2장. Servlet– Ⅰ

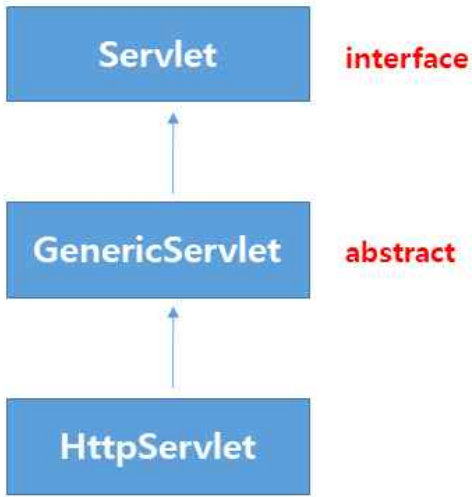
5. 서블릿 프로젝트 만들기

Servlet은 JAVA언어를 사용하여 웹 프로그램을 제작하는 것 입니다. 간단한 Servlet 프로젝트를 만들어 보면서 전체적인 구조(흐름)를 살펴보도록 합니다.

- **Servlet클래스는 HttpServlet 클래스를 상속 받음.**

```
/**
 * Servlet implementation class HelloWorld
 */
@WebServlet("/HelloWorld")
public class HelloWorld extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
```

← HttpServlet 클래스를 상속



5. 서블릿 프로젝트 만들기

- 요청처리객체 및 응답처리객체를 톰캣 서버에서 받음.

요청처리객체 → HttpServletRequest request
응답처리객체 → HttpServletResponse response

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("HelloWorld~~");

    response.setContentType("text/html");
    PrintWriter writer = response.getWriter();

    writer.println("<html>");
    writer.println("<head>");
    writer.println("</head>");
    writer.println("<body>");
    writer.println("<h1>HelloWorld~~~</h1>");
    writer.println("</body>");
    writer.println("</html>");

    writer.close();
```

Console 출력

클라이언트에게 보여줄 때는 html로.!

웹브라우저에 출력하기 위한 스트림(서블릿 파일은 자바파일이기 때문에 출력스트림을 이용해서 html코드를 직접 뿌려준다. JSP는 기본적으로 html파일이기 때문에 서블릿과는 조금 다르다. 기억하자.)

html

ex) 클라이언트가 로그인 정보를 입력하여 로그인 버튼을 누르면 request객체에 담겨서 서버로 넘어간다. 그럼, 서버는 DB를 참조해서 로그인 여부를 결정해서 그 정보를 response객체에 담아서 넘겨준다.

클라이언트의 요청이 들어오면 서버에서 그것을 받아서 처리한다. 여기서 중요한 것은 요청처리객체 및 응답처리객체는 톰캣 서버가 직접 생성해주는 것이다.

5. 서블릿 프로젝트 만들기

- GET & POST 방식

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("HelloWorld~~");

    response.setContentType("text/html");
    PrintWriter writer = response.getWriter();

    writer.println("<html>");
    writer.println("<head>");
    writer.println("</head>");
    writer.println("<body>");
    writer.println("<h1>HelloWorld~~~</h1>");
    writer.println("</body>");
    writer.println("</html>");

    writer.close();

}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IO
    // TODO Auto-generated method stub
}
```

doGet() 호출

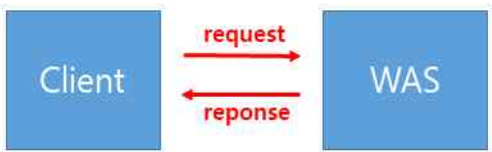
form태그 method 속성값 = get

GET 방식 :
URL값으로 정보가 전송되어 보안에 약함.(중요)
URL경로 뒤, 물음표(?)와 함께 파라메터를 붙여 전송함.
이것을 쿼리문자열이라고 말한다. 쿼리문자열은 각각의 파라미터를 &로 구분한다.

POST 방식 :
header를 이용해 정보가 전송되어 보안에 강함.(중요)

doPost() 호출

form태그 method 속성값 = post



클라이언트 즉 HTML파일에서 form태그를 이용하여
ID, PW등을 보낼 때 보내는 방식에는 Get방식과 POST
방식이 있다. 그것에 따라 서블릿의 메서드가 호출된다.

