KB금융그룹 | 국민의 평생 금융파트너

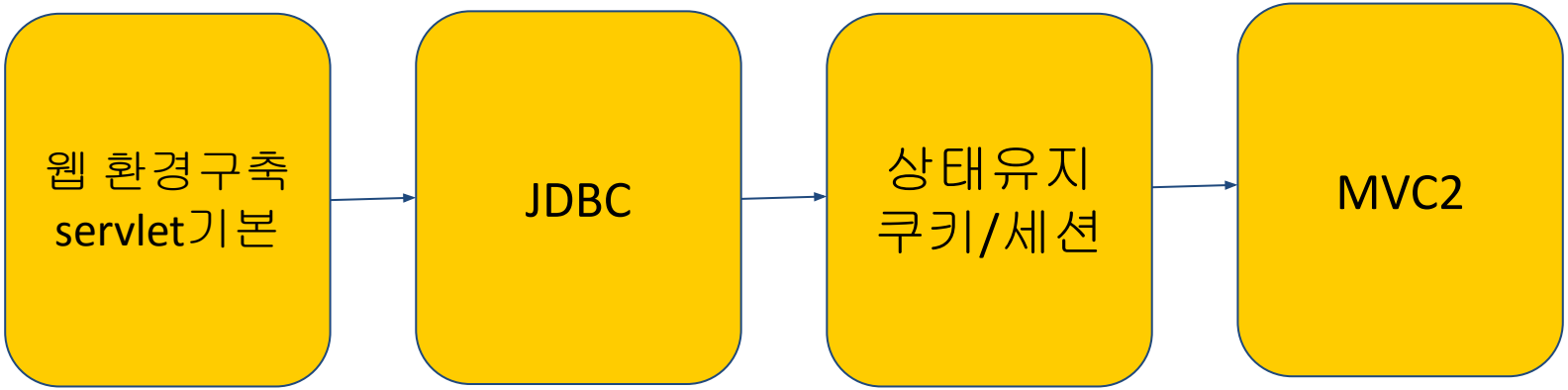
2024년 상반기 K-디지털 트레이닝

JSP/Servlet MVC

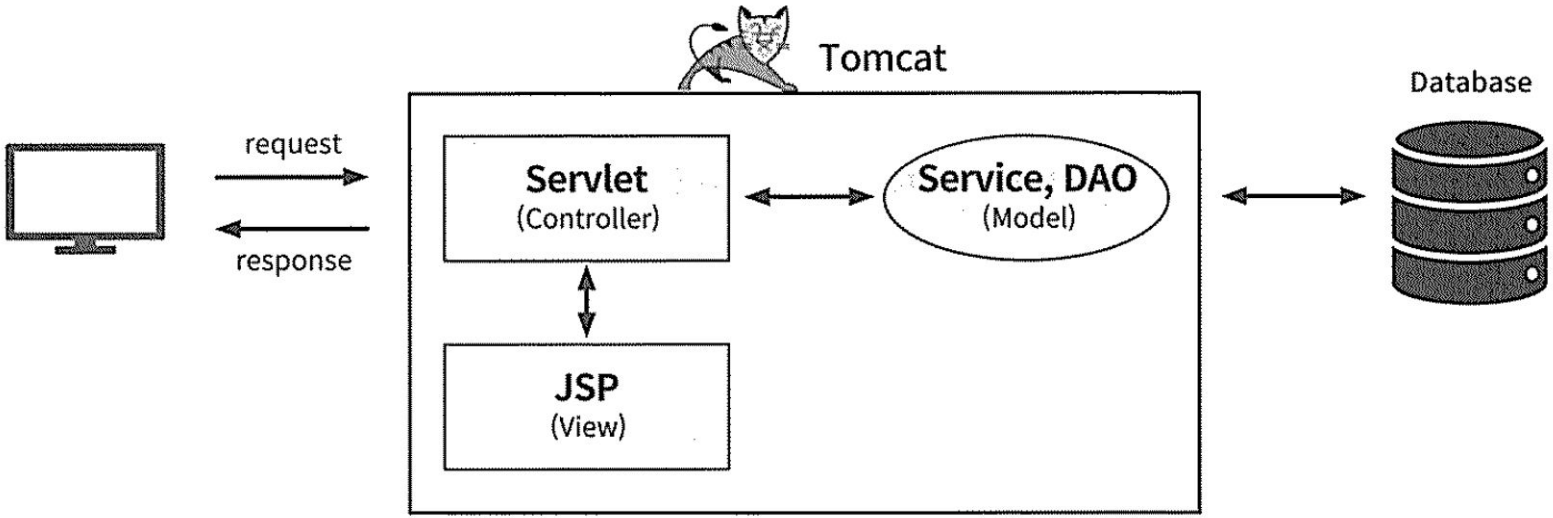
[KB] IT's Your Life

KB 국민은행

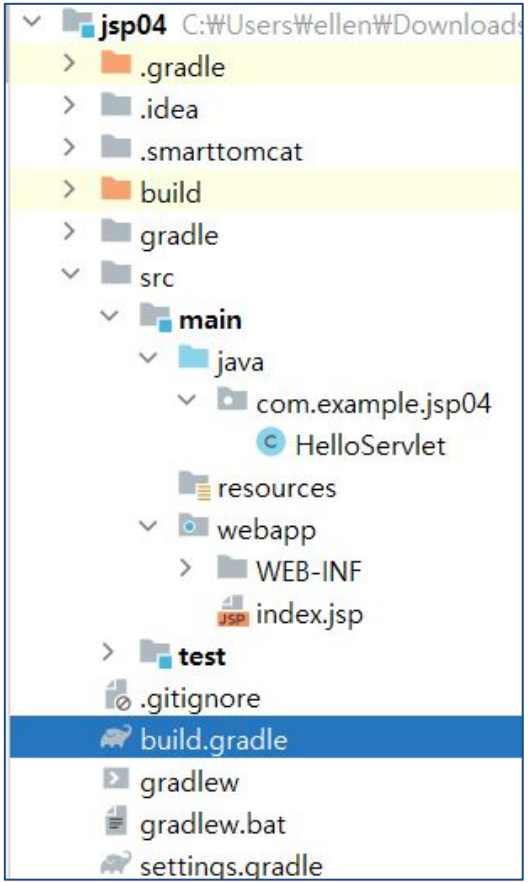
커리큘럼



학습 목표


Tomcat
Database
request
response
Servlet
(Controller)
Service, DAO
(Model)
JSP
(View)

프로젝트 설정




```
sourceCompatibility = '17'
targetCompatibility = '17'

tasks.withType(JavaCompile) {
 options.encoding = 'UTF-8'
}

dependencies {
    // JSP, SERVLET, JSTL
    implementation('javax.servlet:javax.servlet-api:4.0.1')
    compileOnly 'javax.servlet.jsp:jsp-api:2.1'
    implementation 'javax.servlet:jstl:1.2'

 //lombok plugin
 implementation 'org.projectlombok:lombok:1.18.26'  // 최신 버전 사용
 annotationProcessor 'org.projectlombok:lombok:1.18.26'  // 최신 버전 사용

 //driver
 implementation 'mysql:mysql-connector-java:8.0.30'  // MySQL 드라이버 예제


 testImplementation("org.junit.jupiter:junit-jupiter-api:${junitVersion}")
 testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:${junitVersion}")
}
```

