FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

JpaRepository API

Slide 6.1

www.poly.edu.vn

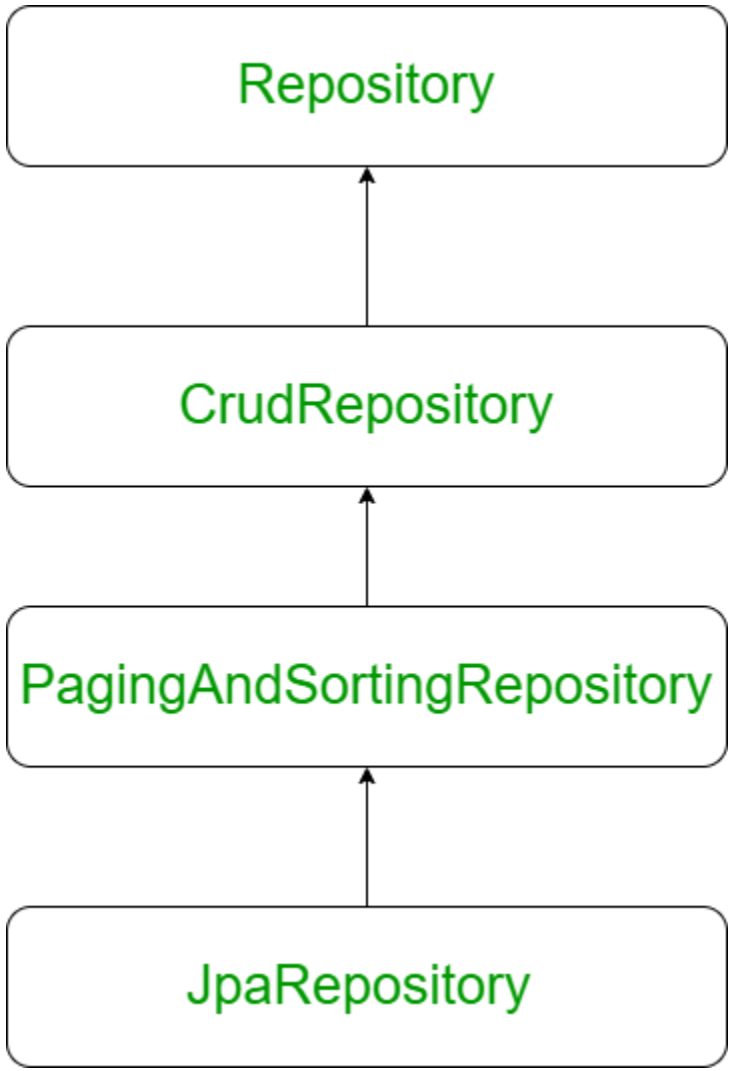
AGENDA

- PHẦN I:
  - GIỚI THIỆU JPAREPOSITORY
  - XÂY DỰNG ENTITY CLASS
  - XÂY DỰNG DAO
  - XÂY DỰNG ỨNG DỤNG CRUD
- PHẦN 2:
  - SẮP XẾP
  - PHÂN TRANG
  - XÂY DỰNG TRANG WEB PHÂN TRANG SẢN PHẨM

