WWW-sovellusten rakentaminen Javalla

Pekka Kosonen

Sisällysluettelo

1 Johdanto

Tämän tutoriaalin pääideana on toimia oppaana WWW-sovellusten[1] rakentamiseen Javalla.

Tutoriaalin tarkoituksena ei ole missään nimessä toimia kaiken tietävänä raamattuna, vaan lähinnä auttaa alkuun pääsyyn. Johtuen tutoriaalin luonteesta on erittäin suositeltavaa hankkia hyvä servletti/JSP kirja (muutama mainittu lähteissä).

Lukijalta oletetaan jonkin verran Java[2] kielen tuntemusta. Jos osaamisesi Javan piirissä on heikkoa, voit joutua turvautumaan alan kirjallisuuteen useasti esimerkkejä pohtiessasi.

Toivottavasti tutoriaalista on sinulle hyötyä!

Tutoriaalin kotisivuilta saat kaikki tarvittavat koodiesimerkit:

http://sinuhe.jypoly.fi/~pkosonen/webapp/

1.1 Sisällys lyhyesti

- Tomcat[3] palvelimen asennus, configurointi ja käyttö.
- Servlettien[4] perusteet.
- JSP[5] (Java Server Pages) perusteet.
- Tomcat tarkemmin:
    - Manager Application.
    - Security Realms.
    - Valves.

1.2 Lukijalle tiedoksi

Hakemistojen nimet ja sijainnit saattavat muuttua seuraavissa Tomcatin versioissa. Jos haluat käyttää tätä tutoriaalia täsmälleen sellaisena kuin se on nyt, niin käytä tomcatin versiota 4.1.12.

Tekstissä monet termit on ilmaistu myös alkuperäisellä (englanninkielisellä) vastineellaan, täten lukijan on helpompi tutkia englanninkielistä kirjallisuutta tietäessään perustermit.

---

[1] Tässä WWW-sovellus käsitetään sovelluksena, jota voidaan käyttää Internetin avulla standardeja protokollia hyödyntäen. Asiakasohjelmaksi käsitetään Internet-selain (esim. IE, NetScape).
[2] Lisätietoa saatavilla osoitteesta http://java.sun.com
[3] Java ”websovellusmoottori”, ks. http://jakarta.apache.org/tomcat/
[4] http://java.sun.com/products/servlet/download.html
[5] http://java.sun.com/products/jsp/download.html

Tutoriaali on kirjoitettu puhtaasti Windowsin käyttäjille. Siksi monet asiat, kuten ympäristömuuttujien asettaminen, jäävät muuta käyttöjärjestelmää käyttävän lukijan itsensä vastuulle.

1.2.1 Koodit

Koko tutoriaalissa olevat koodit on saatavissa yhdessä paketissa, tutor.war[6]. Kaikki mitä sinun tarvitsee tehdä tutoriaalin esimerkkien ajamiseen on kopioida tämä paketti Tomcatin asennuksen jälkeen asennushakemiston alla olevaan webapps hakemistoon ja käynnistää Tomcat.

Ensimmäisestä osasta on olemassa erillinen paketti (tutor_osa1.zip[6]), jota suositellaan käytettäväksi mikäli et ole aiemmin Tomcatin hakemistorakenteeseen ja toimintaan tutustunut. Tällöin laitat itse tiedostot toimintaan oppien samalla Tomcatin käyttöä.

1.2.2 Kenelle tämä tutoriaali on tarkoitettu

Tutoriaali on pääasiallisesti kirjoitettu jonkin verran Javaa osaavalle henkilölle, jota ei pelota tutkia mystiseltä tuntuvaa lähdekoodia alan raamattu kädessä.

Tutoriaalin lukijalta oletetaan perustiedot Javasta, olio-ohjelmoinnista ja appleteista sekä minimaaliset tiedot seuraavista alueista:

- XML[7]
- HTML[8]
- Tietoliikenteen erittäin kevyt ymmärrys (palvelin/asiakas, pyyntö/vastaus eli request/response riittää).

1.2.3 Tarvitsemasi ohjelmat

Tarvitset välttämättä seuraavat ohjelmat

- Zip-pakettien purkamiseen esim. winzip[9].
- Java SDK[10] (Sofware Development kit).
- Tomcat[11], versio 4.12 tai uudempi.

Lisäksi on erittäin suureksi hyödyksi jos sinulla on jokin Java-editorin, kuten NetBeans (www.netbeans.org) tai JBuilder (www.borland.com).

---

[6] Ks. http://sinuhe.jypoly.fi/~pkosonen/webapp/
[7] Ks. http://www.w3.org/XML/
[8] Ks. http://www.w3.org/MarkUp/
[9] Ks. http://www.winzip.com
[10] Ks. http://java.sun.com/j2se/
[11] Ks. http://jakarta.apache.org/tomcat/

1.3 Servletit ja JSP lyhyesti

Servletit ja JSP ovat olennainen osa J2EE:tä[12] (Java 2 Enterpise Edition). *Servletti* on ympäristöriippumaton laajennus web-palvelimeen. Servletit kommunikoivat asiakkaiden (clients) kanssa pyynnön/vastauksen (request/response) avulla. Servletin muistiin lataamisesta ja poistamisesta sekä pyyntöjen välittämisestä yms. asioista huolehtii *servlettimoottori* (servlet container / servlet engine).

Java Server Pages eli JSP on laajennus servletteihin. Useimmiten servletti tulostaa HTML-sivun lähdekoodin (tai XML:ää tms.) muodostaen dynaamisesti www-sivun. JSP teknologialla on mahdollista muodostaa www-sivu, joka sisältää sekä staattisia että dynaamisia osia. JSP sivu on loppujen lopuksi kuitenkin riippuvainen servleteistä, sillä JSP koodi käännetään servlettimoottorin toimesta servletin lähdekoodiksi (joka puolestaan käännetään tavukoodiksi jota virtuaalikone suorittaa) ennen JSP sivun suorittamista.

JSP:n alkaessa yleistyä näytti siltä, että servletit ovat jäämässä pahasti JSP:n jalkoihin. Servlettejä pidettiin vanhentuneena teknologiana. Kuitenkin edelleen servlettejä käytetään lähes kaikessa muussa paitsi käyttöliittymän muodostamisessa.

[12] Ks. http://java.sun.com/j2ee/index.jsp

2 Tomcat

Edellisessä kappaleessa puhuttiin servlettien ja JSP:n vaatimasta servletti moottorista. Tomcat on eräs niistä.

Tomcat on kehitetty avoimena järjestelmänä, ja sen jakelu tapahtuu Apachen lisenssin alla. Tomcatistä julkaistaan uusia versioita nopeaan tahtiin. Seuraava versio (versio 5) on tätä kirjoitettaessa (tammikuu 2003) jo alpha vaiheessa. Eri Tomcatin versioissa on kiinnitettävä huomiota erityisesti siihen, mitä versiota servleteistä ja JSP:stä se tukee. Tomcat 4.1.12 tukee servlettien spesifikaatiota 2.3 ja JSP spesifikaatiota 1.2.

Tomcat on saatavilla osoitteesta

http://jakarta.apache.org/tomcat/

2.1 Huomautus Tomcatin versioista

Jos käytät uudemman spesifikaation piirteitä, joita Tomcat ei tue, ei sovelluksesi todennäköisesti toimi lainkaan.

Tomcat on saatavilla useaan eri alustaan.

2.2 Tomcatin pystytys

Tutoriaalia kirjoitettaessa on asennettu Tomcatin versio 4.1.12. Todennäköisesti ohjeet toimivat myös uudemmille versioille. Asennus on suoritettu ”standalone” serverinä, eli Tomcat-palvelinta ei ole integroitu mihinkään toiseen palvelimeen. Muihin palvelimiin integroinnista löytyy informaatiota Tomcatin dokumentaatiosta.

2.2.1 Asennus ja configurointi

Lataa Java SDK (Java Software Development Kit)

Tarvitset Java SDK:n[13]. Mikäli olet jo SDK:n asentanut (vähintään versio 1.2), ei uuden asentaminen ole välttämätöntä (suosittelen vähintään versiota 1.3). Tutoriaalia tehtäessä on käytetty SDK versiota 1.4. Lisäksi joissakin esimerkeissä on tarvittu Enterprise Editionia. Tarkemmat tiedot SDK-versioista saat Sunin sivuilta

http://java.sun.com/

Lataa Tomcat

Valitse mahdollisimman uusi versio (ei kannata valita alpha- tai beta-julkaisua).

Tomcatin asennus Windowsiin on äärimmäisen helppoa asennusohjelman ansiosta. Tomcatin saat ladattua osoitteesta

http://jakarta.apache.org/tomcat/

[13] Aiemmin käytettiin nimitystä JDK (versiot 1.0, 1.1, 1.2).

2.2.2 Asetusten muokkaaminen eli configurointi

Oletuksena Tomcatin asetukset eivät ole välttämättä parhaat mahdolliset opiskelukäyttöön. Asetukset saattavat vaihdella suurestikin eri versioiden mukaan.

1. Otetaan ROOT web sovellus käyttöön

Tästä edespäin viittaan Tomcatin asennushakemistoon[14] nimellä TOMCAT_HOME.

Mikäli haluat ottaa käyttöön valmiina olevan web sovelluksen nopeata testaamista varten, ota pois kommenteista TOMCAT_HOME\conf\server.xml tiedostosta seuraava rivi

```
<Context path="" docBase="ROOT" debug="0"/>
```

Tällöin saat haluamasi JSP sivut ja servletit näkymään suoraan osoitteessa

http://localhost/sivu.jsp tai http://localhost/servlet/paketin_nimi.servletin_nimi.

Servlet invoker

Tomcatissä on oletuksena servlet invoker sovellus asetettu pois päältä. Tämän sovelluksen saataville asettaminen (enable) helpottaa servlettien testaamista. Mikäli et muuta tätä asetusta, joudut kuvaamaan erikseen jokaisen servletin tekemiisi web sovelluksiin web.xml tiedostoon[15].

Ota pois kommenteista TOMCAT_HOME\conf\web.xml tiedostosta rivit

```
<servlet-mapping>
<servlet-name>invoker</servlet-name>
<url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

Tekemäsi muutos vaikuttaa kaikkiin websovelluksiin, koska muutos tehtiin palvelimen asetuksiin.

Porttimääritys

Mikäli sinulla ei ole muita palvelimia konfiguroituna[16] kuuntelemaan porttia 80, kannattaa se asettaa Tomcat-palvelimen portiksi. Muuta TOMCAT_HOME\conf\server.xml tiedostoa seuraavasti

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector" port="80">
```

[14] Hakemisto, johon Tomcat on asennettu.
[15] Tästä on kerrottu lisää kappaleessa web.xml.
[16] Portti voi olla vain yhden palvelimen käytössä kerrallaan, oletusportti web palvelimissa on 80.

Oletuksena Tomcatin portti on 8080, milloinka joutuisit käyttämään syntaksia:

http://palvelin:8080/

Servlet Reloading

Seuraavaksi määritellään, ladataanko sovelluksen osat automaattisesti uudestaan niitä päivitettäessä.

Asetetaan Tomcat tutkimaan, ovatko servletit/JSP sivut muuttuneet niitä viimeksi ajetusta versiosta. Tomcat siis tarkistaa, tarvitseeko komponentti ladata uudestaan. Etsi server.xml tiedostosta seuraavat sanat:

document roots in places other than the virtual host's appBase directory. -->

ja lisää seuraava rivi suoraan alle

```
<DefaultContext reloadable="true"/>
```

Näin varmistetaan, että kaikkien web sovellusten komponentit ovat aina ajan tasalla. Päivitettyäsi servletin tai JSP sivun uusi versio ladataan ensimmäisen pyynnön tapahtuessa. Tomcatin ikkunassa näet informaatiota servlettien uudelleenlataamisesta.

Mikäli haluat tehdä tämän vain tiettyihin web sovellukseen, aseta vain niiden context-elementtiin[17] reloadable attribuutin arvoksi true.

Mikäli et halua sovellustesi päivittyvän automaattisesti, käytä Tomcatin Manager-ohjelmaa[18] haluamasi web sovelluksen päivittämiseen.

Ympäristömuuttujat

Aseta ympäristömuuttuja **JAVA_HOME** osoittamaan Java SDK:n asennushakemistoasi, esim. c:\j2sdk1.4.1.

Aseta **TOMCAT_HOME** (uusimmissa Tomcat-versioissa **CATALINA_HOME**[19]) Tomcatin asennushakemistoon.

2.2.3 Testing, testing , one two three

Käynnistä Tomcat ajamalla komentokehoitteesta TOMCAT_HOME\bin\startup.bat[20] komentojonotiedosto. Testaa Tomcatin toiminta menemällä selaimella osoitteeseen

http://localhost/[21]

---

[17] Ks. kappale ServletContext.
[18] Ks. kappale Tomcat Manager.
[19] Tarkista ennen asettamista kumpaa syntaksia käyttämäsi Tomcat ymmärtää. Tämä on tosin helppoa vaihtaa. Mikäli nimi on väärin, ei Tomcat käynnisty.
[20] Vastaavasti tapahtuu Tomcatin sulkeminen shutdown.bat komentojonotiedostolla.
[21] Jos Tomcat jossain muussa portissa kuin 80, käytä syntaksia http://localhost:portti_numero .

Kannattaa lisäksi suorittaa jokin valmiista JSP ja servletti esimerkeistä, jotta voit varmistua Tomcatin toimivan oikein.

2.2.4 Tomcatin konfigurointitiedostoista

Tomcatin tärkein konfigurointitiedosto on **TOMCAT_HOME\conf\server.xml**. Pidä aina varmuuskopiota toimivasta versiosta, sillä jos teet vääriä muutoksia tähän tiedostoon, ei Tomcat toimi.

Toinen tärkeä tiedosto on **TOMCAT_HOME\conf\web.xml**, jota ei pidä sekoittaa omien web sovellustesi sisältämään web.xml tiedostoon.

Tiedostojen konfiguroinnista saat lisätietoa osoitteesta

http://jakarta.apache.org/tomcat/tomcat-5.0-doc/index.html

3 Web sovelluksen asentaminen Tomcattiin

Palvelimen pystyttämisen jälkeen päästänkin itse asiaan eli laittamaan toimintaan web sovelluksia (deploy web applications). Seuraavassa kuvataan web sovelluksen rakenteen luominen manuaalisesti (ilman war[22] pakettia) ja Tomcat-palvelimeen toimintaan laittaminen. Koodit saat tutor_osa1.zip[23] tiedostosta.

Normaalisti koko web sovellus pakataan yhteen pakettiin jonka tarkennin on war (web archive). Tämän paketin avulla sovelluksen asentaminen on niinkin yksinkertaista, että riittää sijoittaa kyseinen paketti webapps hakemistoon Tomcatin ollessa käynnissä. Tutoriaalissa ei näin ole toimittu. Syy on se, että lukijalle jää parempi kuva siitä, missä hakemistossa mikäkin tiedosto kuuluu olla.

3.1 Hakemistorakenne

Web sovellusta luodessa hakemistorakenne on oleellinen. Myös kapitaaleilla on merkitystä hakemistojen ja tiedostojen nimissä[24].

Annamme web sovelluksellemme nimeksi tutor. Luo seuraava hakemistorakenne:

1. **TOMCAT_HOME\webapps** tutor
2. **TOMCAT_HOME\webapps** tutor \WEB-INF
3. **TOMCAT_HOME\webapps** tutor \WEB-INF \ classes
4. **TOMCAT_HOME\webapps** tutor \WEB-INF \ lib

Lisäksi voit tehdä muita hakemistoja, esimerkiksi hakemiston 1) alle vaikkapa images hakemiston. Hakemistojen merkitys ja käyttö on seuraavanlainen

- Hakemistoon 1 tai sen alihakemistoihin sijoitat kaikki JSP ja html sivut. sekä kuvat. Luonnollisesti tämä hakemisto näkyy palvelimelta ulospäin (JSP:n lähdekoodeja ei tietenkään ulkopuoliset saa selville, ainoastaan sen suorituksen tuloksena olevan koodin, joka on tyypillisesti html:ää).
- Hakemistoon 2 web.xml tiedosto. Tässä hakemistossa ja sen alla olevat tiedostot eivät näy ulospäin
- Hakemistoon 3 kaikki käännetyt koodit (siis class-tiedostot). Java luokkien pakettirakenne pitää säilyttää samanlaisena kuin niitä tehdessäsi.
- Hakemistoon 4 kaikki sovelluksen tarvitsemat kirjastot (siis sellaiset, joita Tomcat ei oletuksena käytä). Kirjastot ovat useimmiten jar[25] paketteja. Lisäksi

[22] Tarkemmin kerrottu kappaleessa ”WAR ja web sovelluksen käsittely”.
[23] Ks. http://sinuhe.jypoly.fi/~pkosonen/webapp/
[24] Windowsin käyttäjät erityishuomio!
[25] Ks. http://java.sun.com/docs/books/tutorial/jar/

3.1.1 Kirjastoista

Jos haluat sisällyttää uuden kirjaston siten, että kaikki websovellukset voivat sitä käyttää, niin sijoita se TOMCAT_HOME\shared\lib hakemistoon.

Yksittäiset luokat voit sijoittaa kaikkien web sovellusten saataville hakemistoon TOMCAT_HOME\shared\classes.

3.1.2 Hakemistojen nimistä

Hakemistojen nimet ja sijainnit saattavat muuttua seuraavissa Tomcatin versioissa (todella epätodennäköistä, ainoa suurempi muutos käyttäjän kannalta viimeisen kahden vuoden aikana on ollut ympäristömuuttujan TOMCAT_HOME nimeäminen CATALINA_HOME nimellä).

Jos haluat käyttää tätä tutoriaalia täsmälleen sellaisena kuin se on nyt, niin käytä varmuuden vuoksi Tomcatin versiota 4.1.12. Suurella todennäköisyydellä tutoriaalin kaikki osat toimivat myös uudemmissa Tomcat-versioissa.

3.1.3 Vinkki XML tiedostoista

Käsitellessäsi täkeitä XML-tiedostoja server.xml ja web.xml kannattaa niistä pitää varmuuskopioita. Tarina tosielämästä:

Allekirjoittaneella ei Tomcat käynnistynyt XML parserointivirheiden takia. Olin editoinut XML tiedostoja notepadilla, ja sillä kävin ne jälleen läpi. Notepadillä kaikki näytti hyvältä. Avasin XML tiedostot Borlandin JBuilderillä, ja ilmeni että server.xml tiedosto oli vaurioitunut (näkyi tyhjänä). Ilmeisesti ongelma on tiedon siirtämisessä leikepöydän kautta.

Lisäksi xml tiedostot kannattaa validoida. Vanhemmilla Jbuilderin versioilla sovelluksen web.xml tiedosto saattaa vaikuttaa validilta[26], vaikkei se sitä olisi. Tämä saattaa johtaa sovelluksen toimimattomuuteen sekä Tomcatin käynnistymisen hidastumiseen.

3.2 ServletContext

Jokaisella Web sovelluksella (jatkossa kutsun tällaista sovellusta lyhenteellä WEBAPP) on yksi ServletContext. Tämän olion avulla kaikki samaa WEBAPPiin kuuluvat servletit ja JSP:t voivat käyttää yhteisiä olioita. ServletContextin voisi kuvata yhteisenä muistiavaruutena saman web sovelluksen komponenttien kanssa (käytännössä samaan WEBAPPiin kuuluvat komponentit ovat siis samassa alihakemistossa TOMCAT_HOME\webapps\websovelluksen_nimi)

ServletContextin voi ilmoittaa server.xml tiedostossa. Tämä on välttämätöntä, mikäli haluat esimerkiksi saada sovelluksesi lokitiedot eri tiedostoon kuin muiden sovellusten lokitiedot.

---

[26] Dokumentin rakenne vastaa määritettyä DTD:tä (Document Type Definition), ks, http://java.sun.com/dtd/web-app_2_3.dtd

ServletContext ilmoitetaan lisäämällä Context-elementti HOST-elementin sisään TOMCAT_HOME/conf/server.xml tiedostoon.

Lisätään seuraava teksti Host- elementin sisään (juuri ennen elementin lopetustagia </HOST>):

```
<Context path="/tutor" docBase="tutor" debug="0" reloadable="true" />
```

Tässä path=”/tutor” tarkoittaa sitä, että kaikki kutsut, joissa palvelimen URL-osoitteen perään on laitettu teksti /tutor ohjataan kyseiseen WEBAPPiin.

DocBase attribuuttilla määrätään hakemisto jossa sovellus on. Nyt siis hakemisto olisi TOMCAT_HOME\webapps\tutor .

ServletContextin lisäyksen ja poiston saa tehtyä myös Tomcatin administrator[27] ohjelmalla.

Itseasiassa Tomcat luo ServletContextin automaattisesti käynnistyessään, ellei sitä ole ilmoitettu, mutta ei lisää edellä mainittua elementtiä server.xml tiedostoon. Eli voit käyttää ServletContext-oliota sovelluksessasi ilman sen ilmoittamista

4 Web.xml (web application deployment description)

Seuraavaksi luodaan sovelluksen kannalta tärkeä tiedosto, jossa kuvataan web sovellusta ja sen komponentteja (servletit, JSP ym.), eli web.xml tiedosto. Perustiedostomuoto on seuraava.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE web-app PUBLIC
  '-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN'
  'http://java.sun.com/j2ee/dtds/web-app_2_3.dtd'>


<web-app>
  <!-- kuvataan web sovellus -->
</web-app>
```

Tässä tiedostossa määritellään mm. seuraavat asiat web sovelluksestasi <web-app> elementin sisällä:

1. Servlettien määritykset (mm. nimi ja luokan nimi)
2. Servleteille vietävät initialisointi parametrit (alustus, luetaan init()-metodissa)
3. Servlettien ja JSP:n mappaukset (tavallaan annetaan toinen osoite (URI))

[27] Ks. http://localhost/admin/

4. MIME tyyppien mappaukset
5. Resurssien ja ympäristömuuttujien määritykset
6. Session configurointiparametrit
7. Lista virhesivuista (Error pages)
8. Welcome file list (mikä sivu näytetään jollei sivua määritetty, oletuksena ensin index.html ja sitten index.jsp)

Tiedoston pitää olla validi XML-dokumentti. Sovelluksesi saattaa silti toimia, vaikkei dokumenttisi olisikaan validi. Tarkemman kuvauksen tiedostosta näet osoitteesta

http://java.sun.com/dtd/web-app_2_3.dtd

Sijoita tämä tiedosto aina oman web sovelluksesi WEB-INF hakemiston juureen.

4.1 Servlettien määrittely

Servlettien määrittely tapahtuu <servlet> </servlet> tagien väliin. Jokaista servlettiä ei ole pakko (mutta suotavaa) määritellä web.xml tiedostoon. Määrittely on pakollista jos servletti tarvitsee alustusparametreja, jotka se käsittelee init()-metodissa

Servlet elementin sisällä voi olla seuraavat elementit:

Taulukko 1. Servlet elementin rakenne

Elementti	Kuvaus
<servlet-name>	Nimi jolla servlettiä käytetään
<servlet-class>	Servletin luokan nimi (huom! oltava paketin nimi mukana)
<init-param>	Ei pakollinen, voi olla useita. Sisältää alustusparametrin, joka on nimetty *<param-name> nimi </param-name>* ja arvo annettu *<param-value> arvo </param-value>* elementeillä
<load-on-startup>	Ei pakollinen. Käytettävä vain, jos halutaan ladata servletti muistiin jo Tomcatin käynnistyksen yhteydessä. Positiivinen kokonaislukuarvo ilmaisee servlettien latausjärjestyksen.

4.1.1 Esimerkki – alustusparametrit (initParam)

Servletin alustusparametreilla saadaan helpolla ja elegantilla tavalla vähennettyä ”hard koodausta”, eli saadaan ulkoistettua servletin tarvitsemaa mahdollisesti muuttuvaa informaatiota (toinen vaihtoehto olisi esim. Properties[28]-tiedostojen käyttö).

Esimerkkinä tästä on servletti initParam, joka saa yhden alustusparametrin. Servletti myös tulostaa saamansa parametrien arvot kun sitä kutsutaan (molemmat get ja post toimivat samalla lailla). Servletti luo annetun kokoisen kokonaislukutaulukon ja alustaa sen satunnaisilla arvoilla.

Koska servletti kuuluu pakettiin[29] nimeltä esim (katso initParam.java ensimmäinen rivi), pitää servletti sijoittaa webapps\tutor\classes\esim\ hakemistoon.

Servletti kuvataan seuraavasti Web.xml tiedostossa:

```
<servlet>
<servlet-name>initPara</servlet-name>
<servlet-class>initPara</servlet-class>
<init-param>
    <param-name>koko</param-name>
    <param-value>10</param-value>
</init-param>
<init-param>
    <param-name>otsikko</param-name>
    <param-value> WebApplication tutoriaali - servletin alustusparametrit
    </param-value>
</init-param>
</servlet>
```

Parametrit luetaan seuraavasti

```
String otsikko = this.getServletConfig().getInitParameter("otsikko");
String k = this.getServletConfig().getInitParameter("koko");
```

[28] Properties tiedosto on kuten ini-tiedosto, muodostuu avain-arvo pareista. Tiedoston käsittelyyn on Javassa Properties luokka (http://java.sun.com/j2se/1.4.2/docs/api/java/util/Properties.html)
[2] Javassa luokkia voidaan sijoitella eri hakemistoihin käyttötarkoituksensa mukaan, yhdessä hakemistossa olevia luokkia sanotaan yhdeksi paketiksi

Tämän jälkeen kannattaa tietenkin testata, onnistuiko parametrien lukeminen.

Testataksesi esimerkkiä, laita initPara.class webapps\tutor\WEB-INF\classes hakemistoon. Servlettiä voit kutsua seuraavalla URL osoitteella

http://localhost/tutor/servlet/initPara

Kuva 1. InitPara servletin suoritus

Nyt voit huoletta vaikkapa sulkea selaimen ja tulla uudestaan hetken päästä tälle sivulle. Huomaat, että arvotut luvut ovat samat. Miksi?

Syy liittyy servlettien toimintalogiikkaan. Servletit toimivat siten, että kun servletti ladataan ensimmäistä kertaa muistiin, suoritetaan init-metodi. Tämän jälkeen init-metodia ei enää suoriteta. Jos servlettiä tarvitaan usealle asiakkaalle samanaikaisesti, siitä luodaan uusi säie (toisin kuin tavallisissa CGI-ohjelmissa).

Nyt jos suljet Tomcatin ja käynnistät sen uudelleen niin huomaat, että luvut muuttuvat. Syykin on selvä. Init-metodi on suoritettu uudestaan.

Saman lopputuloksen saat aikaan lataamalla web sovelluksesi uudestaan Tomcatin Manager ohjelmalla

http://localhost/manager/html/reload?path=/tutor

Yrittäessäsi tätä sinulta kysytään käyttäjätunnus ja salasana. Mikäli asensit Tomcatin asennusohjelmalla, annoit loppuvaiheessa käyttäjätunnuksen ja salasanan. Jos et muista

niitä tai et käyttänyt asennuseohjelmaa, voit käydä katsomassa toimivat tunnukset TOMCAT_HOME\conf\tomcat-users.xml[30] tiedostosta.

4.2 Servlet-mapping

Jokaiselle esittelemällesi servletille pitäisi määritellä URI, eli osoite jonka avulla servletti voidaan suorittaa. Tämä osoite on oltava yksilöllinen.

Servlettien mappauksella saat aikaan mielenkiintoisia asioita. Voit esimerkiksi aiheuttaa sen, että kaikki kutsut, joissa on lopussa .hötömölö (esim. sivu.hötömölö) ohjataan servlettisi käsiteltäväksi. Tämän saat tehtyä lisäämällä WEBAPPisi web.xml tiedostoon seuraavan tagin

