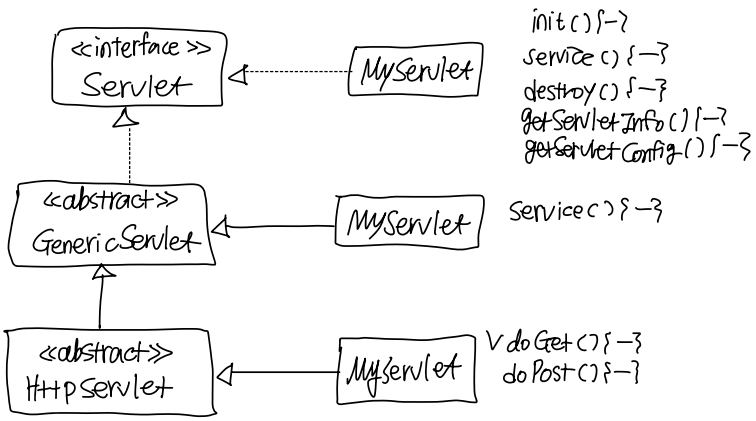
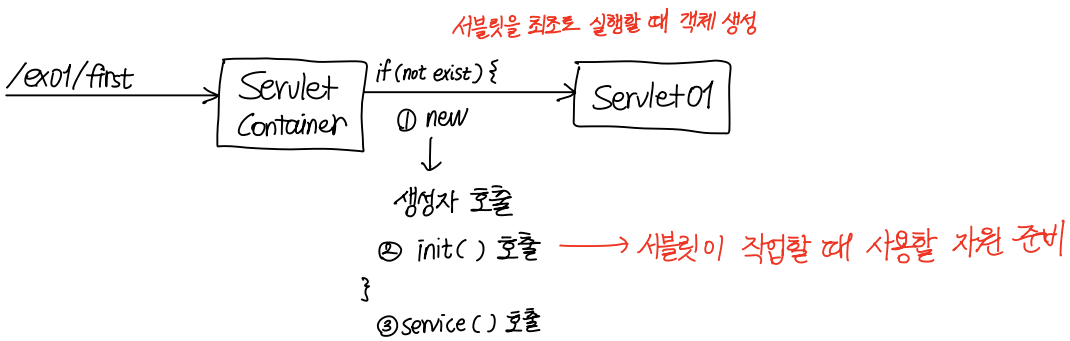
* Servlet 만들기



* Servlet 객체 생성 과정



* Servlet 배치 방법

① 애노테이션

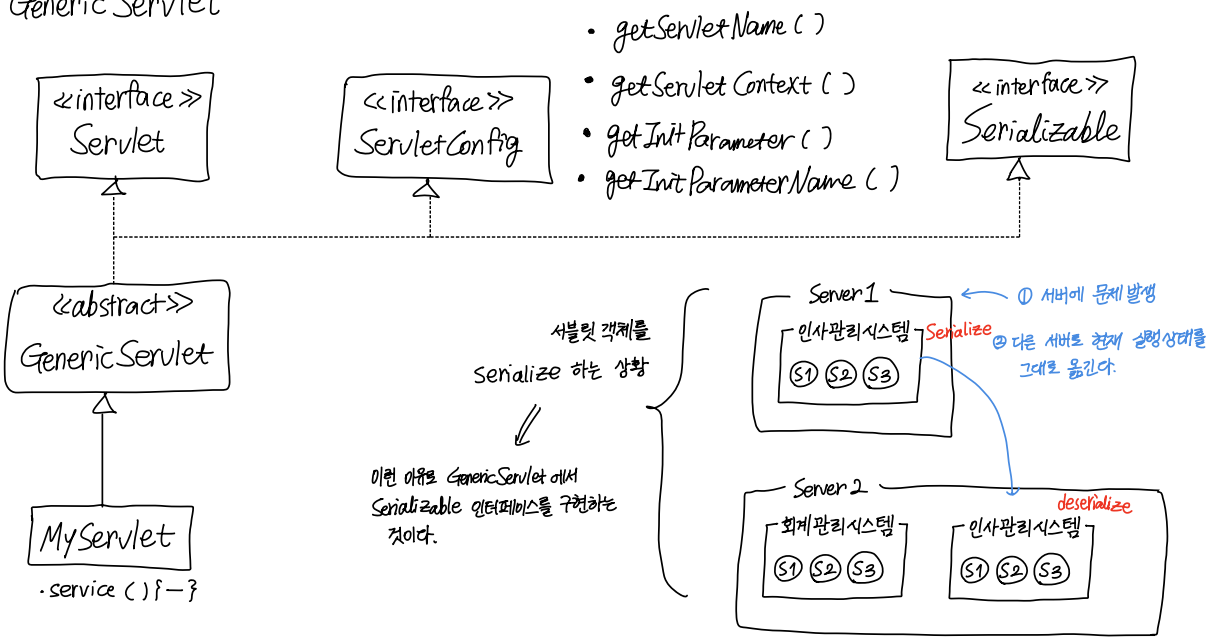
@WebServlet(value={"path", "path", "path"})
class Servlet01 — {—}

