웹프로그래밍 기초(jsp 3~5강)

kopo20
데이터융합SW과 손영식

디렉티브 태그

JSP 페이지를 어떻게 처리할 것인지를 설정하는 태그

JSP 페이지가 서블릿 프로그램에서 서블릿 클래스로 변환할 때

JSP 페이지와 관련된 정보를 JSP컨테이너에 지시하는 메시지

디렉티브 태그	형식	설명
page	〈%@ page … %〉	JSP 페이지에 대한 정보를 설정합니다.
include	〈%@ include … %〉	JSP 페이지의 특정 영역에 다른 문서를 포함합니다.
taglib	〈%@ taglib … %〉	JSP 페이지에서 사용할 태그 라이브러리를 설정합니다.

page 디렉티브 태그

<%@ page 속성1="값1" [속성2="값2" …] %>

<%와 @ 사이에 공백이 없어야 함

속성	설명	기본 값
language	현재 JSP 페이지가 사용할 프로그래밍 언어를 설정합니다.	java
contentType	현재 JSP 페이지가 생성할 문서의 콘텐츠 유형을 설정합니다.	text/html
pageEncoding	현재 JSP 페이지의 문자 인코딩을 설정합니다.	ISO-8859-1
import	현재 JSP 페이지가 사용할 자바 클래스를 설정합니다.	
session	현재 JSP 페이지의 세션 사용 여부를 설정합니다.	true
buffer	현재 JSP 페이지의 출력 버퍼 크기를 설정합니다.	8KB
autoFlush	출력 버퍼의 동작 제어를 설정합니다.	true
isThreadSafe	현재 JSP 페이지의 멀티스레드 허용 여부를 설정합니다.	true
info	현재 JSP 페이지에 대한 설명을 설정합니다.	
errorPage	현재 JSP 페이지에 오류가 발생했을 때 보여줄 오류 페이지를 설정합니다.	
isErrorPage	현재 JSP 페이지가 오류 페이지인지 여부를 설정합니다.	false
isELIgnored	현재 JSP 페이지의 표현 언어(EL) 지원 여부를 설정합니다.	false
isScriptingEnabled	현재 JSP 페이지의 스크립트 태그 사용 여부를 설정합니다.	

import 속성

현재 JSP 페이지에서 사용할 자바 클래스를 설정하는 데 사용
둘 이상의 자바 클래스를 포함하는 경우 쉼표(,)로 구분하여 연속해서 여러 개의 자바 클래스를 설정
또는 여러 개의 자바 클래스를 각각 별도로 설정할 수도 있음

[import 속성 사용 예: 자바 패키지 java.io.* 설정]

```
<%@ page import="java.io.*" %>
```

[import 속성의 다중 자바 클래스 참조 예: java.io.*와 java.lang.* 패키지 설정]

```
<%@ page import="java.io.*, java.lang.*" %>
```

```
<%@ page import="java.io.*" %>
<%@ page import="java.lang.*" %>
```

include 디렉티브 태그

현재 JSP 페이지의 특정 영역에 외부 파일의 내용을 포함하는 태그
현재 JSP 페이지에 포함할 수 있는 외부 파일
    HTML, JSP, 텍스트 파일
include 디렉티브 태그는 JSP 페이지 어디에서든 선언 가능

```
<%@ include file="파일명" %>
```

```jsp
<%@ page import="java.text.SimpleDateFormat"%>
<%@ page import = "java.util.Date" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<%@ page import="java.util.List" %>
<%@ page import="dto.Product" %>
<%@ page import="dao.ProductRepository" %>
<jsp:useBean id="repository" class="dao.ProductRepository" scope="session" />

<!DOCTYPE html>
<html>

<head>
<meta charset="UTF-8">
<title>welcome.jsp</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

</head>

<body>

<jsp:include page="menu.jsp" />

<%! String greeting = "웹 쇼핑몰에 오신 것을 환영합니다.";
String tagline = "Welcome to Web Market!"; %>

<div class="p-5 bg-secondary text-white text-white">
    <div class ="container">
        <h1 class="text-center display-3"><%= greeting %></h1>
    </div>
</div>

<div class="container">
    <div class="text-center">
        <h3 class="p-3"><%= tagline %></h3>
        <%

        //1초에 한번씩 새로고침 할 때 사용하는 문법입니다...
        response.setIntHeader("Refresh", 1);

        Date today = new Date();

        SimpleDateFormat format = new SimpleDateFormat("hh:mm:ss a");

        out.println("현재 접속 시간 : " + format.format(today));
        %>
    </div>
    <hr>
</div>

<jsp:include page="footer.jsp"></jsp:include>

</body>

</html>
```

