

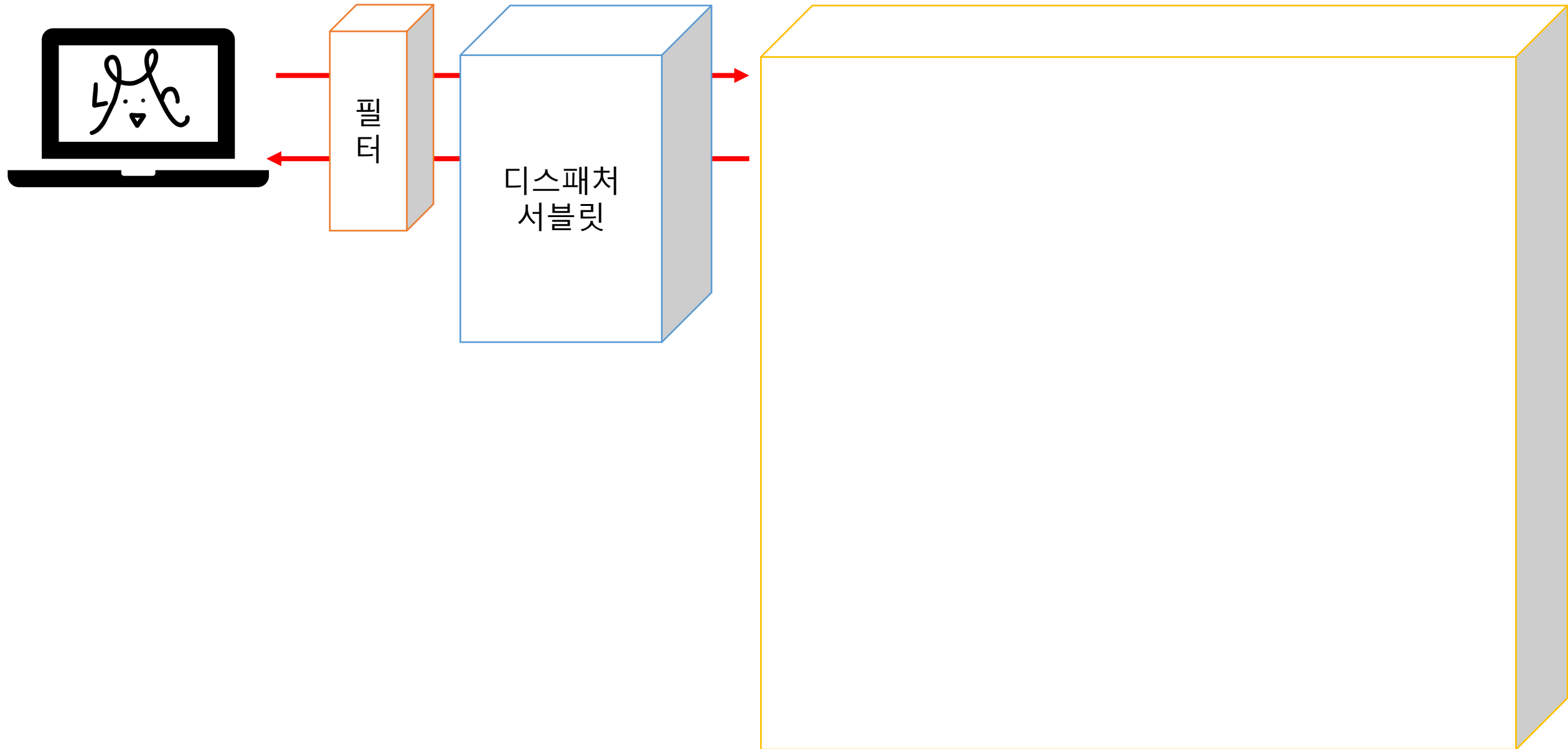
삼성 청년 SW 아카데미

Spring Framework

Filter & Interceptor

- MVC 요청 흐름
- Listener & Filter
- Interceptor

MVC 요청 흐름



Listener & Filter

✓ Listener란?

