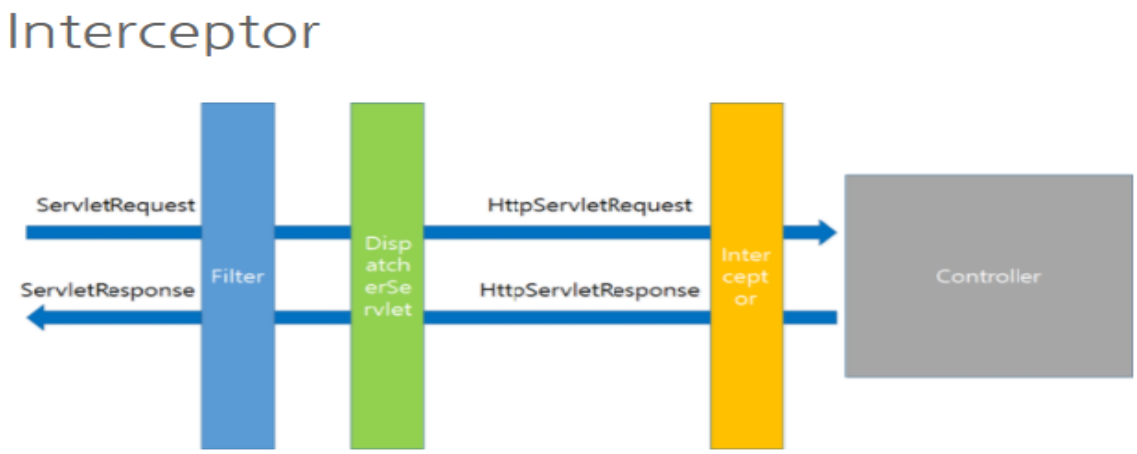
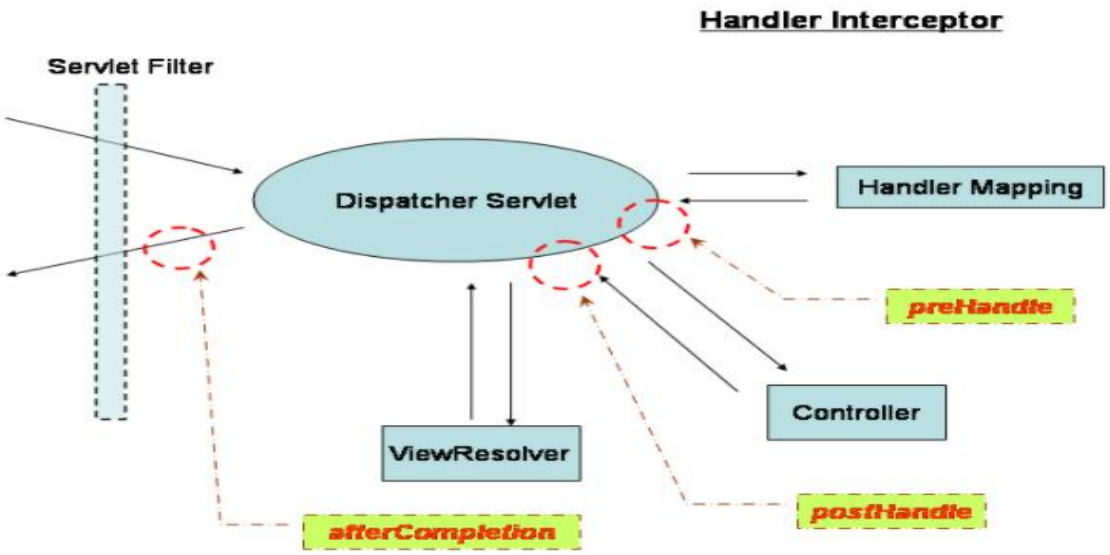
자바 웹프로그래밍을 구현하다보면 공통적인 업무를 추가해야할 것들이 많다. 공통적인 업무에는 로그인처리(세션체크), pc웹과 모바일웹의 분기, 로그 확인, 페이지 인코딩 변환, 권한체크, XSS(Cross site script)방어 등이 있는데 이러한 공통업무에 관련된 코드를 모든 페이지 마다 작성해야한다면 중복된 코드가 많아지게 되고 업무량이 상당히 증가할 것이다. 이러한 공통업무를 프로그램 흐름에서 앞, 중간, 뒤에 추가하여 자동으로 처리할 수 있는 방법이 있는데 서블릿에서 지원하는 서블릿 필터, 스프링 프레임워크를 사용하면 쓸 수 있는 인터셉터, AOP가 있다. 앞서 AOP개념을 정리할 때 언급한 것처럼 개발자는 좀더 핵심로직에 집중하고, 부가로직으로부터 자유로워지게 도와주는 역할을 한다.

Filter, Interceptor, AOP의 실행순서





01) Filter(필터)

말그대로 요청과 응답을 거른뒤 정제하는 역할을 한다.

