Java Web Programming 입문

(Servlet/JSP)

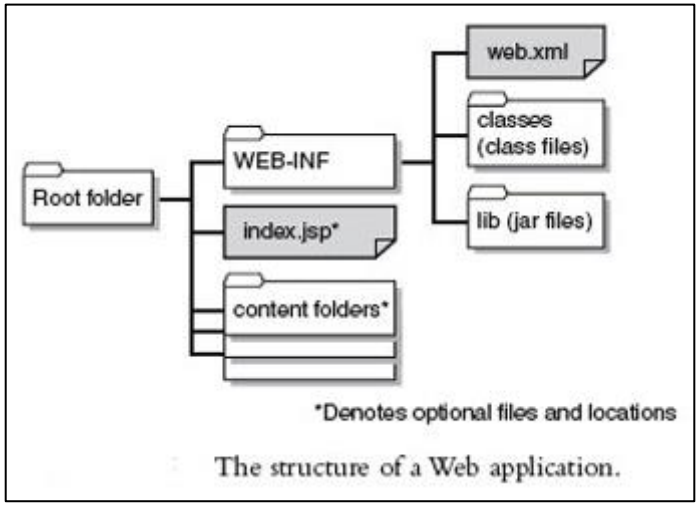
20일차

오늘의 키워드

- Tomcat Web Application
  - Web Application
  - Web Application File 위치
  - Root Application
  - Default Web Page
- Hello, Servlet!
- Hello, JSP!
- JavaBeans

Tomcat Web Application

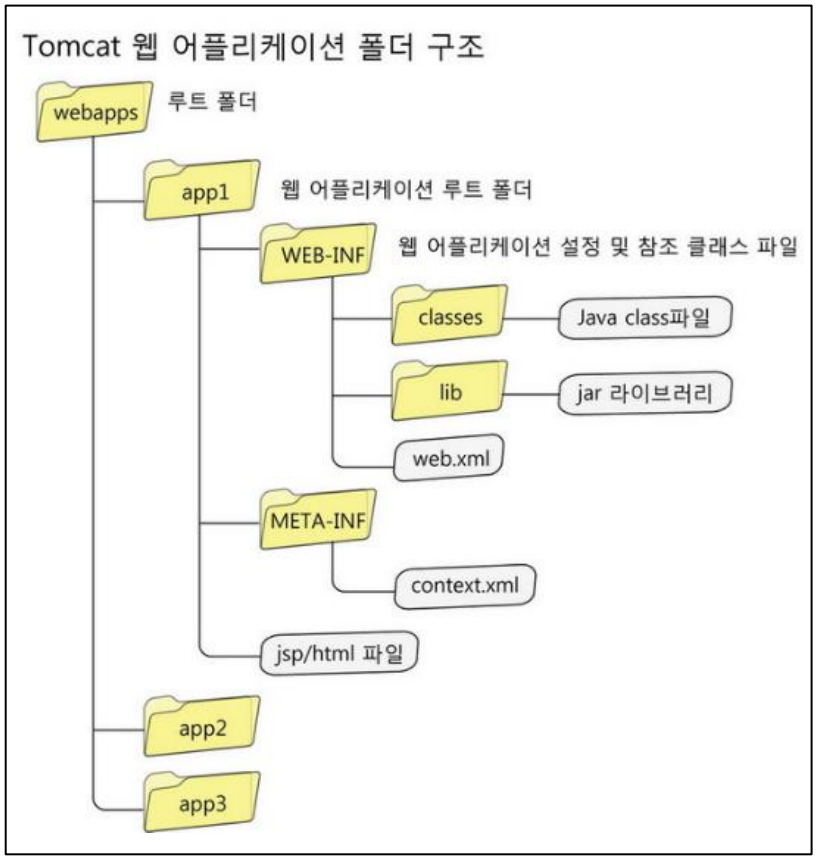
- 웹 애플리케이션 (Web Application)
  - 특정 Service를 제공하는 Directory
  - Web 환경에서의 Directory는 Application의 의미
    - 보통의 Application은 하나의 응용 프로그램을 의미
- Tomcat의 Web Application
  - webapps Directory 내에 존재하는 Directory (들)
- Web Application 구성
  - Web Application Directory
    - HTML, JSP 파일
    - URL 과 File명을 조합하여 직접 접근 가능
  - WEB-INF Directory
    - Class 파일, web.xml 파일
    - 직접 접근 불가
  - web.xml
    - Web Application의 환경설정 파일



The structure of a Web application.

