

JSP 강의



강사 이종구
qoxmoaos@gmail.com

목 차

- 1강. 웹프로그래밍
- 2강. 개발 환경 설정
- 3강. JSP 맛보기
- 4강. Servlet 맛보기
- 5강. Servlet 본격적으로 살펴보기-I
- 6강. Servlet 본격적으로 살펴보기-II
- 7강. Servlet 본격적으로 살펴보기-III
- 8강. Servlet 본격적으로 살펴보기-IV
- 9강. JSP 본격적으로 살펴보기-I
- 10강. JSP 본격적으로 살펴보기-II
- 11강. JSP 본격적으로 살펴보기-III
- 12강. 액션 태그
- 13강. 쿠키
- 14강. 세션
- 15강. 예외 페이지
- 16강. 자바 빈
- 17강. 데이터베이스-I
- 18강. 데이터베이스-II
- 19강. 데이터베이스-III
- 20강. 커넥션 풀
- 21강.회원 인증 프로그래밍
- 23강. EL(Expression Language)
- 24강. JSTL(JSP Standard Tag Language)
- 25강. FrontControll 패턴과 Command 패턴
- 26강. 포워딩(Forwarding)
- 27강. MVC 패턴을 이용한 게시판 만들기-I
- 28강. MVC 패턴을 이용한 게시판 만들기-II
- 29강. MVC 패턴을 이용한 게시판 만들기-III
- 30강. MVC 패턴을 이용한 게시판 만들기-IV

1강. 웹프로그래밍

- 웹프로그래밍이란?
- JAVA 웹
- 웹프로그램의 동작
- 필요한 학습



1-1. 웹프로그래밍이란?

- ◆ 웹프로그래밍이란, 웹어플리케이션을 구현하는 행위
- ◆ 웹어플리케이션이란, 웹을 기반으로 작동되는 프로그램 입니다.
- ◆ 웹이란, 1개 이상의 사이트가 연결되어 있는 인터넷 서비스의 한가지 형태를 말합니다.
- ◆ 인터넷이란, 1개 이상의 네트워크가 연결되어 있는 형태를 말합니다.

- 프로토콜(Protocol) : 네트워크상에서 약속한 통신규약(HTTP, FTP, SMTP, POP, DHCP)
- IP(Internet Protocol) : 네트워크상에서 컴퓨터를 식별할 수 있는 주소
- DNS(Domain Name Server) : IP 주소 값을 인간이 쉽게 외우도록 매핑한 문자열
- PORT : IP주소가 컴퓨터를 식별할 수 있게 해준다면, PORT번호는 해당 컴퓨터의 구동되고 있는 프로그램을 구분할 수 있는 프로그램을 구분할 수 있는 번호

http://www.gugupro.com:9002/kr/index

↓
프로토콜

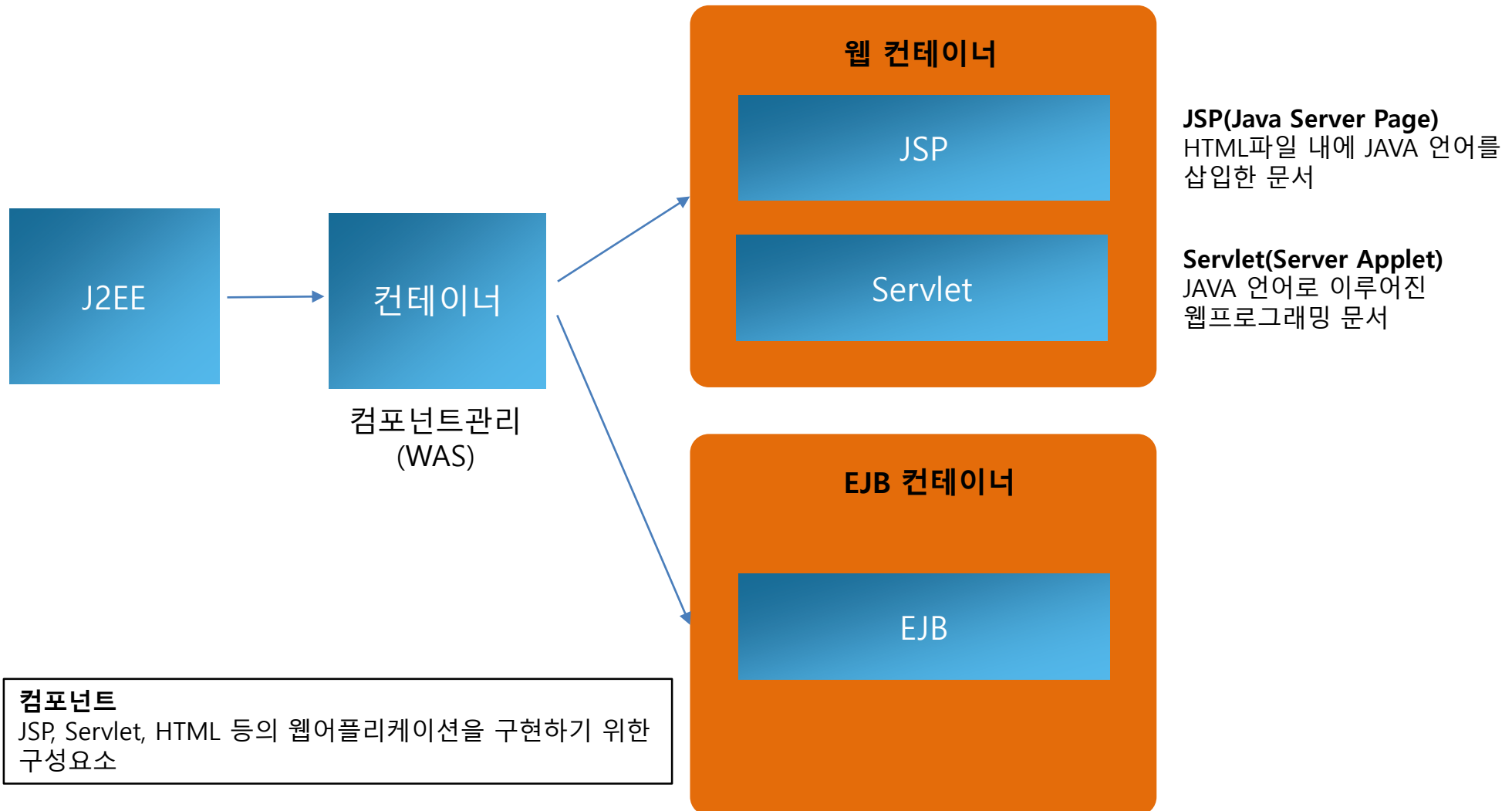
↓
컴퓨터 주소(DNS를 통해 IP주소로 변경)

↓
Port

↓
Information Path

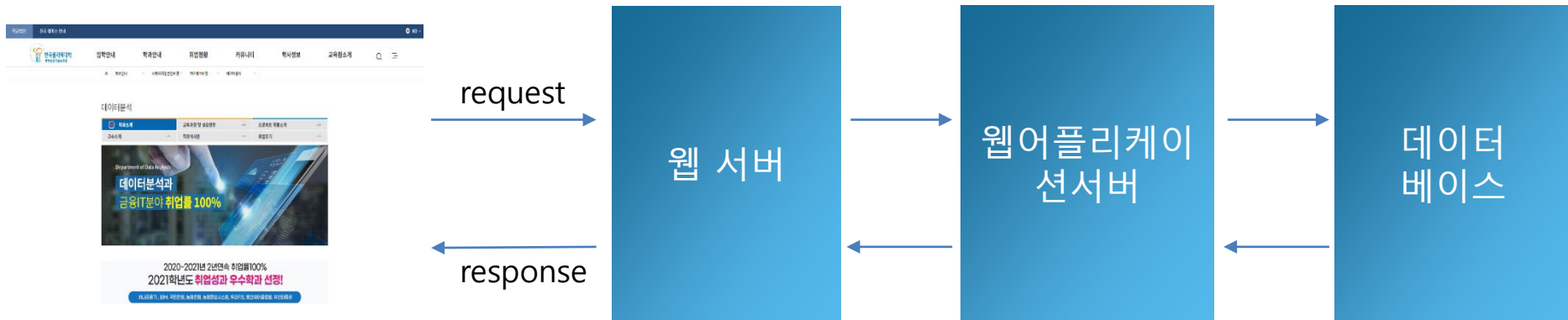
1-2. JAVA웹

- ◆ JAVA플랫폼(J2SE, J2EE)중에서 J2EE를 이용한 웹 프로그래밍입니다.



1-3. 웹프로그램의 동작

- ◆ 웹 서버 : 클라이언트의 요청에 의해 정보를 제공해 주는 서버(정적인 자원 처리: APACHE 등)
-> 별도의 구현이 필요한 로직이 있을 경우 웹 어플리케이션 서버에 요청
- ◆ 웹 어플리케이션 서버 : 별도의 기능 구현 등 로직 처리를 해주는 서버(동적인 처리: Tomcat, JBOSS, NGINX 등)
- ◆ 웹 브라우저 : 웹 서버에 정보를 요청하고, 웹 서버로부터 정보를 받는 매개체, 이때 HTTP 프로토콜을 사용



1-4. 필요한 학습

- ◆ JAVA : JAVA 웹 어플리케이션을 구현하기 위한 선행 학습 필요
- ◆ HTML : 웹 어플리케이션을 구현하기 위한 기본 언어
- ◆ JavaScript : 클라이언트 기능을 구현하기 위한 언어
- ◆ Jquery : JavaScript의 대표적인 라이브러리로, 브라우저의 스크립트 언어를 단순화 할 수 있다.
- ◆ CSS : 웹 어플리케이션의 레이아웃 및 스타일을 지정하는 언어
- ◆ DataBase 시스템 : 구조화된 정보 또는 통합관리되는 데이터의 집합을 데이터베이스라고 하고 이를 체계적으로 관리하는 시스템(Oracle, DB2, SQL Server, MySQL, PostgreSQL 등)

JavaScript/CSS →



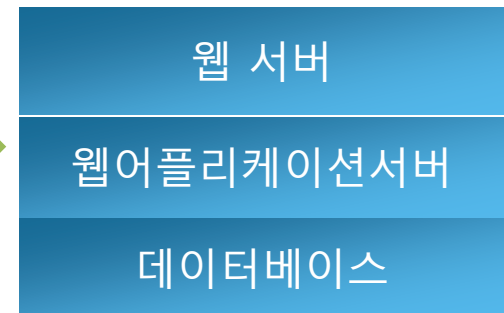
DataBase 자료 →



HTML문서 →



request
response



2강. 개발 환경 설정

- JDK설치
- Path 설정
- 이클립스 설치
- Tomcat 설치
- Tomcat 환경 설정



2-1. JDK설치

- ◆ JSP 및 Servlet은 JAVA를 기본 언어로 사용합니다. JAVA로 작성한 프로그램을 컴파일하기 위해서는 JDK(Java Development Kit)가 필요합니다.

2-2. Path 설정

- ◆ 환경변수 Path에 java_home/bin 폴더를 포함시켜 모든 폴더에서 Java가 실행 가능토록 합니다.

2-3. Eclipse 설치

- ◆ IDE 개발 툴인 Eclipse 를 다운로드 후 설치합니다.(<https://www.eclipse.org/downloads/>)

2-4. Tomcat 설치

- ◆ 웹 컨테이너 Tomcat 설치(<https://tomcat.apache.org>)

Microsoft Standard Proposal

Microsoft PowerPoint 7.0/95

3강. JSP 맛보기

- JSP문서 작성하기
- JSP 아키텍처

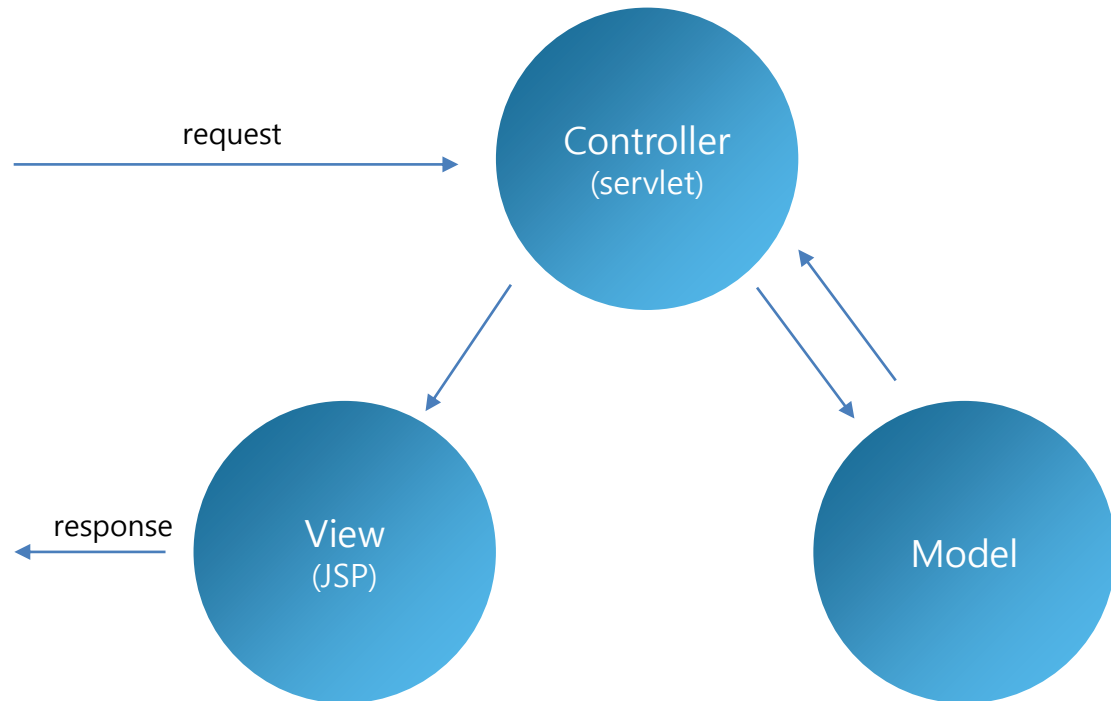
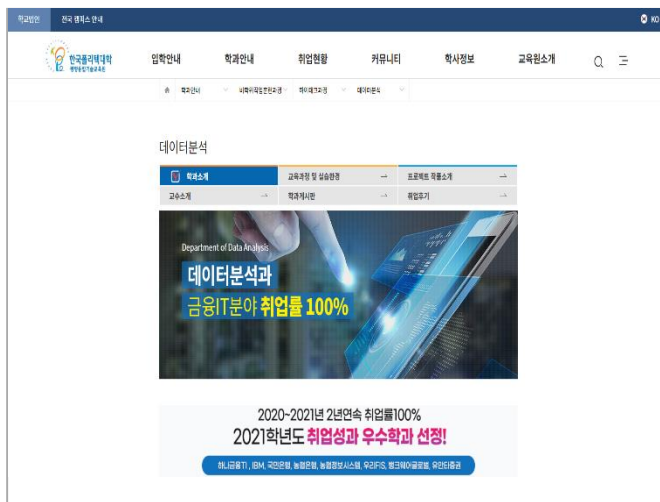
Hansu



3-1. JSP 문서 작성하기

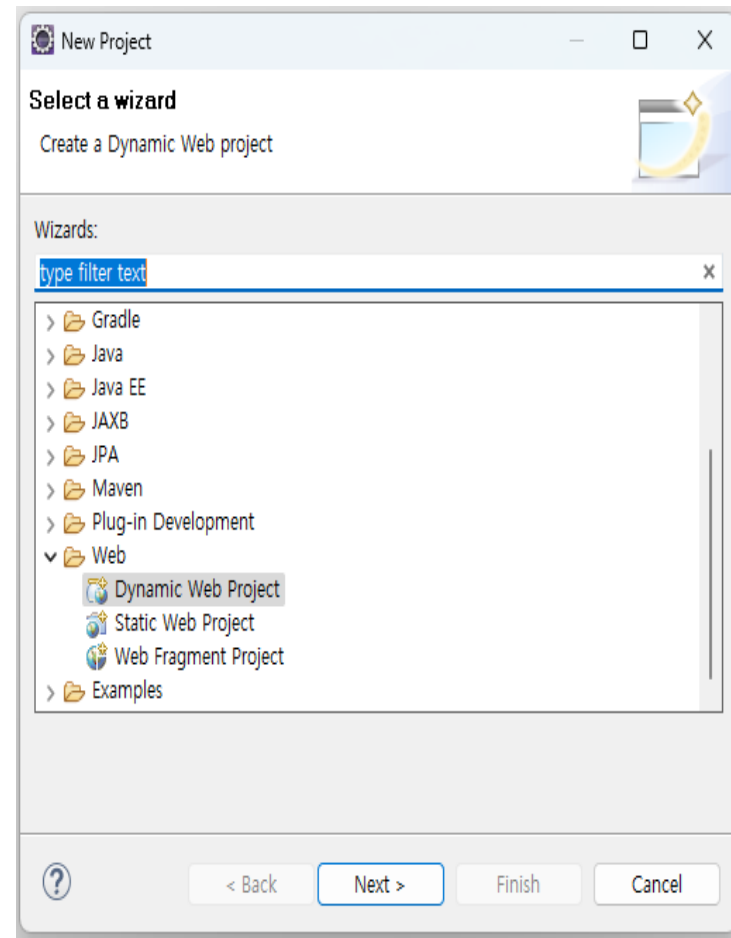
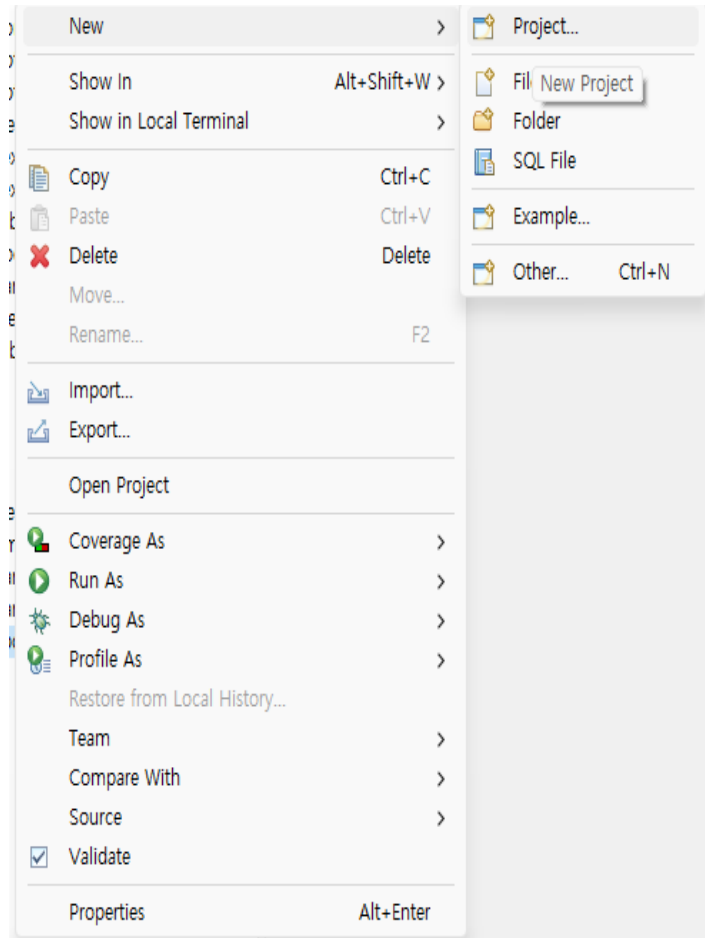
◆ JSP 특징

- 동적 웹어플리케이션 컴포넌트
- jsp 확장자
- 클라이언트의 요청에 동적으로 작동하고, 응답은 html을 이용
- Jsp는 서블릿으로 변환되어 실행
- MVC 패턴에서 View로 이용됨



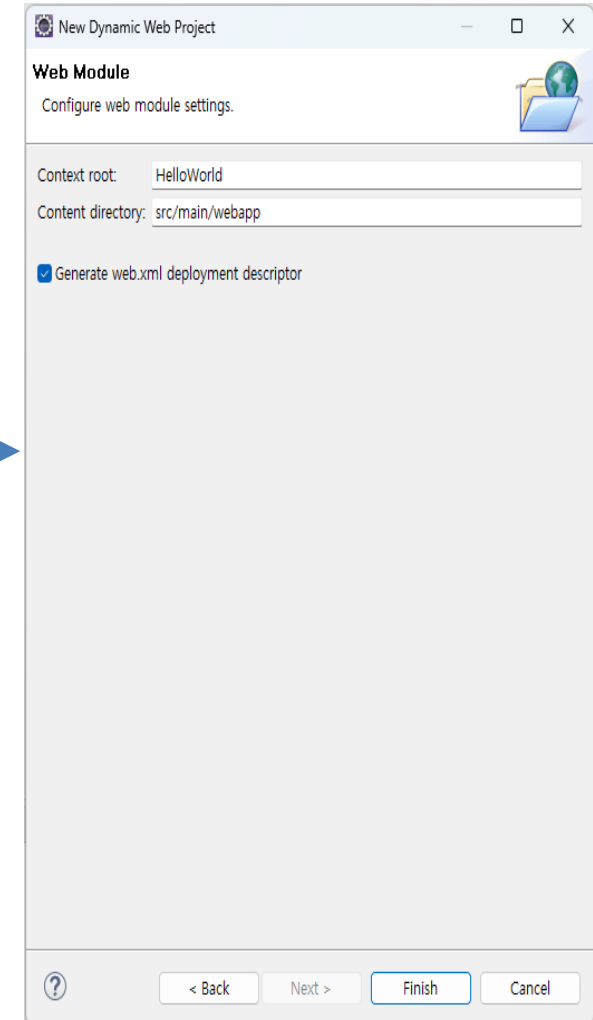
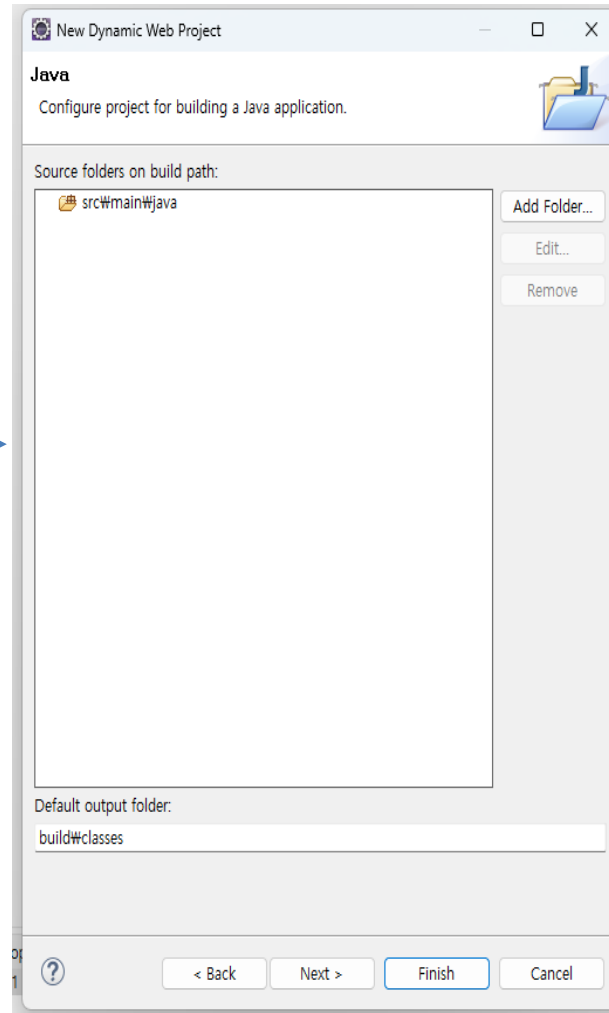
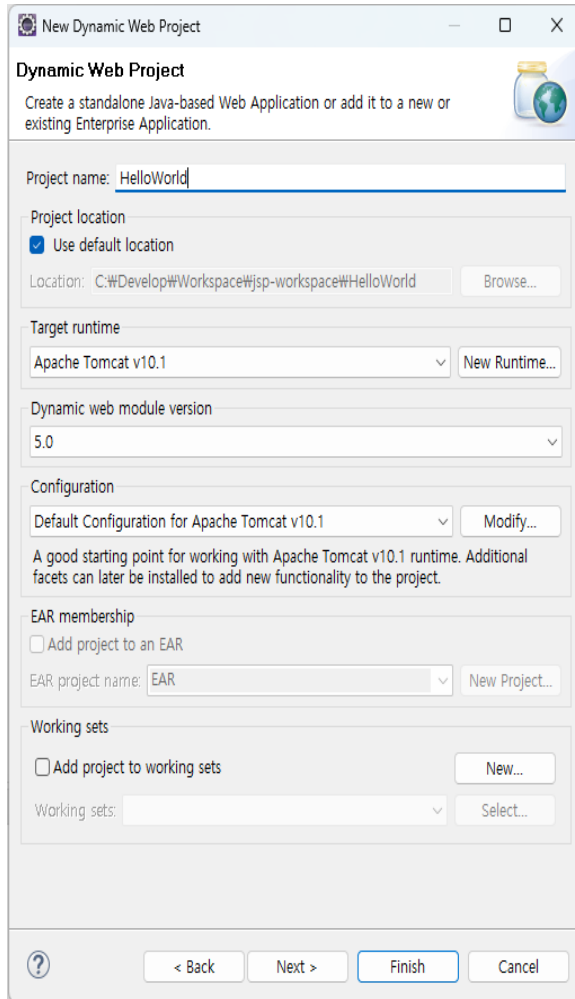
3-1. JSP 문서 작성하기

