**SWP391 – Back-End Technologies in use (SE - FPTU)**

**Phần RESTful API với Spring Boot**

# 1. Tools & Technologies

Spring Boot 3+

JDK 17+

Spring Framework 6+

Hibernate 6+

JPA

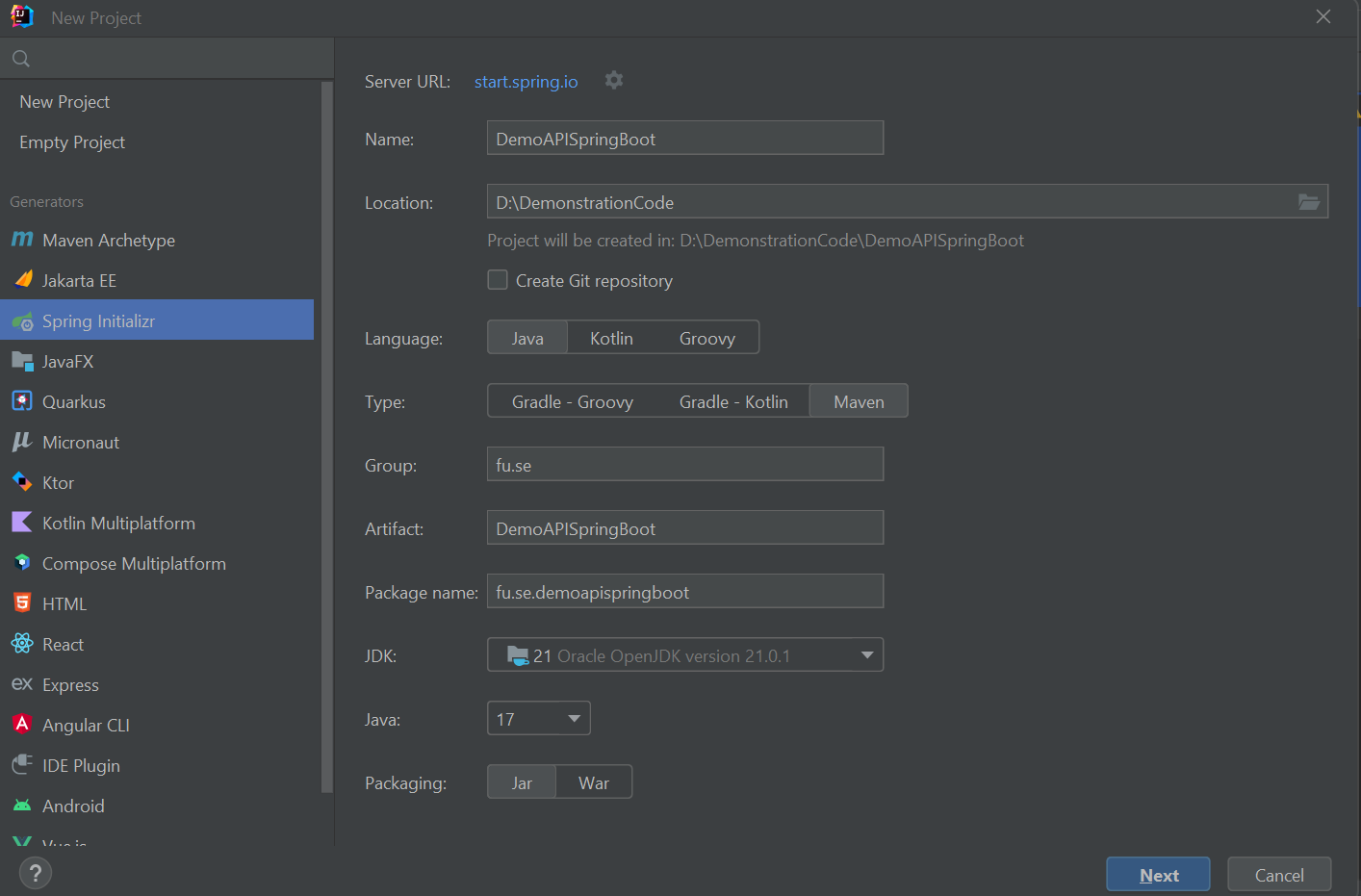
Maven 3.2+

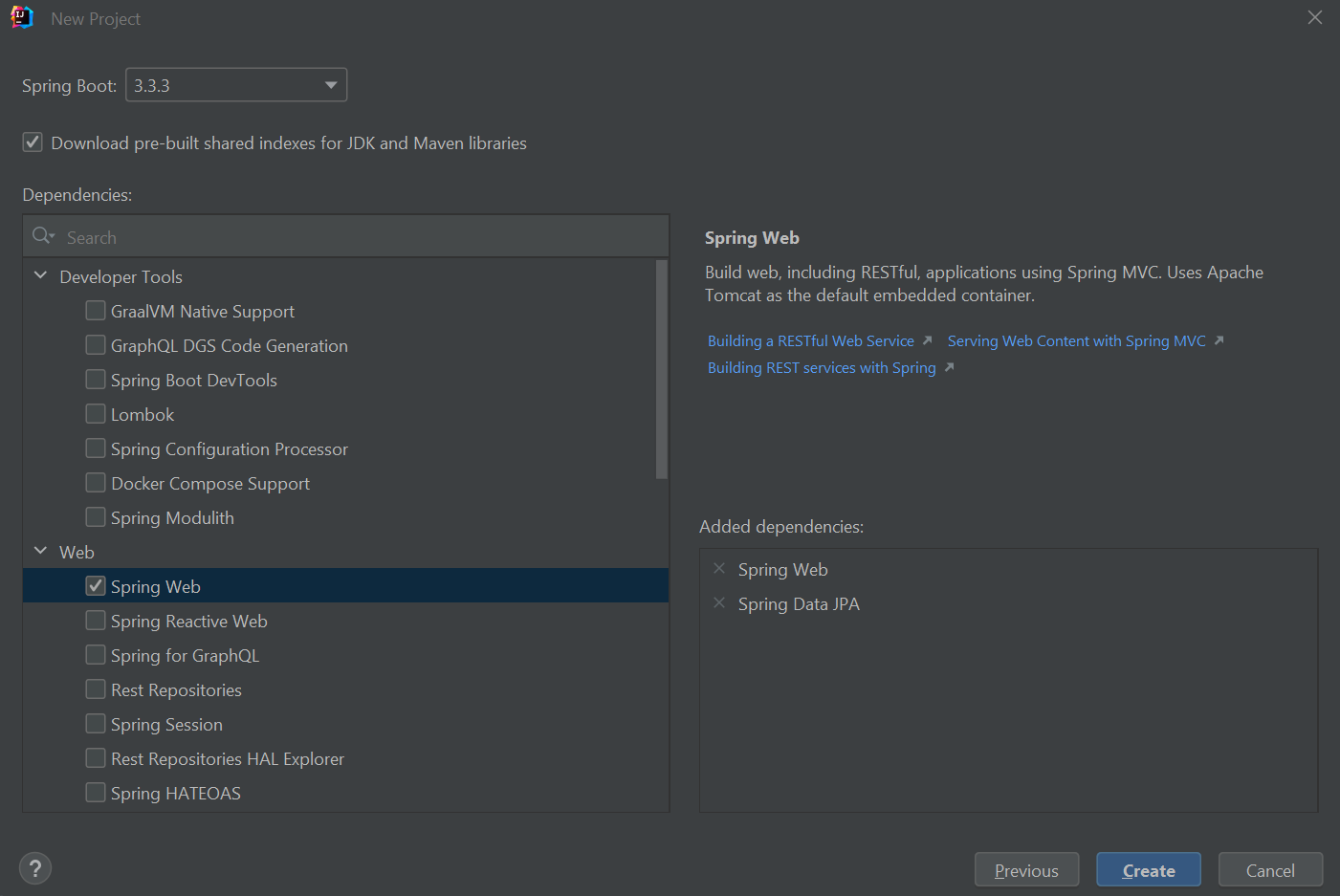
IDE **IntelliJ IDEA** /Eclipse/Spring Tool Suite (STS)