Tomcat Web Application

- 파일들의 위치
  - HTML, JSP
    - WEB-INF 만 제외
  - web.xml
    - WEB-INF
  - Class
    - WEB-INF 아래 classes

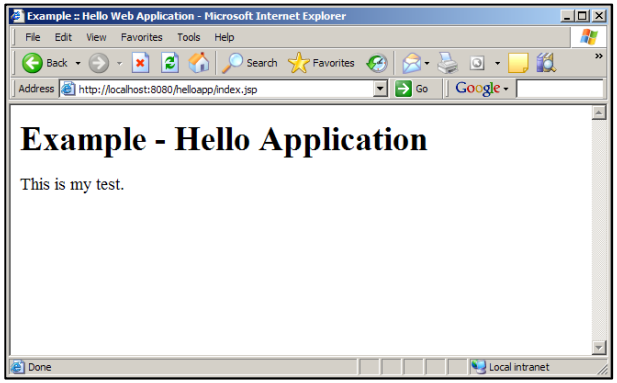


Tomcat Web Application

- Root Application
  - 최상위 Application
  - Tomcat 의 Home Directory
  - …/webapps/ROOT/

- 디폴트 웹 페이지 (Default Web Page)
  - Web Application 실행 시 가장 먼저 실행되는 파일
    - Tomcat 에서는 index.html, index.htm, index.jsp 지정
    - Tomcat은 위 Default Web Page를 찾아보고 존재하는 파일 실행

- Root Application 실행
  - http://localhost:8080/
  - http://localhost:8080/index.jsp
  - http://127.0.0.1:8080/
  - http://127.0.0.1:8080/index.jsp



Tomcat Web Application

- HelloRoot.html

```
<html>
    <body>
        Hello, ROOT Application!
    </body>
</html>
```

  - …/webapps/ROOT/

  - 실행
    - http://localhost:8080/HelloRoot.html
    - http://127.0.0.1:8080/HelloRoot.html

Tomcat Web Application

- 사용자 정의 Web Application
  - 사용자가 직접 생성한 Web Application
  - …/webapps/ 아래Directory
  - 제작 순서
    - …/webapps/ 아래에 Web Application으로 사용할 Directory 생성
    - 생성한 Web Application Directory 아래 WEB-INF Directory 생성
    - 생성한 WEB-INF 아래 classes Directory 생성
    - HTML, JSP 등 Web Application 파일 작성
    - Web Application 실행

Tomcat Web Application

- Web Application – MyWebApps 제작
  - …/webapps/MySample/ Directory 생성
  - …/MySample/WEB-INF Directory 생성
  - …/WEB-INF 에 web.xml 파일 생성 (복사)
  - …/WEB-INF/classes Directory 생성
  - myhello.html

```
<html>
    <body>
        Hello, MyWebApps!
    </body>
</html>
```

  - …/MyWebApps/ 저장
  - http://localhost:8080/Web_Application(Directory)/File
  - http://localhost:8080/MySample/myhello.html
  - http://127.0.0.1:8080/MySample/myhello.html

Hello, Servlet!

- Tomcat 환경 설정
  - Tomcat 환경 설정 파일
    - …/conf/web.xml
    - Servlet 설정 관련 주석 해제 (99~109, 348~351)
    - Tomcat 재 시작
- Servlet 작성 순서
  - Servlet Code 작성 후, Web Application 내 WEB-INF/classes 저장
  - Servlet Code Compile
  - Web Browser 통해 Servlet 실행
    - CLASSPATH 에 servlet-api.jar 파일이 포함되어야 함
      - …/common/lib/servlet-api.jar

Hello, Servlet!

