

Govinda Shrestha

# Web Application in Java for Tour Enrolment

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

2nd February 2016

Author(s) Title	Govinda Shrestha Web Application in Java for Tour Enrolment
Number of Pages Date	34 pages +2 appendices 2 <sup>nd</sup> February 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Olli Hämäläinen , Senior Lecturer
<p>The objective of the project was to develop a web application based on the Java platform for students to send an enrolment request for the course called Study tour to St. Petersburg and for teachers to process the request on similar web interfaces.</p> <p>The course Study Tour to St Petersburg is a course offered by Helsinki Metropolia University of Applied Sciences in which student can get an opportunity to travel to the city of St. Petersburg in Russia. At present, a student willing to be a participant in the course needs to send an email to the teacher and the teacher confirms the request. The goal of the project was to make this request and response for an enrolment easier and organised by using a web application where the students can submit their request with their details in one side, while on the other the teacher can have easier view of the requests of the students and process them.</p> <p>The development of the project was carried out in various steps. The steps included analysing the requirements followed by application design based on the requirements and then the development of the system afterwards. Finally, the project was made ready with the targeted goals.</p> <p>The result of the project was the website which meets the targeted objectives of this project. The website is fully functional and currently being hosted by a cloud service provider called OpenShift Red Hat.</p>	
Keywords	Java EE, JavaServer Faces, JasperReport, iReport, JDBC, Servlet

## Contents

1	Introduction	1
2	Theoretical Background	2
2.1	Java Enterprise Edition	2
2.1.1	J2EE Architecture	3
2.1.2	J2EE Containers	4
2.2	Java Server Faces	6
2.2.1	JavaServer Faces Application	7
2.2.2	Facelets	8
2.2.3	ManagedBeans	9
2.3	Jaspersoft Studio (Report Designer)	12
2.3.1	JasperReports	12
2.3.2	iReport Designer	13
2.3.3	Report Life Cycle	15
3	Development and Deployment	17
3.1	Requirement Analysis	17
3.2	Application Design	18
3.3	Development Tools and System Development	19
3.3.1	Tools and Development Environment Set-up	19
3.3.2	Web Filter	22
3.3.3	Database Tables and the View Controller	24
3.4	Deployment	26
4	Results	29
5	Evaluation	32
5.1	Benefits	32
5.2	Challenges	32
6	Conclusion	34
	References	35
	Appendices	
	Appendix 1. Implementation of Servlet Filter	
	Appendix 2. Maven Project Object Model	

## List of Abbreviations

API	Application Programming Interface
CSV	Comma Separated Values
EJB	Enterprise Java Bean
EL	Expression Language
GUI	Graphical User Interface
HTML	Hypertext Mark-up Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JDBC	Java Database Connectivity
JDK	Java Development Kit
JNDI	Java Naming and Directory Interface
J2EE	Java Enterprise Edition
JRE	Java Runtime Environment
JRXML	JasperReport file format in XML
JSF	Java Server Faces
JSP	Java Server Pages
JSTL	JSP Standard Tag Library
JVM	Java Virtual Machine
LGPL	Lesser General Public License
POJO	Plain Old Java Object
POM	Project Object Model
SQL	Sequential Query Language
SSH	Secure Shell
URL	Uniform Resource Locator
XHTML	Extensible Hypertext Mark-up Language
XML	Extensible Mark-up Language

## 1 Introduction

There are many different ways of learning. Among the different ways of learning, international study, tour is an opportunity for students and learners to travel different countries and experience the cultures, peoples and their lifestyles. The students get chance to see and experience what they have learned from books about the people and culture. For these reasons, Helsinki Metropolia University of Applied Sciences offers a course named "Study Tour to St. Petersburg" in which students get opportunity to travel St. Petersburg in Russia.

For a student willing to be a participant of the course, s/he has to send an email request to the teacher with personal details like name, nationality, passport number, student number, email address, date of birth and other contact details. The teacher on the other hand, has to go through all the emails and process the enrolment requests, which is somehow difficult work since there is no organised view of all the students applying for the course. Therefore, for this reason, some system was required, which would make the work of sending and receiving the enrolment application easier and more organised.

The main goal of this project was to develop a simple web application that would help teachers for the process of enrolling the students to the course. In addition, this project would help students to send their enrolment requests easily with the help of a form available in the application. Additionally, this project would help teachers to keep the records of the students in managed way.

## 2 Theoretical Background

The application has mainly two parts: JSF, at the front-end and the back-end handled by J2EE. A web application is a dynamic extension of the web server. Web applications are of the following types:

1. **Presentation-oriented:** A presentation-oriented web application generates web pages with various mark-up languages such as HTML, XHTML, and XML and dynamic content in response to requests.
2. **Service-oriented:** A service-oriented web application implements the endpoint of the web service. Presentation-oriented applications are often the clients of service-oriented web applications.

The web application developed during this project is a presentation-oriented web application as well as service-oriented. The application has various web pages created using HTML and XHTML on the one hand while on the other the application has the implementation of web services.

### 2.1 Java Enterprise Edition

J2EE is an acronym for Java 2 Enterprise Edition, which is a collection of APIs that can be used to build large-scaled, distributed, component-based, multi-tier applications. The J2EE platform is a set of standard specifications that describe application components, APIs, and the runtime containers and services of an application server [9].

J2EE comprises of many APIs some of which are used by the J2EE environment while some provide application-specific services. A list of the most used technologies in J2EE application is as follows:

- Java Servlet
- Java Server Pages (JSPs)
- Enterprise Java Beans(EJBs)
- Java Naming and Directory Interface(JNDI)
- Java Database Connectivity (JDBC)
- Java Mail

- Java Transaction Service
- Java Transaction API
- J2EE Connector Architecture

Depending upon the requirements, developers may use different technologies during application development. For the purpose of our application, several technologies listed above have been used, including Java Servlet, JDBC, JSP and EJBs.

### 2.1.1 J2EE Architecture

The J2EE uses a multi-tiered application architecture and divides the application into multiple logical parts which are often on different servers. The J2EE applications are made up of different components like JSP, Java Servlet, and EJB modules [10]. The J2EE specification defines the following J2EE components:

#### Client Components

Client components or the client-tier consist of application clients that access the Java EE server and are usually located on different machines from the server [11]. They can be both web-based or non-web-based. The clients can be a web-browser, a standalone application, or other services.

#### Web Components

Web components are responsible for handling the communication between the clients and the business tier. In addition, they collect user input from the client interface and return the results provided by the business-tier. They maintain the state and generate various dynamic contents as per client requests, and hold some data temporarily in JavaBeans. Servlet, JSF, JSP, JSTL, Java Beans Components are used in the web tier in Java EE applications. [12]

#### Business Components

Business components being part of the business tier provide the business logic for the application. A business logic is the functionality specific to the business domain, like the financial industry, or an e-commerce site. The business logics are contained in

different enterprise beans. An enterprise bean receives data from a client, processes it, retrieves data from the database, processes it and sends the results to the client. JAX-RS, JAX-WS, Java Persistence API entities, and Enterprise JavaBeans are the technologies used in the Business Tier. [13]

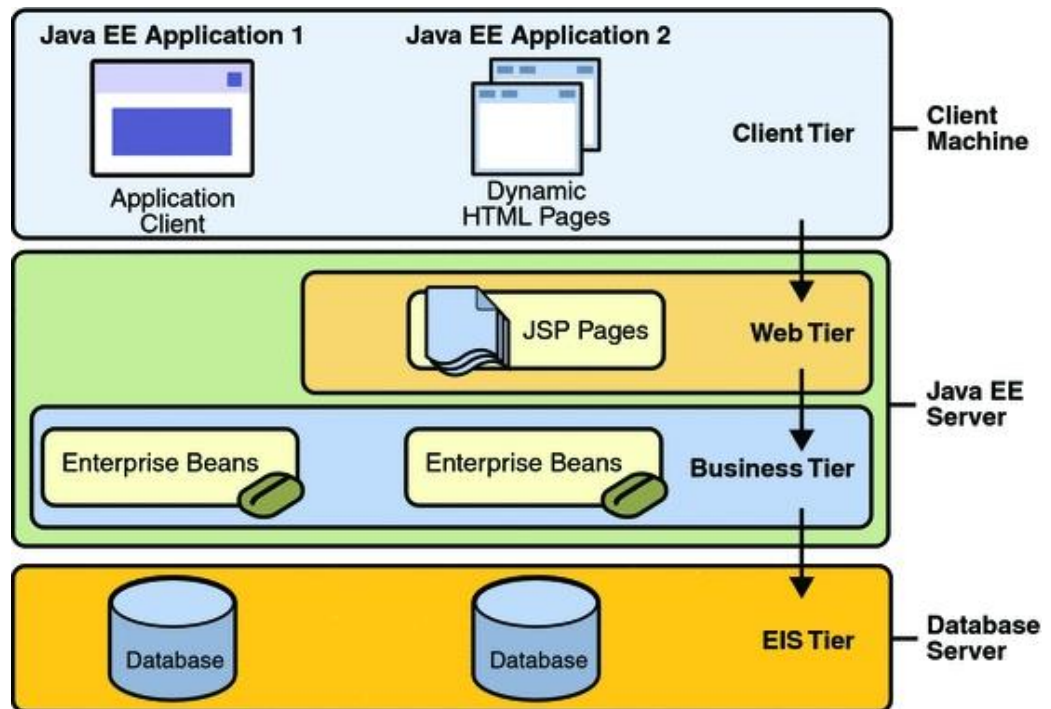


Figure 1 J2EE Multi-Tier Application Models. Reprinted from J2EE Tutorials [14]

Figure 1 is an example of a multi-tier J2EE application model. The figure consists of four different layers, namely client-tier, web-tier, business-tier and Enterprise Information System-tier. The client-tier runs on the client machine whereas the web-tier and the business-tier run on the Java EE server. Similarly, EIS-tier runs on the EIS server.

