < Cell::Rails
   def nayta(parametrit)
      @apujasen = parametrit[:jasen]
      render
   end
end
```

G.24 cells/jasentietolomake/nayta.html.erb

```
<table class="tableJasenLomake">
   <% (@apujasen.eka_kentta..@apujasen.get_kenttia-1).each do |i|
      %>
      <tr>
         <td><b><%= @apujasen.get_kysymys(i)%>:</b></td>

         <td><%= text_field_tag i.to_s, @apujasen.anna(i), :class
            => "jasenkentta", :onkeyup => "aseta_arvo('jasen',this.
            value, this.name)" %></td>

         <td id=<%= "virhe" + i.to_s %> class="virhe"></td>
      </tr>
   <% end %>
</table>
```

G.25 assets/javascripts/kerho.js

```
window.onhashchange = urlHashMuuttunut;

$( window ).load(function() {
   if (window.location.hash == "")
   {
      jQuery.ajax({
         url: "aseta_urlhash",
         type: "GET",
         data: {},
      });
   }
   else urlHashMuuttunut();
});

function urlHashMuuttunut() {
   var hashid = window.location.hash.substring(1);

   jQuery.ajax({
      url: "paivita_jasen",
      type: "GET",
      data: {"hashid": hashid},
   });
};

function paivita_jasentiedot(jasenid) {
   jQuery.ajax({
      url: "nayta_jasentiedot",
      type: "GET",
      data: {"jasenid": jasenid},
   });
}

function aseta_arvo(tyyppi,teksti, i) {
   var onkoVirhetyyli = $("#" + i).hasClass("virheKentta");
   jQuery.ajax({
      url: "aseta_kentta",
      type: "GET",
      data: {"tyyppi": tyyppi, "teksti": teksti, "i": i, "onkovirhe
         ": onkoVirhetyyli},
   });
}

function aseta_uusiharrastuskentta(arvo, kentta) {
   var onkoVirhetyyli = $("#" + kentta).hasClass("virheKentta");

   jQuery.ajax({
      url: "aseta_uusiharrastuskentta",
      type: "GET",
      data: {"arvo": arvo, "kentta": kentta, "onkovirhe":
         onkoVirhetyyli},
   });
}

function aseta_harrastus(valittuKentta) {
   jQuery.ajax({
      url: "aseta_valittuharrastus",
      type: "GET",
      data: {"valittukentta": valittuKentta},
   });
}

function nayta_odotusIlmaisin() {
   $('#odotusIlmaisin').show();
```

```
}
```

G.26 assets/stylesheets/kerho.css.scss

```
.jasenlista {
   float: left;
}
.jasenlista p {
   margin-top: 20px;
}
.jasentiedot {
   float: left;
   margin-left: 30px;
}
.jasentiedot p {
   margin-top: 20px;
}
.harrastukset {
   float: left;
   margin-left: 70px;
}
.selectJasen {
   min-width: 130px;
}
.tableHarrastukset {
   margin-top: 20px;
   text-align: center;
   border: 1px solid black;
   border-collapse: collapse;
}
.tableHarrastukset tr td {
   border: 1px solid black;
   padding: 5px;
}
.tableEka {
   text-align: left;
}
.tableJasenLomake {
   border: none;
   border-collapse: collapse;
}
.tableJasenLomake tr {
   border: none;
}
.tableJasenLomake tr td {
   border: none;
}
.virhe {
   color: red;
}
.virheKentta {
   border-color: red;
}
.virheilmoitus {
   text-align: center;
}
.jasenkentta {
   width: 150px;
}
```

G.27 assets/stylesheets/jasen.css.scss

```
.tableHarrastukset {
   margin-top: 20px;
   text-align: center;
   border: 1px solid black;
   border-collapse: collapse;
}
.tableHarrastukset tr td {
   border: 1px solid black;
   padding: 5px;
}
.jasenlomake {
   float: left;
   margin-left: 20px;
}
.harrastuslomake {
   float: left;
   margin-left: 100px;
}
.virhe {
   color: red;
}
.virheKentta {
   border-color: red;
}
.virheilmoitus {
   text-align: center;
}
.tallennusilmoitus {
   padding: 5px;
   text-align: center;
   width: 300px;
}
.harrastusekakentta {
   width: 130px;
}
.harrastuskentta {
   width: 50px;
}
.jasenkentta {
   width: 150px;
}
```

G.28 config/routes.rb

```
Rails.application.routes.draw do
   get 'jasen/jasensivu'
   get 'kerho/kerhosivu'

   get "kerho/nayta_jasentiedot" => "kerho#nayta_jasentiedot"
   get "kerho/aseta_kentta" => "kerho#aseta_kentta"
   get "kerho/poista_jasen" => "kerho#poista_jasen"
   get "kerho/paivita_jasen" => "kerho#paivita_jasen"
   get "kerho/aseta_urlhash" => "kerho#aseta_urlhash"
   post "kerho/hae_jasenet" => "kerho#hae_jasenet"
   post "kerho/jasendialogi_lahetys" => "kerho#jasendialogi_lahetys
       "

   get "jasen/aseta_kentta" => "jasen#aseta_kentta"
   get "jasen/poista_harrastus" => "jasen#poista_harrastus"
```

```
   get "jasen/aseta_uusiharrastuskentta" => "jasen#
      aseta_uusiharrastuskentta"
   get "jasen/aseta_valittuharrastus" => "jasen#
      aseta_valittuharrastus"
   post "jasen/tallenna_jasen" => "jasen#tallenna_jasen"
   post "jasen/harrastusdialogi_lahetys" => "jasen#
      harrastusdialogi_lahetys"
end
```

H Seaside-kerhon lähdekoodit

H.1 JasenTietoLomake

```
WAComponent subclass: #JasenTietoLomake
   instanceVariableNames: 'jasen virheita nimikenttaArvo'
   classVariableNames: ''
   poolDictionaries: ''
   category: 'WWWKerho'


initialize
   super initialize.

   jasen := Jasen new.
   virheita := 0.
   nimikenttaArvo := ''.


renderContentOn: html
   html div
      id: #lomakeSisalto;
      with:  [ self luoJasenLomake: html ].


luoJasenLomake: html
   html table
      class: 'tableJasenKentat';
      with: [
         jasen ekaKentta to: jasen getKenttia -1 do: [ :k |
            html tableRow: [ self luoJasenKenttaRivi: html and: k
               ]]].


luoJasenKenttaRivi: html and: k
| kysymys |
   kysymys := jasen getKysymys: k.

   html tableData: [ html text: kysymys, ': ' ].

   html tableData: [ self luoJasenKentta: html and: kysymys and: k
      ].

   html tableData: [
      html label
         id: 'virhe',kysymys;
         class: 'virheLabel';
         with: ''. ].
```