- Servlet Class Import

```
import javax.servlet.*;
import javax.servlet.http.*;
```

  - javax.servlet
    - Servlet Programming에 필요한 Class
  - javax.servlet.http
    - HTTP Protocol로 제공되는 Servlet 구현을 위한 Class

- Servlet Class

```
public class HelloServlet extends HttpServlet { ... }
```

- doGet() Method

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException { ... }
```

Hello, Servlet!

- HelloServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {

        PrintWriter out = response.getWriter();
        out.println("<html>                    ");
        out.println("   <body>                ");
        out.println("       Hello Servlet     ");
        out.println("   </body>               ");
        out.println("</html>                   ");
    }
}
```

Hello, Servlet!

- HelloServlet Code 적용
  - .../webapps/MySample/WEB-INF/classes/
  - HelloServlet.java Compile
    - Javac HelloServlet.java
  - http://localhost:8080/MySample/servlet/HelloServlet
  - http://127.0.0.1:8080/MySample/servlet/HelloServlet
    - /HelloServletTest/
      - Web Application 의 Directory
    - /servlet/
      - Servlet을 실행하기 위한 URL Pattern
    - /HelloServlet
      - Servlet 이름

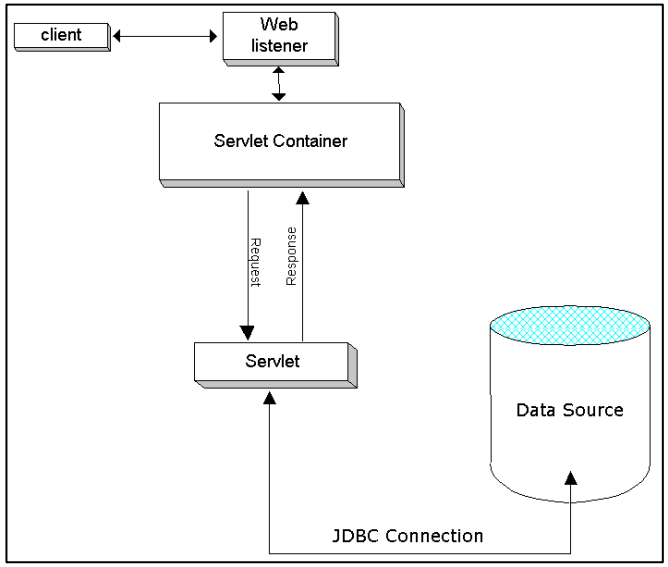
Hello, Servlet!

- Servlet Package
  - 연관된 Servlet 을 Package로 묶어서 관리
  - 한 곳에 Servlet 을 전부 모아놓는 것 보다 관리가 편함
  - 작업 별로 분류하여 사용할 수 있으므로 관리가 용이
  - Package Servlet

```
package my.servlet.test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloPackageServlet extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {

        PrintWriter out = response.getWriter();
        out.println("<html>                              ");
        out.println("    <body>                         ");
        out.println("        Hello, Package Servlet!    ");
        out.println("    </body>                        ");
        out.println("</html>                             ");
    }
}
```

Hello, Servlet!

- HelloPackageServlet Code 적용
  - HelloPackageServlet.java Compile
    - Javac –d HelloPackageServlet.java
  - http://localhost:8080/HelloServletTest/servlet/my.servlet.test.HelloPackageServlet
  - http://127.0.0.1:8080/HelloServletTest/servlet/my.servlet.test.HelloPackageServlet



Hello, JSP!

