TAMPEREEN
AMMATTIKORKEAKOULU

LIFERAY PORTAL-OHJELMISTON VERSIO-PÄIVITYS

Opinnäytetyö

Tero Mielikäinen

Opinnäytetyö
Huhtikuu 2018
Tieto- ja Viestintätekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja Viestintätekniikan koulutus
Ohjelmisto- ja Tietoliikennetekniikka

TERO MIELIKÄINEN:
Liferay Portal ohjelmiston versiopäivitys

Opinnäytetyö 59 sivua, joista liitteitä 0 sivua
Huhtikuu 2018

Opinnäytetyön tarkoitus on Oscar Softwaren cERP toiminnanohjausjärjestelmän hyödyntävän Liferay Portal Community Editionin versiopäivitys. Päivitysprosessi ei ole tarkoitus olla minkäänlainen malli päivittämiseen vaan yksi mahdollisuus toteutukseen käyttäen yrityksen valmiita koodeja. Päivitys on toteutettu hyödyntäen Liferayn omaa dokumentaatiota sekä yrityksen työntekijöiden tietotaitoa vanhan järjestelmän osalta.

Työn teoria osuudessa esitellään käytetty teknologia Liferay yleisesti, sekä työssä hyödynnetyt työkalut. Teoria osuuden jälkeen esitellään päivitysprosessi sekä siihen liittyvät asiat.

Työn lopputuloksena syntyi päivitetty Liferay MVC portlet sekä Liferay 7 portaalin toimiva Tomcat palvelin. Lopuksi on saatu paljon hyödyllistä tietoa muiden portletien päivitysmahdollisuuksista.

Asiasanat: Portlet, OSGi, Java, JAR, WAR, Teema, Hook, Tomcat, JSP, Liferay, Liferay Portal, Spring, Eclipse

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
ICT Engineering
Software Engineering & Telecommunications and Networks

TERO MIELIKÄINEN:
Liferay Portal version upgrade

Bachelor's thesis 59 pages, appendices 0 pages
April 2018

The purpose of this thesis is version upgrade of Liferay Portal Community Edition, used by OscarSoftware cERP enterprise resource planning system. Upgrade process is not meant to be any tutorial, but one possibility how it's done using company's existing codes. Upgrade is done with utilizing Liferays own documentation and coworkers knowledge of the current system.

In the theory part of this thesis there is introduction on fundamentals of Liferay and shortly introduce used tools for this project.

Result of this thesis is a working Liferay MVC portlet and working Liferay 7 Portal Tomcat server. As an end result, there is a lot of information about possibilities of upgrading rest of the portlets used in this project.

Key words: Portlet, OSGi, Java, JAR, WAR, Theme, Hook, Tomcat, JSP, Liferay, Liferay Portal, Spring, Eclipse

SISÄLLYS

ERITYISSANASTO

ERP	Toiminnanohjausjärjestelmä (Enterprise Resource Planning).
Front-end	Käyttöliittymäkoodit.
Back-end	Palvelinpuolen koodit.
JSP	JavaServer Pages, voidaan luoda dynaamisia web sivuja.
Portlet	Käyttöliittymään lisättävä ohjelmistokomponentti.
IDE	Kehitysympäristö. Integrated Development Environment
API	Ohjelmointirajapinta, mahdollistetaan ohjelmien väliset pyynnöt ja tiedonsiirto
Intranet	Tietyn ryhmän käyttöön eristetty lähiverkko. Yleisesti organisaation sisäisessä käytössä viestintään sekä tietojenkäsittelyyn.
Extranet	Yrityksen halutuille sidosryhmille tarkoitettu suljettu verkkopalvelu.
Gartner	ICT-alan johtava tutkimus- ja konsultointiyritys
JVM	Java Virtual Machine
JRE	Java Runtime Environment, sisältää kaiken mitä tarvitaan Java ohjelmien ajoon. JVM toteutuksen ja Javan luokkakirjaston.
Framework	Ohjelmistokehys, tarkoittaa ohjelmistotuotetta, joka muodostaa rungon sen päälle rakennettavalle ohjelmalle.
Sovelluspalvelin	Ohjelmistokehys, joka tarjoaa palvelut Web sovellusten luontiin ja niiden ajamisen palvelimella
Servlet container	Javan Servlet API:n toteutus, tarjoaa Web palvelimen, jonka toiminnallisuutta voidaan laajentaa omilla komponenteilla.
Web sovellus	Sovellus jota ajetaan asiakaspäässä selaimella
OSGi	Open Service Gateway initiative, Java ohjelmistokehys modulaaristen sovellusten ja kirjastojen luomiseen.
OSGi paketti	OSGi bundle, OSGi mallin mukainen modulaarinen ohjelmistokomponentti
MVC	Model View Controller- malli, jossa toteutus jaetaan kolmeen toisiinsa yhteydessä oleviin osiin, Malli, näkymä ja ohjain.

WAR	Web Application Archive, Java EE tyylinen web sovellusartefakti
JAR	Java Archive, OSGi pakettiartefakti
WAB	Web Application Bundle, artefakti, joka luodaan, kun WAR artefaktit muutetaan WAB muotoon, siihen tarkoitetun muuntajan avulla. Muunnoksen jälkeen ne ovat yhteensopivia OSGi mallin kanssa.
BND	Työkalu, jota käytetään OSGi pakettien metatietojen luomiseen.
URL	Verkko-osoite, osoittaa resurssin sijainnin tietokoneverkossa.
DTD	Document Type Definition, dokumentin tyyppimääritelmä, määrittelee rakenteen, lailliset elementit ja XML attribuutit.
REST	HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen.
Catalina	Tomcat-palvelimen servlet-säiliö.
Bootstrap	Avoimen lähdekoodin ilmainen front-end kirjasto sivustojen ja web-sovelluksien suunnitteluun.

1 JOHDANTO

Opinnäytetyössä käsitellään Oscar Softwaren cERP:n hyödyntämän Liferay Portal 6.2 Community Editionin päivittäminen uuteen 7.0 versioon. Liferay Portal on avoimen lähdekoodiin perustuva, lähinnä organisaatioille tarkoitettu portaaliratkaisu.

Uudessa versiossa on useita ominaisuuksia, jotka hyödyttäisivät yrityksen nykyistä järjestelmää. Modulaarisuutensa ansiosta projekti voidaan pilkkoa järkevämmin osiin, sekä moduuleja voidaan käyttää myös toisissa projekteissa. Uudella versiolla on luonnollisesti uusien riippuvuuksien, kuten uuden Java-version, ansiosta myös parempi tietoturva. Vanhalla versiolla ei ole ollut mahdollista hyödyntää täysin REST endpointeja, uudessa versiossa on niihinkin uusia mahdollisuuksia ja parannuksia. Uudella portaaliversiolla on parempia suorituskykyä parantavia ominaisuuksia ja säätömahdollisuuksia

Työ keskittyy vahvasti Liferayn uusimpaan versioon, lähinnä siitä syystä, että se on ajankohtaisempi ja sisältää uudempaa dokumentaatiota. Koska tarkoituksena on päivittää uuteen versioon, on myös sen takia relevantimpaa keskittyä työhön sen kannalta, käyttäen toki referenssejä vanhempiin versioihin.

Liferay Portalin back-end kielenä käytetään Javaa. Käyttöliittymästä tulee valmiina oleva pohja, jonka päälle voidaan alkaa rakentamaan omanlaista sivustoa. Tätä voidaan muokata omilla teema-liitännäisillä ja asettelumallien avulla. Toinen mahdollisuus käyttöliittymän muokkaamiseen on Liferayn Hook-liitännäiset. Niillä voidaan ylikirjoittaa olemassa olevien moduulien logiikkaa. Syvemmin Liferay Portalin toimintaan ja moduuleihin perehdytään työn toisessa kappaleessa. (Liferay Fundamentals, 2017)

Opinnäytetyön toisessa kappaleessa esitellään käytetty teknologia ja siihen liittyvät yleiset asiat. Kolmannessa kappaleessa tutustutaan työn aikana käytettyihin työkaluihin. Työssä käytettyjen Maven arkkityypeillä luotujen liitännäisten sekä palvelimen kansiorakenteita analysoidaan kappaleessa neljä. Kappaleessa viisi keskitytään itse päivitykseen, tavoitteista aina prosessin kautta lopputulokseen. Viimeisessä kappaleessa analysoidaan työn onnistumista, sekä pohditaan jatkokehitystarpeita.

2 KÄYTETYN TEKNOLOGIAN ESITTELY

2.1 Liferay Portal yleisesti

Liferay Portal on ilmainen ja avoimeen lähdekoodiin perustuva yritysportaaliohjelmisto. Se julkaistiin ensimmäisen kerran vuonna 2000. Ohjelmiston pääkäyttökohteena on tarjota yrityksille intranet ja extranet-toteutuksia.

Ohjelmisto on Java-pohjainen, joten se toimii kaikilla alustoilla, jotka kykenevät pyörittämään JRE:tä eli Java Runtime Environmentia ja sovelluspalvelinta. Liferay on saatavilla paketoituna servlet containerin, eli Tomcatin tai sovelluspalvelimen WildFly, aiemmin JBoss, kanssa. (Liferay Developer Documentation, 2017, Fundamentals)

Liferay on saanut useita tunnustuksia toteutuksestaan, merkityksellisempänä varmasti valinta suurimmaksi avoimen lähdekoodin portaaliohjelmistoksi InfoWorldin toimesta vuonna 2007. (Infoworld, 2007) Gartnerin vuonna 2014 julkaistussa raportissa ”Magic Quadrant for Horizontal Portals” Liferay todettiin suoriutuneen paremmin kuin SAP:n, Oraclen ja Microsoft Sharepointin vastaavat tuotteet, nousten johtavaksi avoimenlähdekoodin tuotteeksi omassa sarjassaan. Liferay on ollut sen jälkeen vuosittain Gartnerin Leader eli johtaja luokassa. (Gartner & Liferay, 2014-2018)

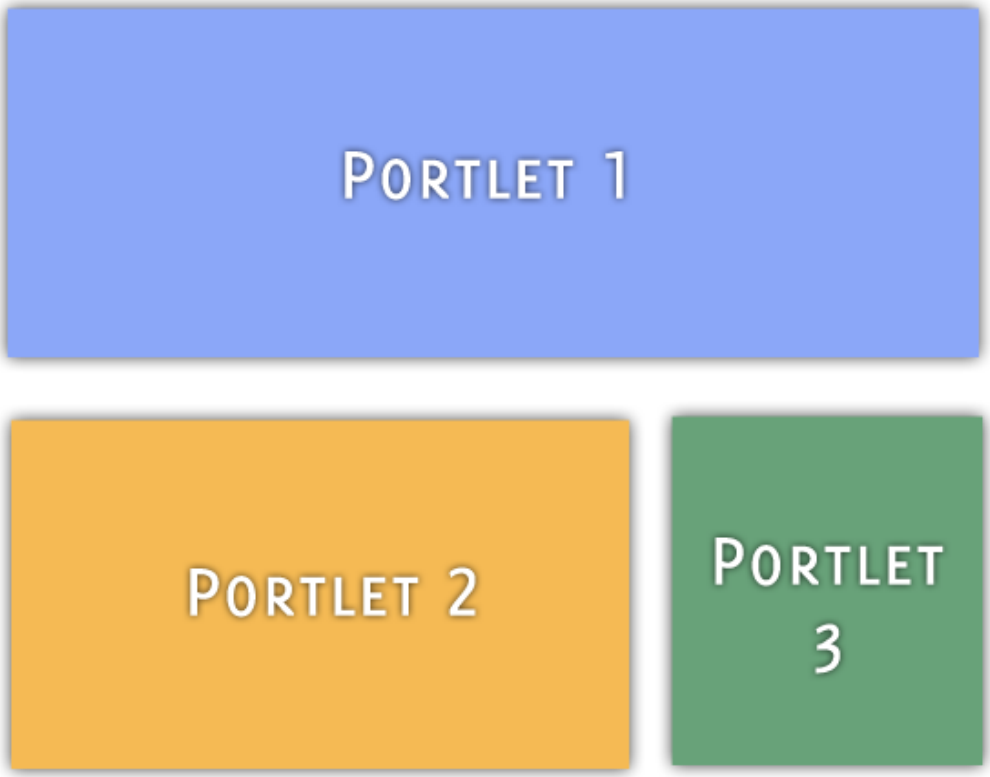
Liferay Portalista on kaksi eri tuotetta; ilmainen Community Edition, johon kuuluu viimeisimmät toiminnallisuudet sekä yhteisön tuki ja Enterprice Edition, myöhemmin Liferay DXP, joka on kaupallinen toteutus portaalista. Kaupalliseen versioon kuuluu lisäpalveluita kuten päivitykset ja täysi tekninen tuki. (Liferay, 2017)

2.2 Portlet liitännäiset

Portletit ovat Liferay Portalin sivustoille asetettavia web-sovelluksia. Kuten monet muutkin web-sovellukset, portletit käsittelevät erilaisia kutsuja sekä luovat vastauksia kutsuihin. Erona tavanomaisiin web-toteutuksiin on niiden toiminta tietyssä osassa sovellusta, tarkoittaen, että portletin omat komponentit pitävät huolen sen omasta sisällöstä. Liferay

Portalin tapauksessa sivusto koostuu monista paloista, jotka toimivat erikseen omina osioinaan. Portleteja voidaan ajaa vain portaalipalvelimella, täten voidaan hyödyntää palvelimen olemassa olevia toimintoja, kuten käyttäjän- ja sivuston hallintaa.

Portleteja voidaan asettaa sivustolle järjestelmänvalvojan tai käyttäjän itsensä toimesta, niiden ominaisuuksiin voidaan määritellä, kenellä on oikeus niitä nähdä, lisätä tai poistaa sivustolta. Sivustolle voidaan asettaa portletit usealla tavalla, esimerkiksi kuvan (kuva 1) mukaisesti. (Liferay Developer Documentation, 2017, Portlets)



Kuva 1 Portletien asettelu (Liferay Developer Documentation, 2017, Portlets)

Portletit perustuvat Javan Portlet Specification standardeihin, joista kirjoitushetkellä uusin on huhtikuussa 2017 julkaistu JSR-362. (Liferay Fundamentals, Open source and based on standards, 2017)
Liferay Portalissa on käytettävissä erilaisilla teknologioilla toteutettuja portleteja. Seuraavissa alakappaleissa keskitytään työn kannalta merkittävimpiin toteutusmahdollisuuksiin, Liferay MVC portletiin ja Spring MVC portletiin. (Liferay Portlets, 2017)

2.2.1 Liferay MVC Portlet liitännäinen

Liferayn oma implementaatio MVC mallin portletista, joka on yksinkertainen laajennus Javan GenericPortlet luokkaan. Liferayn omissa dokumentaatioissa portlet-toteutusta kuvataan kevyemmäksi verraten muihin Java MVC mallin toteutuksiin. Oman toteutuksen tarkoituksena on myös virtaviivaistaa monia portletien ominaisuuksia.

Ohjain eli controller voidaan jakaa useisiin MVC mallin komentoluokkiin, joista jokainen hallitsee ohjaimen koodeja tiettyjen portletin vaiheen mukaan. Render, action ja resurssien palveluvaiheet.

Jokainen portaalissa valmiina oleva Liferayn oma portlet käyttää kyseistä luokkaa, joten niiden toteutuksista saa viitteitä omien portletien kehitykseen ja suunnitteluun. (Liferay Portlets, Liferay MVC Portlet, 2017)

2.2.2 Spring MVC Portlet liitännäinen

Liferay Portalin ollessa avoin ekosysteemi, se mahdollistaa myös muiden ulkopuolisten työkalujen käytön, kuten Spring Framework. Spring Framework on Java-alusta, joka tarjoaa kattavan infrastruktuurin Java-ohjelmien kehitykselle. Spring mahdollistaa ohjelmien rakentamisen tavallisista Java objekteista (engl. plain old Java objects, POJO), sekä asettaa niihin yrityspalveluja. Tämä kyky koskee Java SE, Standard Edition ohjelmointimallia ja täysin tai osittain Java EE, eli Enterprise Editionia.

Seuraavaksi lista esimerkkejä, kuinka Spring-alustaa voidaan käyttää kehityksessä hyödyksi.

- Java-metodi voi suorittaa tietokantatransaktion ilman, että sen täytyy käsitellä transaktiorajapintoja (engl. transaction APIs).
- Lokaali Java metodi voidaan luoda etäaliohjelmaksi (engl. remote procedure), ilman että tarvitsee käsitellä etärajapintoja (engl. remote APIs)
- Lokaali Java metodi voidaan luoda käsittelyoperaatioksi (engl. management operation), ilman JMX, eli Java käsittelylisäosa (engl. Java management extension) rajapintoja.

- Lokaali Java metodi voidaan luoda viestinkäsittelijäksi (engl. message handler) ilman JMS, eli Java viestipalvelu (engl. Java Message Service) rajapintoja. (Spring Framework Introduction, 2014)

Koska Spring perustuu Java Enterprise Editioniin, täytyy se paketoida legacy tyypin WAR moduuliksi. OSGi mallin mukaisesti moduulit pakataan OSGi pakettiartefakti eli JAR moduuleiksi, tästä syystä ne täytyy sijoittaa palvelimelle joko WAR muodossa, tai muuttaa WAB, eli web application bundle muotoon. Tämä tapahtuu käyttämällä WAB Generaattoria, joka käsittelee moduulit, muuttaen ne OSGi malliin sopiviksi moduuleiksi.

Työn aikana Spring Framework aiheutti valtavan määrän päänvaivaa, johtuen esimerkiksi sen OSGi-mallin yhteensopimattomuudesta. WAB malliin muuttaminen on mahdollista, mutta yrityksen kehittäjien mielestä tämä on turha väliaskel. Todettiin, että Spring MVC portletit jätetään toistaiseksi päivittämättä, kunnes ne joko tehdään uudelleen käyttäen jotain toista teknologiaa, tai keksitään, miten portletit saadaan ajettua toimivana WAR muodossa palvelimelle.

2.3 Teema liitännäinen ja asettelumallit

Liferay Portal sisältää valmiin perusteeman, joka monessa käyttötarkoituksessa riittää varsin hyvin. Monet yritykset haluavat kuitenkin muista erottuvan, henkilökohtaisen teeman sivustoilleen.

Käyttötuntumaa voidaan muokata ulkonäöllisesti pääasiassa kahdella eri tavalla, teema liitännäisen tai layout templaten, eli asettelumallin avulla. Kaikki Liferay liitännäiset ovat niin sanottuja hot deployable liitännäisiä, joka tarkoittaa sitä, että niitä voidaan asentaa palvelimelle sen ollessa käynnissä. Asettelupohjilla voidaan suoraan vaikuttaa miten portletit näkyvät sivulla. Pohjalla luodaan CSS-pohjaisia säiliöitä, joissa portletit ”elävät” sivuston rakenteessa. Liferaylla on sisäänrakennettuna muutamia asettelupohjia, mutta on mahdollista luoda omiin tarpeisiin henkilökohtaisia pohjia. (Liferay Themes and Layout Templates, 2017)

2.4 Hook liitännäinen

Koska Liferay Portalissa on sisäänrakennettuna niin paljon valmiita ominaisuuksia, pienemmillä tarpeilla on mahdollista tulla yksinään niillä toimeen. Kuitenkin suuremilla yrityksillä on tarvetta edelleen muokata kokemusta omanlaisekseen, Liferay Portalin ominaisuuksia voidaan ylikirjoittaa käyttäen hook liitännäisiä.

