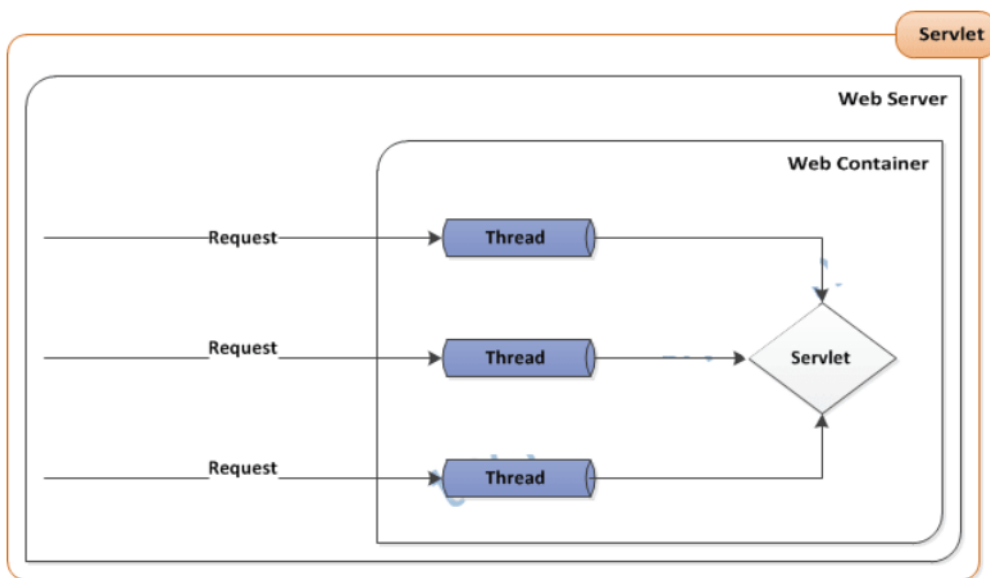


JAVA 개발자 양성과정

Servlet & JSP

서블릿 소개

- 자바를 사용하여 웹페이지를 동적으로 생성하는 서버측 프로그램을 말한다.



Apache Tomcat (Web Application Server)

- web(Servlet & Jsp) container 라고도 한다.
- 네트워크 통신
- 서블릿의 라이프 사이클 관리
- 멀티 스레딩 지원

Servlet & JSP 개발환경 설정

1. Apache Tomcat 설치

- <http://tomcat.apache.org> 사이트에서 Apache Tomcat 다운로드한다.

Download

Which version?

Tomcat 10

Tomcat 9

Tomcat 8

Tomcat Migration Tool
for Jakarta EE

Tomcat Connectors

Tomcat Native

Taglibs

Archives

Documentation

Tomcat 10.1 (alpha)

Tomcat 10.0

Tomcat 9.0

Tomcat 8.5

Tomcat Connectors

Tomcat Native

Wiki

Migration Guide

Presentations

Specifications

Problems?

Security Reports

Find help

FAQ

Mailing Lists

[KEYS](#) | [9.0.53](#) | [Browse](#) | [Archives](#)

Release Integrity

You **must** [verify](#) the integrity of the downloaded files. We provide OpenPGP signatures for every release. This signature should be matched against the [KEYS](#) file which contains the public keys of the Apache Tomcat Managers. We also provide [SHA-512](#) checksums for every release file. After downloading a file, you should calculate a checksum for your download, and make sure it is the same as ours.

Mirrors

You are currently using <https://mirror.navercorp.com/apache/>. If you encounter any problems, please select another mirror. If all mirrors are failing, there are *backup* mirrors that should be available.

Other mirrors:

9.0.53

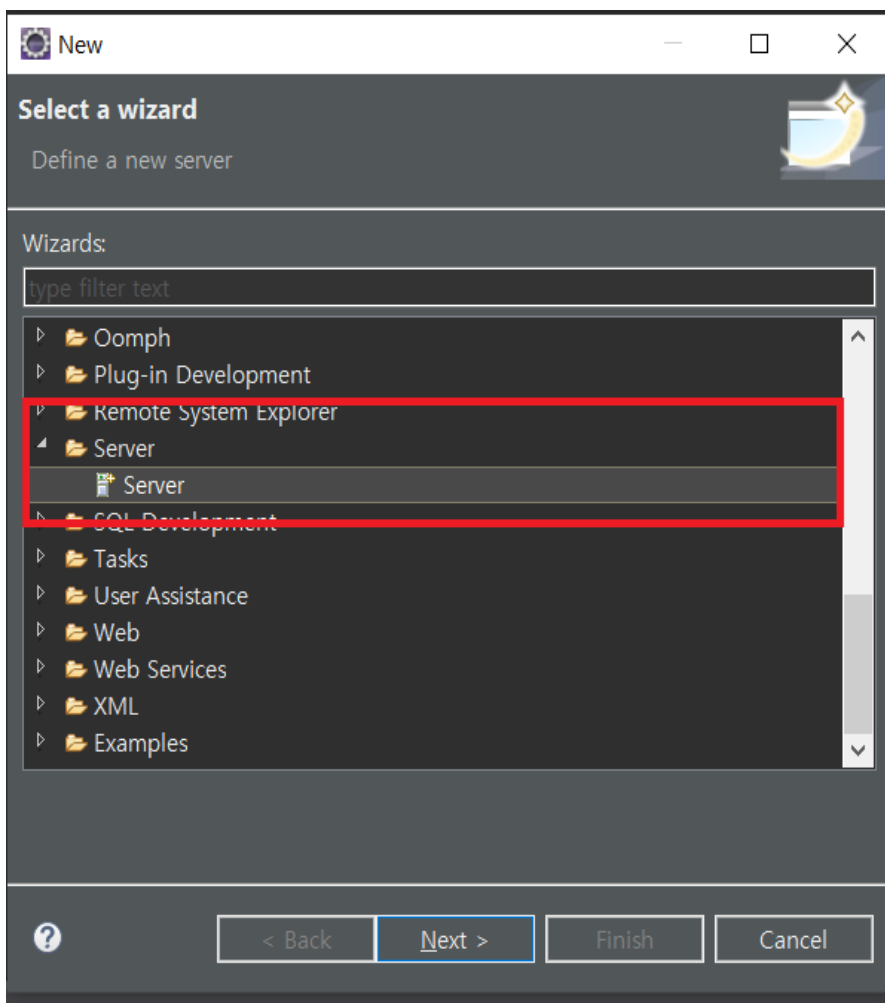
Please see the [README](#) file for packaging information. It explains what every file is for.

Binary Distributions

- Core:
 - [zip](#) ([pgp](#), [sha512](#))
 - [tar.gz](#) ([pgp](#), [sha512](#))
 - [32-bit Windows zip](#) ([pgp](#), [sha512](#))
 - [64-bit Windows zip](#) ([pgp](#), [sha512](#))
 - [32-bit/64-bit Windows Service Installer](#) ([pgp](#), [sha512](#))