RDBMS Microsoft SQL Server

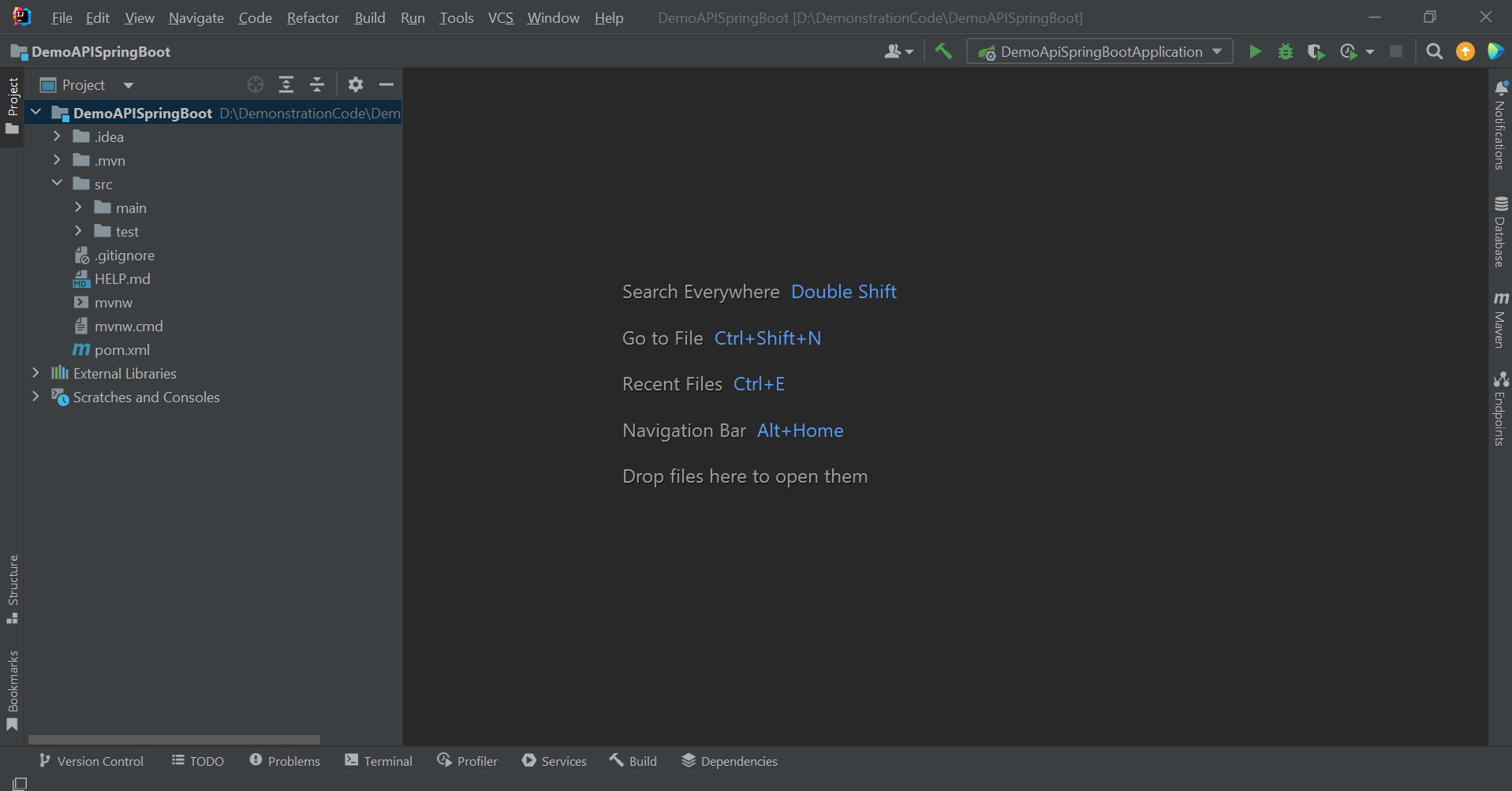
# 2. Phần Hướng dẫn làm Project với Spring Initializr

B1. Tạo Project dùng IntelliJ, các IDE khác cũng tương tự.





