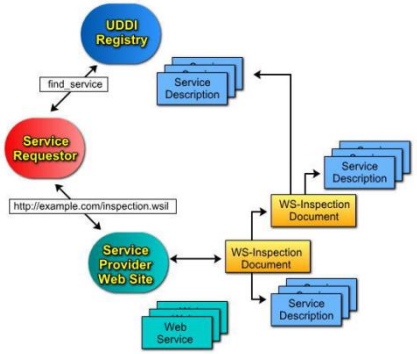
Web Application & Service

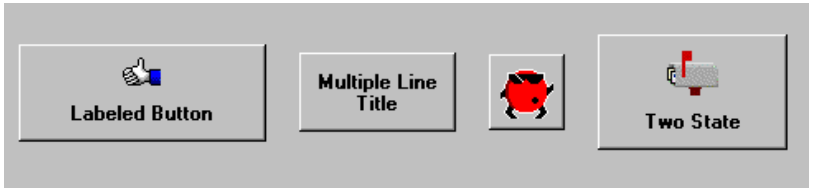


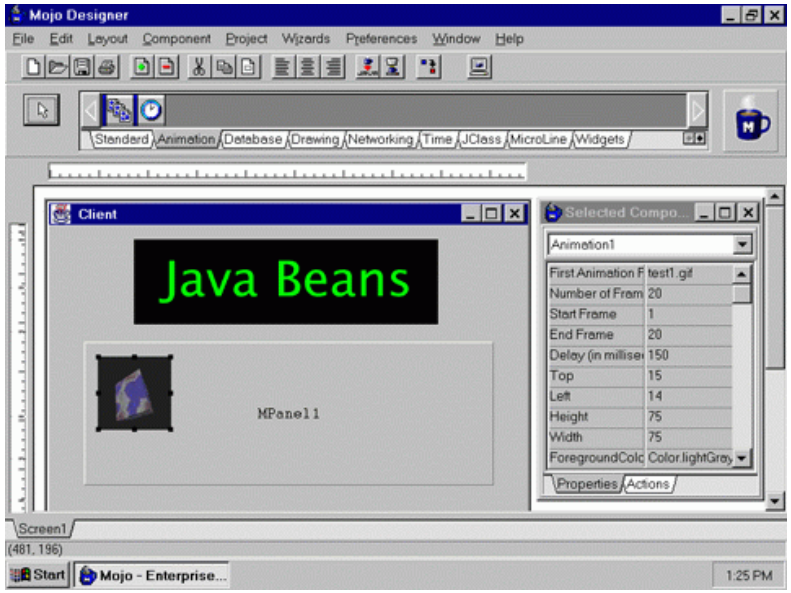
JSP Beans & MVC pattern

컴퓨터 과학과
김 희 열

JavaBeans

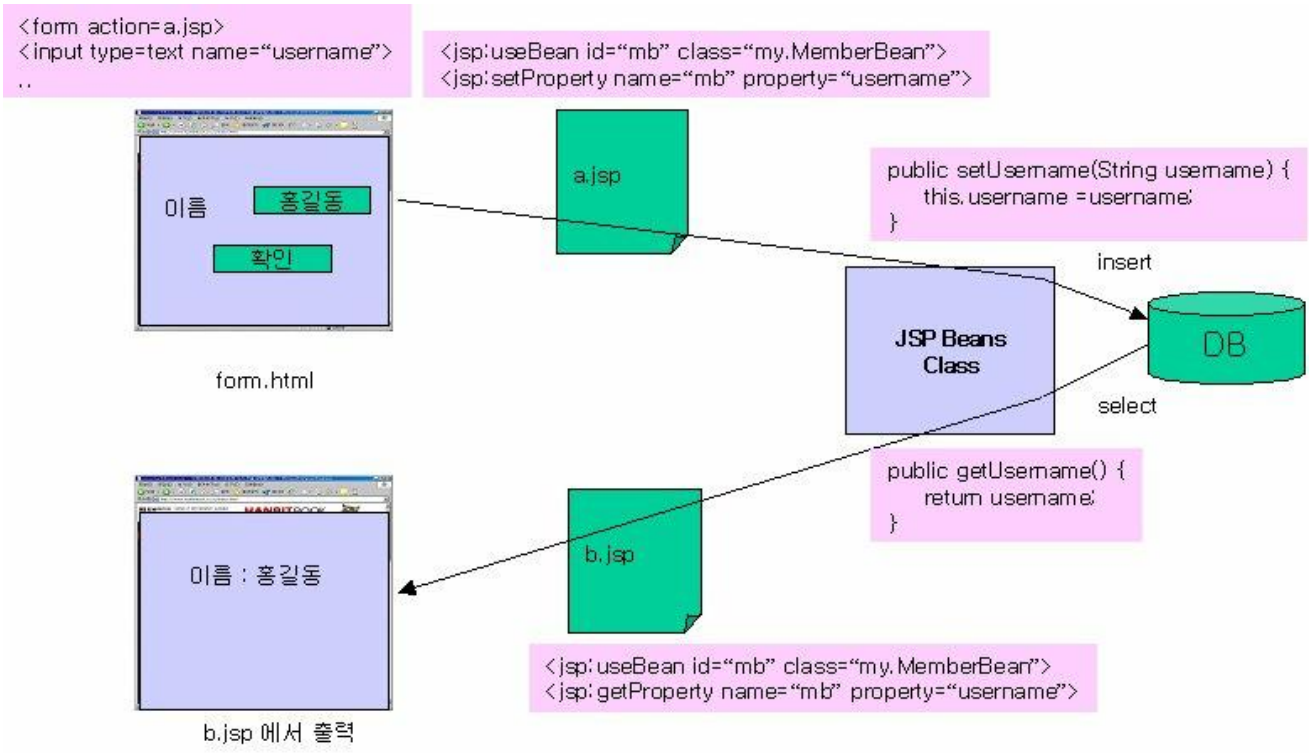
- A portable, platform-independent component model for java
- JavaBeans Components (Beans)
  - Self-contained, reusable software unit that can be visually assembled into composite components, applets, apps, and servlets using visual tools
  - Beans' properties can be dynamically changed or customized