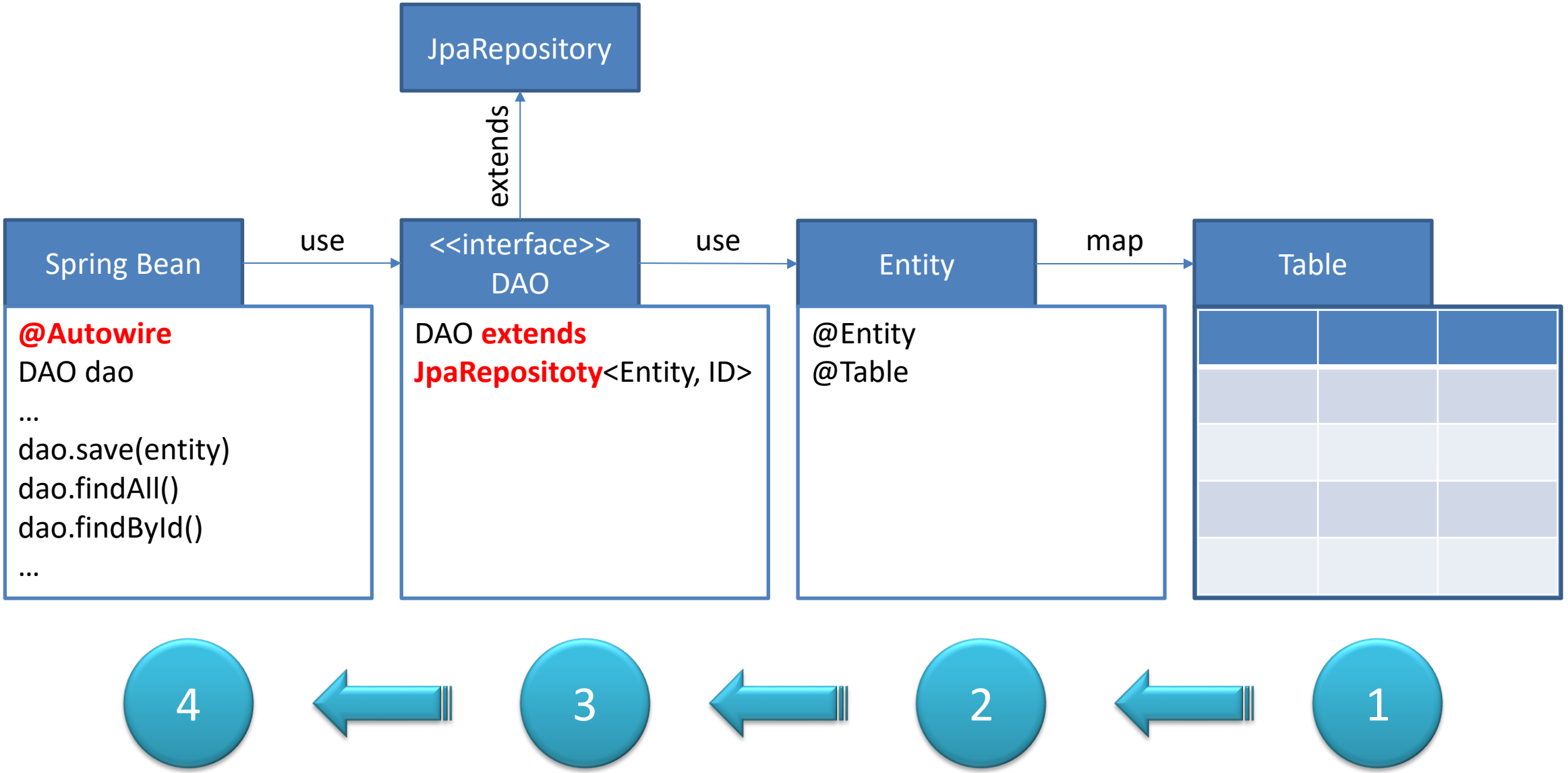
JpaRepository Introduction

GIỚI THIỆU JPAREPOSITORY

- JpaRepository là một API được hỗ trợ bởi Spring Boot giúp lập trình tương tác với CSDL theo phương pháp truy vấn đối tượng bằng JPQL (Java Persistence Query Language) một cách đơn giản và hiệu quả.
- Các thao tác dữ liệu như thêm, sửa, xóa và truy vấn đơn giản đã được JpaRepository xây dựng sẵn mà không cần phải viết thêm bớt kỳ một dòng mã nào.
- Các truy vấn phức tạp bạn chỉ cần chú thích với @Query với đối số là JPQL
- Cơ chế sắp xếp, phân trang linh hoạt, dễ dàng
- DSL (Domain Specific Language) giúp bạn không cần phải viết JPQL mà vẫn có thể truy vấn dữ liệu một cách đơn giản.



ỨNG DỤNG JPAREPOSITORY



JpaRepository API

Methods	Methods
findAll(): List<T>	delete(T)
findAll(Sort): List<T>	deleteById(ID)
findAllByIds(Iterable<ID>):List<T>	deleteAllById(Iterable<ID>)
findAll(Pageable): Page<T>	deleteAllByIdInBatch(Iterable<ID>)
findById(ID): Optional<T>	deleteAll()
getReferenceById(ID): T	deleteAll(Iterable<T>)
save(T): T	deleteAllInBatch()
saveAndFlush(T): T	deleteAllInBatch(Iterable<T>)
saveAll(Iterable<T>): List<T>	count(): long
saveAllAndFlush(Iterable<T>): List<T>	existsById(ID): boolean

CATEGORY - ENTITY CLASS

2

```
@Data
@Builder
@Entity
@Table(name = "Categories")
public class Category {
        @Id
        private String id;
        private String name;
}
```

map

1

```
CREATE TABLE Categories
(
    Id NVARCHAR(5) NOT NULL,
    Name NVARCHAR(50) NOT NULL,
    PRIMARY KEY (Id)
)
```

CATEGORYDAO AND CATEGORYSERVICE

```
public interface CategoryService {
        List<Category> findAll() ;
        Category findById(String id) ;
        Category create(Category entity) ;
        void update(Category entity) ;
        void deleteById(String id;
        boolean existsById(String id) ;
}
```

use → Category

implements

4.1

use

```
@Service
public class CategoryServiceImpl implements CategoryService {
        @Autowired
        CategoryDAO dao;
        ...
}
```

use

```
public interface CategoryDAO
        extends JpaRepository<Category, String> {
}
```

3

CategoryServiceImpl

4.2

```
@Autowired
CategoryDAO dao;

@Override
public List<Category> findAll() {
        return dao.findAll();
}

@Override
public Category findById(String id) {
        return dao.findById(id).get();
}

@Override
public Category create(Category entity) {
        return dao.save(entity);
}

@Override
public void update(Category entity) {
        dao.save(entity);
}

@Override
public void deleteById(String id) {
        dao.deleteById(id);
}

@Override
public boolean existsById(String id) {
        return dao.existsById(id);
}
```