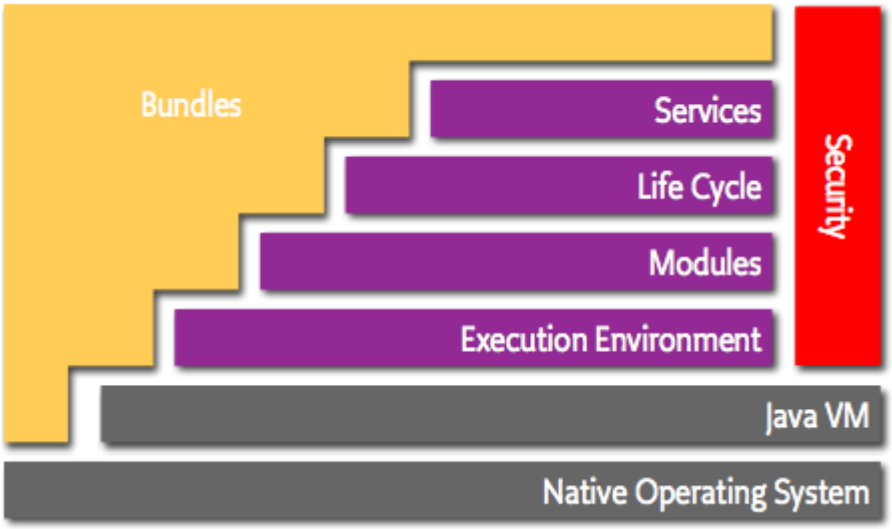
Hook liitännäisillä on mahdollista ylikirjoittaa dynaamisesti tiettyjä portaalin ominaisuuksia, jotka määrittelevät tapahtumatoimintoja (engl. event actions), mallikuuntelijoita (engl. model listeners), validaattoreita, generaattoreita ja sisällönpuhdistajia (engl. content sanitizers). Näiden ominaisuuksien ylikirjoittamisella voidaan muun muassa luoda omanlainen kirjautumiskäytäntö. Oscar Softwaren cERP:ssä on esimerkiksi luotu hook, joka ohjaa käyttäjän organisaation mukaiselle sivulle kirjautumisen jälkeen. Sekä synkronoi käyttäjän Outlook-tapahtumat tietokantaan. (Liferay Overriding Portal Properties Using a Hook, 2017)

2.5 OSGi teknologia

OSGi teknologia on kokoelma spesifikaatioita, jotka määrittelevät Javan dynaamisia komponentteja. Nämä spesifikaatiot mahdollistavat ohjelmistokehitysmallin, jossa sovellukset ovat muodostettu monista erilaisista, uudelleenkäytettävistä komponenteista. OSGi spesifikaatiot mahdollistavat komponenttien toteutuksen piilottamisen toisilta komponenteilta ja kommunikaation palveluiden avulla, jotka ovat nimenomaan eri komponenttien välillä jaettuja objekteja.

OSGi:n tarkoitus on vähentää huomattavasti monimutkaisuutta kehitysprosessin lähes joka näkökulmasta. Koodia on helpompi kirjoittaa ja testata, komponenttien uudelleenkäytettävyys on parempi, kääntämisprosessit ovat yksinkertaisempia, sovellusten käyttöönotto on hallittavampaa, virheet havaitaan aiemmin ja ajoympäristö antaa paljon tietoa siitä mitä ajetaan.
OSGi mallin olennainen käsite on modulaarisuus, joka tarkoittaa käytännössä, että OSGi paketti on tavallinen JAR tiedosto. Kuitenkin erona Javan JAR malliin kaikki sen sisältö ei ole näkyvissä muille, vaan OSGi paketti pyrkii piilottamaan kaiken sen sisällön, ellei

niitä erikseen viedä toiseen pakettiin. OSGi malli koostuu kerroksista, jotka on esitelty alla olevassa kuvassa (kuva 2).



KUVA 2 OSGi kerrokset (OSGi Alliance, Architecture, 2018)

Seuraavassa listassa avataan hieman kerroksien termejä.

- Bundles – Kehittäjien luomia OSGi komponentteja
- Services – Palvelukerros yhdistää paketit dynaamisesti, tarjoamalla publish-find-bind -mallin tavallisille Java Objekteille
- Life-Cycle – API, jota käytetään pakettien käynnistykseen, pysäyttämiseen, päivittämiseen ja niiden poistamiseen.
- Modules – Kerros, joka määrittelee miten paketti voi tuoda ja viedä koodia muihin paketteihin
- Security – Tietoturvasta huolehtiva kerros
- Execution Environment – Määrittelee mitkä metodit ja luokat ovat kyseisellä alustalla käytössä.

(OSGi Alliance, Architecture, 2018)

OSGi paketteja on kahta eri tyyppiä; application eli sovelluspaketti ja shared eli jaettu paketti. Sovelluspaketit ovat paketteja, jotka luodaan erityisesti sovellukselle, ne ovat istuntokohtaisia tai eristyksissä, eikä niitä ole tarkoitus jakaa. Jaettavat paketit eivät ole sovelluskohtaisia, vaan niitä voidaan jakaa käytettäväksi monille sovelluksille. Ne eivät voi tuoda paketteja tai palveluita sovelluspaketilta.

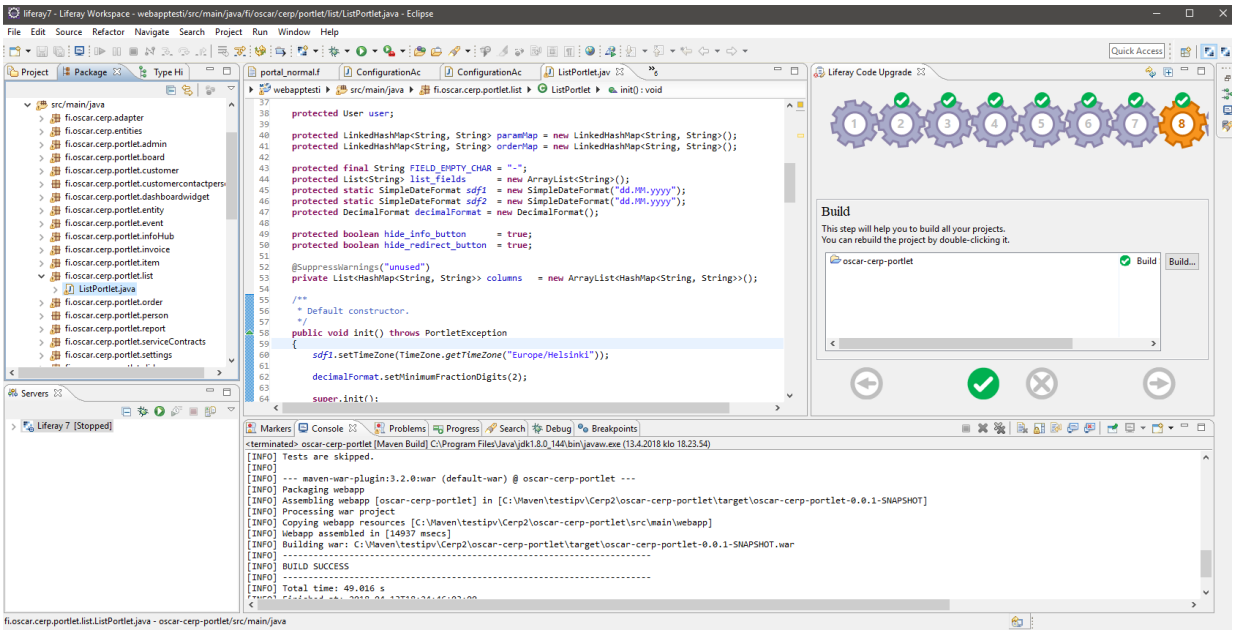
Jaettavat paketit luokitellaan edelleen kahteen alakategoriaan, use eli käyttöpaketteihin ja provision eli varustuspaketteihin. Käyttöpaketit välittävät vähintään yhden paketin sovelluspaketille, varustuspaketit välittävät sovelluspaketille vähintään yhden paketin tai palvelun sillä erolla, että niitä ei määritellä sovelluksen otsikkotiedostossa, jolloin sovellus ei tiedä miten sille välittävän paketin vaatimukset toteutetaan. (IBM, OSGi bundles, 2018)

3 KÄYTETYT TYÖKALUT

Kappaleessa esitellään lyhyesti työn tärkeimmät työkalut ja niiden käyttötarkoitus työssä. Työkalujen toiminnasta on avattu vain niiden perusperiaatteet, Eclipsen käyttöä on selvennetty kuitenkin tarkemmin työvaiheisiin liittyvissä kappaleissa.

3.1 Eclipse kehitysympäristö

Eclipse on avoimen lähdekoodin tuoteperhe. Sillä on mahdollista tehdä ohjelmistokehitystä Javalla, C/C++ ja PHP:llä. Eclipse on käytetyin Java IDE, eli kehitysympäristö. Eclipsellä on paljon sisäänrakennettuja lisäosia, kuten git-ominaisuudet ja niitä voidaan laajentaa kaupasta ladattavilla lisäosilla. Työssä käytettävä Liferay IDE on lisäosa Liferay sovelluskehitykseen. (Eclipse, 2018)
Eclipseen on myös saatavilla Liferay 7 Code Upgrade Tool koodin päivittämisen avuksi, kuvassa (kuva 3) kehitystyötä Eclipsellä ja päivitystyökalulla.



KUVA 3 Liferay kehitystä Eclipsellä

Työssä on käytetty Liferay 7-kehitykseen Eclipse Neonia ja Liferay IDE:n 3.1.3 GA3 versiota. Liferay 6.2-projektin tutkimiseen ja testiprojektien kehitykseen on käytetty Eclipse Lunaa ja Liferay IDE versiota 2.2.2 GA5.

3.2 Maven projektihallintatyökalu

Maven on kääntötyökalu, jolla voidaan hallita projektikohtaisia riippuvuuksia. Työkalulle konfiguroidaan kaikki projektissa käytettävät riippuvuudet ja liitännäiset, joita Maven käyttää sovellusta kääntäessä. Mavenia voidaan käyttää Liferay liitännäisten luontiin käyttäen Maven arkkityyppejä.

Koko työkalun ydin on POM tiedostot. Jokaisella Maven isäntäprojektiin kuuluvalla lapsiprojektilla on oma tiedosto projektin riippuvuuksien hallintaan. Alla olevassa kuvassa (kuva 4) on esimerkki isäntäprojektin POM tiedostoon lisätystä liitännäisestä.

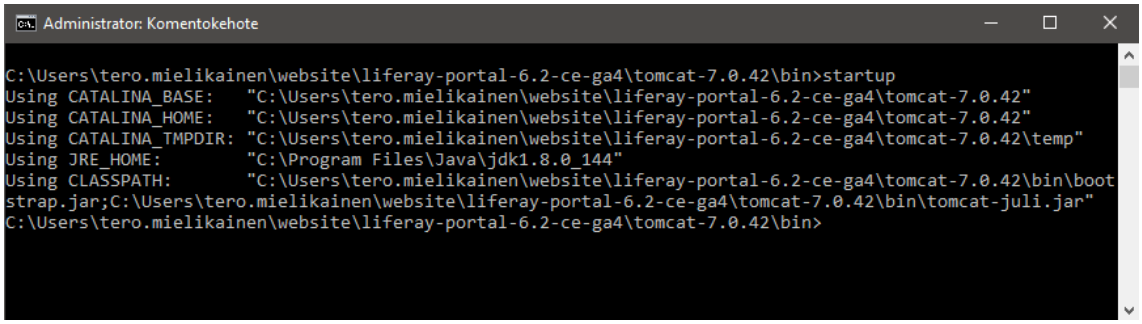
```
<plugin>
    <groupId>org.sonatype.plugins</groupId>
    <artifactId>nexus-staging-maven-plugin</artifactId>
    <version>1.6.8</version>
    <executions>
        <execution>
            <id>default-deploy</id>
            <phase>deploy</phase>
            <goals>
                <goal>deploy</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <serverId>nexus</serverId>
        <nexusUrl>http://nexus.oscar.fi:8084/</nexusUrl>
        <skipStaging>true</skipStaging>
    </configuration>
</plugin>
```

KUVA 4 POM tiedostoon lisätty liitännäinen

3.3 Command Prompt komentotulkki

Command Prompt, eli lyhyemmin cmd on Windowsin komentotulkki, jolla voidaan suorittaa erilaisia komentoja ja työkaluja, jotka toimivat komentoriviympäristössä. Esimerkiksi Node.js:n paketinhallinta npm, Maven komennot, Java komennot sekä Java koodin kääntäminen.

Opinnäytetyössä komentotulkkia on käytetty Maven komentoihin, Apache Tomcatin käynnistykseen, sekä tietokannan päivitystyökalun käyttöön. Kuvassa (kuva 5) Apache Tomcatin käynnistys komentotulkilta



KUVA 5 Tomcatin käynnistäminen cmd:llä

3.4 7-ZIP arkistointiohjelma

7-ZIP on avoimen lähdekoodin arkistointiohjelma Windowsille. Ohjelma on yksinkertainen ja sillä on kevyt graafinen käyttöliittymä. (7-ZIP kotisivut, 2018)
Liferayn sivuilta ladattu portaalipaketti sisälsi tiedostoja, joissa polku oli liian pitkä Windowsin paketinpurkuohjelmalle. 7-ZIP on kevyt ja siitä syystä oiva valinta työn tarpeisiin. Sillä on myös kätevää tarkastaa WAR ja JAR moduulien sisällä olevat tiedostot nopeasti.

3.5 GIT versionhallinta

Ilmainen ja avoimen lähdekoodin versionhallintaohjelma, jolla voidaan käsitellä pieniä tai isoja projekteja helposti projektiryhmän sisällä. Sillä mahdollistetaan monen kehittäjän samanaikainen ohjelmistokehitys. Samalla kaikki projektin koodit ovat verkossa kaikkien saatavilla. (Git kotisivut, 2018)

Git-versionhallintaa voidaan käyttää suoraan komentokehotteelta käyttäen sen omaa git bash sovellusta. Sen ympärille on rakennettu myös monta muuta sovellusta kuten SourceTree, jossa on graafinen käyttöliittymä. Työssä käytetyssä Eclipse kehitystyökalussa on myös git-liitännäinen, jolla voidaan projektit hakea suoraan kehitystyökaluun. Työssä käytettiin Eclipsen git-liitännäisen lisäksi Windows ympäristössä toimivaa TortoiseGit ohjelmaa.

3.6 Notepad++ tekstieditori

Avoimen lähdekoodin tekstieditori, joka tukee monia ohjelmointi- ja merkintäkieliä (engl. markup language), kuten Java, HTML, Javascript ja XML. Notepad++ on nopea ja kevyt käyttää. Työssä sitä on käytetty koodien pieniin muutoksiin, joita on turha avata raskaalla ohjelmalla, kuten Eclipse.

4 PROJEKTIN RAKENNE

Seuraavaksi analysoidaan varsinaisen työssä käytettyjen liitännäisprojektien kansiorakennetta. Ohjelmistokehitys ja arkkityyppien luominen on tehty Eclipse Neonin Liferay IDE lisäosalla.

4.1 Maven projekti

Kaikki projekteihin luodut liitännäiset on käännetty käyttäen Maven työkalua. Kääntämiseen tarvittavat liitännäiset ja riippuvuudet luetaan pom.xml tiedostosta. Jokaisella Maven isäntäprojektiin liitetyllä projektilla on oma tiedosto, johon määritellään projektikohtaisia riippuvuuksia.

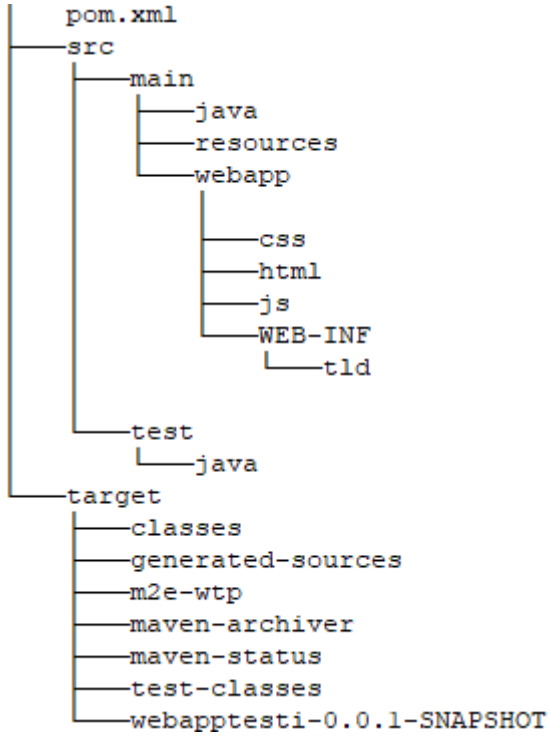
Liitännäiset on luotu käyttäen Maven-arkkityyppejä. Arkkityypeillä saadaan generoituna valmis kansiorakenne sovellusten kehittämiseen. Seuraavissa kappaleissa kerrotaan portlet ja teema-liitännäisen Maven-arkkityyppien kansiorakenne ja mitä kukin kansio pitää sisällään.

4.1.1 Portlet liitännäinen

Portletit on nykyisellä versiolla luotu käyttäen liferay-portlet-archetype Maven arkkityyppiä. OSGi mallin mukainen Maven arkkityyppi Liferay MVC portleteille on com.liferay.project.templates.mvc.portlet, joka luo automaattisesti tarvittavat OSGi mallin mukaiset BND paketointitiedoston ja Gradle käännöstiedoston. Spring MVC portleteille on oma com.liferay.project.templates.spring.mvc.portlet, jota työssä käytettiin yksittäisen Spring MVC portletin päivittämiseen. Kun työssä päädyttiin päivittämään vain yksi Liferay MVC portlet WAR paketiksi, on siihen tarkoitukseen käytetty projekti luotu käyttäen liferay-web-archetype arkkityyppiä.

Kansiorakenteen kannalta näillä arkkityypeillä ei ole varsinaisesti eroa, vaan syy monien arkkityyppien käyttöön on esimerkiksi OSGi mallin vaatimien liitännäisten ja tiedostojen automaattinen luominen projektiin, joka nopeuttaa kehitystyötä. Alla olevassa kuvassa

(kuva 6) on esitelty liferay-web-archetype arkkityypillä luodun portlet liitännäisen kansiorakenne.



