Opinnäytetyö (AMK)

Tietotekniikka

Sulautetut ohjelmistot

2015

Anssi Kivinen

ECLIPSE BIRT -RAPORTOINTIALUSTAN INTEGROINTI JA KÄYTTÖ VAADIN-POHJAISESSA JAVA-SOVELLUKSESSA

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut ohjelmistot

2015 | 35

Tiina Ferm

Anssi Kivinen

ECLIPSE BIRT -RAPORTOINTIALUSTAN INTEGROINTI JA KÄYTTÖ VAADIN-POHJAISESSA JAVA-SOVELLUKSESSA

Opinnäytetyössä selvitettiin Eclipse BIRT -raportointialustan ominaisuuksia ja integrointia Java-ohjelmointikielellä ja Vaadin-sovelluskehyksellä toteutettuun WWW-sovellukseen. WWW-sovelluksessa käytettiin tietokantana PostgreSQL-relaatiotietokantaa, jota tarkasteltiin myös tässä opinnäytetyössä. Opinnäytetyö tehtiin toimeksiantona Trivore Oy:lle.

Opinnäytetyössä tehtiin Java-ohjelmointikielellä ja Vaadin-sovelluskehyksellä yksinkertainen WWW-sovellus, jonka avulla voitiin selvittää käytännössä Eclipse BIRT -raportointialustan integrointiin liittyviä yksityiskohtia. Sovellukseen tehtiin näkymät käyttäjien ja tietojen muokkaukseen sekä näkymät raporttipohjien muokkaamiseen ja raporttien lataamiseen. Sovellukseen toteutettiin lisäksi yksinkertainen käyttäjien oikeuksienhallinta sekä raporttien latausten lokikirjaus tietokantaan. Sovelluksen tietokantayhteyksiin käytettiin Vaadinin SQLContainer-luokkaa, joka puolestaan käyttää JDBC-rajapintaa.

Opinnäytetyön tuloksena Eclipse BIRT –alustasta ja sen rajapinnasta saatiin hyödyllistä tietoa, jota voidaan käyttää myöhemmin muissa projekteissa. Opinnäytetyössä toteutetusta WWW-sovelluksesta saatiin ohjelmakoodia, jota voidaan myös hyödyntää tulevissa projekteissa.

ASIASANAT:

Java, Vaadin, Eclipse BIRT, PostgreSQL

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information Technology | Embedded Software

2015 | 35

Tiina Ferm

Anssi Kivinen

INTEGRATION OF ECLIPSE BIRT TO A VAADIN-BASED JAVA WEB APPLICATION

The purpose of the thesis was to examine the Eclipse BIRT reporting platform, it's properties and the integration of Eclipse BIRT to a Vaadin-based Java web application. The thesis also includes some examination of PostgreSQL object relational database management software, which was also used in the project. The thesis was commissioned by Trivore Oy.

In the thesis, a simple Vaadin-based Java web application was created and it was used to clarify some of the practical details of the integration of Eclipse BIRT. The application has views for editing users and the data in the database as well as views for editing report designs and downloading reports. In addition, simple user right management and logging of report downloads were implemented to the application.

As a result of the thesis, useful information was gained of Eclipse BIRT, it's architecture and it's API. The information as well as the source code of the thesis project can be used in future projects.

KEYWORDS:

Java, Vaadin, Eclipse BIRT, PostgreSQL

SISÄLTÖ

1 JOHDANTO **6**

2 TOIMEKSIANNON KUVAUS **7**

~~3~~ RAPORTOINTI **9**

4 ECLIPSE BIRT -RAPORTOINTIALUSTA **10**

4.1 Arkkitehtuuri 10

4.1.1 Raporttisuunnittelija 11

4.1.2 Raportointimoottori 11

4.1.3 Charting Engine 12

4.2 Raportointimoottorin integrointi Java-sovellukseen 12

5 POSTGRESQL-RELAATIOTIETOKANTA **13**

6 VAADIN-SOVELLUSKEHYS **15**

7 JAVA DATABASE CONNECTIVITY (JDBC) **17**

8 SOVELLUSKEHITYSYMPÄRISTÖ JA –TYÖKALUT **18**

9 POSTGRESQL-TIETOKANNAN ASENTAMINEN JA TIETOKANNAN KUVAUS **19**

10 RAPORTTIMALLIN LUONTI RAPORTTISUUNNITTELIJALLA **23**

11 SOVELLUKSEN TOTEUTUS **25**

11.1 Käyttöliittymä 25

11.2 Eclipse BIRT-alustan integrointi 30

11.3 Raporttien latausten lokikirjaus 32

11.4 Käyttäjien oikeuksienvalvonta 33

12 YHTEENVETO **34**

LÄHTEET **35**

LIITTEET

ReportEngineHandler-luokan toteutus 37

KUVAT

Kuva 1. Kuvakaappaus kirjautumisnäkymästä. 26
Kuva 2. Kuvakaappaus käyttäjienhallintanäkymästä. 26
Kuva 3. Kuvakaappaus tietojenmuokkausnäkymästä. 27
Kuva 4. Kuvakaappaus raporttienhallintanäkymän raporttienlatausvälilehdestä. 28
Kuva 5. Kuvakaappaus raporttienhallintanäkymän raporttipohjienhallintavälilehdestä. 29
Kuva 6. Kuvakaappaus latauslokien tarkastelunäkymästä. 30

KUVIOT

Kuvio 1. Kaavio Eclipse BIRT-alustan arkkitehtuurista [3]. 10
Kuvio 2. Kaavio Vaadinin palvelinpuolen sovelluksen arkkitehtuurista [11]. 16

SQL-LAUSEET

SQL-lause 1. Käyttäjän luontiin käytetty SQL-lause. 19
SQL-lause 2. Tietokannan luontiin käytetty SQL-lause. 19
SQL-lause 3. Käyttäjätaulun luontiin käytetty SQL-lause. 20
SQL-lause 4. "Values1"-taulun luontiin käytetty SQL-lause. 21
SQL-lause 5. "Audit_log" taulun luontiin käytetty SQL-lause. 22
SQL-lause 6. SQL-kysely, jolla haettiin taulun ”values1” tiedot raporttiin. 23

OHJELMAKOODIT

Ohjelmakoodi 1. BirtEngineFactory-luokan toteutus. 31
Ohjelmakoodi 2. Yksinkertaisen lokikirjauksen toteuttava ohjelmakoodi. 32
Ohjelmakoodi 3. Esimerkki käyttäjien oikeuksienhallinnasta. 33

1 JOHDANTO

Eclipse BIRT -raportointialustan käyttö sovelluksessa mahdollistaa erilaisten raporttipohjien helpon luonnin ja muokkaamisen, mikä nopeuttaa sovelluskehitystä. Raporttipohjien luonti voidaan erottaa sovelluskehitysprosessissa omaksi kokonaisuudekseen, ja raporttipohjia voidaan muokata sovelluksen loppukäyttäjien haluamalla tavalla. Raporttipohjien luonti tai muokkaaminen ei vaadi muutoksia sovelluksen lähdekoodiin eikä sovelluksen uudelleenkääntämistä, mikä mahdollistaa sen, että sovelluksen loppukäyttäjät voivat myös itse muokata tai luoda uusia raporttipohjia.

Tässä opinnäytetyössä selvitetään Eclipse BIRT -raportointialustan käyttöönottoa ja integrointia Vaadin- ja Spring-ohjelmistokehyksillä toteutettuun Java-sovellukseen. Tietokantana sovelluksessa käytetään PostgreSQL-relaatiotietokantaa, jota tarkastellaan myös tässä opinnäytetyössä. Tämä opinnäytetyö tehtiin toimeksiantona Trivore Oy:lle.

Opinnäytetyössä toteutetaan Java-ohjelmointikielellä yksinkertainen pilvipalveluna toimiva WWW-sovellus, johon integroidaan työssä tarkasteltava Eclipse BIRT-raportointialusta. Huomioitavaa on, että työssä toteutettava WWW-sovellus ei tule käyttöön sellaisenaan, vaan sen avulla on tarkoitus havainnoida käytännössä, kuinka Eclipse BIRT –alusta integroituu WWW-sovellukseen. Tästä syystä osia sovelluksen toteutuksesta on yksinkertaistettu ja jätetty pienemmälle huomiolle. Tässä työssä ei myöskään tarkastella erityisesti tietoturvaan tai käyttöliittymän ulkoasuun liittyviä asioita.

2 TOIMEKSIANNON KUVAUS

Tämän työn toimeksiantoon kuuluu Eclipse BIRT -raportointialustan tarkastelu, raporttipohjien luominen ja muokkaaminen BIRT Report Creator –työkalulla. Toimeksiantoon liittyy lisäksi BIRT-alustan Report Engine -komponentin integroinnin soveltamista yksinkertaiseen Java-ohjelmointikielellä ja Vaadin-sovelluskehyksellä toteutettuun WWW-sovellukseen sekä PostgreSQL-tietokantaan.

