**Employee REST App**

**Objective** :

***In this exercise, you will create RESTful webservices using Spring Boot for HTTP GET request.***

# 1.0 Functional Requirements

Global technologies wanted to use a web service for manipulating the Employee details. To start with they need a web service to view all the employee details. Help them to automate the above process by developing Rest Service using Maven to retrieve the employee details.

Your application should support the below service :

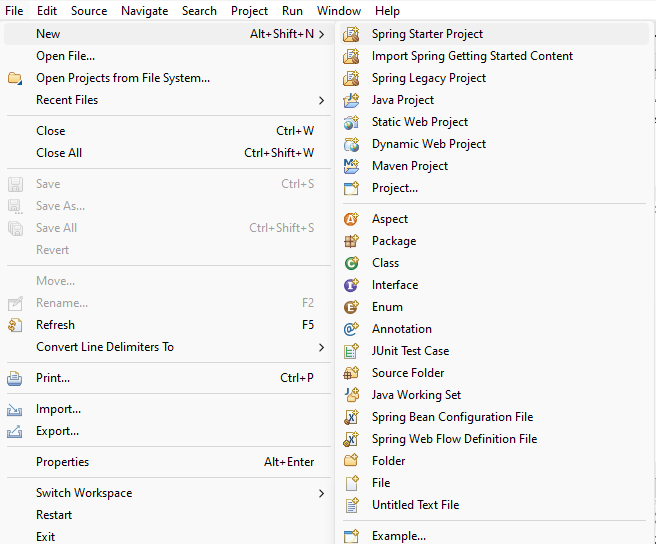
Get Request --> /employees : This service should retrieve all the employee details.

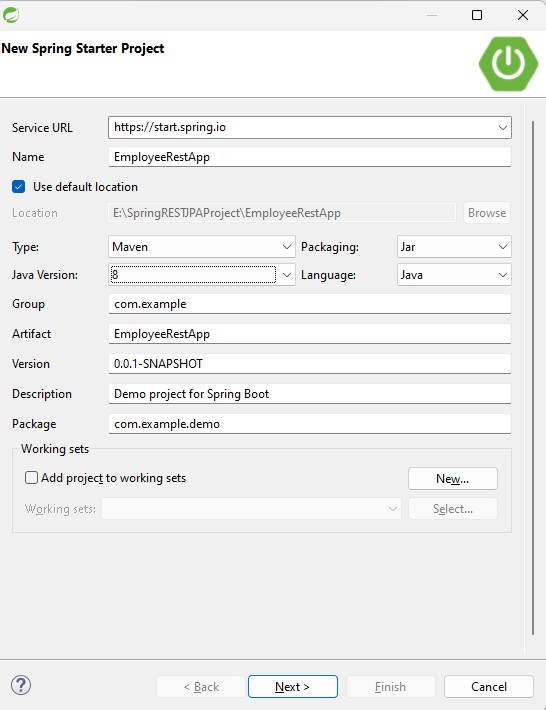
# 2.0 Technical Specifications

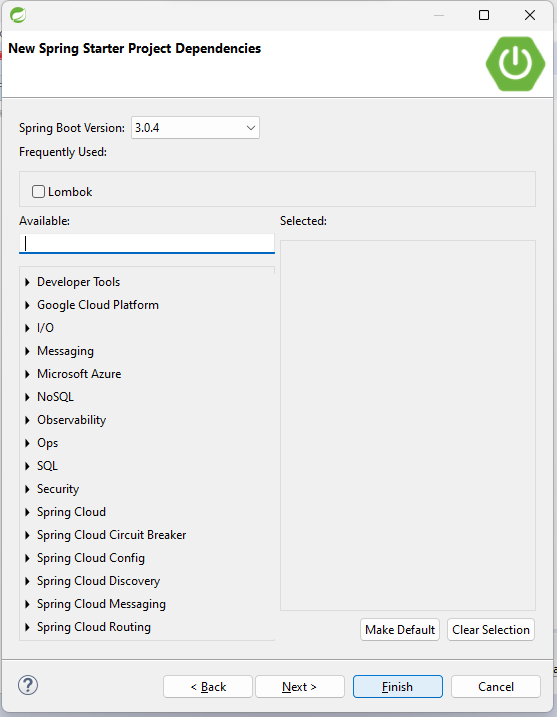
To start with this project, open STS, create a Spring Starter project and follow the instructions below.

