Servlet LifeCycle

# 수업차시	1

01. Servlet

01-01. Servlet 개요

01-01-01. Servlet이란

- Server + Applet의 합성어로, JAVA 언어를 이용하여 사용자의 요청을 받아 처리하고 처리 결과를 다시 사용자에게 전송하는 역할의 Class 파일을 말한다.
  - Applet = Application + let (작은) = 작은 애플리케이션
- 웹에서 동적인 페이지를 java로 구현한 서버측 프로그램으로 보면 된다.
- 관련 패키지와 클래스는 tomcat에서 제공하는 API문서에서 확인 가능하다.

  https://tomcat.apache.org/tomcat-8.0-doc/servletapi/

01-01-02. Servlet 역사

- Java 언어의 창시자인 제임스 고슬링(James Gosling)은 1995년 자바를 발표하며 자바로 구현할 수 있는 서버 프로그래밍 기술에 대해서도 염두에 두고 있었지만, 해당 개념이 실제 구현이 가능한 정도로 제품화 되지 않은 상태였다.
- 당시 Sun사에서는 이를 서버로 구현할 수 있는 제품이 없어 잠시 미룰 수 밖에 없었다. 즉, Java EE Platform이 제품화 되어 있지 않은 시기의 이야기인 것이다.
- 얼마 뒤, .Java 팀의 일원이었던 파바니 디완지(Pavni Diwanji)는 자바 서버 기술에 대한 필요성을 느껴 Servlet 개념을 고안하였고, 이 개념을 토대로 프로젝트를 진행하여 Servlet 구현 및 제품화에 성공하였다.
- 그리고 이 기술은 1997년 6월에 Servlet 1.0을 공식 발표하면서 Java EE의 제품화에 포함되었다.

01-01-03. Servlet 버전 변천사

서블릿 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. 서블릿 보안 강화. 비동기 서블릿 지원
Servlet 3.1	2013.05	Java EE 7	Non-blocking I/O 지원. HTTP1.1 지원

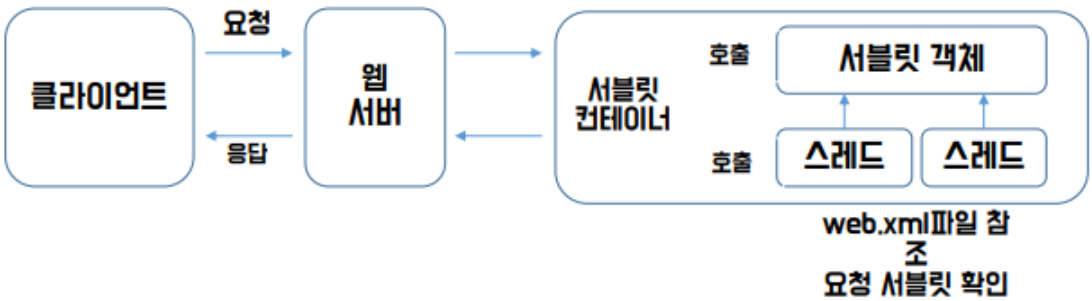
01-01-04. Servlet 설계 규약

1. 모든 Servlet은 javax.servlet.Servlet interface를 상속 받아 구현한다.
2. Servlet 구현 시 Servlet interface와 ServletConfig interface를 javax.servlet.GenericServlet에 구현한다.
3. HTTP 프로토콜을 사용하는 Servlet은 httpServlet Class를 상속 받는다.

   (javax.servlet.http.HttpServlet Class는 javax.servlet.GenericServlet를 상속받은 Class이다.)
4. Servlet의 Exception을 처리하기 위해서는 javax.servlet.ServletException을 상속 받아야 한다.

01-02. Servlet 동작 구조

01-02-01. Servlet의 동작 구조



01-02-02. Servlet Container란

- 웹 서버 또는 응용 프로그램 서버의 일부로, 웹 서버에서 온 요청을 받아 Servlet class를 관리하는 역할(= 생명 주기 관리)을 한다.
- Servlet에 대한 Containe 설정은 Deployment Descriptor(web.xml) 파일을 이용한다.

▼ 추가 설명

1. 쉽게 말해 Servlet을 포함하고 있는 Container가 Servlet Container 이다.
2. Tomcat의 핵심이 바로 이 Servlet Container이다.
3. 작성한 Servlet class를 보관하고, 특정 요청에 따라 Servlet을 결정해서 객체를 생성하거나 이미 생성된 경우 service() method를 호출하여 Servlet이라는 작은 프로그램을 실행한다.
4. Tomcat을 종료하면 Servlet이 전부 소멸하기 때문에 자동으로 destroy() method를 호출한다.