### 2.1.2 J2EE Containers

A container is a runtime environment that provides support for the J2EE components. A container provides methods and protocols for J2EE application components to interact with other components along with services such as security and transaction management. There are three server-side and two client-side containers. These are listed below:



- The **server** itself is a container providing a J2EE runtime environment.
- The **Web container** that hosts web applications and extends the web server functionality by providing developers with the environment to run servlets and JSPs
- The **EJB Container** that provides local and remote access to enterprise beans. It is also responsible for creating enterprise beans and binding to the naming service, ensuring the authorized access to the bean's methods and managing the EJB components.
- The **application container** on the client side is responsible for providing runtime environment for GUIs, consoles and batch-type programs. It is a combination of Java classes, libraries and other files. It is used to distribute along with Java client programs that execute on their own JVM [14].
- The **applet container**, a client side container, specifically is a browser with Java Plug-in. An applet can be embedded in a HTML page by using the tags `<APPLET>` and `</APPLET>`. They are used to indicate the browser that it should load the Java applet [14].

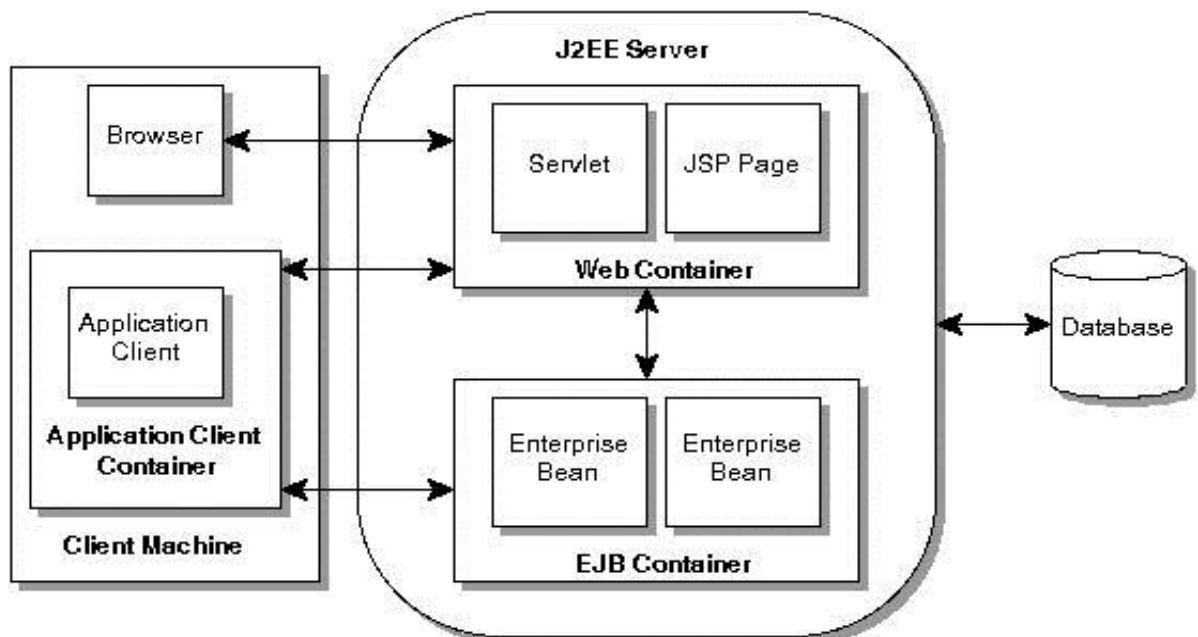


Figure 2 J2EE Server and Containers. Reprinted from pawlan.com [16]

Figure 2 shows the types of containers in the J2EE application. The EJB and the EJB container run on the J2EE server. The EJB Container manages the execution of the enterprise beans for the J2EE application. Similarly, the web container is responsible

for the execution of the JSP pages and servlet components whereas the application client container manages the execution of all application client components for the J2EE application. The application clients and the containers run the client machine. The applet container resides in the client machine and is the web browser and Java Plug-in combined. [16]

## 2.2 Java Server Faces

JavaServer Faces is a web-based server-side application framework intended to simplify the development integration of web-based user interfaces. This MVC-based web framework defines a set of standard UI components and provides the Application Programming Interface (API) for developing components. [5]

Benefits of JSF technology include the following:

- One of the most important advantages of JSF is that it offers clean separation of logic and presentation for web applications and hence allows the developers to focus on a single piece during the development process and provides an easy way to link them programmatically.
- JSF technology APIs are layered directly on top of the Servlet API which enables the developers to create their own custom components directly from the component classes and generating output for various client devices. [5]



Figure 3 Layering of Java Web Application Technologies [5].

- With the release of JSF 2.0, third-party libraries like PrimeFace and OpenFaces supporting JSF 2.0 have been released; they enable developers to have more flexible tools in web development.

- JSF provides a rich architecture for managing and processing component data, validating user inputs, and handling events.
- JSF allows custom UI components to be easily built and re-used [6].

JSF contains all the necessary code for event handling and component organisation. JSF is the view layer in the Java EE standard and is included in every Java EE application server. Unlike most web frameworks, JSF is one of the most popular frameworks for Java web applications with multiple implementations. [3]

### 2.2.1 JavaServer Faces Application

A JSF application is similar to other Java based web applications. A JSF application basically consists of Java beans. A Java bean is a serializable Java class having no argument constructor with `getter` and `setter` methods for its properties and coded according to the JavaBeans API specifications. Moreover, a JSF application consists of the following elements:

- a set of backing beans also called **managed beans**, acting as controllers which are basically Java classes containing application specific functions
- a set of tag libraries which may be customised for representing event handlers
- UI components
- validators, event handlers, navigation handlers
- a web deployment descriptor (**web.xml** file)
- a set of custom tags for custom objects representation
- optionally, one or more **application configuration resource files**, like *faces-config.xml* file, which can be used to define page navigation rules and configure beans and other custom components.

The development of web applications using the JavaServer Faces technology is easy since the technology is easy and user-friendly. The process of the development is very straight forward and involves the following tasks:

- Developing the managed beans
- Creating the web interfaces using the component tags and tag libraries
- Mapping the `javax.faces.webapp.FacesServlet` instance.

## 2.2.2 Facelets

The term **Facelets** refers to a powerful lightweight page declaration language for building JSF views. The facelets also uses simple HTML and XHTML style templates to create a JSF view. It supports Facelets tag libraries along with JSF and JSTL tag libraries, also the Expression Language (EL). The tag library refers to the set of tags used for defining the UI Component which get rendered into corresponding html output. One of the most important feature of Facelets is it allows the reuse of components. It uses JSF custom components natively and hence it becomes easier to combine JSF and Facelets. In other words, Facelets is declaration of JSF components in a Facelets tag library. [26]

As JSF has great support to various tag libraries to be implemented in a web page, Facelets uses XML namespace declarations to support JSF. Table 1 shows different tag libraries supported by the Facelets: [26]

Table 1 Tag Libraries Supported by Facelets. Reprinted from The Java EE 6 Tutorial (2013) [26]

Tag Library	URI	Prefix	Example	Contents
JavaServer Faces Facelets Tag Library	<a href="http://java.sun.com/jsf/facelets">http://java.sun.com/jsf/facelets</a>	ui:	ui:component ui:insert	Tags for templating
JavaServer Faces HTML Tag Library	<a href="http://java.sun.com/jsf/html">http://java.sun.com/jsf/html</a>	h:	h:head h:body h:outputText h:inputText	JavaServer Faces component tags for all UI Component objects
JavaServer Faces Core Tag Library	<a href="http://java.sun.com/jsf/core">http://java.sun.com/jsf/core</a>	f:	f:actionListener f:attribute	Tags for JavaServer Faces custom actions that are independent of any particular render kit
JSTL Core Tag Library	<a href="http://java.sun.com/jsp/jstl/core">http://java.sun.com/jsp/jstl/core</a>	c:	c:forEach c:catch	JSTL 1.2 Core Tags
JSTL Functions Tag Library	<a href="http://java.sun.com/jsp/jstl/functions">http://java.sun.com/jsp/jstl/functions</a>	fn:	fn:toUpperCase fn:toLowerCase	JSTL 1.2 Functions Tags

Facelets makes the web application development easier and faster. Some of other features of Facelets are listed below:

- uses HTML style templates and XHTML for web pages
- supports Facelets tag libraries along with JSTL
- supports EL

### 2.2.3 ManagedBeans

JavaServer Faces managed beans are Java Bean classes that store the application data and are managed by the container. They are registered with the JSF runtime via an XML file and are initialized during runtime when they are needed by the application. [4]

A JSF ManagedBean, generally annotated as `@ManagedBean`, is a regular Java Bean Class associated with components and is used to manage the components used in a particular page. It contains the getter and setter methods, business logic of the application or even a bean containing the HTML form values, a backing bean. Hence, a managed bean works as a model for the UI component and is easily accessed from a JSF page. A managed bean can have several functions like validating a component's data, handling an event called by a component and navigating between the pages and. [8]

JavaServer Faces support bean annotations for configuring JSF applications. The scope annotations set the scope where the managed bean will be used. There are various scopes to which a bean can be configured. The managed bean types and their scopes are listed in table below:

Table 2 Managed Bean annotation and Scope description [9]

<b>Managed Bean with annotation</b>	<b>Scope description</b>
<b>@RequestScoped</b>	A bean without a specified scope is by default request scope. A bean with a request scope is created by a HTTP request and destroyed after the HTTP response.
<b>@NoneScoped</b>	A bean annotated with none scope is created by an EL evaluation and destroyed immediately after evaluation.
<b>@ViewScoped</b>	A bean annotated with a view scope lasts until the user interacts with the same JSF view in the browser. It is created upon a HTTP request and destroyed whenever user moves to a different view.
<b>@SessionScoped</b>	A session scoped bean is created with the first HTTP request involving the bean and is destroyed after end of session.
<b>@ApplicationScoped</b>	An application scoped bean is created with the first HTTP request involving the bean and is destroyed after the web application shuts down.
<b>@CustomScoped</b>	A custom scoped bean lives as long as the bean's entry in the custom Map created for this scope.

