SangamOne
Connected Services

Dynamic Web Projects using Java Spring Boot:

Objective:

To display customer data on the local web browser using Java Spring Boot, Tomcat Apache, Maven, MySQL database and Web Programming tools (HTML, CSS, JavaScript, JSP) – eg :http://localhost:8080/pincode

Pre-Requisites:

SN	Tool	Remarks
1	MySQL Workbench, MySQL Server	For Database
2	Spring Tool Suite (STS)	For Java Programming
3	Notepad, Sublime Text or VS Code	For HTML and JSP web pages

Step1: New -> Spring Boot Starter Project -> Add the following Dependency

1.1) Spring Boot Devtools
1.2) MySQL Driver
1.3) Spring Data JPA
1.4) Spring Data JDBC
1.5) Spring Web Service

After that click on finish.

Step 2: Check pom.xml pages (Project → target → pom.xml)

After that we should add the Tomcat Jasper and JSTL dependency in pom.xml pages.

Browser → MVN Repository → Tomcat jasper dependency(as per our version)

```
<!-- https://mvnrepository.com/artifact/org.apache.tomcat/tomcat-jasper -->

<dependency>

   <groupId>org.apache.tomcat</groupId>

   <artifactId>tomcat-jasper</artifactId>

   <version>9.0.53</version>

</dependency>
```

Browser → MVN Repository → jstl (latest version)

```
<!-- https://mvnrepository.com/artifact/jstl/jstl -->
<dependency>
   <groupId>jstl</groupId>
   <artifactId>jstl</artifactId>
   <version>1.2</version>
</dependency>
```

Step3: application. properties

Here we give our following details.

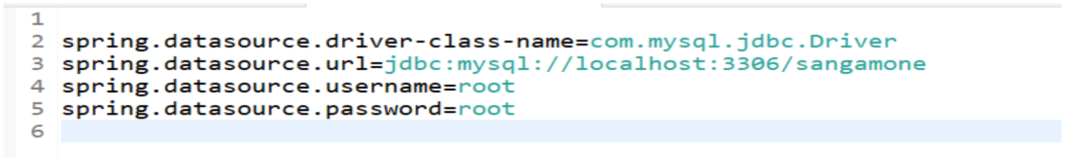
3.1) We should initialize the JDBC-Driver (already we added the dependency)

3.2) We should provide the database connection details such localhost and database name.

3.3) We should provide our database username.

3.4) We should provide our database password also.

Finally, our application. properties like this



```

spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/sangamone
spring.datasource.username=root
spring.datasource.password=root

```

Step 4: Create the JSP Page.

Project → webapp → JSP File → Give JSP page Name → click finish → JSP page will generate inside the webapp (Eg: XXXXX.jsp).

4.1) First just give the heading inside that <h2> tag.

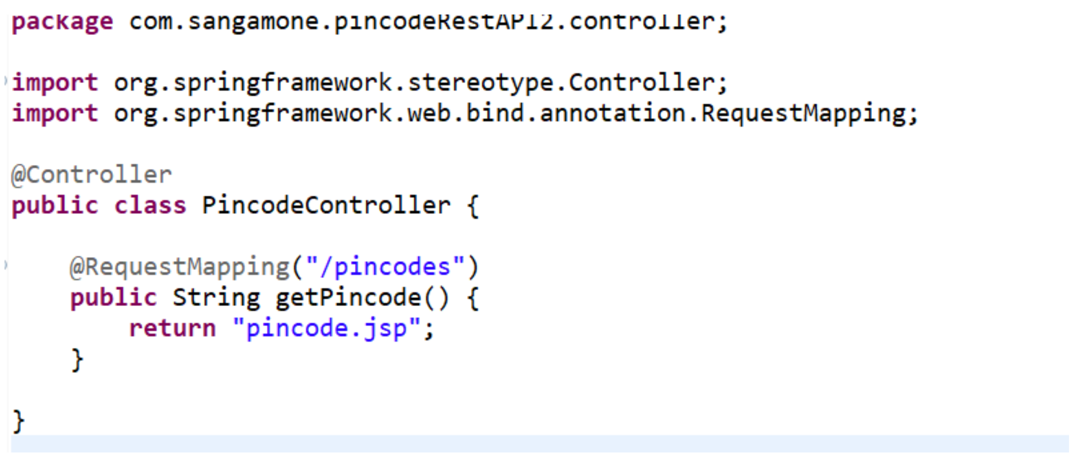
Step 5: Create the controller package and controller class. Name it "PincodeController.class".

5.1) add @Controller above the controller class

5.2) add @RequestMapping inside the controller class and create the one method and call the "pincode.jsp" page in return function. After that give the @RequestMapping("/pincode").

5.3) Check the Project. And Start Run(Right click on the project) → Run as → spring boot App.

Now, out controller class i.e. "PincodeController.class" look like this



```
package com.sangamone.pincodeRestAPI12.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class PincodeController {

    @RequestMapping("/pincodes")
    public String getPincode() {
        return "pincode.jsp";
    }

}
```

For checking: localhost//pincodes

Step 6: Most important creating the database in the MYSQL-Workbench.

6.1) Create the database.

6.2) Create the table and filed name according to our CSV file.