```
<servlet-mapping>
 <servlet-name>
  initPara
 </servlet-name>
 <url-pattern>
  *.hötömölö
 </url-pattern>
</servlet-mapping>
```

kaikkien servlet tagien ulkopuolelle. Nyt jos menet selaimellasi osoitteeseen

http://localhost/tutor/p.hötömölö

niin huomaat, että kutsuttiin initPara servlettiä.

4.3 JSP tiedostot

Valmiin JSP sivun käyttöönotto on mahdollisimman helppoa. Koska JSP käännetään servletiksi servlet-enginen toimesta, ei tarvitse tehdä muuta kuin sijoittaa tekemänsä JSP sivut oman WEBAPP-hakemiston juureen. Täten JSP sivua ei tarvitse erikseen ilmoittaa web.xml tiedostoon.

Tässä vaiheessa käynnistä Tomcat ja testaa

http://localhost/tutor/hello.jsp

tai ellet ole muuttanut Tomcatin porttia (oletusasetuksilla portti on 8080)

http://localhost:8080/tutor/hello.jsp

[30] Tiedostossa kerrotaan kaikki käyttäjätiedot, ellei tiedot ole tietokannassa (ks. kappale ”JDBC realms)

Tässä on oikeastaan yksinkertaisin mahdollinen JSP tiedosto, sillä se ei sisällä tavuakaan Javaa. Saman olisi voinut tehdä myös pelkällä html-sivulla. Kuitenkin JSP sivu käännetään servletiksi ja ajetaan vaikkei kyseinen toimenpide olisikaan välttämätöntä. Opetus on: Älä tee turhaan JSP sivuja.

Kun menet sivulle uudestaan, latausaika on huomattavasti pienempi. Tämä johtuu siitä, että JSP-sivua ei enää uudestaan käännetä servletiksi. Jos teet muutoksia JSP-sivuun, niin tällöin käännös tapahtuu.

5 Sovelluksen eri osat

Web sovellus jakaantuu perinteisesti kolmeen osaan (MVC[31])

- Käyttöliittymä
- Äly eli logiikka
- Tieto

Näiden kolmen kerroksen erillään pitäminen on tärkeää varsinkin sovelluksen ylläpidon kannalta.

5.1 Äly eli logiikka

Javalla web sovellusta tehdessä on oleellista käyttää hyväkseen papuja (Java Beans[32]), joihin sovelluksen varsinainen äly eli toiminnallisuus koodataan. Näin toimiessa on helppoa muuttaa käyttöliittymää tai ohjelmalogiikkaa myöhemmin. Tämä mahdollistaa myös sen, että eri henkilöt tekevät käyttöliittymän ja toiset logiikan. Kannattaa huomata myös hyvin tehtyjen luokkien uudelleenkäytön mahdollisuus.

Eräs esimerkki papujen hyödyistä olisi valuuttojen käyttö. Mikäli ennen euroihin siirtymistä aiempi valuutta (Suomen markka) olisi koodattu suoraan käyttöliittymään, olisi kaikki nämä käyttöliittymän osat jouduttu koodaamaan uudestaan. Jos sovelluksessa olisi käytetty papua, olisi riittänyt koodata uudestaan get-metodi, joka välittää valuutan tiedot.

Tyypillisesti esimerkiksi tietokannasta haku tehdään seuraavasti papuja hyödyntäen

- Asiakas antaa hakutiedot käyttöliittymään (voi olla esim. html-sivu, jsp-sivu, apletti tai flash-sovellus)
- Syöttötiedot lähetetään servletille
- Servletti hakee tiedot tietokannasta
- Hakutulokset ladataan papuun ja papu lisätään sessioon
- Servletti ohjaa asiakkaan selaimen JSP sivulle, joka näyttää pavun sisällön

[31] Ks. http://java.sun.com/blueprints/patterns/MVC-detailed.html
[32] Spesifikaatio nähtävillä osoitteessa http://java.sun.com/products/javabeans/docs/spec.html

Pavuista lisää myöhemmin[33].

5.2 Tieto

Tyypillisesti sovelluksen käsittelemä tieto on tietokannassa. Tietokanta voi sijaita samalla palvelimella kuin sovelluskin.

Javassa tietokantoja käsitellään JDBC[34] (Java Database Connectivity) API:n avulla.

5.3 Käyttöliittymä

Käyttöliittymä muodostetaan tyypillisesti html-sivuilla ja JSP sivuilla. Tyypillisesti käyttöliittymä on siis html-pohjainen, mutta tietyissä tapauksissa ohjelman logiikkaa voidaan siirtää asiakkaan puolelle.

Periaatteessa kaiken voisi tehdä palvelinpuolella, mutta on huomattavasti järkevämpää hoitaa osa toiminnasta asiakkaan selaimessa. Varsinkin asiakkaan antamat tiedot kannattaa tarkastaa ennen kuin ne lähetetään palvelimelle esimerkiksi talletettavaksi tietokantaan.

Tietojen tarkastamisessa asiakkaan puolella on käytännössä kaksi vaihtoehtoa (muita esim. flash[35]):

1. Appletit
2. Scriptikielet kuten JavaScript

5.3.1 Käyttöliittymän ohjelmointi appleteilla

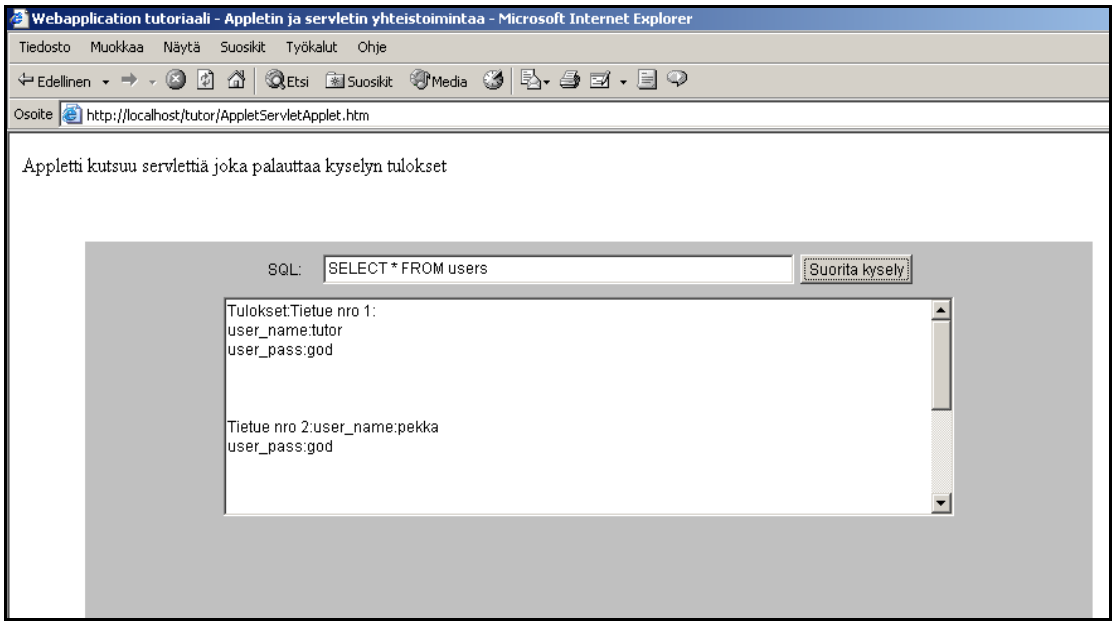
Appletteja[36] käyttämällä pystyt hyödyntämään Java kielen ominaisuuksia. Appletin rakentaminen on helppoa visuaalisten editorien kuten Borlandin Jbuilderin[37] avulla. Tosin applettien lataaminen on hidasta niiden suhteellisen ison koon takia.

Appletilla saat tosin tehtyä kohtuullisen helposti dynaamisia käyttöliittymiä, sillä ne voivat kommunikoida servlettien ja JSP sivujen kanssa (lähde [1]. sivu 228) käyttäen URLConnection[38] luokan oliota jonka avulla appletti saa syötteenä kaiken servletin standardi tulostuksen.

Esimerkkinä AppletServletApplet.html.

---

[33] Ks. kappale ” Pavuista eli JavaBeaneista”.
[34] Ks. http://java.sun.com/products/jdbc/
[35] Ks. http://www.macromedia.com/
[36] Ks. http://java.sun.com/applets/
[37] Ks. http://www.borland.com
[38] Ks. http://java.sun.com/j2se/1.4.2/docs/api/java/net/URLConnection.html



Kuva 2. Appletti, joka kutsuu servlettiä ja näyttää haun tulokset

Esimerkissä tarvittavat tiedostot:

- Tiedosto tutor\AppletServletApplet.html, pääsivu jossa appletti ajetaan.
- Tiedosto tutor\ServletApplet.class, appletti kutsuu servlettiä ja nappaa servletin tulostuksen.
- Tiedosto tutor\WEB-INF\classes\AppletServlet.class servletti suorittaa kyselyn ja tulostaa tulokset standardivirtaan.

Koodeihin ei aloittelijan kannata vielä tässä vaiheessa tarkemmin perehtyä.

Jos et saa esimerkkiä toimimaan, koita käyttäen sivua AppletServletAppletConverted.html. Sivu on käsitelty HTMLConverter[39] ohjelmalla, jonka avulla varmistetaan Java PlugIn:n käyttö. Toinen testattava asia on korvata localhost koneesi IP osoitteella[40].

Toinen tapa, jolla palvelimelta voidaan välittää tietoa appletille on appletin parametrit. Palvelin generoi html-sivun, johon appletti ladataan. Samalla generoidaan sivulle vaihteleva määrä parametreja.

Mielestäni appletteja kannattaa käyttää vain jos tarvitaan todella näyttävää (jos html ja skriptit eivät riitä) tai korkeampaa logiikkaa vaativaa käyttöliittymää. Appleteissa on omat hyvät puolensa, mutta valitettavasti suurin haitta on selainten tuki ja toiminta. Eri selaimet saattavat näyttää appletin erilailla ja osassa selaimista appletti ei välttämättä toimi lainkaan. Muista aina testata ohjelmasi usealla selaimella.

---

[39] Ks. http://java.sun.com/j2se/1.4.2/docs/guide/plugin/developer_guide/html_converter.html
[40] Windowsissa IP osoitteesi saat selville komennolla ipconfig.

5.3.2 Käyttöliittymän ohjelmointi JavaScriptillä

Scriptikielillä kuten JavaScriptillä[41] on näppärää ja helppoa (ei tosin kovin ohjelmoinnillisesti eleganttia) suorittaa pieniä tarkastuksia ja käyttöliittymän viilauksia tapahtumien sekä funktioiden avulla. Huomattavaa on, että tarkastus pitää suorittaa myös palvelimella, sillä asiakkaalla ei välttämättä ole JavaScript päällä selaimessa.

Seuraavassa pieni esimerkki, jossa tarkastetaan seuraavat asiat ennen kuin lomakkeen tiedot lähetetään palvelimelle. Palvelimella servletti näyttää annetut tiedot. Asiakkaan selaimella tarkastetaan seuraavat tiedot JavaScriptiä käyttäen:

- Tarkastetaan, ettei yksikään kenttä ole tyhjä
- Nimi kentän pituus ei saa ylittää 10 merkkiä (todellisessa elämässä raja olisi ehkä 255)
- Syntymävuosi oltava kokonaisluku ja tietyllä välillä (tässä 1850-kuluva vuosi).

Mikäli jokin näistä ei pidä paikkaansa, käytetään selaimen alert-metodia näyttämään virheilmoitus käyttäjälle. Väärät kentät myös voitaisiin tyhjentää (tässä oltava erittäin varovainen. Jos käyttäjä on kirjoittanut esim. 256 merkkiä ja raja on 255, niin häntä saattaa ottaa rajusti pannuun jos ohjelma tyhjentää kentän ja käyttäjä joutuu näpyttelemään tekstinsä uudestaan), scriptissä kenttien tyhjennys kommentteina.

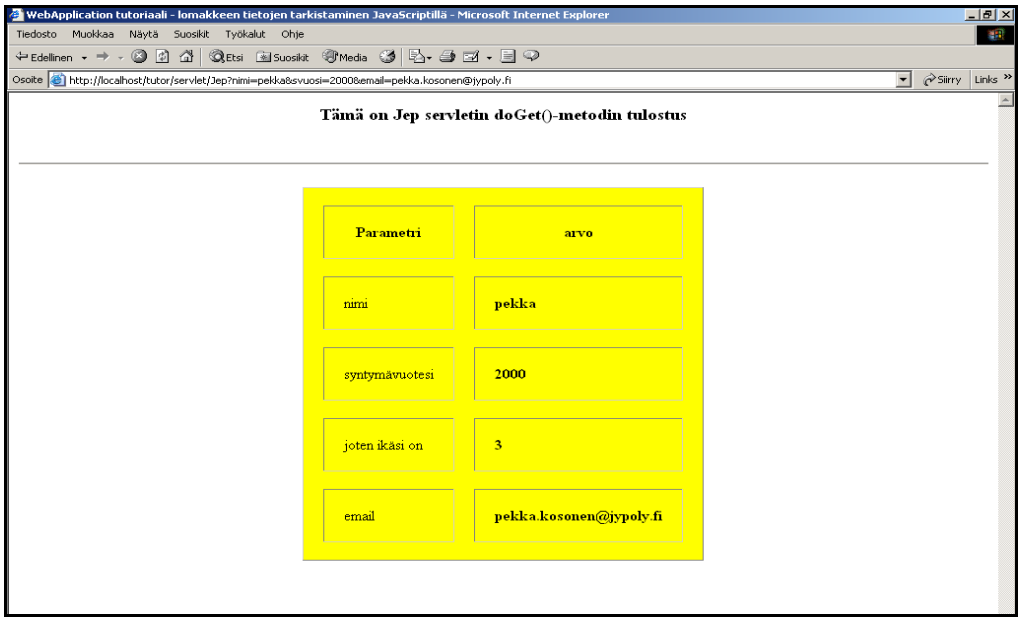
- Laita tarkistus.htm TOMCAT_HOME\webapps\tutor hakemistoon.
- Laita Jep.class (käännetty servletti) tutor\WEB-INF\classes hakemistoon.
- Lisää web.xml tiedostoon seuraavat rivit ennen </webapp> tagia.

```
<servlet>
  <servlet-name>Jep</servlet-name>
  <servlet-class>Jep</servlet-class>
</servlet>
```

Nyt testaa lomakkeen toimintaa osoitteessa tutor/tarkistus.htm. Testaa ensin kaikki väärät syötteet.

Painettuasi lähetä tiedot painiketta kutsutaan Jep servlettiä, joka tulostaa seuraavanlaisen HTML-sivun. Parametrit ja niiden arvot on laitettu taulukkoon (sivujen ulkonäöstä ei kannata välittää. Tästä ei ollut tarkoituskaan tulla käyttöliittymäopasta).

[41] Selaimen ymmärtämä ohjelmointikieli. Huomaa, että eri selaimet ymmärtävät JavaScriptiä varsin vaihtelevasti.



Kuva 3. Servletti tulostaa lomakkeen tiedot

5.3.3 Tietojen validointi palvelinpuolella

Lomakkeen tietojen tarkistaminen on toki mahdollista myös JSP:llä ja servleteillä. Jos tiedot eivät ole oikein, tulostetaan käyttäjälle lomake uudestaan. HTTP:n tilattomuudesta johtuen tässä on huonona puolena se, että käyttäjä saa eteensä tyhjän lomakkeen ja joutuu kirjoittamaan kaiken uudestaan. Jos käytät tällaista tekniikkaa, niin pyri siihen, että tulostaessasi lomakkeen käyttäjälle uudestaan täytettäväksi, laita käyttäjän aiemmat syöttötiedot suoraan lomakkeelle valmiiksi.

Yhteenveto ensimmäisestä osasta

Nyt kun olet saanut Tomcatin asennettua ja configuroitua, olet saanut toimintaan jsp-sivun sekä servletin, on aika pitää tauko. Seuraavassa osassa tutustutaan Tomcattiin tarkemmin.

Osan 1 koodit

Saat koko ensimmäisen osan websovelluksen toimintaan purkamalla tutor_osa1.zip paketin ja kopioimalla tutor hakemiston webapps\tutor hakemiston päälle. Paketti sisältää kokonaisuudessaan seuraavat tiedostot:

- Jep.class (käännetty servletti)
- Jep.java (servletin koodit)
- web.xml (tutor webapplicationin)
- server.xml (conf hakemistosta, Tomcatin configutointitiedosto)
- web.xml (conf hakemistosta, Tomcatin configutointitiedosto)
- ServletApplet.java
- AppletServlet.java
- initPara.java
- Hello.jsp

Lisäksi

- Tarkista.htm (web sivu joka kutsuu Jep servlettiä tarkastettuaan käyttäjän syötteen)
- AppletServletApplet.htm (lataa appletin joka kutsuu servlettiä)
- AppletServletAppletConverted.htm (edellisen HTMLConverter-ohjelmalla käsitelty versio)
- homersuklaa.gif, doh.gif

6 Osa 2 - Servletit

Web sovellusten ohjelmointi Javalla on kohtuullisen helppoa ja selkeää. Servletit tarjoavat helpon tavan kommunikoida web asiakkaiden[42] (web clients), tyypillisesti selainten, kanssa. Servlettien ohjelmoijan ei tarvitse huolehtia verkkoyhteyksien luomisesta, pyyntöjen (request) kiinniottamisesta, protokollien eri versioista eikä vastauksen (response) ohjaamisesta oikealle asiakkaalle. Lisäksi kuten Javassa on tapana, servlettien muistiin lataaminen ja muistin vapauttaminen eivät ole ohjelmoijan rasitteena. Nämä kaikki yksityiskohdat hoitaa servlettimoottori (servlet engine, servlet container) kuten Tomcat.

Seuraavat esimerkit on tehty Java Servlet Development Kit (JSDK) versiolla 2.3, mutta esimerkit toimivat myös aiemmilla JSDK versioilla (testasin esimerkit myös versiolla 2.1).

Servlettien testaamiseen on olemassa hyvä työkalu ServletRunner[43] (JSDK:ssa mukana) minkä kanssa et tarvitse erillistä servlettimoottoria.

6.1 Servlettien elinkaari

Tyypillisesti servlettien elinkaari kattaa seuraavat vaiheet [1].

1. Servletti ladataan muistiin joko servlettimoottorin käynnistyessä (vaatii määrityksen sovelluksen web.xml tiedostoon) tai kun servlettiä ensimmäistä kertaa kutsutaan asiakkaan toimesta (oletus).
2. Servlettimoottori kutsuu servletin init-metodia . Servlet API takaa, että metodi suoritetaan loppuun ennen kuin yhtään pyyntöä aletaan käsitellä. Init-metodissa hoidetaan yleensä servletille tyypilliset alustukset kuten connection poolingin[44] alustus.
3. Servlettimoottori luo asiakkaan pyynnössä välittyvästä datasta oliot joita käyttäen servletti kommunikoi asiakkaan kanssa (HttpServlet[45] luokan perivissä servleteissä (yleisin) luodaan HttpServletRequest[46]- ja HttpServletResponse[47]–rajapinnan toteuttavat oliot)
4. Servletti vastaa asiakkaan pyyntöihin ylikirjoittamillaan metodeilla (tyypillisesti doGet tai doPost)
5. Vaiheita 3 ja 4 toistetaan satunnaisesti aina kun servlettiä kutsutaan.

---

[42] Asiakkaalla tarkoitetaan ohjelmaa, joka kommunikoi palvelimen kanssa
[43] Ks. http://java.sun.com/developer/onlineTraining/Servlets/Fundamentals/magercises/ServletRunnerHosting/
[44] Yhteysallas, käytetään useasti esimerkiksi tietokantojen kanssa.
[45] Ks. http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/servlet/http/HttpServlet.html
[46] Ks. http://java.sun.com/j2ee/sdk_1.2.1/techdocs/api/javax/servlet/http/HttpServletRequest.html
[47] Ks. http://java.sun.com/j2ee/sdk_1.2.1/techdocs/api/javax/servlet/http/HttpServletResponse.html

6. Lopuksi servletti poistetaan muistista. Tällöin suoritetaan servletin destroy-metodi, jossa tyypillisesti esimerkiksi varmistetaan tietokantayhteyksien sulkeminen.

6.2 Säikeet

Oletetaan, että asiakkaan palveleminen kestää suhteellisen kauan, esim. doGet-metodin suoritus. Miten voidaan tällöin taata, että samanaikaisesti voidaan palvella muita asiakkaita? Kuten kappaleen otsikko vihjaa, vastaus on säikeet. Servletti on koodattava säieturvalliseksi (thread safe), sillä Servlet API ei sitä automaattisesti takaa. Tämä suoritetaan käytännössä viljelemällä synchronized termiä joko metodien otsikossa tai suoritettavien lauseiden yhteydessä.

Säieturvallinen koodi edellyttää luokan attribuuttien käytön suojaamista. Paikallisista muuttujista ei tarvitse olla huolissaan. Helppo mutta kustannustehoton tapa on synkronoida koko metodi seuraavasti

```
private synchronized void metodi ()
```

Tämä on hyväksyttävää lähinnä suoritukseltaan lyhyiden metodien yhteydessä tai jos kuitenkin lähes kaikki metodin koodi joudutaan synkronoimaan. Toinen tapa on synkronoida ainoastaan osia metodista seuraavasti.

```
private void metodi()
{
   synchronized ( this ) {
     //Nyt vain yksi säie voi suorittaa tämän koodilohkon yhtäaikaa
   }
   //koodia, joka ei ole synkronoitu
}
```

Voidaan myös käyttää SingleThreadModel-rajapintaa[48], jolloin taataan, ettei kaksi säiettä pääse yhtä aikaa samaan metodiin käsiksi, vaan odotetaan kunnes yksi asiakas on palveltu. Tässä on haittapuolena palvelun huomattava hidastuminen asiakasmäärän kasvaessa.

Lisää säikeistä ja synkronoinnista luettavissa esimerkiksi osoitteesta http://java.sun.com/docs/books/tutorial/essential/threads/

6.3 ServletContext ja alustusparametrien käyttö

Alustusparametreilla voidaan välittää servleteille tietoja, joita ne tarvitsevat käynnistyessään ja joita on helppo muokata. Esimerkkinä mainittakoon tietokantayhteyksissä tarvittavat tiedot.

[48] Ks. http://java.sun.com/j2ee/sdk_1.2.1/techdocs/api/javax/servlet/SingleThreadModel.html

Oletetaan, että haluaisit asettaa jokaiselle sovelluksesi servletille ja JSP:lle jonkin yhteisen tiedon, tavallaan globaalin muuttujan. Miten tämän voisi toteuttaa? Yksi vaihtoehto olisi käyttää ServletContext-oliota, ja ladata sinne servlettiä käyttäen tarvittava data.

6.3.1 Esimerkki

Esimerkissä katsotaan seuraavia asioita:

- ServletContexti-olion käyttäminen
- Servletin lataaminen palvelimen käynnistyksen yhteydessä

Esimerkissä käytetään kahta servlettiä. Ensimmäinen servletti LataaContextServlet.java on oleellinen, sillä se alustaa ServletContext-olioon tarvittavan tietorakenteen. Tärkeätä onkin, että tämä servletti ladataan muistiin Tomcatin käynnistyessä ja vieläpä ensimmäisenä. Tämä saadaan aikaan muokkaamalla web.xml tiedostoa seuraavasti[49]:

```
<servlet>
 <servlet-name>LataaContextServlet</servlet-name>
 <servlet-class>LataaContextServlet</servlet-class>
  <init-param>
   <param-name>titleBegin</param-name>
   <param-value>WebApplication tutoriaali - </param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
</servlet>
```

Esimerkin saat toimintaan asettamalla servletit WEB-INF/classes hakemistoon, muokkaamalla web.xml tiedostoa ja käynnistämällä Tomcatin uudelleen (tai pelkän tutor WEBAPPin käyttäen Tomcatin manager-apuohjelmaa). Uudelleenkäynnistys aiheuttaa sen, että servletit initialisoidaan. Itse asiassa servletit initalisoidaan joka kerta automaattisesti, kun olet muokannut web.xml tiedostoa, ja kutsut jotakin servlettiä/JSP sivua (Jos testaat tämän, muista sulkea kaikki selainikkunat ettei selaimen välimuisti sotke testauksiasi). Käynnistyksen jälkeen suunnista osoitteeseen:

http://localhost/tutor/servlet/LueContextServlet

Tällöin huomaat, että selaimen otsikkorivillä lukee web.xml tiedostoon määritelty parametri, jonka LataaContextServlet on asettanut ServletContext-olioon. LueContextServlet puolestaan lukee arvon init-metodissa.

---

[49] Huomaa erityisesti load-on-startup elementti

ServletContext-olion parametriolioiden asettaminen tapahtuu seuraavasti (ks. **LataaContextServlet.java**)

```
//luetaan alustusparametri, katso web.xml
String titleBegin = this.getServletConfig().getInitParameter("titleBegin");
System.out.println("titleBegin ==" + titleBegin);
 //asetetaan servletcontextiin kaikkien servlettien ja JSP-sivujen saataville
this.getServletContext().setAttribute("titleBegin", titleBegin);
```

ServletContext-oliosta ”lukeminen” eli parametriolioiden käyttöönotto tapahtuu seuraavasti (ks. **LueContextServlet.java ja LueContext.jsp**):