Tietokannan rakenne on määritelty siten, että tauluja on yhteensä viisi. Taulut on määritelty toimeksiannossa seuraavan listan mukaisesti:

- Taulu 1 on käyttäjätaulu, joka sisältää ainakin käyttäjätunnuksen, salasanan, sähköpostiosoitteen sekä puhelinnumeron.
- Taulu 2 sisältää kolme kokonaislukusaraketta sekä kolme tekstisaraketta.
- Taulun 3 rakenne on taulun nimeä lukuun ottamatta identtinen taulun 2 kanssa.
- Taulun 4 rakenne on taulun nimeä lukuun ottamatta identtinen taulujen 2 ja 3 kanssa.
- Taulu 5 on auditointiloki, johon tallennetaan tietoja käyttäjien lataamista raporteista. Auditointilokitaulussa on tapahtuman ajankohta sekä tapahtuman kuvaus.

Eclipse BIRT -raportointialustalla on tarkoitus generoida raportteja taulujen 2, 3 ja 4 sisältämien tietojen pohjalta.

Käyttäjätauluun tehdään kolme käyttäjää: admin, user1 ja user2. Käyttäjien oikeudet tehdään niin että admin-käyttäjällä on oikeudet poistaa ja luoda uusia raporttityyppejä sekä muokata jo olemassa olevia raporttityyppejä. User1-käyttäjällä on oikeus tulostaa raportteja, mutta ei lisätä, poistaa tai muokata raporttipohjia. User2-käyttäjällä ei ole oikeutta muokata tai edes tulostaa raportteja.

Työssä tehdään vähintään kaksi erilaista raporttipohjaa ja toteutetaan Vaadin-ohjelmistokehyksellä WWW-sovellus, jossa raporttipohjia voidaan tulostaa, muokata, lisätä ja poistaa.

~~3~~ RAPORTOINTI

Tässä opinnäytetyössä raportoinnilla tarkoitetaan raportin ohjelmallista tekoa relaatiotietokannassa olevien tietojen pohjalta ja raportilla taas raportoinnin tuloksena syntynyttä dokumenttia.

Ohjelmallisen raportoinnin haasteita ohjelmistonkehityksessä ovat esimerkiksi lokalisointi ja raporttien mallipohjien muokattavuuden tekeminen mahdollisimman helpoksi.

Lokalisoinnilla tarkoitetaan raportin muodon ja kielen sovittamista käyttäjän kielelle. Lokalisointi ei rajoitu pelkästään kieleen vaan kattaa myös muun muassa päivämäärät, valuutat sekä muut yksiköt. [1]

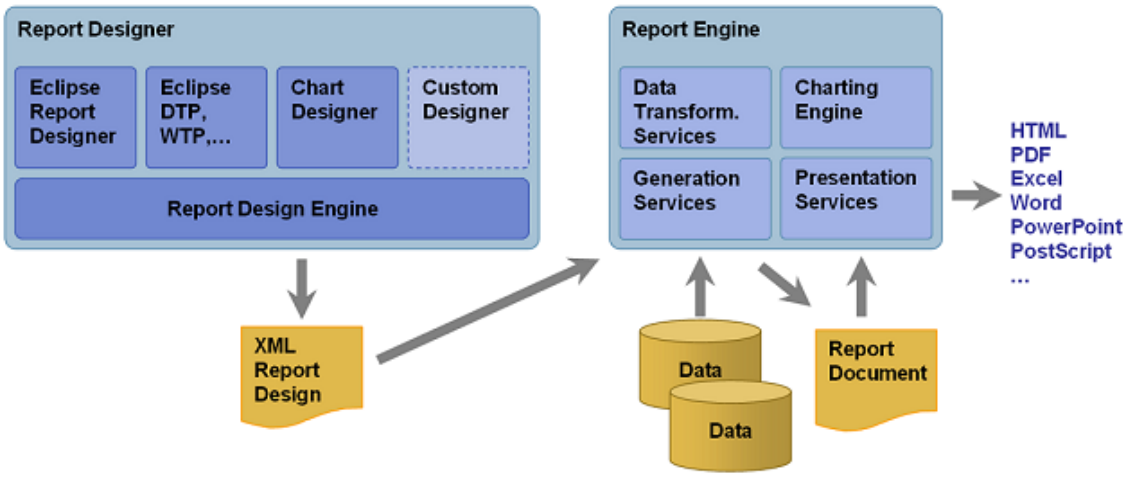
Raporttien mallipohjien muokkaaminen mahdollistaa esimerkiksi fontin, mahdollisen yrityksen logon tai muun yksityiskohdan muokkaamisen, mutta ei rajoitu pelkästään ulkoasun muokkaamiseen vaan sovellukseen voidaan tehdä myös kokonaan uusia raporttipohjia. Raporttipohjien muokkaaminen ja uusien raporttipohjien luominen olisi hyvä voida tehdä ilman ohjelman lähdekoodin muokkaamista ja sovelluksen uudelleen kääntämistä. Tämä voisi mahdollistaa esimerkiksi sen, että sovelluksen loppukäyttäjä voisi itse helposti vaikuttaa raporttipohjien ja siten myös generoitujen raporttien ulkoasuun ja rakenteeseen.

4 ECLIPSE BIRT -RAPORTOINTIALUSTA

Eclipse BIRT on avoimeen lähdekoodiin perustuva, Eclipse Public Licensen alainen, voittoa tavoittelemattoman Eclipse-yhtiön kehittämä alusta, jolla voidaan luoda erilaisia raportteja. Raportit voidaan liittää osaksi WWW-sovellusta tai paikallisesti ajettavaa sovellusta (ns. rich client). BIRT on tarkoitettu erityisesti Javalla tai Java EE:llä toteutettujen sovellusten raportointialustaksi. Eclipse-yhtiön mukaan BIRT on nykyään yksi käytetyimmistä tiedon visualisoinnin ja raportoinnin alustoista, jota käyttää yli 2,5 miljoonaa sovelluskehittäjää 157 eri maassa. [2]

4.1 Arkkitehtuuri

BIRT jakautuu karkeasti kahteen eri komponenttiin: Eclipse-ohjelmointiympäristön päällä toimivaan raporttisuunnittelijaan (Report Designer) ja raporttimoottoriin (Report Engine). Raporttimoottori generoi raportit raporttisuunnittelijan luoman mallin mukaisesti. Report Engine -komponentti voidaan integroida osaksi Java-sovellusta. Kuviossa 1 on kaavio BIRT-alustan arkkitehtuurista. [3]



Kuvio 1. Kaavio Eclipse BIRT-alustan arkkitehtuurista [3].

4.1.1 Raporttisuunnittelija

Raporttisuunnittelija (Report Designer) on Eclipse-kehitysympäristön päälle tehty Eclipsen liitännäisistä koostuva näkymä, jolla voidaan luoda erilaisia raporttimalleja. Raporttimallit ovat XML-formaatin mukaisia, tavallisesti reportdesign-päätteisiä tiedostoja, joiden perusteella varsinainen raportti generoidaan.

Raporttisuunnittelijassa on erilaisia työkaluja, joita ovat muun muassa Data Explorer, Raportin asettelunmuokkausnäkymä (Layout View) sekä Property- ja skriptieditorit. Data Explorer –työkalulla voidaan hallita tietolähteitä (Data Source) ja tietojoukkoja (Data Set). Tietolähteet ovat yhteyksiä tietovarastoihin ja tietojoukot taas ovat kyselyitä tietolähteiden tietoihin. Raportin asettelunmuokkausnäkymä on graafinen editori, jolla raportin ulkoasua voidaan muokata ”drag & drop” –menettelyllä. Property-editorilla voidaan katsoa ja muokata valitun elementin ominaisuuksia. Skriptieditorilla voidaan kirjoittaa skriptejä, jotka voidaan ajaa tietoja haettaessa tai raportin generoinnin yhteydessä. [4]

Raporttimallien generoimiseen käytetään BIRT-alustan Design Engine – komponenttia. Design Engineä voidaan käyttää BIRTin tarjoaman Design Engine API:n kautta ja tätä samaa rajapintaa käyttää myös Eclipse-ohjelmointiympäristön raporttisuunnittelija. [3]

4.1.2 Raportointimoottori

Raportointimoottori (Report Engine) on BIRT-alustan osa, joka on vastuussa raporttien generoinnista. Raportointimoottori tarvitsee raporttimallin, jonka perusteella raportti luodaan. Raportointimoottoria voidaan käyttää sen oman rajapinnan, Report Engine API:n (RE API) kautta, minkä ansioista se voidaan integroida osaksi Java-sovellusta. [3]

RE API:lla voidaan raportin generoinnin lisäksi muun muassa hakea raporttiin tarvittavat parametrit ja niiden oletusarvot, hakea kuva tai kaavio raporttiin sekä viedä raportin tiedot csv-formaattiin. [5]

4.1.3 Charting Engine

Charting Engine –komponentilla voidaan luoda raporttiin tai muualle sovellukseen erilaisia kaavioita annetuista tiedoista. Tällaisia kaavioita ovat esimerkiksi pylväs- ja piirakkadiagrammit. Sekä raportointimoottori, että Design Engine käyttävät Charting Engineä kaavioiden luomiseen. Charting Engineä voidaan käyttää Charting Engine API –rajapinnan avulla. [3]

4.2 Raportointimoottorin integrointi Java-sovellukseen

Raportointimoottorin voi integroida sovellukseen muutamalla eri tavalla. Yksinkertaisin tapa on ladata Eclipsen tarjoama war-paketti ja ottaa se käyttöön tomcat-palvelimella. Paketti sisältää sovelluksen, joka tarjoaa käyttöliittymän generoidun raportin katseluun ja lataamisen usealla eri formaatilla, joita ovat muun muassa PDF, HTML sekä LibreOffice- ja Microsoft Office –dokumentit. Raporttimallit siirretään palvelimelle tiettyyn hakemistoon ja generoitavan raporttimallin tiedoston nimi ja raporttikohtaiset muuttujat annetaan parametreina URL-osoitteessa, jollainen voi olla esimerkiksi *http://localhost:8080/birt/frameset?__report=test1.rptdesign&sample=myparam*.

