**Employee management system**

**Purpose**

* Employee management system is a simple maven CRUD app for capturing and editing employee details. The employee management system uses hibernate as data access layer, JSF as UI framework and Spring in business layer. The main advantage of this combination is that there is very limited configuration through XML and by annotations. This allows developers to focus on the UI and he business layer. The application provides the following basic functionality

1. Create employee
2. Delete employee
3. Edit employee
4. View employee details

**Frameworks and tools used**

**Language**

* Java

**Service layer**

* Spring framework 5.0.8

**Persistence**

* Hibernate 5.0.7 Final
* PostgreSQL 42.1.2
* PgAdmin 4-3.6

**User Interface**

* JSF 2.2.18
* Bootfaces 1.3.0
* Primefaces 6.2
* Bootstrap 3.3.7

**Application server (servlet container)**

* Wildfly 10.0.0. Final

**Build tool (Dependency management)**

* Maven 3.5.4

**IDE**

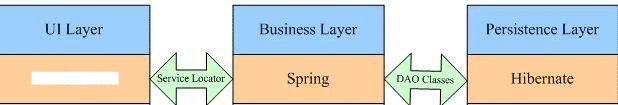
* NetBeans IDE 8.2(One can use any IDE which is compatible with maven or just a plain maven command line is fine. For this project is created a maven project and imported it into as a NetBeans project.

**Testing**

* Junit 4.12

**Project overview**

The following section provides the project overview and how all the components of the project fit together to form a solution.



* A maven project was created using the command line and imported to a NetBeans IDE.
* This maven project is based on a Project Object Model (POM) which is an XML file where all the project dependencies are outlined and resolved.
* After the declaration of maven dependencies, I ran a **Maven clean install** command on the command line to download all the required dependencies to my local machine.
* After downloading all the required dependencies, the following packages were created:

1. **Model**

This package consists of all entities that were extracted from the use case. This includes Person, Employee, Address and the BaseEntity which consist of common attributes.

1. **Common**

This package consists of common code that is shared across the entire project.

1. **Persistence**

The persistence package consists of a persistent service which performs CRUD operations on the database. The sole responsibility of this persistent service is to interact with the database. On this persistence service, hibernate template was used to access the database. Hibernate template is thread safe and reusable. You don’t have to open and close a session, hibernate template will do that for you.

1. **Service package**

This package contains spring services, more specifically an employee service. The business services are spring managed and wrapped into transactions managed by spring interceptors.

1. **UI package**

The UI package contains the managed beans which are regular java classes registered with JSF. These managed beans work as Model for UI component. Managed Bean can be accessed from JSF page.

1. **Test packages**

The test package contains all the unit tests created using Junit framework.

1. **Configuration package**

The configuration package stores configuration classes for the datasource config which traditionally could have been captured on the application context.

**Web pages**

**User interface layout**

For the user interface, JSF page templates were used. This allowed me to define static parts of the web page such as the header and the footer and define placeholders to insert non-static page content. Facelets are used a templating framework. Templates contains placeholders which we will push our page content into. This method of creating application views allowed me to create web pages that are consistent and easy to maintain since boilerplate content is limited to the actual template and reused everywhere so if there are changes we can change it in one place.

|  |
| --- |
| Header |
| Body |
| Footer |

The diagram above represents the structure and layout of a typical web page with templating applied. The header and the footer will only be loaded when then the user makes a request to have that page. Upon a menu click to another page, on the body gets replaced because of the application of templates.

**Project setup**

**The runtime environment for the project are**

1. Java 8
2. Wildfly 10.0
3. PostgreSQL 11 and PgAdmin IV

**Installation**

1. Install maven
2. Clone the repository from https://github.com/gershom12/IQ-Business-Assessment
3. Go to project repository via **cmd** and type “**mvn clean install**” to build the project and download all the required dependencies.

**Database setup**

1. Download and install **postgresl** and **PgAdmin IV**
2. Create a **postgres** database named “empdb” on PgAdmin
3. Make sure that the username and password on DataSourceUtilty.java matches the the credentials you used on the postgreSQL server created on PgAdmin

**How to run**

1. After the environment has been setup, compile the application by running **mvn clean install** on the command line.
2. Copy the war file from the target folder of the application and drop it into the **deployment folder** of your **wildfly** application server.
3. Once the **war** file has been pasted on the deployment folder, go to the bin folder of **wildfly** and start the application server by running the following command **./Standalone.sh**
4. Open your browser and go to ["http://localhost:8080/crudwebapp"](%22http://localhost:8080/crudwebapp%22), you should see a welcome page explaining what the application does and links on the menu section to navigate to the employee management section.

**The end**