6.3) Database (Right Click) → Import → Browse the file → Click to Import → Finish

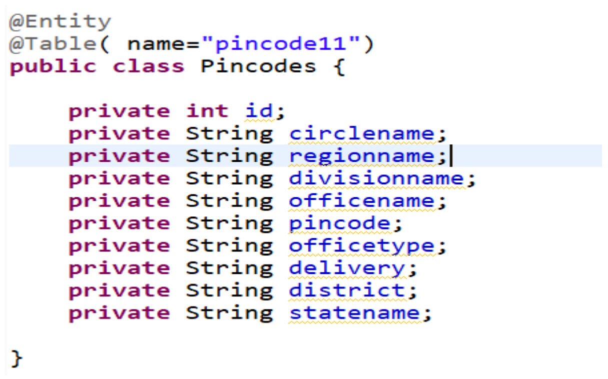
Step 7: Most important to create the Model class. Here we should specify our entity, table name, Database Id and "generated value".

7.1) Create the model package and inside that model package create the model class name it as "Pincodes.java".

7.2) Annotate the @Entity(above the model class)

7.3) Annotate the @Table(name = "tablename") .It should be a below the @Entity.

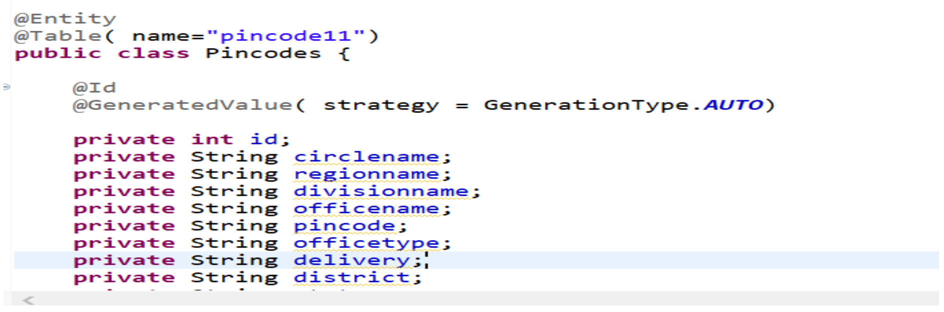
7.4) Cretae the datatype with values,



```
@Entity
@Table( name="pincode11")
public class Pincodes {

    private int id;
    private String circlename;
    private String regionname;
    private String divisionname;
    private String officename;
    private String pincode;
    private String officetype;
    private String delivery;
    private String district;
    private String statename;

}
```

7.5) Create the @Id inside that model class. i.e Pincodes.java

7.6) Create the @GeneratedValue (strategy = GenerationType.AUTO) inside that Model Class.



```
@Entity
@Table( name="pincode11")
public class Pincodes {

    @Id
    @GeneratedValue( strategy = GenerationType.AUTO)

    private int id;
    private String circlename;
    private String regionname;
    private String divisionname;
    private String officename;
    private String pincode;
    private String officetype;
    private String delivery;
    private String district;
```

7.7) Add the constructor Right Click → Source → Generate Constructor Using Fields → SelectAll → Click Generate.

7.8) Add one more Empty Constructor. Right Click → Source → Generate constructor using super class Fields → SelectAll → Click Generate.

7.9) Create the getter & setter's methods for all the attributes.

Right Click → Source → Generate getters and setters → SelectAll → Click Generate

7.10) Create the toString() method

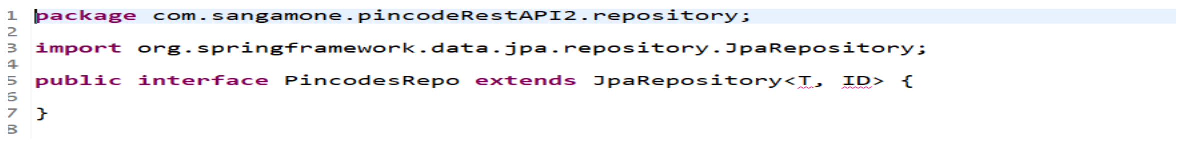
Right Click → Source → Generate toString()... → SelectAll → Click Generate.

Step 8: Create the repository package and create the interface inside the repo packages.

8.1) Annotate the @Repository above the interface class.

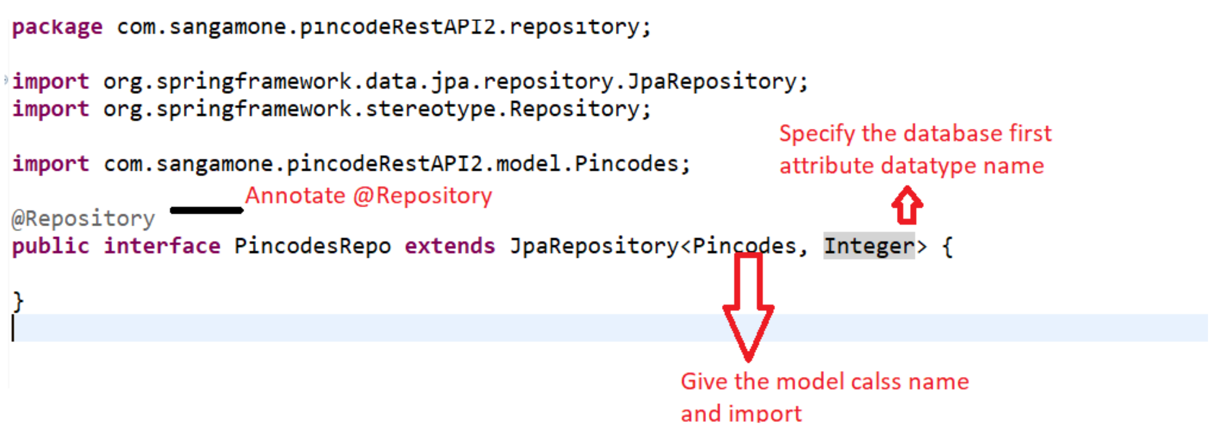
8.2) Add the Model Class name and database First entity datatype.