JSF 2.0 has annotation to register the managed bean as `@ManagedBean (name="beanName")` or simply `@ManagedBean`. There are two ways of configuring a managed bean, listed below:

1. Configuring Managed Beans with scope annotation: The managed beans can be configured with the help of annotation. A managed bean can be annotated by writing `@ManagedBean` in the bean class just before the class definitions as shown in the listing below. The managed bean can be given a name which will afterwards be used to access the bean and its properties from the JSF pages. Also, the session of the bean can be defined in the class file as shown in the listing below:

```

package example;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import java.io.Serializable;
@ManagedBean(name="helloBean")
@SessionScoped
public class HelloBean implements Serializable {
    private static final long serialVersionUID = 1L;
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;    }
public HelloBean(){
}
}
}

```

Listing 1. Configuring managed bean and session in class file

2. **Configuring Managed Beans with faces-config.xml file:**The managed beans can be configured in the `faces-config.xml` file as well as the session can be defined in the same file. The application scans the configuration file at the start-up for managed beans and their scopes. The listing 2 shows the configuration of managed bean in configuration file:

```

<?xml version="1.0" encoding="UTF-8"?>
<faces-config
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-
facesconfig_2_0.xsd"
    version="2.0">
<managed-bean>
    <managed-bean-name>helloBean</managed-bean-name>

```

```

        <managed-bean-class>com.package.HelloBean</managed-
bean-class>
        <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
</faces-config>

```

Listing 2. Configuring managed bean and session in `config.xml` file

Listing 2 is an example of `config.xml` file. The tag `<managed-bean-name></managed-bean-name>` includes the name of the managed bean being used in the application.

## 2.3 Jaspersoft Studio (Report Designer)

Jaspersoft Studio (JSS) is an Eclipse-based stand-alone report designer application for JasperReports and JasperReports Sever which can be used to create and design sophisticated layouts containing charts, images, sub reports, crosstabs, and other features. This application can access data and database using JDBC, TableModels, JavaBeans, XML, Hibernate, and CSV along with custom sources and is able to print the report in a variety of formats like PDF, RTF, CSV, XLS, DOCX, HTML, XHTML and OpenOffice. JSS has developed APIs like iReports and JasperReports for designing and publishing such reports.

### 2.3.1 JasperReports

JasperReports, an API developed by JSS, is the world's most popular open-source reporting engine entirely based on Java. It can use data from any kind of source and process to produce pixel perfect documents that can easily be viewed and printed or exported to various formats like HTML, PDF, Excel and Word. JasperReports is distributed under two licenses: Apache-style and the LGPL license.

JasperReports Library contains all the required library files for the JasperReports and offers an interface to the reporting engine. The library requires the Java JDK or 1.6 or higher and also JDBC 2.1 driver. The JasperReports is basically a *jar* file which can be installed by adding the jar file to the Java classpath along with other required jar files supporting the library. [17]



Features of JasperReports are listed below:

- Pixel-perfect page-oriented output for print or web
- Textual or graphical presentation of data
- Possibility of generating sub reports
- Possibility of exporting reports to variety of formats like PDF,HTML,DOCX,XLS etc
- Dashboards, tables, crosstabs, charts, gauges, and widgets
- Interactive table elements and sub-reports for interactive and complex layouts
- Possibility of any data source connectivity like JDBC, XML file, JavaBeans, Hibernate connection, EJBQL connection
- Possibility of parsing parameters from the application to the JasperReports[18].

### 2.3.2 iReport Designer

iReport is a Java-based, open-source report designing application using JasperReports Library which allows visual designing of reports. With iReport and its simple yet rich GUI, graphical reports can be designed easily without having to know about the JasperReports library.

iReport provides a wide range of connectivity to different data sources: relational database being the most common one. It provides the JDBC driver for the SQL-compliant database systems like MySQL, PostgreSQL, HSQL, Oracle 10g/XE, Microsoft SQL, and JavaDB.

Features of iReport are listed below:

- Support for wide range of JasperReports tags
- Support for visual designer with tools like rectangles, ellipses, text fields, charts etc
- Support for Unicode and non-Latin language
- Built-in editor with syntax highlighting for write expression
- Support for all JDBC complaint databases and JRDataSource
- Support for sub reports and templates [19].

## User Interface

iReport Designer has a well-equipped powerful design environment for designing reports. Reports can be designed from scratch or also from various templates. The user interface environment is very simple and user friendly and has different sections for easy work flow. The figure 4 shows the user interface of iReport Designer with different sections. The UI has Report Designer section where a report can be designed visually by dragging and positioning the report elements. The Report Inspector contains the structure of the report while the Element Palette contains drag gable elements for easy report designing. Likewise, the Property Sheet allows to set the properties of the currently selected components in the report like field, element, and band. [20]

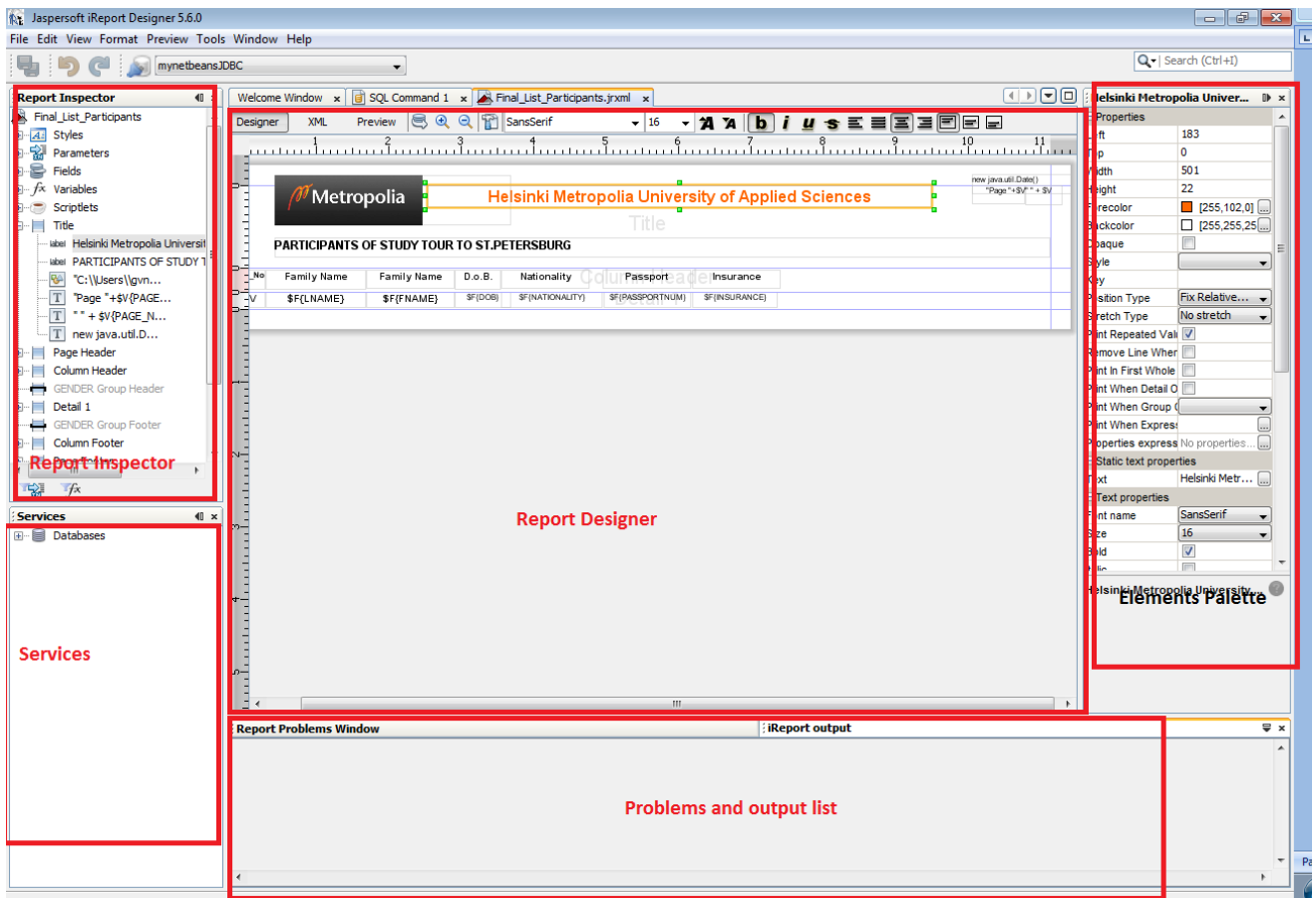


Figure 4 UI of iReport Designer with different sections.

The Figure 4 shows the main UI of the iReport Designer. The UI has three different tabs: Design, Source, and Preview. The Design tab is the tab that is active when a report is opened to be designed while the Source tab contains the JRXML source code

of the report being created or opened. Likewise, the Preview tab lets the user to preview the actual output of the design with the data. The UI window has different sections such as the Report Inspector for viewing the outline of the report, the Services palette for establishing new database connections for the data source to be used in the report, the Report Designer window to design the report, the Problems and Output list section for providing the views of problems incurred during compiling the report, and the Elements palette for providing the information about the elements being used.

### 2.3.3 Report Life Cycle

iReport makes the work of designing and publishing a report very easy and understandable. The report generation takes place in various steps. Firstly, the report is designed in iReport using different elements, like draggable elements, graphics as per requirement which will create a **JRXML** file, an XML document containing the definition of the report layout. In short, to design a report is creating a JRXML file. In iReport Designer, the designing of a report is completely visual.

Next, the JRXML file should be compiled into a binary object called **Jasper file (\*.jasper)**. The report execution is done by passing a Jasper file and a data source to JasperReports. The data source can be a SQL query, XML file, a file, CSV file, an HQL query, collection of JavaBeans, etc. or a custom data source.

The configuration of a data source and filling the data in the report can be easily done in iReport. After the data source has been configured, the final report can be generated in the desired format like PDF, CSV, DOCX, XLX, etc.