Valmiin war-paketin käyttöönotto on yksinkertaista ja sen voi ottaa käyttöön jo olemassa olevaan sovellukseen, mutta sen käyttöliittymä jää erilliseksi kokonaisuudekseen, eikä se valvo tarvittavia käyttäjien oikeuksia. Näiden puutteiden takia tässä työssä valittiin integroinnin lähestymistavaksi BIRT-alustan rajapintojen käyttö suoraan Vaadinilla toteutetusta sovelluksesta.

5 POSTGRESQL-RELAATIOTIETOKANTA

PostgreSQL on avoimen lähdekoodin, PostgreSQL-lisenssin alainen olio-relaatiotietokantapalvelin (object-relational database management system (ORDBMS)). PostgreSQL:ää kehittää nykyään Californian yliopiston Berkeley Computer Science Department. [6]

PostgreSQL:n toteutus alkoi alun perin vuonna jo 1986, jolloin se tunnettiin POSTGRES-projektina. SQL-tuen Postgres sai vuonna 1994, jolloin sen nimi muutettiin Postgres95:ksi. Vuonna 1996 Postgres95 sai nykyisen nimensä, PostgreSQL. [7]

PostgreSQL tukee monia eri tietotyyppejä, joita ovat muun muassa erilaiset kokonaislukuihin perustuvat tietotyypit, kuten smallint, integer sekä bigint, jotka kuvaavat 2-, 4- sekä 8-tavuisia etumerkillisiä kokonaislukuja. PostgreSQL:ssä on erikseen myös smallserial-, serial-, ja bigserial-tietotyypit, jotka ovat automaattisesti kasvavia 2-, 4- sekä 8-tavuisia kokonaislukuja. [8] Serial ei ole varsinainen tietotyyppi, vaan merkintätapa, jolla on yksinkertaistettu ainutlaatuisten pääavainten (engl. primary key) luonti.

Tekstipohjaisia tietotyyppejä PostgreSQL:ssä ovat character varying(n), varchar(n), character(n) ja char(n). Kirjain n on positiivinen kokonaisluku, joka kuvaa merkkijonon pituutta. Character(n) ja char(n) tarkoittavat keskenään samaa asiaa ja niihin tallennettu merkkijono täytetään automaattisesti välilyönneillä n:n pituiseksi. Character varying(n) ja varchar(n) tarkoittavat myös keskenään samaa tietotyyppiä ja niiden ero character(n) –tyyppiin on se, että merkkijonon pituus voi vaihdella ja sitä ei täytetä kuten character(n)-tyypissä. Text-tietotyyppi on rajoittamattoman pituinen merkkijono. [9]

Muita PostgreSQL:ssä olevia tietotyyppejä ovat esimerkiksi kiinteän ja vaihtelevan mittaiset bittijonot, bit(n) ja bit varying(n) sekä erilaisia ajan tai desimaalilukujen tallentamiseen käytettäviä tietotyyppejä. PostgreSQL:ssä on tietotyypit myös rahasummille ja IP-, IPv6- ja MAC-osoitteille. [8]

PostgreSQL on laaja kokonaisuus, joten sen syvällinen tarkastelu ei sisälly tähän opinnäytetyöhön. Tässä työssä käytettävässä tietokannassa käytetään integer-, text- ja character varying(n) – tietotyyppejä sekä serial-tietotyppiä, jolla tehdään tietueiden pääavaimet. Lisäksi työssä käytetään timestamptz-tietotyyppiä auditointilokin merkinnän ajankohdan tallentamiseen.

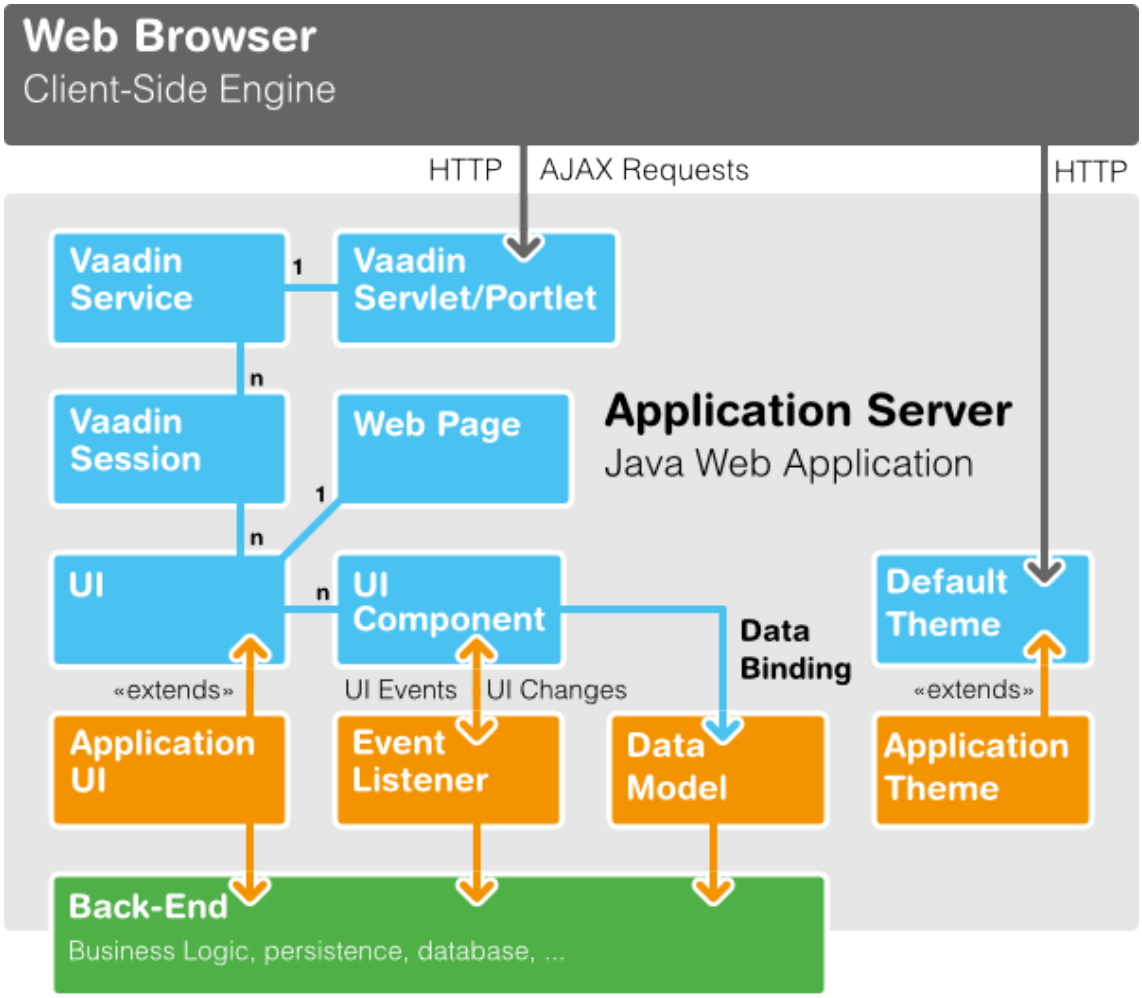
6 VAADIN-SOVELLUSKEHYS

Vaadin on suomalaisen samannimisen yrityksen tekemä avoimen lähdekoodin WWW-sovelluksien kehittämiseen tarkoitettu Java-ohjelmistokehys. Vaadin tukee sekä palvelin-, että asiakaslähtöistä ohjelmointia ja se käyttää selaimessa HTML5-, JavaScript- ja CSS3-teknologioita sekä AJAX-tekniikoita. Vaadin piilottaa web-selaimien käyttämät teknologiat sovelluslogiikasta, jolloin web-selaimen voidaan ajatella olevan kevyt asiakas, joka näyttää käyttöliittymän ja kommunikoi palvelimen kanssa alhaisella abstraktiotasolla. Vaadinin asiakaspuolen ohjelmistokehys käyttää Google Web Toolkitiä, jolla Java-koodi käännetään JavaScriptiksi, joka voidaan ajaa web-selaimissa. [10]

Tässä työssä tehdään palvelinpuolen sovellus, joten Vaadinin tarkastelussa keskitytäänkin suurimmaksi osaksi juuri palvelinpuolen sovelluksiin. Työssä jätetään myös Vaadinin käyttämät CSS-pohjaiset teemat vähemmälle huomiolle, sillä työn pääpaino on sovelluksen toimintojen toteutuksessa eikä käyttöliittymän ulkoasun muokkaamisessa.