② xml

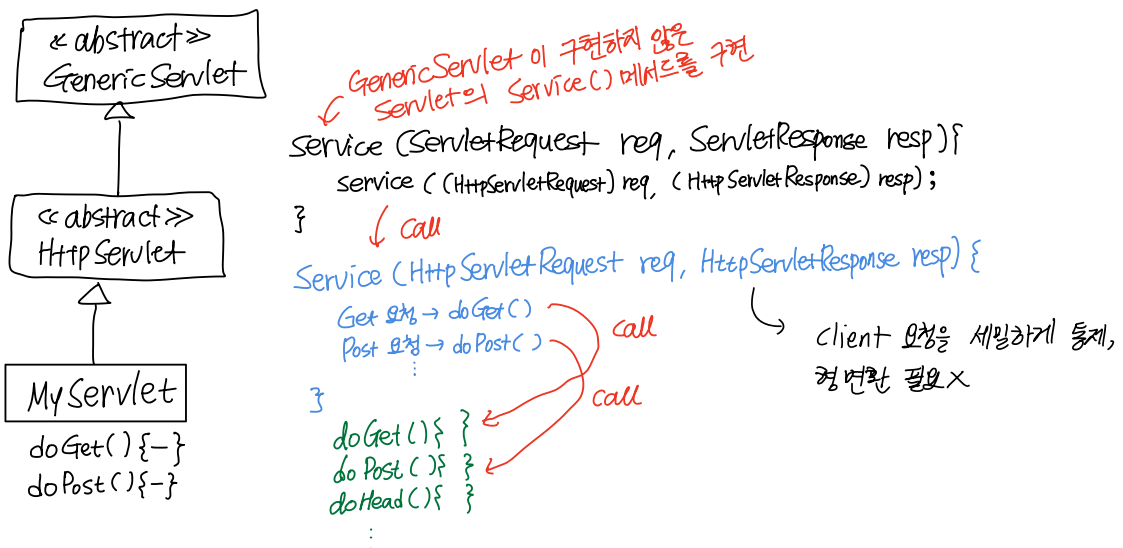
/WEB-INF/web.xml

```
:
<servlet>
    <servlet-name>서블릿명</servlet-name>
    <servlet-class>서블릿 클래스의 FQName</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>        </servlet-name>
        <url-pattern>/hello</url-pattern>
```