welcome.jsp

```jsp
<nav class="navbar navbar-expand navbar-dark bg-dark">
    <div class="container">
        <div class="navbar-header">
        <a class="navbar-brand" href="./welcome.jsp">Home</a>
        <a class="navbar-brand" href="./products.jsp">Products</a>

        </div>
    </div>
</nav>
```

menu.jsp

```jsp
<footer class="container">
    <p>&copy; WebMarket</p>
</footer>
```

footer.jsp

Home Products

웹 쇼핑몰에 오신 것을 환영합니다.

Welcome to Web Market!

현재 접속 시간 : 07:10:28 오전

액션 태그

서버나 클라이언트에게 어떤 행동을 하도록 명령하는 태그

JSP 페이지에서 페이지와 페이지 사이 제어

다른 페이지의 실행 결과 내용을 현재 페이지에 포함

자바 빈즈(JavaBeans) 등의 다양한 기능 제공

XML 형식 <jsp: ... /> 사용

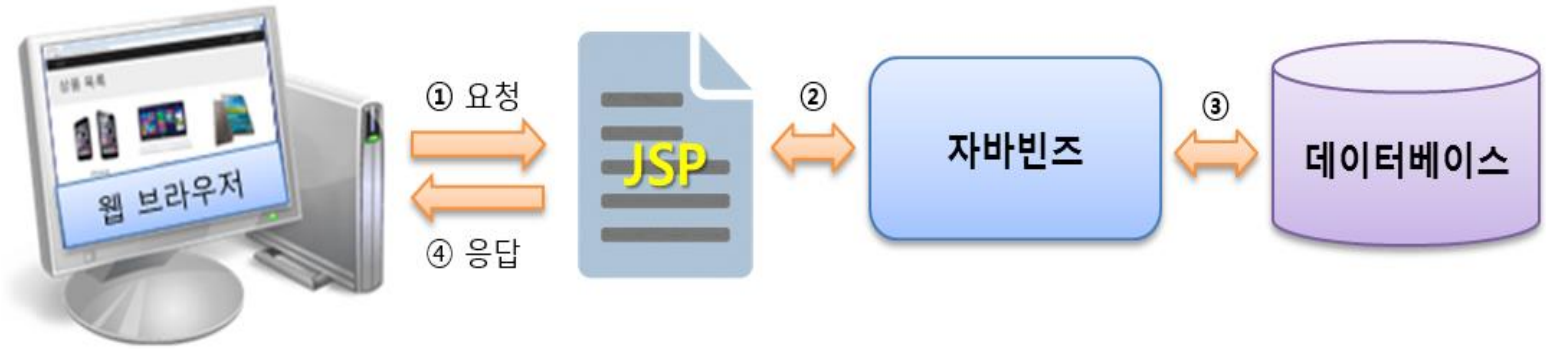
액션 태그	형식	설명
forward	〈jsp:forward ··· /〉	다른 페이지로의 이동과 같은 페이지 흐름을 제어합니다.
include	〈jsp:include ··· /〉	외부 페이지의 내용을 포함하거나 페이지를 모듈화합니다.
useBean	〈jsp:useBean ··· /〉	JSP 페이지에 자바빈즈를 설정합니다.
setProperty	〈jsp:setProperty ··· /〉	자바빈즈의 프로퍼티 값을 설정합니다.
getProperty	〈jsp:getProperty ··· /〉	자바빈즈의 프로퍼티 값을 얻어옵니다.
param	〈jsp:param ··· /〉	〈jsp:forward〉, 〈jsp:include〉, 〈jsp:plugin〉 태그에 인자를 추가합니다.
plugin	〈jsp:plugin ··· /〉	웹 브라우저에 자바 애플릿을 실행합니다. 자바 플러그인에 대한 OBJECT 또는 EMBED 태그를 만드는 브라우저별 코드를 생성합니다.
element	〈jsp:element ··· /〉	동적 XML 요소를 설정합니다.
attribute	〈jsp:attribute ··· /〉	동적으로 정의된 XML 요소의 속성을 설정합니다.
body	〈jsp:body ··· /〉	동적으로 정의된 XML 요소의 몸체를 설정합니다.
text	〈jsp:text ··· /〉	JSP 페이지 및 문서에서 템플릿 텍스트를 작성합니다.