```
luoJasenKentta: html and: kysymys and: k
| virhe arvo onkoVirheTyyli |
   html textInput
      id: 'text',kysymys;
      class: 'jasenkentta';
      value: (jasen anna: k);
      onKeyUp: (html jQuery ajax
         callback: [ :value | onkoVirheTyyli := value ] value: ((
            html jQuery this) hasClass: 'virheKentta');
         callback: [ :value | arvo := value ] value: (html jQuery
            this) value;

         script: [ :s |
            k = jasen ekaKentta
               ifTrue: [ nimikenttaArvo := arvo. ].

            virhe := jasen aseta: k and: arvo.

            virhe = '' & onkoVirheTyyli = 'true'
               ifTrue: [
                  virheita := virheita - 1.
                  s << (s jQuery id: 'text',kysymys) removeClass: '
                     virheKentta' ].

            virhe ~= '' & onkoVirheTyyli = 'false'
               ifTrue: [
                  virheita := virheita + 1.
                  s << (s jQuery id: 'text',kysymys) addClass: '
                     virheKentta' ].

            s << (s jQuery id: 'virhe',kysymys) html: [ :h | h
               render: virhe ]. ]).


asetaJasen: asetettavaJasen
   jasen := asetettavaJasen.
   nimikenttaArvo := jasen getNimi.
   virheita := 0.


getJasen
   ^ jasen.


onkoVirheita
   ^ (virheita > 0).


nimiTyhja
   ^ nimikenttaArvo = ''.


canBeRoot (Class-side method)
   ^false
```

H.2 Kerhosivu

```
WAComponent subclass: #Kerhosivu
   instanceVariableNames: 'kerho tallennettavaJasen jasenLista
      tietoLomake'
   classVariableNames: ''
```

```
   poolDictionaries: ''
   category: 'WWWKerho'


canBeRoot (class-side method)
   ^true


initialize
   super initialize.
   kerho := Kerho new.
   kerho lueTiedostosta: 'kelmit'.
   tallennettavaJasen := Jasen new.
   tietoLomake := JasenTietoLomake new.


children
   ^ Array with: tietoLomake


updateRoot: anHtmlRoot
   super updateRoot: anHtmlRoot.
   anHtmlRoot title: kerho getNimi.

   anHtmlRoot link
      type: 'text/css';
      beStylesheet;
      addAll;
      url: KerhoFileLibrary  / 'kerho.css';
      yourself.


renderContentOn: html
| kentat apuJasen jasenId hakuehto hakukentta |
   apuJasen := Jasen new.

   self session onkoTallennettu
      ifTrue: [
         hakuehto := self session getHakuehto.
         hakukentta := self session getHakukentta ];
      ifFalse: [
         hakuehto := ''.
         hakukentta := apuJasen ekaKentta ].

   jasenLista := kerho etsi: hakuehto and: hakukentta asInteger.
   jasenId := self requestContext request at: 'id'.

   jasenId = nil
      ifTrue: [
         jasenLista size > 0
            ifTrue: [
               apuJasen := jasenLista at: 1 ]];
      ifFalse: [ apuJasen := kerho annaJasenId: jasenId ].

   html heading with: kerho getNimi.
   html paragraph
      id: #labelJasenet;
      class: 'labelJasenet';
      with: 'Jäseniä: ',kerho getJasenia asString.

   html div
      class: 'hakuLomake';
      with: [ self luoHakuLomake: html and: hakuehto and:
          hakukentta ].
```

```
   html paragraph
      class: 'piilotettu';
      id: 'odotusilmaisin';
      with: [
         html text: 'Haetaan jäseniä...'.
         html image url: KerhoFileLibrary / #ajaxloaderGif. ].

   html label
      class: 'labelJasenet';
      with: 'Jäsenet'.
   html break.

   html div
      id: #jasenValintalista;
      class: 'jasenValintalista';
      with: [ self luoJasenValinnat: html and: apuJasen
         getTunnusNro ].

   html div
      id: #jasenTiedot;
      class: 'jasenTiedot';
      with: [ self naytaJasenTiedot: html and: apuJasen
         getTunnusNro. ].

   html div
      id: #jasenHarrastukset;
      class: 'jasenHarrastukset';
      with: [ self naytaJaseneneHarrastukset: html and: apuJasen
         getTunnusNro. ].


luoHakuLomake: html and: hakuehto and: hakukentta
   html label with: 'Hae jäseniä: '.
   html textInput
      id: #textHae;
      with: hakuehto.
   html select
      id: #selectHae;
      with: [ self hakuValinnat: html and: hakukentta ].
   self hakuNappi: html.


hakuValinnat: html and: hakukentta
| apuJasen |
   apuJasen := Jasen new.

   apuJasen ekaKentta to: apuJasen getKenttia - 1 do: [ :k |
      k asString = hakukentta
         ifTrue: [
            html option
               selected: true;
               value: k;
               with: (apuJasen getKysymys: k) ]
         ifFalse: [
            html option
               value: k;
               with: (apuJasen getKysymys: k) ]]


hakuNappi: html
| hakuEhto hakuKentta viive |
   html submitButton
      onClick: (html jQuery: #odotusilmaisin) show;
      onClick: (html jQuery ajax
```

```
         callback: [ :value | hakuEhto := value ] value: (html
            jQuery id: #textHae) value;
         callback: [ :value | hakuKentta := value ] value: (html
            jQuery id: #selectHae) value;

         script: [ :s |
            viive := Delay forSeconds: 3.
            viive wait.

            self session tallennaHaku: hakuEhto.
            jasenLista := kerho etsi: hakuEhto and: hakuKentta
               asInteger.
            jasenLista size > 0
               ifTrue: [ self paivitaJasenLista: s and: 1. ]
               ifFalse: [ self paivitaJasenLista: s and: 0. ].

            s << (s jQuery id: #odotusilmaisin) hide ]);
      with: 'Hae'.


paivitaJasenLista: s and: valittavaJasen
| valittuId |
   valittavaJasen = 1
      ifTrue: [ valittuId := (jasenLista at: 1) getTunnusNro. ];
      ifFalse: [
         valittavaJasen = 0
            ifTrue: [ valittuId := '' ] ;
            ifFalse: [ valittuId := (jasenLista at: (jasenLista
               size)) getTunnusNro. ]].

   s << (s jQuery: #jasenValintalista) html: [ :r | r render: [
      self luoJasenValinnat: r and: valittuId ]].
   s << (s jQuery: #jasenTiedot) html: [ :h | h render: [ self
      naytaJasenTiedot: h and: valittuId ]].
   s << (s jQuery: #jasenHarrastukset) html: [ :h | h render: [
      self naytaJaseneneHarrastukset: h and: valittuId ]].
   s << (s jQuery: #labelJasenet) html: 'Jäseniä: ',kerho
      getJasenia asString.

   s << (s jQuery: #selectJasen) fadeOut: 400 milliseconds.
   s << (s jQuery: #jasenTiedot) fadeOut: 400 milliseconds.
   s << (s jQuery: #jasenHarrastukset) fadeOut: 400 milliseconds.
   s << (s jQuery: #selectJasen) fadeIn: 400 milliseconds.
   s << (s jQuery: #jasenTiedot) fadeIn: 400 milliseconds.
   s << (s jQuery: #jasenHarrastukset) fadeIn: 400 milliseconds.


