MVC모델

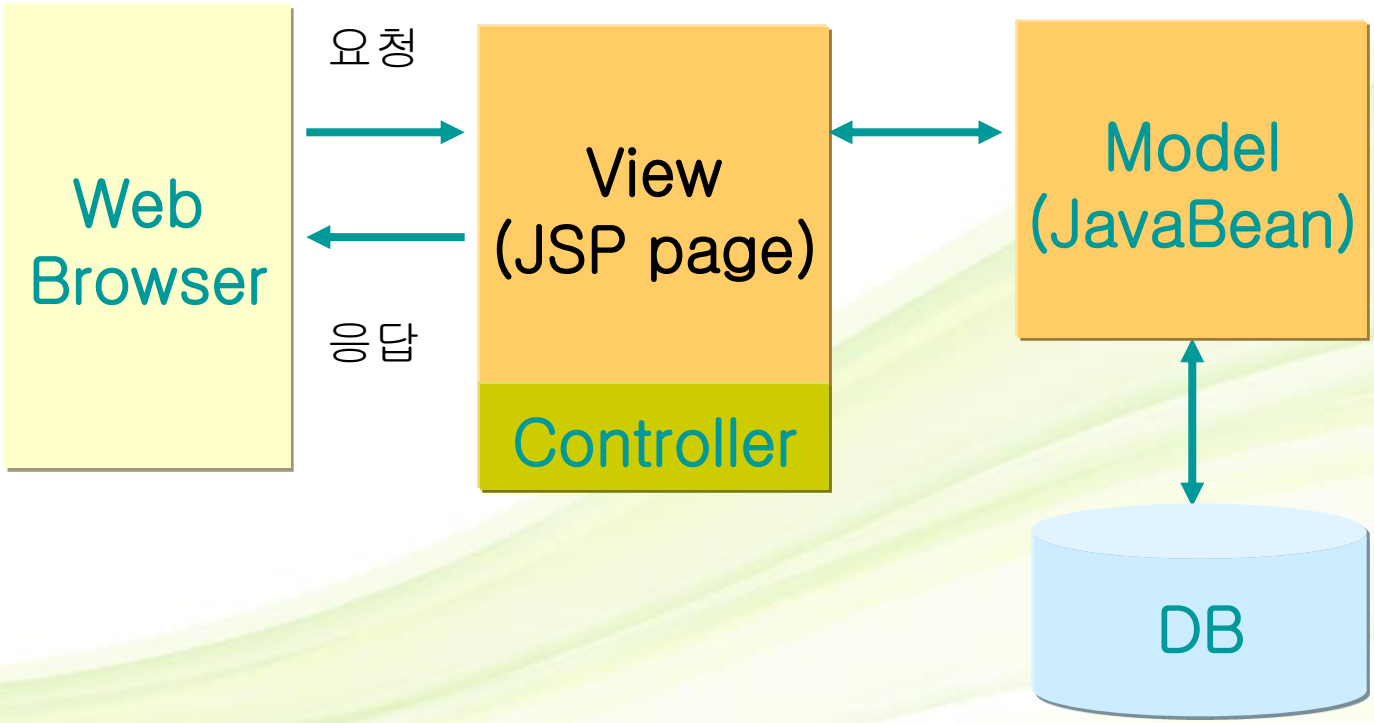
설계 모델과 웹 템플릿에 대하여

❖ JSP와 서블릿 기반의 설계 모델 – 모델 1과 모델 2

- JSP 규격서의 초기 버전에서 소개하고 있는 설계 모델(design model)인 모델 1과 모델 2는 웹 애플리케이션이 해야 할 일을 다음 세가지로 구분하여 모듈화하는 방법을 제시하고 있다.
  - 데이터 입력
  - 데이터 처리
  - 데이터 출력
- 모델 1은 비교적 간단한 웹 애플리케이션에 적합한 설계 모델이고, 모델 2는 비교적 복잡한 웹 애플리케이션에 적합한 설계 모델이다.

모델1(Model 1)

- 모델1(Model 1)구조에서는, 웹 브라우저의 요청(request)을 받아들이고 , 웹 브라우저에 응답(response) 해주는 처리에 대해 JSP page 단독으로 처리하는 구조이다.