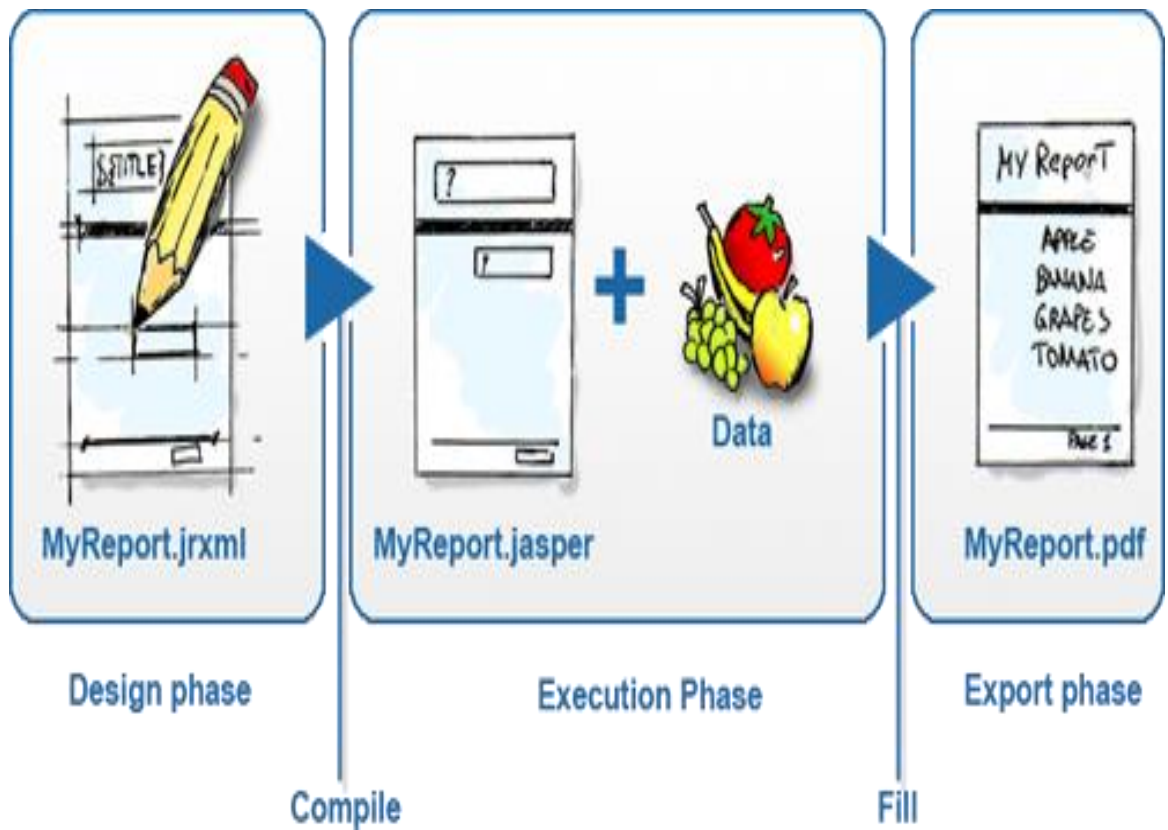


Figure 5 Report Life Cycle. Reprinted from TIBCO Software, Inc. [20]

The step-wise process of a report life cycle is shown in Figure 5. First, the report is designed using various tools as per requirement and saved with file extension **.jrxml**. Once the design is ready, it is then compiled by the jasper compiler and produces a compiled file with an extension **.jasper**. During execution phase, the data required are fed into the jasper file by the JasperFillManager from the specified data source. Once the JasperFillManager feeds the jasper file with the data, a printable format of the report becomes ready. The report can be exported to PDF, DOCX, XLS, RTF or CSV as per the user requirement.

### 3 Development and Deployment

#### 3.1 Requirement Analysis

As mentioned in section 1, the main purpose of the project was to develop a simple and systematic web application for students to enrol for the course by filling in an application form online rather than sending an email request to the teacher. The request sent by the student then can be processed by the teacher afterwards. The student data is collected by the web-based form filled up by the student at the time of the enrolment application which can later be used by the teacher for visa procedures. Hence, analysing the requirements, the basic requirements can be listed as:

- A web-based interface where a student can request for an enrolment request with a form containing their details
- A web interface where the teacher can view the requests of the students with their detailed information and respond afterwards
- A web interface where a teacher can add and edit student information
- A web interface to generate a report about the students who have been enrolled along with their other details
- To provide views of a managed dimensional data.

Analysing the requirements, the application required two basic interfaces: first, a client-side interface for students where they can fill in a form and submit the request for enrolment and secondly an admin side interface where a teacher can view the details of the requests and, add and edit the information about the student and other data regarding the visa procedures. The admin side has to be secured with a login form.

The other requirement of generating the report about the student can be done by creating a template which will afterwards dynamically create the report of the student. The template can be created by using the iReport Designer. The designer will create the report template in `.jrxml` format which after compiling will be in `.jasper` format so that the student data during the runtime can be easily used to generate the report in the required format.

The web application developed for this project cannot only be used to enrol for this course but can also be used to keep the records of students in a systematic way. On

the other hand, the application also simplifies the job of a teacher to keep track of students who have requested for enrolment and the process of visa application.

### 3.2 Application Design

After analysing the requirements of the application, the following application components needed to be designed:

- MVC module with different web interfaces based on the requirements
- HTTP form for authentication of admin login
- Controller classes for database handling, report generating and other utility classes.

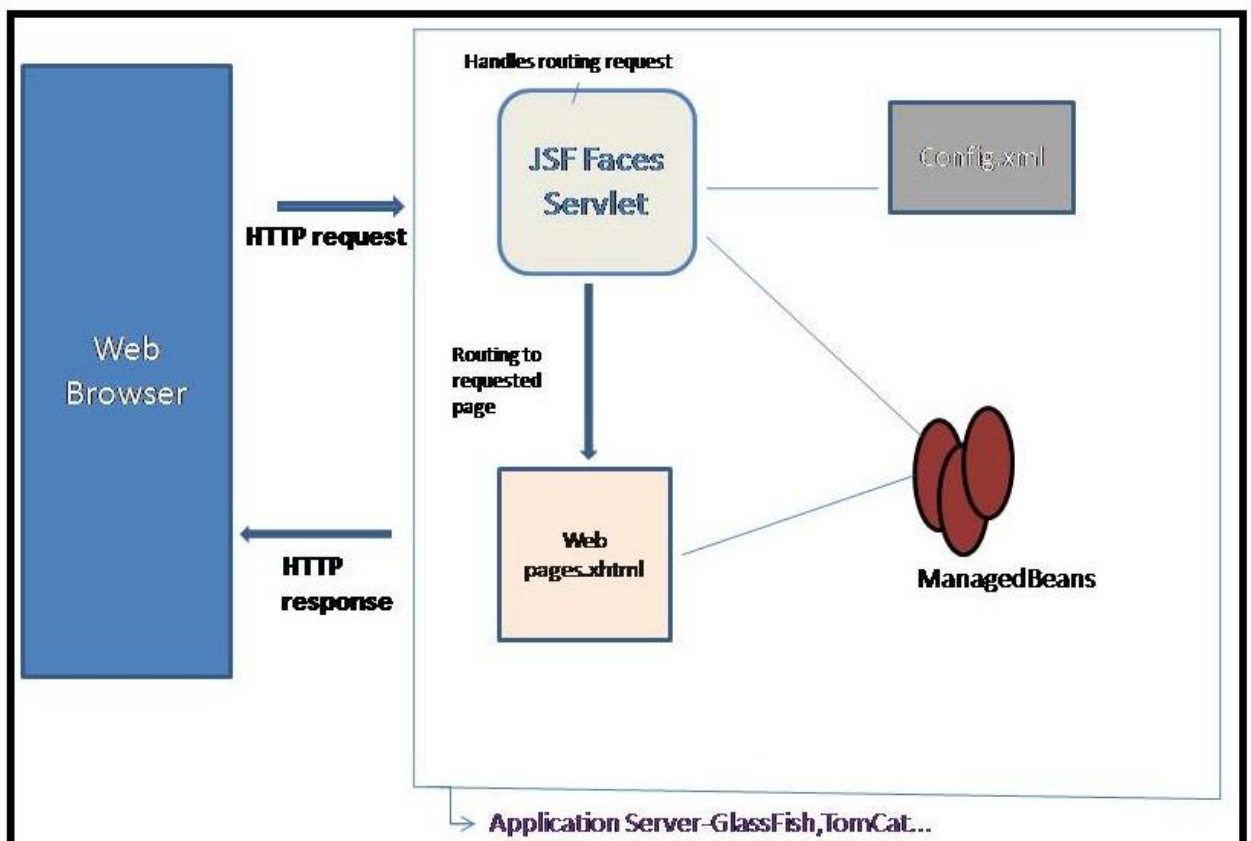


Figure 6 Application Architecture

Based on the required components and features, the overall application architecture was designed as shown in Figure 6. The architecture has a web browser at the client side which is used by both the student and the teachers for submitting and processing

the request of the enrolment. The users in the client side send their requests to the application via an HTTP request. The HTTP request is handled by the JSF Servlet. The Servlet then routes the request to the requested pages which are managed by the managed beans. Finally, the response is sent to the client as an HTTP response. All the servlets, managed beans and the pages reside in the application server which provides the runtime environment for the application.

### 3.3 Development Tools and System Development

After the application requirements had been analysed and the overall design was done the next step in the project development was the development environment set up. This was done in two phases: At first, development and deployment was done on a local machine and secondly, the same project was deployed in a public cloud service.

#### 3.3.1 Tools and Development Environment Set-up

For the project environment set up, different tools were used and programs had to be installed. The following programs were installed in order to carry out the development of the application on a local machine:

- Java Development Kit (JDK)
- NetBeans 7.3.1 IDE for Java EE Developers
- GlassFish Server
- JSF Framework
- JasperReports/ iReport Designer
- Derby Database.