1. 프로젝트 생성



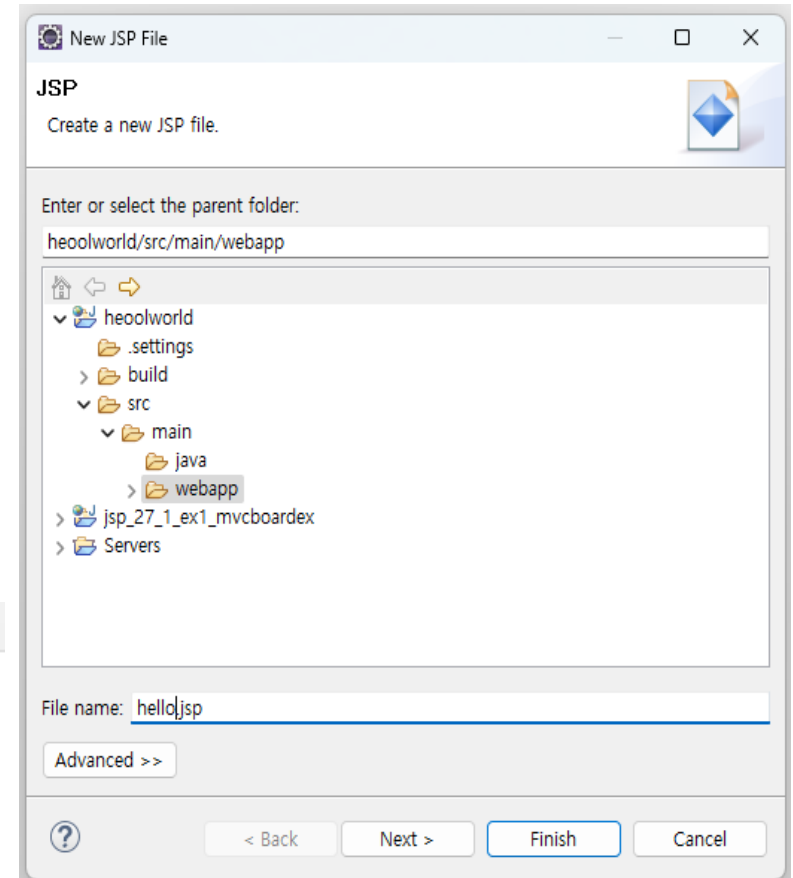
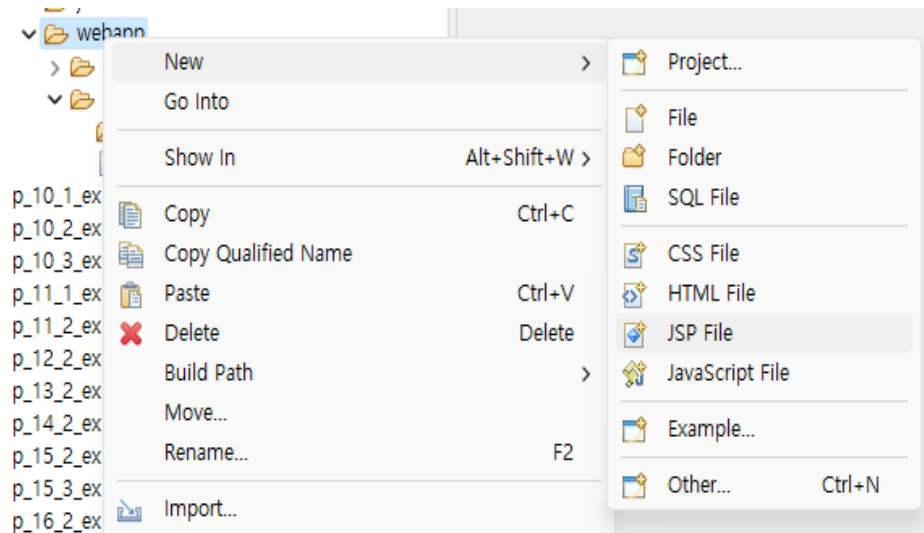
3-1. JSP 문서 작성하기

1. 프로젝트 생성



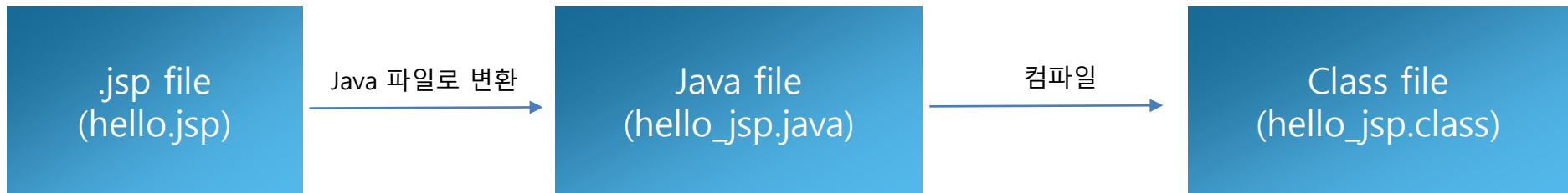
3-1. JSP 문서 작성하기

2. JSP 파일 생성



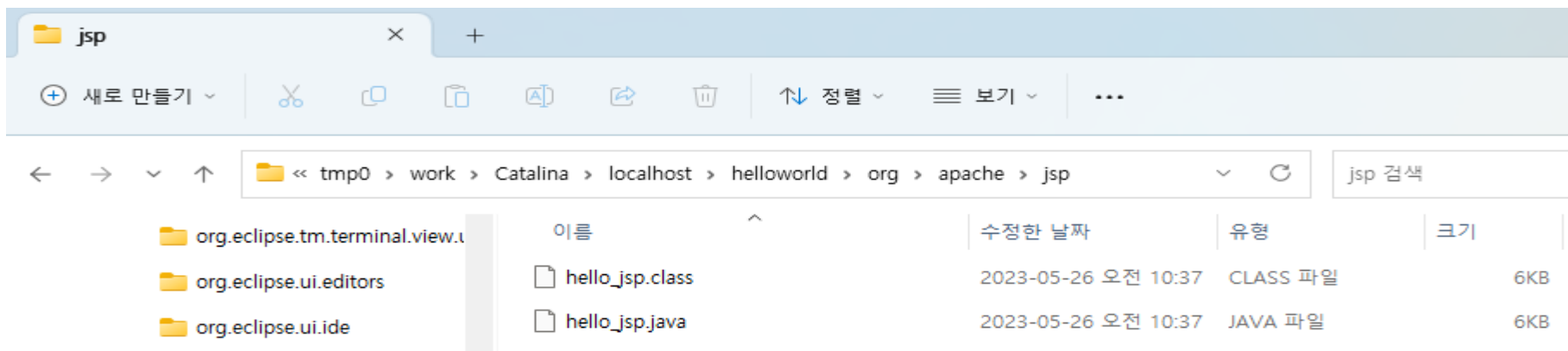
3-2. JSP 아키텍처

1. 아키텍처



2. 생성 파일 위치

C:\프로젝트_워크스페이스\metadata\plugins\org.eclipse.wst.server.core\tmp0\work\Catalina\localhost\helloworld\org\apache\jsp



4강. Servlet 맛보기

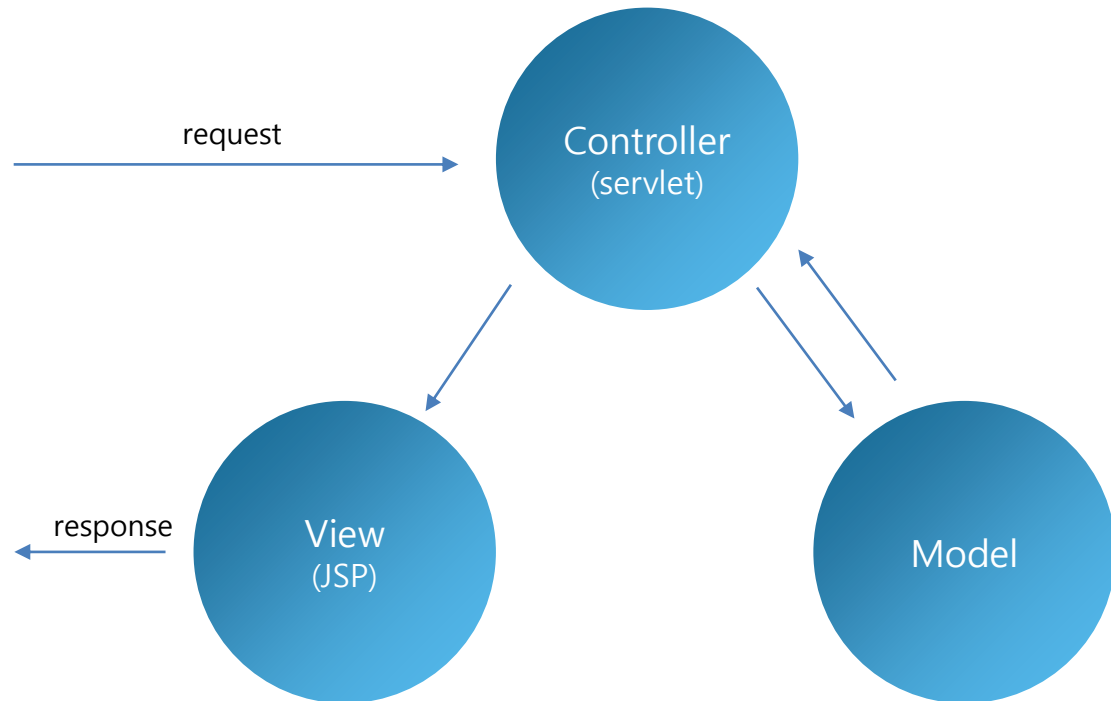
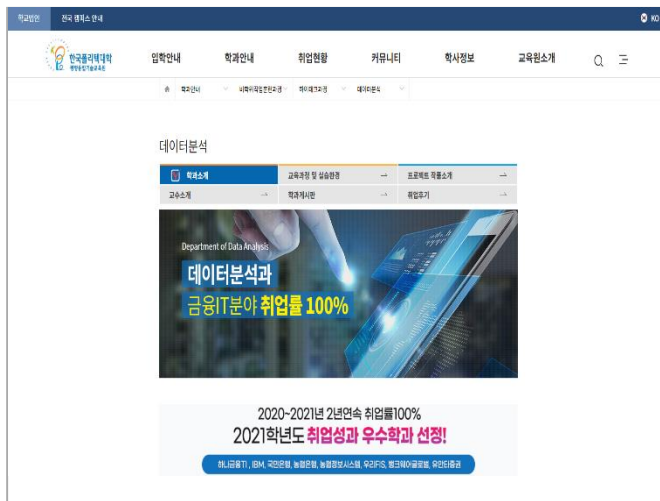
- Servlet 문서 작성하기
- web.xml에 servlet 맵핑
- Annotation을 이용한 servlet 맵핑



4-1. Servlet 문서 작성하기

◆ Servlet 특징

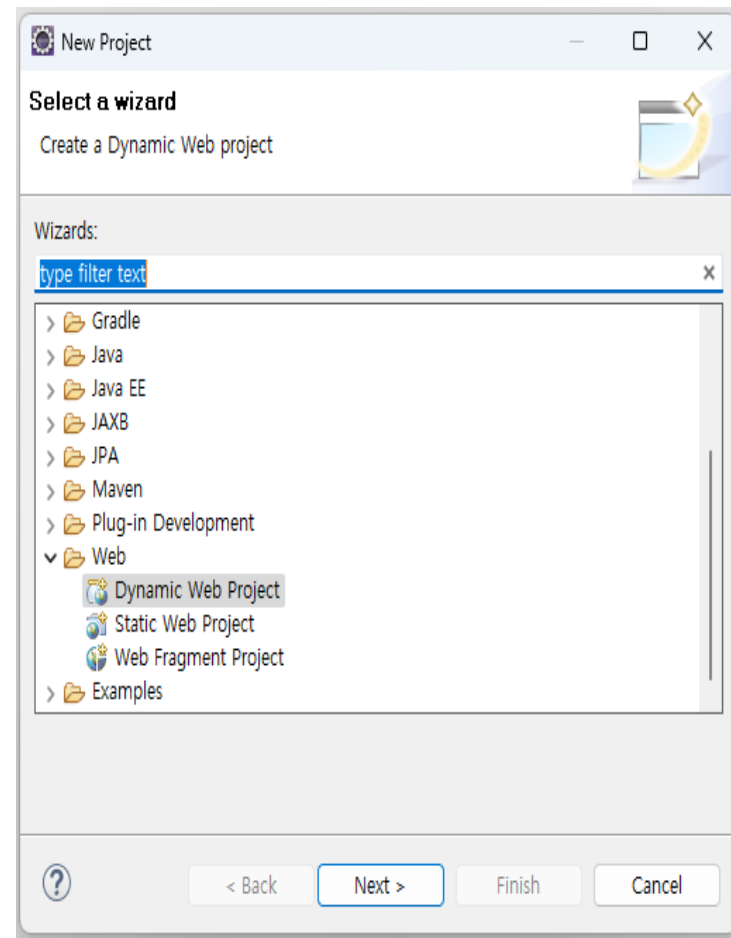
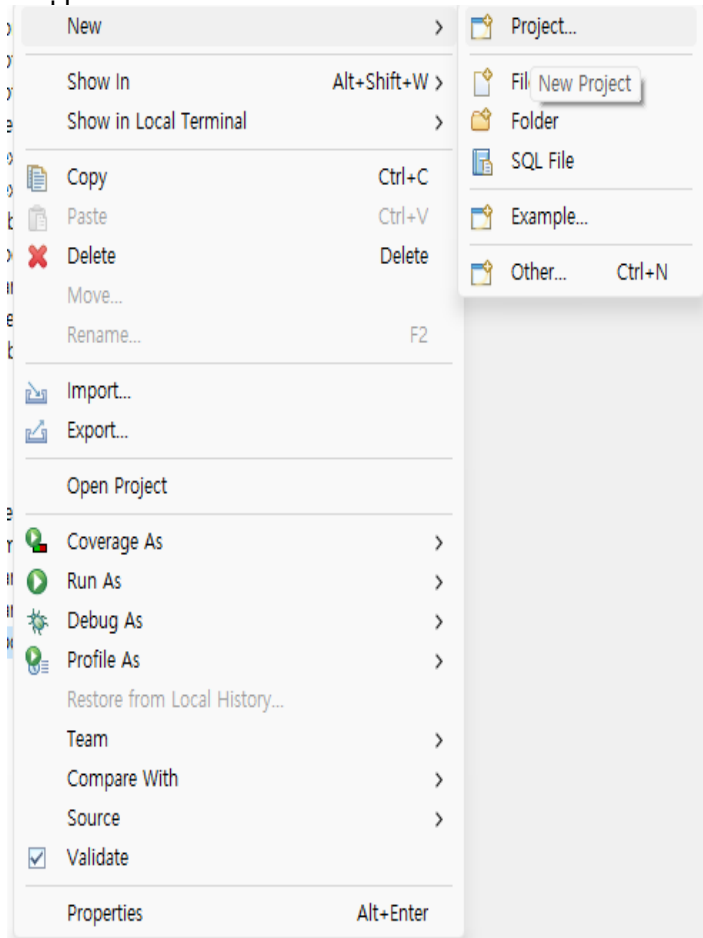
- 동적 웹어플리케이션 컴포넌트
- java 확장자
- 클라이언트의 요청에 동적으로 작동하고, 응답은 html을 이용
- Java thread 이용하여 동작
- MVC 패턴에서 Controller로 이용됨



4-1. Servlet 문서 작성하기

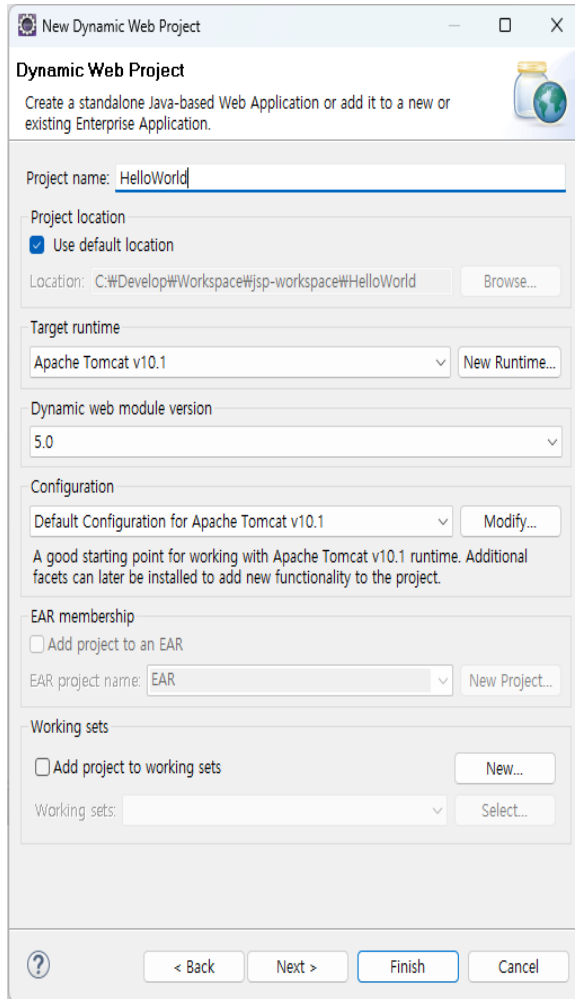
(jsp_04_1_ex1_helloworldex)

1. 프로젝트 생성



4-1. Servlet 문서 작성하기

1. 프로젝트 생성



New Dynamic Web Project

Dynamic Web Project
Create a standalone Java-based Web Application or add it to a new or existing Enterprise Application.

Project name: HelloWorld

Project location
☒ Use default location
Location: C:\Develop\workspace\jsp-workspace\HelloWorld Browse...

Target runtime
Apache Tomcat v10.1 New Runtime...

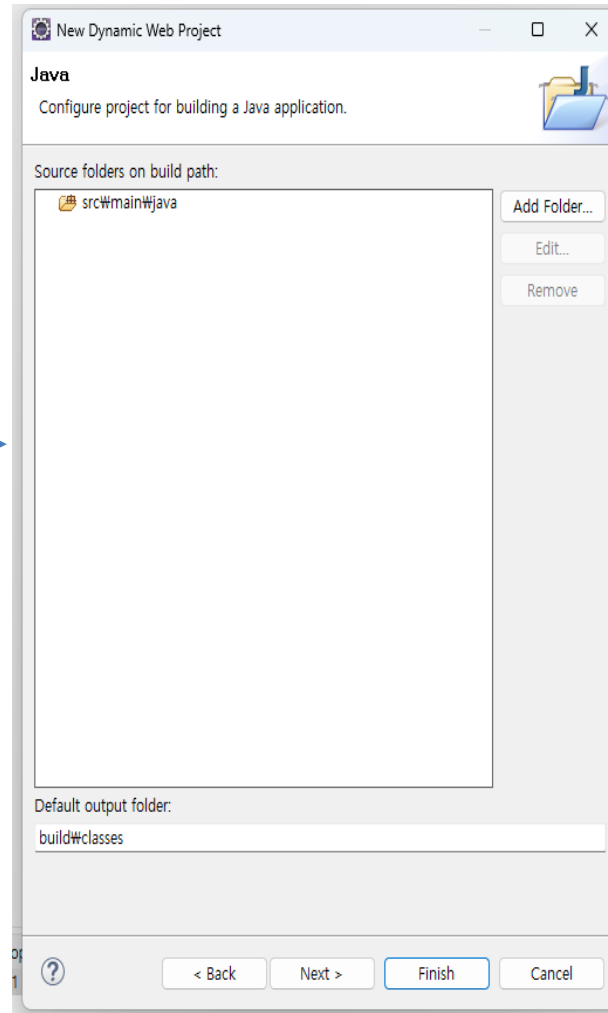
Dynamic web module version
5.0

Configuration
Default Configuration for Apache Tomcat v10.1 Modify...
A good starting point for working with Apache Tomcat v10.1 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership
☐ Add project to an EAR
EAR project name: EAR New Project...

Working sets
☐ Add project to working sets New...
Working sets: Select...

< Back Next > Finish Cancel



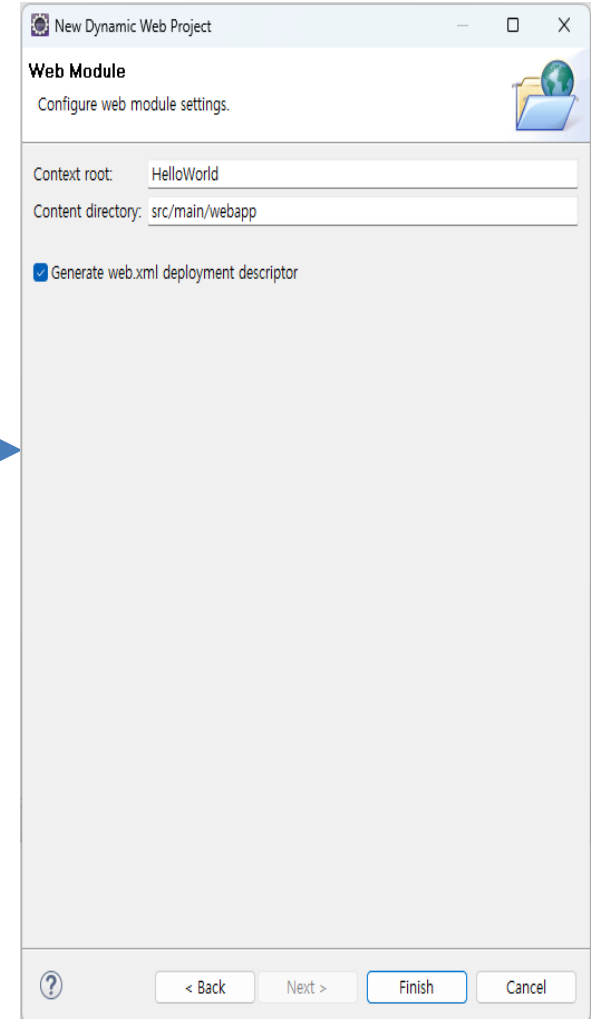
New Dynamic Web Project

Java
Configure project for building a Java application.

Source folders on build path:
src/main/java Add Folder... Edit... Remove

Default output folder:
build/classes

< Back Next > Finish Cancel



New Dynamic Web Project

Web Module
Configure web module settings.

Context root: HelloWorld

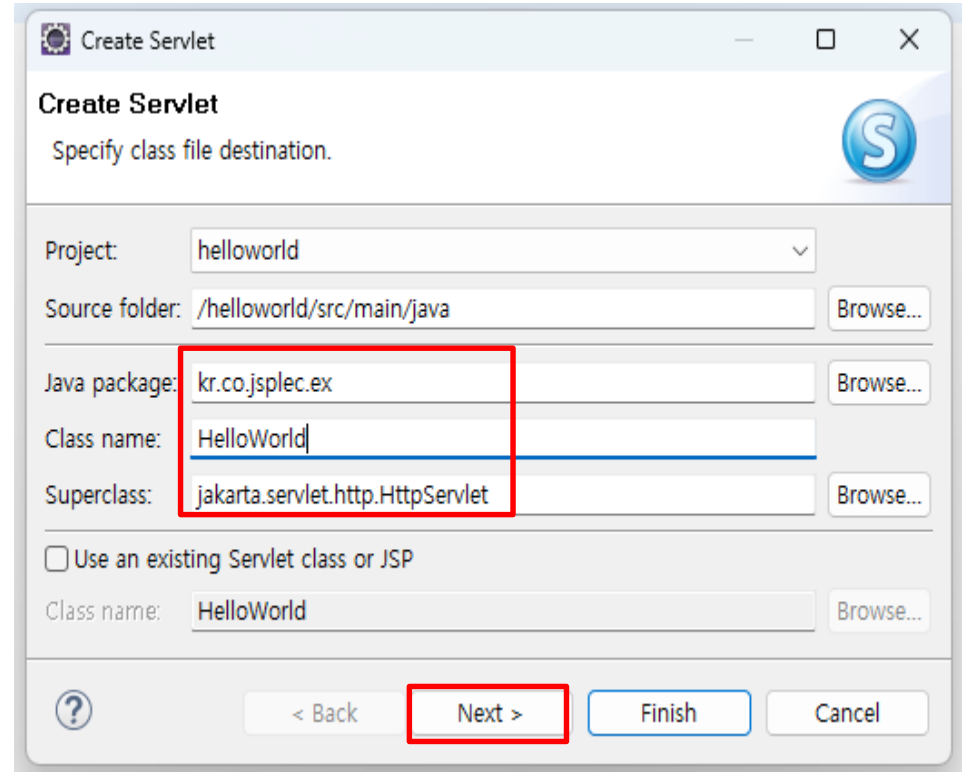
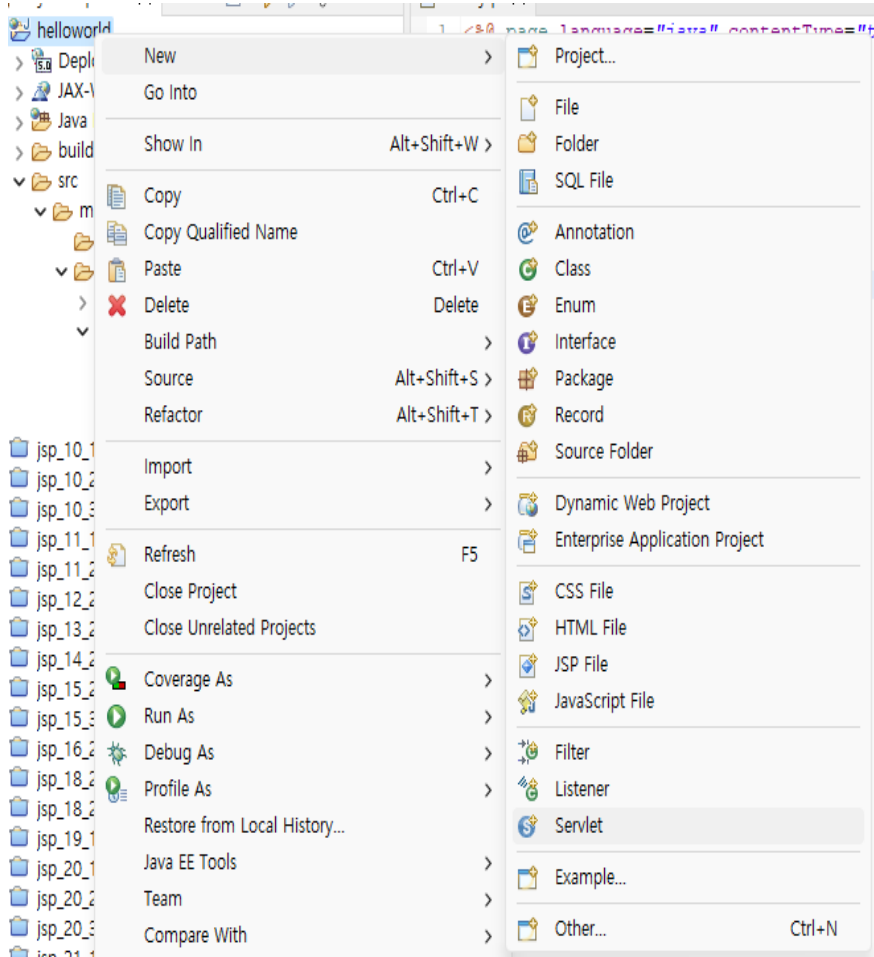
Content directory: src/main/webapp

☒ Generate web.xml deployment descriptor

< Back Next > Finish Cancel

4-1. Servlet 문서 작성하기

2. Servlet 파일 생성



4-1. Servlet 문서 작성하기

2. Servlet 파일 생성

Create Servlet
Enter servlet deployment descriptor specific information.

Name: HelloWorld

Description:

Initialization parameters:

URL Mappings

Name	Pattern
	/HWorld

URL mappings:

/HelloWorld

☐ Asynchronous Support

< Back **Next >** Finish Cancel



Create Servlet
Specify modifiers, interfaces, and method stubs to generate.

Modifiers: ☒ public ☐ abstract ☐ final

Interfaces:

Which method stubs would you like to create?