2. Eclipse 와 Apache Tomcat 연동

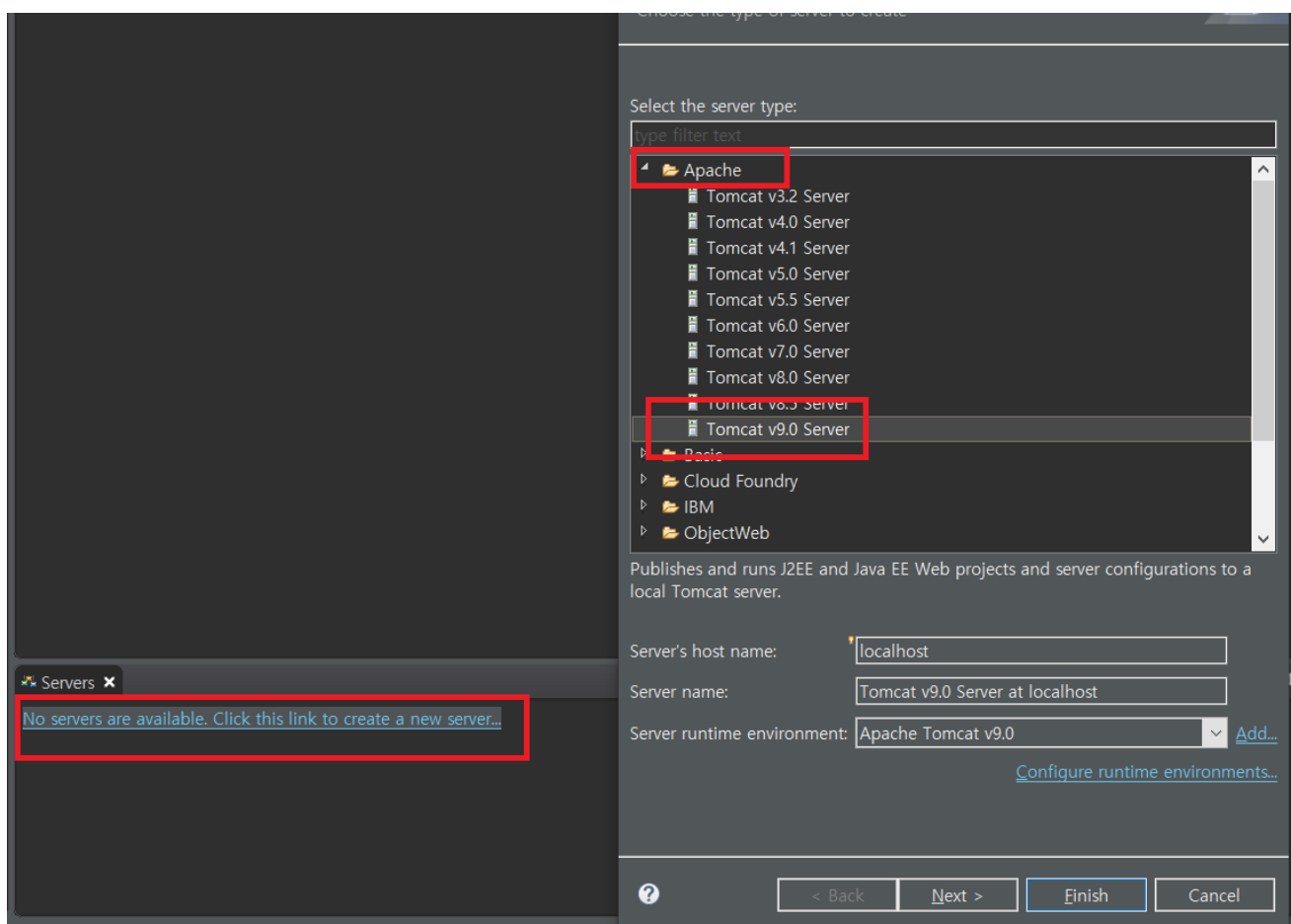
- Eclipse 메뉴에서 File -> New 선택한다.
- Server -> Server 선택한다.



- Eclipse Servers 탭에서 [No servers are available. Click this link to create a new servers](#)

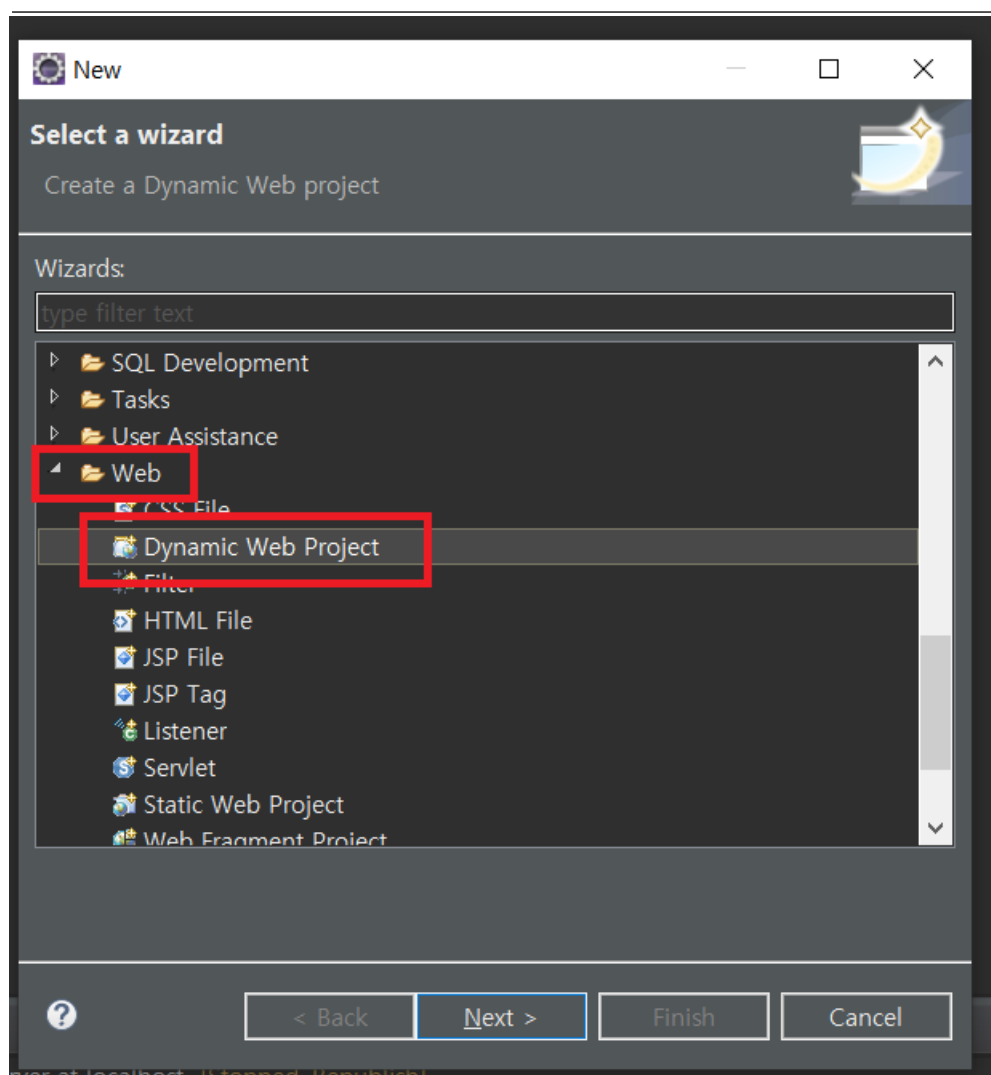
항목을 선택한다.

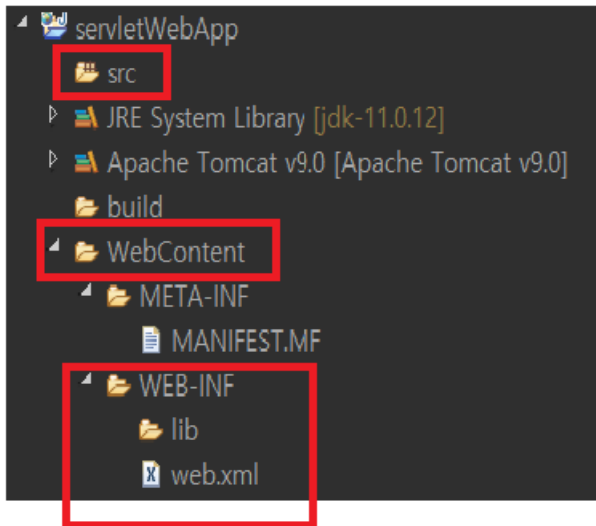
- Apache -> Tomcat v9.0 Server 를 선택한다.



3. Dynamic Web Project 생성하기

- Eclipse 메뉴에서 File -> New -> Other 를 선택한다
- Web -> Dynamic Web Project 를 선택한다.

