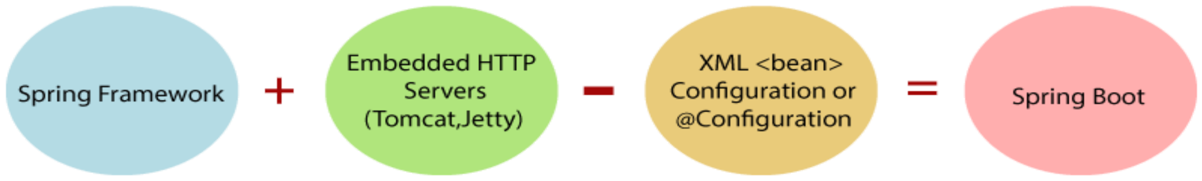
Spring Boot

Spring boot is a project which is used to develop both simple and web-based applications in Java. It is very convenient for the developer as it is a combination of spring framework and embedded server.


Spring Framework
+
Embedded HTTP Servers (Tomcat,Jetty)
-
XML <bean> Configuration or @Configuration
=
Spring Boot

In a spring boot project, we do not need to configure beans and external dependencies. We just need to add maven dependencies in pom.xml files and everything will be auto configured for the developer.

Steps for a Spring Boot Crud application-

1. Go to https://start.spring.io/ and kick start your project. Add necessary dependencies like Spring Data JPA, Spring Web, Spring Boot DevTools, MySql Driver. Select as maven and enter.

2. Import as maven project and check pom for dependencies we added.

3. Create some basic packages named as entity (for entity classes), dao (for dao classes), service (for service classes), rest (for controller classes).

4. Create entity class in entity package and write the required code with annotations, getters and setters, toString() and constructors.

5. Update application.properties in resources folder with below code.
```
spring.datasource.url=jdbc:mysql://localhost:3306/employee_directory?useSSL=false
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.hikari.maximum-pool-size=100
spring.datasource.hikari.minimum-idle=10
spring.jpa.show-sql=true
logging.level.web=DEBUG
```

6. There are two ways to connect a project to database either by Dao classes or JPA repository. If developer uses Dao classes, it will be a lot of code and must write code for different functions in implementation class. On the other have if developer uses Jpa Repository, you just need to create interface in Dao package with “org.springframework.data.jpa.repository.JpaRepository<T, ID>” extension. After you create JpaRepository just replace <T, ID> with <Entity_name, Integer> and then there is no need to code for Dao (It’s just this easy).

7. Create an interface in the service package with the name service.java and declare the necessary methods for your project and then create a class with name ServiceImpl in the same package with the extension of the service interface

we just created to define the methods declared in the service class. Use @Service before ServiceImpl class name, Inject JpaRepository with an object, @Autowired a constructor injection and define JpaRepository with tempJpaObject in the parenthesis and initialize the main JpaObject with tempJpaObject in the constructor body. Now one by one write codes for different methods like you want them to function in your application. Basically, in ServiceImpl all you need is to return the methods with the JpaRepository.

8. Now create a controller class in the rest package to control your serviceImpl class functions. First define @RestController and @RequestMapping("/api") before controller class name, inject service Impl with its object, @Autowired a constructor injection and define serviceImpl with tempImplObject in the parenthesis and initialize the main ImplObject with tempImplObject in the constructor body. Map with annotations before all the functions. There are several annotations for mapping like-

   @GetMapping to get the data
   @PostMapping to add the data
   @PutMapping to update the data
   @DeleteMapping to delete the data

9. After doing the necessary coding in the above classes, run the project as an application and after it is up in IDE, check localhost:8080/ in Browser.

10. Do CRUD operations in postman with the same URL from the browser and verify from the database.

11. Find the code, postman collections and SQL query here: https://github.com/Leaf-Co-Kb/Employee-Managment