1. Open IDE STS – Spring Tool Suite

2. Go to File ->Spring Starter Project



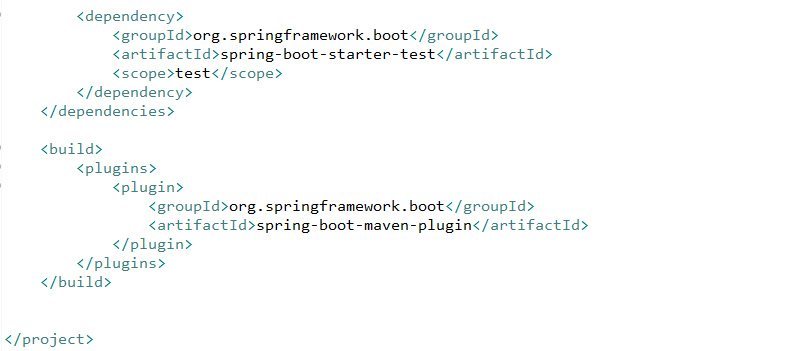


.

Click Finish.

3. Once the project is created, check / add the required dependency in pom.xml





4. Go to src/main/resources. Configure application.properties file

****

5. Go to src/main/java. Create packages com.model and com.controller.

# Model class

You need to create a Model class, Employee with attributes

int empId

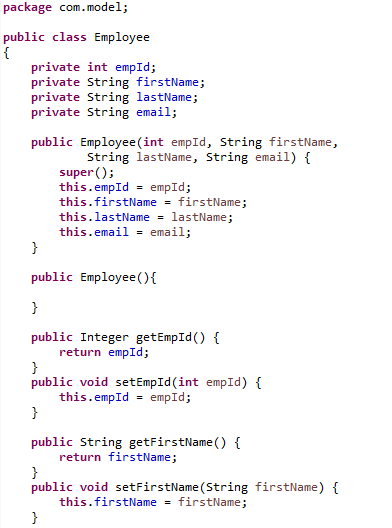
String firstName

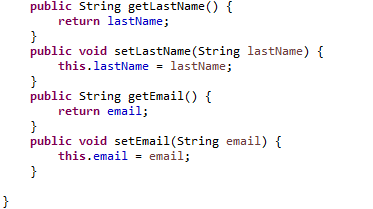
String lastName

String email

Write public getters and setters, a no argument constructor and a parameterized constructor with parameters in the order empId, firstName, lastName and email.

6. Create the class Employee in com.model package





# Service class

Create a class Company in com.model package. This class will have a list of Employee objects as attribute. The methods in this class should perform retrieval operations on Employee objects in the list.

7. Create the Company class in com.model package with the attribute as shown below.



Note that we have written a constructor and added few Employee objects into empList. This is optional.

You can do this, so that you can test your application easily in Postman Client.

**Note that to retrieve all the employee details, we have written the method getEmpList in the Company class.**

# Controller class

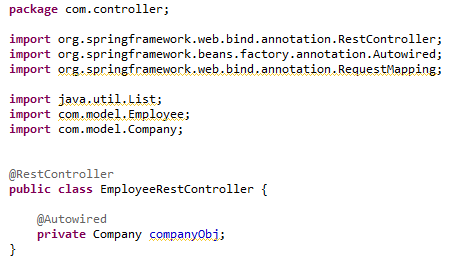
Create the REST API Controller in Spring Boot. For this, create a class EmployeeRestController in com.controller package.

EmployeeRestController class should act as the RestController, wherein the required service has to be created to perform the RETRIEVE operation.

This controller class should inject the Company class and invoke the method in it.

The data returned from the Controller should be a JSON which is by default.

8. Create the EmployeeRestController class with the below code



* **@RestController** makes the class as Spring Boot RESTful web service. It indicates that the data returned by each method in it will be written straight into the response body instead of rendering a template.

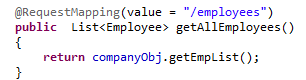
The class marked with @RestController performs CRUD operations and returns the output in JSON format to the user.

Also service class Company is injected in controller.

9. Implement the below services in the EmployeeRestController class

* **Get Request --> /employees** – This service should invoke the getEmpList()  of the Company class and return the list of employees.

To do this write the below code inside Company class



Here method is not provided. So default method taken is GET.