Vaadinilla tehdyn sovelluksen arkkitehtuuri on kaavion 2 kaltainen. WWW-sivulla on vähintään yksi *com.vaadin.ui.UI* –luokasta periytetty käyttöliittymäluokka, johon kuuluu erilaisia käyttöliittymäkomponentteja. Komponentteja ovat esimerkiksi yksinkertainen *Label*-luokan teksti tai paljon monimutkaisempi *Table*-luokan taulukko. Käyttöliittymäkomponentit toimivat tapahtumalähtöisen (engl. event-driven) ohjelmointiparadigman kautta tai ne voidaan sitoa jostain muualta saatavaan dataan. Vaadin ohjaa automaattisesti käyttäjien *VaadinServlet*-luokkaan tulevat pyynnöt *VaadinSession*-luokan istuntoihin ja niihin liittyviin käyttöliittymiin. [10]

Vaadin-käyttöliittymässä voidaan myös näyttää erilaisia resursseja, joita ovat esimerkiksi kuvat. Tässä työssä käytetään käyttöliittymän resursseja generoidun raportin lähettämiseen sovelluksen käyttäjälle.



Kuvio 2. Kaavio Vaadinin palvelinpuolen sovelluksen arkkitehtuurista [11].

7 JAVA DATABASE CONNECTIVITY (JDBC)

Java Database Connectivity (JDBC) on standardi, jolla Java-sovellukset voivat ottaa yhteyden tietokantapalvelimeen. JDBC API mahdollistaa komentojen ja SQL-kyselyiden lähettämisen tietokantasovellukseen, sekä tietokantapalvelimen antamien vastausten käsittelemisen. [12]

PostgreSQL:ään on JDBC-ajuri, jonka voi ladata PostgreSQL:n omilta sivuilta. [13] Tässä työssä toteutettavaan Java-sovellukseen ladattiin ko. ajuri, jota tarvitsevat sekä Vaadinin SQLContainer että BIRT-raportointialustan Report Engine -komponentti.

8 SOVELLUSKEHITYSYMPÄRISTÖ JA –TYÖKALUT

Tätä työtä varten pystytettiin oma kehitysympäristö, joka koostui PostgreSQL-tietokantahallintajärjestelmästä, PostgreSQL:n hallintaan käytetystä pgAdmin III –sovelluksesta sekä Eclipse-ohjelmointiympäristöstä. Kehitysympäristöön kuului myös Oraclen Java SE Development Kit 8, Maven ja Mercurial-versionhallintatyökalu, jolla pidettiin kirjaa sovelluksen lähdekoodin muutoksista. Kaikki työssä käytetyt työkalut ovat alustariippumattomia ja toimivat Linux-pohjaisissa sekä Mac OS X- ja Windows-käyttöjärjestelmissä.

9 POSTGRESQL-TIETOKANNAN ASENTAMINEN JA TIETOKANNAN KUVAUS

Tässä työssä PostgreSQL-relaatiotietokannan hallintajärjestelmä asennettiin Windows 8.1 -käyttöjärjestelmälle ja tietokannan muokkaamiseen käytettiin pgAdmin III -sovellusta. Tietokantaan luotiin uusi käyttäjä ”birtuser”, jonka salasanaksi valittiin ”mypassword”. Valittu salasana ei ole hyvien salasanasääntöjen mukainen, mutta se soveltuu tässä opinnäytetyössä esimerkkinä.

Mikäli käyttäjä luotaisiin komentorivipohjaista asiakasohjelmaa käyttäen, komento olisi ”*createuser -U postgres -e -P birtuser*”, jossa “*-U*”-asetuksella kerrotaan, millä käyttäjällä uusi käyttäjä halutaan luoda. Seuraavaksi ”*-P*”-asetuksella luodaan uudelle käyttäjälle salasana, ja ”*-e*”-asetuksella voidaan tulostaa varsinainen tietokantakomento, jota käytettiin uuden käyttäjän luontiin. [14] Uudelle käyttäjälle annettiin rajoitetut oikeudet tietokantojen käsittelyyn, sillä se ei tarvitse pääsyä kuin sen omaan tietokantaan. SQL-lause, jolla uusi käyttäjä luotiin, on kohdassa SQL-lause 1.

SQL-lause 1. Käyttäjän luontiin käytetty SQL-lause.

```
CREATE ROLE birtuser PASSWORD
'md511975f2f2da93ff8b31bcd0bfad4f3ae' NOSUPERUSER
NOCREATEDB NOCREATEROLE INHERIT LOGIN;
```

Seuraavaksi luotiin uusi tietokanta. Uuden tietokannan omistajaksi valittiin edellä luotu käyttäjä ”birtuser”, jolloin se saa täydet oikeudet tietokannan muokkaamiseen superuserin lisäksi. Tietokannan luontiin käytettiin pgAdmin III:a. Käytetty SQL-lause on kohdassa SQL-lause 2.

SQL-lause 2. Tietokannan luontiin käytetty SQL-lause.

```
CREATE DATABASE birtpdb
  WITH ENCODING='UTF8'
       OWNER=birtuser
       CONNECTION LIMIT=-1;
```

Tietokannan ja käyttäjän luonnin jälkeen ”birtpdb”-tietokantaan lisättiin tarvittavat taulut. Tauluja luotiin yhteensä viisi, jotka ovat ”users”, ”values1”, ”values2”, ”values3” ja ”audit_log”.

Käyttäjätaulun nimeksi valittiin ”users”, ja sille luotiin sarakkeet ”username”, ”password”, ”enabled”, ”email”, ”phone”, ”auhority” sekä sarakkeet käyttäjän oikeuksien määrittelemiseen. ”Username”-sarake on character varying(50) -tyyppiä ja sisältää käyttäjännimen, jolla voidaan kirjautua sisään järjestelmään. Tämä sarake on myös rivin pääavain, joten sen pitää olla ainutlaatuinen eikä se saa olla tyhjäarvo. Password-sarake on niin ikään character varying(50) -tyyppiä ja sisältää käyttäjän salasanan. Normaalisti salasanan pitäisi olla tallennettuna tietokantaan kryptograafisena tiivisteenä, mutta koska tässä työssä ei erityisesti käsitellä tietoturvaan liittyviä asioita, salasana tallennetaan tietokantaan selkomuotoisena yksinkertaisuuden vuoksi. ”Phone”- sekä ”email”-sarakkeet ovat molemmat character varying(n) -tyyppiä ja sisältävät käyttäjän puhelinnumeron ja sähköpostiosoitteen. Boolean-tyyppinen ”enabled”-sarake ja character varying(50) -tyyppinen ”authority”-sarake ovat Spring Securityn käyttämiä sarakkeita. ”Enabled” –sarake kertoo, onko käyttäjätili käytössä ja ”authority”-sarake sisältää tiedon käyttäjän roolista. Taulun luontiin käytetty SQL-lause on kohdassa SQL-lause 3.

SQL-lause 3. Käyttäjätaulun luontiin käytetty SQL-lause.

```
CREATE TABLE users
(
  username character varying(50) NOT NULL,
  password character varying(50) NOT NULL,
  enabled boolean NOT NULL DEFAULT true,
  email character varying(256),
  phone character varying(50),
  authority character varying(50),
  right_add_report boolean NOT NULL DEFAULT false,
  right_print_report boolean NOT NULL DEFAULT false,
  right_remove_report boolean NOT NULL DEFAULT false,
  right_edit_report boolean NOT NULL DEFAULT false,
  CONSTRAINT pk_users PRIMARY KEY (username)
)
WITH (
  OIDS=FALSE
); ALTER TABLE users OWNER TO birtuser;
```

Taulut ”values1”, ”values2” ja ”values3” ovat nimiä lukuun ottamatta keskenään identtisiä. Jokainen näistä kolmesta taulusta sisältää sarakkeet ”id”, ”int_val1”, ”int_val2”, ”int_val3”, ”text_val1”, ”text_val2” ja ”text_val3”. Nimiensä mukaisesti ”int_val” -sarakkeet ovat integer-tyyppisiä ja ”text_val”-sarakkeet ovat text-tyyppisiä sarakkeita. ”Id”-sarake on rivin serial-tyyppinen pääavain. Pääavaimena ”id” ei saa olla tyhjäarvo, ja sen pitää olla ainutlaatuinen. Serial-tietotyyppi pitää huolen siitä, että jokainen rivi saa ainutlaatuisen pääavaimen. Esimerkkinä ”values1” taulun luontiin käytetty SQL-lause on kohdassa SQL-lause 4.

SQL-lause 4. "Values1"-taulun luontiin käytetty SQL-lause.

```
CREATE TABLE values1
(
  id serial NOT NULL,
  text_val1 text,
  text_val2 text,
  text_val3 text,
  int_val1 integer,
  int_val2 integer,
  int_val3 integer,
  CONSTRAINT values1_id PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE values1
  OWNER TO birtuser;
```

Raporttien latausten lokikirjaukseen käytettävä ”audit_log”-taulu sisältää ”date”-nimisen timestamptz-tyyppisen sarakkeen, johon tallennetaan tapahtuman ajankohta sekä text-tyyppisen sarakkeen, johon tallennetaan itse tapahtuman kuvaus. Taulu käyttää lisäksi rivin pääavaimena serial-tyyppistä ”id”-saraketta. ”Audit_log”-taulun luontiin käytetty SQL-lause on kohdassa SQL-lause 5.