Chọn Spring Web (cho việc xây dựng RESTful API) và Spring Data JPA (cho việc thao tác với CSDL)



Thêm Maven dependency vào <dependencies> trong file pom.xml

*<dependency>*

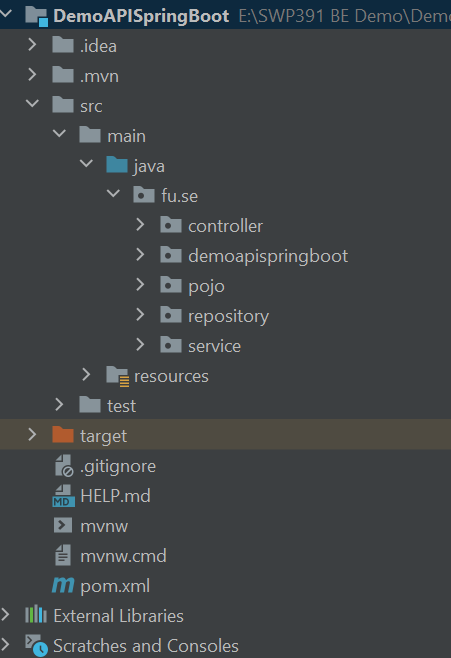
*<groupId>com.microsoft.sqlserver</groupId>*

*<artifactId>****mssql-jdbc****</artifactId>*

*<scope>runtime</scope>*

*</dependency>*

Cấu trúc Project

****

B2. Làm việc với cấu hình cho Data Source (MS SQL Server) trong /src/main/resources/**application.properties**

spring.application.name=DemoAPISpringBoot  
  
spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver  
  
spring.datasource.url=jdbc:sqlserver://localhost:1433;encrypt=true;trustServerCertificate=true;databaseName=Orchid  
spring.datasource.username=**sa**  
spring.datasource.password=**1234567890**  
  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.SQLServerDialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.hibernate.ddl-auto có 4 giá trị ***create, create-drop, validate, và update***

B3. Tạo JPA Entity

package fu.se.pojo;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
  
@Entity  
@Table(name = "orchid")  
public class Orchid {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name="orchid\_id")  
 private int orchidID;  
  
 @Column(name="orchid\_name")  
 private String orchidName;  
  
 @Column(name="is\_natural",columnDefinition="bit default 0")  
 private boolean isNatural;  
  
 @Column(name="orchid\_description")  
 private String orchidDescription;  
  
 @Column(name="orchid\_category")  
 private String orchidCategory;  
  
 @Column(name="is\_attractive",columnDefinition="bit default 0")  
 private boolean isAttractive;  
  
 @Column(name="orchid\_url")  
 private String orchidURL;  
  
 public Orchid() {  
  
 this.orchidName = "";  
 this.isNatural = false;  
 this.orchidDescription = "";  
 this.orchidCategory = "";  
 this.isAttractive = false;  
 this.orchidURL = "";  
 }  
 public Orchid(int orchidID, String orchidName, boolean isNatural, String orchidDescription, String orchidCategory, boolean isAttractive, String orchidURL) {  
  
 this.orchidID = orchidID;  
 this.orchidName = orchidName;  
 this.isNatural = isNatural;  
 this.orchidDescription = orchidDescription;  
 this.orchidCategory = orchidCategory;  
 this.isAttractive = false;  
 this.orchidURL = orchidURL;  
 }  
  
 public int getOrchidID() {  
 return orchidID;  
 }  
  
 public void setOrchidID(int orchidID) {  
 this.orchidID = orchidID;  
 }  
  
 public String getOrchidName() {  
 return orchidName;  
 }  
  
 public void setOrchidName(String orchidName) {  
 this.orchidName = orchidName;  
 }  
  
 public String getOrchidDescription() {  
 return orchidDescription;  
 }  
  
 public void setOrchidDescription(String orchidDescription) {  
 this.orchidDescription = orchidDescription;  
 }  
  
