Servlet

Servlet

Servlet

Servlet

- Server + Applet의 합성어
- JAVA 언어를 이용하여 사용자의 요청을 받아 처리하고 그 결과를 다시 사용자에게 전송하는 역할의 Class파일
- 웹에서 동적인 페이지를 JAVA로 구현한 서버측 프로그램

※ 관련 패키지와 클래스는 tomcat에서 제공하는 API문서에서 확인 가능
https://tomcat.apache.org/tomcat-8.5-doc/servletapi/

Servlet

Servlet의 역사

JAVA 언어의 창시자인 제임스 고슬링은 1995년 자바를 발표하며 자바로 구현할 수 있는 서버프로그래밍 기술에 대해서도 염두에 두고 있었지만, 해당 개념이 실제 구현 가능할 정도의 제품화가 되어있지 않은 상태였다. 당시 SUN사에서는 이를 서버로 구현할 수 있는 제품이 없어 잠시 미룰 수 밖에 없었다.
즉, JAVA EE Platform이 제품화 되어있지 않은 시기였다.

얼마 뒤, 자바 팀의 일원이었던 파바니 디완지는 자바 서버 기술에 대한 필요성을 느껴 Servlet의 개념을 고안하였고, 이 개념을 토대로 프로젝트를 진행하여 Servlet의 구현 및 제품화에 성공하였고, 이 기술은 1997년 6월에 Servlet 1.0을 공식 발표하면서 JAVA EE의 제품화에 포함되었다.

Servlet

Servlet의 버전

버전	발표일	지원 platform	주요 변경 내용
Servlet 1.0	1997.06	-	
Servlet 2.1	1998.11	JDK 1.1	RequestDispatcher, ServletContext 요소를 처음으로 정의
Servlet 2.3	2001.08	J2EE 1.3 J2SE 1.2	Filter 클래스 추가
Servlet 2.4	2003.11	J2EE 1.4 J2SE 1.3	web.xml문서를 통한 XML 스키마 사용
Servlet 2.5	2005.09	JAVA EE 5 JAVA SE 5	어노테이션 지원 기능 추가
Servlet 3.0	2009.12	JAVA EE 6 JAVA SE 6	어노테이션 지원 범위 확대, 보안강화, 비동기서블릿 지원
Servlet 3.1	2013.05	JAVA EE 7	Non-blocking I/O 지원, HTTP 1.1 지원

※ J2EE, J2SE 1.5 버전 부터 표기 방식을 JAVA EE / SE로 변경

Servlet

Servlet 설계 규약

1. 모든 서블릿은 javax.servlet.Servlet 인터페이스를 상속 받아 구현한다.
2. 서블릿을 구현 시 Servlet 인터페이스와 ServletConfig 인터페이스를 javax.servlet.GenericServlet에 구현한다.
3. HTTP 프로토콜을 사용하는 서블릿은 javax.servlet.http.HttpServlet클래스를 상속받는데 HttpServlet클래스는 javax.servletGenericServlet을 상속하여 구현된 클래스다.
4. 서블릿의 Exception을 처리하기 위해서는 javax.servlet.ServletException을 상속 받아야 한다.

Servlet

Servlet 동작 구조

Servlet 컨테이너
웹 서버 또는 응용 프로그램 서버의 일부로, 웹 서버에서 온 요청을 받아서 서블릿 class를 관리하는 역할(생명주기)을 한다. 컨테이너의 서블릿에 대한 설정은 Deployment Descriptor(web.xml)파일을 이용한다.