After the development environment had been set up with installation of the required programs, a new web project for the application was created in NetBeans IDE as shown in the figure 7 and 8:

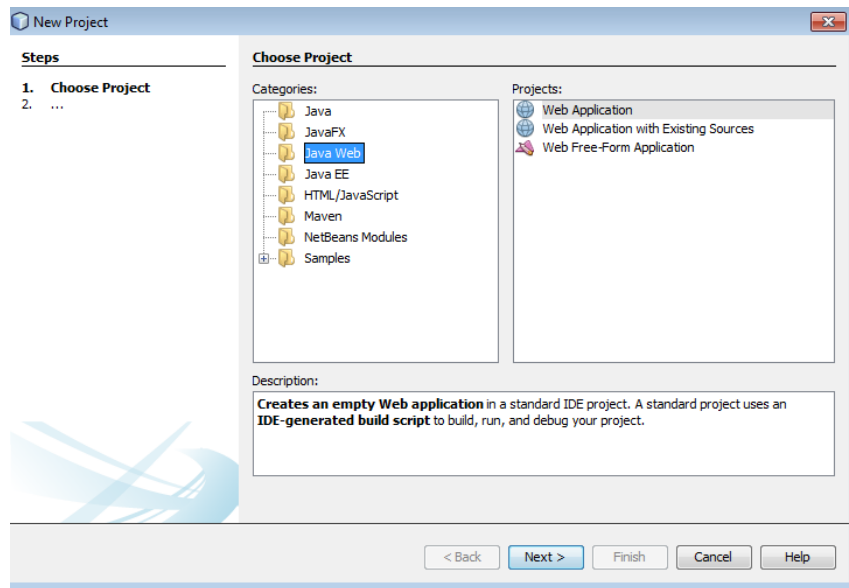


Figure 7 Creating a new JSF Application in NetBeans

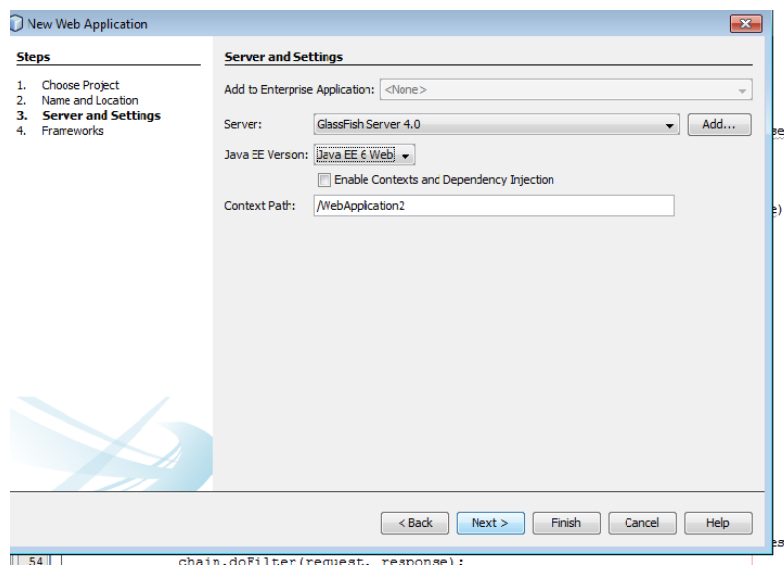


Figure 8 Configuring the Server and Settings of the new project.

The web project was built under the JSF Framework and the Glass Fish Server was used. Next, web pages were created with component tags. The application has two targeted user groups: students and teachers, so different web interfaces were created for both user groups having different features.



The web interface for students was kept inside *'/www'* folder so that the web page is easily accessible to all students. The interfaces for the admin/teachers were kept inside *'/securedPages'* folder for blocking unauthorized access.

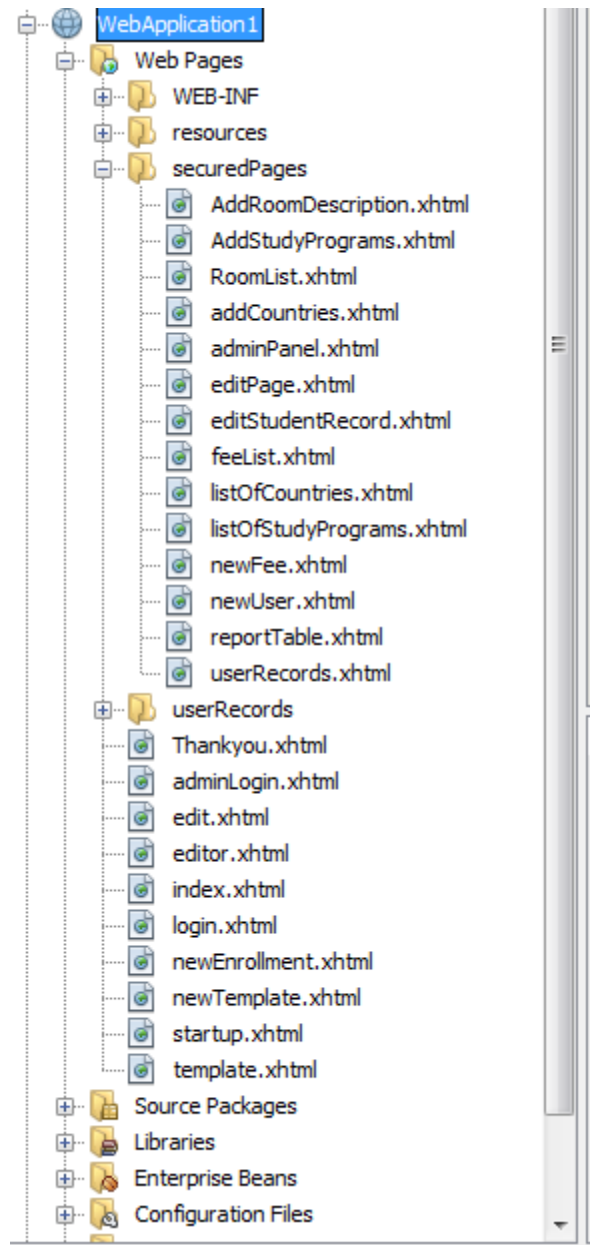


Figure 9 Folder structure of the application

After the project had been developed and deployed on the local machine, it was deployed in a cloud service provided by OpenShift Red Hat Cloud. For the purpose of cloud deployment different tools and environment had to be used. Eclipse for Java Enterprise Edition was used as IDE whereas MySQL was used as the database.

Similarly, Tomcat 7 was used as the application container. The development of the project was done in the cloud by using an open-source plugin for Eclipse IDE called OpenShift Eclipse plugin which was downloaded from Eclipse Marketplace. This tool uses JBoss tools to create a Maven application. Maven is a project management and comprehension tool. It describes how software is built and the dependencies of the project. The Maven application generates a `pom.xml` which contains the Project Object Model (POM) for the project. This file contains all the information about the project and a list of all the external dependencies the project needs. Maven dynamically downloads Java libraries from different repositories. The `pom.xml` of the project is listed in Appendix 2.

### 3.3.2 Web Filter

Since access to web interfaces for both the student and teacher can be done from a web browser, there had to be some mechanism in order to control unauthorized access to the admin web interfaces. Hence, in order to access the secured pages a **Servlet Filter** was used. The servlet filters are pluggable Java components that can be used to intercept and process requests before they are sent to servlets, and responses after the servlet code is finished and before the container sends the responses back to the clients[22].The Servlet Filter can be created by implementing `javax.servlet.Filter` interface. The filters can perform the following functions:

- Authentication : Blocking requests based on the user identity
- Logging and auditing : Tracking users of a web application
- Image conversation: Scaling maps
- Localization: Targeting the request and response to a particular locale.[21]

#### Implementation of a Filter

Filter API consists of three different interfaces: `Filter`, `FilterChain` and `FilterConfig`. The `javax.servlet.Filter` is the interface that was used to implement the filter. It has three methods:

- `void init(FilterConfig filterConfig)` :Called by the web container to indicate to a filter that is being placed into service.
- `void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)` :Called by the container each time a

request/response pair is passed through the chain due to a client request for a resource at the end of the chain.

- `void destroy()`: Called by the web container to indicate to a filter that is being taken out of service.

When the filter is created, the `init()` method is called by the container in which parameters are passed through the `FilterConfig` interface. The container calls `doFilter()` method for the service requests and the `destroy()` method is used to end the filter lifecycle.[10]

A Servlet Filter with annotation was used by implementing `javax.servlet.annotation.WebFilter`. By the use of this annotation, init parameters, filter name and description, servlets, url patterns and dispatcher types to apply the filter can be defined. Any changes to the filter configuration can be done by using the **web.xml** file.

```

<welcome-file-list>
  <welcome-file>faces/startup.xhtml</welcome-file>
</welcome-file-list>
<filter>
  <filter-name>UserFilter</filter-name>
  <filter-class>controller.UserFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>UserFilter</filter-name>
  <url-pattern>/securedPages/*</url-pattern>
</filter-mapping>

```

Listing 3 Implementation of servlet filter in `web.xml` file.

Listing 3 shows the actual implementation of servlet filter configuration in the `web.xml` file. The filter uses the `controller.UserFilter` to configure the web filter. The filter mapping restricts the filter to URLs that start with `/securedPages/`. Alternatively, it could have been done by the use of annotation in the filter class.

```

@WebFilter(filterName = "UserFilter1z", urlPatterns =
{"/faces/securedPages/*"})
public class UserFilter implements Filter {
    private static final boolean debug = true;
    private FilterConfig filterConfig = null;
    public UserFilter() {
    }
    private void doBeforeProcessing(ServletRequest request,
ServletResponse response)
        throws IOException, ServletException {
    }
}

```

#### Listing 4 Implementing filter in filter class

The listing 4 shows the implementation of the filter by the use of annotation in the filter class. The filter name specifies the name of the filter to be implemented and the `urlPatterns` refers to the urls requested by the client to be filtered for. Furthermore, a detailed view of the filter can be found in Appendix 1.

### 3.3.3 Database Tables and the View Controller

A database named *tourManager* was created with a number of tables for students and the admin, however all being handled by the administrator. The database was created using the **Derby Database** system provided by the Net Beans during the time of development in the local machine whereas **MySQL** database was used during development in the cloud. The Derby consists of a database engine and a JDBC driver. Applications use JDBC to interact with the database. The Derby driver class name is "org.apache.derby.jdbc.ClientDriver" and the driver class name for MySQL is "com.mysql.jdbc.Driver". In a Java application, initially the driver with a static `Class.forName` method is loaded [7]. The following tables were created under the database:

- `Participant` : holds the basic information about the students
- `Admin` : holds the admin username and password which is in due course of time used for authenticating the admin user login
- `ContactDetails` : contains the contact details of the student.

- StudyPrograms: contains the study programs a student is admitted to.
- Visa: holds the information about visa fee for different nations.
- Room: holds the information about the rooms like type of room and its virtual id.
- Course: holds a list of the names of the courses.
- Country: holds a list of the names of the countries.

The basic Entity Relationship Diagram (ERD) for the system is shown below in Figure 10 in a simple form. The tables-Participant, Course and Admin are the basic entities of the system whose consecutive attributes have been listed below along with other functional entities of the system. The figure 10 also shows the relationship between these entities.

