# Web02_JSP_Servlet

작성자 양성호

# Servlet과 JSP의 차이와 관계를 이해한다.

- Servlet과 JSP의 개념을 이해한다.
- Servlet과 JSP의 차이를 이해한다.
- Servlet과 JSP의 관계를 이해한다.

# Servlet과 JSP의 개념

- 기능의 차이는 없고 역할의 차이만 있다. (하는 일은 동일)

# Servlet이란

- 웹 기반의 요청에 대한 동적인 처리가 가능한 Server Side에서 돌아가는 Java Program
- Java 코드 안에 HTML 코드 (하나의 클래스)
- 웹 개발을 위해 만든 표준

# JSP란

- Java 언어를 기반으로 하는 Server Side 스크립트 언어
- HTML 코드 안에 Java 코드
- Servlet를 보완하고 기술을 확장한 스크립트 방식 표준
- Servlet의 모든 기능 + 추가적인 기능

# Servlet과 JSP의 차이

- Servlet
- Java 코드 안에 HTML 코드 (하나의 클래스)
- data processing(Controller)에 좋다.
- 즉 DB와의 통신, Business Logic 호출, 데이터를 읽고 확인하는 작업 등에 유용하다.
- Servlet이 수정된 경우 Java 코드를 컴파일(.class 파일 생성)한 후 동적인 페이지를 처리하기 때문에 전체 코드를 업데이트하고 다시 컴파일한 후 재배포하는 작업이 필요하다. (개발 생산성 저하)

- JSP
- HTML 코드 안에 Java 코드
- presentation(View)에 좋다.
- 즉 요청 결과를 나타내는 HTML 작성하는데 유용하다.
- JSP가 수정된 경우 재배포할 필요가 없이 WAS가 알아서 처리한다. (쉬운 배포)

# 서블릿 예시

```
1  package test.com.controller;
2* import java.io.IOException;
9*  * Servlet implementation class MemberController
11 @WebServlet({ "/index.do", "/insert.do", "/selectAll.do", "/selectOne.do" ,
12     "/insertOK.do","/updateOK.do","/deleteOK.do",
13     "/result.do","/result2.do"})
14 public class MemberController extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
18*     * @see HttpServlet#HttpServlet()
20*    public MemberController() {
24
26*     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse resp
28*    protected void doGet(
29            HttpServletRequest request,
30            HttpServletResponse response) throws ServletException, IOException {
31        //response.getWriter().append("Served at: ").append(request.getContextPath
32
33        String sPath = request.getServletPath();
34        System.out.println(sPath);
35
36        if(sPath.equals("/index.do")) {
37            request.getRequestDispatcher("index.jsp").forward(request, response);
38        }else if(sPath.equals("/insert.do")) {
39            request.getRequestDispatcher("member/insert.jsp").forward(request, res
```

# jsp 예시

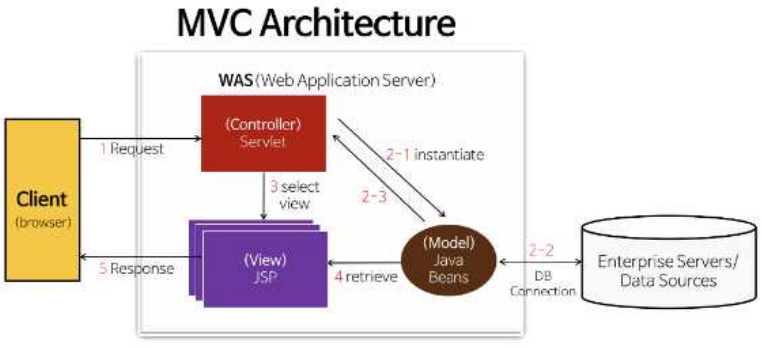
```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>insert.jsp</h1>

    회원가입입력폼 만드세요. 전송방식:get
    id,pw,name,tel

    <form action="insertOK.do" method="get">
        id:<input type="text"><br>
        pw:<input type="text"><br>
        name:<input type="text"><br>
        tel:<input type="text"><br>
        <input type="submit">
    </form>

</body>
</html>
```

# JSP와 Servlet을 모두 이용하는 모델 (MVC Architecture)



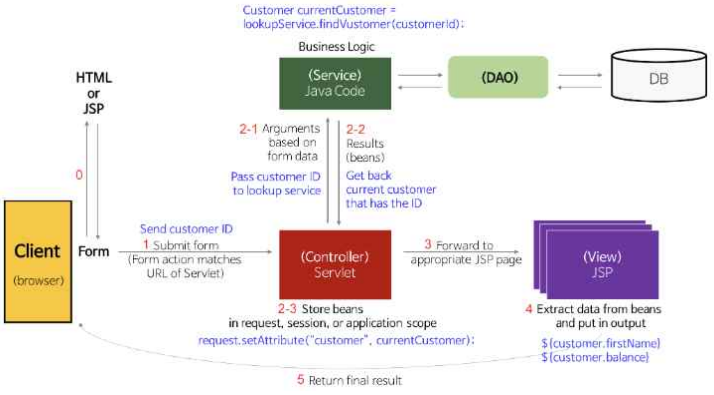
# MVC Architecture란

- JSP와 Servlet을 모두 사용하여 프레젠테이션 로직(View)과 비즈니스 로직(Controller)을 분리한다.
- View(보여지는 부분)는 HTML이 중심이 되는 JSP를 사용
- Controller(다른 자바 클래스에 데이터를 넘겨주는 부분)는 Java 코드가 중심이 되는 Servlet을 사용
- Model은 Java Beans로, DTO(VO)와 DAO를 통해 Mysql,Oracle 등과 같은 Data Storage에 접근

- Model, View, Controller를 분리한 디자인 패턴

- Model
- 애플리케이션의 상태(data)를 나타낸다.
- Java Beans
- View
- 디스플레이 데이터 또는 프리젠테이션
- JSP
- Controller
- View와 Model 사이의 인터페이스 역할
- Model/View에 대한 사용자 입력 및 명령을 수신하여 그에 따라 적절하게 변경
- Servlet
- https://gmlwjd9405.github.io/2018/11/05/mvc-architecture.html

# 구체적인 MVC Flow of Control(Annotated)



감사합니다.