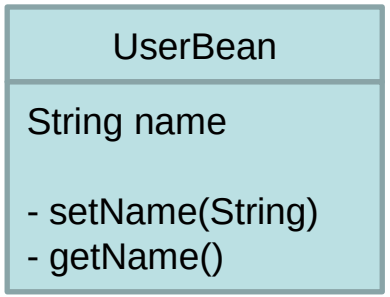
JSP Beans

- JSP와의 연동을 위해 만들어진 beans
- 비즈니스 로직과 프리젠테이션 로직을 분리
- JSP beans는 컨테이너에 위치
  - HTML form을 통한 입력 처리
  - DB등 backend와 연동 후 JSP에 결과 제공



Java Beans Class 구성

- 일반적인 Java class 구성과 유사하지만, Java Beans만의 규칙을 따름
- Class name
  - XxxBean 형식
  - Ex) UserBean
- Parameter가 없는 default constructor 제공
  - Ex) public UserBean()
- 멤버 변수
  - 클래스 내의 멤버 변수는 private으로 선언해서 direct access를 방지
  - Ex) private String name;
- getter/setter method
  - 각 멤버 변수의 access를 위해 getter/setter method 제공
  - 멤버 변수명에 기반하는 method를 생성해야 함
  - setter : set property for the bean
    - Ex) setName(String username)
  - getter : get property for the bean
    - Ex) getName()



JSP에서 Beans 사용

- Scriptlet 내에 java code를 통해 특정 bean 객체 접근이 가능하지만,
- JSP action을 이용해 bean 객체 사용 가능

- Bean 사용 선언
  - <jsp:useBean id="bean id" class="bean class" scope="범위" />
  - bean id
    - Bean의 인스턴스 명
  - bean class
    - Bean 클래스 명으로 패키지 경로 포함
  - 범위
    - Bean 객체는 해당 범위 내의 attribute로 관리됨
    - application : 웹 어플 종료시까지 사용
    - session : 현재 세션 종료시까지 사용
    - request : 현재 request가 처리 완료될 때까지 사용
    - page : 현재 페이지 내에서만 사용

JSP에서 Beans 사용

JSP

```
<jsp:useBean id="user" class="UserBean" scope="request">
```

Servlet

```
UserBean user = (UserBean)request.getAttribute("user");
if (user == null) {
        user = new UserBean();
        request.setAttribute("user", user);
}
```

UserBean
setName(String) getName()

해당 scope내에 user Bean이 이미 생성되어 관리되고 있으면 그 인스턴스 획득

새로운 인스턴스 생성 후 해당 scope에 맞게 attribute로 저장

JSP에서 Beans 사용

- Bean 객체의 property 획득
  - <jsp:getProperty name="bean id" property="property name" />
  - bean id
    - Bean 객체의 인스턴스 명
  - property name
    - 획득하려는 property 명
    - 내부에서는 getter()를 통해 획득

```
<HTML><body>
<% UserBean user = (UserBean)session.getAttribute("user"); %>
User name is: <%= user.getName() %>
</body></HTML>
```

```
<HTML><body>
<jsp:useBean id="user" class="UserBean" scope="session" />
User name is: <jsp:getProperty name="user" property="name" />
</body></HTML>
```

JSP에서 Beans 사용

- Bean 객체의 property 설정
  - <jsp:setProperty name="bean id" property="property name" value="값" />
  - bean id
    - Bean 객체의 인스턴스 명
  - property name
    - 설정하려는 property 명
    - 내부에서는 setter()를 통해 설정
  - 값
    - 설정하려는 value

```
<jsp:useBean id="user" class="UserBean" scope="session" />
<jsp:setProperty name="user" property="name" value="alice" />
```

VS.

```
<jsp:useBean id="user" class="UserBean" scope="session" >
        <jsp:setProperty name="user" property="name" value="alice" />
</jsp:useBean>
```

JSP에서 Beans 사용

- HTML form과 Bean 연결

UserBean
setName(String) getName() setPasswd(String) getPasswd()

```
<form action="testBean.jsp">
        name : <input type="text" name="name">
        password : <input type="password" name="passwd">
        <input type="submit">
</form>
```

```
<jsp:useBean id="user" class="UserBean" />
<% user.setName(request.getParameter("name"));
     user.setPasswd(request.getParameter("passwd")); %>
```

better

```
<jsp:useBean id="user" class="UserBean" />
<jsp:setProperty name="user" property="name" param="name" />
<jsp:setProperty name="user" property="passwd" param="passwd" />
```

JSP에서 Beans 사용

- HTML form과 Beans 연결

```
<form action="testBean.jsp">
        name : <input type="text" name="name">
        password : <input type="password" name="passwd">
        <input type="submit">
</form>
```

