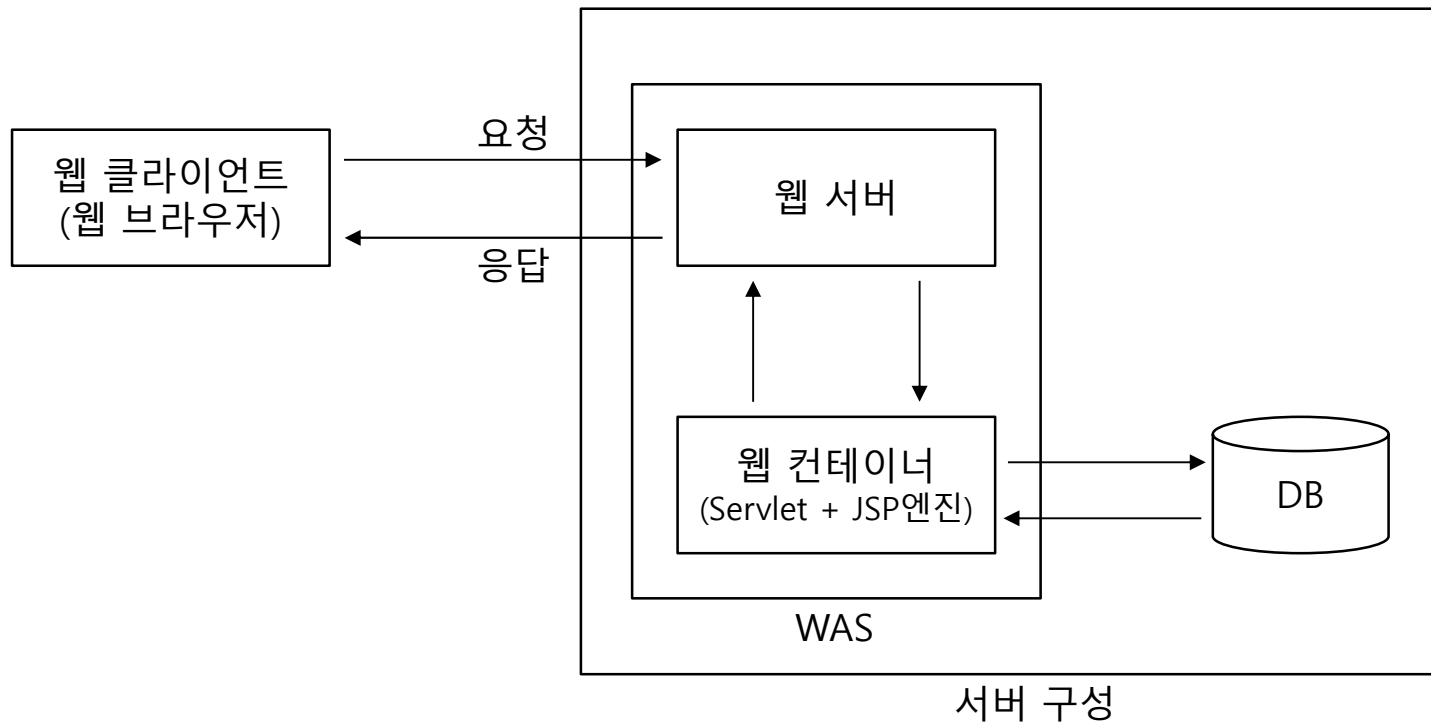


JSP(Java Server Page)

1. JSP 기본

JSP(Java Server Page)

- 동적 웹 페이지를 만들기 위한 자바 서버 프로그래밍 기술
 - 서블릿과 달리 HTML 코드 내에 Java 코드를 삽입함으로써 코드 작성량 감소
(내부적으로는 서블릿으로 실행됨)

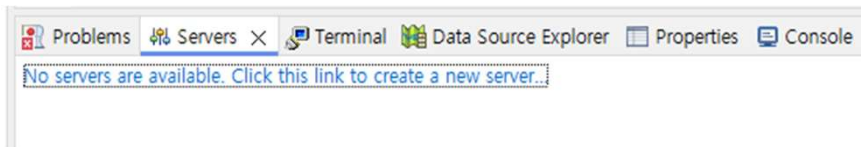


- ① 클라이언트의 요청이 들어오면 웹 컨테이너는 해당 .jsp 파일을 서블릿 코드로 변환
- ② 변환된 서블릿 코드를 클래스 파일로 컴파일
- ③ 서블릿 엔진에서 서블릿 실행

개발환경 구축(1)

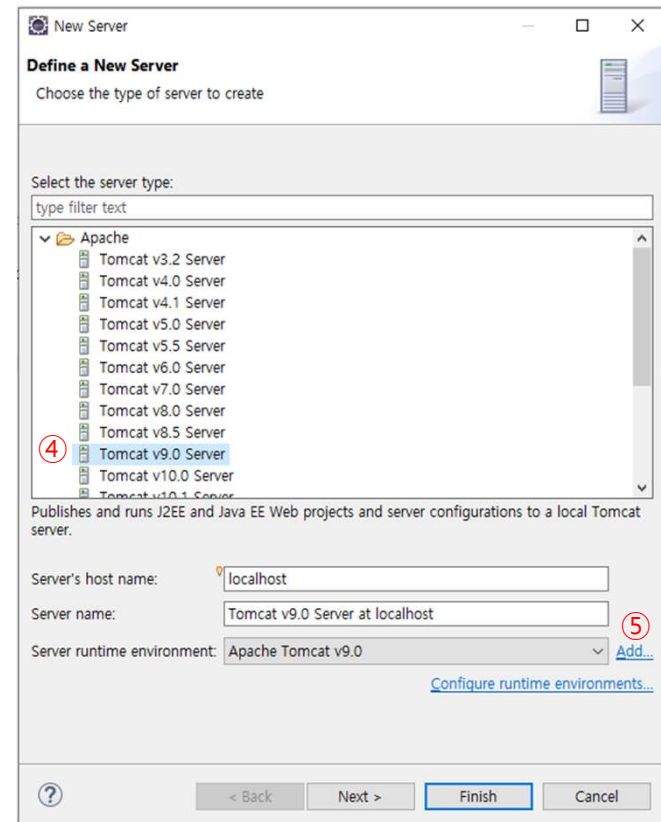
- 톰캣(WAS) 설치

- ① <https://tomcat.apache.org/download-90.cgi> 에서 [64-bit Windows zip](#) 파일 다운로드
- ② 압축을 푼 후, 적절한 위치에 폴더 복사
- ③ 이클립스의 서버 탭에서 'No servers are available. Click ...' 를 클릭



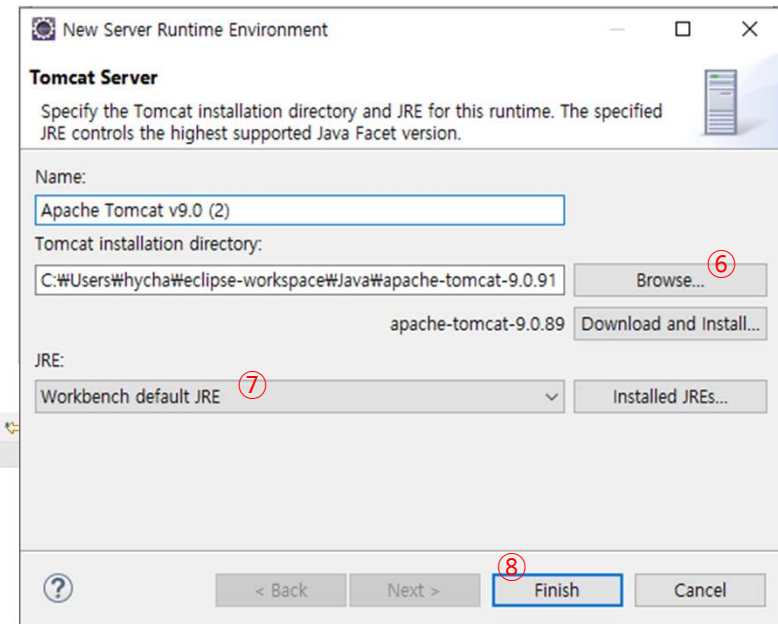
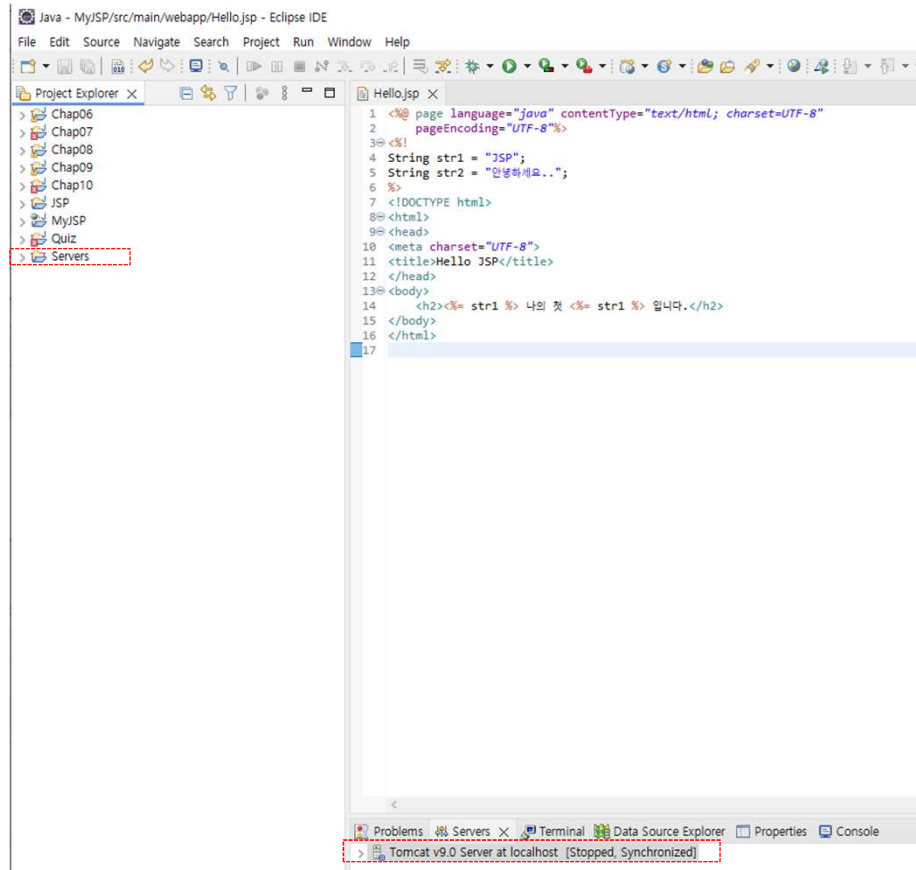
- ④ Apache > Tomcat v9.0 Server 선택

