Cách xây dựng website spring boot với JSP, sử dụng JPA (Inteliji IDEA)

**1. Cấu hình:**

- File **application.properties**:

+ Cấu hình kết nối database

spring.datasource.url=jdbc:mysql://${MYSQL\_HOST:localhost}:3306/test\_db  
spring.datasource.username=root  
spring.datasource.password=  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

+ Cấu hình View

spring.mvc.view.prefix=/views/  
spring.mvc.view.suffix=.jsp

các file jsp sẽ nằm trong thư mục main/webapp/views/

+ Cấu hình khác:

spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.format\_sql=true

- File **pom.xml**: các thư viện cần thiết

+ Thư viện spring

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>

+ Thư viện JPA

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

+ Thư viện jstl

<dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
</dependency>

+ Thư viện Tomcat

<dependency>  
 <groupId>org.apache.tomcat</groupId>  
 <artifactId>tomcat-jasper</artifactId>  
 <version>9.0.58</version>  
</dependency>

+ Thư viện driver connect MySql

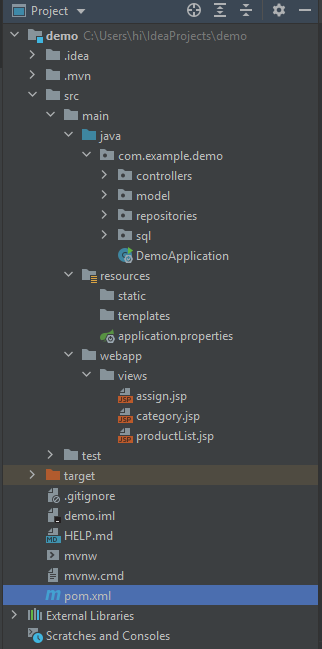
<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
</dependency>

+ Thư viện khác (tùy chọn)

<dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
</dependency>

**2. Cấu trúc code:**

Lưu ý: phải tạo thêm thư mục webapp, mặc định khi tạo project ở Intelliji IDEA ko có thư mục này



**3. Giao tiếp giữa view và controller**

a. Truyền data từ Controller qua View sử dụng ModelMap, vd:

@GetMapping("getProductByCategoryId/{categoryId}")  
public String getProductByCategoryId(ModelMap model, @PathVariable String categoryId){  
 var products = repository.findProductsByCategoryId(categoryId);  
 model.addAttribute("products", products);  
 return "productList";  
}

Tại View thì truy xuất data từ Controller gửi qua ${}

<tbody>  
 <**c:forEach** var="item" items="${products}">  
 <tr>  
 <td>${item.productId}</td>  
 <td>${item.categoryId}</td>  
 <td>${item.productName}</td>  
 <td>${item.price}</td>  
 <td>${item.description}</td>  
 <td><a href="/product/changeCategory/${item.productId}">Change category</a></td>  
 </tr>  
 </**c:forEach**>  
</tbody>

b. Truyền data từ View sang Controller: sử dụng ModelAttrbute

- View: sử dụng sform của spring

<**sform:form** method="post" action="/product/changeCategory/${product.productId}" **modelAttribute**="product">  
 <**sform:select** path="categoryId">  
 <**c:forEach** items="${categories}" var="item">  
 <option value="${item.categoryId}">${item.categoryName}</option>  
 </**c:forEach**>  
 </**sform:select**>  
 <br />  
 <input type="submit" value="Save" />  
</**sform:form**>

- Controller:

@PostMapping("changeCategory/{productId}")  
public String changeCategory(@ModelAttribute("product") Product product){  
 var p = repository.findById(product.getProductId());  
 if (p.isPresent()){  
 var pro = p.get();  
 pro.setCategoryId(product.getCategoryId());  
 repository.save(pro);  
 }  
  
 return "redirect:/product/getProductByCategoryId/" + product.getCategoryId();  
}

**4. Flow update data:**

- Step 1: request GET, show ra UI cho user:

@GetMapping("changeCategory/{productId}")  
public String changeCategory(ModelMap model, @PathVariable String productId){  
 model.addAttribute("categories", crepository.findAll());  
 model.addAttribute("product", repository.findById(productId).get());  
 return "assign";  
}