UserBean

setName(String)
getName()
setPasswd(String)
getPasswd()

better

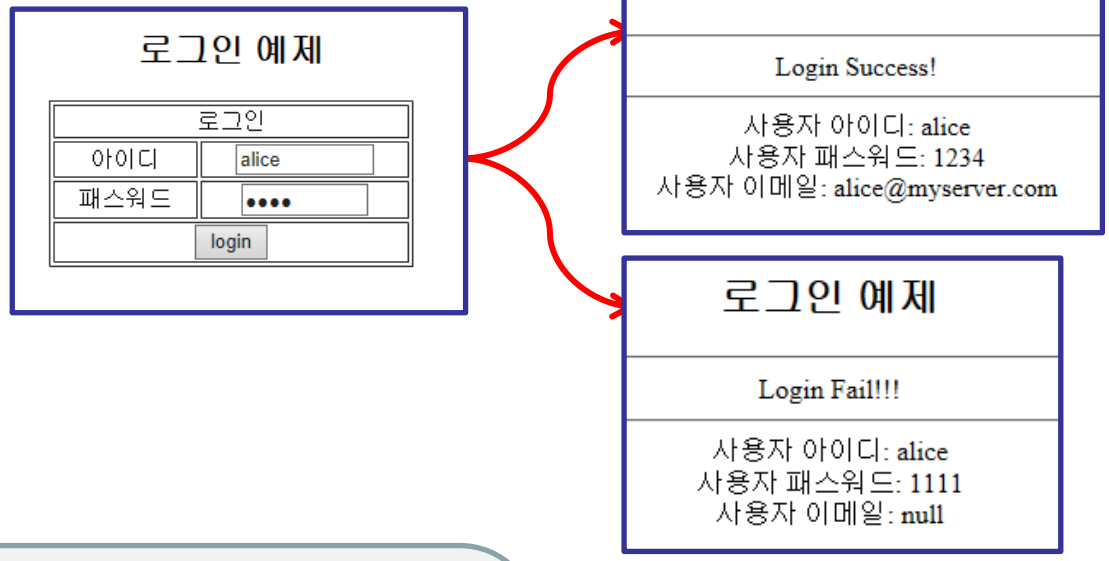
이름이 일치해야 함

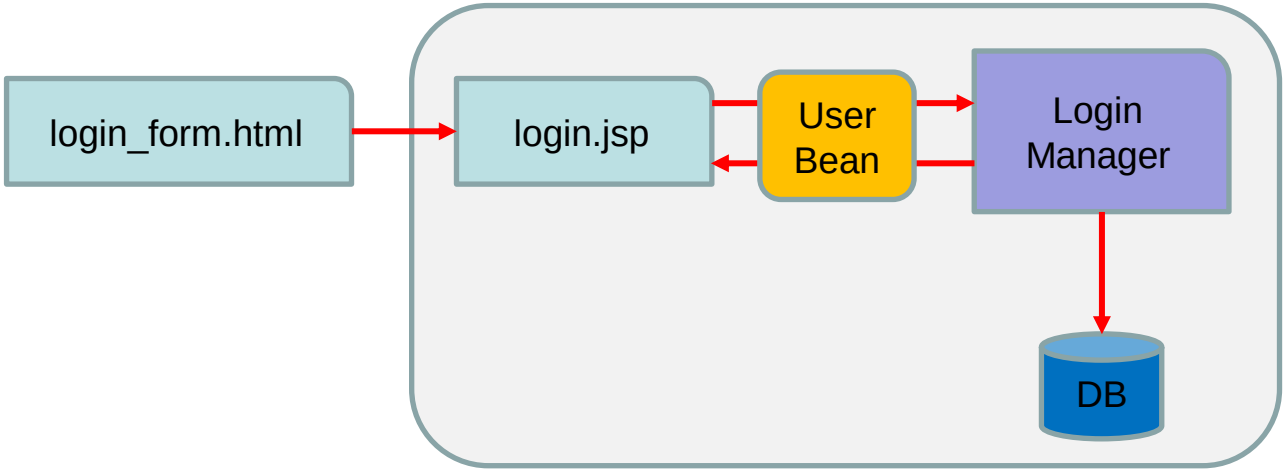
```
<jsp:useBean id="user" class="UserBean" />
<jsp:setProperty name="user" property="name" />
<jsp:setProperty name="user" property="passwd" />
```

better

```
<jsp:useBean id="user" class="UserBean" />
<jsp:setProperty name="user" property="*" />
```

JSP Beans 예제


로그인 예제
로그인
아이디
alice
패스워드
login
로그인 예제
Login Success!
사용자 아이디: alice
사용자 패스워드: 1234
사용자 이메일: alice@myserver.com
로그인 예제
Login Fail!!!
사용자 아이디: alice
사용자 패스워드: 1111
사용자 이메일: null


login_form.html
login.jsp
User Bean
Login Manager
DB

JSP Beans 예제

login_form.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<style>
    body {text-align:center;}
</style>
<title>Insert title here</title>
</head>
<body>
    <H2>로그인 예제</H2>
    <form method="post" action="/webservice/ch07/login.jsp">
        <table width="250" border="1" align="center">
        <tr>
            <td colspan="2">로그인</td>
        </tr>
        <tr>
            <td>아이디</td>
            <td><input type="text" name="userid" size="10"></td>
        </tr>
        <tr>
            <td>패스워드</td>
            <td><input type="password" name="passwd" size="10"></td>
        </tr>
        <tr>
            <td colspan="2">
                <input type="submit" name="Submit" value="login"></td>
        </tr>
        </table>
    </form>
</body>
</html>
```

JSP Beans 예제

UserBean.java

```java
package myapp.ch07;

public class UserBean {
    // member variables : property
    private String  userid;
    private String  passwd;
    private String  email;

    // getter/setter methods
    public String getUserid() {
        return userid;
    }
    public void setUserid(String userid) {
        this.userid = userid;
    }
    public String getPasswd() {
        return passwd;
    }
    public void setPasswd(String passwd) {
        this.passwd = passwd;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}
```

자동 생성

JSP Beans 예제

LoginManager.java