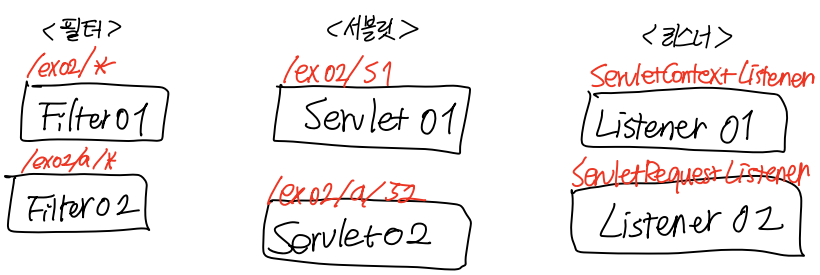
* Generic Servlet



* HttpServlet

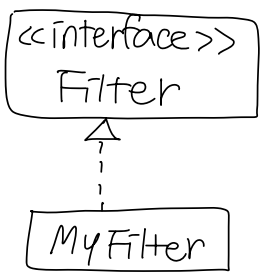


* 주요 웹 컴포넌트

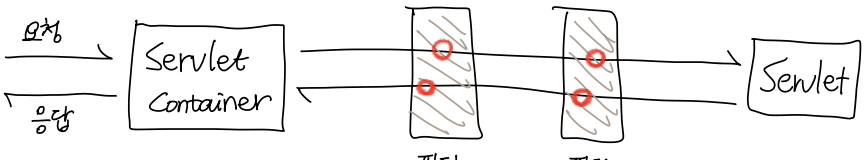


* Filter 만들기

Web app이 시작될 때 객체 생성됨



- init() 필터 객체 생성 후 ← 웹 애플리케이션 실행
- doFilter() 서블릿 요청에 필터가 적용될 때
- destroy() 웹 애플리케이션 종료