It will open like this,



```
package com.sangamone.pincodeRestAPI2.repository;

import org.springframework.data.jpa.repository.JpaRepository;

public interface PincodesRepo extends JpaRepository<T, ID> {

}
```

After that change the <T,ID>



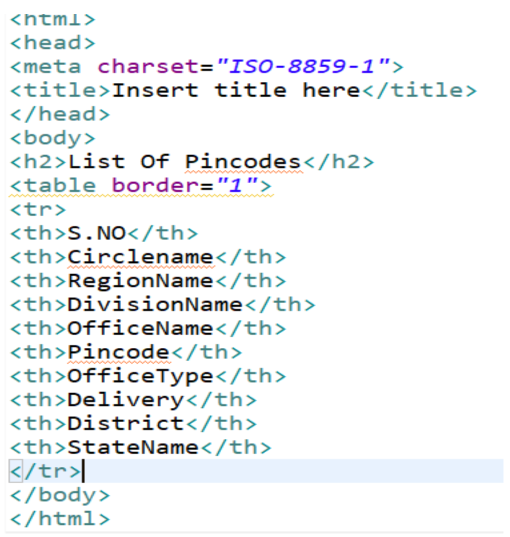
```
package com.sangamone.pincodeRestAPI2.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.sangamone.pincodeRestAPI2.model.Pincodes;

@Repository
public interface PincodesRepo extends JpaRepository<Pincodes, Integer> {

}
```

Specify the database first attribute datatype name
Annotate @Repository
Give the model calss name and import

Step 9: Change the view page. Inside the we create the table format

Model is entity class and view is a jsp page.

9.1) Add the <table> format. Specify the table row<tr> and table heading <th>

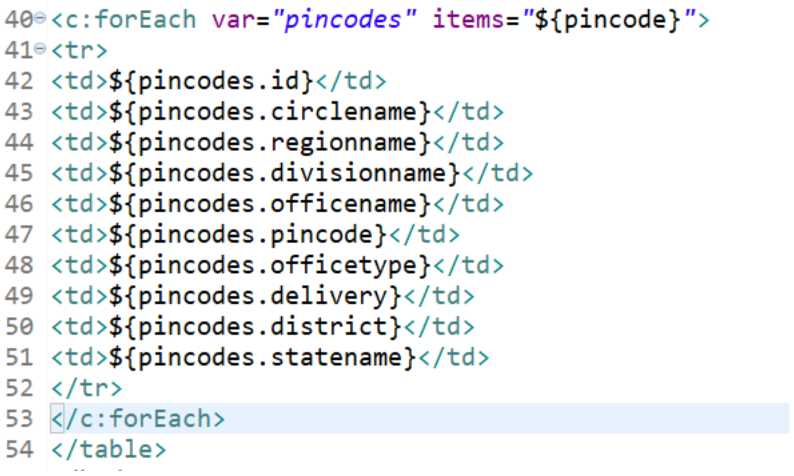


```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>List Of Pincodes</h2>
<table border="1">
<tr>
<th>S.NO</th>
<th>Circlename</th>
<th>RegionName</th>
<th>DivisionName</th>
<th>OfficeName</th>
<th>Pincode</th>
<th>OfficeType</th>
<th>Delivery</th>
<th>District</th>
<th>StateName</th>
</tr>
</body>
</html>
```

9.2) Add the following tag inside the jsp page. It should be a above the head tag.

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

9.3) After that create <c:forEach var="anyname" items="${anyname}"></c:forEach>

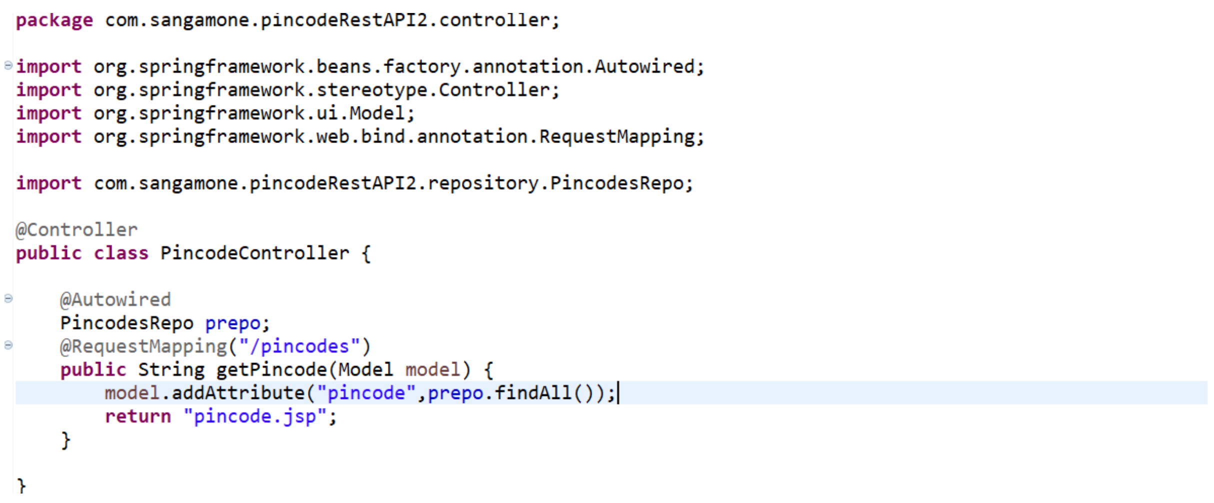


```
<c:forEach var="pincodes" items="${pincode}">
<tr>
<td>${pincodes.id}</td>
<td>${pincodes.circlename}</td>
<td>${pincodes.regionname}</td>
<td>${pincodes.divisionname}</td>
<td>${pincodes.officename}</td>
<td>${pincodes.pincode}</td>
<td>${pincodes.officetype}</td>
<td>${pincodes.delivery}</td>
<td>${pincodes.district}</td>
<td>${pincodes.statename}</td>
</tr>
</c:forEach>
</table>
```