서블릿 필터는 DispatcherServlet 이전에 실행이 되는데 필터가 동작하도록 지정된 자원의 앞단에서 요청내용을 변경하거나, 여러가지 체크를 수행할 수 있다. 또한 자원의 처리가 끝난 후 응답내용에 대해서도 변경하는 처리를 할 수가 있다.
필터는 web.xml 에 등록하는데 대표적으로 인코딩 변환, 로그인 여부확인, 권한체크, XSS 방어 등의 요청에 대한 처리로 사용된다.

```
<!-- 한글 처리를 위한 인코딩 필터 -->
<filter>
    <filter-name>encoding</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter<
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>encoding</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

등록한 encoding 는 이름은 encoding 이고, 값은 UTF-8 인 파라미터를 정의하고 있다. 필터를 /* 에 맵핑하여 필터가 servlet, jsp 뿐만 아니라 이미지와 같은 모든 자원의 요청에도 호출 된다는 것을 의미한다.

필터의 실행메서드

- init() - 필터 인스턴스 초기화
- doFilter() - 전/후 처리
- destroy() - 필터 인스턴스 종료

---

02) Interceptor(인터셉터)

요청을 가로챈다(작업 전/후)

앞서 말했던 것처럼 필터와 인터셉터는 호출 시점이 다르다. 필터는 스프링과 무관하게 지정된 자원에 대해 동작한다. 스프링은 DistpatcherServlet 으로부터 시작되므로 필터는 스프링 컨텍스트 외부에 존재하게 된다. 하지만 인터셉터는 스프링의 DistpatcherServlet 이 컨트롤러를 호출하기 전, 후로 끼어들기 때문에 스프링 컨텍스트 내부에 존재하게된다. 그리고 스프링 내의 모든 객체(bean) 접근이 가능하다. 인터셉터는 여러 개를 사용할 수 있고 로그인 체크, 권한체크, 프로그램 실행시간 계산작업 로그확인, 업로드 파일처리등에 사용된다.

인터셉터의 실행메서드

- preHandler() - 컨트롤러 메서드가 실행되기 전
- postHanler() - 컨트롤러 메서드 실행직 후 view 페이지 렌더링 되기 전
- afterCompletion() - view 페이지가 렌더링 되고 난 후

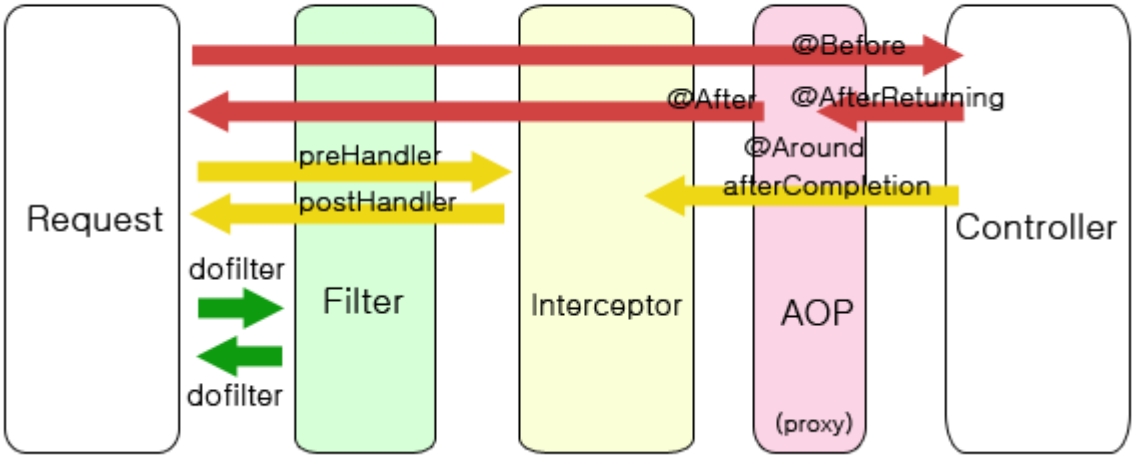
03) AOP

관점 지향 프로그래밍

구분	Filter	Interceptor	AOP
실행 위치	서블릿	서블릿	메서드
실행 순서	1	2	3
설정 위치	web.xml	xml or java	xml or java
실행 메서드	init(), doFilter(), destroy()	preHandler(), postHanler(), afterCompletion()	포인트 컷으로 @After, @Before, @Around 위치를 지정

Interceptor와 filter는 서블릿 단위에서 실행된다. 반면에 AOP는 메소드의 앞에 Proxy패턴을 이용해서 실행된다. 그래서 실행순서에서도 차이가 생긴다. Filter가 가장 겉에 있고, 그 안에 interceptor 그리고 그 안에 aop가 들어있는 구조이다. Request가 filter를 거쳐 interceptor쪽으로 들어가고 aop를 거쳐 다시 나오면서 interceptor와 filter를 거치는 식이다.

실행되는 메소드를 기준으로 설명하면, 서버를 실행시켜 서블릿이 올라오는 동안에 init이 실행되고, 그 후에 dofilter가 실행된다. 그 후 컨트롤에서 들어가기 전에 preHandler가 살행되고, aop가 실행된 후에 컨트롤러에서 나와 postHandler, afterCompletion, dofilter순서대로 진행되고, 서블릿 종료시 destory가 실행 될 것이다.



AOP 의 경우에는 Interceptor 나 Filter 와 달리 메소드 전후의 지점을 자유롭게 설정가능하고, interceptor 와 filter 가 주소로 밖에 걸러낼 대상을 구분 할 수 없는 것에 비해서 AOP 는 주소, 파라미터, 어노테이션등 다양한 방법으로 대상을 지정할 수 있는 장점이 있다.