Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Eemeli Pyöttiö

Opinnäytetyö

Java EE -sovelluksen resurssien suojaus

Case: Toyme Lab Oy

Työn ohjaaja Tieto- ja viestintäteknologian lehtori, FL Paula Hietala
Työn tilaaja Toyme Lab Oy
Tampere 6/2009

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä	Eemeli Pyöttiö
Työn nimi	Java EE -sovelluksen resurssien suojaus Case: Toyme Lab Oy
Sivumäärä	43
Valmistumisaika	**K**esäkuu, 2009
Työn ohjaaja	Paula Hietala
Työn tilaaja	Toyme Lab Oy

Tiivistelmä

Aloitin vuoden 2009 tammikuussa opintoihini liittyvän harjoittelujakson Toyme Lab Oy:ssä. Samoihin aikoihin alkoi opinnäytetyön tekeminen olla ajankohtaista. Toyme Lab Oy:n päätuotteesta Toyme-aloitepalvelusta oltiin samaan aikaan tekemässä uutta versiota nimeltään Toyme2. Tähän uuteen ohjelmistoversioon liittyen löytyi myös opinnäytetyöni aihe, joka liittyy Java EE -sovelluksen resurssien suojaamiseen. Suojaaminen käsittää käyttäjän tunnistamisen sekä sovelluksen sisällön jakamisen käyttäjälle asetettujen oikeuksien mukaisesti.

Keskityn työssäni Java-kielellä ohjelmoidun sovelluksen suojaamiseen Apache Tomcat-sovelluspalvelimella. Työssäni tutustuin Javan sekä Tomcatin tietoturvaominaisuuksiin. Kartoitin erilaisia mahdollisuuksia toteuttaa sovelluksen resurssien suojaaminen sekä jakaminen eri käyttäjien kesken.

Tutkimustyön tuloksena päädyin toteuttamaan Toyme2-sovelluksen resurssien suojaamisen Javan JAAS-sovelluskehyksen sekä Tomcatin JAASRealm-toiminnon avulla. Toyme2-sovelluksen sisältöä voidaan nyt jakaa käyttäjälle tunnistamisen yhteydessä asetettavien käyttäjäroolien mukaisesti.

Asiasanat Java EE, Tomcat, Tietoturva, Resurssien hallinta, JAAS

TAMK University of Applied Sciences
Business Information Systems

Writer	Eemeli Pyöttiö
Thesis	Securing Java EE Application Case: Toyme Lab Ltd
Pages	43
Graduationt time	**J**une, 2009
Thesis Supervisor	Paula Hietala
Co-operating Company	Toyme Lab Ltd

Abstract

In January 2009 I started working in Toyme Lab Ltd. as a trainee. At the same time, writing my Thesis became topical. Toyme Lab Ltd. was developing a new version of their main product called Toyme, an Internet based system for processing ideas and suggestions. The new version would be called Toyme2. Related to this new version of Toyme, I found the topic for my thesis: securing the Java EE application. In this thesis, securing means authentication and authorization of the user using the application.

I concentrated on studying various techniques to secure Java EE application on Apache Tomcat application server. As a result, I ended up programming the security for Toyme2 using the JAAS framework and Tomcat's JAASRealm. Now the resourses of the Toyme2 application may be shared for the users by using the user roles set in the authentication process.

Keywords Java EE, Tomcat, Security, Access control, JAAS

Sisällysluettelo

Sanastoa

Apache Tomcat	Ilmainen www-sovelluspalvelin, jolla voidaan julkaista Java-sovelluksia.
CSS	CSS tyylitiedostojen avulla voidaan luoda HTML- ja XHTML-sivujen ulkoasu.
HTML	Hyper Text Markup Language on kieli, jonka avulla voidaan ohjelmoida staattisia www-sivuja.
JAAS	Java Authentication and Authorization Service. Sovelluskehys, jonka avulla voidaan toteuttaa Java-sovelluksen tietoturva.
Java Doc	Java Doc on standardi Java-sovelluksen dokumetointiin.
Java EE	Standardi Java-ohjelmointikielellä tehtyjen monitasoisten web-sovellusten suunnitteluun.
JDBC	Java Database Connector on rajapinta, jolla voidaan hallita tietokantayhteyksiä Java-sovelluksessa.
JNDI	JNDI on Javan rajapinta, jolla objekti voidaan hakea nimen perusteella.
JSP	Java Server Pages on standardi, jonka avulla voidaan luoda dynaamisia www-sivuja Java-kielellä.
JVM	Java Virtual Machine on on virtuaalinen kone, joka hoitaa käännettyjen Java-sovellusten ajamisen.
LDAP	LDAP on palvelin, joka toimii tietovarastona.
MD5, MD2, SHA	MD5, MD2 ja SHA ovat algoritmeja, joiden avulla selkokielinen teksti voidaan tiivistää muotoon, josta ei ole mahdollista saada selville alkuperäistä tietoa.
MySql	Ilmainen SQL-relaatiotietokannan hallintajärjestelmä.
Pääsynvalvonta	Käyttäjän pääsyn rajoittamista tietojärjestelmän tiettyihin osiin *(Access Control)*.

Realm	Realm on tietovarasto, esimerkiksi relaatiotietokanta, joka sisältää käyttäjän todentamiseen tarvittavat tiedot.
SaaS	*Software As A Service* on malli, jossa sovellus myydään palveluna.
Servlet	Servlet on Java-luokka, jonka avulla voidaan käsitellä HTML-pyyntöjä.
SSO	Single Sign On on Java EE -sovelluspalvelimen ominaisuus, jolla voidaan mahdollistaa useiden erillisten Java web-sovellusten käyttämisen siten, että palveluihin tarvitsee kirjautua vain kerran.
Sun Microsystems	Yritys, joka kehittää Java-ohjelmointikieltä sekä MySql-tietokantaa.
Säiliö	Säiliö (container) on sovelluspalvelimen tuottama palvelu, joka huolehtii Java-kielellä ohjelmoitujen komponenttien toiminnasta.
UML	UML toimintojen ja sovellusten suunnitteluun kehitetty standardi.
UTF-8	UTF-8 on merkistökoodaus, joka sisältää skandinaaviset merkistöt.
XHTML	XHTML on XML-pohjainen kieli, jolla voidaan luoda staattisia internetsivuja.

1 Johdanto

Palvelinympäristöjen kehitys ja Internet-yhteyksien nopeuden kasvu on mahdollistanut 2000-luvulla yhä monipuolisempien interaktiivisten palveluiden kehittämisen. Erilaiset yhteisölliset palvelut, kuten Facebook ovat kuluttajille jo arkipäivää. Myös erilaisia liiketoimintasovelluksia, kuten esimerkiksi pankkipalvelut, ohjelmoidaan yhä useammin toimimaan palvelinympäristössä. Tällöin sovellusta on mahdollista käyttää missä tahansa, kunhan käytössä on Internet-yhteys, ja voidaan siirtyä toimintamalliin, jossa sovellus myydään palveluna perinteisen lisenssikaupan sijaan.

SaaS-toimintamallissa *(Software As A Service)* myytävä sovellus toimii palveluntarjoajan omalla www-palvelimella ja sovellusta käytetään yleisimmin Internet-selaimella. Tästä toimintamallista on monia hyötyjä kaikille osapuolille. Sovelluksen käyttöönotto nopeutuu, sovelluksen ylläpito ja päivittäminen on keskitettynä nopeaa ja sovelluksen myyminen vaatii vähemmän resursseja. Asiakas taas maksaa kuukausittain maksua toimivasta sovelluksesta. Näin ollen suuret kertaluontoiset korvaukset jäävät pois, ja pääomaa ei tarvitse sitoa kalustoon. Uusien lisenssien ostamisesta ei myöskään seuraa työläitä ja aikaa vieviä asennustöitä.

Toisaalta SaaS-mallissa sovelluksen tietoturvaan on kiinnitettävä huomiota, jotta samalla palvelimella sijaitsevat eri yhteisöjen mahdollisesti arkaluontoiset tiedot eivät joudu vääriin käsiin. Tämä on erittäin tärkeää palveluna myytävän sovelluksen uskottavuuden kannalta. Jos sovellus menettää kerran luotettavuutensa, voi asiakkaiden luottamuksen palauttaminen olla vaikeaa. Lisäksi mahdollisesti arkaluontoisten tietojen vuotaminen voisi aiheuttaa asiakkaalle taloudellisia menetyksiä ja pahimmassa tapauksessa johtaa jopa oikeustoimiin.

Toyme Lab Oy on vuodesta 2001 kehittänyt Toyme-aloitepalvelua, jonka avulla voidaan kerätä niin sanottua hiljaista tietoa ja jalostaa sitä organisaation tietopääomaksi. Hiljainen tieto voi sisältää idean, palautteen tai vaikkapa kokemuksen kautta saavutetun taitotiedon. Tällaisen tiedon valjastaminen auttaa yritystä kehittämään liiketoimintaansa ja lisäksi tietoa voidaan hyödyntää myöhemmin, vaikka työntekijä olisi poistunut yrityksen palveluksesta.

Ohjelmistosta on nyt tekeillä uusi versio, joka mahdollistaa tuotteen tarjoamisen asiakkaiden käyttöön tehokkaammin SaaS-toimintamallin mukaisesti ympäri maailmaa. Sovellus ohjelmoidaan Java-ohjelmointikielellä ja sitä käytetään Internet-selaimella. Opinnäytetyössäni selvitän eri mahdollisuuksia toteuttaa sovellukseen luotettava resurssien suojaaminen, jonka avulla ohjelmiston sisältöä voidaan jakaa eri käyttäjien kesken.

Työn lukijalta odotetaan perustietämystä oliopohjaisesta Java-ohjelmoinnista sekä monitasoisten web-sovellusten suunnittelusta. Sen sijaan Java EE -standardiin liittyvät termistöt käydään lyhyesti läpi.

2 Tavoitteet, metodit ja materiaali

Toimeksiantaja

Työn toimeksiantajana toimii Toyme Lab Oy, joka on vuonna 2000 perustettu ohjelmistoalan yritys. Sen päätuote on Toyme-aloitepalvelu, jonka avulla asiakasyritys voi seurata mitä yrityksen sisällä tapahtuu ja kerätä työntekijöiltä aloitteita, joilla parantaa yrityksen toimintaa. Aloitepalvelun lisäksi Toyme-sovelluksesta on räätälöity muun muassa reklamaatiojärjestelmiä ja muita raportointityökaluja. Toyme Lab Oy:n suurimpia asiakkaita ovat Lassila & Tikanoja, Tampereen Sairaanhoitopiiri ja Metso Power. Yrityksen liikevaihto vuonna 2008 oli noin 170 000 Euroa.

Lähtökohta

Toyme2-sovelluksesta on työn aloittamishetkellä valmiina osa sovelluslogiikkaa, kuten tietokantayhteyksien hallinta sekä alustava käyttöliittymä, mikä mahdollistaa sovelluksen testaamisen.

Itselläni on kokemusta sekä teoriatason tietoa Java EE -sovellusten tekemisestä sekä yleisesti www-sovelluksen tietoturvan toteuttamisesta. Joudun kuitenkin tutkimaan paljon Javan tietoturvaominaisuuksia, koska niihin en ole aikaisemmin perehtynyt.

