Chapter

03

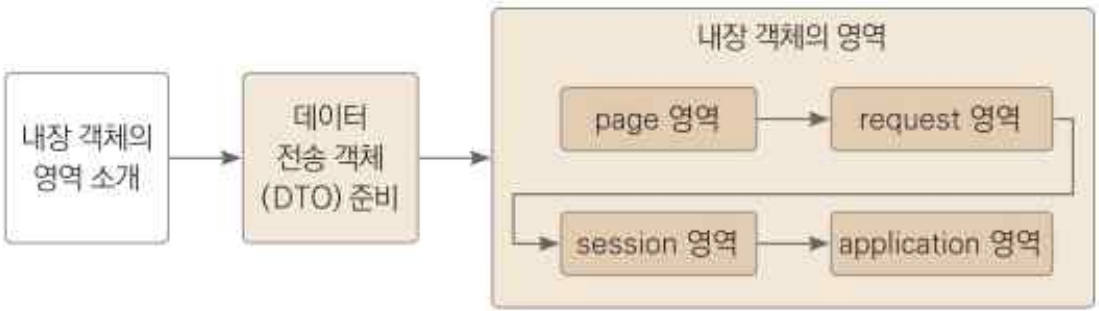
내장 객체의 영역 (Scope)

학습 목표 및 순서

▣ 학습 목표

- 내장객체 중 4가지는 영역이라는 개념을 가지고 있습니다.
- 메모리의 일종으로 Object를 기반으로 데이터를 저장하고 공유할 수 있습니다.
- 이번 장에서는 내장객체의 유효기간이라 할 수 있는 영역에 대해 학습해보겠습니다.

▣ 학습 순서



학습 목표 및 순서

▣ 활용 사례

- 웹에서는 페이지(page)들이 모여 하나의 요청(request)을 처리하며, 요청들이 모여 하나의 세션(session)을, 다시 세션들이 모여 하나의 웹 애플리케이션 (application)을 이룹니다.
- 따라서 이 4가지 내장 객체의 영역 개념을 잘 이해해야 웹 애플리케이션을 효율적으로 설계하고 구현할 수 있습니다.

3.1 내장 객체의 영역이란?

