JSP 2.3 & Servlet 3.1

SERVLET

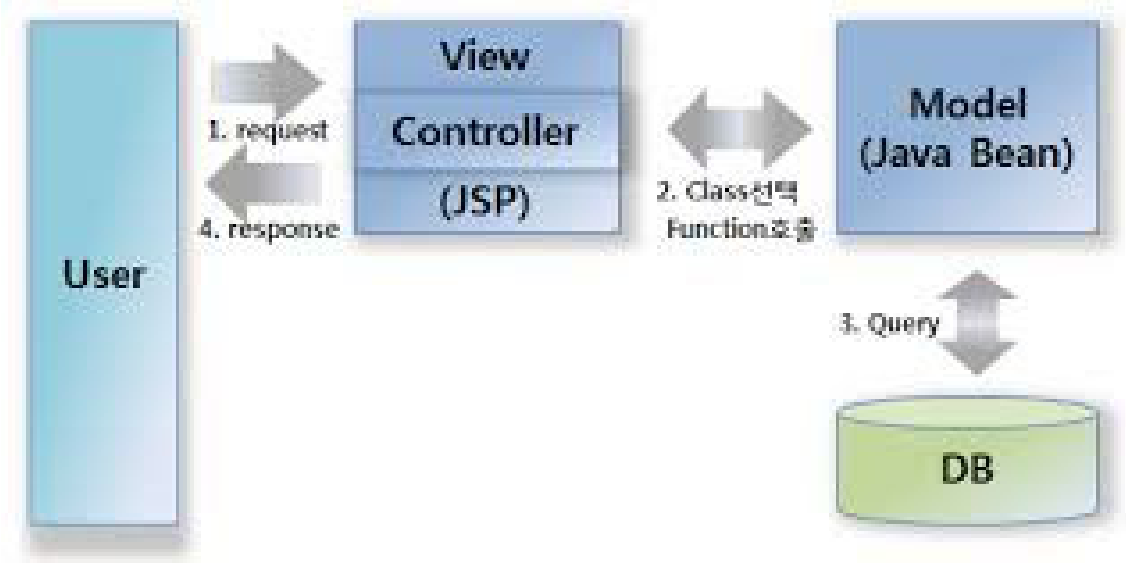
- 김근형 -

MVC

- MVC(Model View Controller)
  - Model : 프로그램의 내부 상태, 즉 프로그램의 정보(데이터)를 말하는 것이다.
  - Controller : 데이터와 비즈니스 로직 간의 상호 작용을 뜻함
  - View : 사용자 인터페이스 요소를 뜻하는데, 유저에게 보여지는 것을 말한다.

MVC1 모델

- MVC1

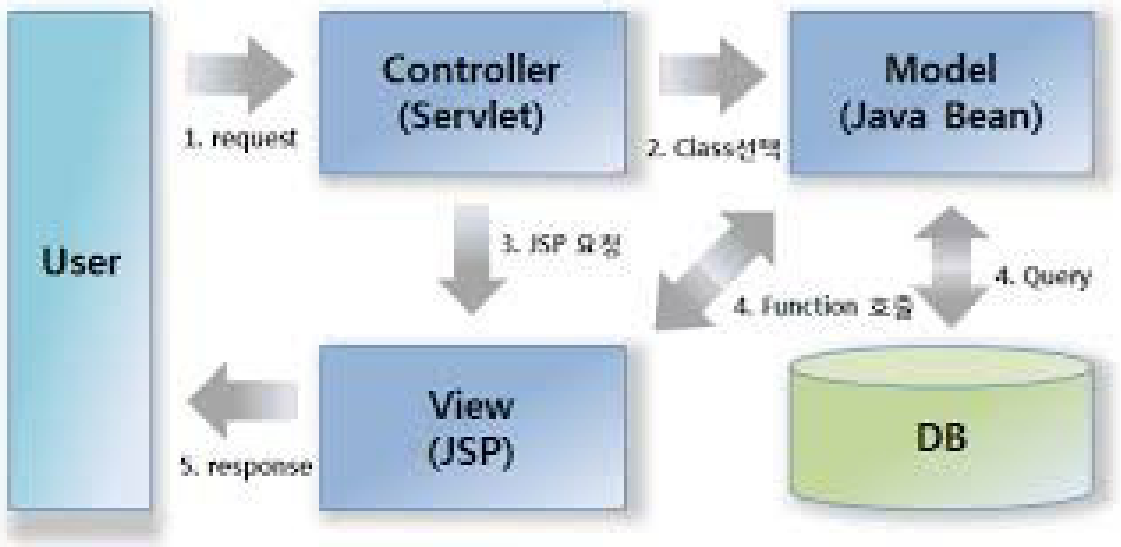

User
1. request
4. response
View
Controller
(JSP)
2. Class선택
Function호출
Model
(Java Bean)
3. Query
DB