luoJasenValinnat: html and: valittuId
   html div
      with: [
         self jasenValintaLista: html and: valittuId.
         html break.
         self luoUusiJasenDialogi: html.
         html break.
         self poistaNappi: html. ].


jasenValintaLista: html and: valittavaJasen
| valittuId |
   html select
      id: #selectJasen;
      class: 'selectJasen';
      size: 20;
      onChange: (html jQuery ajax
```

```
         callback: [ :value | valittuId := value] value: (html
            jQuery this) value;

         script: [ :s |
            s << (s jQuery: #jasenTiedot) html: [ :r | r render: [
               self naytaJasenTiedot: r and: valittuId ]].
            s << (s jQuery: #jasenHarrastukset) html: [ :h | h
               render: [ self naytaJaseneneHarrastukset: h and:
               valittuId ]]]);

      with: [ self jasenValinnat: html and: valittavaJasen ].


jasenValinnat: html and: valittuId
   jasenLista size > 0
      ifFalse: [
         html option
            value: '';
            with: '' ];
      ifTrue: [
         jasenLista do: [ :jasen |
            jasen getTunnusNro = valittuId
               ifTrue: [
                  html option
                     selected: true;
                     value: jasen getTunnusNro;
                     with: jasen getNimi ];
               ifFalse: [
                  html option
                     value: jasen getTunnusNro;
                     with: jasen getNimi ]]]


naytaJasenTiedot: html and: id
| jasen |
   id = ''
      ifFalse: [
         jasen := kerho annaJasenId: id.

         html div
            with: [
               html label
                  class: 'jasenOtsikko';
                  with: 'Jäsen: ',jasen getNimi.
               html label with: ' ['.
               html anchor
                  url: '/jasen?id=',id;
                     with: 'muokkaa'.
               html label with: ']'.
               html break.
               html break.
               html table
                  class: 'tableJasenTiedot';
                  with: [ self luoJasenTietoRivit: html and: jasen.
                     ]]];
      ifTrue: [
         html paragraph
            class: 'boldText';
            with: 'Ei löytynyt yhtään jäsentä!'. ].


luoJasenTietoRivit: html and: jasen
| kysymys arvo |
   jasen ekaKentta to: jasen getKenttia - 1 do: [ :k |
```

```
      kysymys := jasen getKysymys: k.
      arvo := jasen anna: k.

      html tableRow: [
         html tableData
            class: 'boldText';
            with: kysymys,': '.
         html tableData with: arvo. ]].


naytaJaseneneHarrastukset: html and: id
| jasen harrastukset apuHarrastus |
   id = ''
      ifFalse: [
         jasen := kerho annaJasenId: id.
         harrastukset :=kerho annaHarrastukset: jasen.

         apuHarrastus := Harrastus new.
         html div: [
            html table
               class: 'tableHarrastukset';
               with: [
                  html tableCaption: 'Harrastukset'.
                  self luoHarrastusOtsikkoRivi: html and:
                     apuHarrastus.

                  harrastukset size > 0
                     ifTrue: [
                        harrastukset do: [ :harrastus  |
                           self luoHarrastusTietoRivi: html and:
                              harrastus. ]];
                     ifFalse: [
                        html tableRow: [ html tableData colSpan: 3;
                           with: 'Ei harrastuksia!' ]]]]


luoHarrastusOtsikkoRivi: html and: apuHarrastus
   html tableRow
      class: 'boldText';
      with: [
         apuHarrastus ekaKentta to: apuHarrastus getKenttia - 1 do:
             [ :k |
            html tableData: [
               html label with: (apuHarrastus getKysymys: k). ]]].


luoHarrastusTietoRivi: html and: harrastus
| tdClass |
   html tableRow: [
      harrastus ekaKentta to: harrastus getKenttia - 1 do: [ :k |

         k = harrastus ekaKentta
            ifTrue: [ tdClass := 'tableEka' ];
            ifFalse: [ tdClass := '' ].

         html tableData
            class: tdClass;
            with: [
               html label with: (harrastus anna: k). ]]].


luoUusiJasenDialogi: html
   html div
      id: #uusiJasenDialog;
```

```
      script:
         (html jQuery new dialog
            title: 'Lisää uusi jäsen';
            autoOpen: false;
            width: 500;
            height: 600;
            modal: true;
            addButton: 'Peruuta' do: (html jQuery this dialog close
               );
            addButton: 'Lisää uusi jäsen'
               do: (html jQuery ajax
                  script: [ :s |
                     tietoLomake nimiTyhja
                        ifTrue: [
                           s << (s jQuery: #virheIlmoitus) html: [
                              :r | r render: 'Nimi ei saa olla
                              tyhjä!' ].
                           s << (s jQuery: #virheIlmoitus) fadeIn:
                              1000 milliseconds.
                           s << (s jQuery: #virheIlmoitus) fadeOut:
                               1500 milliseconds ];
                        ifFalse: [
                           tietoLomake onkoVirheita
                              ifTrue: [
                                 s << (s jQuery: #virheIlmoitus)
                                    html: [ :r | r render: '
                                    Virheellisiä syötteitä!' ].
                                 s << (s jQuery: #virheIlmoitus)
                                    fadeIn: 1000 milliseconds.
                                 s << (s jQuery: #virheIlmoitus)
                                    fadeOut: 1500 milliseconds ];
                              ifFalse: [
                                 tallennettavaJasen := tietoLomake
                                    getJasen.
                                 self tallennaUusiJasen.
                                 jasenLista := kerho etsi: '' and:
                                    ''.
                                 self paivitaJasenLista: s and: -1.
                              s << (s jQuery: #labelJasenet) html:
                                  [ :r | r render: 'Jäseniä: ',kerho
                                   getJasenia asString ].
                              s << (s jQuery: #uusiJasenDialog)
                                 dialog close ]]]));
      with: [
         html render: tietoLomake.
         html paragraph
            id: #virheIlmoitus;
            class: 'virheIlmoitus';
            with: 'Virheellisiä syötteitä!' ].

   self dialoginAukaisuNappi: html.


dialoginAukaisuNappi: html
   html submitButton
      onClick: (html jQuery ajax
         script: [ :s |
            tallennettavaJasen := Jasen new.
            tietoLomake asetaJasen: tallennettavaJasen.
            "Tässä pitää päivittää tyhjä jäsen lomakkeeseen"
            s << (s jQuery: #lomakeSisalto) html: [ :r | r render:
               [ tietoLomake luoJasenLomake: r ]]]);

      onClick: (html jQuery: #uusiJasenDialog) dialog open;
```

```
      with: 'Lisää uusi jäsen'


tallennaUusiJasen
   tallennettavaJasen rekisteroi.
   kerho korvaaTaiLisaa: tallennettavaJasen.
   kerho tallenna.