☒ Constructors from superclass
☒ Inherited abstract methods

<input type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input type="checkbox"/> service	<input checked="" type="checkbox"/> doGet
<input checked="" type="checkbox"/> doPost	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> doHead	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace

? < Back Next > **Finish** Cancel

4-1. Servlet 문서 작성하기

2. Servlet 파일 생성

Create Servlet
Enter servlet deployment descriptor specific information.

Name: HelloWorld

Description:

Initialization parameters:

URL Mappings

Name	Pattern
	/HWorld

URL mappings:

/HelloWorld

☐ Asynchronous Support

< Back **Next >** Finish Cancel



Create Servlet
Specify modifiers, interfaces, and method stubs to generate.

Modifiers: ☒ public ☐ abstract ☐ final

Interfaces:

Which method stubs would you like to create?

☒ Constructors from superclass
☒ Inherited abstract methods

<input type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input type="checkbox"/> service	<input checked="" type="checkbox"/> doGet
<input checked="" type="checkbox"/> doPost	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> doHead	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace

? < Back Next > **Finish** Cancel

4-2. web.xml 서블릿 맵핑

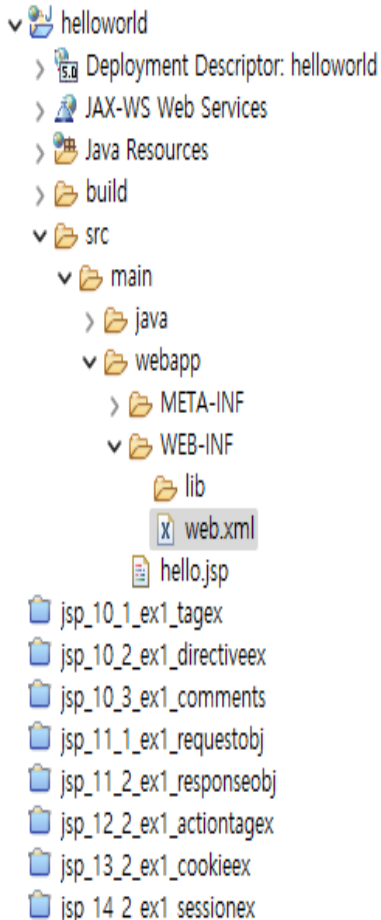
- ◆ 문자열이 너무 길고, 보안에 노출되어 있는 경로를 간편하게 축약하여 Mapping 하는 것입니다.

기존 경로 : http://localhost:8080/helloworld/servlet/com.javalec.ex.HelloWorld

URL맵핑 경로 : http://localhost:8080/helloworld/HWorld



4-2. web.xml 서블릿 맵핑



```

https://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd (xsi:schema
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" :
3   <display-name>helloworld</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.jsp</welcome-file>
7     <welcome-file>index.htm</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.jsp</welcome-file>
10    <welcome-file>default.htm</welcome-file>
11  </welcome-file-list>
12  <servlet>
13    <description></description>
14    <display-name>HelloWorld</display-name>
15    <servlet-name>HelloWorld</servlet-name>
16    <servlet-class>kr.co.jsplec.ex.HelloWorld</servlet-class>
17  </servlet>
18  <servlet-mapping>
19    <servlet-name>HelloWorld</servlet-name>
20    <url-pattern>/HWorld</url-pattern>
21  </servlet-mapping>
22 </web-app>

```

<servlet-name>

: 임의의 이름을 만들어 줍니다.

<servlet-class>

: 매핑할 클래스의 파일명을 포함하여 정확하게 입력합니다.

<url-pattern>

: servlet-class의 클래스를 매핑할 임의의 이름을 입력합니다. '/'로 시작해야 합니다.

4-3. 어노테이션을 이용한 서블릿 맵핑

```

1 package kr.co.jsplec.ex;
2
3 import jakarta.servlet.ServletException;
4
5 /**
6  * Servlet implementation class HelloWorld
7  */
8 @WebServlet("/HWorld")
9 public class HelloWorld extends HttpServlet {
10     private static final long serialVersionUID = 1L;
11
12     /**
13      * @see HttpServlet#HttpServlet()
14      */
15     public HelloWorld() {
16         super();
17         // TODO Auto-generated constructor stub
18     }
19
20     /**
21      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
22      */
23     protected void doGet(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25         // TODO Auto-generated method stub
26         response.getWriter().append("Served at: ").append(
27             System.currentTimeMillis());
28     }
29 }

```

@WebServlet("/HWorld")

: 매핑명(HWorld)을 java 소스에 직접 입력합니다.

5강. Servlet 본격적으로 살펴보기 - I

- 프로젝트 만들기
- doGet()
- doPost()
- 컨텍스트 패스(Context Path)



5-1. 프로젝트 만들기

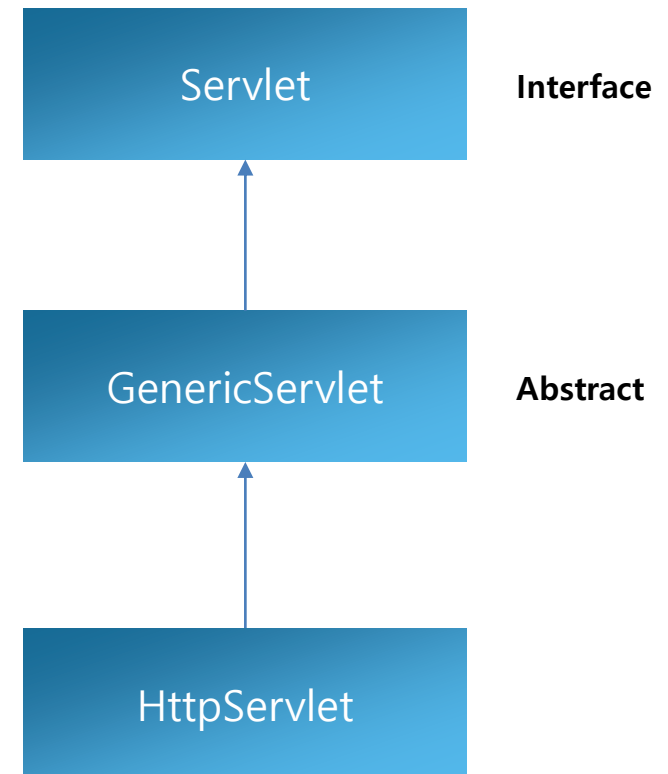
- ◆ Servlet 은 JAVA언어를 사용하여 웹 프로그램을 제작하는 것입니다. 간단한 Servlet 프로젝트를 만들어 보면서 전체적인 구조(흐름)을 살펴보도록 합니다.
(jsp_5_1_ex1_servletox)

Servlet 클래스는 HttpServlet 클래스를 상속 받음.

```
/**
 * Servlet implementation class HelloWorld
 */
@WebServlet("/HWorld")
public class HelloWorld extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public HelloWorld() {
        super();
        // TODO Auto-generated constructor stub
    }
}
```

HttpServlet 클래스 상속



5-1. 프로젝트 만들기

요청처리객체 및 응답처리객체를 Tomcat에서 받음.



5-1. 프로젝트 만들기

GET & POST 방식

```

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    response.getWriter().append("Served at: ").append(request.getContextPath());
    System.out.println("HelloWorld !!!");

    response.setContentType("text/html");
    PrintWriter pw = response.getWriter();

    pw.println("<html>");
    pw.println("<head>");
    pw.println("</head>");
    pw.println("<body>");
    pw.println("<h1>HelloWorld!!!</h1>");
    pw.println("</body>");
    pw.println("</html>");

    pw.close();
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
}

```

doGet() 호출

Form 태그 method 속성 값 = get

GET 방식

: URL 값으로 정보가 전송되어 보안에 취약

doPost() 호출

POST 방식

: header를 이용하여 정보가 전송되어 보안에 강함

Form 태그 method 속성 값 = post

5-2. doGet()

- ◆ html내 form태그의 method속성이 get일 경우 호출됩니다.
- ◆ 웹브라우저의 주소 창을 이용하여 servlet을 요청한 경우에도 호출됩니다.

doGet메소드는 매개변수로 HttpServletRequest와 HttpServletResponse를 받습니다.



doGet()

HttpServletRequest > 클라이언트의 요청처리객체

HttpServletResponse > 클라이언트의 응답처리객체

5-2. doGet()

HttpServletResponse 객체의 `setContentType()` 메서드를 호출하여 응답 방식을 결정합니다.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    response.getWriter().append("Served at: ").append(request.getContextPath());
    System.out.println("HelloWorld !!!");

    response.setContentType("text/html; charset=euc-kr");
    PrintWriter pw = response.getWriter();

    pw.println("<html>");
    pw.println("<head>");
    // ...
}
```

HttpServletResponse 객체의 `getWriter()` 메서드를 이용하여 출력스트림을 얻습니다.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    response.getWriter().append("Served at: ").append(request.getContextPath());
    System.out.println("HelloWorld !!!");

    response.setContentType("text/html; charset=euc-kr");
    PrintWriter pw = response.getWriter();

    pw.println("<html>");
    pw.println("<head>");
    // ...
}
```

5-2. doGet()

출력스트림의 `println()` 메서드를 이용하여 출력하면 웹브라우저에 출력됩니다.

```
response.setContentType("text/html; charset=euc-kr");  
PrintWriter pw = response.getWriter();
```




```
pw.println("<html>");  
pw.println("<head>");  
pw.println("</head>");  
pw.println("<body>");  
pw.println("<h1>HelloWorld!!!</h1>");  
pw.println("</body>");  
pw.println("</html>");
```

```
pw.close();
```

마지막에 출력 객체를 닫습니다.

5-3. doPost()

- ◆ Html내 form태그의 method 속성이 post일 경우 호출됩니다.
(jsp_5_1_ex1_servletox)

 HelloWorld.java
  posthtml.html
 

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="EUC-KR">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <form action="HWorld" method="post">
9   <input type="submit" value="post">
10 </form>
11 </body>
12 </html>

```

HTML

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
// TODO Auto-generated method stub
System.out.println("doPost");

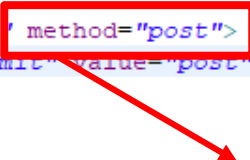
response.setContentType("text/html; charset=euc-kr");
PrintWriter pw = response.getWriter();

pw.println("<html>");
pw.println("<head>");
pw.println("</head>");
pw.println("<body>");
pw.println("<h1>POST 방식으로 호출되었습니다. HelloWorld!!!</h1>");
pw.println("</body>");
pw.println("</html>");

pw.close();
}

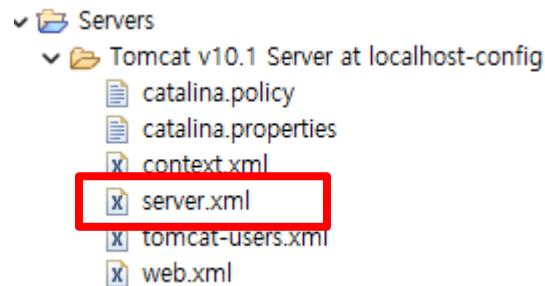
```

Servlet



5-4. 컨텍스트 패스(ContextPath)

- ◆ WAS(Web Application Server)에서 웹어플리케이션을 구분하기 위한 Path 입니다.
- ◆ Eclipse에서 프로젝트를 생성하면, 자동으로 server.xml에 추가 됩니다.



```
<Host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">

  <!-- SingleSignOn valve, share authentication between web applications
       Documentation at: /docs/config/valve.html -->
  <!--
  <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
  -->

  <!-- Access log processes all example.
       Documentation at: /docs/config/valve.html
       Note: The pattern used is equivalent to using pattern="common" -->
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs" pattern="%h %l %u %t &quot;%r&quot; %s %b" prefix="localhost_access_log" suf

  <Context docBase="jsp_05_1_ex1_servletex" path="/jsp_05_1_ex1_servletex" reloadable="true" source="org.eclipse.jst.jee.server:jsp_05_1_ex1_servletex"/></Host>
```

6강. Servlet 본격적으로 살펴보기 - II

- Servlet 작동 순서
- Servlet 라이프사이클(생명주기)
- Servlet 선처리, 후처리



6-1. Servlet 작동 순서

- ◆ 클라이언트에서 servlet요청이 들어오면 서버에서는 servlet컨테이너를 만들고, 요청이 있을 때마다 스레드가 생성됩니다.



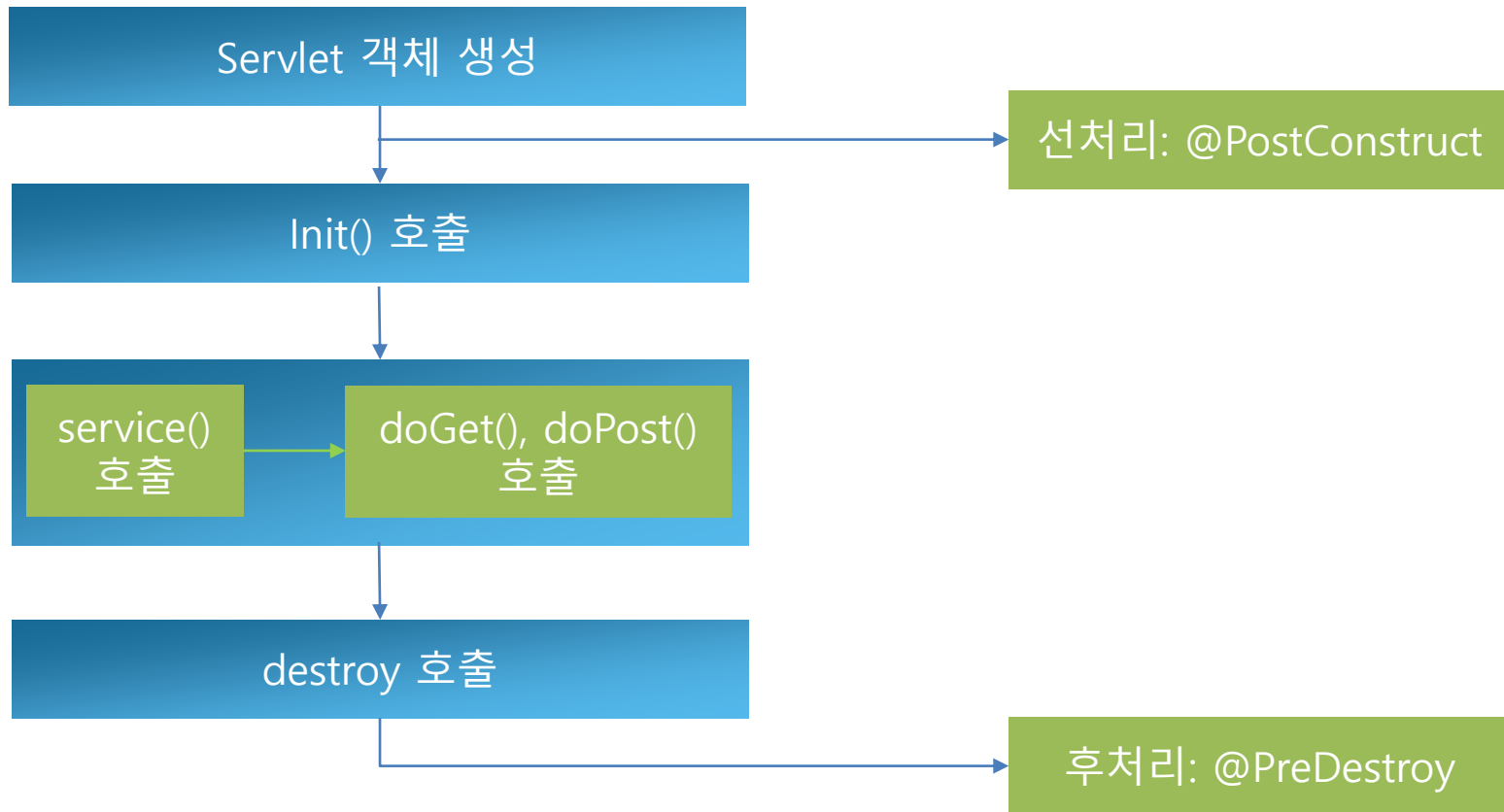
6-2. Servlet 라이프사이클(생명주기)

- ◆ Servlet의 사용도가 높은 이유는 빠른 응답 속도 때문 입니다.
- ◆ Servlet은 최초 요청 시 객체가 만들어져 메모리에 로딩되고, 이후 요청 시에는 기존의 객체를 재활용하게 됩니다. 따라서 동작 속도가 빠릅니다.
- ◆ Servlet의 라이프사이클을 살펴 봅니다.



6-3. Servlet 선처리, 후처리

- ◆ Servlet의 라이프 사이클 중 init()과 destroy() 메서드와 관련하여 선처리(init())와 후처리(destroy()후) 작업이 가능합니다.
(jsp_6_2_ex1_lifecycleex)



Servlet Standard Proposal

Servlet API Specification

7강. Servlet 본격적으로 살펴보기 - III

- HTML Form태그
- Servlet Parameter
- 한글처리



7-1. HTML form 태그

- ◆ HTML form 태그는 서버쪽으로 정보를 전달할 때 사용하는 태그입니다.
(jsp_7_1_ex1_formex)

input

- ◆ 태그의 용도를 지정합니다.

- 속성(type, name, value)
 - type : 태그종류지정(예. Text, password, submit, checkbox, radio, reset 등)
 - name : input 태그 이름
 - value : name에 해당 하는 값(예. Name = value)

type = submit

- ◆ Form 데이터 전송 시 사용

- <input type="submit" value="전송">

type = reset

- ◆ Form 데이터 초기화 시 사용

- <input type="reset" value="초기화">

7-1. HTML form 태그

- ◆ HTML form 태그는 서버쪽으로 정보를 전달할 때 사용하는 태그입니다.
(jsp_7_1_ex1_formex)

form 태그

- ◆ Input 태그들의 값을 서버로 전송하기 위한 정보를 담습니다.

<form action="FormEx" method="post">

요청하는 컴포넌트 이름
(ex. join.jsp, info.html, HWorld)

요청을 처리하는 방식
(ex. get, post)

Get : [http://IP주소:port번호/컨텍스트/path/MemberJoin?id="abcd"&name="홍길동 "](http://IP주소:port번호/컨텍스트/path/MemberJoin?id='abcd'&name='홍길동')

Post : <http://IP주소:port번호/컨텍스트/path/MemberJoin>

7-2. Servlet Parameter

- ◆ Form태그의 submit버튼을 클릭하여 데이터를 서버로 전송하면, 해당파일(Servlet)에서는 HttpServletRequest 객체를 이용하여 Parameter 값을 얻을 수 있다.
(jsp_7_1_ex1_formex)

HTML 파일

```
<form>
  <input type="submit" value="전송">
  .
  .
  .
</form>
```

Servlet 파일

HttpServletRequest객체를 사용하여 Parameter 값을 얻음

<관련 메서드>

```
: getParameter(name)
: getParameterValues(name)
: getParameterName()
```

```
<form action="FormEx" method="post">
  이름 : <input type="text" name="name" size="10"><br/>
  아이디 : <input type="text" name="id" size="10"><br/>
  비밀번호 : <input type="password" name="paswd" size="10"><br/>
  <input type="submit" value="전송">
</form>
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
// TODO Auto-generated method stub
System.out.println("doPost");
```

```
String name = request.getParameter("name");
String id = request.getParameter("id");
String paswd = request.getParameter("paswd");
System.out.println("name: " + name + "| id: " + id + "| paswd: " + paswd);
```

```
response.setContentType("text/html; charset=UTF-8");
PrintWriter pw = response.getWriter();
```

```
pw.write("<html>");
pw.write("<head>");
pw.write("</head>");
```

7-3. 한글처리

- ◆ Tomcat 서버의 기본문자처리방식은 IOS-8859-1 방식입니다. 개발자가 별도의 한글 인코딩을 하지 않으면 한글이 깨져 보이는 현상이 있습니다.
- ◆ Get방식과 Post방식에 따라 한글 처리 방식에 차이가 있습니다.
(jsp_7_1_ex1_formex)

GET방식 요청

<server.xml 수정>

```
<Connector URIEncoding="UTF-8" connectionTimeout="20000" port="8080" />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
  port="8080" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443" />
-->
```

* UTF-8 이 아닌 EUC-KR 사용해도 됩니다.

POST방식 요청

<request.setCharacterEncoding() 메서드 이용>

```
protected void doPost (HttpServletRequest request,
  // TODO Auto-generated method stub
  System.out.println("doPost");
  request.setCharacterEncoding("UTF-8");
```

8강. Servlet 본격적으로 살펴보기 - IV

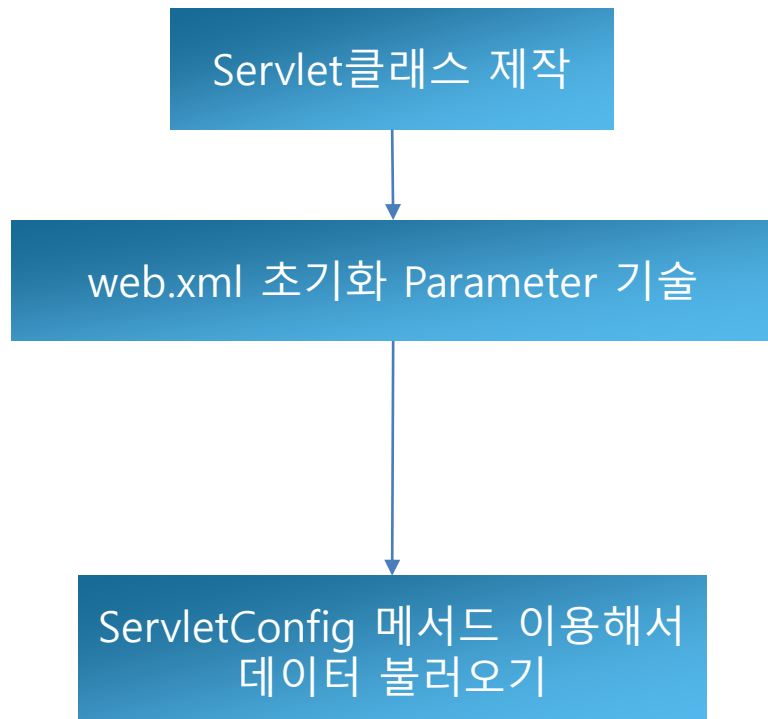
- 서블릿 초기화 파라미터 : ServletConfig
- 데이터 공유 : ServletContext
- 웹어플리케이션 감시 : ServletContextListener



8-1. 서블릿 초기화 파라미터 : ServletConfig

- ◆ 특정 Servlet이 생성될 때 초기에 필요한 데이터들이 있습니다. 예를 들어 특정 경로 및 아이디 정보 등 이러한 데이터들을 초기화 파라미터라고 하며, web.xml에 기술하고 Servlet파일에서는 ServletConfig 클래스를 이용해서 사용합니다.
- ◆ 초기화 파라미터를 web.xml 이 아닌 Servlet파일에 직접 기술하는 방법도 살펴봅니다.
(jsp_8_1_ex1_initparamex)

web.xml파일에 초기화 파라미터(Initialization Parameter) 기술



```

<!--
  LifeCycleEx
  -->
<servlet>
  <description></description>
  <display-name>LifeCycleEx</display-name>
  <servlet-name>LifeCycleEx</servlet-name>
  <servlet-class>kr.co.jsplec.ex.LifeCycleEx</servlet-class>
  <init-param>
    <param-name>dbName</param-name>
    <param-value>oracle</param-value>
  </init-param>
  <init-param>
    <param-name>dbPaswd</param-name>
    <param-value>123456</param-value>
  </init-param>
  <init-param>
    <param-name>path</param-name>
    <param-value>c:\\jsplec\\workspace</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>LifeCycleEx</servlet-name>
  <url-pattern>/LifeCycleEx</url-pattern>
</servlet-mapping>
  
```

```

String dbnm = getInitParameter("dbName");
String dbpwd = getInitParameter("dbPaswd");
String path = getInitParameter("path");
  
```

8-1. 서블릿 초기화 파라미터 : ServletConfig

Servlet파일에 초기화 파라미터(Initialization Parameter) 기술

Servlet클래스 제작

@WebInitParam에 초기화 Parameter 기술

```
@WebServlet(urlPatterns= {"/InitParamEx"}, initParams= {@WebInitParam(name="dbId", value="fghij"),
```

ServletConfig 메서드 이용해서
데이터 불러오기

```
String dbnm = getInitParameter("dbName");  
String dbpwd = getInitParameter("dbPaswd");  
String path = getInitParameter("path");
```

8-2. 데이터 공유 : ServletContext

- ◆ 여러 Servlet에서 특정 데이터를 공유해야 할 경우 context parameter를 이용, web.xml에 기술하고, Servlet 에서 공유하면서 사용할 수 있다.
(jsp_8_2_ex1_contextparamex)

web.xml파일에 context parameter 기술



8-3. 웹어플리케이션 감시 : ServletContextListener

- ◆ 웹어플리케이션의 생명주기(LifeCycle)를 감시하는 리스너(Listener : ServletContextListener)가 있습니다.
- ◆ ServletContextListener 메서드가 웹어플리케이션의 시작과 종료 시 호출됩니다.(contextInitialized(), contextDestroyed())
(jsp_8_3_ex1_contextlistenerex)

web.xml파일에 Listener Class 기술

리스너 클래스 제작

web.xml 리스너 클래스 기술

```
public class ContextListenerEx implements ServletContextListener {
```

```
@Override
```

```
public void contextDestroyed(ServletContextEvent sce) {  
    // TODO Auto-generated method stub  
    ServletContextListener.super.contextDestroyed(sce);  
  
    System.out.println("contextDestroyed");  
}
```

```
@Override
```

```
public void contextInitialized(ServletContextEvent sce) {  
    // TODO Auto-generated method stub  
    ServletContextListener.super.contextInitialized(sce);  
  
    System.out.println("contextInitialized");  
}
```

```
<listener>
```

```
<listener-class>kr.co.jsplec.ex.ContextListenerEx</listener-class>
```

```
</listener>
```


8-3. 웹어플리케이션 감시 : ServletContextListener

리스너 클래스에 기술(@WebListener) (jsp_8_3_ex1_contextlistenerex)

리스너 클래스 제작



@WebListener 추가

```
@WebListener
```

```
public class ContextListenerEx implements ServletContextListener {
```

8-1. 서블릿 초기화 파라미터 : ServletConfig - 실습

- ◆ 초기화 변수: dbId, dbPwd, dbSid
- ◆ ServletConfig를 사용하여 임의의 값을 Parameter로 넘겨 화면에 출력 하시오.(jsp_08_1_ex1_Initparamex)

8-2. 데이터 공유 : ServletContext - 실습

- ◆ 공유 변수: dbId, dbPwd, dbSid
- ◆ ServletContext를 사용하여 임의의 값을 Parameter로 넘겨 화면에 출력 하시오.(jsp_08_2_ex1_contextparamex)

8-3. 웹어플리케이션 감시 : ServletContextListener - 실습

- ◆ ServletContextListener를 상속받은 클래스를 작성하고 Servlet LifeCycle 메서드 init(), destroy(), @PostConstruct, @PreDestroy 의 실행 순서를 확인 하시오.
- ◆ (jsp_08_3_ex1_contextlistenerex -> ResultListenerTest.java)

9강. JSP 본격적으로 살펴보기 - I

- JSP 태그의 개념 이해
- JSP 동작 원리
- JSP 내부 객체



9-1. JSP 태그의 개념 이해

- ◆ Servlet은 JAVA언어를 이용하여 문서를 작성하고, 출력 객체(ex. PrintWriter)를 이용하여 HTML코드를 삽입하였습니다.
- ◆ JSP는 Servlet과 반대로 HTML코드에 JAVA언어를 삽입하여 동적 문서를 만들 수 있습니다.
- ◆ HTML코드 안에 JAVA코드를 삽입하기 위해서는 태그를 이용합니다.

