

Phase-3 Practice Project: Assisted Practice

7. You are given a project to demonstrate RESTful with Spring Boot.

```
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);
    }

}
```

ProductEntity: package

com.test.example;

import jakarta.persistence.Column; import jakarta.persistence .Entity ; import
jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType;

@ Entity

public class ProductEntity {

@jakarta.persistence.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;

}

}

ProductRepository: package

com.test.example; import

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

```
public interface ProductRepository extends JpaRepository<ProductEntity, Integer>{  
  
}
```

```
ProductService: 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);
```

```
}
```

```
}
```

```
ProductController: 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);  
}  
}
```

Output:



The screenshot displays the output of a Spring Boot application startup. On the left, a 'Failure Trace' panel is visible. The main console shows a series of log messages from the application's main method. The logs indicate that the application is running on Java 17.0.7 on a DESKTOP. It shows the initialization of the Spring Data JPA repositories, the Tomcat web server, and the HikariPool-1 database connection pool. The application successfully starts on port 8080 (http) with the context path '/'. The logs also show the Hibernate ORM core version 5.6.15.Final and the Hibernate Commons Annotations 5.1.2.Final. The application is using the org.hibernate.dialect.H2Dialect. The logs conclude with the message 'Started RestApiTestApplication in 9.762 seconds (JVM running)'.

```
2023-06-16 09:22:37.693 INFO 9872 --- [main] com.test.example.RestApiTestApplication : Starting RestApiTestApplication using Java 17.0.7 on DESKTOP
2023-06-16 09:22:37.706 INFO 9872 --- [main] com.test.example.RestApiTestApplication : No active profile set, falling back to 1 default profile: "default"
2023-06-16 09:22:39.835 INFO 9872 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2023-06-16 09:22:39.966 INFO 9872 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 103 ms. Found 1 JPA repository.
2023-06-16 09:22:42.041 INFO 9872 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-06-16 09:22:42.074 INFO 9872 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-06-16 09:22:42.075 INFO 9872 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.76]
2023-06-16 09:22:42.507 INFO 9872 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-06-16 09:22:42.507 INFO 9872 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 4500 ms
2023-06-16 09:22:42.866 INFO 9872 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-06-16 09:22:43.290 INFO 9872 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-06-16 09:22:43.406 INFO 9872 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH0000204: Processing PersistenceUnitInfo [name: default]
2023-06-16 09:22:43.528 INFO 9872 --- [main] org.hibernate.Version : HHH0000412: Hibernate ORM core version 5.6.15.Final
2023-06-16 09:22:43.957 INFO 9872 --- [main] o.hibernate.annotations.common.Version : HCA00000001: Hibernate Commons Annotations {5.1.2.Final}
2023-06-16 09:22:44.226 INFO 9872 --- [main] org.hibernate.dialect.Dialect : HHH0000400: Using dialect: org.hibernate.dialect.H2Dialect
2023-06-16 09:22:45.291 INFO 9872 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000498: Using JtaPlatform implementation: [org.hibernate.jta.platform.internal.NoJtaPlatform]
2023-06-16 09:22:45.311 INFO 9872 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-06-16 09:22:45.830 WARN 9872 --- [main] jpase.BaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database access may cause requests to hang or timeouts (timeout due to socket options). To avoid this issue, add this property to your configuration: 'spring.jpa.open-in-view=false'.
2023-06-16 09:22:46.622 INFO 9872 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path '/'
2023-06-16 09:22:46.644 INFO 9872 --- [main] com.test.example.RestApiTestApplication : Started RestApiTestApplication in 9.762 seconds (JVM running)
```