POM.XML - THƯ VIỆN CẦN THIẾT

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
        <groupId>com.microsoft.sqlserver</groupId>
        <artifactId>mssql-jdbc</artifactId>
        <scope>runtime</scope>
</dependency>
```

FPT Education
FPT POLYTECHNIC

APPLICATION.PROPERTIES - THÔNG SỐ KẾT NỐI CSDL

spring.datasource.url=<dburl>
spring.datasource.username=<username>
spring.datasource.password=<password>
spring.datasource.driverClassName=<driver>

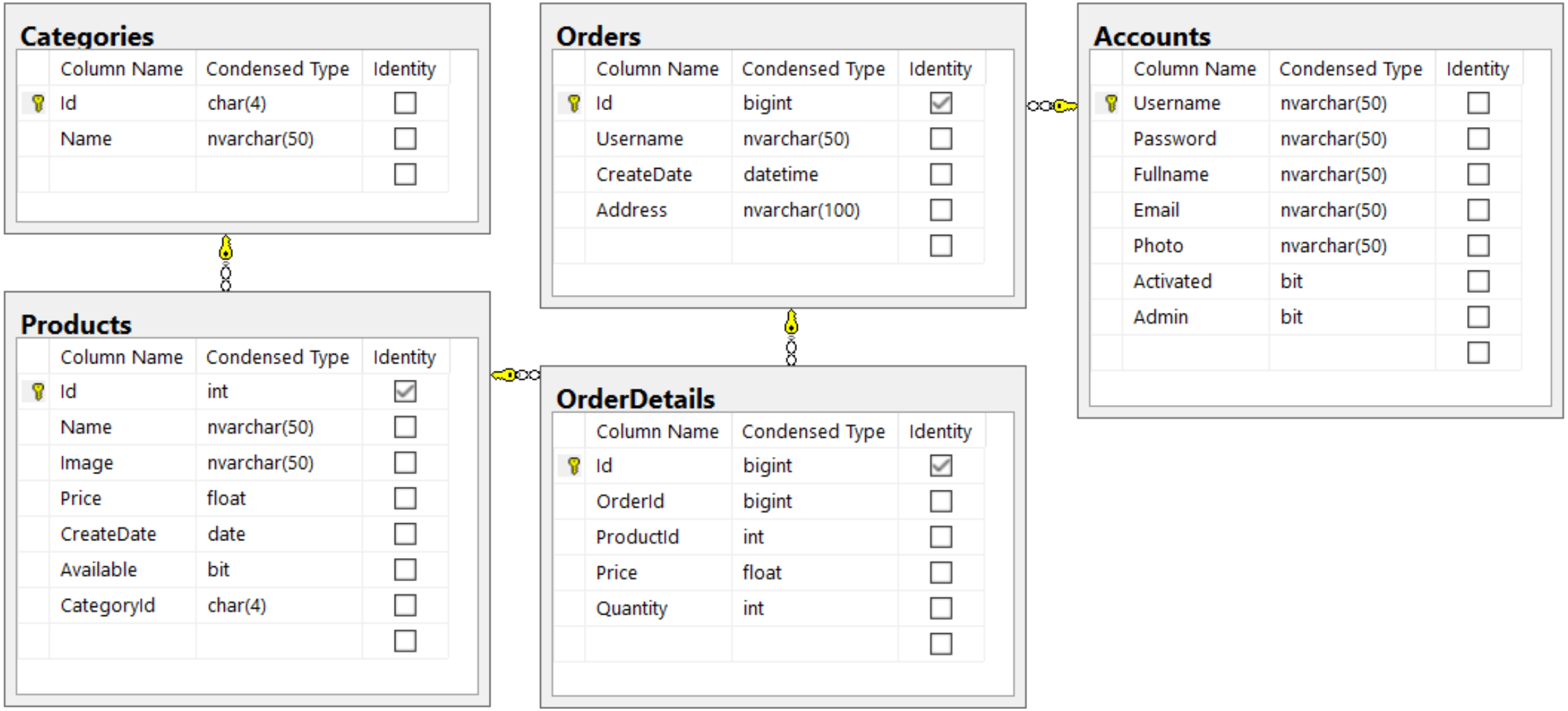
spring.jpa.hibernate.dialect=<dialect>