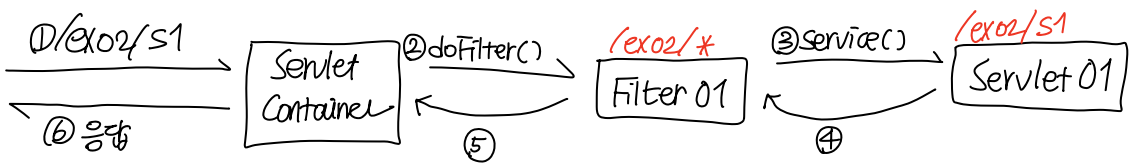
서블릿 실행 전후에 기능을 추가할 때 사용

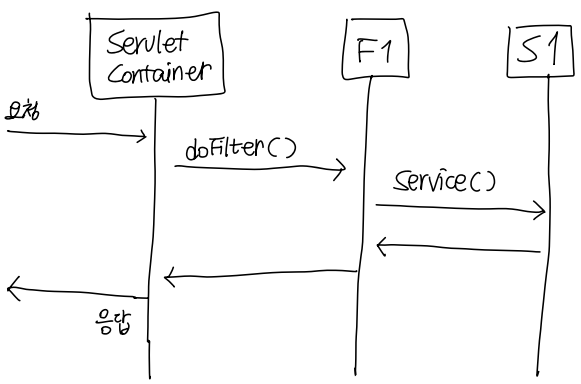
{ 암호해제/암호화
압축해제/압축
디코딩/인코딩
로깅
⋮ }

GoF의 Chain of Responsibility

→ 기존 코드를 손대지 않고 필요할 때 기능을 추가·제거할 수 있는 설계기법

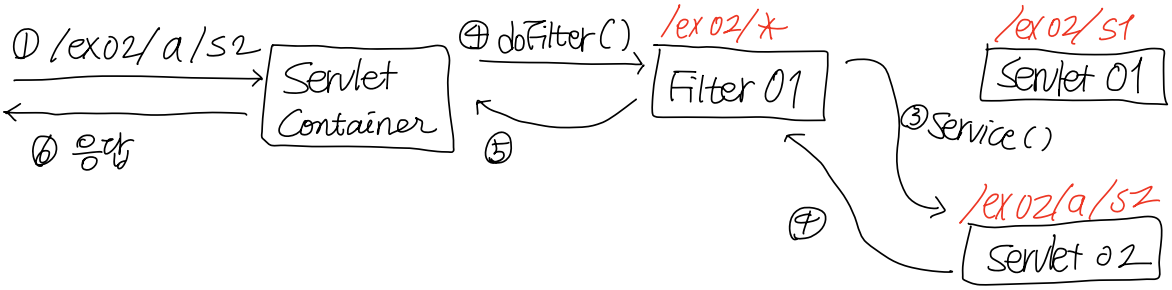
* 필터 실행

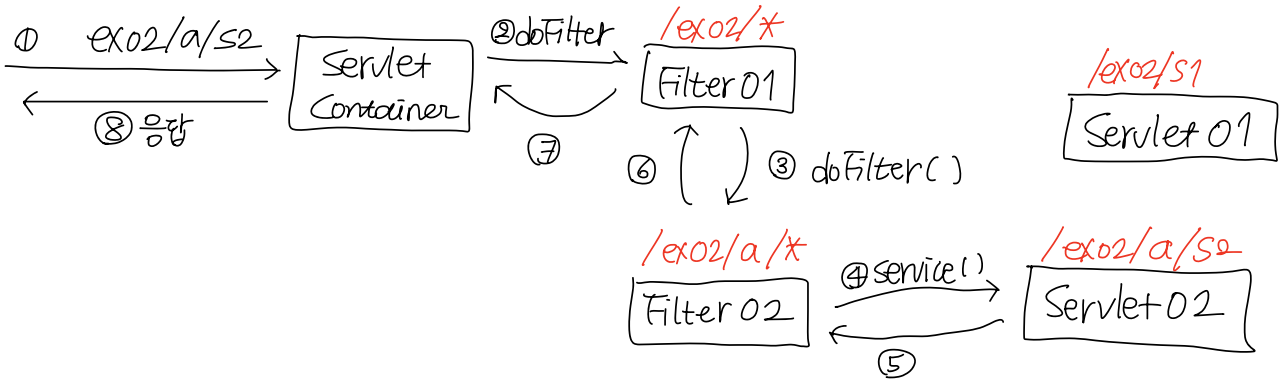


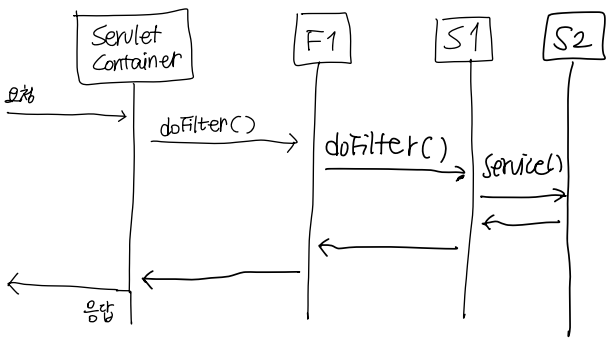


UML의 Interaction Diag 중에서 "Sequence Diag."