모델 1 : JSP개발 초기에 주로 사용하던 것

check_form.jsp

```
<%@ page contentType="text/html;charset=euc-kr" %>
<HTML><HEAD><TITLE>로그인 </TITLE></HEAD>
<center>
<BODY>
<H2> 로그인 </H2>
<form name=form1 method=post action=passwd_check.jsp>
<table cellpadding=5 cellspacing=0 border="1">
  <tr>
    <td bgcolor="#99CCFF">비밀번호를 입력하세요</td>
    <td><input type="password" name="passwd" size="20"></td>
  </tr>
    <td colspan=2 align=center><input type=submit value="확인"> </td>
</tr></table></form></center>
</BODY></HTML>
```

password_check.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="service.IdCheck"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Insert title here</title></head><body>
<%  String id = request.getParameter("id");
      String pass = request.getParameter("pass");
      IdCheck ic = new IdCheck();
      int result = ic.check(id, pass);
      if (result==1) {
%>
<h2>메뉴</h2>
1. 환경보호<p>
2. 세계평화<p>
3. 국가발전<p>
<% }  else { %>
<h2>로그인 실패</h2>
아이디나 암호가 다릅니다<p>
<a href="check.jsp">다시 시도</a>
<% } %>
</body>
</html>
```

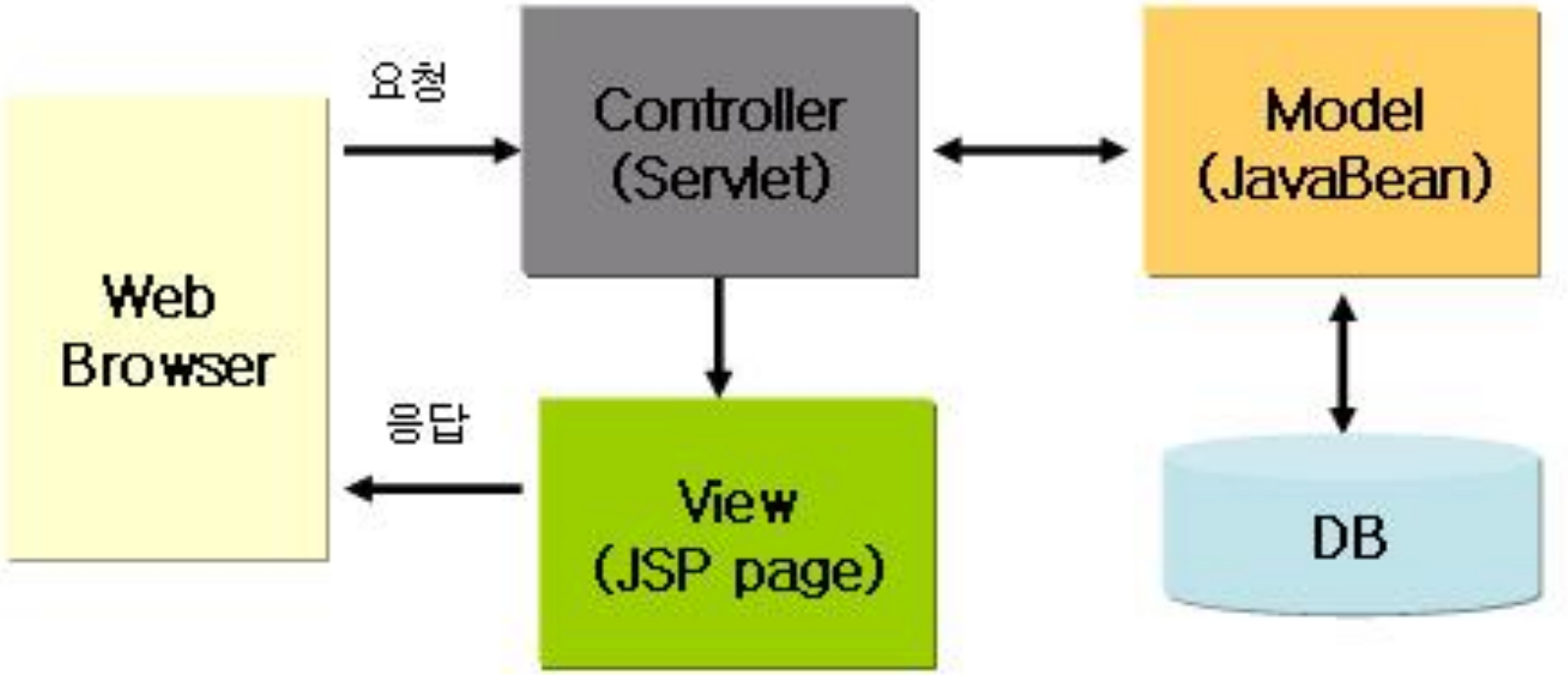
IdCheck.java