poistaNappi: html
| valittuId |
   html submitButton
      onClick: (html jQuery ajax
         confirm: 'Poistetaanko valittu jasen?';
         callback: [ :value | valittuId := value ] value: (html
             jQuery: #selectJasen) value;

         script: [ :s |
            jasenLista remove: (kerho annaJasenId: valittuId).
            kerho poista: valittuId.
            kerho tallenna.
            s << (s jQuery: #labelJasenet) html: [ :r | r render: '
               Jäseniä: ',kerho getJasenia asString ].
            self paivitaJasenLista: s and: 1 ]);
      with: 'Poista jäsen'.
```

H.3 Jasensivu

```
WAComponent subclass: #Jasensivu
   instanceVariableNames: 'jasenId kerho jasen tallennettavaJasen
      virheita harrastukset tallennettavaHarrastus tietoLomake
      harrastuslomakeVirheita uusiharrastusVirheita'
   classVariableNames: ''
   poolDictionaries: ''
   category: 'WWWKerho'


canBeRoot (class-side method)
   ^true


initialize
   super initialize.
   kerho := Kerho new.
   kerho lueTiedostosta: 'kelmit'.
   jasenId  := ''.
   virheita := 0.
   harrastuslomakeVirheita := 0.
   tietoLomake := JasenTietoLomake new.


children
   ^ Array with: tietoLomake


updateRoot: anHtmlRoot
   super updateRoot: anHtmlRoot.
   anHtmlRoot title: kerho getNimi,': Jäsenen muokkaus'.

   anHtmlRoot link
      type: 'text/css';
```

```
      beStylesheet;
      addAll;
      url: KerhoFileLibrary  / 'jasen.css';
      yourself.


renderContentOn: html
   jasenId := self requestContext request at: 'id' .
   jasen := kerho annaJasenId: jasenId.
   tietoLomake asetaJasen: jasen.

   html heading with: jasen getNimi.
   html div
      id: #ilmoitus;
      class: 'ilmoitus'.
   html div
      id: #jasenLomake;
      class: 'jasenTietoLomake';
      with: [
         html render: tietoLomake.
         html break.
         html anchor
            url: '/kerho?id=',jasenId;
            with: 'Takaisin kerhoon'.
         html space.
         self tallennusNappi: html ].
   html div
      id: #harrastusLomake;
      with: [ self naytaHarrastusLomake: html. ].


tallennusNappi: html
   html submitButton
      onClick: (html jQuery ajax
         script: [:s  |
            tietoLomake nimiTyhja
               ifTrue: [
                  s << (s jQuery: #ilmoitus) html: [ :r | r render:
                       [ self naytaIlmoitus: r and: 'Nimi ei saa
                      olla tyhjä!' ]].
                  s << (s jQuery: #ilmoitus) hide: 400 milliseconds
                     .
                  s << (s jQuery: #ilmoitus) show: 400 milliseconds
                      . ];
               ifFalse: [
                  harrastuslomakeVirheita > 0 | tietoLomake
                     onkoVirheita
                     ifFalse: [
                        kerho korvaaTaiLisaa: jasen.
                        kerho tallenna.
                        s << (s jQuery: #ilmoitus) html: [ :r | r
                           render: [ self naytaIlmoitus: r and: '
                           Jäsen tallennettu!' ]].
                        s << (s jQuery: #ilmoitus) hide: 400
                           milliseconds.
                        s << (s jQuery: #ilmoitus) show: 400
                           milliseconds. ];
                     ifTrue: [
                        s << (s jQuery: #ilmoitus) html: [ :r | r
                           render: [ self naytaIlmoitus: r and: '
                           Virheellisiä syötteitä' ]].
                        s << (s jQuery: #ilmoitus) hide: 400
                           milliseconds.
```

```
                        s << (s jQuery: #ilmoitus) show: 400
                           milliseconds. ]]]);
      with: 'Tallenna jäsen'.


naytaIlmoitus: html and: teksti
   harrastuslomakeVirheita > 0 | tietoLomake onkoVirheita |
      tietoLomake nimiTyhja
      ifFalse: [
         html div
            class: 'ui-corner-all ui-state-highlight';
            with: [
               html span class: 'ui-icon ui-icon-info'.
               html text: teksti. ]];
      ifTrue: [
          html div
            class: 'ui-corner-all ui-state-error';
            with: [
               html span class: 'ui-icon ui-icon-alert'.
               html text: teksti. ]].


naytaHarrastusLomake: html
| apuHarrastus |
   harrastukset := kerho annaHarrastukset: jasen.

   harrastukset size > 0
      ifTrue: [
         harrastuslomakeVirheita := 0.
         apuHarrastus := harrastukset at: 1.
         html div
            id: #harrastusLomake;
            class: 'harrastusLomake';
            with: [
               html table
                  class: 'tableHarrastukset';
                  with: [
                     html tableCaption: 'Harrastukset'.
                     self luoHarrastusOtsikkoRivi: html and:
                        apuHarrastus.

                     harrastukset do: [ :harrastus  |
                         self luoHarrastusKenttaRivit: html and:
                            harrastus. ]].
            html break.
            self luoUusiHarrastusDialogi: html.
            self poistaHarrastusNappi: html. ]];
      ifFalse: [
         html div
            class: 'harrastusLomake';
            with: [
               html paragraph
                  class: 'boldText';
                  with: 'Harrastukset'.
               html paragraph with: 'Ei harrastuksia!'.
               self luoUusiHarrastusDialogi: html. ]].


luoHarrastusOtsikkoRivi: html and: harrastus
   html tableRow
      class: 'boldText';
      with: [
         harrastus ekaKentta to: harrastus getKenttia - 1 do: [ :k
            |
```

```
            html tableData: [
               html label with: (harrastus getKysymys: k) ]]]


luoHarrastusKenttaRivit: html and: harrastus
| id |
   id := harrastus getId.

   html tableRow: [
      harrastus ekaKentta to: harrastus getKenttia - 1 do: [ :k |
         html tableData: [
            self luoHarrastusKentta: html and: harrastus and: k and
               : id.

            html label
               id: 'virhe',id,(harrastus getKysymys: k);
               class: 'virheLabel';
               with: ''. ]]]


luoHarrastusKentta: html and: harrastus and: k and: id
| arvo virhe tdClass onkoVirheTyyli |
   k = harrastus ekaKentta
      ifTrue: [ tdClass := 'harrastusekakentta' ];
      ifFalse: [ tdClass := 'harrastuskentta' ].

   html textInput
      id: id,(harrastus getKysymys: k);
      class: tdClass;
      value: (harrastus anna: k);
      onKeyUp: (html jQuery ajax
         callback: [ :value | onkoVirheTyyli := value ] value: ((
            html jQuery this) hasClass: 'virheKentta');
         callback: [ :value | arvo := value ] value: (html jQuery
            this) value;

         script: [ :s |
            virhe := harrastus aseta: k and: arvo.