Step 10: Finally edit our controller class. i.e "PincodeController.java".

10.1) Add @Autowired annotate.

10.2) Create the object for repository interface. Finally our controller class look like this



```
package com.sangamone.pincodeRestAPI2.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import com.sangamone.pincodeRestAPI2.repository.PincodesRepo;

@Controller
public class PincodeController {

    @Autowired
    PincodesRepo prepo;
    @RequestMapping("/pincodes")
    public String getPincode(Model model) {
        model.addAttribute("pincode",prepo.findAll());
        return "pincode.jsp";
    }

}
```

Finally, our pom.xml looks like this,

```
<?xml version="1.0" encoding="UTF-8"?>
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.4</version>
        <relativePath/> <!-- lookup parent from repository -->

        </parent>
        <groupId>com.sangamone</groupId>
        <artifactId>SampleProgram-6-RESTFULServices</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>war</packaging>
        <name>PincodeRestAPI1-3</name>
        <description>Demo2</description>
        <properties>
               <java.version>1.8</java.version>
        </properties>
        <dependencies>
        <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jdbc</artifactId>
        </dependency>
        <dependency>

<groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web-services</artifactId>
        </dependency>
        <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
        </dependency>
        <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
        </dependency>
        <dependency>



<groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
        </dependency>
        <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.tomcat/tomcat-jasper -->
<dependency>

<groupId>org.apache.tomcat</groupId>
        <artifactId>tomcat-jasper</artifactId>
        <version>9.0.53</version>
        </dependency>

<!-- https://mvnrepository.com/artifact/jstl/jstl -->
<dependency>
<groupId>jstl</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin
</plugins>
</build>
</project>
```

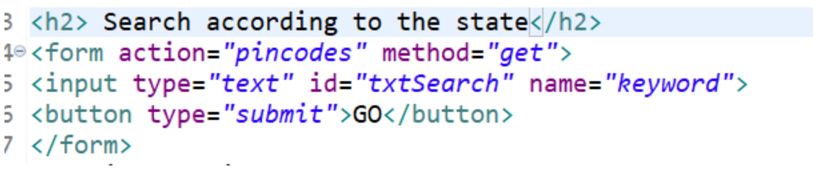
Appendix1 – Implementing the "SearchOption" i.e how to find the particular values.

For implementing the Search option we should modify the following changes.

Note: We need not change anything in the model class.

Step1.1: Goto the JSP page then add the one text filed and one button. After that just run program and check the http:/localhost:8080/pincodes.

Now our pincode.jsp page consist



```
<h2> Search according to the state</h2>
<form action="pincodes" method="get">
<input type="text" id="txtSearch" name="keyword">
<button type="submit">GO</button>
</form>
```

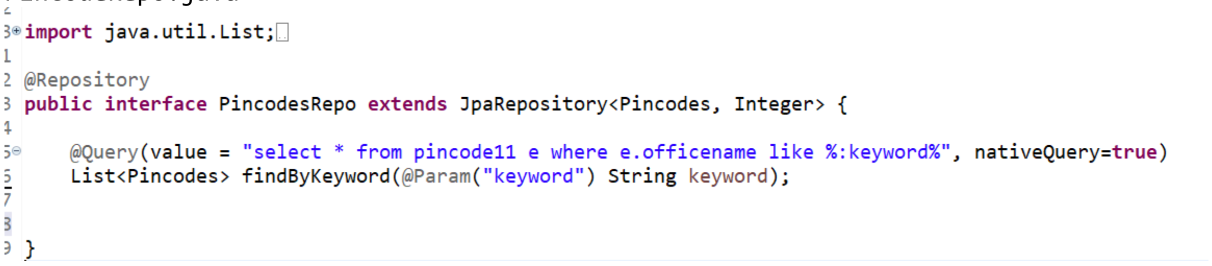
Step 1.2: Now go to the repository package. ("PincodeRepo.java")

Now, in repository package we initialize which values want to search, according to that we should list queries.

```
        List<Pincodes> findByKeyword(@Param("keyword") String keyword);
```

```
    a) Import the List(util packages)
After that we should write the queries. It used to fetch the values from the database. Now we our repo class have following function.
```