```
otsikonAlku = (String) this.getServletContext().getAttribute("titleBegin");
```

Tulostuksesta

Kaikki System.out virtaan kohdistettu tulostus on nähtävissä Tomcatin ikkunassa. Tämä on yksi keino tutkia servlettiesi suorituksia.

Tähän on kuitenkin yksi poikkeus: Servletin init-metodi. Itse varmistin, että servlettien init metodit todella tapahtuvat oikeaan aikaan luomalla molempien servlettien init-metodeissa tiedoston. Luotujen tiedostojen timestampistä selviää, milloin servlettien init-metodit tapahtuivat.

Init-metodista

Jos init metodissasi on virheitä, saat seuraavan virheilmoituksen:

```
javax.servlet.ServletException: Cannot allocate servlet instance for path
/tutor/servlet/LataaContextServlet
```

Kirjastoista

Kun laitat omia kirjastojasi WEB_INF\lib hakemistoon, laita tiedostot jar-pakettiin, vaikkei tiedostoja olisi kuin yksi. Muuten kirjastoasi ei oteta käyttöön, eivätkä servletit, joissa kirjastoja käytetään, toimi. Toki voit laittaa yksittäiset class-tiedostot WEB-INF\ classes hakemistoon

6.4 Sessiot eli istunnot

HTTP protokollan tilattomasta luonteesta johtuen sessioiden käytön hallinta on lähes pakollista kaikissa web sovellusten tekemiseen soveltuvissa kielissä, niin myös Javassa.

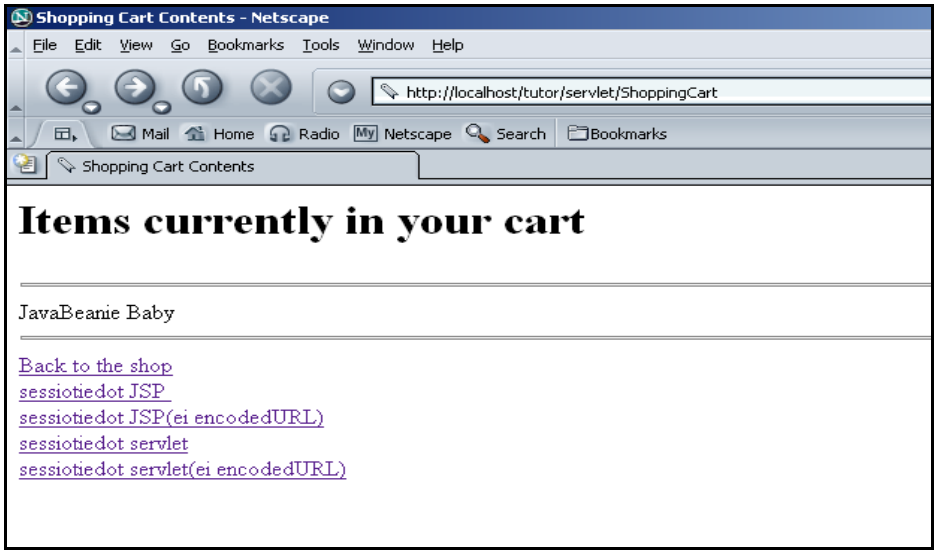
Seuraavassa esimerkissä demonstroidaan sessioiden käyttö sekä servleteissä että JSP sivuissa.

6.4.1 Ostoskori

Esimerkkinä on rakennettu ostoskori, jossa ostokset tallennetaan sessiotietoihin. Lisäksi esimerkki sisältää yhden servletin ja JSP sivun, jotka näyttävät session tietoja.

Aseta ennen esimerkin testaamista selaimestasi evästeet (cookies) pois päältä!

Osta yksi JavaBean sivulta http://localhost/tutor/ShoppingCart.html. Sen jälkeen sinulle näytetään korissasi olevat ostokset.



Kuva 4. Sessiotietojen käyttö ShoppingCart servletissä.

Testaa myös http://localhost/tutor/sessionTestaus.jsp, jossa on demonstroitu encodeURL metodin tärkeys myös JSP sivujen yhteydessä.

EncodeURL

Jos olet asettanut evästeet (cookies) pois päältä, niin linkit joissa lukee sulkeissa ”ei encodedURL” eivät toimi oikein. Seuraavassa selitys tähän.

Normaalisti sessiotietojen tallennukseen käytetään evästeitä. Mikäli kuitenkin evästeet on poistettu käytöstä, pitää session id numero tallentaa URL osoitteeseen (tai lomakkeen piilotettuihin kenttiin). Tämän vuoksi luodessasi sovelluksia, jotka käsittelevät sessioita, pitää kaikki osoitteet käsitellä erityisellä metodilla nimeltä encodeURL.

Eri asiakkaiden sessiotiedot erotellaan id-numerolla. Session id-numero pitää olla joko asiakkaan selaimessa evästeenä, URL-osoitteessa parametrina tai lomakkeen piilotettuna kenttänä. Id numeron avulla palvelin tietää, kuka asiakas on kyseessä.

Metodeilla encodeURL ja encodeRedirectURL (käyttö kun haluat ohjata pyyntöjä eteenpäin) saat sessio id:n tallennettua osoitteeseen. Esimerkissä encodeURL metodia on käytetty seuraavasti

```
String URL = "http://localhost/tutor/session.jsp";
String encodedURL = res.encodeURL( URL);
```

Sessio id:tä ei kuitenkaan tallenneta osoitteeseen kahdessa tapauksessa

- Session tietoja ei ole, sessiota ei ole olemassa
- Evästeiden käyttö on mahdollista

Session luominen ja käyttö

Session luominen ja käyttö tapahtuu käyttämällä HttpServletRequest[50] luokan metodia getSession.

```
// otetaan session objekti, luodaan jos ei ole olemassa
HttpSession session = req.getSession(true);
```

Mikäli parametrilla on arvo true, sessio luodaan ellei sitä ole jo olemassa. Jos sessio on olemassa, se ainoastaan otetaan käyttöön. Mikäli parametrin arvo on false ja sessiota ei olemassa, palauttaa metodi arvon null.

Session attribuutteja käsitellään seuraavilla metodeilla (JSP 1.1 spesifikaatiossa)

- **putValue** (String nimi, Object objekti) – asettaa uuden attribuutin nimen ja arvon
- **getValue** (String nimi) – palauttaa attribuutin arvon ja null jos attribuuttia ei ole

Uudemmissa spesifikaatioissa suositellaan käytettäväksi **setAttribute** ja **getAttribute** metodeja. Metodien käyttö on täsmälleen samanlaista kuin niiden ”esi-isien” (esimerkeissä on käytetty molempia).

Huomautus sessioista JSP sivuissa

JSP sivuihin sessioiden käyttö on ”sisään-rakennettu”, eli yhtenä JSP sivun implisiittisenä objektina on session. Älä siis luo sessiota äläkä ota käyttöön sessio objektia eksplisiittisesti, sillä objekti on olemassa automaattisesti.

Huomautus selaimista

Netscape Navigatorilla[51] sessioiden ja evästeiden testaaminen on huomattavasti helpompaa, koska evästeiden manageröinti on hoidettu paremmin kuin Internet Explorerissa. Lisäksi Netscape näyttää tilarivillä linkissä olevan sessioID:n, IE[52] puolestaan ei. Itse asi-

[50] Ks. http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/servlet/http/HttpServletRequest.html
[51] Ks. http://www.netscape.com
[52] Microsoft Internet Explorer

assa en itse saanut IE:tä poistamaan evästeiden käyttöä käyttäessäni localhostia[53], vaikka kuinka yritin. Kielsin selaimen asetuksista evästeiden käytön, tyhjensin temporary internet files kansion ja suljin selaimen uudestaan. Silti sessiotiedot kyettiin lukemaan vaikkei linkkiä ollutkaan käsitelty encodeURL metodilla. Toiselta koneelta ajettuna IE:kin suostui toimimaan. Outoako?

Netscapella katsottaessa ShoppingCart servletin tulostaman html-sivun lähdekoodi on seuraavanlainen.

```
<html><head><title>Shopping Cart Contents</title></head><body>
<h1>Items currently in your cart</h1>
<hr>JavaBeanie Baby<br><hr><a HREF="../ShoppingCart.html">Back to the shop</a><br>
<a HREF="../session.jsp;jsessionid=9F590CDBA347EB83CAD40EB31C530C2A">
sessiotiedot JSP </a>
<br>
<a HREF="../session.jsp">sessiotiedot JSP(ei encodedURL)</a>
<br>
<a
HREF="SessionInfoServlet;jsessionid=9F590CDBA347EB83CAD40EB31C530C2A">
sessiotiedot servlet</a>
<br>
<a HREF="SessionInfoServlet">sessiotiedot servlet(ei encodedURL)</a>
<br>
</body></html>
```

Kuten huomaat, session id-numero on liitetty linkin osoitteeseen. IE:llä vastaavaa ei tapahtunut localhostia käytettäessä, joten olettaa saattaa, että kaikesta kielloista huolimatta IE käytti evästeitä sessiotiedon tallentamiseen.

[53] Samalla koneella sijaitsevaa palvelinta voi kutsua URL:lla 127.0.0.1 tai nimellä localhost

7 Osa 3 - Java Server Pages

Tässä osassa tutoriaalia paneudutaan JSP sivujen toimintaan.

7.1 Yleistä

Ennen JSP-tekniikkaa pelkästään servlettejä käytettiin palvelinsovellusten ohjelmointiin. Dynaamisen käyttöliittymien muodostaminen osoittautui erittäin hankalaksi, sillä kaikki tulostettava HTML-koodi piti upottaa servlettiin. Käyttöliittymän ylläpito osoittautui hankalaksi. JSP oli tähän ongelmaan pelastava enkeli. JSP on tekniikkana verrattavissa ASP:hen[54] ja PHP:hen[55]

JSP sivut ovat puhtaita tekstidokumentteja. Selaimen pyytäessä JSP sivua JSP moottori muuttaa JSP koodit servletiksi, kääntää servletin ja upottaa servletin tulostuksen vastaukseen. JSP sivujen avulla on mahdollista prosessoida asiakkaan pyyntö (request) ja vastata siihen (response) aivan kuten servleteilläkin.

JSP sivujen avulla on helpompaa erotella staattinen ja dynaaminen sisältö sovelluksessasi kuin servleteillä. Syy tähän on selvä; servletit tulostavat HTML-sivun, kun taas JSP koodi upotetaan HTML:n joukkoon. Okei, tästä saattaa tulla sellainen kuva, että JSP sivut ovat yhtä sekamelskaa. Välillä HTML:ää ja välillä Java koodia. Oikein koodattuna lopputulos on kaukana tästä. Itse asiassa JSP ohjaa kirjoittajan mielestä paremmin olio-pohjaiseen ohjelmointiin kuin servletit (tästä esimerkkinä myöhemmin osassa käytetyt pavut eli JavaBeanit)! Servlettejä kehittäessä tulee helposti tehtyä lähes proseduraalista ohjelmointia sovelluslogiikan nojatessa doGet ja doPost metodeihin sekä muihin aliohjelmiin.

JSP sivuilla usein käytetty tekniikka on käyttää papuja[56] (JavaBeans). Näin toimiessa ohjelmoija luo luokan, jonka ilmentymää (siis oliota) JSP sivuilla käytetään. Tästä lisää kappaleessa JSP ja JavaBeans.

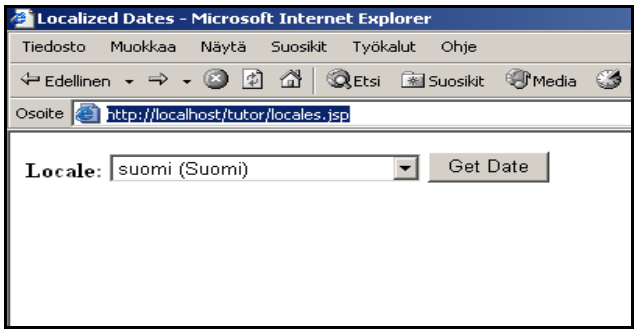
7.2 JSP sivu

JSP sivu on tekstipohjainen, ja sisältää kahdenlaisia osia:

- Staattista tietoa, kuten HTML, XML, WML jne.
- JSP elementtejä, joilla sivun dynaaminen sisältö luodaan.

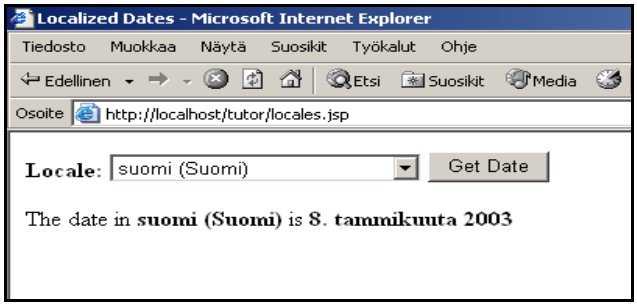
Esimerkissä (muokattu lähteestä [9]) luodaan JSP sivu ja kaksi papua. Sovelluksessa käyttäjä valitsee kielen (lokaalin) jonka mukaan nykyinen päivä näytetään.

[54] Active Server Pages
[55] Ks. http://www.php.net/
[56] Ks. Kappale ”Pavuista eli JavaBeaneistä”



Kuva 5. locales.jsp sivu

Kun olet klikannut get date - painiketta, sivu ladataan uudestaan ja siihen sisällytetään date.jsp sivu.



Kuva 6. locales.jsp sivun suoritus

Huomautus tiedostojen nimeämisestä

Ole varovainen, kun vaihdat kapitaaleja JSP tiedostojesi nimissä (myös Windowsin käyttäjät)!

Jos sinulla on esimerkiksi ollut tiedosto nimellä testisivu.jsp ja olet ajanut sen, ja myöhemmin nimeät sen uudelleen testiSivu.jsp, niin saatat joutua käymään poistamassa TOMCAT_HOME\work\standalone\localhost\tutor\testisivu_jsp.java[57] tiedoston ennen kuin saat uudelleen nimeämäsi JSP sivun toimintaan.

Edellä mainittu hakemisto sisältää JSP sivuista tuotettujen servlettien lähdekoodit! Kyseisiä koodeja voit käyttää hyväksesi esimerkiksi debugatessasi[58] JSP sivujasi.

Huomautus pavuista

Jos sinulla on käytössä Java versio 1.4 tai uudempi , niin kaikki pavut pitää olla paketissa (package paketin_nimi;)! Aiemmissa Java-versioissa papuja ei tarvitse paketoida (lisäinformaatiota http://www.thejspbook.com/faq/details.jsp?id=1007) .

---

[57] Tänne hakemistoon Tomcat kääntää JSP sivut servleteiksi.
[58] Debuggaaminen eli virheiden korjaaminen.

7.2.1 Esimerkki toimintaan

Esimerkin toimintaan laittamisen vaiheet:

1. Kopioi locales.jsp ja date.jsp webapps\turor hakemistoon
2. Kopioi pavut hakemisto (myös hakemisto, ei pelkät tiedostot webapps\turor\WEB-INF\classes hakemistoon)

7.2.2 Esimerkin koodit

Oletan, että olet tottunut HTML:n rakenteeseen ja syntaksiin, joten sitä en selitä. Seuraavassa lyhyet selitykset eri JSP elementeistä joita esimerkissä on käytetty. Tarkemmat selitykset löydät ensi kappaleesta. Katso tiedostojen

- Locales.jsp
- Date.jsp
- MyLocales.java (papu)
- MyDate.java (papu)

lähdekoodeja. Papujen koodit kannattaa ehkäpä tutkia vasta kappaleen ”pavuista” lukemisen jälkeen. Seuraavassa taulukossa on lyhyesti selitetty käytettyä JSP syntaksia.

Taulukko 2. JSP syntaksista

KOODI	KUVAUS
<%@ page import="java.util.*" %>	JSP:n page direktiivi, attribuuttina import. Otetaan tarvittavat kirjastot käyttöön.
<jsp:useBean id="locales" scope="application" class="pavut.MyLocales"/>	Jsp_useBean elementillä luodaan olio määritetystä luokasta.
<% … %>	Skripletti, eli java koodia, jossa on varsinainen sivun toiminta.
<%= … %>	Tämän elementillä (expression-elementti) sisällä olevien muuttujien arvot tulkitaan ja muunnetaan tarvittaessa merkkijonoiksi sekä näytetään sivulla.
<jsp:include … />	kutsutaan toista sivua ja sivun vastaus (response) liitetään tämän sivun vastaukseen.

7.3 Implisiittiset objektit

Jokaisella JSP sivulla on muutama implisiittinen objekti valmiina käytettäväksi. Objektit näet katsomalla JSP sivusta tuotetun servletin lähdekoodia (käännettyjen servlettien koodit löytyvät hakemistosta TOMCAT_HOME\work\Standalone\localhost\tutor).

```
javax.servlet.jsp.PageContext pageContext = null;
HttpSession session = null;
ServletContext application = null;
ServletConfig config = null;
JspWriter out = null;
Object page = this;
```

Lisäksi implisiittisiä objekteja ovat

- request (välitetään parametrina)
- response (myös parametrina)
- exception

7.4 jspInit ja jspDestroy

Aivan kuten servleteissäkin, on JSP sivulle mahdollista määritellä init ja destroy metodit (ks. sendEmailGroup.jsp). Tämä tapahtuu määrittelemällä metodit

- **jspInit (vastaa servletin itit metodia)**
- **jspDestroy (vastaa servletin destroy metodia)**

Metodien toiminta on aivan vastaavaa kuin servleteillä. JspInit() metodi tapahtuu, kun JSP sivua pyydetään ensi kertaa. JspDestroy metodi puolestaan tapahtuu, kun sivua vastaava servletti vapautetaan muistista. Tähän voi olla syynä joko JSP moottorin sulkeminen tai että sivua ei ole tarvittu vähään aikaan jolloin JSP moottori saattaa vapauttaa muistit tehokkuussyistä.

7.5 Pavuista eli JavaBeaneista

Pavut ovat aivan tavallisia Javalla kirjoitettuja luokkia, mutta papujen anatomian tulisi toteuttaa seuraavat kolme sääntöä[59]:

- Pavulla tulee olla parametriton konstruktori!
- Luokalla ei saisi olla julkisia (public) muuttujia.
- Jokaiselle attribuutille tulisi toteuttaa get- ja set –metodit.

[59] Tarkemmin http://java.sun.com/products/javabeans/docs/spec.html

Seuraavassa esimerkissä on toteutettu muutama papu.

7.5.1 Esimerkin toimintaan laittaminen:

Esimerkin toimintaan laittamiseksi pitää suorittaa seuraavat vaiheet:

1. Kopioi koko tutor hakemisto vanhan hakemiston päälle (webapps\tutor).
2. Lataa WEBAPP uudestaan http://localhost/manager/html/reload?path=/tutor
3. Surffaa selaimella osoitteeseen http://localhost/tutor/papuTesti.jsp .

Kun menet sivulle papuTesti.jsp ensimmäistä kertaa, näet tekstin: papu luotu. Nyt kun käyt jossakin toisella sivulla tai suljet selaimen niin huomaat, että papu luodaan uudestaan. Tämä johtuu pavun scope-ominaisuudesta.

Muuta paputesti.jsp tiedostosta teksti

```
<jsp:useBean id="papu" class="pavut.LukuPapu" scope="page"/>
```

seuraavanlaiseksi

```
<jsp:useBean id="papu" class="pavut.LukuPapu" scope="session"/>
```

Muuta pavun arvoa ja sulje selain (pavun alkuarvona on 1, ks Papu.java). Palaa sivulle takaisin. Huomaat, kuinka papu on säilyttänyt antamasi arvon. Kun muutat pavun arvoa uudestaan, uusi arvo lisätään vanhaan ja tulos näytetään.

Huomautus 1

Älä ikinä käytä skripleteissä luomiasi olioita <jsp:useBean .. /> tagissa. Skripleteissä luodut oliot eivät välttämättä (lue:todennäköisesti) ole saatavilla pageContextista useimmissa JSP moottoreissa.

Huomautus 2

Pyri aina käyttämään <jsp:setProperty … /> ja <jsp:getProperty … /> tageja vaikka voitkin kutsua olion get ja set-metodeja suoraan.

Huomautus 3

JSP sivusta käännetyt käännetyt servletit löydät lähdekoodeineen TOMCAT_HOME\work\Standalone\localhost\tutor hakemistosta.

7.5.2 Pavun elinaika ja näkyvyys eli scope

Voit asettaa pavun scope ominaisuuteen yhden seuraavista arvoista

1. page (oletus, jos muuta ei määrätä niin tätä käytetään)
2. request

3. session (papu pidetään ”hengissä” niin kauan kuin istunto kestää (riippuu session asetuksista, sessio on hengissä tietyn ennaltamäärätyn ajan)
4. application (papu poistetaan muistista kun serveri (Tomcat) ajetaan alas)

Jossain tapauksissa voit haluta poistaa pavun ennen sen ”luonnollista kuolemaa”. Tämä parantaa palvelimesi suorituskykyä sekä lisää turvallisuutta. Esimerkiksi voisit haluta poistaa käyttäjän loggautumistiedot käyttäjän loggautuessa ulos. Tämän teet seuraavasti pavun nimeä käyttäen. Pavun poistaminen on riippuvainen pavun scopen asetuksesta kuten seuraavasta taulukosta ilmenee:

Taulukko 3. Papujen käyttö

Pavun scope	Poisto skripletissä	Poisto servletissä
session	session.removeAttribute(name)	HttpSession. removeAttribute(name)
request/page	pageContext.removeAttribute(name)	ServletRequest. removeAttribute(name)
Application	application.removeAttribute(name)	ServletContext. removeAttribute(name)

7.6 Kuinka käsitellä ongelmatilanteet JSP-sivuilla

Palataanpa äskeiseen papuesimerkkiin. Syötä pavulle uudeksi arvoksi ”iso”. Tomcat palvelin generoi virheilmoituksen. Mistä tämä johtuu? Katsotaan pavun koodia (**LukuPapu.jsp**), jota JSP sivu kutsuu.

```
public int muutaLukua( String Luku )
{
 if( Luku != null && Luku != "" )
  return muutaLukua( Integer.parseInt(Luku) ); //Integer.parseInt aiheuttaa poikkeuksen
  return luku; //jos ehto epätosi
}
```

Mitä sitten voimme tehdä tämän estämiseksi? Seuraavassa on esitelty kolme vaihtoehtoa.

1. Lisätään poikkeuksen käsittely papuun (katso **ErrorPapu.java**, testaa **errorPapuTesti.jsp**)

```
public int muutaLukua( String Luku )
```

```
  {
    int muutos = 0;
    try
    {
      muutos = Integer.parseInt(Luku);
      this.muutaLukua(muutos);
    }catch ( NumberFormatException e ) //poikkeuksen käsittely (error
handling)
    { //merkkijonoa ei voitu muuntaa kokonaisluvuksi
      virheLippu = true;
    }
    finally {
      return luku;
    }
  }
```

2. Tarkastetaan ennen kutsua ettei käyttäjä ole antanut vääränlaista informaatiota **(papuTestiTarkastus.jsp**) esimerkiksi seuraavasti (huonoin vaihtoehto, samat koodit joutuu lisäämään jokaiseen papua käyttäjään sivuun)

```
<%
  String arvo = "";
  int muutos = 0;
  try {
    arvo = request.getParameter("arvo");


    if( arvo == null )
      out.print("papu luotu");
    else
    {
      muutos = Integer.parseInt( arvo );
      out.print("<h3>Pavun arvo nyt " + papu.muutaLukua( muutos ) );
    }
```

```
  }
  catch (NumberFormatException e ) //käsittellään NumberFormatException
  {
    out.print("Anna kokonaislukuarvo.<BR>");
    out.print( "(" + e.toString() + ")"); //tulostetaan virheilmoitus suluissa
  }
  catch (Exception ee ) //muut poikkeukset
  {
    out.print("Tapahtui odottamaton poikkeus" + ee.toString());
  }
%>
```

3. Luodaan poikkeuksia varten oma JSP sivunsa (errorPage). Katso esimerkki **aiheutaPoikkeus.jsp**. Esimerkissä jsp sivu suorittaa nollalla jakamisen, jolloinka kontrolli siirtyy sivun alussa määriteltyyn virhesivuun (**poikkeusSivu.jsp**).

Kolmannessa tekniikassa (eli errorPages) käytössä on huomattava, että minkäänlaista tulostusta ei saisi olla syntynyt (eli puskurointi on pois päältä tai tulostuspuskuri on täyttynyt) ennen virheen syntymistä. Jos näin on, tämä tulostus näkyy virhesivulla. Katso esimerkki (**aiheutaPoikkeusEiToimi.jsp**). Esimerkissä tulostuspuskuri täyttyy mikä puolestaan johtaa puskurin tyhjenemiseen, jolloin tulostuksemme ehtii tapahtua ennen kontrollin siirtymistä (forward) virhesivulle. Huomaa, ettei koko silmukan tulostus ehdi tapahtua. Mieti miksi!

JSP sivusta tehdään poikkeussivu seuraavalla määrittelyllä (ks. **poikkeusSivu.jsp**)

```
<%@ page isErrorPage="true" %>
```

7.7 Pieniä esimerkkejä

Seuraavaan olen koonnut muutamia pieniä esimerkkejä, jotka saattavat olla sinulle hyödyksi.

1. Milloin JSP sivua on muokattu viimeksi (ks. pavut.MyDate.java ja lastModified.jsp)

   koodeissa lisäksi päivämäärä ja aika formatoitu Suomen mallisiksi

```
<%
  File f = new File(application.getRealPath( request.getServletPath() ));
  Date modified = new Date( f.lastModified() );
%>
<jsp:useBean id="date" class="pavut.MyDate" type="MyDate" />

<HTML>
<BODY>
Sivua on viimeksi muokattu
<%
  out.print( date.toFinnishDateTime(modified) );
%>
```

2. JSP moottorin käyttämä JSP versio (JSPversio.jsp)

```
<%@ page import="javax.servlet.jsp.JspFactory" %>
<% JspFactory factory = JspFactory.getDefaultFactory(); %>

<body>
<p>
Käytössä on JSP versio
<%= factory.getEngineInfo().getSpecificationVersion() %>
</body>
```

3. Järjestelmäkomentojen suorittaminen. Esimerkissä ajetaan C:\temp\winver.bat tiedosto, jonka tulostus näkyy JSP sivulla (sysCommand.jsp). Kommenteista näet, miten saat selville Unix palvelimen uptimen. Erittäin näppärä tapa lisätä toiminnallisuutta sovelluksiin. JNI:n (Java Native Interface) avulla saat suoritettua myös natiivikoodia esim. DLL:stä