            virhe = '' & onkoVirheTyyli = 'true'
               ifTrue: [
                  harrastuslomakeVirheita :=
                     harrastuslomakeVirheita - 1.
                  s << (s jQuery id: id,(harrastus getKysymys: k))
                     removeClass: 'virheKentta' ].

            virhe ~= '' & onkoVirheTyyli = 'false'
               ifTrue: [
                  harrastuslomakeVirheita :=
                     harrastuslomakeVirheita + 1.
                  s << (s jQuery id: id,(harrastus getKysymys: k))
                     addClass: 'virheKentta' ].

            s << (s jQuery id: 'virhe',id,(harrastus getKysymys: k)
               ) html: [ :h | h render: virhe ]. ]);

      onFocus: (html jQuery ajax
         script: [:s | tallennettavaHarrastus := harrastus. ]).


luoUusiHarrastusDialogi: html
| ala |
   html div
      id: #uusiHarrastusDialog;
```

```
      script: (html jQuery new dialog
            title: 'Lisää uusi harrastus';
         autoOpen: false;
         width: 500;
         height: 300;
         modal: true;
         addButton: 'Peruuta' do: (html jQuery this dialog close);
         addButton: 'Luo harrastus'
            do: (html jQuery ajax
               callback: [ :value | ala := value ] value: (html
                  jQuery id: 'textala') value;
               script: [ :s |
                  ala = ''
                     ifTrue: [
                        s << (s jQuery id: 'virheala') html: '
                           Ala ei saa olla tyhjä!'.
                        s << (s jQuery: #virheIlmoitus) fadeIn:
                           1000 milliseconds.
                        s << (s jQuery: #virheIlmoitus) fadeOut:
                            1500 milliseconds ];
                     ifFalse: [
                        uusiharrastusVirheita > 0
                           ifTrue: [
                              s << (s jQuery: #virheIlmoitus)
                                 fadeIn: 1000 milliseconds.
                              s << (s jQuery: #virheIlmoitus)
                                 fadeOut: 1500 milliseconds ];
                           ifFalse: [
                              tallennettavaHarrastus rekisteroi.
                              kerho lisaa:
                                 tallennettavaHarrastus.

                              s << (s jQuery: #
                                 uusiHarrastusDialog) dialog
                                 close.
                              s << (s jQuery: #harrastusLomake)
                                 html: [ :r | r render: [ self
                                 naytaHarrastusLomake: r ]].
                              s << (s jQuery: #harrastusLomake)
                                 fadeOut: 400 milliseconds.
                              s << (s jQuery: #harrastusLomake)
                                 fadeIn: 400 milliseconds. ]]]))
                                 ;
         with: [
            self luoUusiHarrastusLomake: html.
            html paragraph
               id: #virheIlmoitus;
               class: 'virheIlmoitus';
               with: 'Virheellisiä syötteitä!' ].

   self dialogiAukaisuNappi: html.


dialogiAukaisuNappi: html
   html submitButton
      onClick:
         (html jQuery ajax
            script: [ :s |
               uusiharrastusVirheita := 0.
               tallennettavaHarrastus := Harrastus forJasen:
                  jasenId.
               s << (s jQuery: #uusiHarrastusLomake) html: [ :r | r
                   render: [ self luoHarrastusLomakeTaulu: r ]]]);
      onClick: (html jQuery: #uusiHarrastusDialog) dialog open;
```

```
      with: 'Lisää uusi harrastus'.


luoUusiHarrastusLomake: html
   tallennettavaHarrastus := Harrastus forJasen: jasenId.

   html div
      id: #uusiHarrastusLomake;
      with: [ self luoHarrastusLomakeTaulu: html ].


luoHarrastusLomakeTaulu: html
   html table
      class: 'tableUusiHarrastusKentat';
      with: [
         tallennettavaHarrastus ekaKentta to:
            tallennettavaHarrastus getKenttia - 1 do: [ :k |
            html tableRow: [ self luoUusiHarrastusRivit: html and:
               k ]]].


luoUusiHarrastusRivit: html and: k
| kysymys |
   kysymys := tallennettavaHarrastus getKysymys: k.

   html tableData: [ html text: kysymys, ': '. ].

   html tableData: [ self luoUusiHarrastusKentta: html and: k and:
      kysymys. ].

   html tableData: [
      html label
         id: 'virhe',kysymys;
         class: 'virheLabel';
         with: ''. ].


luoUusiHarrastusKentta: html and: k and: kysymys
| tdClass virhe arvo onkoVirheTyyli |
   k = tallennettavaHarrastus ekaKentta
      ifTrue: [ tdClass := 'harrastusekakentta' ];
      ifFalse: [ tdClass := 'harrastuskentta' ].

   html textInput
      id: 'text',kysymys;
      class: tdClass;
      value: '';
      onKeyUp: (html jQuery ajax
         callback: [ :value | onkoVirheTyyli := value ] value: ((
            html jQuery this) hasClass: 'virheKentta');
         callback: [ :value | arvo := value ] value: (html jQuery
            this) value;

         script: [ :s |
            virhe := tallennettavaHarrastus aseta: k and: arvo.

            virhe = '' & onkoVirheTyyli = 'true'
               ifTrue: [
                  uusiharrastusVirheita := uusiharrastusVirheita -
                     1.
                  s << (s jQuery id: 'text',kysymys) removeClass: '
                     virheKentta' ].

            virhe ~= '' & onkoVirheTyyli = 'false'
```

```
                ifTrue: [
                    uusiharrastusVirheita := uusiharrastusVirheita +
                        1.
                    s << (s jQuery id: 'text',kysymys) addClass: '
                        virheKentta' ].

            s << (s jQuery id: 'virhe',kysymys) html: [ :h | h
                render: virhe ]]).


poistaHarrastusNappi: html
   html submitButton
      onClick: (html jQuery ajax
         confirm: 'Poistetaanko valittu harrastus?';

         script: [ :s |
            kerho poistaHarrastus: tallennettavaHarrastus.
            s << (s jQuery: #harrastusLomake) html: [ :r | r render
                : [ self naytaHarrastusLomake: r ]].
            s << (s jQuery: #harrastusLomake) fadeOut: 400
                milliseconds.
            s << (s jQuery: #harrastusLomake) fadeIn: 400
                milliseconds. ]);
      with: 'Poista harrastus'.
```

H.4 KerhoSession

```
WASession subclass: #KerhoSession
   instanceVariableNames: 'hakuehto hakukentta'
   classVariableNames: ''
   poolDictionaries: ''
   category: 'WWWKerho'


getHakuehto
   ^ hakuehto.


getHakukentta
   ^ hakukentta.


onkoTallennettu
   ^ hakuehto notNil.


tallenna: ehto and: kentta
   hakuehto := ehto.
   hakukentta := kentta.
```

H.5 KerhoFileLibrary