Työn tavoitteet

Työn tavoitteena on toteuttaa resurssien suojaaminen Toyme2-sovellusversioon. Suojaaminen käsittää käyttäjän tunnistamisen sekä käyttäjien pääsynvalvonnan sovelluksen sisältöön ja toiminnallisuuksiin. Toyme2-sovellus ohjelmoidaan Java-kielellä ja sovelluksessa käytetään Java EE -standardin mukaisia komponentteja. Sovellus julkaistaan Tomcat-sovelluspalvelimella, joten keskityn työssäni tutkimaan tekniikoita, joita tämä palvelin tukee. Tämä työ on minulle hieno tilaisuus syventää tietämystäni Java-ohjelmoinnista web-ympäristössä sekä tietoturvasta.

Tutkimusmetodit ja lähdeaineisto

Työssä käytetään kvalitatiivisia tutkimusmenetelmiä. Lähdemateriaalia läpi käymällä ja analysoimalla pyrin valitsemaan parhaan tekniikan, jolla resurssien suojaaminen voidaan tässä tapauksessa toteuttaa.

Suurin osa käytettävissä olevasta lähdeaineistosta löytyy Internetistä. Internet-aineistoa tutkiessani olen kiinnittänyt erityistä huomiota lähteen alkuperään ja sen luotettavuuteen. Kirjastoista painettuna löytyvä, lähinnä web-ohjelmointiin sekä tietoturvaan liittyvä materiaali on usein vanhaa, joten siihen on suhtauduttava siitä syystä tietyllä varauksella. Työn kannalta merkittävimmät aineistot löytyvät Sun Microsystemsin Internet-sivuilta, jossa viralliset Java-dokumentaatiot ja oppaat sijaitsevat. Lisäksi Apache Tomcat -sovelluspalvelimen kotisivuilta löytyvät spesifikaatiot ovat tärkeitä tiedonlähteitä.

Internet-aineiston lisäksi materiaalia on kerätty Tampereen ammattikorkeakoulun sekä Tampereen seudun kirjastoista. Näistä löytyvä materiaali vastaa lähinnä Java EE -sovelluksen yleistä rakennetta koskeviin kysymyksiin.

Niin sanotun teknisen lähdeaineiston lisäksi olen pyrkinyt etsimään Javan tietoturvaan liittyviä artikkeleita alan lehdistä sekä lehtien Internet-julkaisuista vahvistamaan omaa näkemystäni eri tekniikoiden toimivuudesta ja mahdollisista puutteista.

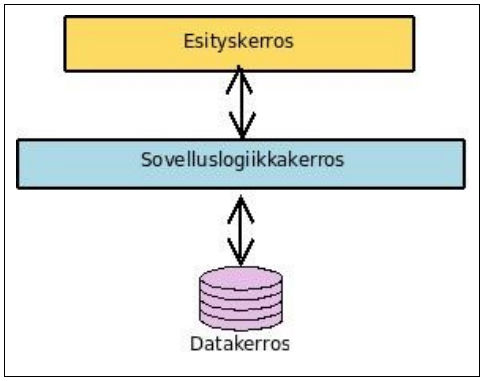
3 Johdatus Java EE -standardiin

Java EE *(Enterprise Edition)* on standardikokoelma, joka määrittelee Java-kielellä ohjelmoidun web-sovelluksen rakenteen, ja sen kuinka sovelluksen sisältämät komponentit kommunikoivat sovelluspalvelimen kanssa. Java EE:n avulla voidaan luoda monitasoisia hajautettuja web-sovelluksia, jotka ovat siirreltävissä Java EE -stantardin mukaiselta sovelluspalvelimelta toiselle ilman muutoksia.

Sovelluksen tasot

Sovelluksen arkkitehtuuri voi koostua ***n*** määrästä tasoja, mutta yleisimmin web-sovellukset rakennetaan kolmen *(kuva 1)* eri tason päälle (Wutka 2001, 8):

- Esityskerroksessa sijaitsee käyttöliittymä. Esimerkiksi loppukäyttäjälle näkyviin generoituvat HTML-sivut sijaitsevat tällä tasolla.
- Sovelluslogiikkakerros sisältää sovelluksen dataa käsittelevät toiminnat, eli toisin sanoen ohjelmakoodin, joka käsittelee tietoa.
- Datakerros käsittelee tallennettavaa dataa/tietoa. Yleensä tässä kerroksessa sijaitsee tietokanta ja sitä käsittelevät rajapinnat.

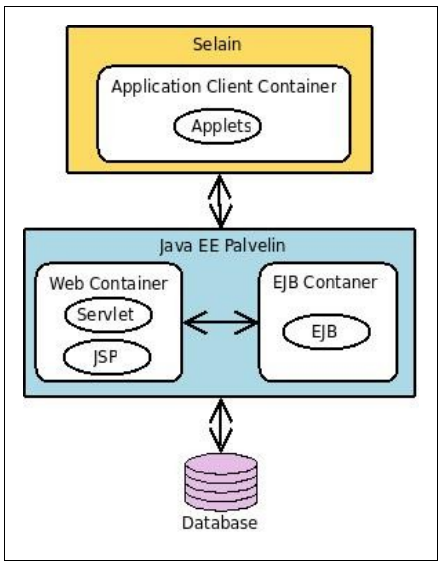


Kuva 1 *Web-sovelluksen tasot.*

Säiliöt (Container)

Säiliö on sovelluspalvelimen tuottama palvelu, joka hallinnoi web-sovelluksen eri komponentteja, kuten JSP-sivuja. Säiliöt huolehtivat komponenttien elinkaarista, sekä komponenttien ja palvelimen välisestä vuorovaikutuksesta *(kuva 2)*. (J2EE Containers n.d.)

Java EE -sovelluspalvelin toteuttaa kaksi eri säiliötä: Web Container sekä EJB Container. Web Container säilöö ja käsittelee sovellukseen liittyvät Servletit sekä JSP-sivut. EJB Container käsittelee nimensä mukaisesti Enterprise Java Beanseja. Appletteja käsittelevä säiliö sijaitsee sovellusta käyttävän asiakkaan selaimessa. (J2EE Containers n.d.)



Kuva 2 *Internet-selaimen ja Java EE -sovelluspalvelimen tarjoamat säiliöt.*

Java Appletit ja Servletit

Java Applet on pieni Java-sovellus, joka lähetetään sovelluspalvelimelta selaimelle, missä se ajetaan. Appletit vaativat toimiakseen selaimelle asennetun Java-laajennuksen. (Vesterholm & Kyppö 2001, 227)

Servlet on pieni Java-sovellus joka toimii, toisin kuin Applet, sovelluspalvelimella. Näin ollen Servlettien ajaminen ei vaadi laajennuksien asentamista selaimelle, mutta vaatii toisaalta sovelluspalvelimen, josta löytyy säiliö *(Web Container)* Servlettien käsittelyä varten. Servlettejä käytetään dynaamisen sisällön generoimiseen staattiselle HTML-sivulle. Servletit perivät HttpServlet luokan ja niiden avulla voidaan käsitellä selaimelta tulevia GET- ja POST-pyyntöjä.(Vesterholm & Kyppö 2001, 477)

Java Beans ja Enterprise Java Beans

Beansit eli pavut ovat Java-luokkia, joita käytetään usein uudelleenkäytettävän tiedon säilömiseen. Hyvänä esimerkkinä toimii käyttöliittymäluokkien toteuttaminen papujen avulla. Pavut rakennetaan tietyn standardin mukaan ja ne toteuttavat *Serializable* rajapinnan. (Westerholm & Kyppö 2001, 390)

Enterprise Java Beansit ovat papuja, jotka käsitellään Java EE -sovelluspalvelimella EJB-säiliössä. EJB-pavut jaetaan kahteen ryhmään. Session Beans sisältää sovelluslogiikkaan liittyviä toimintoja. Yhdellä asiakkaalla voi olla vain yksi Session Bean käytössä. Entity Beans on papu, johon voidaan tallentaa tietyn yksikön tietoja, kuten esimerkiksi asiakkaan yhteystiedot. Entity Beansin sisältämä tieto tallennetaan useimmiten tietokantaan. Suuremmissa sovelluksissa EJB-komponentteja käytetään monesti myös sovelluslogiikan sekä tietokantayhteyksien toteuttamiseen. (Niskanen , Kontio, Vierimaa 2000, 643)

JSP

JSP (*Java Server Pages)* on teknologia, joka mahdollistaa dynaamisten sivujen generoinnin Java-kielellä. Sivuilla generoitavaan staattiseen, tavallisesti HTML-koodiin voidaan sisällyttää myös ”skriptejä” Java-kielellä. Koodi käännetään palvelimella, ja selaimeen palautetaan staattinen sisältö. JSP-sivun ohjelmakoodin ulkonäkö ja toiminta muistuttaa hyvin paljon esimerkiksi PHP-sivua *(Kuva 3)*. (Ahonen, Hämeen-Anttila, Åstrand 2003, 21)

```
<html>
    <body>
        <%
        String name = "Eemeli";
        int age = 26;
        %>
        <h1>Hello World!</h1>
        <p>
            Tervehdys <%=name%>!<br />
            Ikäsi on <%=age%>.
        </p>
    </body>
</html>
```

Kuva 3 *Yksinkertaistettu JSP-sivu joka tulostaa tervehdyksen.*

web.xml

Xml on tärkeä osa nykyaikaista tiedonkäsittelyä. Ennen tietoa saatettiin käsitellä lukemattomilla eri tavoilla eri tyyppisillä tiedostomuodoilla, mutta Xml loi mahdollisuuden yhtenäistää tekstimuotoisen tiedon tallentamisen tiedostoon. Java EE -sovelluksessa xml-tiedostoja käytetään sovelluksen rakenteen kuvaamiseen ja kommunikoimiseen palvelimen kanssa. Tiedosto *web.xml* toimii Java EE -sovelluksen asetustiedostona, josta löytyy määriteltynä esimerkiksi Servlettien nimet ja sijainnit sekä tietoturvaan ja tietokantayhteyksiin liittyviä määrityksiä. **(**Ahonen, ym. 2003, 146)

Sovelluspalvelin

Appletteja lukuun ottamatta Java-sovelluksen ajaminen www-ympäristössä vaatii siihen tarkoitukseen suunnitellun sovelluspalvelimen. Tavallinen www-palvelin ei tähän tarkoitukseen riitä. Markkinoilla on useita eri hintaisia vaihtoehtoja. Ehkä suosituin on ilmainen Apache Tomcat, jota myös tämän työn puitteissa käytetään. Muita yleisesti käytössä olevia palvelimia on esimerkiksi IBM WebSphere ja BEA WebLogic.
(Ahonen, ym. 2003, 29-30)

Apache Tomcat

Apache Tomcat on ehkä käytetyin sovelluspalvelin Java web-sovelluksille, mutta se ei kuitenkaan ole täysimittainen Java EE -palvelin. Tomcat toteuttaa ainoastaan Servlet-säiliön, jolloin siinä on mahdollista ajaa Servlettejä sekä JSP-sivuja. EJB-komponentteja siinä ei sen sijaan voi ajaa. Tomcatin kanssa on kuitenkin mahdollista käyttää erillistä EJB-säiliötä, jolloin siitä saadaan täysimittainen Java EE -palvelin.
(Kuha 2008, 376)

4 Tietoturva

Käyttäjät ja roolit (Users & roles)

Java EE -sovelluksen tietoturva ja resurssien jako perustuu rooleihin. Jokaiselle yksittäiselle käyttäjälle (user) annetaan rooli (role). Rooli määrittelee sen, mihin sovelluksen osiin käyttäjällä on oikeus. Rooleja ei ole ennalta määritelty vaan niitä voidaan nimetä ja luoda sovelluksen tarpeiden mukaan. Roolit määritellään sovelluksen *web.xml*-tiedostossa *(kuva 4)*. (Ahonen, ym. 2003, 241)

```
<security-role>
    <role-name>admin</role-name>
</security-role>
<security-role>
    <description/>
    <role-name>author</role-name>
</security-role>
<security-role>
    <description/>
    <role-name>public</role-name>
</security-role>
```

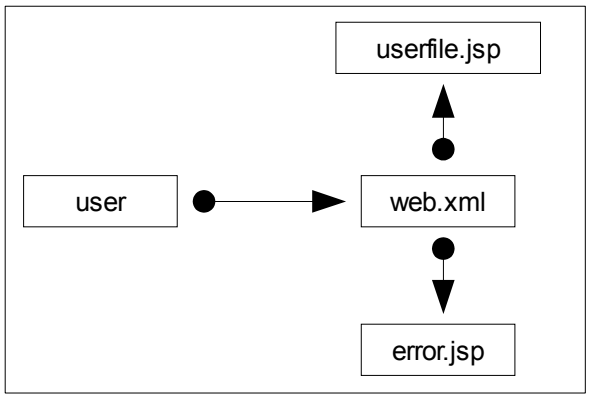
Kuva 4 Esimerkissä sovellukselle on määritelty 3 eri roolia.