```
<%@ page import="java.io.*" %>

<%!
public String run(String cmd) {
```

```
 try {
  Runtime rt = Runtime.getRuntime();
  Process p = rt.exec(cmd);
  InputStreamReader in = new InputStreamReader(p.getInputStream());
  BufferedReader reader = new BufferedReader(in);
  StringBuffer buf = new StringBuffer();
  String line;
  String newline = "\n";
  while ((line = reader.readLine()) != null) {
   buf.append(line);
   buf.append(newline);
  }
  reader.close();
  p.getInputStream().close();
  p.getOutputStream().close();
  p.getErrorStream().close();
  p.waitFor();
  return buf.toString();
 }
 catch (Exception e) {
  return (e.getMessage());
 }
}
%>
<html>
<head>
 <title>Webapplication tutoriaali - järjestelmäkomentojen suorittaminen</title>
</head>
<body>
<%-- Unix ollut pystyssä <%= run("/usr/bin/uptime") %> --%>
```

```
Ajetaan c:\temp\winver.bat tiedosto, joka sisältää ainoastaan komennon: <BR>
ver <BR>
Tulos näkyy alla <BR>
<%= run("c:\\temp\\aja.bat") %>
</body>
</html>
```

4. Bean, jolla saat selville käytettävissä olevan selaimen (ks. Selain.java, selain.jsp)
5. JSP sivun lähdekoodin näyttäminen

   (eihän tässä muuten mitään ongelmaa ole, mutta kun Tomcat suorittaa kaikki JSP tiedostot niiden koodisisällön näyttämisen sijaan)

   - viewSource.jsp (kutsu esim.

     http://localhost/tutor/viewSource.jsp?url=JSPversio.jsp)

   - vsButton.jsp. Incluudaamalla tämän JSP sivun seuraavasti

     ```
     <jsp:include page="vsbutton.jsp" flush="true" />
     ```

     saat sivullesi painikkeen, jonka avulla sivun koodi näytetään uudessa ik-kunassa (apuna käytetty JavaScriptiä)

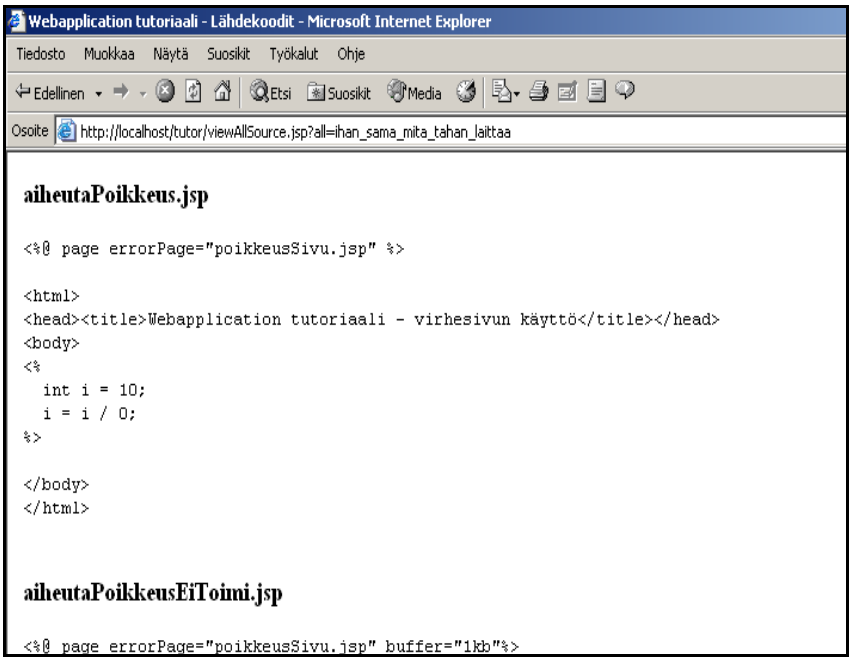
   - viewAllSource.jsp. ViewAllSource tulostaa linkkilistan hakemistossa olevista jsp sivuista. Linkkiä klikkaamalla kutsutaan viewSource.jsp tie-dostoa, jonka tulostus näytetään. (Antamalla parametrin all listataan kaikkien JSP tiedostojen koodit allekkain. Tätä käyttämällä listasin itse tutoriaalin loppuun kaikki JSP koodit. Lisäksi antamalla parametrille ha-ke arvon, listataan kyseisessä hakemistossa olevat tiedostot, myös muut kuin JSP tiedostot)

     Listaa kaikki JSP koodit:

     http://localhost/tutor/viewAllSource.jsp?all=ihan_sama_mita_arvoksi

     Listaa kaikki tiedostot tutor/src hakemistosta

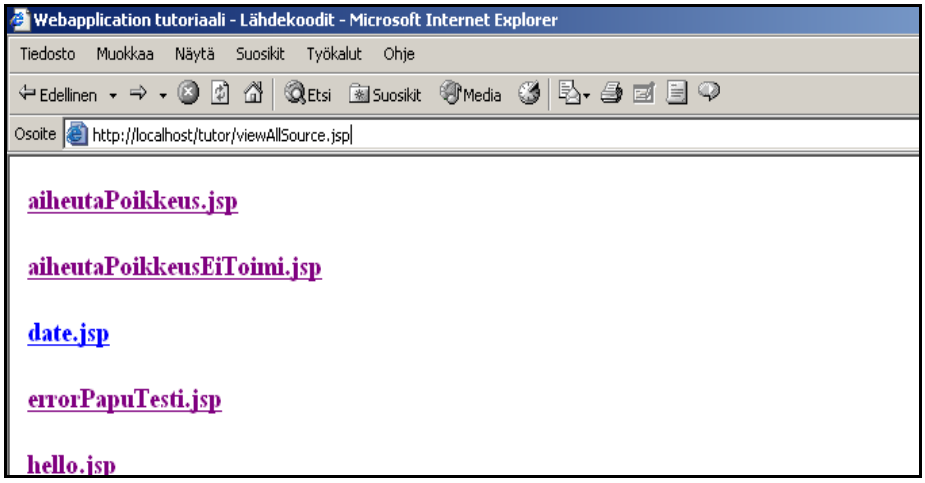
     ```
     http://localhost/tutor/viewAllSource.jsp?all=1&hake=src
     ```



Kuva 7. aiheutaPoikkeus.jsp lähdekoodit viewSource.jsp sivulla katsottuna

Listan tulostus

http://localhost/tutor/viewAllSource.jsp



Kuva 8. viewAllSource.jsp sivun suoritus

Seuraavassa koodien kiinnostavat osat kommentoituna:

ViewAllSource.jsp

```
<%@ page import="java.io.*" %>
<%
File path = new
File(application.getRealPath("viewAllsource.jsp")).getParentFile();
%> //Haetaan viewAllsource.jsp tiedoston absoluuttisen polun ha-
                kemisto
    //eli esim c:\programs\…\tomcat\webapps\tutor
…
<%
 File [] files = path.listFiles(); //Tiedostolistaus tästä hakemistosta
 if( files != null )
 for( int i = 0; i < files.length; i++ ) {
  // tiedostot käydään läpi silmukalla, jos ,ei jsp tiedosto niin continue
  if(   tarkennin( files[i].getName() ).compareToIgnoreCase(".jsp") != 0)
    continue;
%>
…
<%
 if( request.getParameter("all") == null )
 {  // jos all parametria ei ole annettu, niin listataan tiedostojen nimet
  out.print("<a href=/tutor/viewSource.jsp?url=");
  out.print(files[i].getName() + ">");
  out.print("<h3>"+files[i].getName()+"</h3></a>");
 }
 else
 { //muutoin tulostetaan tiedoston nimi
  out.println("<h3>" + files[i].getName() + "</h3>");
  //out.print("TEsti");
```

```
%> //ja incluudataan viewSource.jsp:n tulostus eli tiedoston sisältö
 <jsp:include page="/viewSource.jsp">
  <jsp:param name="url" value="<%=files[i].getName()%>"/>
 </jsp:include>
<%
 }
}
%>
…
<%! //declaration tagi, aliohjelma palauttaa tarkentimen
 public String tarkennin(String file)
 {
  int i = file.lastIndexOf(".");
  if( i == -1)
   return "";
  return file.substring(i, file.length() );
 }
%>
```

ViewSource.jsp

String url = request.getParameter("url");**//url==tiedoston nimi**

```
if (url.indexOf("..") > -1)
 throw new java.io.IOException("Relative paths are not allowed");
File realPath = new File(application.getRealPath(url));
%>
…
<%
FileInputStream fis = null;
```

```
try {
  BufferedReader reader;
  reader = new BufferedReader(new InputStreamReader(fis));
  String line; //luetaan tiedostoa rivi kerrallaan
  while ((line = reader.readLine()) != null) {
    line = replace(line, "&", "&amp;"); //html muotoon
    line = replace(line, "<", "&lt;");
    line = replace(line, ">", "&gt;");
    out.println(line);
  }
}
catch (IOException e) {
  out.println("IOException: " + e.getMessage());
}
finally { if (fis != null) fis.close(); }
%>
</pre></body></html>
<%!//aliohjelmalla korvataan old osat replacement osalla
public String replace(String s, String old, String replacement) {
   int i = s.indexOf(old);
   StringBuffer r = new StringBuffer();
   if (i == -1) return s;
   r.append(s.substring(0,i) + replacement);
   if (i + old.length() < s.length())
     r.append(replace(s.substring(i + old.length(), s.length()),
      old, replacement));
   return r.toString();
} %>
```

8 JSP tagit

JSP spesifikaatiosta 1.1[60] lähtien ohjelmoijilla on ollut mahdollisuus tehdä omia JSP-tageja. Ohjelmoija määrittelee JSP uuden tagin sekä sen attribuutit, ja kaikki tagit kerätään yhteen muodostaen JSP-tagi kirjaston.

8.1.1 Miksi omia tageja?

Perusidea omien tagien luomisessa on sama kuin jsp:useBean direktiivissä. Muodoltaan yksinkertaisella ja selkeällä komennolla saadaan piilotettua monimutkaisuus JSP sivulta.

Pavut vs. tagit

- Pavuilla ei ole mahdollista muokata JSP sivun sisältöä.
- Monimutkaiset operaatiot saadaan paremmin yksinkertaistettua omilla tageilla kuin pavuilla.
- Omien tagien luominen on hieman tuskallisempaa kuin papujen.
- Pavut ovat käytössä jo JSP spesifikaatiossa 1.0.
- Mahdollisuus tehdä omia tageja on ollut olemassa JSP spesifikaatiosta 1.1.
- Omat tagit vaativat Tomcat version 3.1 tai uudemman.

8.1.2 Omien tagien luominen

Käyttääksesi omia tageja ohjelmoijan tarvitsee luoda kolme komponenttia

1. Tagin käsittelijä luokka, joka määrää tagin käyttäytymisen.
2. Tagi kirjaston kuvaamistiedosto (descriptor file), liittää yhteen XML elementit tagin toteutukseen.
3. JSP sivu, jossa tagia käytetään.

Tagin käsittelijä luokka

Ensimmäinen tehtävä luodessasi uutta tagia on luoda luokka, joka kertoo järjestelmälle mitä tehdään kun tällaiseen tagiin törmätään. Esimerkissä määritelty tagi korvataan tekstillä

```
<title>
Webapplication tutoriaali - selainversion testaaminen
</title>
```

kaikkien tagin esiintymien kohdalla.

[60] Ks. http://java.sun.com/products/jsp/download.html

TitleTag luokka

```
package tagit;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;

public class TitleTag extends TagSupport {
 public int doStartTag() {
  try {
   JspWriter out = pageContext.getOut();
   out.print("<title> Webapplication tutoriaali</title>");
  } catch(IOException ioe) {
   System.out.println("Error in ExampleTag: " + ioe);
  }
  return(SKIP_BODY);
 }
}
```

Yksinkertaiset tagit käsitellään yleensä luokalla, joka perii TagSupport[61] luokan. Jos tagilla ei ole attribuutteja eikä varsinaista sisältöä, riittää doStartTag-metodin ylikirjoittaminen (override). Tällöin kyseisen metodin pitäisi lisäksi palauttaa SKIP_BODY vakio. Näin jätetään huomioimatta mahdollinen tagin sisältö. Metodissa tulostetaan JSPWriter[62] luokan oliolla.

Käännetty luokka sijoitetaan Tomcatissä WEB-INF\classes\tagit hakemistoon.

Tagikirjaston kuvaamistiedosto (Tag Library Descriptor)

Tehtyäsi tagin käsittelijä luokan, seuraava tehtäväsi on luoda XML-tiedosto, jossa luokka yhdistetään määrättyihin tageihin. Tätä koodia voit käyttää suoraan omissa sovelluksissasi. Ainoastaan lihavoidut kohdat tarvitsevat muokkausta. Tiedostossa tutor_taglib.tld on kuvattuna ainoastaan yksi tagi. Tomcatissä tiedosto sijoitetaan WEB-INF\lib hakemistoon.

---

[61] Ks. http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/servlet/jsp/tagext/TagSupport.html
[62] Ks. http://jakarta.apache.org/tomcat/tomcat-5.0-doc/jspapi/javax/servlet/jsp/JspWriter.html

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
 PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
 "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<taglib>
 <tlibversion>1.0</tlibversion>
 <jspversion>1.1</jspversion>
 <shortname>tutor</shortname>
 <uri>/tutor</uri>
 <info>
    Tagi kirjasto webApplication tutoriaaliin
 </info>

 <tag>
  <name>title</name>
  <tagclass>tagit.TitleTag</tagclass>
  <info>tagi korvataan puhtaalla tekstillä</info>
  <bodycontent>EMPTY</bodycontent>
 </tag>
</taglib>
```

Tagikirjaston kuvaamistiedosto on muuttunut huomattavasti JSP spesifikaatioiden 1.1 ja 1.2 välillä. Erot huomaat vertailemalla vastaavia DTD-dokumentteja:

http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd

http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd

Tagikirjaston kuvaaminen web.xml sivulla

Tomcatissä kaikki tagikirjastot kuvataan sovelluksesi web.xml tiedostossa taglib element-tiä käyttäen seuraavasti:

```
<taglib>
  <taglib-uri>/tutor</taglib-uri>
  <taglib-location>/WEB-INF/lib/tutor_taglib.tld</taglib-location>
</taglib>
```

Esimerkki otettu tutor/web.xml tiedostosta.

Tagin käyttö JSP sivulla

Kun olet tehnyt tagin käsittelyyn tarvittavan luokan ja tagikirjaston kuvaavan tld-tiedoston, voit käyttää tagiasi JSP sivulla. Ennen ensimmäistä tagia kirjasto pitää ottaa käyttöön seuraavasti

```
<%@ taglib uri="tutor_taglib.tld" prefix="tutor" %>
```

Tagin käyttöä demonstroidaan tagTest.jsp sivulla

```
<%@ taglib uri="/tutor" prefix="tutor" %>
<html>
<head>
  <tutor:title />
</head>
<body>
  <h1> Tagia käytetty sivun otsikon luomiseen </h1>
</body>
</html>
```

Attribuuttien lisääminen tagiin

Edellisen esimerkki toimii kuten symbolinen vakio. Ennalta määrätty teksti korvataan uudella arvolla. Lisätäksemme toiminnallisuutta "makro-tasolle" pitää tagimme pystyä käsittelemään attribuutteja.

Attribuutin käyttö JSP sivulla aiheuttaa vastaavan set_attribuutin_nimi kutsun. Tagin käsittelijä-luokkaan lisätään jokaiselle attribuutille vastaavan niminen attribuutti sekä set-metodi[63] (lisäksi on hyvä lisätä get-metodit[64], mikäli muut luokat käyttävät tagin käsittelijää).

Tagin attribuutit pitää myös kuvata tld-tiedostossa seuraavasti:

[63] Metodi, joka asettaa attribuutille arvon, esim. setNimi(String uusiArvo).
[64] get-metodi palauttaa vastaavan attribuutin arvon, esim. getNimi()-metodi.

```
<attribute>
  <name><!-- attribuutin nimi --></name>
  <required><!-- true/false. Onko attribuutti pakollinen--> <required>
  <rtexprvalue><!-- true/false. Voidaanko attribuuttia käyttää JSP expressi-
on tagissa --></rtexprvalue>
</attribute>
```

<rtexprvalue> elementti ei ole pakollinen (oletus false).

Attribuutin käyttö JSP sivulla on suoraviivaista

```
<tutor:title attribuutin_nimi=arvo />
```

Attribuutit asetetaan ennen doStartTag metodin suorittamista, joten attribuutit ovat käytettävissä metodissa.

Esimerkin vuoksi muokataan edellisiä tiedostoja s.e sivun otsikkoon on mahdollista lisätä tarkennusta tagia käytettäessä. Annetaan attribuutille nimeksi titleEnd. Tarvittavat muutokset ovat:

Tagin käsittelijä luokkaan:

- Lisätään attribuutti titleEnd
- Lisätään setTitleEnd-metodi

```
public void setTitleEnd(String jono)
{
  titleEnd = jono;
}
```

- Muokataan doStartTag-metodia

```
    out.print("<title> Webapplication tutoriaali ");

    if( titleEnd != null)
      out.print("- " + titleEnd);

    out.print("</title>");
```

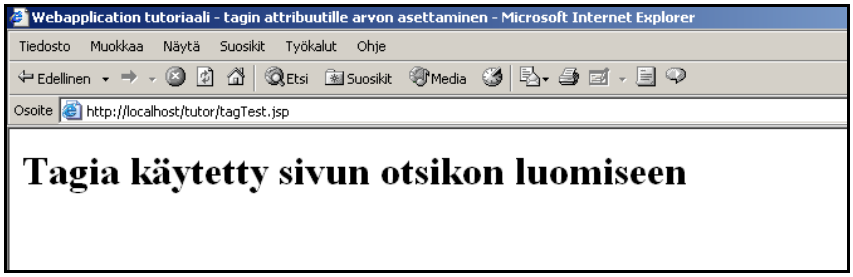
- Tagin kuvaus tiedostoon lisätään attribuutin nimi titleEnd ja required arvoon false

```
...
<attribute>
  <name>titleEnd</name>
  <required>false</required>
 </attribute>
</tag>
```

- Muokataan JSP sivua asettamaan attribuutille arvo (huomaa lainausmerkkien käyttö)

```
<tutor:title titleEnd="tagin attribuutille arvon asettaminen"/>
```

Lopputulos tagTest.jsp



Kuva 9. tagTEst.jsp. Oman tagin testaaminen. Huomaa selaimen otsikko (title)

Tagin rungon lisääminen

Seuraavaksi lisätään tagiin runko (body). Tagi kokonaisuudessaan on seuraavaa muotoa

```
<prefix:tagname attribute1="value1" attribute2="value2" > body
</prefix:tagname>
```

Ainoa muutos bodyn sisällyttämiseksi tagin käsittelijään on muuttaa doStartTag() metodin paluuarvo olemaan EVAL_BODY_INCLUDE. Lisäksi tagin kuvaustiedostoon pitää muuttaa bodycontent elementti saamaan arvo JSP.

Yleensä rungon käsittelyn jälkeen halutaan tehdä jokin toimenpide. Tämä onnistuu ylikirjoittamalla doEndTag metodi. Ellei haluta lopettaa sivun suorittamista tagin käsittelyn

jälkeen, doEndTag-metodin pitää palauttaa EVAL_PAGE. Mikäli halutaan lopettaa sivun suoritus jostain syystä, palauta SKIP_PAGE.

Huomaa, että tagin runko voi sisältää myös JSP skriptejä ja direktiivejä!

8.1.3 Parametrien lukeminen tagin käsittelijässä

Koska tagin käsittelijäluokan metodit eivät saa parametreina HttpServletRequest eikä HttpServletResponse olioita, on pyynnön parametrien lukeminen hieman monimutkaisempaa. Seuraavassa luetaan parametrin useBody arvo, jonka mukaan päätetään otetaanko tagin runko mukaan tulostukseen vai ei.

```
public int doStartTag() {
  ServletRequest request = pageContext.getRequest();
  String value = request.getParameter("useBody");
  If( value != null && value.equalsIgnoreCase("true") )
    return (EVAL_BODY_INCLUDE);
  return (SKIP_BODY);
}
```

Tagin rungon käsitteleminen

TagSupport luokan periminen on riittänyt hyvin tähän asti käsitellyissä yksinkertaisissa tageissa. Mikäli kuitenkin tagin runkoa pitää muokata, on tagin käsittelijäluokan perittävä BodyTagSupport[65] luokka (joka puolestaan perii TagSupport[66] luokan).

BodyTagSupport luokassa määritellyt tärkeimmät metodit ovat:

- getBodyContent, joka palauttaa BodyContent tyyppisen olion. Olio sisältää tagin rungon
- doAfterBody, jossa tagin runkoa voidaan manipuloida. Yleensä metodi palauttaa SKIP_BODY arvon, jottei enempää tagin rungon käsittelyä tapahdu

BodyContent[67] luokassa on kolme oleellista metodia

- **getEnclosingWriter,** palauttaa JspWriter olion jota voidaan käyttää doStartTag- ja doEndTag- metodeissa.
- **getReader**, palauttaa Reader luokan olion jolla tagin runko voidaan lukea

[65] Ks. http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/jsp/tagext/BodyTagSupport.html
[66] Ks. http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/jsp/tagext/TagSupport.html
[67] Ks. http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/jsp/tagext/BodyContent.html

- **getString**, palauttaa tagin koko rungon

Tagin rungon käsitteleminen moneen kertaan

Mikäli haluat käsitellä tagin rungon useasti, pitää doAfterBody metodin palauttaa EVAL_BODY_TAG arvo. Halutessasi lopettaa toiston palauta arvo SKIP_BODY.

9 Case

Tarkastellaan kolmea eri tapausta:

- Sähköpostiryhmät
- Chat
- ”Hitaasti valmistuva” servletti

9.1 Case 1 – Sähköpostiryhmät

Seuraavassa rakennetaan servlettejä käyttäen järjestelmä, jonka tarkoituksena on toimia sähköpostiryhmänä. Esimerkistä käsitellään seuraavia asioita:

- Sähköpostin lähetys (SendMail.java)
- Servlettien välinen kommunikointi (RequestDispatcher)
- Tietokannan käyttö

9.1.1 Sähköpostin lähetys (SendMail.java, sendMail.html)

Servletti, joka suorittaa sähköpostin lähetykset on melko suoraviivainen. Katso lähdekoodit tarkempien tietojen saamiseksi.

Lähettämisen saat toimintaan kopioimalla SendMail hakemiston classes hakemiston alle ja laittamalla sendMail.html tiedoston webapps\tutor hakemistoon. Tomcatissä JavaMail API on valmiina (TOMCAT_HOME\ common\lib\mail.jar). Saatat joutua asettamaan SMTP palvelimen osoitteen ja kääntämään koodit uudestaan. Testaa lähettämällä itsellesi posti

http://localhost/tutor/SendMail.htm

Vinkki JBuilderin käyttäjille

Jos haluat käyttää JavaMail APIA, niin muista lisätä kirjasto (mail.jar paketissa) projektiin (Project->Project Properties->Paths välilehti->Required Libraries->Add)

9.1.2 Sähköpostin lähettäminen ennalta määrätylle ryhmälle (SendEmailGroup.java)

Sähköpostin lähetys ryhmille onkin jo paljon mielenkiintoisempi tapaus. Ryhmien sähköpostiosoitteet on tallennettu tietokantaan (emailGroup.mdb), josta ne luetaan ja käyttäen edellistä servlettiä sähköposti lähetetään.

Toimintaan laittaminen

1. Kopioi tietokanta haluamaasi hakemistoon.
2. Lisää tietokannalle **alias nimellä emailgroup** (Windowsissa avaa control panel-> Administrative tools -> ODBC datasources. Valitse Microsoft access driver)**.**

3. Kopioi sendMailGroup.jsp webapps\tutor hakemistoon.
4. Testaa ryhmiä osoitteessa

   http://localhost/tutor/sendMailGroup.jsp

Tietokanta (emailgroup.mdb)

Tietokannassa on kaksi taulua

1. **Ryhmat**, kentät ryhmaID (laskuri) ja nimi(teksti), avaimena ryhmaID.
2. **Osoitteet**, kentät ryhmaID(luku) ja osoite(teksti), yhdistetty avain.

Käyttöliittymä (sendEmailGroup.jsp)

Esimerkissä käytetään JSP sivua (JSP esitelty seuraavassa osassa) hakemaan ryhmien nimet pudotusvalikkoon. Käyttäjän valittua ryhmä ja kirjoitettua viesti kutsutaan SendEmailGroup servlettiä, joka lähettää postit. SendEmailGroup servletti kutsuu SendMail servlettiä, ja välittää tälle saamansa request ja response oliot. Tämä tehdään seuraavasti

```
RequestDispatcher rd =
this.getServletContext().getRequestDispatcher("/servlet/sendmail.SendMail");

//SendMailin output incluudataan

rd.include(request, response);
```

Huomautus koodeista

Esimerkin koodeissa on monet asiat ”hard koodattu” selvyyden vuoksi. Oikeasti varmasti haluaisit asettaa mahdollisimman monen asian ohjelman ulkopuolelta käyttäen esimerkiksi web.xml tiedoston alustusparametreja tai properties tiedostoa (ks. chat esimerkki).

9.2 Case 2 – Chat

Useimmissa web järjestelmissä on toteutettu chat eli keskustelupalsta. Seuraavassa näet, miten chat voidaan toteuttaa servlettejä käyttäen.

9.2.1 Toteutus

Esimerkin chat sovellus koostuu seuraavista osista:

- Administrointi, jossa voidaan hoitaa uusien huoneiden (chat room) luominen ja olemassa olevien poisto (tämän hoitaa **ChatAdminServlet**).
- Huoneiden listaus, josta käyttäjä valitsee huoneen (**ListRoomsServlet**)
- Huone, jossa käyttäjät keskustelevat (**ChatRoomServlet**)

Datan käsittelystä vastaavat luokat

Sovelluksessa käytetään seuraavia luokkia datan käsittelyyn:

- RoomList
    - Tämän luokan instanssi sisältää viittaukset kaikkiin ChatRoom objekteihin. Tämä objekti on sijoitettu ServletContextiin, jotta kaikki sovelluksen servletit voivat sitä käsitellä. Tästä luokasta luodaan vain yksi instanssi. Tietysti, jos huoneita jaoteltaisiin aiheiden mukaan, voisi tästä luokasta luoda montakin oliota (instanssia).
- ChatRoom
    - Yhtä huonetta kohden on yksi ChatRoom olio, joka sisältää yleiset tiedot huoneesta sekä kaikki ChatEntry objektit.
- ChatEntry
    - Yhden henkilön yksi kommentti muodostaa yhden ChatEntry objektin. Objekti sisältää tiedon siitä, mitä sanottiin ja kuka sanoi.