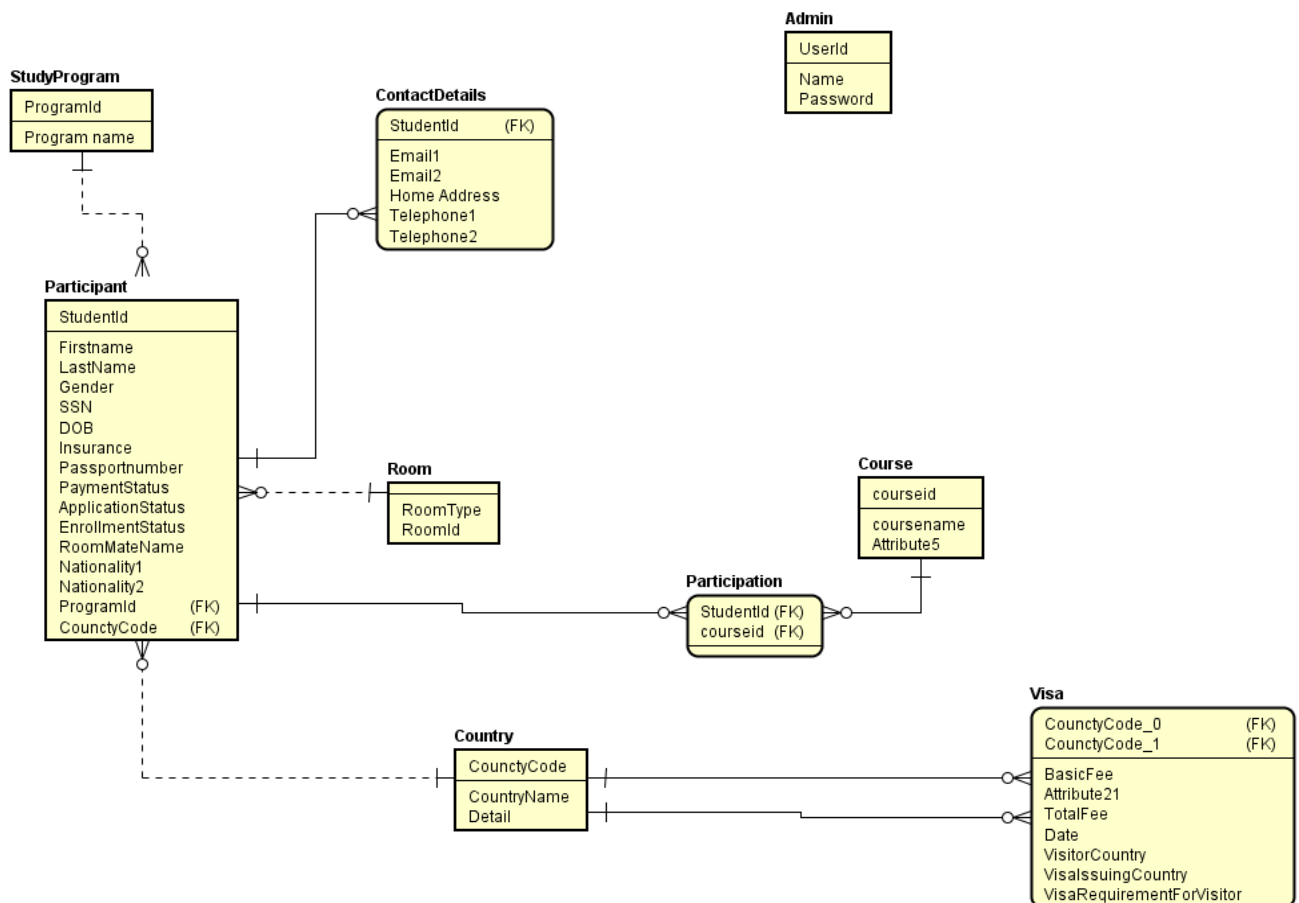


Figure 10 Basic ER-diagram of the system

The JSF views are controlled by backing beans, each of which is a JSF managed bean that is associated with the UI components used on a particular page. The backing bean supports the logic with the particular JSF view. The JSF views were made with specific

functionalities backed up by specific backing beans. The JSF views were created to perform following tasks:

- User Authentication for Admin users
- A web form for students to send enrolment requests
- Enrolling students directly by teacher for admin
- Updating student records
- Viewing records of students
- Creating reports of students.

In addition, the database-related activities are controlled by the bean classes specific to the database controller. The beans were developed for the following database activities:

- Add, edit and update student information
- Add records of student nationalities for visa processing
- Add, edit and update information about visa.

### 3.4 Deployment

The deployment of the project was done in two phases. At first it was developed and deployed on the local machine and then it was hosted using some public cloud so that it can be viewed by everyone.

#### Deployment in localhost

The application was developed using the NetBeans7.3.1 and Glass Fish Server. It was deployed using the Glass Fish Server 4.0. For the deployment in Glass Fish Server, however needs some basic requirement for deployment. In the local server, the application is available at <http://localhost:8080/WebApplication1/faces/in> local server. Depending upon the operating system being deployed the hardware and software requirement for the server differs. Since the deployment was done in Windows 7 Professional OS, the server had the minimum memory requirement of 1GB but 2GB is recommended. Likewise, the minimum hard disk space requirement is 250MB; however 500MB is recommended. Similarly, the GlassFish Server requires JDK version 7 or higher Java Virtual Machine version in both 32-and 64-bit of system. In addition, the installation of the server automatically detects the free ports. However,

three main ports commonly used for the system are 4848 for Administrative Console of Glassfish, 8080 for HTTP requests and 8081 for handling HTTPS.

#### Deployment in a public cloud

The deployment of the system over the web-server was done using OpenShift, which is a Linux-based public cloud application development and hosting service easily integrated with an Eclipse environment. This freeware is provided by OpenShift RedHat Online. A new OpenShift application was created using JBoss Tools OpenShift support which needed account credentials. Then, a new OpenShift domain was created, which is a unique namespace, and all the user application exists under the namespace. The domain name forms a part of the application url. Next, it rendered a view which required to upload the SSH keys to OpenShift so that OpenShift can perform Git operations and access the application gear remotely.

Next, the name of the key and the name of the private and public key file name was provided which directed to application creation wizard. The wizard needed details about the application. The application details include the name of the application, the type of the application, gear profile, scalability of the application, and option to embed cartridges like MySQL, PostgreSQL, MongoDB, phpMyAdmin and others. For the purpose of deployment, phpMyAdmin and MySQL were embedded as cartridges. Then server adapter settings configuration was done and the location to clone the git repository was specified. With this the application was configured and the application container was ready to deploy the application.

A private git repository was setup and cloned to the local system by OpenShift. After this, the project can be accessed with its DNS from any part of the world. The original project files were copied to this project location. The database credentials of the application were changed with the OpenShift database credentials. Similarly, the changes to the code were committed and pushed to the cloud. [24]

Openshift uses JBoss EWS 2.0 for Tomcat 7. The Java web applications created using OpenShift uses Maven to download dependencies and build/deploy. The dependencies of the application were added to the OpenShift Java web application by modifying the `pom.xml` and performing `git commit`. The required dependency can be found in the

Maven Central Repository. The dependencies are included inside the `<dependencies></dependencies>` of the `pom.xml` file. [25]

The project is created with the name `studentmgmt` and namespace `studentenrolment`. The project can be accessed on the server at the url <http://studentmgmt-studentenrolment.rhcloud.com>. Similarly, the database can be found at <http://studentmgmt-studentenrolment.rhcloud.com/phpmyadmin> which prompts for the database credentials generated by OpenShift during the project setup. The same credentials have been used to commit the database transactions of the project.



## 4 Results

The main objective of the project was to develop a web platform through which a student can request a teacher for enrolment for a course. This objective was met. Likewise, the other objective to enable the teacher who is also the admin of the system, to view, process and update the request was also fulfilled. Similarly, the other task to allow the admin to generate a report of the student data was also achieved.

**Metropolia**

**Application for Enrollment for Study Tour to St. Petersburg**

Student ID

First Name

Last Name

Study Program

Study Type

Primary Email

Alternate Email :

Gender :

Insurance(Vakuutus)

Nationality :

Nationality(other, if any)

Passport number

DOB

SSN

Telephone (home/personal)

Telephone (work/other)

Address(Home)/Contact address

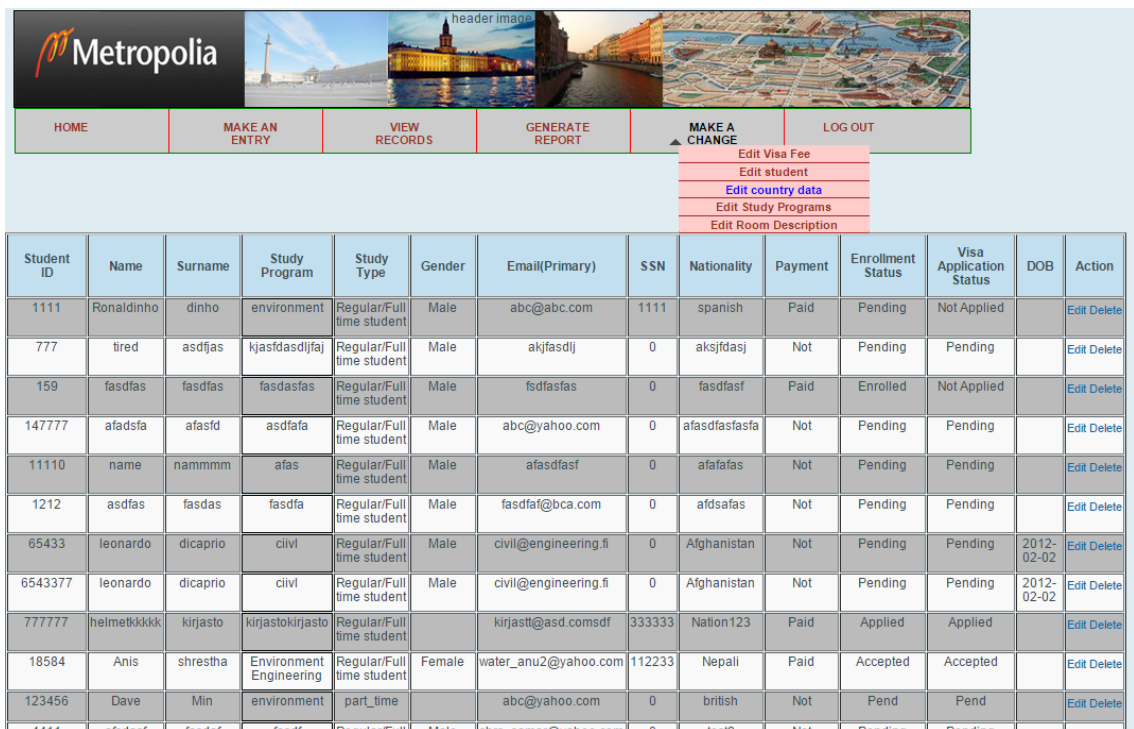
Address(Work)/Contact address

Desired friend to share room with

Figure 11 UI showing enrolment form for students

Figure 11 is a snapshot of the web interface where a student can make a request for enrolment for the course. The student has to fill in the form as in the illustration, and then submit it. The student has to provide the personal details, contact details, student ID and program, name of the program and other details.