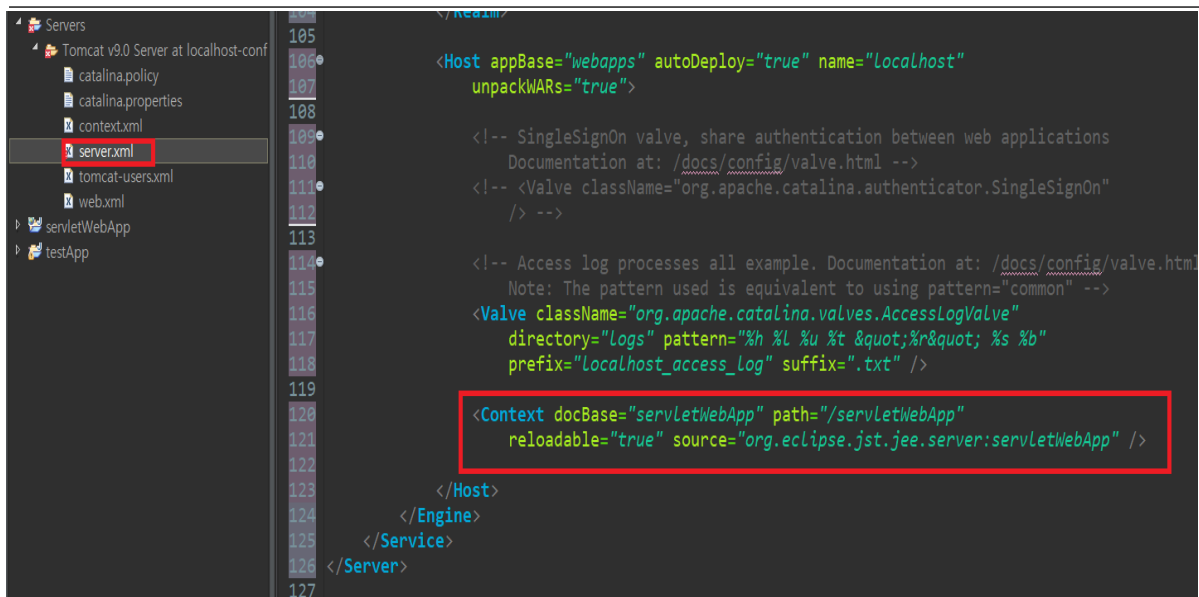




src	서블릿, 자바 파일 (*.java)
WebConent	html, css, js, jsp
WEB-INF/lib	라이브러리 파일 (mysql jdbc driver 등)
WEB-INF/web.xml	DD (Deployment Descriptor) 배포 서술자

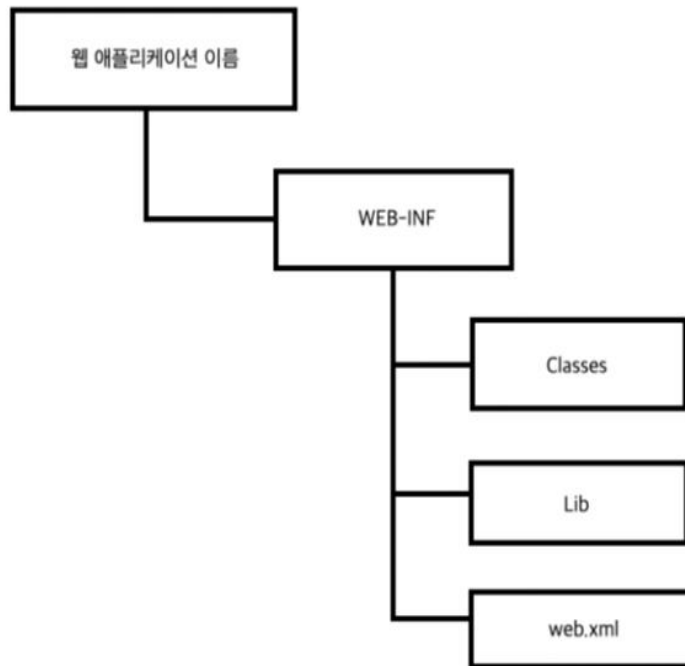
● Context

- 톰캣에서 실행되는 하나의 웹 어플리케이션을 의미한다.



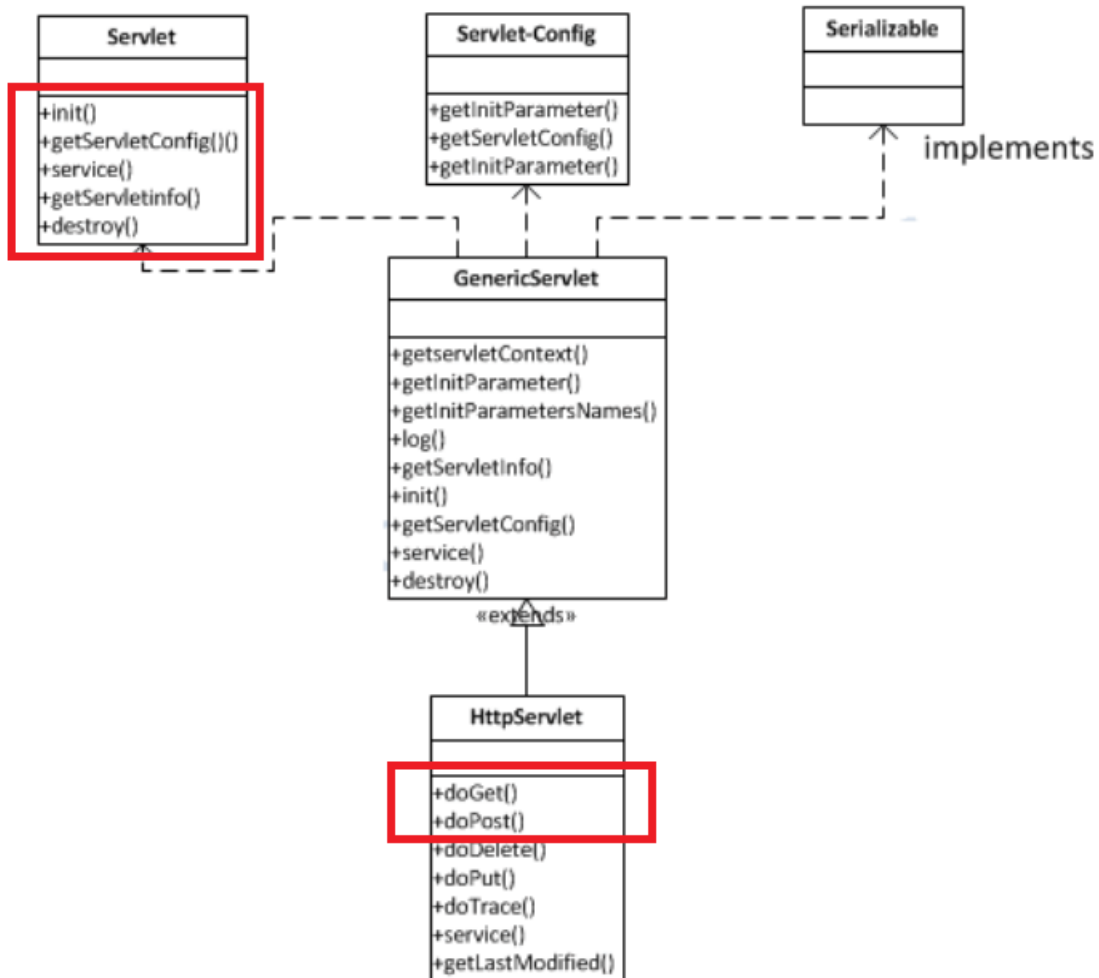
```
105
106
107 <Host appBase="webapps" autoDeploy="true" name="localhost"
108       unpackWARs="true">
109
110     <!-- SingleSignOn valve, share authentication between web applications
111          Documentation at: /docs/config/valve.html -->
112     <!-- <Valve className="org.apache.catalina.authenticator.SingleSignOn"
113          /> -->
114
115     <!-- Access log processes all example. Documentation at: /docs/config/valve.html
116          Note: The pattern used is equivalent to using pattern="common" -->
117     <Valve className="org.apache.catalina.valves.AccessLogValve"
118           directory="logs" pattern="%h %l %u %t &quot;%r&quot; %s %b"
119           prefix="localhost_access_log" suffix=".txt" />
120
121     <Context docBase="servletWebApp" path="/servletWebApp"
122           reloadable="true" source="org.eclipse.jst.jee.server:servletWebApp" />
123
124 </Host>
125 </Engine>
126 </Service>
127 </Server>
```

