



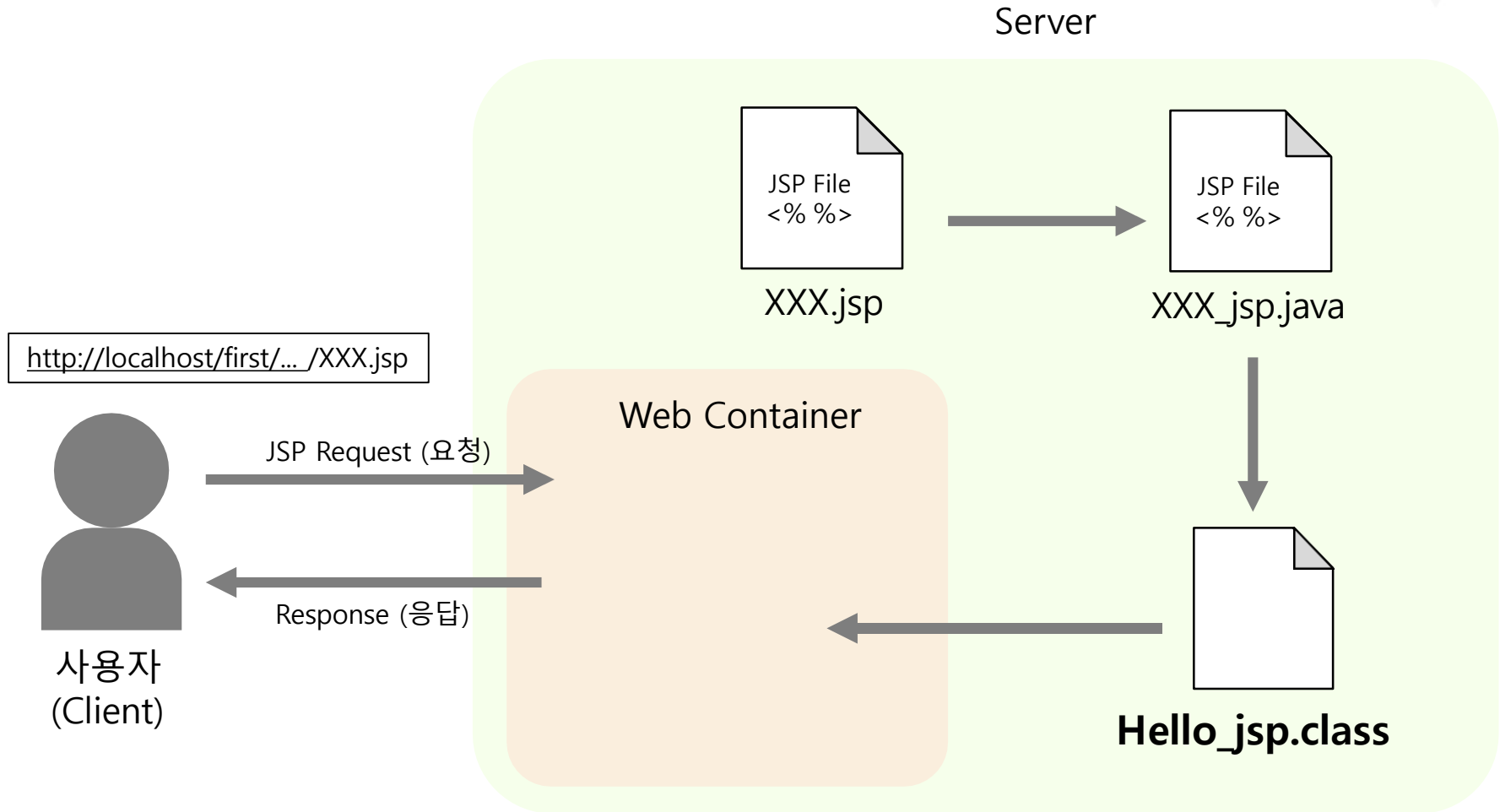
JSP

▶ Servlet VS JSP

| | Servlet | JSP |
|----|---|--|
| 형태 | Java코드에 HTML코드 삽입 | HTML코드에 Java코드 삽입 |
| 예시 | <code>out.println("<HTML">);</code> | <code><% for(int k = 0; k < 10; k++){ %></code> |
| 특징 | Business 로직 처리에 적합 | 화면 로직 처리에 적합 |