- 프로그래밍에서 Listener란 특정 이벤트가 발생하기를 기다리다가 실행되는 객체
- 이벤트란 특정한 사건발생.
EX) 버튼클릭, 키보드입력, 컨테이너 빌드완료, 웹어플리케이션시작, HTTP요청수신 등등..
- 이벤트 소스란 이벤트가 발생한 근원지(객체)

✓ Listener 종류

Listener Interface	Description
javax.servlet.AsyncListener	Listener that will be alerted if an asynchronous action started on a ServletRequest to which the listener was attached has completed, timed out, or failed.
javax.servlet.ServletContextListener	Interface for receiving ServletContext lifecycle change notification events.
javax.servlet.ServletContextAttributeListener	Interface for receiving ServletContext attribute changes notification events.
javax.servlet.ServletRequestListener	Interface for receiving notifications about requests entering and exiting a web application's scope.
javax.servlet.ServletRequestAttributeListener	Interface for receiving ServletRequest attribute changes notification events.
javax.servlet.http.HttpSessionListener	Interface for receiving HttpSession lifecycle changes notification events.
javax.servlet.http.HttpSessionBindingListener	When an object is tied to or freed from a session, it is alerted by javax.servlet.http.HttpSessionBindingListener.
javax.servlet.http.HttpSessionAttributeListener	Interface for receiving HttpSession attribute change notification events.
javax.servlet.http.HttpSessionActivationListener	Objects that are connected to a javax.servlet.http.HttpSessionActivationListener

✓ Listener 사용 (Annotation)

```
package com.ssafy.mvc;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;

@WebListener
public class MyListener implements ServletContextListener {

    public void contextDestroyed(ServletContextEvent sce) {
        System.out.println("웹어플리케이션 종료될때 호출1");
    }

    public void contextInitialized(ServletContextEvent sce) {
        System.out.println("웹어플리케이션 시작될때 호출1");
    }

}
```


✓ Listener 사용 (web.xml)

```
package com.ssafy.mvc;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

public class MyListener2 implements ServletContextListener {

    public void contextDestroyed(ServletContextEvent sce) {
        System.out.println("웹어플리케이션 종료될때 호출2");
    }

    public void contextInitialized(ServletContextEvent sce) {
        System.out.println("웹어플리케이션 시작될때 호출2");
        ServletContext context = sce.getServletContext();

        System.out.println("welcome : "+context.getInitParameter("welcome"));
    }
}
```

✓ Listener 사용 (web.xml)

```
<listener>
  <listener-class>com.ssafy.mvc.MyListener2</listener-class>
</listener>
<context-param>
  <param-name>welcome</param-name>
  <param-value>Hello SSAFY</param-value>
</context-param>
```