서블릿/JSP 웹 어플리케이션 디렉토리 구조



웹어플리케이션 폴더	HTML, 이미지, CSS, js, jsp
WEB-INF/classes	서블릿, 자바 클래스 파일 (*.class)
WEB-INF/lib	라이브러리 파일 (*.jar)
WEB-INF/web.xml	<p>DD (Deployment Descriptor) : 배포 서술자</p> <p>웹 어플리케이션을 실행하는데 필요한 설정정보를 기술한다.</p> <ul style="list-style-type: none"> · URL과 서블릿 매핑 · 컨텍스트 초기화 파라미터와 서블릿 초기화 파라미터 · 서블릿 리스너 · 서블릿 필터

Servlet API



Servlet Life Cycle

서블릿의 생명주기와 관련된 메서드: init(), service(), destroy()

- 서블릿의 생성과 실행, 소멸, 즉 생명주기와 관련된 메서드

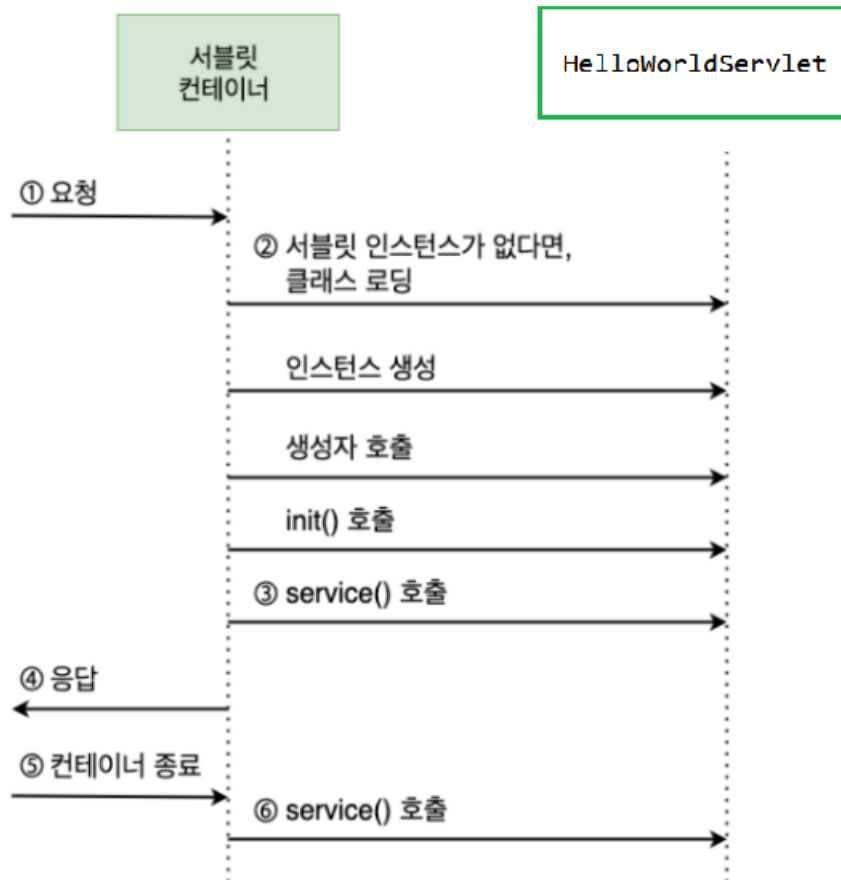
구분	설명
init()	서블릿 컨테이너가 서블릿을 생성한 후 초기화 작업을 수행하기 위해 호출하는 메서드
service()	실질적으로 서비스 작업을 수행하는 메서드
destroy())	서블릿 컨테이너가 종료되거나 웹 애플리케이션이 멈출 때, 또는 해당 서블릿을 비활성화 시킬 때 호출하는 메서드 ex) 자원의 해제, 데이터 저장, 마무리 작업

Servlet 인터페이스 기타 메서드: getServletConfig(), getServletInfo()

- 서블릿 정보를 추출할 필요가 있을 때 호출하는 보조 메서드

구분	설명
getServletConfig()	서블릿 설정 정보를 다루는 ServletConfig 객체를 반환 ex) 서블릿 이름, 서블릿 초기 매개변수 값, 서블릿 환경정보
getServletInfo()	서블릿을 작성한 사람에 대한 정보 ex) 서블릿 버전, 권리

How Servlet Works



Example of HelloWorldServlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

@WebServlet("/hello")
public class HelloWorldServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        PrintWriter pw = res.getWriter();
        pw.println( "Hello World!!" );
        pw.close();
    }
}
```

1. Annoation을 이용한 서블릿 설정

urlPatterns	웹 브라우저에서 서블릿 요청 시 사용하는 매핑 이름
name	서블릿 이름
loadOnStartup	웹 컨테이너 실행 시 서블릿이 로드되는 순서 지정
initParams	@WebInitParam 어노테이션을 이용한 초기화 파라미터 설정

2. /WEB-INF/web.xml 을 이용한 서블릿 설정

```
<servlet>
  <servlet-name>Hello</servlet-name>
  <servlet-class> HelloWorldServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>Hello</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

Get vs Post 요청 방식

1. GET 요청

● 특징

- URL 에 데이터를 포함 -> 데이터 조회에 적합
- 바이너리 및 대용량 데이터 전송 불가
- 요청라인과 헤드 필드의 최대 크기
 - HTTP 사양에는 제한사항 없음
 - 대용량 URL 로 인한 문제 발생 -> 웹 서버에 따라 최대 크기 제한
 - Microsoft IIS 6.0+: 16KB
 - Apache 웹 서버: 8KB

● 데이터 전달 형식

GET /servletWebApp/getData?name=kim&age=10 HTTP/1.1

● GET 요청의 문제점

- 주소창에 사용자가 입력한 정보가 그대로 노출되어 보안에 취약하다.
- 이미지나 동영상과 같은 바이너리 데이터 전송 불가하다.

2. POST 요청

● 특징

- URL에 데이터가 포함되지 않음 -> 외부 노출 방지
- 메시지 본문(Message body)에 데이터 포함

● 데이터의 전달 형식

- POST /servletWebApp/LoginServlet HTTP/1.1
- Content-Length: 22
- Content-Type: application/x-www-form-urlencoded
- id=java&password=1111

서버에 보내는 데이터는 메시지 본문(Message Body)에 포함되어 전송된다.

Redirect vs Dispatch 방식

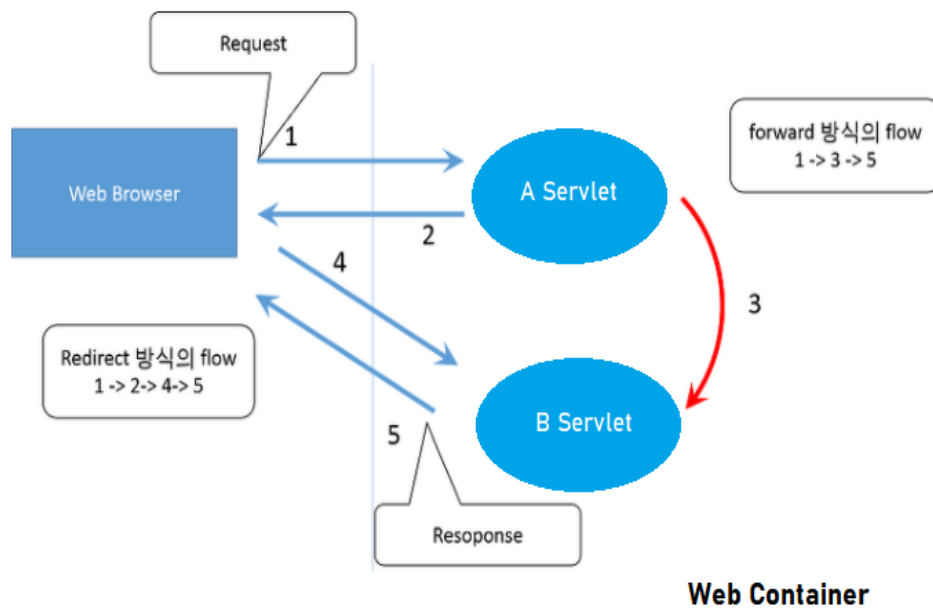
- Redirect

- `sendRedirect()` method sends the request with the new URL, which connects to this URL.
- Every time it creates a new request or response object.