Servlet

배포 서술자(Deployment Descriptor)

- 어플리케이션에 대한 전체 설정정보를 가지고 있는 파일로 이 정보를 가지고 웹 컨테이너가 서블릿을 구동
- xml파일로 태그로 이루어져 있음
- 어플리케이션 폴더의 WEB-INF폴더에 web.xml 파일

설정정보

- Servlet 정의 / Servlet 초기화 파라미터
- Session 설정 파라미터
- Servlet/jsp 매핑 / MIME type 매핑
- 보안설정
- Welcome file list 설정
- 에러페이지, 리소스, 환경변수 설정

Servlet

web.xml 파일 주요 태그

태그	설명
<web-app>	루트속성, 문법식별자 및 버전의 정보를 속성값으로 설정
<context-param>	웹 어플리케이션에서 공유하기 위한 파라미터 설정
<mime-mapping>	특정 파일 다운로드시 파일이 깨지는 현상을 방지하기 위한 설정
<servlet> <servlet-class> <servlet-mapping>	서블릿 매핑
<welcome-file-list>	시작페이지 설정
<filter>	필터정보 등록
<error-page>	에러발생시 안내페이지 설정

※ 이외에도 더 많은 태그들이 존재

Servlet

서블릿 매핑

- client가 servlet에 접근할 때 원본 클래스명이 아닌 다른 명칭으로 접근 시 사용 접근명칭과 클래스명을 매핑해주는 것

- web.xml을 이용하는 방법
  web.xml에 태그를 이용하여 매핑 정보를 등록

- @annotation을 이용하는 방법
  해당 서블릿 클래스 상단에 @annotation을 이용하여 매핑정보 등록

Servlet

web.xml을 이용한 방법

```
<servlet>
        <servlet-name>mapping 명칭</servlet-name>
        <servlet-class>실제 클래스 명칭 </servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>mapping 명칭</servlet-name>
        <url-pattern>사용자 접근 명칭</url-pattern>
</servlet-mapping>
```

Servlet

@annotation을 이용한 방식

```
@web-servlet(“매핑명칭”)
public class 서블릿명칭 extends HttpServlet{

        Servlet 클래스 내용

}
```

※ @annotation을 이용한 방식은 Servlet 파일 생성 시 바로 지정이 가능하고 편리하여 더 많이 사용

Servlet

server.xml

- WAS서버에 대한 설정을 변경할 수 있는 파일

설정정보

- Context Path 설정(서버 내 애플리케이션 설정)
- 어플리케이션 포트 설정
- default 접속 경로 설정
- 특정 이벤트 설정

Servlet

Context Path

- 어플리케이션에 접근하는 경로
- 어플리케이션의 root 경로(최상위 경로)
- 하나의 WAS에 여러 프로젝트를 이용하여 다양한 어플리케이션을 사용이 가능한데 이를 구별해 주는 역할

접속방법

http://서버아이피:[포트번호]/[Context Path]/Servlet 매핑 값

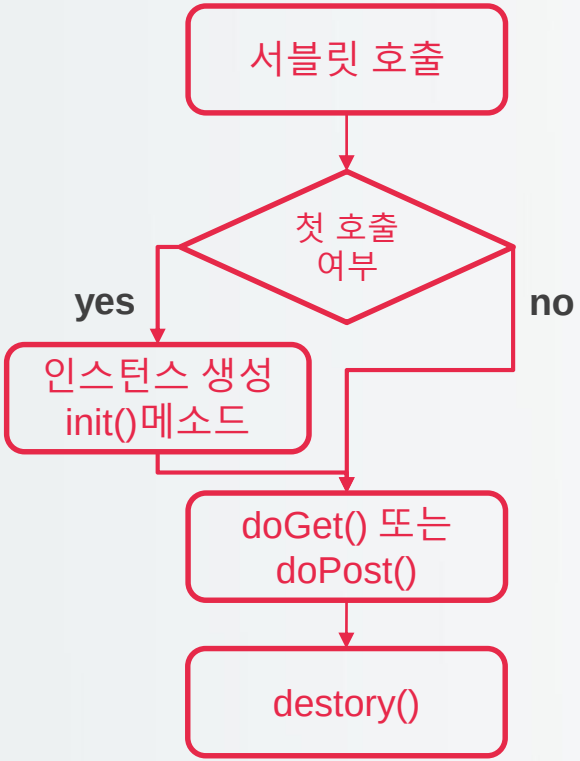
- 포트번호 80을 사용하는 경우 포트번호 생략가능(그 외 포트는 포트번호 입력)
- 사용하는 어플리케이션이 1개 인 경우 Context Path또한 생략 가능하게 설정 가능