자바빈즈

동적 콘텐츠 개발을 위해 자바 코드를 사용하여 자바 클래스로 로직을 작성하는 방법

JSP 페이지에서 화면을 표현하기 위한 계산식이나 자료의 처리를 담당하는 자바코드를 따로 분리하여 작성하는 것

JSP 페이지가 HTML과 같이 쉽고 간단한 코드만으로 구성



자바빈즈를 작성할 때 규칙

- 자바 클래스는 java.io.Serializable 인터페이스를 구현해야 함
- 인수가 없는 기본 생성자가 있어야 함
- 모든 멤버 변수인 프로퍼티는 private 접근 지정자로 설정해야 함
- 모든 멤버 변수인 프로퍼티는 getter/setter( ) 메소드가 존재해야 함
  - getter( ) 메소드는 멤버 변수에 저장된 값을 가져올 수 있는 메소드이고,
  - setter( ) 메소드는 멤버 변수에 값을 저 장할 수 있는 메소드임

```java
package dto;

import java.io.Serializable;

//모델 클래스
//JSP : 자바빈즈 (getter/setter, 기본 생성자(매개변수 없음), Serializable - 주소값 지정)

//public class Product implements Serializable { // implement ~ 생략해도 됩니다...
// javabeans는 getter-setter만 있으면 자바빈즈로 해석합니다...

public class Product {

    private static final long serialVersionUID = 2867748905925104542L;

    private String productId;
    private String name;
    private int unitPrice;
    private String description;
    private String manufacturer;
    private String category;
    private long unitsInStock;
    private String condition;

    public Product() {

    }

    public Product(String productId, String name, int unitPrice) {
        this.productId = productId;
        this.name = name;
        this.unitPrice = unitPrice;
    }

    public String getProductId() {
        return productId;
    }
```

useBean 액션 태그

JSP 페이지에서 자바빈즈를 사용하기 위해 실제 자바 클래스를 선언하고 초기화하는 태그

id 속성과 scope 속성을 바탕으로 자바빈즈의 객체를 검색하고, 객체가 발견되지 않으면 빈 객체를 생성

```
<jsp:useBean  id= "자바빈즈 식별이름"  class= "자바빈즈 이름" scope= "범위"/>
```

