

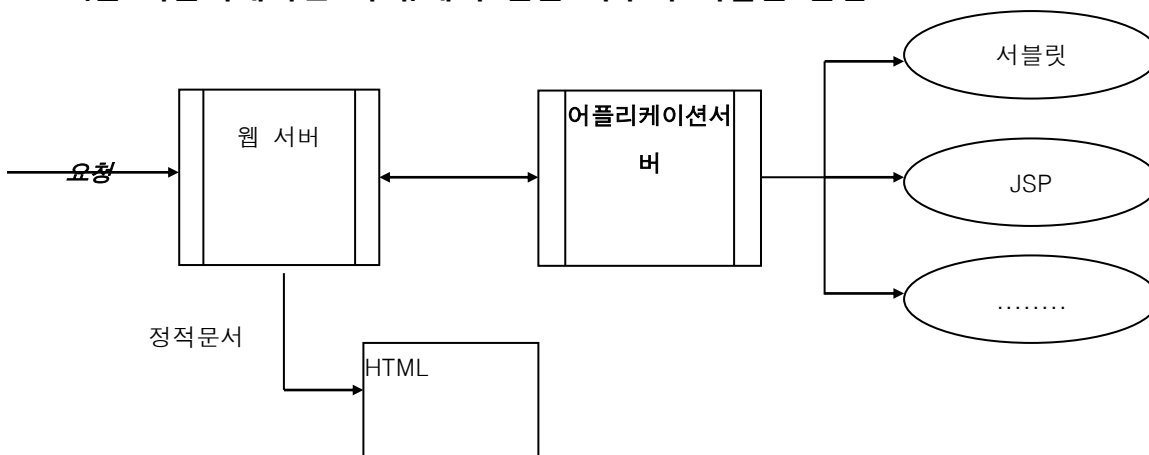
웹 문서 : 웹 문서란 웹에서 클라이언트가 서버에게 정보를 요청하면 응답되는 콘텐츠

- 정적인 웹 문서 : 사이트 관리자가 문서를 변경하지 않는 한 항상 동일한 내용을 전달, HTML, GIF, JPG, PDF, 영상, 음원 등
- 동적인 웹 문서 : 요청 시마다 다른 웹 문서의 내용을 클라이언트로 전달하는 것으로 다음과 같은 두 가지 방법 중 하나에 의해 처리된다 - 서블릿, JSP, ASP, PHP,

웹 서버 : HTTP 라는 프로토콜을 기반으로 하여 웹 클라이언트(브라우저)로부터의 요청을 서비스하는 기능을 담당한다.

웹 어플리케이션 서버 : 웹 서버의 기능들을 구조적으로 분리하여 처리하려는 목적으로 제시된 것이 바로 웹 어플리케이션 서버(WAS). 서블릿 컨테이너, JSP 컨테이너, EJB 컨테이너, 네이밍 서버 등으로 구성. JEUS, Web Logic, **Tomcat**, JBOSS, WebSphere, IAS 등

클라이언트로부터 요청 받는 일과 화면에 표현하는 로직 (Presentation Logic)까지만 웹 서버에서 담당하고 다양한 기능들을 수행하는 로직 (Business Logic)은 컨테이너가 담당하도록 WAS(웹 어플리케이션 서버)에서 일을 나누어 역할을 분담.



