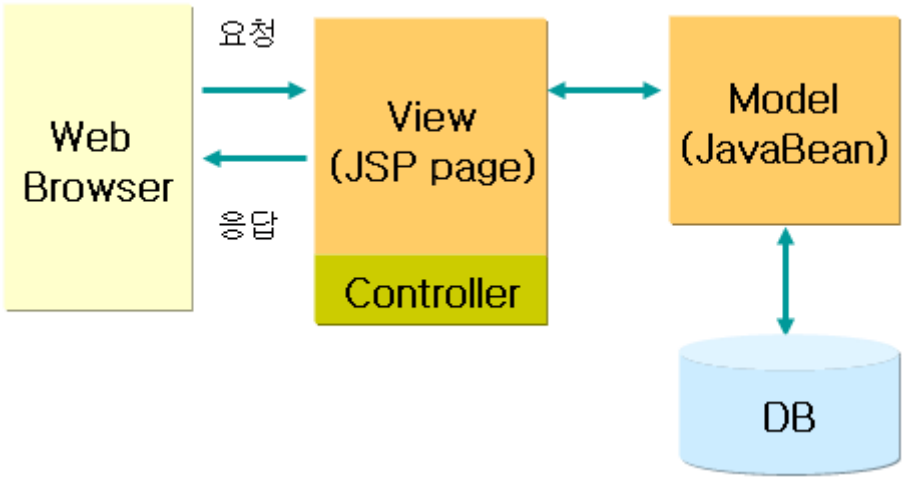
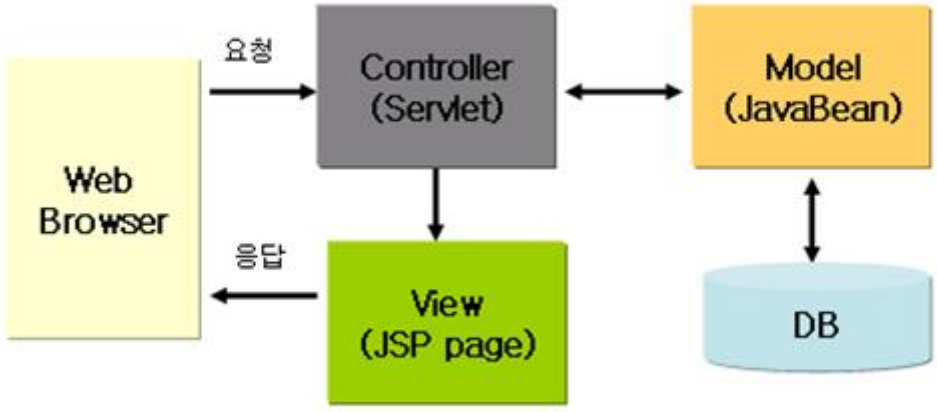
1. 모델 1

: 모델 1(Model 1)구조에서는, 웹 브라우저의 요청(request)을 받아들이고 , 웹 브라우저에 응답(response) 해주는 처리에 대해 JSP page 단독으로 처리하는 구조이다.



2. 모델 2

: 모델 2(Model 2)구조에서는 요청(request)처리, 데이터접근(data access), 비즈니스 로직(business logic)을 포함하고 있는 컨트롤 컴포넌트(control component)와 뷰 컴포넌트(view component)는 엄격히 구분되어져 있다.



3. MVC 패턴(Model-View-Controller pattern)

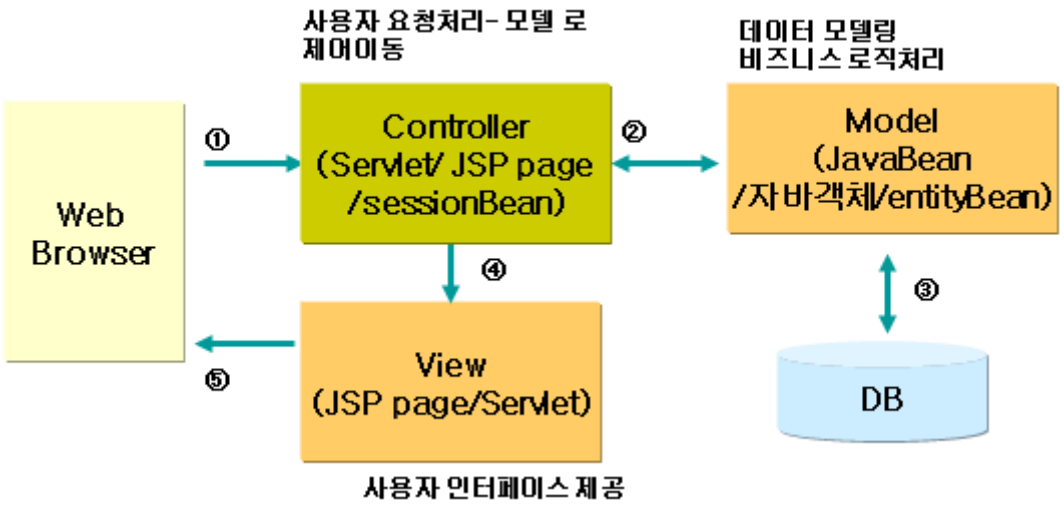
: **MVC(Model-View-Controller)** 구조는 전통적인 GUI(Graphic User interface) 기반의 어플리케이션을 구현하기 위한 디자인 패턴이다. MVC 구조는 사용자의

입력을 받아서, 그 입력 혹은 이벤트에 대한 처리를 하고, 그 결과를 다시 사용자에게 표시하기 위한 최적화된 설계를 제시한다.