JSP

서블릿의 단점

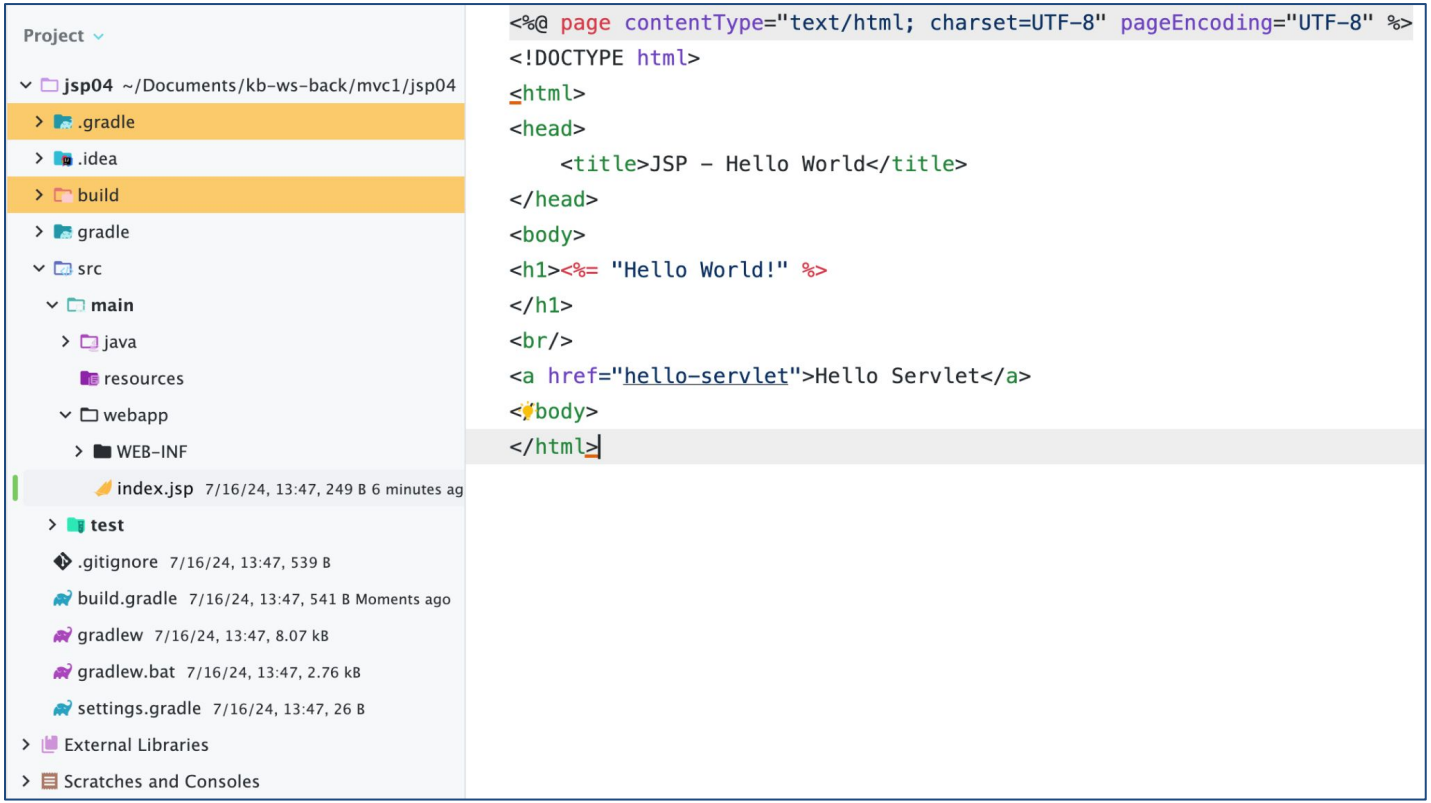
- 정해진 규칙을 지켜서 작성해야 한다.

  (import, public 클래스, HttpServlet 상속, 기본 생성자, 콜백 메소드 오버라이딩)

- web.xml 파일이나 어노테이션을 통해 요청 URL에 대한 매핑을 설정해야 한다.
- 소스를 수정한 후에는 반드시 재컴파일을 하고 리로딩될 때까지 기다려야 한다.

```
protected void service(HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException {
    // 1. 사용자 입력 정보 추출
    String id = request.getParameter("id");

    // 2. 응답 화면 구성
    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();

    out.println("<html>");
    out.println("<head>");
    out.println("<meta http-equiv='Content-Type' content='text/html;
    out.println("<title>Hello Servlet</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<center>");
    out.println("<h1>" + id + "님 환영합니다.</h1>");
    out.println("</center>");
    out.println("</body>");
    out.println("</html>");

    out.close();
}
```

- **jsp(HTML + Java) → Java Servlet으로 자동 변환**



JSP 주요 문법

태그	종류	예시	설명
<%-- --%>	주석(Comment)	<%-- 이것은 주석입니다 --%>	코드에 대한 설명을 추가
<%@ %>	지시자(Directive)	<%@ page language="java" %>	페이지 전반에 걸쳐 적용되는 설정을 지정
<% %>	스크립틀릿(Scriptlet)	<% System.out.println("안녕"); %>	서버 측에서 실행되는 코드 블록을 포함
<%= %>	표현식(Expression)	<%= "안녕하세요" %>	값을 계산하고 그 결과를 출력하는데 사용
<%! %>	선언문(Declaration)	<%! int a = 0; %>	변수나 메서드를 선언

JSP

```
<%= test %>
```

servlet

```
out.println(test);
```

scriptlet → servlet service()로 변환

- 지역변수: 반드시 초기화

```jsp
<jsp:useBean id="bag" class="bean.BbsDTO"></jsp:useBean>
<jsp:setProperty property="id" name="bag"/>
<%
BbsDAO dao = new BbsDAO();
BbsDTO dto = dao.one(bag);
%>
```

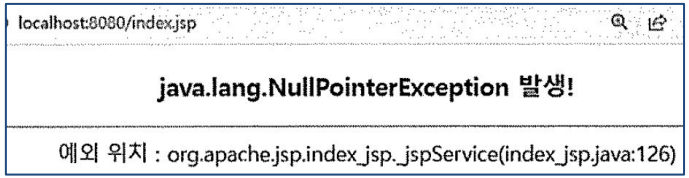
```
Servlet{
        public void service(request, response){
                BbsDTO bag = new BbsDTO();
                 bag.setId(request.getParameter("id"));
                 ——----------------------
        }
```

지시자

지시어	태그	설명	예시 사용
page	<%@ page %>	페이지 속성 설정, 서블릿 환경 구성	인코딩 설정, 오류 페이지 지정
include	<%@ include %>	다른 페이지 또는 컴포넌트 포함	공통 헤더, 푸터 포함
taglib	<%@ taglib %>	사용자 정의 태그 라이브러리 사용	커스텀 태그 사용, 데이터 표시

지시자

- **page 지시자 (<%@ page %>):**
  - **isErrorPage**
    - 현재 페이지가 에러로 작동되는 페이지인지 확인
  - **errorPage**
    - 에러가 발생했을 때 선보일 JSP 페이지를 지정
    - 오류 페이지 지정
  - **import**
    - 번들로서 서블릿에서 import할 클래스를 선언
  - **session**
    - 현재 페이지에서 세션 객체를 사용할 것인지 설정
    - default session = true




```jsp
<%@ page isErrorPage="true" %>
<html>
<head><title>오류 페이지</title></head>
<body>
  <h1>오류 발생!</h1>
  <p><%= exception.getMessage() %></p>
</body>
</html>
```

```jsp
<%@ page errorPage="errorPage.jsp" %>
<html>
<head><title>메인 페이지</title></head>
<body>
  <% if (1 == 2) { %>  <!-- 의도적으로 오류를 발생시키는 조건 -->
    <p>이 조건은 참입니다.</p>
  <% } else {
    throw new Exception("에러 발생: 조건이 거짓입니다.");
  } %>
</body>
</html>
```

JSTL (JavaServer Pages Standard Tag Library)

- **taglib 지시자 사용**
- **JSTL은 조건문, 반복문, 포매팅, 데이터베이스 접근 등을 위한 태그를 제공하여 JSP 페이지의 Java 코드 사용을 최소화하고, 보다 효율적으로 작성**
- **build.gradle**
  - implementation group: 'javax.servlet', name: 'jstl', v
- **Core 태그 라이브러리 :**
  - URI: http://java.sun.com/jsp/jstl/core
  - 접두어 (권장): c
  - 기능: 변수 선언, 조건문, 반복문, URL 처리 등
- **Formatting 태그 라이브러리**
  - URI: http://java.sun.com/jsp/jstl/fmt
  - 접두어: fmt
  - 기능: 날짜, 시간, 숫자 포매팅 및 국제화

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
   <title>JSTL Example</title>
</head>
<body>
   <h2>조건문 예제</h2>
   <c:if test="${5 > 2}">
      <p>5는 2보다 큽니다.</p>
   </c:if>

   <h2>반복문 예제</h2>
   <ul>
      <c:forEach var="i" begin="1" end="5">
         <li>번호 ${i}</li>
      </c:forEach>
   </ul>
</body>
</html>
```

- **formatting**

– 날짜 formatting

```
<%@ page import="java.util.Date" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

<%
    Date now = new Date();
    request.setAttribute("now", now);
%>

<fmt:formatDate value="${now}" pattern="yyyy-MM-dd HH:mm:ss" />
```

– 숫자 formatting

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<fmt:formatNumber value="${price}" type="currency" />
```

EL(Expression Language)

- JSP 페이지 내에서 Java 코드를 사용하지 않고도 간단한 표현식을 사용하여 데이터를 처리하고 출력하는 기능을 제공
- EL은 주로 속성값의 처리, 조건부 로직의 간소화 및 JSP 페이지 내에서의 데이터 표현을 목적
  - 변수 접근: `${variableName}` 구문을 사용하여 변수에 접근
  - 속성 접근: .(점) 또는 [](대괄호)를 사용하여 객체의 속성이나 맵의 키에 접근 예: `${user.name}` 또는 `${user["name"]}`
  - 컬렉션 접근: 배열이나 리스트의 요소에는 인덱스를, 맵의 경우 키를 사용하여 접근 예: `${numbers[0]}` 또는 `${map["key"]}`
  - 조건 표현: `? :` 조건 연산자를 사용하여 간단한 조건을 처리. 예: `${empty user ? "Guest" : user.name}`
- 내장 객체
  - `pageScope` : 페이지 스코프에 저장된 속성에 접근
  - `requestScope` : 요청 스코프에 저장된 속성에 접근
  - `sessionScope` : 세션 스코프에 저장된 속성에 접근
  - `applicationScope` : 애플리케이션 스코프에 저장된 속성에 접근
  - 내장 객체 이름 생략하는 경우, 내장 객체 전체 설정 중 이름으로 찾음.

```
<%@ page contentType="text/html;charset=UTF-8"
language="java" %>
<html>
<head>
    <title>EL Example</title>
</head>
<body>
    <h2>Welcome ${user.name}!</h2>
    <p>Your role is: ${sessionScope.userRole}</p>
    <p>Today's date: ${applicationScope.todayDate}</p>
</body>
</html>
```