 public String getOrchidCategory() {  
 return orchidCategory;  
 }  
  
 public void setOrchidCategory(String orchidCategory) {  
 this.orchidCategory = orchidCategory;  
 }  
 public boolean isNatural() {  
 return isNatural;  
 }  
 public void setNatural(boolean isNatural) {  
 this.isNatural = isNatural;  
 }  
  
 public boolean isAttractive() {  
 return isAttractive;  
 }  
  
 public void setAttractive(boolean attractive) {  
 isAttractive = attractive;  
 }  
  
 public String getOrchidURL() {  
 return orchidURL;  
 }  
  
 public void setOrchidURL(String orchidURL) {  
 this.orchidURL = orchidURL;  
 }  
}

*@Entity*: Xác định Java class là Entity. An entity is a lightweight persistence domain object that is mapped to a database table. @Entity là required cho lớp Entity.  
*@Table*: Default thì tên bảng trùng tên Java class.  
*@Id*: Khóa chính của 1 bảng (primary key).  
*@Column*: Default thì tên cột trùng tên thuộc tính.  
*@GeneratedValue*: Primary key được phát sinh, gồm các giá trị AUTO, IDENTITY, SEQUENCE, và TABLE (by default, the strategy is AUTO).

B3. Create Spring Data Repository - EmployeeRepository.java

package fu.se.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
import fu.se.pojo.Orchid;  
  
@Repository  
public interface IOrchidRepository extends JpaRepository<Orchid, Integer> {  
  
}

Interface IOrchidRepository thừa kế JpaRepository interface. IOrchidRepository thừa kế các phương thức trong *JpaRepository: findAll(), findById(), save(), và deleteById()*. (CRUD cơ bản, không cần dùng câu lệnh SQL)

B4. Tạo Service (tương ứng với việc xử lý nghiệp vụ (Business Logic Layer) gồm IOrchidService.java và OrchidService.java

package fu.se.service;  
  
import fu.se.pojo.Orchid;  
import java.util.List;  
import java.util.Optional;  
  
public interface IOrchidService {  
 public List<Orchid> getAllOrchids();  
  
 public Orchid insertOrchid(Orchid orchid);  
  
 public Orchid updateOrchid(int orchiD, Orchid orchid);  
  
 public void deleteOrchid(int orchiD);  
  
 public Optional<Orchid> getOrchidById(int orchiD);  
  
 public void test();  
}

package fu.se.service;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import fu.se.pojo.Orchid;  
import fu.se.repository.IOrchidRepository;  
  
import java.util.List;  
import java.util.Optional;  
  
  
@Service  
public class OrchidService implements IOrchidService{  
  
 @Autowired  
 private IOrchidRepository iOrchidRepository;  
  
 @Override  
 public List<Orchid> getAllOrchids() {  
 // *TODO Auto-generated method stub* return iOrchidRepository.findAll();  
 }  
  
 @Override  
 public Orchid insertOrchid(Orchid orchid) {  
 // *TODO Auto-generated method stub* return iOrchidRepository.save(orchid);  
 }  
  
 @Override  
 public Orchid updateOrchid(int orchiD, Orchid norchid) {  
 // *TODO Auto-generated method stub* Orchid orchid = iOrchidRepository.getById(orchiD);  
 if(orchid !=null) {  
 orchid.setOrchidName(norchid.getOrchidName());  
 orchid.setNatural(norchid.isNatural());  
 return iOrchidRepository.save(orchid);  
 }  
 return orchid;  
 }  
  
 @Override  
 public void deleteOrchid(int orchiD) {  
 // *TODO Auto-generated method stub* iOrchidRepository.deleteById(orchiD);  
 }  
 @Override  
 public Optional<Orchid> getOrchidById(int orchiD) {  
 // *TODO Auto-generated method stub* return iOrchidRepository.findById(orchiD);  
 }  
  
 @Override  
 public void test() {  
 // *TODO Auto-generated method stub* }  
  
}

B5. Tạo REST Controller

package fu.se.controller;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
  
import org.springframework.web.bind.annotation.\*;  
import fu.se.pojo.Orchid;  
import fu.se.service.IOrchidService;  
import java.util.List;  
import java.util.Optional;  
  
