highradius

Technical Documentation for
FinTech B2B Order Management Application

General Information

Direction Document functionality	FinTech B2B Order Management Application
Associated Documentation	PRS to B2B Order Management Application

Collaborators

Name	**Email**
HIghradius Dev Team	autonomous-dev@highradius.com

Reviewed By

Name	**Email**	**Date**

Document History

Version	**Reason for Change**	**Valid as of Date**
1.0	Initial Version	13th November, 2020

Prerequisites

Recommended Softwares Stacks:

1. Front-end Development:

- VS Code Editor

2. Back-end Development:

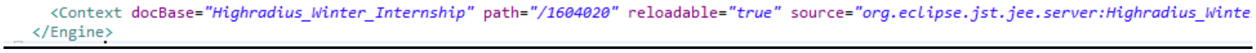
- Eclipse
- Apache Tomcat 8.5
- MySQL Server
- SQLYog Community Edition

Technology Stacks to be used:

2. Front-end Development:
   As per Developers choice (HTML, CSS,JS,BootStrap,ReactJs,JSP,Angular,etc) or any other framework of your choice
2. Back-end Development:
   Java

Project Cloning:

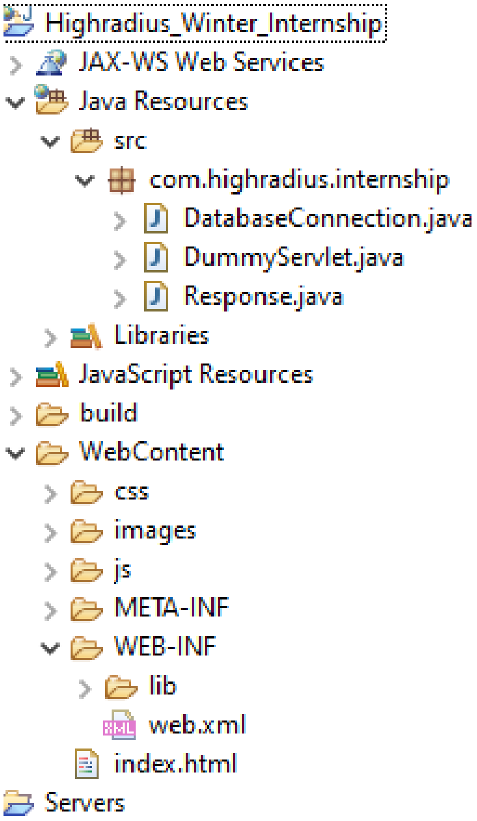
1. Download the must have Software Stack from here and watch this video to complete the installation.
2. Clone the Highradius Winter Internship Sample Project from here and follow the instructions present in README.md file to setup your workspace.
   You must use the cloned project to build your application. Self created projects won't be considered at any cost at the end of the Internship Period.
3. Application endpoint must always be your KIIT Roll Number.
   For Example: if your KIIT Roll Number is 1604020. Your Application endpoint must be: ***http://localhost:8080/1604020/***
   You can set it by changing the path property in the Context tag present inside server.xml in your Tomcat Directory.



```
    <Context docBase="Highradius_Winter_Internship" path="/1604020" reloadable="true" source="org.eclipse.jst.jee.server:Highradius_Winte
  </Engine>
```

Detailed Technical Direction:

Folder Structure:

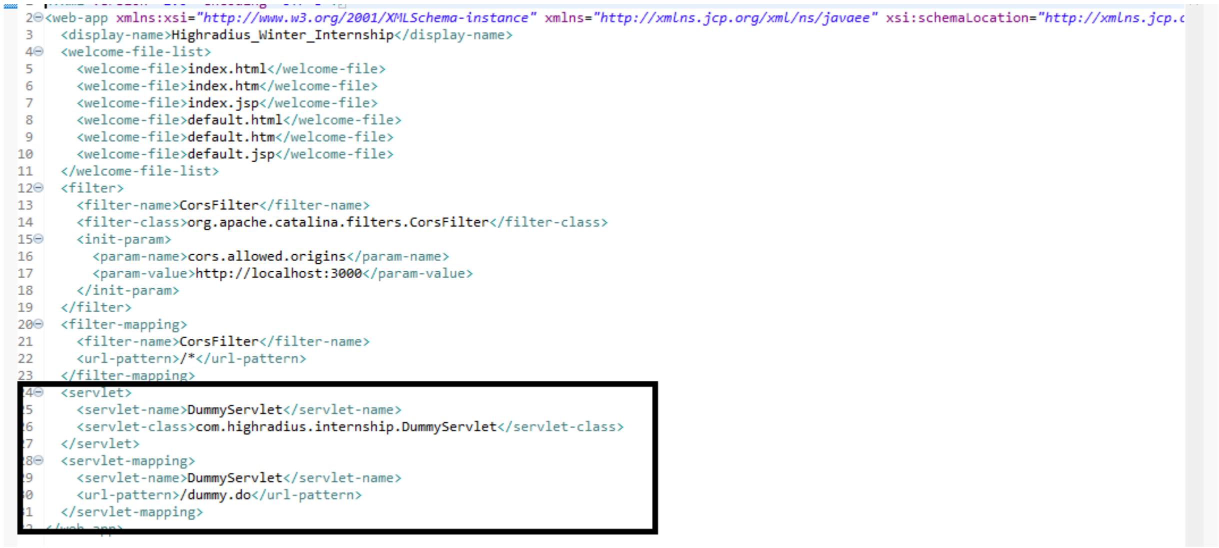


```
Highradius_Winter_Internship
> JAX-WS Web Services
v Java Resources
    v src
        v com.highradius.internship
            > DatabaseConnection.java
            > DummyServlet.java
            > Response.java
    > Libraries
> JavaScript Resources
> build
v WebContent
    > css
    > images
    > js
    > META-INF
    v WEB-INF
        > lib
          web.xml
      index.html
Servers
```

Create your Servlets and Pojo Classes inside the package ***com.highradius.internship*** . You can create your UI files inside WebContent Folder. If you are using ReactJs or any other equivalent technologies like Vui.js you can create a separate workspace for Front-end.

Creating your First Servlet:

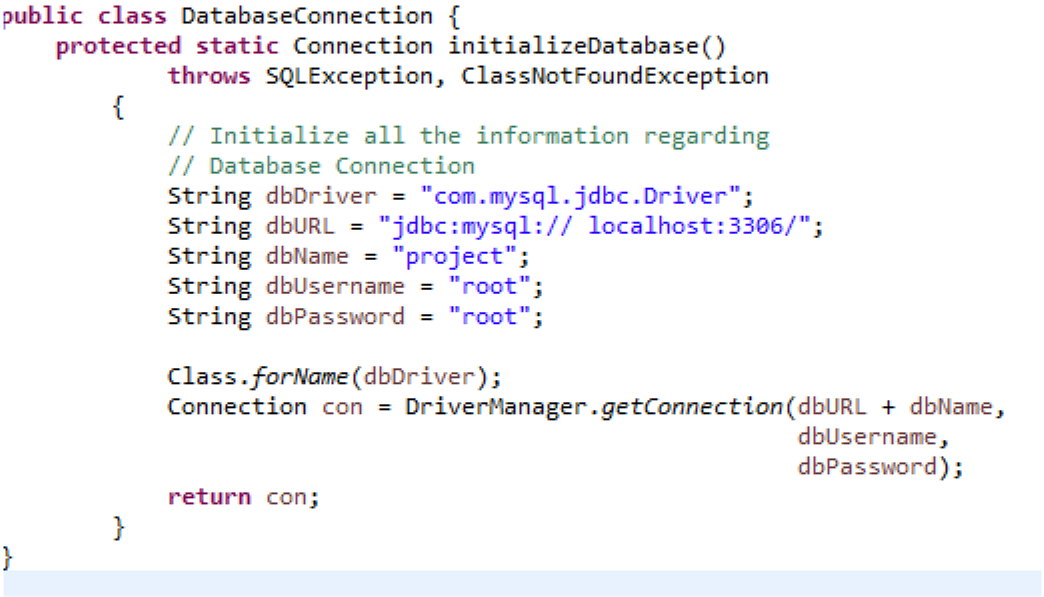
You can create a Servlet by creating a Java class and extending the *HttpServlet* Class. Override the doGet or doPost methods according to your need and use. You can find a sample Servlet by the name *DummyServlet.java* present in the project. Once the Class is created, map the servlet with a specific url in the Deployment Descriptor File (web.xml). You can find a sample mapping in the web.xml file.



