|  |
| --- |
| JSP/Servlet  2015년 5월 7일(목) 과제 |
| 통합ICTO팀 이영주 사원 |

**클라이언트에서 서블릿으로 요청하는 방식**

1. Get 방식
   1. GET 방식으로 요청이 전송되는 경우 :
      1. 브라우저 주소 표시 중에 주소를 직접 입력해서 요청하는 경우
      2. Html의 a(앵커) 태그를 사용해서 링크를 걸어서 요청하는 경우
      3. Html 폼 태그에서 method 속성을 GET으로 지정하여 요청하는 경우

<form action="" name="" method="GET">

* + 1. 간단한 파라미터를 필요하고 보안상 노출되어도 상관없을 경우

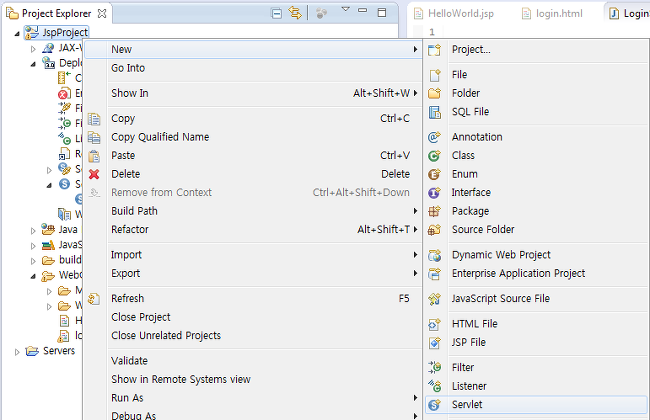
1. Post 방식
   1. 특정 페이지로 많은 양의 파라미터를 전송하거나 보안상 노출되어서는 안되는 파라미터를 전송할 경우
   2. 회원 가입, 글쓰기, 자료실 업로드 등 처리에 사용
   3. 사용 방식 : <form action="" name="" method="POST">

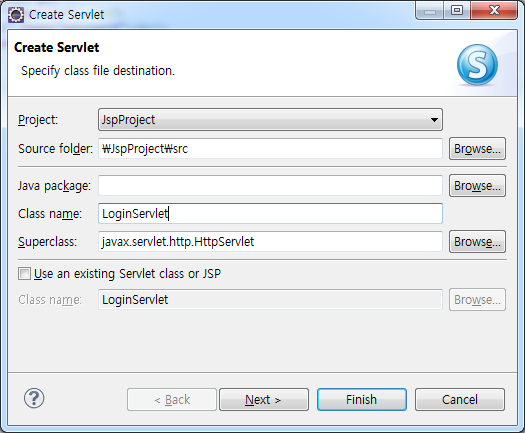
**서블릿을 이용한 클라이언트에서 전송되는 요청 처리**

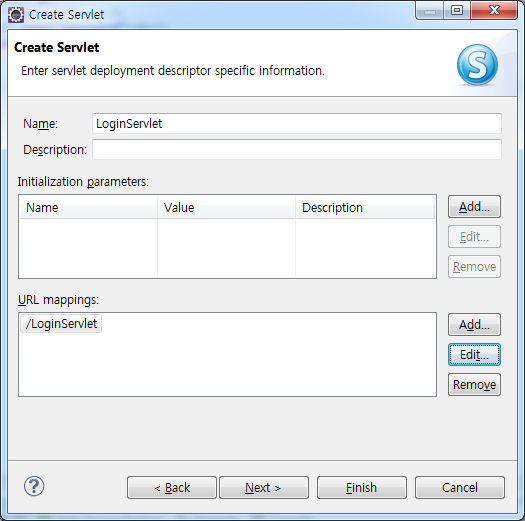
1. 요청이 GET 방식으로 전송되어 올 경우
   1. 클라이언트 페이지 생성하기

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <!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=EUC-KR">  <title> login.html</title>  </head>  <body>  <h1>로그인 </h1>  <form action="login" method="get">  아이디 : <input type="text" name="id"/><br/>  비밀번호 : <input type="text" name="passwd"><br/>  <input type="submit" value="로그인"/>  </form>  </body>  </html> |

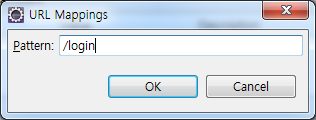
* + - 폼태그 안에서 아이디와 패스워드를 받아 login이라는 이름의 서블릿에 요청
  1. 서블릿 생성하기