```
WAFileLibrary subclass: #KerhoFileLibrary
   instanceVariableNames: ''
   classVariableNames: ''
   poolDictionaries: ''
   category: 'WWWKerho'
```

```
kerhoCss
^ '
    .hakuLomake {
       margin-bottom: 30px;
    }
    .labelJasenet {
       margin-top: 0px;
       font-weight: bold;
    }
    .jasenValintalista {
       float: left;
    }
    .jasenTiedot {
       float: left;
       margin-left: 30px;
    }
    .jasenOtsikko {
       margin-top: 0px;
       font-weight: bold;
       text-decoration: underline;
    }
    .boldText {
       font-weight: bold;
    }
    .jasenHarrastukset {
       float: left;
       margin-left: 30px;
       margin-top: 0px;
    }
    .selectJasen {
       min-width: 120px;
    }
    .tableHarrastukset {
       text-align: center;
       border: 1px solid black;
       border-collapse: collapse;
    }
    .tableHarrastukset, tr, td {
       border: 1px solid black;
       padding: 5px;
    }
    .tableHarrastukset caption {
       font-weight: bold;
    }
    .tableEka {
       text-align: left;
    }
    .tableJasenTiedot tr {
       padding: 0px;
       margin: 0px;
    }
    .tableJasenTiedot tr td{
       padding: 0px;
       margin: 0px;
       border: none;
    }
    .tableJasenKentat tr {
       padding: 0px;
       margin: 0px;
    }
    .tableJasenKentat tr td{
       padding: 0px;
       margin: 0px;
```

```
      border: none;
   }
   .virheKentta {
      border-color: red;
   }
   .virheLabel {
      color: red;
   }
   .virheIlmoitus {
      color: white;
      text-align: center;
      background-color: red;
      width: 100%;
      display: none;
   }
   .piilotettu {
      display: none;
   }
   .jasenkentta {
      width: 150px;
   }
'


jasenCss
^ '
   .jasenTietoLomake{
      float: left;
      margin-left: 30px;
      margin-top: 20px;
   }
   .jasenTietoLomake p{
      margin-top: 0px;
      font-weight: bold;
      text-decoration: underline;
   }
   .harrastusLomake{
      float: left;
      margin-left: 30px;
      margin-top: 40px;
   }
   .boldText {
      font-weight: bold;
   }
   .ilmoitus {
      width: 200px;
      display: none;
   }
   .tableHarrastukset {
      text-align: center;
      border: 1px solid black;
      border-collapse: collapse;
   }
   .tableHarrastukset, tr, td {
      padding: 5px;
   }
   .tableHarrastukset, tr {
      border: 1px solid black;
      padding: 5px;
   }
   .tableHarrastukset caption {
      font-weight: bold;
   }
   .tableJasenet tr {
      padding: 0px;
```

```
      margin: 0px;
   }
   .tableJasenet tr td{
      padding: 0px;
      margin: 0px;
   }
   .tableUusiHarrastusKentat tr {
      padding: 0px;
      margin: 0px;
   }
   .tableUusiHarrastusKentat tr td {
      padding: 0px;
      margin: 0px;
   }
   .tableJasenKentat tr {
      padding: 0px;
      margin: 0px;
   }
   .tableJasenKentat tr td{
      padding: 0px;
      margin: 0px;
      border: none;
   }
   .virheKentta {
      border-color: red;
   }
   .virheLabel {
      color: red;
   }
   .virheIlmoitus {
      color: white;
      text-align: center;
      background-color: red;
      width: 100%;
      display: none;
   }
   .harrastusekakentta {
      width: 130px;
      padding: 0px;
      margin: 0px;
   }
   .harrastuskentta {
      width: 50px;
      padding: 0px;
      margin: 0px;
   }
   .jasenkentta {
      width: 150px;
   }
'


ajaxloaderGif
   (kuva lisätty konfiguraatiosivulla)
```

I Valuuttamuunnin ilman sovelluskehyksiä -lähdekoodit

I.1 index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
    www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fi" lang="fi">
<head>
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript" src="muunnin.js"></script>
<title>Valuuttamuunnin</title>
</head>
<body>
<form>
   <p>
      <input type="text" id="textSumma" />
      <select id="selectValuutta"><option value=""> </option></
          select>
      <input type="button" id="buttonMuunna" value="Muunna" />
      <strong id="labelTulos">0 &#8364;</strong>
   </p>
</form>
</body>
</html>
```

I.2 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This web.xml file is not required when using Servlet 3.0
    container,
      see implementation details http://jersey.java.net/nonav/
         documentation/latest/jax-rs.html#d4e194 -->
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="
    http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/
    javaee/web-app_2_5.xsd" version="2.5">
  <servlet>
    <servlet-name>Valuuttamuunnin</servlet-name>
    <servlet-class>valuuttaMuunnin.MuunninServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Valuuttamuunnin</servlet-name>
    <url-pattern>/valuuttaMuunnin</url-pattern>
  </servlet-mapping>
</web-app>
```

I.3 muunnin.js

```
window.onload = function() {
   $.ajax({
      async: true,
      url: "valuuttaMuunnin",
      data: "",
      processData: false,
      type: "POST",
      success: function(data, textStatus, request){
         $("#selectValuutta").html(request.responseText);
      }}
   );

   $('#buttonMuunna').click(laheta);
```

```
   function laheta(e) {
      var summa = $("#textSumma").val();
      var valuutta = $("#selectValuutta").val();
      var tulos;

      $.ajax({
         async: true,
         url: "valuuttaMuunnin",
         data: "summa=" + summa + "&valuutta=" + valuutta,
         processData: false,
         type: "POST",
         success: function(data, textStatus, request){
            if (request.responseText === "virhe") {
               $("#labelTulos").text("Virheellinen syöte!");
            }
            else {
               tulos = request.responseText + " euroa";
               $("#labelTulos").text(tulos);
            }
         }
      });
   }
};
```

I.4 MuunninServlet.java

```
package valuuttaMuunnin;

import java.io.IOException;
import java.util.List;
import javax.servlet.*;
import javax.servlet.http.*;

public class MuunninServlet extends HttpServlet  {
   private static final long serialVersionUID = 1L;

   private ValuuttaMuunnin muunnin;

   public void init() throws ServletException {
      muunnin = new ValuuttaMuunnin();
   }

   public void doPost(HttpServletRequest request,
       HttpServletResponse response)
          throws ServletException, IOException {

      response.setContentType("text/plain");

      if (request.getParameter("summa") == null) {
         List<String> nimet = muunnin.getValuuttaNimet();
         String selectHTML = "";

         for (String nimi : nimet) {
            selectHTML = selectHTML + "<option value=\"" + nimi + "
               \">" + nimi + "</option>";
         }

         response.getWriter().write(selectHTML);
      }
      else {
```

```
            String maara = request.getParameter("summa");
            String valuutta = request.getParameter("valuutta");

            String tulos = muunnin.muunnaValuutta(valuutta, maara);
            response.getWriter().write(tulos);
        }
    }