KUVA 6 Portlet projektin rakenne

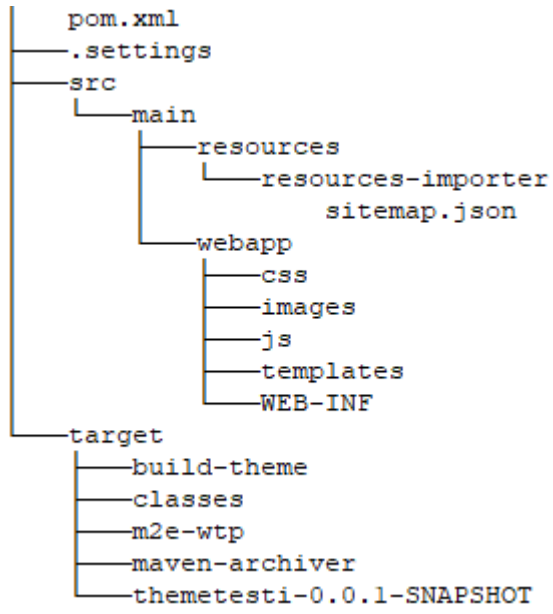
Seuraavassa luettelossa avataan rakenteen oleellisimmat kansiot, sekä niiden sisältöä:

- src/main/java
  - Sisältää projektin kaikki käytettävät Java luokkatiedostot.
- src/main/webapp
  - Kaikki front-end resurssit ja portletien määritystiedostot.
  - Css-kansiossa portletin käyttämät tyylitiedostot
  - Js-kansiossa kaikki hyödynnettävät Javascript-kirjastot sekä
  - html kansio sisältää kaikki portletien JSP-tiedostot
- src/main/WEB-INF Sisältää portlettien config-tiedostot
  - portlet.xml
    - Määritellään JSR standardin mukaiset konfiguraatiot. Tiedostoon määritellään portletin tiedot, kuten esimerkiksi nimi ja mitä luokkaa se hyödyntää.
  - liferay-portlet.xml

    - Portletin rekisteröintiin tarkoitettu tiedosto, jotta Liferay Portal osaa käyttää määritettyä portletia. Tiedostoon voidaan määritellä monia ominaisuuksia, kuten portletin käyttämien CSS- sekä Javascript-tiedostojen sijainti. Voidaan määrittää, onko portletin asettaminen monelle sivulle sallittua vai ei.
  - liferay-display.xml
    - Käytetään portletien organisointiin eri kategorioihin portaalin lisäysikkunassa (Roufid Understanding Liferay portlet configuration files, 2016)
  - Liferay-plugin-packgage.properties
    - Nimensä mukaisesti liittyy paketointiominaisuuksiin. Aiemmin tähän tiedostoon määriteltiin liitännäisen tarvitsemat JAR tiedostot, mutta ominaisuus on poistunut uudessa versiossa.
- target
  - Projektin käännöksen jälkeiset ja sen yhteydessä luodut resurssit, joiden avulla luodaan lopullinen portlet palvelimelle siirrettävässä JAR- tai WAR-paketti muodossa.
- pom.xml
  - Portletissa tarvittavat Maven-riippuvuudet, sekä kääntämisessä käytettävät liitännäiset

4.1.2 Teema liitännäinen

Alkuperäinen projektissa käytetty teema on luotu käyttäen liferay-theme-archetype Maven arkkityyppiä. Päivitykseen lopulta haluttiin saada ainoastaan teeman tuomat CSS tiedostot, jotta portletit saadaan toimimaan ja näyttämään kunnollisilta uudessa versiossa. Tähän tarkoitukseen luotiin erillinen testiprojekti käyttäen uudempaa com.liferay.project.templates.theme arkkityyppiä. Alla olevassa kuvassa (kuva 7) on kyseisellä arkkityypillä luodun teema-projektin kansiorakenne.



KUVA 7 Teema testiprojektin rakenne

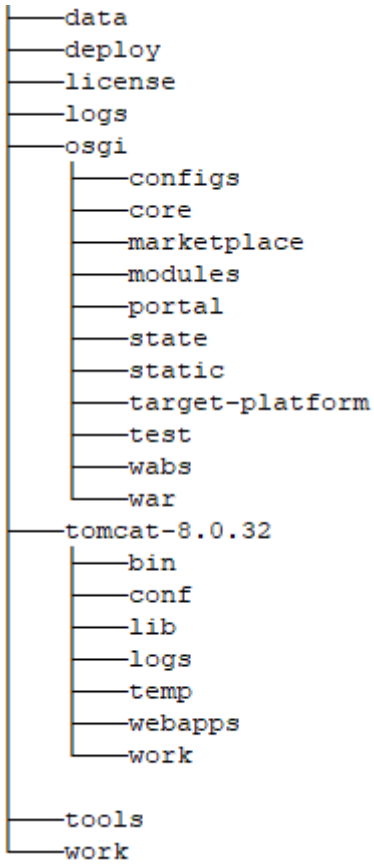
Rakenteen oleellisimpia kansioita ja niiden sisältämiä tärkeitä tiedostoja on selitetty alla olevassa luettelossa:

- src/main/resources/resources-importer/sitemap.json
  - Tiedosto, joka määrittelee mallipohjan web-sisällölle, sivuille ja portleteille sivustoilla. (Liferay Creating the Sitemap.json File, 2017)
- src/main/webapp
  - Projektin käyttämät tyylitiedostot sijaitsevat css-kansiossa
    - Tärkein tiedosto on _custom.scss, johon sijoitetaan kaikki itse tehdyt CSS tyylimuutokset.
  - Käytettävät Javascript-tiedostot sijaitsevat js-kansiossa
    - Itse tehdyt Javascriptit tehdään main.js tiedostoon.
  - Sivuilla käytettävät kuvat images kansiossa
- templates
  - Freemarker mallitiedostot portaalin näkymän ominaisuuksiin.
  - portal_normal.ftl
    - Samankaltainen internetsivujen index.html tiedoston kanssa. Tiedosto on solmukohta kaikille muille teema malleille.
  - portal_pop_up.ftl
    - Teema-malli ponnahdun teeman portletien dialogeille.
- target sisältää kääntämisen jälkeiset resurssit joista luodaan palvelimelle sijoitettava pakettitiedosto, sekä varsinaisen valmiin teema, WAR paketin.

(Liferay Theme Reference Guide, 2017)

4.2 Tomcat Server Bundle

Liferayn 6.2 version Tomcat palvelimen kansiorakenne on hyvin samanlainen. Suurin ero näiden välillä on se mihin moduulit sijoitetaan. Liferayn 7.0 versiossa hyödynnetään OSGi paketteja ja tämä aiheuttaa muutoksia moduulien sijainteihin. Vanhassa versiossa moduulit sijoitetaan tomcat/webapps kansioon. Liferay 7 versiossa moduulit sijaitsevat tyypistä riippuen joko osgi/modules tai osgi/wabs kansiossa. Kuvassa (kuva 8) on Liferay Portal Tomcat bundlen kansiorakenne.



KUVA 8 Tomcat palvelimen kansiorakenne

Tomcat-palvelimeen liittyy useita kansioita, joiden käyttötarkoituksia on selvennetty seuraavassa luettelossa:

- data:
  - Kansio sisältää sisäänrakennetun HSQL-tietokannan, Liferay Portalin tietosäilön, sekä hakuindeksit. Oletuksena Liferay Portal käyttää HSQL tietokantaa, mutta se

on tarkoitettu lähinnä esittely- ja koekäyttöön. Tässä projektissa HSQL on korvattu MySQL-tietokannalla

- deploy:
  - Kansio johon siirretään liitännäisten paketoidut tiedostot, jotka puretaan Tomcatilla ajettavaksi. Kansiosta tiedostot siirretään automaattisesti niille kuuluvaan kansioon. Kansioon voidaan siirtää Legacy tyyppisiä WAR , Liferay 7.0 CE mallin JAR tai kaupasta ostettuja LPKG-tiedostoja.

- license:
  - Tekijänoikeus- ja versiotiedostot
- logs:
  - Sisältää Portalin logit, kirjaa kaikki tapahtumat Portalin sisällä, kuten Portletin tai muiden moduulien asennuksessa tapahtuvat mahdolliset virheet. Sisältää arvokasta tietoa virheiden korjauksessa.
- osgi:
  - Hakemisto, joka sisältää portaaliin sijoitetut moduulit sekä muutamat Liferay OSGi runtime määrittelytiedostot.
  - configs:
    - Sisältää komponenttien konfiguraatiotiedostot
  - core:
    - Kansiossa on Liferay Portalin keskeiset moduulit, kuten OSGi sekä Bootstrap
  - marketplace:
    - Sisältää kaupasta ladatut sovellukset ja sovelluskokoelmat
  - modules:
    - Sisältää sovellukset, jotka on itse siirretty palvelimelle deploy-kansiota käyttäen. Kansioon sijoitetaan ainoastaan Liferay 7 CE JAR moduulit
  - portal:
    - Sisältää vähemmän keskeiset Portalin moduulit, oletuksena kansio on tyhjä
  - state:
    - OSGi:n sisäiset tiedostot, kuten OSGi-pakettien asennukset ja pakettien säilytystila yms.
  - target-platform:
    - Kohdealustan hakemisto
  - test:

    - Sisältää moduulit, jotka tukevat testausta
  - war
    - Sisältää *.war päätteiset* moduulit. Liferay 6.2 moduulit asennetaan tästä kansiosta
  - wabs
    - Koska joitakin sovelluksia ei voi muuttaa JAR muotoon, ne voidaan muuttaa WAB moduuleiksi. Jotkut WAR moduulit eivät suoraan ole yhteensopivia OSGi mallin kanssa, siksi ne muutetaan WAB muotoon käyttäen WAB Generatoria. Tämä kansio sisältää muutetut WAR moduulit WAB muodossa.

- tomcat-8.0.32
  - Sisältää Tomcatin palvelimen
  - bin
    - Sisältää kaikki tarvittavat komentojono-tiedostot, esimerkiksi käynnistykseen(startup) ja sammuttamiseen(shutdown). Näitä komentoja voidaan ajaa Windowsin komentokehotteella.
  - conf
    - Serverin konfiguraatio-tiedostot, kuten Catalina, Tomcat yms.
  - lib
    - Serverin toimintaan vaadittavat kirjastot, sisällä on ext-kansio, jossa on myös portletien toimintaan vaikuttavia kirjastoja. Esimerkiksi portal-kernel.jar sekä portlet.jar yms.
  - logs
    - Käynnistyksen aikana kerätyt tapahtumatiedot, esimerkiksi Catalinan käynnistyminen, ja HTTP kutsut ja niiden tilatiedot.
  - temp
    - Kun moduuleja sijoitetaan Portaliin, luodaan väliaikaistiedostot, jotta uudelleensijoittaminen on helpompaa ja luotettavampaa.
  - webapps
    - Kansio web-sovelluksille, Liferay 6.2 luodut sovellukset siirtyvät deploy-kansion jälkeen tähän kansioon. Liferay 7 versiossa kansiossa on pelkästään ROOT sovellus, joka sisältää Portalin oletusportletit ja resurssit, kuten kuvakkeet.
  - work
    - Jasper-moduulin työkansio, Jasper on Tomcatin JSP-moottori.

(Liferay Installing Liferay Portal, 2017)

4.3 Tietokanta

Liferayn Tomcat-paketti käyttää oletuksena HSQL-tietokantaa. Kokeilu- ja esittelytarkoituksiin tämä on sopiva (Liferay Wiki, Database Configuration, 2009), mutta tarjolla on paljon tehokkaampia teknologioita varsinaisen tuotantosovelluksen tarkoituksiin. Oscar Softwarella on käytössään Oraclen, Firebirdin sekä MySQL:n tietokantoja. cERP:ssä on käytössä MySQL 5.5.54, uudessa Liferay 7 versiossa vaaditaan vähintään 5.6.4 versio, joten se täytyy päivittää myös päivitysprosessissa. Päivitettäessä päädyttiin sillä hetkellä uusimpaan 5.7.16 versioon.

4.4 Muutokset

Kun liitännäisiä ajetaan palvelimella WAR muodossa, rakenteellisesti versioiden erot eivät ole kovinkaan mullistavia. Portlet-, teema-, sekä hook-projektit pysyvät päivityksen läpi lähes muuttumattomana. OSGi malli tuo muutoksia moduulien paketointiin, eli ne paketoidaan Java-arkistoiksi eli JAR moduuleiksi. On myös mahdollista kääntää WAR paketit WAB Generatorin avulla WAB eli web application bundleiksi.
Palvelimen puolella kansioissa on muutoksia lähinnä OSGi-mallin tuomien modulaarisuusmuutosten aiheuttamia.

5 PÄIVITYS

Seuraavat kappaleet käsittelevät työn päivityksen tavoitteita, mahdollisuuksia, päivitysprosessia ja siihen liittyviä haasteita. Päivitysprosessissa noudatetaan Liferayn omia dokumentaatioita päivittämiseen.

5.1 Tavoitteet

Alkuperäisiksi tavoitteiksi asetettiin koko Oscar Softwaren cERP-järjestelmän päivitys. Kuitenkin nopeasti huomattiin projektin olevan liian suuri kokonaisuudessaan päivitettäväksi, joten päädyttiin päivittämään osa kerrallaan. Tärkein päivitettävä kokonaisuus on sivustolla käytettävät portletit. Työn edetessä törmättiin useisiin ongelmiin, esimerkiksi Spring MVC portletien kanssa. Nämä ongelmat sivuutettiin ja lopulta oli siirryttävä päivittämään Liferay MVC, sekä muita jäljelle jääneitä portleteja yksi kerrallaan. Yritys toivoo myös uudelta versiolta modulaarisuutta ja työn purkamista järkevämpiin osiin, jolloin portlet kerrallaan päivittäminen ei ollut askel taaksepäin.
Päivityksen toisena osiona oli tarkoitus päivittää teema uuteen versioon, koska osa portleteista käyttää teemojen tuomia resursseja, kuten JavaScript-kirjastoja.

5.2 Mahdollisuudet

Päivittäessä on mahdollista päivittää projekti paketoitavaksi kolmeen eri muotoon. Alkuperäinen projekti sijoitettiin palvelimelle WAR arkistointipaketteina. Tämä onnistuu myös uudella versiolla, kuitenkin niin, että koodiin tehtiin tarvittavat muutokset ennen moduulin sijoittamista palvelimelle. Spring MVC portleteilla tämä on ainoa mahdollinen keino päivittää projektissa käytetyt portletit, johtuen sen perustumisesta Java EE teknologiaan.

Toinen mahdollinen päivityskeino on paketoida liitännäiset JAR muodossa, jota OSGi-malli myös käyttää. Pelkän koodin kääntämisen lisäksi tähän tarvitaan BND työkalun paketointitiedosto. Tähän tiedostoon määritellään OSGi paketin käyttämät resurssit ja Java luokat.

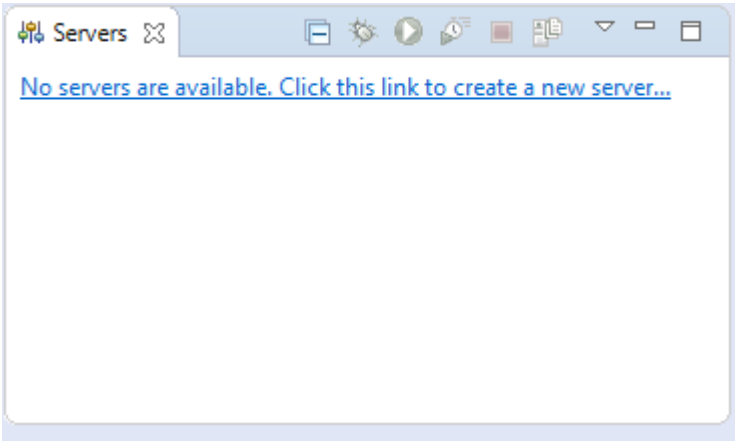
Kääntäessä projektia, luodaan arkistopakettiin metatiedosto MANIFEST.MF. Kyseinen tiedosto pitää sisällään tietoja, kuten paketin luojan tiedot, paketointiversion, sekä listan tiedostoista, joita sovellukset käyttävät. WAR paketeissa kyseiseen tiedostoon ei lisätä OSGi paketointidirektiivejä, eikä sen sisältämiä tiedostoja.

Kolmas keino päivittää projekti on muuntaa tavalliset WAR paketit WAB muotoon käyttäen WAB Generaattoria. WAB, eli Web Application Bundle on yksinkertaistettuna arkisto, jolla on sama asettelumalli kuin WAR arkistopaketilla, mutta se sisältää MANIFEST.MF metatiedostossa OSGi pakettidirektiivejä, eli se on toisin sanoen OSGi mallin paketti. (Liferay, Using WAB Generator)

5.3 Palvelimen lisääminen Eclipsessä

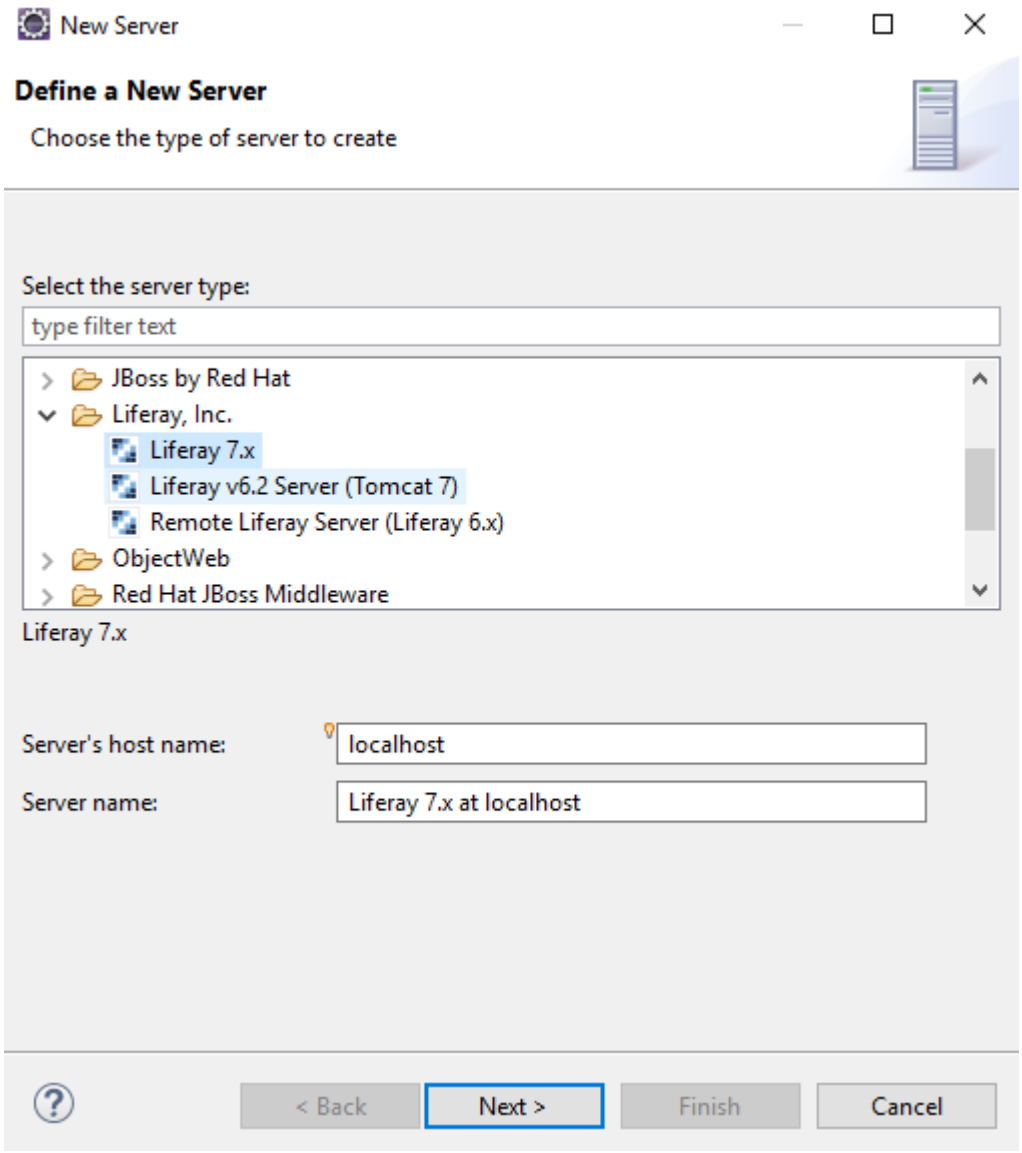
Tomcat-palvelinta voidaan käyttää täysin erillään Eclipse-ympäristöstä käynnistämällä ja sijoittamalla liitännäiset palvelimelle Mavenilla komentokehotteelta. Päivittäessä projektin koodeja Code Upgrade Tool -työkalun avulla, tarvitaan palvelimen ajoympäristöä tiettyjen prosessien, kuten hook liitännäisten päivittämiseksi erillisiksi moduuleiksi.