```
package service;

public class IdCheck {
    public int check(String id, String pass) {
       int result = 0;
       if (id.equals("kk") && pass.equals("1234")) {
            result = 1;
       }
       return result;
    }
}
```

모델2(Model 2)

- 모델2(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



JSP모델 2 : MVC(Model View-Controler)
LoginCheck2.jsp

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/LoginCheck2")
public class LoginCheck2 extends HttpServlet {
   protected void doGet(HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException {
          String id = request.getParameter("id");
          String pass = request.getParameter("pass");
          if (id.equals("kk") && pass.equals("1234")) {
                        response.sendRedirect("loginSuccess.jsp");
          } else {
                        response.sendRedirect("loginFail.jsp");
          }
   }
  protected void doPost(HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException {
  }
}
```

admin_menu.jsp

```
<%@ page contentType="text/html;charset=euc-kr" %>
<HTML>
<HEAD><TITLE>비밀번호 확인</TITLE></HEAD>
<center>
<BODY>
 <H2> 관리자 메뉴 </H2>
 <HR>
   1. <a href=>사용자 추가</a><p>
   2. <a href=>비밀번호 변경</a><p>
   3. <a href=>리포트 보기</a><p>
  </center>
</BODY>
</HTML>
```

error.jsp

```
<%@ page contentType="text/html;charset=euc-kr" %>
 <HTML>  <HEAD><TITLE>관리자 메뉴</TITLE></HEAD>
 <center>
 <BODY>
 <H2>비밀번호가 잘못되었습니다.</H2>
 <a href=check_form2.jsp>돌아가기</a>
 </center>
</BODY>
</HTML>
```

모델1 vs 모델2

- 어플리케이션의 복잡도(규모)에 따라, 유지보수의 빈도에 따라, 어플리케이션 컴포넌트의 재사용성에 따라 그리고 팀원의 수와 수준에 따라 결정해야 한다.

- 개발하려는 웹 어플리케이션의 복잡도(규모)가 적고, 유지보수가 빈번하지 않다면 모델1(Model 1)의 구조로 개발하는 것이 적합하며, 반대로 웹 어플리케이션의 복잡도(규모)가 크고, 유지보수가 빈번하게 발생한다면 모델2(Model 2)구조를 사용하는 것이 적합할 것이다. 반드시 무엇이 정답이라고 할 수없다. 상황 따라 맞는 것을 선택하는 것이 가장 좋은 방법이다.