```
package myapp.ch07;

public class LoginManager {

    public boolean authenticate(UserBean user) {
        if (user.getUserid().equals("alice") && user.getPasswd().equals("1234")) {
            user.setEmail("alice@myserver.com");
            return true;
        }
        else return false;
    }
}
```

JSP Bea

login.jsp

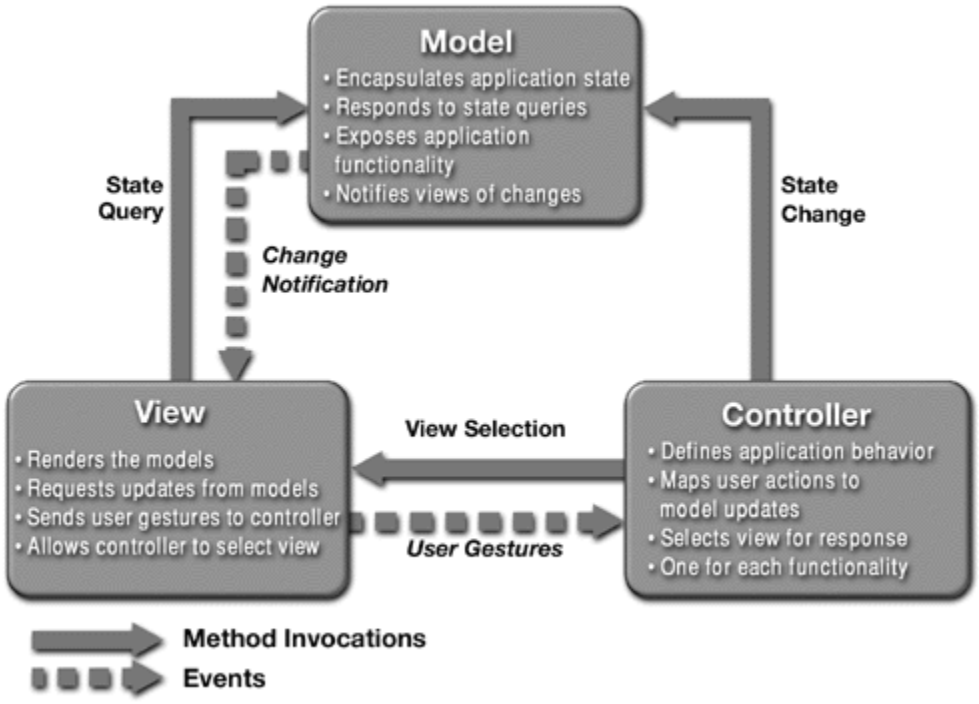
```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="myapp.ch07.LoginManager" %>
<!DOCTYPE html>

<jsp:useBean id="user" class="myapp.ch07.UserBean" scope="request" />
<jsp:setProperty name="user" property="*" />