Määrittelevä tietoturva (Declarative Security)

Määrittelevä tietoturva tarkoittaa sitä, että ohjelmiston resursseja suojataan kansiotasolla. Määritykset tehdään sovelluksen ***web.xml***-tiedostoon, joten itse ohjelmakoodiin ei tarvitse koskea. (Ahonen, ym. 2003, 232)

Voidaan esimerkiksi ajatella, että sovelluksella on kaksi eri käyttäjätasoa eli roolia: user ja admin. Sovelluksen sisältöä sijaitsee kansioissa *adminfile* ja *userfile*. Tiedostossa w*eb.xml* voidaan määritellä käyttäjille pääsy tiedostoihin siten että admin-käyttäjillä on pääsy *adminfile*-kansioon ja vastaavasti user-käyttäjillä *userfile*-kansioon. Käyttäjällä siis joko on pääsy sovelluksen tiettyyn resurssiin tai ei. Tietyn resurssin, esimerkiksi JSP-sivun sisällä, voidaan sisältöä jakaa ohjelmoitavan tietoturvan avulla.

Kuvan 5 kaaviossa käyttäjä (user) lähettää palvelimelle pyynnön päästä sivulle *userfile.jsp*. Palvelin tarkistaa *web.xml* -tiedostosta, onko kyseiselle sivulle vapaa pääsy vai onko pääsy sallittu vain tietyille rooleille. Jos käyttäjälle määritellyllä roolilla on oikeus sivun sisältöön, sivu näytetään. Muussa tapauksessa käyttäjä ohjataan joko kirjautumissivulle tai selaimeen tulostetaan virheilmoitus.



Kuva 5 *Käyttäjän oikeudet tiettyyn resurssiin tarkistetaan web.xml-tiedostossa.*

Servlettien ja JSP-sivujen käyttöoikeus voidaan antaa myös vain tietyille rooleille. *(kuva 6)*. Elementin **<run-as>** sisällä voidaan määritellä rooli, jolla Servlettiä tai JSP-sivua saa käyttää. Elementin **<security-role-ref>** avulla voidaan tehdä viittaus rooliin. Viittaukseen voidaan sisällyttää useampiakin rooleja. Viittauksista on erityisesti hyötyä ohjelmoitaessa tietoturvaa, jolloin ohjelmakoodin sisällä olevat roolien nimet eivät muutu. Muutokset viittauksiin tehdään *web-xml*:n sisällä. (Ahonen, ym. 2003, 243)

```
<servlet>
...
    <run-as>
        <role-name>admin</role-name>
    </run-as>

    <security-role-ref>
        <description/>
        <role-name>admin</role-name>
        <role-link>admin-link</role-link>
    </security-role-ref>
...
</servlet>
```

Kuva 6 *Servletin käyttöön voidaan antaa oikeudet vain tietyille rooleille.*

Ohjelmoitava tietoturva (Programmatic Security)

Nimensä mukaisesti ohjelmoitava tietoturva ohjelmoidaan sovelluslogiikan sisään. Ohjelmakoodia ajettaessa voidaan tarkistaa, onko kyseisellä käyttäjällä oikeus kyseiseen toimintoon. Tämä tekniikka lisää joustavuutta sovelluksen rakenteessa. (Ahonen, ym. 2003, 232)

Esimerkiksi navigointipalkin sisältävällä JSP-sivulla voidaan ohjelmallisesti tarkistaa, mikä sivun ladanneen käyttäjän rooli on *(kuva 7)*. Tämän roolin mukaisesti voidaan käyttäjälle tulostaa tai olla tulostamatta joitain osia sivusta. Näin kaikille rooleille ei tarvitse tehdä erikseen omaa versiota navigointipalkista. Tämä ei tietenkään yksinään riitä suojaamaan esimerkin admin-sivua. Näin kuitenkin vältytään turhilta virheilmoituksilta käyttäjän selaimessa, kun sivulle johtavan linkin painaminen on estetty, jos siihen ei ole oikeutta.

```
<%
if (request.isUserInRole("admin")) {
%>
<a href="/adminpage.jsp">Hallinta</a>
<% }%>
<a href="/index.jsp">Etusivu</a>
<a href="/news.jsp">Uutiset</a>
```

***Kuva 7** Esimerkissä selaimeen tulostetaan linkkejä. Jos käyttäjän rooli on admin, tulostetaan selaimeen myös linkki hallintasivulle.*

Metodit, joita tarkistukseen voidaan käyttää, sisältyvät **javax.servlet.http.HttpServletRequest**-rajapintaan. Esimerkissäkin käytetty **isUserInRole(**String role**)**-metodi palauttaa boolean arvon sen mukaan, onko käyttäjälle määritelty parametrina annettu rooli. Metodi **getRemoteUser()** palauttaa käyttäjän nimen. Jos käyttäjälle on liitetty käyttäjää kuvaava Principal-luokan olio *(katso luku 7 JAAS)*, voidaan se palauttaa metodilla **getUserPrincipal()**. (Interface HttpServletRequest n.d.)

4.2 Todentaminen (Authentication)

Ennen kuin sovelluksen suojattuja resursseja voidaan jakaa käyttäjälle, tulee tämä tunnistaa. Yleisin käytössä oleva tunnistusmetodi lienee käyttäjätunnus-salasana yhdistelmä. Useimmiten tunnus ja siihen liittyvä salasana on tallennettu tietokantaan, josta käyttäjän kirjautuessa sivulle haetaan tunnukseen liittyvä salasana ja verrataan sitä käyttäjän antamaan syötteeseen. Muitakin vaihtoehtoja on kuitenkin olemassa.

4.2.1 Käyttäjätietojen säilöminen (Security Realms)

Security Realm on tietovarasto, johon on tallennettu käyttäjiä koskevat tiedot, kuten käyttäjätunnus ja salasana. Jokaiselle käyttäjälle on myös määritelty rooli, jonka mukaan sisältöä jaetaan. (Apache Tomcat n.d. c)

Tomcat Realms

Tomcatissa valmiiksi määriteltyjä Realmeja on useita. Nämä Realmit sisältävät luokan, joka käsittelee valitun tietovaraston todentamisen yhteydessä. Tallennuspaikkana on mahdollista käyttää esimerkiksi tietokantaa tai tekstipohjaista tiedostoa. Toteuttamalla Tomcatin *Realm*-rajapinta *(org.apache.catalina.Realm)*, on mahdollista kirjoittaa oma Realm-luokka. Itse ohjelmoitu Realm-luokka käännetään ja sijoitetaan Tomcatin kansioon ***/lib***. (Apache Tomcat n.d. a)

Realmin konfigurointi Tomcat-sovelluspalvelimella

Käytettävä Realm määritellään palvelimella. Tomcat -sovelluspalvelimella määritys tapahtuu *server.xml*-tiedostossa elementin **<Realm/>** attribuuteilla *(Kuva 8)*. Sama määritys voidaan tehdä yksittäisen sovelluksen *context.xml*-tiedostossa, jolloin asetus vaikuttaa vain kyseiseen sovellukseen. (Apache Tomcat n.d. c)

```
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
       resourceName="UserDatabase"/>
```

Kuva 8 *Realmin konfigurointi Tomcatin server.xml-tiedostossa elementillä <Realm/>.*

Määritys voi sisältää myös muita attribuutteja, kuten esimerkiksi käyttäjätietojen tallentamiseen käytettävän tietokannan sijainnin, nimen ja taulujen nimet riippuen siitä, mitä Realmia sovelluksen halutaan käyttävän. (Apache Tomcat n.d. a)

MemoryRealm

MemoryRealm-luokka käyttää käyttäjätietojen tallentamiseen Tomcat-sovelluspalvelimen *tomcat-users.xml*-tiedostoa *(Kuva 9)*. Tiedostoon tallennetaan käyttäjätunnus, salasana sekä kyseisen tunnuksen rooli. (Apache Tomcat n.d. a)