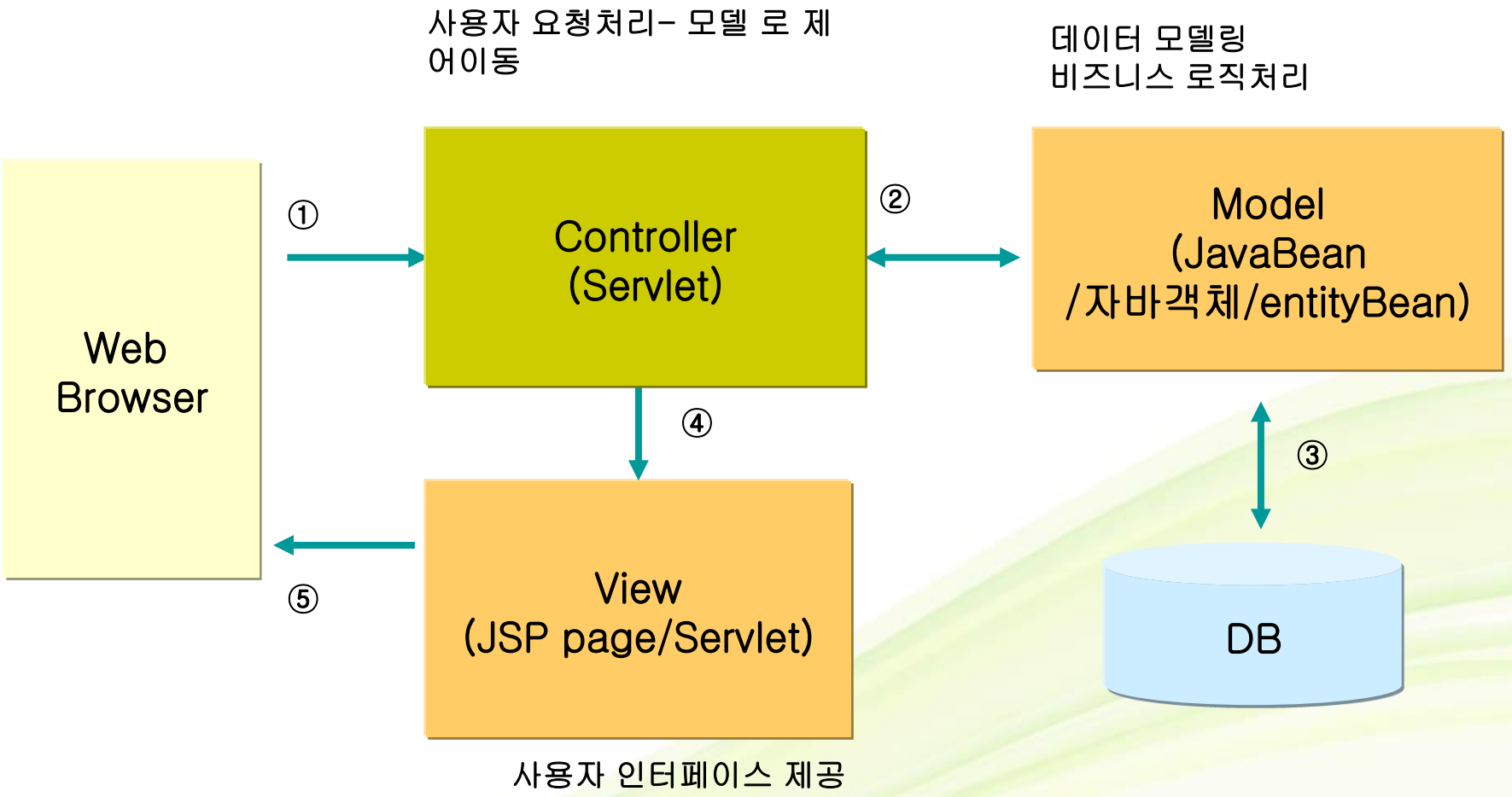
MVC패턴(Model-View-Controller pattern)

- 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.
- 뷰(View)는 화면에 내용을 표출하는 역할을 담당하는 것으로 데이터가 어떻게 생성되고, 어디서 왔는지에 전혀 관여하지 않는다. 단지 정보를 보여주는 역할만을 담당한다. JSP기반의 웹 어플리케이션에서는 JSP페이지가 뷰(View)에 해당한다.
- 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. JSP기반의 웹 어플리케이션에서는 보통 서블릿(Servlet)을 컨트롤러(Controller)로 사용한다.

MVC패턴(Model-View-Controller pattern)

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

MVC패턴(Model-View-Controller pattern)



Controller에 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 request또는 session의 속성에 저장한다.
5. 적당한 뷰(View)(JSP페이지)를 선택 후 해당 뷰(View)로 포워딩(forwarding)한다.
6. 서블릿에서 dispacher.forward(request,response)로 해당 JSP페이지와 request, response를 공유한 경우 해당 JSP페이지에서 request.getAttribute("result")와 같이 사용해서 결과를 화면에 표출한다.

작업순서

1. 컨트롤러(Controller)의 요청을 받는다.
2. 비즈니스 로직을 처리한다.
3. 처리한 비즈니스 로직의 결과를 컨트롤러(Controller)로 반환한다.