- ⑤ 'Add...' 클릭



개발환경 구축(2)

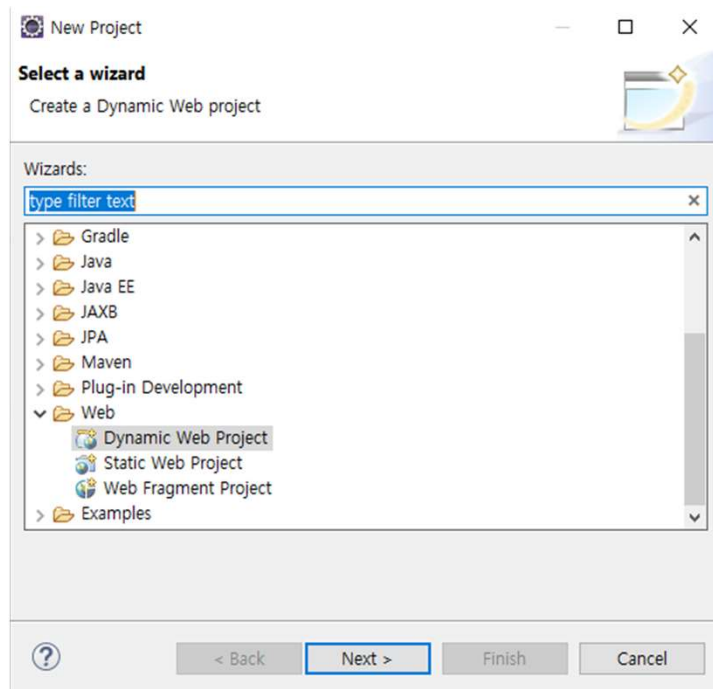
- ⑥ 'Browse' 버튼을 클릭 후, 톰캣 설치 위치 설정
- ⑦ 현재 설치된 JDK 버전(JDK 22) 를 선택
- ⑧ 'Finish' 버튼 클릭
- ⑨ 왼쪽 'Project Explorer' 창의 프로젝트 트리에 'Servers' 가 보이는지 확인



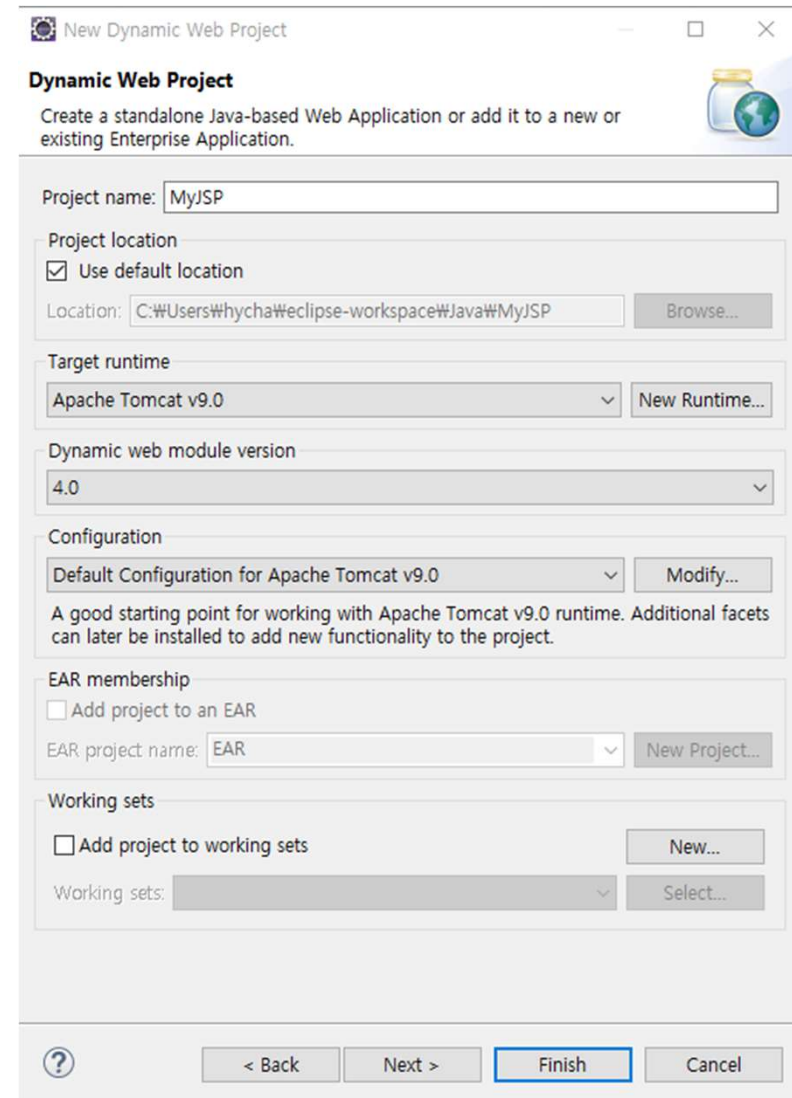
첫 JSP 구동(1)

- Hello.jsp

① File > New > Project... 로 New Project 창을 띄우고 Web > Dynamic Web Project 선택 후, 'Next' 클릭



② Project Name 입력 후, 'Finish' 클릭



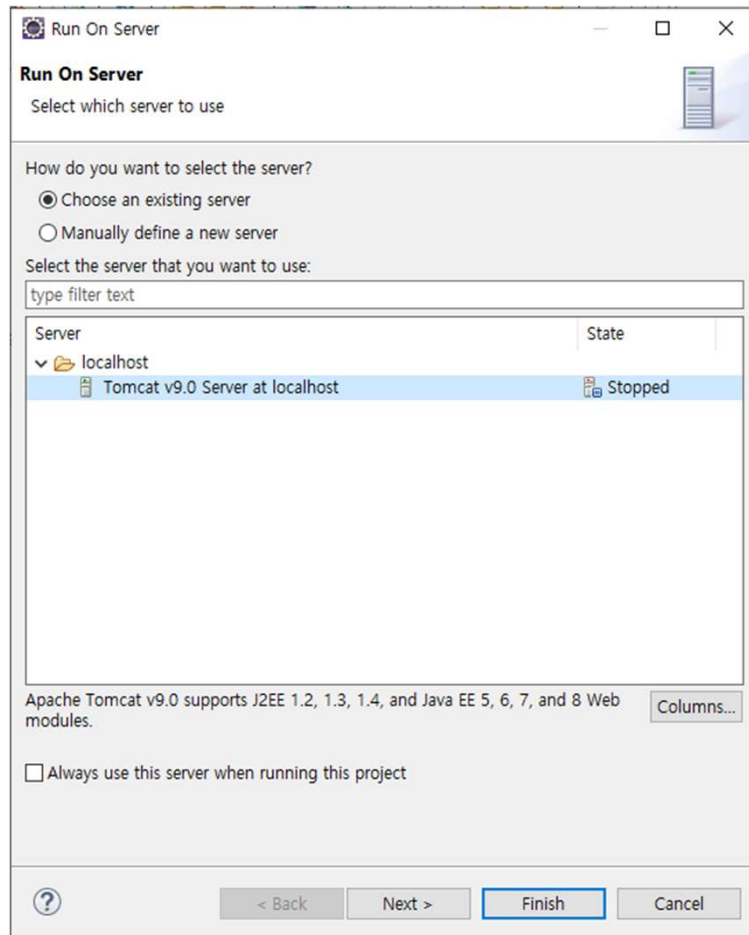
첫 JSP 구동(2)

- ③ Project Explorer 창의 프로젝트명 위에 마우스를 대고 오른쪽 버튼 클릭 후, New > JSP File 선택
- ④ 아래의 코드 입력

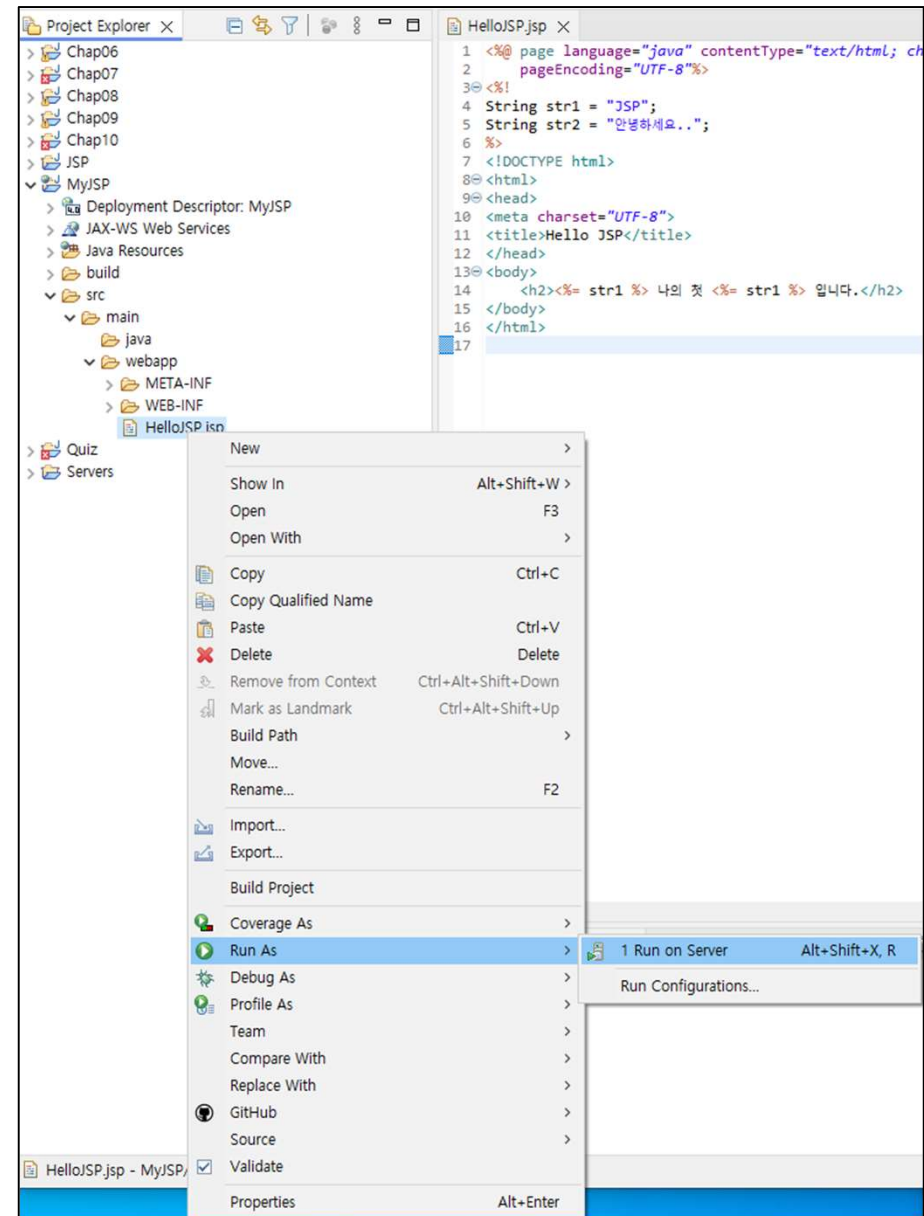
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%!
String s1 = "JSP";
String s2 = "안녕하세요. ";
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Hello JSP</title>
</head>
<body>
<h2>나의 첫 <%= s1 %> 입니다.</h2>
<%
out.println(s2 + s1 + " 세상에 오신 것을 환영합니다.");
%>
</body>
</html>
```

첫 JSP 구동(3)

- ⑤ 소스 파일 위에 마우스를 대고 오른쪽 버튼 클릭 후, Run As > Run on Server 선택



- ⑥ 서버 선택 후, Finish 버튼 클릭



JSP 파일 구조

- 지시어(Directive), 스크립트 요소(Script Elements), 주석(Comment)로 구분
 - 스크립트 요소는 선언부(Declaration), 스크립틀릿(Scriptlet), 표현식(Expression)으로 구분

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

지시어
(Directive)