spring.jpa.hibernate.ddl-auto=<**none**|create|create-drop|validate|update>

spring.jpa.show-sql=<true|**false**>
spring.jpa.properties.hibernate.format_sql=<true|**false**>

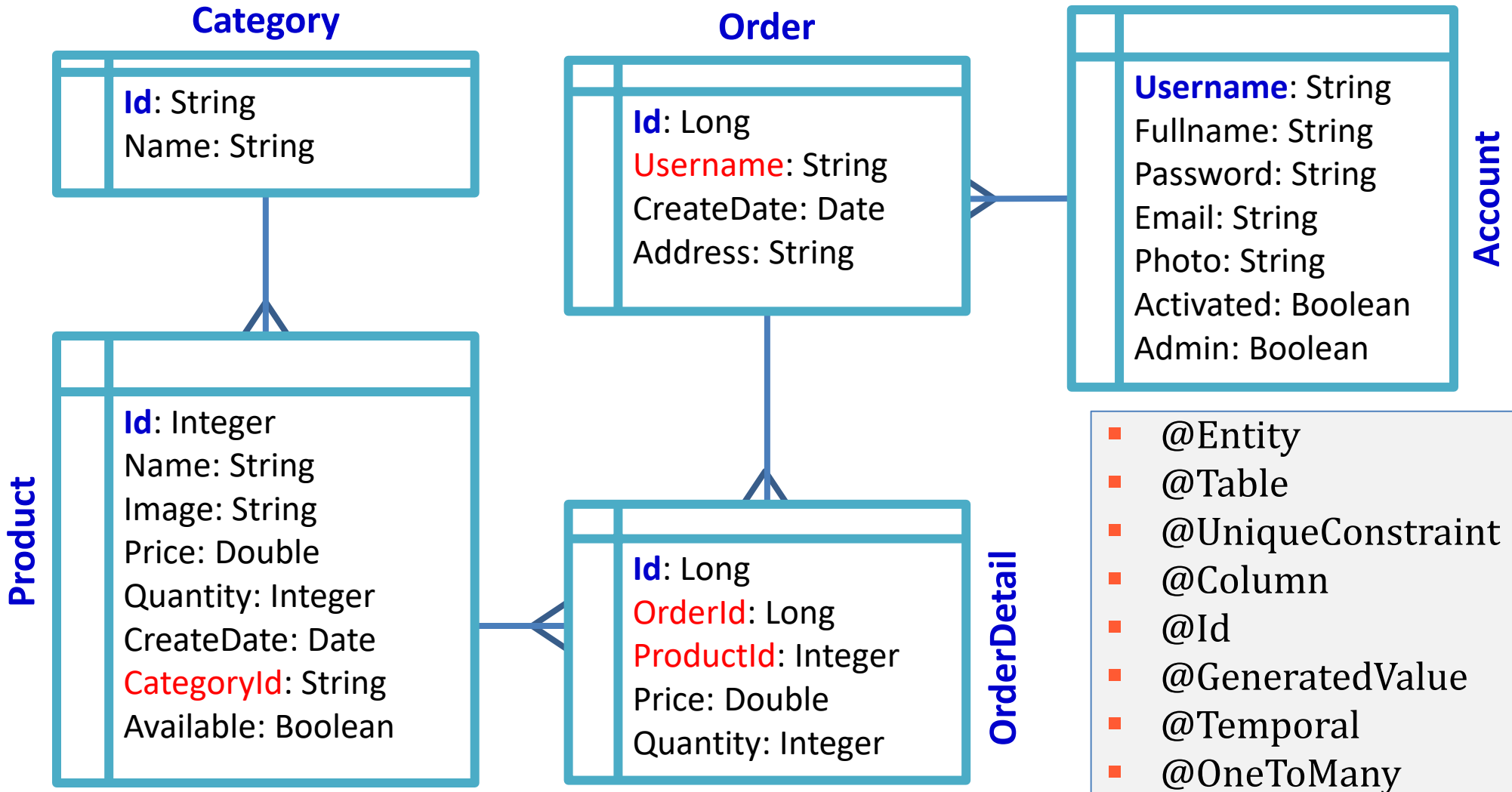
Building Entity Class

FPT Education
FPT POLYTECHNIC

WEBSHOP DATABASE INTRODUCTION



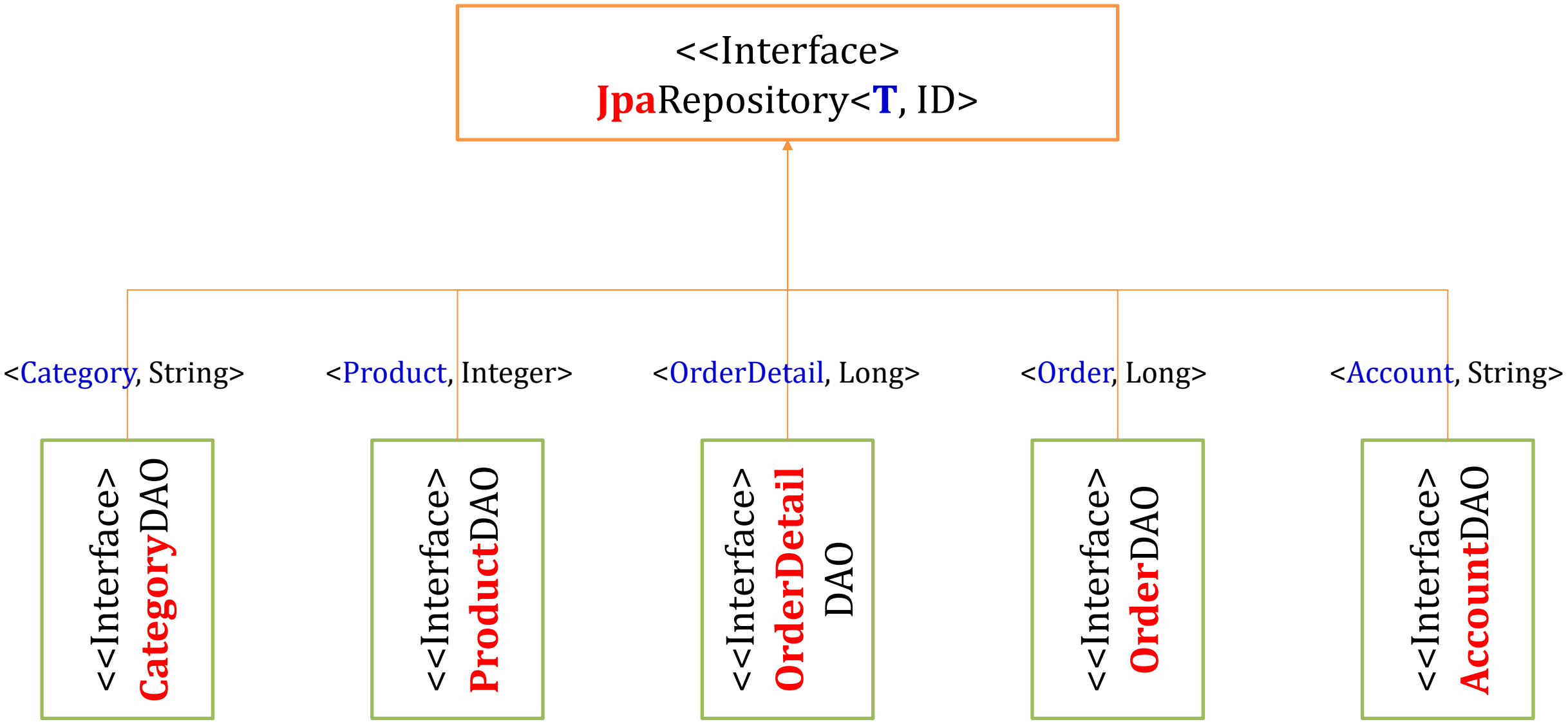
Entity Relational Diagram



- @Entity
- @Table
- @UniqueConstraint
- @Column
- @Id
- @GeneratedValue
- @Temporal
- @OneToMany
- @ManyToOne
- @JoinColumn

Building WebShop DAO

WEBSHOP DAO CLASS DIAGRAM



WebShop DAO Implementation