* Step 2: sau khi user bấm submit ở form, sẽ gọi đến POST để save change
* @PostMapping("changeCategory/{productId}")  
  public String changeCategory(@ModelAttribute("product") Product product){  
   var p = repository.findById(product.getProductId());  
   if (p.isPresent()){  
   var pro = p.get();  
   pro.setCategoryId(product.getCategoryId());  
   repository.save(pro);  
   }  
    
   return "redirect:/product/getProductByCategoryId/" + product.getCategoryId();  
  }

**5. Sử dụng JPA và MySql**

Không cần cấu hình gì nhiều, MySql và JPA sẽ tự nhận database, với JPA chỉ cần tạo interface kế thừa từ JPA và khai báo thêm hàm nếu cần

public interface ProductRepository extends CrudRepository<Product, String> {  
 Iterable<Product> findProductsByCategoryId(String categoryId);  
}

**6. Validation data nhập vào**

- Add thêm vào file pom thư viện

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-validation</artifactId>  
 <version>2.6.3</version>  
</dependency>

* Trong class entity cần validation thêm các annotation vào:
* @NotNull  
  @Size(min = 3, max = 30)  
  @NotBlank(message = "Name can't be null")  
  private String productName;
* Trong Controller ở hàm POST, sử dụng @Valid ở ModelAttribute và BindingResult:

@PostMapping("insert")  
public String insert(ModelMap model,  
 **@Valid** @ModelAttribute Product product,  
 BindingResult result){  
 if (result.hasErrors()){  
 return "insert";  
 }  
  
 try {  
 repository.save(product);  
 return "redirect:/product/getProductByCategoryId/" + product.getCategoryId();  
 }catch (Exception e){  
 model.addAttribute("error", e.getMessage());  
 return "insert";  
 }  
}

**7. Flow khi insert thêm 1 bản ghi**

- Step 1:

+ Controller: gọi đến UI, nhớ thêm vào 1 thuộc tính ở Model gọi là ModelAttribute để UI nhập data vào object này

@GetMapping("insert")  
public String insert(ModelMap model){  
 model.addAttribute("product", new Product());  
 model.addAttribute("categories", crepository.findAll());  
 return "insert";  
}

+ View: sử dụng model attribute ở Controller truyền qua và nhập data vào đây

<**sform:form** method="post" action="/product/insert" modelAttribute="product">  
 <**sform:input** path="productId" type="text" placeholder="Enter Id" />  
 <**sform:errors** path="productId" cssClass="red" />  
 <br />  
 <**sform:input** path="productName" type="text" placeholder="Enter name" />  
 <**sform:errors** path="productName" cssClass="red" />  
 <br />  
 <**sform:input** path="price" type="number" placeholder="Enter price" />  
 <br />  
 <**sform:textarea** rows="3" path="description"  
 type="text" placeholder="Enter description" />  
 <br />  
 <**sform:select** path="categoryId">  
 <**c:forEach** items="${categories}" var="item">  
 <option value="${item.categoryId}">${item.categoryName}</option>  
 </**c:forEach**>  
 </**sform:select**>  
 <br />  
 <input type="submit" value="Save" />  
</**sform:form**>

* Step 2:

+ View: sau khi user nhập đủ data vào và bấm submit sẽ gọi đến hàm POST của Controller

+ Controller:

@PostMapping("insert")  
public String insert(ModelMap model,  
 @Valid @ModelAttribute Product product,  
 BindingResult result){  
 if (result.hasErrors()){  
 return "insert";  
 }  
  
 try {  
 repository.save(product);  
 return "redirect:/product/getProductByCategoryId/" + product.getCategoryId();  
 }catch (Exception e){  
 model.addAttribute("error", e.getMessage());  
 return "insert";  
 }  
}

**8. Delete 1 bản ghi**

- Có thể dùng thẻ <a> để gọi request đến hàm xóa

<td><a href="/product/delete/${item.productId}">Delete product</a></td>

* Dùng thẻ form của spring để gọi hàm POST
* <**sform:form** method="post" action="/product/delete/${product.productId}"  
   onsubmit="return confirm('Do you want delete?') ? true : false;">  
   <input type="submit" value="Delete" />  
  </**sform:form**>
* Controller:

@RequestMapping("delete/{productId}")  
public String delete(@PathVariable String productId){  
 var product = repository.findById(productId);  
 repository.deleteById(productId);  
 return "redirect:/product/getProductByCategoryId/" + product.get().getCategoryId();  
}