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%!
String str1 = "JSP";
String str2 = "안녕하세요..";
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>HelloJSP</title>
</head>
<body>
    <h2>처음 만들어보는 <%= str1 %></h2>
    <p>
        <%
        out.println(str2 + str1 + "입니다. 열공합시다^^*");
        %>
    </p>
</body>
</html>
```

지시어

스크립트 요소(선언부)

스크립트 요소(표현식)

스크립트 요소(스크립틀릿)

예제1-6 :

지시어(Directive)

- JSP를 Java코드로 변환시 필요한 정보 기술(필수)

스크립트요소(Scripting Elements)

- 선언부
  - 멤버변수나 메서드 선언
- 표현식
  - 변수 출력
- 스크립틀릿

3.1 내장 객체의 영역이란?

▣ 내장 객체의 영역이란..??

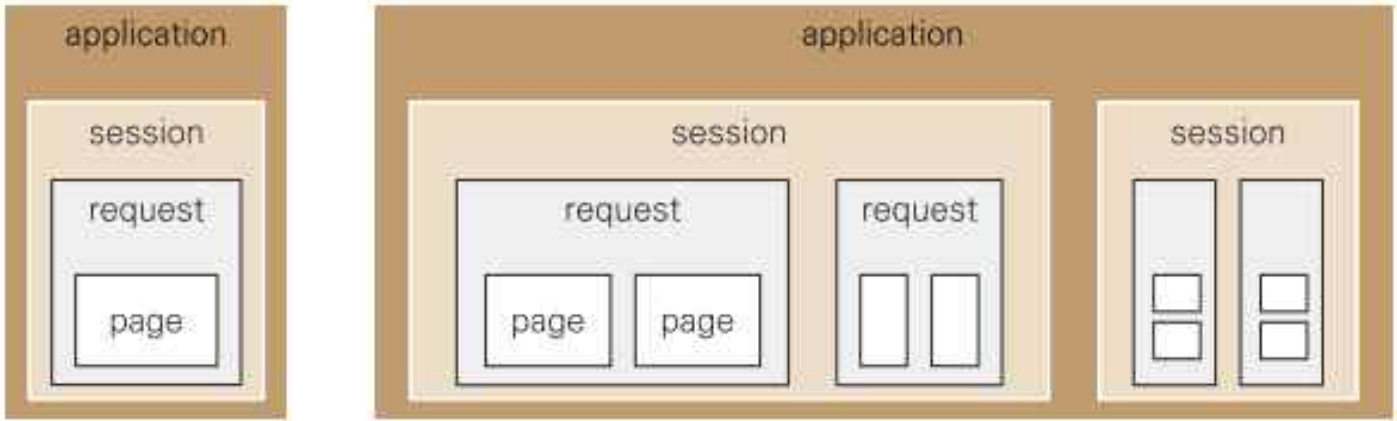
- 각 객체가 저장되는 메모리의 유효기간
- JSP는 페이지 단위로 구성되므로 4가지 영역을 통해 데이터를 공유할 수 있음

▣ 내장 객체의 4가지 영역

- page 영역 : 동일한 페이지에서만 공유되어 페이지를 벗어나면 소멸, pageContext 객체를 사용
- request 영역: 하나의 요청에 의해 호출된 페이지와 포워드(요청 전달)된 페이지까지 공유, request 객체를 사용
- session 영역: 클라이언트가 처음 접속한 후 웹 브라우저를 닫을 때까지 공유, session 객체를 사용
- application 영역: 한 번 저장되면 웹 애플리케이션이 종료될 때까지 유지 , application 객체를 사용

3.1 내장 객체의 영역이란?

▣ 내장 객체의 영역별 접근범위 및 포함관계



- 범위의 크기 : application > session > request > page
- 애플리케이션 구조에 따라 큰 영역은 작은 영역을 하나 이상 포함할 수 있음

3.1 내장 객체의 영역이란?

▣ 주요 메서드