01-03. Deployment Descriptor(DD)

01-03-01. 배포서술자

- Application에 대한 전체 설정 정보를 가지고 있는 파일로, xml 형식 파일이며 요소(= 태그)로 이루어져 있다.
- 파일 내 설정 정보를 가지고 웹 컨테이너가 Servlet을 구동한다.
  - 설정 정보
    - Servlet 정의 및 Servlet 초기화 파라미터
    - Session 설정 파라미터
    - Servlet-jsp mapping 및 MIME type mapping
    - 보안 설정
    - Welcome file list 설정
    - Error page list, resources, 환경변수
- 위치: webapp > WEB-INF > web.xml

01-03-02. DD 파일 세부 내용

- <web-app> : 루트 속성, 문법 식별자 및 버전 정보를 속성 값으로 설정
- <context-param> : 웹 어플리케이션에서 공유하기 위한 파라미터 설정
- <mime-mapping> : 특정 파일 다운로드 시 파일이 깨지는 현상 방지

- <servlet>~<servlet-class> : 컨테이너에 Servlet 설정
- <servlet-mapping> : Servlet mapping
- <welcome-file-liest> : 시작 페이지 설정
- <filter> : 필터 정보 등록
- <error-page> : 에러 발생 시 안내 페이지 설정
- <session-coinfig> : session 기간 설정
- <listener> : 이벤트 처리 설정 (6가지)

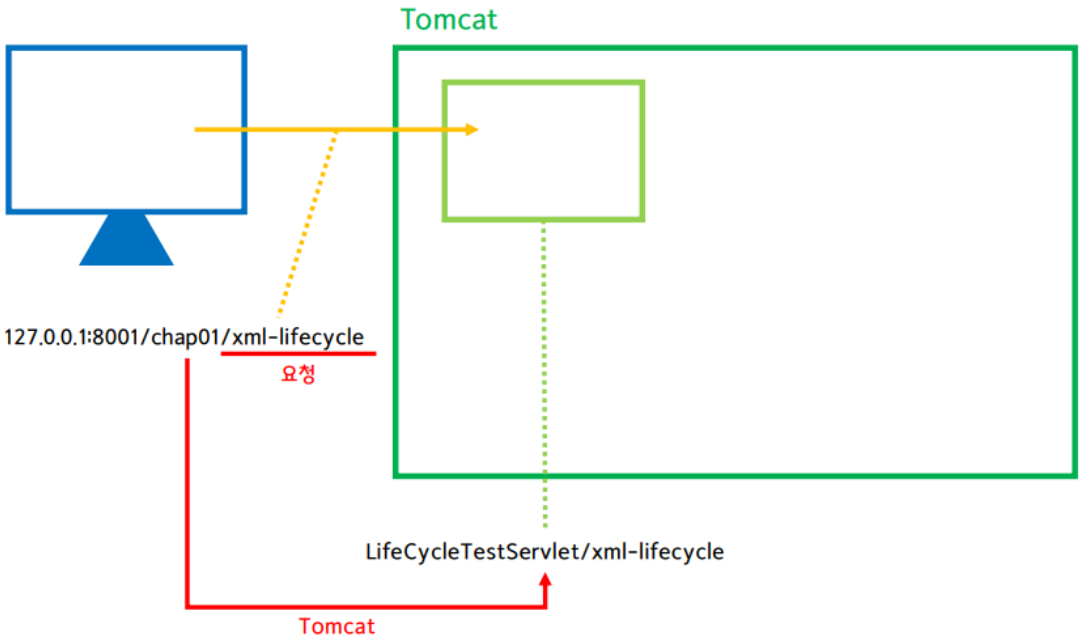
01-04. Servlet mapping

01-04-01. Servlet mapping 방법

- client가 servlet에 접근할 때 원본 클래스명이 아닌 다른 명칭으로 접근할 수 있다.
- 별칭 사용 설정 방법은 web.xml을 이용한 적용과 @annontation으로 두 가지 방법이 있다.

01-04-02. Servlet mapping 구조

- 클라이언트가 서블릿의 url로 요청을 보내면, 톰캣이 이를 해석해 매핑된 서블릿으로 연결하는 구조이다.



- mapping 경로 잘못되면 tomcat 실행 자체가 되지 않는다.

- 정의되지 않은 경로이거나 중복된 경로를 사용하면 톰캣 서버 실행 오류가 발생한다.

01-04-02. web.xml 이용

- xml 파일로 매핑 정보를 관리하면 한눈에 확인하고 관리할 수 있다는 장점이 있지만, 새로운 서블릿을 추가할 때마다 파일을 이동해 매핑하고 일일이 확인해야 하는 단점이 있다.
- <servlet>과 <servlet-mapping>은 묶어서 함께 설정해야 한다.
- 사용법

```
<servlet>
  <servlet-name>mapping명칭</servlet-name>
  <servlet-class>실제 클래스명</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>mapping명칭</servlet-name>
  <url-pattern>사용자 접근명칭</url-pattern>
</servlet-mapping>
```

- 사용 예시

```
<servlet>
    <servlet-name>xmlmapping</servlet-name>
    <servlet-class>com.section01.xml.LifeCycleTestServlet</servlet-class>
    <load-on-startup>100</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>xmlmapping</servlet-name>
    <url-pattern>/xml-lifecycle</url-pattern>
</servlet-mapping>
```

  - <load-on-startup> 속성을 설정하고 숫자로 우선순위를 주면, 낮은 숫자를 우선하여 서버가 구동될 때 서블릿을 생성한다.