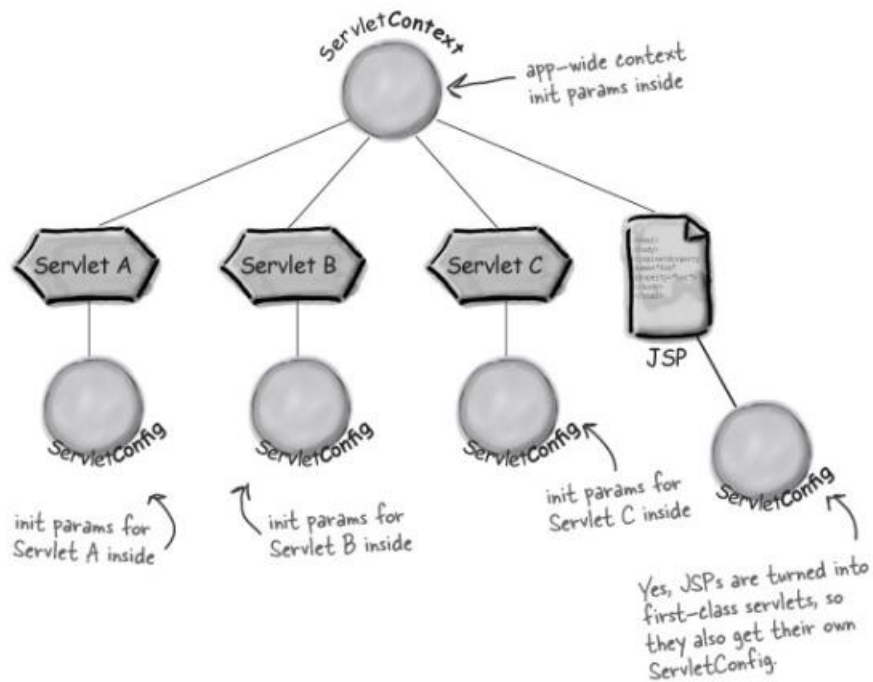
- Dispatch

- Inter-Servlet Communication
- RequestDispatcher
 1. forward
 2. include

forward	다른 서블릿으로 제어의 흐름을 넘기는 방식다. forward를 호출한 곳에서 출력한 내용은 버퍼에서 비워지며, 호출된 서블릿에서 클라이언트로 최종 응답을 수행한다.
include	현재 서블릿의 결과에 include된 서블릿의 결과를 삽입하는 방식이다.



ServletConfig vs ServletContext



• `javax.servlet.ServletContext`

특징

- 컨텍스트(웹 애플리케이션)마다 하나의 `ServletContext` 가 생성된다.
- 서블릿끼리 자원을 공유하는 데 사용한다.
- 컨테이너 실행 시 생성되고 종료 시 소멸한다.

제공하는 기능

- 서블릿에서 파일 접근
- 자원 바인딩
- 로그 파일
- 컨텍스트에서 제공하는 설정 정보 제공

● javax.servlet.ServletConfig

- 서블릿에 대한 초기화 작업이 가능하다.
- ServletContext 객체를 구할수 있다.

※ load-on-startup이란?

- 톰캣 컨테이너가 실행되면서 미리 서블릿 실행한다.
- 지정한 숫자가 0 보다 크면 톰캣 컨테이너가 실행되면서 서블릿 초기화 작업을 수행한다.
- 지정한 숫자는 우선순위를 의미, 작은 숫자부터 먼저 초기화 작업이 수행된다.

Attributes Types in Servlet

1. `setAttribute(String name, Object value)`
2. `getAttribute(String name)`
3. `removeAttribute(String name)`

※ Scope(영역) 이란?

- 웹 컨테이너에서 객체 또는 변수가 생성된 후 유효할 수 있는 범위를 말한다.

● Request Scope

- `HttpServletRequest`
- 요청을 받아서 응답할때 까지 객체가 유효한 영역이다.
- Servlet 에서 `forward` 또는 `include` 를 사용하여 request 영역에 바인딩된 객체를 공유한다.

● Session Scope

- `HttpSession`
- 하나의 웹 브라우저 당 1 개의 session 객체가 생성된다.
- 같은 브라우저 내에서 요청되는 서블릿 또는 JSP 들은 웹 브라우저의 세션이 유지되는 동안

객체를 공유하게 되는데, 이를 세션 영역이라고 한다.

- `request.getSession()` 메서드를 호출하여 세션 객체를 얻을 수 있다.

● Application(Context) Scope

- `ServletContext`
- 하나의 웹 어플리케이션 당 1 개의 `application` 객체가 생성된다.
- 같은 웹 어플리케이션 내에서 요청되는 서블릿 또는 JSP 들은 해당 웹 어플리케이션이 종료될 때 까지 객체를 공유하게 되는데 이를 어플리케이션 영역이라고 한다.
- `request.getServletContext()` 메서드를 호출하여, 어플리케이션(컨텍스트) 영역 객체를 구할 수 있다.

HTTP 프로토콜

- stateless(상태를 유지할 수 없는) 프로토콜이다.

세션 트래킹

- 로그인 상태, 장바구니

● 쿠키

- 사용자 정보를 웹 브라우저가 실행되는 클라이언트 PC 에 저장한다. (파일)
- (name, value) 쌍으로 이루어진 정보
- 저장 정보 용량 제한
- 보안 취약
- 도메인당 쿠키가 생성된다.
- 서블릿에서 `javax.servlet.http.Cookie` 제공한다.

● 쿠키 생성 과정

1. 브라우저로 사이트에 접속한다.
2. 서버는 정보를 저장한 쿠키를 생성한다.
3. 생성한 쿠키를 브라우저로 전송한다.
4. 브라우저는 서버로부터 받은 쿠키를 파일에 저장한다.
5. 재접속 시, 서버는 브라우저에게 쿠키를 요청하고, 브라우저는 서버로 쿠키를 보낸다.
6. 서버는 쿠키를 이용해서 작업한다.

Session Management

● Session 소개

- 세션은 웹 브라우저 당 하나가 생성된다. (SessionID)
- 세션은 사용자 정보를 서버 메모리에 저장하므로 쿠키에 비해 보안에 강하다.
- 세션은 세션 아이디를 **session cookie(웹 브라우저 메모리에 생성)**로 관리하며 세션 아이디는 `server` 가 종료되거나, 유효기간이 만료하거나, `client browser` 가 종료되면 삭제된다.
- 서블릿에서 `javax.servlet.http.HttpSession` 제공한다.

● Session mechanics

1. 클라이언트가 서버에 리소스를 요청한다.
2. 서버는 클라이언트 요청 헤더의 session cookie 를 통해 session-id 를 확인한다.
3. session-id 가 존재하면, server 는 session-id 가 유효한지 확인 후 client 의 request 를 처리하고 응답(response) 한다.
4. session-id 가 존재하지 않으면, server 는 session-id 를 생성한 후 set-cookie 를 통해 response-header 에 추가하여 client 에 response 한다.
5. 위 4 번 항목 처리 후 client 는 server 에 request 시 server 로부터 response 한 session-id 를 request-header 에 추가하여 request 한다.