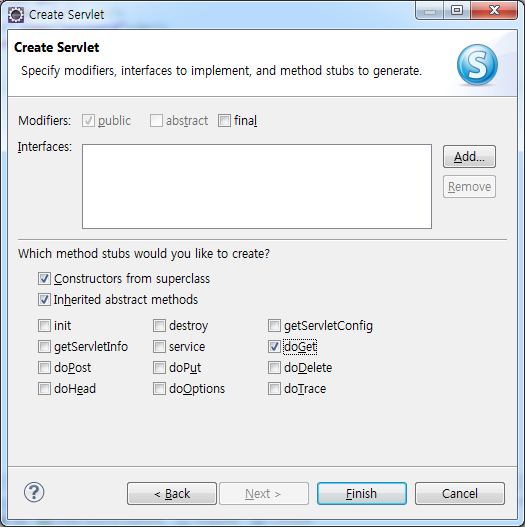


* + - Class name : LoginServlet, 이부분이 web.xml의 <servlet-class> 항목에 설정되는 값이다. 
    - [URL mappings] 항목을 선택하고 [Edit]을 선택한다 이 부분이 web.xml의 <url-pattern> 항목에 설정 된다.



* + - Pattern : /login, 요청 폼태그의 action 부분이 login으로 설정되어 있음으로 설정한다. [OK] - [Next]



* + - 예제는 GET방식임으로 [doGet]만 선택하고 [Finish]
    - 
    - 추가된 LoginServlet.java를 아래와 같이 수정한다.

|  |  |
| --- | --- |
|  | import java.io.IOException;  import java.io.PrintWriter;    import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;    /\*\*   \* Servlet implementation class LoginServlet   \*/  @WebServlet("/login")  public class LoginServlet extends HttpServlet {      private static final long serialVersionUID = 1L;        /\*\*       \* @see HttpServlet#HttpServlet()       \*/      public LoginServlet() {          super();          // TODO Auto-generated constructor stub      }        /\*\*       \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)       \*/      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {          // TODO Auto-generated method stub            String id = request.getParameter("id");          String passwd = request.getParameter("passwd");          response.setContentType("text/html;charset=euc-kr");          PrintWriter out = response.getWriter();            out.print("아이디="+id+"<br>");          out.print("비밀번호="+passwd+"<br>");      }   } |

→ 생성된 Servlet를 보면 doGet 메소드가 생성되었고 중요한 부분은 jsp2.2는 서블릿 설정이 web.xml 보다는 어노테이션 이반을 우선 제공하고 있어 @WebServlet("/login") 코드가 생성된 것이다.

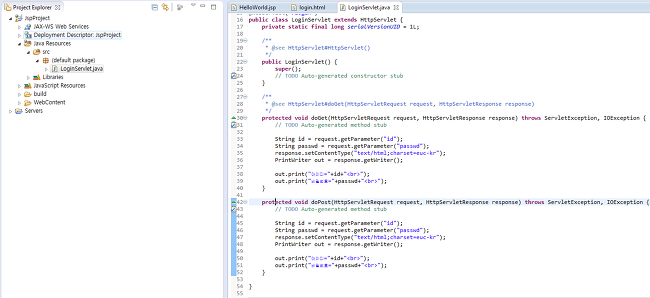
* + - 프로젝트를 실행하고 <http://localhost:8088/JspProject/login.html>로 접속하여 아이디와 비밀번호를 입력하여 Servlet이 정상 작동하는지 확인한다.

1) 요청이 GET 방식으로 전송되어 올 경우

→ 요청방식과 요청을 처리하는 부분만 변경하면 된다.

1. login.html의 method="GET" 부분을  method="POST" 로 변경한다.

2. 아래 그림과 같이 LoginServlet.java 를 열어서 doGet() 메소드 전체를 복사하여 이름만 doPost로 변경한다.



3. login.html을 웹브라우저에서 오픈하여 결과를 확인한다. 이때 URL에 parameter가 표시되지 않음을 기억한다.

**서블릿에서 한글 처리하기**

한글 처리를 제대로 하기 위해선 클라이언트(브라우저)가 처리하는 방식과 서버에서 처리하는 방식이 같아야 한다. 클라이언트는 EUC-KR로 처리하는데 서버는 ISO-8859-1로 처리하는 한글이 깨지는 결과가 발생한다.

