Project Preparation Final

University Database Web App

This is a very simple and straight-forward application which simply displays list of students enrolled and their information such as ID, First Name, Last Name, email id, etc. It consists of features such as to add a new student, to update and existing student and to delete a particular student from the record all together. This application again is very elementary and I developed it as an assignment project during my internship at Persistent Systems to hone my skills and for getting a better grasp on core java concepts and understand how to integrate frontend with the backend.

This application is developed using Java Server Pages(JSPs), servlet and Java Database Connectivity (JDBC). It follows MVC approach which is a software design pattern consisting of a MODEL, VIEW and a CONTROLLER.

Now, both JSP and Servlets are server side technologies which processes the request on the server and returns a response accordingly. A JSP file is basically an HTML Page with some embedded java code in it, which is essentially used for adding dynamic content inside our page. For example, let’s say we have a web browser and it sends a request to our server for some data, now on our server we have created a particular Java Server page i.e., JSP to deal with such requests. So what will happen is, that JSP page will get processed on the server, now by processing it means the java code embedded inside the JSP will get processed and the output of that will be added to the HTML page and sent back as a response back to the browser. So this is how JSP works. In fact it serves as the VIEW in MVC approach for software design.

Now coming to servlet, it is a java class which extends the functionality of a web server. It generates dynamic content and interacts with the client through REQUEST and RESPONSE objects. In MVC architecture servlets works as a controller and handles all the business logic of our web application.

Now as I have talked about two of the terms in MVC model i.e., VIEW and CONTROLLER. All that remains now is to talk about MODEL, What is a model ?

Now model is as I understand a java class which is called upon by the controller when it need some additional information to or data to process the request sent by the client. Now, model itself does not contain any data but it fetches the data from an external resource, maybe it can be an api or from some database.

Now, one of the most essential part of any application is connectivity to the database. In this application I have used TOMCAT as my server to run the application on and I have used MySql for database. Now, as I mentioned before that I did this project to clear my concepts and to better understand on how to integrate backend with frontend. It really helped me a lot in that aspect.

Now talking about how I connected TOMCAT with my database,

Whenever we need to connect tot a database using a web application, we always think that we only need one database connection. But this approach to connect to the database is not scalable for multiple users because one database connection can only be used by a single user at a time.

This is analogous to having only one telephone inside an office consisting of 100s of people waiting in a queue for their turn on the telephone. To avoid this best practice is to use database connection pools, which will essentially allow our app to scale and handle multiple users.

Now to set up connection pool we need to have JDBC Driver jar file of the database we are using (In my case MySql) and context.xml file which will contain the name of the database to connect to, and other information such as the URL to connect to the database, no. of connections upfront, no. of idle connections, etc.

After adding these two files to our database we need to setup connection pool which is done by using resource injection. Using this, the tomcat will automatically set the connection pool in our servlet file.

For resource injection we use special annotation,

@Resource(name=”jdbc/web\_student\_tracker”)

Private DataSource dataSource;

dataSource variable will have a connection pool to our database which we can use to setup connections to our database.

Covid-19 Alert (Spring Boot)

This Project is about covid-19 alert. In this project I created a spring boot microservice which basically process the data about the covid-19 cases and according to that data it labels a particular state with red, orange or green zone. Red considered to be Highly contagious, orange to be moderately contagious and green to be a safe zone. Apart from the labels, it also tells us security measures to be taken according to our zone. Another feature of this application is that it can give us the overall summary of the covid breakout throughout the country, like, no. of total cases, total no. of casualties, total recovered, etc.

We’ll have three main classes :

AlertController, AlertService and CovidDataProvider. The AlertController class will be having two endpoints i.e., (/india/summary) and (/india/{state}). First endpoint will give the overall summary regarding the corona virus cases in India at a particular time, while second endpoint will take a state name as an input and will be providing the alert label for that particular state which will be Red, orange or green based on the no. of cases in that particular state at a given time.

Now this AlertController class will be calling upon the AlertService class which contains the business logic to classify and label red, orange or green zones. Now this AlertController class will be having two functions corresponding to each of the endpoints in the AlertController class, i.e., getSummary() and getStateAlert(state).

Now, all the label classification and summary is done based on some data, and for that data, AlertService class calls upon the CovidDataProvider class to provide the necessary data to complete the request by the client. Furthermore, the CovidDataProvider class will not be having that exact particular data but it will contact an external api source to fetch the data. There are many external rest apis which are providing real-time data for covid-19 cases in India. But for this project I have used data from a website called api.rootnet.in. This is the same api that is used on the website of Ministry of wealth and Family Welfare of India, so it quite reliable.

Helper class (Model) StudentDbUtil.java

Class StudentDbUtil {

Private DataSource dataSource;

Public StudentDbUtil(DataSource d) {

This.dataSource = d;

Myconn conn = dataSource.getConnection();

Stmt myStmt = conn.createStatement();

String sql = “select \* from student”;

Res myRes = myStmt.executeQuery(sql);

While(myRes.next()) {

String name = myRes.getString(1);

}

}

}

StudentController.java

// connection pool

@Resource(name=”jdbc/web-student-tracker”)

Private DataSource dataSource;

Init() {

StudentDbUtil obj1 = new StudentDbUtil(dataSource);

}

Do get(Req, res) {

listStudent(req, res);

}

Public listStudent(req, res) {

List<Student> studentList = obj1.getStudent();

}

Class StateData {

String loc;

Int confirmedCasesIndian;

Int confirmedCasesForeign;

}

Class CovidApiData {

Boolean success;

DateTimeZone lastRefreshed;

DateTimeZone lastOriginUpdate;

}

Class covidData {

Summary summary;

Regional regional;

}

String URL=”xjhhjb”

CovidApiData covidDataApi = restTemplate.getForObject(URL, class.CovidDataClass);

@Resource(name=”nameofthedb”)

Private DataSource dataSource;

// controller file (StudentController) servlet

@Resource(name=”jdbc/student\_tracker”)

Private DataSource dataSource;

doGet(httpRequest res, httpResp res) {

listStudent(req, res);

}

Public void listStudent(req, res) {

List<Student> s\_list = getStudent();

}

}

// model class StudentDbUtil

{

DataSource dataSource;

Public StudentDbUtil(DataSource dataSource) {

This.dataSource = dataSource;

}

Public static List<Student> getStudent() {

Connection myConn=null;

Statement myStmt = null;

ResultSet myRes;

myConn = dataSource.getConnection();

myStmt = myConn.createStatement();

//

String sql = “Select \* from student”;

myRes = myStmt.executeQuery(sql);

List<Student> studentList ;

while(myRes.next()) {

int id = myRes.getInt(1);

String name = myRes.getString(2);

.

.

studentList.add(new Student(id, name, roll))

}

}

}

//

String URL = [www.xyz.com](http://www.xyz.com);

CovidApiData covidApiData = restTemplate.getForObject(URL, CovidApiData);

Public class CovidApiData {

Boolean success;

Data data;

DateTimeZone lastRefreshed;

DateTimeZone lastoriginalUpdate;

}

Public class Data {

Summary summary;

StateData data;

}