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<style>
    body {text-align:center;}
</style>
<title>Insert title here</title>
</head>
<body>
    <H2>로그인 예제</H2>
    <HR>
    <%
        LoginManager loginManager = new LoginManager();
        if (loginManager.authenticate(user))
            out.println("Login Success!");
        else
            out.println("Login Fail!!!");
    %>
    <HR>
    사용자 아이디: <jsp:getProperty property="userid" name="user" />
    <BR>
    사용자 패스워드: <jsp:getProperty property="passwd" name="user"/>
    <BR>
    사용자 이메일: <jsp:getProperty property="email" name="user" />
</body>
</html>
```

MVC Pattern

- Model-View-Controller pattern
  - 객체간의 coupling 레벨을 줄임으로써 보다 유연한 data처리 가능
  - Web application에서 많이 사용되는 중요 패턴



Three Logical Layers

- Model (Business process layer)
  - Backend단에서의 비즈니스 로직 담당
  - data와 그에 대한 행동을 모델링
  - 실제 처리해야 하는 작업을 수행
    - DB와 연동해서 query 실행
    - Business process 수행
  - Presentation layer와 무관하게 data를 encapsulate

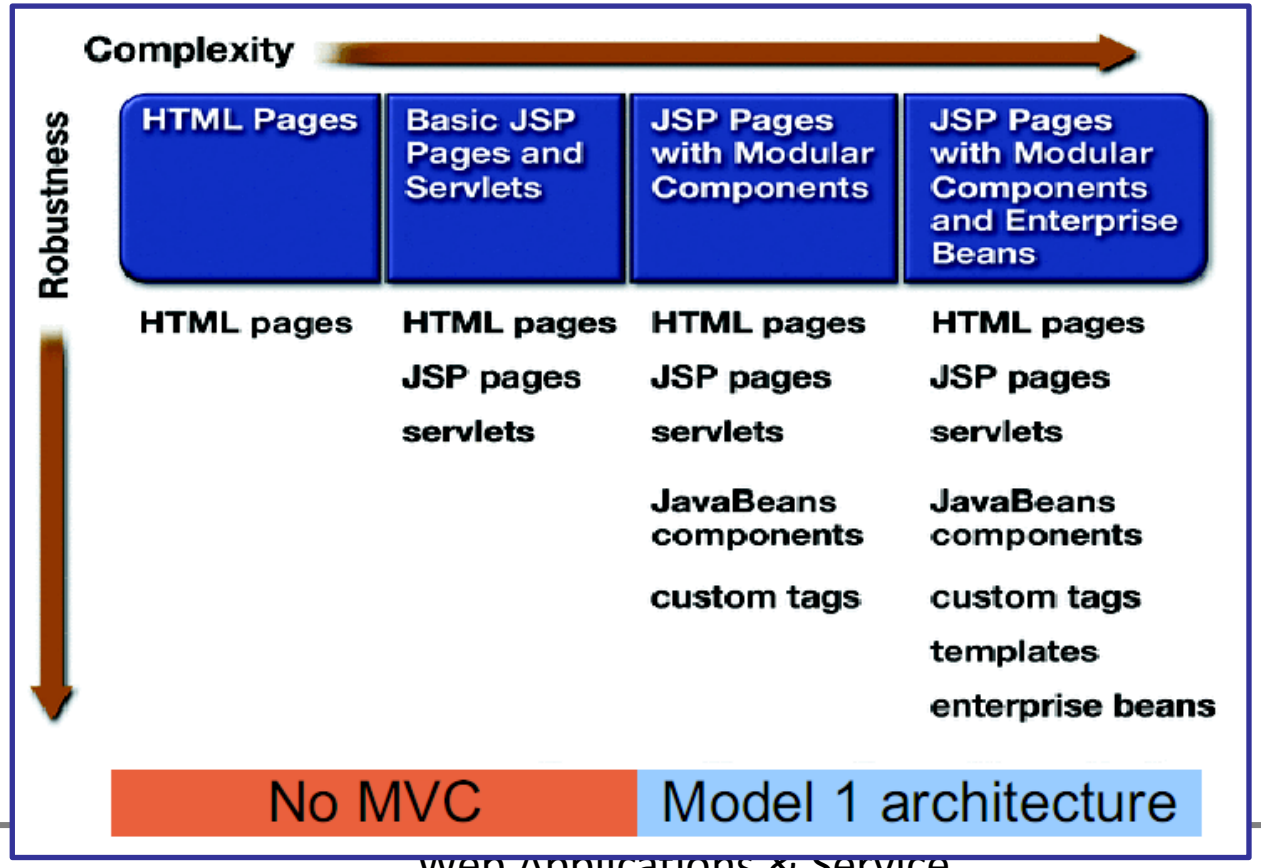
- View (Presentation layer)
  - 비즈니스 로직(Model)을 통해 얻어진 결과를 출력
  - 사용자 타입과 환경에 맞게 다양한 정보를 출력
  - 정보 자체가 어디로부터 왔는지, 어떻게 생성되었는지는 상관하지 않음

Three Logical Layers

- Controller (Control layer)
  - 사용자와 backend의 비즈니스 로직을 연결해 주는 역할
  - 여러 presentation중 어느 것을 제공할지 결정해 주는 역할
    - Ex) 사용자 언어, 위치, access level등에 따라 다른 presentation 제공
  - 사용자의 요청을 받은 후, 요청을 어떻게 처리하고 어떤 결과를 얻어서 표현해야 하는지를 결정

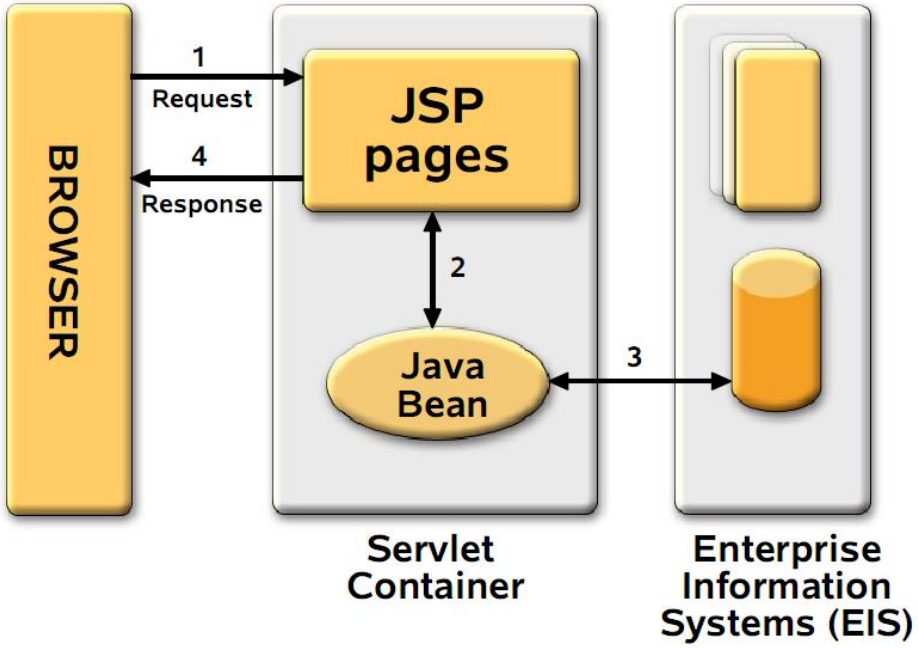
Evolution of Web Application Design Architecture