After the form, shown in the Figure 11, is submitted by the student with the information required, the information is inserted into the database. The teacher can view the information of the student and then process the request after logging into the system. There is a mandatory login whenever the pages on the admin side are requested. Once the login is successful the admin has the access to those resources. The admin has to enter the username and password at the time of login. The login credentials are stored in the database. With a successful login, the teacher can have access to the resources shown in Figure 12 and 13.



The screenshot shows the Metropolia web interface. At the top, there is a navigation menu with buttons for HOME, MAKE AN ENTRY, VIEW RECORDS, GENERATE REPORT, MAKE A CHANGE, and LOG OUT. Below the menu, there is a dropdown menu for 'MAKE A CHANGE' with options: Edit Visa Fee, Edit student, Edit country data, Edit Study Programs, and Edit Room Description. The main content area displays a table of enrolment requests.

Student ID	Name	Surname	Study Program	Study Type	Gender	Email(Primary)	SSN	Nationality	Payment	Enrollment Status	Visa Application Status	DOB	Action
1111	Ronaldinho	dinho	environment	Regular/Full time student	Male	abc@abc.com	1111	spanish	Paid	Pending	Not Applied		<a href="#">Edit</a> <a href="#">Delete</a>
777	tired	asdftas	kjasdfasdjfaj	Regular/Full time student	Male	akjfasdlj	0	aksjfdasj	Not	Pending	Pending		<a href="#">Edit</a> <a href="#">Delete</a>
159	fasdfas	fasdfas	fasdfasfas	Regular/Full time student	Male	fsdfasfas	0	fasdfasfas	Paid	Enrolled	Not Applied		<a href="#">Edit</a> <a href="#">Delete</a>
147777	afadsfa	afasfd	asdfafa	Regular/Full time student	Male	abc@yahoo.com	0	afasdfasfasfa	Not	Pending	Pending		<a href="#">Edit</a> <a href="#">Delete</a>
11110	name	nammmm	afas	Regular/Full time student	Male	afasdfasf	0	afafafas	Not	Pending	Pending		<a href="#">Edit</a> <a href="#">Delete</a>
1212	asdfas	fasdfas	fasdfa	Regular/Full time student	Male	fasdfaf@bca.com	0	afdsafas	Not	Pending	Pending		<a href="#">Edit</a> <a href="#">Delete</a>
65433	leonardo	dicaprio	civil	Regular/Full time student	Male	civil@engineering.fi	0	Afghanistan	Not	Pending	Pending	2012-02-02	<a href="#">Edit</a> <a href="#">Delete</a>
6543377	leonardo	dicaprio	civil	Regular/Full time student	Male	civil@engineering.fi	0	Afghanistan	Not	Pending	Pending	2012-02-02	<a href="#">Edit</a> <a href="#">Delete</a>
777777	helmetkkkkk	kirjasto	kirjastokirjasto	Regular/Full time student		kirjastt@asd.comsd	333333	Nation123	Paid	Applied	Applied		<a href="#">Edit</a> <a href="#">Delete</a>
18584	Anis	shrestha	Environment Engineering	Regular/Full time student	Female	water_anu2@yahoo.com	112233	Nepali	Paid	Accepted	Accepted		<a href="#">Edit</a> <a href="#">Delete</a>
123456	Dave	Min	environment	part_time		abc@yahoo.com	0	british	Not	Pend	Pend		<a href="#">Edit</a> <a href="#">Delete</a>
4411	afdfasf	fasdfaf	fasdff	Regular/Full time student	Male	lehre_camar@yahoo.com	0	fact0	Not	Pending	Pending		<a href="#">Edit</a> <a href="#">Delete</a>

Figure 12 List of enrolment request

Figure 12 is a snapshot of the web interface where the admin can view the list of enrolment requests from students in tabular form. Furthermore, the table has options allowing the admin to update and delete the requests of the students. The **Edit** option allows the admin to update the request whereas the link next to it **Delete** allows to delete the request.

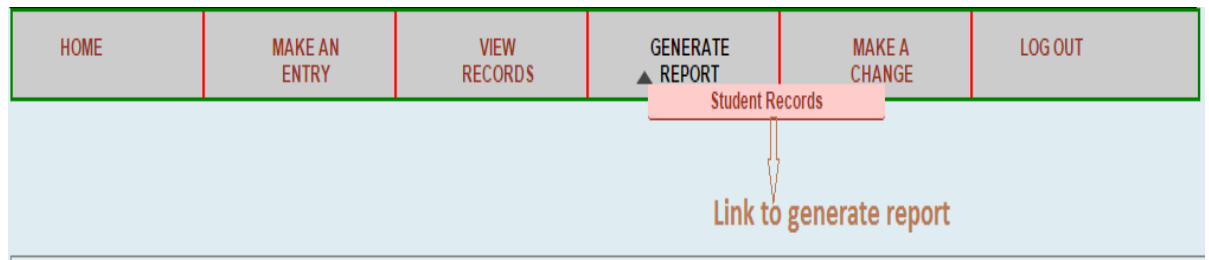


Figure 13 Link to generate report

Figure 13 is a snapshot of the interface which appears when the admin mouse pointer hovers over the Generate Report menu. This menu allows the admin to generate various reports of the students in a printable format. The reports are the templates designed in iReport Designer, then compiled such that whenever the admin requests for the report, the data of the students are compiled as per the design of the report. A demo of the report containing the name of students participating in the tour is shown in Figure 14. The report in the figure is generated in the **.pdf** format and it contains a list of students data received from the database.



**Metropolia**

**Helsinki Metropolia University of Applied Sciences**

November 06, 2015  
Page 1 of 1

**PARTICIPANTS OF STUDY TOUR TO ST.PETERSBURG**

S_No	Family Name	Family Name	D.o.B.	Nationality	Passport	Insurance
1	dinho	Ronaldinho	null	spanish	14725	null
2	asdfjas	tired	null	aksjfdasj	0	null
3	afasfd	afadsfa	null	afasdfasfasfa	0	null
4	nammmm	name	null	afafafas	0	null
5	fasdas	asdfas	null	afdsafas	0	null
6	dicaprio	leonardo	02/02/2012	Afghanistan	0	null
7	dicaprio	leonardo	02/02/2012	Afghanistan	0	null
8	fasdaf	afsdasf	null	test9	0	null
9	asdfa	fasdf	null	cad	0	null

Figure 14 A demo report of student generated by the system

Similarly, Figure 15 is a snapshot of the interface which asks for the admin credentials for authentication whenever the admin controlled resources are tried to be accessed. The resources illustrated in Figure 12-14 can only be accessed if this authentication is successful.

## 5 Evaluation

### 5.1 Benefits

Although the project was a simple web application based on Java platform, with the completion of the project the targeted goals were achieved. As the project has the web interface for submitting enrolment requests, one of the benefits of the project is that it makes the process of enrolling easier for students by just filling in an enrolment form and submitting it using the web interface. Likewise, it becomes easier for teachers to view the list of enrolment requests and process the requests. Furthermore, the teachers can easily edit and save the student data.

In addition, the project was an opportunity to become more familiar with my field of study. The project offered me a chance to use the theoretical studies in the practical implementation. With this practical implementation, I got chance to understand the various aspects of Java programming, various development tools used in programming and I had a close look at the real-world software development. Also, by carrying out the project I learned trouble shooting faced during the development of the project. Overall, the project helped me learn more about Java programming and move one step further in the field of programming.

### 5.2 Challenges

Although the project was completed and the targeted goals were met, there were several challenges and problems faced during the process of the project development. The first challenge was to use the right framework for the project since there are several frameworks in Java for web development. Since the project was simple and small, the JSF framework was used. Similarly, during the time of development there were several errors faced which took some time to be fixed and delayed the process of development. The solutions for most of the problems were found on the web portals and forums like [stackoverflow.com](https://stackoverflow.com), [coderanch.com](https://coderanch.com) and also [youtube.com](https://youtube.com) helped to solve several problems.

The theoretical studies did seem to be enough for the project hence it needed self-studies in different new topics within Java programming. Since, I did not have

knowledge about report generating in Java programming, I had to go through a number of tutorials about iReport Designer. There were many reporting tools available for the purpose of report designing, which caused some confusion. However learning about the iReport Designer tool seemed to be easy and useful for the project. I found that this tool was easy to integrate with NetBeans IDE. However, going through the tutorials and documentation and solving the errors occurring during the time of development took more time than expected, which somehow delayed the process of development.

Furthermore, several problems were faced during the time of hosting the application in the cloud. Though I had done web hosting for normal `php/html` based websites, I had not hosted a Java web application. So, the deployment in cloud took more time than expected as I had to go through documentations and tutorials of OpenShift. Moreover, some further problems occurred during the time of hosting as the development environment was different than that of localhost hosting in terms of IDE, servers and database. So, I had to make many changes to the original project so that it could run on a different server.

## 6 Conclusion

The main objectives of the project were to develop a web-based platform based on Java, enabling students to send enrolment requests for a course to the teacher and enabling the teacher to accept or reject the requests along with enabling the teacher to update the student information. Lastly, to enable the teacher to generate a report of the student information using the application was another requirement.

With the requirements focused, the project was carried out stepwise. First, the requirements were analysed and a study was carried out for the methods and technology to be used. Then the basic architecture was discussed several times and designed. Then for the development of the project, a development platform was chosen. Next, the web interfaces were designed and developed along with their back-ends. The database-related activities were also carried out side by side yet following the MVC architecture. At several points of the project development, a study was carried out on different technologies as per the requirements of the system. The application development progressed along with testing and changes were done wherever needed, thus resulting in the final product.

The final product hence achieved the targeted goals on the one hand, while on the other hand different ideas regarding software development and techniques in problem solving were learnt. The project provides an easy way for students and teachers for the process of enrolment. Likewise, it provides an easy way for the teachers to store the student data in an organised way.

