**[ 4 ] Servlet 웹프로그래밍 II**

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");
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");



#D4A190(버튼과 선들) #A47160(회원가입 글씨) #3F3F48(th 글씨)



3. Servlet 초기화 파라미터 : ServletConfig의 메소드 이용

* 특정 Servlet이 생성될 때 초기에 필요한 데이터들이 있습니다. 예를 들어 특정 경로 및 아이디 정보 등이다.
* 이러한 데이터들을 초기화 파라미터라고 하며, 어노테이션을 이용하거나, web.xml에 기술하고 Servlet파일에서는 getInitParameter(“파라미터”) 메서드를 이용해서 접근(사용)한다.
* Servlet파일에 초기화 파라미터 기술하는 법
  1. Servlet 클래스 제작 -> @WebInitParam에 초기화 파라미터 기술 -> ServletConfig 메소드 이용해서 데이터 불러오기
  2. Servlet소스 내

@WebServlet(urlPatterns= {"/ServletInitParam"},

initParams= {@WebInitParam(name="id",value="root"),

@WebInitParam(name="pw",value="mysql"),

@WebInitParam(name="jdbcdriver", value="com.mysql.jdbc.Driver")})

**public** **class** ServletInitParam **extends** HttpServlet {

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

String id=getInitParameter("id");

String pw = getInitParameter("pw");

String jdbcdriver = getInitParameter("jdbcdriver");

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<h2>ID : "+id+"</h2>");

out.println("<h2>PW : "+pw+"</h2>");

out.println("<h2>jdbcdriver : "+jdbcdriver+"</h2>");

out.close();

System.***out***.println("id:"+id);

System.***out***.println("pw:"+pw);

System.***out***.println("jdbcdriver:"+jdbcdriver);

}

}

* Web.xml에 초기화 파라미터 기술하는 법
  1. Servlet 클래스 제작 -> web.xml파일에 초기화 파라미터 기술 -> GenericServlet 메소드 이용해서 데이터 불러오기
  2. 소스

<servlet>

<servlet-name>ServletInitParam</servlet-name>

<servlet-class>com.tj.ex2.ServletInitParam</servlet-class>

<init-param>

<param-name>id</param-name>

<param-value>root</param-value>

</init-param>

<init-param>

<param-name>pw</param-name>

<param-value>mysql</param-value>

</init-param>

<init-param>

<param-name>jdbcdriver</param-name>

<param-value>com.mysql.jdbc.Driver</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>ServletInitParam</servlet-name>

<url-pattern>/ServletInitParam</url-pattern>

</servlet-mapping>

4. 데이터 공유 : ServletContext의 메소드 이용

* 여러 Servlet에서 특정 데이터를 공유해야 할 경우 context parameter를 이용해서 web.xml에 데이터를 기술하고, Servlet에서 공유하면서 사용 할 수 있다. Servlet에 대한 기술보다는 위에 기술해야 한다.

<context-param>

<param-name>contextId</param-name>

<param-value>scott</param-value>

</context-param>

<context-param>

<param-name>contextPw</param-name>

<param-value>tiger</param-value>

</context-param>

<context-param>

<param-name>contextPath</param-name>

<param-value>d:/mega-IT/IDE</param-value>

</context-param>

String cId = getServletContext().getInitParameter("contextId");

String cPw = getServletContext().getInitParameter("contextPw");

String cPath = getServletContext().getInitParameter("contextPath");

5. 웹 어플리케이션 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**