* Servlet이란??

- 웹 서비스를위한 자바 클래스

  (자바를 이용하여 웹을 만들기 위해 필요한 기술)

- 웹 프로그래밍에서 클라이언트의 요청을 처리하고

  그 결과를 다시 클라이언트에게 전송(응답)하는

  Servlet클래스의 구현 규칙을 지킨 자바 프로그래밍 기술

  (ex. 사용자가 로그인을 하려고 할 때 아이디와 비밀번호를 입력하고 로그인 버튼을 누르면

  서버는 아이디와 비밀번호를 확인하고 다음 페이지를 띄워주는 역할 수행)

즉, Servlet은 자바 어플리케이션 코딩을 하듯 웹 브라우저용 출력 화면을 만드는 방법

* 서블릿 특징

- 클라이언트의 요청에 대해 동적으로 작동하는 웹 애플리케이션 컴포넌트.
- HTML을 사용하여 요청에 응답
- java thread를 이용하여 동작
- mvc패턴에서 controller로 이용
- http프로토콜 서비스를 지원하는javax.servlet.http.HttpServlet 클래스를 상속 받음

- servlet에 작성한 html 코드 변경 시재컴파일 해야하는 단점이 있음

* 서블릿 상속 관계

- 서블릿 클래스는 반드시 javax.servlet.http.HttpServlet 클래스를 부모 클래스로 상속 받아야 함

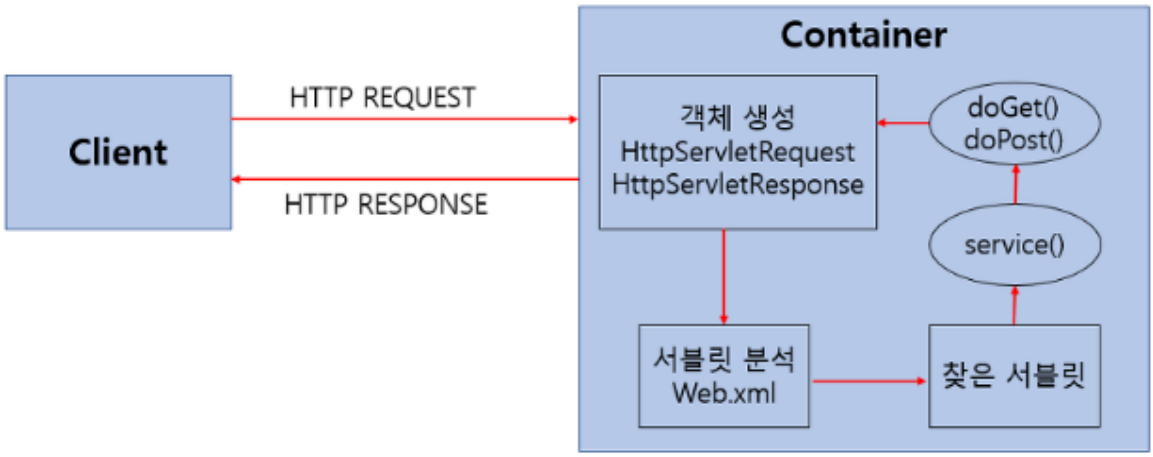
- 서블릿 상속 관계도

javax.servlet.Servlet 인터페이스

ㄴ javax.servlet.GenericServlet 추상클래스

ㄴ javax.servlet.http.HttpServlet 추상클래스

*** Servlet 동작 방식**



1. 사용자(클라이언트)가 URL(Uniform Resource Locator)을

   클릭하면 HTTP Request를 Servlet Container로 전송

2. Http Request를 전송 받은 Servlet Container는

   HttpServletRequest, HttpServletResponse 두 객체 생성

3. DD (배포서술자, Deployment Descriptor) = web.xml은

   사용자가 요청한 URL을 분석하여 어느 서블릿에 요청할 것인지 찾음

4. 해당 서블릿에서 init() 메소드를 먼저 호출한 후 service() 메소드를 호출하여

클라이언트로부터 전송 받은 방식인 GET, POST 여부에 따라 해당 메소드(doXXX()) 를 호출함.

5. doGet() / doPost() 메소드는 동적 페이지를 생성 후 HttpServletResponse객체에 응답을 보냄

6. 응답이 끝나면 destroy()메소드를 호출하여 HttpServletRequest, HttpServletResponse 객체 소멸

* Get과 Post방식의 비교/차이

- 클라이언트가 서버로 요청을 보내는 방법

1. GET방식 : (데이터를) 가져오다, 얻어오다

- URL에 변수(데이터)를 포함시켜 요청

  보안 유지를 안하기 때문에 로그인 같은 경우는 get방식으로 하면 부적합

- 데이터를 HTTP Header에 포함하여 전송

  GET방식에서 바디는 보통 빈 상태로 전송 되며

  헤더의 내용 중 Body의 데이터를 설명하는 Content-type헤더필드도 들어가지 않음

- 전송하는 길이 제한(보내는 길이가 너무 길면 초과데이터는 절단됨)

- 캐싱 가능 (ex. 즐겨찾기, 북마크)

  (한번 접근 후, 또 요청할 시 빠르게 접근하기 위해 데이터를 저장시켜 놓는 것)

