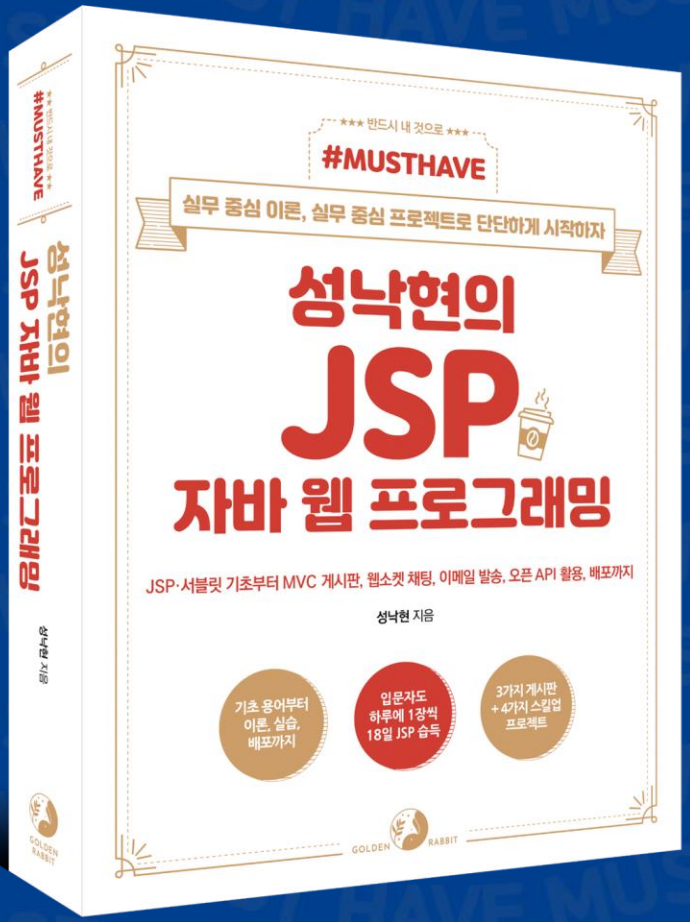


Chapter

03

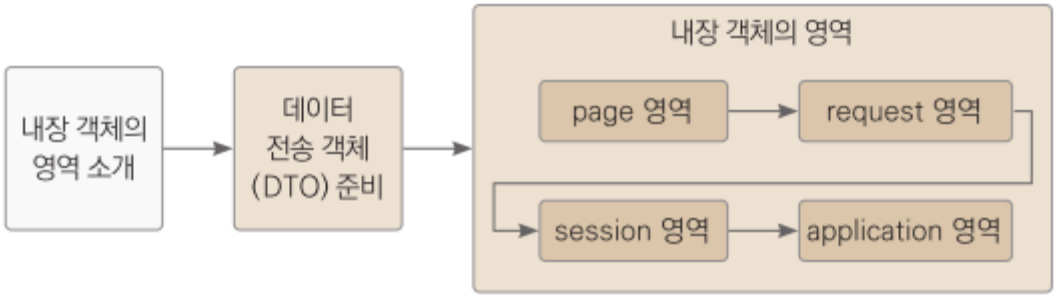
내장 객체의 영역 (Scope)

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?

▣ 내장 객체의 영역이란..??

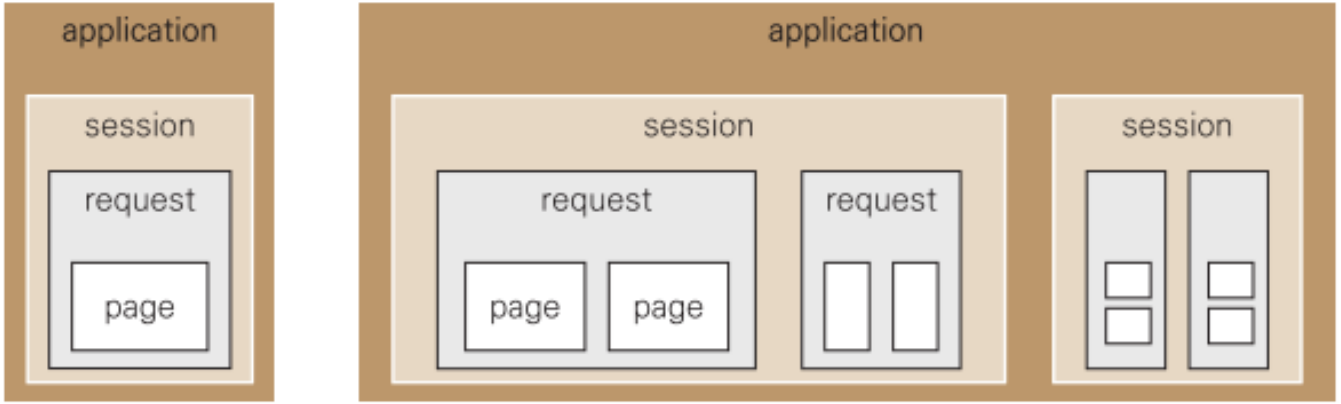
- 각 객체가 저장되는 메모리의 유효기간
- JSP는 페이지 단위로 구성되므로 4가지 영역을 통해 데이터를 공유할 수 있음