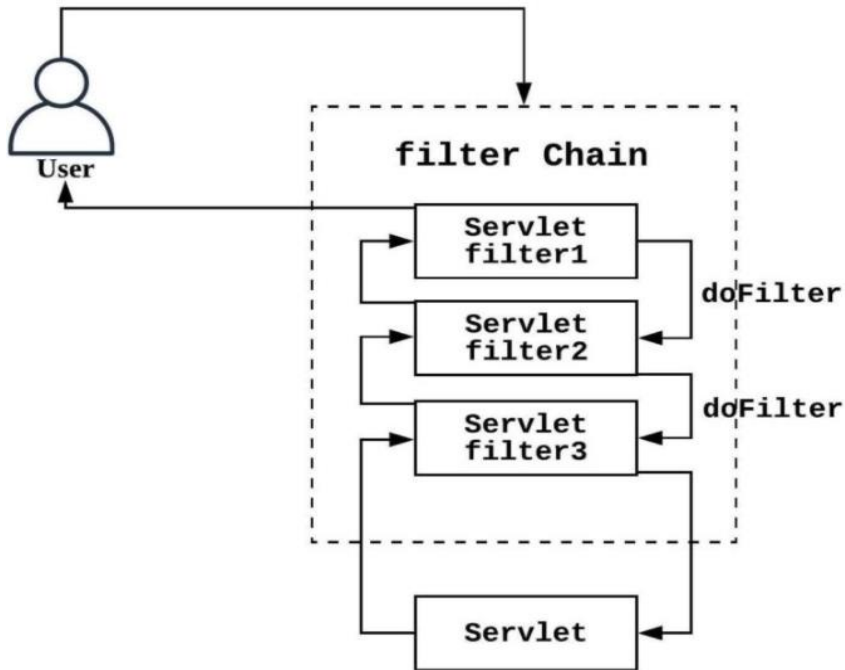
● Session 종료 시기

1. 타임아웃(timeout)
2. HttpSession 객체의 invalidate() 호출

3. 애플리케이션(application) 또는 server 종료

Servlet Filter

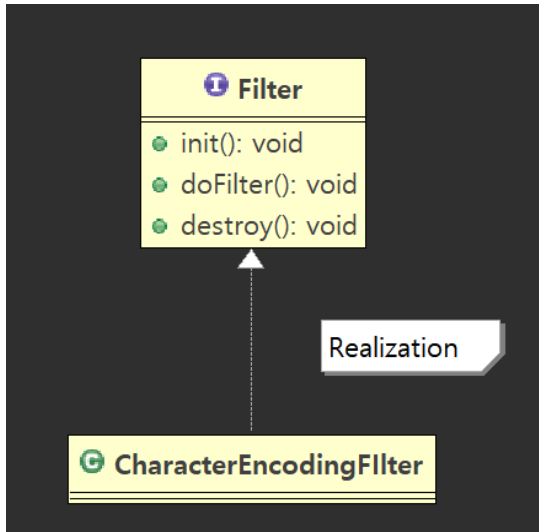
- 서블릿 필터를 사용하면 클라이언트의 요청을 가로채어서 서버 컴포넌트의 공통적인 기능 (사용자 인증, 로깅)들을 수행할 수 있다.



● 필터

- 사용자 인증 필터
- 로깅 및 감시(Audit) 필터
- 인코딩
- 암호화 필터
- URL 및 기타 정보들을 캐싱하는 필터

● 인코딩 필터



1. 어노테이션 설정

```
@WebFilter(  
    value= "/*",  
    initParams = {@WebInitParam(name="encoding", value="utf-8")})
```

2. /WEB-INF/web.xml 설정

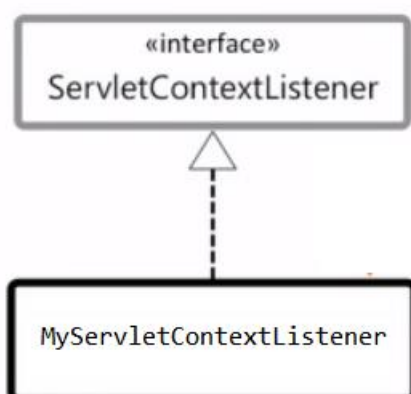
```
<!-- 필터 선언 -->  
<filter>  
    <filter-name>CharacterEncodingFilter</filter-name>  
    <filter-class> util.CharacterEncodingFilter </filter-class>  
    <init-param>  
        <param-name>encoding</param-name>  
        <param-value>UTF-8</param-value>  
    </init-param>  
</filter>  
  
<!-- 필터 URL 매핑 -->  
<filter-mapping>  
    <filter-name>CharacterEncodingFilter</filter-name>  
    <url-pattern>/*</url-pattern>  
</filter-mapping>
```

Servlet Listener

- 서블릿에서 발생하는 이벤트를 처리할 수 있도록 제공한다.

- ServletContextListener

- 컨텍스트 객체의 생성/소멸 시 발생하는 이벤트를 처리한다.
- HttpSessionBindingListener를 제외한 모든 Listener는 **@WebListener**를 사용해서 Listener로 등록해야 한다.



/WEB-INF/web.xml 설정

```
<listener>
```

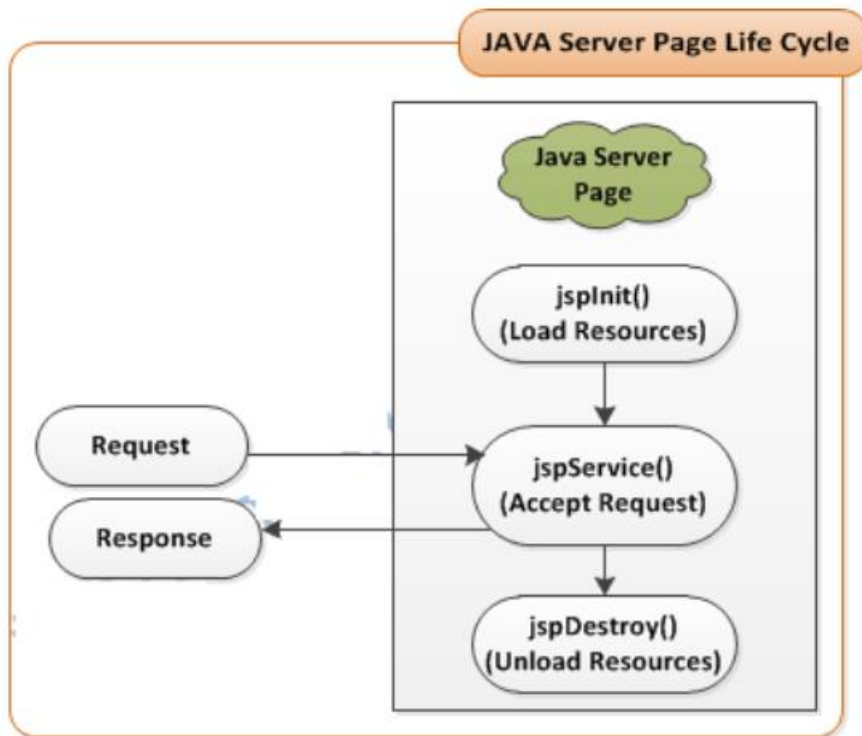
```
    <listener-class>MyServletContextListener</listener-class>
```

```
</listener>
```

JSP 소개

JSP Life Cycle

1. `jspInt()`
2. `jspService()`
3. `jspDestroy()`



JSP Directives

- JSP 지시어는 JSP 파일의 속성을 기술 하는 곳이다.
- 웹 컨테이너에게 해당 페이지를 어떻게 처리해야 하는지 정보를 담고 있다.

```
<%@ 지시어 속성1="값1" 속성2="값2" ... %>
```

· 지시어는 크게 3가지 로 나눌 수 있다.