PincodeRepo.java

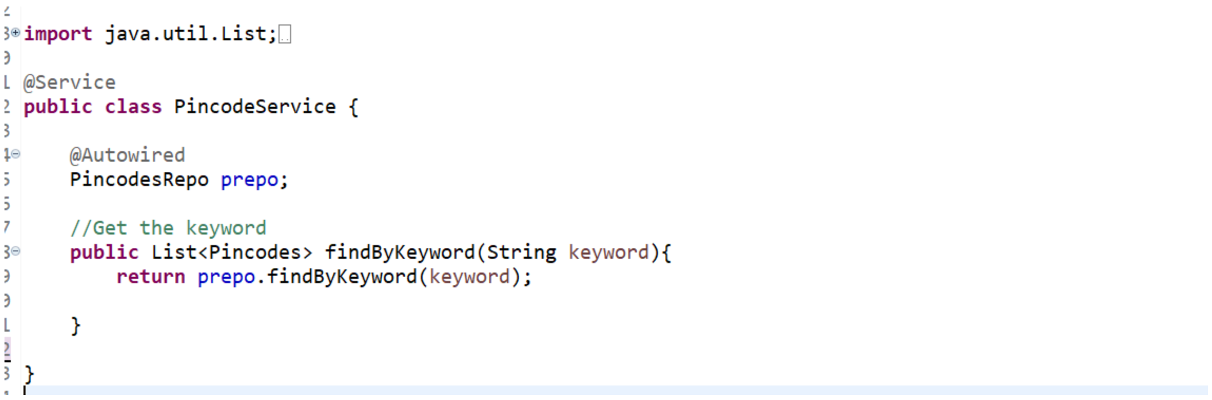


```java
import java.util.List;

@Repository
public interface PincodesRepo extends JpaRepository<Pincodes, Integer> {

    @Query(value = "select * from pincode11 e where e.officename like %:keyword%", nativeQuery=true)
    List<Pincodes> findByKeyword(@Param("keyword") String keyword);


}
```

Step1.3: Now we need to add one more Service Package.
1.3.1) Create the Service package and create the class like "PincodeService.java".
1.3.2) Inside that we implement our method.
1.3.3) Here we are implementing from repository class so that we should add the annotate to the repository class.

```
Ex:
        @Autowired
        PincodeRepo pRepo;
```

1.3.4) After that for searching the values always prefere the "findByKeyword"(findBy is normal format for spring)
1.3.5) Now our service class consists follows.



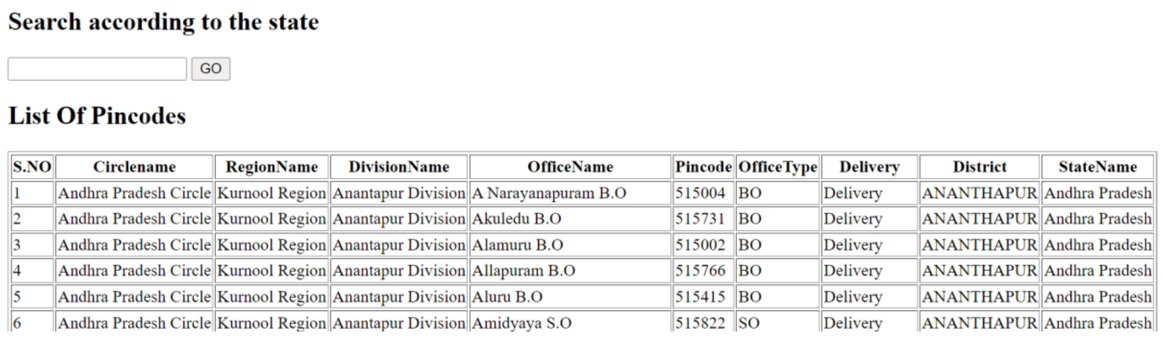
```java
import java.util.List;

@Service
public class PincodeService {

    @Autowired
    PincodesRepo prepo;

    //Get the keyword
    public List<Pincodes> findByKeyword(String keyword){
        return prepo.findByKeyword(keyword);

    }

}
```

Step 1.4: Now we modify our controller class.

1.4.1) Inside the method we should out keyword with datatype.
1.4.2) Autowired to service class

```
@Controller
public class PincodeController {

    @Autowired
    PincodesRepo prepo;

    @Autowired
    PincodeService pservice;

    @RequestMapping("/pincodes")
    public String getPincode(Model model, String keyword) {
        if(keyword != null) {
            model.addAttribute("pincode",pservice.findByKeyword(keyword));

        }
        else {
        model.addAttribute("pincode",prepo.findAll());
        }

        return "pincode.jsp";
    }
```

Finally run the program and check it in the localhost.
http://localhost:8080/pincodes