EL

1. EL의 기본 문법
${expression}

2. 속성값 접근
// 예제: JSP 페이지에서 변수 접근
<%
    request.setAttribute("name", "John Doe");
%>
Hello, ${name}

3. 스코프 객체
Hello, ${sessionScope.name}, ${name}

4. 객체의 프로퍼티 접근
<%
    Person person = new Person();
    person.setName("Jane Doe");
    request.setAttribute("person", person);
%>
Hello, ${person.name}

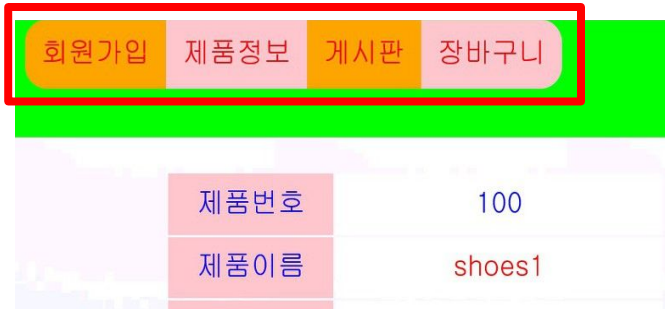
5. 배열 및 컬렉션 접근
<%
    String[] fruits = {"Apple", "Banana", "Cherry"};
    request.setAttribute("fruits", fruits);
%>
First fruit: ${fruits[0]}

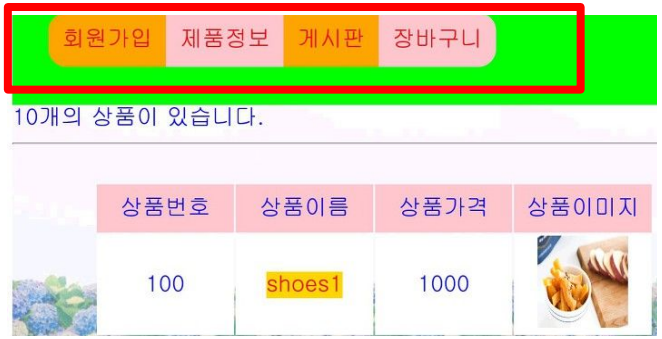
5. 배열 및 컬렉션 접근
<%
    List<String> animals = Arrays.asList("Dog", "Cat", "Elephant");
    request.setAttribute("animals", animals);
%>
Second animal: ${animals[1]}

6. 연산자
<%
    int a = 10;
    int b = 20;
    request.setAttribute("a", a);
    request.setAttribute("b", b);
%>
Sum: ${a + b}
Is a greater than b? ${a > b}

7. 내장 함수
<%
    String[] fruits = {"Apple", "Banana", "Cherry"};
    request.setAttribute("fruits", fruits);
%>
Is fruits empty? ${empty fruits}

include지시자






```
<div id="total">
    <div id="top">
        <jsp:include page="top.jsp"></jsp:include>
    </div>
    <div id="top2">
        <jsp:include page="top2.jsp"></jsp:include>
    </div>
    <div id="center">
        <span class="badge bg-primary">제품리스트 개수: <%= list.size() %>개
        <table class="table-hover table-info table">
```

jsp예제

```jsp
<%@ page language="java" contentType="text/html;
charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
   <title>Login Page</title>
</head>
<body>
<h2>회원 로그인</h2>
<form action="login_result.jsp" method="post">
   <div>
       <label for="username"> 아이디:</label>
       <input type="text" id="username" name="username">
   </div>
   <div>
       <label for="password"> 비밀번호:</label>
       <input type="password" id="password" name="password”>
   </div>
   <div>
       <button type="submit"> 로그인</button>
   </div>
</form>
</body>
</html>
```

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
 <title>Login Result</title>
</head>
<body>
<%
 String username = request.getParameter("username");
 String password = request.getParameter("password");

 // 예제를 위한 간단한 사용자 정보 검증
 String validUsername = "user123";
 String validPassword = "pass123";

