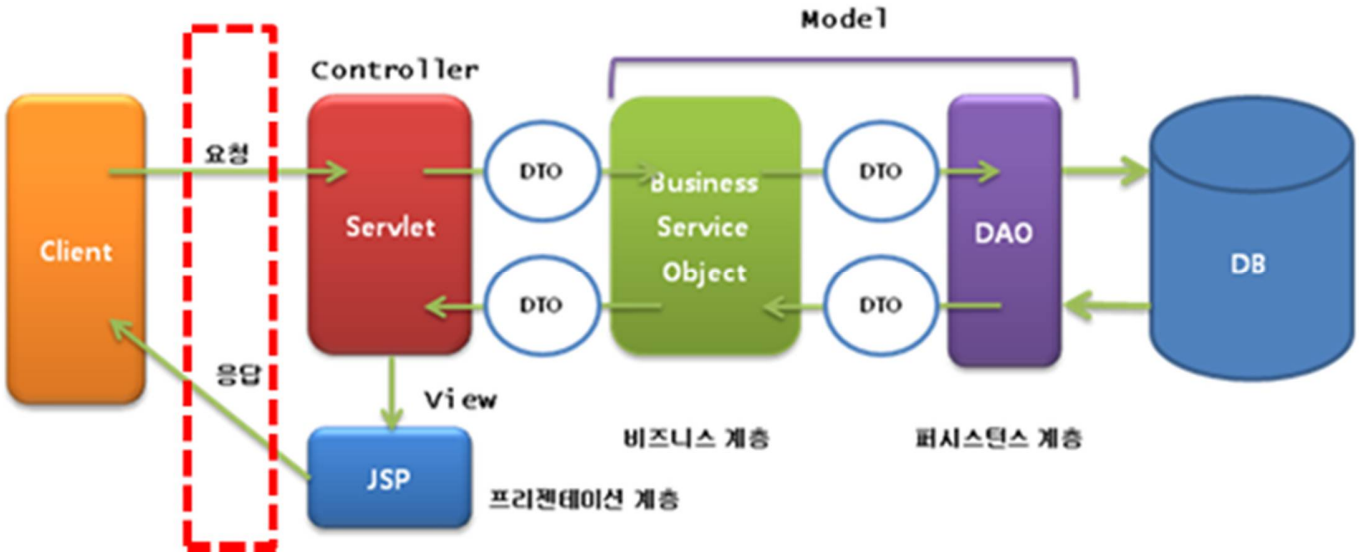
st13.인터셉터

목차

1. 인터셉터란? 2
2. interceptor 를 이용한 로그인 처리 3
2.1 코드를 이용한 로그인 체크 3
2.2 인터셉터를 이용한 로그인 체크 3
3. 인터셉터 설정 4
4. Interceptor 클래스 생성 5
5. 컨트롤러에 코드 추가 6
6. JSP 파일 추가 8
6.1 WEB-INF/view/login.jsp 추가 8
6.2 WEB-INF/view/admin_main.jsp 추가 8
7. 실행 9
7.1 "http://localhost/admin_main" 페이지 접속 9
7.2 로그인 테스트 9
8. 인터셉터, 필터, AOP 비교 11
8.1 인터셉터나 필터, AOP 의 사용 용도 : 전후 처리기 역할 11
8.2 차이점 13
8.2.1 일단 호출되는 시점이 다르다. 13
8.2.2 설정 파일이 다르다 13
8.2.3 멤버함수의 용도가 다르다. 13
9. Reference 14

1. 인터셉터란?

인터셉터는 중간에 무엇인가를 가로챈다는 의미이다. 스프링에서도 말 그대로 중간에 요청을 가로채서 어떠한 일을 하는 것을 의미한다. 서블릿(Servlet)을 사용해본 사람이라면 필터(Filter)를 들어봤을 텐데, 비슷한 의미로 사용된다. 그럼 어느 중간에서 요청을 가로채서 무엇을 하는지를 간단히 살펴보자.



인터셉터는 위 이미지의 빨간색 박스 부분에서 동작한다. 인터셉터의 정확한 명칭은 핸들러 인터셉터(Handler Interceptor)이다. 인터셉터는 DispatcherServlet 이 컨트롤러를 호출하기 전,후에 요청과 응답을 가로채서 가공할 수 있도록 해준다.

예를 들어, 로그인 기능을 구현한다고 했을 때, 어떠한 페이지를 접속하려고 할 때, 로그인 된 사용자만 보여주고, 로그인이 되어있지 않다면 메인 화면으로 이동시키려고 한다. 어떻게 하면 될까?