Console ⌵ Markers Properties Servers Snippets

Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Zulu\zulu-8\bin\javaw.exe

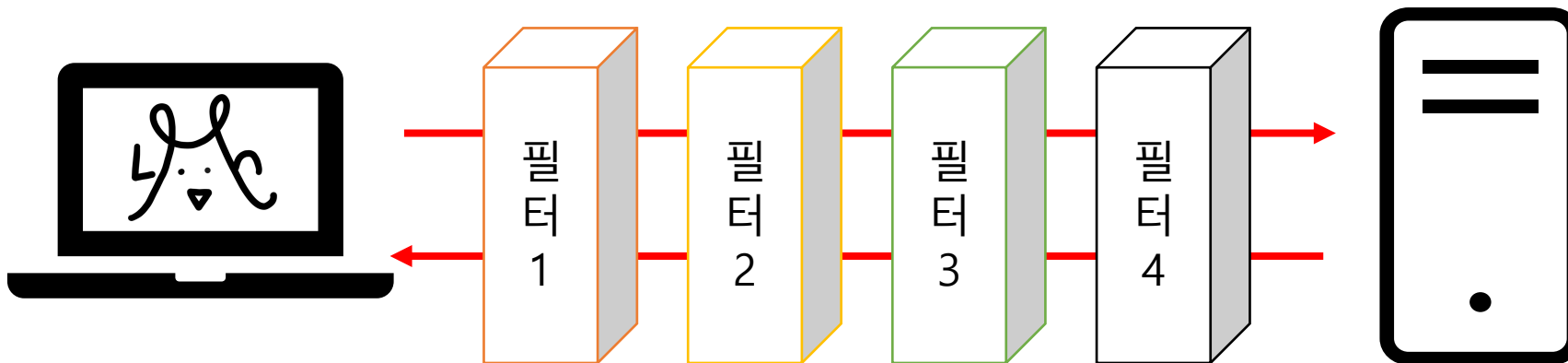
웹어플리케이션 시작될때 호출2

welcome : Hello SSAFY

웹어플리케이션 시작될때 호출1

✓ Filter

- 요청과 응답 데이터를 필터링하여 제어, 변경하는 역할
- 사용자의 요청이 Servlet에 전달되어지기 전에 Filter를 거침
- Servlet으로부터 응답이 사용자에게 전달되어지기 전에 Filter를 거침
- FilterChain을 통해 연쇄적으로 동작 가능



✓ Filter 사용

```
public class MyFilter implements Filter {
    public FilterConfig filterConfig;

    // 필터 초기화
    public void init(FilterConfig fConfig) throws ServletException {
        this.filterConfig = fConfig;
    }

    // 필터 종료
    public void destroy() {
    }

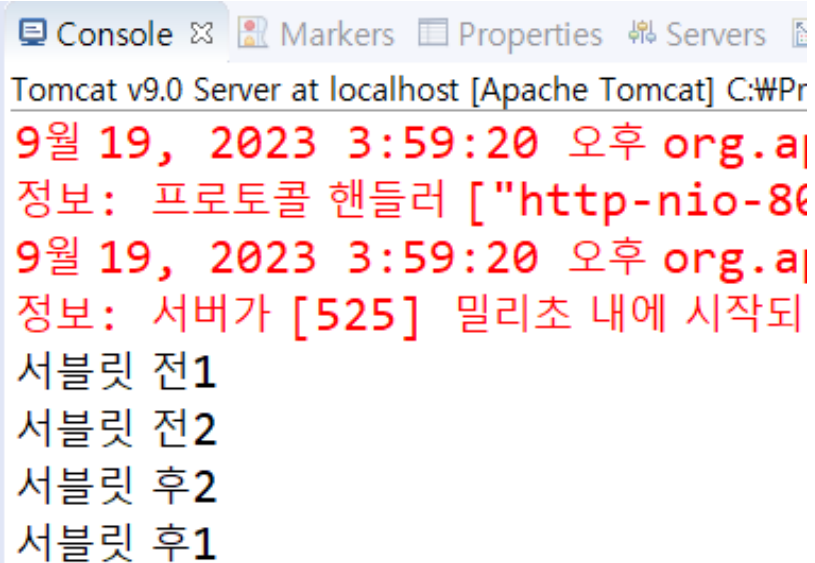
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        // 해당 요청이 서블릿에 들어가기 전 수행하고자 하는 코드
        System.out.println("서블릿 전1");
        String encoding = this.filterConfig.getInitParameter("encoding");
        request.setCharacterEncoding(encoding);
        chain.doFilter(request, response); // 다음 필터로 전달
        // 해당 요청이 서블릿에 의해 처리된 후 수행하고자 하는 코드
        System.out.println("서블릿 후1");
    }
}
```

✓ Filter 사용

```
<filter>
  <filter-name>MyFilter</filter-name>
  <filter-class>com.ssafy.mvc.MyFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>utf-8</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>MyFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

✓ Filter 사용 (순서)