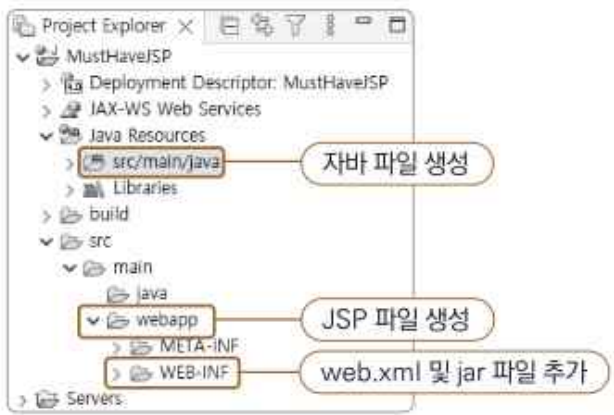
메서드명	설 명
void setAttribute(String name, Object value)	각 영역에 속성을 저장 첫번째 인수는 속성명, 두번째 인수는 저장할 값 값의 타입은 Object이므로 모든 타입의 객체 저장 가능
Object getAttribute(String name)	영역에 저장된 속성값을 얻어옴 Object로 자동 형변환되어 저장되므로 원래 타입으로 형변환 후 사용해야 함
void removeAttribute(String name)	영역에 저장된 속성을 삭제 삭제할 속성명이 존재하지 않더라도 에러는 발생하지 않음

- 위 메서드는 4가지 내장객체의 영역에서 공통으로 사용
- 즉 사용법의 차이는 없고, 단지 공유 범위에서만 차이가 있음

3.2 데이터 전송 객체(DTO) 준비

▣ DTO란..??

- DTO(Data Transfer Object)는 주로 데이터를 저장하거나 전송하는 데 쓰이는 객체
- 별 다른 로직 없이 순수하게 데이터만 저장하는 기능을 가짐
- 데이터만 가지고 있으므로 VO(Value Object)라고 하기도 함
- 자바빈즈(Java Beans) 규약에 따라 작성
  a. 자바빈즈는 기본(default) 패키지 이외의 패키지에 속해야 함
  b. 멤버 변수(속성)의 접근 지정자는 private
  c. 기본 생성자가 있어야 함
  d. 멤버 변수에 접근할 수 있는 게터(getter)/
  e. 게터/세터 메서드의 접근 지정자는 public

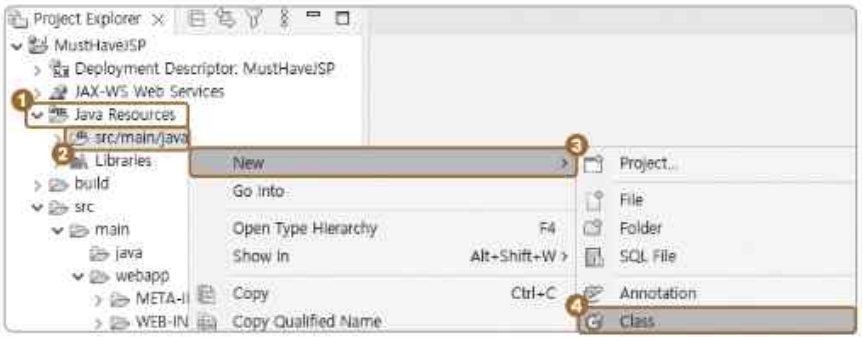


▣ Person 클래스 생성

3.2 데이터 전송 객체(DTO) 준비

▣ Person 클래스 생성

- Java Resources → src/main/java → 마우스 우클릭 → [New] → [Class] 클릭



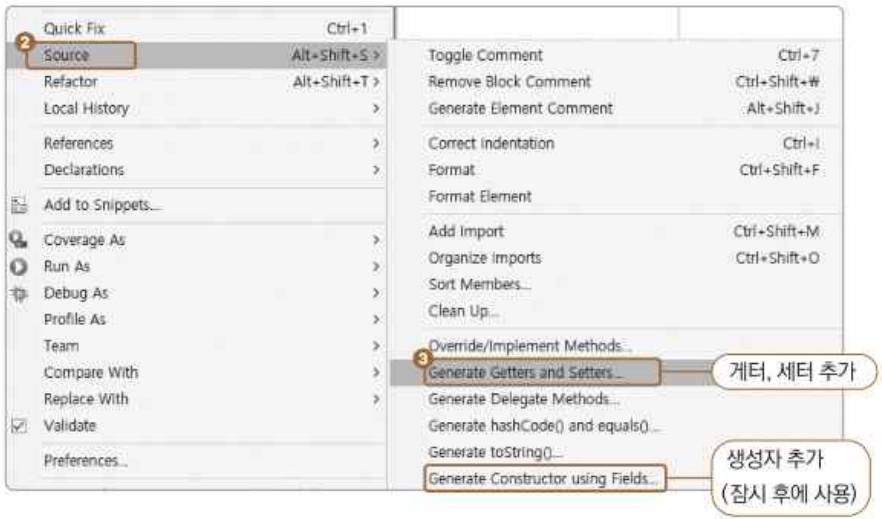
예제 3-1] Java Resources/common/Person.java

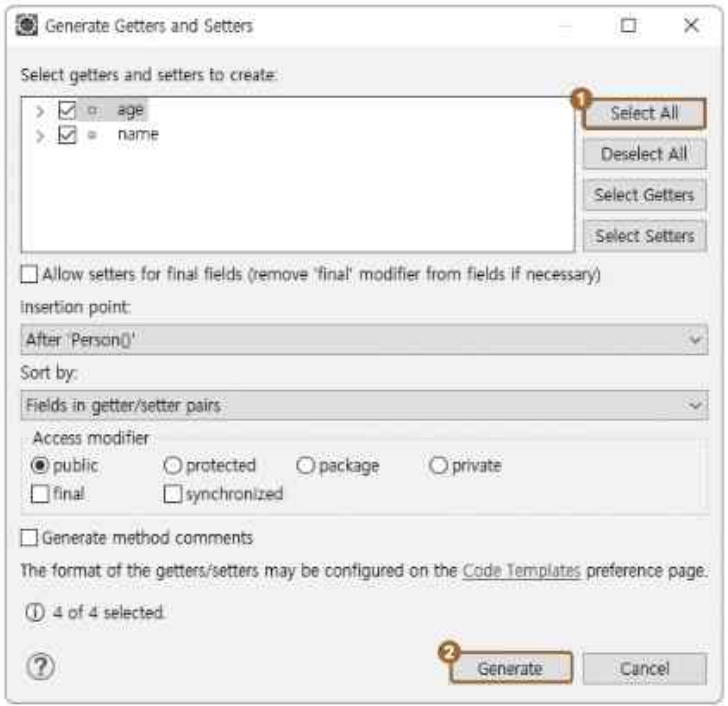
```
package common;            // 기본 패키지 이외의 패키지(규약 1번)