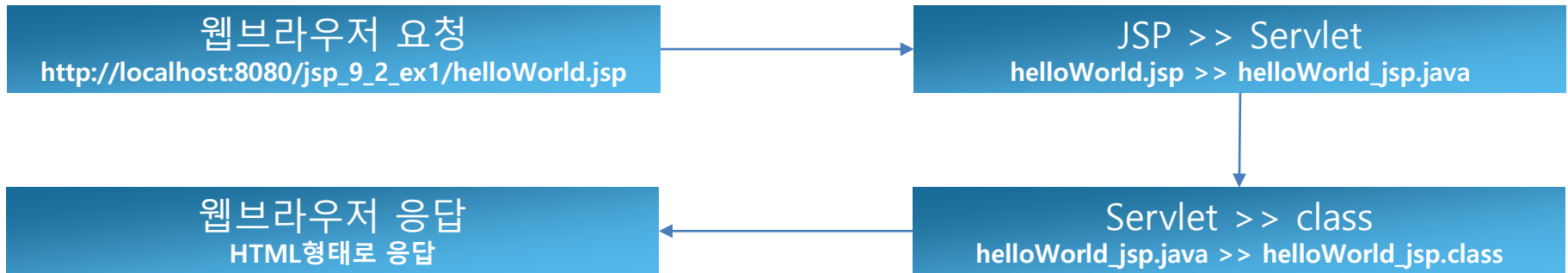
JSP태그 종류		
구분	표현방법	설명
지시자(directive)	<%@ %>	페이지 속성 <ul style="list-style-type: none"> - JSP가 JSP컨테이너에 어떠한 메시지를 보내기 위한 것 - 전역적인 값 설정(ex> 클래스 선언, 구현 메서드, 출력 내용 형식 등) - page 지시자, include 지시자, taglib 지시자
주석	<%-- --%>	주석처리
선언	<%! %>	변수, 메서드 선언
표현식	<%= %>	결과 값 출력
스크립트릿	<% %>	JAVA 코드 작성
액션태그	<jsp:action> </jsp:action>	JAVA Bean 연결

9-2. JSP 동작원리

- ◆ JSP가 요청되어 응답하기까지의 과정을 이해하면 , 개발에 많은 도움이 됩니다.

(jsp_9_2_ex1)

- ◆ 클라이언트가 웹브라우저로 helloWorld.jsp를 요청하게 되면 JSP컨테이너가 JSP파일을 Servlet파일(.java)로 변환합니다.
- ◆ 이후 Servlet파일(.java)은 컴파일 된 후 클래스 파일(.class)로 변환되고, 요청한 클라이언트한테 html파일 형태로 응답 됩니다.



9-3. JSP 내부 객체

- ◆ 개발자가 객체를 생성하지 않고 바로 사용할 수 있는 객체가 내부 객체입니다.
- ◆ JSP에서 제공되는 내부 객체는 JSP컨테이너에 의해 Servlet으로 변화될 때 자동으로 객체가 생성됩니다.

내부 객체 종류

구분	내부객체
입출력 객체	request, response, out
서블릿 객체	page, config
세션 객체	session
예외 객체	exception

10강. JSP 본격적으로 살펴보기 - II

- 스크립트릿, 선언, 표현식
- 지시자
- 주석



10-1. 스크립트릿, 선언, 표현식

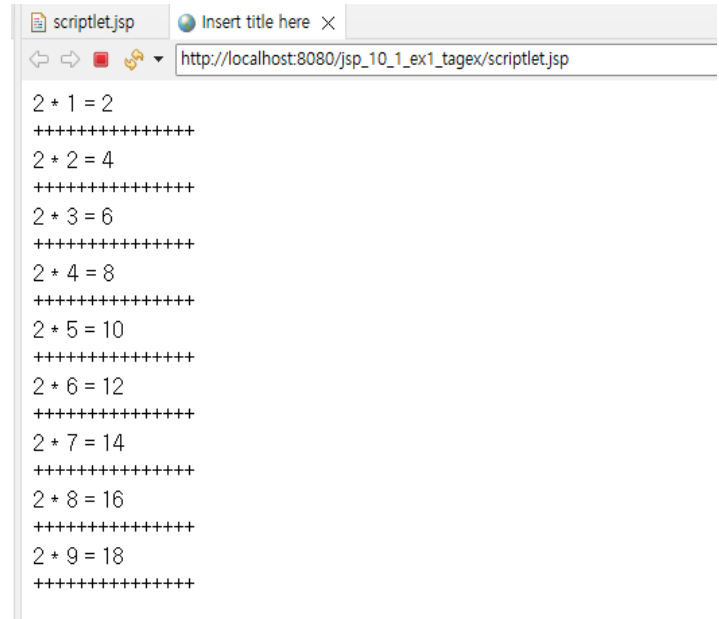
- ◆ JSP는 태그를 이용하여 문법을 기술합니다. 이는 JSP문서 안에 JAVA언어를 넣기 위한 방식들 입니다.
(jsp_10_1_ex1_tagex)

스크립트릿(scriptlet) : <% java코드 기술 %>

- ◆ JSP페이지에서 JAVA언어를 사용하기 위한 요소 중 가장 많이 사용되는 요소 입니다.
우리가 알고 있는 거의 모든 JAVA코드를 사용할 수 있습니다.

```
<body>
<%
    int i = 0;
    while(true) {
        i++;
        out.println("2 * " + i + " = " + (2*i) + "<br />");
    }
    if(i >= 9) break;
%>
```

```
<body>
2 * 1 = 2<br />
+++++++<br />
2 * 2 = 4<br />
+++++++<br />
2 * 3 = 6<br />
+++++++<br />
2 * 4 = 8<br />
+++++++<br />
2 * 5 = 10<br />
+++++++<br />
2 * 6 = 12<br />
+++++++<br />
2 * 7 = 14<br />
```



- 스크립트릿(scriptlet) : JSP페이지에서 Java코드를 작성하고 실행할 수 있도록 돕는 코드 블록

10-1. 스크립트릿, 선언, 표현식

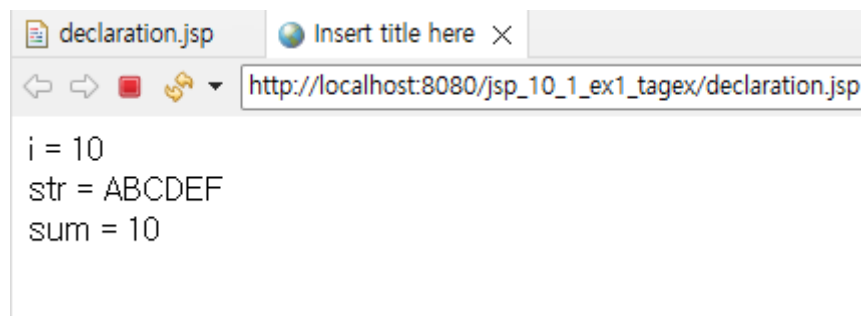
선언(declaration) : <%! java코드 기술 %>

- ◆ JSP페이지 내에서 사용되는 변수 또는 Method를 선언할 때 사용합니다.
여기서 선언된 변수 및 Method는 전역의 의미로 사용됩니다.

```
<body>
<%!
    int i = 10;
    String str = "ABCDEF";
%>

<%!
    public int sum(int x, int y){
        return x + y;
    }
%>

<%
    out.println("i = " + i + "<br />");
    out.println("str = " + str + "<br />");
    out.println("sum = " + sum(1, 9) + "<br />");
%>
</body>
```



10-1. 스크립트릿, 선언, 표현식

표현식(expression) : `<%= java코드 기술 %>`

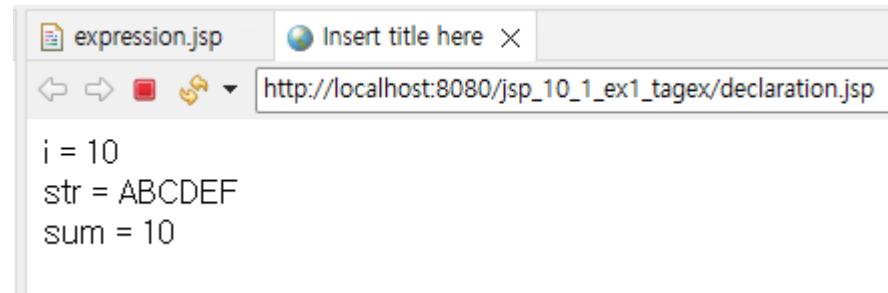
- ◆ JSP페이지 내에서 사용되는 변수의 값 또는 Method 호출 결과 값을 출력하기 위해 사용됩니다. 결과 값은 String 타입이며, ','를 사용할 수 없습니다.

```
<body>
<%!
    int i = 10;
    String str = "ABCD";

    private int sum(int x, int y){
        return x + y;
    }
%>

<%= i %><br />
<%= str %><br />
<%= sum(1, 10) %><br />

</body>
```



10-2. 지시자

- ◆ JSP페이지의 전체적인 속성을 지정할 때 사용합니다.
- ◆ page, include, taglib가 있으며, `<%@ 속성 %>` 형태로 사용합니다.

지시자	설명
page	해당 페이지의 전체적인 속성 지정
include	별도의 페이지를 현재 페이지에 삽입
taglib	태그라이브러리의 태그 사용

page 지시자

- ◆ 페이지의 속성을 지정할 때 사용합니다. 주로 사용되는 언어 지정 및 import문을 많이 사용합니다.
(jsp_10_2_ex1_directiveex)

```

page.jsp ×
1  <%@page import="java.util.Arrays"%>
2  <%@ page language="java" contentType="text/html; charset=EUC-KR"
3      pageEncoding="EUC-KR"%>
.
.
.
<%
    int[] iArr = {10, 20, 30};
    out.println(Arrays.toString(iArr));
%>

```

10-2. 지시자

include 지시자

- ◆ 현재 페이지 내에 다른 페이지를 삽입할 때 사용됩니다. File 속성을 이용합니다.
(jsp_10_2_ex1_directiveex)

```
<h1>include.jsp 페이지 입니다.</h1><br />
<%@ include file = "include01.jsp" %><br />
<h1>다시 include.jsp 페이지 입니다.</h1>
```

http://localhost:8080/jsp_10_2_ex1_directiveex/include.jsp

include.jsp 페이지 입니다.

include01.jsp 페이지 입니다.

다시 include.jsp 페이지 입니다.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
</head>
<body>
    <h1>include.jsp 페이지 입니다.</h1><br />
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
</head>
<body>
    <h1>include01.jsp 페이지 입니다.</h1>
</body>
</html><br />
```

```
    <h1>다시 include.jsp 페이지 입니다.</h1>
</body>
</html>
```

10-2. 지시자

taglib 지시자

- ◆ 사용자가 만든 tag들을 태그라이브러리라고 합니다. 그리고 이러한 태그라이브러리를 사용하기 위해 taglib 지시자를 사용합니다.
- ◆ uri 및 prefix 속성이 있으며, uri는 태그라이브러리의 위치 값을 가지며, prefix는 태그를 가리키는 이름 값을 가집니다.
- ◆ taglib 지시자에 대한 학습은 추후 JSTL학습 때 살펴보기로 합니다.

10-3. 주석

- ◆ 실제 프로그램에는 영향이 없고, 프로그램 설명들의 목적으로 사용되는 태그입니다.
- ◆ HTML 및 JSP 주석이 별도로 존재합니다.
(jsp_10_3_ex1_comments)

HTML 주석 : `<!-- comments -->`

- ◆ 테스트 용도 및 프로그램 설명 용도 등으로 사용

```
<!-- 여기는 html 주석입니다. -->
여기는 html 주석이 아닙니다.
```

```
<!-- 여기는 html 주석입니다. -->
여기는 html 주석이 아닙니다.
```

JSP주석 : `<%-- comments --%>`

- ◆ HTML 주석과 마찬가지로 테스트 용도 및 프로그램 설명 용도 등으로 사용
- ◆ JAVA 언어의 주석도 사용됩니다.(`//` , `/* */`)

```
<%-- 여기는 jsp 주석입니다. --%>
여기는 jsp 주석이 아닙니다.
```

```
여기는 jsp 주석이 아닙니다.
```

- 브라우저 소스 보기에서는 jsp 는 was에서 컴파일되어 html 결과 파일로 전달되어지기 때문에 소스 보기에는 나타나지 않습니다.

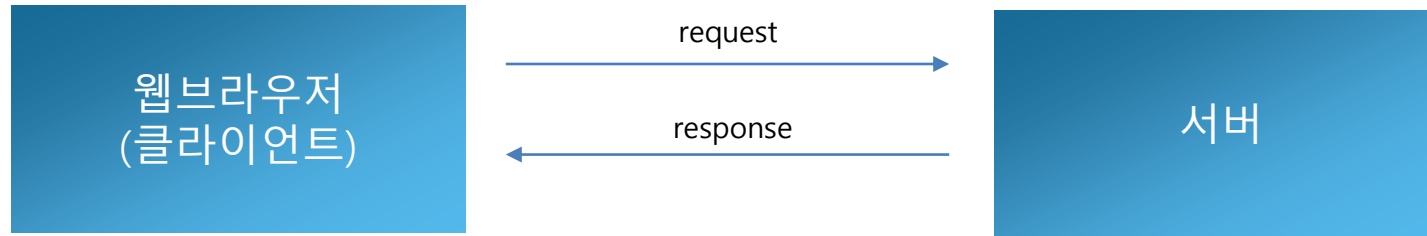
11강. JSP 본격적으로 살펴보기 - III

- request 객체의 이해
- response 객체의 이해



11-1. request 객체의 이해

- ◆ 웹 브라우저를 통해 서버에 어떤 정보를 요청하는 것을 request라고 합니다. 이러한 요청 정보는 request 객체가 관리합니다.



Request객체 관련 HTTP관련 Methods(jsp_11_1_ex1_requestobj)

- ◆ JSP페이지에서 JAVA언어를 사용하기 위한 요소 중 가장 많이 사용되는 요소 입니다. 우리가 알고 있는 거의 모든 JAVA코드를 사용할 수 있습니다.

Method	기능
getContextPath()	웹어플리케이션의 Context Path를 얻습니다.
getMethod()	get 방식과 post 방식을 구분할 수 있습니다.
getSession()	세션 객체를 얻습니다.
getProtocol()	해당 프로토콜을 얻습니다.
getRequestURL()	요청 URL를 얻습니다.
getRequestURI()	요청 URI를 얻습니다.
getQueryString()	쿼리스트링을 얻습니다.

```

<code>
out.println("서버: " + request.getServerName() + "<br />");
out.println("포트번호: " + request.getServerPort() + "<br />");
out.println("요청방식: " + request.getMethod() + "<br />");
out.println("프로토콜: " + request.getProtocol() + "<br />");
out.println("URL: " + request.getRequestURL() + "<br />");
out.println("URI: " + request.getRequestURI() + "<br />");
</code>
  
```


11-1. request 객체의 이해

Request 객체 Parameter 관련 메서드(jsp_11_1_ex1_requestparam)

- ◆ 앞에서 살펴본 요청 관련 메서드 보다 실제 많이 쓰이는 메서드는 parameter와 관련된 메서드들입니다. Jsp페이지를 제작하는 목적이 데이터 값을 전송하기 위해서 이므로, parameter 관련 메서드는 중요합니다.

getParameter(String name): name에 해당하는 parameter 값을 구함.

getParameterNames(): 모든 parameter 명칭을 구함.

getParameterValues(String name): name에 해당하는 parameter 값들을 구함.

```
<body>
<form action="paramRequest.jsp" method="post">
    이름<input type="text" name="name" size="10">
    아이디<input type="text" name="id" size="10">
    비밀번호<input type="password" name="paswd" size="10">
    취미<input type="checkbox" name="hobby" value="cook">
        <input type="checkbox" name="hobby" value="running">
        <input type="checkbox" name="hobby" value="sleeping">
    전공<input type="radio" name="major" value="korean">
        <input type="radio" name="major" value="english">
        <input type="radio" name="major" value="math">
    프로토콜<select name="protocol">
        <option value="http">http</option>
        <option value="ftp" selected="selected">ftp</option>
        <option value="smtp">smtp</option>
        <option value="https">https</option>
    </select>
    <input type="submit" name="submit" value="전송">
    <input type="reset" name="reset" value="초기화">
</form>
</body>
```

```
<%!
String name, id, paswd, major, protocol;
String[] hobby;
%>
<%
request.setCharacterEncoding("EUC_KR");

name = request.getParameter("name");
id = request.getParameter("id");
paswd = request.getParameter("paswd");
major = request.getParameter("major");
protocol = request.getParameter("protocol");

hobby = request.getParameterValues("hobby");
%>

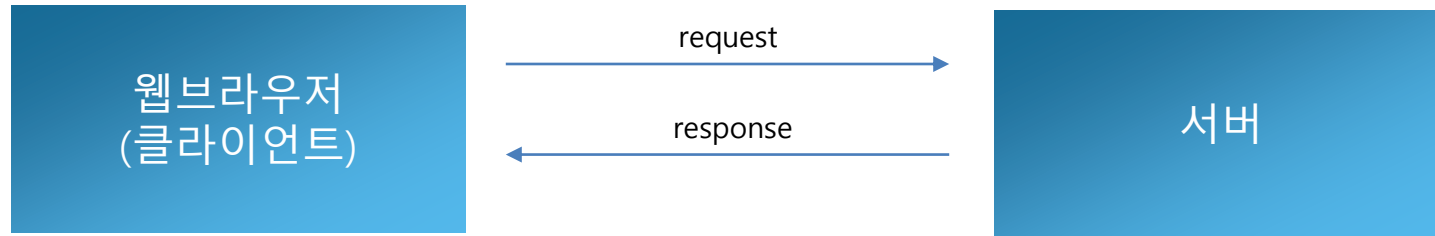
이름<input type="text" value=name %><br />
아이디<input type="text" value=id %><br />
비밀번호<input type="password" value=paswd %><br />
취미<input type="checkbox" value=Arrays.toString(hobby) %><br />
전공<input type="radio" value=major %><br />
프로토콜<input type="radio" value=protocol %><br />
```

http://localhost:8080/jsp_11_1_ex1_requestobj/paramRequest.jsp

이름 : 홍길동
 아이디 : test
 비밀번호 : 1234
 취미 : [swim, running]
 전공 : kor
 프로토콜 : https

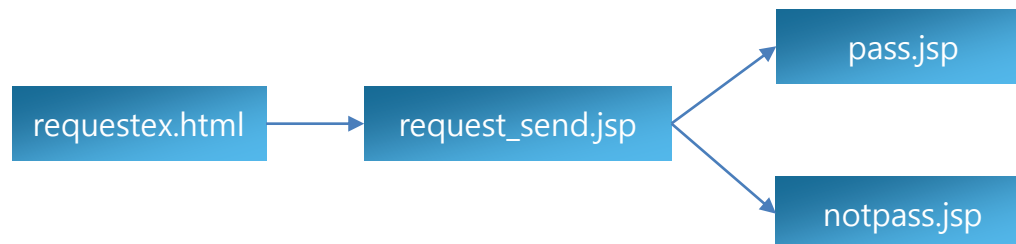
11-2. response 객체의 이해

- ◆ 웹브라우저의 요청에 응답(response) 하는 것을 response라고 하며, 응답(response)의 정보를 가지고 있는 객체를 response객체라고 합니다.



Response객체 관련 Methods(jsp_11_2_ex1_responseobj)

Method	기능
getCharacterEncoding()	응답할 때 문자의 인코딩 형태를 구합니다.
addCookie(Cookie)	쿠키를 지정합니다.
sendRedirect(URL)	지정한 URL로 이동합니다.



```

<!--
    int age;
-->

<--
String strAge = request.getParameter("age");
age = Integer.parseInt(strAge);

if (age >= 20)
    response.sendRedirect("pass.jsp?age=" + age);
else
    response.sendRedirect("notpass.jsp?age=" + age);
-->
  
```

11-1. request 객체 - 실습

- ◆ Request 내장 객체를 이용하여 다음 조건에 맞는 JSP 애플리케이션을 만들고 실행 결과를 확인 하시오.
(jsp_11_1_ex1_requestobj)

http://localhost:8080/jsp_11_1_ex1_requestobj/paramRequestForm.htm

이름
 아이디
 비밀번호
 취미 ☐swim ☐cook ☐running ☐sleeping
 전공 ☐국어 ☐영어 ☐수학
 프로토콜

http://localhost:8080/jsp_11_1_ex1_requestobj

이름 : 홍길동
 아이디 : test
 비밀번호 : 1234
 취미 : [cook, running]
 전공 : eng
 프로토콜 : ftp

11-2. response 객체 - 실습

- ◆ Request, response 내장 객체를 이용하여 다음 조건에 맞는 JSP 애플리케이션을 만들고 실행 결과를 확인 하시오.
(jsp_11_2_ex1_responseobj)

http://localhost:8080/jsp_11_2_ex1_responseobj

당신의 나이는 :

```
if(age >= 20)
    response.sendRedirect("pass.jsp?age=" + age);
else
    response.sendRedirect("notpass.jsp?age=" + age);
```

http://localhost:8080/jsp_11_2_ex1_response

당신의 나이는 30세로 성인입니다.
[처음으로](#)

http://localhost:8080/jsp_11_2_ex1_response

당신의 나이는 19세로 미성년자입니다.
[처음으로](#)

the Standard Proposal

the Standard Proposal

12강. 액션태그

- 액션태그란?
- forward, include, param 태그 살펴보기



12-1. 액션태그란?

- ◆ JSP페이지 내에서 어떤 동작을 하도록 지시하는 태그입니다. 페이지와 페이지 사이를 제어(페이지 이동)하거나 다른 페이지 실행결과 내용을 현재 페이지에 포함(include)하거나, javaBeans 등 입니다.
javaBeans는 추후에 학습하도록 하겠습니다.
- ◆ 우선은 forward, include, param 태그만 살펴보고, 추후 bean을 학습할 때 추가로 학습하도록 하겠습니다.
- ◆ 액션 태그는 XML형식 <jsp:... />를 사용합니다.

12-2. forward, include, param 태그 살펴보기

forward: 다른 페이지로의 이동과 같은 페이지 흐름을 제어합니다.

- ◆ 현재의 페이지에서 다른 지정하는 페이지로 전환할 때 사용합니다. 사용방법이 간단하여 아래의 예제로 쉽게 이해 할 수 있습니다. URL 변경이 일어나지 않습니다.
(jsp_12_2_ex1_actiontagex)

```
<hl>main.jsp 페이지입니다.</hl>
<jsp:forward page="sub.jsp"></jsp:forward>
```

http://localhost:8080/jsp_12_2_ex1_actiontagex/main.jsp

```
<hl>sub.jsp 페이지 입니다.</hl>
```

sub.jsp 페이지 입니다.

12-2. forward, include, param 태그 살펴보기

include: 외부 페이지의 내용을 포함하거나 페이지를 모듈화합니다.

- ◆ 현재의 페이지에 다른 페이지를 삽입할 때 사용합니다.
(jsp_12_2_ex1_actiontagex)

```
<h1> include01.jsp 페이지 입니다.</h1>  
<jsp:include page="include02.jsp" flush="true" />  
<h1> 다시 include01.jsp 페이지 입니다.</h1>
```



http://localhost:8080/jsp_12_2_ex1_actiontagex/include01.jsp

include01.jsp 페이지 입니다.
include02.jsp 페이지 입니다.
다시 include01.jsp 페이지 입니다.