Käännettyjen moduulien käyttöönotto palvelimella on myös yksinkertaisempaa, kun projekteja voidaan siirtää fyysisesti ikkunassa vetämällä hiirellä palvelimelle. Palvelimet lisätään Eclipseen seuraavasti. Liferay IDE:n aloitusruudussa on oma näkymä palvelimille, joka on nähtävissä kuvassa (kuva 9).



KUVA 9 Palvelinnäkymä

Palvelimen lisääminen on yksinkertaista, klikataan kuvassa näkyvää linkkiä, jolloin aukeaa kuvan (kuva 10) mukainen palvelimen luomisikkuna.



KUVA 10 Palvelimen luonti Eclipsessä

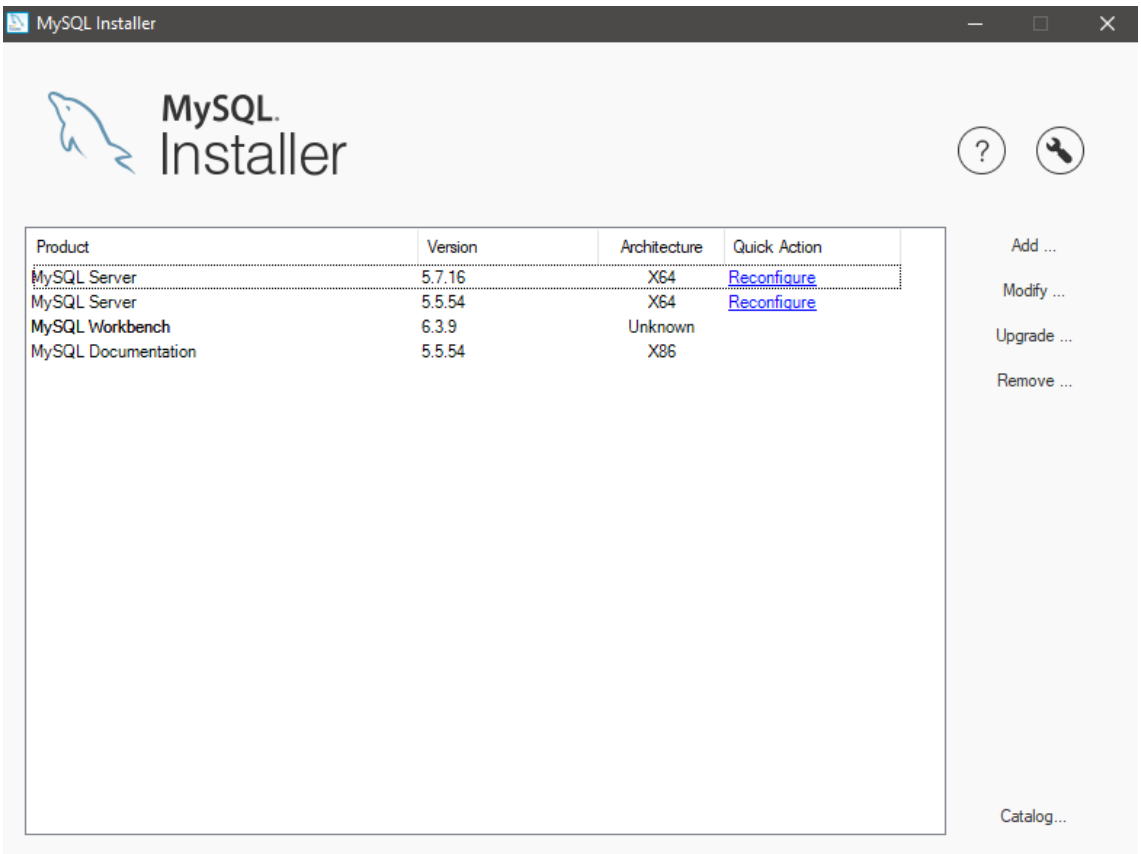
Valikosta etsitään Liferay, Inc-kansio, jonka alta valitaan halutun Liferay-version palvelin. Kun palvelimen versio on valittu, seuraavassa valikossa määritellään palvelinkansion sijainti, sekä mitä JRE versiota halutaan käyttää. Palvelimen luominen on sama Liferay-versiosta riippumatta. Päivittäessä molemmat tulee olla lisättynä, mutta kehitysvaiheessa tarvitaan vain Liferay 7-palvelin.

5.4 Prosessi

Liferay Portalin päivittäminen koostuu pääasiassa kahdesta osasta, Liferay Portal asennuksen ja tietokannan päivittämisestä. Liferayn dokumentaatioista löytyy ohjenuorat ja esitiedot päivityksessä onnistumiseen. (Liferay Upgrading to Liferay Portal, 2017)
Kun portaali on onnistuneesti päivitetty, voidaan aloittaa liitännäisten koodien ja muun niihin liittyvien osien päivittäminen. Seuraavissa kappaleissa käsitellään sekä Liferay Portalin, että liitännäisten päivittämistä.

5.4.1 Tietokanta

Tietokannan päivitys aloitettiin projektissa käytetyn MySQL-palvelimen uuden version lataamisella ja asentamisella. Koska samanaikaisesti käytettiin myös vanhaa MySQL-palvelinta, siirrettiin uusi palvelin uuteen porttiin. Vanha palvelin on portissa 3306 ja uusi portissa 3310, näin mahdollistetaan yhdenaikainen toiminta. MySQL palvelin asennettiin käyttäen MySQL Installer 1.2 versiota. Kuvassa (kuva 11) näkyy kaikki asennetut tuotteet.

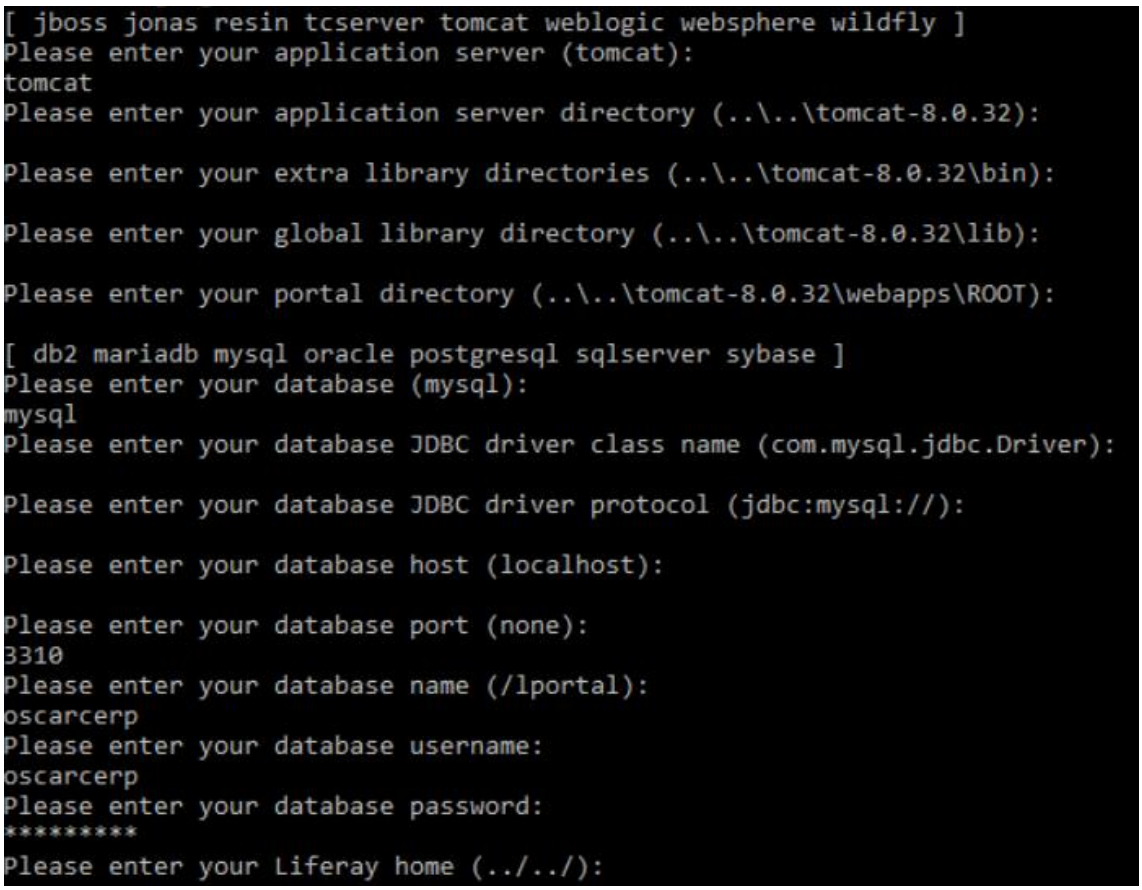


KUVA 11 MySQL Installer ikkuna

Sivussa olevilla painikkeilla voidaan lisätä ja muokata tuotteita, sekä päivittää ja poistaa tuotteita luettelosta.

5.4.2 Palvelin

Liferay Portal Tomcat-paketissa on uusi palvelimen versio, mutta sen ytimen ja tietokannan liitokset täytyy päivittää käyttäen mukana tulevaa DB Upgrade Client komentokehotteella toimivaa työkalua. Työkalu käynnistetään cmd komennolla *java -jar com.liferay.portal.tools.db.upgrade.client.jar*, sekä siihen voidaan lisätä tietoja, kuten kieli, maa ja aikavyöhyke. Käynnistyksen jälkeen syötetään asennetun palvelimen hakemiston lisäksi tietokannan tiedot, kuten kannan nimi, portti, käyttäjätunnus sekä salasana. Kuvassa (kuva 12) työkalun käyttöprosessi.



KUVA 12 Upgrade Client työkalun käyttöprosessi

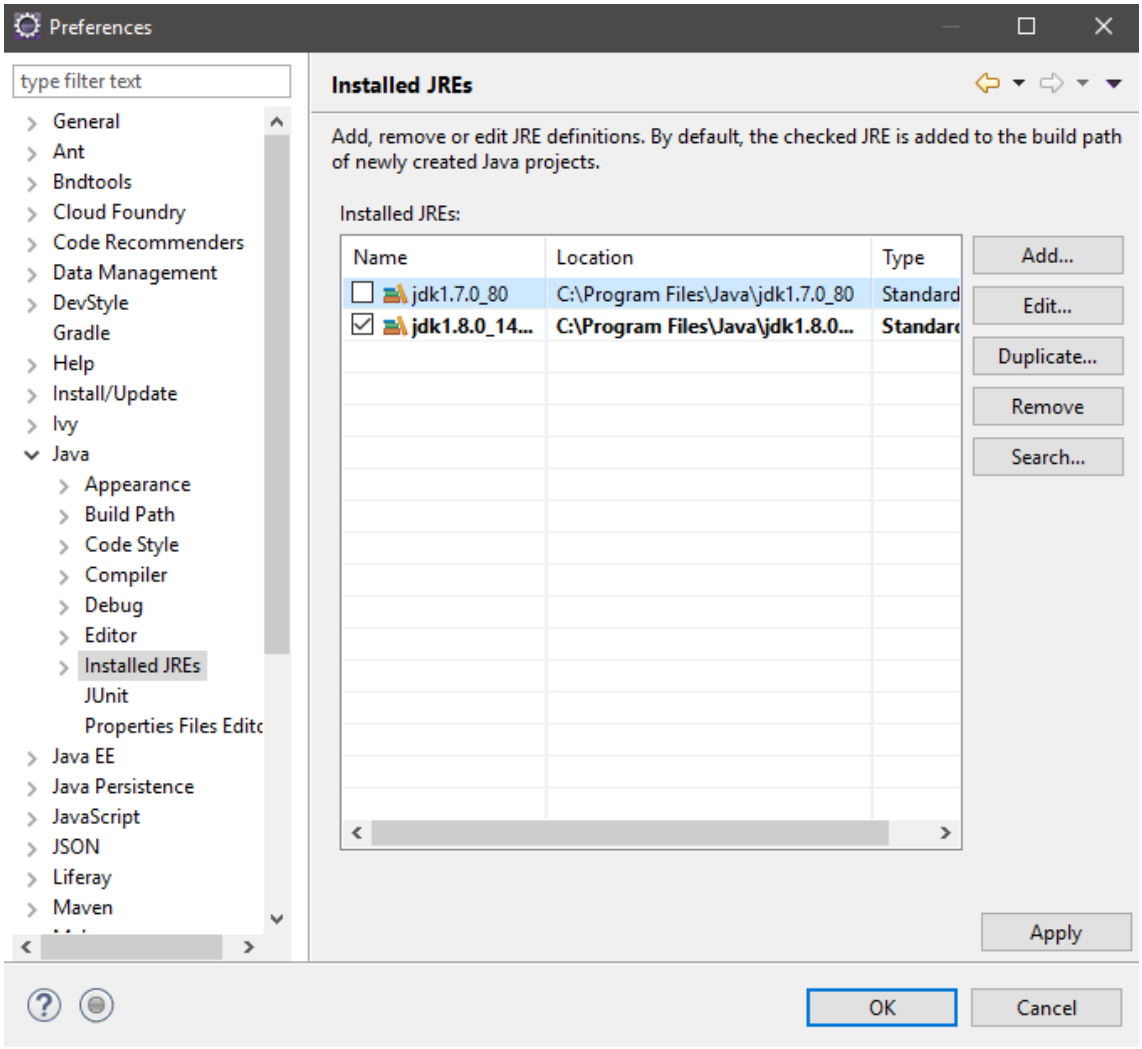
Kun prosessi on valmis, se ilmoittaa konsolissa onnistuiko päivittäminen ja tarkistaa onko kaikki osat päivitetty onnistuneesti. Päivitystyökalu luo prosessin jälkeen tiedostoja, joita Portal lukee käynnistäessä.

Päivityksessä tuli bugi, joka aiheutti Liferay Portalin kotihakemistossa tiedostoon, *tools/portal-tools-db-upgrade-client/**portal-upgrade-database.properties***, väärän URL:in tietokantaan. Ongelman korjaamiseksi täytyi tiedostoon manuaalisesti korjata väärin asetettu parametri. Tiedostoon kirjaamisessa työkalu ei jostain syystä luonut vino-viivaa tekstiin, ja se täytyi siihen itse lisätä. Korjauksen jälkeen todettiin Portalin päivitys tietokannan osalta onnistuneeksi.

5.4.3 Javan päivittäminen

Liferay Portal 7.0 käyttää uudempaa Java 8 versiota, joten vanha täytyy päivittää. Javan itsensä päivittäminen ei ole suuri operaatio; ladataan verkkosivuilta uusin Java 8 JDK, eli kehittäjätyökalut ja asennetaan haluttuun kansioon. Javasta asennettiin versio 8u144.

Pelkän Javan päivittämisen lisäksi täytyy varmistaa, että Eclipseen on määritetty käyttöön oikea JRE. Tämä tapahtuu Eclipsen valikosta Window → Preferences → Java → Installed JREs. Kuvassa (kuva 13) on näkymä JREn valitsemiseen.

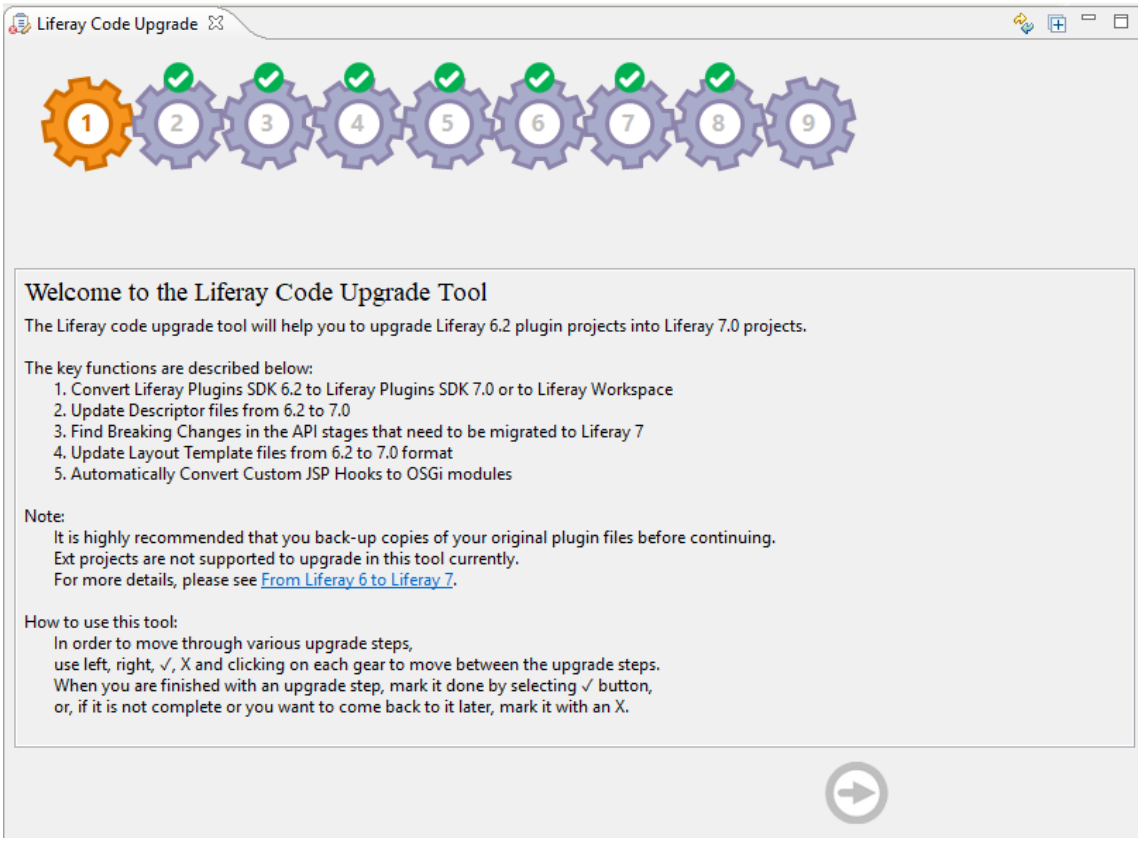


KUVA 13 Eclipsen JRE valinta

Päivitetyssä versiossa valittiin käytettäväksi Java 8 versiota, joten listassa on oltava 1.8 JDK lisättynä. Add.. painikkeella voidaan lisätä uusi, mikäli JDK puuttuu. Täytyy myös muistaa, että kyseinen JRE on valittuna kuten kuvassa.