```
<%!  
String s1 = "JSP";  
String s2 = "안녕하세요. ";  
%>
```

스크립트요소(선언부-Declaration)

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="UTF-8">  
<title>Hello JSP</title>
```

```
</head>
```

```
<body>
```

```
<!-- 이것도 주석입니다. -->
```

```
<%-- 이것은 주석입니다. --%>
```

주석(Comment)

```
<h2>나의 첫 <%= s1 %> 입니다.</h2>
```

스크립트요소(표현식-Expression)

```
<%  
out.println(s2 + s1 + " 세상에 오신 것을 환영합니다.");  
%>
```

스크립트요소
(스크립틀릿
-Scriptlet)

```
</body>
```

```
</html>
```

지시어(Directive)

- 현재의 JSP 코드를 컨테이너에서 처리하는데 필요한 각종 속성을 기술
 - JSP 코드를 서블릿 코드로 변환하는 데 필요한 메타 정보
 - 다음과 같은 3가지 종류의 지시어가 있음
 - page : JSP 페이지에 대한 정보 설정
 - include : 외부 파일을 현재 JSP 페이지에 포함시킴
 - taglib : 표현 언어에서 사용할 자바 클래스나 JSTL을 선언

[page 지시어 속성]

| 속성 | 내용 | 기본값 |
|--------------|---|------------|
| info | 페이지에 대한 설명 | |
| language | 페이지에서 사용할 스크립팅 언어 | java |
| contentType | 페이지에서 생성할 MIME 타입 및 캐릭터 셋 설정 | |
| pageEncoding | charset과 같이 인코딩 지정 | ISO-8859-1 |
| import | 페이지에서 사용할 자바 패키지와 클래스 지정 | |
| session | 세션 사용 여부 | true |
| buffer | 출력 버퍼 크기 지정(버퍼를 사용하지 않으려면 'none'으로 지정) | 8KB |
| autoFlush | 출력버퍼가 모두 채워졌을 때 자동으로 버퍼를 비울지를 결정 | true |
| errorPage | 해당 페이지에서 에러가 발생했을 때, 에러 발생 여부를 보여준 페이지 지정 | |
| isErrorPage | 해당 페이지가 에러를 처리할 지 여부 | false |

실습(1) – page 지시어

[Ex1] import.jsp

```
<%@ page info="import에 대한 예제" %>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.text.SimpleDateFormat"%>__<!--필요한 외부 클래스 импорт-->
<%@ page import="java.util.Date"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>page 지시어 - import 속성</title>
</head>
<body>
<%
Date today = new Date( );
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
String todayStr = dateFormat.format(today);
out.println("오늘 날짜 : " + todayStr);
%>
</body>
</html>
```

실습(2) – page 지시어

[Ex2-1] errorHandler.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"
errorPage = "errorPage.jsp"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>page 지시어 - errorPage, isErrorPage 속성</title>
</head>
<body>
<%
int myAge = Integer.parseInt(request.getParameter("age")) + 10; // 에러 발생
out.println("10년 후 당신의 나이는 " + myAge + "입니다."); // 실행되지 않음
%>
</body>
</html>
```

실습(3) – page 지시어

[Ex2-2] errorPage.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"
isErrorPage="true"%> <!--isErrorPage 속성에 true를 지정-->
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>page 지시어 - errorPage/isErrorPage 속성</title>
</head>
<body>
<h2>서비스 중 일시적인 오류가 발생하였습니다.</h2>
<p>
오류명 : <%= exception.getClass().getName() %> <br />
오류 메시지 : <%= exception.getMessage() %>
</p>
</body>
</html>
```

실습(4) – page 지시어

[Ex3] buffer.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" buffer="1kb" autoFlush="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>page 지시어 - buffer, autoFlush 속성</title>
</head>
<body>
<%
for (int i = 1; i <= 100; i++) {
out.println("abcde12345");
}
%>
</body>
</html>
```

위의 page 지시문에서 autoFlush="true"로 수정하여 실행

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" buffer="1kb" autoFlush="true"%>
```

위의 page 지시문에서 buffer="8kb"로 수정하여 실행

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" buffer="8kb" autoFlush="true"%>
```

실습(5) – include 지시어

[Ex4-1] include1.jsp

```
<%@ page import="java.time.LocalDateTime"%>
<%@ page import="java.time.LocalDate"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<% LocalDate today = LocalDate.now(); %>
```

[Ex4-2] include2.jsp

```
<%@ page import="java.time.LocalDateTime"%>
<%@ page import="java.time.LocalDate"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<% LocalDateTime tomorrow = LocalDateTime.now().plusDays(1); %>
```

[Ex4] includeMain.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ include file = "include1.jsp" %>
<%@ include file = "include2.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>include 지시어</title>
</head>
<body>
<%
out.println("오늘 날짜 : " + today);
out.println("<br/>");
out.println("내일 날짜 : " + tomorrow);
%>
</body>
</html>
```

스크립트 요소(Script Elements)(1)

- 선언부(Declaration)

- 스크립틀릿이나 표현식에서 사용할 멤버 변수나 메서드를 선언하는 블록

```
. . .  
<%!  
String str = "문자열";  
int num = 10;  
%>
```

- 표현식(Expression)

- 변수의 출력이나 메소드 호출을 하는 블록
- out.println()과 같은 자바 출력 메소드를 이용하지 않고 간단하게 출력하기 위한 기능

```
. . .  
<%!  
String str = "문자열";  
int num = 10;  
%>  
<%!  
public int add(int num1, int num2) { return num1 + num2; }  
%>  
. . .  
<%= str %>  
<%= add(10, 20) %>  
. . .
```


스크립트 요소(Script Elements)(2)

- 스크립틀릿(Scriptlet)
 - JSP 문서 내에서 자바 코드를 작성하는 영역
 - 순수 자바 코드로만 작성되어야 하며, 메서드에 대한 선언은 할 수 없다.
(메서드 선언은 선언부에서 해야 함)

```
. . .  
<%  
String str = "test";  
for(int i = 0; i < 10; i++){  
    out.println(str);  
}  
%>  
. . .
```

실습 - 스크립트 요소

[Ex1] gugudan.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>구구단</title>
</head>
<body>
<%
for(int i = 2; i < 10; i++){
for(int j = 1; j < 10; j++){
out.println(i + " * " + j + " = " + i * j);
out.println("<br>");
}
out.println("<hr>");
}
}%>
</body>
</html>
```

2. 내장 객체(Implicit Object)

내장객체(Implicit Object)

- JSP 코드 내에서 객체 생성 없이 사용 가능한 객체
 - 서블릿 코드로 변환 시, 서블릿 코드 안에 자동으로 선언되는 객체
 - 스크립틀릿과 표현식 블록 안에서만 사용 가능

| 내장객체 | 객체 타입 | 설명 |
|-------------|--|---|
| request | javax.servlet.http.HttpServletRequest | 클라이언트의 요청 정보를 가지고 있는 객체 |
| response | javax.servlet.http.HttpServletResponse | 클라이언트의 요청에 대한 응답 정보를 가지고 있는 객체 |
| pageContext | javax.servlet.jsp.PageContext | 실행되는 JSP 페이지에 대한 context 정보를 가지고 있는 객체 |
| session | javax.servlet.http.HttpSession | 클라이언트의 세션을 유지하기 위한 정보를 가지고 있는 객체 |
| application | javax.servlet.ServletContext | 웹 서버의 어플리케이션 처리와 관련된 정보를 가지고 있는 객체 |
| out | javax.servlet.jsp.JspWriter | 현재 실행되는 JSP 페이지에 출력할 내용을 가지고 있는 출력 스트림 객체 |
| config | javax.servlet.ServletConfig | 현재 실행되는 JSP 페이지의 설정 관련 정보를 가지고 있는 객체 |
| page | java.lang.Object | 현재 JSP 페이지를 구현한 자바 클래스의 인스턴스를 가지고 있는 객체 |
| exception | java.lang.Throwable | 예외가 발생한 경우 예외를 처리하는 객체 |

실습 – request 객체

[Ex1-1] requestMain.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>내장 객체 - request</title></head>
<body>
<h2>1. 클라이언트와 서버의 환경정보 읽기</h2>
<a href="./requestWebInfo.jsp?eng=Hello&han=안녕">GET 방식 전송</a><br>
<form action="requestWebInfo.jsp" method="post">
영어 : <input type="text" name="eng" value="Bye" /><br>
한글 : <input type="text" name="han" value="잘 가" /><br>
<input type="submit" value="POST 방식 전송" />
</form>
<h2>2. 클라이언트의 요청 매개변수 읽기</h2>
<form method="post" action="requestParameter.jsp">
아이디 : <input type="text" name="id" value="" /><br>
성별 :
<input type="radio" name="sex" value="man" />남자
<input type="radio" name="sex" value="woman" checked="checked" />여자<br>
관심사항 :
<input type="checkbox" name="favo" value="eco" />경제
<input type="checkbox" name="favo" value="pol" checked="checked" />정치
<input type="checkbox" name="favo" value="ent" />연예<br>
자기소개:
<textarea name="intro" cols="30" rows="4"></textarea><br>
<input type="submit" value="전송하기" />
</form>
<h2>3. HTTP 요청 헤더 정보 읽기</h2>
<a href="requestHeader.jsp">요청 헤더 정보 읽기</a>
</body>
</html>
```