1) **뷰(View)**는 화면에 내용을 표출하는 역할을 담당하는 것으로 데이터가 어떻게 생성되고, 어디서 왔는지에 전혀 관여하지 않는다. 단지 정보를 보여주는 역할만을 담당한다. JSP 기반의 웹 어플리케이션에서는 JSP 페이지가 뷰(View)에 해당한다.

2) **모델(Model)**은 로직을 가지는 부분으로 DB 와의 연동을 통해서 데이터를 가져와 어떤 작업을 처리하거나, 처리한 작업의 결과를 데이터로서 DB 에 저장하는 일을 처리한다. 모델(Model)은 어플리케이션의 수행에 필요한 데이터를 모델링하고 비즈니스 로직을 처리한다. 즉, 데이터를 생성하고 저장하고 처리하는 역할만을 담당한다. JSP 기반의 웹 어플리케이션에서는 자바빈(JavaBean)이 모델(Model)에 해당한다.

3) **컨트롤러(Controller)**는 어플리케이션의 흐름을 제어하는 것으로 뷰(View)와 모델(Model)사이에서 이들의 흐름을 제어한다. 컨트롤러(Controller)는 사용자의 요청을 받아서 모델(Model)에 넘겨주고, 모델(Model)이 처리한 작업의 결과를 뷰(View)에 보내주는 역할을 한다. JSP 기반의 웹 어플리케이션에서는 보통 서블릿(Servlet)을 컨트롤러(Controller)로 사용한다.



4. JSP 기반의 웹 어플리케이션에서 **Controller** 에 포함되어야 할 작업

1) 웹 브라우저(클라이언트)의 요청을 받는다.

2) 웹 브라우저(클라이언트)가 요구하는 작업을 분석한다.

3) 요청한 작업을 처리하기 위해서 비즈니스 로직을 처리하는 모델(Model)(JavaBean)을 사용한다.

4) 처리결과를 request 또는 session 의 속성에 저장한다.

5) 적당한 뷰(View)(JSP 페이지)를 선택 후 해당 뷰(View)로 포워딩(forwarding)한다.

5. JSP 기반의 웹 어플리케이션에서 **View** 에 포함되어야 할 작업

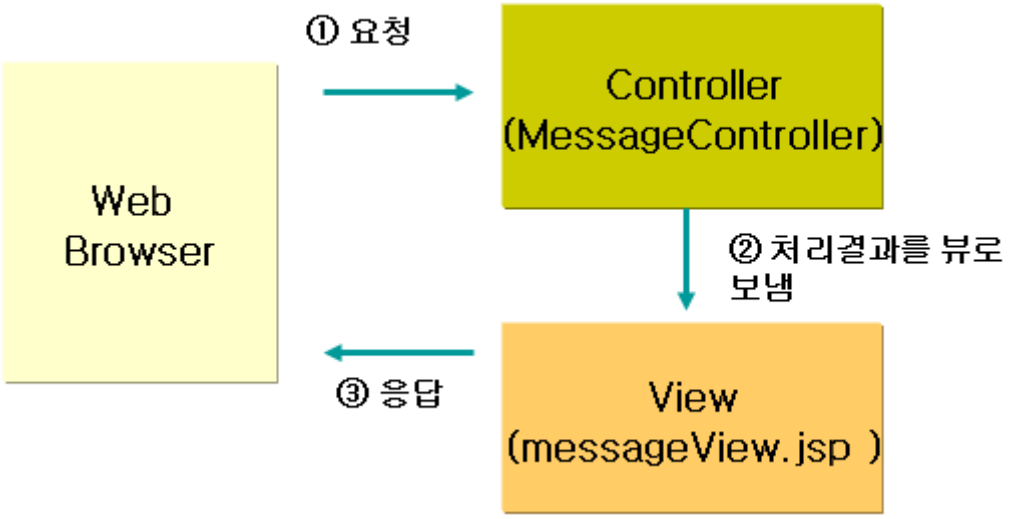
: 서블릿에서 dispacher.forward(request,response)로 해당 JSP 페이지와 request, response 를 공유한 경우 해당 JSP 페이지에서 request.getAttribute("result")와 같이 사용해서 결과를 화면에 표출한다.

6. JSP 기반의 웹 어플리케이션에서 **Model** 에 포함되어야 할 작업

1) 컨트롤러(Controller)의 요청을 받는다.

2) 비즈니스 로직을 처리한다.

3) 처리한 비즈니스 로직의 결과를 컨트롤러(Controller)로 반환한다.



7. 요청 URI 자체를 명령어로 사용하는 방법