```
<h1>include02.jsp 페이지 입니다.</h1>
```

12-2. forward, include, param 태그 살펴보기

param: forward 및 include 태그에 인자를 추가합니다.

- ◆ forward 및 include 태그에서 데이터 전달을 목적으로 사용되는 태그입니다. 이름과 값으로 이루어져 있습니다.
(jsp_12_2_ex1_actiontagex)

```
<jsp:forward page="forward_param.jsp">
  <jsp:param name="id" value="test" />
  <jsp:param name="paswd" value="1234" />
</jsp:forward>
```

```
<%!
    String id, paswd;
%>

<%
    id = request.getParameter("id");
    paswd = request.getParameter("paswd");
%>

<H1>forward_param.jsp 입니다.</H1>
아이디 : <%= id %> 입니다.<br/>
비밀번호 : <%= paswd %> 입니다.
```

forward_param.jsp 입니다.

아이디 : test 입니다.
비밀번호 : 1234 입니다.

the Standard Proposal

the Standard Proposal

13강. 쿠키

- 쿠키란?
- 쿠키 문법



13-1. 쿠키란?

- ◆ 웹브라우저에서 서버로 어떤 데이터를 요청하면, 서버 측에서는 알맞은 로직을 수행한 후 데이터를 웹브라우저에 응답합니다. 그리고, 서버는 웹브라우저와의 관계를 종료합니다. 이렇게 웹브라우저에 응답 후 관계를 끊는 것은 http프로토콜의 특징입니다.
연결이 끊겼을 때 어떤 정보를 지속적으로 유지하기 위한 수단으로 쿠키라는 방식을 사용합니다.
- ◆ 쿠키는 서버에서 생성하여, 서버가 아닌 클라이언트 측에 특정 정보를 저장합니다. 그리고 서버에 요청할 때마다 쿠키의 속성 값을 참조 또는 변경할 수 있습니다.
- ◆ 쿠키는 4kb로 용량이 제한적이며, 300개까지 데이터 정보를 가질 수 있습니다.

13-2. 쿠키 문법

- ◆ 쿠키는 서버에서 생성되고, 클라이언트에 전송되어 저장됩니다.
- ◆ 쿠키 생성 방법 및 관련 메서드를 살펴봅니다.



13-2. 쿠키 문법

쿠키 관련 메서드

메서드	설명
setMaxAge()	쿠키 유효기간을 설정
setPath()	쿠키 사용의 유효 디렉토리를 설정
setValue()	쿠키의 값을 설정
setVersion()	쿠키 버전을 설정
getMaxAge()	쿠키 유효기간 정보를 얻습니다.
getName()	쿠키 이름을 얻습니다.
getPath()	쿠키 사용의 유효 디렉토리 정보를 얻습니다.
getValue()	쿠키의 값을 얻습니다.
getVersion()	쿠키의 버전을 얻습니다.

13-2. 쿠키 문법

쿠키 생성

- ◆ Cookie 클래스를 사용하여 쿠키를 생성해야 하며, Cookie() 메서드를 사용하며 형식은 아래와 같습니다.
- ◆ 쿠키를 생성한 후에는 반드시 response 내부 객체의 addCookie() 메서드로 쿠키를 설정해야 합니다.

생성 형식: `Cookie Cookie(String name, String value)`

<Cookie() 메서드 사용 예제>

```
Cookie cookie = new Cookie("memberId", "admin");  
response.addCookie(cookie);
```

쿠키 정보

- ◆ 생성된 쿠키의 정보는 request 내부 객체의 get_cookies() 메서드를 이용 쿠키 객체를 얻은 후 getName(), getValue() 메서드를 사용 쿠키 이름과 쿠키 값을 얻습니다.

<쿠키 객체 얻기(get_cookies() 메서드 사용) 예제>

```
Cookie[] cookies = request.get_cookies();
```

<쿠키 객체 정보 얻기(getName(), getValue() 메서드 사용) 예제>

```
Cookie[] cookies = request.get_cookies();
```

```
For(int i=0; i < cookies.length; i++) {  
    out.println(cookies[i].getName() + " | " + cookies[i].getValue() + "<br/>");  
}
```

13-2. 쿠키 문법

쿠키 삭제

- ◆ Cookie 클래스는 쿠키 삭제 기능을 별도로 제공하지 않고, 쿠키를 더 유지할 필요가 없으면 쿠키의 유효 시간을 만료하면 됩니다. 즉, 유효기간 결정 메서드 `setMaxAge()`에 유효 기간을 0으로 설정하여 쿠키를 삭제합니다.

유효기간 설정 메서드: `void setMaxAge(int age);`

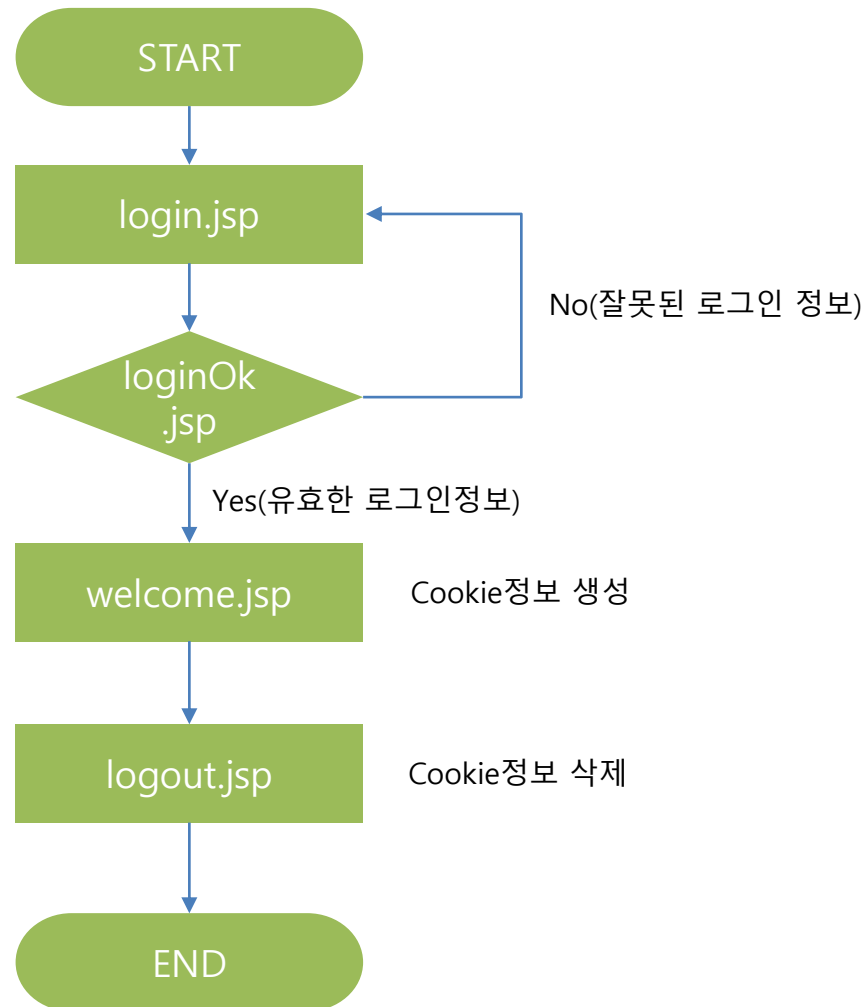
<setMaxAge() 메서드 사용 예제>

```
Cookie cookie = new Cookie("memberId", "admin");  
cookie.setMaxAge(0);  
response.addCookie(cookie);
```

쿠키 예제(jsp_13_2_ex1_cookieex)

- ◆ 예제를 통해서 쿠키 생성, 속성 설정, `response`객체 탑재 방법, 쿠키 삭제 등을 학습합니다.

13. 쿠키 실습



the Standard Proposal

the Standard Proposal

14강. 세션

- 세션이란?
- 세션 문법



14-1. 세션이란?

- ◆ 쿠키와 마찬가지로 서버와의 관계를 유지하기 위한 수단입니다.
단, 쿠키와 달리 클라이언트의 특정 위치에 저장되는 것이 아니라, 서버 상에 객체로 존재 합니다.
따라서 세션은 서버에서만 접근이 가능하여 보안이 좋고, 저장할 수 있는 데이터에 한계가 없습니다.

14-2. 세션 문법

- ◆ 세션은 클라이언트의 요청이 발생하면 자동 생성(jsp 컨테이너) 됩니다.
- ◆ Session이라는 내부 객체를 지원하여 세션의 속성을 설정할 수 있습니다.



14-2. 세션 문법

세션 관련 메서드

메서드	설명
setAttribute(String name, Object value)	세션에 데이터를 저장 합니다.
getAttribute(String name)	세션에서 데이터를 얻습니다.
getAttributeName()	세션에 저장되어 있는 모든 데이터의 이름(유니크한 키값)을 얻습니다.
getId()	자동 생성된 유니크한 아이디를 얻습니다.
isNew()	세션이 최초 생성 되었는지, 이전에 생성된 세션인지 구분합니다.
setMaxInactiveInterval(int interval)	세션의 유효 시간을 설정합니다. 초 단위로 값을 지정합니다.
getMaxInactiveInterval()	세션의 유효 시간을 얻습니다. 가장 최근 요청 시점을 기준으로 카운트 됩니다. (참고로 web.xml <session-config>에 30분으로 설정되어 있고 변경 가능합니다.)
removeAttribute(String name)	세션에서 특정 데이터를 삭제 합니다.
invalidate()	세션의 모든 데이터를 삭제 합니다.

14-2. 세션 문법

세션 생성

- ◆ 세션 생성은 session 내부 객체의 `setAttribute()` 메서드를 사용하며 형식은 아래와 같습니다.
- ◆ `setAttribute()` 메서드로 세션의 속성을 설정하면 계속 세션 상태를 유지할 수 있으며, 동일 세션의 속성 이름으로 세션을 설정하면 마지막에 설정한 것이 세션 속성 값이 됩니다.

생성 형식: `void setAttribute(String name, Object value);`

<setAttribute() 메서드 사용 예제>

```
session.setAttribute("memberId", "admin");
```

세션 정보

- ◆ 생성된 세션의 정보를 얻어오려면 session 내부 객체의 `getAttribute()` 또는 `getAttributeNames()` 메서드를 사용합니다.
- 단일 세션 정보 얻기 : 세션에 저장된 하나의 세션 속성 이름에 대한 속성 값 얻기

메서드 형식: `Object getAttribute(String name);`

<getAttribute() 메서드 사용 예제>

```
String id = (String)session.getAttribute("memberId");
```

14-2. 세션 문법

세션 정보

- ◆ 생성된 세션의 정보를 얻어오려면 session내부 객체의 `getAttribute()` 또는 `getAttributeNames()` 메서드를 사용합니다.
 - 다중 세션 정보 얻기 : 세션에 저장된 여러 개의 세션 속성 이름에 대한 속성 값 얻기

메서드 형식: `Enumeration getAttributeNames();`

<getAttributeNames() 메서드 사용 예제>

```
Enumeration enum = session.getAttributeNames();
```

```
while(enum.hasMoreElements) {  
    String name = enum.nextElement().toString();  
    String value = session.getAttribute(name).toString();  
}
```

➤ Enumeration 사용: `import java.util.Enumeration`

14-2. 세션 문법

세션 삭제

- ◆ 생성된 세션을 더 유지할 필요가 없으면 session 내부 객체의 removeAttribute() 또는 invalidate() 메서드로 삭제합니다.

- 단일 세션 삭제하기 : 세션에 저장된 하나의 세션 속성 이름을 삭제

메서드 형식: void removeAttribute(String name);

<removeAttributeNames() 메서드 사용 예제>

```
Session.removeAttribute("memberId");
```

- 다중 세션 삭제하기 : 세션에 저장된 모든 세션 속성 이름을 삭제

메서드 형식: void invalidate()

<invalidate() 메서드 사용 예제>

```
session.invalidate();
```

14-2. 세션 문법

세션 유효 시간 설정

- ◆ 세션 유효 시간은 세션을 유지하기 위한 세션의 일정 시간을 말합니다. 웹브라우저에 마지막으로 접근한 시간부터 일정 시간 이내에 다시 웹브라우저에 접근하지 않으면 자동으로 세션이 종료됩니다. 이러한 세션 유효 시간 설정은 session 내부 객체의 `setMaxInactiveInterval()` 메서드를 사용합니다.

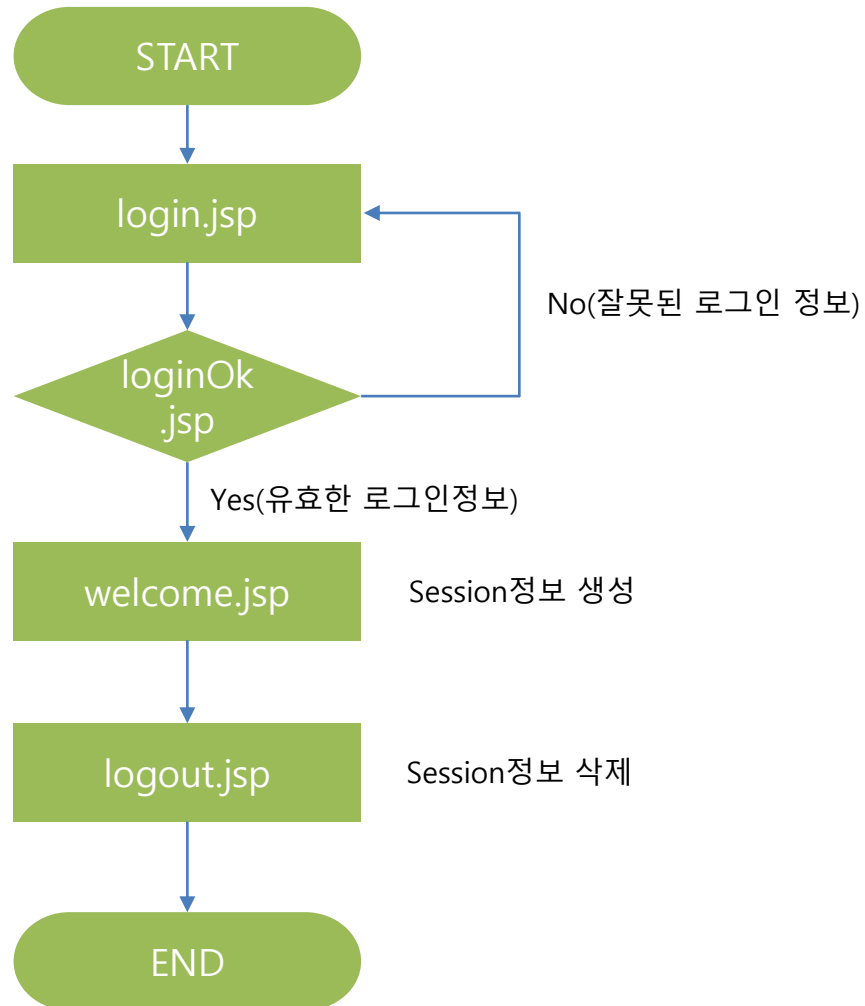
세션 유효시간 설정 메서드: `void setMaxInactiveInterval(int interval)`

<setMaxInactiveInterval() 메서드 사용 예제>
`session. setMaxInactiveInterval(60*60);`

세션 예제(jsp_14_2_ex1_sessionex)

- ◆ `sesseion` 메서드를 이용해서 데이터를 저장 및 삭제해 봅니다.

14. 세션 실습



15강. 예외 페이지

- 예외 페이지의 필요성
- page 지시자를 이용한 예외처리
- web.xml 파일을 이용한 예외처리



15-1. 예외 페이지의 필요성

- ◆ JAVA언어에서 아마도 예외 처리에 대해서 사전 학습을 했을 것 입니다. JSP, Servlet에서도 예외가 발생할 수 있습니다. 예외적인 상황이 발생했을 경우 웹 컨테이너(톰캣)에서 제공되는 기본적인 예외 페이지가 보여 진다면, 사용자로 하여금 뭔가 불쾌한 느낌이 들면서, 다시는 해당 사이트에 접속하려 들지 않을 것 입니다. 따라서 약간은 딱딱한 에러 페이지를 보다 친근한 느낌이 느껴지는 페이지로 유도 할 수 있습니다.

← → ↻ 📄 http://localhost:8080/jsp_15_2_ex1_exceptionex/exceptionex1.jsp

HTTP 상태 500 – 내부 서버 오류

타입 예외 보고

메시지 행 [12]에서 [/exceptionex1.jsp]를(를) 처리하는 중 예외 발생

설명 서버가, 해당 요청을 충족시키지 못하게 하는 예기치 않은 조건을 맞닥뜨렸습니다.

예외

```
org.apache.jasper.JasperException: 행 [12]에서 [/exceptionex1.jsp]를(를) 처리하는 중 예외 발생

9:  </head>
10:  <body>
11:      <%
12:          int i = 40 / 0;
13:      %>
14:
15:  </body>
```

Stacktrace:

```
org.apache.jasper.servlet.JspServletWrapper.handleJspException(JspServletWrapper.java:599)
org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:488)
org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:380)
org.apache.jasper.servlet.JspServlet.service(JspServlet.java:328)
jakarta.servlet.http.HttpServlet.service(HttpServlet.java:631)
org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
```

근본 원인 (root cause)

```
java.lang.ArithmeticException: / by zero
org.apache.jsp.exceptionex1_jsp._jspService(exceptionex1_jsp.java:133)
org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:70)
jakarta.servlet.http.HttpServlet.service(HttpServlet.java:631)
org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:456)
org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:380)
org.apache.jasper.servlet.JspServlet.service(JspServlet.java:328)
jakarta.servlet.http.HttpServlet.service(HttpServlet.java:631)
org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
```

비고 근본 원인(root cause)의 풀 스택 트레이스를, 서버 로그들에서 확인할 수 있습니다.

15-2. page 지시자를 이용한 예외처리

(jsp_15_2_ex1_exceptionex)

```
<%@ page language="java" contentType="text"
    pageEncoding="EUC-KR"%>
<%@ page errorPage="errorPage.jsp" %>
```

```
<%
    int i = 40 / 0;
%>
```

```
<%@ page isErrorPage="true" %>
<%
    response.setStatus(200);
%>
```

```
<%= exception.getMessage() %>
```

(isErrorPage="true" 가 되어야 exception 객체 사용가능)

예외 발생



예외 페이지

15-3. web.xml 파일을 이용한 예외처리

(jsp_15_3_ex1_exceptionex)

```
<!-- 예외페이지정의 추가 -->
```

```
<error-page>
```

```
<error-code>404</error-code>
```

← 404에러 발생 시 error404.jsp로 페이지 이동

```
<location>/error404.jsp</location>
```

```
</error-page>
```

```
<error-page>
```

```
<error-code>500</error-code>
```

← 500에러 발생 시 error500.jsp로 페이지 이동

```
<location>/error500.jsp</location>
```

```
</error-page>
```

16강. 자바 빈

- 빈 이란?
- 빈 만들기
- 빈 관련 액션 태그(useBean, getProperty, setProperty)



16-1. 빈 이란?

- ◆ 반복적인 작업을 효율적으로 하기 위해 빈을 사용 합니다. 빈이란? JAVA언어의 데이터(속성)와 기능(메서드)로 이루어진 클래스입니다.
jsp페이지를 만들고, 액션 태그를 이용하여 빈을 사용 합니다. 그리고 빈의 내부 데이터를 처리 합니다.

16-2. 빈 만들기

- ◆ JAVA언어를 학습하면서 데이터 객체를 많이 만들어본 경험이 있을 것입니다. 데이터 객체에는 데이터가 있어 그에 해당하는 getter와 setter가 있습니다.
빈을 만든다는 것은 데이터 객체를 만들기 위한 클래스를 만드는 것입니다.
(jsp_16_2_ex1_beanex)

```
1 package kr.co.jsplec.ex;
2
3 public class Student {
4     private String name;
5     private int age;
6     private int grade;
7     private int studentNum;
8 }
```

```
    public String getName() {
        return this.name;
    }

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

    public int getAge() {
        return this.age;
    }

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

16-3. 빈 관련 액션 태그(useBean, setProperty, getProperty)

- ◆ 액션태그 중에서 Bean과 관련한 태그가 있습니다. 주로 데이터를 업데이트하고, 얻어오는 역할을 합니다.

useBean

- ◆ 특정 Bean을 사용한다고 명시할 때 사용합니다.

```
<jsp:useBean id="student" class="kr.co.jsplec.ex.Student" scope="page" />
```

빈 이름

클래스 이름

스코프 범위

Scope

page: 생성된 페이지 내에서만 사용 가능합니다.

request: 요청된 페이지 내에서만 사용 가능합니다.

session: 웹 브라우저의 생명 주기와 동일하게 사용 가능합니다.

application: 웹 어플리케이션 생명 주기와 동일하게 사용 가능합니다.

16-3. 빈 관련 액션 태그(useBean, setProperty, getProperty)

setProperty

- ◆ 데이터 값을 설정할 때 사용합니다.

```
<jsp:setProperty name="student" property="name" value="홍길동" />
```

빈 이름

속성 이름

속성(데이터) 값

getProperty

- ◆ 데이터 값을 가져올 때 사용합니다.

```
<jsp:getProperty name="student" property="name" />
```

빈 이름

속성 이름

Oracle Standard Proposal

Oracle Standard Proposal

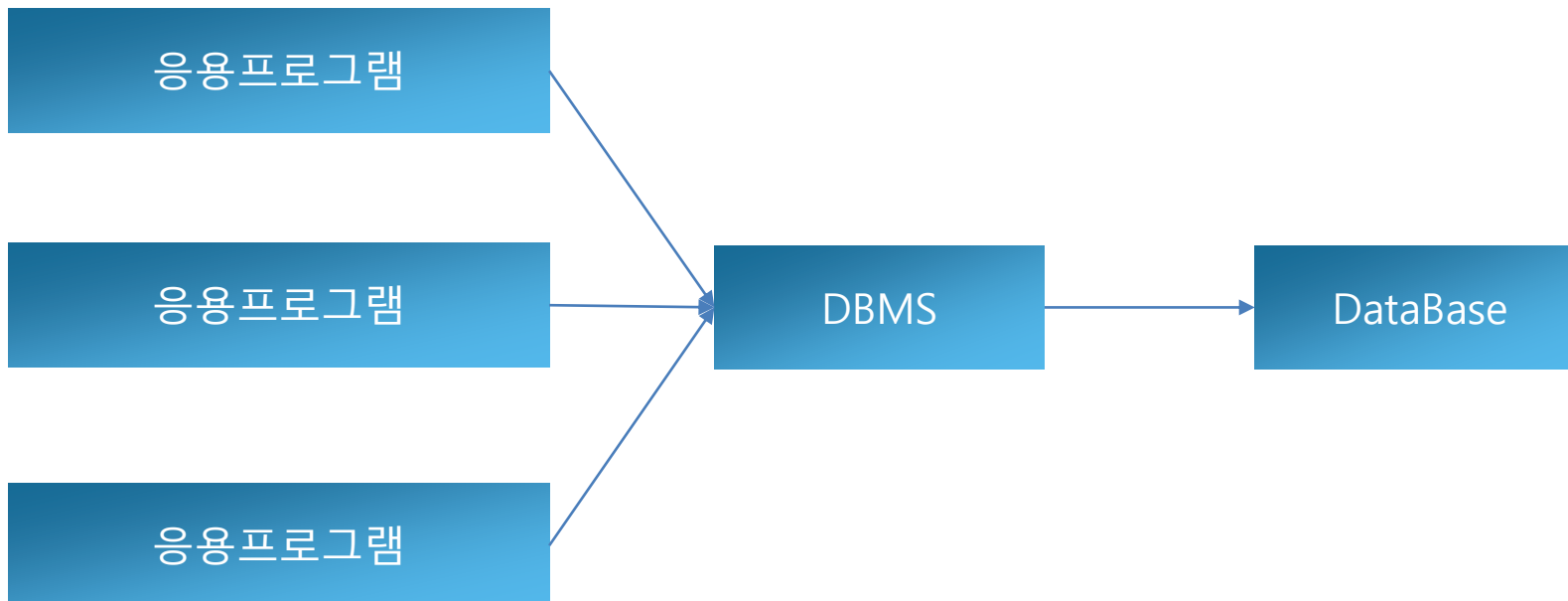
17강. 데이터 베이스 - I

- 데이터베이스 개요
- Oracle 설치
- 기본적인 SQL문 익히기



17-1. 데이터베이스 개요

- ◆ 다량의 데이터를 관리하기 위한 수단으로 데이터베이스가 있습니다. 데이터베이스는 데이터의 추가, 수정, 삭제, 검색, 이동 등의 기능이 쉽게 되어 있어 사용자로 하여금 원하는 데이터를 빠르게 이용할 수 있게 합니다.
- ◆ 이러한 데이터베이스를 관리하는 도구가 DBMS(DataBase Management System, 데이터베이스 관리시스템) 입니다. DBMS는 언어와 데이터베이스를 연결해 주는 도구이며, 일반적으로 데이터베이스와 동일 시 합니다.
- ◆ DBMS의 종류는 다양하며, 우리가 사용할 것은 RDBMS(Relational DataBase Management System)인 Oraclec 입니다.



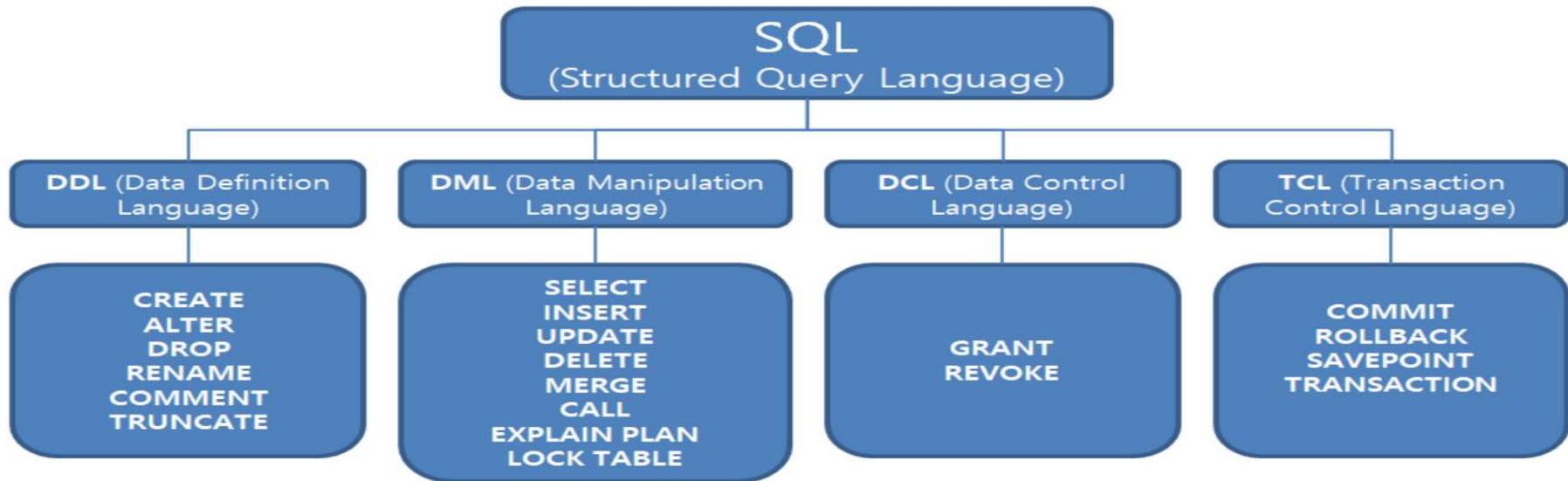
17-2. 오라클 설치

- ◆ otn.oracle.com 에 접속하여 회원 가입 후 원하는 버전을 다운 받아 설치합니다.

17-3. SQL(Structured Query Language) 익히기

- ◆ SQL(Structured Query Language)로 데이터베이스의 데이터를 관리하기 위한 언어입니다.
- ◆ 관계형 데이터베이스 관리시스템에서 자료의 검색과 관리, 데이터베이스 스키마 생성과 수정, 데이터베이스 객체 접근 조정 관리를 위해 고안되었다.

SQL 구조



17-3. SQL(Structured Query Language) 익히기

DDL(Data Definition Language) : 데이터 정의 언어

- 테이블과 컬럼을 정의하는 명령어로 생성, 수정, 삭제 등의 데이터 전체 골격을 결정하는 역할을 한다.
- DDL은 명령어를 입력하는 순간 작업이 즉시 반영(Auto Commit) 되기 때문에 사용할 때 주의해야 한다.

명령어	내용
CREATE	테이블 생성
ALTER	테이블 구조 수정
DROP	테이블 삭제
RENAME	테이블명 변경
TRUNCATE	테이블 초기화

17-3. SQL(Structured Query Language) 익히기

DML(Data Manipulation Language) : 데이터 조작 언어

- 데이터베이스의 내부 데이터를 관리하기 위한 언어로 데이터 조회, 추가, 변경, 삭제 등의 작업을 수행하기 위해 사용된다.

명령어	내용
SELECT	테이블 데이터 조회
INSERT	테이블 데이터 추가
UPDATE	테이블 데이터 변경
DELETE	테이블 데이터 삭제

17-3. SQL(Structured Query Language) 익히기

DCL(Data Control Language) : 데이터 제어 언어

- 데이터 관리 목적으로 보안, 무결성, 회복, 병행 제어 등을 정의하는데 사용한다.
- 데이터베이스에 접근하여 읽거나 쓰는 것을 제한할 수 있는 권한을 부여하거나 박탈할 수 있다.