 if (username.equals(validUsername) && password.equals(validPassword))
{
   out.println("<h2> 로그인 성공! 환영합니다, " + username + "!</h2>");
 } else {
   out.println("<h2> 로그인 실패: 아이디 또는 비밀번호가
잘못되었습니다.</h2>");
 }
%>
</body>
</html>
```

MVC

MVC 방법론

- MVC (Model-View-Controller) 방법론은 소프트웨어 엔지니어링에서 널리 사용되는 아키텍처 패턴
- 어플리케이션의 데이터 로직(Model), 사용자 인터페이스(View), 그리고 입력 처리(Controller)를 분리 목적
- 유지보수와 확장이 용이한 코드 구조를 만드는 것
- 한 부분의 변경이 다른 부분에 미치는 영향을 최소화하고, 개발 프로세스를 보다 체계적으로 관리

MVC 방법론

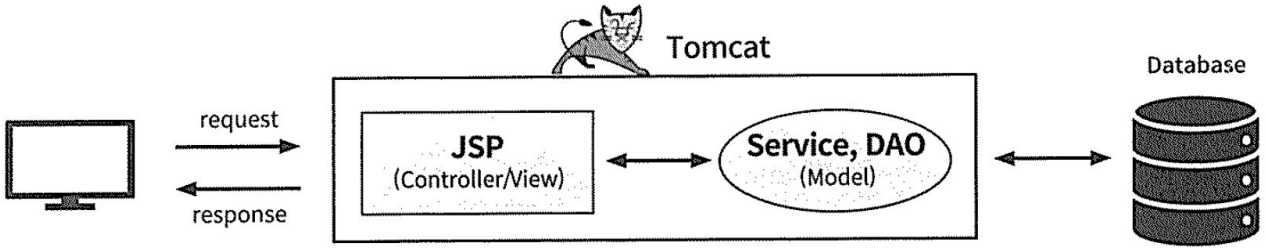
- **Model**
  - Model은 어플리케이션의 데이터와 관련된 로직을 처리
  - 데이터베이스, 계산, 또는 다른 데이터 소스와의 상호작용을 담당
  - 데이터의 상태 변화에 대해 View와 Controller에 통지할 수 있는 기능 담당
- **View**
  - View는 사용자에게 보이는 부분, 즉 사용자 인터페이스(UI)
  - Model에서 전달 받은 데이터를 기반으로 사용자에게 정보를 표시하는 방법을 정의
  - 일반적으로 HTML/CSS, JavaScript 등을 사용하여 구현
- **Controller**
  - Controller는 사용자의 입력을 처리하고 그에 따라 Model을 업데이트하고 View를 선택하는 역할
  - 사용자의 행동에 따라 Model에서 데이터를 요청하거나 데이터를 업데이트하고, 그 결과를 바탕으로 적절한 View를 응답으로 제공
  - 사용자 입력 정보 추출, 화면 이동(내비게이션) 등의 기능 담당

MVC1, MVC2

구분	MVC1	MVC2
구조	Controller와 View가 혼합	Model, View, Controller가 명확히 분리
기술적 측면	JSP, ASP 등 페이지 중심	서블릿, JSP 등을 사용한 분리된 처리
유지보수성	낮음 (로직 혼재)	높음 (로직 분리)
확장성	제한적	높음
적용 분야	간단한 웹 어플리케이션	복잡한 웹 및 엔터프라이즈 어플리케이션

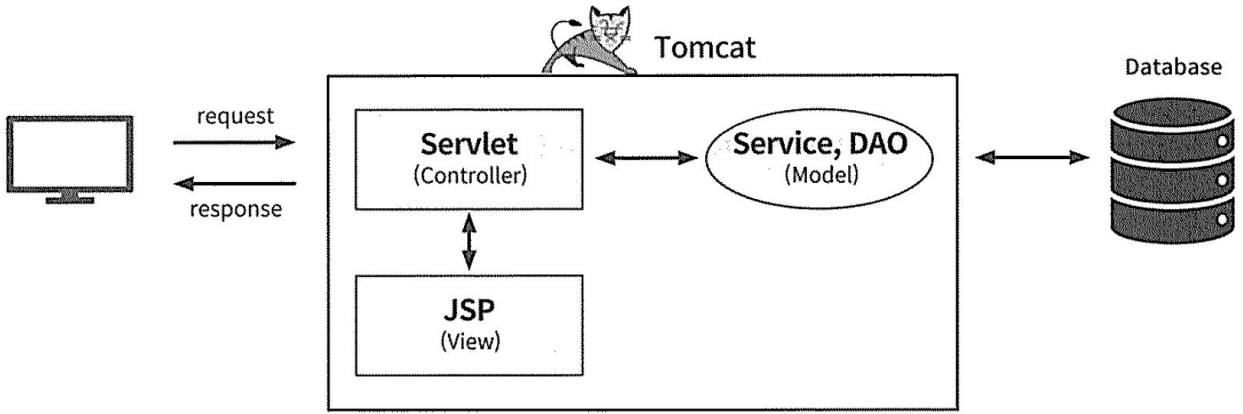
MVC1, MVC2

- **MVC1**
  - 초기 웹 애플리케이션에서 사용된 패턴으로, JSP(Java Server Pages)나 ASP(Active Server Pages) 같은 서버 페이지 기술을 사용하여 Controller와 View의 기능이 혼합된 형태
  - 이JSP나 ASP 페이지가 데이터 처리와 사용자 인터페이스를 동시에 담당하기 때문에, 비즈니스 로직과 프리젠테이션 로직이 혼재
  - 결과적으로, 애플리케이션의 유지보수와 확장성이 저하될 수 있음.



MVC1, MVC2

- **MVC2**
  - MVC2는 MVC1의 단점을 개선하기 위해 등장
  - Controller 역할을 서블릿과 같은 서버측 컴포넌트가 담당하고, View는 JSP 또는 다른 템플릿 기술을 사용하여 순수하게 사용자 인터페이스만을 처리
  - Model, View, Controller가 더욱 명확하게 분리되어 있어, 각각 독립적으로 개발 및 관리할 수 있으며, 전체적인 애플리케이션의 테스트와 유지보수가 용이

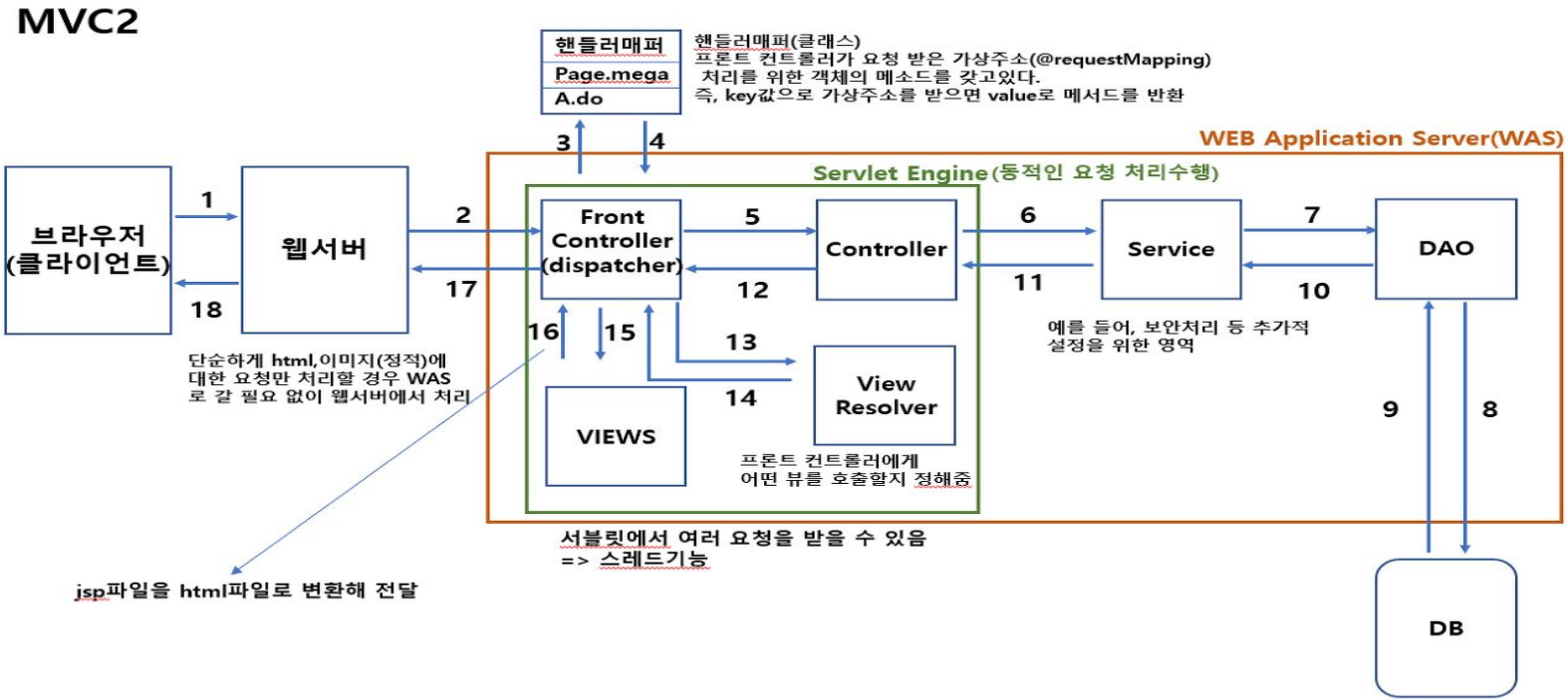


MVC1, MVC2

- **MVC2**

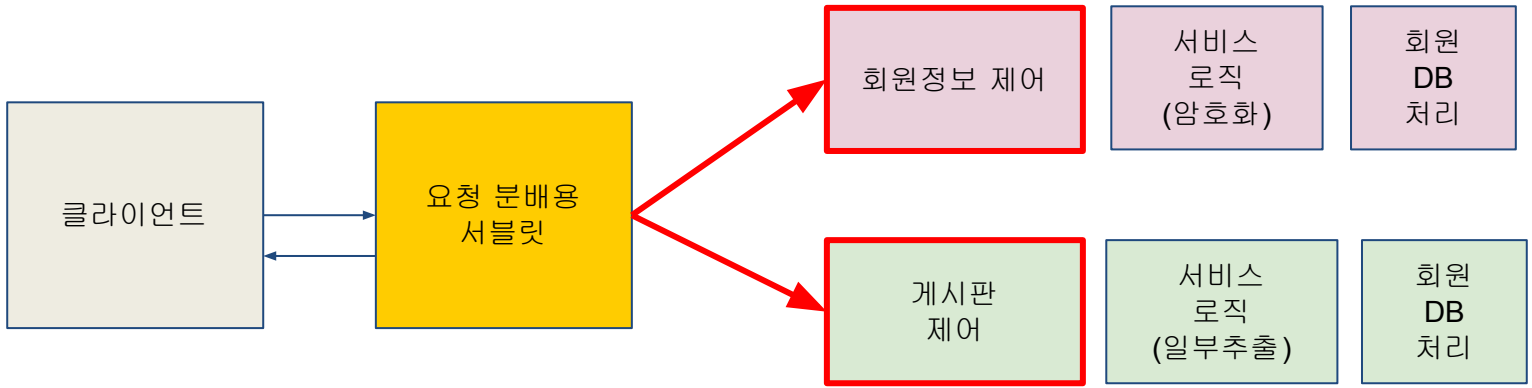
주체	역할	책임	중요성
Model	데이터 및 비즈니스 로직 처리	데이터 관리, 비즈니스 규칙 수행, 데이터 상태 변경 알림	데이터의 무결성 및 일관성 유지, 다른 구성 요소와의 독립성 보장
View	사용자 인터페이스 표시	사용자에게 정보 제공, 입력 인터페이스 제공, 데이터 렌더링	사용자 경험 및 인터페이스의 사용성과 접근성 결정
Controller	사용자 입력 및 응답 조정	입력 처리, 데이터 처리 요청 및 업데이트, View 선택 및 데이터 전달	애플리케이션의 흐름 관리, Model과 View의 중개자 역할

spring framework mvc2

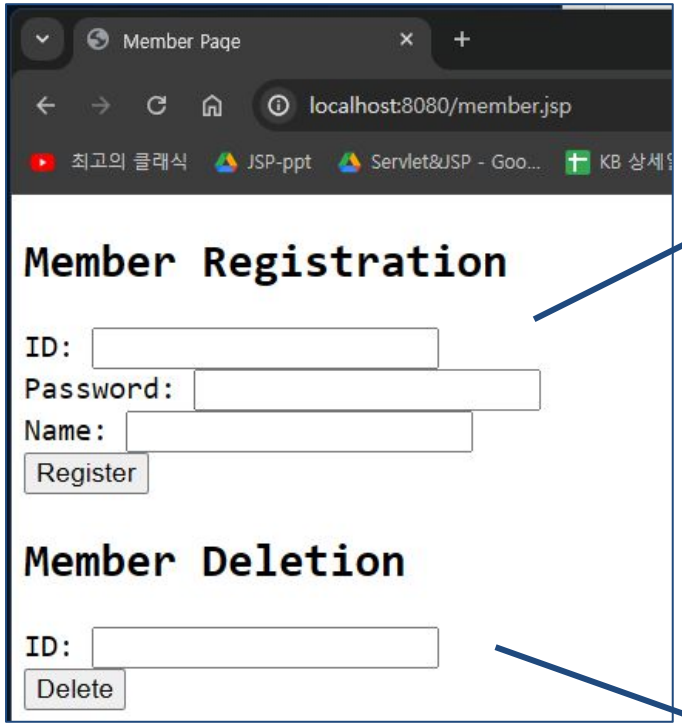