기본적으로 톰캣 서버는 ISO-8859-1 방식이므로 한글을 제대로 처리할 수 없다.

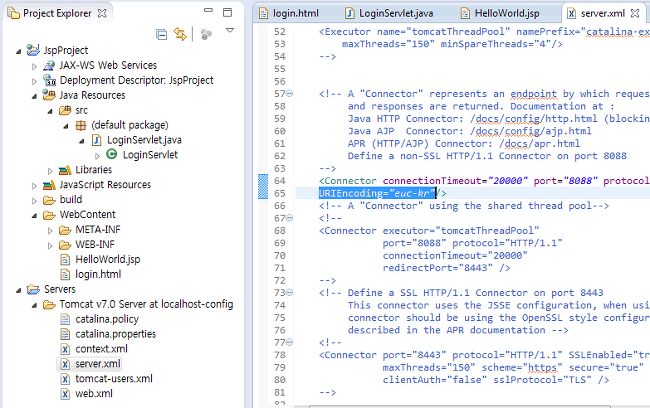
1. GET 방식으로 요청이 전송되어 올 경우
   1. 링크가 걸려서 요청 하는 경우, 폼 태그의 method 속성의 GET 방식인 경우는 페이지에서 사용하고 있는 charset 으로 인코딩 되어 파라미터가 전송된다.
   2. 주소 표시줄에 파라미터를 직접 입력한 경우 해당 브라우저에서 사용하고 있는 charset을 이용해서 인코딩을 처리한다.
      1. 개별 문자열의 캐릭터셋을 변경하여 처리하는 방법
         1. doPost() 메소드안의 패스워드 파라미터를 받는 곳을 다음과 같이 수정한다.

String passwd = new String(request.getParameter("passwd").getBytes("8859\_1"),"euc-kr");

getBytes()는 지정된 charset를 사용하여 bytes로 변환한다

new String(Byte[] bytes, String CharsetName)은 첫번째 인자 bytes를 두번째 인자 charset의 String으로 변환한다.

* + - 1. 다시 login.html 로 돌아가서 아이디, 비밀번호 필드에 각각 한글을 입력한 후 결과를 확인한다.
      2. 아이디는 한글이 깨지고 비밀번호는 정상으로 출력 됨을 확인한다.
    1. server.xml의 URIEncoding을 추가하는 방법
       1. 그림과 같이 Servers 밑에 server.xml 파일은 수정한다.

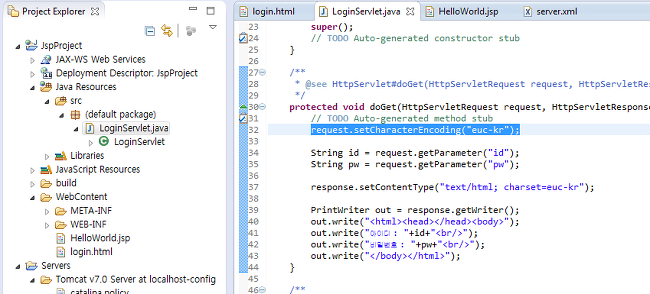


<Connector> 태그안에 URIEncoding="euc-kr" 을 추가한다.

* + - 1. Server 설정이 변경되었으니 서버를 재시작한다.
      2. 아이디, 비밀번호에 각각 한글을 입력하여 결과를 확인한다.
      3. 개별문자열 처리하는 방법과 반대의 결과가 나온다. 방금의 설정으로 아이디의 한글이 정상출력되고 비밀번호는 정상적으로 인코딩된 parameter를 다른 charset으로 인코딩하였기 때문이다.
    1. server.xml의 useBodyEncodingForURI="true" 를 추가하는 방법
       1. 1번과 2번에서 추가 했던 부분을 되돌린다. (차이를 명확히 하기 위함)
       2. 2번의 URIEncoding을 추가 했던 부분에 아래 속성을 추가한다.