    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        doPost(request,response);
    }

    public void destroy() {
        // do nothing.
    }
}
```

J Django-valuuttamuunninsovelluksen lähdekoodit

J.1 index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

{% load dajaxice_templatetags %}

<head>
<title>Testi</title>
<script type="text/javascript" src='http://code.jquery.com/jquery
    -1.8.2.js'></script>
<script type="text/javascript" src="/static/dajax/jquery.dajax.core
    .js"></script>
{% dajaxice_js_import %}
</head>
<body>
<form action="" method="post">
    {% csrf_token %}
    <p>
        <input type="text" value="" id="textSumma">
        {{ valuuttaField }}
        <input type="button" value="Muunna" onclick="muunna();"> =
        <label id="labelTulos">0 &euro;</label>
    </p>
</form>

<script type="text/javascript">
function muunna(){
    Dajaxice.djangoMuunnin.laskeTulos(Dajax.process, {'valuutta':$('
        #id_selectValuutta').val(), 'summa':$('#textSumma').val()});
}
</script>
</body>
</html>
```

J.2 forms.py

```
from django import forms
from djangoMuunnin.valuuttamuunnin import ValuuttaMuunnin

class ValuuttaSelectField(forms.Form):

   muunnin = ValuuttaMuunnin()
   valuutat = muunnin.getValuuttaNimet()

   VALUUTTA_VALINNAT = [(valuutta, valuutta) for valuutta in
      valuutat]

   selectValuutta = forms.ChoiceField(choices=VALUUTTA_VALINNAT,
      widget=forms.Select(), label='')
```

J.3 ajax.py

```
from dajax.core import Dajax
from dajaxice.decorators import dajaxice_register
from djangoMuunnin.valuuttamuunnin import ValuuttaMuunnin

@dajaxice_register
def laskeTulos(request, valuutta, summa):
   dajax = Dajax()

   muunnin = ValuuttaMuunnin()
   tulos = muunnin.muunnaValuutta(valuutta, summa)

   if tulos == "virhe":
      dajax.assign('#labelTulos','innerHTML','Virheellinen syöte!')
   else:
      dajax.assign('#labelTulos','innerHTML',tulos + ' &euro;')

   return dajax.json()
```

J.4 views.py

```
# -*- coding: ISO-8859-1 -*-

from django.http import HttpResponse
from django.core.context_processors import csrf
from django.shortcuts import render_to_response
from djangoMuunnin.forms import ValuuttaSelectField

def djangoMuunninLomake(request):

   c = {}

   valuuttaField = ValuuttaSelectField()

   c.update(csrf(request))
   c.update({'valuuttaField': valuuttaField})

   return render_to_response('djangoMuunnin/index.html', c)
```

J.5 urls.py

```
from django.conf.urls import patterns, include, url
from dajaxice.core import dajaxice_autodiscover, dajaxice_config
from django.conf import settings
from django.contrib.staticfiles.urls import staticfiles_urlpatterns
from djangoMuunnin.views import djangoMuunninLomake

dajaxice_autodiscover()

urlpatterns = patterns('',
   url(r'^valuuttamuunnin/$', djangoMuunninLomake),
   url(dajaxice_config.dajaxice_url, include('dajaxice.urls')),
)

urlpatterns += staticfiles_urlpatterns()
```

K JavaServer Faces -valuuttamuunninsovelluksen lähdekoodit

K.1 index.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:h="http://java.sun.com/jsf/html">

<h:head>
   <title>JSF 2.0 ValuuttaMuunnin</title>
</h:head>
<h:body>
   <h3>ValuuttaMuunnin JSF:llä</h3>
   <h:form>
      <h:panelGrid id="panel" columns="4" border="0" cellpadding="0
          " cellspacing="6">
         <h:inputText id="arvo" value="#{muunnin.arvo}"></h:
             inputText>

         <h:selectOneMenu id="selectValuutta" value="#{muunnin.
             selectValuutta}">
            <f:selectItems value="#{muunnin.valuutat}" />
         </h:selectOneMenu>

         <h:commandButton value="Muunna">
            <f:ajax execute="arvo selectValuutta" render="output"
                />
         </h:commandButton>

         <h:outputText id="output" value=" #{muunnin.muunna} " />
      </h:panelGrid>
   </h:form>
</h:body>
</html>
```

K.2 Muunnin.java

```
package muunnin;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.model.SelectItem;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

@ManagedBean
@SessionScoped
public class Muunnin implements Serializable {
   private static final long serialVersionUID = 1L;

   private ValuuttaMuunnin muunnin;
   private List<SelectItem> valuutat;
   private String arvo = "";
   private String selectValuutta = "0";

   public Muunnin() {
      muunnin = new ValuuttaMuunnin();
      String[] nimet = muunnin.getValuuttaNimet();

      valuutat = new ArrayList<SelectItem>();
      for (String nimi : nimet) {
         valuutat.add(new SelectItem(nimi));
      }
   }

   public String getArvo() {
      return arvo;
   }
   public void setArvo(String value) {
      this.arvo = value;
   }

   public String getSelectValuutta() {
      return selectValuutta;
   }
   public void setSelectValuutta(String value) {
      this.selectValuutta = value;
   }

   public List<SelectItem> getValuutat() {
      return valuutat;
   }

   public String getMuunna() {
      if (arvo.equals("")) return "0,00 euroa";

      String tulos = muunnin.muunnaValuutta(selectValuutta, arvo);

      if (tulos.equals("virhe")) return "Virheellinen luku!";

      return tulos + " euroa";
   }
}
```

K.3 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="
    http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/
    javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
   <display-name>JSFMuunnin</display-name>
   <servlet>
      <servlet-name>Faces Servlet</servlet-name>
      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class
         >
      <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
      <servlet-name>Faces Servlet</servlet-name>
      <url-pattern>*.xhtml</url-pattern>
   </servlet-mapping>
   <context-param>
      <description>State saving method: 'client' or 'server' (=
         default). See JSF Specification 2.5.2</description>
      <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
      <param-value>client</param-value>
   </context-param>
   <context-param>
      <param-name>javax.servlet.jsp.jstl.fmt.localizationContext</
         param-name>
      <param-value>resources.application</param-value>
   </context-param>
   <listener>
      <listener-class>com.sun.faces.config.ConfigureListener</
         listener-class>
   </listener>
</web-app>
```

L Wicket-valuuttamuunninsovelluksen lähdekoodit

L.1 Muunnin.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
   <title>Wicket ValuuttaMuunnin</title>
</head>
<body>
<form wicket:id="form">
   <input type="text" wicket:id="textSumma"/>
   <select wicket:id="selectValuutta"></select>
   <input type="button" value="Muunna" wicket:id="buttonMuunna"/>
   <span wicket:id="labelTulos">0</span>
</form>
</body>
</html>
```

L.2 Muunnin.java