  
@RestController  
@RequestMapping("/orchids")  
public class OrchidController {  
  
 @Autowired  
 private IOrchidService iOrchidService;  
  
 @GetMapping("/fetchAll")  
 public ResponseEntity<List<Orchid>> fetchAll() {  
 return ResponseEntity.*ok*(iOrchidService.getAllOrchids());  
 }  
  
  
  
 @PostMapping("/insert")  
 @ResponseStatus(HttpStatus.*CREATED*)  
 public Orchid saveProduct(@RequestBody Orchid orchid) {  
 return iOrchidService.insertOrchid(orchid);  
 }  
  
// @PostMapping("/insert")  
// public ResponseEntity<Orchid> saveProduct(@RequestBody Orchid orchid) {  
// Orchid newProduct = iOrchidService.insertOrchids(orchid);  
// return ResponseEntity.ok().body(newProduct);  
// }  
  
 @PutMapping("/update/{id}")  
 public ResponseEntity<Orchid> updateProduct(@PathVariable int id, @RequestBody Orchid product) {  
 Orchid updatedProduct = iOrchidService.updateOrchid(id, product);  
 return ResponseEntity.*ok*(updatedProduct);  
 }  
  
 @DeleteMapping("/delete/{id}")  
 public ResponseEntity<String> deleteProduct(@PathVariable int id) {  
 iOrchidService.deleteOrchid(id);  
 return ResponseEntity.*ok*("Product deleted successfully");  
 }  
  
 @GetMapping("/{id}")  
 public ResponseEntity<Optional<Orchid>> getProductByID(@PathVariable int id) {  
 Optional<Orchid> orchid = iOrchidService.getOrchidById(id);  
 return ResponseEntity.*ok*(orchid);  
 }  
}

B6. Dùng @ComponentScan, @EnableJpaRepositories, @ EntityScan

@ComponentScangiúp tự động tìm kiếm và quản lý các bean trong ứng dụng.

package fu.se.demoapispringboot;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.boot.autoconfigure.domain.EntityScan;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;  
  
@SpringBootApplication  
  
@ComponentScan({ "fu.se.controller","fu.se.service" })  
@EnableJpaRepositories(basePackages = "fu.se.repository")  
@EntityScan(basePackages = "fu.se.pojo")  
public class DemoApiSpringBootApplication {  
  