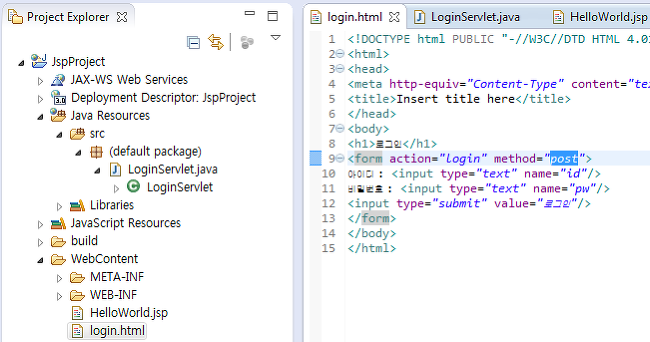
useBodyEncodingForURI="true"

* + - 1. LoginServlet.java의 doGet()의 첫번째 라인에 아래와 같이 입력한다.

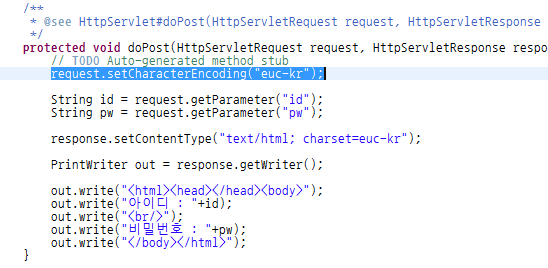


request.setCharacterEncoding("euc-kr");

* + - 1. 결과를 확인한다.
    1. POST 방식으로 요청이 전송되어 올 경우
       1. login.html의 폼 태그의 method="GET" 부분은 method="POST"로 변경한다.



* + - 1. LoginServlet.java의 doPost() 메소드의 첫줄에 아래 항목을 입력한다.



request.setCharacterEncoding("euc-kr");

get 방식과의 차이점은 server측 설정없이 위와 같은 한줄 코드로 인코딩 방식을 변경할 수 있다.

**하나의 파라미터 이름으로 여러 개의 파라미터 값 처리하기**

[ 클라이언트 측 ]

 소스 : car.html

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <!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=EUC-KR">  <title>Insert title here</title>  </head>  <body>    <h1>자동차 선택</h1>  <form action="selectCar" method="post">  <input type="checkbox" name="car" value="소나타"/>소나타  <input type="checkbox" name="car" value="너나타"/>너나타  <input type="checkbox" name="car" value="말리부"/>말리부  <input type="checkbox" name="car" value="똥차"/>똥차  <input type="submit" value="선택"/>  </form>    </body>  </html> |

[ 서블릿 페이지 ]

 소스 : SelectCarServlet.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | import java.io.IOException;  import java.io.PrintWriter;    import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;    /\*\*   \* Servlet implementation class SelectCarServlet   \*/  @WebServlet("/selectCar")  public class SelectCarServlet extends HttpServlet {      private static final long serialVersionUID = 1L;        /\*\*       \* @see HttpServlet#HttpServlet()       \*/      public SelectCarServlet() {          super();          // TODO Auto-generated constructor stub      }        /\*\*       \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)       \*/      protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {          // TODO Auto-generated method stub          request.setCharacterEncoding("euc-kr");            String[] car = request.getParameterValues("car");            response.setContentType("text/html; charset=euc-kr");          PrintWriter out = response.getWriter();            out.println("<html>");          out.println("<head>");          out.println("</head>");          out.println("<body>");          out.print("선택된 차 : ");          for(int i=0; i<car.length; i++)          {              out.print(car[i]);              out.print(i==car.length-1 ? "" : ", ");          }          out.print("</body></html>");      }    } |

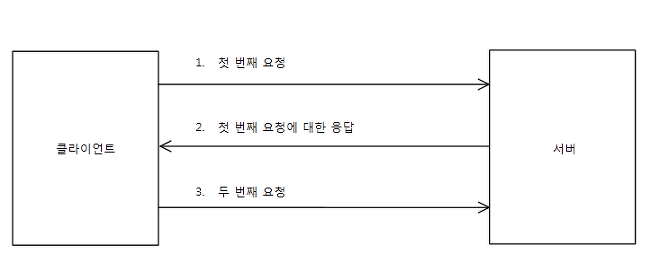
- car.html : 체크박스를 예로 하나의 이름으로 여러개의 value를 전송한다.

- SelectCarServlet.java : 하나의 이름으로 여러개의 value가 있는 parameter를 받기위해 request.getParameterValues()를 사용했다.

배열의 갯수 만큼 for문을 사용하여 response 한다.