- No MVC
- MVC Model 1 (Page-centric)
- MVC Model 2 (Servlet-centric)
- Web application frameworks



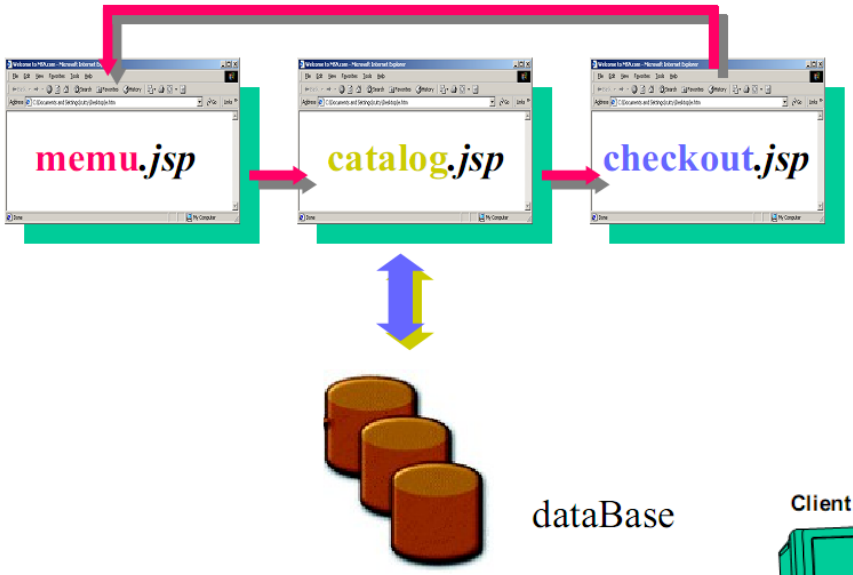
Evolution of Web Application Design Architecture

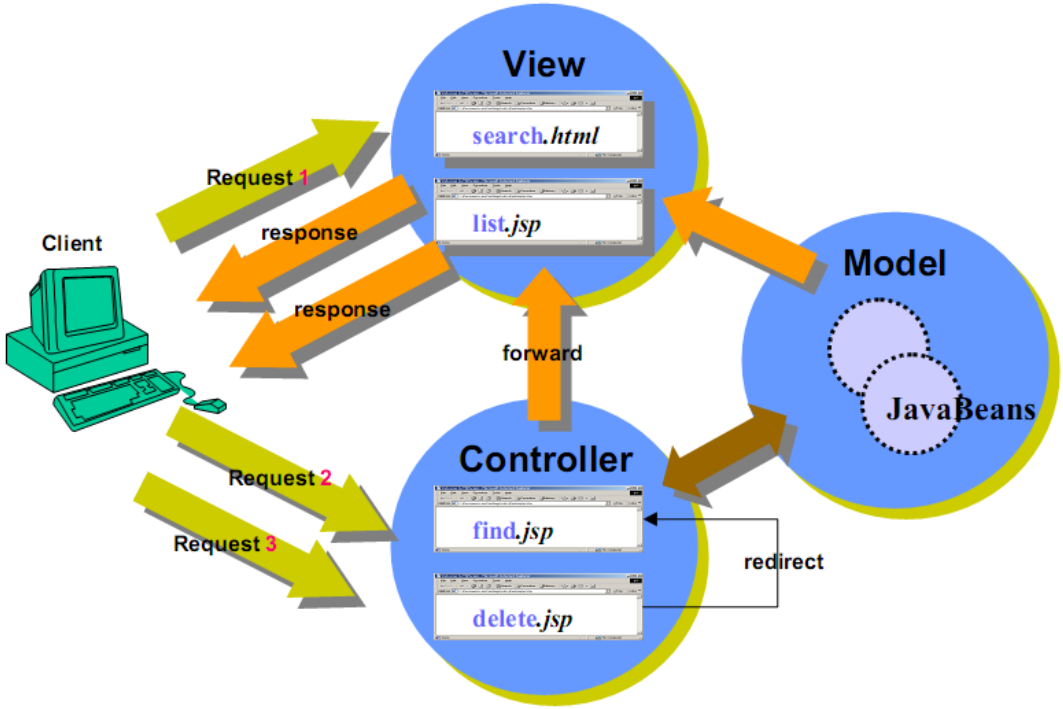
- Model 1 architecture
  - 서로 연결되어있는 JSP 페이지들로 구성
    - 각 JSP 페이지가 presentation, control, business logic을 모두 담당
    - 심지어 한 페이지가 모든 역할을 수행하기도 함
  - Business logic과 control도 모두 JSP 페이지에 하드코딩되어 있음
  - 주로 하이퍼링크와 폼을 통해 다음 JSP 페이지와 연결



Evolution of Web Application Design Architecture

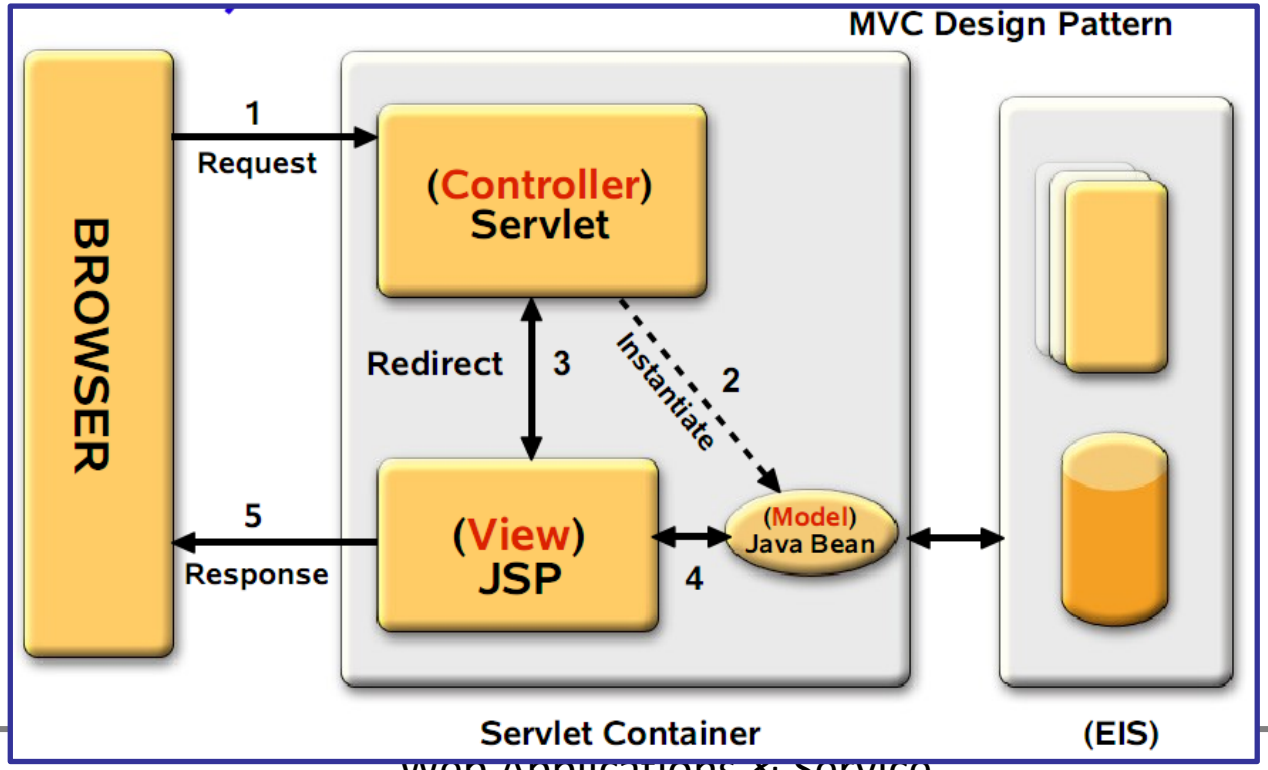
- Model 1 architecture





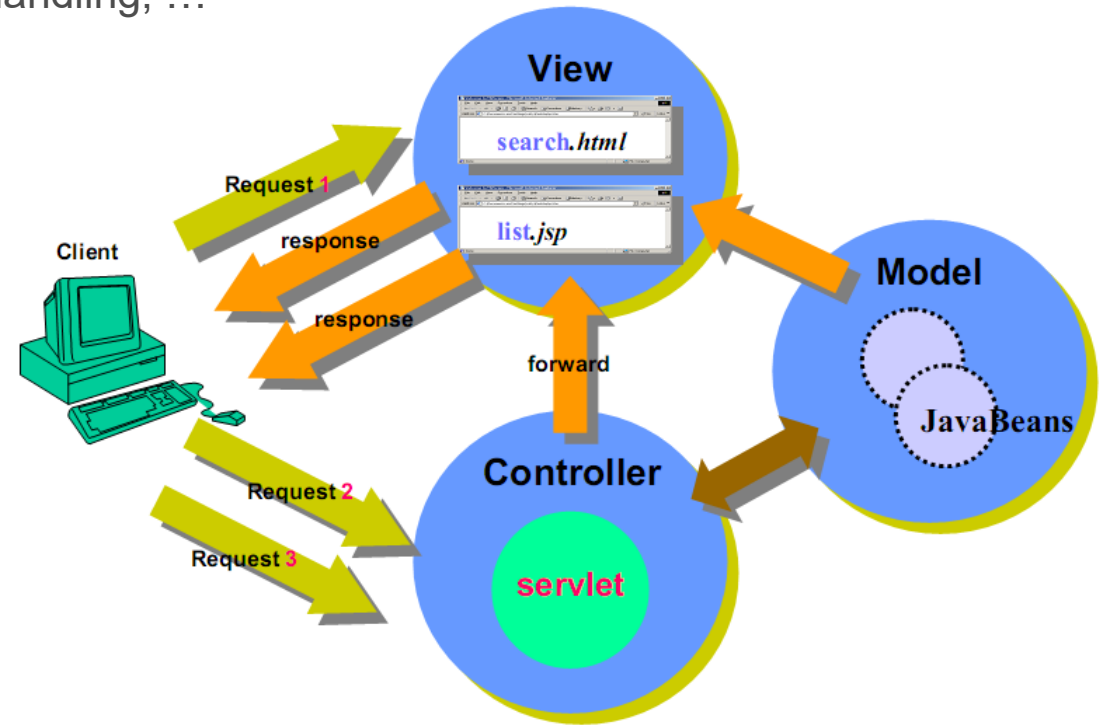
Evolution of Web Application Design Architecture

- Model 2 architecture
  - Servlet
    - Easy to control & manipulate logic & data
    - Hard to generate HTML
  - JSP
    - Easy to display presentation of dynamic contents



Evolution of Web Application Design Architecture

- Model 2 architecture
  - JSP pages
    - Used only for presentation
  - Servlet
    - Control and handle application logic for the request
    - Authentication, error handling, …



Evolution of Web Application Design Architecture

- Web application frameworks
  - Based on MVC Model 2 architecture
  - 많은 web app에서 필요로 하는 공통 기능 제공
    - DB access
    - Template system
    - Authentication/authorization framework
    - Ajax framework
    - …
  - Reusable한 component를 보다 쉽게 제작하는 방법 제공

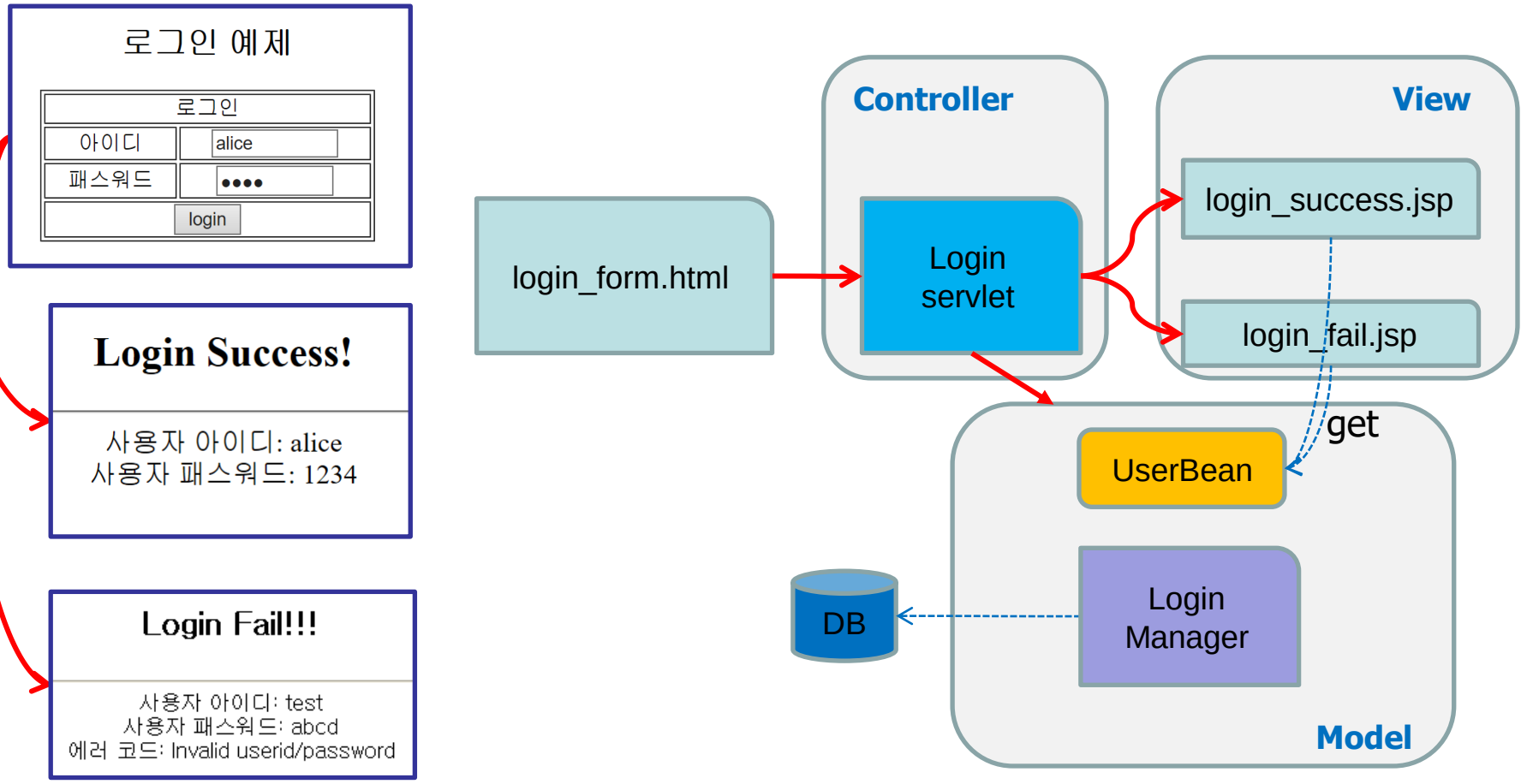
  - Apache Struts / Struts2
  - Spring framework