SQL-lause 5. "Audit_log" taulun luontiin käytetty SQL-lause.

```
CREATE TABLE audit_log
(
  date timestampz,
  log text,
  id serial NOT NULL,
  CONSTRAINT log_id PRIMARY KEY (id)

)
WITH (
  OIDS=FALSE
);
ALTER TABLE audit_log
  OWNER TO birtuser;
```

10 RAPORTTIMALLIN LUONTI RAPORTTISUUNNITTELIJALLA

Raporttisuunnittelija yhdistetään PostgreSQL-tietokantaan käyttäen JDBC-rajapintaa. Raporttisuunnittelijaan lisättiin ensin ”Data Explorer” –näkymästä uusi tietolähde (Data Source), johon määriteltiin tietokannan osoite, portti, käyttäjännimi ja salasana. Lisäksi tietolähteeseen lisättiin tarvittava JDBC-ajuri.

BIRT raporttisuunnittelija ei sisällä oletuksena PostgreSQL:n tarvitsemaa JDBC-ajuria, vaan se pitää ladata erikseen PostgreSQL:n sivuilta osoitteesta https://jdbc.postgresql.org/download.html. Tässä työssä ajuriksi valittiin JDBC41, koska työssä käytettiin Javan versiota 1.8 ja PostgreSQL:n versiota 9.4. Ladatun tiedoston nimi on ”postgresql-9.4-1201.jdbc41.jar”.

Kun raporttisuunnittelija on yhdistetty tietokantaan, voidaan luoda tietojoukko (Data Set). Tietojoukkoon määritetään mitä tietoja tietokannasta halutaan hakea kirjoittamalla sille SQL-kysely. Tässä työssä tehtiin raportti taulun ”values1” tietueista, joten SQL-kyselyksi kirjoitettiin kohdassa SQL-lause 6 oleva kysely. Pienillä muutoksilla kohdan SQL-lause 6 kyselyä voidaan käyttää myös taulujen ”values2” ja ”values3” tietojen hakemiseen.

SQL-lause 6. SQL-kysely, jolla haettiin taulun ”values1” tiedot raporttiin.

```
select values1.text_val1,
values1.text_val2,
values1.text_val3,
values1.int_val1,
values1.int_val2,
values1.int_val3
from values1
```

Tietojoukosta voidaan muokata muun muassa sarakkeiden otsikoita, haettavien rivien määrää, suodattimia ja parametreja. Tietojoukon suodattimilla voidaan määrittää ehtoja, jolloin raportti tehdään vain riveistä, jotka toteuttavat suodattimien ehdot. Näin saadaan esimerkiksi näytettyä tietoja, joita on muokattu vii-

meisen viikon aikana, mikäli taulussa on sarake, joka kertoo koska tietoja on viimeksi muokattu.

Tietojoukon parametreilla voidaan antaa tietojoukolle muuttujia, joilla tietokantakyselyä voidaan muokata raporttikohtaisesti. Tämä mahdollistaa raportin generoimisen esimerkiksi yhden käyttäjän tiedoista, mikäli käyttäjännimi annetaan parametrina raportin generoinnin yhteydessä. Parametreilla voidaan myös lisätä raportoijan omia kommentteja raporttiin.

Tietojoukkoon voidaan myös lisätä laskettuja sarakkeita (Computed Columns), joiden tietueiden arvo lasketaan tietokannan tietojen perusteella, kun raportti generoidaan. Näin voidaan esimerkiksi laskea kokonaislukujen summa tai kahden aikaleiman välinen aikaero, jolloin näitä tietoja ei tarvitse erikseen pitää tietokannassa. Tietojoukon muokkausikkunassa on myös osio tulosten esikatseluun, jossa voidaan nähdä, mitä tietoja tietojoukko hakee tietokannasta. Esikatselusta näkyy myös laskettujen sarakkeiden arvot.

Tietojoukon luomisen jälkeen, raporttipohja voidaan rakentaa raporttieditorin asettelunmuokkausnäkymällä. Raporttiin halutut elementit vedetään työkalupalkista asettelunmuokkausnäkymään, jonka jälkeen niiden ominaisuuksia voidaan muokata Property-editorilla.

11 SOVELLUKSEN TOTEUTUS

Sovellus toteutettiin Java-ohjelmointikielellä Eclipse-ohjelmointiympäristössä. Käytetyn Javan versio oli 1.8, mikä mahdollisti Java 8:n tuomien ominaisuuksien, esimerkiksi lambada-lausekkeiden, käytön sovelluksessa.

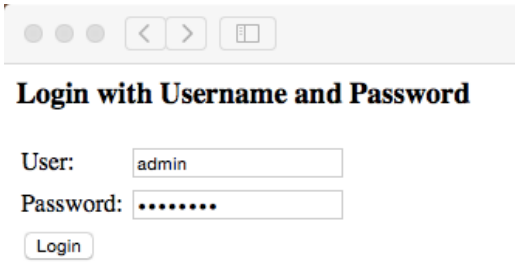
Sovelluksen käyttöliittymä toteutettiin Vaadin-sovelluskehyksellä, sekä osia sovelluksesta toteutettiin käyttäen Spring-sovelluskehystä. Springistä käytettiin muun muassa IoC (Inversion of Control) containeria, joka yksinkertaistaa monimutkaisten Java-olioiden luontia ja käyttöä sovelluksen ohjelmakoodissa. Spring-sovelluskehyksestä käytettiin lisäksi Spring Boot –ratkaisua, johon on integroitu Tomcat 8 –palvelin, jolloin sitä ei tarvitse asentaa erikseen. Spring Boot hoitaa myös monia Spring-sovelluskehykseen liittyviä konfiguroinnin yksityiskohtia automaattisesti.

Vaadin-sovelluskehyksellä tehtiin sovellukseen käyttöliittymä, josta voidaan katsella ja muokata tietokannan sisältämiä tietoja. Käyttöliittymästä voidaan muokata käyttäjien lisäksi varsinaisia raportoitavia tietoja sekä eri raporttityyppejä. Käyttöliittymästä voidaan myös valita raportti tulostettavaksi. Vaadin kytkeytyy PostgeSQL-tietokantaan käyttäen SQLContainer-luokkaa, joka puolestaan käyttää JDBC-rajapintaa. SQLContainerin käyttö on kätevää, sillä sen sisällön saa helposti näkymään Vaadinin taulukkokomponentissa. Lisäksi SQLContainerin sisältämiä tietoja voidaan helposti muokata Vaadin-käyttöliittymässä.

11.1 Käyttöliittymä

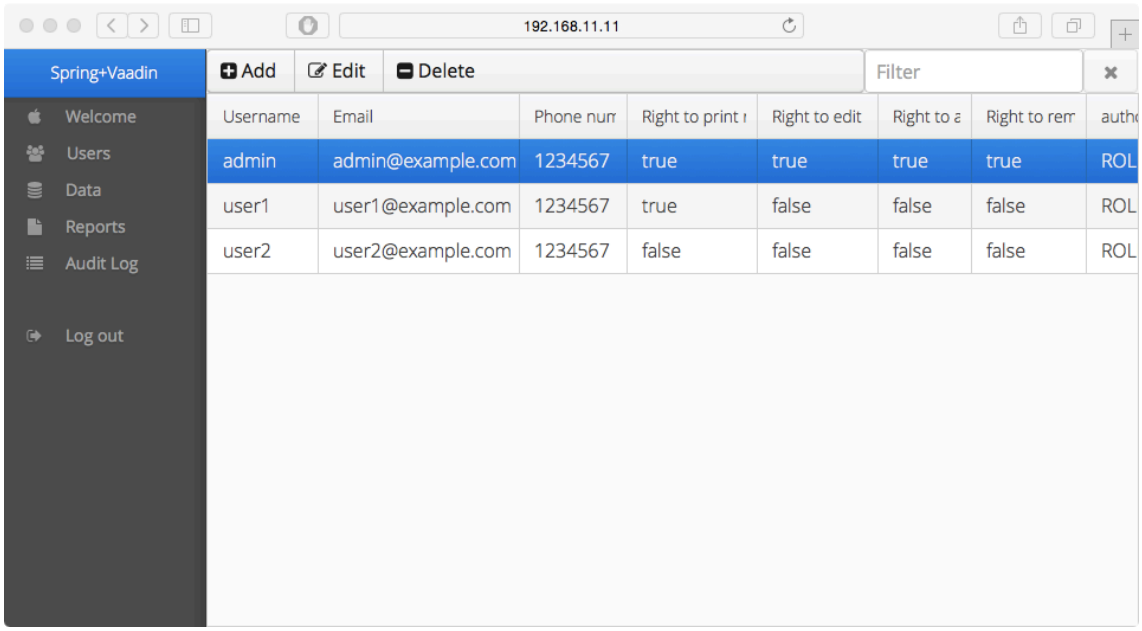
Käyttöliittymä rakentuu siten, että käyttöliittymän vasemmalla reunalla on listattu eri sovelluksen eri näkymät. Käyttöliittymän näkymiä ovat Aloitus-näkymän lisäksi, näkymät käyttäjien ja tietojen katseluun sekä muokkaamiseen. Käyttöliittymässä on myös erillinen näkymä raporttityyppien muokkaamiseen ja erilaisten raporttien tulostamiseen.

