Servlet

서블릿 이해

- 자바 플랫폼에서 컴포넌트 기반의 웹 애플리케이션 개발기술
- **JSP**는 서블릿 기술에 기반함.
- 서블릿의 프리젠테이션 문제를 해결하기 위해 **JSP**가 등장
- **JSP** 모델**2** 가 주목받으며 다시 서블릿에 대한 중요성 부각

서블릿 변천

① 서블릿 문제점 대두

프로그램에서 HTML 핸들링
컨텐츠와 비즈니스 로직이 분리되지 않음
개발과 관리의 어려움

② JSP 등장

HTML에서 프로그램 핸들링이 가능
→ JSP 스크립팅 기술

③ JSP 스크립트 기술의 한계

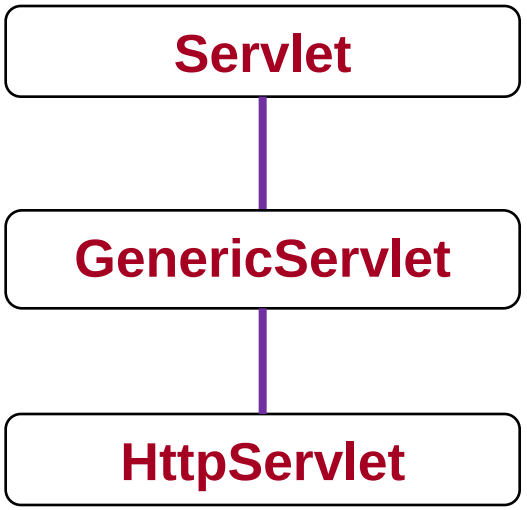
HTML에서 프로그램 핸들링
컨텐츠와 비즈니스 로직이 분리되지 않음
컨텐츠 관리는 쉬워졌지만 프로그램 관리는 이전보다 더 복잡해짐

④ MVC 패턴 주목받기 시작

애플리케이션 구성 요소 단위로 역할 분담
모델 : 자바 클래스(DAO, VO)
뷰 : JSP(JSTL, EL)
컨트롤러 : 서블릿

서블릿 API

- 웹에서 동작하는 서블릿 클래스가 되기 위해서는 다음 3가지 클래스 중 하나를 상속받아서 작성



인터페이스

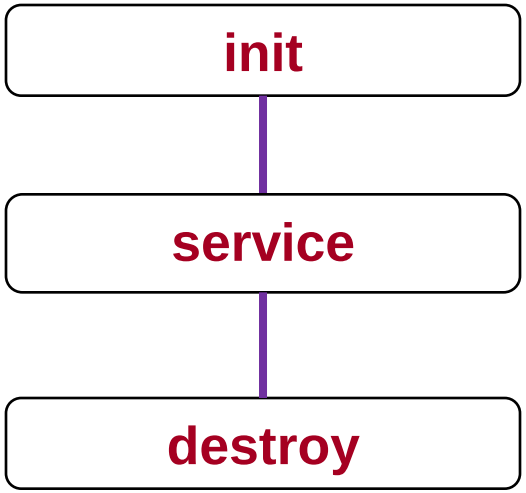
추상클래스

추상클래스

대부분의 사용자 정의 서블릿 클래스는 **Httpservlet** 클래스를 상속받아 구현

서블릿 생명주기

- 서블릿 클래스는 기본적인 메소드 호출 순서(LifeCycle)가 존재
  - init : 최초 한번만 실행
  - service : 사용자 요청에 대한 실제 응답 처리 진행
  - destroy : 메모리에서 해제될 경우 호출(예>서블릿 내용이 변경, 서버 재구동..)
  - init -> service -> destroy



web.xml에 서블릿 등록

- 서블릿은 보안이 적용되어 있는 WEB-INF/classes 폴더에 있으므로 직접 접근이 불가능(웹 URL에서 직접 호출 불가능)
- 서블릿을 web.xml에 등록하여 서버에서 특정한 경로를 맵핑 하여 호출

```
<servlet>
   <servlet-name>hello</servlet-name>
   <servlet-class>kr.co.bit.servlet.HelloServlet</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>hello</servlet-name>
   <url-pattern>/hi.bit</url-pattern>
</servlet-mapping>
```

서블릿 생명주기

- 만약, 사용자 정의 서블릿 클래스가 HttpServlet를 상속받아서 구현 했다면
  - 사용자가 service 메소드를 오버라이딩(재정의) 하지 않았다면