**[ 4 ] Servlet 웹프로그래밍 II**

1. HTML Form 태그
   1. Html의 form태그는 서버쪽으로 정보를 전달할 때 사용하는 태그
   2. Input 태그의 속성(type, name, value)
      1. type : 태그 종류 지정(text, password, submit, checkbox, radio, reset, button, hidden)

* input 태그외에도 서버쪽에 정보 전달을 하는 select태그가 있다

<select name="menu" size=”5” multiple="multiple">

<option>떡라면</option>

<option selected="selected">김치</option>

<option>미역국</option>

</select>

* input 태그외에도 서버쪽에 여러줄 text를 잔달하는 textarea 태그가 있다

<textarea name=”content” cols=”30” rows=”5”></textarea>

* + 1. name : 서버측에 전달하고자 하는 파라미터 이름
    2. value : name에 해당하는 값(name=value가 넘어간다)

 

독서, 요리

**join.jsp /join**

border : #D4A190 회원가입 글씨 색상:#A47160 th글씨색상 : #3F3F48



@charset *"UTF-8"*;

*#joinForm\_wrap* {width:*430px*; margin: *10px* *auto*; border: *1px* *solid* *#D4A190*; padding:*20px*;}

*#join\_title*{

width:*400px*;

margin:*10px* *auto*;

padding-bottom:*20px*;

border-bottom:*1px* *solid* *#D4A190*;

color:*#A47160*;

font-size:*1.5em*;

text-align:*center*;

font-weight:*bold*;

}

*#joinForm\_wrap* **table**{margin:*0* *auto*; color:*#A47160*;}

*#joinForm\_wrap* **table** **tr**{height:*40px*; }

*#joinForm\_wrap* **table** **th**{

width:*120px*;

font-size:*0.9em*;

color:*#3F3F48*;

text-align:*left*;

padding-left:*10px*;

}

*#joinForm\_wrap* **table** **td** {width: *300px*;}

*#name***,** *#id***,** *#pw***,** *#pwChk***,** *#birth***,** *#hobby***,** *#email***,** *#mailSend* {

border: *1px* *solid* *#D4A190*; height:*20px*;

width:*95%*; padding:*3px*;

}

*#mailSend* {height: *60px*;}

*.joinBtn\_style* {

color: *#ffffff*;

background: *#A47160*;

border-style: *none*;

}

*.joinBtn\_style*{width:*32%*; height:*120%*; padding:*3px*; box-sizing: *border-box*;}

* 1. form 태그는 Input 태그들의 값을 서버로 전송하기 위한 정보를 담고 있다

<form action="요청하는 컴포넌트이름" method="요청방식">

GET방식 : <http://ip>주소:port번호/컨텍스트/path/Ex?id=”1234”&name=”Lee”

POST방식 : <http://ip>주소:port번호/컨텍스트/path/Ex

1. Servlet Parameter
   * Form태그의 submit 버튼을 클릭하여 데이터를 서버로 전송하면, 해당파일(Servlet)에서는 HttpServletRequest객체를 이용하여 Parameter값을 얻을 수 있다.
     1. request.getParameter("파라미터이름");
     2. request.getParameterValues("파라미터배열이름");
     3. request.getParameterNames() ; 파라미터 이름들이 배열로
2. 한글처리
   1. Tomcat 서버의 기본 문자 처리 방식은 IOS-8859-1 방식 입니다. 따라서 개발자가 별도의 한글 인코딩을 하지 않으면 한글이 깨져 보이는 현상이 있습니다.
   2. response객체인지, request객체인지 따라서 한글처리 방식에 차이가 있습니다
      1. response객체 : response.setContentType("text/html;charset=utf-8");
      2. request객체(POST방식) : request.setCharacterEncoding("utf-8");
      3. Get방식의 경우 server.xml에 <Connector>부분에 URIEncoding="utf-8"을 추가하기도 합니(tomcat 7.0이전 버전의 경우)
3. 웹어플리케이션 감시 : ServletContextListener
   * 웹 어플리케이션의 생명주기를 감시하는 리스너(Listener)인 ServletContextListener가 있다.
   * 리스터의 해당 메소드가 웹 어플리케이션의 시작과 종료시 호출된다(contextInitialized메소드와 contextDestroyed()메소드)
   * ServletContextListener에 implements 받은 리스터 클래스 제작하고 web.xml 파일에 리스너 클래스 기술한다

**public** **class** ListenerEx **implements** ServletContextListener{

@Override

**public** **void** contextDestroyed(ServletContextEvent arg0) {

System.***out***.println("★ ★ ★ context 종료될 때 ★ ★ ★");

}

@Override

**public** **void** contextInitialized(ServletContextEvent arg0) {

System.***out***.println("★ ★ ★ context 시작될 때 ★ ★ ★");

}

}

<listener>

<listener-class>com.tj.ex2.ListenerEx</listener-class>

</listener>

Web.xml에 명시하지 않으면 리스너 클래스에 어노테이션으로 처리할 수 있다.

@WebListener

**public** **class** ListenerEx **implements** ServletContextListener {

@Override

**public** **void** contextDestroyed(ServletContextEvent arg0) {

// **TODO** Auto-generated method stub

System.***out***.println("종료될 때");

}

@Override

**public** **void** contextInitialized(ServletContextEvent arg0) {

// **TODO** Auto-generated method stub

System.***out***.println("시작될 때");

}

}

**처음 웹 어플리케이션 실행 될 때 contextInitialized()실행 -> 선처리작업 -> init()메소드 실행 -> doGet()이나 doPost() 실행**

**종료될 때 Destroye()실행 -> 후처리작업 -> contextDestroyed()실행**