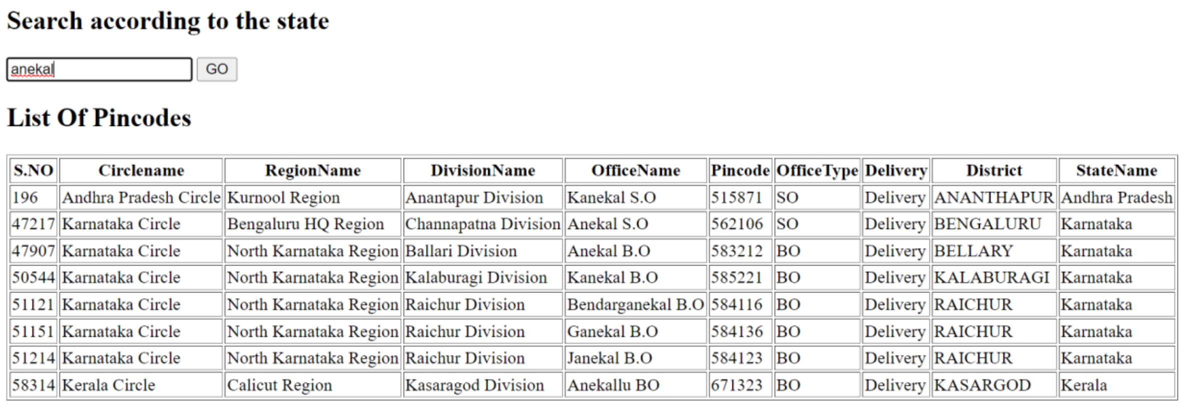
Search according to the state

GO

List Of Pincodes

S.NO	Circlename	RegionName	DivisionName	OfficeName	Pincode	OfficeType	Delivery	District	StateName
1	Andhra Pradesh Circle	Kurnool Region	Anantapur Division	A Narayanapuram B.O	515004	BO	Delivery	ANANTHAPUR	Andhra Pradesh
2	Andhra Pradesh Circle	Kurnool Region	Anantapur Division	Akuledu B.O	515731	BO	Delivery	ANANTHAPUR	Andhra Pradesh
3	Andhra Pradesh Circle	Kurnool Region	Anantapur Division	Alamuru B.O	515002	BO	Delivery	ANANTHAPUR	Andhra Pradesh
4	Andhra Pradesh Circle	Kurnool Region	Anantapur Division	Allapuram B.O	515766	BO	Delivery	ANANTHAPUR	Andhra Pradesh
5	Andhra Pradesh Circle	Kurnool Region	Anantapur Division	Aluru B.O	515415	BO	Delivery	ANANTHAPUR	Andhra Pradesh
6	Andhra Pradesh Circle	Kurnool Region	Anantapur Division	Amidyaya S.O	515822	SO	Delivery	ANANTHAPUR	Andhra Pradesh

Search the element according to the place.



Search according to the state

anekal GO

List Of Pincodes

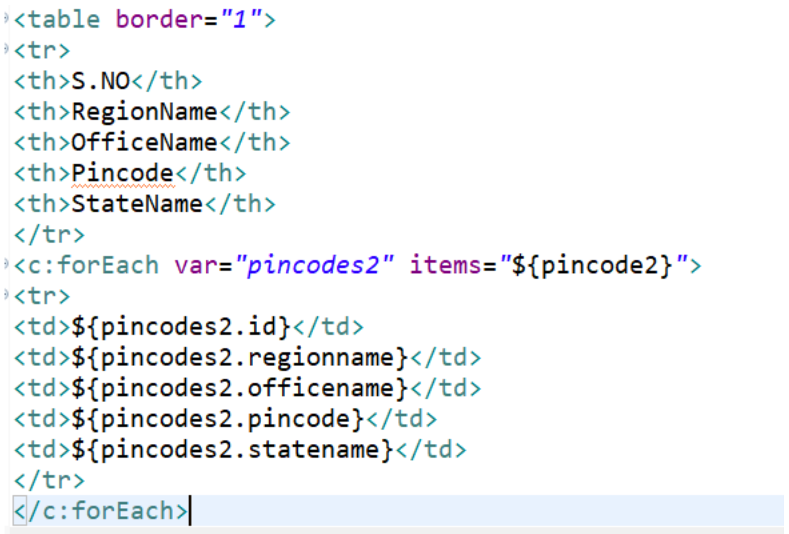
S.NO	Circlename	RegionName	DivisionName	OfficeName	Pincode	OfficeType	Delivery	District	StateName
196	Andhra Pradesh Circle	Kurnool Region	Anantapur Division	Kanekal S.O	515871	SO	Delivery	ANANTHAPUR	Andhra Pradesh
47217	Karnataka Circle	Bengaluru HQ Region	Channapatna Division	Anekal S.O	562106	SO	Delivery	BENGALURU	Karnataka
47907	Karnataka Circle	North Karnataka Region	Ballari Division	Anekal B.O	583212	BO	Delivery	BELLARY	Karnataka
50544	Karnataka Circle	North Karnataka Region	Kalaburagi Division	Kanekal B.O	585221	BO	Delivery	KALABURAGI	Karnataka
51121	Karnataka Circle	North Karnataka Region	Raichur Division	Bendarganekal B.O	584116	BO	Delivery	RAICHUR	Karnataka
51151	Karnataka Circle	North Karnataka Region	Raichur Division	Ganekal B.O	584136	BO	Delivery	RAICHUR	Karnataka
51214	Karnataka Circle	North Karnataka Region	Raichur Division	Janekal B.O	584123	BO	Delivery	RAICHUR	Karnataka
58314	Kerala Circle	Calicut Region	Kasaragod Division	Anekallu BO	671323	BO	Delivery	KASARGOD	Kerala

Appendix 2: How filter the selected columns using JSP Page

Step 2.1: Here we are changing the view page.
2.1.1) Create the one more jsp page . “Pincodes2.jsp”
2.1.2) Create the table.
2.1.3) After that add following function in the jsp page.