- Page 지시문

```
<%@page contentType="text/html; charset=euc-kr"%>
```

- JSP Page에서 Java Code 작성

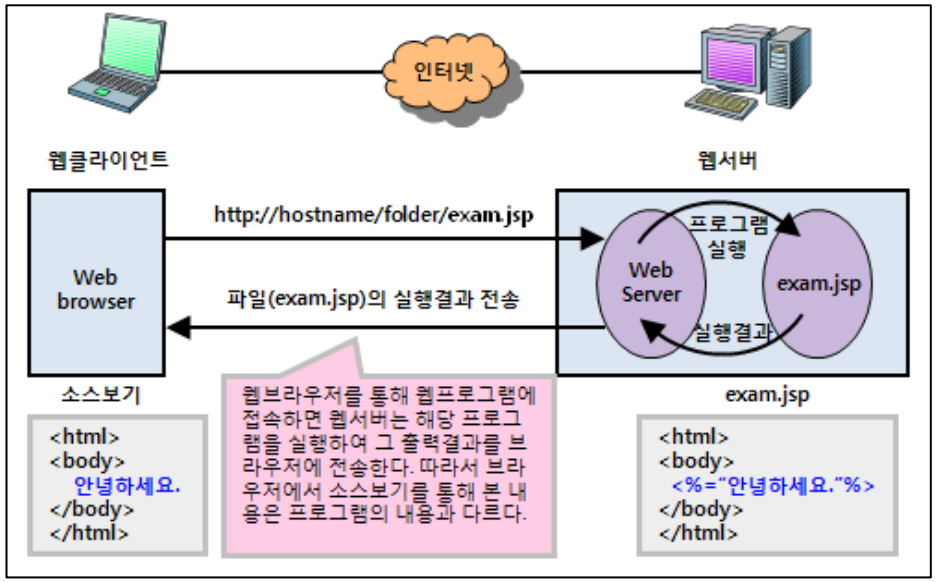
```
<% out.println("Hello, JSP!"); %>
```

- HelloJSP.jsp

```
<%@ page contentType="text/html; charset=euc-kr"%>
<html>
    <head>
        <title>Hello, JSP</title>
    </head>
    <body>
        <%out.println("Hello, JSP!");%>
    </body>
</html>
```

Hello, JSP!

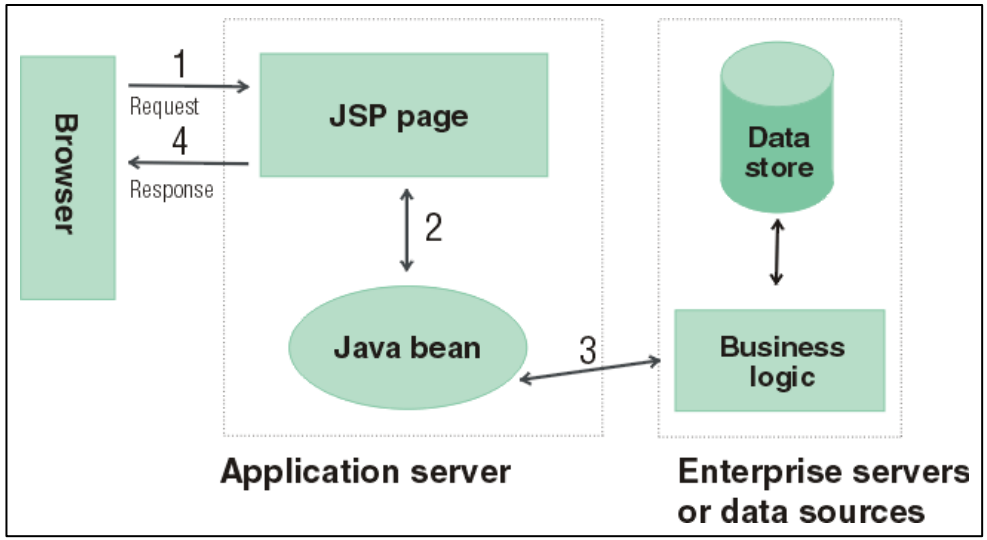
- HelloJSP JSP Code 적용
  - …/webapps/HelloJSPTest/
  - http://localhost:8080/HelloJSPTest/HelloJSP.jsp
  - http://127.0.0.1:8080/HelloJSPTest/HelloJSP.jsp