* JSP기술의 목표는 Servlet의 Business로직으로부터 화면 로직을 분리하는데 있음

▶ JSP 실행 방식



▶ JSP 특징

1. JSP파일이 변경되지 않는다면

 'jsp'파일에 대한 컴파일은 다시 일어나지 않음

2. JSP파일이 변경될 때마다 Web Container는 translation, compile, load, initialization과정 수행

* 구 버전의 JSP파일을 overwrite할 경우 제대로 반영되지 않는 경우가 발생할 수 있음

3. JSP파일의 배포 환경(위치)은 HTML과 동일(WEB ROOT 폴더 하단)

▶ JSP Elements 표기법

| 태그 | 표기법 |
|-----------------|--------------|
| Comments tag | <%-- 주석 --%> |
| Directive tag | <%@ 지시자 %> |
| Declaration tag | <%! 선언문 %> |
| Scriptlet tag | <% 코드 %> |
| Expression tag | <%= 표현식 %> |



예시

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" %>
<%!
    // JSP 선언 태그입니다.
    public static final String NAME = "JSP World";
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSP Elements 실습</title>
</head>
<body>
<%-- JSP 주석 태그입니다. --%>
<%
    // JSP 스크립트릿 태그입니다.
    out.println("Welcome :");
    String name = request.getParameter("name");
    if(name == null || name.isEmpty()){
        name = NAME;
    }
%>
<!-- JSP 값 표현 태그입니다. -->
<h3>Hello, <%= name %></h3>
</body>
</html>
```

JSP 지시자 태그

JSP 선언 태그

JSP 주석 태그

JSP 스크립트릿 태그

JSP 값 표현 태그

▶ JSP Elements

✓ Directive tag

JSP page 전체에 영향을 미치는 정보를 기술할 때 쓰임

```
<%@ 지시자 [속성 명="value"] ... %>
```

✓ 지시자 종류

✓ 지시자 종류

page, include, taglib 3종류로 나뉨

```
<%@ page import="java.io.*"%>
```

```
<%@ include file="header.html"%>
```

```
<%@ taglib uri="/WEB-INF/tags/abc" prefix="abc" %>
```

▶ JSP Elements

✓ Declaration tag

Servlet클래스의 멤버변수/메소드에 해당하는 코드를 작성할 때 사용

✓ 멤버 변수 선언

```
<%! public static final String DEFAULT_NAME = "홍길동"; %>
```

```
<%! int counter = 0; %>
```

✓ 멤버 메소드 선언

```
<%!
```

```
    public String getName(HttpServletRequest request){  
        return request.getParameter("name");  
    }  
}
```

```
%>
```

▶ JSP Elements

✓ Scriptlet tag

jspServlet 메소드의 로컬 변수와 코드를 작성할 때 사용

✓ 로컬 변수 선언

```
<% int i = 0; %>
```

✓ 자바 코드 내용 기술

```
<% if(i > 10) { %>
```

i가 10보다 큽니다.

```
<% } else { %>
```

i가 10보다 작습니다.

