**Assisted Practice: 4.1 Build RESTful with Spring Boot**

This section will guide you to:

* Build Restful with Spring Boot

This lab has eight sub-sections, namely:

4.1.1 Opening Spring Tool Suite and creating a new project using Spring Initializer

4.1.2 Selecting the required project dependencies

4.1.3 Creating an Entity Class

4.1.4 Creating a Repository Class

4.1.5 Creating a Service Class

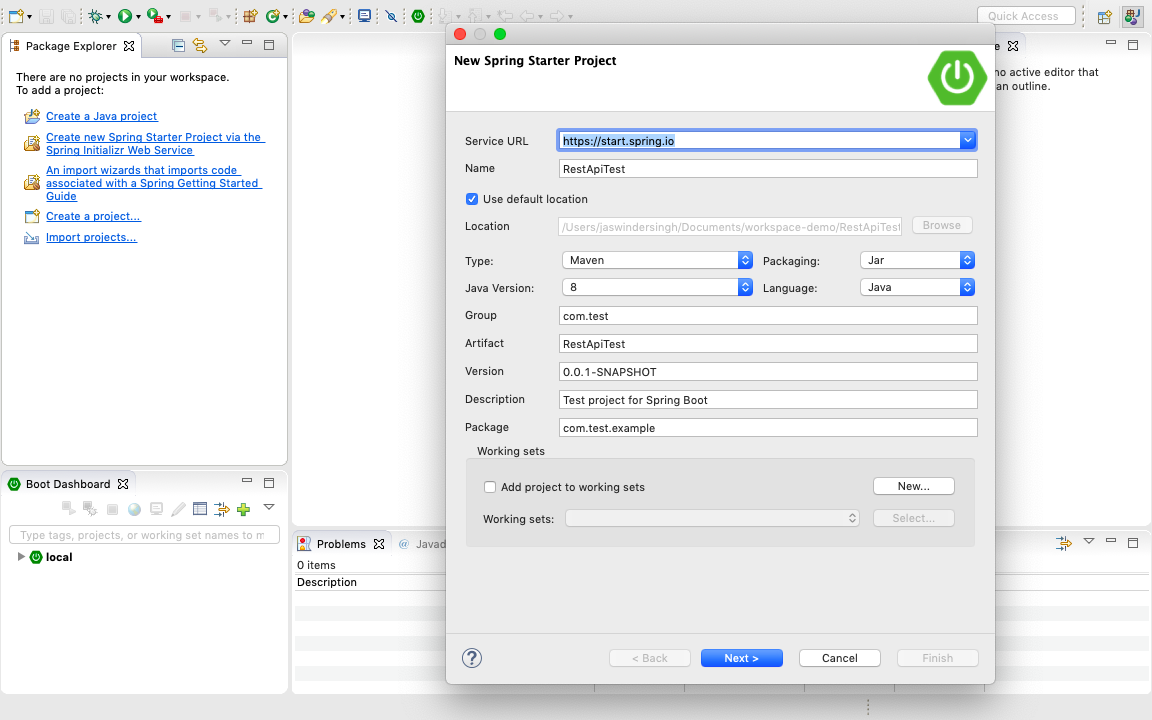
4.1.6 Creating a Controller Class

4.1.7 Executing the project as ‘Spring Boot App’

4.1.8 Pushing the code to your GitHub repositories

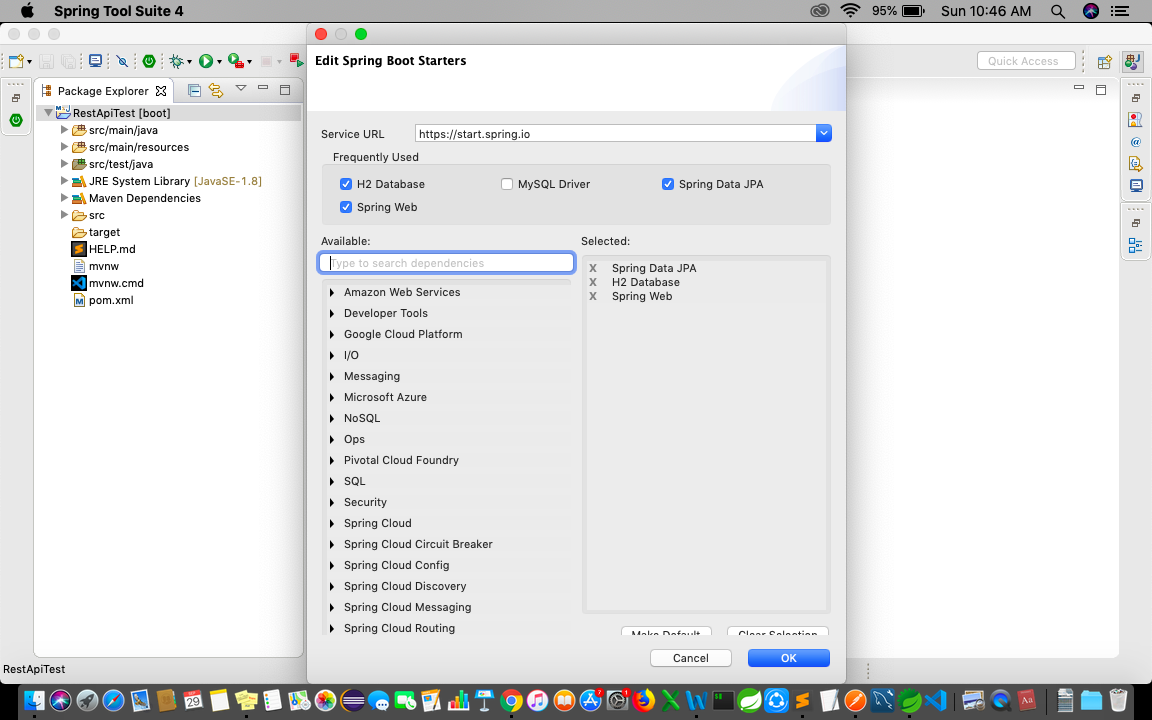
**Step 4.1.1:** Opening Spring Tool Suite and creating a new project using Spring Initializer

* Fill in the required fields.

****

* It will automatically create the class with the ‘main’ method.

**Step 4.1.2:** Selecting the required project dependencies namely, Spring Web, H2 Database (Inbuilt Database of Spring Boot) and Spring Data JPA. Then, click on **Finish**.

****

* It will automatically create the class with the ‘main’ method.

package com.test.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RestApiTestApplication {

public static void main(String[] args) {

SpringApplication.run(RestApiTestApplication.class, args);

}

}

**Step 4.1.3:** Creating an Entity Class

**package com.test.example;**

**import javax.persistence.Column;**

**import javax.persistence.Entity;**

**import javax.persistence.GeneratedValue;**

**import javax.persistence.GenerationType;**

**import javax.persistence.Id;**

**@Entity**

**public class ProductEntity {**

**@Id**

**@GeneratedValue(strategy = GenerationType.AUTO)**

**@Column(name = "id", updatable = false, nullable = false)**

**private int id;**

**@Column**

**private String name;**

**@Column**

**private String description;**

**public ProductEntity() {**

**super();**

**}**

**public ProductEntity(int id, String name, String description) {**

**super();**

**this.id = id;**

**this.name = name;**

**this.description = description;**

**}**

**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 String getDescription() {**

**return description;**

**}**

**public void setDescription(String description) {**

**this.description = description;**

**}**

**}**

**Step 4.1.4:** Creating a Repository Class

package com.test.example;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<ProductEntity, Integer>{

}

**Step 4.1.5:** Creating a Service Class

package com.test.example;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class ProductService {

@Autowired

ProductRepository productRepository;

public List<ProductEntity> getAllProduct(){

return productRepository.findAll();

}

public ProductEntity getProduct(int id){

return productRepository.findById(id).get();

}

public void addProduct(ProductEntity pe){

productRepository.save(pe);

}

public void updateProduct(int id, ProductEntity pe){

if(productRepository.findById(id).isPresent()) {

ProductEntity oldProductEntity=productRepository.findById(id).get();

oldProductEntity.setName(pe.getName());

oldProductEntity.setDescription(pe.getDescription());

productRepository.save(oldProductEntity);

}

}

public void deleteProduct(int id){

productRepository.deleteById(id);