01-04-03. @annotation 이용

- 어노테이션(Annotation)이란
  - 사전적 의미로는 '주석'을 뜻한다.
  - 실제로 주석처럼 쓰이며 특별한 의미, 기능을 수행하도록 하는 기술이다.
  - 프로그램에 추가 정보를 제공해주는 메타데이터로 볼 수 있다.

    (meta data : 데이터를 위한 데이터)
- 서블릿을 새로 개발할 때 클래스 바로 윗쪽에 어노테이션만 추가하면 되므로 간편하다는 장점이 있다.

- 하지만 별도의 문서 등으로 관리하지 않으면 시스템의 모든 서블릿 매핑 정보를 한눈에 볼 수 없다는 단점이 있다.
- 사용법

```
@WebServlet(“/mapping명칭”)
public class Servlet명 extends HttpServlet {
  (Servlet code)
}
```

- 사용 예시

```
@WebServlet(value = "/annotation-lifecycle", loadOnStartup = 1)
public class LifeCycleTestServlet extends HttpServlet {
  (Servlet code)
}
```

01-05. Servlet 초기값 설정

01-05-01. ServletConfig

- web.xml의 정보를 가져와 저장한 Interface 객체로, getServlet method로 정보가 저장된 객체를 호출할 수 있다.
- getInitParam(“*저장명*”)으로 초기화된 값을 가져올 수 있다.
- 지정된 servlet에서만 활용 가능한 초기값이며, 동적 수정이 불가한 상수값으로 보면 된다.
- servlet이 초기화된 이후에 활용 가능하다.

```
<web-app>
  <servlet>
    <servlet-name>mappin명</servlet-name>
    <servlet-class>설정 Class명</servlet-class>
    <init-param>
      <param-name>저장명</param-name>
      <param-value>저장값</param-value>
    </init-param>
  </servlet>
</web-app>
```

01-05-02. ServletContext

- ServletConfig의 초기값은 지정된 Servlet에서만 사용이 가능하나, ServletContext는 모든 Application 이 공용으로 사용하는 초기값을 설
  정한다.

- 값은 getContext().getInitParam("*저장명*")으로 호출한다.
- <servlet> 태그 내부가 아닌 <web-app> 내부에 설정한다.
- getContext 앞에는 getConfig() / this가 생략되어 있다.

```
<web-app>
  <context-param>
    <param-name>저장명</param-name>
    <param-value>설정값</param-value>
  </context-param>
</web-app>
```

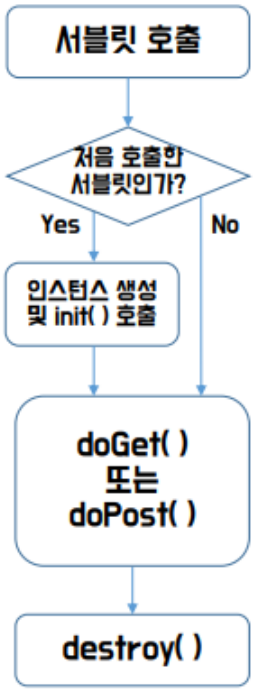
01-05-03. Context Path

- Application에 접근하는 경로로, 컨테이너에서 Application을 구분하는 root경로라고 볼 수 있다.
- 프로젝트의 별칭은 Tomcat 서버 설정의 server.xml 파일 내 <Context> 설정을 따른다.
- http://localhost:[*PORT번호*]/[*프로젝트 별칭*]/[*Servlet 명*]

  예시: http://localhost:8800/first/test1.do

01-06. Servlet의 LifeCycle과 구동

01-06-01. Servlet LifeCycle



1. 첫 요청이면, 객체를 생성하며 init( ) 메소드를 호출한다.
    a. init( ) : 서블릿 컨테이너에 의해 호출되며 최초 요청 시에만 실행하고 두 번째 요청부터는 호출하지 않는 메소드이다.
    b. load-on-startup 속성값을 설정할 경우 서버가 구동할 때 객체를 생성하며 init( ) 메소드를 호출한다.
    c. 서블릿도 클래스이므로 객체를 생성해 작동한다. 따라서 서버 구동 시 서블릿을 생성하면 톰캣 최초 수행 시에는 시간이 좀 걸리지만, 클라이언트 요청에는 더 빠르게 응답할 수 있다.
2. 이후 작업이 실행 될 때마다 service() 메소드가 요청한 HTTP Type에 따라 doGet(), doPost() method를 호출한다.
    a. service() : 서블릿 컨테이너에 의해 호출되며 최초 요청 시에는 init() 메소드 동작 이후에 동작하고, 두 번째 요청부터는 init() 메소드 호출 없이 바로 동작한다.
3. 최종적으로 Servlet이 서비스 되지 않을 때 destroy() 메소드가 호출된다.
    a. destroy() : 컨테이너, 즉 서버가 종료될 때 또는 서블릿의 내용이 변경되어 재컴파일 될 때 호출되는 메소드로 주로 자원을 반납하는 용도로 사용한다.

- Servlet의 주요 메소드는 Servlet의 Life cycle과 유사하다.

01-06-02. Servlet 구동