5.4.4 Liferay Code Upgrade Tool -työkalun käyttö

Koodin päivittämisen tueksi Liferay IDE sisältää Liferay Code Upgrade Tool -työkalun, jonka aloitusnäkymä on kuvassa (kuva 14).



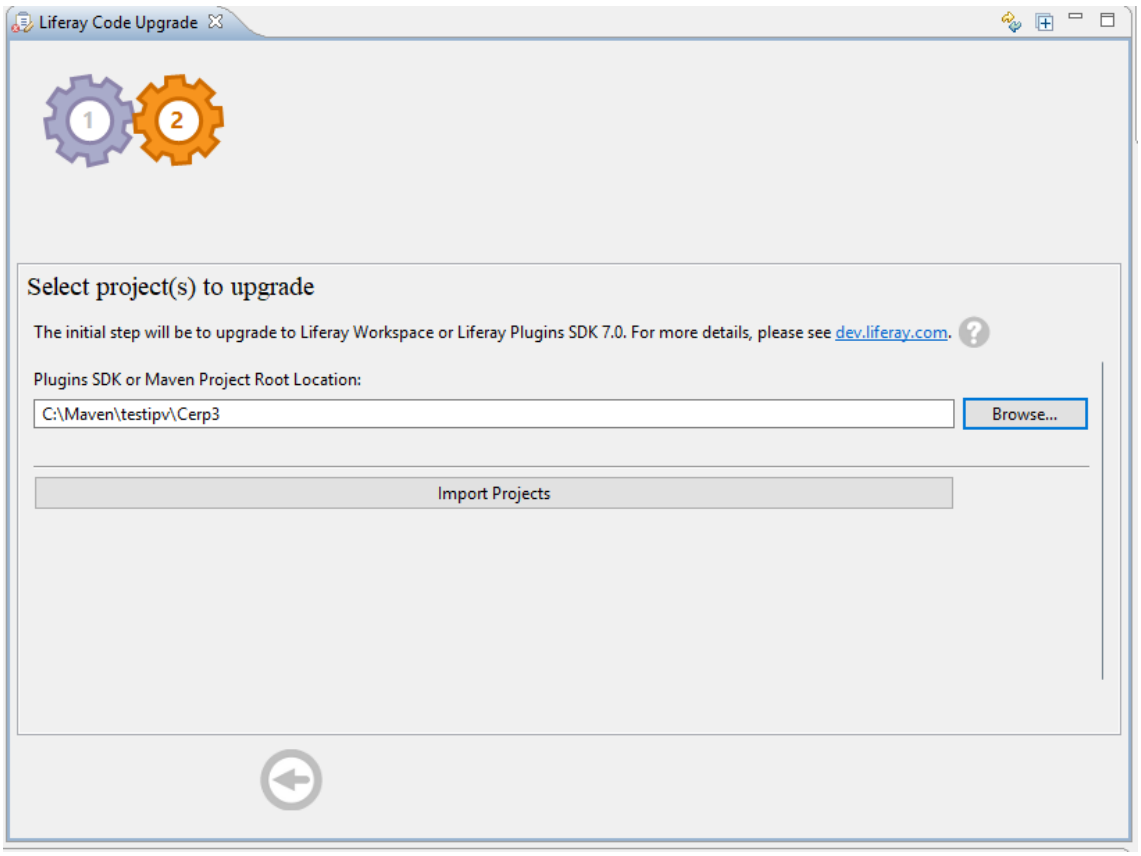
KUVA 14 Liferay Code Upgrade Tool -työkalun ensinäkymä

Työkalun käyttö on suoraviivaista ja melko yksinkertaista ja prosessi on jaettu moneen osaan, jotta päivittäminen on yksinkertaisempaa. Työvaiheet löytyvät alla olevasta taulukosta (taulukko 1).

TAULUKKO 1 Koodin päivitystyökalun työvaiheet, hieman muokattu (Liferay Adapting to Liferay 7 API, 2017)

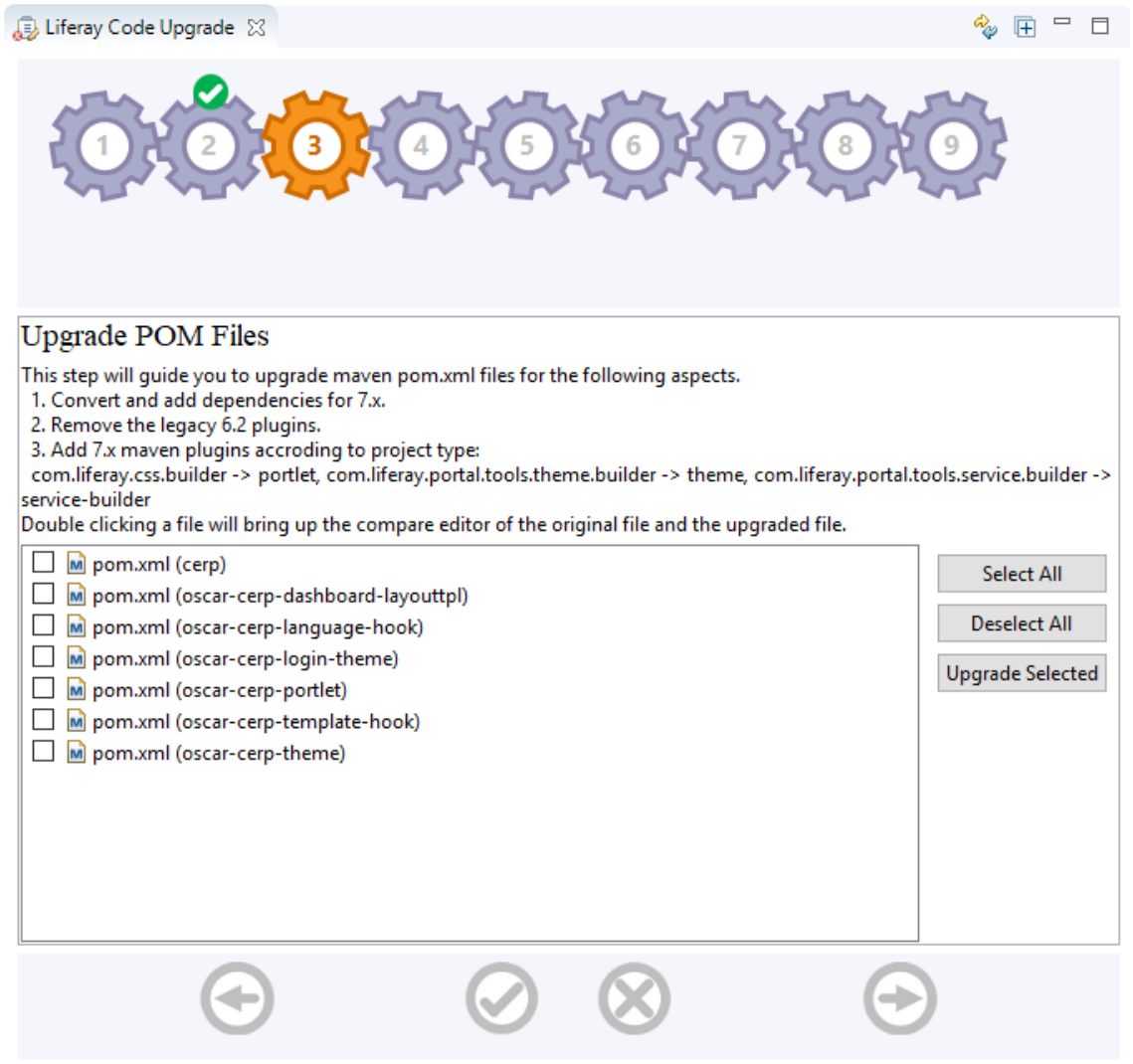
Step	**Name**	**Description**
1.	Welcome to the Liferay Code Upgrade Tool	Introduces the Code Upgrade Tool.
2.	Configure the Project	Imports an existing Maven project or Plugins SDK project, and prepares its plugins for upgrading.
3. (Maven only)	Upgrade POM Files	Upgrades POM files so they use the latest Maven plugins and Liferay dependencies.
4.	Find Breaking Changes	Finds breaking changes, describes them, and prescribes adaptations (some of which can be applied automatically).
5.	Upgrade Descriptor Files	Moves descriptor files to their new versions.
6.	Upgrade Layout Templates	Upgrades layout templates.
7.	Convert Custom JSP Hooks	Converts JSP hooks to modules or module fragments.
8.	Build	Compiles the plugins
9.	Summary	Lists each upgrade step’s status

Ensimmäinen varsinainen työvaihe on projektin tuominen työkalulle. Projektin päivittämisessä on hieman erilainen polku, riippuen siitä onko kyseessä Plugins SDK vai Maven projekti. Työn projekti on rakennettu Mavenilla, joten toista vaihtoehtoa ei tarvitse tässä käsitellä. Projektin sijainti lisätään kuvan (kuva 15) mukaisesti valitsemalla, tai manuaalisesti kirjoittamalla oikea polku.



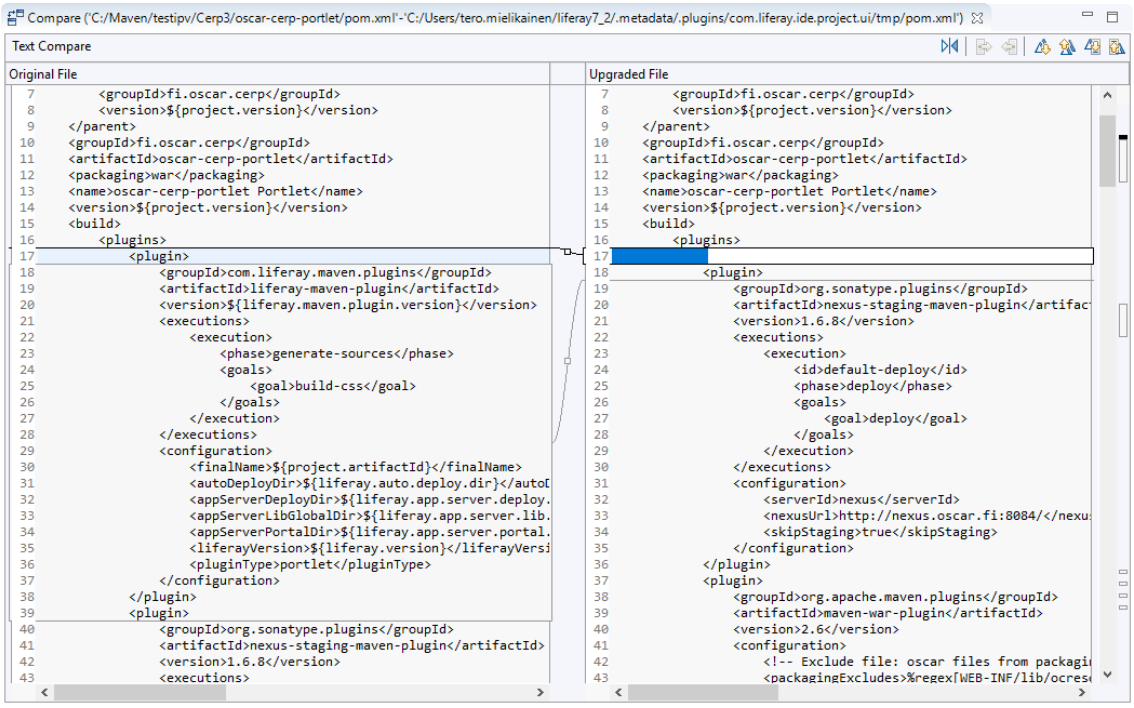
KUVA 15 Projektin tuonti päivitettäväksi

Kun projektien tuonti on onnistunut, näkyvät lähdekoodit Eclipsen Project Explorer ikkunassa. Seuraava askel on päivittää Maven projektien pom.xml tiedoston riippuvuudet ja liitännäiset. Työkalu tunnistaa automaattisesti kaikki tiedostot, jotka vaativat päivitystä, kuvassa (kuva 16) näkyy kaikki päivitettävät tiedostot.



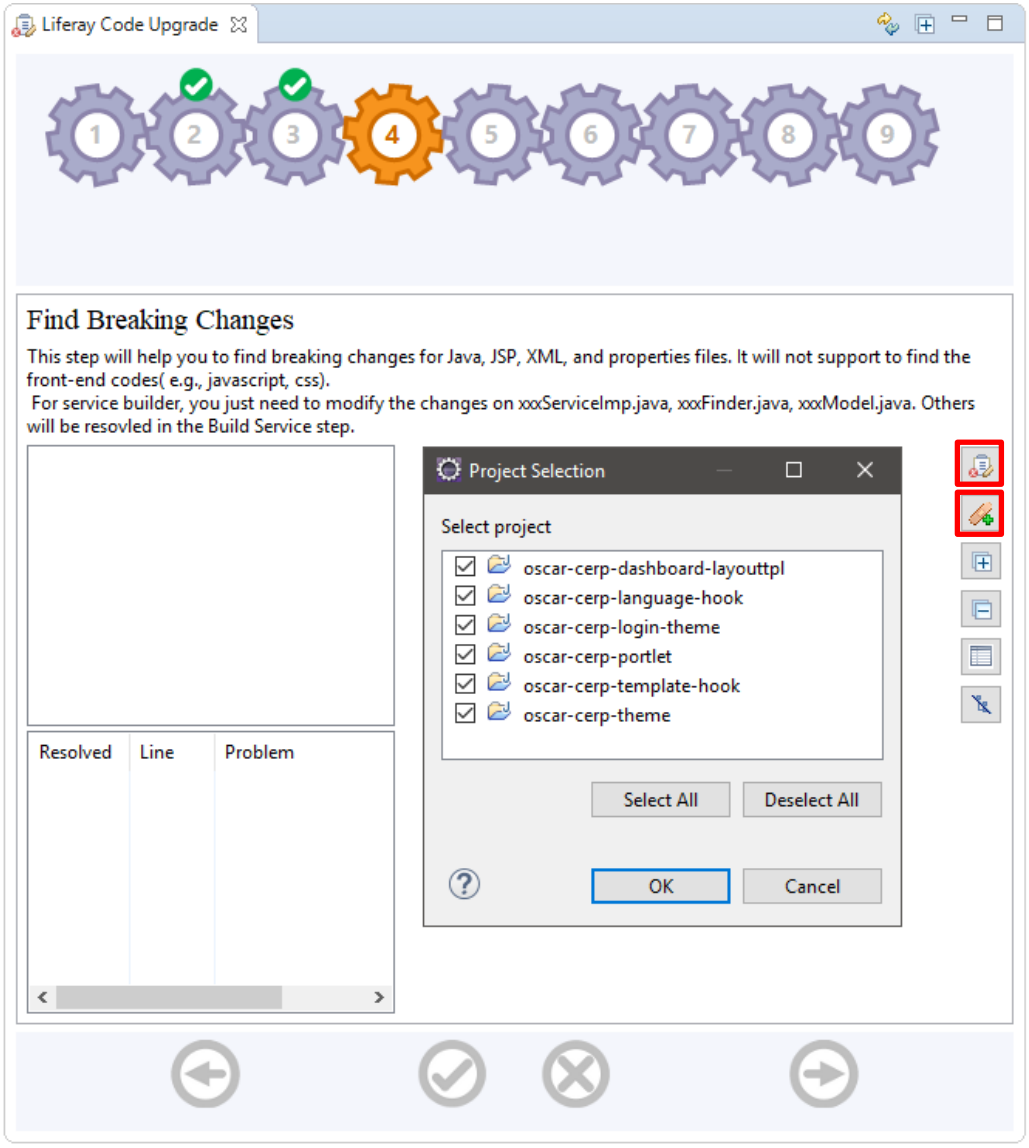
KUVA 16 POM tiedostojen päivittäminen

Työkalu hoitaa muutosten asettamisen automaattisesti. Valitaan kaikki tiedostot klikkaamalla Select All painiketta. Seuraavaksi jatketaan Upgrade Selected painikkeella, jolloin kaikkien tiedostojen muutokset suoritetaan. Muutoksia on mahdollista tutkia tuplaklikkaamalla jotakin tiedostoa listasta. Muutokset esitetään kuvan (kuva 17) mukaisesti.



KUVA 17 POM tiedoston muutoksia

Onnistuneiden muutosten jälkeen seuraava vaihe on etsiä työkalun avulla vanhentuneet koodit, jotka voivat rikkoa toteutuksen ja päivittää ne uuden API:n mukaiseksi. Kuvassa (kuva 18) Breaking Changes työvaiheen näkymä.