6. doGet()

- html내 form태그의 method속성이 get일 경우 호출 됩니다.
- 웹브라우저의 주소창을 이용하여 servlet을 요청한 경우에도 호출 됩니다.

doGet메소드는 매개변수로 HttpServletRequest와 HttpServletResponse를 받습니다.(톰캣 서버가 자동 생성해 줌)

6. doGet()

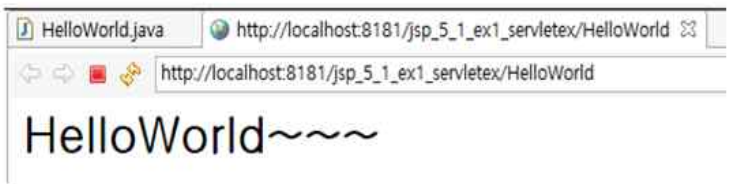
1. HttpServletResponse 객체의 **setContentType()** 메소드 호출하여 응답방식 결정합니다.
2. HttpServletResponse 객체의 **getWriter()** 메소드를 이용하여 출력 스트림을 얻습니다.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("HelloWorld~~");
    기본적으로 응답할 때 html코드로 응답하고, 문자셋은 utf-8로 설정함(필히 설정해야 함)
    response.setContentType("text/html; charset= utf-8 ;
    PrintWriter writer = response.getWriter();

    writer.println("<html>");
    writer.println("<head>");
```

6. doGet()

출력스트림의 println() 메소드를 이용하여 출력하면, 웹브라우저에 출력 됩니다.

```
response.setContentType("text/html; charset=euc-kr");
PrintWriter writer = response.getWriter();

writer.println("<html>");
writer.println("<head>");
writer.println("</head>");
writer.println("<body>");
writer.println("<h1>HelloWorld~~~</h1>");
writer.println("</body>");
writer.println("</html>");

writer.close();
```



마지막에 출력객체 닫습니다.(리소스 해제)

```
writer.close();
```

7. doPost()

- html내 form태그의 method속성이 post일 경우 호출 됩니다.

post.html | PostMethod.java | http://localhost:8181/jsp_5_1_ex1_servletex/PostMethod

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset=" utf-8 ">
5  <title>Insert title here</title>
6  </head>
7  <body>
8
9      <form action="PostMethod" method="post">
10         <input type="submit" value="post">
11     </form>
12
13 </body>
14 </html>
```

HTML

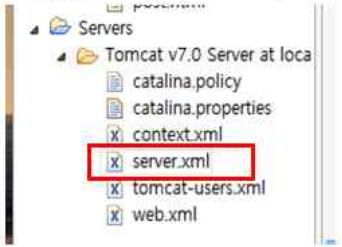
```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throw
    // TODO Auto-generated method stub
    System.out.println("doPost");

    response.setContentType("text/html; charset=euc-kr");
    PrintWriter writer = response.getWriter();
    writer.println("<html>");
    writer.println("<head>");
    writer.println("</head>");
    writer.println("<body>");
    writer.println("<h1>POST 방식 입니다. 따라서 getPost 메소드 호출 되었습니다.</h1>");
    writer.println("</body>");
    writer.println("</html>");
}
```

Servlet

8. 컨텍스트 패스(Context Path)

WAS(Web Application Server)에서 웹어플리케이션을 구분하기 위한 path입니다.
(통상 웹어플리케이션(프로젝트명) 이름으로 봐도 무방하다.)

이클립스에서 프로젝트를 생성하면, 자동으로 server.xml에 추가 됩니다.



post.html | PostMethod.java | http://localhost:8181/jsp_5_1_ex1_servletex/PostMethod | server.xml

```
101        /docs/config/cluster.html (reference documentation) -->
102      <!--
103      <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
104      -->
105
106      <!-- Use the LockOutRealm to prevent attempts to guess user passwords
107           via a brute-force attack -->
108      <Realm className="org.apache.catalina.realm.LockOutRealm">
109        <!-- This Realm uses the UserDatabase configured in the global JNDI
110             resources under the key "UserDatabase".  Any edits
111             that are performed against this UserDatabase are immediately
112             available for use by the Realm.  -->
113        <Realm className="org.apache.catalina.realm.UserDatabaseRealm" resourceName="UserDatabase"/>
114      </Realm>
115
116      <Host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
117
118        <!-- SingleSignOn valve, share authentication between web applications
119             Documentation at: /docs/config/valve.html -->
120        <!--
121        <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
122        -->
123
124        <!-- Access log processes all example.
125             Documentation at: /docs/config/valve.html
126             Note: The pattern used is equivalent to using pattern="common" -->
127        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs" pattern="%h %l %u %t &quot;%r&quot; %s %b" prefix="localhost_access_log." suffix=".txt"/>
128
129        <Context docBase="jsp_5_1_ex1_servletex" path="/jsp_5_1_ex1_servletex" reloadable="true" source="org.eclipse.jst.jee.server:jsp_5_1_ex1_servletex"/></Host>
130      </Engine>
131    </Service>
```

감사합니다.