```
public interface CategoryDAO extends JpaRepository<Category, String>{}

public interface ProductDAO extends JpaRepository<Product, Integer>{}

public interface AccountDAO extends JpaRepository<Account, String>{}

public interface OrderDAO extends JpaRepository<Order, Long>{}

public interface OrderDetailDAO extends JpaRepository<OrderDetail, Long>{}
```

Building CRUD Application

Category Management

Category Id:

Category Name:

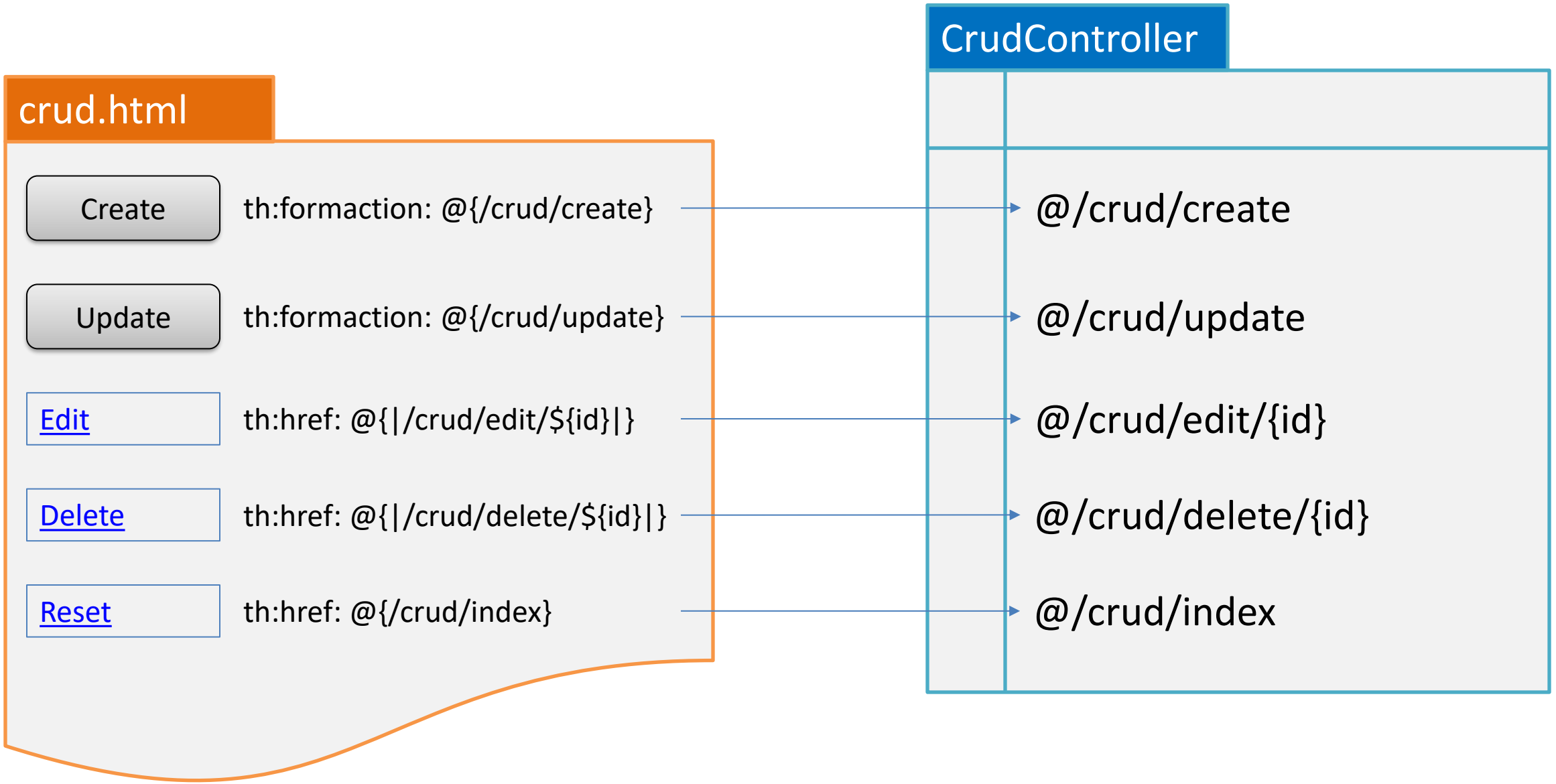
Create Update Delete Reset

FORM

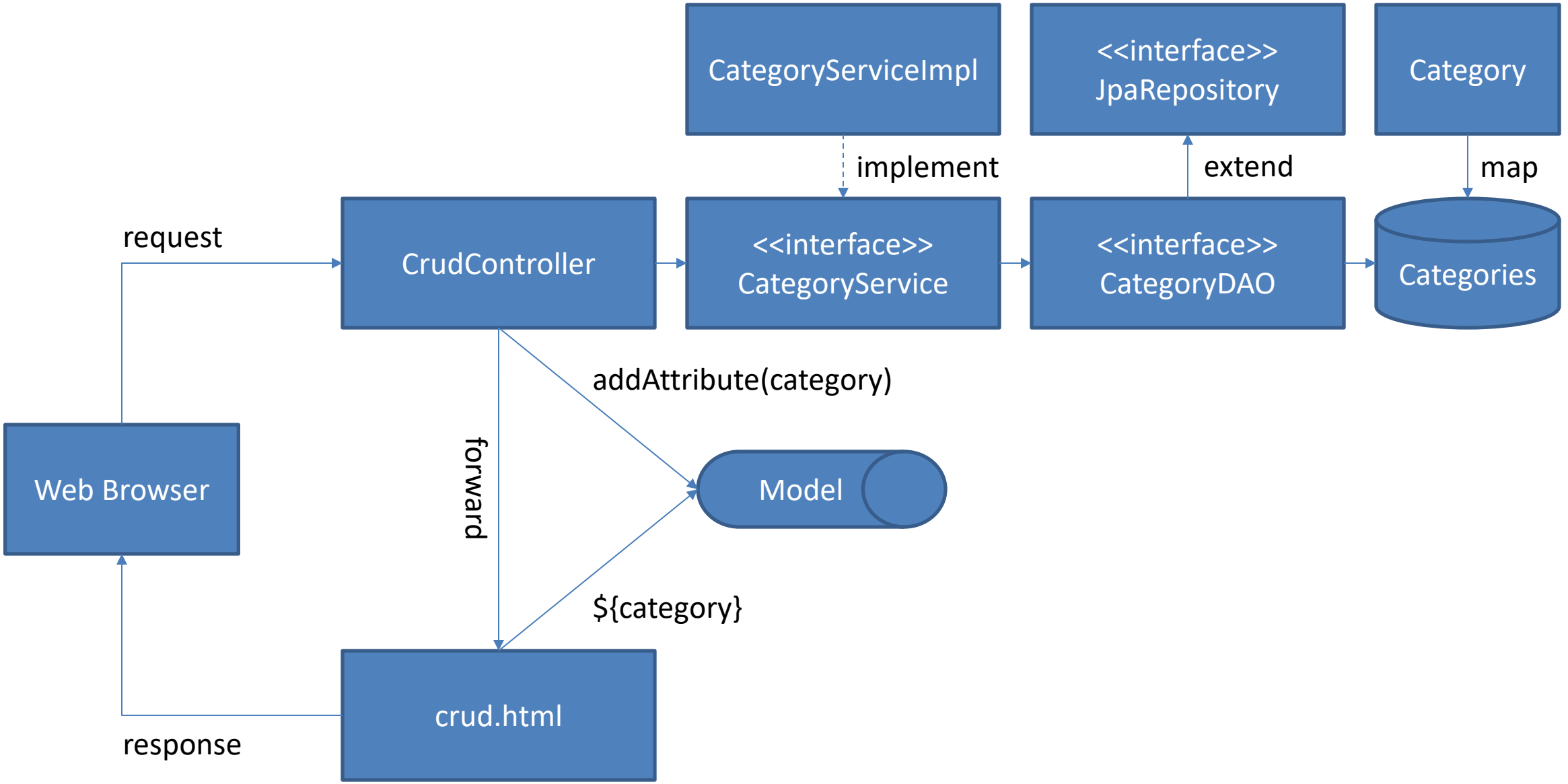
No.	Id	Name	
1	CAT01	Điện thoại di động	Edit Delete
2	CAT02	Máy tính	Edit Delete
3	CAT03	TV thông minh	Edit Delete
4	CAT04	Tủ lạnh	Edit Delete
5	CAT05	Quạt máy AI	Edit Delete
6	CAT06	Máy tính để bàn	Edit Delete
7	CAT07	Máy hút bụi	Edit Delete
8	CAT08	Thiết bị Wifi 5G	Edit Delete

TABLE

REQUEST HANDLING



CRUD Application Architecture



Component Check List

Thành phần	Thể loại	Status
Categories	Table	✓ Sẵn sàng
Category	Entity Class	✓ Sẵn sàng
CategoryDAO	Interface	✓ Sẵn sàng
CategoryService	Interface	✓ Sẵn sàng
CategoryServiceImpl	@Service	✓ Sẵn sàng
CrudController	Controller Class	⊕ Chưa có, xây mới
crud.html	Template	⊕ Chưa có, xây mới

CrudController Structure