2. POST방식 : (데이터) 부치다

- 데이터를 서버로 제출하여 추가 또는 수정하기 위해 데이터를 전송하는 방식

- URL에 변수(데이터)를 노출하지 않고 요청 데이터를 HTTP Body에 포함하여 전송

- 헤더필드 중 Body의 데이터를 설명하는 Content-Type이라는 헤더필드가 들어가고
  어떤 데이터 타입인지 명시해주어야 함

- 전송하는 길이 제한이 없음.
  Body에 데이터가 들어가기 때문에 길이에 제한이 없지만
  최대 요청을 받는 시간(Time Out)이 존재해서
  페이지 요청, 기다리는 시간 존재

- 캐싱할 수 없음.
  URL에 데이터가 노출 되지 않으므로 즐겨찾기나 캐싱 불가능
  하지만 쿼리스트링(문자열)데이터, 라디오 버튼, 텍스트 박스와 같은
  객체들의 값도 전송 가능

* Servlet Container

- 서블릿을 관리해주는 컨테이너

(서블릿 : 어떤 역할을 수행하는 정의서 / 서블릿 컨테이너 : 정의서를 보고 수행)

- 클라이언트의 요청을 받아주고 응답할 수 있게 웹서버와 소켓을 만들어 통신 (ex. 톰캣)

* Servlet Container 역할

1. 웹서버와의 통신 지원

서블릿과 웹 서버가 손 쉽게 통신할 수 있게 함

일반적으로 소켓을 만들고 listen, accept 등을 해야하지만

서블릿 컨테이너는 이러한 기능을 API로 제공하여 복잡한 과정 생략하게 함

2. 서블릿 생명주기(Life Cycle) 관리

서블릿 클래스를 로딩하여 인스턴스화 하고, 초기화 메소드를 호출하고,

요청이 들어오면 적절한 서블릿 메소드 호출

서블릿이 생명을 다 한 순간에는 적절하게 가비지 컬렉션을 진행하여 편의 제공

3. 멀티쓰레드 지원 및 관리

서블릿 컨테이너는 요청이 올 때마다 새로운 자바 쓰레드 생성하는데

http서비스 메소드를 실행하고 나면 쓰레드는 자동으로 죽음

(원래는 쓰레드를 관리해야하지만

서버가 다중 쓰레드를 생성/운영 해주어서

쓰레드의 안정성에 관해서는 걱정할 필요 없음)

4. 선언적인 보안 관리

서블릿 컨테이너를 사용하면 개발자는

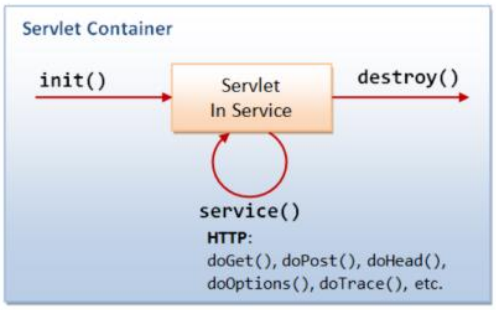
보안에 관련된 내용을 서블릿 또는 자바 클래스에 구현하지 않아도 됨

일반적으로 보안관리는 xml배포 서술자(DD (web.xmk)에다가 기록하므로

보안에 대해 수정할 일이 생겨도

자바소스코드를 수정하여 다시 컴파일 하지 않아도 보안관리 가능

* Servlet 생명주기



1. 클라이언트의 요청이 들어오면 컨테이너는

- 해당 서블릿이 메모리에 있는지 확인하고,

없는 경우 init()메소드를 호출하여 적재함.

- init()메소드는 처음 한번만 실행되기 때문에

서블릿의 쓰레드에서 공통적으로 사용해야하는 것이 있다면 오버라이딩 하여 구현

실행 중 서블릿이 변경될 경우, 기존 서블릿을 파괴하고

init()메소드를 통해 새로운 내용을 다시 메모리에 적재

2. init()메소드가 호출된 후 클라이언트의 요청에 따라서

- service()메소드를 통해 요청에 대한 응답이 doGet()나 doPost()로 분기

- 이때 서블릿 컨테이너가 클라이언트의 요청이 오면

가장 먼저 처리하는 과정으로 생성된 HttpServletRequest, HttpServletResponse에 의해

request와 response객체가 제공됨

3. 컨테이너가 서블릿에 종료 요청을 하면 destroy()메소드가 호출되는데 마찬가지로 한 번만 실행되며, 종료 시에 처리해야하는 작업들은 destroy()메소드를 오버라이딩 하여 구현하면 됨

* 메소드 별 오버라이딩

웹 클라이언트의 요청 방식이 GET방식으로 요청 해오면 doGet() 메소드로 응답

웹 클라이언트의 요청 방식이 POST방식으로 요청 해오면 doPost() 메소드로 응답

그러므로 반드시 doGet()메소드와 doPost()메소드는 Overriding을 해주어야 함

첫번째 파라미터는 HttpServletRequest 타입

두번째 파라미터는 HttpServletResponse 타입

* Request, Response

- 클라이언트 플랫폼 정보 및 브라우저 정보 : String request.getHeader("User-Agent")
- Request 관련 쿠키 : Cookies[] cookies = request.getCookies();
- 클라이언트 세션 정보 : HttpSession session = request.getSession();
- HTTP 메소드 : String method = request.getMethod();
- 출력스트림(PrintWriter, ServletOutputStream)을 이용하여 HTML 등을 작성

  클라이언트에게 돌려보낼 setContentType()을 정함
- 그 밖에 헤더정보 설정, 오류를 발생시키거나 쿠키를 추가함