**JSP & Servlet**

**제 1장. 서블릿과 JSP 개요**

**1.1 서블릿**

Server + Applet의 합성어로 서버에서 실행되는 Applet이란 의미. 자바를 이용하여 웹에서 실행되는 프로그램을 작성하는 기술을 말한다.

서블릿은 자바 클래스 형태의 웹 애플리케이션을 말하는데, 브라우저를 통해 자바 클래스가 실행되도록 하기 위해서는 javax.servlet.http 패키지에서 제공하는 HttpServlet 클래스를 상속받아 구현해야 한다. 또한 HttpServlet 클래스를 상속받아 만든 서브 클래스를 서블릿 클래스라고 한다.

웹 프로그래밍에서는 클라이언트가 어떻게 서버에 요청하는 지가 중요하다. 이는 get방식과 post 두가지 방식 중 하나로 요청한다.

|  |  |
| --- | --- |
| **get방식** | 주소 창을 닫고 넘어가기 때문에 서버로 보내는 데이터를 사용자가 그대로 볼 수 있음. 보안에 취약. 255자 이하의 데이터만을 전송  doGet()메소드를 호출 |
| **post방식** | Html header를 타고 넘어가기 때문에 보안에 강함. 대용량의 데이터를 전송  doPost()메소드를 호출 |

**1.2 JSP**

Java Server Page의 줄임말로 자바로 서버 페이지를 작성하기 위한 언어. HTML과 JSP태그(스크립트릿)로 구성되어 화면을 작성하는 데 유리한 웹 프로그래밍 기술. 서버 페이지는 웹 서버에서 실행되는 페이지를 말하며 요청에 필요한 페이지를 로직이나 데이터베이스와의 연동을 위해 필요한 것들을 포함함. JSP에서는 자바 언어로 조릭을 구현. (서블릿은 자바 코드 내부에 HTML코드가 들어가는 구조이지만, JSP는 이와 반대로 HTML 문서 내부에 자바 코드가 들어가는 구조)

HTML 문서 일부분에서 자바를 사용할 수 있도록 하기 위해 JSP는 다양한 태그를 제공한다.

|  |
| --- |
| Ex)  <%@ page language=*"java"* contentType=*"text/html; charset=EUC-KR"*  pageEncoding=*"EUC-KR"*%>  - <%@ %>태그는 해당 페이지 내에서 사용되는 전반적인 환경을 결정해 주는 태그  (위의 경우 사용하는 언어는 자바, 한글 인코딩은 EUC-KR로 하겠다는 의미)  <% %>  - 스크립트릿. 자바 코드 기술을 위해 사용  <%= %>  - 표현식. 변수에 저장된 값이나 함수의 결과값을 출력할 때 사용 |

서블릿과 비교하여 JSP로 개발하는 것이 쉽고 간단함. 하지만 복잡한 웹 프로그래밍 구현에서는 JSP페이지에 자바 코드를 기술하면 업무 상 효율이 떨어질 수 있음. 그렇기 때문에 실무에서는 ‘복잡한 로직은 서블릿에 기술, 결과만은 JSP페이지를 통해 클라이언트에 제공’하는 형태를 취함

**제 2장. 서블릿의 기초**

**2.1 서블릿 클래스 정의하기**

|  |
| --- |
| Ex)  **public** **class** AddBoard **extends** HttpServlet {  } |

서블릿 정의를 위해서는 javax.servlet.http 패키지에서 제공하는 HttpServlet 클래스를 상속받아 구현해야 함. 또한, 브라우저를 통해 외부에서 실행되기 때문에 접근제한자는 반드시 public.