1. page 지시어
2. include 지시어
3. taglib 지시어

● include Directive

· 해당 JSP 파일 내에 다른 HTML파일이나 JSP 페이지를 삽입할 수 있게 도와주는 기능을 하는 지시어이다.

```
<%@ include file="삽입할 파일 이름" %>
```

● taglib Directive

· JSP 기능을 확장하기 위해 만들어진 커스텀 태그 라이브러리를 JSP 파일에서 사용 하기 위한 지시어이다.

```
<%@ taglib uri="삽입할 파일 이름" prefix="삽입할 파일 이름" %>
```

● page Directive

· JSP 페이지를 컨테이너에서 처리하는 데 필요한 속성을 기술하는 부분이다.

language	스크립트 언어의 유형 (기본값 java)
contentType	MINME 형식 및 charset 설정
pageEncoding	JSP 소스 파일이 저장될 시 사용될 인코딩을 지정한다.
import	JSP 파일에서 외부 자바 패키지나 클래스를 불러올때 사용한다.
errorPage	현재 페이지에서 오류가 발생할 경우 오류를 처리할 페이지를 지정한다.
isErrorPage	예외를 처리하는 페이지인지 여부를 설정할 때 사용한다. (기본값 false)
session	JSP 페이지가 HttpSession을 사용할 지에 대한 여부를 지정한다. (기본값 true)
buffer	출력 버퍼의 크기를 지정하는 속성이다. (기본값 8kb)
autoFlush	버퍼를 자동으로 비울지 여부를 나타내는 속성이다. (기본값 : true)

JSP 스크립트 요소

1. 스크립틀릿 (scriptlet)
2. 표현식 (Expression)
3. 선언부 (Declaration)

- 스크립틀릿 (scriptlet)

- JSP에서 자바코드를 실행할 때 사용하는 자바코드블록이다.
- `<% %>` 사이에 자바코드가 온다.

- 표현식

- 어떤 값을 출력 결과에 포함시키고자 할 때 사용한다.
- `<%= %>` 사이에 출력할 값이 위치한다.

- 선언부

- JSP의 스크립틀릿이나 표현식에서 사용할 수 있는 메소드를 작성할 때 사용한다.
- `<%! %>`

JSP 액션 태그

- 액션태그란 JSP 페이지 내에서 어떤 동작을 하도록 지시하는 태그이다.

- forward

- forward는 현재 페이지를 다른 페이지로 전환할 때 사용한다.


```
<jsp:forward page="이동하고자하는 페이지 경로">
```

● param

- forward 액션 태그와 param을 이용해서 다른 페이지로 데이터를 전달할 수 있다.

```
<jsp:forward page="이동하고자하는 페이지 경로">  
    <jsp:param name="email" value="java@gmail.com">  
    <jsp:param name="userName" value="Julia">  
</jsp:forward>
```

● include

- JSP 페이지 내에 다른 페이지를 삽입하는 액션태그 입니다.

```
<jsp:include page="삽입하고자 하는 페이지 경로">
```

● 자바빈과 <jsp:useBean> 액션태그

- <jsp:useBean>은 application, session, request, page 영역에 저장된 자바 객체를 꺼낼 수 있다.

```
<jsp:useBean id="이름" scope="page | request | session | application"  
            class="클래스명" />
```

id	JSP 페이지에서 자바빈 객체에 접근할 때 사용하는 이름
class	패키지 이름을 포함한 자바빈 클래스 이름 (FQCN)
scope	자바빈 객체가 저장될 영역을 지정한다. (기본값 page)

● <jsp:setProperty>

```
<jsp:setProperty name="자바빈" property="이름" value="값" />
```

name	프로퍼티 값을 변경할 자바빈 객체의 이름. <jsp:useBean> 액션 태그의 id 속성에서 지정한 값을 사용
property	값을 지정할 프로퍼티의 이름
value	프로퍼티 값. 표현식 사용가능

Implicit Objects

- JSP 페이지 내에서 제공하는 특수한 레퍼런스 타입의 변수이다.
- JSP 페이지가 서블릿으로 변환될 때 웹 컨테이너에 의해서 자동으로 생성된다.

· 객체 생성 없이 바로 사용할 수 있는 JSP의 객체를 뜻한다.

● 기본 객체와 영역(scope)

page 영역	하나의 JSP 페이지를 처리할 때 사용되는 영역. <code>pageContext</code> 기본객체
request 영역	하나의 HTTP 요청을 처리할 때 사용되는 영역. <code>request</code> 기본객체
session 영역	하나의 웹 브라우저와 관련된 영역. <code>session</code> 기본객체
application	하나의 웹 어플리케이션과 관련된 영역. <code>application</code> 기본객체

● JSP 내장 객체

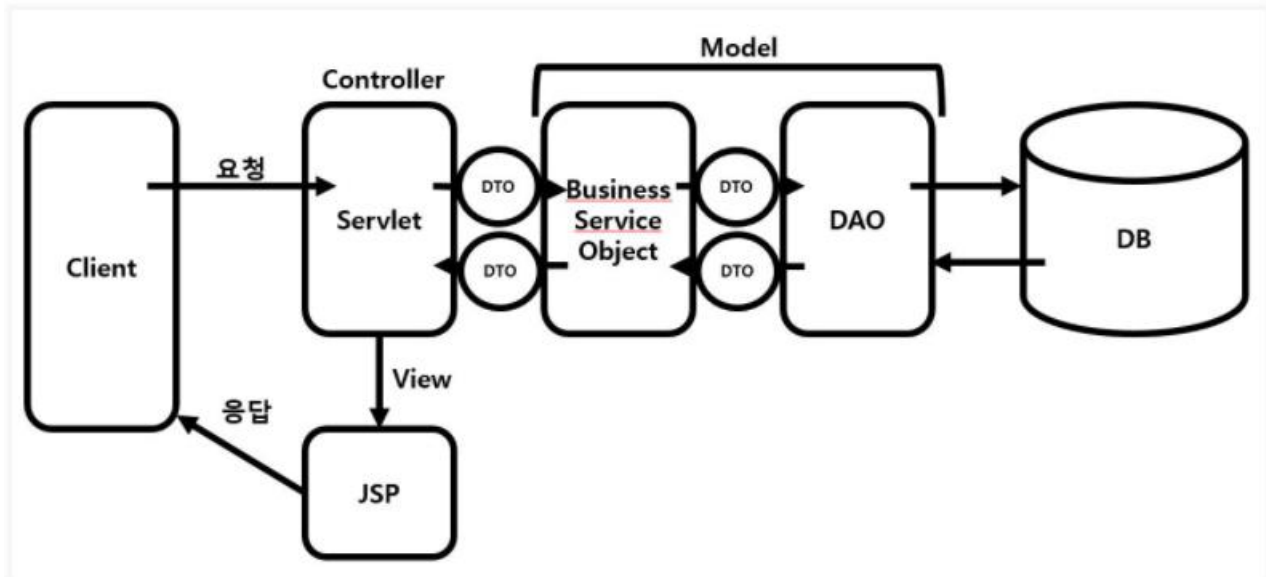
request	웹 브라우저의 요청 정보를 저장하고 있는 객체 <code>javax.servlet.http.HttpServletRequest</code>
---------	---

response	웹 브라우저의 요청에 대한 응답 정보를 저장하고 있는 객체 <code>javax.servlet.http.HttpServletResponse</code>
out	JSP 페이지에 출력할 내용을 가지고 있는 출력 스트림 객체 <code>javax.servlet.jsp.JspWriter</code>
session	하나의 웹 브라우저의 정보를 유지하기 위한 세션 정보를 저장하고 있는 객체 <code>javax.servlet.http.HttpSession</code>
application	웹 어플리케이션 Context 의 정보를 저장하고 있는 객체 <code>javax.servlet.ServletContext</code>
pageContext	JSP 페이지에 대한 정보를 저장하고 있는 객체 <code>javax.servlet.jsp.PageContext</code>
page	JSP 페이지를 구현한 자바 클래스 객체 <code>java.lang.Object</code>
config	JSP 페이지에 대한 설정 정보를 저장하고 있는 객체 <code>javax.servlet.ServletConfig</code>
exception	JSP 페이지에서 예외가 발생한 경우 사용되는 객체 <code>javax.lang.Throwable</code>

MVC Model 2 Pattern

- 웹 어플리케이션 개발에 사용되는 디자인 패턴이다.