9.2.2 Järjestelmän toimintaan laittaminen

Järjestelmän toimintaan laittamisen vaiheet

1. Kopioi koko chat hakemisto tutor/WEB_INF/classes hakemiston alle
2. Lisää seuraavat rivit web.xml tiedostoon:

```
<servlet>
 <servlet-name>chatAdmin</servlet-name>
 <servlet-class>chat.ChatAdminServlet</servlet-class>

 <init-param>
  <param-name>chatprops</param-name>
  <param-value>chat.conf</param-value>
 </init-param>

 <load-on-startup>2</load-on-startup>
</servlet>

<servlet>
 <servlet-name>listRooms</servlet-name>
 <servlet-class>chat.ListRoomsServlet</servlet-class>
</servlet>
```

```
<servlet>
 <servlet-name>chatRoom</servlet-name>
 <servlet-class>chat.ChatRoomServlet</servlet-class>
</servlet>
```

3. Tee chat.conf niminen tiedosto tutor hakemistoon (samassa kaikki JSP tiedostot) ja lisää sinne seuraavat rivit (luodaan automaattisesti huoneet java ja cpp):

```
java.description = java_testi_chat
cpp.description = cpp_testi_chat
```

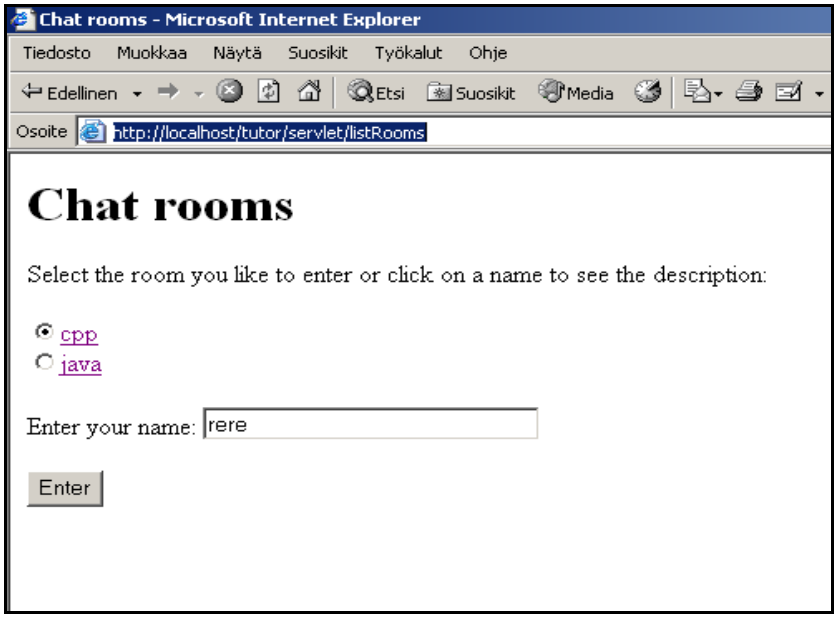
   HUOM! Älä laita välilyöntejä huoneiden nimiin, vaan korvaa ne + merkillä.

4. Lataa tutor WEBAPP uudestaan ja suuntaa osoitteeseen

   http://localhost/tutor/servlet/chat.ListRoomsServlet

   tai (ks. web.xml)

   http://localhost/tutor/servlet/listRooms



Kuva 10. Huoneiden listaaminen ja valittuun huoneeseen liittyminen

Klikkaamalla huoneen nimeä saat huoneesta näkyviin kuvauksen.

Chatin administrointi hoidetaan osoitteesta

http://localhost/tutor/servlet/chatAdmin

Lisää uusi chat huone nimellä ”uusi testi” (tässä saat käyttää välilyöntejä) ja anna huoneelle jokin kuvaus. Seuraavaksi lataa WEBAPP uudestaan komennolla

http://localhost/manager/html/reload?path=/tutor

Käy katsomassa chat.conf tiedostoa ja huomaat, kuinka uusi huone on tallennettu tiedostoon. Esimerkki on muokattu lähteestä [1].

9.2.3 Luokat

Seuraavaksi paneudutaan chat sovelluksen toimintaan.

ChatAdminServlet

Tämän luokan vastuulla ovat seuraavat asiat

- Init-metodissa:
  - RoomList objektin alustus. Tämä hoidetaan java.util.Properties[68] luokkaa käyttäen. Luotavien huoneiden tiedot luetaan tiedostosta, jonka nimi luetaan alustusparametrissa nimeltä ”chatprops”. Properties-luokan toimintaan kannattaa tutustua tarkemmin!
  - RoomList objektin lisäys ServletContextiin. Servletti on configuroitu niin, että se ladataan Tomcatin käynnistyessä. Näin voidaan varmistaa, että roomList on asetettu ennen kuin muita sovelluksen servlettejä käytetään.
- DoGet-metodissa muodostetaan käyttöliittymä, jolla huoneiden lisäys ja poisto hoidetaan.
- DoPost-metodi hoitaa poistot ja lisäykset, ja lopuksi kutsuu doGet-metodia.
- SaveList() metodi tallentaa huoneiden tiedot (voisi toki kutsua destroy metodistakin, ks. koodit).

WEBAPPin polku on esimerkissä otettu selville metodissa getPropsFileName seuraavasti:

```
String propsFile = getServletConfig().getInitParameter("chatprops");

File realPath = new File(
this.getServletContext().getRealPath(propsFile));

return realPath.getAbsolutePath();
```

ListRoomsServlet

Chatin käyttäjä liittyy tämän servletin avulla haluamaansa huoneeseen. Servletti listaa kaikki huoneet ja antaa käyttäjän valita haluamansa huoneen sekä antaa käyttäjätunnuksen (avatar). Servletissä on ainoastaan doGet-metodi, jonka avulla käyttöliittymä muo-

[68] Ks. http://java.sun.com/j2se/1.4.2/docs/api/java/util/Properties.html

dostetaan. Käyttäjän valittua huoneen ja klikattua painiketta kutsutaan ChatRoomServlettiä.

ChatRoomServlet

Varsinainen chatin toiminta on koodattu tähän luokkaan. Servletin avulla käyttäjä kirjoittaa viestinsä, ja servlettiä kutsutaan uudestaan. Mielenkiintoisin osa onkin, miten muiden käyttäjien viestit saadaan näkyviin. Ei ole käytännössä mahdollista (ainakaan helposti) palvelimen kertoa kaikille asiakkaille, että uusia viestejä on tullut ja ne pitäisi näyttää.

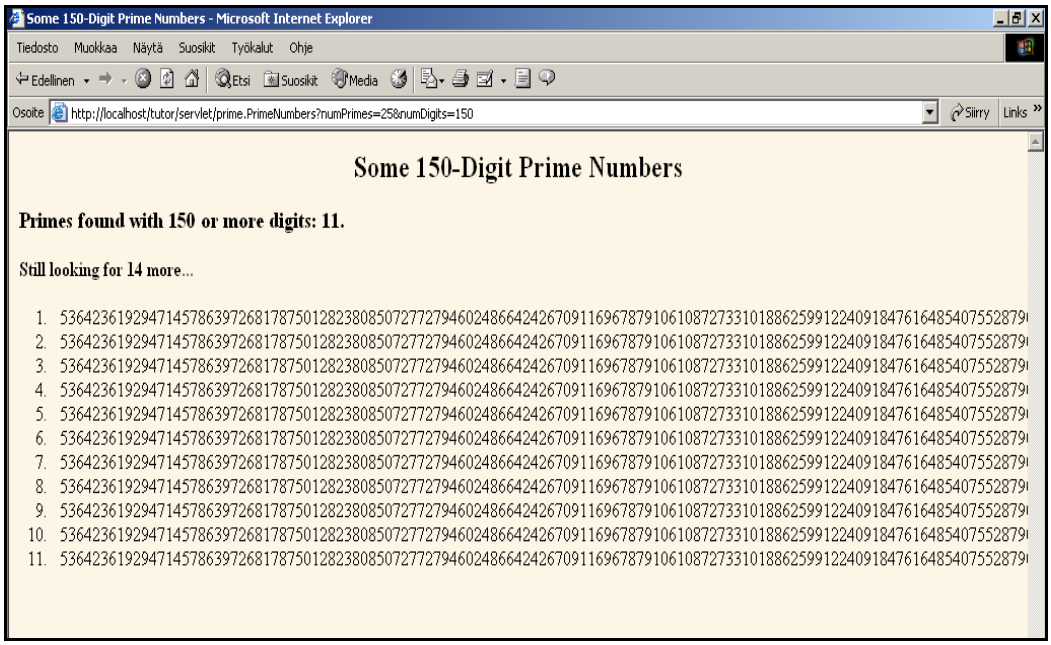
Viestien lukeminen on toteutettu siten, että käyttäen HTML META tagia sivu päivittää itsensä viiden sekunnin välein. Entä jos käyttäjä onkin kirjoittamassa viestiä ja sivu päivitetään? Ei hätää, käyttäjä ei joudu kirjoittamaan viestiä uudestaan toivoen, että hän olisi riittävän nopea. Tästä ongelmasta on päästy eroon käyttäen kehyksiä (frames, frameset). Yksi kehys (joka päivitetään) on viestilistalle ja toinen viestien lähettämistä varten.

9.3 ”Hitaasti valmistuva” servletti

Oletko joskus eksynyt sivulle, jossa olet joutunut odottamaan tuhottoman kauan? Varmasti jokaiselle on käynyt niin. Useimmiten syynä on liiallinen verkkoliikenne kapasiteettiin verrattuna, mutta osansa saattaa olla myös hitaasti suoriutuvilla ohjelmilla.

Joskus saattaa tulla eteesi tehtävä, jonka suoritus kestää melko kauan. Kuitenkin käyttäjälle olisi hyvä näyttää jonkinlaista väliaikaisinformaatiota sovelluksesi tilasta. Miten tällaiseen lopputulokseen päästään servlettejä käyttäen? Seuraavassa esimerkissä (lähde [4]) näytetään vastaus tähän.

Esimerkissä on tehty servletti PrimeNumbers, joka laskee alkulukuja annetun määrän annetulla suuruudella. Itse laskeminen on toteutettu säikeenä. Servletti näyttää tuloksia aina 5 sekunnin välein, kunnes haluttu määrä alkulukuja on saatu laskettua.



Kuva 11. PrimeNumbers servletin suoritus.

Servletti asettaa vastauksen header tietoihin refresh attribuutille arvon 5, mikäli säie ei ole saanut laskutoimitustaan valmiiksi

```
if (!isLastResult) {
   response.setHeader("Refresh", "5");
  }
```

Itse säie on toteutettu luokassa PrimeList. Esimerkin saat ajettua

http://localhost/tutor/PrimeNumbers.html

10 Kaikki tutoriaalin lähdekoodit

Saat näkyviin kaikki tutoriaalissa käytetyt koodit

http://localhost/tutor/viewAllSource.jsp?all=1&hake=src

Parametrilla hake varustettuna viewAllSource listaa kyseisessä hakemistossa olevat tiedostot (ei ainoastaan jsp tiedostot, vaan kaikki).

JSP lähdekoodit saat näkyviin

http://localhost/tutor/viewAllSource.jsp?all=1

11 Osa 4 - Tomcatistä tarkemmin

Tässä osassa tutustutaan tarkemmin Tomcatin toimintaan. Ensin kuitenkin tarkastetaan Java SDK:n toimintaan vaikuttavat asetukset.

11.1 Java SDK:n asetukset kuntoon

Ennen kappaleen aloittamista, testaa että olet tehnyt riittävät asetukset jotta SDK:n työkalut toimivat (alla ohjeet Windowsiin).

- Avaa komentokehoite (cmd / command).
- Anna komento javac.

Mikäli komentoa ei löydetä, lisää SDK:n bin hakemisto PATH- ympäristömuuttujaan.

Lisäys win2000:ssa tapahtuisi seuraavasti:

- Avaa Control Panel->System->Advanced->Environment options.
- Muokkaa path muuttujaa lisäämällä loppuun ; ja Java asennushakemisto bin alihakemisto.
- Käynnistä komentokehoite uudestaan ja testaa komennolla jar.

Lisäys win9X:ssa:

- "Start", "Run" **sysedit** klikkaa **OK**.
- Valitse ikkuna jossa lukee AUTOEXEC.BAT.
- Etsi PATH sana, jollei löydy niin lisää se seuraavasti:

```
PATH C:\WINDOWS;C:\WINDOWS\COMMAND;C:
\J2SDK1.4.1_<version number>\BIN
```

- Avaa komentokehoite ja aja c:\autoexec.bat.

Jollet ohjeista huolimatta saa SDK:n työkaluja käyttöön, katso http://java.sun.com/j2se/.

11.2 WAR ja web sovelluksen käsittely

Tässä kappaleessa käsitellään sovellusten levittäminen käyttäen WAR (web application archive) pakettia. WAR pakettiin sijoitat kaikki tarvittavat tiedostot (ja hakemistot) jotka sovelluksesi tarvitsee.

11.3 WAR paketin tekeminen

Monet Java työkalut (kuten NetBeans) mahdollistavat war-paketin tekemisen helpolla tavalla. Mikäli tällaista työkalua ei ole käytettävissä, joudut tekemään paketin Java SDK:n mukana tulevalla jar-työkalulla.

Paketin tekeminen vaatii oikeanlaisen hakemistorakenteen, eli samanlaisen kuin normaalilla web sovelluksella.

Kun olet saanut hakemistot luotua, lähdekoodit kirjoiteltua ja käännettyä, voit luoda sovellukselle war paketin käyttäen Java SDK:n jar työkalua seuraavasti: Mene ensin juurihakemistoon, eli hakemistoon jonka alta löytyy suoraan WEB-INF hakemisto ja anna seuraava komento:

```
jar cvf paketin_nimi.war
```

Paketti kannattaa tehdä kunnon työkaluilla (kuten Jbuilder), mikäli sinulta sellainen löytyy. Paketin luomisessa SDK:n työkaluilla ei ole mitään hankalaa, mikäli paketti toimii oikein. Paketin configurointi onkin sitten hankalempaa. Lisää war paketeista voit lukea

http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/WebComponents3.html

11.4 WAR paketin asentaminen Tomcattiin

Tomcatissä sijoitat war paketin TOMCAT_HOME\webapps hakemiston juureen. Kun käynnistät tomcatin, puretaan war pakettisi ja web sovelluksesi on valmis käyttöönotettavaksi. Tomcatin ollessa käynnissä voit päivittää war-paketin uuteen versioon. Asetuksista riippuen Tomcat asentaa uuden version sovelluksestasi.

Toinen vaihtoehto asentaa WAR paketti on käyttää tomcatin manageri –apuohjelmaa, jota käyttäen asennus onnistuu selaimen avulla (voit asentaa sovelluksesi myös eri koneelta).

Seuraavalla komennolla asentaminen hoidetaan s.e contextiksi tulee uusi ja war-paketti on ollut hakemistossa c:\java\

http://localhost/manager/install?path=/tutor&war=jar:file:c:/java/tutor.war!/

Itse asiassa paketin voi asentaa miltä tahansa koneelta seuraavalla syntaksilla

```
war=jar:http://hostname:port/relativepath/warfile.war!/
```

esim.

http://192.168.23.20/manager/install?path=/testi&war=jar:http://192.168.23.19/java/testi.war!/

11.5 Tomcat manager

Tomcat manager ohjelman avulla voit hoitaa kaikki seuraavissa kappaleissa kuvatut tehtävät. Katso

http://localhost/manager/html/

11.6 Asennettujen web sovellusten listaus

Listaus tapahtuu komennolla

http://localhost/manager/list

11.6.1 Asennetun web sovelluksen uudelleen lataaminen (reload)

Web sovelluksen uudelleen lataaminen tapahtuu komennolla

http://localhost/manager/reload?path=/sovelluksen_nimi

esim.

http://localhost/manager/reload?path=/tutor

Tällöin kaikki servletit (myös JSP-sivuista käännetyt) ajetaan alas, ja niiden destroy metodit tapahtuvat. Servlettien käynnistyessä uudelleen (määritelty load-on-startup elementillä web.xml tiedostossa) tapahtuvat init-metodit.

11.7 Sessiotiedot

Web sovelluksen sessiotiedot saat selville

http://localhost/manager/sessions?path=/sovelluksen_nimi

11.8 Web sovelluksen pysäyttäminen ja käynnistäminen

Pysäyttäminen tapahtuu stop komennolla seuraavasti

http://localhost/manager/stop?path=/sovelluksen_nimi

Vastaavasti sovellus käynnistetään start komennolla

http://localhost/manager/start?path=/sovelluksen_nimi

11.9 Asennetun sovelluksen poistaminen

Sovelluksen poistaminen (undeploy) tapahtuu seuraavasti

http://localhost/manager/remove?path=/sovelluksen_nimi

Tällä komennolla ei kuitenkaan mitään tiedostoja tai hakemistoja poisteta. Sovellus vain poistetaan Tomcatin sisäisesti listasta, jolla ylläpidetään sovelluksia.

11.10 Security realms

Tomcatin uusiin piirteisiin (versiossa 4) kuuluu security realms, vapaasti suomennettuna turvallisuusalueet, joiden ansiosta Tomcat kestää vertailun entistä paremmin vastaavien kaupallisten tuotteiden kanssa. Tällä tekniikalla saadaan suojattua web sovelluksen resurssit.

Tomcatissä on kaksi eri realm-toteutusta valmiina.

11.10.1 Memory realms

Tomcatin käyttäjien roolit, käyttäjätunnukset ja salasanat on asetettu tällä tekniikalla. Tietoja pääset muokkaamaan TOMCAT_HOME\conf\tomcat-users.xml tiedostosta. Tiedos-

to on yksinkertainen XML notaatiota noudattava konfigurointitiedosto. Alla tiedoston sisältö

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
 <role rolename="tomcat"/>
 <role rolename="role1"/>
 <role rolename="manager"/>
 <role rolename="admin"/>
 <user username="tomcat" password="tomcat" roles="tomcat"/>
 <user username="role1" password="tomcat" roles="role1"/>
 <user username="both" password="tomcat" roles="tomcat,role1"/>
 <user username="admin" password="21902190" fullName="Pekka Kosonen" roles="admin,manager"/>
</tomcat-users>
```

Kuten huomaat, tiedoston alussa on luotu rooleja <role> tagilla, ja myöhemmillä riveillä käyttäjiä. Käyttäjä voi kuulua moneen ryhmään. Tällöin ryhmien nimet erotellaan roles attribuutin arvossa toisistaan pilkulla. Admin ja Manager roolissa olevat käyttäjät voivat käyttää Tomcatin manager sovellusta, jonka graafisen käyttöliittymän saat näkyviin osoitteessa

http://localhost/manager/html/list

Admin roolissa olevat käyttäjät pääsevät lisäksi käsiksi Tomcatin administrointi sovellukseen:

http://localhost/admin/login.jsp

Oman sovelluksen suojaaminen MemoryRealm-tekniikalla

Seuraavaksi suojataan oma web sovelluksemme, tutor.

1. Lisää tomcat-users.xml tiedostoon uusi käyttäjä:

```
<user username="tutor" password="god" roles="tutorUser"/>
```

2. Avaa sovelluksen web.xml tiedosto, ja lisää sinne seuraavat rivit:

```
<security-constraint>
 <web-resource-collection>
  <web-resource-name>tutor webapp</web-resource-name>
```

```
  <url-pattern>/*</url-pattern>
 </web-resource-collection>
 <auth-constraint>
  <role-name>tutorUser</role-name>
 </auth-constraint>
</security-constraint>
<login-config>
 <auth-method>BASIC</auth-method>
  <realm-name>tutor webapp</realm-name>
</login-config>
```

3. Reloadaa tutor webapp komennolla

   http://localhost/manager/html/reload?path=/tutor

Yrittäessäsi käyttää jotain resurssia tutor sovelluksesta, saat eteesi seuraavan ikkunan:

Kuva 12. Autentikointi-ikkuna

11.10.2 JDBC Realms

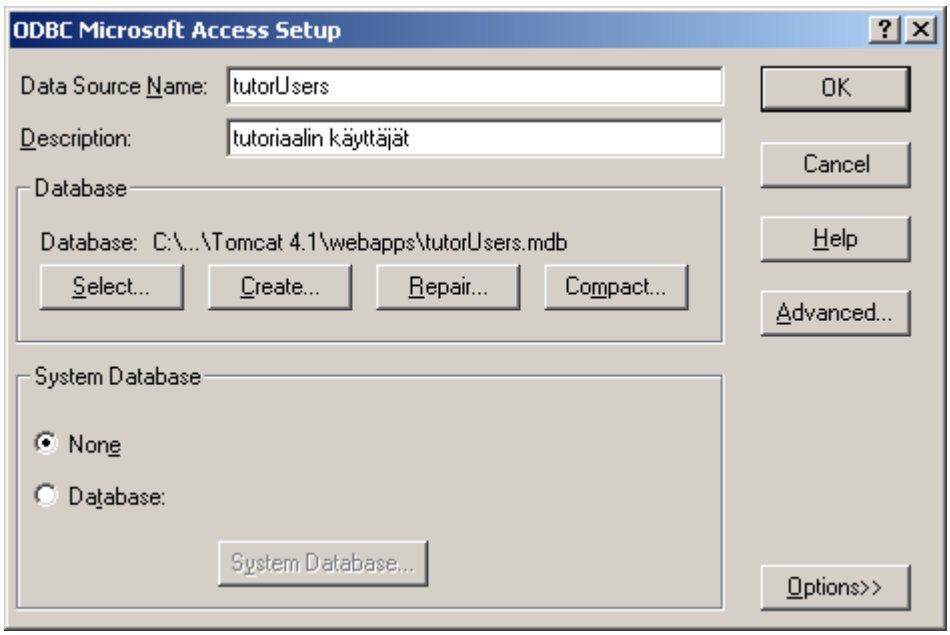
Toinen tapa suojata sovelluksesi on JDBC Realms. Erona edelliseen on se, että käyttäjätiedot tallennetaan tietokantaan. Seuraavassa kerrotaan vaiheet jossa käyttäjätiedot tallennetaan MS Access kantaan (johtuen myöhemmistä vaiheista tietokantaratkaisu ei ole kovin hyvä).

1. Luo seuraavat taulut:

Kuva 13. Taulujen rakenne ja yhteydet

2. Aseta kenttien tyypit seuraavasti:
   - user_name: varchar(12)
   - user_pass: varchar(12)
   - role_name: varchar(12)
3. Syötä rooleihin seuraavat roolinimet:
   - tutorUser
   - tutorDeveloper
   - tomcat
   - manager
   - admin
4. Syötä user_name ja user_pass seuraavasti:
   - user | password
   - pekka | god
   - tutor | god
5. Liitä käyttäjät rooleihinsa lisäämällä seuraavat tietueet (name | role) user_roles tauluun:
   - user | tutorUser
   - pekka | tutorUser
   - pekka | tutorDeveloper
   - pekka | manager
   - pekka | admin
   - tutor | tutorUser

6. Tallenna tietokanta nimellä tutorUsers.mdb.
7. Lisää alias nimellä tutorUsers (Windowsissa avaa Control Panel ->Administrative tools->ODBC datasources->Add ja valitse Microsoft access driver sekä kirjoittele seuraavat tiedot):



Kuva 14. Aliaksen luominen

8. Configuroi server.xml tiedostoa. Kommentoi kaikki muut <realm> elelementit pois ja lisää seuraava elementti:

```
<Realm  className="org.apache.catalina.realm.JDBCRealm" debug="99"
        driverName="sun.jdbc.odbc.JdbcOdbcDriver"
      connectionURL="jdbc:odbc:tutorUsers"
        userTable="users" userNameCol="user_name" userCredCol="user_pass"
      userRoleTable="user_roles" roleNameCol="role_name" />
```

9. Käynnistä Tomcat uudestaan.

Mennessäsi http://localhost/tutor/ hakemistossa mihin tahansa resurssiin saat eteesi loggautumisikkunan.

JDBC Realmin edut

Ensiksi mieleen tulevia etuja ovat ainakin helpompi konfigurointi. Myös käyttäjien lisäys ”lennossa” on helpompaa, eikä muutosten jälkeen tarvitse käynnistää mitään uudestaan.

Käyttäjätietojen selvittäminen

Kun käyttäjä on autentikoinut itsensä (antanut käyttäjätunnuksen ja salasanan, loggautunut) on helppoa lukea käyttäjän tiedot käyttäen HttpServletRequest objektia. JSP sivulla käyttäjätunnuksen lukeminen tapahtuu seuraavasti

```
request.getRemoteUser();
```

11.11 Tomcat valves

Kyseinen tekniikka on tullut käyttöön vasta Tomcatin versiossa 4. Tekniikka mahdollistaa eräänlaisten esiprosessorien käytön (valve =venttiili) Tomcatissä versiossa 4.1.12 on valmiina neljä valvea.

1. Access log valve
   - logi tiedostot käyttäjätiedoista mm.
     - sivujen kävimäärät
     - käyttäjien autentikointi tiedot
2. Remote address Filter
   - Mahdollista rajata käyttäjiä IP-osoitteiden avulla (jokerimerkit sallittuja maskeissa, onneksi!).
3. Remote Host Filter
   - Erona edelliseen on se, että verrataan käyttäjän IP:n jakaneen palvelimen (host) IP:tä, ei itse käyttäjän IP osoitetta.
4. Request Dumper Valve
   - Voit ”dumpata” HTTP otsikkotiedot määritettyyn logiin tietyistä pyynnöistä.

11.12 Servlet filters

Servlettifiltterien[69] avulla pystyt tutkimaan ja muuttamaan sekä pyyntöjä että vastauksia (request/reply). Servlettifiltterit liitetään (mapataan) tiettyihin URL-osoitteisiin. Servlettifilttereitä on mahdollista ketjuttaa. Filtterit esiteltiin servletti spesifikaatioversiossa 2.3.

Filtterien avulla pystyt tekemään monia asioita. Niiden avulla voit esimerkiksi varmistaa autentikoinnin, upottaa vastauksen alkuun ja loppuun jonkin sisällön, muuntaa tulostuksen esim. XML-muotoon tai kryptata tulostuksen.

---

[69] Tarkemmin http://java.sun.com/products/servlet/Filters.html

11.12.1 Servlettifiltterin toteutus

Seuraavassa on toteutettu luokka, joka toimii servlettifiltterinä lisäten jokaiseen pyyntöön yhden attribuutin.

```
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public final class ExampleFilter implements Filter {

  public void init(FilterConfig config) throws
javax.servlet.ServletException {
    System.out.println("ExampleFilter: init"); //debuggausta
  }

  public void destroy() {
    System.out.println("ExampleFilter: destroy ");
  }

public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws java.io.IOException,
javax.servlet.ServletException
{
    System.err.println("ExampleFilter: ennen doFilter()");
    request.setAttribute("alias", new String("jdbc:odbc:tutorUser"));
    chain.doFilter(request, response); //kontrolli ketjun seuraavalle filtterille,
//mikäli sellainen on
```

```
    System.err.println("---->ExampleFilter: jälkeen doFilter()<----");
  }
}
```

Sovelluksen web.xml tiedostoon lisätään seuraavat rivit, joilla filtteri ensin esitellään ja sitten määritellään osoitteet, joihin filtteri vaikuttaa (tässä filtteri käsittelisi kaikki jsp sivut).