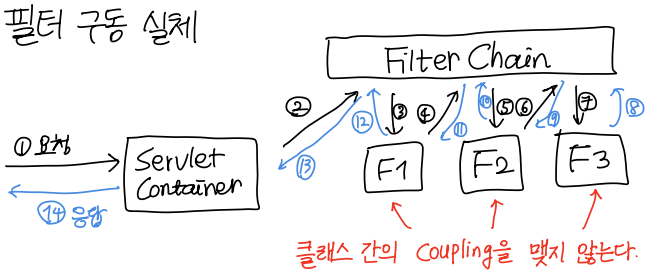
시간의 흐름에 따라 실행 흐름을 도식화







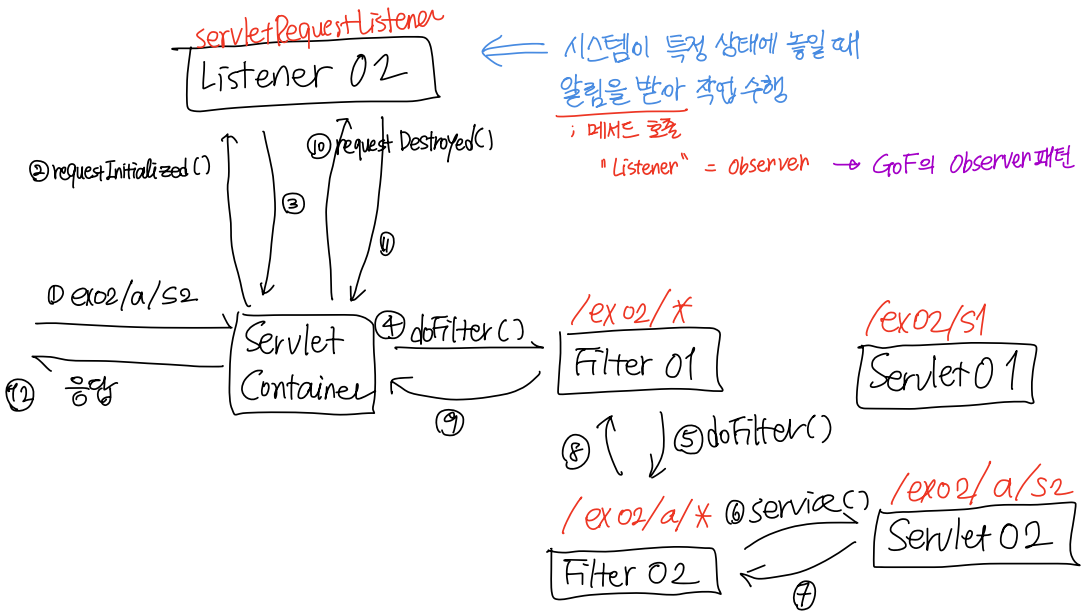
* 필터 구동 실체



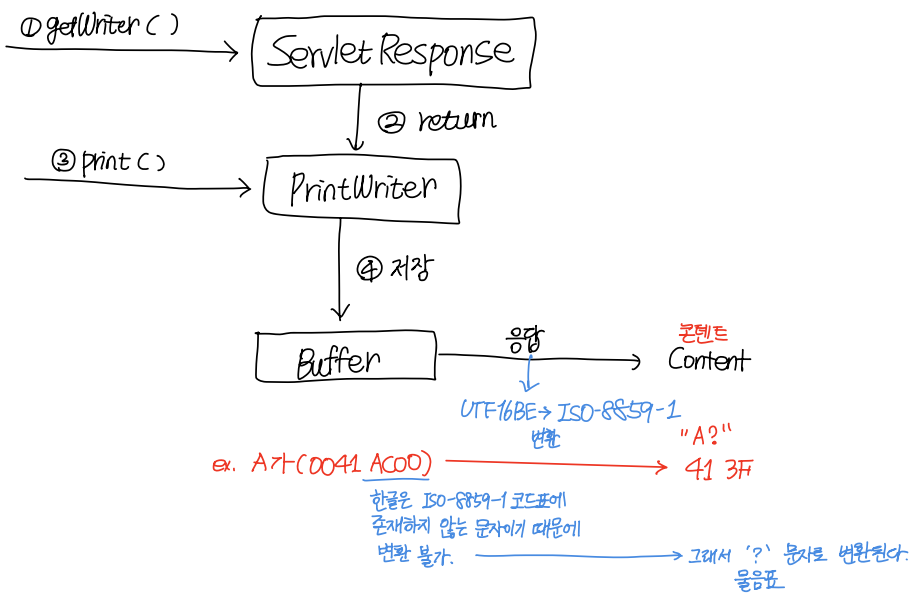
"@WebFilter" 애노테이션 사용

✳ 리스너 실행 : ServletRequestListener

↳ 어떤 특정 상황에 놓일 때 자동으로 실행되는 객체! Observer Pattern
(기존 코드를 손대지 않고 ✌)



✳ 서블릿에서 출력하기



* Binary Data 응답

☆ Binary Data를 보낼땐 get write가 아닌 outputstream 사용!! flush 필수☆