 public static void main(String[] args) {  
  
 SpringApplication.*run*(DemoApiSpringBootApplication.class, args);  
 }  
  
}

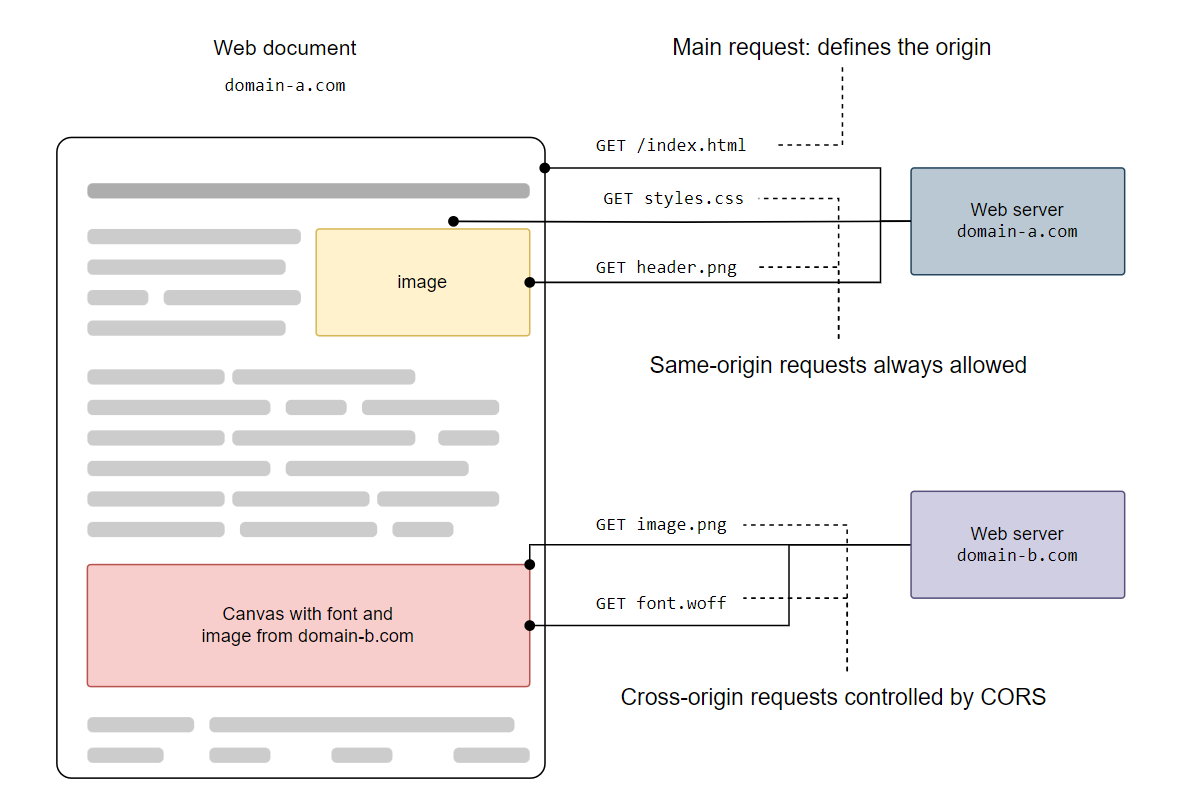
# 3. Test API dùng Postman

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | API name | HTTP Method | Path | Status Code |
| 1 | GET Orchids | GET | http://localhost:8080/orchids/fetchAll | 200 OK |
| 2 | POST Orchid | POST | *http://localhost:8080/orchids/insert* | 201 Created |
| 3 | DELETE Orchid | DELETE | *http://localhost:8080/orchids/delete/{id}* | 204 No Content |
| 4 | PUT Orchid | PUT | *http://localhost:8080/orchids/update/{id}* | 200 OK |
| 5 | GET Orchid | GET | *http://localhost:8080/orchids/{id}* | 20 OK |

# 4. Note bổ sung Spring Boot

## 4.1. Xử lý CORS trong Spring REST - Origin (domain, scheme, port)

Tham khảo CORS <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>



* CORS (Cross Origin Resource Sharing) trong controller method

@RestController  
@RequestMapping("/category")  
public class CategoryController {  
  
 **@CrossOrigin**  
 @RequestMapping(method = RequestMethod.*GET*, path = "/{id}")  
 public Category retrieve(@PathVariable Long id) {  
 // ...  
 }  
  
 @RequestMapping(method = RequestMethod.*DELETE*, path = "/{id}")  
 public void remove(@PathVariable Long id) {  
 // ...  
 }  
}

* @CrossOrigin trong Controller

@CrossOrigin(origins = "http://example.com", maxAge = 3600)  
@RestController  
@RequestMapping("/category")  
public class CategoryController {  
  
 @RequestMapping(method = RequestMethod.*GET*, path = "/{id}")  
 public Category retrieve(@PathVariable Long id) {  
 // ...  
 }  
  
 @RequestMapping(method = RequestMethod.*DELETE*, path = "/{id}")  
 public void remove(@PathVariable Long id) {  
 // ...  
 }  
}

* @CrossOrigin cho cả class và phương thức của controller

@CrossOrigin(maxAge = 3600)  
@RestController  
@RequestMapping("/category")  
public class CategoryController {  
  
 @CrossOrigin("http://example.com")  
 @RequestMapping(method = RequestMethod.*GET*, "/{id}")  
 public Category retrieve(@PathVariable Long id) {  
 // ...  
 }  
  
 @RequestMapping(method = RequestMethod.*DELETE*, path = "/{id}")  
 public void remove(@PathVariable Long id) {  
 // ...  
 }  
}