```
 <filter>
   <filter-name>Filter 1</filter-name>
   <filter-class>ExampleFilter</filter-class>
</filter>

<filter-mapping>
   <filter-name>Filter 1</filter-name>
   <url-pattern>*.jsp</url-pattern>
</filter-mapping>
```

Testataan filtterin toiminta luomalla JSP sivu, jossa attribuutin arvo luetaan. Ensin kuitenkin web sovellus pitää ladata uudelleen. Filtterin toiminta testataan seuraavalla skriptletillä (filterTest.jsp):

```
<%
  String attrib = (String) request.getAttribute("alias");
  if( attrib == null )
    out.println("Attribuuttia ei saatu luettua. Filtteri ei taida toimia");
  else
   out.println("Filtteri toimii. Attribuutin arvo on " + attrib);

  String user = request.getRemoteUser(); //autentikoitu käyttäjä

  if( user != null )
    out.println("<BR>" + "Olet loggautunut käyttäjätunnuksella " + user);
%>
```

Ole erittäin huolellinen käyttäessäsi servlettifilttereitä. Koodivirheet saattavat aiheuttaa sen, ettei koko sovelluksesi toimi.

Yhteenveto

Tutoriaalissa on tutkittu servlettien ja JSP sivujen perusteita sekä tutustuttu Tomcat-servlettimoottoriin.

Servleteistä on tarkemmin tutkittu servlettifilttereitä. JSP sivujen hankalin osuus on tagikirjastojen muodostaminen.

Tämän tutoriaalin on tarkoitus elää ja kehittyä, tarkista uusin versio osoitteesta

http://sinuhe.jypoly.fi/~pkosonen/webapp/

Seuraavaksi aion lisätä JDBC osuuden sekä isomman esimerkin, joka tuo paremmin papujen (JavaBeans) käyttämisen hyödyt esille.

Mitä seuraavaksi?

Tutoriaalin jälkeen on hyvä tutustua muutamiin kehystöihin (framework), kuten Jakarta Struts.[70] Kehystöjen avulla saadaan web sovelluksiin valmis runko, jonka osat täydentämällä sovellus saadaan toimintaan.

Toinen tutkimisen arvoinen asia on web services, jota voi lähteä tutkimaan esimerkiksi osoitteesta

http://ws.apache.org/

Toivottavasti tutoriaalista on hyötyä! Palautetta otetaan vastaan osoitteeseen pekka.kosonen@jypoly.fi

[70] Ks. http://jakarta.apache.org/struts/

Tehtäviä tutoriaaliin liittyen

Tähtien määrä (1-5) kertoo tehtävän vaikeudesta

***** ohjelma, jolla voidaan rajoittaa aika / katsottujen tiedostojen määrä, jonka yksi käyttäjä saa olla päivässä sivuillamme. Käyttäjätunnus lähetetään sähköpostiin, sähköpostiosoite tallennetaan (periaatteessa monella email-osoitteella pääsee lukemaan monta kertaa)

**** Tietovisa. Kysymyksiä pystyttävä lisäämään ja muokkaamaan selaimella. Tee ensin kiinteällä määrällä vaihtoehtoja. Mieti sitten, miten saisit ohjelman tehtyä vaihtelevalla määrällä vaihtoehtoja

*** Lisää ryhmäposteihin liittyminen ja sieltä poistuminen. Lisää myös mahdollisuus luoda uusia postilistoja.

** Muokkaa viewAllSource.jsp sivun toimintaa siten, että parametrina voidaan myös tiedoston tarkennin. Tällöin ainoastaan tällä tarkentimella olevat tiedostot tulostetaan. Joudut muokkaamaan myös viewSource.jsp:tä.

** Tee JSP sivu printTable.jsp, joka tulostaa tietokannassa olevan taulun kaikkien tietueiden tiedot. Tietokannan alias ja taulun nimi tuodaan parametreina. Myös kenttien nimet tulostetaan.

* Tee JSP sivu laske.jsp, joka laskee parametrina saatujen kahden kokonaisluvun summan, erotuksen ja keskiarvon sekä tietenkin tulostaa ne näkyviin.

*** Tee seuraavanlainen ajanvaraussovellus käyttäen JSP:tä ja JavaBeaneja:

Sovellukseen listataan vapaita aikoja, esim. maanantai kello 8-10. Tee JSP sivu, joka tulostaa kaikki vapaat ajat. Sivua käyttäen aika on mahdollista myös varata. Varatessa aikaa on annettava varaajan sähköpostiosoite. Varatut ajat tulostetaan erillisellä JSP sivulla, jolloin näytetään myös linkki varaajan sähköpostiosoitteeseen (mailto linkki)

Lähteet

Kirjat

1. Professional Java server Programming (WROX 1999, www.wrox.com)
2. Apache Jakarta-Tomcat (James Goodwill, Apress, www.virtuas.com/publications.html)
3. DHTML (2001, Tero Linjama, Docendo)
4. Core servlets and Java server pages (Marty Hall, Sun, 2001)
5. More servlets and Java server pages (Marty Hall, Sun, 2002)

Internet lähteet

6. http://jakarta.apache.org/tomcat/
7. http://java.sun.com/
8. http://www.moreservlets.com/
9. http://java.sun.com/j2ee/tutorial/1_3-fcs/ (J2EE tutorial)
10. http://developer.iplanet.com/viewsource/fields_jspcomp/fields_jspcomp.html

12 Koodilistaus

Koodilistaus on tuotettu viewSource.jsp:llä

12.1 JSP sivut

12.1.1 aiheutaPoikkeus.jsp

```
<%@ page errorPage="poikkeusSivu.jsp" %>

<html>
<head><title>Webapplication tutoriaali - virhesivun käyttö</title></head>
<body>
<%
  int i = 10;
  i = i / 0;
%>

</body>
</html>
```

12.1.2 aiheutaPoikkeusEiToimi.jsp

```
<%@ page errorPage="poikkeusSivu.jsp" buffer="1kb"%>

<html>
<head><title>Webapplication tutoriaali - virhesivun käyttö</title></head>
<body>
<%
  for ( int i = 0; i < 100; i++ )
    out.println("Täytetään puskuria " + i + ". kertaa.");
  int i = 10;
  i = i / 0;
%>

</body>
</html>
```

12.1.3 date.jsp

```
<%--

  Copyright 2001 Sun Microsystems, Inc. All Rights Reserved.

  This software is the proprietary information of Sun Microsystems, Inc.
  Use is subject to license terms.

  Modified 8.1.2003 for Java SDK 1.4 by Pekka Kosonen (all beans must be in packages in SDK1.4)
--%>

<%@ page import="java.util.*" %>

<html>
<body bgcolor="white">
<%@ page import="pavut.*" %>
```

```
<jsp:useBean id="date" class="pavut.MyDate" type="MyDate" />
<jsp:useBean id="locales" scope="application" class="pavut.MyLocales"/>
<%
  Locale locale = locales.getLocale( request.getParameter("locale") );
  if (locale != null) {
%>
<jsp:setProperty name="date" property="locale" value="<%=locale%>"/>
The date in <b><%=locale.getDisplayName()%></b> is <b><%=date.getDate()%></b>
<% } %>
</body>
</html>
```

12.1.4 errorPapuTesti.jsp

```
<html>
<head>
  <title>WebApplication tutoriaali - JavaBeanin käyttö</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body BACKGROUND="/tutor/images/papu.gif" >


<jsp:useBean id="papu" class="pavut.ErrorPapu" scope="session"/>

<table align="center" width="50%">
<tr align="center">
<td>
<% if (request.getParameter("arvo") != null )
   {
      papu.muutaLukua(request.getParameter("arvo"));
      if ( papu.getVirheLippu() )
      {
        out.print("Anna kokonaislukuarvo");
        papu.virheLippuAlas();
      }
      %>

      <h3>
        Pavun arvo on nyt <jsp:getProperty name="papu" property="luku" />
      </h3>
      <%
   }
   else
     out.print("Papu luotu");
%>
<%--
<% if (request.getParameter("arvo") != null )
     out.print("<h3>Pavun arvo nyt " +
     papu.muutaLukua(request.getParameter("arvo")) );
   else
     out.print("Papu luotu");
%>
--%>
</td>
</tr>
</table>

<br>
<FORM name="papuTesti" METHOD=POST ACTION="errorPapuTesti.jsp">

<br>
```

```
<table align="center" width="50%" cellspacing="10">
<tr align="center">
  <td>
    Kasvata arvoa:
  </td>
  <td>
    <INPUT TYPE="text" name="arvo"><br>
  </td>
<tr align="center">
  <td>
    <INPUT TYPE=submit value="submit">
  </td>
</tr>
</table>
</FORM>

</body>
</html>
```

12.1.5 filterTest.jsp

```
<html>
<head>
<title>WebApplication tutoriaali - Servlet filterin testaus</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<%
  String attrib = (String) request.getAttribute("alias");
  if( attrib == null )
    out.println("Attribuuttia ei saatu luettua. Filtteri ei taida toimia");
  else
   out.println("Filtteri toimii. Attribuutin arvo on " + attrib);

  String user = request.getRemoteUser();

  if( user != null )
    out.println("<BR>" + "Olet loggautunut käyttäjätunnuksella " + user);
%>

</body>
</html>
```

12.1.6 hello.jsp

```
<html>
<head>
<title>WebApplication tutoriaali - Eka JSP</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

  <table width="500" border="0" cellspacing="0" cellpadding="0">
    <tr>
      <td> </td>
    </tr>
    <tr>
    <td>
      <img src="/tutor/images/homersuklaa.gif"></td>
    <td>
      <b>Life is like a box of chocolates. <BR>One minute it's full of goodies and before you relise it's empty.</b>
```

```
   </td>
   </tr>
   <tr>
     <td> </td>
   </tr>
  </table>
</body>
</html>
```

12.1.7 jep.jsp

```
<html>
<head>
  <title>WebApplication tutoriaali - lomakkeen tietojen tarkistaminen JavaScriptillä</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

  <table width="500" border="0" cellspacing="0" cellpadding="0">
    <tr>
      <td> </td>
    </tr>
    <tr>
    <td>
      <b>Terve <%= request.getParameter("nimi") %> ! </b>
      <p> Pääsit sitten läpi tarkastuksesta.
          Annoit sitten syntymävuodeksesi <%= request.getParameter("svuosi") %> ja
                  emailiksi <%= request.getParameter("email") %>
        </p>
    </td>

    </tr>

    <tr>
    <td>
      <a href="" onClick = "this.href=history.go(-1)"  > takaisin </a>
    </td>
    </tr>

    <tr>
      <td> </td>
    </tr>
  </table>
</body>
</html>
```

12.1.8 JSPversio.jsp

```
<html>
<title>Webapplication tutoriaali - JSP version näyttäminen</title>
</html>

<%@ page import="javax.servlet.jsp.JspFactory" %>
<% JspFactory factory = JspFactory.getDefaultFactory(); %>

<body>
<p>
Käytössä on JSP versio
<%= factory.getEngineInfo().getSpecificationVersion() %>
</body>
```

```
</html>
```

12.1.9 lastModified.jsp

```
<%@ page import="java.io.*,java.util.*, pavut.*" %>

<%
  File f = new File(application.getRealPath( request.getServletPath() ));
  Date modified = new Date( f.lastModified() );
%>
<jsp:useBean id="date" class="pavut.MyDate" type="MyDate" />

<HTML>
<BODY>
Sivua on viimeksi muokattu
<%
  out.print( date.toFinnishDateTime(modified) );
%>
</BODY>
</HTML>
```

12.1.10 locales.jsp

```
<%--

  Copyright 2001 Sun Microsystems, Inc. All Rights Reserved.

  This software is the proprietary information of Sun Microsystems, Inc.
  Use is subject to license terms.

  Modified 8.1.2003 for Java SDK 1.4 by Pekka Kosonen (all beans must be in packages in SDK1.4)
--%>

<%@ page import="java.util.*" %>
<%@ page import="pavut.MyLocales" %>
<%@ page contentType="text/html; charset=ISO-8859-5" %>
<html>
<head><title>Localized Dates</title></head>
<body bgcolor="white">
<jsp:useBean id="locales" scope="application" class="pavut.MyLocales"/>
<form name="localeForm" action="locales.jsp" method="post">
<b>Locale:</b>
<select name=locale>
<%
  Iterator i = locales.getLocaleNames().iterator();
  String selectedLocale = request.getParameter("locale");
                while (i.hasNext()) {
                                String locale = (String)i.next();
                                if (selectedLocale != null && selectedLocale.equals(locale) ) { %>
    <option selected><%=locale%></option>
<%                              } else { %>
    <option><%=locale%></option>
<%                              }
  }
%>
</select>
<input type="submit" name="Submit" value="Get Date">
</form>
<p>
```

```
<jsp:include page="date.jsp" flush="true" />
</body>
</html>
```

12.1.11 LueContext.jsp

```
<html>
<body>
<head>
  <meta http-equiv=\"Content-Type\" content="text/html; charset=iso-8859-1">
<title>
<%
  String alku = (String) this.getServletContext().getAttribute("titleBegin");
  out.print( alku + "ServletConfig käyttö" );
%>
</title>
</head>
<body>
  <H1 align="center"> Esimerkkissä luettiin ServletContextista otsikko (ks selaimen otsikkopalkki). </H1>
</body>
</html>
```

12.1.12 papuTesti.jsp

```
<html>
<head>
  <title>WebApplication tutoriaali - JavaBeanin käyttö</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body BACKGROUND="/tutor/images/papu.gif" >


<jsp:useBean id="papu" class="pavut.LukuPapu" scope="page"/>

<table align="center" width="50%">
<tr align="center">
<td>
<% if (request.getParameter("arvo") != null )
     out.print("<h3>Pavun arvo nyt " +
     papu.muutaLukua(request.getParameter("arvo")) );
   else
     out.print("Papu luotu");
%>

</td>
</tr>
</table>

<br>
<FORM name="papuTesti" METHOD=POST ACTION="papuTesti.jsp">

<br>
<table align="center" width="50%" cellspacing="10">
<tr align="center">
  <td>
    Kasvata arvoa:
  </td>
  <td>
    <INPUT TYPE="text" name="arvo"><br>
```

```
  </td>
<tr align="center">
  <td>
    <INPUT TYPE=submit value="submit">
  </td>
</tr>
</table>
</FORM>

</body>
</html>
```

12.1.13 papuTestiTarkastus.jsp

```
<html>
<head>
  <title>WebApplication tutoriaali - JavaBeanin käyttö</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body BACKGROUND="/tutor/images/papu.gif" >


<jsp:useBean id="papu" class="pavut.LukuPapu" scope="page"/>

<table align="center" width="50%">
<tr align="center">
<td>
<% String arvo = "";
   int muutos = 0;
   try {
     arvo = request.getParameter("arvo");

     if( arvo == null )
       out.print("papu luotu");
     else
     {
       muutos = Integer.parseInt( arvo );
       out.print("<h3>Pavun arvo nyt " + papu.muutaLukua( muutos ) );
     }
   }
   catch (NumberFormatException e )
   {
     out.print("Anna kokonaislukuarvo.<BR>");
     out.print( "(" + e.toString() + ")");
   }
   catch (Exception ee )
   {
     out.print("Tapahtui odottamaton poikkeus" + ee.toString());
   }
%>

</td>
</tr>
</table>

<br>
<FORM name="papuTesti" METHOD=POST ACTION="papuTestiTarkastus.jsp">

<br>
<table align="center" width="50%" cellspacing="10">
<tr align="center">
```

```
  <td>
    Kasvata arvoa:
  </td>
  <td>
    <INPUT TYPE="text" name="arvo"><br>
  </td>
<tr align="center">
  <td>
    <INPUT TYPE=submit value="submit">
  </td>
</tr>
</table>
</FORM>

</body>
</html>
```

12.1.14 poikkeusSivu.jsp

```
<%@ page isErrorPage="true" %>
<html>
<head><title>Webapplication tutoriaali - poikkeussivu</title></head>
<body BACKGROUND="/tutor/images/doh.gif">
<h1>Tapahtui seuraava poikkeus:<br>
<%= exception.toString() %><br>
Palaa takaisin ja yritä uudelleen. Jos se ei auta, ota yhteyttä järjestelmänvalvojaan. <br>
</h1>
</body>
</html>
```

12.1.15 selain.jsp

```
<%--
  selain.jsp

  Testataan selaimen versio, käyttäen beaniä
--%>

<html>
<title>
Webapplication tutoriaali - selainversion testaaminen
</title>

<%@ page import="pavut.*" %>

<jsp:useBean id="selain" class="pavut.Selain"/>

<%
  out.print("<BR>");
  selain.setRequest(request);
  if ( selain.isIE() )
    out.print("Selaimesi on IE");
  if ( selain.isNS7() )
    out.print("Selaimesi on Netscape 7");
  if ( selain.isNS6() )
    out.print("Selaimesi on Netscape 6");
  if ( selain.isNS4() )
    out.print("Selaimesi on Netscape 4");
```

```
 out.print( "<BR>" + selain.getUseragent() );
 out.print("<BR>");
%>
</body>
</html>
```

12.1.16 sendEmailGroup.jsp

```
<html>
<head>
 <title>WebApplication tutoriaali - Ryhmäpostin lähetys</title>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<%@ page import="java.sql.*" %>

<%!
  Connection conn = null;
  Statement stmt = null;
  ResultSet rs = null;
  String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
  String db = "jdbc:odbc:emailgroup";
  String table = "ryhmat";
  String field = "nimi";
  String selectName = "ryhma";
%>

<%! public void jspInit()
   {
    try {
     connect();
    } catch (Exception e) { log("senEmailGroup.jsp jspInit: ", e); }
   }

   public void jspDestroy()
   {
     try {
       conn.close();
     } catch( SQLException e) { }
     conn = null;
   }

 private synchronized void connect() throws Exception {
   if( conn == null || conn.isClosed() )
   {
     Class.forName(driver).newInstance();
     conn = DriverManager.getConnection(db);
     stmt = conn.createStatement();
   }
 }
%>
<FORM action="servlet/sendmail.SendEmailGroup" method="post">
  <TABLE>
   <tbody align="right">
     <TR><TD>Ryhmä </TD><TD> <%= haeRyhmat() %> </TD></TR>
     <TR><TD>Vastausosoite:</TD><TD><input type="text" name="from" size="64"></TD></TR>
     <TR><TD>Aihe:</TD><TD><input type="text" name="subject"
        size="64"></TD></TR>
     <TR><td colspan="2"><textarea name="text" rows="8"
        cols="64"></TEXTAREA></TD></TR>
     <TR><td colspan="2"><input type="submit" value="Lähetä"> <input
        type="reset" value="tyhjennä kentät"></TD></TR>
```

```
   </TBODY>
  </TABLE>
</FORM>


</body>
</html>
<%!
String haeRyhmat() throws Exception
{
 StringBuffer sb = new StringBuffer("");


 rs = stmt.executeQuery("SELECT " + field + " FROM " + table);

 sb.append("<select size=\"1\" name=\"" + selectName  +"\">");
 while ( rs.next() )
 {
  sb.append("<option>");
  sb.append( rs.getString(field) );
  sb.append("</option>");
 }
 sb.append("</select>");
 return sb.toString();
}
%>
```

12.1.17 session.jsp

```
<%@ page import="java.util.*" %>
<html>
<head>
 <title>WebApplication tutoriaali - Sessio tietojen lukeminen</title>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<%
 if ( request.getParameter("tapa") == null )
 {
  out.println("Uusi " + session.isNew()+ "<BR>" );
  out.println( "ID " + session.getId() + "<BR>");
  out.println( "Luotu " + session.getCreationTime() + "<BR>");
  out.println( "Käsitelty viimeksi " + session.getLastAccessedTime() + "<BR>");
  out.println( "Voimassa " + session.getMaxInactiveInterval() + "<BR>");
  out.println("<h3><a href = \"session.jsp?tapa=1\">InAktivoi sessio tästä </a> </h3>");

  Enumeration e = session.getAttributeNames();

  out.println("Sessiolla on seuraavat attribuutit:<BR>" );
  while ( e.hasMoreElements() )
  {
   out.println( e.nextElement()+ "<BR>");
  }
 }
 else
 {
  session.invalidate();
  out.println("Sessio inaktivoitu.");
 }
%>

</body>
```

```
</html>
```

12.1.18 sessionTestaus.jsp

```
<%@ page import="java.util.*" %>
<html>
<head>
  <title>WebApplication tutoriaali - Sessio tietojen lukeminen</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<%!
  String URL;
  String encodedURL;
%>

<%
  if (session.getAttribute("testi") == null )
  {
    session.setAttribute("testi", "testi");
    out.println("Attribuutti lisätty");
  }
  URL = "session.jsp";
  encodedURL = response.encodeURL(URL);
%>

<BR> <a href="session.jsp"> JSP (ei encodedURL) </a>
<BR> <a href=" <%=encodedURL%> "> JSP (encodedURL </a>
</body>
</html>
```

12.1.19 sysCommand.jsp

```
<%@ page import="java.io.*" %>

<%!
public String run(String cmd) {
  try {
    Runtime rt = Runtime.getRuntime();
    Process p = rt.exec(cmd);
    InputStreamReader in = new InputStreamReader(p.getInputStream());
    BufferedReader reader = new BufferedReader(in);
    StringBuffer buf = new StringBuffer();
    String line;
    String newline = "\n";
    while ((line = reader.readLine()) != null) {
      buf.append(line);
      buf.append(newline);
    }
    reader.close();
    p.getInputStream().close();
    p.getOutputStream().close();
    p.getErrorStream().close();
    p.waitFor();
    return buf.toString();
  }
  catch (Exception e) {
    return (e.getMessage());
  }
```

```
}
%>
<html>
<head>
 <title>Webapplication tutoriaali - järjestelmäkomentojen suorittaminen</title>
</head>
<body>
<%-- Unix ollut pystyssä <%= run("/usr/bin/uptime") %> --%>
Ajetaan c:\temp\winver.bat tiedosto, joka sisältää ainoastaan komennon: <BR>
ver <BR>
Tulos näkyy alla <BR>
<%= run("c:\\temp\\aja.bat") %>
</body>
</html>
```

12.1.20 tagTest.jsp

```
<%@ taglib uri="/tutor" prefix="tutor" %>
<html>
<head>
  <tutor:title titleEnd="tagin attribuutille arvon asettaminen"/>
</head>
<body>
  <h1> Tagia käytetty sivun otsikon luomiseen. </h1>
</body>
</html>
```

12.1.21 testi.jsp

```
<%@ page info="Hello" %>
<html>
<head>
<title>
testi
</title>
</head>
<body>
<h1>
JBuilder Generated JSP
</h1>

  <jsp:include page="/viewSource.jsp">
   <jsp:param name="url" value="selain.jsp"/>
  </jsp:include>

<%! int i=0; %>

<% for (i=0; i < 5; i++ )
    out.print(i);
 %>
<BR>Silmukan jälkeen i on <%= i %>
<BR>

</body>
</html>
```

12.1.22 viewAllSource.jsp

```
<%--
  viewAllsource.jsp
  -----------------

  Listaa samassa WEBAPPissa (samassa hakemistossa) olevien jsp tiedostojen
  sisällöt.

  1) jos et anna parametria all, tulostetaan linkkilista
     http://localhost/tutor/viewAllSource.jsp
  2) jos annat, tulostetaan kaikkien tiedostojen koodit lomakkeelle
     http://localhost/tutor/viewAllSource.jsp?all=jotain

  Mikäli annetaan parametrille hake jokin arvo, listataan
  tiedostot tästä hakemistosta. Esim, jos annat parametrille
  hake arvon src, listataan kaikki tiedostot, jotka löytyvät
  hakemistosta tutor/src. Tällöin listataan myös muut kuin jsp tiedostot.

  Käyttää apuna viewSource.jsp:tä, jonka responset incluudataan
  silmukassa.