**서블릿에서의 세션**

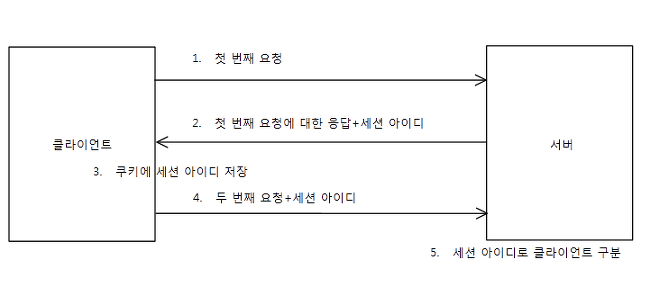
1. 세션의 개념



- Http 프로토콜의 특성은 연결을 유지 하지 않는다는 점이다. 즉 한 클라이언트가 여러차례에 요청을 보내도 서버는 클라이언트를 구분 하지 않는다는 것이다.

- 이러한 특성 때문에 동시에 여러개의 요청을 효과적으로 처리할 수 있다.

- 하지만 단점도 있기 때문에 이를 보안하기 위한 것이 세션이다.



- 서블릿의 세션 개념이 들어가면 위와 같이 처리 방법이 달라진다.

1. 클라이언트가 첫 번째 요청을 하고

2. 번의 세션 기능을 이용하여 클라이언트의 요청에 대한 응답을 할 때 서버는 중복되지 않는 세션 아이디를 클라이언트에 부여해서 응답에 같이 보낸다.

3. 응답을 받은 클라이언트는 세션아이디를 쿠기 저장소에 저장하고

4. 클라이언트가 다시 요청을 보낼때 저장된 세션 아이디가 요청에 같이 전송된다.

5. 세션 아이디로 클라이언트를 구분할 수 있다.

**서블릿에서 특정 페이지로 포워딩 하는 방법**

1. Dispatcher 방식
   1. 이 방식을 사용하면 주소표시줄이 변경된지 않는다. 즉, 하나의 요청이여서 request영역을 공유하고 서버에서 다른페이지를 호출한다

소스 : DispatcherServlet.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | import java.io.IOException;    import javax.servlet.RequestDispatcher;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;    /\*\*   \* Servlet implementation class DispatcherServlet   \*/  @WebServlet("/DispatcherServlet")  public class DispatcherServlet extends HttpServlet {      private static final long serialVersionUID = 1L;        /\*\*       \* @see HttpServlet#HttpServlet()       \*/      public DispatcherServlet() {          super();          // TODO Auto-generated constructor stub      }        /\*\*       \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)       \*/      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {          RequestDispatcher dispatcher = request.getRequestDispatcher("dispatcher.jsp");          request.setAttribute("request", "requestValue");          dispatcher.forward(request, response);      }    } |

추가된 LoginServlet.java를 아래와 같이 수정한다.

소스 : dispatcher.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <%@ page language="java" contentType="text/html; charset=EUC-KR"      pageEncoding="EUC-KR"%>  <!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=EUC-KR">  <title>Insert title here</title>  </head>  <body>  <%= request.getAttribute("request") %>  </body>  </html> |

1. Redirect 방식
   1. 포워딩 될때 클라이언트의 주소표시줄의 URL이 변경되어 요청이 바뀌게 된다. 포워딩된 jsp 페이지에서는 request영역이 달라 서블릿에서 request영역에 공유한 속성값에 접근할 수 없다.

소스 : RedirectServlet.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | 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;    /\*\*   \* Servlet implementation class RedirectServlet   \*/  @WebServlet("/RedirectServlet")  public class RedirectServlet extends HttpServlet {      private static final long serialVersionUID = 1L;        /\*\*       \* @see HttpServlet#HttpServlet()       \*/      public RedirectServlet() {          super();          // TODO Auto-generated constructor stub      }        /\*\*       \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)       \*/      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {          request.setAttribute("request", "requestValue");          response.sendRedirect("redirect.jsp");      }    } |

소스 : redirect.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <%@ page language="java" contentType="text/html; charset=EUC-KR"      pageEncoding="EUC-KR"%>  <!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=EUC-KR">  <title>Insert title here</title>  </head>  <body>  <%= request.getAttribute("request") %>  </body>  </html> |

참조 : http://yaic.tistory.com/entry/JSP-%EA%B8%B0%EC%B4%88-3-Servlet%EC%9D%98-%ED%95%B5%EC%8B%AC