명령어	내용
GRANT	권한 부여
REVOKE	권한 박탈

17-3. SQL(Structured Query Language) 익히기

TCL(Transaction Control Language) : 트랜잭션 제어 언어

- 데이터 제어가 아닌 트랜잭션을 제어할 때 사용한다.
- 논리적인 작업 단위를 묶어 DML에 의해 조작된 결과를 트랜잭션별로 제어한다.

명령어	내용
COMMIT	모든 작업을 정상적으로 처리하겠다는 명령어, 물리적으로 저장
ROLLBACK	모든 작업을 다시 돌려 놓겠다는 명령어, 물리적으로 취소
SAVEPOINT	Commit 전에 특정시점 까지만 반영(COMMIT)하거나, 롤백(ROLLBACK)하겠다는 명령어

- 트랜잭션 : 데이터베이스의 상태를 변화시키기 위해 수행하는 작업의 단위

18강. 데이터 베이스 - II

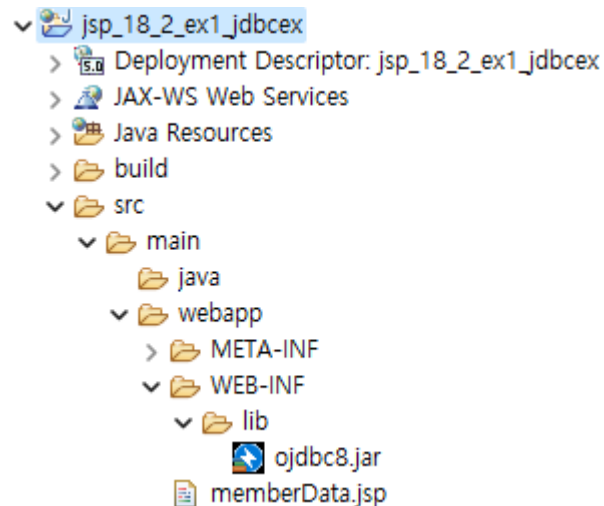
- JDBC 살펴보기
- Statement객체 살펴보기



18-1. JDBC 살펴보기

- ◆ JAVA 프로그램에서 SQL문을 실행하여 데이터를 관리하기 위한 API 입니다.
- ◆ JDBC의 특징은 다양한 데이터베이스에 대해서 별도의 프로그램을 만들 필요없이, 해당 데이터베이스의 JDBC를 이용하면 하나의 프로그램으로 데이터베이스를 관리할 수 있습니다.
- ◆ 우리는 Oracle을 사용하므로, Oracle용 JDBC를 사용하며, 이것은 Oracle을 설치하면 자동으로 설치되고, 이클립스에서 해당 클래스 파일을 복사하면 됩니다.

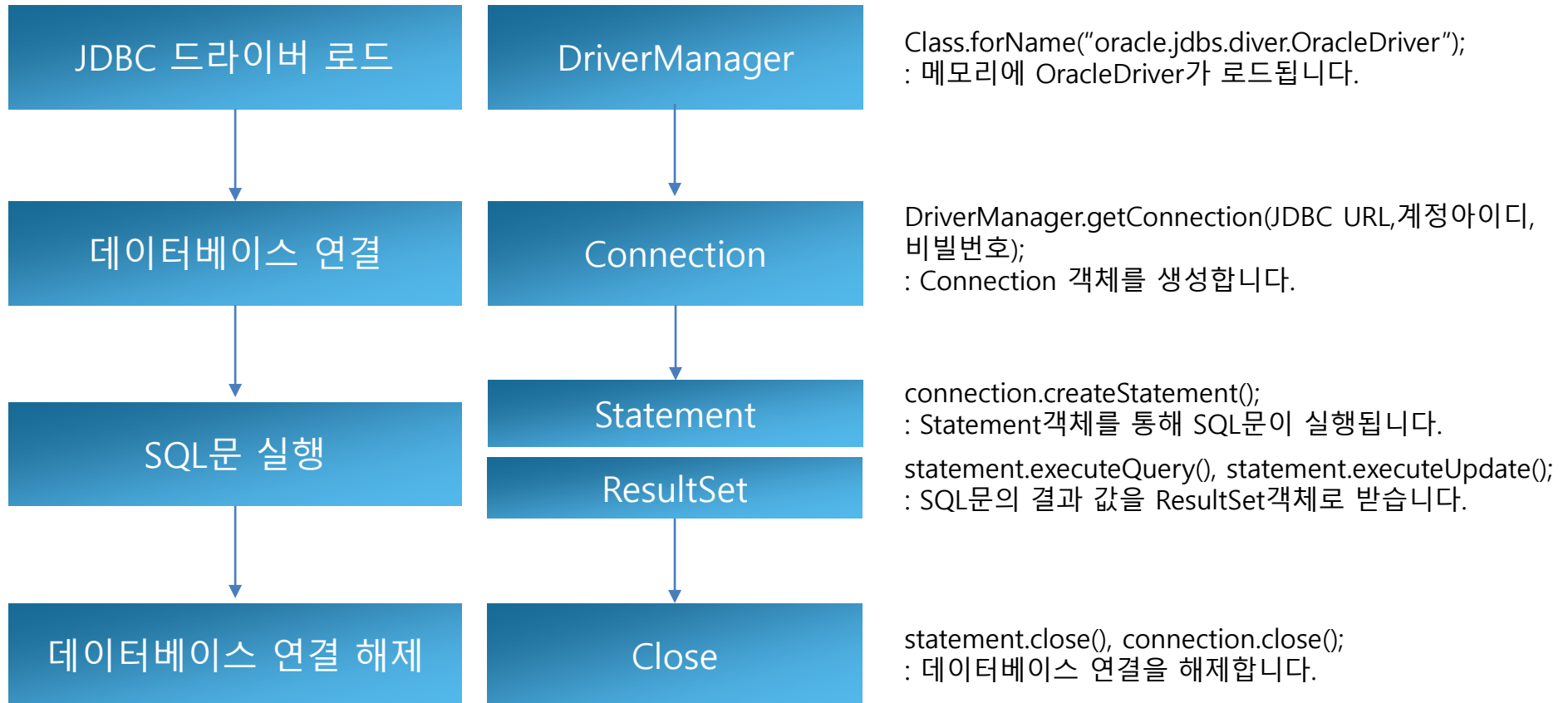
오라클 드라이버를 사용하기 위한 ojdbc8.jar 파일 복사



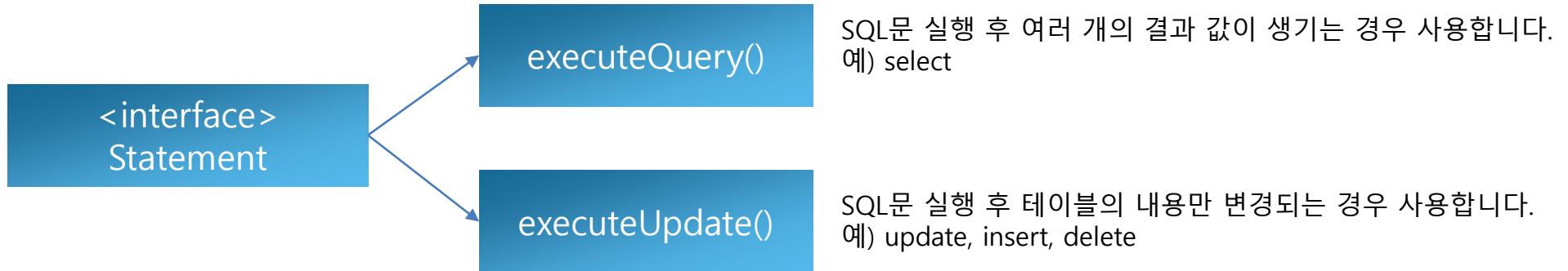
해당 프로젝트의 WEB-INF/lib 에 ojdbc8.jar 파일 복사

18-1. JDBC 살펴보기

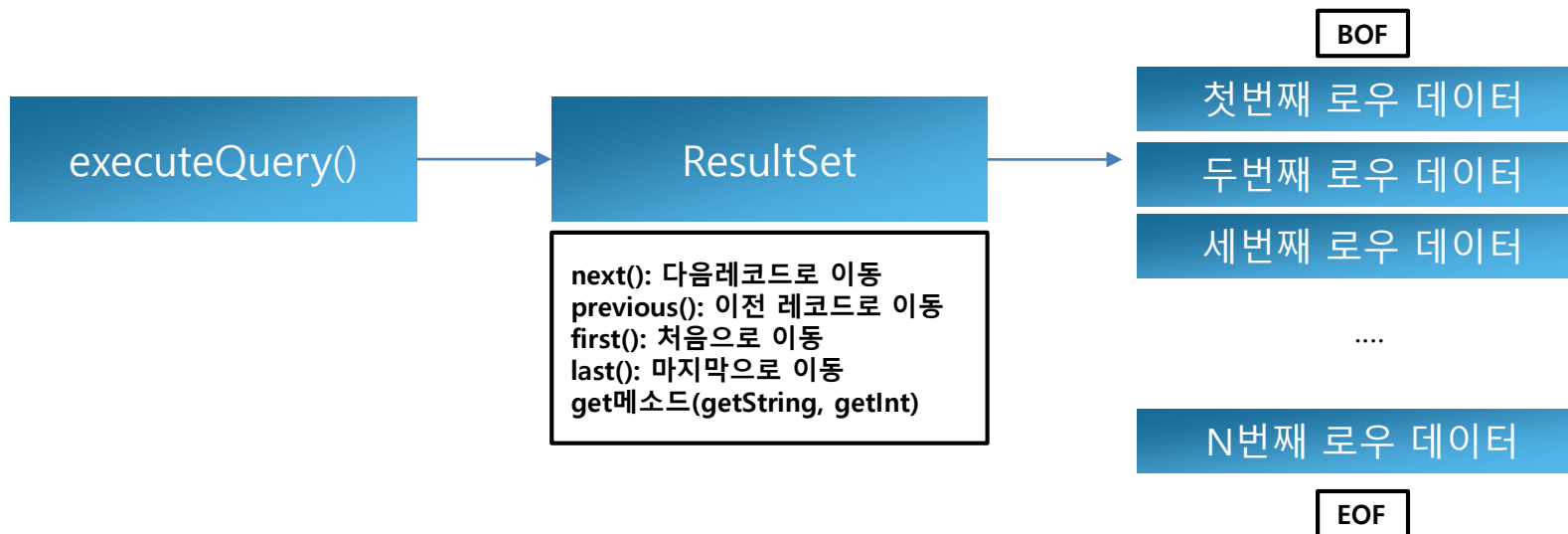
데이터베이스 연결 순서



18-2. Statement 객체 살펴보기



executeQuery() 실행 후 반환되는 레코드 Set



18-2. Statement 객체 살펴보기

JDBC 예제(jsp_18_2_ex1_jdbcex)

```
create table member (  
  id varchar2(20),  
  pw varchar2(20),  
  name varchar2(20),  
  phone varchar2(20)  
);
```

```
insert into member(id, pw, name, phone) values ('abc', '123', '홍길동', '010-1234-5678');  
insert into member(id, pw, name, phone) values ('def', '456', '홍길순', '010-2234-5678');  
insert into member(id, pw, name, phone) values ('ghi', '789', '홍길이', '010-3234-5678');  
insert into member(id, pw, name, phone) values ('jkl', '234', '홍길남', '010-4234-5678');
```

18-2. Statement 객체 살펴보기

JDBC 예제(jsp_18_2_ex1_jdbcex)

<%!

```
Connection connection;
Statement statement;
ResultSet resultSet;

String driver = "oracle.jdbc.driver.OracleDriver";
String url = "jdbc:oracle:thin:@tlf.co.kr:1521:orcl";
String uid = "kopo";
String upw = "kopo";
String query = "select * from member";
```

>%

<%

```
try {
    Class.forName(driver);
    connection = DriverManager.getConnection(url, uid, upw);
    statement = connection.createStatement();
    resultSet = statement.executeQuery(query);

    while(resultSet.next()) {
        String id = resultSet.getString("id");
        String pw = resultSet.getString("pw");
        String name = resultSet.getString("name");
        String phone = resultSet.getString("phone");

        out.println("아이디: " + id + ", 비밀번호: " + pw + ", 이름: " + name + ", 전화번호: " + phone + "<br/>");
    }
}
```

memberData.jsp × Insert title here ×

jsp_18_2_ex1_jdbcex/src/main/webapp/memberData.jsp jdbcex/memberData.jsp

아이디: abc, 비밀번호: 123, 이름: 홍길동, 전화번호: 010-1234-5678
 아이디: def, 비밀번호: 456, 이름: 홍길순, 전화번호: 010-2234-5678
 아이디: ghi, 비밀번호: 789, 이름: 홍길이, 전화번호: 010-3234-5678
 아이디: jkl, 비밀번호: 234, 이름: 홍길남, 전화번호: 010-4234-5678

18. 요약

1. JDBC 개요

- JDBC는 JAVA/JSP 프로그램 내에서 데이터베이스와 관련된 작업을 처리할 수 있도록 도와주는 자바 표준 인터페이스로, 관계형 데이터베이스 시스템에 접근하여 SQL문을 실행하기 위한 JAVA API 입니다.

2. JDBC 드라이버 로딩 및 DBMS 접속/해제

- JDBC 드라이버 로딩 단계에서는 드라이버 인터페이스를 구현하는 작업으로 Class.forName() 메서드를 이용, JDBC 드라이버를 로딩합니다. JDBC 드라이버가 로딩되면 자동으로 객체가 생성되고 DriverManager 클래스에 등록됩니다.
- JDBC 드라이버에서 데이터베이스와 연결된 커넥션을 가져오기 위해 DriverManager 클래스의 getConnection() 메서드를 사용합니다.
- SQL 사용을 위해 Connection 객체로 부터 Statement 객체를 생성하여 작업을 수행하고, 데이터베이스 연결이 더 이상 필요하지 않으면 close() 메서드로 Connection 객체를 해제합니다.

3. SQL 쿼리 실행

- Statement 객체를 사용하며, executeQuery() 메서드는 SELECT 쿼리 문을 통해 데이터를 검색하는데 사용하고, executeUpdate() 메서드는 INSERT, UPDATE, DELETE 쿼리 문을 통해 데이터를 추가, 수정, 삭제하는 데 사용합니다.
- SELECT 쿼리 문의 실행 결과를 가져오기 위해 ResultSet 객체를 사용합니다.

19강. 데이터 베이스 - III

- 회원가입 및 회원정보 수정 프로그래밍

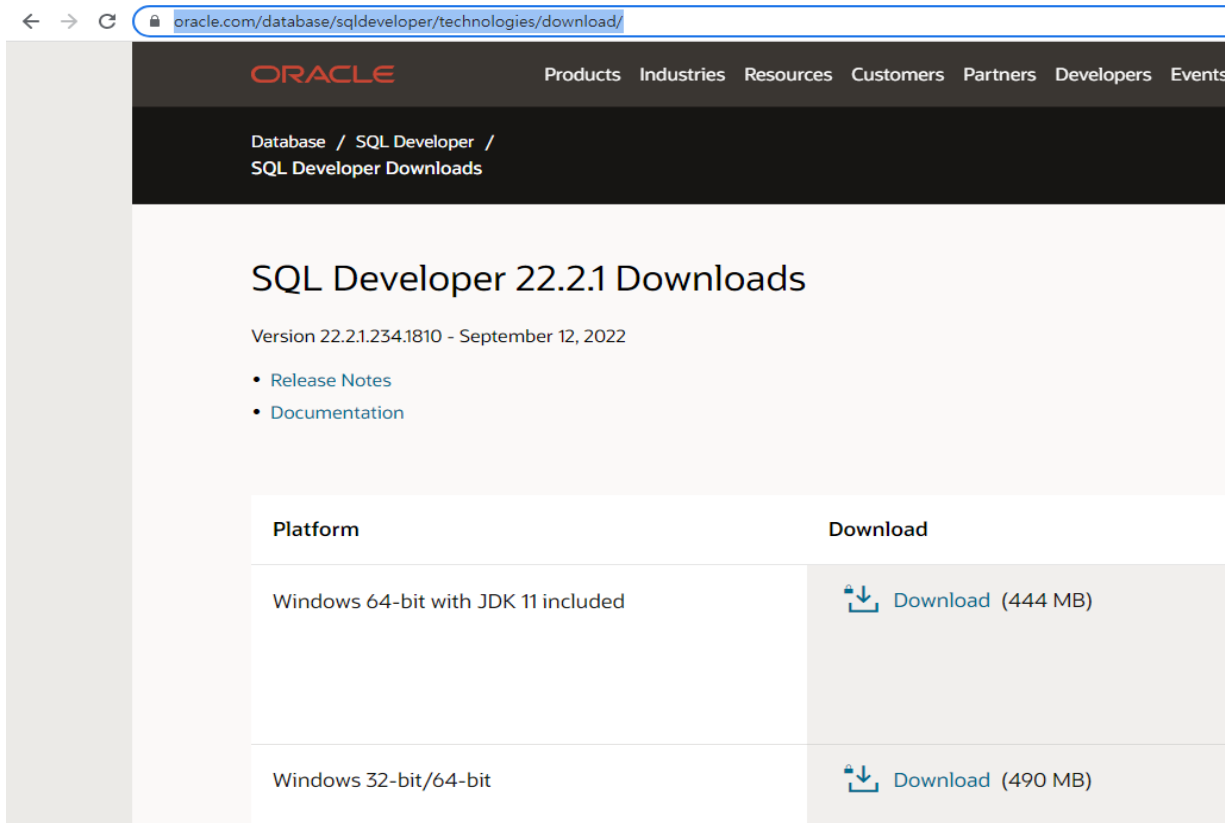


19-1. 회원가입 및 회원정보 수정 프로그래밍

오라클 SQL Developer(오라클에서 제공하는 데이터베이스 전용 GUI툴) 설치

아래 주소로 접속 후 다운 받습니다.

<https://www.oracle.com/database/sqldeveloper/technologies/download/>



← → ↻ [oracle.com/database/sqldeveloper/technologies/download/](https://www.oracle.com/database/sqldeveloper/technologies/download/)

ORACLE Products Industries Resources Customers Partners Developers Events

Database / SQL Developer / SQL Developer Downloads

SQL Developer 22.2.1 Downloads

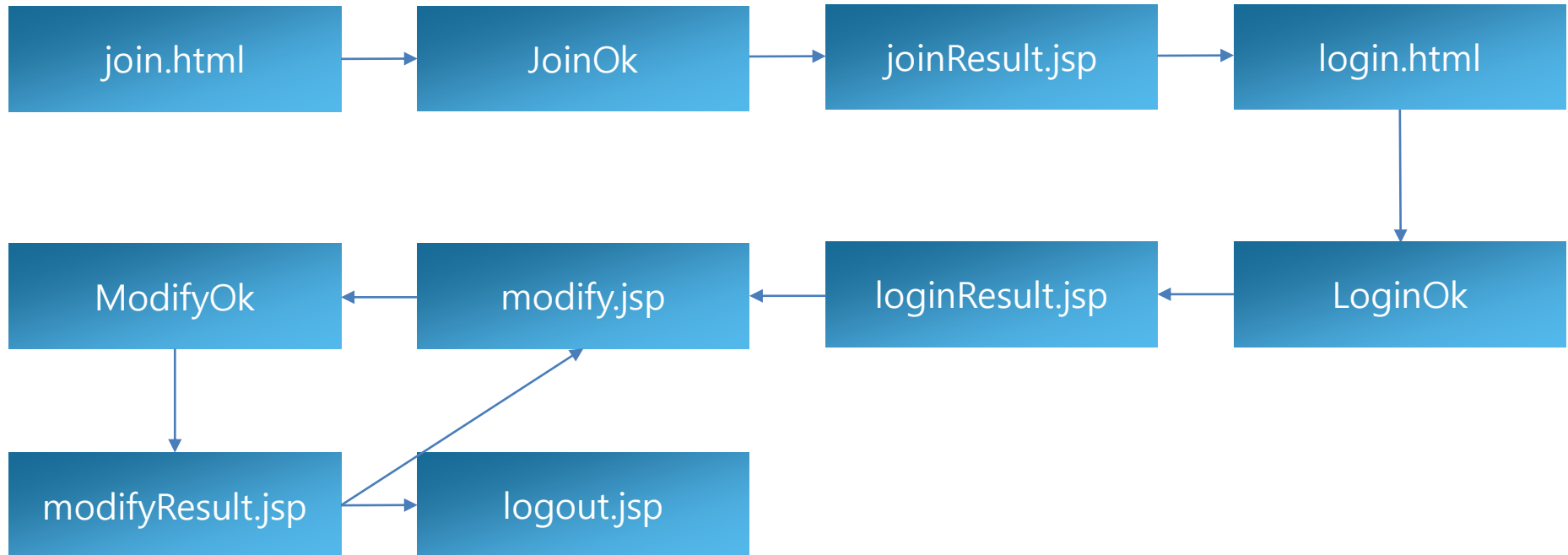
Version 22.2.1.234.1810 - September 12, 2022

- [Release Notes](#)
- [Documentation](#)

Platform	Download
Windows 64-bit with JDK 11 included	Download (444 MB)
Windows 32-bit/64-bit	Download (490 MB)

19-1. 회원가입 및 회원정보 수정 프로그래밍

- ◆ 회원가입 및 회원정보 수정 프로그래밍을 통한 JDBC 사용법을 학습합니다.
(jsp_19_1_ex1_memberex)



19. 회원가입 및 회원정보 수정 프로그래밍 실습 (jsp_19_1_ex1_memberex)

◆ 테이블 생성

```
CREATE TABLE KOPO.MEMBER
(
  ID      VARCHAR2(20 BYTE),      -- 아이디
  PW      VARCHAR2(20 BYTE),      -- 비밀번호
  NAME    VARCHAR2(20 BYTE),      -- 이름
  PHONE1  VARCHAR2(20 BYTE),      -- 전화번호
  PHONE2  VARCHAR2(20 BYTE),      -- 전화번호
  PHONE3  VARCHAR2(20 BYTE),      -- 전화번호
  GENDER  VARCHAR2(5 BYTE)        -- 성별구분
);
```

- 스키마 : 각자 개인 스키마 사용
- 테이블 명 : 개인별 지정
- 컬럼 속성 : varchar2로 개별 지정

20강. 커넥션 풀

- DAO, DTO
- PreparedStatement
- 커넥션 풀(DBCP)



20-1. DAO, DTO

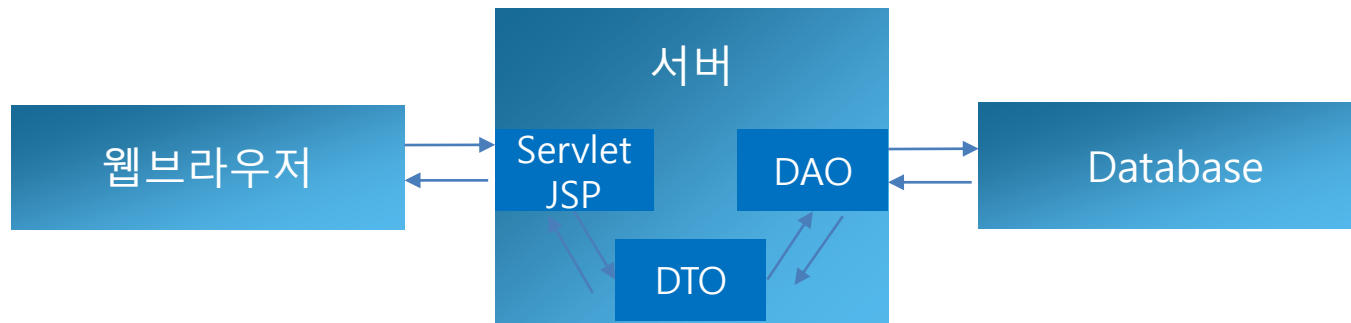
- ◆ DAO : Data Access Object
 - ◆ DTO : Data Transfer Object
- (jsp_20_1_ex1_daotoex)

DAO

- ◆ 데이터베이스에 접속해서 데이터 추가, 삭제, 수정 등의 작업을 하는 클래스입니다. 일반적인 JSP 혹은 Servlet 페이지 내에 위의 로직을 함께 기술 할 수도 있지만, 유지보수 및 코드의 모듈화를 위해 별도의 DAO 클래스를 만들어 사용합니다.

DTO

- ◆ DAO클래스를 이용하여 데이터베이스에서 데이터를 관리할 때 데이터를 일반적인 변수에 할당하여 작업 할 수도 있지만, 해당 데이터의 클래스를 만들어 사용합니다.



20-2. PreparedStatement객체 살펴보기

- ◆ SQL문 실행을 위해 Statement객체를 이용하였습니다. Statement객체의 경우 중복 코드가 많아지는 단점이 있습니다. 이러한 단점을 보완한 PreparedStatement객체에 대해 살펴봅니다.