속성	설명
id	자바빈즈를 식별하기 위한 이름입니다.
class	패키지 이름을 포함한 자바빈즈 이름입니다. 자바빈즈는 인수가 없는 기존 생성자가 있어야 하며 추상 클래스를 사용할 수 없습니다.
scope	자바빈즈가 저장되는 영역을 설정합니다. page(기본 값), request, session, application 중 하나의 값을 사용합니다.

```java
package dto;

import java.io.Serializable;

//모델 클래스
//JSP : 자바빈즈 (getter/setter, 기본 생성자(매개변수 없음), Serializable - 주소값 지정)

//public class Product implements Serializable { // implement ~ 생략해도 됩니다...
// javabeans는 getter-setter만 있으면 자바빈즈로 해석합니다...

public class Product {

    private static final long serialVersionUID = 2867748905925104542L;

    private String productId;
    private String name;
    private int unitPrice;
    private String description;
    private String manufacturer;
    private String category;
    private long unitsInStock;
    private String condition;

    public Product() {

    }

    public Product(String productId, String name, int unitPrice) {
        this.productId = productId;
        this.name = name;
        this.unitPrice = unitPrice;
    }

    public String getProductId() {
        return productId;
    }

    public void setProductId(String productId) {
        this.productId = productId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getUnitPrice() {
        return unitPrice;
    }

    public void setUnitPrice(int unitPrice) {
        this.unitPrice = unitPrice;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public long getUnitsInStock() {
        return unitsInStock;
    }

    public void setUnitsInStock(long unitsInStock) {
        this.unitsInStock = unitsInStock;
    }

    public String getCondition() {
        return condition;
    }

    public void setCondition(String condition) {
        this.condition = condition;
    }

//  @Override
//  public String toString() {
//      return "Product [productId=" + productId + ", name=" + name + ", unitPrice=" + unitPrice + ", description="
//              + description + ", manufacturer=" + manufacturer + ", category=" + category + ", unitsInStock="
//              + unitsInStock + ", condition=" + condition + "]";
//  }
```

dto 폴더 내 Product 클래스 소스코드
: jsp 페이지에서 데이터를 게터세터 하기 위한 운송(?) 작업 폴더 생성

[자바빈즈로 사용할 상품 데이터 접근 클래스 만들기]

```java
package dao;

import java.util.ArrayList;


public class ProductRepository {

    private List<Product> products = new ArrayList<>();

    public ProductRepository() {

        Product phone = new Product("P1234", "iPhone6s", 800000);
        phone.setDescription("4.7-inch, 1334X750 Retina HD display, 8-megapixel iSight Camera");
        phone.setCategory("Smart Phone");
        phone.setManufacturer("Apple");
        phone.setUnitsInStock(1000);
        phone.setCondition("New");

        Product notebook = new Product("P1235", "LG PC 그램", 1500000);
        notebook.setDescription("13.3-inch, IPS LED display, 5rd Generation Intel Core processors");
        notebook.setCategory("Notebook");
        notebook.setManufacturer("LG");
        notebook.setUnitsInStock(1000);
        notebook.setCondition("Refurbished");

        Product tablet = new Product("P1236", "Galaxy Tab S", 900000);
        tablet.setDescription("212.8*125.6*6.6mm, Super AMOLED display, Octa-Core processor");
        tablet.setCategory("Tablet");
        tablet.setManufacturer("Samsung");
        tablet.setUnitsInStock(1000);
        tablet.setCondition("Old");

        products.add(phone);
        products.add(notebook);
        products.add(tablet);

    }
    public List<Product> getAllProducts(){
        return products;
    }
```