실습 – request 객체

[Ex1-2] requestWebInfo.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>내장 객체 - request</title></head>
<body>
<h2>1. 클라이언트와 서버의 환경정보 읽기</h2>
<ul>
<li>데이터 전송 방식 : <%= request.getMethod() %></li>
<li>URL : <%= request.getRequestURL() %></li>
<li>URI : <%= request.getRequestURI() %></li>
<li>프로토콜 : <%= request.getProtocol() %></li>
<li>서버명 : <%= request.getServerName() %></li>
<li>서버 포트 : <%= request.getServerPort() %></li>
<li>클라이언트 IP 주소 : <%= request.getRemoteAddr() %></li>
<li>쿼리스트링 : <%= request.getQueryString() %></li>
<li>전송된 값 1 : <%= request.getParameter("eng") %></li>
<li>전송된 값 2 : <%= request.getParameter("han") %></li>
</ul>
</body>
</html>
```

실습 – request 객체

[Ex1-3] requestParameter.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>내장 객체 - request</title></head>
<body>
<%
request.setCharacterEncoding("UTF-8");
String id = request.getParameter("id");
String sex = request.getParameter("sex");
String[] favo = request.getParameterValues("favo");
String favoStr = "";
if (favo != null) {
for (int i = 0; i < favo.length; i++) {
favoStr += favo[i] + " ";
}
}
String intro = request.getParameter("intro").replace("\r\n", "<br/>");
%>
<ul>
<li>아이디 : <%= id %></li>
<li>성별 : <%= sex %></li>
<li>관심사항 : <%= favoStr %></li>
<li>자기소개 : <%= intro %></li>
</ul>
</body>
</html>
```

실습 – request 객체

[Ex1-4] requestHeader.jsp

```
<%@ page import="java.util.Enumeration"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>내장 객체 - request</title></head>
<body>
<h2>3. 요청 헤더 정보 출력하기</h2>
<%
Enumeration headers = request.getHeaderNames();
while (headers.hasMoreElements()) {
String headerName = (String)headers.nextElement();
String headerValue = request.getHeader(headerName);
out.print("헤더명 : " + headerName + ", 헤더값 : " + headerValue + "<br/>");
}
%>
<p>이 파일을 직접 실행하면 referer 정보는 출력되지 않습니다.</p>
</body>
</html>
```


실습 – response 객체

[Ex2-1] responseMain.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>내장 객체 - response</title></head>
<body>
<h2>1. 로그인 폼</h2>
<%
String loginErr = request.getParameter("loginErr");
if (loginErr != null) out.print("로그인 실패");
%>
<form action="responseLogin.jsp" method="post">
아이디 : <input type="text" name="user_id" /><br />
패스워드 : <input type="text" name="user_pwd" /><br />
<input type="submit" value="로그인" />
</form>

<h2>2. HTTP 응답 헤더 설정하기</h2>
<form action="responseHeader.jsp" method="get">
날짜 형식 : <input type="text" name="add_date" value="2021-10-25 09:00" /><br />
숫자 형식 : <input type="text" name="add_int" value="8282" /><br />
문자 형식 : <input type="text" name="add_str" value="홍길동" /><br />
<input type="submit" value="응답 헤더 설정 & 출력" />
</form>
</body>
</html>
```

실습 – response 객체

[Ex2-2] responseLogin.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>내장 객체 - Response</title></head>
<body>
<%
String id = request.getParameter("user_id");
String pwd = request.getParameter("user_pwd");
if (id.equalsIgnoreCase("must") && pwd.equalsIgnoreCase("1234")) {
response.sendRedirect("responseWelcome.jsp");
//request.getRequestDispatcher("responseWelcome.jsp").forward(request, response);
}
else {
request.getRequestDispatcher("responseMain.jsp?loginErr=1").forward(request, response);
}
%>
</body>
</html>
```

실습 – response 객체

[Ex2-3] responseWelcome.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>내장 객체 - response</title></head>
<body>
<h2>로그인 성공</h2>
<%
String id = request.getParameter("user_id");
String pwd = request.getParameter("user_pwd");
out.println("id : " + id);
out.println("pwd : " + pwd);
%>
</body>
</html>
```

실습 – response 객체

[Ex2-4] responseHeader.jsp(1)

```
<%@page import="java.util.ArrayList"%>
<%@ page import="java.util.Collection"%>
<%@ page import="java.text.SimpleDateFormat"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%
// 응답 헤더에 추가할 값 준비
SimpleDateFormat s = new SimpleDateFormat("yyyy-MM-dd HH:mm");
long add_date = s.parse(request.getParameter("add_date")).getTime();

java.sql.Date date2 = new java.sql.Date(add_date);
System.out.println(date2);

int add_int = Integer.parseInt(request.getParameter("add_int"));
String add_str = request.getParameter("add_str");

// 응답 헤더에 값 추가
response.addDateHeader("myBirthday", add_date);
response.addIntHeader("myNumber", add_int);
response.addIntHeader("myNumber", 1004); // 추가
response.addIntHeader("myNumber", 2004); // 추가
response.addHeader("myName", add_str);
response.setHeader("myName", "안중근"); // 수정
%>
```

실습 – response 객체

[Ex2-4] responseHeader.jsp(2)

```
<html>
<head><title>내장 객체 - response</title></head>
<body>
<h2>응답 헤더 정보 출력하기</h2>
<%
Collection<String> headerNames = response.getHeaderNames();
for (String hName : headerNames) {
String hValue = response.getHeader(hName);
%>
<li><%= hName %> : <%= hValue %></li>
<%
}
%>
<h2>myNumber만 출력하기</h2>
<%
Collection<String> myNumber = response.getHeaders("myNumber");
for (String myNum : myNumber) {
%>
<li>myNumber : <%= myNum %></li>
<%
}
%>
</body>
</html>
```

실습 - out 객체

[Ex3] outObject.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>내장 객체 - out</title></head>
<body>
<%
// 버퍼 내용 삭제하기
out.print("출력되지 않는 텍스트"); // 버퍼에 저장
out.clearBuffer(); // 버퍼를 비움(위 줄의 출력 결과 사라짐)

out.print("<h2>out 내장 객체</h2>");

// 버퍼 크기 정보 확인
out.print("출력 버퍼의 크기 : " + out.getBufferSize() + "<br>");
out.print("남은 버퍼의 크기 : " + out.getRemaining() + "<br>");

out.flush(); // 버퍼 내용 출력
out.print("flush 후 버퍼의 크기 : " + out.getRemaining() + "<br>");

// 다양한 타입의 값 출력
out.print(1);
out.print(false);
out.print('가');
%>
</body>
</html>
```

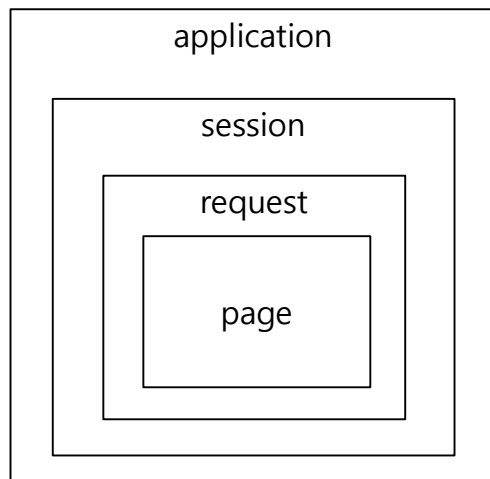
실습 – session 객체

[Ex4] sessionObject.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>session 객체 예제</title>
</head>
<body>
<h2>session 예제</h2>
<hr>
<%
if(session.isNew()){
out.println("<script> alert('세션이 해제되어 다시 설정합니다.') </script>");
session.setAttribute("login", "홍길동");
}
%>
<%=session.getAttribute("login") %>님 환영합니다.<br>
세션 ID : <%= session.getId() %><br>
<% session.setMaxInactiveInterval(30); %>
세션 유지시간 : <%= session.getMaxInactiveInterval() %><br>
</body>
</html>
```

내장객체 영역(Scope)

- 각 Object의 정보가 저장되는 유효 기간 또는 영역을 의미
- 4개의 내장 객체 영역으로 구분
 - page 영역 : 동일한 페이지에서만 유지되며 페이지를 벗어나면 소멸
 - request 영역 : 하나의 요청에 의해 호출된 페이지와 포워드된 페이지까지 유지
 - session 영역 : 클라이언트가 처음 접속한 후 웹 브라우저를 닫을 때까지 유지
 - application 영역 : 한 번 저장되면 웹 애플리케이션이 종료될 때까지 유지



각 영역은 하위 영역을 1개 이상 포함할 수 있음
(즉, session은 1개 이상의 request 영역을 포함할 수 있음)

실습 – page 영역

[Ex1-1] Person.java

프로젝트의 Java Resources > src/main/java > New > Class 를 통해 파일 생성

```
package common;

public class Person {
    private String name;
    private int age;
    public Person() {}

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

실습 – page 영역

[Ex1-2] PageContextMain.jsp

```
<%@ page import="common.Person"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%
pageContext.setAttribute("pageInteger", 1000);
pageContext.setAttribute("pageString", "페이지 영역의 문자열");
pageContext.setAttribute("pagePerson", new Person("한석봉", 99));
%>
<html>
<head><title>page 영역</title></head>
<body>
<h2>page 영역의 속성값 읽기</h2>
<%
int pInteger = (Integer)(pageContext.getAttribute("pageInteger"));
String pString = pageContext.getAttribute("pageString").toString();
Person pPerson = (Person)(pageContext.getAttribute("pagePerson"));
%>
<ul>
<li>Integer 객체 : <%= pInteger %></li>
<li>String 객체 : <%= pString %></li>
<li>Person 객체 : <%= pPerson.getName() %>, <%= pPerson.getAge() %></li>
</ul>

<h2>include된 파일에서 page 영역 읽어오기</h2>
<%@ include file="PageInclude.jsp" %>
<h2>페이지 이동 후 page 영역 읽어오기</h2>
<a href="PageLocation.jsp">PageLocation.jsp 바로가기</a>
</body>
</html>
```

