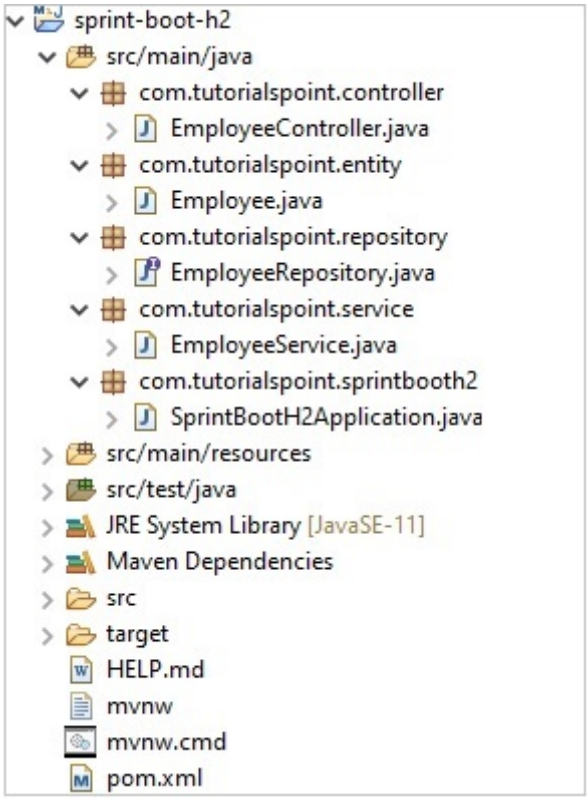
Spring Boot JPA - Application Setup

As in previous chapter Environment Setup , we've imported the generated spring boot project in eclipse. Now let's create the following structure in **src/main/java** folder.



- **com.tutorialspoint.controller.EmployeeController** − A REST Based Controller to implement REST based APIs.
- **com.tutorialspoint.entity.Employee** − An entity class representing the corresponding table in database.
- **com.tutorialspoint.repository.EmployeeRepository** − A Repository Interface to implement the CRUD operations on the database.
- **com.tutorialspoint.service.EmployeeService** − A Service Class to implement the business opearations over repository functions.
- **com.tutorialspoint.springbooth2.SprintBootH2Application** − A Spring Boot Application class.

SprintBootH2Application class is already present. We need to create the above packages and relevant classes and interface as shown below −

Entity - Entity.java

Following is the default code of Employee. It represents a Employee table with id, name, age and email columns.

```
package com.tutorialspoint.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table
public class Employee {
   @Id
   @Column
   private int id;

   @Column
   private String name;

   @Column
   private int age;

   @Column
   private String email;

   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
```

```
    public String getEmail() {
       return email;
    }
    public void setEmail(String email) {
       this.email = email;
    }
}
```

Repository - EmployeeRepository.java

Following is the default code of Repository to implement CRUD operations on above entity, Employee.

```
package com.tutorialspoint.repository;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import com.tutorialspoint.entity.Employee;

@Repository
public interface EmployeeRepository extends CrudRepository<Employee, Intege
}
```

Service - EmployeeService.java

Following is the default code of Service to implement operations over repository functions.

```
package com.tutorialspoint.service;

import java.util.ArrayList;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.tutorialspoint.entity.Employee;
import com.tutorialspoint.repository.EmployeeRepository;

@Service
public class EmployeeService {
   @Autowired
   EmployeeRepository repository;

   public Employee getEmployeeById(int id) {
      return repository.findById(id).get();
   }
```

```
    public List<Employee> getAllEmployees(){
       List<Employee> employees = new ArrayList<Employee>();
       repository.findAll().forEach(employee -> employees.add(employee));
       return employees;
    }
    public void saveOrUpdate(Employee employee) {
       repository.save(employee);
    }
    public void deleteEmployeeById(int id) {
       repository.deleteById(id);
    }
}
```

Controller - EmployeeController.java

Following is the default code of Controller to implement REST APIs.

```
package com.tutorialspoint.controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.tutorialspoint.entity.Employee;
import com.tutorialspoint.service.EmployeeService;

@RestController
@RequestMapping(path = "/emp")
public class EmployeeController {
   @Autowired
   EmployeeService employeeService;

   @GetMapping("/employees")
   public List<Employee> getAllEmployees(){
      return employeeService.getAllEmployees();
   }
   @GetMapping("/employee/{id}")
   public Employee getEmployee(@PathVariable("id") int id) {
      return employeeService.getEmployeeById(id);
```

```
    }
    @DeleteMapping("/employee/{id}")
    public void deleteEmployee(@PathVariable("id") int id) {
       employeeService.deleteEmployeeById(id);
    }
    @PostMapping("/employee")
    public void addEmployee(@RequestBody Employee employee) {
       employeeService.saveOrUpdate(employee);
    }
    @PutMapping("/employee")
    public void updateEmployee(@RequestBody Employee employee) {
       employeeService.saveOrUpdate(employee);
    }
}
```

Application - SprintBootH2Application.java

Following is the updated code of Application to use above classes.