```
<filter-mapping>
  <filter-name>MyFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>MyFilter2</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```



The screenshot shows a web application console with tabs for Console, Markers, Properties, Servers, and a file icon. The console output is from Tomcat v9.0 Server at localhost [Apache Tomcat] C:\WP. It displays two log entries in red text, both timestamped '9월 19, 2023 3:59:20 오후 org.a'. The first entry is '정보: 프로토콜 핸들러 ["http-nio-86' and the second is '정보: 서버가 [525] 밀리초 내에 시작되'. Below these, the execution order of filters is shown: '서블릿 전1', '서블릿 전2', '서블릿 후2', and '서블릿 후1'.

Console ✕ Markers Properties Servers

Tomcat v9.0 Server at localhost [Apache Tomcat] C:\WP

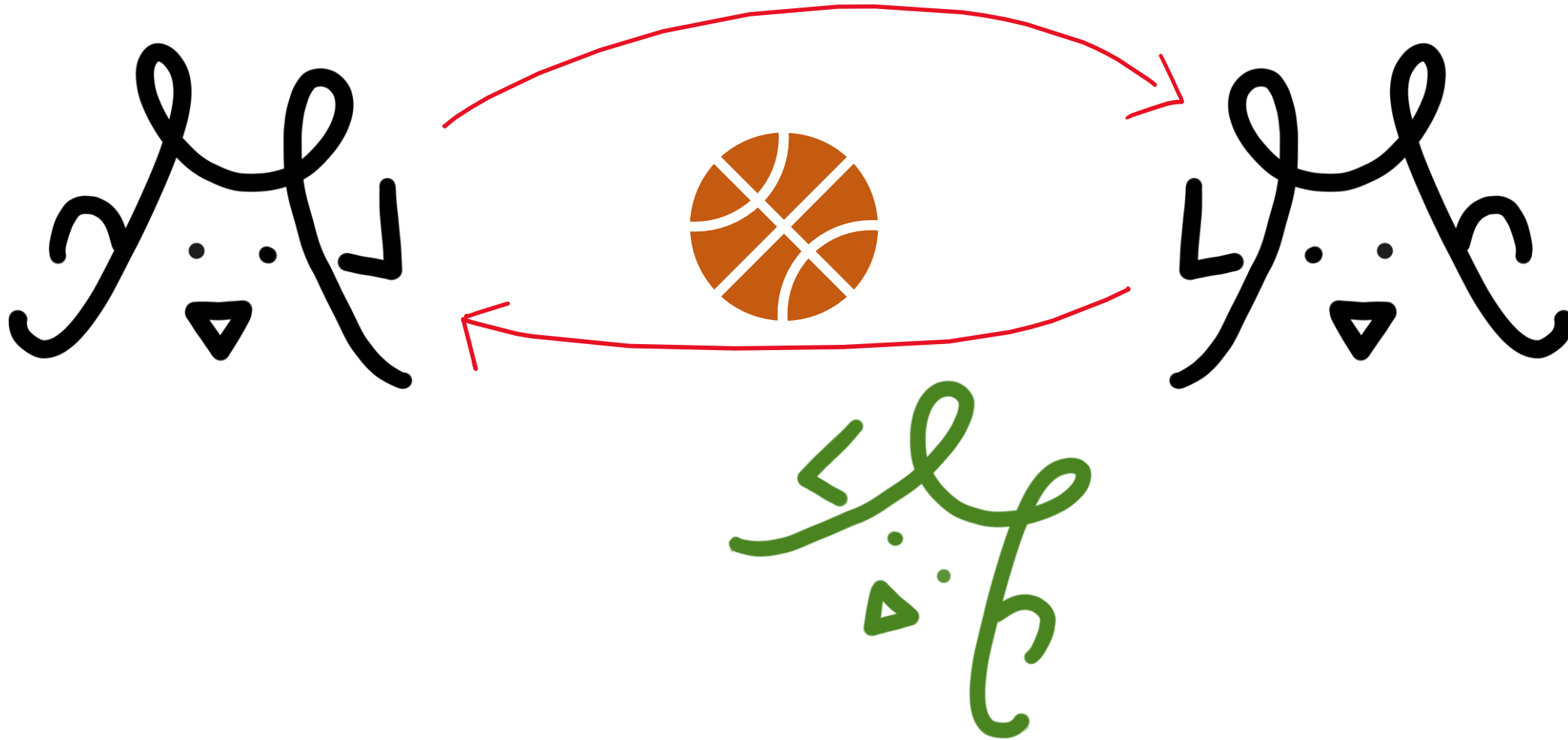
9월 19, 2023 3:59:20 오후 org.a
정보: 프로토콜 핸들러 ["http-nio-86