실습 – page 영역

[Ex1-3] 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>
```

실습 - page 영역

[Ex1-4] PageLocation.jsp

```
<%@ page import="common.Person"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<html>
<head><title>page 영역</title></head>
<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>
</html>
```

실습 – request 영역(1)

[Ex2-1] RequestScopeMain.jsp

```
<%@ page import="common.Person"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%
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
%>
<h2>request 영역의 속성값 읽기</h2>
<%
Person rPerson = (Person)(request.getAttribute("requestPerson"));
%>
<ul>
<li>String 객체 : <%= request.getAttribute("requestString") %></li>
<li>Person 객체 : <%= rPerson.getName() %>, <%= rPerson.getAge() %></li>
</ul>
<h2>포워드된 페이지에서 request 영역 속성값 읽기</h2>
<%
request.getRequestDispatcher("RequestForward.jsp?paramHan=한글&paramEng=English")
.forward(request, response);
%>
</body>
</html>
```

실습 – request 영역(1)

[Ex2-2] RequestForward.jsp

```
<%@ page import="common.Person"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>request 영역</title></head>
<body>
<h2>포워드로 전달된 페이지</h2>
<h4>RequestMain 파일의 리퀘스트 영역 속성 읽기</h4>
<%
Person pPerson = (Person)(request.getAttribute("requestPerson"));
%>
<ul>
<li>String 객체 : <%= request.getAttribute("requestString") %></li>
<li>Person 객체 : <%= pPerson.getName() %>, <%= pPerson.getAge() %></li>
</ul>
<h4>매개변수로 전달된 값 출력하기</h4>
<%
request.setCharacterEncoding("UTF-8");
out.println(request.getParameter("paramHan"));
out.println(request.getParameter("paramEng"));
%>
</body>
</html>
```

실습 – request 영역(2)

[Ex3-1] RequestScopeMain2.jsp

```
<%@ page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%
ArrayList<String> lists = new ArrayList<String>();
lists.add("리스트");
lists.add("컬렉션");
request.setAttribute("lists", lists);
%>
<html>
<head><title>request 영역</title></head>
<body>
<h2>페이지 이동 후 request 영역의 속성 읽기</h2>
<%
request.getRequestDispatcher("RequestLocation2.jsp")
.forward(request, response);
%>
>
</body>
</html>
```

실습 – request 영역(2)

[Ex3-2] RequestLocation2.jsp

```
<%@ page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>request 영역</title></head>
<body>
<h2>페이지 이동 후 request 영역의 속성 읽기</h2>
<%
ArrayList<String> lists = (ArrayList<String>)request.getAttribute("lists");
for (String str : lists)
out.print(str + "<br/>");
%>
</body>
</html>
```


실습 – session 영역

[Ex4-1] SessionMain.jsp

```
<%@ page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%
ArrayList<String> lists = new ArrayList<String>();
lists.add("리스트");
lists.add("컬렉션");
session.setAttribute("lists", lists);
%>
<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>
</body>
</html>
```

실습 – session 영역

[Ex4-2] SessionLocation.jsp

```
<%@ page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>session 영역</title></head>
<body>
<h2>페이지 이동 후 session 영역의 속성 읽기</h2>
<%
ArrayList<String> lists = (ArrayList<String>)session.getAttribute("lists");
for (String str : lists)
out.print(str + "<br/>");
%>
</body>
</html>
```

실습 – application 영역

[Ex5-1] ApplicationMain.jsp

```
<%@ page import="java.util.HashMap"%>
<%@ page import="common.Person"%>
<%@ page import="java.util.Map"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>application 영역</title></head>
<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);
%>
application 영역에 속성이 저장되었습니다.
</body>
</html>
```

실습 – application 영역

[Ex5-2] ApplicationResult.jsp

```
<%@ page import="java.util.Set"%>
<%@ page import="common.Person"%>
<%@ page import="java.util.Map"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head><title>application 영역</title></head>
<body>
<h2>application 영역의 속성 읽기</h2>
<%
    Map<String, Person> maps
    = (Map<String, Person>)application.getAttribute("maps");
    Set<String> keys = maps.keySet();
    for (String key : keys) {
        Person person = maps.get(key);
        out.print(String.format("이름 : %s, 나이 : %d<br/>",
            person.getName(), person.getAge()));
    }
%>
</body>
</html>
```

3. 쿠키(Cookie)&세션(Session)

쿠키(Cookie)

- 클라이언트의 상태 정보를 유지하기 위해 클라이언트에 저장되는 값
 - 서버에서 생성된 정보이며, 보안상 문제가 발생할 여지 있음
 - 해당 서버에 재요청 시, 저장된 쿠키 정보는 HTTP 헤더에 실려 서버로 전송됨
 - 쿠키의 값에는 콤마, 콜론, 공백, 세미콜론을 포함할 수 없음

| 쿠키 속성 | 설명 | 메소드 |
|---------------|----------------|--|
| 이름(name) | 쿠키를 구별하는 이름 | String getName() * name은 Cookie 인스턴스 생성 시, 생성자를 통해 초기화하기 때문에 setName() 메소드가 없음(즉, 한번 name이 지정되면 변경 안 됨) |
| 값(value) | 쿠키에 저장할 실제 데이터 | void setValue(String value) String getValue() |
| 도메인(domain) | 쿠키를 적용할 도메인 | void setDomain(String domain) String getDomain() |
| 경로(path) | 쿠키를 적용할 경로 | void setPath(String path) String getPath() |
| 유지기간(max age) | 쿠키를 유지할 시간 | void setMaxAge(int seconds) int getMaxAge() |

실습 – Cookie

[Ex1-1] CookieMain.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<html>
<head><title>Cookie</title></head>
<body>
<h2>1. 쿠키(Cookie) 설정</h2>
<%
Cookie cookie = new Cookie("myCookie", "쿠키맛나요"); // 쿠키 생성
cookie.setPath(request.getContextPath()); // 경로를 컨텍스트 루트로 설정
cookie.setMaxAge(3600); // 유지 기간을 1시간으로 설정
response.addCookie(cookie); // 응답 헤더에 쿠키 추가
%>

<h2>2. 쿠키 설정 직후 쿠키값 확인하기</h2>
<%
Cookie[] cookies = request.getCookies(); // 요청 헤더의 모든 쿠키 얻기
if (cookies!=null) {
for (Cookie c : cookies) { // 쿠키 각각의
String cookieName = c.getName(); // 쿠키 이름 얻기
String cookieValue = c.getValue(); // 쿠키 값 얻기
// 화면에 출력
out.println(String.format("%s : %s<br/>", cookieName, cookieValue));
}
}
%>

<h2>3. 페이지 이동 후 쿠키값 확인하기</h2>
<a href="CookieResult.jsp">
다음 페이지에서 쿠키값 확인하기
</a>
</body>
</html>
```

실습 - Cookie

[Ex1-2] CookieResult.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>CookieResult.jsp</title>
</head>
<body>
<h2>쿠키값 확인하기(쿠키가 생성된 이후의 페이지)</h2>
<%
Cookie[] cookies = request.getCookies();
if (cookies != null) {
for (int i = 0; i < cookies.length; i++) {
String cookieName = cookies[i].getName();
String cookieValue = cookies[i].getValue();
out.println(String.format("쿠키명 : %s - 쿠키값 : %s<br/>",
cookieName, cookieValue));
}
}
%>
</body>
</html>
```


실습 – Cookie

[Ex2] CookieSpace.jsp(쿠키값에 공백문자 포함)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import = "java.net.URLEncoder"%>
<%@ page import = "java.net.URLDecoder"%>
<html>
<head><title>Cookie</title></head>
<body>
<h2>1. 쿠키(Cookie) 설정</h2>
<%
String value = URLEncoder.encode("쿠키 맞나요", "UTF-8");
Cookie cookie = new Cookie("myCookie", value); // 쿠키 생성
cookie.setPath(request.getContextPath()); // 경로를 컨텍스트 루트로 설정
cookie.setMaxAge(3600); // 유지 기간을 1시간으로 설정
response.addCookie(cookie); // 응답 헤더에 쿠키 추가
%>
<h2>2. 쿠키 설정 직후 쿠키값 확인하기</h2>
<%
Cookie[] cookies = request.getCookies(); // 요청 헤더의 모든 쿠키 얻기
if (cookies!=null) {
for (Cookie c : cookies) { // 쿠키 각각의
String cookieName = c.getName(); // 쿠키 이름 얻기
String cookieValue = URLDecoder.decode(c.getValue(), "UTF-8"); // 쿠키 값 얻기
// 화면에 출력
out.println(String.format("%s : %s<br/>", cookieName, cookieValue));
}
}
%>
<h2>3. 페이지 이동 후 쿠키값 확인하기</h2>
<a href="CookieResult.jsp">
다음 페이지에서 쿠키값 확인하기
</a>
</body>
</html>
```

세션(Session)

- 서버에서 관리하는 클라이언트와의 연결 상태 정보
 - 서버에서 생성하는 세션 아이디를 통해 구분
 - 세션 유효 시간이나 브라우저 종료 전까지 유지되기 때문에 로그인 상태 유지, 장바구니 등 다양한 상황에서 사용됨

| 주요 메소드 | 설명 |
|---|----------------------------------|
| String getId() | 각 접속에 대한 세션 고유의 ID를 문자열 형태로 반환 |
| HttpSession getSession() | 요청한 클라이언트에 지정된 HttpSession 객체 반환 |
| long getCreationTime() | 현재 세션 생성 시간 반환 |
| long getLastAccessedTime() | 현재 세션으로 클라이언트에서 마지막으로 접근한 시간 반환 |
| int getMaxInactiveInterval() | 설정된 세션 유효시간 반환 |
| void setMaxInactiveInterval(int t) | 세션 유효시간 설정(초 단위) |
| void invalidate() | 세션을 종료시키고 세션의 모든 데이터를 삭제 |
| Object getAttribute(String name) | 세션의 속성을 반환 |
| void setAttribute(String name, Object attr) | 세션에 속성값을 저장 |

실습 – Session

[Ex3-1] SessionMain2.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>session 객체 예제</title>
</head>
<body>
<h2>session 예제</h2>
<hr>
<%
if(session.isNew()){
out.println("<script>alert('세션이 해제되어 다시 설정합니다.')
```

실습 – Session

[Ex3-2] Session2.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>session 객체 예제</title>
</head>
<body>
<h2>session 예제</h2>
<hr>
<%
if(session.isNew()){
out.println("<script>alert('세션이 해제되어 다시 설정합니다.')
```