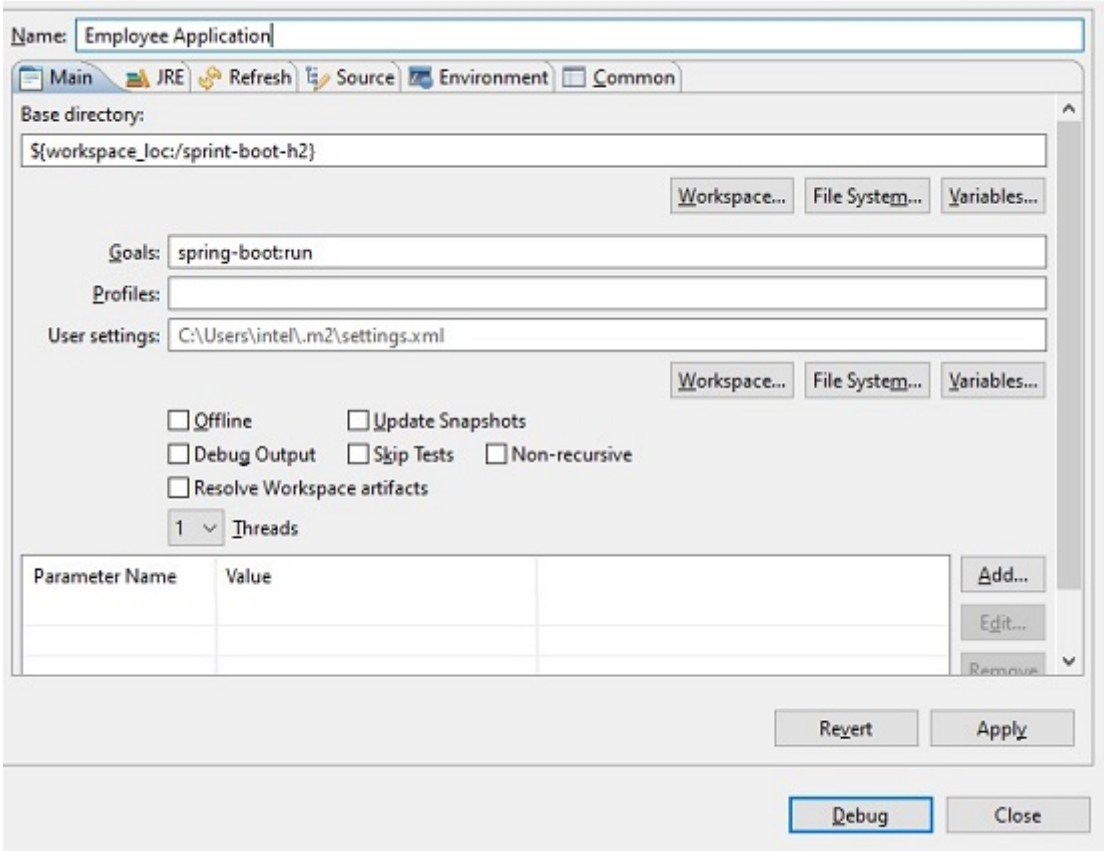
```
package com.tutorialspoint.sprintbooth2;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.data.jpa.repository.config.EnableJpaRepositorie

@ComponentScan({"com.tutorialspoint.controller","com.tutorialspoint.servic
@EntityScan("com.tutorialspoint.entity")
@EnableJpaRepositories("com.tutorialspoint.repository")
@SpringBootApplication
public class SprintBootH2Application {
   public static void main(String[] args) {
      SpringApplication.run(SprintBootH2Application.class, args);
   }
}
```

Run/Debug Configuration

Create following **maven configuration** in eclipse to run the springboot application with goal **spring-boot:run**. This configuration will help to run the REST APIs and we can test them using POSTMAN.



Run the application

In eclipse, run the **Employee Application** configuration. Eclipse console will show the similar output.

```
[INFO] Scanning for projects...
...
2021-07-24 20:51:14.823  INFO 9760 --- [restartedMain] c.t.s.SprintBootH2Applic
Started SprintBootH2Application in 7.353 seconds (JVM running for 8.397)
```

Once server is up and running, Use Postman to make a POST request to add a record first.

Set the following parameters in POSTMAN.

- HTTP Method - **POST**
- URL - **http://localhost:8080/emp/employee**
- BODY - **An employee JSON**

```
{
   "id": "1",
   "age": "35",
   "name": "Julie",
   "email": "julie@gmail.com"
}
```

Click on Send Button and check the response status to be OK. Now make a GET Request to get all records.

Set the following parameters in POSTMAN.

- HTTP Method - **GET**
- URL - **http://localhost:8080/emp/employees**

Click the send button and verify the response.

```
[{
   "id": "1",
   "age": "35",
   "name": "Julie",
   "email": "julie@gmail.com"
}]
```