  - Ruby on Rails
  - JavaServer Faces (JSR-127)
  - …

Implementing MVC Pattern

- Separate business logic & data access layers from presentation layer

1. Define beans to represent the data
2. Use a servlet to handle requests
   - Read request parameters
3. Populate the beans
   - Servlet invokes business logic or data access code
   - The results are placed in the beans in step 1
4. Store the bean in the request, session, or servlet context
   - Store as an attribute
5. Forward the request to a JSP page
   - Servlet determines which JSP is appropriate
6. Extract the data from the beans
   - JSP access beans & gets properties
   - JSP merely extracts & displays data the servlet created

MVC Pattern 예제



MVC Pattern 예제

login_form.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<style>
    body {text-align:center;}
</style>
<title>Insert title here</title>
</head>
<body>
    <H2>로그인 예제</H2>
    <form method="post" action="/webservice/Login">
        <table width="250" border="1" align="center">
        <tr>
            <td colspan="2">로그인</td>
        </tr>
        <tr>
            <td>아이디</td>
            <td><input type="text" name="userid" size="10"></td>
        </tr>
        <tr>
            <td>패스워드</td>
            <td><input type="password" name="passwd" size="10"></td>
        </tr>
        <tr>
            <td colspan="2">
                <input type="submit" name="Submit" value="login"></td>
        </tr>
        </table>
    </form>
</body>
</html>
```