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="manager"/>
  <role rolename="admin"/>
  <user username="admin" password="salasana" roles="manager,admin"/>
</tomcat-users>
```

Kuva 9 *Esimerkissä käyttäjätunnuksella admin on kaksi roolia: manager ja admin.*

JDBCRealm

JDBCRealm eroaa MemoryRealm-luokasta lähinnä käyttäjätietojen tallennuspaikan suhteen. Jo nimestä voidaan päätellä, että tiedot tallennetaan relaatiotietokantaan ja tietoihin päästään käsiksi Java DataBase Connectorin avulla. Tämä mahdollistaa tietojen sujuvamman muokkaamisen verrattuna Tomcatin *tomcat-users.xml*-tiedostoon. (Apache Tomcat n.d. a)

DataSourceRealm

DataSourceRealm pääsee käsiksi relaatiotietokantaan JNDI:n, eli Javan nimipalvelun avulla *(Java Naming and Directory Interface)* nimetyn JDBC yhteyden kautta. JNDI on Javan rajapinta, joka tarjoaa metodit hakemistojen hallintaan. Tallennettua tietoa, eli objektia voidaan etsiä kansiorakenteesta esimerkiksi sille asetettujen attribuuttien avulla. (JNDI Overview n.d.; Apache Tomcat n.d. a)

JNDIRealm

JNDIRealm käyttää käyttäjätietojen tallentamiseen LDAP-palvelinta, johon se pääsee niin ikään käsiksi JNDI:n avulla. LDAP *(Lightweight Directory Access Protocol)* on verkkoprotokolla, jonka avulla voidaan tuottaa hakemistopalveluja. Saman tyyppisen palvelun tuottaa useimmille tuttu DNS *(Domain Name Server),* jossa selaimeen kirjoi-

tettu Domain johdattaa käyttäjän tietylle Internet-sivulle tai IP-osoitteeseen. LDAP-palvelimelle voidaan siis tallentaa tietovarasto-objekti *(Data Access Object),* joka sisältää objektille kuuluvia tietoja. Tietoihin päästään käsiksi objektin nimen perusteella. (Ahonen, ym. 2003, 337)

JAASRealm

JAASRealmia käytettäessä käyttäjätietoihin päästään käsiksi JAAS *(Java Authentication and Authorization Service)* sovelluskehyksen kautta. JAAS-sovelluskehyksen käyttöön syvennytään tarkemmin luvussa 7. JAASin suurin etu lienee mahdollisuus eriyttää tietoturva muusta ohjelmakoodista. Lisäksi periaatteessa minkä tahansa Realmin käyttö on mahdollista ja turvamekanismia voidaan muokata halutunlaiseksi Principal-objektien avulla. (Apache Tomcat n.d. a)

Digest

Tomcatin Realmit tukevat salasanojen tiivistämistä eli salasanat voidaan salakirjoittaa käyttäen **java.security.MessageDigest**-luokan tukemia algoritmeja, joita ovat *MD5*, *MD2* ja *SHA*. Digest-toimintoa käyttäessä ohjelmaan kirjautuva henkilö antaa salasanansa selkokielisenä syötteenä, jonka jälkeen ohjelma muuttaa sen algoritmin avulla ja vertaa muunnosta aikaisemmin Realmiin tallennettuun salasanaan. Näin selkokielistä salasanaa ei lähetetä verkon yli palvelimelle, joten riski tunnistustietojen vuotamisesta vääriin käsiin on merkittävästi pienempi. Esimerkiksi sana ”*salasana*” tiivistettynä MD5 -algoritmilla näyttää tältä: *e7e941b1f09f266540c6780db51d5f58*. Digest konfiguroidaan sovelluksen **<Realm>**-elementin sisällä. Attribuutin digest arvoksi annetaan jokin tuetuista algoritmeista. (Ahonen, ym. 2003, 253; Apache Tomcat n.d. a)

4.2.2 Käyttäjän tunnistaminen

Käyttäjän tunnistamiseen on tarjolla monia eri vaihtoehtoja. Www-sovelluksien yhteydessä yleisin käytetty tapaa lienee käyttäjätunnus-salasana -yhdistelmä. Muita mahdollisia tunnistusmekanismeja voisivat olla esimerkiksi sormenjälki tai iiris. HTTP-protokollaan on sisällytetty valmiiksi muutamia tapoja tunnistaa käyttäjä ja lisäksi on mahdollista luoda oma käyttöliittymä käyttäjän tunnistamiseen.

HTTP basic

Käyttäjän pyrkiessä suojatulle sivustolle HTTP basic kysyy tunnusta sekä salasanaa ja lähettää tunnistustiedot palvelimelle. Tämä tapa ei ole kovinkaan turvallinen, koska tiedot välitetään Base64-koodattuna palvelimelle. Tämä koodaus on helposti purettavissa ja siihen löytyy internetistä valmiita työkaluja. Base64:n tarkoituksena on estää merkistömuutoksista johtuva datan korruptoituminen siirrettäessä HTTP-protokollan yli. Salauksen kanssa sillä ei siis ole juurikaan tekemistä. (Ahonen, ym. 2003, 234-235)

HTTP digest

HTTP digest on turvallisempi versio HTTP basic tunnistuksesta. Tätä tapaa käyttäessä salasanasta, tunnuksesta, pyydetystä URL:stä, käytetystä HTTP metodista sekä satunnaisesta merkkijonosta luodaan tiivistelmä käyttäen MD5-algoritmia. Palvelin tekee vastaavan tiivistelmän ja vertaa omaansa selaimen lähettämään. Jos nämä kaksi täsmäävät, tunnistaminen on onnistunut. (Ahonen, ym. 2003, 237)

Form based *(lomakepohjainen)*

Usein sovellukseen kirjautuminen halutaan toteuttaa sovelluksen graafiseen ilmeeseen sopivalla lomakkeella. Tällöin sovelluksen *web.xml*-tiedostossa määritellään kirjautumissivu sekä virhesivu, jonne käyttäjä ohjataan tunnistamisen epäonnistuessa. Lomake lähettää tunnistetiedot POST-kutsuna palvelimelle. Oletusarvoisesti palvelin käsittelee tunnistuksen kuten basic-tunnistuksessa. Ohjelmoijan on kuitenkin mahdollista luoda omat metodinsa käyttäjän tunnistamiseen esimerkiksi JAAS-sovelluskehyksen avulla ja näin parantaa sovelluksen tietoturvaa. (Ahonen, ym. 2003, 238, 240)

Lomaketta tehtäessä on sen kentät nimettävä tietyllä tavalla. Lomakkeen kohteeksi (*action*) merkitään ***j_security_check***. Tunnuskenttä tulee olla nimeltään ***j_ username*** ja salasanakenttä ***j_password***. Virhesivun ulkoasua tai toiminnallisuuksia varten ei ole olemassa mitään määrityksiä. (Ahonen, ym. 2003, 239)

SSO (*Single Sign On*)

Usein hajautetuissa web-sovelluksissa on tarpeen yhdistää monta pientä sovellusta yhdeksi suureksi palveluksi. SSO mahdollistaa sen, että käyttäjää ei tarvitse todentaa erikseen jokaista sovellusta varten. Riittää kun käyttäjä kirjautuu palveluun kerran. (Ahonen, ym. 2003, 244)

Tomcatissä SSO konfiguroidaan *server.xml*-tiedostossa *(Kuva 10)* Kaikkien SSO:n alla toimivien sovellusten tulee käyttää samaa Realmia käyttäjän tunnistamiseen. (Apache Tomcat n.d. d)

```
<Host>
...
<Valve className="org.apache.catalina.authenticator.SingleSignOn" />
...
</Host>
```

Kuva 10 *SSO:n käyttöönotto Tomcatin server.xml-tiedostossa.*

SQL Injections

Ennen kuin käyttäjän syöttämiä parametreja liitetään tietokantakyselyyn on tarpeen tarkistaa syötteiden sisältö mahdollisten tietokantaan kohdistuvien hyökkäysten varalta. Tietyillä syötteillä paha-aikeisen käyttäjän voi olla mahdollista päästä käsiksi sisältöön, jolle tällä ei todellisuudessa olisi oikeutta. Syötteet on mahdollista tarkistaa esimerkiksi säännönmukaisilla lausekkeilla (Regular Expressions). Tällöin voidaan esimerkiksi estää tiettyjen sql-komentojen, kuten DROP tai OR, viemisen tietokantakyselyyn.

Toinen, ehkä tyylikkäämpi vaihtoehto estää tietokantaan kohdistuvat hyökkäykset on toteuttaa tietokantakyselyiden koostaminen Javan **PreparedStatement**-rajapinnan avulla *(Kuva 11),* jolloin sql-komentojen tai merkkien **'** ja **”** vieminen kyselyyn ei vaikuta kantaan haitallisesti. Merkkijonot ja kokonaisluvut voidaan sisällyttää kyselyyn set-metodeilla. (Ahonen, ym. 2003, 201)

```
try {

    connection = DriverManager.getConnection(protocol);
    PreparedStatement statement = null;
    statement = connection.prepareStatement("UPDATE user SET age = ? WHERE name = ?");

    statement.setInt(1, 24);
    statement.setString(2, "Iines");

} catch (Exception e) {
    ...
}
```

Kuva 11 *PreparedStatement -rajapinnan avulla syötteet kirjoitetaan kysymysmerkkien paikoille.*

4.3 Valtuutus (Authorization)

Kun käyttäjä on tunnistettu luotettavasti ja käyttäjän oikeudet sovellukseen tiedetään, voidaan kyseiselle käyttäjälle antaa pääsy sovelluksen resursseihin Realmissa määritellyn roolin mukaisesti. Valtuutusta voidaan tehdä joko määrittelevästi *web.xml* -tiedoston avulla tai ohjelmallisesti sovelluksen ohjelmakoodin sisällä.

Turvallisuusalueet

Realmissa määriteltyjen ryhmien pääsy sovelluksen eri osiin määritellään Java EE -sovelluksen *web.xml*-tiedostossa. Elementin ***<security-constraint>*** *(Kuva 12)* sisällä määritellään sovelluksen osa, mihin pääsyä halutaan rajoittaa sekä roolit, joilla on pääsy määritellyn kansion sisältöön. (Ahonen ym. 2003, 24)

```
<security-constraint>
    <display-name>Author</display-name>
    <web-resource-collection>
        <web-resource-name>Author Area</web-resource-name>
        <description/>
        <url-pattern>/WEB-INF/author/*</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <http-method>HEAD</http-method>
        <http-method>PUT</http-method>
        <http-method>OPTIONS</http-method>
        <http-method>TRACE</http-method>
        <http-method>DELETE</http-method>
    </web-resource-collection>
    <auth-constraint>
        <description/>
        <role-name>author</role-name>
        <role-name>admin</role-name>
    </auth-constraint>
</security-constraint>
```

Kuva 12 *Esimerkkikoodissa tiedoston /WEB-INF/author/ sisältöön on annettu oikeus käyttäjärooleille author ja admin.*

<web-resource-collection> määrittelee suojattavan alueen sijainnin. Sijainnille annetaan nimi, jolla siihen voidaan viitata sekä todellinen URL, mistä suojattava kansio löytyy. Lisäksi elementin **<http-method>** sisällä voidaan määritellä, mitä HTTP-pyyntöjä kyseiseen resurssiin sallitaan. Elementin **<auth-constraint>** sisällä määritellään roolit, joilla on pääsy suojattuun resurssiin. (Ahonen ym. 2003, 24)

7 JAAS

Java-sovelluksen tietoturva on myös mahdollista toteuttaa erilaisten valmiiden sovelluskehysten avulla. Yksi mahdollinen käytettävissä oleva kehys on JAAS. JAAS tulee sanoista Java Authentication and Authorization Service. Nimensä mukaisesti tällä sovelluskehyksellä voidaan todentaa käyttäjä tai palvelu ja valtuuttaa tämä käyttämään sovelluksen osia. Vielä Javan versioon 1.3 asti JAAS oli vaihtoehtoinen paketti, mutta se on sisällytetty osaksi Javaa versiosta 1.4 eteenpäin. (Sun Microsystems n.d.)

JAAS toteuttaa sovelluksen tietoturvan erillisenä muusta sovelluslogiikasta. Tämä mahdollistaa erilaisten tietoturvamekanismien käytön ja vaihtamisen ilman että itse ohjelmakoodiin tai palvelimen konfigurointiin tarvitsee koskea. JAAS mahdollistaa minkä tahansa Realmin käytön käyttäjätietojen tallentamiseksi.

7.1 Keskeiset luokat

Subject

Ennen kun resursseja voidaan jakaa, tulee sovelluksen tunnistaa resursseihin kohdistuvan pyynnön kohde. Termi Subject tarkoittaa tämän pyynnön kohdetta, joka tunnistetaan. *Subject* voi olla mikä tahansa objekti, kuten henkilö tai palvelu ja yksittäisellä Subjectilla voi olla useampi identiteetti. Kun *Subject* on tunnistettu onnistuneesti, sille voidaan liittää Principal- ja Credential-objekteja. (Sun Microsystems n.d.)

Principal

Principal kuvastaa Subjectille annettua identiteettiä. Yksittäinen Subject voi siis saada monta Principalia. Esimerkiksi, Subjectin ollessa henkilö, voidaan sille asettaa Principal, joka kertoo että kyseinen henkilö on Hartikainen. Hartikainen on töissä Autokorjaamo Oy:ssä, joten Subjectille voidaan antaa toinen Principal-objekti, joka kuvastaa yrityksen nimeä. Käytännössä Principal-olio kuvaa Subjektille annettua roolia. Principalina käytettävä luokka toteuttaa rajapinnat **java.security.Principal** sekä **java.io.Serializable**. Toteutettavia metodeita ovat **equals(Object another)**, **getName()**, **hashCode()** sekä **toString()**. (Sun Microsystems n.d.)

Credentials

Credentials ei toteuta mitään tiettyä rajapintaa. Yksittäinen credential voi sisältää minkä tahansa luokan olion. Credentialin tarkoitus on periaatteessa sama kuin Principal-luokan olion. Se kuvaa todennettua Subjektia jollain tavalla, mutta ei kuitenkaan kuvaa roolia. Credential voisi esimerkiksi sisältää todennetun henkilön yhteystiedot. Credential liitetään todentamisen yhteydessä listaan, jolla on joko Public- tai Private-määre. (Sun Microsystems n.d.)

config-tiedosto

Config-tiedostossa (*Kuva 13)* määritellään, mitä LoginModule-luokkia käyttäjän todentamiseen käytetään. Todentamiseen käytettäviä luokkia voi olla useita ja LoginContext yrittää todentaa käyttäjän kaikkia määriteltyjä luokkia käyttämällä. (Sun Microsystems n.d.)