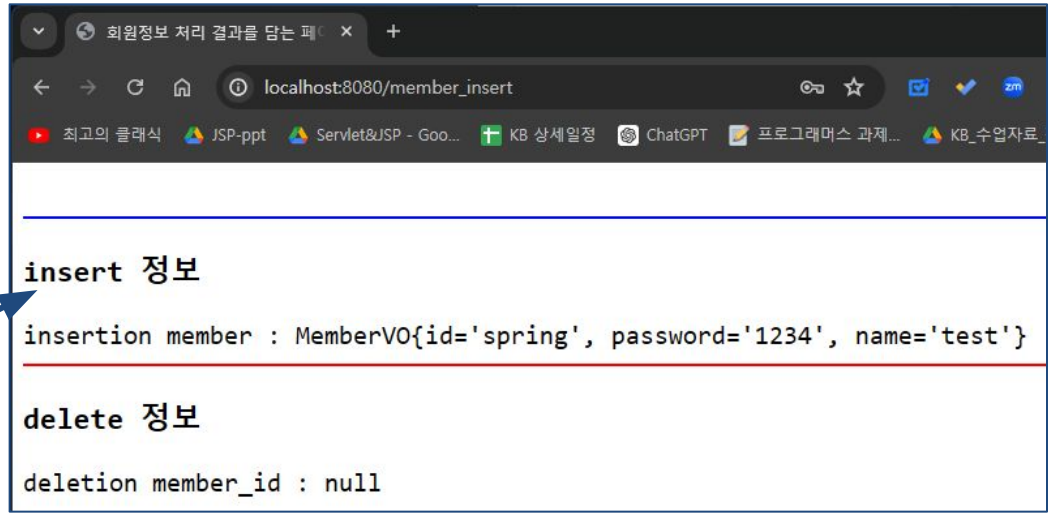
DispatcherServlet

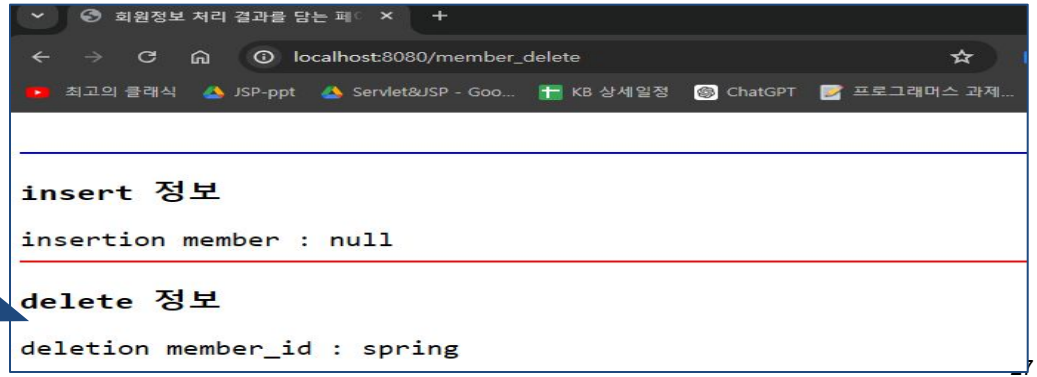
- 요청 분배용 서블릿
- Spring MVC 프레임워크에서 매우 중요한 역할을 하는 핵심 구성 요소 중 하나
- 요청(request)과 응답(response)을 처리하는 중앙 허브 역할
- 클라이언트로부터 들어오는 모든 HTTP 요청은 DispatcherServlet이 수신.
  DispatcherServlet은 요청을 적절한 핸들러(Controller)로 전달하고, 응답을 생성하여 클라이언트에게 돌려줌.



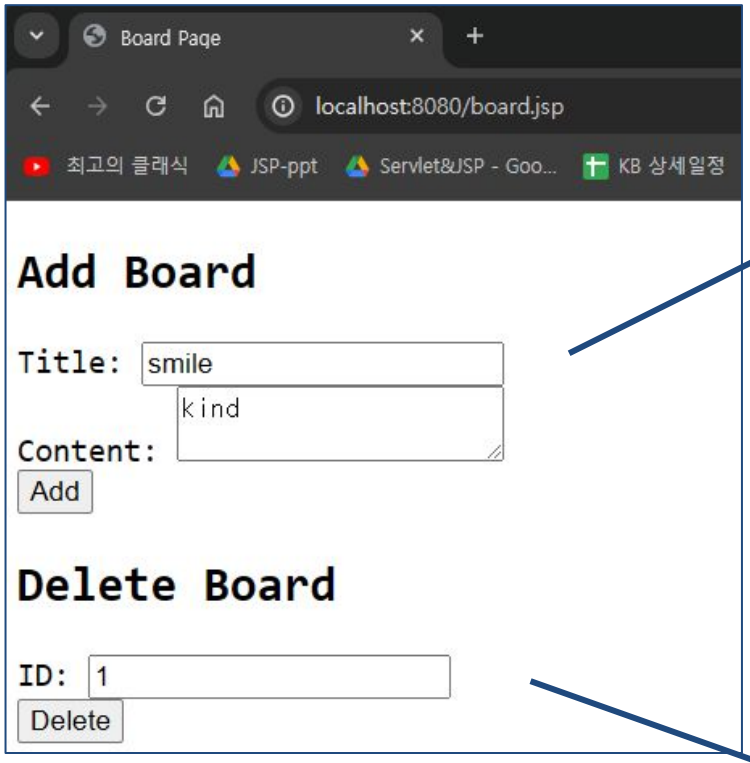
회원 정보 처리 요청

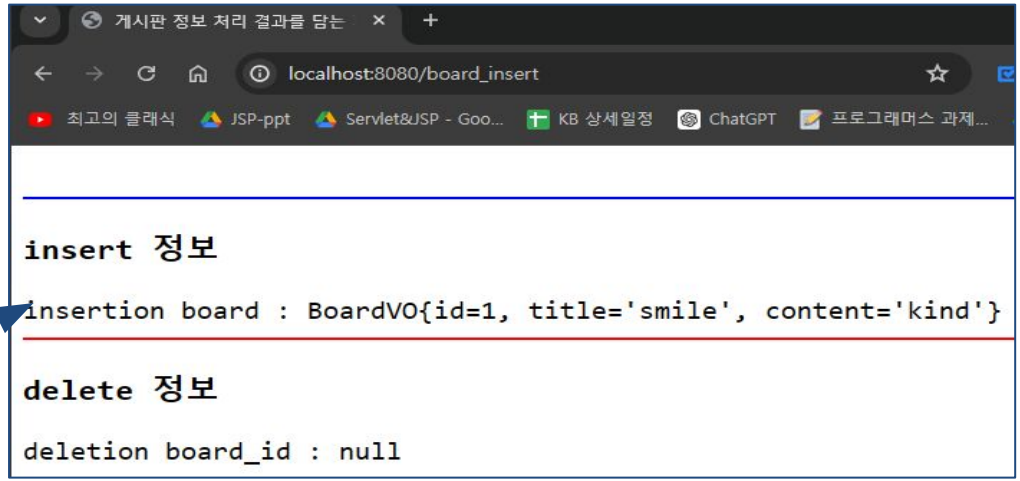

Member Page
localhost:8080/member.jsp
최고의 클래식
JSP-ppt
Servlet&JSP - Goo...
KB 상세일
Member Registration
ID:
Password:
Name:
Register
Member Deletion
ID:
Delete

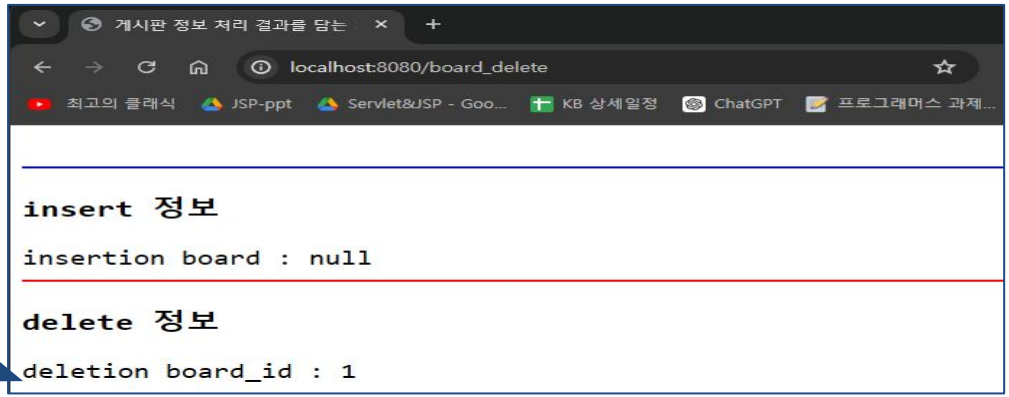

회원정보 처리 결과를 담는 페
localhost:8080/member_insert
최고의 클래식
JSP-ppt
Servlet&JSP - Goo...
KB 상세일정
ChatGPT
프로그래머스 과제...
KB_수업자료_
insert 정보
insertion member : MemberVO{id='spring', password='1234', name='test'}
delete 정보
deletion member_id : null


회원정보 처리 결과를 담는 페
localhost:8080/member_delete
최고의 클래식
JSP-ppt
Servlet&JSP - Goo...
KB 상세일정
ChatGPT
프로그래머스 과제...
insert 정보
insertion member : null
delete 정보
deletion member_id : spring

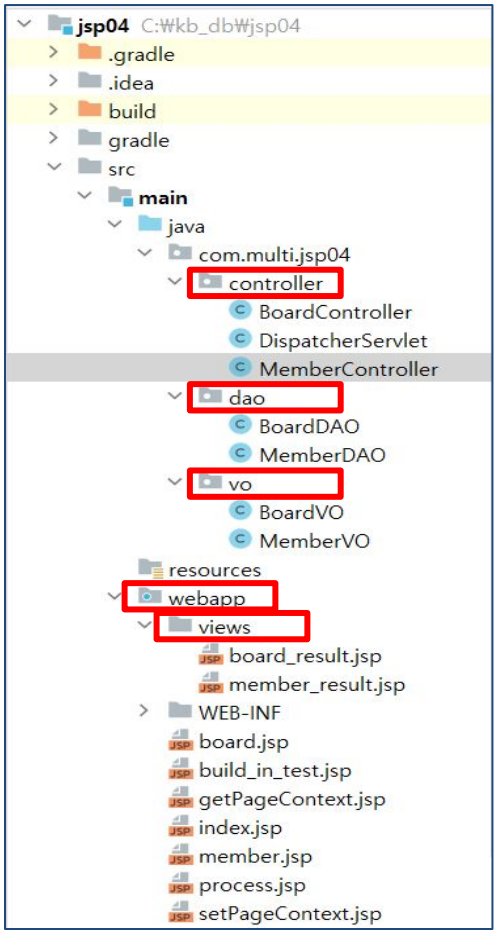
게시판 정보 처리 요청


Board Page
localhost:8080/board.jsp
최고의 클래식
JSP-ppt
Servlet&JSP - Goo...
KB 상세일정
Add Board
Title:
smile
Content:
kind
Add
Delete Board
ID:
1
Delete


게시판 정보 처리 결과를 담는
localhost:8080/board_insert
최고의 클래식
JSP-ppt
Servlet&JSP - Goo...
KB 상세일정
ChatGPT
프로그래머스 과제...
insert 정보
insertion board : BoardVO{id=1, title='smile', content='kind'}
delete 정보
deletion board_id : null


게시판 정보 처리 결과를 담는
localhost:8080/board_delete
최고의 클래식
JSP-ppt
Servlet&JSP - Goo...
KB 상세일정
ChatGPT
프로그래머스 과제...
insert 정보
insertion board : null
delete 정보
deletion board_id : 1

프로젝트 트리



구성 요소	파일/폴더	설명
Model	com.multi.jsp04.vo	BoardVO.java, MemberVO.java 데이터 전달 객체, 데이터베이스의 데이터를 담고 이동하는 객체
Model	com.multi.jsp04.dao	BoardDAO.java, MemberDAO.java 데이터 접근 객체, 데이터베이스와 상호작용하여 데이터를 처리
View(결과출력용)	webapp/views	board_result.jsp, member_result.jsp 사용자에게 데이터를 보여주는 역할
View(입력용)	webapp	board.jsp, member.jsp, index.jsp 사용자 인터페이스를 제공, 데이터를 입력받음
Controller	com.multi.jsp04.controller	BoardController.java, MemberController.java, DispatcherServlet.java 사용자의 요청을 처리하고 적절한 모델과 뷰를 호출

View(입력용)

- webapp/member.jsp