MVC Pattern 예제

UserBean.java

```
package myapp.ch07;

public class UserBean {

    private String  userid;
    private String  passwd;
    private String  email;
    private String  error;

    public String getUserid() {
        return userid;
    }
    public void setUserid(String userid) {
        this.userid = userid;
    }
    public String getPasswd() {
        return passwd;
    }
    public void setPasswd(String passwd) {
        this.passwd = passwd;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getError() {
        return error;
    }
    public void setError(String error) {
        this.error = error;
    }
}
```

MVC Pattern 예제

Login.java

```
package myapp.ch07;

import java.io.IOException;

@WebServlet("/Login")
public class Login extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) t
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        LoginManager loginMgr = new LoginManager();
        UserBean userBean = new UserBean();
        String addr;

        userBean.setUserid(request.getParameter("userid"));
        userBean.setPasswd(request.getParameter("passwd"));
        if (loginMgr.authenticate(userBean)) addr = "/ch07/login_success.jsp";
        else addr = "/ch07/login_fail.jsp";

        request.setAttribute("userinfo", userBean);
        RequestDispatcher dispatcher = request.getRequestDispatcher(addr);
        dispatcher.forward(request, response);
    }
}
```

MVC Pattern 예제

login_success.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.

<jsp:useBean id="userinfo" class="myapp.ch07.LoginBean" scope="request" />

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<style>
    body {text-align:center;}
</style>
<title>Insert title here</title>
</head>
<body>
    <H2> Login Success!</H2>
    <HR>
    사용자 아이디: <jsp:getProperty name="userinfo" property="userid" />
    <BR>
    사용자 패스워드: <jsp:getProperty name="userinfo" property="passwd" />
</body>
</html>
```

MVC Pattern 예제

login_fail.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.

<jsp:useBean id="userinfo" class="myapp.ch07.LoginBean" scope="request" />

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<style>
    body {text-align:center;}
</style>
<title>Insert title here</title>
</head>
<body>
    <H2> Login Fail!</H2>
    <HR>
    사용자 아이디: <jsp:getProperty name="userinfo" property="userid" />
    <BR>
    사용자 패스워드: <jsp:getProperty name="userinfo" property="passwd" />
    <BR>
    에러 코드: <jsp:getProperty name="userinfo" property="error" />
</body>
</html>
```