BSc (Hons) in Information Technology
Specializing in Software Engineering
Year 3 - 2021
SE3040 – Application Frameworks
Lab 09 – Spring Boot

1. Check the Java setup.
   a. JAVA_HOME environment variable should be set to JDK directory.
   b. {JAVA_HOME}/bin should be added to Path variable.
   c. 'java -version' and 'java -version' should give you the version of the installation if setup correctly.
2. Setup Maven.
   a. Download Maven https://maven.apache.org/download.cgi
   b. Extract the archive.
   c. Setup M2_HOME environment variable to Maven directory.
   d. Add {M2_HOME}/bin to Path variable.
   e. 'mvn version' should give you the Maven version of the installation if setup correctly.
3. Go to https://start.spring.io/
   a. Select Project type Maven (default).
   b. Select Language Java (default).
   c. Spring boot version should be latest stable (default).
   d. Type meaningful Group and Artifact (Name).
   e. Packaging should be Jar.
   f. Java version 11 should be selected.
4. Adding dependencies
   a. Click on add dependencies.
   b. Add following dependencies.
      i. Spring Web
      ii. Spring Boot Actuator
5. Click generate and download the project. Extract and open in the IDE.
6. Explore the project.
   a. POM file
      i. Dependencies
      ii. Parent
      iii. Plugin
   b. Application file in the src directory.
7. Let's change the default web server from Tomcat to Undertow.
   a. Exclude the tomcat dependency from spring-boot-web-starter.
```
<exclusions>
    <exclusion>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
    </exclusion>
</exclusions>
```
   b. Add the undertow dependency.
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-undertow</artifactId>
</dependency>
```

8. Run the application.

   a. With the spring configuration you should be able to run the application by running the main method in Application class.

9. Check http://localhost:8080/ in browser, you should receive an error page from undertow.
10. Add sub package named ‘controller’ into the main package.
11. Add PostEndpoint.java into the package ‘controller’.
12. Annotate PostEndpoint class with @RestController and @RequestMapping(“/posts”).
13. Add packages for business logic as ‘api’ and ‘domain’.
14. Add a class named Post.java into ‘domain’ package with properties id, name and, description (String) with getters and setters.
15. Add a class PostApi.java into ‘api’ package.
    a. Annotate it with @Service.
    b. Add a method to return List of Post.java.
    c. Add an instance level variable a Map of Post.java.
    d. In the method body return all values of Map as a list.
16. Add a method to return all Post.java as a list to the PostEndpoint.java class. Annotate the method with @GetMapping. This method should call PostApi to get the list of Posts.
17. Add PostApi as dependency to PostEndpoint (instance level variable). Add a constructor which accepts a PostApi instance and setting into the instance variable. Annotate the constructor with @Autowired.
18. Try accessing http://localhost:8080/posts you should receive an empty JSON array.
19. Add a package named ‘dto’. Add a class to the package named ‘PostDto’ with only ‘name’ and ‘description’ with getters and setters.
20. Let’s add a method which to create a new Post. Use annotation @RequestBody (should be in the method argument for PostDto class) to accept JSON in the HTTP request body.
21. Add a method to PostApi to add a new Post to the Map (method should accept a Post in the method).
22. In the method in PostEndpoint create a new instance of Post using the data passed in the PostDto in the method argument.
23. You should not expect ID to be passed by the user, thus generate the ID to the Post object in the PostApi using UUID class.
24. Add a method to remove a post by ID to the PostAPI.
25. Add a method to remove a post in PostEndpoint. Use @DeleteMapping(“/{id}”) to mark the path and @ResponseStatus(HttpStatus.NO_CONTENT) to change the default successful response code to 204. Use @PathVaraible to mark the argument ID to extract from the URL path.
26. Try these endpoints in Postman
    a. GET http://localhost:8080/posts
    b. POST http://localhost:8080/posts
    c. DELETE http://localhost:8080/{id}
27. Add a method to update a post using @PutMapping.
28. Try Spring Actuator endpoints.
    a. http://localhost:8080/actuator/health/