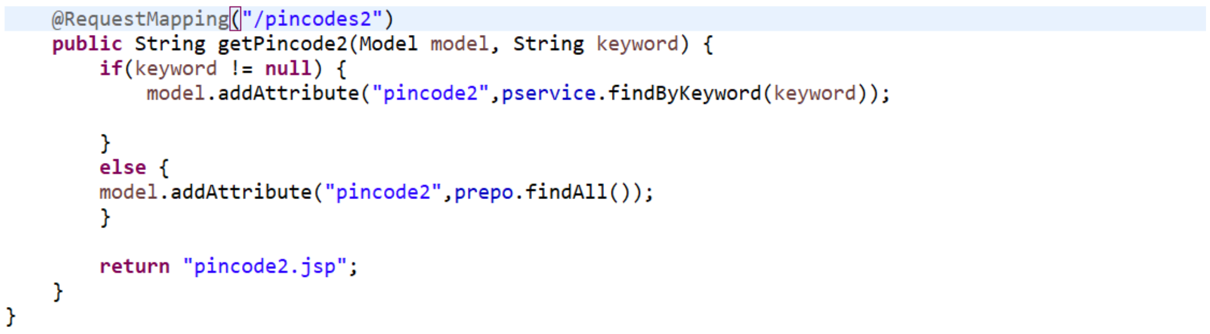
```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

2.1.4) Create the <c:forEach> tag, inside that fetch columns which we have required.
 Here we are going to fetch the only five columns from the database.



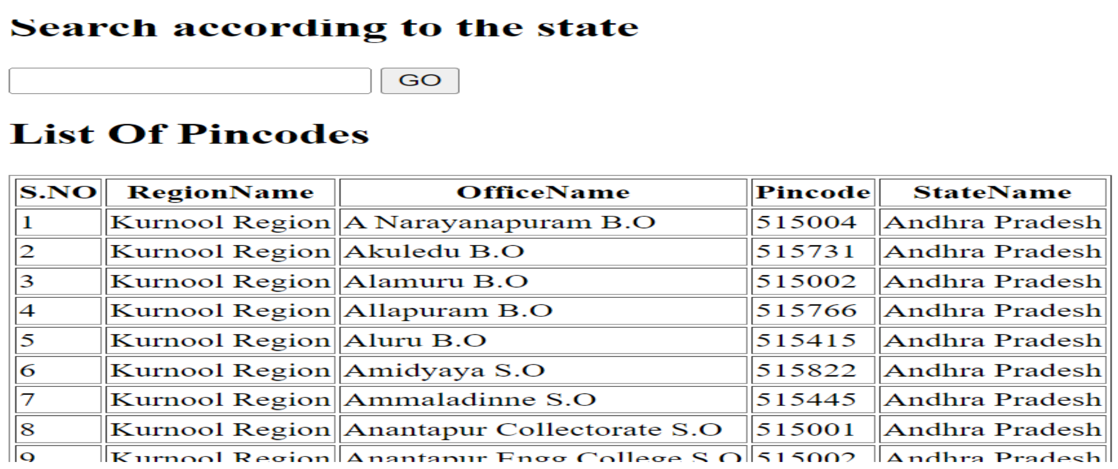
```
<table border="1">
<tr>
<th>S.NO</th>
<th>RegionName</th>
<th>OfficeName</th>
<th>Pincode</th>
<th>StateName</th>
</tr>
<c:forEach var="pincodes2" items="${pincode2}">
<tr>
<td>${pincodes2.id}</td>
<td>${pincodes2.regionname}</td>
<td>${pincodes2.officename}</td>
<td>${pincodes2.pincode}</td>
<td>${pincodes2.statename}</td>
</tr>
</c:forEach>
```

Step 2.2: After that we should modify our controller class.
2.2.1) Create the one more method.
2.2.2) Give the annotate method.



```
    @RequestMapping("/pincodes2")
    public String getPincode2(Model model, String keyword) {
        if(keyword != null) {
            model.addAttribute("pincode2",pservice.findByKeyword(keyword));

        }
        else {
        model.addAttribute("pincode2",prepo.findAll());
        }