## References

- 1 Oracle Corporation. Sun GlassFish Enterprise Server v3 Release Notes[online].2010  
URL:<https://docs.oracle.com/cd/E19226-01/820-7688/abpaj/index.html>.  
Accessed: 29 September 2015.
- 2 Beal Vangie. J2EE- Java 2 Platform Enterprise Edition [online].  
URL:<http://www.saint-petersburg.com/>. Accessed: 29 September 2015.
- 3 David Geary, Cay Horstmann. Core JavaServer Faces. January 2012. Prentice Hall, New Jersey, United States. Third Edition. Ch-1 Getting Started.p.4
- 4 Chris Schalk, Ed Burns, James Holmes. Java ServerFaces: TheComplete Reference.2007. The Mc Graw Hill Companies. United States. Ch-4 Managed Beans and the JSF Expression Language.p.53-54.
- 5 docs.oracle.com. JavaServer Faces Technology Benefits[online]. 2013.  
URL:<https://docs.oracle.com/javaee/6/tutorial/doc/bnapj.html>. Accessed: 30 September 2015.
- 6 Oracle Corporation. Introduction to JavaServer Faces 2.x[online]. 2015.  
URL:<https://netbeans.org/kb/docs/web/jsf20-intro.html>. Accessed: 30 September 2015.
- 7 db.apache.org. Derby JDBC Driver[online].  
URL:<http://db.apache.org/derby/docs/10.4/devguide/cdevdvlp40653.html>.  
Accessed: 1 October 2015.
- 8 docs.oacle.com. Managed Beans in JavaServer Faces Technology[online]. 2013.  
URL:<http://docs.oracle.com/javaee/6/tutorial/doc/bnaqm.html>. Accessed: 4 October 2015.
- 9 Tutorialspoint.com. JSF- Managed Bean[online]. 2015.  
URL:[www.tutorialspoint.com/jsf/jsf\\_managed\\_beans.htm](http://www.tutorialspoint.com/jsf/jsf_managed_beans.htm) Accessed: 9 October 2015.
- 10 Mukhar Kevin, Zelenak Chirs,Weaver James L. and Crume Jim. Beginning Java EE 5 Platform From Novice to Professional. Apress, 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705, United States. 2006. Ch-6 Servlets:p.281-282.
- 11 Oracle Corporation. The Client Tier -Your First Cup: An Introduction to the Java EE Platform [online].2010.

- URL:<https://docs.oracle.com/cd/E19798-01/821-1770/gcrla/index.html>.  
Accessed: 12 October 2015.
- 12 Oracle Corporation. The Web Tier -Your First Cup: An Introduction to the Java EE Platform [online].2010.  
URL:<https://docs.oracle.com/cd/E19798-01/821-1770/gcrla/index.html>.  
Accessed: 22 October 2015.
  - 13 Oracle Corporation. The Business Tier -Your First Cup: An Introduction to the Java EE Platform [online].2010.  
URL:<https://docs.oracle.com/cd/E19798-01/821-1770/gcrla/index.html>.  
Accessed: 15 October 2015.
  - 14 Oracle. Distributed Multitiered Applications [online].2010.  
URL:<http://docs.oracle.com/javaee/5/tutorial/doc/bnaay.html>. Accessed: 2 October 2015.
  - 15 careerride.com.J2EE Architecture [online].2008.  
URL:<http://www.careerride.com/J2EE-Architecture.aspx>. Accessed: 20 October 2015.
  - 16 Pawlan, Monica. Introduction to the J2EE Architecture [online].March 23,2001..  
URL:<http://pawlan.com/monica/articles/j2eeearch/>. Accessed: 16 October 2015.
  - 17 TIBCO Software Inc, Getting Started with JasperReports Library[online].2015..  
URL:<http://community.jaspersoft.com/wiki/getting-started-jasperreports-library>.  
Accessed: 19 October 2015.
  - 18 TIBCO Software Inc, JasperReports Library-Features and Highlights [online].2015.  
URL:<http://community.jaspersoft.com/wiki/getting-started-jasperreports-library-features-and-highlights>. Accessed: 21 October 2015.
  - 19 Toffoli, Giulio, JasperReports A design too iReport for JasperReports: Dcoumentation and tutorials [online].2015..  
URL:<http://ireport.sourceforge.net/manual0.2.0.html#1.1>. Accessed: 22 October 2015.
  - 20 Toffoli, Giulio, iReport Designer Getting Started[online].2015..  
URL:<http://community.jaspersoft.com/wiki/ireport-designer-getting-started>.  
Accessed: 22 October 2015.
  - 21 Integrated Cloud Applications & Platform Services. The Essentials of Filters [online].  
URL:<http://www.oracle.com/technetwork/java/filters-137243.html>. Accessed: 20 October 2015.



- 22 Pankaj, Journal Dev. Java Servlet Filter Example Tutorial[online].2015.  
URL:<http://www.journaldev.com/1933/java-servlet-filter-example-tutorial>.  
Accessed: 20 October 2015.
- 23 Oracle Corporation. Sun Java System Application Server Enterprise Edition 8.2 Administration Guide. Application Server Architecture [online].2015.  
URL:<https://docs.oracle.com/cd/E19900-01/819-4733/ablat/index.html>.  
Accessed: 22 October 2015.
- 24 Gulati Shekhar, OpenShift.com. Day 28: OpenShift Eclipse Integration for Java Developers [online].November 25, 2013.  
URL: <https://blog.openshift.com/day-28-openshift-eclipse-integration-for-java-developers/>. Accessed: 27November 2015.
- 25 OpenShift Developers, OpenShift.com. Tomcat Dependencies [online].  
URL:<https://developers.openshift.com/en/tomcat-dependencies.html>.  
Accessed: 27 November 2015.
- 26 Oracle Corporation. Sun Java System Application Server Enterprise Edition 8.2 Administration Guide. Application Server Architecture [online].2015.  
URL:<https://docs.oracle.com/javaee/6/tutorial/doc/gijtu.html> Accessed: 22 October 2015.

## Appendix 1: Implementation of Servlet Filter

```
package controller;

import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.io.StringWriter;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebFilter(filterName = "UserFilter1z", urlPatterns =
{"/faces/securedPages/*"})
@WebFilter(filterName = "UserFilter1z", urlPatterns =
{"/faces/securedPages/*"})
public class UserFilter implements Filter {
    private static final boolean debug = true;
    private FilterConfig filterConfig = null;
    public UserFilter() {
    }
    private void doBeforeProcessing(ServletRequest request,
ServletResponse response)
        throws IOException, ServletException {
    }
    private void doAfterProcessing(ServletRequest request,
ServletResponse response)
        throws IOException, ServletException {
    }
    @Override
    public void doFilter(ServletRequest request, ServletResponse
response,
```

```

        FilterChain chain)
        throws IOException, ServletException {
    System.out.println("Filter Initiated");
    HttpServletRequest req = (HttpServletRequest) request;
    HttpServletResponse resp = (HttpServletResponse) response;
    HttpSession session = (HttpSession) req.getSession(false);
    if (session==null || session.getAttribute("username") == null)
{
        resp.sendRedirect(req.getContextPath()
"/faces/login.xhtml");
        System.out.println("    context    path    =    "    +
req.getContextPath());
    } else {
        System.out.println(" request = " + request + "    and
response = " + response);
        chain.doFilter(request, response);
    }
}

public FilterConfig getFilterConfig() {
    return (this.filterConfig);
}

public void setFilterConfig(FilterConfig filterConfig) {
    this.filterConfig = filterConfig;
}

public void destroy() {
}

public void init(FilterConfig filterConfig) {
    this.filterConfig = filterConfig;
    if (filterConfig != null) {
    }
}

@Override
public String toString() {
    if (filterConfig == null) {
        return ("UserFilter()");
    }
    StringBuffer sb = new StringBuffer("UserFilter(");
    sb.append(filterConfig);
    sb.append(")");
}

```

```
        return (sb.toString());
    }
    private void log(String userFilterInitializing_filter) {
        throw new UnsupportedOperationException("Not supported yet.");
    }
    //To change body of generated methods, choose Tools | Templates.
}
```

## Appendix 2: Maven Project Object Model pom.xml

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001
/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>studentmgmt</groupId>
    <artifactId>studentmgmt</artifactId>
    <packaging>war</packaging>
    <version>1.0</version>
    <name>studentmgmt</name>
    <repositories>
        <repository>
            <id>eap</id>
            <url>http://maven.repository.redhat.com/techpreview/all</url>
            <releases>
                <enabled>>true</enabled>
            </releases>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
    </repositories>
    <pluginRepositories>
        <pluginRepository>
            <id>eap</id>
            <url>http://maven.repository.redhat.com/techpreview/all</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots> <enabled>true</enabled>
        </snapshots>
        </pluginRepository>
    </pluginRepositories>

    <properties>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <maven.compiler.source>1.6</maven.compiler.source>
        <maven.compiler.target>1.6</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>com.sun.faces</groupId>
            <artifactId>jsf-api</artifactId>
            <version>2.2.2</version>
        </dependency>
        <dependency>
            <groupId>com.sun.faces</groupId>
            <artifactId>jsf-impl</artifactId>
            <version>2.2.2</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>

```

```

<artifactId>javax.servlet-api</artifactId>
  <version>3.0.1</version>
</dependency>

<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>9.2-1003-jdbc4</version>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.25</version>
</dependency>
<dependency>
  <groupId>net.sf.jasperreports</groupId>
  <artifactId>jasperreports</artifactId>
  <version>6.1.0</version>
</dependency>
</dependencies>
<profiles>
  <profile>
<!-- When built in OpenShift the 'openshift' profile will be used when
      invoking mvn. -->
<!-- Use this profile for any OpenShift specific customization your app
      will need. -->
<!-- By default that is to put the resulting archive into the 'webapps'
      folder. -->
<!-- http://maven.apache.org/guides/mini/guide-building-for-different-
environments.html -->
<id>openshift</id>
  <build>
    <finalName>studentmgmt</finalName>
    <plugins>
      <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.1.1</version>
        <configuration>

          <outputDirectory>webapps</outputDirectory>

          <warName>ROOT</warName>

        </configuration>
      </plugin>
    </plugins>
  </build>
</profile>
</profiles>
</project>

```