Spring Boot web application with JDBC (Part 2)

Overview

This part adds functionality to insert new products into the database via a web form. The application lifecycle is as follows:

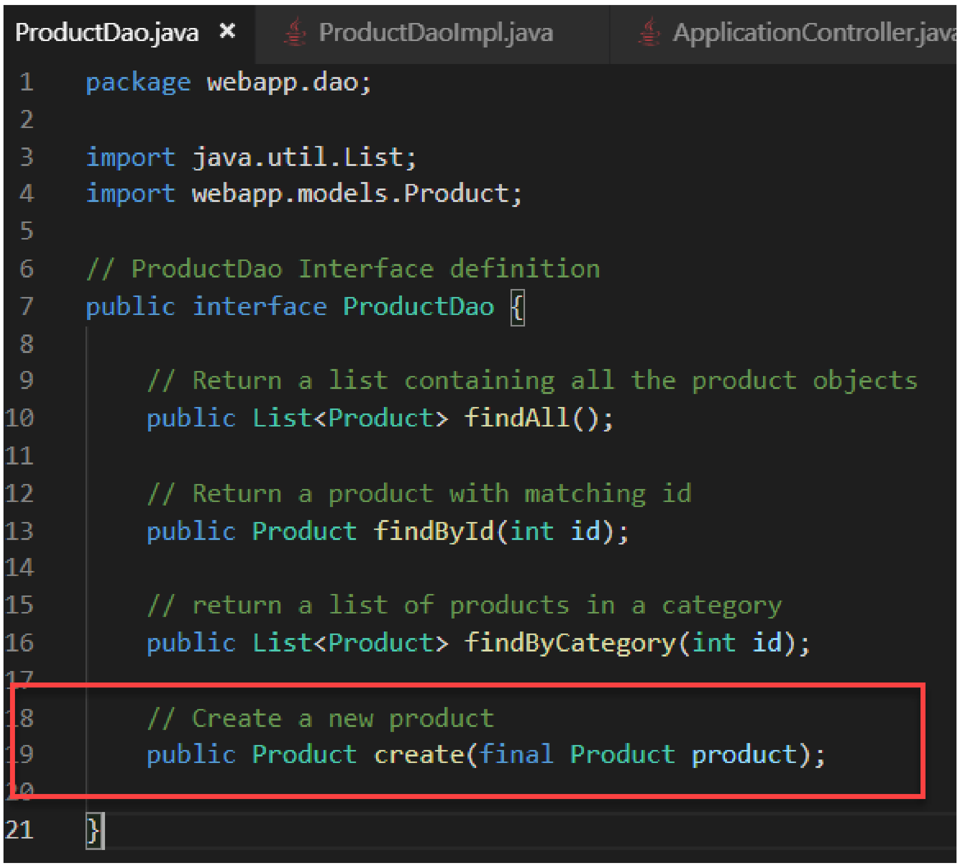
1. Display the form (`/newProduct`)
2. Fill in form values and `submit`
3. Insert new product into database.

1. Add DAO method

Add a method to `ProductDao` and its implementation to perform the insert

First `ProductDao` (the interface)

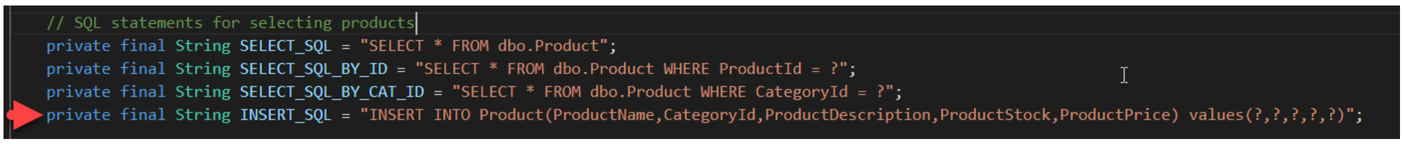
The create method accepts a product parameter (which will be empty) and returns a new product after it has been created.



```java
package webapp.dao;

import java.util.List;
import webapp.models.Product;

// ProductDao Interface definition
public interface ProductDao {

    // Return a list containing all the product objects
    public List<Product> findAll();

    // Return a product with matching id
    public Product findById(int id);

    // return a list of products in a category
    public List<Product> findByCategory(int id);

    // Create a new product
    public Product create(final Product product);

}
```

The product implementation, **ProductDaoImpl.java**

The SQL statement required for insert



```
// SQL statements for selecting products
private final String SELECT_SQL = "SELECT * FROM dbo.Product";
private final String SELECT_SQL_BY_ID = "SELECT * FROM dbo.Product WHERE ProductId = ?";
private final String SELECT_SQL_BY_CAT_ID = "SELECT * FROM dbo.Product WHERE CategoryId = ?";
private final String INSERT_SQL = "INSERT INTO Product(ProductName,CategoryId,ProductDescription,ProductStock,ProductPrice) values(?,?,?,?,?)";
```