```
@Controller
public class CrudController {
        @Autowired
        CategoryService categoryService;

        @RequestMapping("/crud/index")
        public String index(Model model) {...}

        @RequestMapping("/crud/edit/{id}")
        public String edit(Model model, @PathVariable("id") String id) {...}

        @RequestMapping("/crud/delete/{id}")
        public String delete(Model model, @PathVariable("id") String id) {...}

        @RequestMapping("/crud/create")
        public String create(Model model, @ModelAttribute("form") Category category) {...}

        @RequestMapping("/crud/update")
        public String update(Model model, @ModelAttribute("form") Category category) {...}
}
```

CrudController Methods

```
@Autowired
CategoryService categoryService;

@RequestMapping("/crud/index")
public String index(Model model) {
        model.addAttribute("form", new Category());
        return "/crud/index";
}

@RequestMapping("/crud/edit/{id}")
public String edit(Model model, @PathVariable("id") String id) {
        model.addAttribute("form", categoryService.findById(id));
        return "/crud/index";
}
```

CRUDCONTROLLER METHODS

```
@RequestMapping("/crud/delete/{id}")
public String delete(Model model, @PathVariable("id") String id) {
        categoryService.deleteById(id);
        return "redirect:/crud/index";
}

@RequestMapping("/crud/create")
public String create(Model model, @ModelAttribute("form") Category category) {
        categoryService.create(category);
        return "redirect:/crud/index";
}

@RequestMapping("/crud/update")
public String update(Model model, @ModelAttribute("form") Category category) {
        categoryService.update(category);
        return "redirect:/crud/edit/"+category.getId();
}
```

PRODUCT.HTML - FORM

```
<form th:object="${form}" method="post">
        <label>Category Id:</label>
        <input th:field="*{id}" type="text">
         <br>
        <label>Category Name:</label>
        <input th:field="*{name}" type="text">
         <hr>
        <button formaction="/crud/create">Create</button>
        <button formaction="/crud/update">Update</button>
        <a th:href="@{|/crud/delete/*{id}|}">Delete</a>
        <a href="/crud/index">Reset</a>
</form>
```

PRODUCT.HTML - TABLE

```
<table border="1" style="width:100%">
    <thead> <tr> <th>No.</th> <th>Id</th> <th>Name</th> <th></th> </tr> </thead>
    <tbody>
        <tr th:each="e, s : ${@categoryServive.findAll()}">
            <td th:text="${s.count}">No.</td>
            <td th:text="${e.id}">Id</td>
            <td th:text="${e.name}">Name</td>
            <td>
                <a th:href="@{|/crud/edit/${e.id}|}">Edit</a>
                <a th:href="@{|/crud/delete/${e.id}|}">Delete</a>
            </td>
        </tr>
    </tbody>
</table>
```

FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

SẮP XẾP VÀ PHÂN TRANG

SLIDE 6.2

www.poly.edu.vn

Sort ang Pagination

SẮP XẾP

```
Direction direction = Direction.DESC;
String property = "name";
Sort sort = Sort.by(direction, property);
List<Category> list = dao.findAll(sort);
list.forEach(c -> System.out.println(">> " + c.getName()));
```

- Direction
  - Direction.ASC
  - Direction.DESC
- Sort
  - Sort.by(String…properties)
  - Sort.by(Direction direction, String…properties)

PHÂN TRANG

```
int pageNumber = 0;
int pageSize = 8;
Sort sort = Sort.by(Direction.DESC, "name");
Pageable pageable = PageRequest.of(pageNumber, pageSize, sort);

Page<Category> page = dao.findAll(pageable);