  Tehty:hyvin myöhään yöllä 10.1.2003, Pekka Kosonen
--%>

<%@ page import="java.io.*" %>
<%! String hake = null; %>
<%
File path = new File(application.getRealPath("viewAllSource.jsp")).getParentFile();
hake = request.getParameter("hake");
if( hake != null )
{
 String newPath = path.toString() + "\\" + hake;
 path = new File(newPath);
}
%>
<html>
<head>
  <title>
    Webapplication tutoriaali - Lähdekoodit
  </title>
</head>
<body>
<table border="0" width = "100%" >
<%
  File [] files = path.listFiles();
  if( files != null )
  for( int i = 0; i < files.length; i++ ) {
    String nimi = files[i].getName();

    if (tarkennin( nimi ).compareToIgnoreCase(".jsp") != 0 && hake == null)
      continue;
%>
<tr>
<td>
<%
  if( request.getParameter("all") == null )
  {
    out.print("<a href=/tutor/viewSource.jsp?url=");
    out.print(nimi + ">");
    out.print("<h3>"+nimi+"</h3></a>");
  }
```

```
  else
  {
%>
  <jsp:include page="/viewSource.jsp">
   <jsp:param name="url" value="<%=files[i].getName()%>"/>
  </jsp:include>
<%
  }
}
%> <%-- Silmukka päättyy --%>
</td>
</tr>
<%!
  public String tarkennin(String file)
  {
    int i = file.lastIndexOf(".");
    if( i == -1)
      return "";
    return file.substring(i, file.length() );
  }
%>
</table>
</body></html>
```

12.1.23 viewSource.jsp

```
<%@ page import="java.io.*" %>
<%
String url = request.getParameter("url");

if (url.indexOf("..") > -1)
  throw new java.io.IOException("Relative paths are not allowed");

String hake = request.getParameter("hake");
String polku = (new File (application.getRealPath(url) ).getParent() );
String fileName =(new File (application.getRealPath(url) ).getName() );

File realPath = null;

if( hake == null )
{
  realPath = new File(application.getRealPath(url) );
  out.println("<h3>" + "<a href=\"" + url + "\">" + url + "</a></h3>");
}
else
{
  realPath = new File(polku + "\\" + hake + "\\" + fileName);
  out.println("<h3>" + "<a href=\"" + hake + "/" + url + "\">" + url + "</a></h3>");
}
%>
<html><head><title>Tiedosto <%= url %></title></head><body><pre>
<%
FileInputStream fis = null;
try {
  if( realPath == null )
    throw new IOException("realpath null");
  fis = new FileInputStream(realPath);
  if( fis == null )
    throw new IOException("realpath null");
  BufferedReader reader;
```

```
  reader = new BufferedReader(new InputStreamReader(fis));
  String line;
  while ((line = reader.readLine()) != null) {
    line = replace(line, "&", "&amp;");
    line = replace(line, "<", "&lt;");
    line = replace(line, ">", "&gt;");
    out.println(line);
  }
}
catch (IOException e) {
  out.println("IOException: " + e.getMessage());
}
finally { if (fis != null) fis.close(); }
%>
</pre></body></html>
<%!
public String replace(String s, String old, String replacement) {
   int i = s.indexOf(old);
   StringBuffer r = new StringBuffer();
   if (i == -1) return s;
   r.append(s.substring(0,i) + replacement);
   if (i + old.length() < s.length())
     r.append(replace(s.substring(i + old.length(), s.length()),
      old, replacement));
   return r.toString();
}
%>
```

12.1.24 vsButton.jsp

```
<%@ page import="java.io.*,java.util.*,java.text.*" %>
<% String me = request.getRequestURI(); %>
<script language="JavaScript">
function show(url) {
  opts="height=400,width=600,scrollbars=yes,resizable=yes"
  window.open("viewsource.jsp?url="+escape(url), "src", opts);
}
</script>
<form><div align="right">
<input type="button" value="View Source"
onClick="show('<%= me %>')"></div>
</form>
```

12.2 Muut koodit

Koodeista jätetty pois aliohj.java listaus paperin säästämiseksi. Lisäksi aliohjelmia on käytetty hyvin vähän tutoriaalissa.

12.2.1 AppletServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;

import java.util.*;
```

```
import java.sql.*;
import java.io.*;

public class AppletServlet extends HttpServlet {

  Connection conn;

  public void init() throws ServletException {

    try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      String dbURL = "jdbc:odbc:tutorUsers";
      conn = DriverManager.getConnection(dbURL);
    } catch (Exception e) {
      System.out.println("tietokantayhteyttä ei saada avattua <BR> "
      + e.toString() );
      return;
    }
  }
  public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
   doPost(req, res);
 }

 public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    PrintWriter out = res.getWriter();
    res.setContentType("text/html");

    String qry = req.getParameter("qry");

    if( qry == null )
    {
      out.println("Parametria ei saatu luettua.");
      out.close();
      return;
    }
/*    else //debug
     {
       out.println("qry" + qry);
       out.close();
     }
*/
    try {
      Statement s = conn.createStatement();
      ResultSet rs = s.executeQuery(qry);
      int col = rs.getMetaData().getColumnCount();

      //käydään tulosjoukko tietue kerrallaan läpi
      while ( rs.next() ) {
        out.println("Tietue nro " + rs.getRow() + ":");

        for( int i=1; i <= col; i++)
        {
          //tulostataan kentän nimi
          out.print( rs.getMetaData().getColumnName(i) + ":");
          //ja tietueen kyseisen kentän arvo
          out.println( rs.getString(i));
        }
      }
    } catch (SQLException e) {
      out.println( "SQL-VIRHE " + e.toString() );
```

```
      System.out.println( e.toString() );
      return;
    }
    out.println();
    out.close();
  }

  public void destroy() {
    try {
      conn.close();
    } catch (Exception e) {
      System.out.println("Tietokantayhteyden sulkeminen ei onnistu <BR>"
      + e.toString() );
    }
  }
}
```

12.2.2 ChatAdminServlet.java

```
package chat;

import javax.servlet.*;
import javax.servlet.http.*;

import java.io.*;
import java.util.*;

import chat.*;
import Aliohj;

public class ChatAdminServlet extends HttpServlet
{
  public void init() throws ServletException
  {
    String propsFile = this.getPropsFileName();

    Properties props = new Properties();

    try
    {
      InputStream is = new FileInputStream(propsFile);
      props.load(is);
      is.close();
    }
    catch (Exception e)
    {
      //Aliohj.kirjoitaRivi("C:\\temp\\props.txt", "Chatprops " + e.toString(), true);
      throw new UnavailableException(this, "Can't read the chatprops file " +
          propsFile);
    }

    RoomList roomList = createRooms(props);
    getServletContext().setAttribute("roomList", roomList);
    //Aliohj.kirjoitaRivi("C:\\temp\\props.txt", "Chatprops Asetettu", true);
  }

  public void doGet(HttpServletRequest req, HttpServletResponse res)
            throws IOException, ServletException
  {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
```

```
    writePage(out);
    out.close();
  }

  public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws IOException, ServletException
  {

    boolean isListModified = false;
    RoomList roomList =
      (RoomList) getServletContext().getAttribute("roomList");

    // Update the room list
    String[] removeList = req.getParameterValues("remove");
    if (removeList != null)
    {
      roomList.removeRooms(removeList);
      isListModified = true;
    }

    String roomName = req.getParameter("roomname");
    String roomDescr = req.getParameter("roomdescr");
    if (roomName != null && roomName.length() > 0)
    {
      roomList.addRoom(new ChatRoom(roomName, roomDescr));
      isListModified = true;
    }

    if (isListModified)
    {
      saveList(roomList);
    }

    doGet(req, res);
  }

  private void saveList(RoomList roomList) throws UnavailableException
  {
    String propsFile = this.getPropsFileName();

    Properties props = new Properties();
    Enumeration rooms = roomList.getRooms();
    while (rooms.hasMoreElements())
    {
      ChatRoom room = (ChatRoom) rooms.nextElement();
      String roomName = StringUtils.replaceInString(room.getName(), " ", "+");
      props.put(roomName + ".description", room.getDescription());
    }
    try
    {
      OutputStream os = new FileOutputStream(propsFile);
      props.store(os, "Generated by ChatAdminServlet");
      os.close();
    }
    catch (Exception e)
    {
      throw new UnavailableException(this, "Can't write the chatprops file " +
        propsFile);
    }
  }

  private RoomList createRooms(Properties props)
  {
```

```
    RoomList roomList = new RoomList();
    Enumeration propKeys = props.keys();
    while (propKeys.hasMoreElements())
    {
      String key = (String) propKeys.nextElement();
      //Aliohj.kirjoitaRivi("C:\\temp\\props.txt", "Key =" + key, false);
      if (key.endsWith(".description"))
      {
        String roomDescription = props.getProperty(key);
        String roomName = key.substring(0, key.lastIndexOf("."));
        roomName = StringUtils.replaceInString(roomName, "+", " ");
        //Aliohj.kirjoitaRivi("C:\\temp\\props.txt", "name =" + roomName, false);
        //Aliohj.kirjoitaRivi("C:\\temp\\props.txt", "kuvaus =" + roomDescription, false);
        roomList.addRoom(new ChatRoom(roomName, roomDescription));
      }
    }
    return roomList;
  }

  private void writePage(PrintWriter out)
  {
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Chat room administration</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<H1>Chat room administration</H1>");
    out.println("<FORM METHOD=POST ACTION=chatAdmin>");

    // Add check boxes for removing rooms
    out.println("Check off the rooms you would like to remove:<P>");
    RoomList roomList = (RoomList) getServletContext().getAttribute("roomList");
    Enumeration rooms = roomList.getRooms();
    while (rooms.hasMoreElements())
    {
      ChatRoom room = (ChatRoom) rooms.nextElement();
      out.println("<INPUT TYPE=CHECKBOX NAME=remove VALUE='" +
            room.getName() + "'>" + room.getName() + "<BR>");
    }
    // Add fields for adding a room
    out.println("<P>Describe the room you would like to add:<P>");
    out.println("<TABLE>");
    out.println("<TR><TD>Name:</TD><TD><INPUT NAME=roomname SIZE=50></TD></TR>");
    out.println("<TR><TD>Description:</TD>");
    out.println("<TD><TEXTAREA NAME=roomdescr COLS=50 ROWS=15>");
    out.println("</TEXTAREA></TD></TR>");
    out.println("</TABLE>");

    // Add submit button
    out.println("<P><INPUT TYPE=SUBMIT VALUE='Update List'>");

    out.println("</FORM>");
    out.println("</BODY></HTML>");
  }

  private String getPropsFileName() throws UnavailableException
  {
    String propsFile = getServletConfig().getInitParameter("chatprops");

    if (propsFile == null || propsFile.length() == 0)
    {
      throw new UnavailableException(this, "chatprops not set");
    }

   File realPath = new File( this.getServletContext().getRealPath(propsFile));
```

```
    //Aliohj.kirjoitaRivi("C:\\temp\\filee.txt", realPath.getAbsolutePath(), true);

    return realPath.getAbsolutePath();
  }
}
```

12.2.3 ChatEntry.java

```
package chat;

import java.util.*;

public class ChatEntry
{
  private String avatarName;
  private String chatMessage;

  public ChatEntry(String avatarName, String chatMessage)
  {
    this.avatarName = avatarName;
    this.chatMessage = chatMessage;
  }

  public String getAvatarName()
  {
    return avatarName;
  }

  public String getChatMessage()
  {
    return chatMessage;
  }
}
```

12.2.4 ChatRoom.java

```
package chat;

import java.util.*;

public class ChatRoom
{
  private static final int MAX_ENTRIES = 5;
  private String name;
  private String description;
  private Vector chatEntries = new Vector();
  public ChatRoom(String name, String description)
  {
    this.name = name;
    this.description = description;
  }

  public synchronized void addChatEntry(ChatEntry entry)
  {
    chatEntries.addElement(entry);
    if (chatEntries.size() > MAX_ENTRIES)
    {
      chatEntries.removeElementAt(0);
```

```
    }
  }

  public Enumeration getChatEntries()
  {
    return chatEntries.elements();
    }

  public String getDescription()
  {
    return description;
  }

  public String getName()
  {
    return name;
  }
}
```

12.2.5 ChatRoomServlet.java

```
package chat;

import javax.servlet.*;
import javax.servlet.http.*;

import java.io.*;
import java.net.*;
import java.util.*;

import chat.*;

public class ChatRoomServlet extends HttpServlet
{

  public void doGet(HttpServletRequest req, HttpServletResponse res)
            throws IOException, ServletException
  {

    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    ChatRoom room = getRoom(req, res);
    if (room == null)
    {
      return;
    }

    // Check if it's a request for a message list or a form
    String listPar = req.getParameter("list");
    if (listPar != null && listPar.equals("true"))
    {
      writeMessages(out, room, getAvatarName(req));
    }
    else
    {
      writeForm(out);
    }
    out.close();
  }
```

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws IOException, ServletException
{

  res.setContentType("text/html");
  PrintWriter out = res.getWriter();

  ChatRoom room = getRoom(req, res);
  if (room == null)
  {
     return;
  }
  String avatarName = getAvatarName(req);

  // Save message if any
  String msg = req.getParameter("msg");
  if (msg != null && msg.length() != 0)
  {
    room.addChatEntry(new ChatEntry(avatarName, msg));
  }
  writeFrame(out, room);
  out.close();
}

private String getAvatarName(HttpServletRequest req) {
  HttpSession session = req.getSession();
  String avatarName = (String) session.getValue("avatarName");
  if (avatarName == null)
  {
    // Entered a room for the first time?
    avatarName = req.getParameter("avatarName");
    if (avatarName == null || avatarName.length() == 0)
    {
      avatarName = "A spineless spy";
    }
    session.putValue("avatarName", avatarName);
  }
  else
  {
    // Entered a new room with a new name?
    String newName = req.getParameter("avatarName");
    if (newName != null && newName.length() > 0 &&
          !newName.equals(avatarName))
    {
      avatarName = newName;
      session.putValue("avatarName", avatarName);
    }
  }
  return avatarName;
}

private ChatRoom getRoom(HttpServletRequest req, HttpServletResponse res)
        throws IOException
{

  HttpSession session = req.getSession();
  PrintWriter out = res.getWriter();

  String roomName = (String) session.getValue("roomName");
  if (roomName == null)
  {
    // Just entered?
    roomName = req.getParameter("roomName");
```

```
      if (roomName == null || roomName.length() == 0)
      {
        writeError(out, "Room not specified");
        return null;
      }
      session.putValue("roomName", roomName);
    }
    else {
      // Entered a new room?
      String newRoom = req.getParameter("roomName");
      if (newRoom != null && newRoom.length() > 0 &&
        !newRoom.equals(roomName))
      {
        roomName = newRoom;
        session.putValue("roomName", roomName);
      }
    }

    RoomList roomList =
      (RoomList) getServletContext().getAttribute("roomList");
    ChatRoom room = roomList.getRoom(roomName);
    if (room == null)
    {
      writeError(out, "Room " + roomName + " not found");
      return null;
    }
    return room;
  }

  private void writeError(PrintWriter out, String msg)
  {

    out.println("<HTML>");
    out.println("<HEAD><TITLE>Error</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<H1>Error</H1>");
    out.println(msg);
    out.println("</BODY></HTML>");
  }

  private void writeForm(PrintWriter out)
  {
    out.println("<HTML>");
    out.println("<BODY>");
    out.println("<FORM METHOD=POST ACTION=chatRoom TARGET=_top>");

    // Add field for new message
    out.println("<P>Enter your message:<P>");
    out.println("<TEXTAREA NAME=msg COLS=50 ROWS=3 WRAP=HARD>");
    out.println("</TEXTAREA>");

    // Add submit button and hidden field with room name
    out.println("<P><INPUT TYPE=SUBMIT VALUE='Send Message'>");

    // Add an Exit button
    out.println("</FORM>");
    out.println("<FORM ACTION=listRooms METHOD=GET TARGET=_top>");
    out.println("<INPUT TYPE=SUBMIT VALUE=Exit>");
    out.println("</FORM>");

    out.println("</BODY></HTML>");
  }
```

```
  private void writeFrame(PrintWriter out, ChatRoom room)
  {

    out.println("<HTML>");
    out.println("<HEAD><TITLE>" + room.getName() + "</TITLE></HEAD>");
    out.println("<FRAMESET ROWS='60%,30%' BORDER=0 FRAMEBORDER=NO>");
    out.println("<FRAME SRC=chatRoom?list=true NAME=list SCROLLING=AUTO>");
    out.println("<FRAME SRC=chatRoom?list=false NAME=form SCROLLING=AUTO>");
    out.println("<NOFRAMES>");
    out.println("<BODY>");
    out.println("Viewing this page requires a browser capable of displaying frames.");
    out.println("</BODY>");
    out.println("</NOFRAMES>");
    out.println("</FRAMESET>");
    out.println("</HTML>");
  }

  private void writeMessages(PrintWriter out, ChatRoom room,
     String avatarName)
  {

    out.println("<HTML>");
    out.println("<HEAD><META http-equiv=\"refresh\" content=\"5\"></HEAD>");
    out.println("<BODY>");
    out.println("<H1>You're in " + room.getName() + " as " +
      avatarName + "</H1>");

    // List all messages in the room
    Enumeration entries = room.getChatEntries();
    if (!entries.hasMoreElements())
    {
       out.println("<FONT COLOR=RED>There are no messages in this room yet</FONT>");
    }
    while (entries.hasMoreElements())
    {
      ChatEntry entry = (ChatEntry) entries.nextElement();
      String entryName = entry.getAvatarName();
      if (entryName.equals(avatarName))
      {
        out.print("<FONT COLOR=BLUE>");
      }
      out.println(entryName + " : " + entry.getChatMessage() +
          "<BR>");
      if (entryName.equals(avatarName))
      {
        out.print("</FONT>");
      }
    }
    out.println("</BODY></HTML>");
  }
}
```

12.2.6 ErrorPapu.java

```
/**
 * Papu.java
 * ---------
 *
 * Yksinkertainen JavaBean, jonka yhden attribuutin arvoa
 * muutetaan papuTesti.jsp sivun avulla.
 *
```

```
 * Tehty: 9.1.2003, Pekka Kosonen
 *
 */
package pavut;

public class ErrorPapu extends LukuPapu
{
  private boolean virheLippu;

  public ErrorPapu()
  {
    virheLippu = false;
//    luku = 0;
  }

  public int muutaLukua( String Luku )
  {
    int muutos = 0;
    try
    {
      muutos = Integer.parseInt(Luku);
      this.muutaLukua(muutos);
    }catch ( NumberFormatException e )
    { //merkkijonoa ei voitu muuntaa kokonaisluvuksi
      virheLippu = true;
    }
    finally {
      return luku;
    }
  }

  public void virheLippuAlas()
  {
    virheLippu = false;
  }

  public boolean getVirheLippu()
  {
    return virheLippu;
  }
}
```

12.2.7 ExampleFilter.java

```
import java.io.IOException;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.*;

public final class ExampleFilter implements Filter {

  public void init(FilterConfig config) throws javax.servlet.ServletException {
    System.out.println("ExampleFilter: init"); //debuggausta
  }

  public void destroy() {
    System.out.println("ExampleFilter: destroy ");
  }
```

```
public void doFilter(ServletRequest request, ServletResponse response,  FilterChain chain) throws
java.io.IOException, javax.servlet.ServletException
{
    System.err.println("ExampleFilter: ennen doFilter()");
    request.setAttribute("alias", new String("jdbc:odbc:tutorUser"));
    chain.doFilter(request, response); //kontrolli ketjun seuraavalle filtterille, mikäli
                                                    //sellainen on
    System.err.println("---->ExampleFilter: jälkeen doFilter()<----");
  }
}
/*
long before = System.currentTimeMillis();
    chain.doFilter(request, response);
    long after = System.currentTimeMillis();

    String name = "";
    if (request instanceof HttpServletRequest) {
      name = ((HttpServletRequest)request).getRequestURI();
    }
    config.getServletContext().log(name + ": " + (after - before) + "ms");
  }
}


When the server calls init(), the filter saves a reference to the config
in its config variable, which is later used in the doFilter() method
to retrieve the ServletContext. When the server calls doFilter(), the filter
times how long the request handling takes and logs the time once processing
 has completed. This filter nicely demonstrates before- and after-request
 processing. Notice that the parameters to the doFilter() method are not
 HTTP-aware objects, so to call the HTTP-specific getRequestURI() method
 requires a cast of the request to an HttpServletRequest type.
*/
```

12.2.8 HelloTag.java

```
package tutoriaali;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspTagException;
import javax.servlet.jsp.tagext.TagSupport;

public class HelloTag extends TagSupport
{
    public void HelloTag() {

    }

    // Method called when the closing hello tag is encountered
    public int doEndTag() throws JspException {

        try {

            // We use the pageContext to get a Writer
            // We then print the text string Hello
            pageContext.getOut().print("Hello");
        }
        catch (Exception e) {

            throw new JspTagException(e.getMessage());
```

```
        }
        // We want to return SKIP_BODY because this Tag does not support
        // a Tag Body
        return SKIP_BODY;
    }

    public void release() {

        // Call the parent's release to release any resources
        // used by the parent tag.
        // This is just good practice for when you start creating
        // hierarchies of tags.
        super.release();
    }
}
```

12.2.9 HTML.java

```
package util;

public class HTML
{

  public static final int NORMAL   = 0;
  public static final int HEADING  = 1;
  public static final int LINE     = 2;

  public StringBuffer buffer;

  public HTML(String title)
  {
    buffer = new StringBuffer(4096);
    this.buffer.append("<HTML><HEAD><TITLE>");
    this.buffer.append(title);
    this.buffer.append("</TITLE></HEAD><BODY>");
  }

  public void add(int style, String text, boolean linebreak)
  {
    switch(style)
    {
      case NORMAL:
        this.buffer.append(text);
        break;
      case HEADING:
        this.buffer.append("<H1>");
        this.buffer.append(text);
        this.buffer.append("</H1>");
        break;
      case LINE:
        this.buffer.append("<HR>");
        break;
      default:
        break;
    }
    if(linebreak)
    {
      buffer.append("<BR>");
    }
  }
```

```
  public String getPage()
  {
    this.buffer.append("</BODY></HTML>");
    return this.buffer.toString();
  }
}
```

12.2.10 initPara.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class initPara extends HttpServlet {
  private static final String CONTENT_TYPE = "text/html";
  //attribuutit
  private int luvut[];
  private int koko;
  private String otsikko;
  private boolean ok;


  public void init() throws ServletException {
    //luetaan alustusparametrit, asetettu web.xml tiedostossa
    otsikko = this.getServletConfig().getInitParameter("otsikko");
    String k = this.getServletConfig().getInitParameter("koko");

// tai näin:
//  ServletConfig asetukset = getServletConfig();
//  otsikko = asetukset.getInitParameter("otsikko");
//  String k = asetukset.getInitParameter("koko");

    if( otsikko == null || k == null ) //testataan onnistuiko luku
    {
      ok = false;
      return;
    }
    else
      ok = true;

    try {
    koko = Integer.parseInt( k );
    }
    catch (NumberFormatException e )
    { //voisi tehdä jotain järkevämpääkin
      ok = false;
      otsikko = "WebApplication tutoriaali - koko oltava kokonaisluku!";
      return;
    }

    luvut = new int[koko]; //luodaan annetun kokoinen taulukko

    for( int i=0; i < koko; i++ )
      luvut[i] = (int) Math.round( 1 / Math.random() ); //1 / (arvo välillä 0 - 1)
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IO-
Exception {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
```

```
    out.println("<html>");

    out.println("<head><title>" + otsikko + "</title></head>");
    out.println("<body>");

    if ( !ok )
    {
      out.println("<H1>Alustusparameteja ei saatu luettua oikein. Tarkista web.xml tiedosto</H1>");
      out.println("</body></html>");
      return;
    }

    out.println("<H3 align=\"center\">Arvotut luvut</H3>");

    out.println("<HR> <BR> <BR>");

    for( int i=0; i < koko; i++ )
      out.println( luvut[i] + "<BR>");

    out.println("</body></html>");
  }

  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    doGet(request, response); //kutsutaan doGet-metodia
  }

  /**Clean up resources*/
  public void destroy() {
    luvut = null; //ei pakollista
  }

 /* Esimerkkejä erilaisista tavoista lukea init parametrit
    // This method is called by the servlet container just before this servlet
    // is put into service.
    public void init() throws ServletException {
        ok = true;
        otsikko = "Alustettu";

        //getServletContext().log("getinit init");
        // Get the value of an initialization parameter
        //String value = getServletConfig().getInitParameter("param1");

        // Get all available intialization parameters
        java.util.Enumeration enum = getServletConfig().getInitParameterNames();
        for (; enum.hasMoreElements(); ) {
            otsikko = "Oli param";
            // Get the name of the init parameter
            String name = (String)enum.nextElement();
            if( name != null )
              otsikko.concat(name);

            otsikko.concat(":");
            // Get the value of the init parameter
            String value = getServletConfig().getInitParameter(name);
            if( value != null )
              otsikko.concat(value);
            otsikko.concat("<BR>");
        }

        // The int parameters can also be retrieved using the servlet context
        //value = getServletContext().getInitParameter("param1");
    }
```

```
*/

}
```

12.2.11 Jep.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class Jep extends HttpServlet {
  private static final String CONTENT_TYPE = "text/html";
  /**Initialize global variables*/
  public void init() throws ServletException {
  }
  /**Process the HTTP Get request*/
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IO-
Exception {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>WebApplication tutoriaali - lomakkeen tietojen tarkistaminen JavaScrip-
tillä</title></head>");
    out.println("<body>");
    String nimi = request.getParameter("nimi");
    String svuosi = request.getParameter("svuosi");
    String email = request.getParameter("email");

    out.println("<H3><center>Tämä on Jep servletin doGet()-metodin tulostus </center></H3><BR>");
    out.println("<HR><BR>");

    out.println("<table width=\"30%\" border=\"1\" cellspacing=\"20\" " +
    "cellpadding=\"20\" align=\"center\" bgcolor=\"yellow\" >");


    out.println("<tr><th>");
    out.println("Parametri </th> <th><b> arvo </b>");
    out.println("</th></tr>");

    out.println("<tr><td>");
    out.println("nimi </td> <td><b> " + nimi + "</b>");
    out.println("</td></tr>");

    out.println("<tr><td>");
    out.println("syntymävuotesi </td> <td> <b> " + svuosi + "</b>");
    out.println("</td></tr>");


    GregorianCalendar kal = new GregorianCalendar();
    int ika = (kal.getTime().getYear() + 1900) - Integer.parseInt(svuosi);

    out.println("<tr><td>");
    out.println("joten ikäsi on </td> <td> <b> " + ika + "</b>");
    out.println("</td></tr>");

    out.println("<tr><td>");
    out.println("email </td> <td> <b>" + email + "</b>");
    out.println("</td></tr> </table>");
```

```
//    out.println("<a href=\"\" onClick = \"this.href=history.go(-1)\"  > takaisin </a> ");

    out.println("</body></html>");
  }
  /**Clean up resources*/
  public void destroy() {
  }
}
```

12.2.12 LataaContextServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class LataaContextServlet extends HttpServlet {
  private static final String CONTENT_TYPE = "text/html";
  /**Initialize global variables*/
  public void init() throws ServletException {
    // Testaamista varten (initissä System.out ei tule Tomcat ikkunaan!)
    Aliohj.kirjoitaRivi("c:\\temp\\lataa.txt","Init",true);
    //mihinkähän tulostus menee, luultavasti bittien taivaaseen.
    System.out.println("LataaContext: init()");

    //luetaan alustusparametri, katso web.xml
    String titleBegin = this.getServletConfig().getInitParameter("titleBegin");
    System.out.println("titleBegin ==" + titleBegin);

    //asetetaan servletcontextiin kaikkien servlettien ja JSP-sivujen saataville
    this.getServletContext().setAttribute("titleBegin", titleBegin);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IO-
Exception {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();

    out.println("<html>");

    out.println("<body>");
    out.println("<p>Tällä servletillä asetetaan yksi parametri Servlet Contextiin.</p>");
    out.println("</body></html>");
  }

}
```

12.2.13 ListRoomsServlet.java

```
package chat;

import javax.servlet.*;
import javax.servlet.http.*;

import java.io.*;
import java.util.*;
import java.net.URLEncoder;
```

```
import chat.*;

public class ListRoomsServlet extends HttpServlet
{

  public void doGet(HttpServletRequest req, HttpServletResponse res)
          throws IOException
  {

    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    String expand = req.getParameter("expand");
    HttpSession session = req.getSession();
    String avatarName = (String) session.getValue("avatarName");
    if (avatarName == null)
    {
      avatarName = "";
    }

    writePage(out, expand, avatarName);
    out.close();
  }

  private void writePage(PrintWriter out, String expand, String avatarName)
  {

    out.println("<HTML>");
    out.println("<HEAD><TITLE>Chat rooms</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<H1>Chat rooms</H1>");
    out.println("<FORM METHOD=POST ACTION=chatRoom>");

    // Add radio boxes for selecting a room
    out.println("Select the room you like to enter " +
      "or click on a name to see the description:<P>");
    RoomList roomList = (RoomList) getServletContext().getAttribute("roomList");
    Enumeration rooms = roomList.getRooms();
    boolean isFirst = true;
    while (rooms.hasMoreElements())
    {
      ChatRoom room = (ChatRoom) rooms.nextElement();
      String roomName = room.getName();
      out.println("<INPUT TYPE=RADIO NAME=roomName VALUE='" +
        roomName + "'" +
        (isFirst ? " CHECKED" : "") + ">" +
        "<A HREF=listRooms?expand=" +
        URLEncoder.encode(roomName) + ">" +
        roomName + "</A><BR>");
      isFirst = false;

      // Show description if requested
      if (expand != null && expand.equals(roomName))
      {
        out.println("<BLOCKQUOTE>");
        out.println(room.getDescription());
        out.println("</BLOCKQUOTE><BR>");
      }
    }

    // Add a field for the avatar name
    out.println("<P>Enter your name: ");
    out.println("<INPUT NAME=avatarName VALUE='" +
```

```
        avatarName + "' SIZE=30>");

      // Add submit button
      out.println("<P><INPUT TYPE=SUBMIT VALUE='Enter'>");
      out.println("</FORM>");
      out.println("</BODY></HTML>");
    }
}
```