MVC를 이용한 자료실

- Dynamic Web Project 생성
- DBCP 사용을 위한 jar 파일을 lib 폴더에 복사
- 파일 업로드를 위한 jar 파일을 복사
- 프로젝트에 upload 폴더 추가
- context.xml 파일을 작성해서 프로젝트에 추가

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Resource name="DBConn"
   auth="Container"
   type="javax.sql.DataSource"
   username="system"
   password="wnddkd"
   driverClassName="oracle.jdbc.driver.OracleDriver"
   factory="org.apache.tomcat.dbcp.dbcp2.BasicDataSourceFactory"
   url="jdbc:oracle:thin:@127.0.0.1:1521:xe"
   maxActive="20"
   maxIdle="30"/>
</Context>
```

cos.jar를 사용

- cos.jar 파일 다운로드 http://www.servlets.com
- cos-05Nov.zip파일의 압축 해제
  - cos\lib폴더 안에 있는 cos.jar 파일을 lib폴더에 복사

upload 폴더 추가 ; "/upload

web.xml 파일을 작성해서 프로젝트에 추가

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"  version="3.0">
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
```

- src 폴더에 controller 패키지를 생성하고 제어를 담당할 FileUploadServlet 서블릿을 생성
- web.xml 파일에 action이 붙은 주소를 입력하면 위에서 생성한 서블릿으로 이동하도록 작성

```
<servlet>
  <servlet-name>UploadServlet</servlet-name>
  <servlet-class>controller.FileUploadServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>UploadServlet</servlet-name>
  <url-pattern>*.action</url-pattern>
 </servlet-mapping>
```

테이블 생성

```
create table pds_item (
    id              number          NOT NULL,
    fileName        VARCHAR2(100)   NOT NULL,
    fileSize        NUMBER,
    description     VARCHAR2(255)
);
```

src 폴더에 dto 패키지를 생성하고 PdsItem클래스를 생성

```
package dto;
public class PdsItem {
   private int id;              private String fileName;
   private long fileSize;    private String description;

   public int getId() {  return id;  }
   public void setId(int id) {    this.id = id;  }
   public String getFileName() {  return fileName;   }
   public void setFileName(String fileName) {
       this.fileName = fileName;
    }
   public long getFileSize() {  return fileSize;  }
   public void setFileSize(long fileSize) {
      this.fileSize = fileSize;
  }
  public String getDescription() {  return description;   }
  public void setDescription(String description) {
      this.description = description;
  }
}
```

- src 폴더에 dao 패키지를 생성하고 테이블에 접근해서 데이터베이스 작업을 수행하는 PdsItemDao 클래스를 생성
- src 폴더에 service 패키지를 생성
- 파일 업로드 폼을 작성(uploadForm.jsp)

```
<%@ page contentType="text/html; charset=utf-8" %>
<html><head><title>파일 등록</title>
    <style type="text/css">span { color: red; }</style>
    <script type="text/javascript">
      function chk() {
         var file = document.getElementById("file");
         var fileSpan = document.getElementById("fileSpan");
         if (file == null || file.value.length == 0) {
              fileSpan.innerHTML = "파일을 선택해야 합니다.";
              return false;
          } else {
              fileSpan.innerHTML = "";
              return true;
         }
   }
</script>
</head>
```

```
<body>
<h1>파일 저장하기</h1>
    <form action="fileUpload.action" method="post" enctype="multipart/form-data"
        onsubmit="return chk()">

        파일: <input type="file" name="file" id="file" />
              <span id="fileSpan"></span><p>

        설명: <input type="text" name="description" /> <p>

        <input type="submit" value="업로드" /> <p>

        <a href="list.action">목록보기</a>

    </form>
</body>
</html>
```