}

}

**Step 4.1.6:** Creating a Controller Class

**package com.test.example;**

**import java.util.List;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.web.bind.annotation.PathVariable;**

**import org.springframework.web.bind.annotation.RequestBody;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.RequestMethod;**

**import org.springframework.web.bind.annotation.RestController;**

**@RestController**

**@RequestMapping(path = "/webapi")**

**public class ProductController {**

**@Autowired**

**ProductService productService;**

**@RequestMapping("/allproduct")**

**public List<ProductEntity> getAllProduct(){**

**return productService.getAllProduct();**

**}**

**@RequestMapping("/product/{id}")**

**public ProductEntity getProduct(@PathVariable int id){**

**return productService.getProduct(id);**

**}**

**@RequestMapping(method=RequestMethod.POST, value="/product")**

**public void addProduct(@RequestBody ProductEntity pe ) {**

**productService.addProduct(pe);**

**}**

**@RequestMapping(method=RequestMethod.PUT, value="/product/{id}")**

**public void updateProduct(@PathVariable int id, @RequestBody ProductEntity pe ) {**

**productService.updateProduct(id, pe);**

**}**

**@RequestMapping(method=RequestMethod.DELETE, value="/product/{id}")**

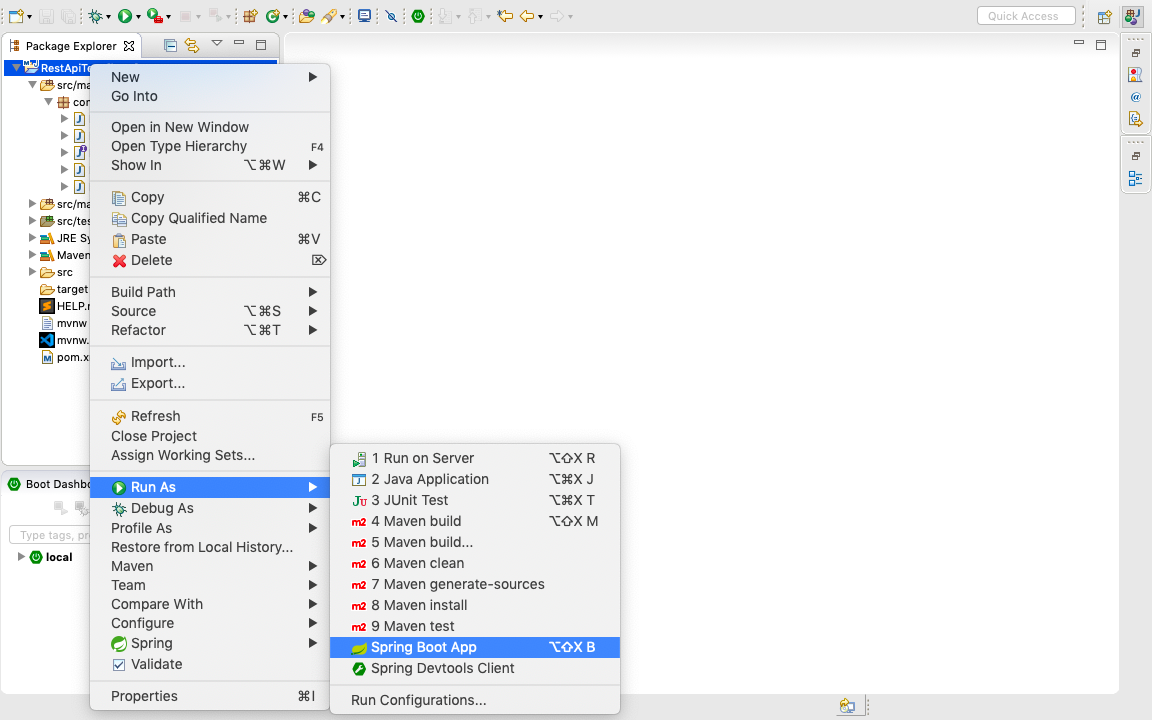
**public void deleteProduct(@PathVariable int id) {**

**productService.deleteProduct(id);**

**}**

**}**

**Step 4.1.7:** Executing the project as ‘Spring Boot App’

****

* Open Postman. Now, you can perform all the CRUD Operations.

**Step 4.1.8:** Pushing the code to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**