```
<%@ page contentType="text/html;charset=UTF-8" %>
<!DOCTYPE html>
<html>
<head>
   <title>Member Page</title>
</head>
<body>
<h2>Member Registration</h2>
<form action="/member_insert" method="post">
   ID: <input type="text" name="id"/><br/>
   Password: <input type="password"
name="password"/><br/>
   Name: <input type="text" name="name"/><br/>
   <input type="submit" value="Register"/>
</form>

<h2>Member Deletion</h2>
<form action="/member_delete" method="post">
   ID: <input type="text" name="id"/><br/>
   <input type="submit" value="Delete"/>
</form>
</body>
</html>
```

- webapp/board.jsp

```
<%@ page contentType="text/html;charset=UTF-8" %>
<!DOCTYPE html>
<html>
<head>
   <title>Board Page</title>
</head>
<body>
<h2>Add Board</h2>
<form action="/board_insert" method="post">
   Title: <input type="text" name="title"/><br/>
   Content: <textarea name="content"></textarea><br/>
   <input type="submit" value="Add"/>
</form>

<h2>Delete Board</h2>
<form action="/board_delete" method="post">
   ID: <input type="text" name="id"/><br/>
   <input type="submit" value="Delete"/>
</form>
</body>
</html>
```

View(결과 출력용)

- webapp/views/member_result.jsp

```
<%@ page contentType="text/html;charset=UTF-8"
language="java" %>
<html>
<head>
   <title>회원정보 처리 결과를 담는 페이지</title>
</head>
<body>
<br>
<hr color="blue">

<h3>insert 정보</h3>
insertion member : <%= request.getAttribute("member")%>

<hr color="red">

<h3>delete 정보</h3>
deletion member id : <%=
request.getAttribute("member_id")%>
</body>
</html>
```

- webapp/views/board_result.jsp

```
<%@ page contentType="text/html;charset=UTF-8"
language="java" %>
<html>
<head>
   <title>게시판 정보 처리 결과를 담는 페이지</title>
</head>
<body>
<br>
<hr color="blue">

<h3>insert 정보</h3>
insertion board : <%= request.getAttribute("board")%>

<hr color="red">

<h3>delete 정보</h3>
deletion board id : <%=
request.getAttribute("board_id")%>
</body>
</html>
```

Controller(controller/DispatcherServlet.java)

```
package com.multi.jsp04.controller;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/")
public class DispatcherServlet extends HttpServlet {
   private MemberController memberController = new MemberController();
   private BoardController boardController = new BoardController();

   @Override
   protected void service(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
       String uri = request.getRequestURI();
       if (uri.equals("/member_insert")) {
           memberController.insert(request, response);
       } else if (uri.equals("/member_delete")) {
           memberController.delete(request, response);
       } else if (uri.equals("/board_insert")) {
           boardController.insert(request, response);
       } else if (uri.equals("/board_delete")) {
           boardController.delete(request, response);
       }
   }
}
```

Controller

- controller/MemberController.java

```
public class MemberController {
   private MemberDAO memberDAO = new MemberDAO();

   public void insert(HttpServletRequest request,
HttpServletResponse response)  throws IOException, ServletException
{
           MemberVO member =  new MemberVO();
           member.setId(request.getParameter( "id"));
           member.setPassword(request.getParameter( "password"));
           member.setName(request.getParameter( "name"));
           memberDAO.insertMember(member);
           request.setAttribute( "member", member);
           RequestDispatcher rd =
request.getRequestDispatcher( "views/member_result.jsp");
           rd.forward(request, response);
   }
   public void delete(HttpServletRequest request,
HttpServletResponse response)  throws IOException, ServletException
{
           String id = request.getParameter( "id");
           memberDAO.deleteMember(id);
           request.setAttribute( "member_id", id);
           RequestDispatcher rd =
request.getRequestDispatcher( "views/member_result.jsp");
           rd.forward(request, response);
       }
   }
```

- controller/BoardController.java

```
public class BoardController {

   private BoardDAO boardDAO = new BoardDAO();

   public void insert(HttpServletRequest request,
HttpServletResponse response)  throws IOException, ServletException
{
       BoardVO board =  new BoardVO();
       board.setTitle(request.getParameter( "title"));
       board.setContent(request.getParameter( "content"));
       boardDAO.insertBoard(board);
       request.setAttribute( "board", board);
       RequestDispatcher rd =
request.getRequestDispatcher( "views/board_result.jsp");
       rd.forward(request, response);
   }
   public void delete(HttpServletRequest request,
HttpServletResponse response)  throws IOException, ServletException
{
       int id = Integer.parseInt(request.getParameter( "id"));
       boardDAO.deleteBoard(id);
       request.setAttribute( "board_id", id);
       RequestDispatcher rd =
request.getRequestDispatcher( "views/board_result.jsp");
       rd.forward(request, response);
       }
   }
```

Model

- dao/MemberDAO.java

```java
package com.multi.jsp04.dao;

import com.multi.jsp04.vo.MemberVO;
import java.util.HashMap;
import java.util.Map;

public class MemberDAO {
   private Map<String, MemberVO> members = new HashMap<>();

   public void insertMember(MemberVO member) {
       members.put(member.getId(), member);
   }

   public void deleteMember(String id) {
       members.remove(id);
   }
}
```

- dao/BoardDAO.java