9월 19, 2023 3:59:20 오후 org.a
정보: 서버가 [525] 밀리초 내에 시작되

서블릿 전1
서블릿 전2
서블릿 후2
서블릿 후1

Interceptor

✓ Interceptor?



✓ Interceptor?

- HandlerInterceptor를 구현한 것 (또는 HandlerInterceptorAdapter를 상속한 것)
- 요청(requests)을 처리하는 과정에서 요청을 가로채서 처리
- 접근 제어(Auth), 로그(Log) 등 비즈니스 로직과 구분되는 반복적이고 부수적인 로직 처리
- HandlerInterceptor의 주요 메서드
 - preHandle()
 - postHandle()
 - afterCompletion()

✓ preHandle

@Override

```
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)  
    throws Exception {
```

- Controller(핸들러) 실행 이전에 호출
- false를 반환 하면 요청을 종료한다.

✓ postHandle

```
@Override  
public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler,  
    ModelAndView modelAndView) throws Exception {
```

- Controller(핸들러) 실행 후 호출
- 정상 실행 후 추가 기능 구현 시 사용
- Controller 에서 예외 발생 시 해당 메서드는 실행되지 않음

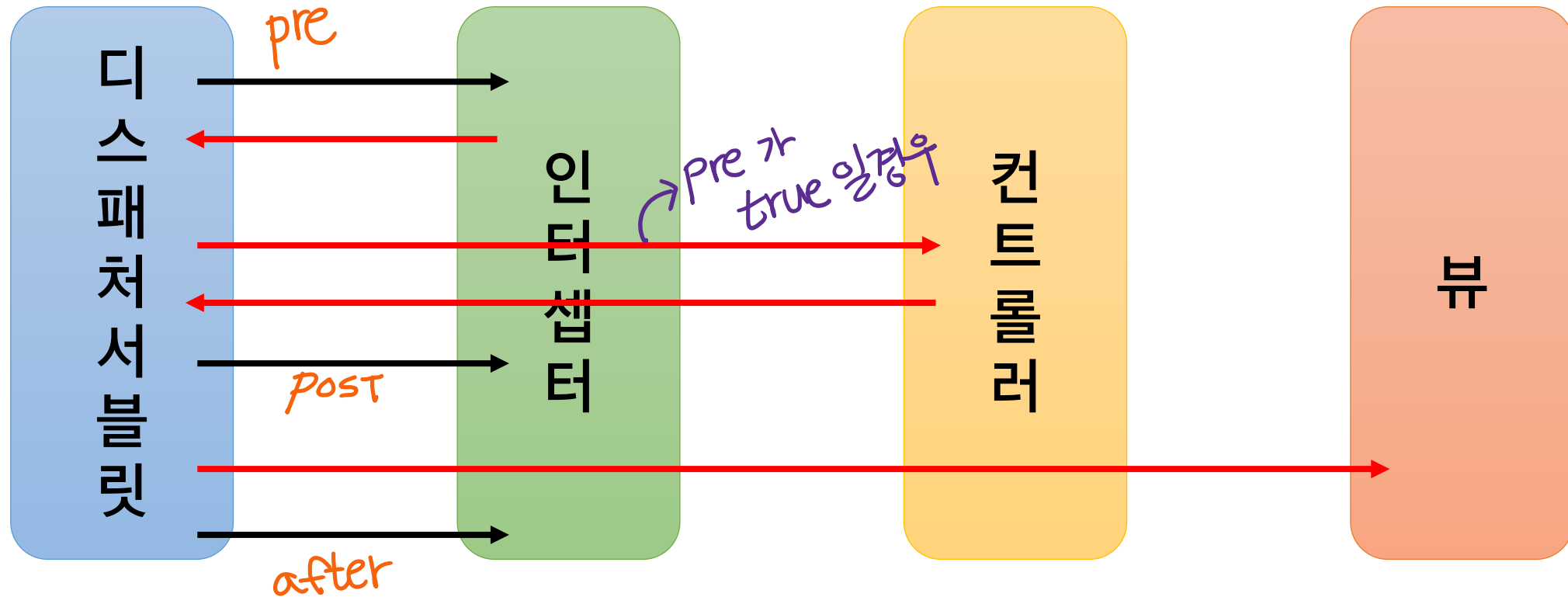
✓ afterCompletion

@Override

```
public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) throws Exception {
```

- 뷰가 클라이언트에게 응답을 전송한 뒤 실행.
- Controller 에서 예외 발생시, 네번째 파라미터로 전달이 된다. (기본은 null)
- 따라서 Controller에서 발생한 예외 혹은 실행 시간 같은 것들을 기록하는 등 후처리 시 주로 사용.

✓ Interceptor 흐름



✓ 로그인 인터셉터 만들기

```
1 package com.ssafy.mvc;
2
3 import javax.servlet.http.HttpServletRequest;
4 import javax.servlet.http.HttpServletResponse;
5 import javax.servlet.http.HttpSession;
6
7 import org.springframework.web.servlet.HandlerInterceptor;
8 import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
9
10
11 public class LoginCheckInterceptor implements HandlerInterceptor{
12     @Override
13     public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