Also instead of



you can also use

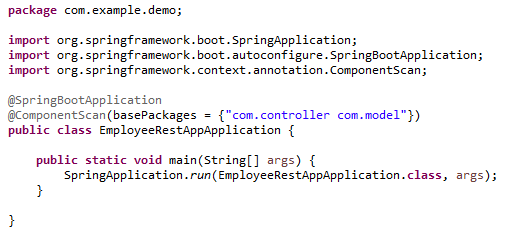


**Create the Launch class for Spring Boot Application**

Every Spring Boot Application needs one launch class. This class is annotated with the **@SpringBootApplication.**

10. In com.example.demo, you will have a class, EmployeeRestAppApplication. Write the below code in this class.

Annotate this class with **@SpringBootApplication** and @ComponentScan. Call the static **method run of SpringApplication** class with the class name as parameter.



**@ComponentScan** indicates in which packages there are annotated classes which should be managed by Spring.

**@SpringBootApplication** annotation makes this class a configuration class.

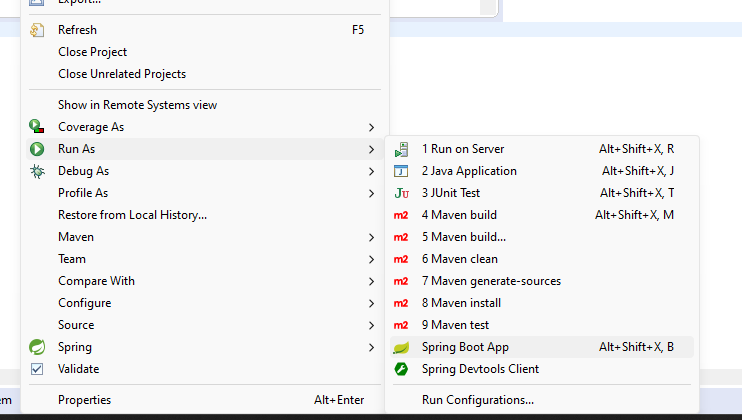
**SpringApplication**  bootstraps and auto-configures our application. It also starts the embedded Tomcat server.

The class name is passed as an argument to run method to indicate that this is primary Spring Component for the project.

11. Comment the code in the java file EmployeeRestAppApplicationTests available inside src/test/java folder.

12. Having completed the application, you can test the correctness by using Postman Client. To do this, Run the Application as Spring Boot App

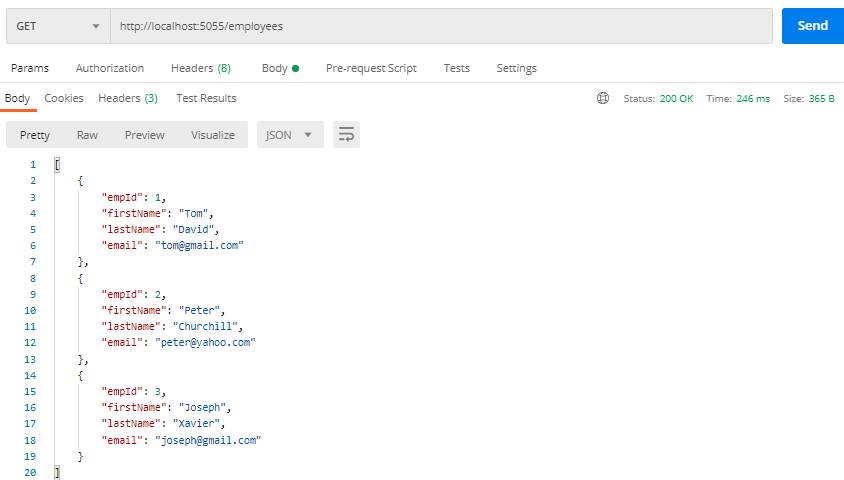
Right click the Project -> Run As -> Spring Boot App



You will see the below code in the console



13. Open Postman Client and check the application as shown below



# 3.0 Overall Design Constraints

When submitting the code to platform ensure the below points

1. **Check if the property name given in the application.properties files is same as given in the sample code. You can change the value if needed.**
2. **In the pom.xml ensure that you have provided only the dependencies provided in this sample. Don’t provide any additional dependency.**
3. **Use the service type and the service names as expected in the specification**
4. Adhere to the design specifications mentioned in the case study.
5. Ensure that you have provided all the classes / interface / attribute name / method name / return type / parameters as mentioned in the problem statement.
6. **Please make sure that your code does not have any compilation errors while submitting your case study solution.**

Congratulations, you have successfully completed the exercise on building a Spring REST API to perform a service using HTTP Get method !