PdsItemDao 클래스에 데이터베이스 삽입 작업을 위한 변수와 메서드를 작성

```
package dao;                    import java.sql.*;
import java.util.ArrayList;     import javax.sql.*;  import javax.naming.*;
public class PdsItemDao {
        private static PdsItemDao instance = null;
        private PdsItemDao() {}
        public static PdsItemDao getInstance() {
                if (instance == null) {         instance = new PdsItemDao();         }
                return instance;
        }
        Connection getConnection() {
                Connection conn = null;
                try {
                        Context init = new InitialContext();
                        DataSource ds = (DataSource)
                                init.lookup("java:comp/env/jdbc/OracleDB");
                        conn = ds.getConnection();
                } catch(Exception e) { System.out.println(e.getMessage());         }
                return conn;
        }
```

```
public int insert(PdsItem item) {
        int result = 0;  Connection conn = null;  int num = 0;
        String sql1 = "select max(id) from pds_item";
        String sql2 = "insert into pds_item values (?,?,?,?)";
        try { conn = getConnection();
                PreparedStatement pstmt = conn.prepareStatement(sql1);
                ResultSet rs = pstmt.executeQuery();
                if (rs.next())    {                num = rs.getInt(1);  }
                num++;
                rs.close();    pstmt.close();
                pstmt = conn.prepareStatement(sql2);
                pstmt.setInt(1, num);
                pstmt.setString(2, item.getFileName());
                pstmt.setLong(3, item.getFileSize());
                pstmt.setString(4, item.getDescription());
                result = pstmt.executeUpdate();
        }catch(Exception e) { System.out.println(e.getMessage()); }
        return result;
}
```

```
    public ArrayList<PdsItem> list() {
        Connection conn = null;
        String sql = "select * from pds_item";
        ArrayList<PdsItem> al = new ArrayList<PdsItem>();
        try {   conn = getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                do { PdsItem pi = new PdsItem();
                    pi.setId(rs.getInt("id"));
                    pi.setFileName(rs.getString("filename"));
                    pi.setFileSize(rs.getLong("filesize"));
                    pi.setDescription(rs.getString("description"));
                    al.add(pi);
                 } while(rs.next());
                rs.close();
            }
            pstmt.close();      conn.close();
        }catch(Exception e) { System.out.println(e.getMessage()); }
        return al;
    }
}
```

service 패키지에 데이터를 삽입하는 FileInsert 클래스

```
package service;
import java.io.File;
import java.io.UnsupportedEncodingException;
import java.util.Enumeration;
import javax.servlet.http.HttpServletRequest;
import com.oreilly.servlet.MultipartRequest;
import com.oreilly.servlet.multipart.DefaultFileRenamePolicy;
import dao.PdsItem;
import dao.PdsItemDao;
public class FileInsert {
	private static FileInsert instance = null;
	private FileInsert() { }
	public static FileInsert getInstance() {
		if (instance == null) { instance = new FileInsert(); }
		return instance;
	}
```

```
public int addFile(HttpServletRequest request) throws
            UnsupportedEncodingException {
    request.setCharacterEncoding("utf-8");
    String real= request.getSession().getServletContext().getRealPath("/upload");
    int fileSize = 10*1024*1024; String fileName="";  int b = 0;
    MultipartRequest multi = null;
    try {  multi = new MultipartRequest(request, real,
                    fileSize,"utf-8", new DefaultFileRenamePolicy());
          Enumeration<String> en = multi.getFileNames();
          String file = en.nextElement();
          fileName = multi.getFilesystemName(file);
    }catch(Exception e) {     System.out.println(e.getMessage());    }
    PdsItemDao pd = PdsItemDao.getInstance();
    PdsItem item = new PdsItem();
    item.setDescription(multi.getParameter("description"));
    item.setFileName(fileName);
    File file = new File(real +"/" + fileName);
    item.setFileSize(file.length());
    b = pd.insert(item);
    return b;
  }
}
```

FileUploadServlet 클래스의 doPost 메서드에 삽입 처리

```
    protected void doPost(HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException {
            String uri  = request.getRequestURI();
            String path = request.getContextPath();
            String command = uri.substring(path.length()+1);
            request.setCharacterEncoding("utf-8");
            RequestDispatcher rd;
            if (command.equals("FileUpload.action")) {
                        FileInsert fi = FileInsert.getInstance();
                        int b = fi.addFile(request);
                        if (b > 0) rd = request.getRequestDispatcher("uploaded.jsp");
                        else rd = request.getRequestDispatcher("fileUploadForm.jsp");
                        rd.forward(request, response);
            }
    }
}
```

업로드 결과를 표시할 uploaded.jsp 파일을 작성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h2>파일을 업로드 했습니다.</h2>
<p>
<a href="list.action">[목록보기]</a>
</body>
</html>
```

service 패키지에 데이터를 가져오는 FileSelect 클래스