▣ 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

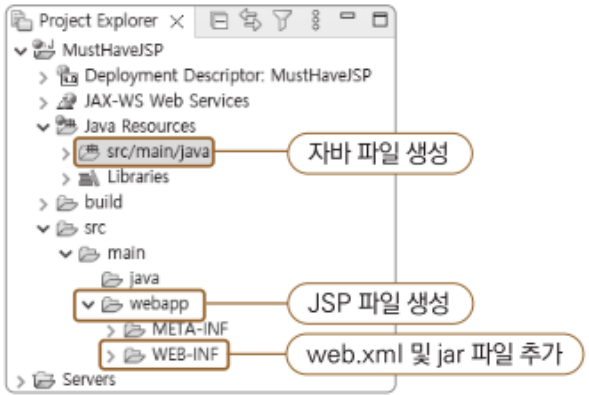
3.2 데이터 전송 객체(DTO) 준비

▣ 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 private으로 선언
  c. 기본 생성자가 있어야 함
  d. 멤버 변수에 접근할 수 있는 게터(getter)/세터(setter) 메서드가 필요
  e. 게터/세터 메서드의 접근 지정자는 public으로 선언

▣ Person 클래스 생성

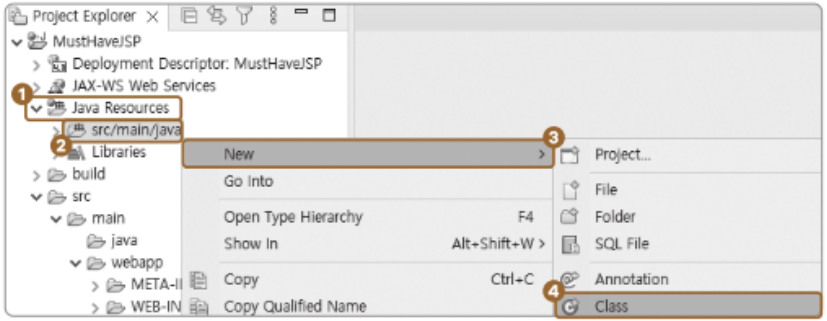
- Java Resources 하위의 src/main/java에 생성



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

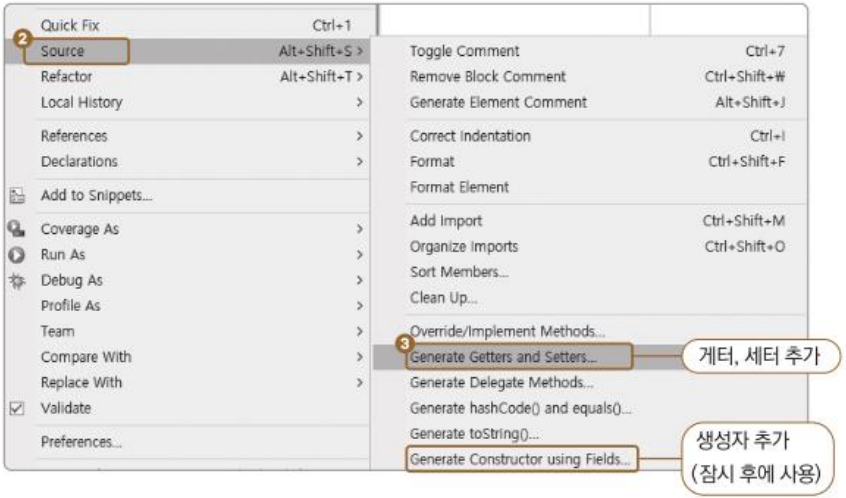
- 멤버변수 선언 및 기본생성자 정의

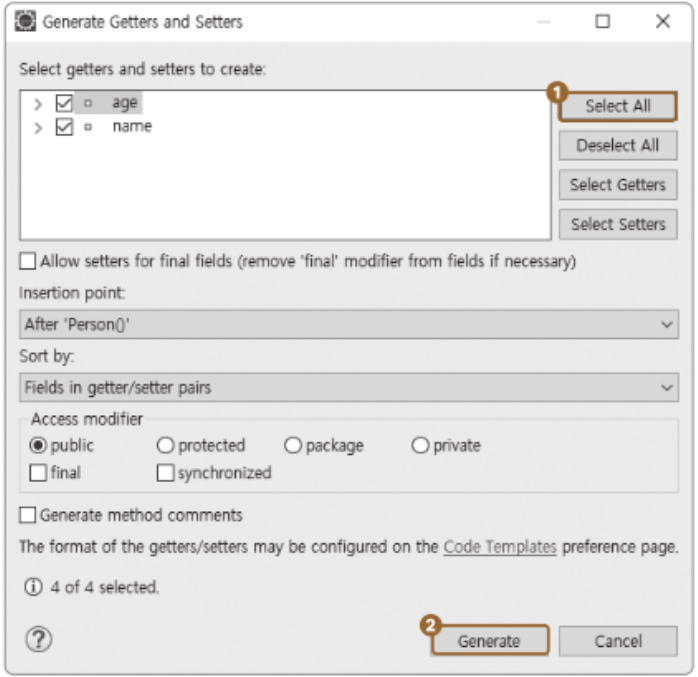
```
package common;           // 기본 패키지 이외의 패키지(규약 1번)

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...] 선택





- [Select All] → [Generate] 클릭해서 완성

3.3 page 영역

▣ page 영역

- 클라이언트의 요청을 처리하기 위해 JSP 페이지 당 하나씩 생성
- pageContext 객체를 통해 영역을 사용함
- 해당 페이지에서만 공유되고 페이지를 벗어나면 소멸
- include 지시어를 통해 포함한 파일도 공유됨(다음 페이지)

예제 3-3] 03Scope/PageContextMain.jsp

```
<%
pageContext.setAttribute("pageInteger", 1000);
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 읽기
❹ 사용시 반드시 원래의 타입으로 형변환

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

▣ request 영역

- request 영역은 하나의 요청에 대한 응답이 완료될 때 소멸
- 페이지 이동시에는 소멸되지만, 포워드 된 페이지까지는 영역을 공유할 수 있음
- 따라서 page 영역보다는 접근 범위가 넓음

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(계속)

```
    ... 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] 03Scope/RequestForward.jsp

```
<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

3.5 session 영역

▣ session 영역

- 웹 브라우저를 최초로 연 후 닫을 때까지 요청되는 모든 페이지는 session 영역을 공유