**2.2 요청 메소드**

doGet() 또는 doPost()를 오버라이딩 해야 함.

|  |
| --- |
| Ex)  **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  } |

요청처리를 하는 HttpServletRequest, 응답처리를 하는 HttpServletResponse를 갖음. 또 doGet()은 반드시 예외처리(ServletException, IOException)를 해주어야 하는데, 일반적으로 throws 절을 이용해서 doGet()메소드를 호출한 웹 서버에게 예외처리를 넘김

|  |
| --- |
| response.setCharacterEncoding("text/html; charset=utf-8") |

결과로 출력할 내용이 한글일 경우, 위와 같이 인코딩 방식을 지정하지 않으면 한글이 깨질 수 있음

**2.3 출력 스트림 얻어오기**

Response.getWriter()메소드를 호출하여 결과 출력을 준비.

|  |
| --- |
| PrintWriter out = response.getWriter();  out.println("홍길동"); |

PrintWriter는 파일 입출력을 위해서 제공해주는 자바클래스. Java.io 패키지에 정의되어 있음. print(), println() 등의 메소드를 활용하여 HTML형태로 결과를 출력

**2.4 서블릿의 동작 원리**

[웹 페이지 – 웹 서버 – 서블릿 컨테이너 – 스레드 – 서블릿 객체]

2.4.1 브라우저에서 서블릿을 요첨

2.4.2 WAS 안에 웹 서버가 서블릿 요청을 인식하여, 서블릿 컨테이너에게 서블릿을 수행하도록 넘김

2.4.3 서블릿은 스레드를 기동하여 서블릿 객체를 생성하고 이를 수행

2.4.4 서블릿 객체의 작업이 종료되면 기동되었던 스레드는 사라짐

2.4.5 서블릿의 수행 결과가 웹 서버에 전송

2.4.6 클라이언트에 전송

**2.5 서블릿의 라이프 사이클**

[인스턴스 생성 – init() – doGet 혹은 doPost() – destroy()]

2.5.1 객체가 생성되면서 init()메소드가 단 한번 호출되고, 초기화 작업을 진행

2.5.2 클라이언트의 요청이 있을 때 doGet() 혹은 doPost() 메소드가 호출됨. (여러 클라이언트가 동시에 요청을 하더라도 여러 doGet(), doPost()가 메소드가 실행되기 때문에 수행속도가 빠름)

2.5.3 서블릿이 더 이상 필요치 않은 경우(서블릿 컨테이너 종료, 서블릿 내용이 변경되어 다시 컴파일 하는 경우 등) destroy() 메소드가 호출됨

**2.6 서블릿의 데이터 통신**

2.6.1 get방식과 post방식

|  |
| --- |
| get방식) <form action=*"getBoardList"* method=*"get"*>  post방식) <form action=*"getBoardList"* method=*"post"*> |

2.6.2 쿼리스트링

: 사용자가 입력한 데이터를 서버로 전달하는 가장 단순한 방법

|  |
| --- |
| Ex) Form태그 안에서 텍스트 박스를 통해 쿼리스트링을 붙일 경우  <input name=*"텍스트 박스 이름"* type=*"text"* /> |

2.6.3 요청 객체와 파라메터 관련 메소드

request객체의 getParameter()메소드를 호출하여 <input>태그를 통해 입력된 값을 읽어옴. 읽어올 때는 <input>태그 안에 name 속성값을 getParameter의 매개변수로 얻어 추출

|  |
| --- |
| Ex)  String id = request.getParameter(“텍스트 박스 이름”) |

**제 3장. JSP 기본 다루기**

**3.1 JSP 기본 태그**

**3.1.1 스크립트릿:** <% %>

웹서버에 실행되는 자바코드를 기술

**3.1.2 선언문:** <%! %>

변수를 선언, 메소드를 정의할 때 사용

|  |
| --- |
| Ex)  <%!  String str="안녕하세요";  **public** **int** abs(**int** n){  **return** n\*n;  }  %> |

**3.1.3 표현식:** <%= %>

<%= %>은 <% out.print() %>로 변환. HTML 형태로 브라우저에 출력하기 위해 사용

**3.1.4 주석문:** <%-- -->, <!-- -->, //, /\* .. \*/

실행에는 영향을 미치지 않음. HTML 주석문은 브라우저에 주석 형태로 공개되면 반면, 위 JSP 주석문은 공개되지 않음

**3.1.5 지시자:** <%@ %>

태그 안에서 @로 시작하며 page, include, taglib 3가지의 역할을 함.