JavaBeans

- JavaBeans
  - Data 의 집합을 표현하기 위한 Class
  - Object가 가진 Member Variable 에 값을 할당(set)/얻기(get)위해 사용
  - Business Logic과 표현을 위한 코드 분리
  - 분리되므로, 내용을 재사용(Reuse) 가능한 Component 제작이 가능

- 구성
  - 생성자 (Constructor)
  - 멤버변수 (Member Variable)
  - set~ Member Method (setter)
  - get~ Member Method (getter)



JavaBeans

- 작성
  - Package
  - Class
  - Member Variable
  - Member Method
    - setter
    - getter

```
package hello;

public class HelloBean {

    private String name;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return "Hello" + this.name;
    }
}
```

JavaBeans

- HelloBean 적용
  - .../webapps/.../WEB-INF/classes
  - Javac -d HelloBean.java

- JSP에서 JavaBeans의 사용
  - Java Code

```
HelloBean testHello = new HelloBean();
```

  - JSP의 Action Tag
    - JSP 내 몇몇 기능을 XML 형태 Tag를 이용하여 쉽게 사용할 수 있도록 정의해 놓은 Tag
  - `<jsp:useBean>` Action Tag
    - JSP 내에서 JavaBeans를 이용하기 위한 Tag

```
<jsp:useBean id="testHello" class="hello.HelloBean" scope="page" />
```

JavaBeans

- Java Code를 이용한 JavaBeans
  - HelloBean Class Import

```
<%@page import="hello.HelloBean"%>
```

  - HelloBean Object 생성

```
HelloBean testHello = new HelloBean();
```

  - setter, getter Method를 사용

```
testHello.setName("Hong GilDong");
result = testHello.getName();
```

JavaBeans

- HelloJspWithJavaBeans.jsp

```jsp
<%@page contentType="text/html; charset=euc-kr"%>

<%@page import="hello.HelloBean"%>

<html>
    <head>
        <title>Hello, JSP with JavaBeans</title>
    </head>
    <body>
        <%
            String str = new String();
            HelloBean testHello = new HelloBean();
            testHello.setName("Hong GilDong");
            result = testHello.getName();
            out.println(str);
        %>
    </body>
</html>
```

- HelloJspWithJavaBeans 적용
  - …/webapps/HelloJSPTest/ 저장
  - http://localhost:8080/HelloJSPTest/HelloJspWithJavaBeans.jsp
  - http://127.0.0.1:8080/HelloJSPTest/HelloJspWithJavaBeans.jsp

JavaBeans

- Action Tag 를 이용한 JavaBeans
  - Action Tag
    - JSP 내 몇몇 기능을 XML 형태 Tag를 이용하여 쉽게 사용할 수 있도록 정의해 놓은 Tag
  - `<jsp:useBean>` Action Tag
    - JSP 내에서 JavaBeans를 이용하기 위한 Tag
  - Object 생성

```
<jsp:useBean id="testHello" class="hello.HelloBean" scope="page" />
```

    - id
      - 생성하려는 JavaBeans Object의 ID
    - class
      - JavaBeans의 Class 명 (Package 명 포함)
    - scope
      - JavaBeans의 유효범위 지정
      - “page” 값일 경우 사용범위를 현재 Page로 제한

JavaBeans

- HelloJspWithBeanTag.jsp

```jsp
<%@page contentType="text/html; charset=euc-kr"%>

<jsp:useBean id="testHello" class="hello.HelloBean" scope="page" />

<html>
    <head>
        <title>Hello, JSP with Bean Tag</title>
    </head>
    <body>
        <%
            String str = new String();
            testHello.setName("Park MoonSoo");
            str = testHello.getName();
            out.println("<h1>" + str + "</h1>");
        %>
    </body>
</html>
```

- HelloJspWithBeanTag 적용
  - …/webapps/HelloJSPTest/ 저장
  - http://localhost:8080/HelloJSPTest/HelloJspWithBeanTag.jsp
  - http://127.0.0.1:8080/HelloJSPTest/HelloJspWithBeanTag.jsp

오늘 숙제

- 집에서 해보자! (인터넷 검색 필수)