```
JaasTest {
    com.toymelab.initiative.login.JaasLoginModule required debug="true"
    url="jdbc:mysql://localhost:3306/toyme2?user=pilotadmin&password=pilotadmin"
    driver="org.gjt.mm.mysql.Driver";
};
```

Kuva 13 *Käytettävä LoginModule-luokka määritellään Comfig-tiedostossa.*

Sovelluksessa käytettävä *config*-tiedosto pitää määritellä JVM:lle ympäristömuuttujaksi palvelimen käynnistyksen yhteydessä, jotta palvelin tietää mistä etsiä tietoa käytettävästä LoginModule-luokasta:

-Djava.security.auth.login.config==[hakemisto/tiedosto.config]
(Apache Tomcat n.d. d)

LoginContext

LoginContext-luokka todentaa Subjectin. LoginContext-olio saa tiedon käytettävästä todentamismenetelmästä tai LoginModule-luokasta configuraatio-tiedostosta.
(Sun Microsystems n.d.)

LoginModule

LoginModule rajapinnan avulla voidaan sovellukselle luoda halutunlainen metodi käyttäjän todentamiseen. Tämän rajapinnan toteuttavan luokan metodit hoitavat todentamisen käytännössä. LoginContext kutsuu LoginModule-luokan metodeita todentaessaan käyttäjää. (Sun Microsystems n.d.)

Rajapinnan toteuttavassa luokassa ohjelmoidaan seuraavat metodit:

- **initialize()** toimii rajapinnan toteuttavan luokan konstruktorina, eli alustaa tarvittavat muuttujat.
- **login()**-metodi yrittää todentaa käyttäjän. Tämän metodin sisällä voidaan vertailla käyttäjän syötteenä antamia tunnisteita esimerkiksi tietokannasta haettuihin vastaaviin tunnisteisiin. Jos todentaminen onnistuu kutsuu LoginContext metodia commit().
- **Commit()** asettaa Subjectille halutut Principal- ja Credential-luokkien oliot, eli roolit.
- **abort()**-metodia kutsutaan, jos todentaminen ei onnistu.
- **logout()**-metodi poistaa todennetun Subjectin Principalit ja Credentialit.

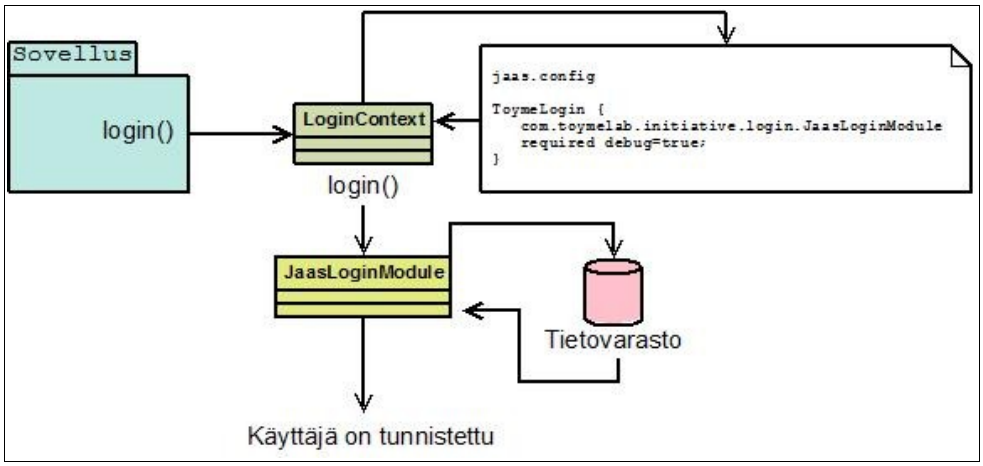
(Sun Microsystems n.d.)

CallbackHandler

Jos LoginModulen on tarpeen kommunikoida käyttäjän kanssa saadakseen tietoonsa todentamiseen liittyviä tietoja, kuten tunnuksen tai salasanan, voidaan se toteuttaa CallbackHandler-rajapinnan avulla. Tämä rajapinta välittää tunnistetiedot käyttäjältä LoginModule-rajapinnan toteuttavalle luokalle. CallbackHandler-rajapinnan avulla tietoja voidaan viedä myös toiseen suuntaan, LoginModulelta käyttäjälle. (Sun Microsystems n.d.)

7.2 JAAS Authentication

Käyttäjän todentaminen JAASin avulla tapahtuu monivaiheisesti: ensiksi ohjelma luo **LoginContext**-luokan olion, joka saa palvelimelle määritellystä JVM:n ympäristömuuttujasta tiedon, missä config-tiedosto sijaitsee. Tämän jälkeen sovellus kutsuu LoginContext-luokan metodia **login()**. Tämä metodi kutsuu kaikkia config-tiedostossa määriteltyjen LoginModule-luokkia, joista jokainen pyrkii todentamaan Subjectin *(Kuva 14)*. Onnistuessaan LoginModule liittää Subjectille Principalit sekä mahdolliset Credentialit ja julistaa Subjectin todennetuksi. Lopuksi LoginContext palauttaa sovellukselle tiedon todennuksen tilasta. Jos todennus onnistuu, LoginContext palauttaa Subjectin sovellukselle. (Sun Microsystems n.d.)



Kuva 14 *LoginContext saa tiedon käytettävästä LoginModulesta jaas.config-tiedostosta.*

7.3 JAAS Authorization

JAAS-sovelluskehystä käytettäessä pääsynvalvonta voidaan hoitaa Policy-tiedostoilla. Jos käytössä on Tomcat sovelluspalvelin, voidaan JAASin yhteydessä käyttää Tomcatin valmista JAASRealmia. Tällöin resurssit voidaan suojata määrittelemällä *web.xml*-tiedostossa turvallisuusalueet. Käytettäessä Policy-tiedostoja pääsynvalvontaan, voidaan Tomcatin kohdalla pääsylistat ohjelmoida Tomcatin *catalina.policy*-tiedostoon. (Sun Microsystems n.d.)

Java Policy

Abstraktin **java.security.Policy**-luokan avulla voidaan laatia sääntöjä, joiden mukaan sovelluksella tai sovellusta käyttävällä Subjectilla on oikeus toimia. Policy-tiedostoon voidaan määritellä myös sääntöjä *(Kuva 15)* joiden mukaan pääsyä voidaan rajoittaa Subjectin roolien perusteella. (Sun Microsystems n.d.)