public class Person {
    private String name;   // private 멤버 변수(규약 2번)
    private int age;       // private 멤버 변수(규약 2번)

    public Person() {}     // 기본 생성자(규약 3번)
}
```

3.2 데이터 전송 객체(DTO) 준비

▣ Person 클래스 생성

- Getters / Setters 생성
- 편집창에서 마우스 우클릭 → [Source] → [Generate Getters and Setters...] 선





3.3 page 영역

- page 영역은 기본적으로 클라이언트의 요청을 처리하는 JSP마다 하나씩 생성
- 각 JSP는 page 영역을 사용하기 위한 pageContext 객체를 할당 받음
- page영역에 속성 저장
  - pageContext.setAttribute(속성명, 값);
- page영역의 속성 읽기
  - pageContext.getAttribute(속성명);
- include 지시어를 통해 포함시킨 파일은 같은 페이지이므로 page 영역을 공유
- 링크를 통해 다른 페이지로 이동하면 page영역에 저장된 데이터는 소멸됨

```
<%
pageContext.setAttribute("pageInteger", 100);
pageContext.setAttribute("pageString", "페이지 영역의 문자열");
pageContext.setAttribute("pagePerson", new Person("한석봉", 99));
%>
    <h2>page 영역의 속성값 읽기</h2>
    <%
    int pInteger = (Integer)(pageContext.getAttribute("pageInteger"));
    String pString = pageContext.getAttribute("pageString").toString();
    Person pPerson = (Person)(pageContext.getAttribute("pagePerson"));
    %>
```

❷ 속성 저장

❸ 속성 읽기

❹

❷ setAttribute()로 속성 저장

❸ getAttribute()로 속성

예제3-1 : Person.java
예제3-3 : PageContextMain.jsp

3.3 page 영역

▣ page 영역

- 예제 3-3]에 포함시킬 JSP문서 생성
- 해당 파일은 page 지시어를 제외한 모든 HTML태그를 제거한 후 작성

예제 3-4] 03Scope/PageInclude.jsp

```
<%@ page import="common.Person"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<h4>Include 페이지</h4>
<%
int pInteger2 = (Integer)(pageContext.getAttribute("pageInteger"));
// String pString2 = pageContext.getAttribute("pageString").toString();
Person pPerson2 = (Person)(pageContext.getAttribute("pagePerson"));
%>
<ul>
    <li>Integer 객체 : <%= pInteger2 %></li>
    <li>String 객체 : <%= pageContext.getAttribute("pageString") %></li>
    <li>Person 객체 : <%= pPerson2.getName() %>, <%= pPerson2.getAge() %></li>
</ul>
```

❶ 속성 읽기

❷ 해당 페이지는 include 지시어를 통해 포함되었으므로 page영역이 공유되어 모든 내용이 출력됨

3.3 page 영역

▣ page 영역

- 예제 3-3]에서 <a>태그의 링크로 이동할 페이지
- 해당 파일은 HTML태그를 제거하지 않고 작성

예제 3-5] 03Scope/PageLocation.jsp