Copy this text:

```
private final String INSERT_SQL = "INSERT INTO
Product(ProductName,CategoryId,ProductDescription,ProductStock,ProductPrice) values(?,?,?,?,?)";
```

The **create** method which will fill in the values and execute the query (**see code comments**)

```
    // Create a new product
    public Product create(final Product product) {

        // A new Primary key (identity) value will be generated by the database on insert
        // This value is retrieved using KeyHolder
        KeyHolder holder = new GeneratedKeyHolder();
        jdbcTemplate.update(new PreparedStatementCreator() {
            @Override
            // Prepared statement replaces ? parameters with values
            // Create the statement and connect
            public PreparedStatement createPreparedStatement(Connection connection) throws SQLException {
                PreparedStatement ps = connection.prepareStatement(INSERT_SQL, Statement.RETURN_GENERATED_KEYS);
                // Set each parameter by index (of ? in SQL) and value
                ps.setString(1, product.getProductName());
                ps.setInt(2, product.getCategoryId());
                ps.setString(3, product.getProductDescription());
                ps.setInt(4, product.getProductStock());
                ps.setDouble(5, product.getProductPrice());

                // return the completed statement
                return ps;
            }
        }, holder);

        // Get the new id and assign it to the new product object
        int newProductId = holder.getKey().intValue();
        product.setProductId(newProductId);

        // Return the newly created product
        return product;
    }
```

2. A Controller Method to load the add Product page

This method will load a view named **newProduct**. The view requires an empty Product object and also a list of categories for the form:

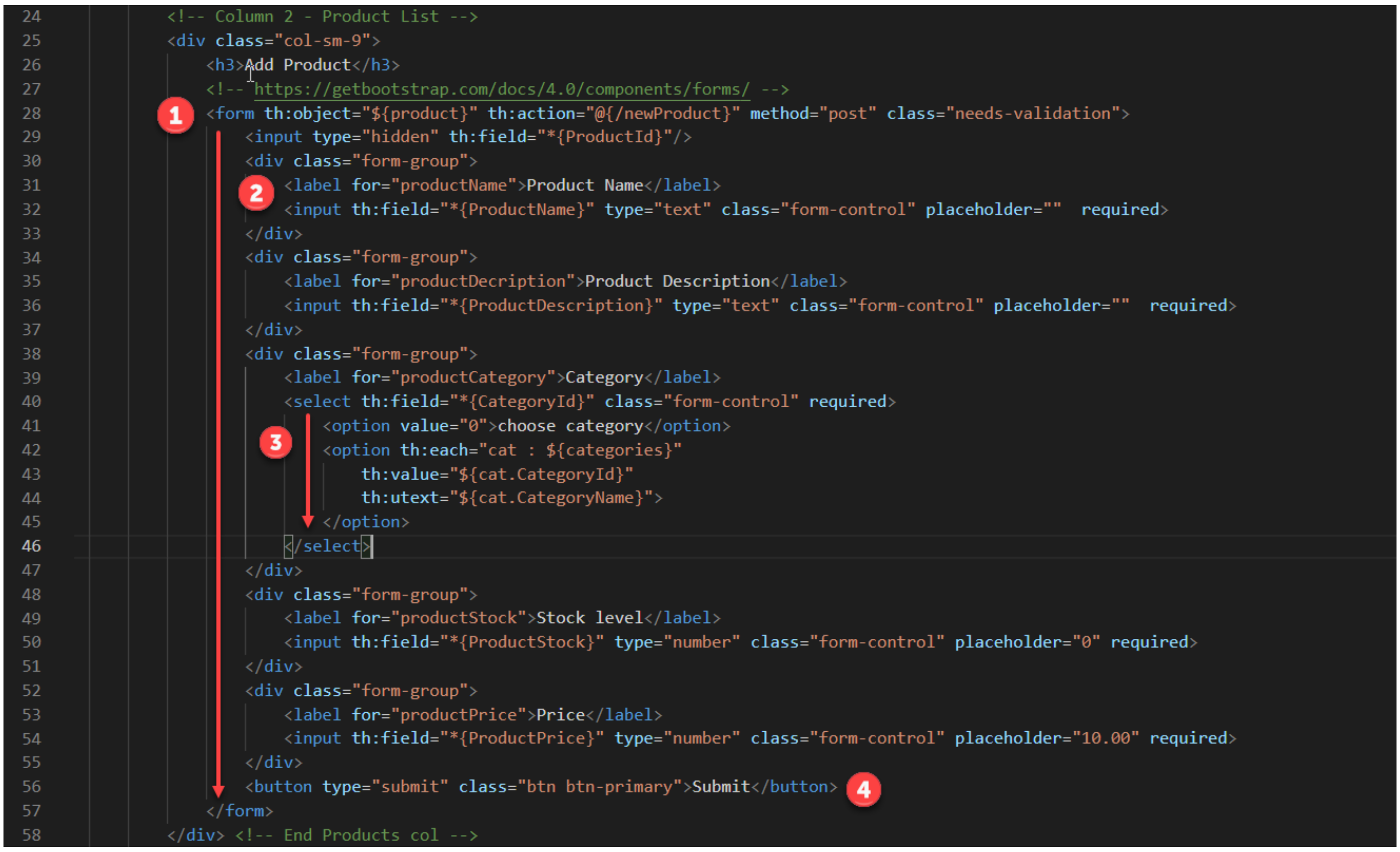
```java
    // The about page will be accessed using /newProduct from the browser
    @RequestMapping(value = "/newProduct", method = RequestMethod.GET)
    public String newProduct(Model model) {

        // add empty Product to the model
        model.addAttribute("product", new Product());

        // Get a list of categories and add to the model
        List<Category> categories = categoryData.findAll();
        model.addAttribute("categories", categories);

        // Return the view
        return "newProduct";
    }
```