## 4.2. JPA vs. Hibernate

**JPA:** JPA là một đặc tính của Java dùng để truy cập, quản lý và lưu trữ dữ liệu giữa đối tượng Java và Relational Database (cách tiếp cận chuẩn cho ORM)

* JPA là một Java feature dùng để ánh xạ dữ liệu quan hệ trong ứng dụng Java.
* JPA không cung cấp bất kỳ lớp implementation nào.
* JPA dùng ngôn ngữ truy vấn **JPQL** (Java Persistence Query Language).
* JPA trong package **javax.persistence**.
* JPA được triển khai trong các công cụ ORM khác nhau như Hibernate, EclipseLink …
* JPA sử dụng EntityManager để xử lý persistence của dữ liệu.

**Hibernate:** Là công cụ ORM mã nguồn mở, nhẹ được sử dụng để lưu trữ các đối tượng Java trong hệ thống Relational Database. Hibernate là một nhà cung cấp (provider)của JPA. Hibernate tuân theo cách tiếp cận do JPA cung cấp.

* Hibernate là một **framework** **ORM** xử lý persistence của dữ liệu.
* Hibernate cung cấp các class implementation.
* Hibernate dùng ngôn ngữ truy vấn của riêng **HQL** (Hibernate Query Language).
* Hibernate trong package **org.hibernate** .
* Hibernate là **nhà cung cấp** của JPA.
* Hibernate sử dụng **Session** để xử lý persistence của dữ liệu.

## 4.3. CrudRepository vs. JpaRepository interface

@NoRepositoryBean

public interface **CrudRepository**<T,ID> extends Repository<T,ID>

<https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/repository/CrudRepository.html>

@NoRepositoryBean

public interface **JpaRepository**<T,ID> extends ListCrudRepository<T,ID>, **ListPagingAndSortingRepository**<T,ID>, QueryByExampleExecutor<T>

<https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html>

## 4.4. Quan hệ One-toOne, One-to-Many, Many-to-Many trong JPA (bidirection)

Quan hệ One to Many (1 Department có nhiều Employee, 1 Employee chỉ thuộc về 1 Department)

@Entity  
public class Department {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String name;  
 @OneToMany(mappedBy = "department", cascade = CascadeType.*ALL*, orphanRemoval = true)  
 private List<Employee> employees = new ArrayList<>();  
 //Getters and Setters  
}

@Entity  
public class Employee {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String name;  
 @ManyToOne  
 @JoinColumn(name = "department\_id")  
 private Department department;  
 //Constructors  
 //Getters, Setters  
}

Quan hệ One to One (1 Employee có 1 Address, 1 Address chỉ thuộc về 1 Employee)

@Entity  
public class Address {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @OneToOne(fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "person\_id")  
 private Employee employee;  
  
 private String city;  
  
 //Constructors  
 //Getters, Setters  
}

@Entity  
public class Employee {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String name;

@ManyToOne  
 @JoinColumn(name = "department\_id")  
 private Department department;

@OneToOne(mappedBy = "employee", cascade = CascadeType.*ALL*, fetch = FetchType.*LAZY*)  
 private Address address;

//Constructors  
 //Getters, Setters  
}

Quan hệ Many to Many (1 Student tham gia nhiều Course, 1 Course có nhiều Student tham gia)

@Entity  
public class Student {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String name;  
  
 @ManyToMany(mappedBy = "students")  
 private Set<Course> courses = new HashSet<>();

//Constructors  
 //Getters, Setters  
  
}

@Entity  
public class Course {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String name;  
  
 @ManyToMany  
 @JoinTable(  
 name = "course\_student",  
 joinColumns = @JoinColumn(name = "course\_id"),  
 inverseJoinColumns = @JoinColumn(name = "student\_id")  
 )  
 private Set<Student> students = new HashSet<>();

//Constructors  
 //Getters, Setters  
  
}

## 4.5. Convert từ Entity sang DTO và ngược lại bằng thư viên ModelMapper.

<https://mvnrepository.com/artifact/org.modelmapper/modelmapper>