```
grant   codebase "file:./Business.jar", Principal com.business.package "admin" {
   permission java.io.FilePermission "companysecret.txt", "read";
};
```

Kuva 15 *esimerkissä paketissa Business.jar sijaitsevaan tiedostoon companysecret.txt annetaan pääsy Subjectille, jolle on määritelty rooli admin.*

catalina.policy

tomcat.policy-tiedosto noudattaa Javan Policy-luokan syntaksia, joten pääsylistojen ja sääntöjen ohjelmointi tapahtuu samalla tavalla. (Apache Tomcat n.d. b)

7.4 Vahvuudet & heikkoudet

JAASin tuomat hyödyt liittyvät lähinnä sen muunneltavuuteen. Sovelluksen todentamismenetelmää on helppo vaihtaa korvaamalla sovelluksen *login.conf*-tiedosto ja luomalla uusi *LoginModule*-luokka tai sisällyttämällä sovellukseen useampia *LoginModule*-luokkia. Tällöin sovellus yrittää todentaa käyttäjän jokaisen määritellyn LoginModulen avulla.

Toisin kuin muissa Tomcatin tarjoamissa todentamismenetelmissä, JAASia käytettäessä voidaan samanaikaisesti käyttää useampia erityyppisiä tietovarastoja, jotka voivat sijaita myös eri palvelimella. Näin saavutetaan aidosti joustava todentamismenetelmä, jota on mahdollista muokata tulevaisuudessa rajattomasti.

Toisaalta kaikkia JAAS-sovelluskehyksen ominaisuuksia ei voida hyödyntää www-ympäristössä aivan suoraviivaisesti. Tämä johtuu siitä, että JAAS on alunperin kehitetty työpöytä-sovelluksia varten, eikä siinä ole huomioitu www-ympäristön erityispiirteitä. Suurin menetetty hyöty on Subjectille asetettavien vapaamuotoisten Principal-luokkien katoaminen liikuttaessa sivulta toiselle. Tämäkän voidaan toki korjata todentamalla käyttäjä tämän siirtyessä uudelle sivulle esimerkiksi Servletille asetettavan Filterin avulla. Tämä ratkaisu on kuitenkin hieman jäykkä. Toivottavasti tulevaisuudessa JAASia kehitetään enemmän www-ympäristöön sopivaksi tai sille kehitetään vakavasti otettava vaihtoehto.

8 Toyme2

Toyme2-aloitepalvelun muutokset liittyvät käytettävyyden ja ohjelmakoodin uudelleenkäytettävyyden parantamiseen. Yksi suurimpia asiakkaalle näkyviä uudistuksia on palveluun kirjautuminen. Sovellusta halutaan markkinoida aidosti palveluna, johon asiakas voi kirjautua. Tällä halutaan korostaa palvelun käyttöönoton helpoutta. Toinen merkittävä muutos edelliseen versioon on se, että itse sovellukseen on mahdollista kirjautua mistä tahansa. Aiemmin yrityskohtaiset sovellukset oli suojattu ip-rajauksella, jolloin palveluun pääsi ainoastaan asiakasyrityksen toimitiloista.

8.1 Vaatimusmäärittely

Sovelluksen osan tulee mahdollistaa seuraavat toiminnot:

- *Yksittäinen käyttäjä tulee pystyä todentamaan.*
- *Kirjautuminen on yhteinen kaikille yrityksille. Kaikki yritykset kirjautuvat omaan sovellukseensa samasta url-osoitteesta.*
- *Todentamisen jälkeen käyttäjä voidaan ohjata oman yrityksensä sovellukseen. Tämän jälkeen käyttäjälle jaetaan oman roolin mukaisesti kyseisen yrityksen sovelluksen resursseja.*
- *Lisäksi työssä otetaan huomioon mahdolliset tietoturva-aukot, jotka mahdollistavat esimerkiksi käyttäjän kirjautumistietojen varastamisen ja pyritään mahdollisimman laaja-alaisesti poistamaan tietoturva-aukot.*

Tekniset määritykset

Sovellus ohjelmoidaan Java-kielellä ja julkaistaan Tomcat-sovelluspalvelimella. Käyttöliittymä toteutetaan JSP-sivuilla, joiden staattinen koodi noudattaa validia Strictiä XHTML-kieltä. Tyylit määritellään CSS-tiedostossa. Sovelluksen Java-koodi kommentoidaan JavaDoc-standardin mukaisesti *(Kuva 16)*.

Ohjelmointikieli	*Java (EE 5 / JDK 1.5)*
Käyttöliittymä	*JSP (Validi XHTML Strict, CSS)*
Merkistö	*UTF-8*
Tietovarasto	*MySql*
Sovelluspalvelin	*Apache Tomcat 5.5*
Dokumentaatio	*JavaDoc + UML*

Kuva 16 *Toyme2-sovelluksen tekniset määritykset.*

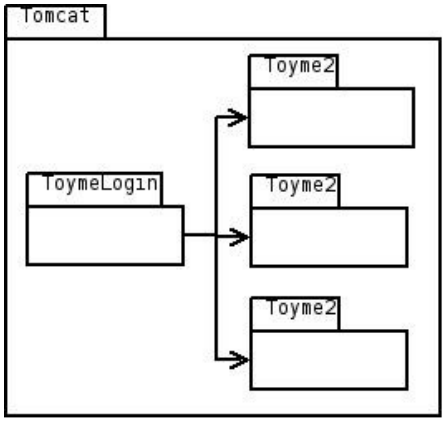
Kehitysympäristönä on NetBeans 6.5 ja paikalliset Apache Tomcat 5.5-sovelluspalvelin sekä MySql 5.0.67 relaatiotietokanta.

8.2 Resurssien suojaaminen Toyme2-sovelluksessa

Koska Toyme2-sovelluksen kehittäminen on vielä kesken, ei tarkkoja roolimäärityksiä voida vielä tehdä. Erilaisia rooleja ja turvallisuusalueita tullaan todennäköisesti lisäämään vielä sovelluksen kehityksen jatkuessa. Sovellusta varten ohjelmoin kuitenkin todentamista varten tarvittavat komponentit. Itse Toyme2-sovellukseen tehtävät turvallisuusalueet käsitellään tässä luvussa periaatteellisella tasolla.

Kirjautuminen

Toyme2-sovellusversion yhteydessä palveluun kirjautuminen halutaan yhtenäistää asiakkaiden kesken. Tämä tarkoittaa käytännössä sitä, että kaikkien asiakasyritysten sovelluksiin kirjaudutaan samasta url-osoitteesta. Jokaisella yritykselle halutaan asentaa kuitenkin oma Toyme2-sovellus, joten ainoastaan kirjautuminen on yhteinen osa palvelua *(Kuva 17)*. Kirjautuminen toteutetaan omana pienenä sovelluksenaan, jonka nimi on **ToymeLogin**.



Kuva 17 *Kirjautuminen Toyme2-sovellukseen toteutetaan erillisenä pienenä sovelluksena, josta käyttäjä ohjataan omaan Toyme2-sovellukseen.*

Toyme2-sovellus halutaan tarjota jokaiselle asiakkaalle erikseen omana sovelluksena, koska tulevaisuudessa on mahdollista, että jollekin yritykselle on tarpeen räätälöidä sovellukseen jokin ominaisuus, mitä muille ei haluta tarjota. Näin sovelluksen räätälöinti on helpompi hallita. Tietokantojen hallinta on myös yksinkertaisempaa, kun ne on eritelty asiakkaiden kesken.

Käytettävä Realm

Toyme2-sovelluksessa tulee olemaan useita eri käyttäjärooleja ja käyttäjien lisääminen myöhemmin tulee olla vaivatonta. Käytettävä tietovarasto tulee siis olla helposti muunneltavissa.

Nämä vaatimukset sulkevat pois sellaiset tavat tallentaa käyttäjätiedot, joissa joudutaan puuttumaan palvelimen konfiguraatiotiedostoihin. Lisäksi tekstipohjainen tallennustapa voisi olla melko epävarma käyttäjämäärän kasvaessa, koska tiedosto voisi helposti korruptoitua useamman käyttäjän pyrkiessä muuttamaan sitä yhtä aikaa. Parhaiten tässä työssä käsitellyistä tekniikoista Toyme2-sovelluksen resurssienhallinnan toteuttamiseen soveltuu JAAS-sovelluskehys yhdistettynä Tomcat-sovelluspalvelimen JAASRealm-luokkaan, jolloin itse tietovarastona käytetään MySql-tietokantaa.

JAASin valintaa puoltaa mahdollisuus eriyttää tietoturva muusta sovelluslogiikasta, mikä mahdollistaa helpon päivitettävyyden ja hallinnan. Lisäksi sovellukseen on mahdollista sisällyttää useampia käyttäjän todentamiseen käytettäviä tietovarastoja. Tämä

on tärkeää, koska käytännössä eri yritysten tietovarastot halutaan pitää erillään toisistaan. Tietokantojen eriyttäminen lisää osaltaan tietoturvan tasoa ja tietokannan taulujen koot eivät kasva liian suuriksi ja hakuajat pysyvät lyhyempinä. Lisäksi joidenkin asiakkaiden kohdalla voi olla tarpeen käyttää ulkopuolisia tietovarastoja, kuten LDAP-palvelinta. Jos osa asiakkaista käyttää eri tietovarastoa kuin toiset, on JAASin käyttäminen käytännössä ainoa järkevä keino toteuttaa erilaisten tietovarastojen yhtäaikainen käyttö.

Käytettäessä JAASin rinnalla Tomcatin JAASRealm-luokkaa, hoitaa Tomcat määrittelevän tietoturvan. Näin erillisten *Policy*-tiedostojen ohjelmointiin ei ole tarvetta, vaan turvallisuusalueet voidaan määritellä yksinkertaisemmin sovelluksen *web.xml*-tiedostossa. Tämä helpottaa palvelinohjelmiston mahdollista päivittämistä, koska palvelimen konfiguraatiotiedostoihin ei tarvitse tehdä muutoksia.

Realmin määrittely

JAASRealm määritellään ToymeLogin-sovelluksen *context.xml*-tiedostossa. Elementin **<Context>** sisällä määritellään käytettävän Realmin-luokan lisäksi luokat, jotka kuvaavat käyttäjää ja sille annettavia rooleja *(Kuva 18)*. Sama asetus tulee tehdä myös jokaisen Toyme2-sovelluksen *context.xml*-tiedostoon, jotta palvelimelle konfiguriutava SSO toimisi.

```
<Realm appName="ToymeLogin"
       className="org.apache.catalina.realm.JAASRealm"
       debug="99"
       roleClassNames="com.toymelab.initiative.login.RolePrincipal"
       userClassNames="com.toymelab.initiative.login.UserPrincipal"/>
```

Kuva 18 *JAASRealmin määrittely sovelluksen context.xml-tiedostossa.*

Tietokanta

Kirjautumista varten luodaan oma tietokanta, johon käyttäjien tiedot tallennetaan. Muut Toyme2-sovellukseen liittyvät tietovarastot ovat yrityskohtaisia ja niihin on pääsy ainoastaan yritykseen kuuluvalla käyttäjällä. ToymeLogin-sovelluksen tietokantaan luodaan kaksi eri taulua *(Kuvat 19 & 20)*. Ensimmäiseen tauluun tallennetaan käyttäjän tiedot, kuten nimi, kirjautumistunnus, salasana ja niin edelleen. Toiseen tauluun tallennetaan kaikki roolit, joita kullekin käyttäjätunnukselle halutaan antaa. Jokaisen

asiakasyrityksen käyttäjien todentamiseen käytetään kyseiselle yritykselle luotua omaa tietokantaa. Taulussa *user* muuttuja *company* kertoo ToymeLogin-sovellukselle minkä yrityksen Toyme2-sovellukseen käyttäjä ohjataan kirjautumisen onnistuessa.

Toinen vaihtoehto olisi ollut eriyttää myös kirjautumiseen käytetyt tietokannat siten, että jokaisella yrityksellä olisi ollut oma kanta käytettävissä. Luotettavaa ja tarpeeksi dynaamista keinoa kannan nimen päättelyyn kirjautumistilanteessa ei kuitenkaan ollut mahdollista toteuttaa. Yksi mahdollisuus olisi ollut lisätä kirjautumislomakkeeseen tekstikenttä, johon käyttäjä olisi voinut kirjoittaa yrityksensä nimen, mutta tämä heikentää käytettävyyttä. Toinen mahdollisuus olisi päätellä yrityksen nimi käyttäjän sähköpostiosoitteesta, mutta tätäkään tekniikkaa ei voi pitää kestävänä: yrityksen sähköpostiosoitteet voivat vaihdella vuosien saatossa jolloin eri työntekijöillä voi olla käytössään erilaiset sähköpostiosoitteen päätteet.

username VARCHAR(100)	password VARCHAR(100)	id INTEGER	company VARCHAR(100)
eemeli@toymelab.fi	O1ZuiwwweuIU98e9w==	100	toymelab
uuno@autokorjaamo.fi	Iuwi e98098wjeEqJWe==	101	autokorjaamo

Kuva 19 *Taulu user*

username VARCHAR(100)	role VARCHAR(100)	id INTEGER
eemeli@toymelab.fi	admin	101
eemeli@toymelab.fi	authenticated	101
eemeli@toymelab.fi	toymelab	101

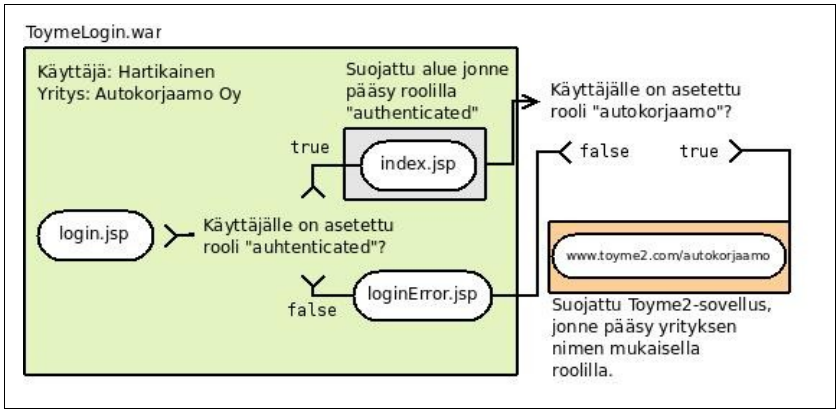
Kuva 20 *Taulu role*

Käytännössä yksi user-taulun tietue vastaa todennetulle Subjectille liitettävän UserPrincipal-luokan olion muuttujia. Vastaavasti role-taulun tietue vastaa RolePrincipal-luokan olion muuttujia.

Roolit ja turvallisuusalueet

Kun käyttäjä tunnistetaan, hänelle asetetaan vähintään kolme eri roolia, joista käytetään selkeyden vuoksi seuraavia nimiä: *authenticated*, *company* ja *role*. *Authenticated* kertoo ToymeLogin-sovellukselle sen, että käyttäjä on ylipäätään todennettu. *Company* kertoo käyttäjän yrityksen ja *role* kertoo käyttäjän oikeudet itse Toyme2-sovelluksen sisällä. Roolit asetetaan käyttäjälle sen jälkeen, kun käyttäjä on tunnistettu onnistuneesti.

Roolin ***authenticated*** tarkoituksena ei sinänsä ole suojata mitään salaista resurssia, vaan sen tarkoitus on vain estää uudelleenohjaaminen, jos käyttäjää ei ole todennettu. Kun ohjelma toteaa, että käyttäjällä on rooli *authenticated*, käyttäjä pääsee jsp-sivulle joka uudelleen ohjaa käyttäjän roolin *company* mukaiseen URL-osoitteeseen *(Kuva 21)*.



Kuva 21 *ToymeLogin-sovellukseen luodaan yksi turvallisuusalue, jolla suojataan sivu index.jsp. Jos käyttäjä on tunnistettu onnistuneesti, eli hänellä on rooli ”authenticated”, on käyttäjällä oikeus siirtyä ToymeLogin-sovelluksen index-sivulle. Tämä sivu ohjaa käyttäjän oman yrityksen sovellukseen.*

Jokainen yksittäinen Toyme2-sovellus on suojattu turvallisuusalueella, jonne on pääsy ainoastaan käyttäjällä, jolle on määritelty yrityksen nimeä vastaava ***company***-rooli. Esimerkiksi yrityksen Autokorjaamo Oy:n sovellus on suojattu roolilla ”*autokorjaamo”,* joten käyttäjältä on löydyttävä tämä rooli päästäkseen sovellukseen. Muussa tapauksessa käyttäjä ohjataan virhesivulle *(Kuva 21)*.

Yrityskohtaisen Toyme2-sovelluksen sisällä yksi tai useampi rooli ***role*** määrittelee pääsyn tiettyihin resursseihin. Näitä resursseja ovat esimerkiksi kansiot tai yksittäiset Servletit ja JSP-sivut. Esimerkiksi admin-tason käyttäjälle kuuluvat resurssit sijoitetaan omaan kansioonsa, johon on pääsy vain tällä roolilla. Ohjelmallisesti Servlettien ja JSP-sivujen sisällä käyttäjän oikeus resurssiin voidaan tarkistaa **isUserInRole(**String role**)**-metodilla.

Tunnistaminen

Tunnistaminen toteutetaan lomakepohjaisena. Käyttäjän tunnistamiseen käytetään uniikkia sähköpostiosoitetta sekä salasanaa. ToymeLogin-sovellus sisältää JSP-sivuilla toteutetun käyttöliittymän kirjautumista varten sekä Tomcatin JAASRealmin rinnalla käytettävän JAAS-sovelluskehyksen vaatimat luokat käyttäjän todentamista varten. Lisäksi suunnitellaan tarvittavan tietokannan rakenne ja luodaan testaamiseen soveltuva tietokanta.

Tomcat-sovelluspalvelimen konfigurointi

Kun kirjautuminen ja käyttäjän todentaminen tapahtuu erillään yksittäisestä sovelluksesta, täytyy sovelluspalvelimella käyttää SSO:ta, jotta kirjautuminen on voimassa siirryttäessä kirjautumissovelluksesta asiakasyrityksen omaan palveluun. ToymeLogin-sovelluksessa käytettävä JAASRealm on konfiguroitava myös yrityskohtaisten sovellusten sisällä, jotta SSO toimisi. SSO konfiguroidaan server.xml-tiedostossa.

jaas.config

Konfiguraatiotiedosto *(Kuva 22)*, josta JAASRealmin luoma LoginContext saa tiedon käytettävästä LoginModule-luokasta, sijoitetaan ToymeLogin-sovelluksen juureen. Sijaintiin viitataan ympäristömuuttujassa, joka määritellään Tomcatin JVM:lle. Tiedostossa määritellään ainoastaan käytettävä LoginModule-luokka.