System.out.println("1. PAGE ELEMENTS:");
List<Category> list = page.getContent();
list.forEach(c -> System.out.println(">> " + c.getName()));
System.out.println("2. PAGINATION INFO");
System.out.println(">> Page Number: " + page.getNumber());
System.out.println(">> Page Size: " + page.getSize());
System.out.println(">> Number of Elements: " + page.getNumberOfElements());
System.out.println(">> Total Elements:  " + page.getTotalElements());
System.out.println(">> Total Pages: " + page.getTotalPages());
```

- Pageable
- PageRequest
  - PageRequest.of()
- Page
  - getContent()
  - getNumber()
  - getSize()
  - getNumberOfElements()
  - getTotalElements()
  - getTotalPages()

SẮP XẾP VÀ PHÂN TRANG

- Pageable: quản lý thông tin yêu cầu phân trang
  - Pageable.ofSize(int pageSize)
  - Pageable.unpaged()
- PageRequest: lớp tiện ích hỗ trợ tạo Pageable
  - PageRequest.of(int number, int size)
  - PageRequest.of(int number, int size, Sort sort)
  - PageRequest.of(int number, int size, Direction direction, String…properties)

PAGE<T> API – KẾT QUẢ PHÂN TRANG

- Nội dung trang
  - getContent(): List<T>
  - toList(): List<T>
- Các thông số phân trang
  - getNumber(): int
  - getSize(): int
  - getTotalPages(): int
  - getNumberOfElements(): int
  - getTotalElements(): int
- Kiểm tra
  - isEmpty() : boolean
  - isFirst() : boolean
  - isLast() : boolean
  - hasContent() : boolean
  - hasNext() : boolean
  - hasPrevious() : boolean
- Điều hướng phân trang
  - getPageable(): Pageable
  - previousPageable() : Pageable
  - previousOrFirstPageable() : Pageable
  - nextPageable() : Pageable
  - nextOrLastPageable() : Pageable

PHÂN TRANG SẢN PHẨM

PAGINATION APPLICATION PROTOTYPE

Pagination Demo

No.	Id	Name
11	1228	Điện thoại 11
12	1229	Điện thoại 12
13	1230	Điện thoại 13
14	1231	Điện thoại 14
15	1232	Điện thoại 15

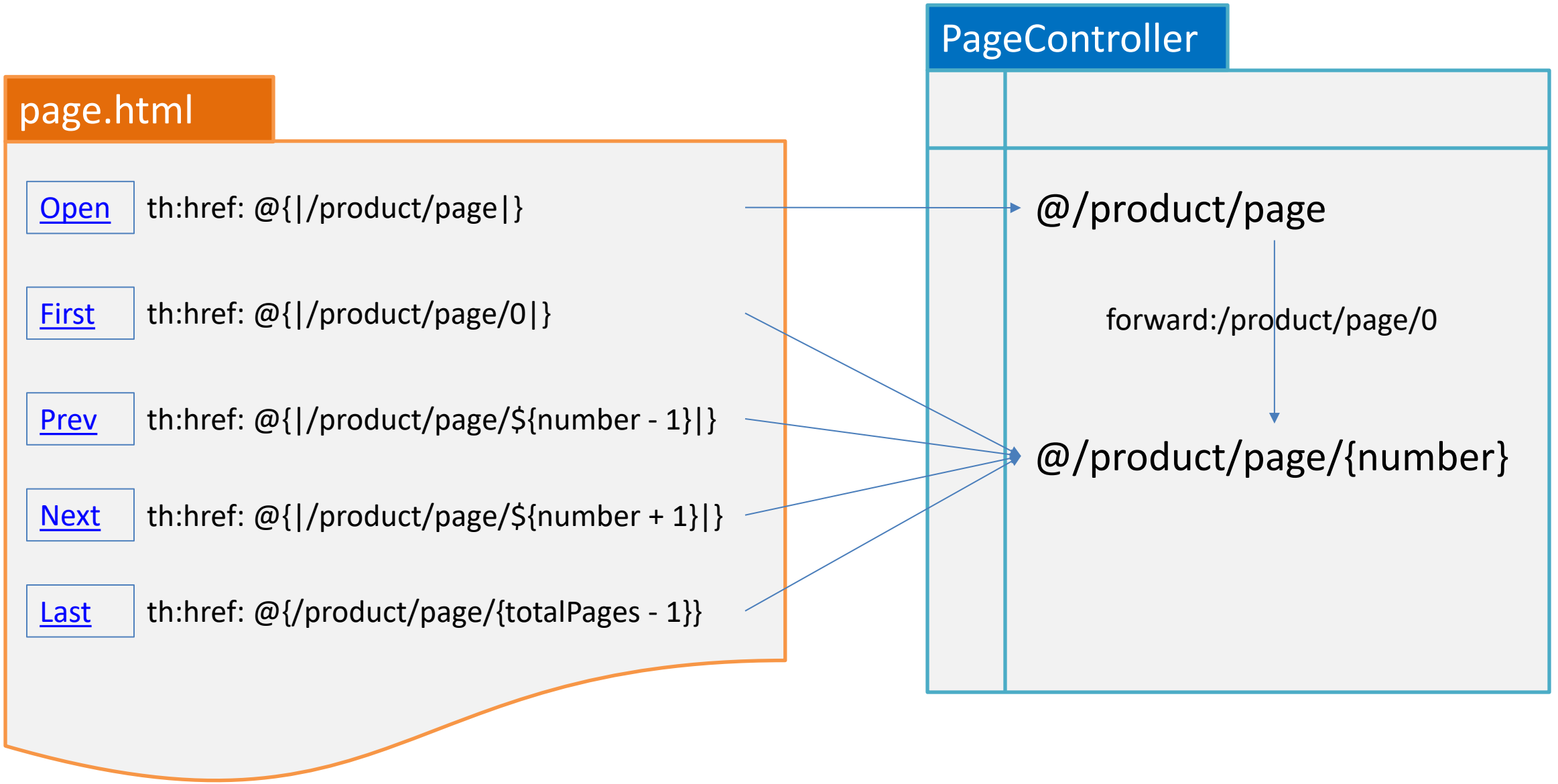
First | Previous | Next | Last

- Page Number: *2*
- Page Size: *5*
- Total Pages: *42*
- Number of Elements: *5*
- Total Elements: *209*

Liệt kê sản phẩm trang số #i

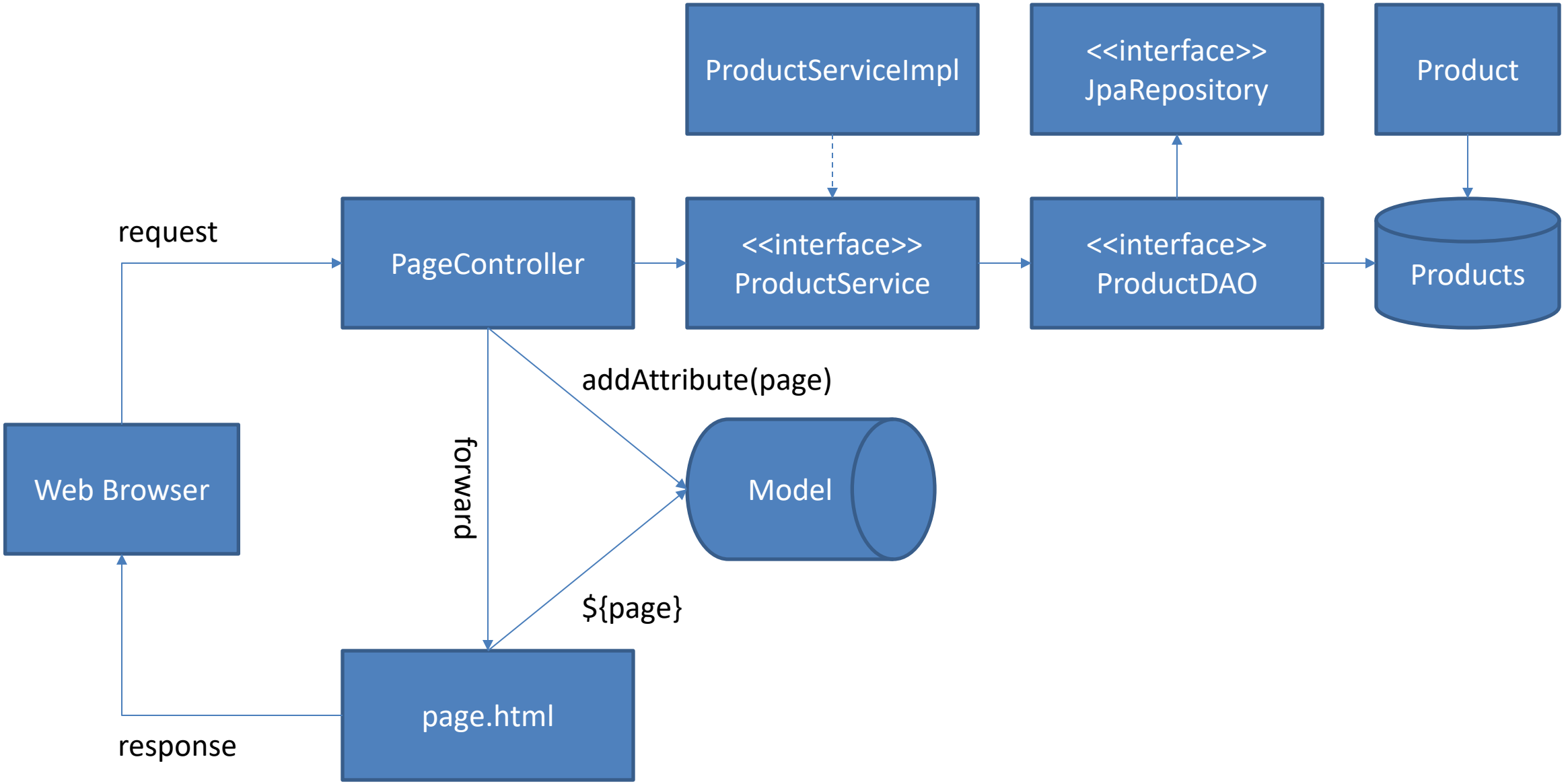
Điều hướng và thông tin trang #i

REQUEST HANDLING



FPT Education
FPT POLYTECHNIC

PAGINATION APPLICATION ARCHITECTURE



Component Check List

Thành phần	Thể loại	Status
Products	Table	✓ Sẵn sàng
Product	Entity Class	✓ Sẵn sàng
ProductDAO	Interface	✓ Sẵn sàng
ProductService	Interface	⊕ Chưa có, xây mới
ProductServiceImpl	@Service	⊕ Chưa có, xây mới
PageController	Controller Class	⊕ Chưa có, xây mới
page.html	Template	⊕ Chưa có, xây mới

PRODUCTSERVICE & PRODUCTSERVICEIMPL