dto 폴더 내 Product.java에 구체적인 데이터베이스를 넣기 위한

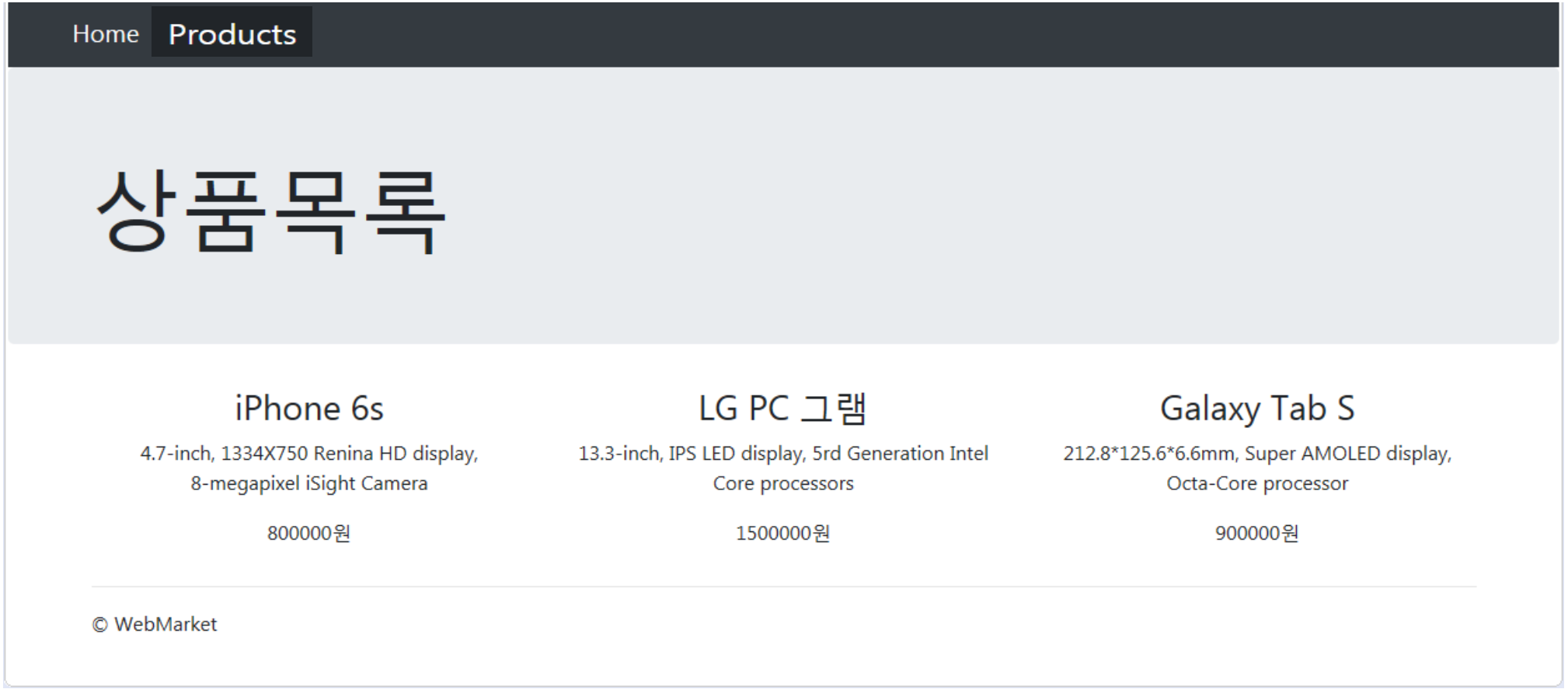
고정 db화 작업을 하고
이를, dao 폴더 ProductRepository 클래스 생성

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.util.List" %>
<%@ page import="dto.Product" %>
<%@ page import="dao.ProductRepository" %>
<jsp:useBean id="repository" class="dao.ProductRepository" scope="session" />
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>상세 정보</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

</head>
<body>
<jsp:include page="menu.jsp" />

<div class="p-5 bg-secondary text-white text-white">
    <div class ="container">
        <h1 class="display-3">상세정보</h1>
    </div>
</div>

    <%

    /* 상품 목록 페이지에서 특정 상품 상세정보로 이동할 때 사용하는 문법입니다... */

    String id = request.getParameter("id");
    Product product = repository.getProductById(id);

    %>

    <div class="container">
    <div class="row">
            <div class="col-md-6">
            <h3><%= product.getName() %></h3>
            <p><%= product.getDescription() %></p>
            <p><%= product.getUnitPrice() %></p>
                <!-- 나머지는 책에 있으므로 혼자 실습 시 찾아보세요! -->
            <p><a href="#" class="btn btn-info">상품주문 &raquo;</a>
<a href="./products.jsp" class="btn btn-secondary">상품목록 &raquo;</a></p>
            </div>