- 세션(session)이란 클라이언트가 서버에 접속해 있는 상태 혹은 단위

예제 3-8] 03Scope/SessionMain.jsp

```
<%
ArrayList<String> lists = new ArrayList<String>();
lists.add("리스트");
lists.add("컬렉션");                                    ❶
session.setAttribute("lists", lists);  ❷
%>
<html>
<head><title>session 영역</title></head>
<body>
    <h2>페이지 이동 후 session 영역의 속성 읽기</h2>
    <a href="SessionLocation.jsp">SessionLocation.jsp 바로가기</a>  ❸
</body>
```

❶ List 컬렉션에 문자열 저장
❷ List를 session영역에 저장
❸ <a> 태그를 통해 페이지 이동

3.5 session 영역

▣ session 영역

- session영역은 웹 브라우저를 닫을때까지 공유되므로 이동한 페이지에서도 데이터를 읽어올 수 있음

예제 3-9] 03Scope/SessionLocation.jsp

```
<body>
    <h2>페이지 이동 후 session 영역의 속성 읽기</h2>
    <%
    ArrayList<String> lists = (ArrayList<String>)session.getAttribute("lists"); ❶
    for (String str : lists)  ❷
        out.print(str + "<br/>");
    %>
</body>
```

❶ session 영역에 저장된 List 컬렉션을 읽어온다. 이때 형변환 해야한다.
❷ 갯수만큼 반복해서 출력한다.

- session 영역의 속성값을 삭제하려면 웹 브라우저를 완전히 닫았다가 다시 열면 됨
- 주의) 탭만 닫아서는 session이 삭제되지 않음

3.6 application 영역

▣ application 영역

- 웹 애플리케이션은 단 하나의 application 객체만 생성하고, 클라이언트가 요청하는 모든 페이지가 application 객체를 공유
- 웹 서버가 시작될때 생성되고, 종료할때 소멸
- 따라서 웹 브라우저를 새롭게 시작해도 소멸되지 않음

예제 3-10] 03Scope/ApplicationMain.jsp

```
<body>
    <h2>application 영역의 공유</h2>
    <%
    Map<String, Person> maps = new HashMap<>();
    maps.put("actor1", new Person("이수일", 30));
    maps.put("actor2", new Person("심순애", 28));
    application.setAttribute("maps", maps);  ❷
    %>
    application 영역에 속성이 저장되었습니다.
</body>
```