```
<body>
    <h2>이동 후 page 영역의 속성값 읽기</h2>
    <%
    Object pInteger = pageContext.getAttribute("pageInteger");
    Object pString = pageContext.getAttribute("pageString");
    Object pPerson = pageContext.getAttribute("pagePerson");
    %>
    <ul>
        <li>Integer 객체 : <%= (pInteger == null) ? "값 없음" : pInteger %></li>
        <li>String 객체 : <%= (pString == null) ? "값 없음" : pString %></li>
        <li>Person 객체 : <%= (pPerson == null) ? "값 없음" : ((Person)pPerson).
                        getName() %></li>
    </ul>
</body>
```

❶ 속성 읽기

❷ 페이지 이동시 page 영역은 소멸되므로 모든 값이 출력되지 않음

3.4 request 영역

- 클라이언트가 요청을 할 때마다 새로운 request 객체가 생성
- request 영역은 같은 요청을 처리하는 데 사용되는 모든 JSP가 공유
- 즉, 현페 페이지와 포워드(forward) 된 페이지까지 공유됨
- 단, 다른 페이지로 이동하면 request영역에 저장된 데이터는 소멸됨
- request영역에 속성 저장 및 읽기
  - page영역과 동일. 뒷 부분의 session, application영역도 동일함.
- request영역의 속성 삭제
  - request.removeAttribute(속성명);
- 요청에 대한 응답이 완료될 때 소멸되므로 page 영역보다 접근 범위가 조금 더 넓

▶ 포워드(forward)란?
: 현재 페이지로 들어온 요청을 다음 페이지로 전달하는 것

형식	request.getRequestDispatcher("포워드할 파일 경로") .forward(request, response);

예제3-6 : RequestMain.jsp

3.4 request 영역

▣ request 영역

예제 3-6] 03Scope/RequestMain.jsp

```
<%
request.setAttribute("requestString", "request 영역의 문자열");
request.setAttribute("requestPerson", new Person("안중근", 31));
%>
<html>
<head><title>request 영역</title></head>
<body>
    <h2>request 영역의 속성값 삭제하기</h2>
    <%
        request.removeAttribute("requestString");
        request.removeAttribute("requestInteger"); // 에러 없음
    %>
```

❶ 속성 저장

❶ 영역에 저장시에는 setAttribute() 를 동일하게 사용

❷ 삭제시 removeAttribute() 를 사용

❸ 삭제할 속성이 없더라도 에러가 발생하지 않음

3.4 request 영역

▣ request 영역

- 포워드는 다소 복잡한 함수의 조합으로 기술해야 함

```
request.getRequestDispatcher("포워드할 파일 경로").forward(request, response)
```

예제 3-6] 03Scope/RequestMain.jsp(

```
    ... 생략 ...
    <h2>포워드된 페이지에서 request 영역 속성값 읽기</h2>
    <%
    request.getRequestDispatcher(
                "RequestForward.jsp?paramHan=한글&paramEng=English")
            .forward(request, response);
    %>
</body>
</html>
```

RequestForward.jsp로 포워드 하면서 쿼리스트링을 통해 파라미터도 함께 전달

3.4 request 영역

▣ request 영역

- 포워드 되는 페이지 작성

예제 3-7]

```
<body>
    <h2>포워드로 전달된 페이지</h2>
    <h4>RequestMain 파일의 리퀘스트 영역 속성 읽기</h4>
    <%
    Person pPerson = (Person)(request.getAttribute("requestPerson")); ❶
    %>
    <ul>
        <li>String 객체 : <%= request.getAttribute("requestString") %></li>
        <li>Person 객체 : <%= pPerson.getName() %>, <%= pPerson.getAge() %></li> ❷
    </ul>
    <h4>매개변수로 전달된 값 출력하기</h4>
    <%
        request.setCharacterEncoding("UTF-8"); ❸
        out.println(request.getParameter("paramHan"));
        out.println(request.getParameter("paramEng")); ❹
    %>
</body>
```

❶ request 영역은 포워드 된 페이지까지는 공유되므로 속성을 읽어올 수 있음

❷ ❶에서 읽은 속성값 출력

❸ Tomcat 10.1에서는 한글 깨짐 현상이 없으므로 생략가능. 9이하에서는 깨짐 현상 발생됨.

❹ 쿼리스트링을 통해 전달된 파라미터 출력