```java
package com.multi.jsp04.dao;

import com.multi.jsp04.vo.BoardVO;
import java.util.HashMap;
import java.util.Map;

public class BoardDAO {
   private Map<Integer, BoardVO> boards = new HashMap<>();
   private int nextId = 1;

   public void insertBoard(BoardVO board) {
       board.setId(nextId++);
       boards.put(board.getId(), board);
   }

   public void deleteBoard(int id) {
       boards.remove(id);
   }
}
```

VO

- vo/MemberVO.java

```
public class MemberVO {
    private String id;
    private String password;
    private String name;

    // getters and setters
    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;
    }

    @Override
    public String toString() {
        return "MemberVO{" +
                "id='" + id + '\'' +
                ", password='" + password + '\'' +
                ", name='" + name + '\'' +
                '}';
    }
}
```

- vo/BoardVO.java

```
public class BoardVO {
    private int id;
    private String title;
    private String content;

    // getters and setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    @Override
    public String toString() {
        return "BoardVO{" +
                "id=" + id +
                ", title='" + title + '\'' +
                ", content='" + content + '\'' +
                '}';
    }
}
```

내장객체

내장객체

내장 객체	데이터 타입	설명
request	javax.servlet.http.HttpServletRequest	클라이언트로부터 들어오는 요청 정보를 담고 있는 객체
response	javax.servlet.http.HttpServletResponse	서버가 클라이언트로 응답을 보낼 때 사용하는 객체
session	javax.servlet.http.HttpSession	사용자별로 정보를 저장할 수 있는 세션 객체
application	javax.servlet.ServletContext	서블릿이 실행되는 컨텍스트에 대한 정보를 제공하며, 애플리케이션 범위의 데이터를 저장할 수 있는 객체
config	javax.servlet.ServletConfig	서블릿의 초기화 파라미터와 관련된 설정 정보를 제공하는 객체
out	javax.servlet.jsp.JspWriter	JSP 페이지에서 클라이언트로 데이터를 출력하는 데 사용하는 객체
pageContext	javax.servlet.jsp.PageContext	JSP 페이지의 컨텍스트 정보를 제공하는 객체

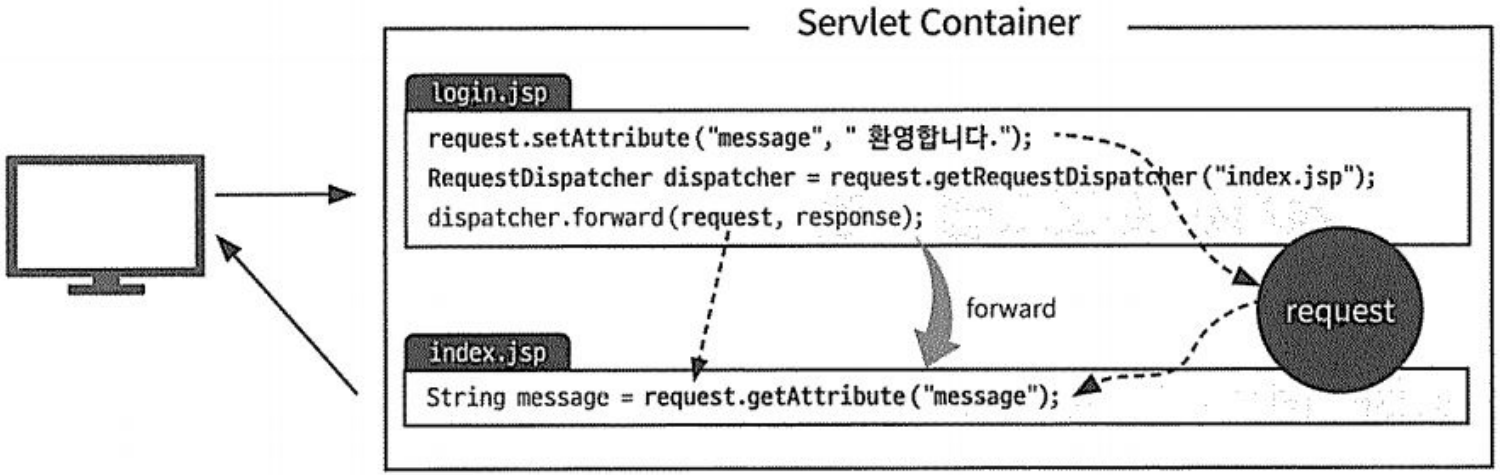
pageContext

- 다양한 정보를 제공하는 중요한 내장 객체
  - pageContext.setAttribute("attributeName", attributeValue);
  - Object value = pageContext.getAttribute("attributeName");
  - pageContext.removeAttribute("attributeName");

메서드	기능
void forward(String url)	특정 페이지로 화면을 이동
Exception getException()	발생한 예외 객체를 리턴
Object getPage()	현재 JSP에 해당하는 객체를 리턴
ServletRequest getRequest()	현재 JSP에서 사용하는 HttpServletRequest 객체를 리턴
ServletResponse getResponse()	현재 JSP에서 사용하는 HttpServletResponse 객체를 리턴
ServletConfig getServletConfig()	현재 JSP에서 사용하는 ServletConfig 객체를 리턴
ServletContext getServletContext()	현재 JSP에서 해당하는 ServletContext 객체를 리턴
HttpSession getSession()	현재 JSP에서 해당하는 HttpSession 객체를 리턴
void include(String url)	특정 URL의 실행 결과를 현재 JSP에 포함

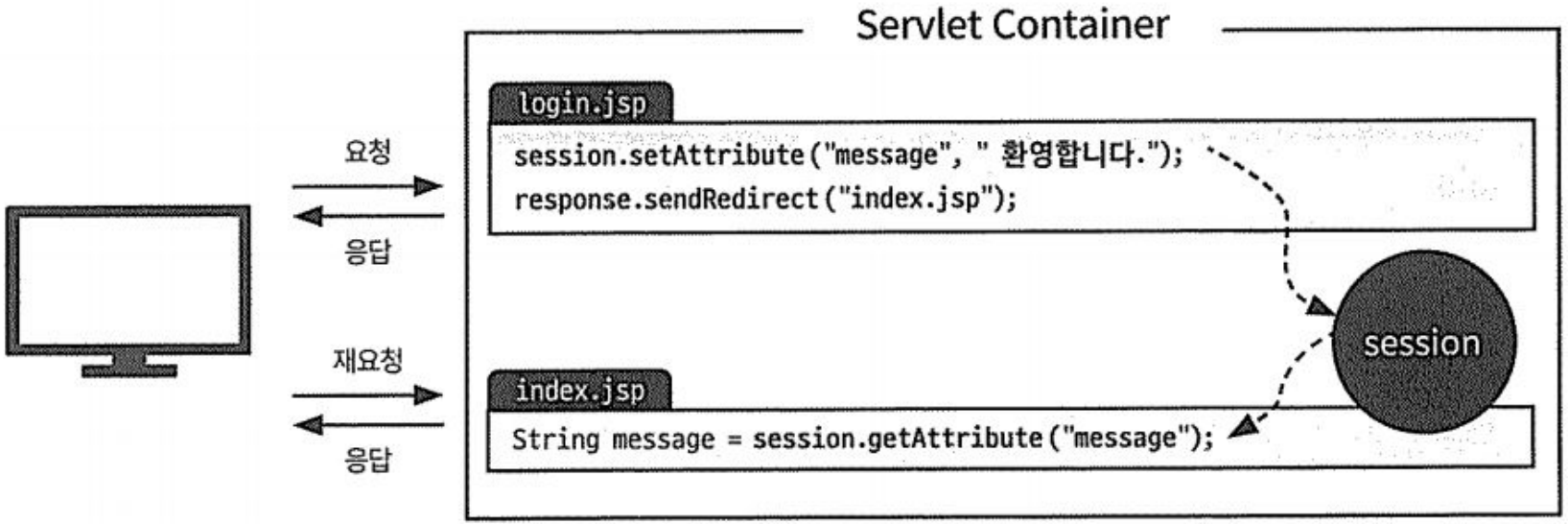
request

- 클라이언트 요청 정보를 처리하는 객체



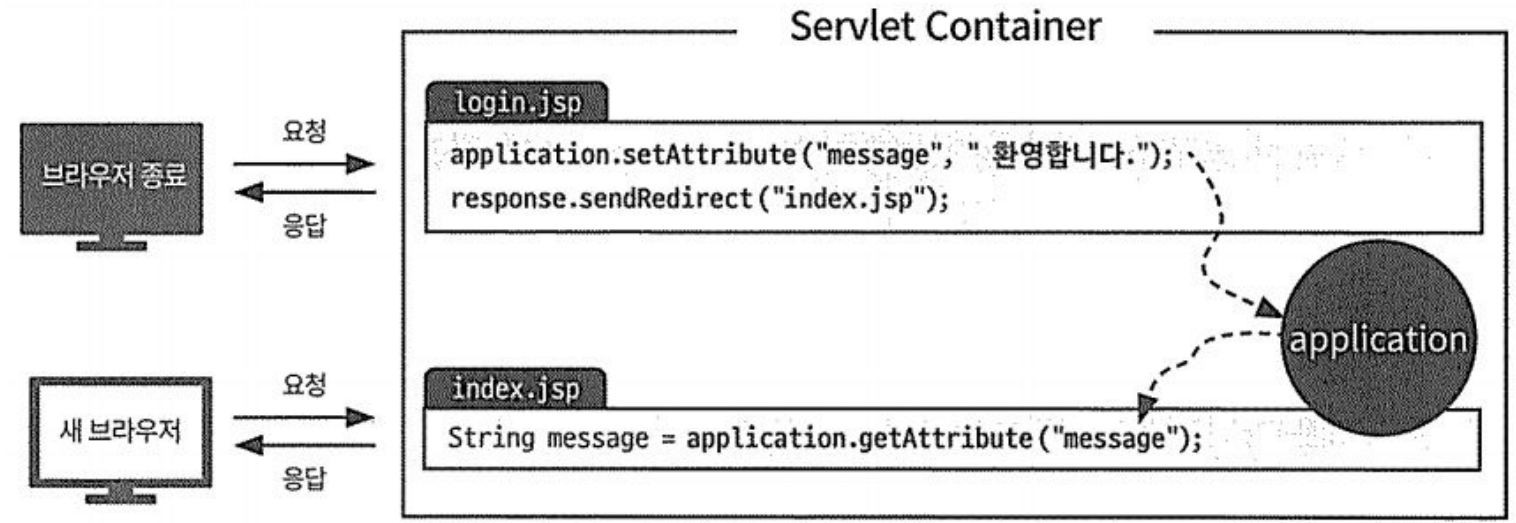
session

- 사용자별 세션 데이터를 저장하고 관리하는 객체



application

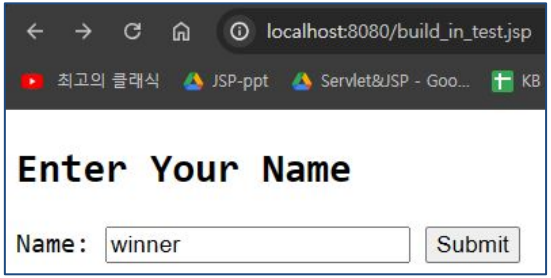
- 애플리케이션 범위의 데이터를 저장하고 초기화 파라미터를 읽는 객체



내장객체 테스트

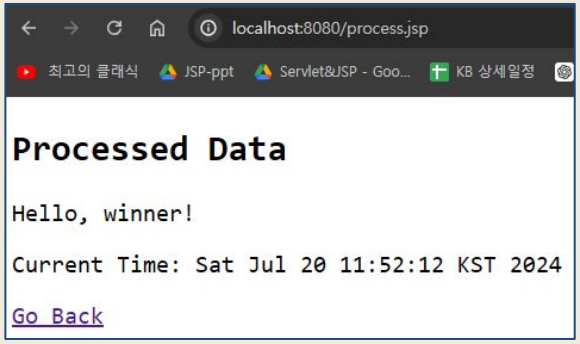
build_in_test.jsp