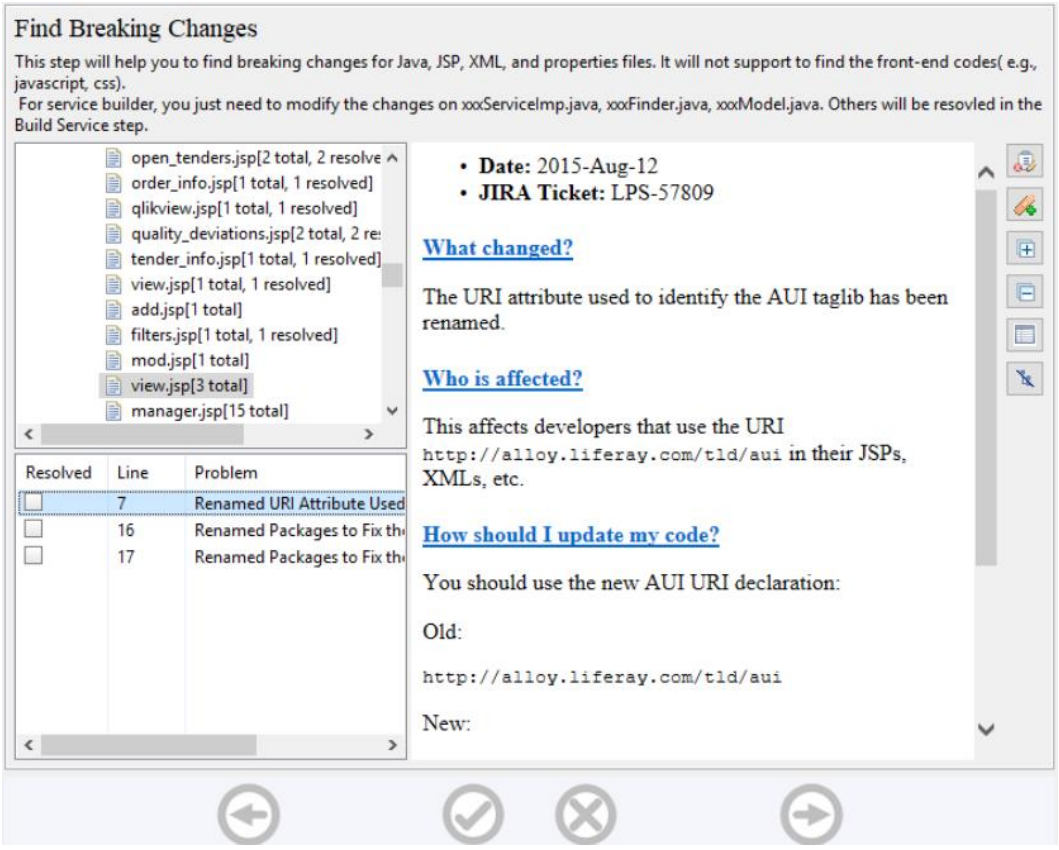


KUVA 18 Breaking Changes työvaihe

Kuten kuvan yhteydessä todetaan, työkalu ei lue front-end koodeja, vaan etsii muutoksia Java, JSP ja XML tiedostoista, sekä properties eli ominaisuustiedostoista. (Liferay Code Upgrade Tool, 2017)

Kuvaan on korostettu punaisilla ääriviivoilla painikkeet, joilla muutosten hakeminen voidaan aloittaa. Ylempi painike ( ) on kaikkien virheiden manuaaliseen korjaamiseen, eli työkalu vain listaa tarvittavat muutokset ja ne tulee itse korjata koodiin. Alempi painike ( ) hakee muutoksia ja yrittää automaattisesti korjata yksinkertaisimmat, kuten pakettien nimen muutokset ja versiotiedot ominaisuustiedostoihin. (Liferay Adapting to Liferay 7s API, 2017)

Työkalun automaattinen korjaus ei kuitenkaan toiminut, joten muutosten korjaaminen tehtiin käsin jokaiseen tiedostoon. Kuvassa (kuva 19) on esimerkkitapaus tarvittavasta muutoksesta.



KUVA 19 Esimerkki tarvittavista muutoksista

Vasemmalla näkyy tiedostot, joissa on jotakin muutettavaa, sen alapuolella tarkemmin rivit tiedostossa, joissa kyseiset muutosta kaipaavat koodit ovat. Oikealla puolella tekstikentässä lukee referenssitietoa, kuten mikä on muuttunut ja miten koodia on päivitettävä.

Kaikki päivittämiseen liittyvät API-muutokset ovat listattuna Liferayn Breaking Changes dokumentaatiossa. Pääasiassa muutokset ovat pakettien ja kirjastojen uudelleen nimeämisiä, esimerkiksi portal-kernel ja portal-impl kirjastojen päällekkäisten nimien aiheuttaman ongelman korjaamiseksi on niiden kirjastojen nimiä muutettu. Kuvassa (kuva 20) on muutokset ennen (ylempi) ja jälkeen.

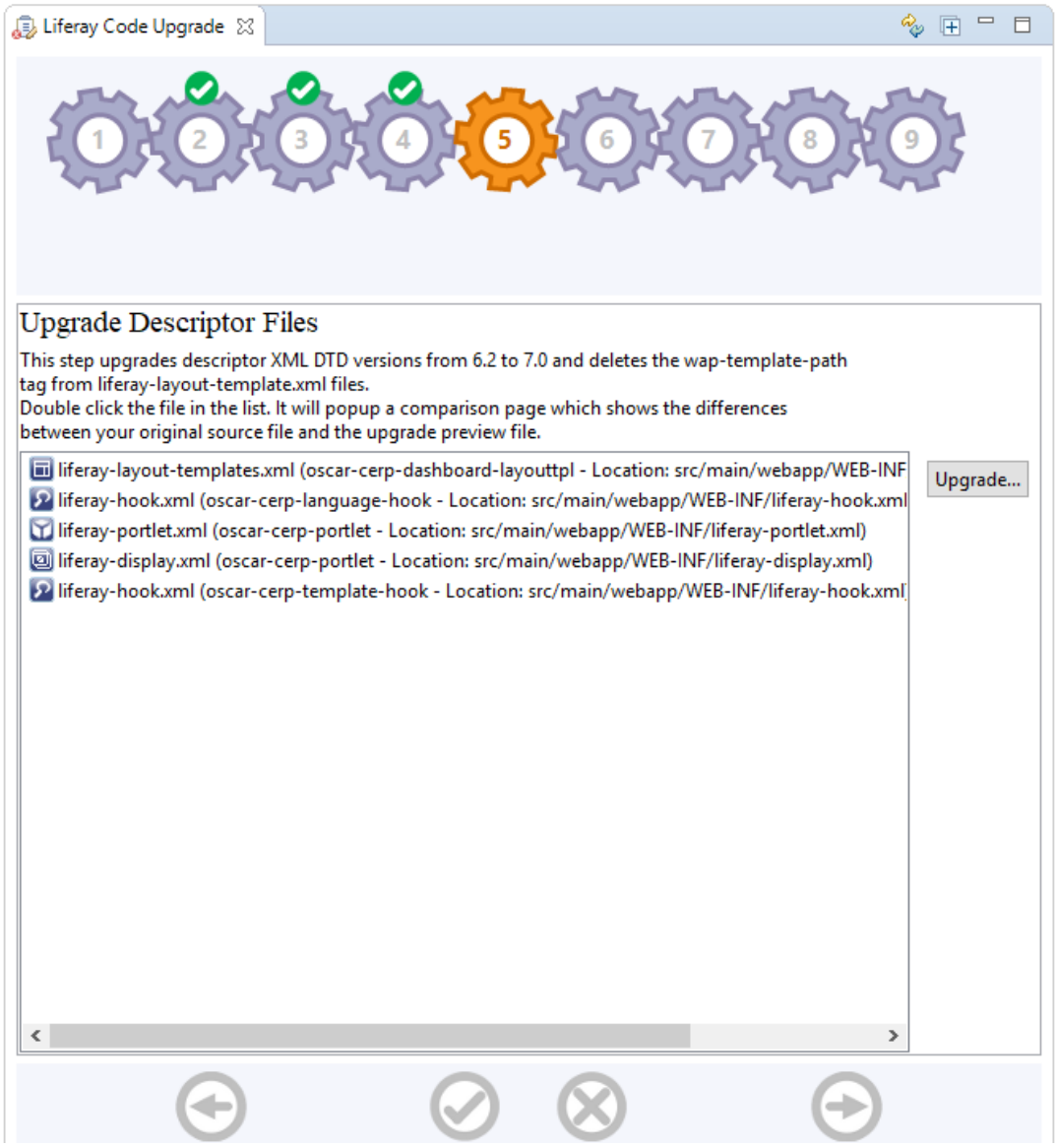
```
 7 import com.liferay.mail.service.MailServiceUtil;
 8 import com.liferay.portal.kernel.mail.MailMessage;
 9 import com.liferay.portal.model.Group;
10 import com.liferay.portlet.expando.model.ExpandoBridge;
```

```
 7 import com.liferay.mail.kernel.service.MailServiceUtil;
 8 import com.liferay.mail.kernel.model.MailMessage;
 9 import com.liferay.portal.kernel.model.Group;
10 import com.liferay.expando.kernel.model.ExpandoBridge;
```

KUVA 20 Kirjastojen nimimuutokset ennen ja jälkeen.

Kirjastojen muutoksien myötä joihinkin portletien metodeihin on tullut muutoksia, joihin keskitytään enemmän seuraavassa portleteja käsittelevässä kappaleessa.

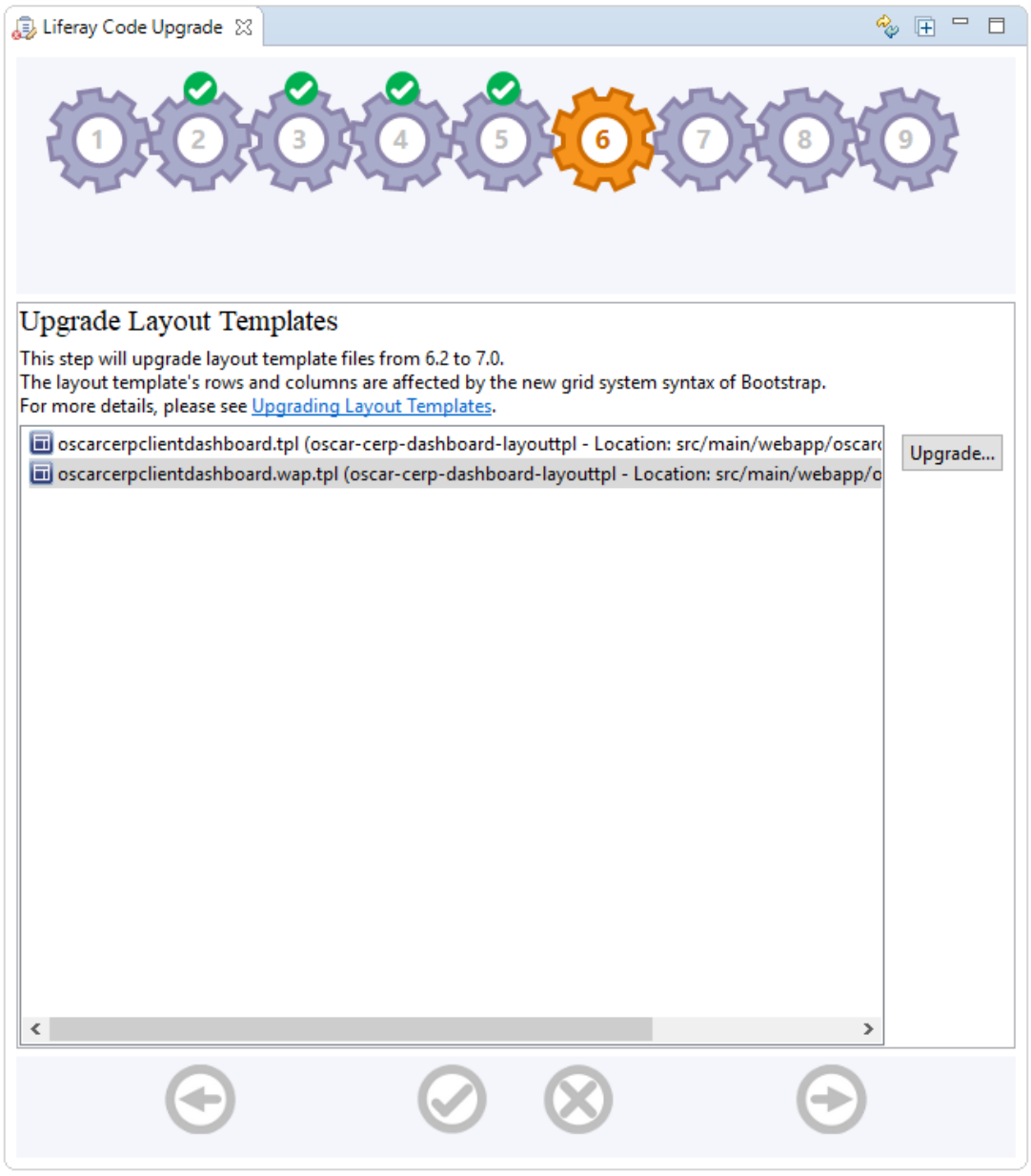
Päivitysprosessin seuraavassa vaiheessa päivitetään liitännäisten määritystiedostot. Tiedostojen muutokset tehdään automaattisesti painaen kuvassa (kuva 21) näkyvää Upgrade painiketta.



KUVA 21 Määritystiedostojen päivittäminen

Työn määritystiedostoihin ei suuria muutoksia tullut. Tiedoston DTD eli dokumentin tyyppimääritelmän versio on vaihdettu Liferay 7: ään, ja merkkikoodaukseksi on määritelty UTF-8.

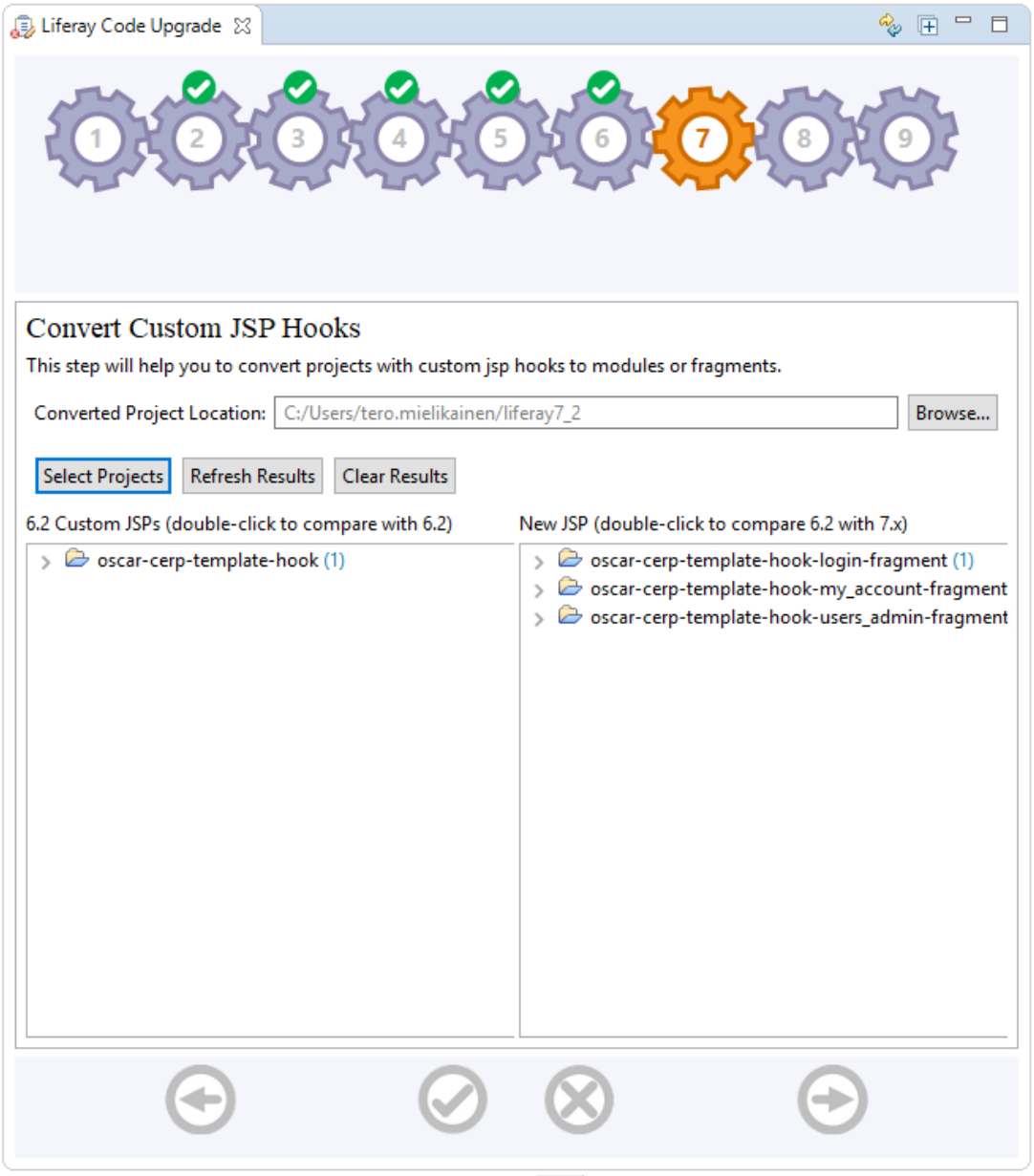
Seuraava työvaihe on Layout Template, eli asettelumallien määritysten päivittäminen. Prosessi on vastaavanlainen, kuin edellisen vaiheen liitännäisten määritystiedostojen päivittäminen. Kuvassa (kuva 22) on työvaiheen näkymä.



KUVA 22 Asettelumallin päivittäminen

Tiedostojen ainoat muutokset koskevat Bootstrapin uuden ruudukkojärjestelmän syntaksia. Uudella ruudukkojärjestelmällä tarkoitetaan elementtien jakamista gridin eli ruudukon sisälle, jolloin voidaan käyttäjän selainikkunan koon perusteella määritellä kuinka paljon ruudukosta elementti vie tilaa.