|  |  |
| --- | --- |
| Page | JSP페이지 전반적인 환경을 설정 |
| Include | 현재 페이지에 다른 파일의 내용을 삽입할 때 사용 |
| taglib | 태그 라이브러리에서 태그를 꺼내와 사용할 수 있는 기능을 제공 |

|  |
| --- |
| Ex)  <%@page import=*"com.samsung.biz.utils.JDBCUtil"*%>  : 자바에서 이미 만들어진 자바클래스를 사용  <%@ extends = "javax.servlet.jsp.HttpJspBase" %>  : 특정 클래스로부터 상속을 받으려면 extends 속성에 상위 클래스로 누구를 설정할 지 기술 |

**제 4장. JSP 내장객체와 액션 태그**

**4.1 JSP 내장 객체**

JSP 페이지에서 프로그래머가 객체를 생성하는 과정 없이 바로 사용할 수 있는 객체. JSP가 서블릿 파일로 변환될 때 JSP(서블릿) 컨테이너가 객체를 자동으로 생성해 주기 때문에 바로 사용이 가능

|  |  |  |
| --- | --- | --- |
| 입출력 관련 | request | 클라이언트와 웹 서버 사이의 요청과 관련된 정보를 관리하는 객체 |
| response | 클라이언트에 대한 응답을 처리.  실행결과를 브라우저로 되돌려 줄 때 사용  sendRedirect(): 페이지 이동시 request, response객체가 유지되지 않음  forward(): requestDispather 객체로 접근해야만 호출이 가능, 서버상에서 페이지가 이동되기 때문에 request, response객체가 유지됨 |
| out | 서버에서 클라이언트로 열려있는 출력 스트림 |
| 서블릿 관련 | page | 한 번의 클라이언트 요헝에 하나의 JSP페이지를 범위로 가짐. |
| 외부 환경정보 제공 | Session | 웹 브라우저를 닫기 전까지 페이지를 이동하더라도 사용자의 저오를 잃지 않고 서버에 보관할 수 있도록 하는 객체 |
| Application | 하나의 웹 애플리케이션을 관리하고, 웹 애플리케이션 안에서 자원을 공유. Application 객체에 저장된 내용은 프로젝트의 모든 JSP페이지에서 공유 |

**4.2 액션태그**

|  |  |
| --- | --- |
| <jsp: forward> | 다른 사이트로 이동할 때 사용, 페이지의 흐름을 제어할 때 사용 |
| <jsp: include> | 정적 혹은 동적인 자원을 현재 페이지의 내용에 포함시킬 때, 페이지를 모듈화할 때 사용 |
| <jsp: param> | 파라메터를 추가할 때 사용 |
| <jsp: useBean> | 빈을 생성하고 사용하기 위한 환경을 정의하는 액션 태그 |
| <jsp: setProperty> | 액션은 빈에서 속성값을 할당 |
| <jsp: getProperty> | 액션은 빈에서 속성 값을 얻어올 때 사용 |

**제 5장. 세션**

다른 페이지로 이동하더라도 그 전 페이지의 정보가 유지되어야 할 때 사용. 쿠키에 비해, JSP에서만 접근 가능하므로 보안에 좋음. 저장할 수 있는 데이터에 한계가 없음. 서버 상에 존재하는 객체로서 브라우저 단위상 한 개씩 존재.

|  |
| --- |
| Ex) 세션에 값을 담을 때  HttpSession session = request.getSession();  session.setAttribute("name", vo.getName());  session.setAttribute("id", vo.getId());  Ex) 세션의 값을 불러올 때  session.getAttribute("name");  session.getAttribute("id");  Ex) 세션 제거  session.removeAttribute("name"); |

(출처: JSP&Servlet 교재 참조)