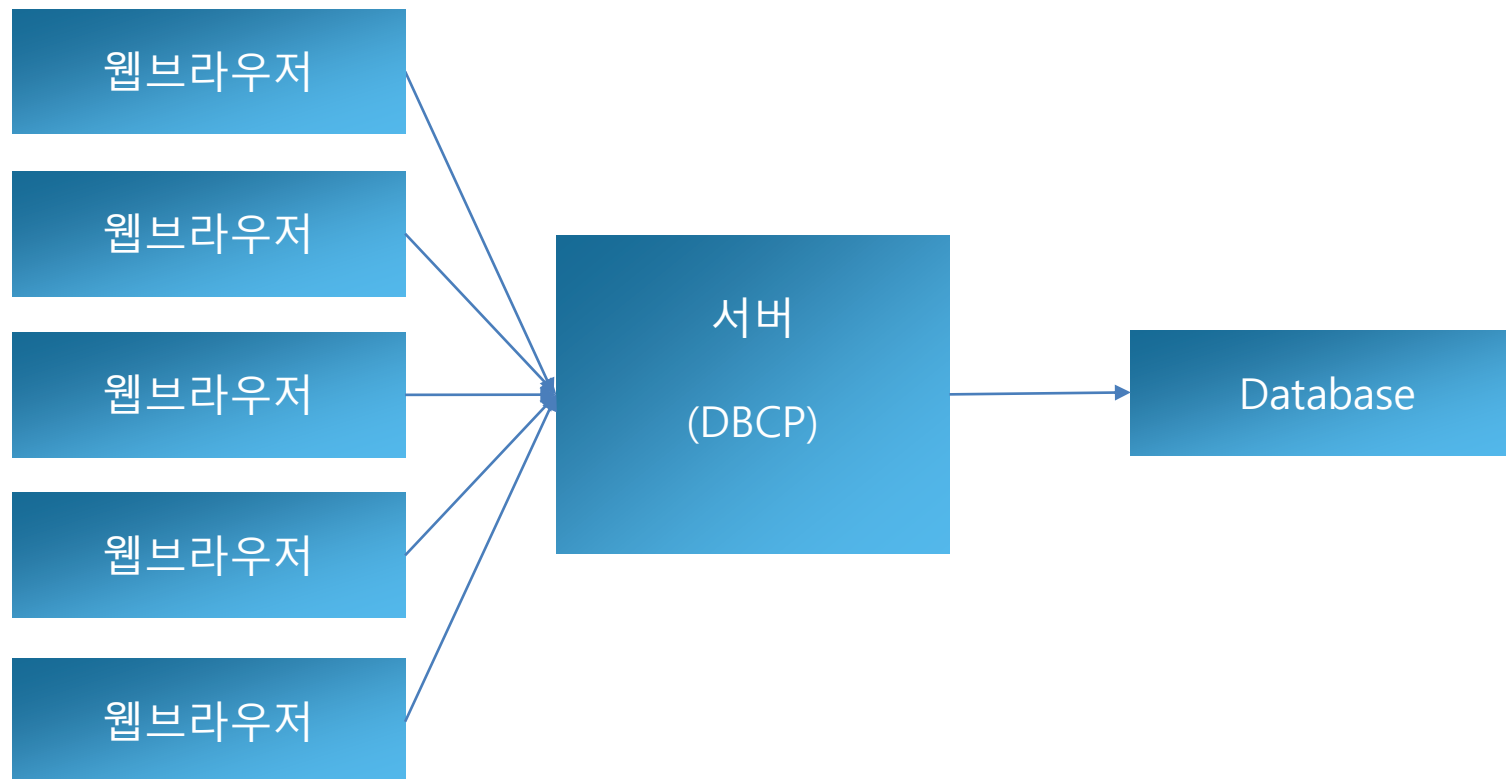
(jsp_20_2_ex1_preparedex)

```
Class.forName(driver);
conn = DriverManager.getConnection(url, uid, upw);
int cnt;
String query = "insert into memberforpre(id, pw, name, phone) values(?, ?, ?, ?)";
pstmt = conn.prepareStatement(query);

pstmt.setString(1, "abc");
pstmt.setString(2, "123");
pstmt.setString(3, "홍길동");
pstmt.setString(4, "010-1234-5678");
cnt = pstmt.executeUpdate();
```

20-3. 커넥션 풀(DBCP)

- ◆ 클라이언트에서 다수의 요청이 발생할 경우 데이터베이스에 부하가 발생합니다. 이러한 문제를 해결하기 위해서 커넥션 풀(Database Connection Pool) 기법을 이용합니다.
(jsp_20_3_ex1_cpex)



20-3. 커넥션 풀(DBCP)

- ◆ Tomcat컨테이너가 데이터베이스 인증을 하도록 context.xml 파일을 열어 아래의 코드를 추가 합니다.

context.xml ×

```
29 |
30 | <Resource
31 |     auth = "Container"
32 |     driverClassName = "oracle.jdbc.driver.OracleDriver"
33 |     url = "jdbc:oracle:thin:@tlf.co.kr:1521:orcl"
34 |     username = "kopo"
35 |     password = "kopo"
36 |     name = "jdbc/Oracle11g"
37 |     type = "javax.sql.DataSource"
38 |     maxActive = "50"
39 |     MaxWait = "1000"
40 | </>
41 | </Context>
42 |
```

Tomcat v10.1 Server at localhost [Stopped, Republish]
jsp_20_3_ex1_dbcpex [Synchronized]

Tomcat v10.1 Server at localhost [Started, Synchronized]
jsp_20_3_ex1_dbcpex [Synchronized]

Microsoft Standard Proposal

Microsoft PowerPoint 2007

21강. 회원 인증 프로그래밍

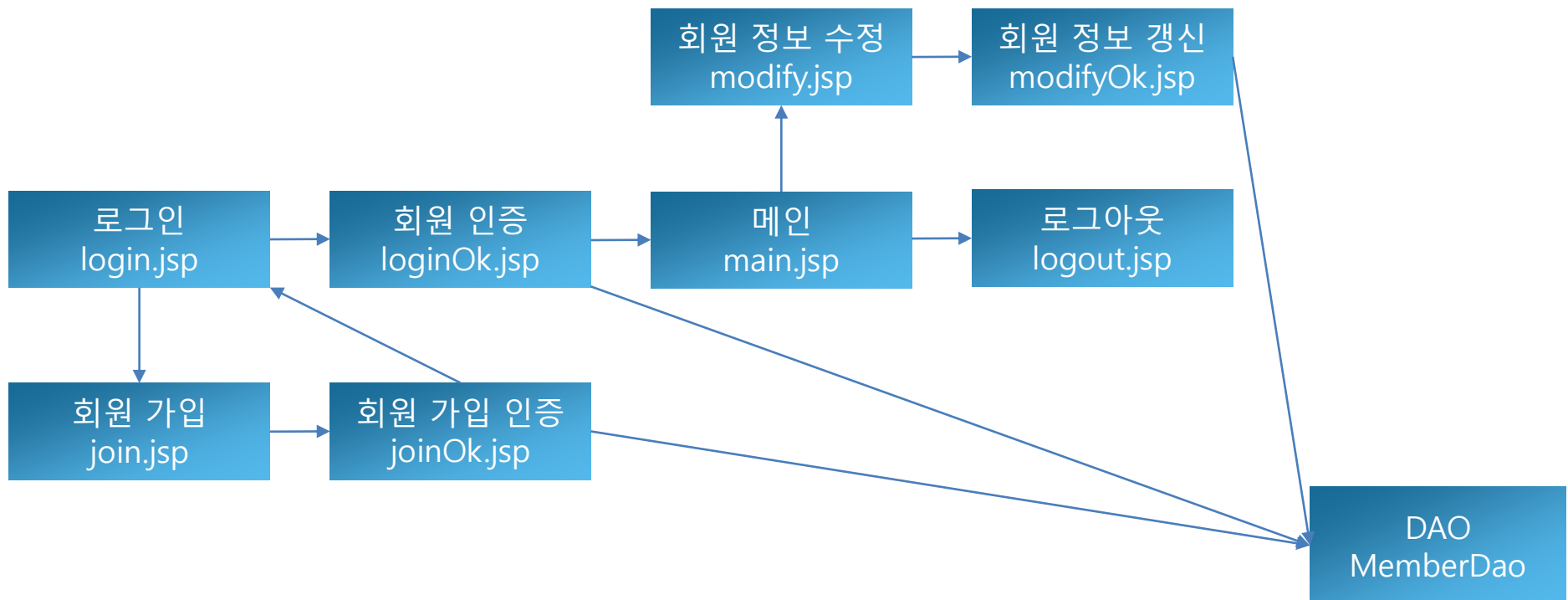
- 회원 인증 프로그래밍



21-1. 회원 인증 프로그래밍

- ◆ 지금까지 학습한 JSP 및 Servlet 관련 기능들을 가지고 회원 인증 프로그램을 만들어 봅니다.
(jsp_21_1_ex1_memberex)

전체적인 흐름도



23강. EL(Expression Language)

- EL(Expression Language)?
- 액션태그로 사용되는 EL
- 내장객체



23-1. EL(Expression Language)

- ◆ EL(Expression Language)란 표현식 또는 액션 태그를 대신해서 값을 표현하는 언어입니다.
(JSP_23_1_ex1_elex)

<%= value %>

표현식

\${ value }

EL

EL연산자

산술 : +, -, /, %
관계형 : ==, !=, <, >, <=, >=
조건 : a? b : c
논리 : &&, ||

23-2. 액션태그로 사용되는 EL

(jsp_23_2_ex1_elex)

```
<jsp:getProperty name="member" property="name" />
```



```
${ member.name }
```

23-3. 내장객체

(jsp_23_3_ex1_elex)

pageScope : page 객체를 참조하는 객체
requestScope : request객체를 참조하는 객체
sessionScope : session객체를 참조하는 객체
applicationScope : applicatio객체를 참조하는 객체

param : 요청 파라미터를 참조하는 객체
paramValues : 요청 파라미터(배열)를 참조하는 객체
initParam : 초기화 파라미터를 참조하는 객체
cookie : cookie객체를 참조하는 객체

24강. JSTL(JSP Standard Tag Library)

- JSTL 개요 및 설치
- JSTL 라이브러리



24-1. JSTL 개요 및 설치

- ◆ JSP의 경우 HTML 태그와 같이 사용되어 전체적인 코드의 가독성이 떨어집니다. 이러한 단점을 보완하고자 만들어진 태그 라이브러리가 JSTL입니다. JSTL은 Tomcat 컨테이너에 포함되어 있지 않으므로, 별도의 설치를 하고 사용합니다.

JSTL 다운로드 URL

```
https:// tomcat.apache.org/taglibs/standard/
```

JSTL 다운로드 파일

```
taglibs-standard-Impl-1.2.5.jar  
Taglibs-standard-jstlel-1.2.5.jar  
Taglibs-standard-spec-1.2.5.jar
```

JSTL 다운로드 설치

```
SRC/WEB-INF/lib 폴더에 다운로드 파일을 복사한다.
```

JSTL 사용법

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

24-2. JSTL 라이브러리

JSTL에서는 다섯 가지의 라이브러리를 제공합니다.(Core, XML, i18N formatting, SQL, Functions)

lib	Prefix	ex
Core태그	c	<c:tag
Formatting태그	x	<x:tag
SQL태그	sql	<sql:tag
Functions태그	fn	<fn:function()

Core

Core라이브러리는 기본 라이브러리로 출력, 제어문, 반복문 등 처리 변수 선언, 삭제 등 변수와 관련된 작업 및 if 문, for문과 같은 제어 기능, URL처리로 페이지 이동 기능을 제공합니다.

(jsp_24_2_ex1_jstlex)

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

24-2. JSTL 라이브러리

출력 태그 : <c:out>

```
<c:out value="출력값" default="기본값" escapeXml="true or false" >
```

변수 설정 태그 : <c:set>

```
<c:set var="변수명" value="설정값" target="객체" property="값" scope="범위" >
```

변수를 제거하는 태그 : <c:remove>

```
<c:remove var="변수명" scope="범위" >
```

예외처리 태그 : <c:catch>

```
<c:catch var="변수명">
```

24-2. JSTL 라이브러리

제어문(if) 태그 : <c:if>

```
<c:if test="조건" var="조건 처리 변수명" scope="범위">
```

제어문(switch) 태그 : <c:choose>

```
<c:choose>  
<c:when test="조건"> 처리 내용 </c:when>  
<c:otherwise> 처리 내용 </c:otherwise>  
</c:choose>
```

반복문(for) 태그 : <c:forEach>

```
<c:forEach items="객체명" begin="시작 인덱스" end="끝 인덱스" step="증감식" var="변  
수명" varStatus="상태변수">
```


24-2. JSTL 라이브러리

페이지 이동 태그 : <c:redirect>

```
<c:redirect url="url">
```

파라미터 전달 태그 : <c:param>

```
<c:param name="파라미터명" value="값">
```

UML Standard Proposal

UML Standard Proposal

25강. FrontController 패턴과 Command 패턴

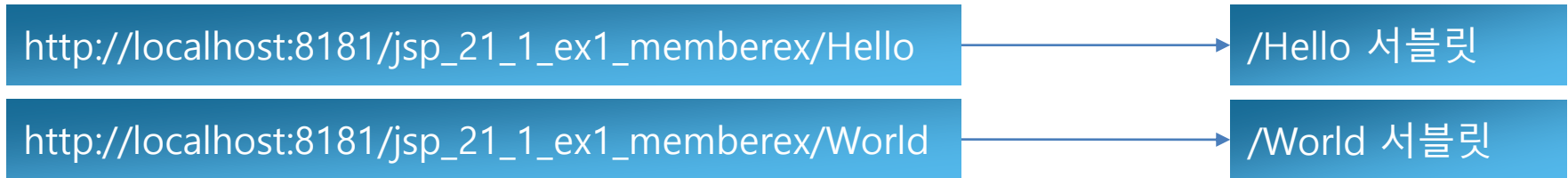
- url-pattern
- FrontController 패턴
- Command패턴



25-1. url-patten

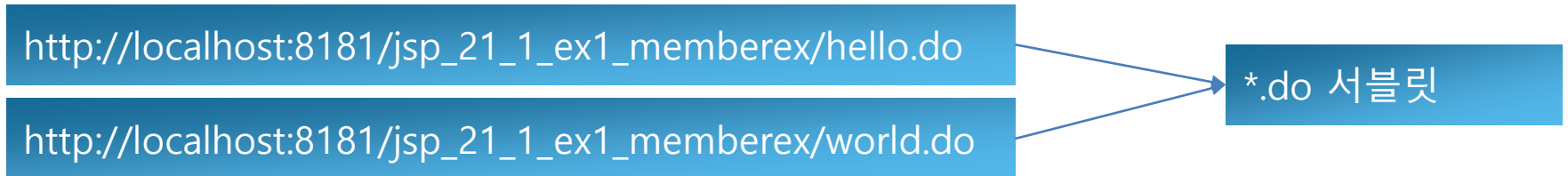
디렉터리 패턴

디렉터리 형태로 서버의 해당 컴포넌트를 찾아서 실행하는 구조



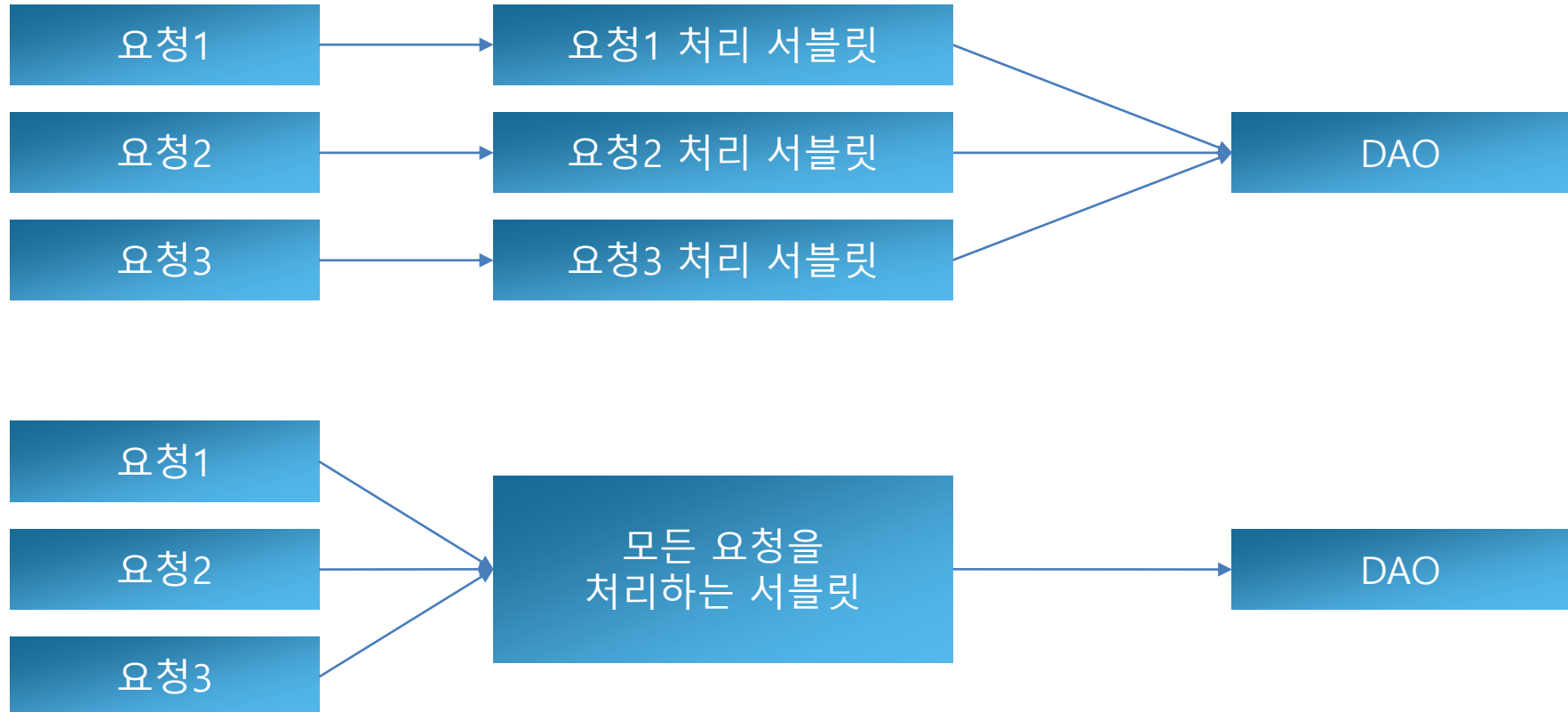
확장자 패턴

확장자 형태로 서버의 해당 컴포넌트를 찾아서 실행하는 구조



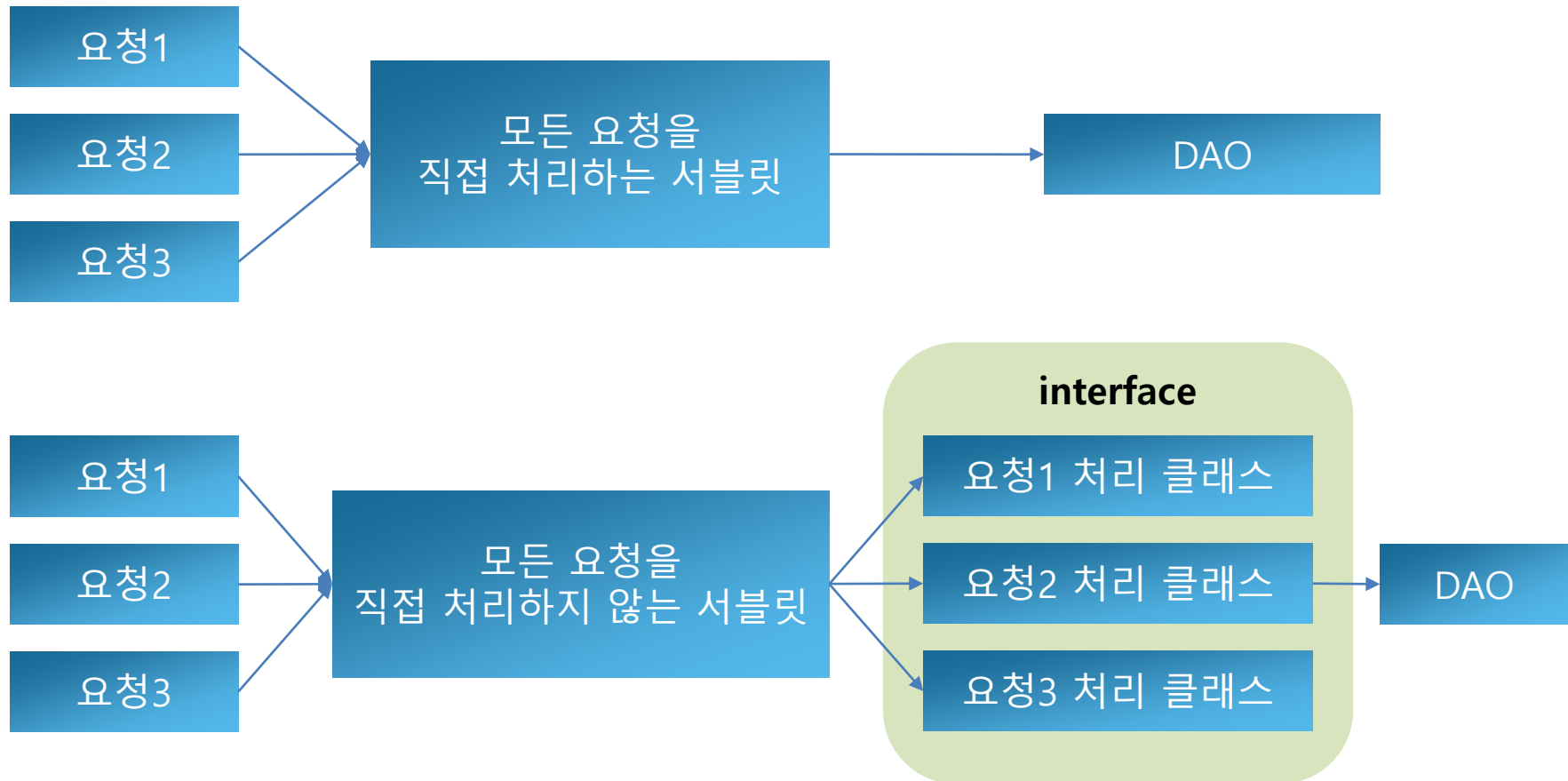
25-2. FrontController 패턴

클라이언트의 다양한 요청을 한 곳으로 집중시켜, 개발 및 유지보수에 효율성을 극대화 합니다.
(jsp_25_2_ex1_frontex)



25-3. Command 패턴

클라이언트로부터 받은 요청들에 대해서, 서블릿이 작업을 직접 처리하지 않고, 해당 클래스가 처리토록 합니다.
(jsp_25_3_ex1_commex)



the Standard Proposal

the Standard Proposal

26강. 포워딩(Forwarding)

- RequestDispatcher 클래스
- HttpServletResponse 클래스



26-1. RequestDispatcher 클래스

- ◆ 서블릿 또는 JSP에서 요청을 받은 후 다른 Component로 요청을 위임할 수 있습니다. 그리고 이러한 위임 방법에는 2개의 클래스를 이용합니다. 하나는 RequestDispatcher 클래스이고, 또 하나는 HttpServletResponse 클래스입니다.
- ◆ RequestDispatcher 클래스의 경우 요청 받은 요청객체(request)를 위임하는 컴포넌트에 동일하게 전달할 수 있습니다.
(jsp_26_1_ex1_forwardingex)

RequestDispatcher 클래스

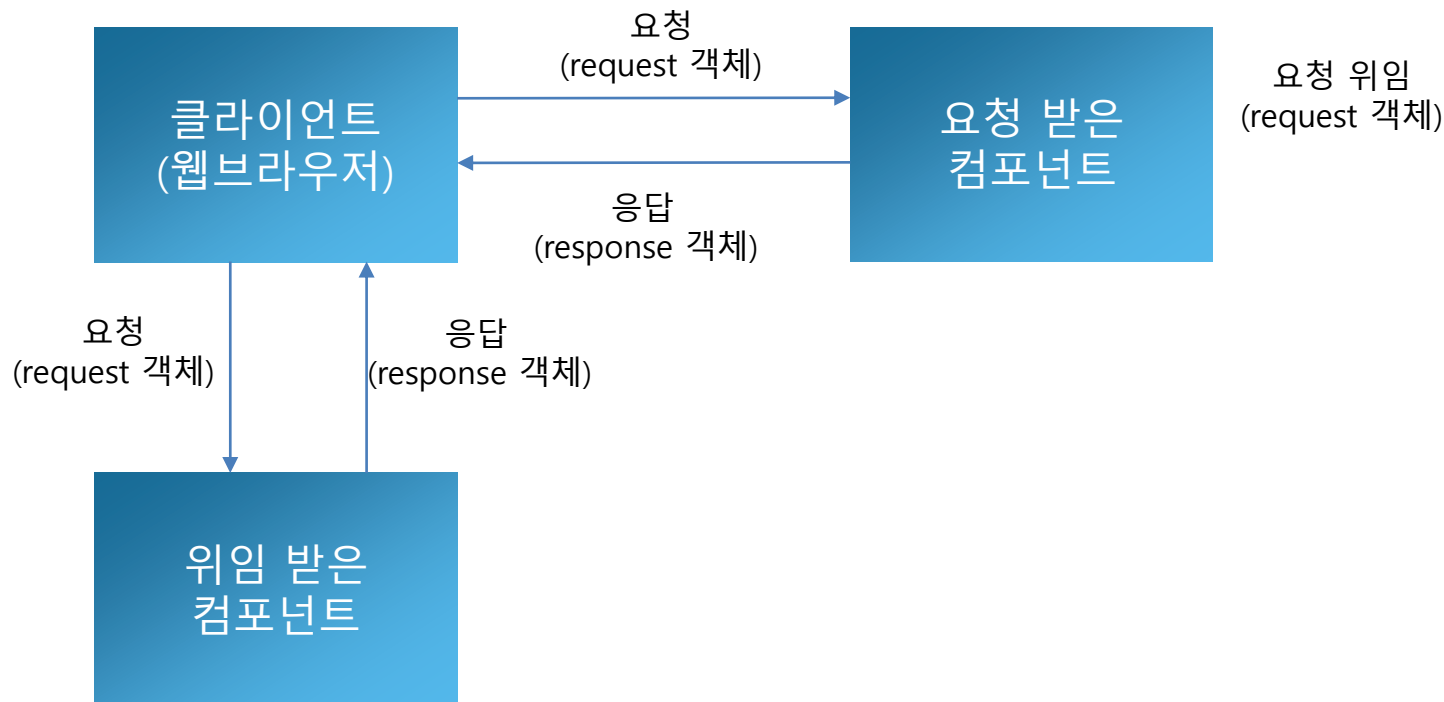


26-2. HttpServletResponse 클래스

- ◆ RequestDispatcher 클래스와 동일하게 요청을 위임하는 클래스입니다.
- ◆ RequestDispatcher 클래스와 차이점은 요청 받은 요청 객체를 위임 받은 컴포넌트에 전달하는 것이 아닌 새로운 요청 객체를 생성합니다.

