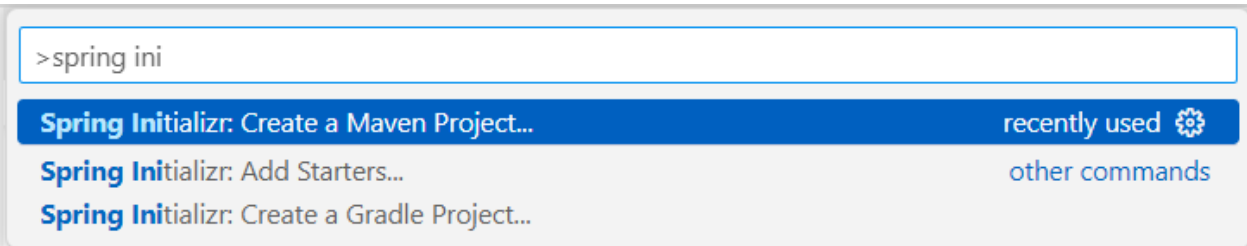
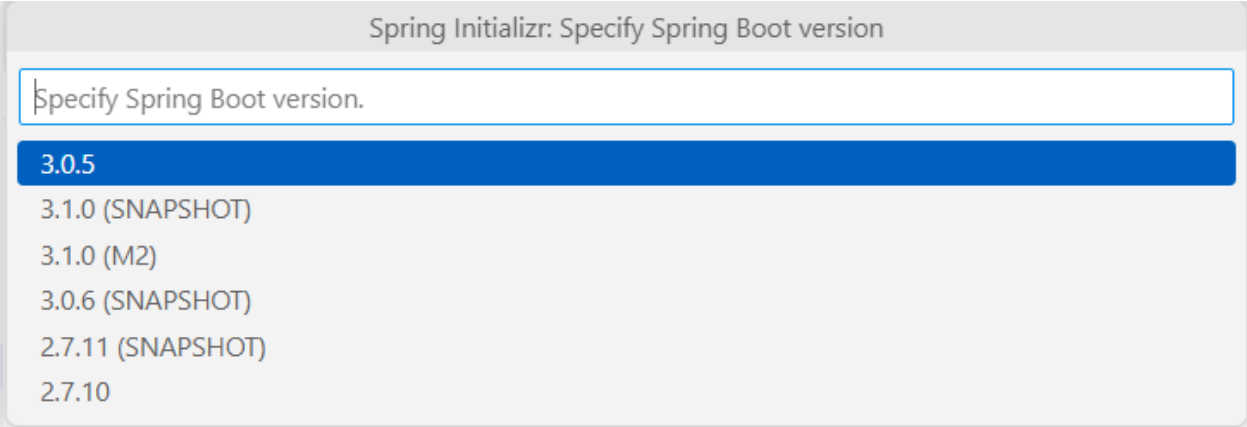
SE3040 – Application Frameworks
BSc (Hons) in Information Technology
Software Engineering Specialization
3rd Year
Faculty of Computing
SLIIT
2023 - Practical
Lab 06

Building REST services with Spring

1. Setting up VS code for Spring Boot. Add the following dependencies to the VS code
    a. Spring Initializr Java Support - Microsoft
    b. Extension Pack for Java – Microsoft
2. Press F1 or Ctrll + Shift + P from the keyboard to open the command palette.
3. Then select the following



4. Then Select the latest Spring Boot version



5. Select the programming language as Java
6. Input the group id and then Artifact Id. To learn about them refer to this.
7. Then select the packaging type as Jar
8. After that select the Java version that is installed on your PC.

9. Select a directory to start the project.
10. Open the directory from the VS Code, if it is not opened automatically.
11. Navigate to the main\java\<<group id>>\<<artifact id>>\<<artifact Id>>Application.java
12. Modify the existing code as follows

```
@SpringBootApplication
@RestController
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    @GetMapping("/")
    public String rootEndpoint(){
        String message = "Hello, world!";
        return message;
    }
}
```

13. Modify the above code to accept a parameter and display the message as follows. Refer to this link.

Hello, <<name>>!

Ex: Hello, Janith!

Building Rest API

1. Create a new package called com.example.springbootrestapi.model, and inside it, create a class called User:

```
package com.example.rest.model;


public class User {
   private Long id;
   private String name;
   private String email;

   public Long getId() {
      return id;
   }

   public void setId(Long id) {
      this.id = id;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getEmail() {
      return email;
   }

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

}
```

2. To interact with the database, we need to create a repository interface. But in this lab we will test the REST API without integrating the Database.

```
package com.example.rest.model;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class UserRepository {
   private List<User> users = new ArrayList<>();
   private long nextId = 1;

   public List<User> findAllUsers() {
      return users;
   }

   public Optional<User> findUserById(Long id) {
      return users.stream()
            .filter(user -> user.getId().equals(id))
            .findFirst();
   }

   public User saveUser(User user) {
      if (user.getId() == null) {
         user.setId(nextId++);
      } else {
         users.removeIf(existingUser -> existingUser.getId().equals(user.getId()));
      }

      users.add(user);
      return user;
   }

   public void deleteUserById(Long id) {
      users.removeIf(user -> user.getId().equals(id));
   }
}
```

3. Now, let's create a REST controller to handle HTTP requests. Create a new package called com.example.springbootrestapi.controller, and inside it, create a class called UserController

```
package com.example.rest.controller;


import java.util.List;
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.example.rest.model.UserRepository;
import com.example.rest.model.User;

@RestController
@RequestMapping("/api/users")
public class UserController {
   private final UserRepository userRepository = new UserRepository();


   @GetMapping
   public List<User> getAllUsers() {
      return userRepository.findAllUsers();
   }

   @PostMapping
   public User createUser(@RequestBody User user) {
      return userRepository.saveUser(user);
   }

   @GetMapping("/{id}")
   public User getUserById(@PathVariable Long id) {
      return userRepository.findUserById(id)
            .orElseThrow(() -> new IllegalArgumentException("User not found with id: " + id));
   }

   @PutMapping("/{id}")
   public User updateUser(@PathVariable Long id, @RequestBody User updatedUser) {
      User user = userRepository.findUserById(id)
            .orElseThrow(() -> new IllegalArgumentException("User not found with id: " + id));
```

```
        user.setName(updatedUser.getName());
        user.setEmail(updatedUser.getEmail());

        return userRepository.saveUser(user);
    }

    @DeleteMapping("/{id}")
    public void deleteUser(@PathVariable Long id) {
        userRepository.deleteUserById(id);
    }
}
```

4. Run the application.

Note that the default port will be 8080