```
package service;
import java.util.ArrayList;
import dao.PdsItem;
import dao.PdsItemDao;
public class FileSelect {
	private static FileSelect instance = null;
	private FileSelect() {}
	public static FileSelect getInstance() {
		if (instance == null) {	instance = new FileSelect(); }
		return instance;
	}
	public ArrayList<PdsItem> list() {
		PdsItemDao pid = PdsItemDao.getInstance();
		return pid.list();
	}
}
```

FileUploadServlet 클래스의 doGet 메서드에 삽입을 처리

```
public class UploadServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
            String uri  = request.getRequestURI();
            String path = request.getContextPath();
            String command = uri.substring(path.length()+1);
            request.setCharacterEncoding("utf-8");
            RequestDispatcher rd;
            if (command.equals("list.action")) {
                        FileSelect fs = FileSelect.getInstance();
                        ArrayList<PdsItem> al = fs.list();
                        request.setAttribute("list", al);
                        rd = request.getRequestDispatcher("list.jsp");
                        rd.forward(request, response);
            }
    }
```

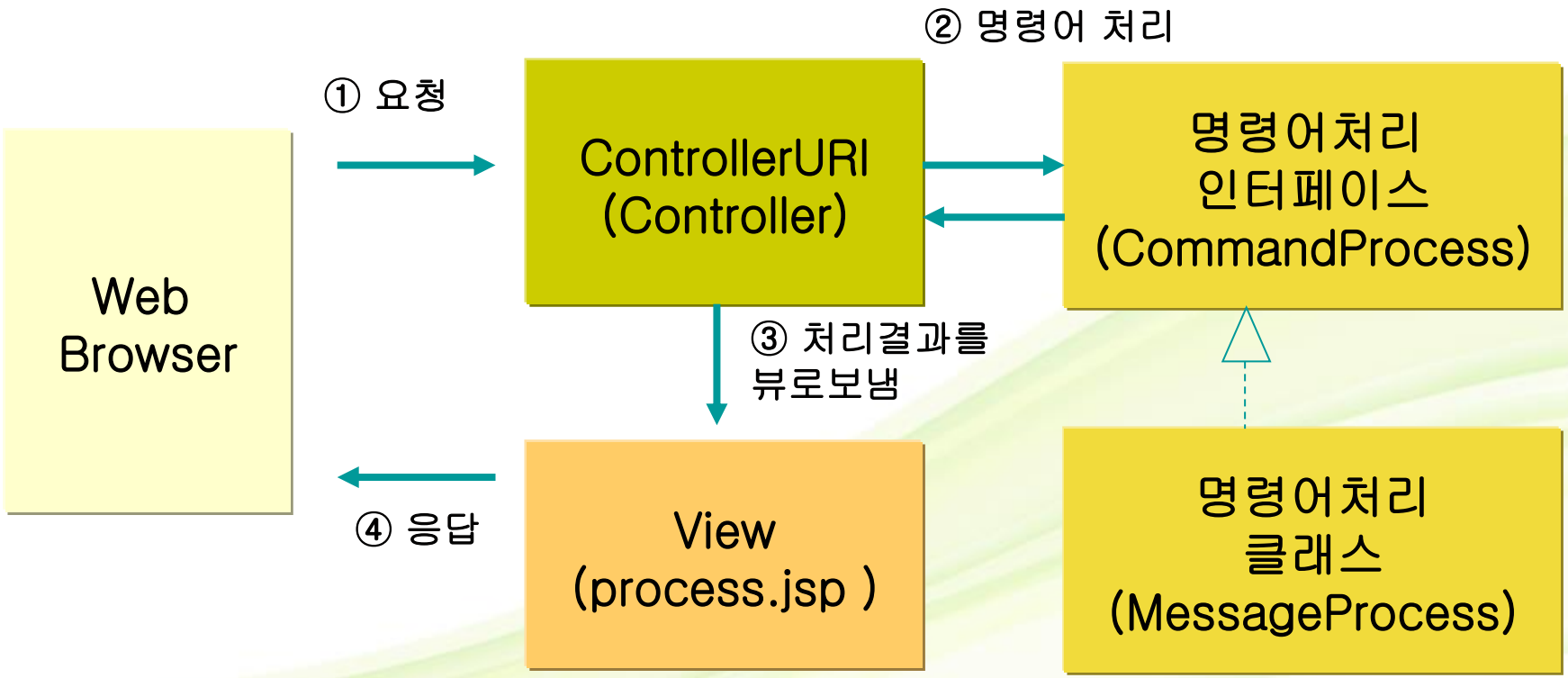
list.jsp 파일을 작성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title></head><body>
<h1>저장 정보</h1>
<table border="1" bgcolor="yellow">
<tr><th>번호</th><th>파일명</th><th>파일크기</th><th>설명</th>
            <th>저장</th></tr>
<c:forEach var="pi" items="${list}">
            <tr><td>${pi.id }</td><td>${pi.fileName }</td><td>${pi.fileSize }</td>
              <td>${pi.description }</td>
              <td><a href="upload/${pi.fileName }">다운받기</a></td></tr>
</c:forEach>
</table>
</body>
</html>
```