(jsp_26_2_ex1_forwardingex)

HttpServletResponse 클래스



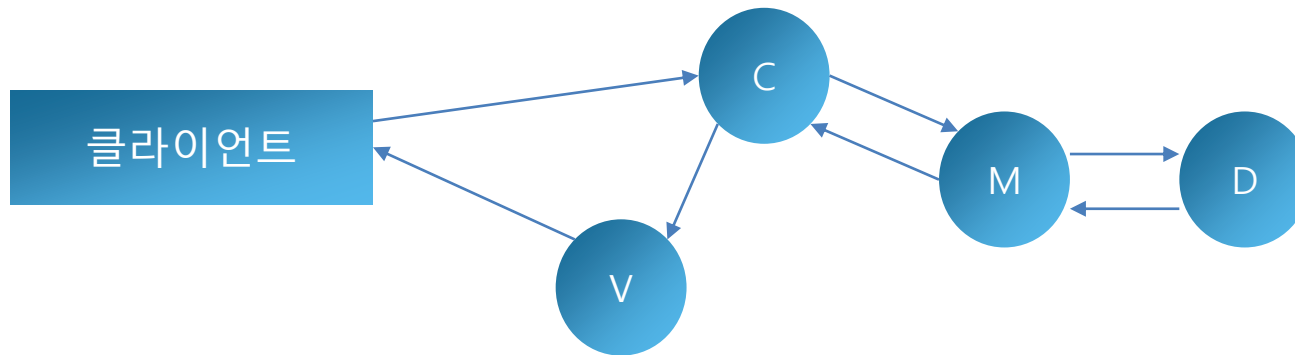
27강. MVC패턴을 이용한 게시판 만들기-I

- MVC패턴의 이해
- 전체적인 컴포넌트 설계
- DB(DataBase) 생성



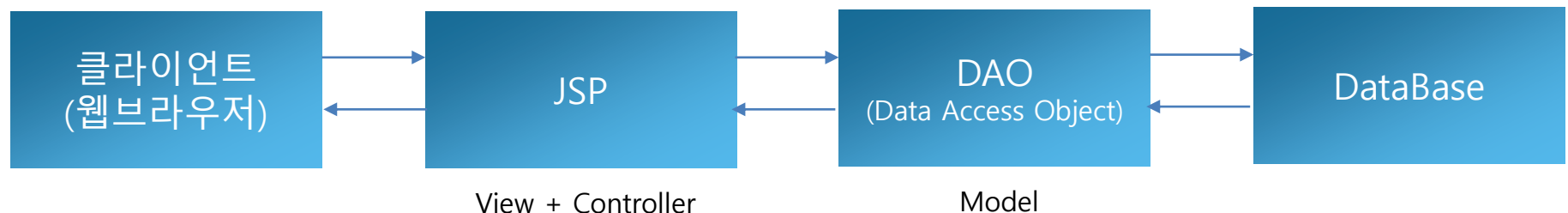
27-1. MVC패턴의 이해

- ◆ MVC란 Model, View, Controller를 뜻하는 용어로 개발 형태의 일종입니다.
- ◆ Model은 데이터베이스와의 관계를 담당합니다. 클라이언트의 요청에서 필요한 자료를 데이터베이스로부터 추출하거나 수정하여 Controller로 전달합니다.
- ◆ View는 사용자한테 보여지는 UI화면입니다. 주로 .jsp파일로 작성하며, Controller에서 어떤 View컴포넌트로 보여줄지 결정합니다.
- ◆ Controller는 클라이언트의 요청을 받고 적절한 Model에 지시를 내이며, Model에서 전달된 데이터를 적절한 View에 전달합니다.
- ◆ 이렇게 작업을 분할하면 추후 유지보수에 효율적입니다.



Model1

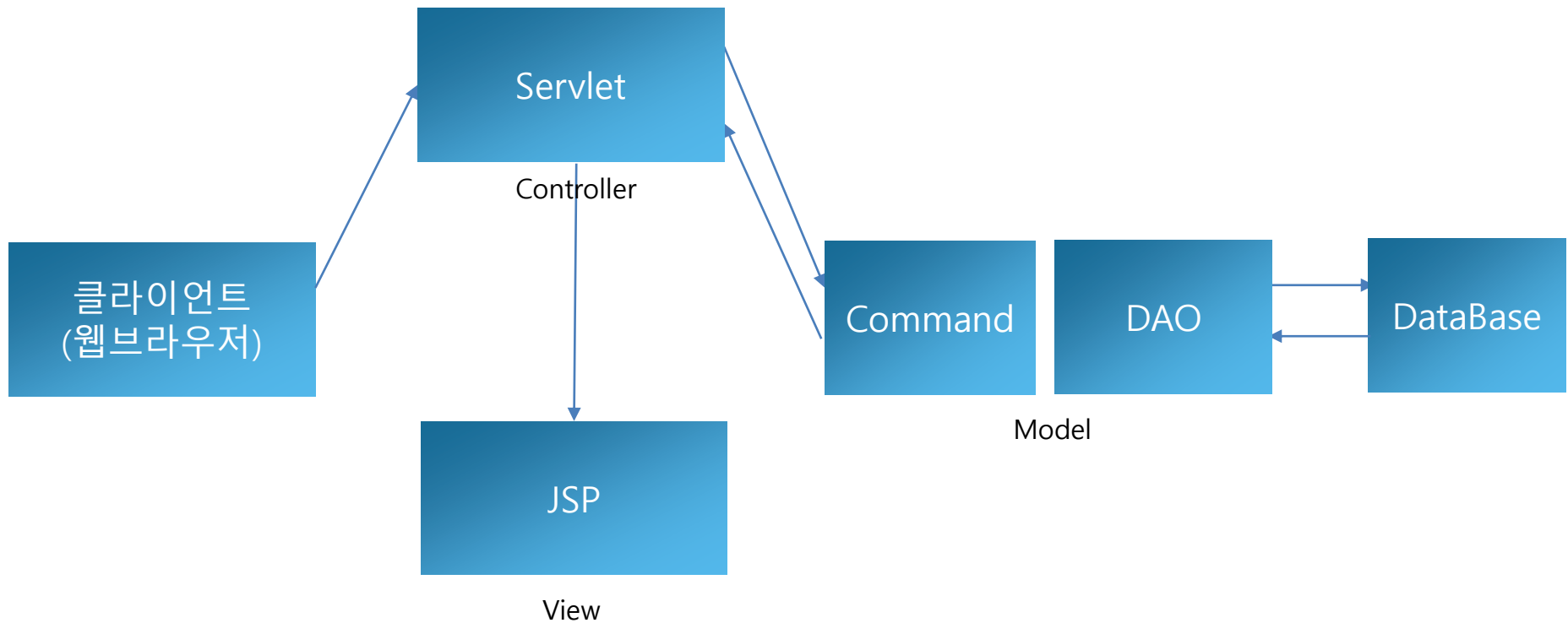
- ◆ MVC에서 View와 Controller가 같이 있는 형태입니다.



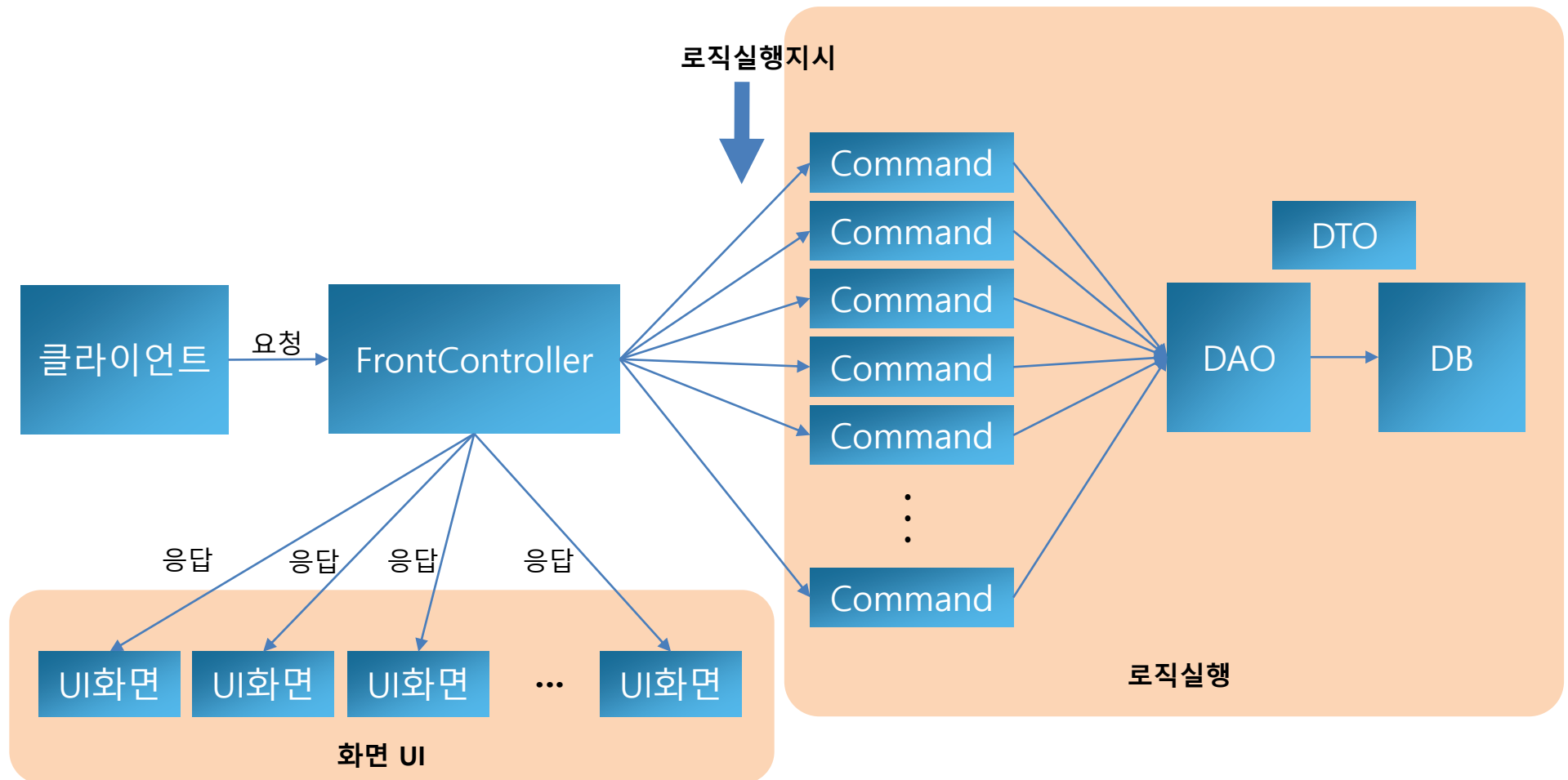
27-1. MVC패턴의 이해

Model2

- ◆ MVC에서 Model, View 그리고 Controller가 모두 모듈화 되어 있는 형태입니다.



27-2. 전체적인 컴포넌트 설계



27-3. DataBase 생성

게시판 테이블 생성

- ◆ 테이블 명 : mvc_board
- ◆ 컬럼 구성 : 식별자, 작성자 명, 제목, 내용, 작성일자, 조회 수, 분류, 레벨, 들여쓰기
- ◆ 시퀀스 : seq_mvc_board

28강. MVC패턴을 이용한 게시판 만들기-II

- FrontController 만들기
- Command 만들기
- DTO(Data Transfer Object) 만들기
- DAO(Data Access Object) 만들기
- View 페이지 만들기



28-1. FrontController 만들기

- ◆ 클라이언트의 요청을 받는 역할을 하는 Controller를 만들어 봅니다.(MBFrontController)
(jsp_27_1_ex1_mvcboardex)
- ◆ 패키지 : com.javalec.ex.frontcontroller
- ◆ 클래스 : MBFrontController

```
private void actionPerformed( HttpServletRequest request, HttpServletResponse response ) throws ServletException, IOException {
    System.out.println("actionDo");

    request.setCharacterEncoding("UTF-8");

    String viewPage = null;
    BCommand cmd = null;

    String uri = request.getRequestURI();
    String conPath = request.getContextPath();
    String com = uri.substring(conPath.length());

    if( com.equals("/write_view.do") ) {
        viewPage = "write_view.jsp";
    } else if(com.equals("/write.do")) {
        cmd = new BWriteCommand();
        cmd.execute(request, response);
        viewPage = "list.do";
    }

    RequestDispatcher reqDpt = request.getRequestDispatcher(viewPage);
    reqDpt.forward(request, response);
}
```

← 웹브라우저에서 글쓰기 UI화면입니다.

← 'write.do' 요청이 들어오면 해당 Command를 생성하여 적절한 로직을 실행 후 'list.do' 페이지로 포워딩 합니다.

28-2. Command 만들기

- ◆ Command 인터페이스를 이용해서 Command 클래스들을 만듭니다.
- ◆ 패키지 : kr.co.tlf.ex.command
- ◆ Command 인터페이스 : Bcommand
- ◆ Write Command 클래스 : BWriteCommand

```

MBCommand.java ×
1 package kr.co.tlf.ex.command;
2
3 import jakarta.servlet.http.HttpServletRequest;
4 import jakarta.servlet.http.HttpServletResponse;
5
6 public interface MBCommand {
7     void execute(HttpServletRequest request, HttpServletResponse response);
8 }
9

```

```

@Override
public void execute(HttpServletRequest request, HttpServletResponse response) {
    // TODO Auto-generated method stub

    String nbMvcBoard = request.getParameter("nbMvcBoard");
    MBDao dao = new MBDao();
    MBDto dto = dao.getContentView(nbMvcBoard);

    request.setAttribute("content_view", dto);
}

```

데이터 베이스와 연결하여 사용자가 지정한 글의 내용을 조회 합니다.

28-3. DTO(Data Transfer Object) 만들기

- ◆ 데이터 베이스의 데이터 DTO 객체를 만듭니다.
- ◆ 패키지 : kr.co.tlf.ex.dto
- ◆ DTO 클래스 : MBDto

```
public class MBDto {  
    private int nbMvcBoard;  
    private String nmName;  
    private String nmTitle;  
    private String nmContent;  
    private Timestamp daWrite;  
    private int cnHit;  
    private int nbGroup;  
    private int nbStep;  
    private int nbIndent;  
  
    public int getNbMvcBoard() {  
        return nbMvcBoard;  
    }  
    public void setNbMvcBoard(int nbMvcBoard) {  
        this.nbMvcBoard = nbMvcBoard;  
    }  
    public String getNmName() {  
        return nmName;  
    }  
    public void setNmName(String nmName) {  
        this.nmName = nmName;  
    }  
    public String getNmTitle() {  
        return nmTitle;  
    }  
}
```

← 생성자 및 속성 설정

← Getter, Setter

28-4. DAO(Data Access Object) 만들기

- ◆ 데이터 베이스에 연결하여 필요한 로직을 수행하는 DAO클래스를 만듭니다.
- ◆ 패키지 : kr.co.tlf.ex.dao
- ◆ DTO 클래스 : MBDao

```
public MBDao() {
    try {
        Context ctx = new InitialContext();
        ds = (DataSource)ctx.lookup("java:comp/env/jdbc/Oracle11g");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

← 생성자에서 DBCP를 만듭니다.

```
public int write(MBDto dto) {
    int ri = 0;

    Connection conn = null;
    PreparedStatement pstmt = null;
    String query = "insert into mvc_board values(seq_mvc_board.nextval, ?, ?, ?, sysdate, ?, ?, ?, ?)";
```

```
    try {
        conn = ds.getConnection();
        pstmt = conn.prepareStatement(query);

        pstmt.setString(1, dto.getNmName());
        pstmt.setString(2, dto.getNmTitle());
        pstmt.setString(3, dto.getNmContent());
        pstmt.setInt(4, 1);
        ...
    }
```

← DBCP로부터 Connection을 얻고, 데이터 베이스와 관련한 필요한 작업을 시작 합니다.

28-5. View 페이지 만들기

- ◆ 클라이언트의 요청에 대해서 FrontController에서 작업을 분기하고, 해당 Command클래스가 작동하여 DAO를 이용한 데이터 베이스 작업을 합니다. DAO클래스의 결과물로 DTO객체가 View(jsp페이지)로 전달되며, View에서는 클라이언트의 요청에 대한 응답으로 화면(UI)를 구성하여 출력합니다.
- ◆ View 파일 : list.jsp

```
<table width="500" cellpadding="0" cellspacing="0" border="1">
  <tr>
    <td>번호</td>
    <td>이름</td>
    <td>제목</td>
    <td>날짜</td>
    <td>조회수</td>
  </tr>
  <c:forEach items="${list}" var="dto">
    <tr>
      <td>${dto.nbMvcBoard}</td>
      <td>${dto.nmName}</td>
      <td>${dto.nmTitle}</td>
      <td>${dto.daWrite}</td>
      <td>${dto.cnHit}</td>
    </tr>
  </c:forEach>
</table>
```



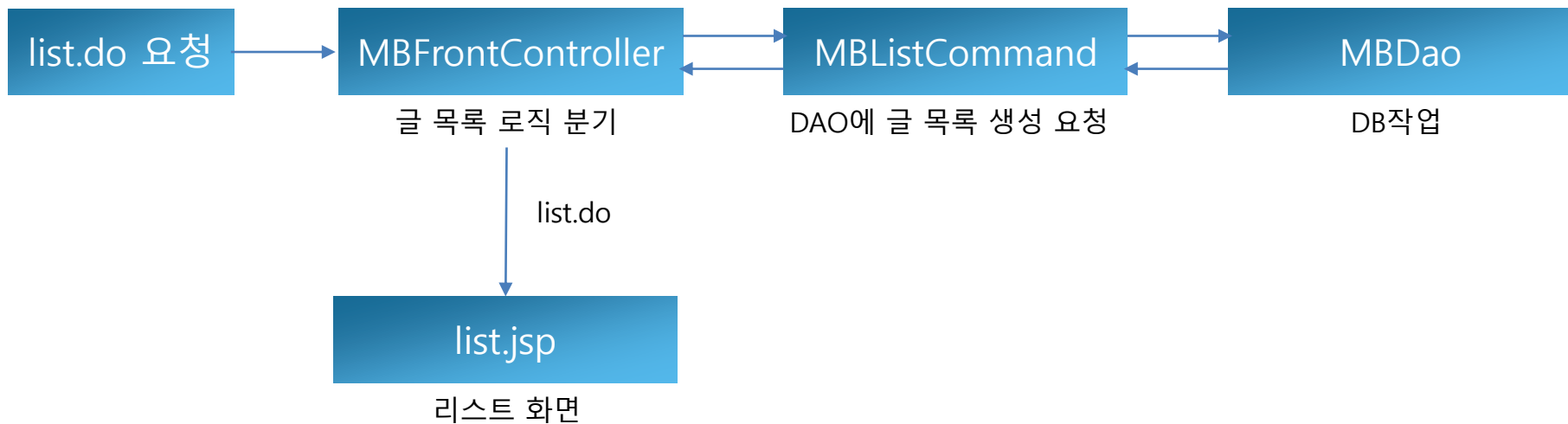
29강. MVC패턴을 이용한 게시판 만들기-III

- 글 목록 페이지 만들기
- 글 내용 보기 페이지 만들기
- 글 내용 수정 페이지 만들기
- 글 삭제 페이지 만들기

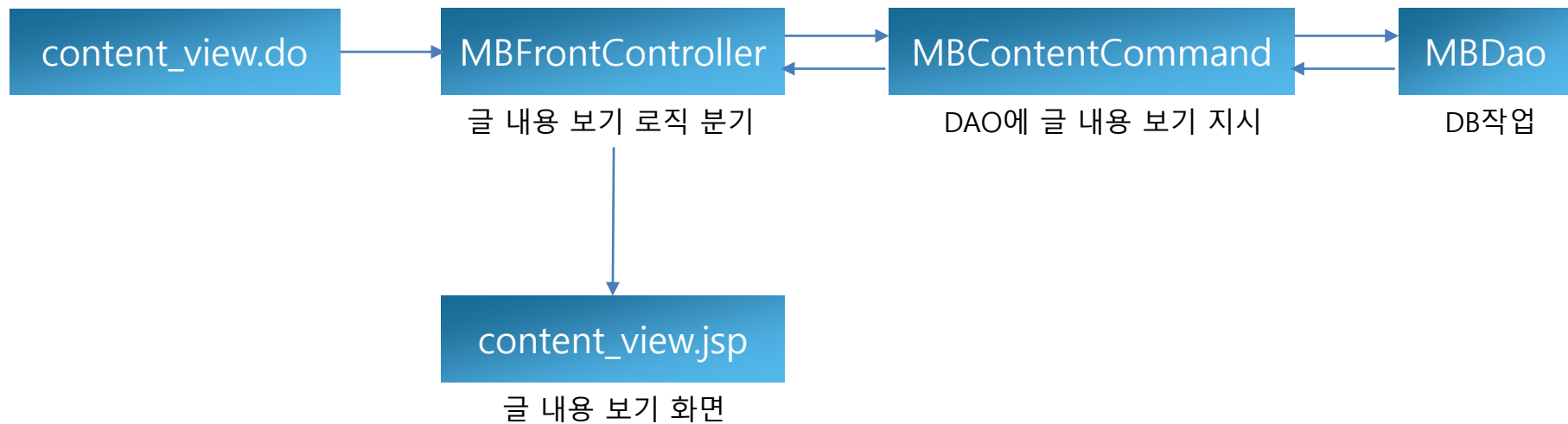


29-1. 글 목록 페이지 만들기

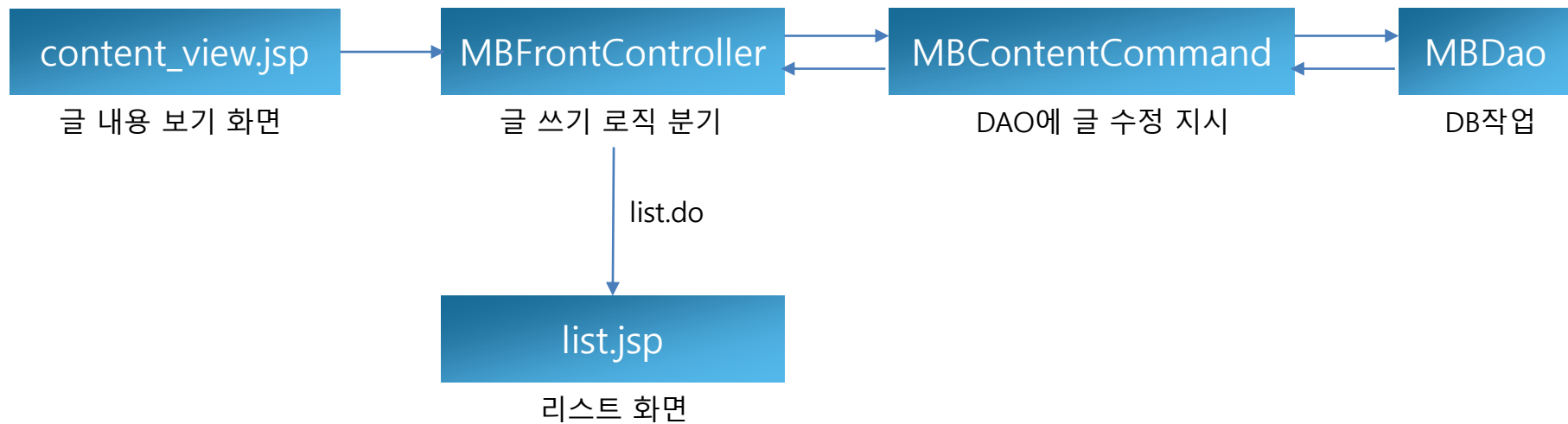
- ◆ 28강에 이어서 여러분 각자가 만든 소스에 추가하여 작업합니다.
(jsp_27_1_ex1_mvcboardex)



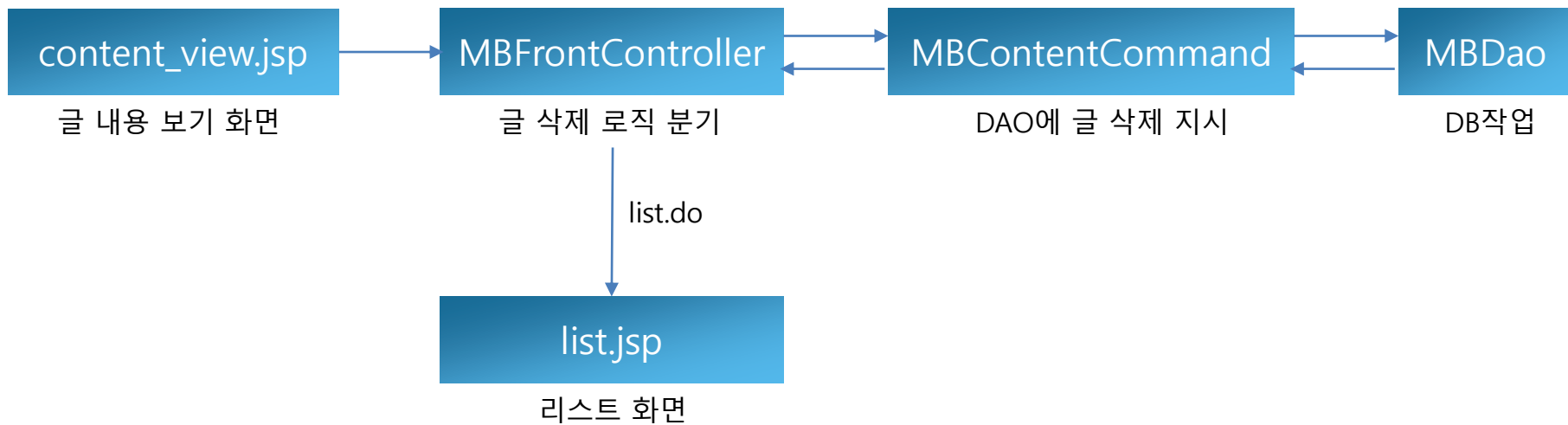
29-2. 글 내용 보기 페이지 만들기



29-3. 글 내용 수정 페이지 만들기



29-4. 글 삭제 페이지 만들기



Microsoft Standard Proposal

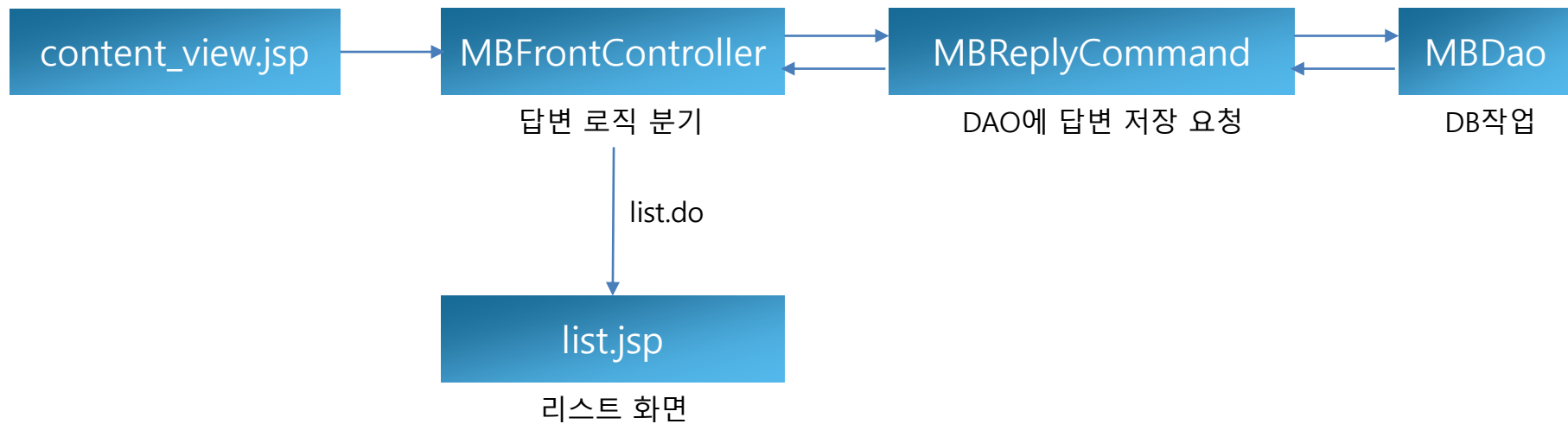
Microsoft PowerPoint 2003

30강. MVC패턴을 이용한 게시판 만들기-IV

- 글 답변 달기



30-1. 글 답변 달기



감사합니다.