        <!-- <div class="col-md-4">1</div>
        <div class="col-md-4">2</div>
        <div class="col-md-4">3</div>
     -->
     </div>
</div>

<jsp:include page="footer.jsp" />

</body>

</html>
```

dao 폴더 내 클래스 db를 활용하여..

dto 폴더 클래스 내 변수로 저장 후..

상세정보 product.jsp로 호출 후 출력



4강 상품목록 창 새로 생성 후 완료!!

내장 객체(implicit object)

JSP 페이지에서 사용할 수 있도록 JSP 컨테이너에 미리 정의된 객체
JSP 페이지가 서블릿 프로그램으로 번역될 때 JSP 컨테이너가 자동으로
내장 객체를 멤버 변수, 메소드 매개변수 등의 각종 참조 변수(객체)로 포함
JSP 페이지에 별도의 import 문 없이 자유롭게 사용 가능
스크립틀릿 태그나 표현문 태그에 선언을 하거나 객체를 생성하지 않고도
직접 호출하여 사용 가능

내장 객체	반환 유형	설명
request	javax.servlet.http.HttpServletRequest	웹 브라우저의 HTTP 요청 정보를 저장합니다.
response	javax.servlet.http.HttpServletResponse	웹 브라우저의 HTTP 요청에 대한 응답 정보를 저장합니다.
out	javax.servlet.jsp.jsp.jspWriter	JSP 페이지에 출력할 내용을 담고 있는 출력 스트림입니다.
session	javax.servlet.http.HttpSession	웹 브라우저의 정보를 유지하기 위한 세션 정보를 저장합니다 (13장 참고).
application	javax.servlet.ServletContext	웹 애플리케이션의 콘텍스트 정보를 저장합니다.
pageContext	javax.servlet.jsp.PageContext	JSP 페이지의 정보를 저장합니다.
page	java.lang.Object	JSP 페이지를 구현한 자바 클래스로 JSP 페이지 자체를 나타냅니다.
config	javax.servlet.ServletConfig	JSP 페이지의 설정 정보를 저장합니다.
exception	java.lang.Throwable	JSP 페이지의 예외 발생을 처리합니다(11장 참고).

request 내장 객체

JSP 페이지에서 가장 많이 사용되는 기본 내장 객체
웹 브라우저에서 서버의 JSP 페이지로 전달하는 정보를 저장
폼 페이지로부터 입력된 데이터를 전달하는 요청 파라미터 값을 JSP 페이지로 가져옴
JSP 컨테이너는 웹 브라우저에서 서버로 전달되는 정보를 처리하기 위해 javax.servlet.http.HttpServletRequest 객체 타입의 request 내장 객체를 사용하여 사용자의 요구 사항을 얻어냄

response 내장 객체

사용자의 요청을 처리한 결과를 서버에서 웹 브라우저로 전달하는 정보를 저장하고 서버는 응답 헤더와 요청 처리 결과 데이터를 웹 브라우저로 보냄
JSP 컨테이너는 서버에서 웹 브라우저로 응답하는 정보를 처리하기 위해 javax.servlet.http.HttpServletResponse 객체 타입의 response 내장 객체를 사용하여 사용자의 요청에 응답

out 내장 객체

웹 브라우저에 데이터를 전송하는 출력 스트림 객체
JSP 컨테이너는 JSP 페이지에 사용되는 모든 표현문 태그와 HTML, 일반 텍스트 등을 out 내장 객체를 통해 웹 브라우저에 그대로 전달
스크립틀릿 태그에 사용하여 단순히 값을 출력하는 표현문 태그(<%= ...%>)와 같은 결과를 얻을 수 있음

5. [웹 쇼핑몰] 상품 상세 정보 표시하기