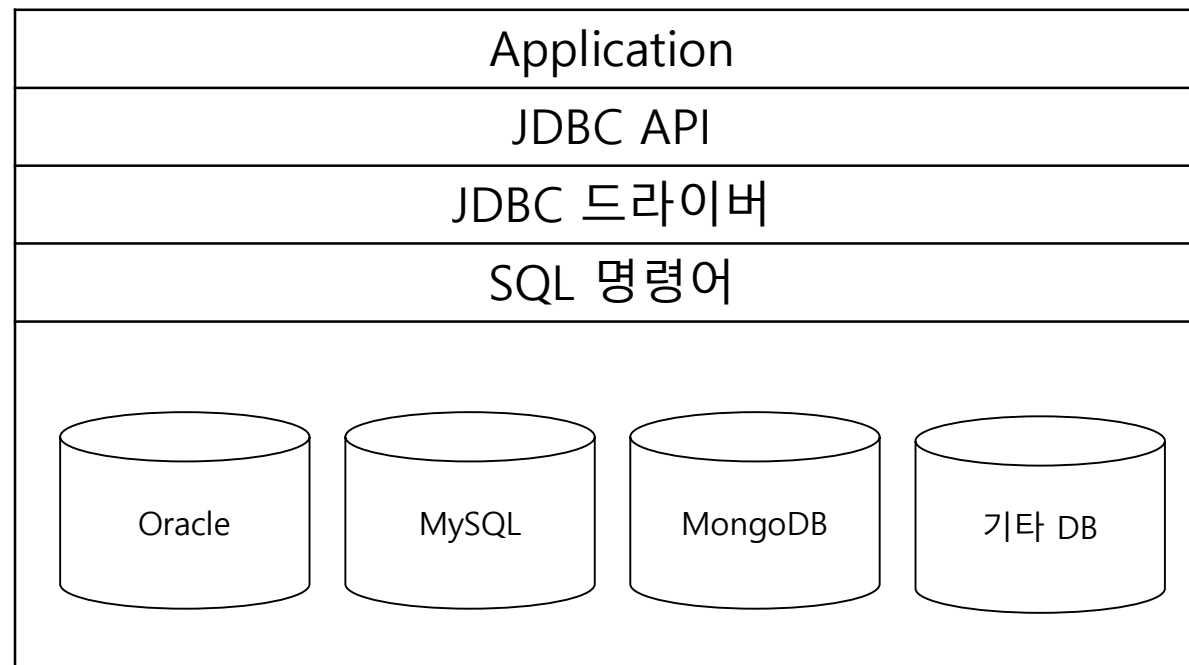
쿠키 vs. 세션

| | 쿠키 | 세션 |
|---------|-------------------------------|--|
| 저장위치/형식 | 클라이언트에 텍스트로 저장 | 서버에 Object 타입으로 저장 |
| 보안 | 보안에 취약 | 비교적 안전 |
| 자원/속도 | 서버 자원을 사용하지 않으므로 세션보다 빠름 | 서버에서 처리하기 때문에 쿠키보다 느림 |
| 크기 | 크기 제한 있음(브라우저마다 다름) | 서버가 허용하는 한 제한 없음 |
| 유지시간 | 쿠키 생성 시 설정(설정 시간이 지나면 무조건 삭제) | 서버에서 설정(설정 시간이 지나도 동작이 있다면 삭제되지 않고 유지) |

4. 데이터베이스 연동

JDBC

- Java DataBase Connectivity의 약자로 자바 코드와 다양한 데이터베이스 간의 연동에 대한 표준 인터페이스 규격을 일컬음
 - 개발자는 데이터베이스 종류와 상관없이 표준 API를 이용하여 연동 프로그램 구현 가능



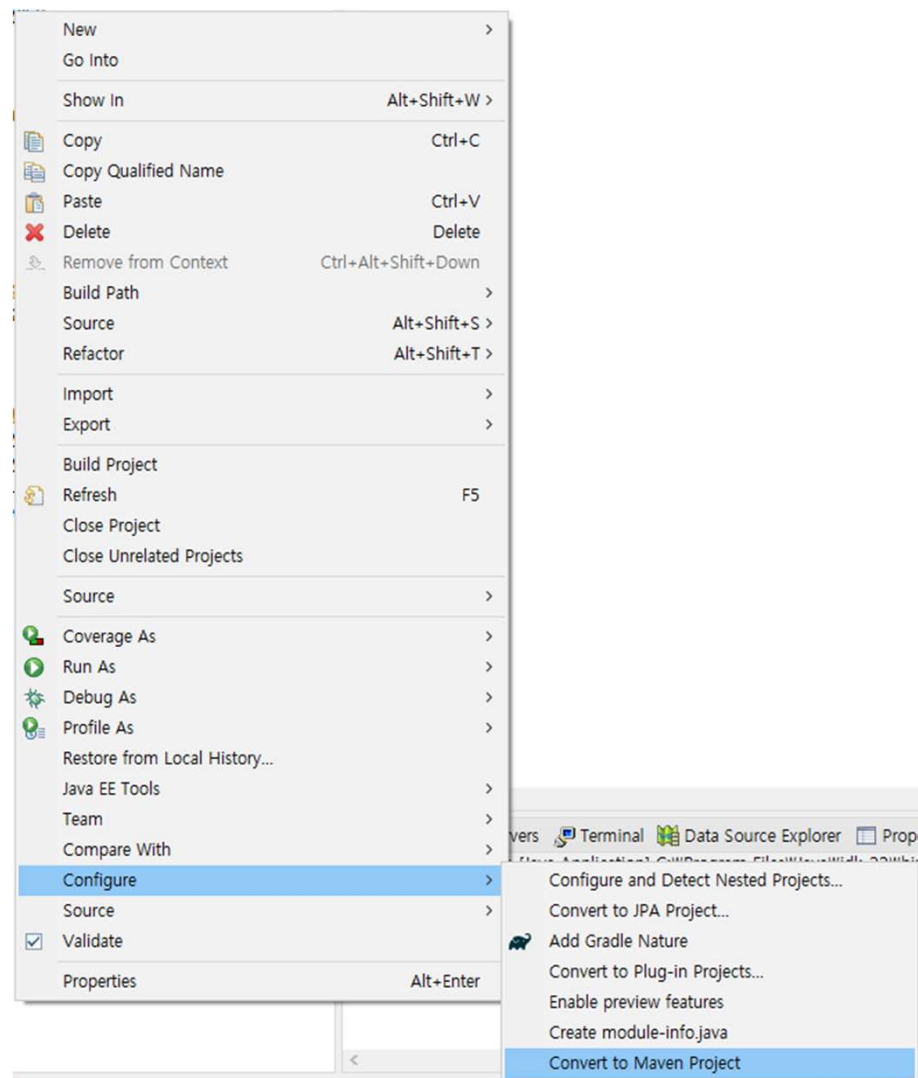
JDBC 드라이버 설치(1)

- Maven이라는 자동 빌드 도구를 이용해 프로그램 빌드 시 설치 가능

① Maven 프로젝트로 변환

'Project Explorer'의 프로젝트명 선택 후
마우스 오른쪽 버튼을 눌러

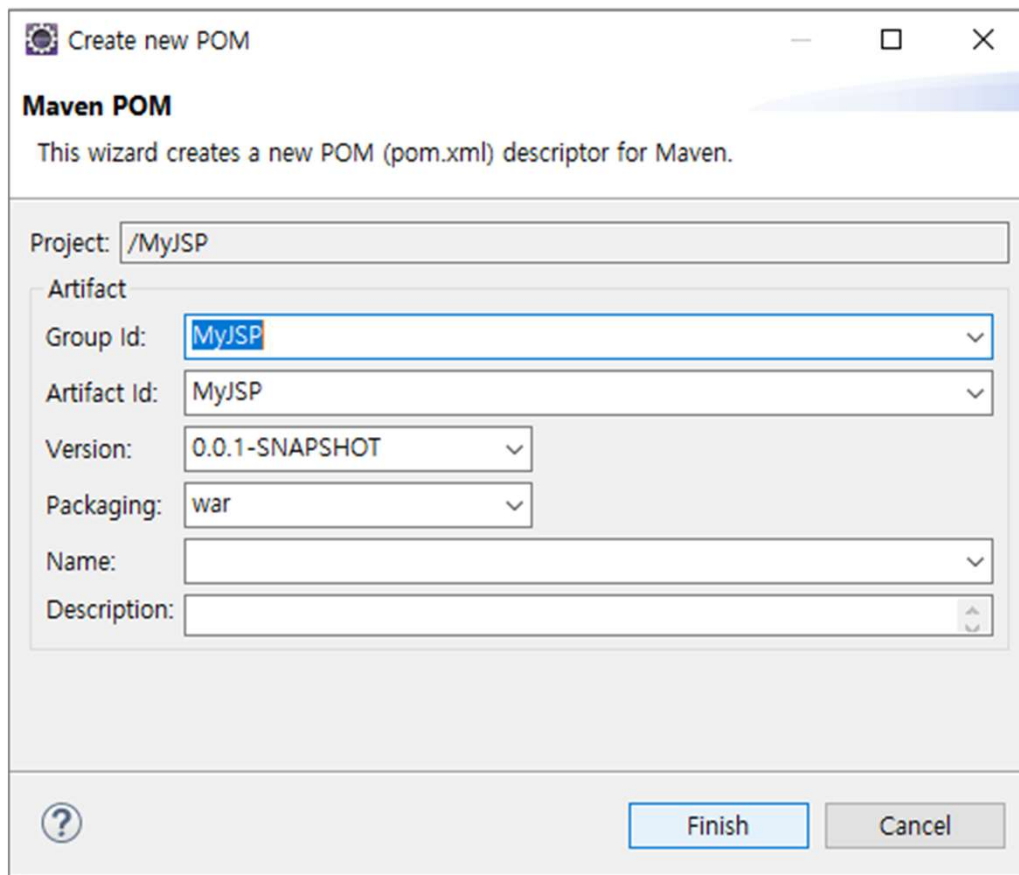
Configure > Conver to Maven Project
선택



JDBC 드라이버 설치(2)

② Maven POM 파일 기본 정보 설정

그대로 'finish' 클릭



Create new POM

Maven POM
This wizard creates a new POM (pom.xml) descriptor for Maven.