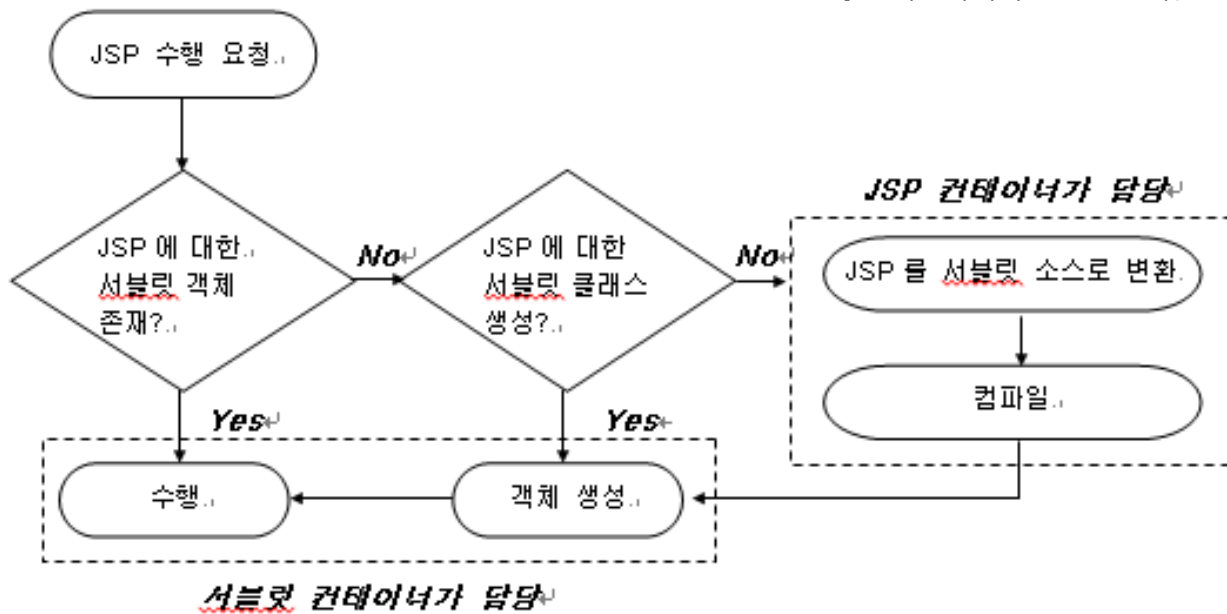
-서블릿 컨테이너 : 클라이언트에서 전송되는 서블릿 요청에 대하여 서블릿을 수행하는 프로그램. 기본 스펙들을 지원 . 서블릿 표준 API 에서 제공되는 추상 클래스와 인터페이스들을 구현한 클래스들을 제공.

-JSP 컨테이너 : JSP 컨테이너는 JSP 의 서블릿 변환까지만을 담당하는 프로그램이며 변환된 서블릿의 수행은 서블릿 컨테이너가 담당.

실습

Tomcat 설치 교재 p44 ~ 51

Eclipse 에 Tomcat 등록하기 p81~ p87 2 번까지만



실습

eclipse 에서 프로젝트선택>오른쪽 마우스 클릭 >new> Servlet>

서블릿 클래스명(ServletTest1) 입력, 패키지명(day01) 입력 > Finish 버튼 클릭

아래와 같이 코딩. (자동 완성되는 일부 코드 삭제 해도 됨)

서블릿의 구현 기초 : Servlet, GenericServlet, HttpServlet 중 하나 상속

```

package day01;
import javax.servlet.*;
import javax.servlet.http.*;

@WebServlet("/ServletTest1")
public class ServletTest1 extends HttpServlet {
    protected void doGet (HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        System.out.println("doGet");
    }
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        System.out.println("doPost");
    }
}

```

```
}
}
```

실행 방법 :

소스창에서 오른쪽 마우스 클릭 > Run as > Run a Server

또는 웹 브라우저에 <http://localhost:8088/02JspServlet/ServletTest1> 로 요청

수행 결과 :

Get 방식 요청 시 서버 콘솔에 doGet 출력되는데, (default 로 get 방식 요청이다.)

Post 방식 요청 시 doPost 출력. (이 예제에서는 Post 로 요청할 수 없음)

서블릿의 구현: GET 방식 요청이나 POST 방식요청이나, WAS 는 service(..) 메소드 호출함. 만약, service(..)가 재정의 되어 있는 것이 있으면 service(..) 메소드가 실행되고, 그렇지 않으면, HttpServlet 의 service(..)가 실행되는데, 이때, service(..)는 GET 방식 요청 시에는 해당 서블릿의 doGet(..)을 다시 호출하고, POST 방식 요청에는 해당 서블릿의 doPost(..)를 호출함.