3. The add product form

Here's the form part (see below for the full html for the **newProduct** page.

This is an HTML form – contained in a **`<form>`** element. The Thymeleaf **`th:`** syntax is used to link the form fields to the **product** object which was passed as a parameter. It is also styled using Bootstrap 4 (using **`class=`**…)

1. The form properties
    a. **`th:object`** associates the form with **product**
    b. **`th:action`** defines where the form will be submitted (the controller will look after this)
    c. Method defines that the form will be sent via HTTP POST (parameters in request body)
    d. Needs-validation will check required fields in browser before submitting.
2. Each form field consists of a **label** and an input wrapped in a Bootstrap div (**form-group**)
    a. **`th:field`** associates the product field and type defines the data input type expected
3. The HTML **`<select>`** element provides a list of options. In this case, categories. It is also assigned to the product object **CategoryId**
4. The submit button sends the values entered to the address defined by the form **action** attribute.



```html
        <!-- Column 2 - Product List -->
        <div class="col-sm-9">
            <h3>Add Product</h3>
            <!-- https://getbootstrap.com/docs/4.0/components/forms/ -->
            <form th:object="${product}" th:action="@{/newProduct}" method="post" class="needs-validation">
                <input type="hidden" th:field="*{ProductId}"/>
                <div class="form-group">
                    <label for="productName">Product Name</label>
                    <input th:field="*{ProductName}" type="text" class="form-control" placeholder=""  required>
                </div>
                <div class="form-group">
                    <label for="productDecription">Product Description</label>
                    <input th:field="*{ProductDescription}" type="text" class="form-control" placeholder=""  required>
                </div>
                <div class="form-group">
                    <label for="productCategory">Category</label>
                    <select th:field="*{CategoryId}" class="form-control" required>
                        <option value="0">choose category</option>
                        <option th:each="cat : ${categories}"
                            th:value="${cat.CategoryId}"
                            th:utext="${cat.CategoryName}">
                        </option>
                    </select>
                </div>
                <div class="form-group">
                    <label for="productStock">Stock level</label>
                    <input th:field="*{ProductStock}" type="number" class="form-control" placeholder="0" required>
                </div>
                <div class="form-group">
                    <label for="productPrice">Price</label>
                    <input th:field="*{ProductPrice}" type="number" class="form-control" placeholder="10.00" required>
                </div>
                <button type="submit" class="btn btn-primary">Submit</button>
            </form>
        </div> <!-- End Products col -->
```

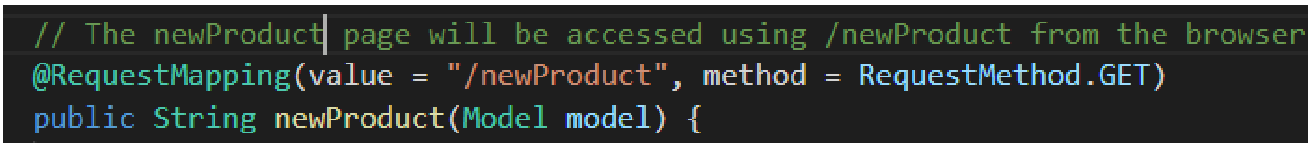
newProduct.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- Link Bootstrap 4 from CDN -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