Sovelluksen sisäänkirjautumisnäkymä on Spring Securityn oletuksena tarjoama JavaServer Pages -sivu, jossa on kentät käyttäjännimelle ja salasanalle, sekä sisäänkirjautumispainike. Näkymästä on kuvakaappaus kuvassa 1.



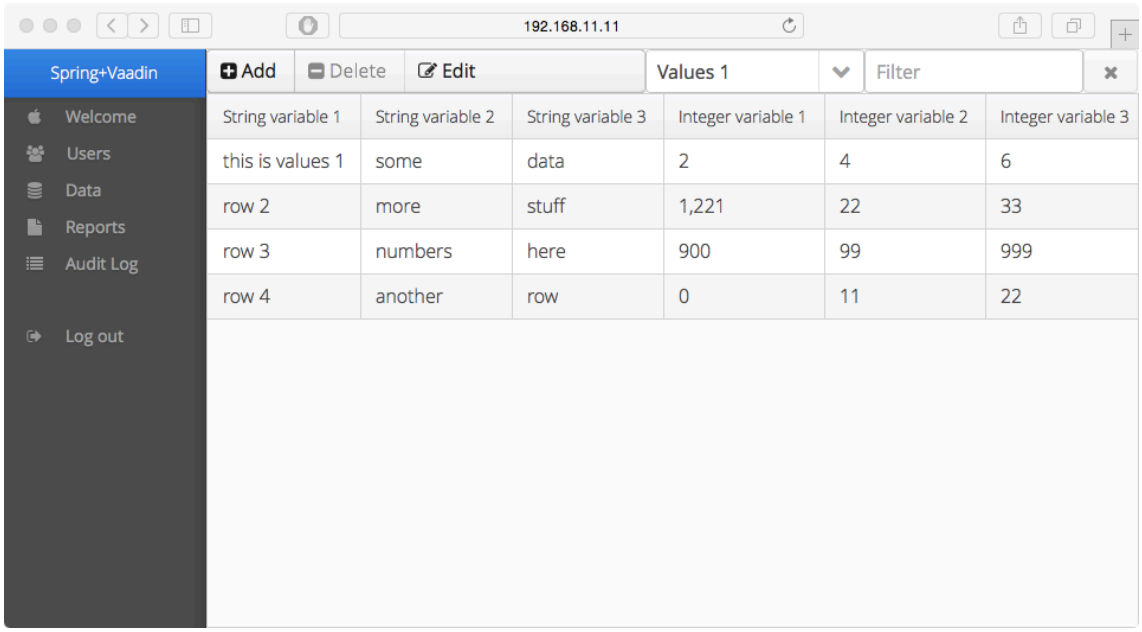
Kuva 1. Kuvakaappaus kirjautumisnäkymästä.

Sovellukseen tehtiin näkymä, jossa voi katsella ja muokata sovelluksen käyttäjien tietoja. Tässä näkymässä voi myös lisätä ja poistaa käyttäjätilejä. Näkymän keskeisin osa muodostuu taulukosta, jossa on listattu käyttäjätilit ja niiden ominaisuuksia. Taulukon yläpuolella on painikkeet uuden käyttäjätilin luomiseen sekä taulukosta valitun käyttäjätilin muokkaamiseen ja poistamiseen. Kuvassa 2 on kuvakaappaus sovelluksen käyttäjienhallintanäkymästä.



Kuva 2. Kuvakaappaus käyttäjienhallintanäkymästä.

Sovelluksessa on tietojenmuokkausnäkymä, jossa voidaan lisätä, muokata ja poistaa ”values”-taulujen sisältämiä tietoja. Näkymän rakenne on samankaltainen käyttäjienhallintanäkymän kanssa. Taulukon yläpuolella on pudotusvalikko, josta voidaan valita taulu, jonka tietoja katsellaan tai muokataan. Kuvassa 3 on kuvakaappaus tietojenmuokkausnäkymästä.

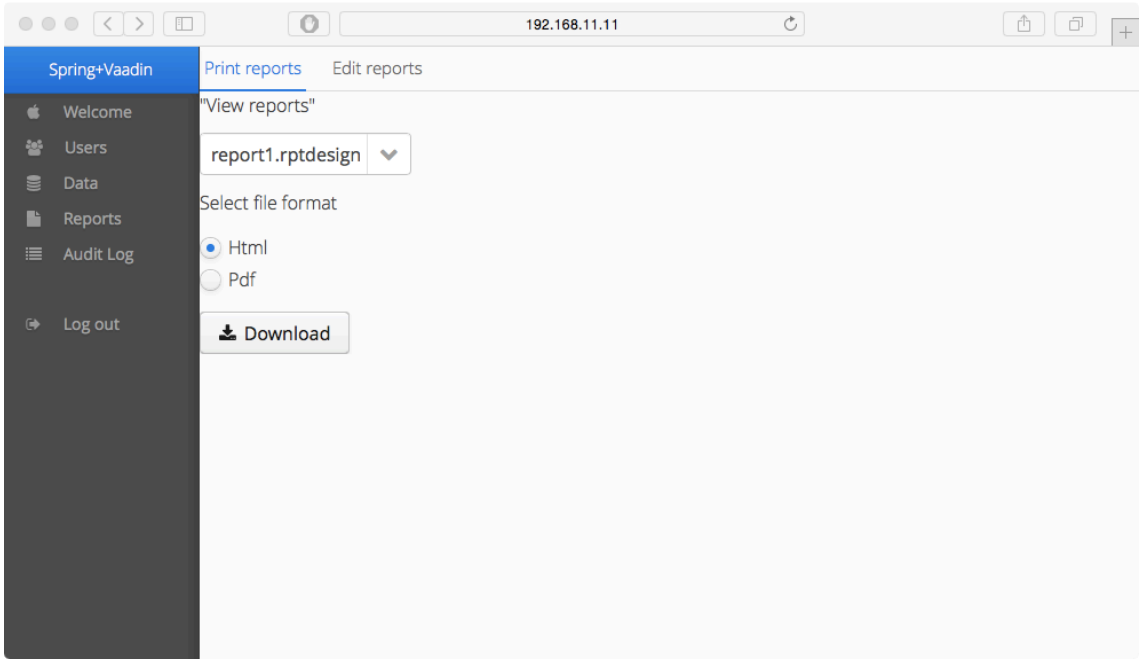


Kuva 3. Kuvakaappaus tietojenmuokkausnäkymästä.

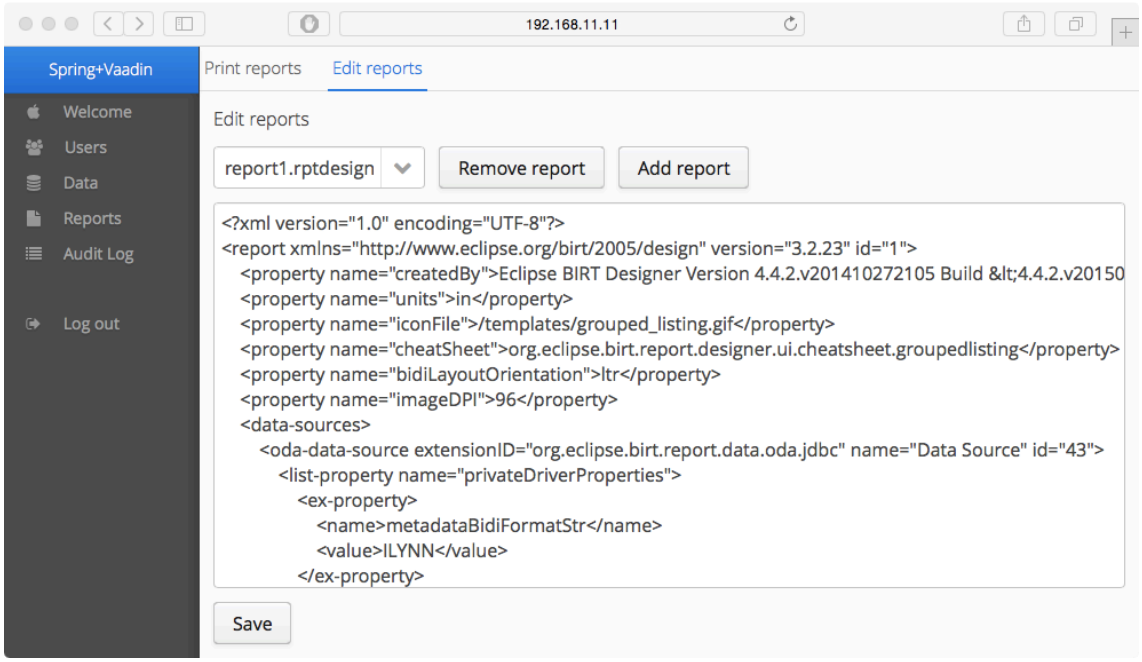
Sovellukseen tehtiin lisäksi näkymät raporttien tulostuksen ja raporttien mallipohjien hallitsemiseen sekä raporttien latauslokin katseluun. Raporttien latausnäkymä ja raporttipohjien hallintanäkymä ovat yhdistetty yhdeksi näkymäksi, joka on erotettu välilehdellä kahteen osaan. Ensimmäisellä välilehdellä voidaan tulostaa raportteja valitsemalla pudotusvalikosta raporttipohjan tiedostonnimi, minkä jälkeen valitaan haluttu tiedostomuoto. Näiden vaiheiden jälkeen painetaan latauspainiketta, jolloin haluttu raportti tallennetaan käyttäjän tietokoneelle.