```
package muunnin;

import java.util.List;
import org.apache.wicket.ajax.AjaxRequestTarget;
import org.apache.wicket.ajax.markup.html.form.AjaxButton;
import org.apache.wicket.markup.html.WebPage;
import org.apache.wicket.markup.html.basic.Label;
import org.apache.wicket.markup.html.form.Button;
import org.apache.wicket.markup.html.form.DropDownChoice;
import org.apache.wicket.markup.html.form.Form;
import org.apache.wicket.markup.html.form.TextField;
import org.apache.wicket.model.Model;
import org.apache.wicket.model.PropertyModel;
import org.apache.wicket.request.mapper.parameter.PageParameters;

public class Muunnin extends WebPage {

   private static final long serialVersionUID = 1L;

   private Label labelTulos;
   private TextField<String> textSumma;
   private List<String> valuutat;
   private DropDownChoice<String> selectValuutta;
   private ValuuttaMuunnin muunnin;
   private String valittuValuutta = "";

   @SuppressWarnings({ "serial", "rawtypes" })
   public Muunnin(final PageParameters parameters) {
      super(parameters);

      muunnin = new ValuuttaMuunnin();
      valuutat = muunnin.getValuuttaNimet();
      valittuValuutta = valuutat.get(0);

      Form form = new Form("form");
      textSumma = new TextField<String>("textSumma", Model.of(""));
      labelTulos = new Label("labelTulos", "0 euroa");
      // Mahdollistaa tuloksen päivittämisen Ajaxilla.
      labelTulos.setOutputMarkupId(true);

      selectValuutta = new DropDownChoice<String>(
            "selectValuutta", new PropertyModel<String>(this, "
               valittuValuutta"), valuutat);

      Button buttonMuunna = new AjaxButton("buttonMuunna"){
         @Override
         public void onSubmit(AjaxRequestTarget target, Form form)
             {
            String tulos = muunnin.muunnaValuutta(selectValuutta.
               getDefaultModelObjectAsString(), textSumma.
               getDefaultModelObjectAsString());
            if (tulos.equals("virhe")) labelTulos.
               setDefaultModelObject("Virheellinen syöte!");
            else labelTulos.setDefaultModelObject(tulos + " euroa")
               ;
            target.add(labelTulos);
         }
      };

      form.add(textSumma);
      form.add(selectValuutta);
      form.add(buttonMuunna);
      form.add(labelTulos);
```

```
        add(form);
    }
}
```

L.3 MuunninApplication.java

```
package muunnin;

import org.apache.wicket.markup.html.WebPage;
import org.apache.wicket.protocol.http.WebApplication;

public class MuunninApplication extends WebApplication
{
    @Override
    public Class<? extends WebPage> getHomePage()
    {
        return Muunnin.class;
    }
    @Override
    public void init()
    {
        super.init();
        // add your configuration here
    }
}
```

L.4 web.xml

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
<display-name>Wicket ValuuttaMuunnin</display-name>
<filter>
    <filter-name>ValuuttaMuunnin</filter-name>
    <filter-class>org.apache.wicket.protocol.http.WicketFilter</
        filter-class>
    <init-param>
        <param-name>applicationClassName</param-name>
        <param-value>muunnin.MuunninApplication</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>ValuuttaMuunnin</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
</web-app>
```

M ZK-valuuttamuunninsovelluksen lähdekoodit

M.1 index.zul

```
ï»¿<zk>
   <window apply="muunnin.ZKMuunnin.MuunninController">
      <hlayout>
         <textbox id="textSumma" />
         <selectbox id ="selectValuutta"></selectbox>
         <button id ="buttonMuunna" label="Muunna" />
         <label id="labelTulos" value ="0 euroa"/>
      </hlayout>
   </window>
</zk>
```

M.2 MuunninController.java

```
package muunnin.ZKMuunnin;

import java.util.List;
import org.zkoss.bind.annotation.Init;
import org.zkoss.zk.ui.Component;
import org.zkoss.zk.ui.select.SelectorComposer;
import org.zkoss.zk.ui.select.annotation.Listen;
import org.zkoss.zk.ui.select.annotation.Wire;
import org.zkoss.zul.Button;
import org.zkoss.zul.Label;
import org.zkoss.zul.ListModelList;
import org.zkoss.zul.Selectbox;
import org.zkoss.zul.Textbox;

public class MuunninController extends SelectorComposer<Component>
    {

   private static final long serialVersionUID = 1L;

   @Wire
   private Textbox textSumma;
   @Wire
   private Label labelTulos;
   @Wire
   private Button buttonMuunna;
   @Wire
   private Selectbox selectValuutta;

   private List<String> valuutat;
   private ValuuttaMuunnin muunnin;

   @Init
   public void init() {}

   @Listen("onClick=#buttonMuunna")
   public void doMuunna() {
      String tulos = muunnin.muunnaValuutta(valuutat.get(
         selectValuutta.getSelectedIndex()), textSumma.getValue());

      if (tulos.equals("virhe")) labelTulos.setValue("Virheellinen
         syöte!");
      else labelTulos.setValue(tulos + " euroa");
   }

   @Override
   public void doAfterCompose(Component comp) throws Exception{
      // Pitää tehdä!
```

```
        super.doAfterCompose(comp);

        muunnin = new ValuuttaMuunnin();
        valuutat = muunnin.getValuuttaNimet();

        ListModelList<String> listModel = new ListModelList<String>(
            valuutat);
        // Laitetaan valituksi ensimmäinen valuutta,
        // muuten Selectboxiin tulee tyhjä valinta.
        listModel.addToSelection(valuutat.get(0));

        selectValuutta.setModel(listModel);
    }
}
```

M.3 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun
    .com/xml/ns/j2ee/web-app_2_4.xsd">

    <description>
        <![CDATA[My ZK Application]]>
    </description>
    <display-name>ZKMuunnin</display-name>

    <!-- //// -->
    <!-- ZK -->
    <listener>
        <description>ZK listener for session cleanup</description>
        <listener-class>org.zkoss.zk.ui.http.HttpSessionListener</
            listener-class>
    </listener>

    <servlet>
        <description>ZK loader for ZUML pages</description>
        <servlet-name>zkLoader</servlet-name>
        <servlet-class>org.zkoss.zk.ui.http.DHtmlLayoutServlet</
            servlet-class>
        <init-param>
            <param-name>update-uri</param-name>
            <param-value>/zkau</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup><!-- Must -->
    </servlet>
    <servlet-mapping>
        <servlet-name>zkLoader</servlet-name>
        <url-pattern>*.zul</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>zkLoader</servlet-name>
        <url-pattern>*.zhtml</url-pattern>
    </servlet-mapping>

    <servlet>
        <description>The asynchronous update engine for ZK</
            description>
        <servlet-name>auEngine</servlet-name>
```

```
      <servlet-class>org.zkoss.zk.au.http.DHtmlUpdateServlet</
         servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>auEngine</servlet-name>
      <url-pattern>/zkau/*</url-pattern>
   </servlet-mapping>
</web-app>
```