Project: /MyJSP

Artifact

Group Id: MyJSP

Artifact Id: MyJSP

Version: 0.0.1-SNAPSHOT

Packaging: war

Name:

Description:

Finish Cancel

JDBC 드라이버 설치(3)

③ 'Project Explorer'의 프로젝트 트리 아래에 생성된 'pom.xml' 파일 수정

아래 내용 추가

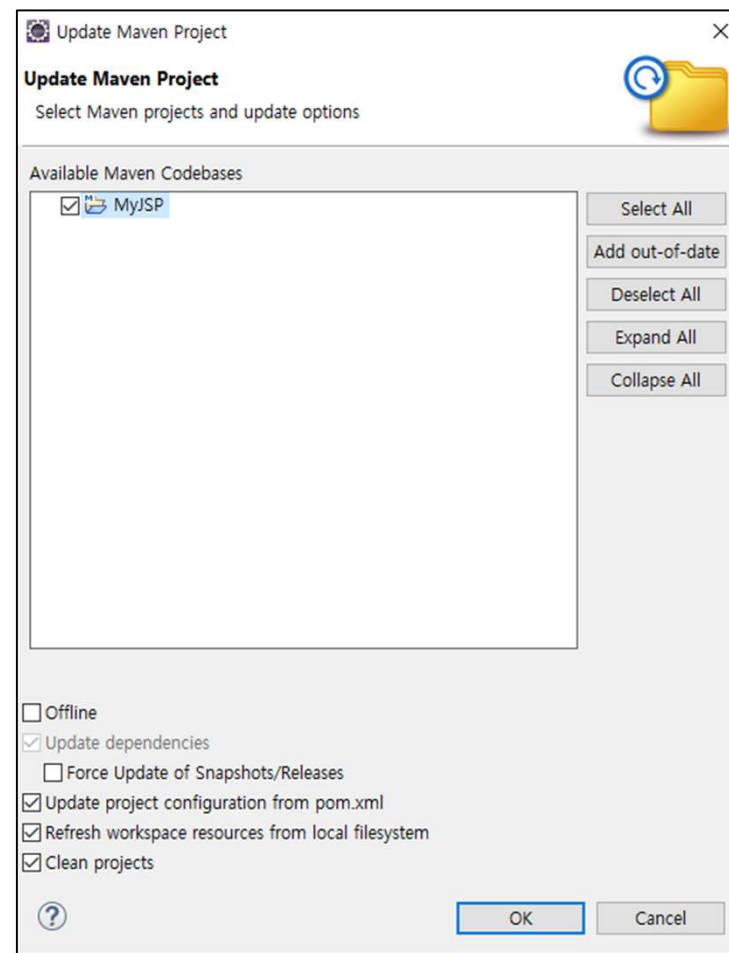
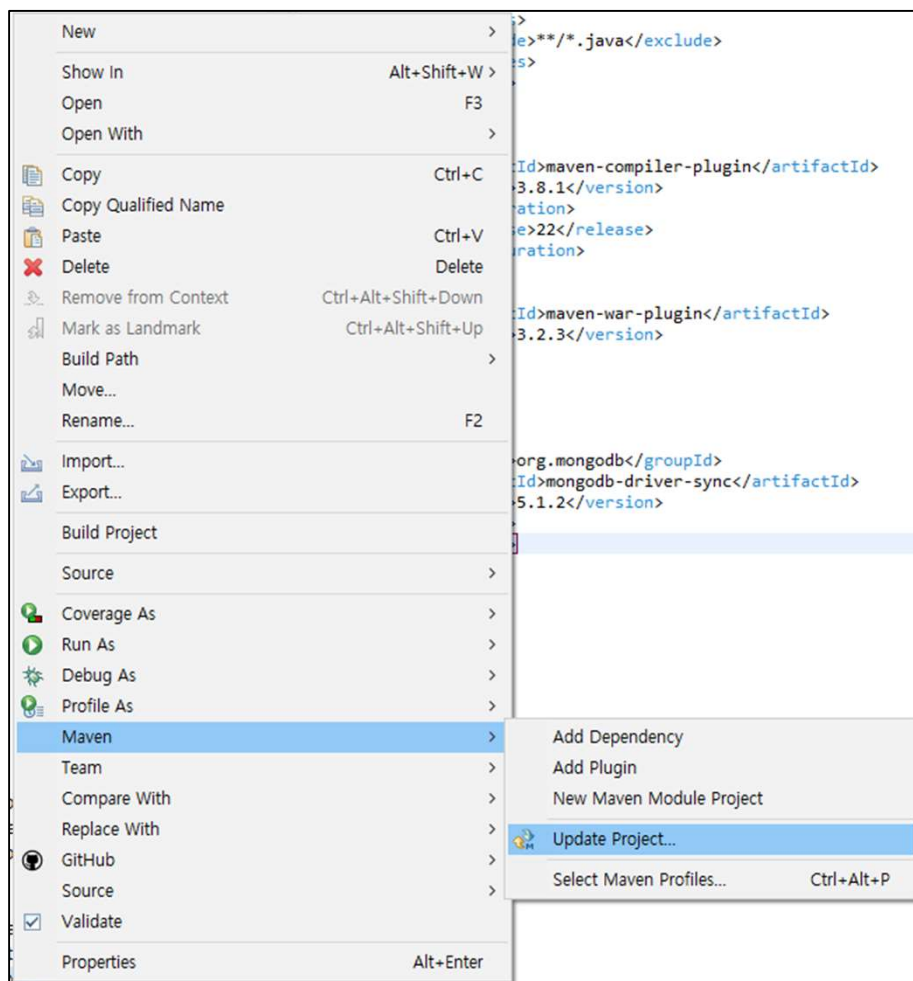
```
<dependency>
<groupId>org.mongodb</groupId>
<artifactId>mongodb-driver-sync</artifactId>
<version>4.9.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client -->
<dependency>
<groupId>org.mariadb.jdbc</groupId>
<artifactId>mariadb-java-client</artifactId>
<version>3.4.1</version>
</dependency>
</dependencies>
```

<https://mvnrepository.com/> 에서 필요한 라이브러리를 검색하여 위의 내용을 복사해오면 됨

```
    </configuration>
  </plugin>
  <plugin>
    <artifactId>maven-war-plugin</artifactId>
    <version>3.2.3</version>
  </plugin>
</plugins>
</build>
<dependencies>
  <dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>5.1.2</version>
  </dependency>
</dependencies>
</project>
```

JDBC 드라이버 설치(4)

④ 수정된 'pom.xml' 파일 적용을 위한 Maven 업데이트



실습 – MariaDB 연동

[Ex1] MariaDBConnect.java(1)

```
package common;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class MariaDBConnect {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        final String driver = "org.mariadb.jdbc.Driver";
        final String DB_IP = "localhost";
        final String DB_PORT = "3306";
        final String DB_NAME = "test";
        final String DB_URL =
            "jdbc:mariadb://" + DB_IP + ":" + DB_PORT + "/" + DB_NAME;

        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        try {
            Class.forName(driver);
            conn = DriverManager.getConnection(DB_URL, "root", "1234");
            if (conn != null) {
                System.out.println("DB 접속 성공");
            }
        } catch (ClassNotFoundException e) {
            System.out.println("드라이버 로드 실패");
            e.printStackTrace();
        } catch (SQLException e) {
            System.out.println("DB 접속 실패");
            e.printStackTrace();
        }
    }
}
```

실습 – MariaDB 연동

[Ex1] MariaDBConnect.java(2)

```
try {
    String sql = "select * from user";
    pstmt = conn.prepareStatement(sql);
    rs = pstmt.executeQuery();
    String userId = null;
    String password = null;
    String name = null;
    while (rs.next()) {
        userId = rs.getString(1);
        password = rs.getString(2);
        name = rs.getString(3);
    }
    System.out.println(userId);
    System.out.println(password);
    System.out.println(name);
} catch (SQLException e) {
    System.out.println("error: " + e);
} finally {
    try {
        if (rs != null) {
            rs.close();
        }
        if (pstmt != null) {
            pstmt.close();
        }
        if (conn != null && !conn.isClosed()) {
            conn.close();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}
```

실습 – MongoDB 연동

[Ex2] MongoDBConnect.java

```
package common;

import org.bson.Document;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;

public class MongoDBConnect {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String uri = "mongodb://localhost:27017";
        MongoClient mc = MongoClients.create(uri);
        MongoDatabase db = mc.getDatabase("local");

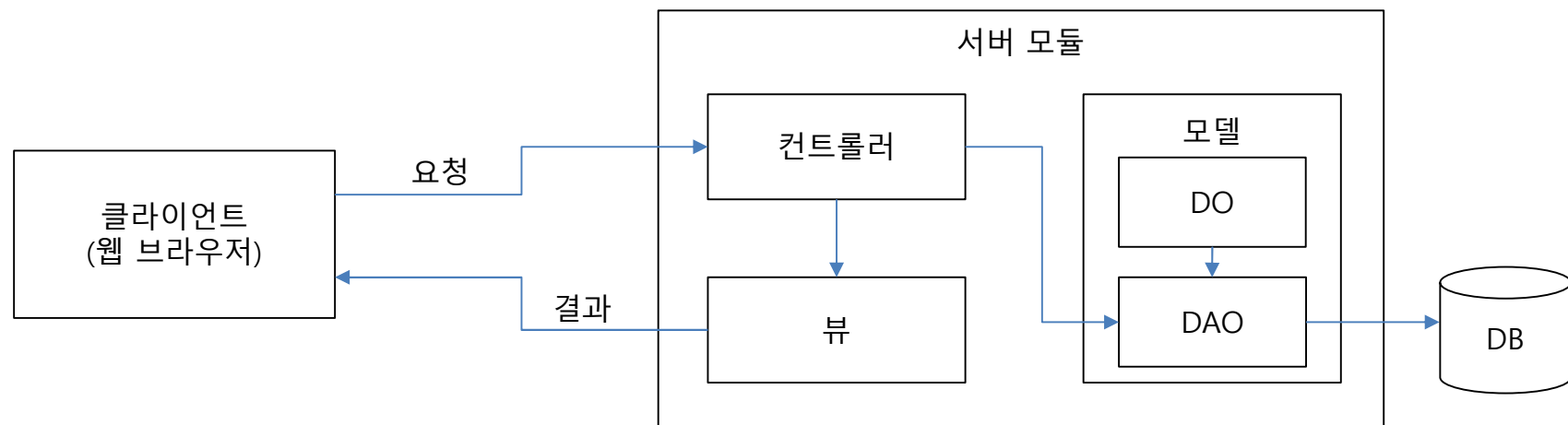
        db.createCollection("myCollection");// 컬렉션 생성
        MongoCollection<Document> col = db.getCollection("myCollection");// 컬렉션 가져오기
        Document document = new Document("name", "홍길동");// 다큐먼트 생성
        document.append("age", 30);// 필드 추가
        document.append("gender", "남자");// 필드 추가
        col.insertOne(document);// 컬렉션에 다큐먼트 추가
        document = new Document("name", "신사임당");
        document.append("age", 45);
        document.append("gender", "여자");
        col.insertOne(document);

        FindIterable<Document> docs = col.find();// 컬렉션 조회
        for(Document doc: docs) {
            System.out.println(doc.toJson());
        }
    }
}
```