❶ Map 컬렉션에 Person 객체를 2개 추가

❷ application 영역에 Map 컬렉션을 저장

3.6 application 영역

▣ application 영역

- 웹 브라우저를 완전히 닫아도 영역이 공유되므로 예제를 단독으로 실행

예제 3-11] 03Scope/ApplicationResult.jsp

```
<body>
    <h2>application 영역의 속성 읽기</h2>
    <%
    Map<String, Person> maps
            = (Map<String, Person>)application.getAttribute("maps");  ❶
    Set<String> keys = maps.keySet();  ❷
    for (String key : keys) {
        Person person = maps.get(key);
        out.print(String.format("이름 : %s, 나이 : %d<br/>",
                person.getName(), person.getAge()));
    }                                                                  ❸
    %>
</body>
```

❶ application 영역에 저장된 속성인 Map 컬렉션을 가져옴

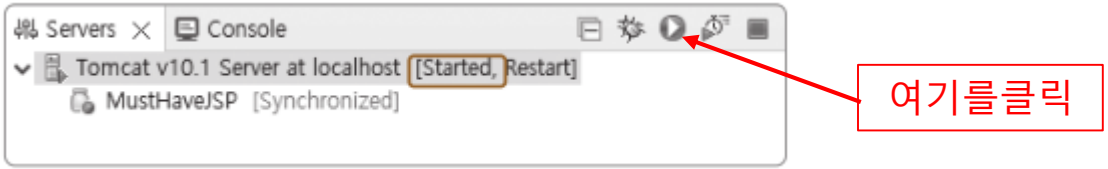
❷ Map은 key를 먼저 얻어온 후 반복해야 함

❸ key의 갯수만큼 반복하여 value를 읽어와서 출력

3.6 application 영역

▣ application 영역

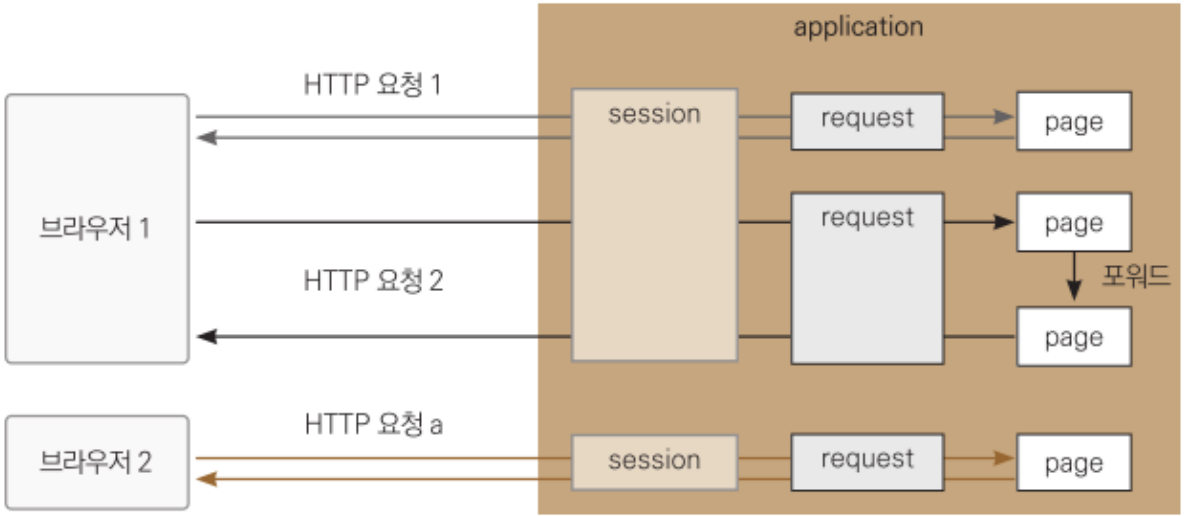
- application 영역에 저장된 속성을 삭제하고 싶다면 서버를 재시작하면 됨
- 이클립스 하단의 [Servers] 탭에서 Restart 버튼을 클릭



- 예제 3-11]을 단독으로 재실행하면 NullPointerException 이 발생됨

학습 마무리

- page 영역 : 동일한 페이지에서만 공유되고, 페이지를 벗어나면 소멸.
- request 영역 : 하나의 요청에 의해 호출된 페이지와 포워드(요청 전달)된 페이지까지 공유. 페이지 이동시 소멸.
- session 영역 : 클라이언트가 최초 접속한 후 웹 브라우저를 닫을 때까지 공유. 페이지 이동시에도 소멸되지 않음.
- application 영역 : 한 번 저장되면 웹 애플리케이션이 종료될 때까지 유지됨. 서버가 셧다운 되었을때 소멸



THE END