```
7
8  public class ProductRepository {
9
10     private List<Product> products = new ArrayList<>();
11
12     public ProductRepository() {
13
14         Product phone = new Product("P1234", "iPhone6s", 800000);
15         phone.setDescription("4.7-inch, 1334X750 Retina HD display, 8-megapixel i
16         phone.setCategory("Smart Phone");
17         phone.setManufacturer("Apple");
18         phone.setUnitsInStock(1000);
19         phone.setCondition("New");
20
21         Product notebook = new Product("P1235", "LG PC 그램", 1500000);
22         notebook.setDescription("13.3-inch, IPS LED display, 5rd Generation Intel
23         notebook.setCategory("Notebook");
24         notebook.setManufacturer("LG");
25         notebook.setUnitsInStock(1000);
26         notebook.setCondition("Refurbished");
27
28         Product tablet = new Product("P1236", "Galaxy Tab S", 900000);
29         tablet.setDescription("212.8*125.6*6.6mm, Super AMOLED display, Octa-Core
30         tablet.setCategory("Tablet");
31         tablet.setManufacturer("Samsung");
32         tablet.setUnitsInStock(1000);
33         tablet.setCondition("Old");
34
35         products.add(phone);
36         products.add(notebook);
37         products.add(tablet);
38
39     }
40     public List<Product> getAllProducts(){
41         return products;
42     }
43
44     //상품 ID로 상품을 찾기
45
46     public Product getProductById(String productId) {
47         return products // 동일한 기존 코드(for문) 교재 p.173을 참고하시기 바랍니다...
48                 .stream() // 상품 3개가 흘러갑니다...
49                 .filter((product) -> product.getProductId().equals(productId)) //
50                 .findFirst() // 가장 먼저 걸린 것을 찾습니다...
51                 .get(); // 가장 먼저 걸린 녀석을 get합니다...
52     }
53
54
```

상품 상세 정보를 가져오는 메소드 만들기

람다식으로 표기..

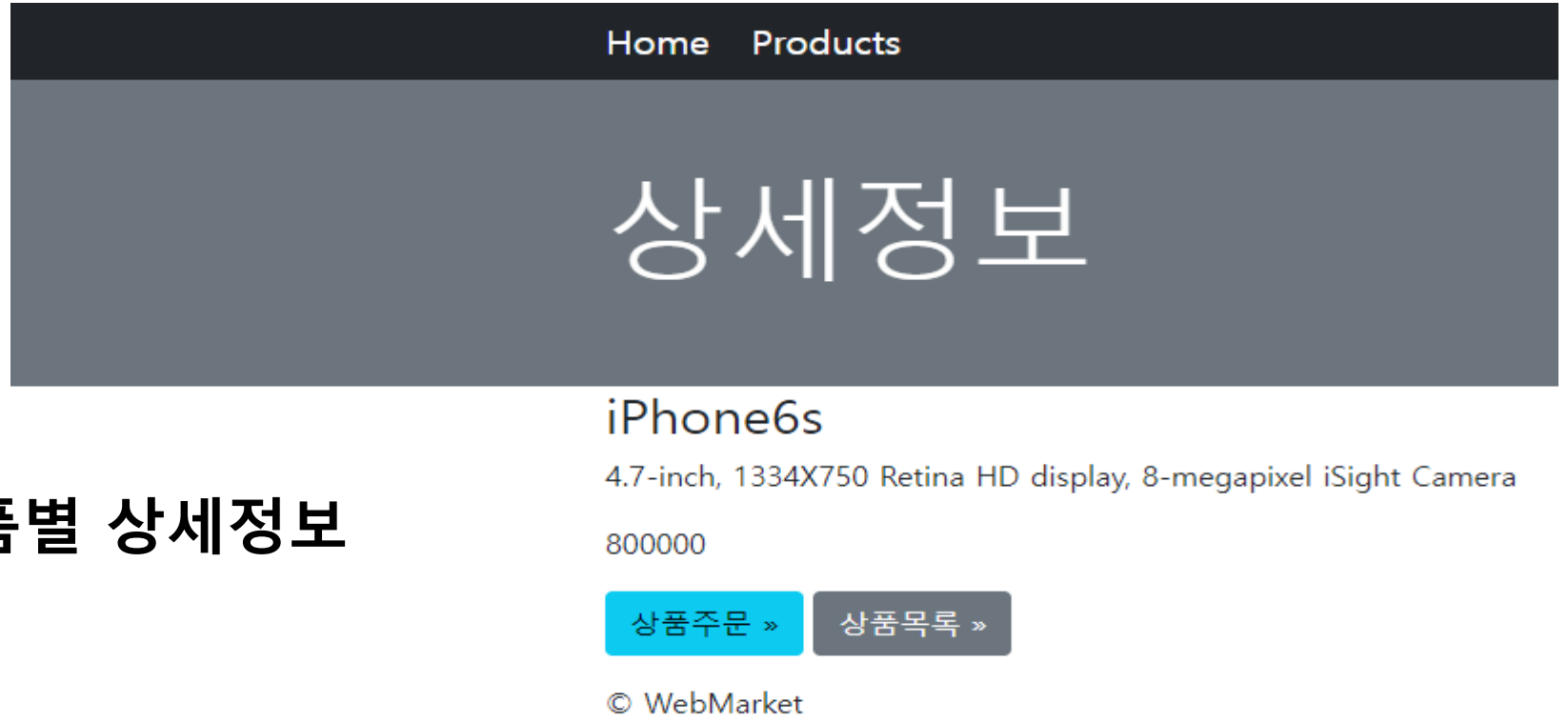
```jsp
<%@ page import="java.util.List" %>
<%@ page import="dto.Product" %>
<%@ page import="dao.ProductRepository" %>
<jsp:useBean id="repository" class="dao.ProductRepository" scope="session" />
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>상품 목록</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