```
<% } %>
```


▶ JSP Elements

✓ Expression tag

Servlet코드에서 out.print()의 역할 수행

✓ 예제

현재 시간은 <%= new java.util.Date() %> 입니다. * ; 를 붙이지 않음

➔ out.print(new java.util.Date());

▶ JSP 내장 객체

JSP에서 기본적으로 제공하는 객체들로 request, response, out 등 Scriptlet tag와 Expression tag에서 사용할 수 있게 암시적으로 선언된 객체

jspElements.jsp

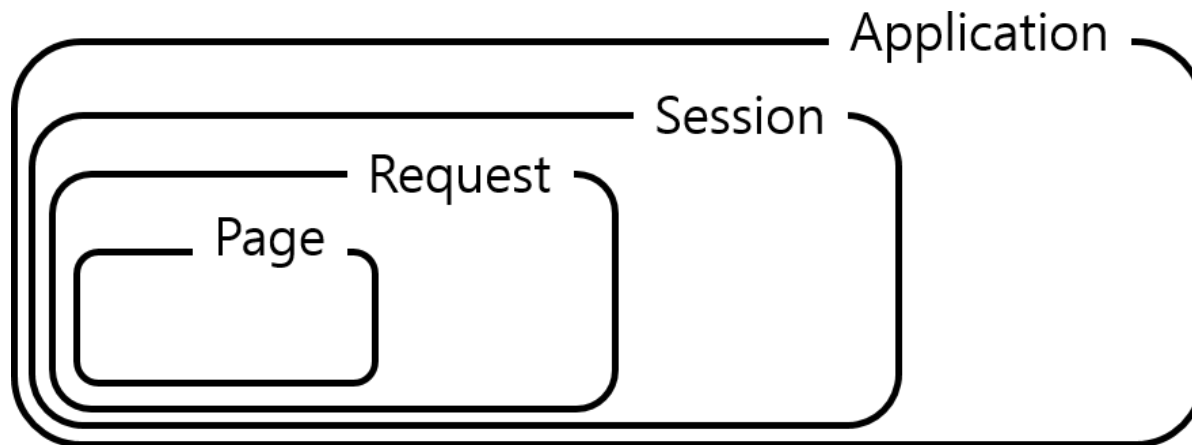
```
<%  
    // JSP 스크립트릿 태그입니다.  
    out.println("Welcome :");  
    String name = request.getParameter("name");  
    if(name == null || name.isEmpty()){  
        name = NAME;  
    }  
%>
```

▶ JSP 내장 객체 종류

| 내장 객체 명 | 설명 |
|-------------|------------------------------|
| request | HttpServletRequest 객체 참조 변수 |
| response | HttpServletResponse 객체 참조 변수 |
| out | JspWriter 객체 참조 변수 |
| session | HttpSession 객체 참조 변수 |
| application | ServletContext 객체 참조 변수 |
| page | 현재 JSP페이지에 대한 참조 변수 |
| exception | 발생하는 Throwable객체에 대한 참조 변수 |

▶ JSP 내장 객체 영역

| 영역 | 설명 |
|-------------|---------------------------|
| page | 하나의 JSP페이지를 처리할 때 사용되는 영역 |
| request | 하나의 요청을 처리할 때 사용되는 영역 |
| session | 하나의 브라우저와 관련된 영역 |
| application | 하나의 웹 애플리케이션과 관련된 영역 |



▶ Request 주요 메소드

| 메소드 명 | 설명 |
|--------------------------|-------------------------------------|
| getParameter(name) | name 파라미터의 값 리턴 |
| getParameterValues(name) | name 파라미터의 값을 배열 형태로 리턴(checkbox 등) |
| getParameterNames() | 요청에 포함된 파라미터 이름 리턴 |
| getMethod() | 현재 요청 방식 리턴 |
| getSession() | 현재 세션 객체 리턴 |
| setCharacterEncoding() | 클라이언트에서 서버로 전달된 값을 지정한 문자 셋으로 변경 |

▶ Response 주요 메소드

| 메소드 명 | 설명 |
|----------------------------|---------------------------------|
| sendRedirect(url) | 응답 결과를 요청으로 하여 지정된 url에 재전송 |
| setStatus(int status_code) | 응답으로 전송될 상태 코드 설정, 성공은 200 / OK |
| sendError(int status_code) | 에러가 발생한 경우 응답 헤더에 상태 코드 설정 |
| setContentType(String) | 서버에서 클라이언트로 전달될 값의 데이터 타입 설정 |

▶ JSP 지시자 태그

✓ page

여러 개의 page구문을 사용할 수 있지만 import 속성을 제외하고는 한 페이지에 한 번씩만 선언 가능

page지시어는 JSP파일 어느 위치에 와도 상관 없으나 가장 첫 부분에 사용하는 것이 좋음

```
<%@ page import="java.io.*"%>
```

```
<%@ page contentType="text/html" %>
```

▶ JSP 지시자 태그

✓ import

변환될 서블릿 클래스에 필요한 자바 클래스의 import문 정의 java.lang, javax.servlet, javax.servlet.http, javax.servlet.jsp는 기본적으로 import 되어 있으며 여러 package import 시 ', '로 구분

```
<%@ page import="java.io.*"%>  
<%@ page contentType="text/html" %>
```

✓ contentType

MIME 타입과 문자 인코딩 설정

```
<%@ page contentType="text/html;charset=euc-kr" %>
```


▶ JSP 지시자 태그

✓ isErrorPage

현재 페이지가 JSP오류 처리용 페이지인지 정의
값은 true/false(default), true인 경우 exception 내장 객체 사용 가능

```
<%@ page isErrorPage="true" %>
```

✓ errorPage

해당 JSP페이지가 발생시키는 모든 runtime exception을 처리할
다른 JSP페이지 지정, 값은 상태적인 URL

```
<%@ page errorPage="/error/errorForm.jsp" %>
```

▶ JSP 지시자 태그

✓ include

다른 페이지(JSP, HTML)를 포함할 수 있음

```
<%@ include file="페이지 경로" %>
```

```
<%@ include file="footer.html" %>
```

▶ JSP Exception 처리

JSP페이지에서 발생하는 Exception을 처리하기 위해서

별도의 예외 처리 페이지 지정

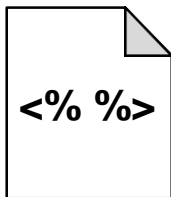
하나의 JSP페이지에 대한 예외 처리 페이지는 하나만 지정할 수 있어

예외마다 다른 예외 처리 불가

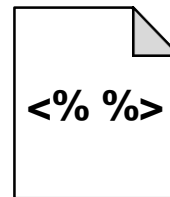
예외가 발생할 페이지 : `<%@page errorPage="/error/exceptionPage.jsp"%>`

예외를 처리할 페이지 : `<%@page isErrorPage="true"%>`

`/throws_error.jsp`



`/error/exceptionPage.jsp`



Web Container는 발생한 오류를 catch하여
예외 처리 페이지로 전달한다.