12.2.14 login.java

```
package tutoriaali;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class login extends HttpServlet {

  private String target = "/welcome.jsp";

  private String getUser(String username, String password) {

    // Just return a statice name
    // If this was reality, we would perform a SQL lookup
    return "Bob";
  }

  public void init(ServletConfig config)
    throws ServletException {

    super.init(config);
  }

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

    // If it is a get request forward to doPost()
    doPost(request, response);
  }

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

    // Get the username from the request
    String username = request.getParameter("username");
    // Get the password from the request
    String password = request.getParameter("password");

    String user = getUser(username, password);

    // Add the fake user to the request
    request.setAttribute("USER", user);

    // Forward the request to the target named
    ServletContext context = getServletContext();

    System.err.println("Redirecting to " + target);
```

```
    RequestDispatcher dispatcher =
      context.getRequestDispatcher(target);
    dispatcher.forward(request, response);
  }

  public void destroy() {
  }
}
```

12.2.15 LueContextServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class LueContextServlet extends HttpServlet {
    String otsikonAlku; //
  private static final String CONTENT_TYPE = "text/html";
  /**Initialize global variables*/

  public void init() throws ServletException {
    otsikonAlku = (String) this.getServletContext().getAttribute("titleBegin");
//    testausta varten:
    Aliohj.kirjoitaRivi("c:\\temp\\lue.txt","Init",true);
    System.out.println("LueContext: init()");
  }
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IO-
Exception {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();

    out.println("<html>");

    if ( otsikonAlku != null )
      out.println("<head><title>" + otsikonAlku + " ServletContextin käyttö</title></head>");
    else
      out.println("<head><title>Attribuuttia ei saada luettua servletcontextista</title></head>");

    out.println("<body>");
    out.println("<p>Esimerkissä luetaan servletContextista merkkijono, joka"
    + " asetetaan html sivun otsikoksi.</p>");
    out.println("</body></html>");
  }
  /**Process the HTTP Post request*/
  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    doGet(request, response);
  }
}
```

12.2.16 LukuPapu.java

```
/**
 * Papu.java
 * --------
 *
 * Yksinkertainen JavaBean, jonka yhden attribuutin arvoa
 * muutetaan papuTesti.jsp sivun avulla.
```

```
 *
 * Bean toimii wrapperina yhdelle kokonaisluvulle.
 *
 * Harjoitus 1:
 * Tee bean nimeltä TaulukkoPapu, joka
 * toimii wrapperinä kokonaislukutaulukolle (koko esim. 10).
 *
 * Tee taulukkoPapuTesti.jsp, jolla voidaan lisätä, poistaa sekä tulostaa
 * taulukon arvon.
 *
 * Lisäharjoitus (myöhemmin):
 * Tutki miten saisit pavun luomaan taulukon halutunkokoisena, luominen
 * pitää kuitenkin tapahtua konstruktorissa.
 * (Vihje: config.getInitParameter ja web.xml tiedoston muokkaus)
 *
 * Tehty: 8.1.2003, Pekka Kosonen
 *
 */
package pavut;

public class LukuPapu extends Papu
{
  public int muutaLukua( int Luku )
  {
    this.setLuku( luku + Luku);
    return this.getLuku();
  }

  public int muutaLukua( String Luku )
  {
    if( Luku != null && Luku != "" )
      return muutaLukua( Integer.parseInt(Luku) );
    return luku; //jos ehto epätosi
  }
}
```

12.2.17 LukuPapu2.java

```
/**
 * Papu.java
 * ---------
 *
 * Yksinkertainen JavaBean, jonka yhden attribuutin arvoa
 * muutetaan papuTesti.jsp sivun avulla.
 *
 * Bean toimii wrapperina yhdelle kokonaisluvulle.
 *
 * Harjoitus 1:
 * Tee bean nimeltä TaulukkoPapu, joka
 * toimii wrapperinä kokonaislukutaulukolle (koko esim. 10).
 *
 * Tee taulukkoPapuTesti.jsp, jolla voidaan lisätä, poistaa sekä tulostaa
 * taulukon arvon.
 *
 * Lisäharjoitus (myöhemmin):
 * Tutki miten saisit pavun luomaan taulukon halutunkokoisena, luominen
 * pitää kuitenkin tapahtua konstruktorissa.
 * (Vihje: config.getInitParameter ja web.xml tiedoston muokkaus)
 *
 * Tehty: 8.1.2003, Pekka Kosonen
 *
```

```
 */
package pavut;

public class LukuPapu2 extends Papu
{
  private int luku;

  public LukuPapu2()
  {
    luku = 0;
  }

  public int getLuku()
  {
    return luku;
  }

  public void setLuku( int Luku )
  {
   luku = Luku;
  }

  public int muutaLukua( int Luku )
  {
   luku += Luku;
   return luku;
  }

  public int muutaLukua( String Luku )
  {
   if( Luku != null && Luku != "" )
     return muutaLukua( Integer.parseInt(Luku) );
   return luku; //jos ehto epätosi
  }
}
```

12.2.18 Moi.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class Moi extends HttpServlet {
 private static final String CONTENT_TYPE = "text/html";
 /**Initialize global variables*/
 public void init() throws ServletException {
 }
 /**Process the HTTP Get request*/
 public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IO-
Exception {
   response.setContentType(CONTENT_TYPE);
   PrintWriter out = response.getWriter();
   out.println("<html>");
   out.println("<head><title>Moi</title></head>");
   out.println("<body>");
   out.println("<p>Moi.</p>");
   out.println("</body></html>");
 }
 /**Clean up resources*/
 public void destroy() {
```

```
  }
}
```

12.2.19 MyDate.java

```
/*
 *
 * Copyright 2001 Sun Microsystems, Inc. All Rights Reserved.
 *
 * This software is the proprietary information of Sun Microsystems, Inc.
 * Use is subject to license terms.
 *
 */


import java.text.DateFormat;
import java.util.*;

public class MyDate {
   Date today;
   DateFormat dateFormatter;

   public MyDate() {
      today = new Date();
   }

   public String getDate() {
      return dateFormatter.format(today);
   }

   public void setLocale(Locale l) {
      dateFormatter = DateFormat.getDateInstance(DateFormat.FULL, l);
   }

}
```

12.2.20 MyLocales.java

```
/*
 *
 * Copyright 2001 Sun Microsystems, Inc. All Rights Reserved.
 *
 * This software is the proprietary information of Sun Microsystems, Inc.
 * Use is subject to license terms.
 *
 */

import java.util.*;
import java.text.DateFormat;

public class MyLocales {
   HashMap locales;
   ArrayList localeNames;

   public MyLocales() {
      locales = new HashMap();
      localeNames = new ArrayList();
      Locale list[] = DateFormat.getAvailableLocales();
      for (int i = 0; i < list.length; i++) {
```

```
        locales.put(list[i].getDisplayName(), list[i]);
        localeNames.add(list[i].getDisplayName());
      }
      Collections.sort(localeNames);
    }
    public Collection getLocaleNames() {
      return localeNames;
    }

    public Locale getLocale(String displayName) {
      return (Locale)locales.get(displayName);
    }
}
```

12.2.21 Papu.java

```
/**
 * Papu.java
 * ---------
 * Yläluokka tutoriaalin osassa 2 käytetyille pavuille.
 *
 * Tehty: 8.1.2003, Pekka Kosonen
 *
 */
package pavut;

public class Papu
{
  protected int luku;

  public Papu()
  {
    luku = 1;
  }

  public int getLuku()
  {
    return luku;
  }

  public void setLuku( int Luku )
  {
   luku = Luku;
  }
}
```

12.2.22 RoomList.java

```
package chat;

import java.util.*;

public class RoomList
{
  private Hashtable chatRooms = new Hashtable();

  public void addRoom(ChatRoom room)
  {
    chatRooms.put(room.getName(), room);
```

```
  }

  public ChatRoom getRoom(String name)
  {
    return (ChatRoom) chatRooms.get(name);
  }

  public Enumeration getRooms()
  {
    return chatRooms.elements();
  }

  public void removeRooms(String[] rooms)
  {
    for (int i = 0; i < rooms.length; i++)
    {
      chatRooms.remove(rooms[i]);
    }
  }
}
```

12.2.23 selain.java

```
/**
 * Selain.java
 *
 * JavaBean, jolla saat selville selaimen version (IE / NS4 /NS6).
 * (ei tosin kovin "beanimainen")
 *
 *
 * Tehty: 23.8.2002, Pekka Kosonen
 * Muokattu: 9.1.2003, Pekka Kosonen
 *  - kommentoitu
 */
package pavut;

import javax.servlet.http.HttpServletRequest;


public class Selain{
  private HttpServletRequest request = null;
  private String useragent = null;
  private boolean netEnabled = false;
  private boolean ie = false;
  private boolean ns7 = false;
  private boolean ns6 = false;
  private boolean ns4 = false;

  public Selain() {
  }

  public void setRequest(HttpServletRequest req) {
    request = req;
    useragent = request.getHeader("User-Agent");
    String user = useragent.toLowerCase();
    if(user.indexOf("msie") != -1) //etsitään osamerkkijonoa, -1 jos ei löydy
    {
       ie = true;
    }
    else if(user.indexOf("netscape/7") != -1)
    {
```

```
      ns7 = true;
    }
    else if(user.indexOf("netscape/6") != -1)
    {
      ns6 = true;
    }
    else if(user.indexOf("netscape/4") != -1) //ei testattu toimiiko
    {
      ns4 = true;
    }

    if(user.indexOf(".net clr") != -1)
      netEnabled = true;
  }

  public String getUseragent() {
    return useragent;
  }

  public boolean isNetEnabled() {
    return netEnabled;
  }

  public boolean isIE() {
    return ie;
  }

  public boolean isNS7() {
    return ns7;
  }

  public boolean isNS6() {
    return ns6;
  }

  public boolean isNS4() {
    return ns4;
  }

}
```

12.2.24 SendEmailGroup.java

```
/**
 *
 */

package sendmail;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.mail.*;
import javax.mail.internet.*;


public class SendEmailGroup extends HttpServlet {
  private String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
  private String db = "jdbc:odbc:emailgroup";
```

```
  private String table = "ryhmat";
  private String table2 = "osoitteet";

  private Connection conn = null;
  private Statement stmt = null;
  private ResultSet rs = null;
  private PreparedStatement preparedStmt = null;
  private PreparedStatement preparedStmt2 = null;

//  private String smtpHost = "mail.jypoly.fi"; //TOIMII

  public void init() throws ServletException {
    try {
      connect();
    } catch (Exception e) { log("servletemalgroup ei lähde käyntiin", e); }
  }

  private synchronized void connect() throws Exception {
    if( conn == null || conn.isClosed() )
    {
      Class.forName(driver).newInstance();
      conn = DriverManager.getConnection(db);
      stmt = conn.createStatement();

      preparedStmt = conn.prepareStatement("SELECT ryhmaID FROM " + table +
      " WHERE nimi=(?)");

      preparedStmt2 = conn.prepareStatement("SELECT osoite FROM " + table2 +
      " WHERE ryhmaID=(?)");
    }
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException
  {
    //luetaan ryhman nimi, sitten kaikki osoitteet jotka on ryhmaan l
    //liitetty ja etiäpäin SendMail servletille.
    PrintWriter out = response.getWriter();

    out.println("<html><head><title> Webapplication tutoriaali - Ryhmäpostit</title></head>");
    out.println("<body>");

    String ryhma = request.getParameter("ryhma");
    int ryhmaID = -1;

    if( ryhma == null || ryhma == "")
    {
      out.println("parametrilla ryhma ei ole arvoa. Posteja ei lähetetty.");
      out.close();
      return;
    }

    StringBuffer osoitteet = new StringBuffer("");

    String virheid = "0";
    synchronized (this) {
      try {
        virheid = "1";
        preparedStmt.setString(1, ryhma);
        rs = preparedStmt.executeQuery();

        if ( !rs.next() ) //onko tietuetta
        {
```

```
          out.print("Ei tämän nimistä ryhmää.");
          out.close();
          return;
        }
        virheid = "2";
        ryhmaID = rs.getInt("ryhmaID");

        preparedStmt2.setInt(1, ryhmaID);
        rs = preparedStmt2.executeQuery();

        virheid = "3";
        if ( !rs.next() )
        {
          out.print("Ei osoitteita. Postia ei lähetetty.");
          return;
        }
        virheid = "4";
        //osoitteet pilkulla eroteltuna
        do {
          String osoite = rs.getString("osoite");
          osoitteet.append( osoite );
          osoitteet.append( ",");
        } while ( rs.next() );

      //pilkku pois lopusta
      osoitteet.deleteCharAt( osoitteet.length() - 1 );
      System.out.print("osoitteet:" + osoitteet.toString() );

      String to = osoitteet.toString();

      //sitten asetetaan requestin attribuutti to uusiksi
      request.setAttribute("to", to);

      //lopuksi käytetään SendMail servlettiä
      RequestDispatcher rd =
      this.getServletContext().getRequestDispatcher("/servlet/sendmail.SendMail");

      //incluudataan tulostukset
      rd.include(request, response);

      //katso mitä eroa on kun käytät forwardia.
      //rd.forward(request, response);

      out.print("</body></html>");
    }
    catch ( Exception e )
    {
      out.print("Sähköpostin lähetys ei onnistunut. virheid " + virheid
      + "<BR>" + e.toString() );
    }
  }
}
  public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException
  {
    doGet(request, response);
  }

  public void destroy() {
    try {
      conn.close();
    } catch( SQLException e) { }
    conn = null;
```

```
 }

}
```

12.2.25 SendMail.java

```
package sendmail;

import javax.mail.*;
import javax.mail.internet.*;

import javax.servlet.*;
import javax.servlet.http.*;

import java.io.*;
import java.util.*;

public class SendMail extends HttpServlet {

  private String smtpHost = "mail.jypoly.fi";

  public void doPost( HttpServletRequest request, HttpServletResponse response)
  throws ServletException, java.io.IOException
  {
    String from = request.getParameter("from");
    String to = request.getParameter("to");

    if( to == null )
      to = (String) request.getAttribute("to"); //ryhmäposti käyttää

    String cc = request.getParameter("cc");
    String bcc = request.getParameter("bcc");
    String subject = request.getParameter("subject");
    String text = request.getParameter("text");

    String status = "Sendmail:";
    System.out.println(status + to);

    try {
      // Luodaan JavaMail sessio
      java.util.Properties properties = System.getProperties();
      properties.put("mail.smtp.host", smtpHost);

      Session session = Session.getInstance(properties, null);

      // Kasataan viesti
      MimeMessage message = new MimeMessage(session);


      Address fromAddress = new InternetAddress(from);
      message.setFrom(fromAddress);

      //parsitaan ja asetetaan vastaanottajien osoitteet
      Address[] toAddresses = InternetAddress.parse(to);
      message.setRecipients(Message.RecipientType.TO,toAddresses);


      if( cc != null )
      {
        Address[] ccAddresses = InternetAddress.parse(cc);
        message.setRecipients(Message.RecipientType.CC,ccAddresses);
```

```
        }

        if( bcc != null )
        {
          Address[] bccAddresses = InternetAddress.parse(bcc);
          message.setRecipients(Message.RecipientType.BCC,bccAddresses);
        }
        if( subject != null )
        {
          message.setSubject(subject);
        }
        if( text != null )
        {
          message.setText(text);
        }

        Transport.send(message);

        status = "Viestisi lähettiin.";

        } catch (AddressException e) {
            status = "Virheellinen osoite.<BR>" + e.toString() ;
      System.out.println(status);
        } catch (SendFailedException e) {
            status = "Viestin lähettäminen epäonnistui.<BR>"+ e.toString();
      System.out.println(status);
        } catch (MessagingException e) {
            status = "Viestin lähettäminen epäonnistui.<BR>"+ e.toString();
      System.out.println(status);
        }

      response.setContentType("text/html");

      PrintWriter out = response.getWriter();

      out.println("<html><head><title>"+status + "</title></head>");
      out.println("<body><p>" + status + "</p></body></html>");

      out.close();
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, java.io.IOException
    {
      doPost(request, response);
    }
}
```

12.2.26 ServletApplet.java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;

public class ServletApplet extends Applet implements ActionListener {

  TextField sql;
  TextArea tulos;
```

```
  Button buttonAja;

  public void init() {
    Panel p1 = new Panel();
    p1.setLayout(new FlowLayout(FlowLayout.LEFT));

    p1.add(new Label("SQL:"));

    sql = new TextField("SELECT * FROM roles", 50);
    p1.add(sql);

    buttonAja = new Button("Suorita kysely");
    buttonAja.addActionListener(this);
    p1.add(buttonAja);

    add("North", p1);

    tulos = new TextArea(10, 80);
    add("Center", tulos);
  }

  public void executeQuery() {

    //luetaan SQL
    String qryString = sql.getText();
    tulos.setText("Tulokset:");

    try {
      // servletin URL
//      URL url = new URL("http://192.168.38.64/tutor/servlet/AppletServlet");

     //"oikeasti käytettäessä" korvaa localhost palvelimesi ip:llä
     URL url = new URL("http","localhost","/tutor/servlet/AppletServlet");

      //kaikki HTTP-pyynnöt pitää käsitellä encode metodilla
      String qry = URLEncoder.encode("qry") + "=" +
                   URLEncoder.encode(qryString);

      //asetetaan vaadittavat asetukset
      URLConnection uc = url.openConnection();
      uc.setDoOutput(true);
      uc.setDoInput(true);
      uc.setUseCaches(false);
      uc.setRequestProperty("Content-type",
                   "application/x-www-form-urlencoded");

      // lähetetään pyynnöt palvelimelle (eli ko. servletille)
      DataOutputStream dos = new DataOutputStream(uc.getOutputStream());
      dos.writeBytes(qry);
      dos.flush();
      dos.close();

      InputStreamReader in = new InputStreamReader(uc.getInputStream());

      //tyhjennetään aiemmat tulokset
      //tulos.setText("");

      int chr = in.read();

      //luetaan vastaus palvelimelta, eli mitä servletti tulostaa
      //HttpServletResponse.getWriter() virtaan (eli out)
      while (chr != -1) {
        tulos.append(String.valueOf((char) chr));
```

```
            chr = in.read();
          }
          in.close();

        } catch(MalformedURLException e) {
          tulos.setText(e.toString());
        } catch(IOException e) {
          tulos.setText(e.toString());
        }
    }
    //aina kun jotain tapahtuu niin suoritetaan, kuuntelijana vain buttonAja
    public void actionPerformed(ActionEvent ae) {
      executeQuery();
    }
}
```

12.2.27 SessionInfoServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class SessionInfoServlet extends HttpServlet {
  private static final String CONTENT_TYPE = "text/html";
  /**Initialize global variables*/
  public void init() throws ServletException {
  }
  /**Process the HTTP Get request*/
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IO-
Exception {
    response.setContentType(CONTENT_TYPE);
    response.setHeader("pragma", "no-cache"); //ei välimuistia
    PrintWriter out = response.getWriter();

    HttpSession session = request.getSession(false); //ei luoda sessiota

    out.println("<html>");
    out.println("<head><title>Webapplication tutoriaali - SessionInfoServlet</title></head>");
    out.println("<body>");
    out.println("<p>");

    if( session != null ) //onko sessio olemassa, ei luoda tässä
    {
      out.println("Uusi " + session.isNew()+ "<BR>" );
      out.println( "ID " + session.getId() + "<BR>");
      out.println( "Luotu " + session.getCreationTime() + "<BR>");
      out.println( "Käsitelty viimeksi " + session.getLastAccessedTime() + "<BR>");
      out.println( "Voimassa " + session.getMaxInactiveInterval() + "<BR>");
      Enumeration e = session.getAttributeNames();

      out.println("Sessiolla on seuraavat attribuutit:<BR>" );
      while ( e.hasMoreElements() )
      {
        out.println( e.nextElement()+ "<BR>");
      }
    }
    else
    {
      out.println("<h3>Sessiota ei ole.</h3>");
    }
```

```
    out.println("</p>");
    out.println("</body></html>");
  }
  /**Process the HTTP Post request*/
  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    doGet(request, response);
  }
  /**Clean up resources*/
  public void destroy() {
  }
}
```

12.2.28 ShoppingCart.java

```
/*
  public java.lang.String encodeURL(java.lang.String url)
  Encodes the specified URL by including the session ID in it, or,
  if encoding is not needed, returns the URL unchanged.

  The implementation of this method includes the logic to determine whether the
  session ID needs to be encoded in the URL. For example, if the browser
  supports cookies, or session tracking is turned off, URL encoding is
  unnecessary. For robust session tracking, all URLs emitted by a servlet
  should be run through this method. Otherwise, URL rewriting cannot be used
  with browsers which do not support cookies.Parameters:url - the url to be
  encoded.Returns:the encoded URL if encoding is needed;
  the unchanged URL otherwise.

  Aseta cookies pois päältä, käynnistä selain uudestaan ja testaa

  session.jsp
*/
import java.util.*;

//   Servlet Libraries
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.net.URL;
import util.*;

public class ShoppingCart extends HttpServlet
{

   public void doPost(HttpServletRequest req, HttpServletResponse res)
          throws ServletException, IOException
   {

      // otetaan session objekti, luodaan jos ei ole olemassa
      HttpSession session = req.getSession(true);

      //   Luetaan yksi session attribuutti
      Integer itemCount = (Integer) session.getValue("itemCount");

      //  Jos sessio on uusi (luotu edellä), niin itemCount oliota ei ole
      if (itemCount == null)
      {
        itemCount = new Integer(0);
      }
```

```
    PrintWriter out = res.getWriter();
    res.setContentType("text/html");


    String[] itemsSelected;
    String itemName;
    itemsSelected = req.getParameterValues("item");

    //Jos oltiin valittu ostoksia, niin lisätään ne sessioon
    if (itemsSelected != null)
    {
      for (int i=0 ; i < itemsSelected.length ; i++)
      {
        itemName = itemsSelected[i];
        itemCount = new Integer(itemCount.intValue() + 1);

        session.putValue("Item" + itemCount, itemName);

        session.putValue("itemCount", itemCount);
      }
    }

    HTML h = new HTML("Shopping Cart Contents");
    h.add(HTML.HEADING, "Items currently in your cart", false);
    h.add(HTML.LINE, "", false);

    for(int i = 1; i <= itemCount.intValue(); i++)
    {
      String item = (String) session.getValue("Item" + i);
      h.add(HTML.NORMAL, item, true);

    }

    h.add(HTML.LINE, "", false);
    h.add(HTML.NORMAL, "<A HREF=\"../ShoppingCart.html\">Back to the shop</A>", true);

    //URL url = new URL("http", "localhost", "tutor\\session.jsp");
    //muuta suhteelliseksi
    String URL = "../session.jsp";
    String encodedURL = res.encodeURL( URL);


    h.add(HTML.NORMAL, "<A HREF=\"" + encodedURL + "\">sessiotiedot JSP </A>" , true );
    h.add(HTML.NORMAL, "<A HREF=\"" + URL + "\">sessiotiedot JSP(ei encodedURL)</A>" , true );

    URL = "SessionInfoServlet";
    encodedURL = res.encodeURL( URL );

    h.add(HTML.NORMAL, "<A HREF=\"" + encodedURL + "\">sessiotiedot servlet</A>" , true );
    h.add(HTML.NORMAL, "<A HREF=\"" + URL + "\">sessiotiedot servlet(ei encodedURL)</A>" , true );
    out.println(h.getPage());
    out.close();
  }
}
```

12.2.29 StringUtils.java

```
package chat;

/**
```

```
 * This class includes a static method for replacing a substring
 * with another substring in a String. Even though this is a very
 * common operation neither the String nor StringBuffer class
 * provides such a method.
 *
 * You can use the StringUtils class in your application like this:
 * import com.wrox.pssjp.util.StringUtils;
 *
 *   public class MyApp {
 *
 *                   public void myMethod() {
 *                                   String hello = "Hello World!";
 *                                   hello = StringUtils.replaceInString(hello, "World", "Java");
 *                   }
 *   }
 *
 * @author Hans Bergsten, Gefion software (www.gefionsoftware.com)
 */
public class StringUtils {

   /**
    * Returns a String with all occurrences of the String from
    * replaced by the String to.
    *
    * @return The new String
    */
   public static String replaceInString(String in, String from, String to) {
      StringBuffer sb = new StringBuffer(in.length() * 2);
      String posString = in.toLowerCase();
      String cmpString = from.toLowerCase();
      int i = 0;
      boolean done = false;
      while (i < in.length() && !done) {
         int start = posString.indexOf(cmpString, i);
         if (start == -1) {
            done = true;
         }
         else {
            sb.append(in.substring(i, start) + to);
            i = start + from.length();
         }
      }
      if (i < in.length()) {
         sb.append(in.substring(i));
      }
      return sb.toString();
   }
}
```

12.2.30 TitleTag.java

```
package tagit;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;

public class TitleTag extends TagSupport {
  String titleEnd = null;

  public int doStartTag() {
```

```
    try {
      JspWriter out = pageContext.getOut();
      out.print("<title> Webapplication tutoriaali ");

      if( titleEnd != null)
        out.print("- " + titleEnd);

      out.print("</title>");
    } catch(IOException ioe) {
      System.out.println("Error in ExampleTag: " + ioe);
    }
    return(SKIP_BODY);
  }

  public void setTitleEnd(String jono)
  {
    titleEnd = jono;
  }
}
```

12.2.31 ViallinenPapu.java

```
/**
 * ViallinenPapu.java
 * ------------------
 *
 * Papu heittää poikkeuksen vika metodia kutsuttaessa
 *
 * Tehty: 9.1.2003, Pekka Kosonen
 */



package pavut;

public class ViallinenPapu {

  public ViallinenPapu()
  {

  }

  public void vika() throws Exception
  {
    Exception e = new Exception("Papu on viallinen");
    throw e;
  }
}
```