Properties를 이용한 MVC모델

컨트롤러(Controller)인 서블릿에 사용자의 요청을 명령어로 전달 – 커맨드 패턴

❖ 요청 파라미터로 명령어를 전달하는 예제



CommandURI.properties

/message.do=mvc.controller.MessageProcess

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>
<%
response.sendRedirect("message.do");
%>
</body>
</html>
```

Web.xml

```
<servlet>
   <description></description>
   <display-name>ControllerURI</display-name>
   <servlet-name>ControllerURI</servlet-name>
   <servlet-class>mvc.controller.ControllerURI</servlet-class>
   <init-param>
    <param-name>propertyConfig</param-name>
    <param-value>/WEB-INF/CommandURI.properties</param-value>
   </init-param>
 </servlet>
 <servlet-mapping>
   <servlet-name>ControllerURI</servlet-name>
   <url-pattern>*.do</url-pattern>
 </servlet-mapping>
</web-app>
```

CommandProcess.java

```
package mvc.controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

//요청 파라미터로 명령어를 전달하는 방식의 슈퍼 인터페이스
public interface CommandProcess {
        public String requestPro(HttpServletRequest request,
                HttpServletResponse response)
    throws Throwable;
}
```

MessageProcess.java

```
package mvc.controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

// Controller로 부터 작업의 처리를 지시받아서 작업을 처리
public class MessageProcess implements CommandProcess {

    public String requestPro(HttpServletRequest
                request,HttpServletResponse response)
                        throws Throwable {
        request.setAttribute("message","요청 파라미터로 명령어를 전달 하삼");
        return "process.jsp";
    }
}
```

```
package mvc.controller;   import java.io.*;   import java.util.*;
import javax.servlet.*;      import javax.servlet.http.*;
public class ControllerURI extends HttpServlet {
   private Map commandMap = new HashMap();//명령어와 처리 클래스를 쌍으로 저장
   public void init(ServletConfig config) throws ServletException {
      String props = config.getInitParameter("propertyConfig");
     //web.xml에서 propertyConfig에 해당하는 init-param 의 값을 읽어옴
      Properties pr = new Properties();
      FileInputStream f = null;
      try { String configFilePath = config.getServletContext().getRealPath(props);
         f = new FileInputStream(configFilePath);
         pr.load(f);//Command.properties파일의 정보를  Properties객체에 저장
      } catch (IOException e) {
         throw new ServletException(e);
      } finally {       if (f != null) try { f.close(); } catch(IOException ex) {}
      }
      Iterator keyIter = pr.keySet().iterator();
      while( keyIter.hasNext() ) {
         String command = (String)keyIter.next();
         String className = pr.getProperty(command);
         try {
            Class commandClass = Class.forName(className);
            Object commandInstance = commandClass.newInstance();
            commandMap.put(command, commandInstance);
         } catch (Exception e) { System.out.println(e.getMessage()); }       }
   }
```

```
public void doGet( HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        requestPro(request, response);
    }
    protected void doPost(  HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        requestPro(request, response);
    }
    //시용자의 요청을 분석해서 해당 작업을 처리
    private void requestPro(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        String view = null;
        CommandProcess com=null;
       try {
            String command = request.getRequestURI();
            if (command.indexOf(request.getContextPath()) == 0) {
                 command = command.substring(request.getContextPath().length()+1);
            }
                com = (CommandProcess)commandMap.get(command);
               view = com.requestPro(request, response);
        } catch(Throwable e) {
             throw new ServletException(e);
        }
        RequestDispatcher dispatcher =request.getRequestDispatcher(view);
        dispatcher.forward(request, response);
    }
}
```