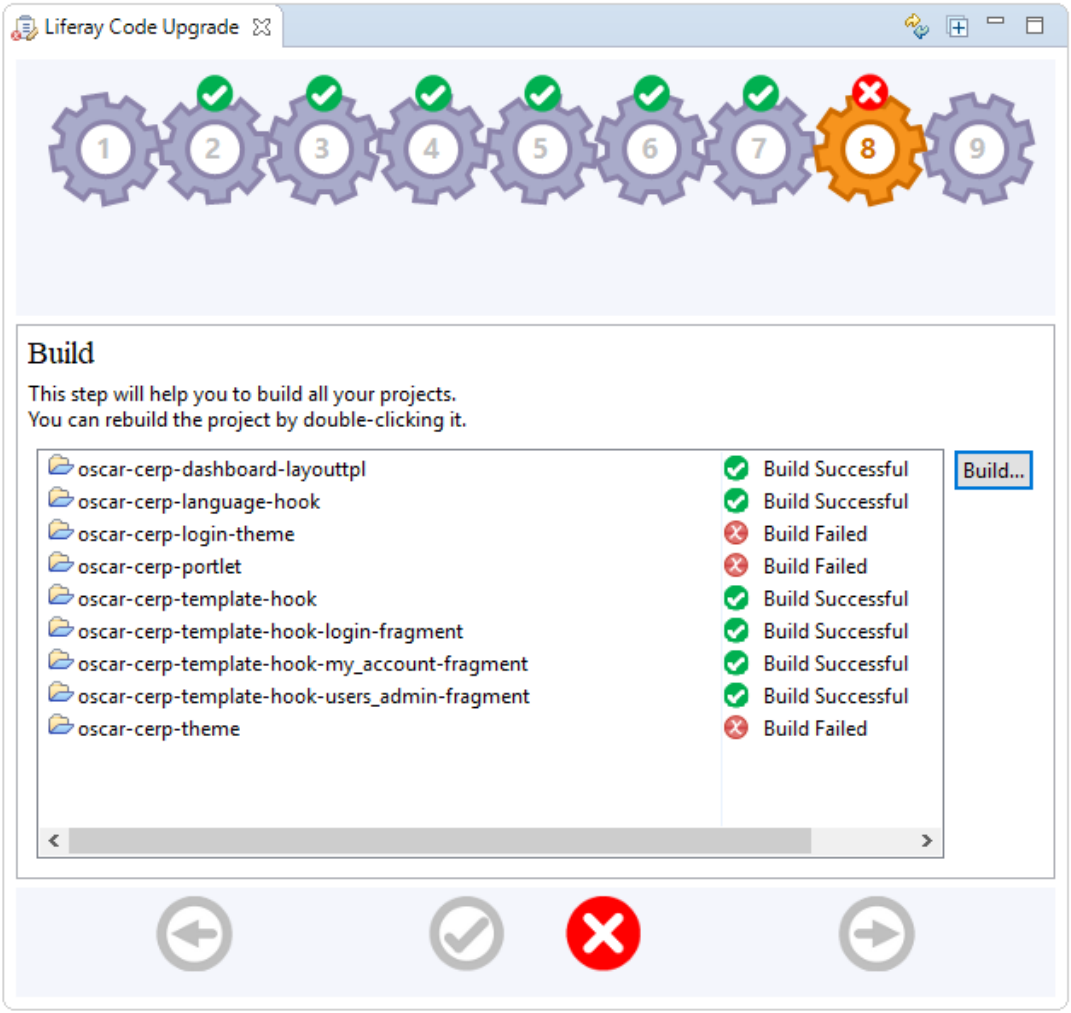
Viimeinen työvaihe ennen projektin kääntämistä on mukautettujen JSP Hookien päivittäminen. Kyseisen vaiheen erikoispiirteenä on, että se vaatii myös vanhan 6.2 palvelimen ajonaikaisen ympäristön, eli runtime environmentin. Palvelin luodaan Eclipseen aiemmin työssä esitetyllä tavalla, joten sitä ei enää tässä osiossa esitellä. Päivittäminen tapahtuu kuvan (kuva 23) mukaisesti valitsemalla hook-projekti valikosta.



KUVA 23 Hook projektin päivitys

Projektin valitsemisen jälkeen työkalu tekee tarvittavat muutokset automaattisesti. Muutoksia voidaan tarkastella tuplaklikkaamalla tiedostoja projektien alta.

Hook-projekteihin tehtyjen muutosten jälkeen projekti voidaan kääntää, mutta kuten kuvasta (kuva 24) huomataan, portlet- ja teema-projektit eivät kääntyneet onnistuneesti.



KUVA 24 Projektien kääntäminen

Syy käännösten epäonnistumiseen selvisi nopeasti ja johtuu siitä, ettei työkalu päivitä portletien tai teemojen POM riippuvuuksia ja liitännäisiä kääntämistä varten. Seuraavissa kappaleissa perehdytään Mavenin, portletien ja teemojen tarvittaviin muutoksiin. Muutosten jälkeen työ käännetään uudelleen, jotta todetaan muutosten onnistuminen.

5.4.5 Maven koontiversion päivitys ja bnd paketointi

Kuten työn aikaisemmassa kappaleessa on todettu, työn ensisijainen tavoite on saada projektit OSGi mallin mukaisesti paketoitua JAR paketeiksi. Tämä tapahtuu pääasiassa POM tiedoston liitännäisiin tapahtuvilla muutoksilla ja riippuvuuksien uusilla versioilla.

Projektien paketoimiseksi uudella versiolla täytyy ensin Mavenin koontiversio (engl. build) päivittää sen mukaiseksi. Liferay 6.2 Maven projektin CSS-tiedostot on käännetty käyttäen liferay-maven-plugin liitännäistä. Uudessa versiossa se on poistettu käytöstä, ja CSS tiedostojen kääntäminen tapahtuu CSS Builderin avulla. Muutos tapahtuu poistamalla Maven Plugin ja korvaamalla se CSS Builder liitännäisellä POM tiedostossa.

Liferay 6.2 portaalissa tarvitsi myös erikseen määritellä palvelimen asetukset POM tiedostoon, mutta uudessa versiossa näitä ei tarvita, koska Liferay 7 portaalissa Maven työkalut eivät ole riippuvaisia asennuksen tietyistä versioista. Nämä kuvassa (kuva 25) näkyvät määritykset voidaan poistaa projektin POM tiedostosta.

```
<liferay.version>6.2.2</liferay.version>
<liferay.maven.plugin.version>6.2.2</liferay.maven.plugin.version>
<liferay.auto.deploy.dir>../../liferay-portal-6.2-ce-ga3/deploy</liferay.auto.deploy.dir>
<liferay.app.server.dir>../../liferay-portal-6.2-ce-ga3/tomcat-7.0.42</liferay.app.server.dir>
<liferay.app.server.deploy.dir>../../liferay-portal-6.2-ce-ga3/tomcat-7.0.42/webapps</liferay.app.server.deploy.dir>
<liferay.app.server.lib.global.dir>../../liferay-portal-6.2-ce-ga3/tomcat-7.0.42/lib/ext</liferay.app.server.lib.global.dir>
<liferay.app.server.portal.dir>../../liferay-portal-6.2-ce-ga3/tomcat-7.0.42/webapps/ROOT</liferay.app.server.portal.dir>
```

KUVA 25 POM tiedostosta poistettavat määritykset

Liferay 6.2 käytettävät artefaktiriippuvuudet ovat myös vaihtuneet eri paketteihin. Yrityksen projektissa käytettiin muun muassa portal-service artefaktia, joka on muuttunut portal-kernel artefaktiin. Muuttumattomiin, projektissa referoituihin artefakteihin täytyy myös päivittää niiden uudet versiot. Alla olevassa taulukossa (taulukko 2) on suosituimpien artefaktipakettien muutokset. (Liferay Upgrading Maven Build, 2017)

TAULUKKO 2 Liferay artefaktien muutoksia (Liferay Upgrading Maven Build, 2017)

Liferay Portal 6.2 Artifact ID	Liferay Portal CE 7.0 Artifact ID
`portal-service`	`com.liferay.portal.kernel`
`util-bridges`	`com.liferay.util.bridges`
`util-java`	`com.liferay.util.java`
`util-slf4j`	`com.liferay.util.slf4j`
`util-taglib`	`com.liferay.util.taglib`

Projektien paketoimiseksi JAR moduuleiksi tarvitaan projektin POM tiedostoon BND ja Maven JAR liitännäiset. BND liitännäinen huolehtii kaikkien Maven resurssien hallinnasta ja valmistelee moduulin muodostamista OSGi paketiksi. Maven JAR liitännäinen huolehtii pakettiarkkitehtuurista ja kääntää projektin OSGi mallin JAR arkistointipaketeiksi nimensä mukaisesti.

BND liitännäinen tarvitsee paketointiin BND direktiivin paketointitiedoston. Tähän tiedostoon määritellään paketin käyttämät kirjastot ja sen nimi sekä versiotiedot. Tiedostot luodaan jokaiseen liitännäisprojektiin, esimerkki BND määritelmästä portlet projektissa on alla olevassa kuvassa (kuva 26). (Liferay Creating A Module JAR Using Maven, 2017)

```
Bundle-Name: OscarCerpPortlet
Bundle-SymbolicName: CerpPortlet
Bundle-Version: 0.0.1-SNAPSHOT
Export-Package: \
    fi.oscar.cerp.adapter,\
    fi.oscar.cerp.entities,\
    fi.oscar.cerp.portlet.admin,\
    fi.oscar.cerp.portlet.board,\
    fi.oscar.cerp.portlet.customer,\
    fi.oscar.cerp.portlet.customercontactperson,\
    fi.oscar.cerp.portlet.dashboardwidget,\
    fi.oscar.cerp.portlet.entity,\
    fi.oscar.cerp.portlet.event,\
    fi.oscar.cerp.portlet.infoHub,\
    fi.oscar.cerp.portlet.invoice,\
    fi.oscar.cerp.portlet.item,\
    fi.oscar.cerp.portlet.list,\
    fi.oscar.cerp.portlet.order,\
    fi.oscar.cerp.portlet.person,\
    fi.oscar.cerp.portlet.report,\
    fi.oscar.cerp.portlet.serviceContracts,\
    fi.oscar.cerp.portlet.settings,\
    fi.oscar.cerp.portlet.slide,\
    fi.oscar.cerp.portlet.slp,\
    fi.oscar.cerp.portlet.spring.contactPerson,\
    fi.oscar.cerp.portlet.tender,\
    fi.oscar.cerp.portlet.travelexpense,\
    fi.oscar.cerp.router,\
    fi.oscar.cerp.utils,\
    fi.oscar.tyse.spring.day,\
    fi.oscar.tyse.spring.inout,\
    fi.oscar.tyse.spring.routines,\
    fi.oscar.tyse.spring.start,\
    fi.oscar.tyse.spring.tasklistbooking,\
    fi.oscar.tyse.spring.worklistbooking,\


Import-Package: *

-jsp: *.jsp,*.jspf
-plugin.jsp: com.liferay.ant.bnd.jsp.JspAnalyzerPlugin
-plugin.resourcebundle: com.liferay.ant.bnd.resource.bundle.ResourceBundleLoaderAnalyzerPlugin
-plugin.sass: com.liferay.ant.bnd.sass.SassAnalyzerPlugin
-sass: *
-sources: true

```

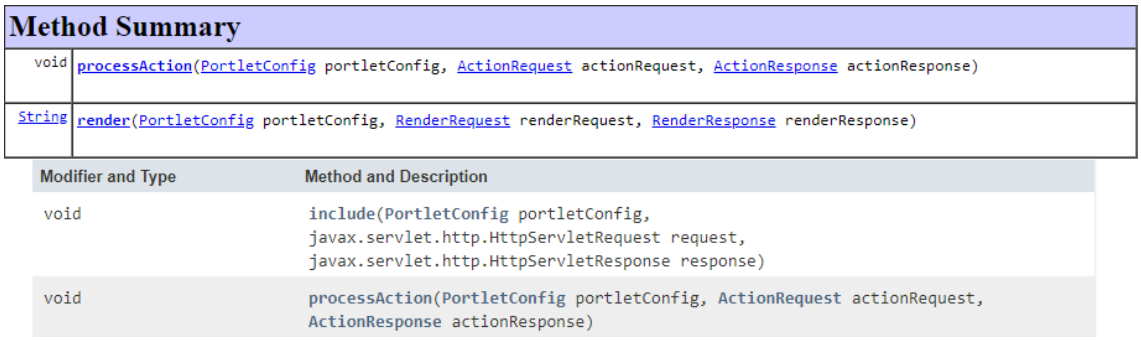
KUVA 26 BND paketointitiedosto portleteille

Kun Mavenin koontiversio on päivitetty ohjeiden mukaisesti ja bnd paketointitiedostot on luotu liitännäisprojekteihin, voidaan aloittaa vastaavasti liitännäisprojektien päivittäminen.

5.4.6 Portletien päivittäminen

Portletien päivittämisessä on otettava huomioon eri teknologioilla toteutetut portletit, työn tapauksessa Liferay MVC ja Spring MVC portletit. Projektin POM tiedostoon päivitetään näihin kuuluvien liitännäisten versiot ja riippuvuudet.

Liferay MVC portletit on hyvin yksinkertaista päivittää, koska suurimmat muutokset korjataan jo Code Upgrade Toolin Breaking Changes työvaiheessa. Kirjastoihin tulleiden muutosten takia kuitenkin jotkin metodit muuttuvat. Työn kannalta merkittävin muutos on ConfigurationAction interfacen toteuttavien luokkien rakennemuutos. ConfigurationAction interface sisälsi aiemmin työssä hyödynnetyn render metodin, joka on version vaihdossa muuttunut. Kuvassa (kuva 27) kyseisen interfacen luokka dokumentaatiot ennen (ylempi) ja jälkeen, josta muutos on helpoiten nähtävissä.



Method Summary

void	processAction(PortletConfig portletConfig, ActionRequest actionRequest, ActionResponse actionResponse)
String	render(PortletConfig portletConfig, RenderRequest renderRequest, RenderResponse renderResponse)

Modifier and Type	Method and Description
void	include(PortletConfig portletConfig, javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response)
void	processAction(PortletConfig portletConfig, ActionRequest actionRequest, ActionResponse actionResponse)

KUVA 27 ConfigurationAction interfacen muutokset (Liferay Dokumentaatio, 2017)

Muutokset render metodiin ovat sen parametreihin, RenderRequest ja RenderResponse ovat vaihtuneet HttpServletRequest ja HttpServletResponse luokkiin. Metodi palautti myös ennen string tyyppisen muuttujan, nyt metodi on void tyypiä, eikä enää palauta mitään. Ajallisista syistä toteutukseen perehtyminen täytyi jättää, joten metodi jäi toteuttamatta uudella tavalla. Korvaavaksi toteutettavaksi luokaksi olisi mahdollista kokeilla DefaultConfigurationAction luokkaa. On myös mahdollista jakaa Render ja Action tapahtumat omiin luokkiinsa.

Aluksi koko portlet-projektia yritettiin kääntää ohjeiden mukaisesti JAR paketeiksi, kuten se nykyisellään tehdään WAR paketeilla. Nopeasti selvisi kuitenkin, että Spring MVC portleteja ei voida kääntää JAR paketeiksi, koska ne perustuvat Java EE-teknologian WAR-moduuleihin.

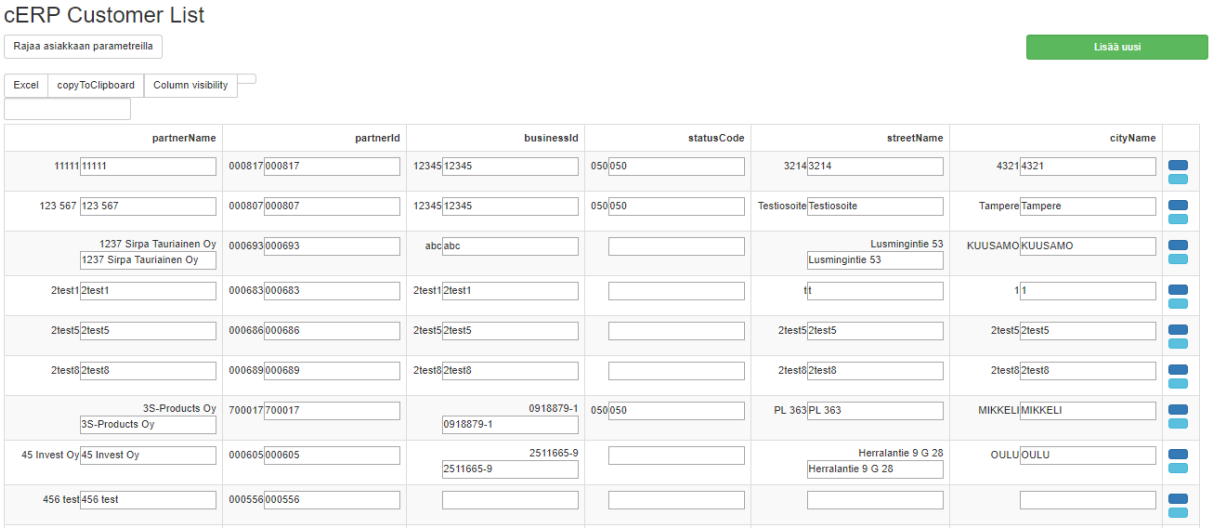
Aiemmin portletin riippuvuudet, kuten sen käyttämät kirjastot on määritelty liferay-plugin-package.properties tiedostoon. Tämä onnistuu Liferay MVC portletien tapauksessa BND paketointitiedostolla ja Maven määrityksillä, mutta Spring portleteilla riippuvuudet täytyy määritellä Maven määritysten lisäksi käyttäen Apache Ivyä, joka on integroitu Apache Ant työkaluun. Riippuvuudet määritellään käyttäen ivy.xml tiedostoa, johon lisätään portletin käyttämät riippuvuudet.

Koska koko projektin päivittäminen epäonnistui Spring portletien takia, päädyttiin projektia purkamaan osiin, ensin yhdellä Liferay MVC portletilla. Erillisen projektin POM-tiedostoon ja BND-paketointitiedostoon määriteltiin samat liitännäiset ja riippuvuudet kuten aiemmin.

JAR-moduuliksi päivittäminen ei kuitenkaan onnistunut, vaikka edettiin täysin Liferayn sivuilta löytyvien ohjeiden mukaisesti. Kääntäessä kaikki Maven käännökset onnistuivat, eikä palvelimelle siirrettäessä konsolissa näkynyt virheitä, joihin se olisi kaatunut. Vaikka kaikki näytti päällisin puolin oikealta, portletia ei näkynyt portaalissa lisättävien portletien listassa.

Konsolissa onnistuneiden portletien siirtämisestä tulee ilmoitus, jossa lukee niiden olevan valmiita käyttöön. Konsolin seuraamisen aikana huomattiin, ettei tuota ilmoitusta tule lainkaan. Syyksi todettiin sen jäävän jollain tavalla jumiin, eikä Liferay Portal ilmoita mihin sovellus on kaatunut.

Seuraavaksi päädyttiin kokeilemaan portletin siirtämistä WAR moduulina aiemman version tapaan. Näin ollen BND paketointitiedostoa ei tarvita ja POM tiedostoon JAR liitännäinen korvattiin WAR liitännäisellä. Muuten prosessi on sama kuin edellisellä yrityksellä. Kaikki käännökset onnistuivat ja palvelimelle siirtäessä konsolissa tuli ilmoitukset normaalisti portletin asennuksesta ja sen käyttövalmiudesta. WAR moduulina portlet siis toimi aivan normaalisti eikä syytä miksei sama portlet toiminut uuden mallin mukaisesti ei selvinnyt ollenkaan. Kuvassa (kuva 28) on portlet sivustolle asetettuna, ilman teema projektin tuomia tyylimuutoksia.



KUVA 28 Muotoilematon päivitetty portlet

Tässä vaiheessa työhön käytetyn ajan takia päätettiin, että työn lopputuloksena on Liferay MVC portletin onnistunut päivitys WAR pakettina. Seuraavaksi portlet vaatii teemasta saatavia CSS-tiedostoja portletin ulkoasun viimeistelyyn.

5.4.7 Teemojen päivittäminen

Teemoihin täytyy tehdä muutamia muokkauksia, ennen kuin vanhat teemat voidaan ajaa uuden version palvelimella. Muutokset aloitetaan päivittämällä liferay-plugin-package.properties tiedostoon oikea Liferay versio sen DTD, eli dokumenttityypin määritelmään.

Aiemmin CSS-tiedostot on nimetty esimerkiksi custom.css, nyt uudessa versiossa vaaditaan tiedostojen nimet muodossa, _custom.scss. Mikäli muutosta ei tee, tulee käännöksenaikainen virhe, jossa uuden mallin CSS-tiedosto luodaan, mutta palvelin ei tiedä kumpaa tiedostoa pitää käyttää.