- Model - View - Controller의 약자이다.
- 사용자 인터페이스(프리젠테이션 로직)와 비즈니스 로직과 데이터 접근 로직을 분리하여 개발할 수 있다.



Model	비즈니스 로직과 데이터 접근 로직을 담당한다. (자바코드)
View	뷰는 사용자에게 보여주는 인터페이스로 JSP 가 담당한다.
Controller	서블릿이 흐름을 제어하는 컨트롤러의 역할을 수행한다. Front Controller pattern Command pattern

EL (Expression Language)

· EL은 JSP의 출력 문법을 대체하는 표현 언어이다.

`${ 영역객체에 바인딩된 속성이름 }`

● EL 내장 객체

pageScope	pageContext 기본 객체에 저장된 속성의 <속성이름, 값> 매핑을 저장한 Map 객체
requestScope	request 기본 객체에 저장된 속성의 <속성이름, 값> 매핑을 저장한 Map 객체
sessionScope	session 기본 객체에 저장된 속성의 <속성이름, 값> 매핑을 저장한 Map 객체
applicationScope	application 기본 객체에 저장된 속성의 <속성이름, 값> 매핑을 저장한 Map 객체
param	요청 파라미터 값을 가져오기 위한 객체 요청 파라미터의 <파라미터 이름, 값> 매핑을 저장한 Map 객체
paramValues	요청 파라미터 값을 가져오기 위한 객체 요청 파라미터의 <파라미터 이름, 값배열> 매핑을 저장한 Map 객체
header	헤더 값을 가져오기 위한 객체 요청 정보의 <헤더 이름, 값> 매핑을 저장한 Map 객체
headerValues	헤더 값을 가져오기 위한 객체 요청 정보의 <헤더 이름, 값배열> 매핑을 저장한 Map 객체
cookie	쿠키 값을 가져오기 위한 객체 <쿠키 이름, Cookie> 매핑을 저장한 Map 객체
initParam	초기화 파라미터 <이름, 값> 매핑을 저장한 객체
pageContext	pageContext 객체에 접근하기 위한 객체

JSTL

- JavaServer Pages Standard Tag Library
- JSTL은 태그를 통해 JSP 코드를 관리하는 라이브러리로서, JSP의 가독성이 좋아진다.

● JSTL 사용법

- JSTL은 추가 적인 jar 파일이 필요하다
- <http://tomcat.apache.org/download-taglibs.cgi> 여기에서 다음의 파일 3개의 받은 후 WEB-INF/lib/ 에 복사한다.

Jar Files

- [Binary README](#)
- Impl:
 - [taglibs-standard-impl-1.2.5.jar](#) (pgp, sha512)
- Spec:
 - [taglibs-standard-spec-1.2.5.jar](#) (pgp, sha512)
- EL:
 - [taglibs-standard-jstlel-1.2.5.jar](#) (pgp, sha512)
- Compat:
 - [taglibs-standard-compat-1.2.5.jar](#) (pgp, sha512)

● Core Tags

· `<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>`

Tags	Description
c:out	<code><% = ... %></code> 태그 작동 방식과 유사한 표현식의 결과를 표시
c:import	상대 또는 절대 URL을 검색하여 내용을 'var'의 문자열, 'varReader'의 Reader 또는 페이지에 표시
c:set	평가중인 표현식의 결과를 'scope'변수에 설정
c:remove	특정 범위에서 지정된 범위 변수를 제거하는 데 사용
c:catch	본문에서 발생하는 Throwable 예외를 포착하는 데 사용
c:if	조건을 테스트하는 데 사용되는 조건부 태그이며 표현식이 참인 경우에만 본문 내용을 표시
c:choose, c:when, c:otherwise	평가 된 조건이 true 인 경우 본문 내용을 포함하는 간단한 조건부 태그
c:forEach	기본 반복 태그, 고정 된 횟수 또는 초과 수집 동안 중첩 된 본문 내용을 반복
c:forEachTokens	제공된 델리 미터로 분리 된 토큰을 반복
c:param	포함하는 'import'태그의 URL에 매개 변수를 추가
c:redirect	브라우저를 새 URL로 리디렉션하고 컨텍스트 기준 URL을 지원
c	url

● function Tags

· 많은 표준 함수를 제공, 대부분은 일반적인 문자열 조작 함수

· `<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>`

JSTL Functions	Description
<code>fn:contains()</code>	프로그램에서 지정된 하위 문자열을 포함하는 입력 문자열인지 테스트하는 데 사용
<code>fn:containsIgnoreCase()</code>	입력 문자열에 대소 문자를 구분하지 않고 지정된 하위 문자열이 포함되어 있는지 테스트하는 데 사용
<code>fn:endsWith()</code>	입력 문자열이 지정된 접미어로 끝나는 지 테스트하는 데 사용
<code>fn:escapeXml()</code>	XML 마크업으로 해석되는 문자를 escape
<code>fn:indexOf()</code>	지정된 하위 문자열이 처음 나타나는 문자열 내에서 인덱스를 반환
<code>fn:trim()</code>	문자열의 양쪽 끝에서 공백을 제거
<code>fn:startsWith()</code>	주어진 문자열이 특정 문자열 값으로 시작되는지 여부를 확인하는 데 사용.
<code>fn:split()</code>	문자열을 하위 문자열 배열로 분할
<code>fn:toLowerCase()</code>	문자열의 모든 문자를 소문자로 변환
<code>fn:toUpperCase()</code>	문자열의 모든 문자를 대문자로 변환
<code>fn:substring()</code>	주어진 시작 및 끝 위치에 따라 문자열의 하위 집합을 반환
<code>fn:substringAfter()</code>	특정 하위 문자열 다음에 문자열의 하위 집합을 반환
<code>fn:substringBefore()</code>	특정 하위 문자열 앞의 문자열 하위 집합을 반환
<code>fn:length()</code>	문자열 내부의 문자 수 또는 컬렉션의 항목 수를 반환
<code>fn:replace()</code>	모든 문자열을 다른 문자열 시퀀스로 바꿈

Connection Pool

- 커넥션 풀이란 DB와 연결하는 커넥션을 미리 생성해두고 풀에 저장해두었다가 필요할 때 꺼내쓰고, 사용후에는 다시 풀에 반환하는 기법을 말한다.
- 커넥션을 미리 생성해 두기 때문에 커넥션을 사용자가 DB를 사용할 때 마다 매번 생성하는 것보다 더 빠른 속도를 보장한다. 또한 커넥션의 최대 생성 갯수도 제어가 가능하므로 과부하를 방지할 수 있다.

• Database Connection Pool (DBCP 2) Configurations

1. \$CATALINA_HOME/lib/tomcat-dbcp.jar
2. JDBC Driver's jar into \$CATALINA_HOME/lib .
3. WebContent/META-INF/context.xml

<Context>

<Resource name="jdbc/mysqlldb" auth="Container"

type="javax.sql.DataSource" maxTotal="100" maxIdle="30"

maxWaitMillis="10000" username="jdbc" password="jdbc1234"

```
driverClassName="com.mysql.cj.jdbc.Driver"
```

```
factory="org.apache.tomcat.dbcp.dbcp2.BasicDataSourceFactory"
```

```
url="jdbc:mysql://localhost:3306/jdbcdtb" />
```

```
</Context>
```

4. WEB-INF/web.xml

```
<resource-ref>
```

```
<description>DB Connection</description>
```

```
<res-ref-name>jdbc/mysqdb</res-ref-name>
```

```
<res-type>javax.sql.DataSource</res-type>
```

```
<res-auth>Container</res-auth>
```

```
</resource-ref>
```

5. 자바 코드

```
Context initContext = new InitialContext();
```

```
Context envContext = (Context)initContext.lookup("java:/comp/env");
```

```
DataSource ds = (DataSource)envContext.lookup("jdbc/mysqdb");
```

```
Connection conn = ds.getConnection();
```