Toisella välilehdellä on raporttien hallintaan käytettävä osa, jossa voidaan lisätä, poistaa ja muokata sovelluksessa käytettäviä raporttipohjia. Muokattavan tai poistettavan raporttipohjan tiedosto valitaan pudotusvalikosta, jonka jälkeen raporttipohjan tiedoston sisältö näytetään tekstialueella. Raporttipohjaa voidaan

muokata tekstialueessa, mutta parempi tapa on viedä raporttipohjan määrittely Eclipse-ohjelmointiympäristön raporttieditoriin, muokata raporttia siellä ja lopuksi tuoda muutokset sovellukseen. Raporttipohjan muokkaamisen jälkeen painetaan tallennuspainiketta, jolloin raporttipohjan muutokset tallennetaan sovelluksen palvelimelle. Pudotusvalikosta valittu raporttipohja voidaan poistaa painamalla raportinpoistopainiketta ja uuden raporttipohjan voi lisätä raportinlisäyspainikkeesta, josta aukeaa ikkuna, jossa voidaan valita palvelimelle lähetettävä raporttipohja. Kuvassa 4 on kuvakaappaus raporttienhallintanäkymän raporttienlatausvälilehdestä ja kuvassa 5 on kuvakaappaus raporttienhallintanäkymän raporttipohjienhallintavälilehdestä

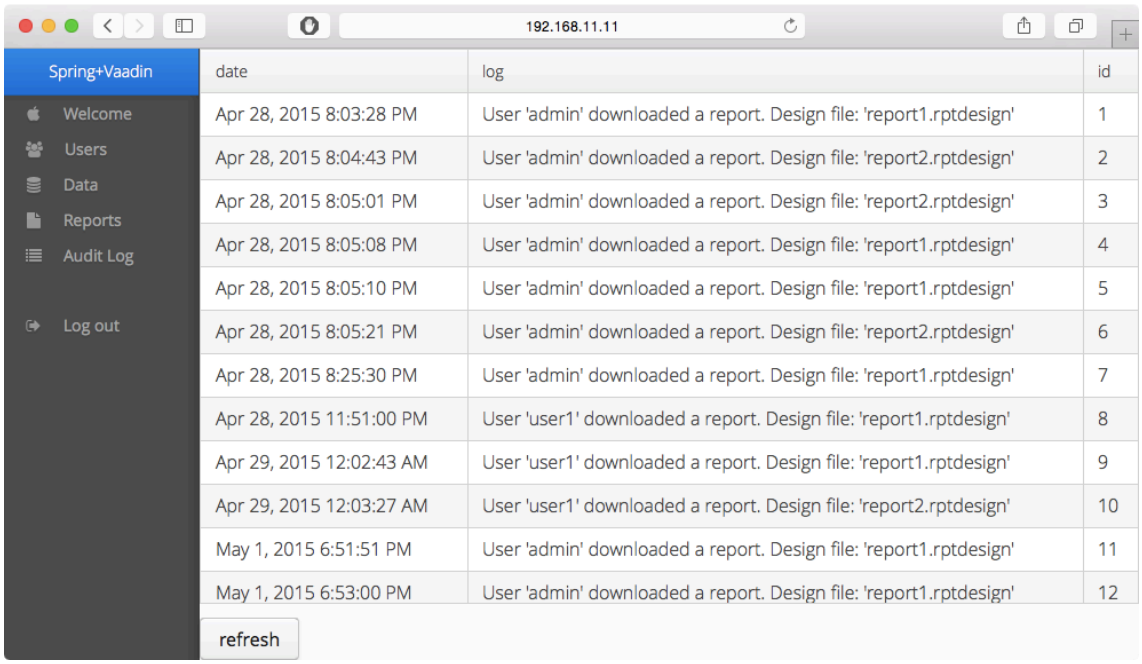


Kuva 4. Kuvakaappaus raporttienhallintanäkymän raporttienlatausvälilehdestä.



Kuva 5. Kuvakaappaus raporttienhallintanäkymän raporttipohjienhallintavälilehdestä.

Sovelluksen latauslokintarkastelunäkymä sisältää yksinkertaisen taulukon, jossa näytetään ”audit_log”-taulun sisältö. Näkymässä on myös painike, josta taulun sisällön voi päivittää. Kuvassa 6 on kuvakaappaus latauslokintarkastelunäkymästä.



date	log	id
Apr 28, 2015 8:03:28 PM	User 'admin' downloaded a report. Design file: 'report1.rptdesign'	1
Apr 28, 2015 8:04:43 PM	User 'admin' downloaded a report. Design file: 'report2.rptdesign'	2
Apr 28, 2015 8:05:01 PM	User 'admin' downloaded a report. Design file: 'report2.rptdesign'	3
Apr 28, 2015 8:05:08 PM	User 'admin' downloaded a report. Design file: 'report1.rptdesign'	4
Apr 28, 2015 8:05:10 PM	User 'admin' downloaded a report. Design file: 'report1.rptdesign'	5
Apr 28, 2015 8:05:21 PM	User 'admin' downloaded a report. Design file: 'report2.rptdesign'	6
Apr 28, 2015 8:25:30 PM	User 'admin' downloaded a report. Design file: 'report1.rptdesign'	7
Apr 28, 2015 11:51:00 PM	User 'user1' downloaded a report. Design file: 'report1.rptdesign'	8
Apr 29, 2015 12:02:43 AM	User 'user1' downloaded a report. Design file: 'report1.rptdesign'	9
Apr 29, 2015 12:03:27 AM	User 'user1' downloaded a report. Design file: 'report2.rptdesign'	10
May 1, 2015 6:51:51 PM	User 'admin' downloaded a report. Design file: 'report1.rptdesign'	11
May 1, 2015 6:53:00 PM	User 'admin' downloaded a report. Design file: 'report1.rptdesign'	12

Kuva 6. Kuvakaappaus latauslokien tarkastelunäkymästä.

11.2 Eclipse BIRT-alustan integrointi

BIRT integrointiin sovellukseen käyttäen Report Engine API (RE API) –rajapintaa. RE API:lla voidaan luoda IReportEngine–olio, jota voidaan käyttää raportin luomiseen. Yhtä sovellusta kohti riittää yksi IReportEngine-olio ja se on myös suositeltavaa, sillä IReportEngine-olion luonti on huomattavan raskas toimenpide. Mikään ei toisaalta kuitenkaan estä useamman IReportEngine-olion käyttöä yhdessä sovelluksessa. IReportEngine-olion luontiin tarvitaan ensin EngineConfig-olio, jolla määritetään IReportEngine-olion asetuksia, muun muassa lokitukseen liittyvät asetukset. BIRT-alusta käynnistetään Platform-luokalla, josta kutsutaan startup-metodia, johon annetaan argumentiksi aikaisemmin luotu EngineConfig-olio. Platform-luokasta saadaan IReportEngineFactory-olio, jota sitten käytetään IReportEngine-olion luontiin. Kun IReportEngine-oliota ei enää tarvita, on tärkeää kutsua destroy-metodia, sekä Platform-luokan shutdown-metodia.

Tämän sovelluksen toteutuksessa käytettiin IReportEngineFactory-oliota, jolla voidaan luoda uusi IReportEngine-olio tai mikäli olio on jo olemassa, siihen annetaan viittaus. ReportEngineFactory-oliota kutsutaan Spring-ohjelmistokyhyksen FactoryBean ja DisposableBean abstraktit luokat toteuttavasta luokasta BirtEngineFactory, jonka toteutus on kohdassa ohjelmakoodi 1.

Ohjelmakoodi 1. BirtEngineFactory-luokan toteutus.

```
public class BirtEngineFactory implements FactoryBean,
ApplicationContextAware, DisposableBean {

    private IReportEngine birtEngine;


    public boolean isSingleton() {
        return true;
    }

    public void destroy() throws Exception {
        birtEngine.destroy();
        Platform.shutdown();
    }

    public IReportEngine getObject() {

        EngineConfig config = new EngineConfig();
        try {
            Platform.startup(config);
        } catch (BirtException e) {
            throw new RuntimeException("Could not start the
            Birt engine!", e);
        }

        IReportEngineFactory factory = (IReportEngineFactory)
        Platform.createFactoryObject(
        IReportEngineFactory.EXTENSION_REPORT_ENGINE_FACTORY);

        IReportEngine be = factory.createReportEngine(config);
        this.birtEngine = be;
        return be;
    }

    @Override
    public Class getObjectType() {
        return IReportEngine.class;
    }
}
```

Sovellukseen tehtiin ReportEngineHandler- luokka, jolla erotettiin BIRT-alustaa käyttävä ohjelmakoodi käyttöliittymän toteutuksesta. Luokassa on metodit raporttipohjan asettamiseen ja varsinaisen raportin generointiin. ReportEngineHandler-luokan toteutus on liitteenä kohdassa Liite 1.