```
<!DOCTYPE html>
<html>
<head>
    <title>JSP Example</title>
</head>
<body>
    <h2>Enter Your Name</h2>
    <form action="process.jsp" method="post">
        Name: <input type="text" name="username" />
        <input type="submit" value="Submit" />
    </form>
</body>
</html>
```



process.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page import="java.util.Date" %>
<%
    // request 객체를 통해 사용자 입력 데이터 가져오기
    String username = request.getParameter("username");

    // 세션에 사용자 이름 저장
    session.setAttribute("username", username);

    // 애플리케이션 컨텍스트에 현재 시간 저장
    application.setAttribute("currentTime", new Date());

    // pageContext를 통해 세션에서 사용자 이름 가져오기
    String sessionUsername = (String) pageContext.getAttribute("username",
PageContext.SESSION_SCOPE);

    // pageContext를 통해 애플리케이션에서 현재 시간 가져오기
    Date currentTime = (Date) pageContext.getAttribute("currentTime",
PageContext.APPLICATION_SCOPE);
%>
<!DOCTYPE html>
<html>
<head>
    <title>Process Page</title>
</head>
<body>
    <h2>Processed Data</h2>
    <p>Hello, <%= sessionUsername %>!</p>
    <p>Current Time: <%= currentTime %></p>
    <p><a href="index.jsp">Go Back</a></p>
</body>
</html>
```



내장객체 테스트

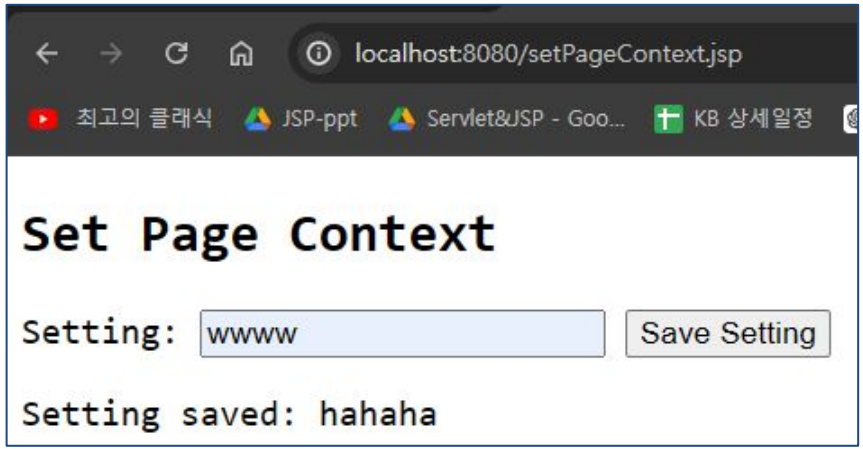
setPageContext.jsp

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
   <title>Set Page Context</ title>
</head>
<body>
<h2>Set Page Context</ h2>
<form action="setPageContext.jsp" method="post">
   Setting: <input type="text" name="settingValue" />
   <input type="submit" value="Save Setting" />
</form>

<%
   if (request.getMethod().equalsIgnoreCase( "POST")) {
       // request 객체를 통해 사용자 입력 데이터 가져오기
       String settingValue = request.getParameter( "settingValue");

       // pageContext 객체를 사용하여 설정값 저장
       pageContext.setAttribute( "pageSetting", settingValue,
PageContext.PAGE_SCOPE);

       out.println( "<p>Setting saved: " + settingValue +  "</p>");
   }
%>
</body>
</html>
```

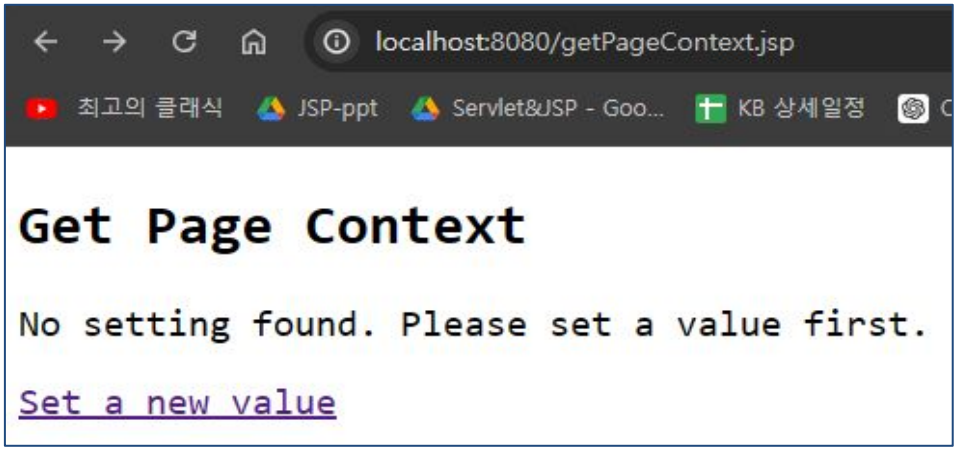
getPageContext.jsp

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
   <title>Get Page Context</ title>
</head>
<body>
<h2>Get Page Context</ h2>

<%
   // pageContext 객체를 사용하여 설정값 불러오기
   String pageSetting = (String)
pageContext.getAttribute( "pageSetting", PageContext.PAGE_SCOPE);

   if (pageSetting != null) {
       out.println( "<p>Saved Setting: " + pageSetting +  "</p>");
   } else {
       out.println( "<p>No setting found. Please set a value
first.</p>");
   }
%>

<p><a href="setPageContext.jsp">Set a new value</ a></p>
</body>
</html>
```

내장객체 테스트



- pageContext로 설정된 값이 있어 출력 가능



- pageContext로 설정된 값이 없어 출력 불가능

핵심정리

- JSP
  - 핵심 문법
  - 내장 객체
  - JSTL, EL
  - 지시자
- **MVC2**
  - DispatcherServlet
  - Model
  - Controller
  - View