    <title>Products</title>
</head>
<body>
    <!--/*/ <th:block th:include="fragments/navigation :: navigation"></th:block> /*/-->
    <p></p>
    <p></p>
    <!-- Bootstrap-->
    <div class="container">
        <!-- Content row with 2 columns-->
        <div class="row">
            <!-- Column 1 - Categories List -->
            <div class="col-sm-3">

            </div> <!-- End Categories col-->

            <!-- Column 2 - Product List -->
            <div class="col-sm-9">
                <h3>Add Product</h3>
                <!-- https://getbootstrap.com/docs/4.0/components/forms/ -->
                <form th:object="${product}" th:action="@{/newProduct}" method="post" class="needs-validation">
                    <input type="hidden" th:field="*{ProductId}"/>
                    <div class="form-group">
                        <label for="productName">Product Name</label>
                        <input th:field="*{ProductName}" type="text" class="form-control" placeholder=""  required>
                    </div>
                    <div class="form-group">
                        <label for="productDecription">Product Description</label>
                        <input th:field="*{ProductDescription}" type="text" class="form-control" placeholder=""  required>
                    </div>
                    <div class="form-group">
                        <label for="productCategory">Category</label>
                        <select th:field="*{CategoryId}" class="form-control" required>
                            <option value="0">choose category</option>
                            <option th:each="cat : ${categories}"
                                th:value="${cat.CategoryId}"
                                th:utext="${cat.CategoryName}">
                            </option>
                        </select>
                    </div>
                    <div class="form-group">
                        <label for="productStock">Stock level</label>
                        <input th:field="*{ProductStock}" type="number" class="form-control" placeholder="0" required>
                    </div>
                    <div class="form-group">
                        <label for="productPrice">Price</label>
                        <input th:field="*{ProductPrice}" type="number" class="form-control" placeholder="10.00" required>
                    </div>
                    <button type="submit" class="btn btn-primary">Submit</button>
                </form>
            </div> <!-- End Products col -->
        </div> <!-- End row -->
    </div> <!-- End container -->

    <!-- JavaScript dependencies for Bootstrap 4-->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous"></script>
</body>
</html>
```

5. Handling form submit

Back to the controller to handle the incoming form values (after it is submission).

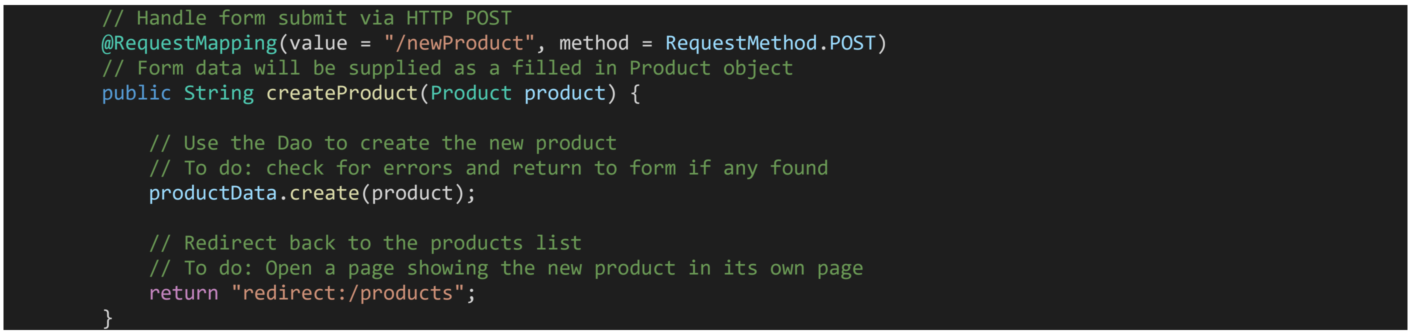
The **`newProduct()`** method already handles requests to **`/newProduct`**



```
// The newProduct page will be accessed using /newProduct from the browser
@RequestMapping(value = "/newProduct", method = RequestMethod.GET)
public String newProduct(Model model) {
```

That version is for requests using the **`GET`** method (e.g. via the browser address bar or a link).

Our new method to handle the form submit will re-use **`/newProduct`** but for **`POST`** requests.



```
    // Handle form submit via HTTP POST
    @RequestMapping(value = "/newProduct", method = RequestMethod.POST)
    // Form data will be supplied as a filled in Product object
    public String createProduct(Product product) {

        // Use the Dao to create the new product
        // To do: check for errors and return to form if any found
        productData.create(product);

        // Redirect back to the products list
        // To do: Open a page showing the new product in its own page
        return "redirect:/products";
    }
```

To be continued…