Servlet

서블릿 라이프 사이클

1. 첫번째 요청인 경우, 객체를 생성하며 init() 메소드 호출
2. 이후 작업이 실행 될 때마다 HTTP Type에 따른 doGet(), doPost()메소드 호출
3. 최종적으로 서블릿이 서비스되지 않을 때 destroy()메소드 호출

※ destroy()메소드는 보통 서버가 종료되었을 때, 내용이 변경되어 재컴파일 될때 호출



Servlet 객체/메소드

Servlet

사용자 데이터 전송 방식

- get 방식
  URL창 “?”뒤에 데이터를 입력하는 방법(쿼리스트링)으로 전송
  전송할 데이터가 여러 개인 경우 &를 이용하여 여러 개 전송
  데이터 검색에 많이 사용되며, 데이터 크기에 한계가 있으며, 보안에 취약
- post 방식
  HTTP 헤더의 내용으로 보내는 방식으로 데이터 크기에 제한이 없고, 보안이 뛰어남

※ Servlet이 두 방식 중 하나로 전달 받으면 해당하는 메소드를 호출
html의 <form>태그에서 method속성을 이용하여 전송 방법 결정 : 명시하지 않으면 get

Servlet

doGet()

- client에서 데이터를 get방식으로 전송하게 되면 호출되는 메소드

doPost()

- client에서 데이터를 post방식으로 전송하게 되면 호출되는 메소드

※ ServletException을 처리해야함

Servlet

HttpServletRequest(interface)

- **Http Servlets을 위한 요청정보(request information)를 제공**
- **인터페이스 구현은 컨테이너가 설정하며, 메소드만 사용**
- **javax.servlet.ServletRequest를 상속**

메소드	설명
getParameter(String)	client가 보내준 값이 저장된 명칭이 매개변수와 같은 명칭에 저장된 값을 불러오는 메소드
getParameterNames()	client가 보내준 값을 저장한 명칭을 불러오는 메소드
getParameterValues(String)	client가 보내준 값이 여러 개일 경우 그 값을 배열로 불러오는 메소드
getParameterMap()	clien가 보내준 값 전체를 Map방식으로 불러오는 메소드

Servlet

HttpServletRequest(interface)

메소드	설명
setAttribute(String,Object)	request 객체에 전달하고 싶은 값을 String 이름으로 Object객체로 저장하는 메소드
getAttribute(String)	매개변수와 동일한 객체 속성 값을 불러오는 메소드
removeAttribute(String)	request 객체에 저장되어 매개변수와 동일한 속성값을 삭제하는 메소드
setCharacterEncoding(String)	전송 받은 request 객체의 값들의 CharacterSet을 설정해주는 메소드
getRequestDispatcher(String)	컨테이너내에서 request,response 객체를 전송하여 처리할 컴포넌트(jsp파일 등)를 불러오는 메소드로 forward() 메소드와 같이 사용

Servlet

HttpServletResponse(interface)

- 요청에 대한 처리결과를 작성하기 위해 사용하는 객체
- 인터페이스 구현은 컨테이너가 설정하며, 메소드만 사용
- javax.servlet.ServletResponse를 상속

메소드	설명
setContentType(String)	응답으로 작성하는 페이지의 MIME type을 정하는 메소드
setCharacterEncoding(String)	응답하는 데이터의 CharacterSet을 지정해주는 메소드
getWriter()	문자를 페이지에 전송하기 위한 Stream을 가져오는 메소드
getOutputStream()	byte 단위로 페이지에 전송을 위한 Stream을 가져오는 메소드
sendRedirect(String)	client가 매개변수의 페이지를 다시 서버에 요청하게 하는 메소드