</head>
<body>
<jsp:include page="menu.jsp" />

<div class="p-5 bg-secondary text-white text-white">
    <div class ="container">
        <h1 class="display-3">상품목록</h1>
    </div>
</div>

    <%
    //ProductRepository repository = new ProductRepository();
    List<Product> products = repository.getAllProducts();

    %>

    <div class="container">
    <div class="row text-center">
    <%
    for(Product product : products){
    %>
            <div class="col-md-4">
            <h3><%= product.getName() %></h3>
            <p><%= product.getDescription() %></p>
            <p><%= product.getUnitPrice() %></p>
            <p><a class="btn btn-secondary" role="button" href="./product.jsp?id=<%= product.getProductId() %>">상세정보&raquo;</a></p>
            </div>

    <%
    }
    %>
        <!-- <div class="col-md-4">1</div>
        <div class="col-md-4">2</div>
        <div class="col-md-4">3</div>
     -->
     </div>
</div>

<jsp:include page="footer.jsp" />

</body>

</html>
```

상세 정보 버튼 생성

```jsp
</head>

<body>

<jsp:include page="menu.jsp" />

<%! String greeting = "Welcome to Web Shopping Mall";
String tagline = "Welcome to Web Market!"; %>

<div class="p-5 bg-secondary text-white text-white">
    <div class ="container">
        <h1 class="text-center display-3"><%= greeting %></h1>
    </div>
</div>

<div class="container">
    <div class="text-center">
        <h3 class="p-3"><%= tagline %></h3>
        <%

        //1초에 한번씩 새로고침 할 때 사용하는 문법입니다...
        response.setIntHeader("Refresh", 1);

        Date today = new Date();

        SimpleDateFormat format = new SimpleDateFormat("hh:mm:ss a");

        out.println("현재 접속 시간 : " + format.format(today));
        %>
    </div>
    <hr>
</div>

<jsp:include page="footer.jsp"></jsp:include>

</body>

</html>
```

response.setIntHeader(“Refresh”, 1)

-> 페이지 새로고침 (1초단위로)



5강 상품목록에서 제품별 상세정보 실습하기 완료 화면!