MVC1

- MVC1
  - JSP로 구현한 기존 웹 어플리케이션은 모델 1 구조로 웹 브라우저의 요청을 JSP 페이지가 받아서 처리 하는 구조이다.
  - JSP 페이지에 비지니스 로직을 처리 하기 위한 코드와 웹 브라우저에 결과를 보여주기 위한 출력 관리 코드가 뒤섞여 있는 구조
  - JSP 페이지 안에서 모든 정보를 표현(view)하고 저장(model)하고 처리(control)되므로 재사용이 힘들고, 읽기도 힘들어 가독성이 떨어진다.

MVC2

- MVC2


User
1. request
Controller
(Servlet)
2. Class선택
Model
(Java Bean)
3. JSP 요청
4. Function 호출
4. Query
View
(JSP)
DB
5. response

MVC2

- MVC2
  - 클라이언트의 요청처리와 응답처리, 비즈니스 로직 처리하는 부분을 모듈화시킨 구조
  - MVC1 구조와 달리 웹 브라우저의 요청을 하나의 컨트롤러가 받게 됨
  - 컨트롤러는 웹 브라우저의 요청을 알맞게 처리한 후 그 결과를 JSP 페이지로 포워딩
  - 모델2 방식의 경우 실질적으로 보여지는 HTML과 JAVA 소스를 분리했기 때문에 모델1 방식에 비해 확장이 용이하고 유지보수하기도 쉽다

Servlet

- Servlet
  - 웹프로그래밍에서 클라이언트의 요청을 처리하고 그 결과를 다시 클라이언트에게 전송하는 자바 프로그래밍 기술
  - 클라이언트가 요청을 하면 그에 대한 결과를 다시 전송해 역할을 하는 자바 프로그램
  - 클라이언트의 요청에 대해 동적으로 작동하는 웹 어플리케이션 컴포넌트
  - html을 사용하여 요청에 응답한다.
  - Java Thread를 이용하여 동작한다.
  - MVC 패턴에서 Controller로 이용된다.
  - HTTP 프로토콜 서비스를 지원하는 javax.servlet.http.HttpServlet 클래스를 상속받는다. UDP 보다 속도가 느리다.
  - 변경 시 Servlet을 재 컴파일해야 하는 단점이 있다.

Servlet

- Servlet

```java
@WebServlet("/ServletExample1")
public class ServletExample1 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ServletExample1() {
        super();
        // TODO Auto-generated constructor stub
    }

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

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }

}
```

Servlet LifeCycle

- Servlet LifeCycle
  - 브라우저가 Servlet에 요청을 하면 Servlet은 바로 호출이 되지 않는다
  - Servlet 객체를 생성하고 초기화 작업을 거친 후에 요청을 처리하게 되는데, Servlet은 아래와 같은 생명주기를 가지고 있다.
    - 요청이오면, Servlet 클래스가 로딩되어 요청에 대한 Servlet 객체가 생성된다.
    - 서버는 init() 메소드를 호출해서 Servlet을 초기화한다.
    - service() 메소드를 호출해서 Servlet이 브라우저의 요청을 처리하도록 한다.
    - service() 메소드는 특정 HTTP 요청(GET, POST 등)을 처리하는 메서드 (doGet(), doPost() 등)를 호출한다.
    - 서버는 destroy() 메소드를 호출하여 Servlet을 제거한다.

Servlet LifeCycle

- Servlet LifeCycle

```java
@WebServlet("/ServletExample2")
public class ServletExample2 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ServletExample2() {

    }

    public void init(ServletConfig config) throws ServletException {
        System.out.println("init() 메서드 호출");
    }
    public void destroy() {
        System.out.println("destroy() 메서드 호출");
    }
    protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("service() 메서드 호출");
        doPost(request, response);
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("doGet() 메서드 호출");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("doPost() 메서드 호출");
        doGet(request, response);
    }

}
```

Servlet Redirect

- Redirect 소스

```
@WebServlet("/ServletRedirect")
public class ServletRedirect extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ServletRedirect() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.sendRedirect("/Chapter14/RedirectPage.jsp");
    }

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

}
```

Servlet Redirect

- Forward 소스

```
@WebServlet("/ServletForward")
public class ServletForward extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ServletForward() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        RequestDispatcher dispatcher = request.getRequestDispatcher("/forward.jsp");
        dispatcher.forward(request, response);
    }

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

}
```

MVC2 제작

- MVC2 제작