11.3 Raporttien latausten lokikirjaus

Sovellus kirjaa tietokantaan merkinnän jokaisesta ladatusta raportista. Lokimerkintään kirjataan tapahtuman ajankohta sekä viesti, josta käy ilmi käyttäjä ja ladatun raportin mallipohjan tiedostonnimi. Sovelluksen raporttien latausten lokikirjaus toteutettiin myös yksinkertaisesti Vaadinin SQLContainer-luokalla. Raportin latauspainikkeeseen liitettiin tapahtuma, jolloin painiketta painettaessa suoritetaan lokimerkinnän kirjoittava koodi. Kyseinen koodi on kohdassa ohjelmakoodi 2.

Ohjelmakoodi 2. Yksinkertaisen lokikirjauksen toteuttava ohjelmakoodi.

```
download.addClickListener(event -> {
          Authentication auth = SecurityContextHolder.
          getContext().getAuthentication();
          String name = auth.getName();
    logContainer = (SQLContainer) context.getBean(
          "sqlContainer","audit_log",false);

    Date date = new Date();
    Object id = logContainer.addItem();
    Item item = logContainer.getItem(id);
    item.getItemProperty("date").setValue(date);
    item.getItemProperty("log").setValue(
          String.format("User '%s' downloaded a report.
          Design file: '%s'",name, cBox.getValue()));

    try {
        logContainer.commit();

   } catch (Exception e) {
        logger.error(
"Exception committing audit log: ", e);
    }
});
```

11.4 Käyttäjien oikeuksienvalvonta

Käyttäjien oikeuksienvalvonta toteutettiin sovelluksessa siten että, Vaadin-käyttöliittymän komponentteja otetaan pois käytöstä tai jätetään kokonaan asettamatta käyttöliittymään, jos ne mahdollistavat jonkin toiminnon, jota käyttäjän ei haluta antaa käyttää. Tiedot käyttäjien oikeuksista ovat tallennettu tietokannan ”users”-tauluun, josta sovellus käy tarkistamassa ne Vaadinin SQLContainer-luokan avulla. Kohdassa ohjelmakoodi 3 on esimerkki käyttäjän oikeuksien tarkistamisesta.

Ohjelmakoodi 3. Esimerkki käyttäjien oikeuksienhallinnasta.

```
private Authentication auth;
private String userName;


auth = SecurityContextHolder.getContext().getAuthentication();
userName = auth.getName();
SQLContainer users = (SQLContainer)
context.getBean("sqlContainer","users", false);

Item user = users.getItem(new RowId(userName));
Property prop = user.getItemProperty("right_print_report");
boolean hasRight = (boolean) prop.getValue();

if (hasRight) {
        //draw user interface
    }

else {
    addComponent(new Label("You are not allowed to
     generate reports"));
   }
```

12 YHTEENVETO

Tämän opinnäytetyön tuloksena saatiin paljon hyödyllistä tietoa BIRT-raportointialustan käyttöönotosta ja integroinnista Vaadin-sovelluskehykseen ja PostgreSQL-tietokantaan. Tietoja voidaan hyödyntää tulevaisuudessa lopullisessa päätöksenteossa siinä, onko BIRTin integrointi Vaadin-pohjaiseen Java-sovellukseen kannattavaa ja kuinka se voidaan tehdä mahdollisimman tehokkaasti. Lisäksi työssä saatiin aikaiseksi mallisovellus, jonka lähdekoodia voidaan käyttää BIRT-alustan integrointiin tulevissa projekteissa.

Tulevaisuudessa selvitettäviä asioita voisivat olla Design Engine -komponentin integrointi suoraan sovellukseen, jolloin raporttien mallipohjia voitaisiin muokata tai luoda dynaamisesti ilman Eclipse-ohjelmointiympäristön raporttieditoria. Lisäksi voitaisiin selvittää raporttien lokalisointiin, eri tietolähteiden käyttöön sekä raporttien parametrien käyttöön liittyviä asioita.

LÄHTEET

[1] The World Wide Web Consortium, ”Localization vs. Internationalization,” 2015. [www-dokumentti]. Saatavilla: http://www.w3.org/International/questions/qa-i18n.en. [Luettu: 18.5.2015].

[2] The Eclipse Foundation, ”About,” 2014. [www-dokumentti]. Saatavilla: http://eclipse.org/birt/about/. [Luettu: 17.1.2015].

[3] The Eclipse Foundation, ”Eclipse BIRT Architecture,” [www-dokumentti]. Saatavilla: http://eclipse.org/birt/about/architecture.php. [Luettu: 16.3.2015].

[4] The Eclipse Foundation, ”Designer Overview,” 2014. [www-dokumentti]. Saatavilla: http://eclipse.org/birt/about/designer.php. [Luettu: 16.3.2015].

[5] The Eclipse Foundation, ”Integrating BIRT,” 2014. [www-dokumentti]. Saatavilla: http://eclipse.org/birt/documentation/integrating/reapi.php. [Luettu: 20.4.2015].

[6] The PostgreSQL Global Development Group, ”What is PostgreSQL?,” 2015. [www-dokumentti]. Saatavilla: http://www.postgresql.org/docs/9.4/static/intro-whatis.html. [Luettu: 8.3.2015].

[7] The PostgreSQL Global Development Group, ”A Brief History of PostgreSQL,” 2015. [www-dokumentti]. Saatavilla: http://www.postgresql.org/docs/9.4/static/history.html. [Luettu: 8.3.2015].

[8] The PostgreSQL Global Development Group, ”Chapter 8. Data Types,” 2015. [www-dokumentti]. Saatavilla: http://www.postgresql.org/docs/9.4/static/datatype.html. [Luettu: 8.3.2015].

[9] The PostgreSQL Global Development Group, ”PostgreSQL 9.4.1 Documentation: 8.3. Character Types,” 2015. [www-dokumentti]. Saatavilla: http://www.postgresql.org/docs/9.4/static/datatype-character.html. [Luettu: 3.4.2015].

[10] M. Grönroos, Book of Vaadin, Vaadin 7 Edition - 4th Revision, Volume 1, Juva: Bookwell Oy, 2014.

[11] Vaadin Ltd, ”Book of Vaadin, Chapter 4. Writing a Server-Side Web Application,” 2015. [www-dokumentti]. Saatavilla: https://vaadin.com/book/-/page/application.html. [Luettu: 2.4.2015].

[12] Oracle, ”JDBC Overview,” [www-dokumentti]. Saatavilla: http://www.oracle.com/technetwork/java/overview-141217.html. [Luettu: 17.1.2015].

[13] The PostgreSQL Global Development Group, ”PostgreSQL JDBC Driver,” 2013. [www-dokumentti]. Saatavilla: https://jdbc.postgresql.org/download.html. [Luettu: 17.1.2015].

[14] The PostgreSQL Global Development Group, ”PostgreSQL 9.4.1 Documentation, createuser,” 2015. [www-dokumentti]. Saatavilla: http://www.postgresql.org/docs/9.4/static/app-createuser.html. [Luettu: 26.3.2015].

ReportEngineHandler-luokan toteutus

```
@Service
public class ReportEngineHandlerImpl {

    public enum Format {
        HTML, PDF
    }


    private IReportEngine engine;
    private Logger logger;
    private IReportRunnable design;

    @Autowired
    ApplicationContext context;

    @PostConstruct
    void init() {
        logger = LoggerFactory.getLogger(
    ReportEngineHandlerImpl.class);


        engine = (IReportEngine) context.getBean(
        "engineFactory");
        logger.info("Loaded ReportEngine: "
        + engine.toString());
    }

    public void setDesing(String desingPath) {
        logger.info("Design path: " + desingPath);
        try {
            design = engine.openReportDesign(desingPath);
        } catch (EngineException e) {
            logger.error("Caught exception: ", e);
        }
    }


    public ByteArrayOutputStream getFormattedReport(
    Format format) {
        if (design == null)
            throw new IllegalStateException(
           "Desing is null. Use setDesign(design)
            before getReport(format)");


        IRunAndRenderTask task =
        engine.createRunAndRenderTask(design);
        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        IRenderOption options = new RenderOption();
```

```
switch (format) {
            case HTML:
                HTMLRenderOption    htmlOptions
                = new HTMLRenderOption(options);
                htmlOptions.setOutputFormat("html");
                // htmlOptions.setEmbeddable(true);
            break;
            case PDF:
                PDFRenderOption pdfOptions =
                new PDFRenderOption(options);
                pdfOptions.setOutputFormat("pdf");
            break;
            default:
                throw new IllegalArgumentException("Format '"
                +format.toString()+"' not supported");
    }

        options.setOutputStream(bos);
        task.setRenderOption(options);

        try {

          task.run();
        } catch (EngineException e) {
            logger.error("Caught exception: ", e);
        } finally {
            task.close();
        }
        return bos;
    }
}
```