        return "pincode2.jsp";
    }
}
```

Output:



Search according to the state

GO

List Of Pincodes

S.NO	RegionName	OfficeName	Pincode	StateName
1	Kurnool Region	A Narayanapuram B.O	515004	Andhra Pradesh
2	Kurnool Region	Akuledu B.O	515731	Andhra Pradesh
3	Kurnool Region	Alamuru B.O	515002	Andhra Pradesh
4	Kurnool Region	Allapuram B.O	515766	Andhra Pradesh
5	Kurnool Region	Aluru B.O	515415	Andhra Pradesh
6	Kurnool Region	Amidyaya S.O	515822	Andhra Pradesh
7	Kurnool Region	Ammaladinne S.O	515445	Andhra Pradesh
8	Kurnool Region	Anantapur Collectorate S.O	515001	Andhra Pradesh
9	Kurnool Region	Anantapur Engg College S.O	515002	Andhra Pradesh

Appendix 3: Creating the REST API

Common Procedure:

1) Initializing the a Spring Boot Starter Project.
2) Goto the create MYSQL-Workbench and create the table and insert the values in to the table.
3) Then create the Pincode model class.
4) Create the Repository for Pincode Interface.
5) Create the Controller class

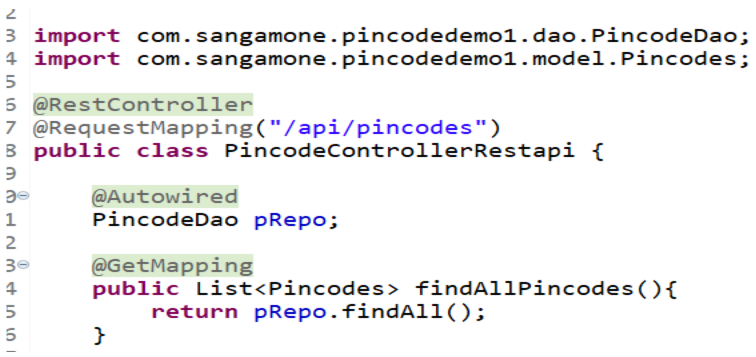
Step 5.1: Only one controller is enough for creating the REST API because already we created the model class and interface class.

5.1.1) Add the @RestController annotation in above the controller class.

5.1.2) Add the @RequestMapping annotation .

5.1.3) Add the @GetMapping Annotation.

5.1.4) Create the one method . Now our controller class is,

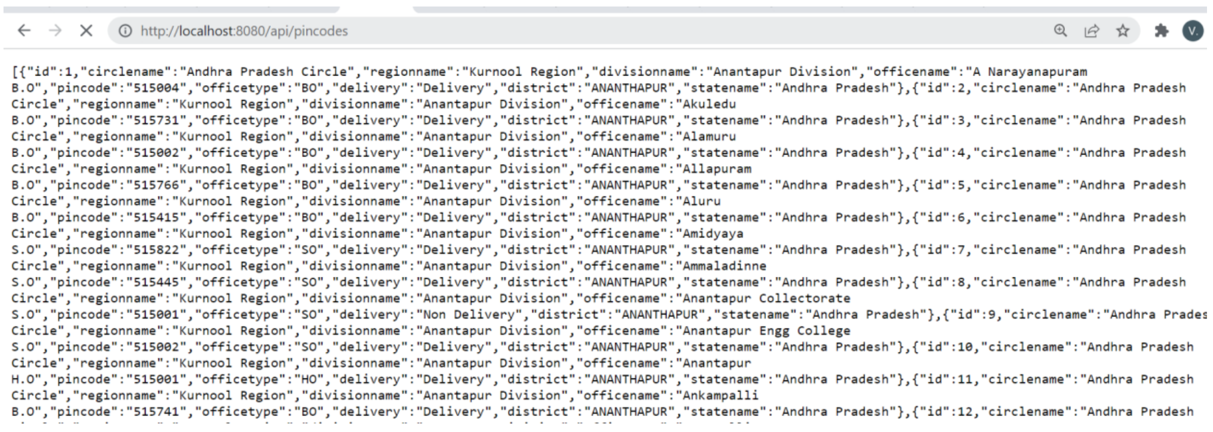


```
import com.sangamone.pincodedemo1.dao.PincodeDao;
import com.sangamone.pincodedemo1.model.Pincodes;

@RestController
@RequestMapping("/api/pincodes")
public class PincodeControllerRestapi {

    @Autowired
    PincodeDao pRepo;

    @GetMapping
    public List<Pincodes> findAllPincodes(){
        return pRepo.findAll();
    }
```

Now Compile and Run the Pincode Project.

Localhost:8080/api/pinodes

Output:



```
[{"id":1,"circlename":"Andhra Pradesh Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"A Narayanapuram
B.O","pincode":"515004","officetype":"BO","delivery":"Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":2,"circlename":"Andhra Pradesh
Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"Akuledu
B.O","pincode":"515731","officetype":"BO","delivery":"Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":3,"circlename":"Andhra Pradesh
Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"Alamuru
B.O","pincode":"515002","officetype":"BO","delivery":"Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":4,"circlename":"Andhra Pradesh
Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"Allapuram
B.O","pincode":"515766","officetype":"BO","delivery":"Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":5,"circlename":"Andhra Pradesh
Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"Aluru
B.O","pincode":"515415","officetype":"BO","delivery":"Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":6,"circlename":"Andhra Pradesh
Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"Amidyaya
S.O","pincode":"515822","officetype":"SO","delivery":"Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":7,"circlename":"Andhra Pradesh
Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"Ammaladinne
S.O","pincode":"515445","officetype":"SO","delivery":"Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":8,"circlename":"Andhra Pradesh
Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"Anantapur Collectorate
S.O","pincode":"515001","officetype":"SO","delivery":"Non Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":9,"circlename":"Andhra Prades
Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"Anantapur Engg College
S.O","pincode":"515002","officetype":"SO","delivery":"Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":10,"circlename":"Andhra Pradesh
Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"Anantapur
H.O","pincode":"515001","officetype":"HO","delivery":"Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":11,"circlename":"Andhra Pradesh
Circle","regionname":"Kurnool Region","divisionname":"Anantapur Division","officename":"Ankampalli
B.O","pincode":"515741","officetype":"BO","delivery":"Delivery","district":"ANANTHAPUR","statename":"Andhra Pradesh"},{"id":12,"circlename":"Andhra Pradesh
```

Release History:

S.NO	Date	Details
1	14-Mar-2022	Initial Release by asharaniv.sangamone@gmail.com, Reviewed by laluprasaddash23.sangamone@gmail.com and riteshkumaryadav.sangamone@gmail.com

SangamOne
Connected Services