برای ایجاد یک برنامه CRUD ساده با استفاده از Spring Boot که شامل محصول ((Productو دسته‌بندی ((Categoryباشد، مراحل زیر را دنبال می‌کنیم. در این برنامه، روابط بین محصول و دسته‌بندی نیز ایجاد می‌شود و متد به‌روزرسانی همراه با اعتبارسنجی و استفاده از Regex پیاده‌سازی خواهد شد. در نهایت، متدهای تست به صورت JSON نیز ارائه می‌شود.

1. ایجاد پروژه Spring Boot

ابتدا یک پروژه Spring Boot جدید با استفاده از Spring Initializr ایجاد کنید و وابستگی‌های زیر را انتخاب کنید:

- Spring Web
- Spring Data JPA
-) H2 Databaseبرای تست)
-) Spring Boot DevToolsاختیاری)

2. تعریف موجودیت‌ها

Category.java

```java
import javax.persistence.*;
import java.util.Set;

@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true)
    private String name;

    @OneToMany(mappedBy = "category")
    private Set<Product> products;

    //Getters and Setters
}
```

Product.java

```java
import javax.persistence.*;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
```

```java
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull(message = "Product name cannot be null")
    @Pattern(regexp = "^[A-Za-z0-9 ]{3,50}$", message = "Product name must be between 3 and 50 characters and can only contain letters, numbers, and spaces")
    private String name;

    @NotNull(message = "Price cannot be null")
    private Double price;

    @ManyToOne
    @JoinColumn(name = "category_id", nullable = false)
    private Category category;

    //Getters and Setters
}
```

3. ایجاد Repositoryها

CategoryRepository.java

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

public interface CategoryRepository extends JpaRepository<Category, Long> {
}
```

ProductRepository.java

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

public interface ProductRepository extends JpaRepository<Product, Long> {
}
```

4. ایجاد Serviceها

CategoryService.java

```java
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class CategoryService {
    @Autowired
    private CategoryRepository categoryRepository;

    public List<Category> findAll() {
        return categoryRepository.findAll();
    }

    public Category save)Category category) {
        return categoryRepository.save(category);
    }

    public Category update)Long id, Category category) {
        category.setId(id);
        return categoryRepository.save(category);
    }

    public void delete)Long id) {
        categoryRepository.deleteById(id);
    }
}
```

ProductService.java

java

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.validation.Valid;
import java.util.List;

@Service
public class ProductService {
    @Autowired
    private ProductRepository productRepository;

    public List<Product> findAll() {
        return productRepository.findAll();
    }

    public Product save@)Valid Product product) {
        return productRepository.save(product);
```

```
    }

    public Product update)Long id,@ Valid Product product) {
        product.setId(id);
        return productRepository.save(product);
    }

    public void delete)Long id) {
        productRepository.deleteById(id);
    }
}
```

5. ایجاد Controllerها

CategoryController.java

java

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/categories")
public class CategoryController {
    @Autowired
    private CategoryService categoryService;

    @GetMapping
    public List<Category> getAllCategories() {
        return categoryService.findAll();
    }

    @PostMapping
    public ResponseEntity<Category> createCategory@)RequestBody Category category)
{
        return ResponseEntity.ok(categoryService.save(category));
    }

    @PutMapping("/{id}")
    public ResponseEntity<Category> updateCategory@)PathVariable Long id,
@RequestBody Category category) {
        return ResponseEntity.ok(categoryService.update(id, category));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteCategory@)PathVariable Long id) {
```

```
        categoryService.delete(id);
        return ResponseEntity.noContent().build();
    }
}
```

ProductController.java

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;

@RestController
@RequestMapping("/api/products")
public class ProductController {
    @Autowired
    private ProductService productService;

    @GetMapping
    public List<Product> getAllProducts() {
        return productService.findAll();
    }

    @PostMapping
    public ResponseEntity<Product> createProduct@)Valid@ RequestBody Product product) {
        return ResponseEntity.ok(productService.save(product));
    }

    @PutMapping("/{id}")
    public ResponseEntity<Product> updateProduct@)PathVariable Long id,@ Valid @RequestBody Product product) {
        return ResponseEntity.ok(productService.update(id, product));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteProduct@)PathVariable Long id) {
        productService.delete(id);
        return ResponseEntity.noContent().build();
    }
}
```

6. تست متدها با استفاده از JSON

ایجاد دسته‌بندی

```json
POST /api/categories
}
    "name": "Electronics"
{
```

ایجاد محصول

```json
POST /api/products
}
    "name": "Smartphone",
    "price": ,699.99
    "category": }
        "id": 1
    {
{
```

به‌روزرسانی محصول

```json
PUT /api/products/1
}
    "name": "Updated Smartphone",
    "price": ,749.99
    "category": }
        "id": 1
    {
{
```

به‌روزرسانی دسته‌بندی

```json
PUT /api/categories/1
}
    "name": "Updated Electronics"
{
```

7. اعتبارسنجی و مدیریت خطاها

برای مدیریت خطاها و اعتبارسنجی می‌توانید از `@ControllerAdvice` و `@ExceptionHandler` استفاده کنید تا خطاها را به صورت مناسب مدیریت کنید.

GlobalExceptionHandler.java

```java
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

import java.util.HashMap;
import java.util.Map;

@ControllerAdvice
public class GlobalExceptionHandler {
    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResponseEntity<Map<String, String>>
handleValidationExceptions)MethodArgumentNotValidException ex) {
        Map<String, String> errors = new HashMap<>();
        ex.getBindingResult().getFieldErrors().forEach(error -> {
            errors.put(error.getField(), error.getDefaultMessage());
        });
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(errors);
    }
}
```

نتیجه‌گیری

با پیاده‌سازی مراحل بالا، شما یک برنامه CRUD ساده با Spring Boot خواهید داشت که شامل محصول و دسته‌بندی است. اعتبارسنجی و مدیریت خطاها نیز به درستی پیاده‌سازی شده‌اند. می‌توانید با استفاده از ابزارهایی مانند Postman یا Curl، متدهای CRUD را آزمایش کنید.