5. MVC 모델

MVC(Model-View-Controller) 패턴

- GUI 기반 어플리케이션 개발에 사용되는 디자인 패턴으로서 웹 어플리케이션 개발에서 백엔드 구현의 기본으로 자리잡은 기술임
 - 모델은 데이터를 처리하는 영역으로서 데이터베이스 연동(DAO : Data Access Object) 및 데이터 구조(Data Object)를 구현하는 모듈임
 - 뷰는 화면 구성을 담당하는 영역으로서 컨트롤러로부터 전달받은 데이터를 출력하는 모듈임. 뷰 구현을 위해 JSP를 이용해 뷰 템플릿을 구성
 - 컨트롤러는 클라이언트의 요청을 처리하는 영역으로서 모델과 뷰의 제어 역할을 하는 모듈임. 클라이언트의 요청을 받아서 요청을 분석하고, 분석 결과에 따라 모델로부터 데이터를 추출하거나, 추출된 데이터를 가공하여 뷰에 전달하는 역할을 함



[Ex1-1] User.java

```
package common;

public class User {
    private String id;
    private String password;
    private String name;
    private String email;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}
```

[Ex1-2] UserDao.java

```
package common;

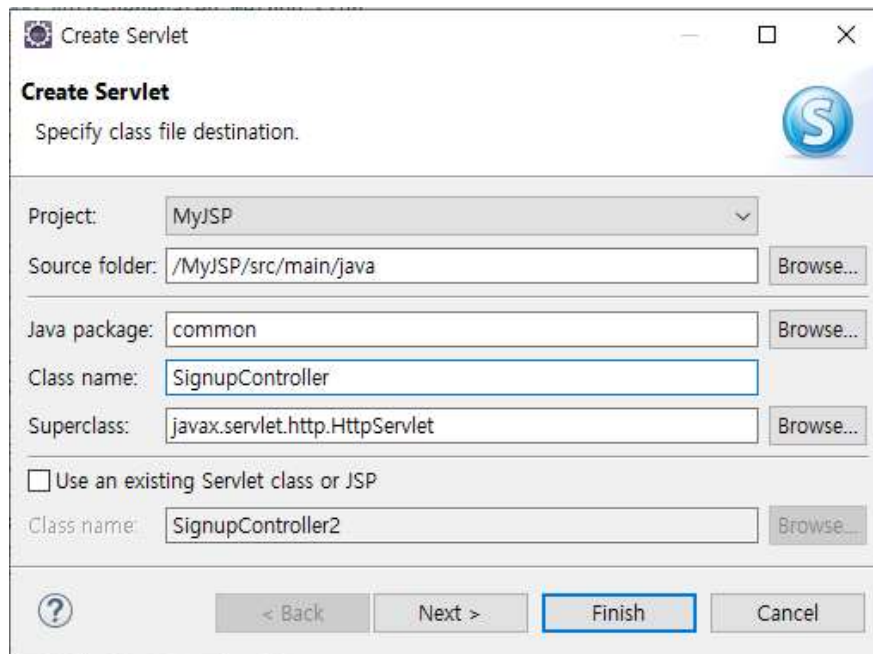
import java.sql.*;

public class UserDao {
    final String driver = "org.mariadb.jdbc.Driver";
    final String DB_IP = "localhost";
    final String DB_PORT = "3306";
    final String DB_NAME = "test";
    final String DB_URL1 = "jdbc:mariadb://" + DB_IP + ":" + DB_PORT + "/" + DB_NAME;
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "1234";

    public void insertUser(User user) throws SQLException{
        Connection connection = null;
        try {
            Class.forName(driver);
            connection = DriverManager.getConnection(DB_URL1, DB_USER, DB_PASSWORD);
            if(connection != null) {
                System.out.println("DB접속 성공");
            }
        } catch (ClassNotFoundException e) {
            System.out.println("드라이버 로드 실패");
            e.printStackTrace();
        } catch (SQLException e) {
            System.out.println("DB 접속 실패");
            e.printStackTrace();
        }
        PreparedStatement stmt = connection.prepareStatement
            ("INSERT INTO users (id, password, name, email) VALUES (?, ?, ?, ?)");
        stmt.setString(1, user.getId());
        stmt.setString(2, user.getPassword());
        stmt.setString(3, user.getName());
        stmt.setString(4, user.getEmail());
        stmt.executeUpdate();
    }
}
```

실습

- SignupController.java는 서블릿으로 생성해야 함
 - ① 'Project Explorer'의 'src/main/java' 의 패키지 명에서 마우스 버튼을 클릭 후, New > Servlet 을 선택
 - ② 'URL mappings'에서 '/SignupController'를 선택 후, 'Edit' 클릭하여 '/signup'으로 수정



Create Servlet
Specify class file destination.

Project: MyJSP

Source folder: /MyJSP/src/main/java

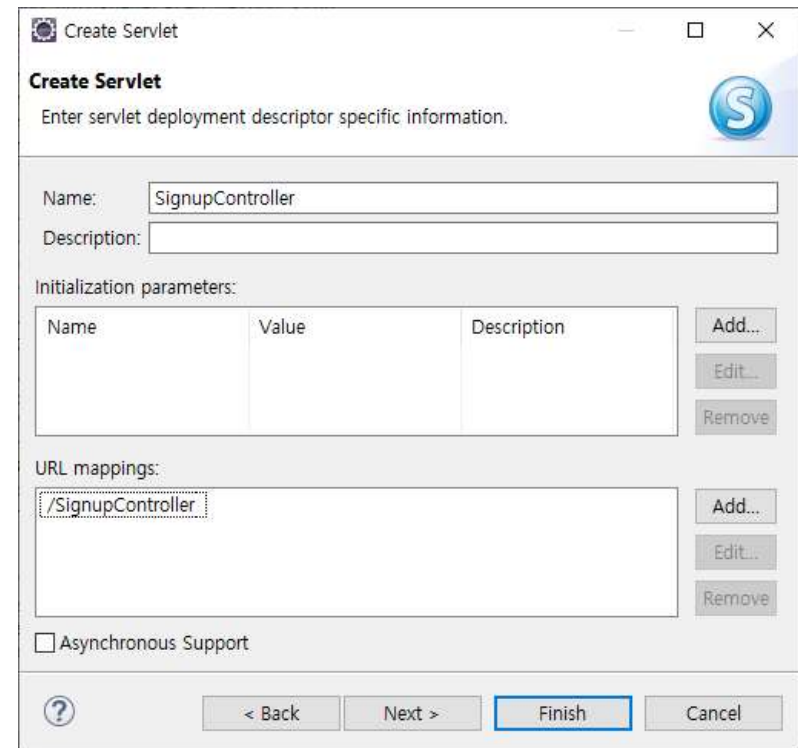
Java package: common

Class name: SignupController

Superclass: javax.servlet.http.HttpServlet

☐ Use an existing Servlet class or JSP

Class name: SignupController2



Create Servlet
Enter servlet deployment descriptor specific information.

Name: SignupController

Description:

Initialization parameters:

| Name | Value | Description |
|------|-------|-------------|
|------|-------|-------------|

URL mappings:

/SignupController

☐ Asynchronous Support

[Ex1-3] SignupController.java(1)

```
package common;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/signup")
public class SignupController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private UserDao userDao = new UserDao();

    public SignupController() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    }
}
```

[Ex1-3] SignupController.java(2)

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // TODO Auto-generated method stub
    request.setCharacterEncoding("UTF-8");
    String id = request.getParameter("id");
    String password = request.getParameter("password");
    String name = request.getParameter("name");
    String email = request.getParameter("email");
    // 모델 객체 생성 및 데이터 저장
    User user = new User();
    user.setId(id);
    user.setPassword(password);
    user.setName(name);
    user.setEmail(email);

    try {
        // 회원 가입 처리
        userDao.insertUser(user);
    } catch (Exception e) {
        // 회원 가입 실패 처리
        request.setAttribute("errorMessage", e.getMessage());
        RequestDispatcher dispatcher = request.getRequestDispatcher("/fail.jsp");
        dispatcher.forward(request, response);
        return;
    }

    // 가입 결과에 따라 적절한 뷰로 이동
    request.setAttribute("id", user.getId());
    RequestDispatcher dispatcher = request.getRequestDispatcher("/success.jsp");
    dispatcher.forward(request, response);
}
```

[Ex1-4] signup.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>회원 가입</title>
</head>
<body>
<h1>회원 가입</h1>
<form action="signup" method="post">
    <label for="id">아이디:</label>
    <input type="text" id="id" name="id" required><br>

    <label for="password">비밀번호:</label>
    <input type="password" id="password" name="password" required><br>

    <label for="name">이름:</label>
    <input type="text" id="name" name="name" required><br>

    <label for="email">이메일:</label>
    <input type="email" id="email" name="email" required><br>

    <button type="submit">가입</button>
</form>
</body>
</html>
```


실습

[Ex1-5] success.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>회원 가입 성공</title>
</head>
<body>
    <h1>회원 가입 성공!</h1>
    <p>회원님의 아이디는 ${id}입니다.</p>
    <p><a href="signup.jsp">홈페이지로 이동</a></p>
</body>
</html>
```

[Ex1-6] fail.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>회원 가입 실패</h1>
    <p>${errorMessage}</p>
    <p><a href="signup.jsp">홈페이지로 이동</a></p>
</body>
</html>
```

첨 부

GET vs. POST

- GET
 - 서버로부터 정보를 조회하기 위한 용도
 - URL에 데이터를 포함시켜 서버에 전달
 - '?' 이후의 값들은 '속성=값'의 형태로 작성되며, 서버에서 QUERY_STRING을 통해 전달됨
 - URL과 함께 데이터가 전달되므로 최대 데이터 크기가 제한적(브라우저마다 다름)
 - URL과 함께 HTTP 헤더부분에 포함되어 전달되므로 보안성 취약
- POST
 - 서버에 정보를 업로드하기 위한 용도
 - 전송할 데이터를 HTTP 메시지의 Body에 실어서 보냄
 - 데이터가 노출되지 않으므로 GET 메소드와 비교해 보안성이 좋음(Body를 암호화할 수 있으므로)
 - HTTP Body에 데이터를 포함시켜 전달하므로 데이터 크기에 제한이 없음