```
ToymeLogin {
   com.toymelab.initiative.login.JaasLoginModule required debug="true"
};
```

Kuva 22 *jaas.config-tiedoston sisältö.*

8.2.1 Toteutetut JSP-sivut

Resurssien suojaamiseen liittyvät JSP-sivut sisältävät kirjautumiseen tarvittavan lomakkeen sekä sivun joka ohjaa käyttäjän oikeaan osoitteeseen. Lisäksi epäonnistunutta kirjautumista varten tehtiin virhesivu.

login.jsp

Login-sivu sisältää lomakkeen *(Kuva 23)*, jolla kirjautumistiedot saadaan käyttäjältä. Sovellukseen toteutetaan lomakepohjainen kirjautuminen, joten lomakkeen kentät on nimettävä nimillä *j_password* ja *j_username*, jonka lisäksi lomakkeen ”action” on oltava *j_security_check*.

```
<form name="form1" method="post" action="j_security_check">
    <table cellspacing="3">
        <tr>
            <th align="right">Tunnus:</th>
            <td align="left"><input type="text" name="j_username"/></td>
        </tr>
        <tr>
            <th align="right">Salasana:</th>
            <td align="left"><input type="password" name="j_password"/></td>
        </tr>
        <tr>
            <th align="right"> </th>
            <td align="left">
                <input type="submit" value="Log In"/> 
            <input type="reset" value="Reset"/></td>
        </tr>
    </table>
</form>
```

Kuva 23 *Kirjautumislomake.*

ToymeLogin-sovelluksen *web.xml*-tiedostossa määritellään kirjautumismetodiksi lomakepohjainen kirjautuminen *(Kuva 24)*. Lisäksi määritellään virhesivu loginError.jsp jonne käyttäjä ohjataan kirjautumisen epäonnistuttua.

```
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name/>
    <form-login-config>
        <form-login-page>/Login.jsp</form-login-page>
        <form-error-page>/loginError.jsp</form-error-page>
    </form-login-config>
</login-config>

<error-page>
    <error-code>403</error-code>
    <location>/loginError.jsp</location>
</error-page>

<security-role>
    <description/>
    <role-name>authenticated</role-name>
</security-role>
```

Kuva 24 w*eb.xml-tiedostossa määritellään tunnistamista varten login-sivu.*

index.jsp

index.jsp on suojattu *(Kuva 25)* sivu, jonne päästäkseen on käyttäjällä oltava rooli *authenticated*. Tämä sivu ohjaa *(Kuva 25)* käyttäjän oman yrityksensä sovellukseen.

```
<security-constraint>
    <display-name>Company Area</display-name>
    <web-resource-collection>
        <web-resource-name>/*</web-resource-name>
        <description>companyarea</description>
        <url-pattern>/index.jsp</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <http-method>HEAD</http-method>
        <http-method>PUT</http-method>
        <http-method>OPTIONS</http-method>
        <http-method>TRACE</http-method>
        <http-method>DELETE</http-method>
    </web-resource-collection>
    <auth-constraint>
        <description>User authenticated</description>
        <role-name>authenticated</role-name>
    </auth-constraint>
</security-constraint>
```

Kuva 25 *index.jsp suojataan sovelluksen web.xml-tiedostossa.*

```
<%
/*
 * If an instance of UserPrincipal can be found from the request,
 * we try to generate url to the users companys software.
 * If not succeed the url leads the user to the error page.
 */
String urlStart = "http://localhost:8088/";
String url = "";
UserPrincipal user = null;
if (request.getUserPrincipal() != null) {

    // Get the Principal object from the request.
    user = (UserPrincipal) request.getUserPrincipal();
    out.println("Redirecting to " + user.getCompanyOid());
    session.setAttribute("UserPrincipal", user);

    // If user is authenticated, but without the companyOid
    if (!user.getCompanyOid().equalsIgnoreCase("")) {
        url = urlStart + user.getCompanyOid();
    } else {
        url = urlStart + "/loginError.jsp";
    }