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.c
  <display-name>Highradius_Winter_Internship</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <filter>
    <filter-name>CorsFilter</filter-name>
    <filter-class>org.apache.catalina.filters.CorsFilter</filter-class>
    <init-param>
      <param-name>cors.allowed.origins</param-name>
      <param-value>http://localhost:3000</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>CorsFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <servlet>
    <servlet-name>DummyServlet</servlet-name>
    <servlet-class>com.highradius.internship.DummyServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>DummyServlet</servlet-name>
    <url-pattern>/dummy.do</url-pattern>
  </servlet-mapping>
</web-app>
```

Connecting your Java Application to your Database:

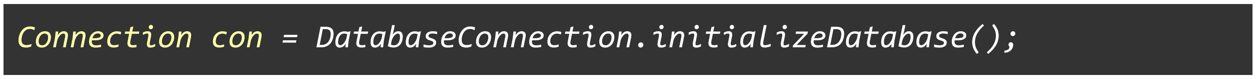
We will use JDBC to connect to the Database. You take the help of *DatabaseConnection.java* class to establish connection.



```
public class DatabaseConnection {
    protected static Connection initializeDatabase()
            throws SQLException, ClassNotFoundException
        {
            // Initialize all the information regarding
            // Database Connection
            String dbDriver = "com.mysql.jdbc.Driver";
            String dbURL = "jdbc:mysql:// localhost:3306/";
            String dbName = "project";
            String dbUsername = "root";
            String dbPassword = "root";

            Class.forName(dbDriver);
            Connection con = DriverManager.getConnection(dbURL + dbName,
                                                         dbUsername,
                                                         dbPassword);
            return con;
        }
}
```

Enter the dbName,dbUsername and dbPassword in the java file and use the method.



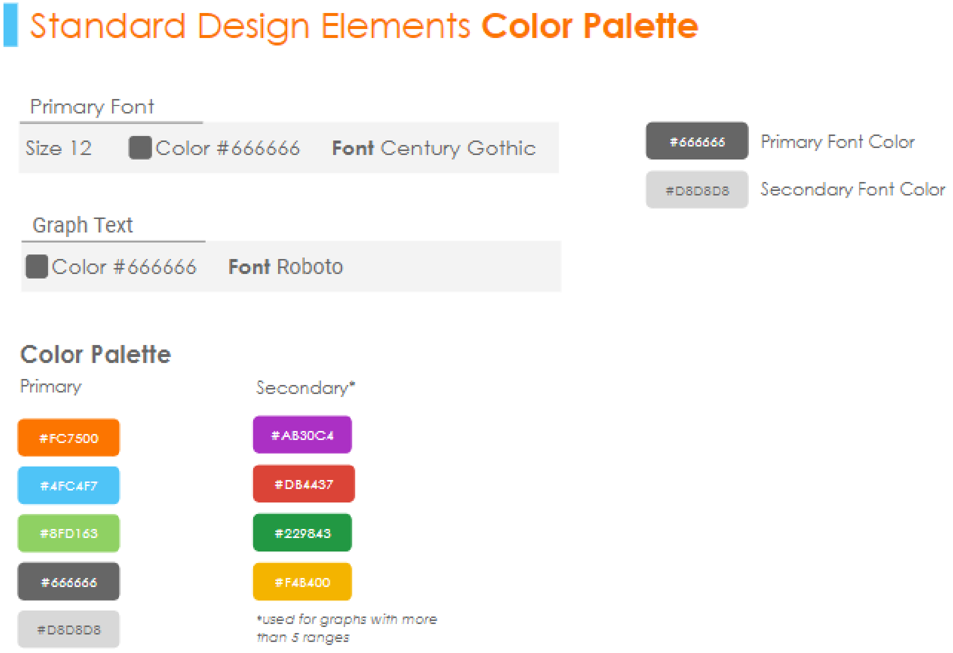
```
Connection con = DatabaseConnection.initializeDatabase();
```

Actions to be performed via Servlet Call:

1) Login Validation.
2) Data Loading in Grid with pagination.

3) Search Functionality.
4) Add Order Functionality.
5) Edit Order Functionality.
6) Approve/Reject Order Functionality.

Color Codes for Front-end Development:



Standard Design Elements **Color Palette**

Primary Font

Size 12 ■ Color #666666 **Font** Century Gothic

Graph Text

■ Color #666666 **Font** Roboto

#666666	Primary Font Color
#D8D8D8	Secondary Font Color

Color Palette

Primary	Secondary*
#FC7500	#AB30C4
#4FC4F7	#DB4437
#8FD163	#229843
#666666	#F4B400
#D8D8D8	

*used for graphs with more than 5 ranges

References(Links to Videos,Playbooks,Docs):

1.