14         throws Exception {
15         HttpSession session = request.getSession();
16         if(session.getAttribute("user") == null) {
17             response.sendRedirect(request.getContextPath() + "/login.do");
18             return false;
19         }
20         return true;
21     }
22 }
```

HandlerInterceptor **Interface** 구현

HandlerInterceptorAdapter **Class**
의 상속은 권장되지 않음(Deprecated)

preHandle 메서드 오버라이드

✓ servlet-context.xml에 bean 등록 & interceptor 매핑 설정

```

20
21<!-- Resolves views selected for rendering by @Controllers to .jsp resources
22      in the /WEB-INF/views directory -->
23<beans:bean
24      class="org.springframework.web.servlet.view.InternalResourceViewResolver">
25      <beans:property name="prefix" value="/WEB-INF/views/" />
26      <beans:property name="suffix" value=".jsp" />
27  </beans:bean>
28
29  <context:component-scan
30      base-package="com.ssafy.mvc" />
31  <beans:bean id="confirm"
32      class="com.ssafy.mvc.LoginCheckInterceptor" />
33
34  <interceptors>
35      <interceptor>
36          <mapping path="/a.do" />
37          <beans:ref bean="confirm" />
38      </interceptor>
39  </interceptors>
40
41
42 </beans:beans>
    
```

Interceptor를 bean으로 등록

요청 경로(path)와 Interceptor bean을 매핑

exclude-mapping
방식으로도 가능

```

34<interceptors>
35  <interceptor>
36    <mapping path="*" />
37    <exclude-mapping path="/b.do" />
38    <!-- <mapping path="/a.do" /> -->
39    <beans:ref bean="confirm" />
40  </interceptor>
41 </interceptors>
    
```

다음 방송에서 만나요!

삼성 청년 SW 아카데미