```
public interface ProductService  extends JpaRepository<Product, Integer>{
        Page<Product> findPage(int number);
}
```

```
@Service
public class ProductServiceImpl implements ProductService {
        @Autowired
        ProductDAO dao;
        @Override
        public Page<Product> findPage(int number) {
                int size = 5;
                return dao.findAll(PageRequest.of(number, size));
        }
}
```

PAGECONTROLLER

```
@Controller
public class PageController {
	@Autowired
	ProductService productService;

	@RequestMapping("/product/page")
	public String paginate() {
		return "forward:/product/page/0";
	}

	@RequestMapping("/product/page/{number}")
	public String paginate(Model model, @PathVariable("number") Integer number) {
		Page<Product> page = productService.findPage(number);
		model.addAttribute("page", page);
		return "/demo/product/page.html";
	}
}
```

PRODUCT.HTML – LIỆT KÊ SẢN PHẨM CỦA TRANG

```
<h3>Pagination Demo</h3>

<table border="1" style="width:100%">
    <thead><tr> <th>No.</th> <th>Id</th> <th>Name</th></tr></thead>
    <tbody>
        <tr th:each="e, s: ${page.content}">
            <td th:text="${s.count + page.number * page.size}"></td>
            <td th:text="${e.id}"></td>
            <td th:text="${e.name}"></td>
        </tr>
    </tbody>
</table>
```

PRODUCT.HTML – ĐIỀU HƯỚNG VÀ THÔNG TIN TRANG

```
<hr>
<nav>
    <a th:href="@{|/product/page/${0}|}">First</a> |
    <a th:href="@{|/product/page/${page.number - 1}|}">Previous</a> |
    <a th:href="@{|/product/page/${page.number + 1}|}">Next</a> |
    <a th:href="@{|/product/page/${page.totalPages - 1}|}">Last</a>
</nav>
<hr>
<ul>
   <li>Page Number: <i th:text="${page.number}"></i></li>
   <li>Page Size: <i  th:text="${page.size}"></i></li>
   <li>Total Pages: <i  th:text="${page.totalPages}"></i></li>
   <li>Number of Elements: <i  th:text="${page.numberOfElements}"></i></li>
   <li>Total Elements: <i  th:text="${page.totalElements}"></i></li>
</ul>
```

SUMMARY

- ☑ JpaRepository API
  - ☑ Giới thiệu CSDL mẫu J5Shop
  - ☑ Tạo các lớp thực thể
  - ☑ Sơ đồ phân cấp thừa kế JpaRepository
  - ☑ Tạo các DAO làm việc với J5Shop
  - ☑ Xây dựng ứng dụng CRUD
- ☑ Sắp xếp và phân trang với JpaRepository
  - ☑ Truy vấn có sắp xếp
  - ☑ Truy vấn có phân trang
  - ☑ Sắp xếp và phân trang sản phẩm


FPT Education
FPT POLYTECHNIC
Thank you