Muut teemojen muutokset koskevat Bootstrapin uuden version tuomia muutoksia. Aiemmassa Liferay-versiossa käytettiin Bootstrap 2 versiota, joten koodit täytyy päivittää Bootstrap 3 version mukaiseksi. Tämä voidaan tehdä käsin tai käyttää Gulp toimintoa käyttäen gulp.js tiedostoa, johon on määritelty tehtävät päivitystyöt.

Yksittäisen portletin päivittämisessä tärkeintä on päivittää teeman mallitiedostot, kuten portal_normal.ftl tiedosto. Tiedostoissa voidaan muun muassa ladata asiakaspäätteelle erilaisia CSS- ja Javascript-kirjastoja. Aiempaan versioon verrattuna syntaksimuutoksia on tullut huomattavasti. Alla olevassa taulukossa (taulukko 3) on osa portal_normal.ftl tiedostoon tehtävistä muutoksista. (Liferay Upgrading Themes, 2017)

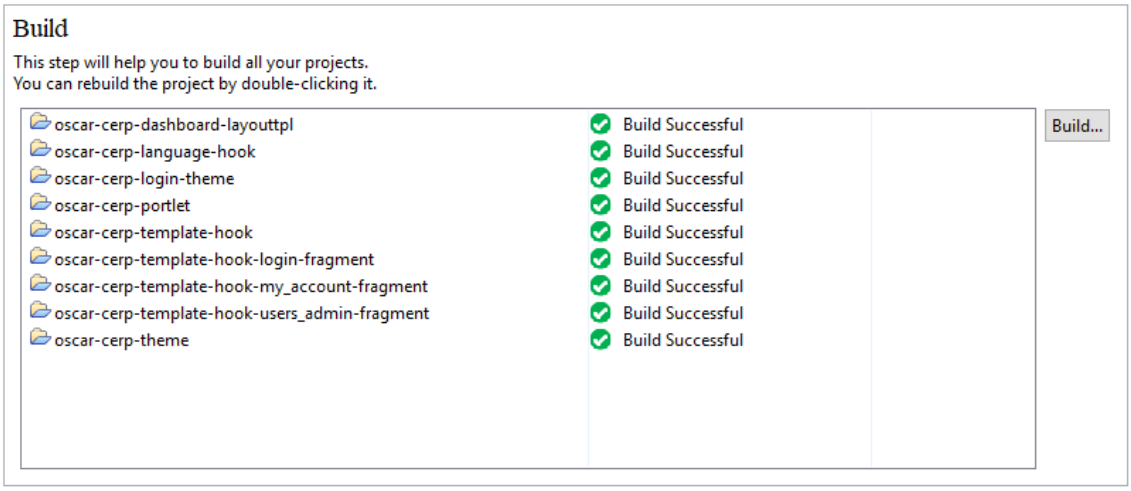
TAULUKKO 3 Portal Normal FTL tiedoston muutoksia (Liferay Upgrading Themes, 2017)

6.2	7.0
`${theme.include(top_head_include)}`	`<@liferay_util["include"] page=top_head_include />`
`${theme.include(body_top_include)}`	`<@liferay_util["include"] page=body_top_include />`
`${theme.include(content_include)}`	`<@liferay_util["include"] page=content_include />`
`${theme.wrapPortlet("portlet.ftl", content_include)}`	`<@liferay_theme["wrap-portlet"] page="portlet.ftl"> <@liferay_util["include"] page=content_include /> </@>`
`${theme.include(body_bottom_include)}`	`<@liferay_util["include"] page=body_bottom_include />`
`${theme.include(bottom_include)}`	`<@liferay_util["include"] page=bottom_include />`
`${theme.getSetting("my-theme-setting")}`	`${theme_settings["my-theme-setting"]}`
`${theme.runtime("56", "articleId=" + my_article_id)}`	`<@liferay_portlet["runtime"] portletName="com_liferay_journal_content_web_portlet_JournalContentPortlet" queryString="articleId=" + my_article_id />`

Template-tiedostojen muutoksilla portletin tarvitsemat CSS ja Javascriptit saadaan portletille. Yrityksen ohjeiden mukaisesti, ei koko teeman päivittämiseen käytetty aikaa, vaan pyrittiin saada toimiva portlet asiallisen näköiseksi.

5.4.8 Projektien ajaminen palvelimella

Kun liitännäisiin tehtiin aiemmin esitellyt muutokset, käännettiin ne uudelleen käyttäen Liferay Code Upgrade Toolin Build-työkalua. Alla olevassa kuvassa (kuva 29) on käännettyjen projektien lopputulos.



KUVA 29 Projektien kääntämisen lopputulos

Kaikkien muutosten jälkeen JAR ei toiminut, mahdollisesti Spring Frameworkin yhteensopimattomuuden takia. Konsoliin ei tullut kunnollista virheilmoitusta miksi näin kävi. Ongelmien vuoksi yritettiin muuttaa yksi portlet JAR muotoon. Portlet saatiin palvelimelle, mutta se ei toiminut ollenkaan sivustolla.

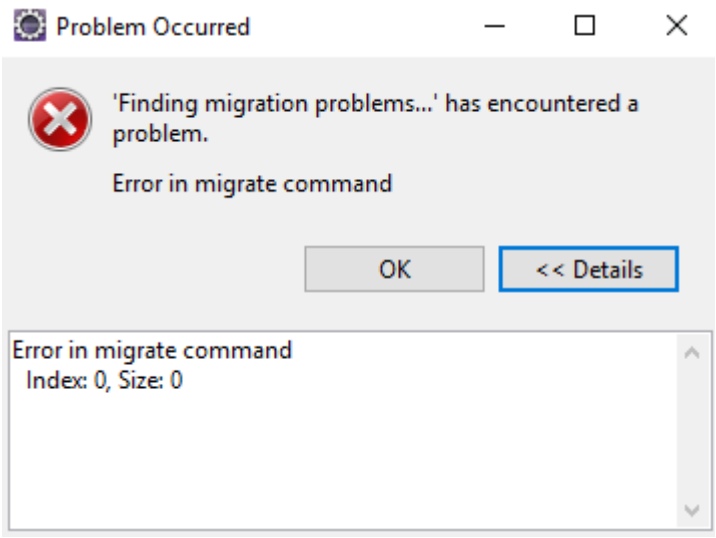
Kun sama portlet ajettiin WAR muodossa, portlet saatiin onnistuneesti näkymään sivustolla. Lopulliseen muotoon se saatiin teeman template tiedostojen muutoksilla.
WAB generatorilla saatiin myös kyseinen portlet toimimaan, mutta todettiin askeleen olevan turha työn kannalta.

Spring portleteja ei saatu toimimaan edes WAR muodossa, joka piti olla Liferay dokumentaation mukaan mahdollista. Syytä ei tarkkaan ehditty selvittää, koska työstä haluttiin

edes yksi toimiva osa. Tällöin lopputulokseksi todettiin Liferay MVC mallilla toteutetun portletin toimivan yrityksen nykyisillä koodeilla.

5.5 Haasteet

Päivitysprosessissa oli monenlaisia haasteita, jotka hidastivat sen edistymistä. Suurimmaksi ongelmaksi osoittautui Liferayn Code Upgrade Tool, jonka toiminta oli enemmän tai vähemmän epävarmaa. Kriittisimmässä, eli Breaking Changes-vaiheessa työkalu ei löytänyt mitään muutettavaa ja kaatui yleensä mystiseen ongelmaan. Kuvassa (kuva 30) on Eclipsen antama virheilmoitus työkalun kaatumisesta.



KUVA 30 Code Upgrade Toolin virheilmoitus

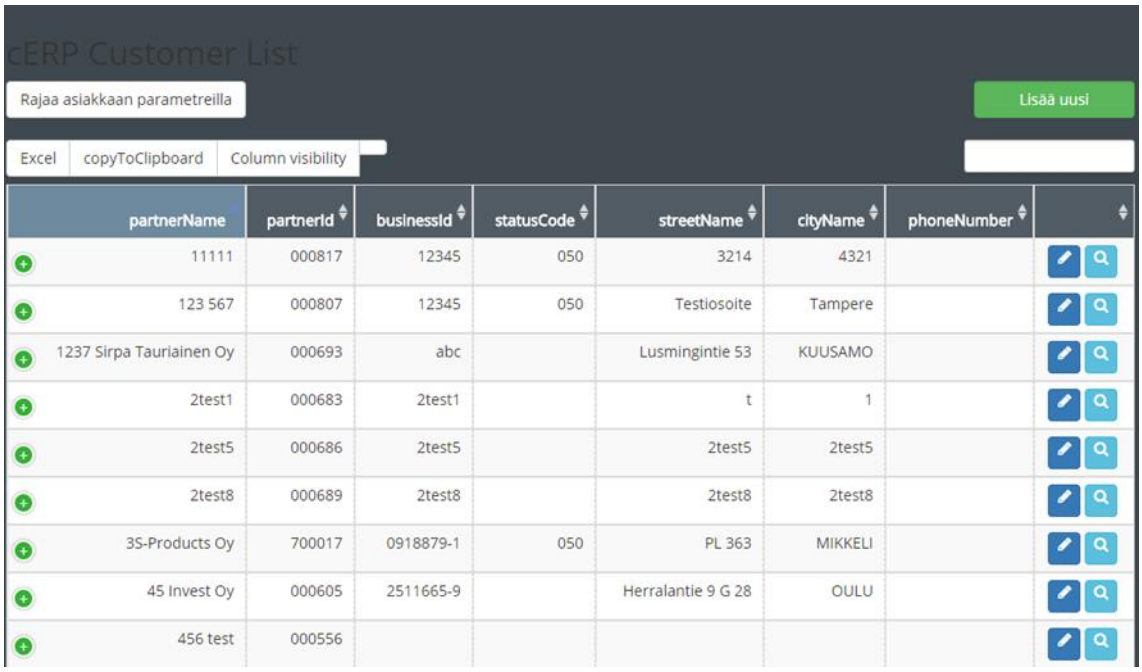
Kuten kuvasta havaitaan, on annetun informaation perusteella vaikea arvioida, miksi työkalu ei saanut työvaihetta loppuun, internetistäkään ongelmaan ei apua löytynyt lainkaan. Toistuvien yritysten jälkeen työkalu yllättäen toimi kerran, jolloin kaikki tarvittavat muutokset saatiin tehtyä.

Spring Framework aiheutti työn aikana valtavan määrän päänvaivaa, yleisesti liittyen sen OSGi yhteensopimattomuuteen. Ongelmien ratkeamattomuuden takia päädyttiin ne pudottamaan päivityslistalta pois.

Portletien päivittäminen OSGi mallin JAR paketeiksi piti olla Liferayn mukaan yksinkertaista. Itse prosessi sitä olikin, mutta jatkuvien ongelmien vuoksi oli luonnollista päivittää ne käyttäen WAR paketointia edellisen version tapaan.

5.6 Lopputulos

Kaikkien aiemmin esiteltyjen muutosten jälkeen lopullinen portlet on kuvan (kuva 31) mukainen.



partnerName	partnerId	businessId	statusCode	streetName	cityName	phoneNumber
11111	000817	12345	050	3214	4321	
123 567	000807	12345	050	Testiosoite	Tampere	
1237 Sirpa Tauriainen Oy	000693	abc		Lusmingintie 53	KUUSAMO	
2test1	000683	2test1		t	1	
2test5	000686	2test5		2test5	2test5	
2test8	000689	2test8		2test8	2test8	
3S-Products Oy	700017	0918879-1	050	PL 363	MIKKELI	
45 Invest Oy	000605	2511665-9		Herralantie 9 G 28	OULU	
456 test	000556					

KUVA 31 Lopullinen päivitetty portlet sivustolla

Koko projektin päivittämisen sijasta päädyttiin, aiemmin esitellyistä syistä, että tarkoituksena oli päivittää ainakin yksittäinen Liferay MVC portlet, joten teemaan ei panostettu kovinkaan paljon. Koska teeman Bootstrap-ominaisuuksia ei päivitetty, sivuston teema on hyvin vajavainen, mutta yksittäinen portlet toimii, kuten sen odotettiin.

6 POHDINTA

Kokonaisuutena työ onnistui lähtökohtiin nähden hyvin, vaikka kaikkiin alussa asetettuihin tavoitteisiin ei päästy. Yrityksellä on ainakin yksi toimiva Liferay MVC portlet, muiden samaa toteutusta käyttävien portletien päivittäminen pitäisi olla vastaavanlainen prosessi. Teemojen päivittämisessä ei suurempia ongelmia tullut, Liferayn päivitysohjeiden avulla prosessi on varsin yksinkertainen, mutta ajankäytöllisistä syistä ne päädyttiin päivittämään vain tärkeimmiltä osa-alueilta. Näin ollen työn teema on vajavainen.

Uuden version palvelin saatiin toimimaan, joten yrityksen toivomat uudet portaalin ominaisuudet ovat hyödynnettävissä. REST-päätepisteitä en ehtinyt työn venymisen takia edes kokeilla, mutta ne pitäisi olla täysin hyödynnettävissä uusissa projekteissa. OSGi malli tukee yrityksen haluamaa projektin jakamista varsin tehokkaasti, jolloin paketteja voidaan uudelleenkäyttää eri projekteissa.

Käytetty teknologia oli minulle täysin tuntematon, joten työn aloittaminen oli erittäin hidasta. Aluksi täytyi tutustua vanhaan versioon riittävän paljon, että on jotakin referenssiä päivitystä aloittaessa. Vanhan version kanssa apuna olivat työtoverit, jotka tarvittaessa selittivät teknologian tärkeimpiä osa-alueita. Päivitystä aloittaessa täytyi tutustua myös uuden version tuomiin muutoksiin, sekä mitä on hyvä tietää version vaihdon aikana. Uudempi versio tuotti hankaluuksia siinäkin suhteessa, että se oli työtovereille uutta, jolloin ongelmien selvittämisessä kesti välillä hyvinkin pitkään.

Liferayn toimintaa oli aluksi hankala ymmärtää, lisäksi sen dokumentaatiot olivat aloittavalle Liferay kehittäjälle paikoitellen heikkoa. Tämä aiheutti turhaa ajanhaaskausta jonkin pienenkin ongelman selvittämiseen. Yhteisöfoorumit olivat yleensä melko suppeita, mutta työn aikana sieltä löytyi kyllä paljon apua. Liferay IDE:llä kehittäminen oli välillä todella ikävää, käännösten aikana virheistä tuli suppeita ilmoituksia konsoliin, jos niitä tuli ollenkaan.

Työ oli kokonaisuudessaan haastava, mutta kun alkukankeuden ja ongelmien ohi pääsi, se oli kiinnostava. Työelämään työstä on hyötyä varsinkin jatkossa järjestelmien päivittämiseen, sekä kerrytti paljon kokemusta Javalla ja Liferayllä kehittämiseen. Pettymyksenä työssä oli se, ettei koko projektia saatu loppuun asti päivitettyä, eikä varsinaisia

OSGi moduuleja päästy täysin hyödyntämään. Syitä taustalla voi olla monia, kuten tarvittava oma tietotaito moduulien tiedostojen konfiguroinnissa.

Työstä jäi vielä avoimia vaihtoehtoja erilaisiin päivitysmahdollisuuksiin. Spring Frameworkin portleteissa mahdollista on eristää ne omaksi projektiksi ja sen jälkeen päivittää WAR paketteina, aivan kuten ennenkin. On mahdollista hyödyntää WAB Generatoria muuttamaan ne OSGi mallin kanssa yhteensopiviksi WAB paketeiksi, jos sille nähdään tarvetta. Myös mahdollisuus niiden päivittämiseen voi olla niiden toteuttaminen jollakin toisella teknologialla, jotta ne voidaan paketoida JAR paketteina.
JAR paketoinnissa tarvitsee perehtyä BND paketointidirektiiviin, sekä mahdollisesti hyödyntää Gradlea onnistuneen päivityksen aikaansaamiseksi.

Jatkossa portletien ajaminen on täysin mahdollista ajaa palvelimella WAR paketteina kuten ennenkin, päivittämällä vain koodin ja riippuvuudet uuden version mukaiseksi.

LÄHTEET

7-ZIP työkalun kotisivut, luettu 13.04.2018
https://www.7-zip.org/

Git versiohallinnan kotisivut, luettu 13.04.2018
https://git-scm.com/

Eclipse IDE, luettu 16.04.2018
https://www.eclipse.org/ide/

Apache Maven esittely, luettu 13.04.2018
https://maven.apache.org/what-is-maven.html

Liferay Adapting to Liferay 7 API
https://dev.liferay.com/develop/tutorials/-/knowledge_base/7-0/adapting-to-liferay-7s-api-with-the-code-upgrade-tool

Liferay Using WAB Generator, luettu 16.04.2018
https://dev.liferay.com/develop/tutorials/-/knowledge_base/7-0/using-the-wab-generator

Liferay Best Open Source Portal of 2007, luettu 26.03.2018
https://www.infoworld.com/article/2659795/computer-hardware/2007-technology-of-the-year-award-winners.html

Liferay Developer Documentation, luettu 08.04.2018
https://dev.liferay.com/develop/tutorials

Liferay Fundamentals, luettu 08.04.2018
https://dev.liferay.com/develop/tutorials/-/knowledge_base/7-0/fundamentals

Liferayn kotisivut, luettu 08.04.2018
https://www.liferay.com/

Gartnerin Magic Quadrant for Horizontal Portals vuodelta 2016, luettu 26.03.2018

https://www.smc.it/documents/945540/975068/REPORT+GARTNER+2016+-+Horizontal+Portals+-+Liferay.pdf/a733154c-cc42-47d2-a319-8d4d19a5da77

Kehittäjien mielipiteitä Liferaysta, 2014, luettu 28.03.2018
https://web.liferay.com/community/forums/-/message_boards/message/43061427

Liferayn portletit, luettu 28.03.2018
https://dev.liferay.com/develop/tutorials/-/knowledge_base/7-0/portlets

IBM OSGi bundles, luettu 08.04.2018
https://www.ibm.com/support/knowledgecenter/en/SSHR6W/com.ibm.websphere.wdt.doc/topics/cbundles.htm

OSGi Architecture, luettu 08.04.2018
https://www.osgi.org/developer/architecture/

Maven arkkityypit, luettu 11.04.2018
https://maven.apache.org/guides/introduction/introduction-to-archetypes.html

Roufid, Understanding Liveray portlet configuration files, 2016, luettu 22.04.2018
http://roufid.com/understanding-liferay-portlet-files/

Liferay Creating the Sitemap.json file, 2017, luettu 23.04.2018
https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/creating-the-sitemap-json-file

Liferay Theme Reference Guide, 2017, luettu 23.04.2018
https://dev.liferay.com/develop/reference/-/knowledge_base/7-0/theme-reference-guide

Liferay Installing Liferay Portal, 2017, luettu 23.04.2018
https://dev.liferay.com/discover/deployment/-/knowledge_base/7-0/installing-product

Liferay Upgrading Themes, 2017, luettu 23.04.2018
https://dev.liferay.com/develop/tutorials/-/knowledge_base/7-0/upgrading-themes