} else {
    url = urlStart + "/loginError.jsp";
}
%>
```

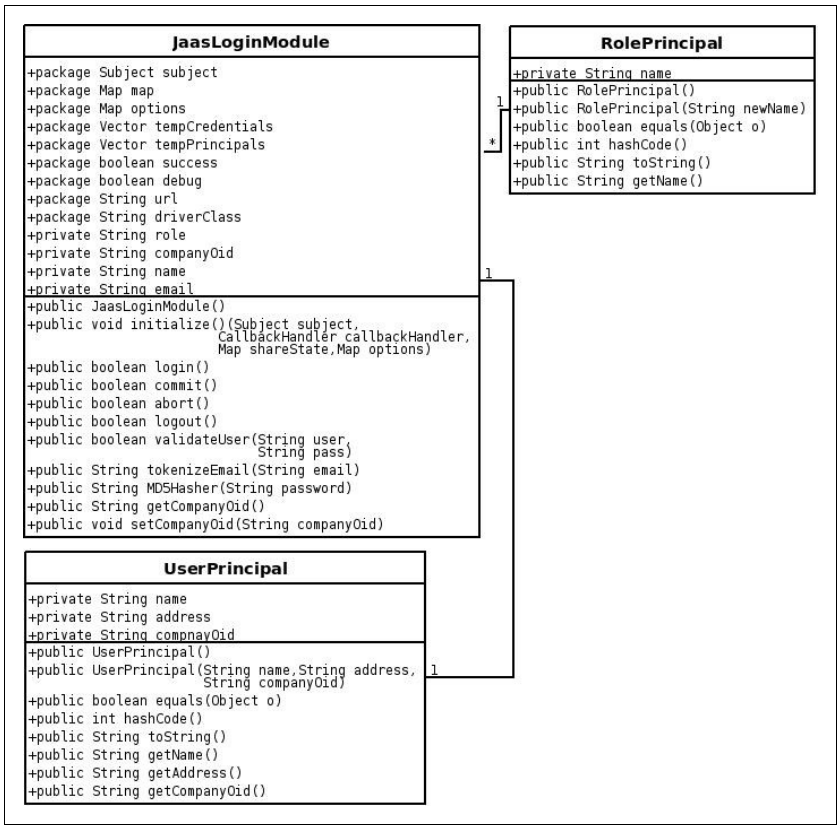
Kuva 26 *ToymeLogin-sovelluksen index.jsp-sivulla haetaan tieto todennetusta käyttäjästä ja luodaan tämän companyOid:n perusteella URL-osoite, johon käyttäjä ohjataan.*

loginError.jsp

Jos kirjautuminen ei jostain syystä onnistu, esimerkiksi käyttäjän syöttäessä virheellisen salasanan, käyttäjä ohjataan *loginError.jsp*-sivulle. Sivu tulostaa kirjautumislomakkeen lisäksi virheilmoituksen, mikä informoi käyttäjää kirjautumisen epäonnistumisesta.

8.2.2 Toteutetut luokat

Tomcatin JAASRealm-luokka vastaa LoginContext-luokan olion luomisesta ja käytöstä, jolloin vain käytettävät Principal-, Credential- ja LoginModule–luokat on ohjelmoitava. ToymeLogin-sovellukseen toteutetaan siis luokat **JaasLoginModule**, **RolePrincipal** sekä **UserPrincipal** *(Kuva 27)*. Luokka JaasLoginModule sisältää käyttäjän tunnistamiseen ja roolien asettamiseen tarvittavat metodit. Lisäksi luokka toteuttaa yhteyden tietokantaan, josta saadaan käyttäjän tunnistamiseen tarvittavat tiedot.



Kuva 27 *ToymeLogin-sovelluksen luokkakaavio*

public class JaasLoginModule

Sovellusta varten ohjelmoitiin LoginModule-rajapinnan toteuttava luokka JaasLoginModule. Rajapinnassa määriteltyjen metodien lisäksi luokka toteuttaa seuraavat metodit:

- private **validateUser(**String user, String pass**)** throws Exception
- public Collection **getRoles(**int userId**)** throws Exception
- public String **MD5Hasher(**String password**)**

validateUser(String user, String pass**)** vertaa käyttäjän syötteenä antamaa salasanaa tietokannasta löytyvää käyttäjän tunnusta vastaavaan salasanaan. Metodi saa salasanan ja tunnuksen parametrina ja palauttaa boolean arvon *true* tai *false* sen mukaan vastaavatko salasanat toisiaan. Jos tämä metodi palauttaa arvon false, toteaa JaasLoginModule, ja sitä kautta myös LoginContext, että todentaminen on epäonnistunut ja LoginContext kutsuu JaasLoginModule-luokan metodia abort().

Metodi **getRoles(**int userId**)** hakee tietokannasta roolit todennetulle Subjectille parametrina saadun käyttäjän id:n perusteella. Metodi palauttaa kokoelman rooleista, josta ne voidaan metodissa commit() liittää Subjectille.

MD5Hasher(String password**)** tiivistää parametrina annetun merkkijonon MD5-algoritmia käyttäen ja palauttaa sen. Salasanan kryptaus olisi mahdollista toteuttaa myös Tomcatin JAASRealm-luokassa kirjautumisen yhteydessä. Tämä olisi kuitenkin voinut tulevaisuudessa aiheuttaa ongelmia, jos käyttöön otetaan ulkopuolinen tietovarasto, kuten esimerkiksi asiakkaan oma LDAP-palvelin. Tällöin salasanat on voitu tiivistää käyttäen jotain muuta algoritmia. Näin ollen salasanan tiivistäminen on järkevämpää toteuttaa erikseen JaasLoginModulessa.

public class RolePrincipal

Luokka RolePrincipal edustaa Subjectille annettavaa yksittäistä roolia. RolePrincipal toteuttaa rajapinnat Principal ja java.io.Serializable. RolePrincipal-luokan oliolle voidaan tallentaa käyttäjän tietoja, kuten nimi, osoite ja puhelinnumero.

public class UserPrincipal

UserPrincipal edustaa käyttäjälle annettavaa identiteettiä, kuten nimeä. Luokka niin ikään toteuttaa rajapinnat Principal ja java.io.Serializable.

8.3 Testaus

Tämän työn puitteissa testaaminen rajoittuu lähinnä kirjautumisen toimivuuden testaamiseen. Koska itse Toyme2-sovellus on vielä kehitysvaiheessa eikä vielä tiedetä tarkkaan sen rakennetta, on tietoturvaa hyödytöntä testata laajemmin. Testaaminen tullaan kuitenkin toteuttamaan muun sovelluksen testaamisen yhteydessä.

Sovelluksen testaaminen keskittyy pääasiassa eri roolien oikeuksien testaamiseen. On tärkeää varmistaa, että sovellukseen kirjautuneella käyttäjällä ei ole mahdollisuutta päästä käyttämään resursseja, joihin hänelle ei ole annettu oikeuksia. Tietoturvan pitävyyden lisäksi on myös hyödyllistä testata sovelluksen tietokantahakujen raskautta. Sovellus testataan myös kuormitustestillä, jotta voidaan varmistua sen toimivuudesta raskaassa käytössä.

JUnit & JMeter

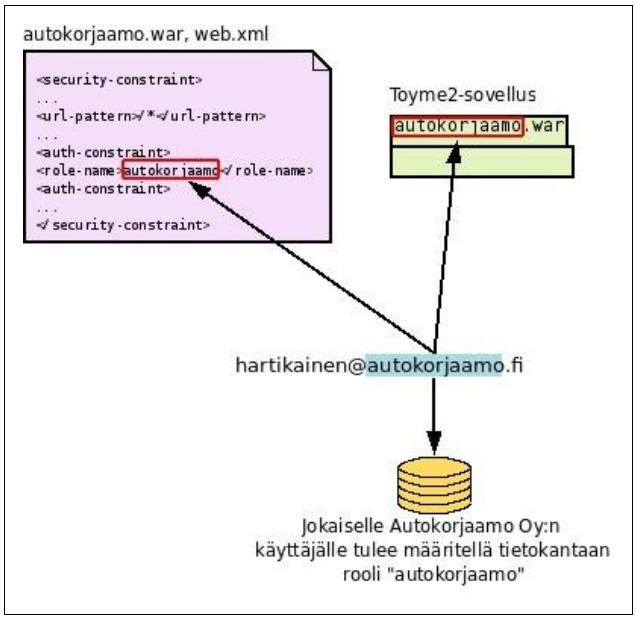
Sovelluksen testaus halutaan automatisoida. Java sovellusta ja Tomcat sovelluspalvelimella toimivaa sovellusta voidaan testata JUnit-testiluokkien avulla, joiden rinnalla hyödynnetään Jmeter-sovellusta. JMeter on Apachen kehittämä sovellus niin staattisten kuin dynaamistenkin Java-sovellusten testaamiseen. Sovelluksella voidaan tehdä kuormitustestejä palvelinsovellukselle ja ajaa laajoja testisuunnitelmia automaattisesti käyttäen JUnit-testiluokkia.

Testitapausten automatisoinnista on paljon hyötyä tulevaisuudessa. Aina, kun sovellusta päivitetään, on tarpeen ajaa tietyt perustestit, jotta voidaan varmistua siitä, ettei päivitysten yhteydessä olla aiheutettu ongelmia aikaisempaan ohjelmakoodiin. Näiden testien ajaminen käsin vie paljon aikaa, joten niiden automatisointi vapauttaa ohjelmoijan muihin töihin.

8.4 Sovelluksen konfigurointi uudelle asiakkaalle

Kun Toyme2-sovellus otetaan käyttöön uutta asiakasta varten, täytyy siihen tehdä muutamia määrityksiä *(Kuva 28)*. Sovelluksen *web.xml*-tiedostoon täytyy määritellä sovelluksen nimi, joka vastaa käyttäjille asetettavaa *company*-roolia. Lisäksi koko Toyme2-sovellus suojataan sovelluksen nimeä vastaavalla roolilla.

Tallennettaessa tietokantaan käyttäjiä on jokaiselle käyttäjätunnukselle liitettävä niin ikään tämä rooli, jotta käyttäjällä olisi pääsy yrityksensä sovellukseen. Myös itse Toyme2-sovellus tulee sijoittaa palvelimella saman nimiseen pakettiin.



Kuva 28 *Toyme2-sovelluksen osoite ja sinne pääsyyn vaadittava rooli määräytyy asiakasyrityksen nimen mukaan.*

9 Tavoitteiden saavuttaminen ja jatkotoimet

Tavoitteena oli tutkia, minkälaisia eri mahdollisuuksia on toteuttaa resurssien suojaaminen web-sovelluksessa, joka on toteutettu Java-kielellä. Ongelma rajattiin sovelluksiin, jotka julkaistaan Apache Tomcat -sovelluspalvelimella, koska toimeksiantajan sovellus Toyme2 julkaistaan kyseisellä palvelimella.

Suurin ja aikaa vievin työ projektin aikana oli etsiä asiaan liittyvää tietoa ja analysoida sitä siten, että oli mahdollista löytää juuri tähän työhön sopivat materiaalit. Työn aikana opin paljon uutta liittyen Javan tietoturvaan sekä kuinka eri käyttäjätasojen välinen sisällönhallinta voidaan toteuttaa. Itse ohjelmointityö vei suhteessa huomattavasti vähemmän aikaa, kuin tiedon etsintä ja analysointi.

Työlle asetetut tavoitteet saavutettiin ja toimeksiantajan toivoma toiminnallisuus saatiin toteutettua. Työn valmistuminen edesauttaa koko sovelluskokonaisuuden valmistumista aikataulussa, jolloin uutta sovellusversiota päästään tarjoamaan uusille asiakkaille mahdollisesti jo kesän 2009 kuluessa.

Jatkossa on mahdollista suunnitella valmiiksi eri tyyppisiä LoginModule-luokkia, joilla voidaan kerätä kirjautumistietoja erilaisista tietolähteistä. Tällä hetkellä sovelluksessa on toteutettuna luokka, joka hakee tiedot SQL-tietokannasta. Valmiit luokat tietojen hakemiseen esimerkiksi LDAP-palvelimen kautta asiakkaan omasta tietovarastosta nopeuttaisivat sovelluksen käyttöönottoa tulevaisuudessa, asiakkaan tällaista toiminnallisuutta toivoessa. Lisäksi sovellusta myytäessä tämä mahdollisuus voisi toimia myyntivalttina.

Työ oli kaiken kaikkiaan erittäin mielenkiintoinen. Sovelluksen tietoturvaan ei voi koskaan kiinnittää liikaa huomiota ja vaikka tietoturvan maailma on laaja tunnen onnistuneeni opinnäytetyössäni hyvin. Halua myös kiittää Toyme Lab Oy:tä kannustuksesta ja hyvästä aiheesta.

Lähdeluettelo

Ahonen, Tero; Hämeen-Anttila, Tapio; Åstrand, Kim. 2003.
Java Servlets. Jyväskylä: Docendo Finland Oy.

Apache Tomcat n.d. a
[online][viitattu 3.3.2009]
http://tomcat.apache.org/tomcat-6.0-doc/realm-howto.html

Apache Tomcat n.d. b
[online][viitattu 3.4.2009]
http://tomcat.apache.org/tomcat-6.0-doc/security-manager-howto.html

Apache Tomcat n.d. c
[online][viitattu 15.3.2009]
http://tomcat.apache.org/tomcat-5.5-doc/realm-howto.html#What%20is%20a%20Realm?

Apache Tomcat n.d. d

[online][viitattu 14.4.2009]
http://tomcat.apache.org/tomcat-6.0-doc/config/valve.html

Interface HttpServletRequest
[online][viitattu 27.4.2009]
http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/http/HttpServletRequest.html

J2EE Containers n.d..
[online][viitattu 9.3.2009]
http://java.sun.com/j2ee/1.3/docs/tutorial/doc/Overview4.html

JNDI Overview n.d.
[online][viitattu10.3.2009]
http://java.sun.com/products/jndi/overview.html

Kuha, Janne 2004.
Tehokas Java WW -sovellustuotanto.
Jyväskylä, WSOYpro/Docento-tuotteet

Niskanen, Pekka; Kontio, Mikko; Vierimaa, Kimmo. 2000.
Inside Enterprise Java. Helsinki: Oy Edita Ab.

Sun Microsystems n.d.
[online][viitattu 30.1.2009]
http://java.sun.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html

Westerholm, Mika; Kyppö, Jorma. 2001.
Java-ohjelmointi Pro Training. Pieksämäki: Talentum Media Oy.

Wutka, Mark. 2001. Inside Java J2EE. Nelimarkka, Mari (Suom.). 2003.
Edita Publishing Oy.