GET 방식과 POST 방식 비교

GET 방식	POST 방식
QUERY_STRING 환경변수를 통해서 가능 (hello?key=value&name=value ...) 최대 240 바이트까지 가능. URL 에 값이 노출되는 것이 단점.	HTML Form 에 입력한 내용을 전송할 때 사용. 데이터크기 제한 없음. URL 에 정보 노출 안됨

입력폼에서 파라미터 전달예제 :

WebContent/day01/[login.html](#)

로그인하세요.

```
<form action="/02JspServlet/LoginServlet" method="post">
    id : <input type="text" name="id" />
    pw : <input type="password" name="pw" />
    <input type="submit" value="login" />
</form>
```

실행 방법 :

소스창에서 오른쪽 마우스 클릭 > Run as > Run a Server

또는 웹 브라우저에 <http://localhost:8088/02JspServlet/day01/login.html> 로 요청

html 화면에서 login 버튼 클릭 시 **post** 방식으로
<http://localhost:8088/02JspServlet/ServletTest2> 로 요청

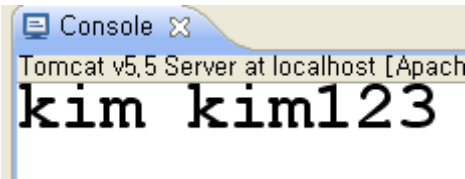
src/ **day01.LoginServlet.java**

```
package day01;
import javax.servlet.*;
import javax.servlet.http.*;

@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
                        HttpServletResponse response){
        String id = request.getParameter("id");
        String pw = request.getParameter("pw");
        System.out.println(id+" "+pw);
    }
}
```

입력폼에서 파라미터 전달예제- 결과

<http://localhost:8088/02JspServlet/day01/login.html> 요청 시

<p>로그인하세요 .</p> <p>id : <input type="text" value="kim"/></p> <p>pw : <input type="password" value="●●●●●"/></p> <p><input type="button" value="login"/></p>	<p>로그인 버튼 클릭</p> 
---	---

서블릿에서 클라이언트로 응답 예제 day01.LoginServlet.java

```
package day01;
```

```

import javax.servlet.*;
import javax.servlet.http.*;

@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    public void doPost(ServletRequest request, ServletResponse response){
        /*서버가 보내는 문자코드 설정*/
        /*response.getWriter()를 호출하기 전 response 문자코드 세팅*/
        response.setContentType("text/html;charset=UTF-8");
        /* PrintWriter 인스턴스를 생성 */
        PrintWriter out = response.getWriter();

        /*클라이언트가 보내는 문자코드 설정 */
        request.setCharacterEncoding("UTF-8");
        String userID = request.getParameter("id") ;
        out.println(userID);    /* 클라이언트로 전송 */
    }
}

```

GET 방식으로 전달되는 한글 정보 처리를 위한 설정. conf\server.xml 파일에서 아래와 같이 **URIEncoding** 부분 추가 후 탐캣 재시작

```

<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" URIEncoding="utf-8" />

```

[서블릿 요청 URL 형식과 요청 방법]

1) 서블릿을 WEB-INF/web.xml 에 등록

```

<servlet>
    <servlet-name>test</servlet-name>

```

```
<servlet-class>ServletTest</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>test</servlet-name>
  <url-pattern>/a</url-pattern>
</servlet-mapping>
```

** 주의 : url-pattern 은 한 application 내에서 유일 해야함.

2) 톰캣 7.0 부터 annotation 으로 등록 가능

@WebServlet("/a")

public class **ServletTest** extends HttpServlet { ..}

다음과 같이 요청