로그인 체크 로직을 만들어서 각 화면마다 일일이 코드를 붙여 넣으면 될 것이다. 그러나 유지보수면에 있어서는 너무 힘들다. 어떻게 하면 반복 코드를 없앨 수 있을 건인가?

스프링에서는 인터셉터를 사용하여 위의 기능을 간단히 만들 수 있다. 인터셉터에서 어떠한 요청이 들어올 때, 그 사람의 로그인 여부를 판단해서 로그인이 되어있으면 요청한 페이지로 이동시키고, 로그인이 되어있지 않을 경우 바로 메인 페이지로 이동시키면 끝이다.

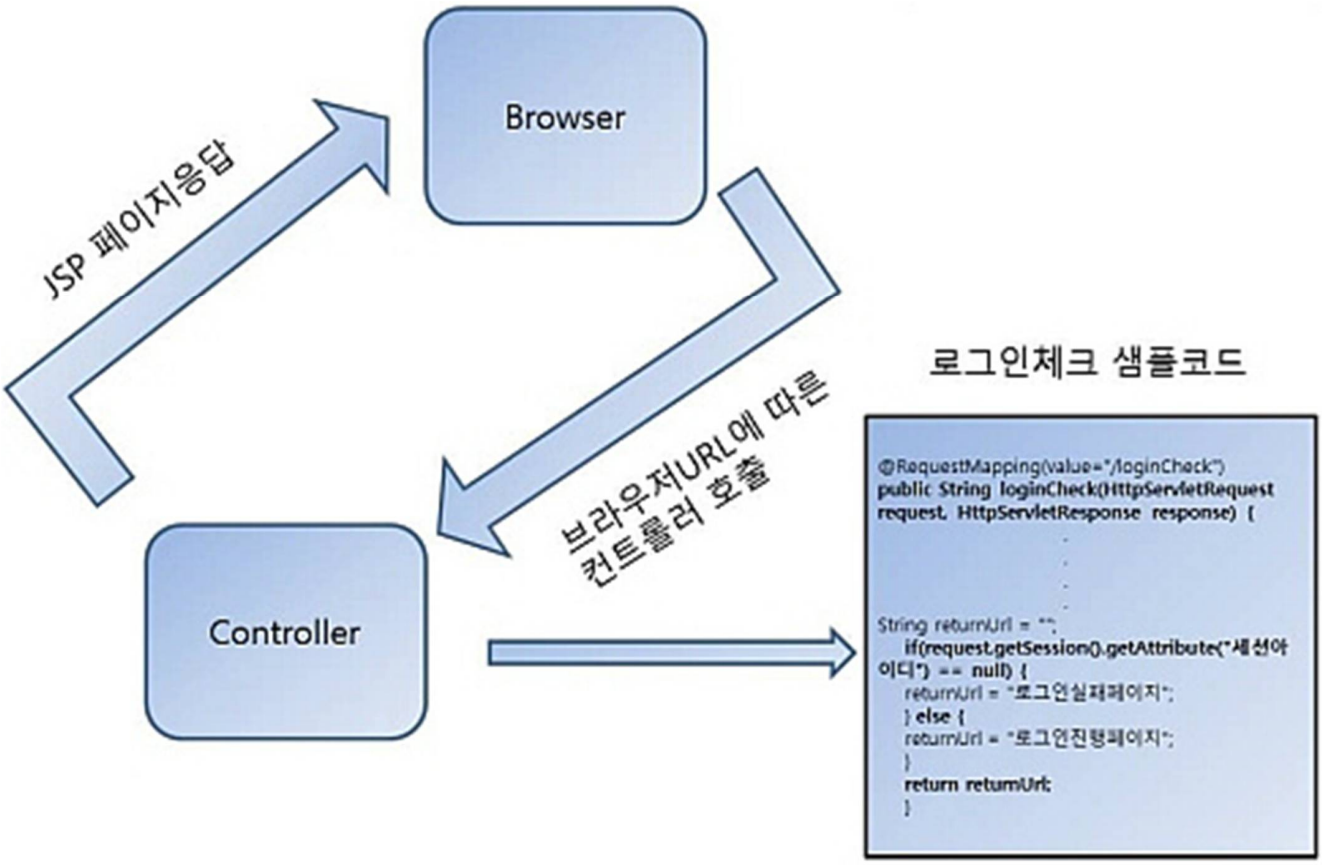
단 하나의 인터셉터로 프로젝트내의 모든 요청에서 로그인 여부를 관리할 수 있는 것이다.

2. interceptor 를 이용한 로그인 처리

- 인터셉터는 주로 컨트롤러 메서드 호출 전에 이벤트를 가로채서 어떠한 처리를 해주기 위해 사용되는 기능입니다.
- 로그인 체크 처리에서 많이 사용됩니다.

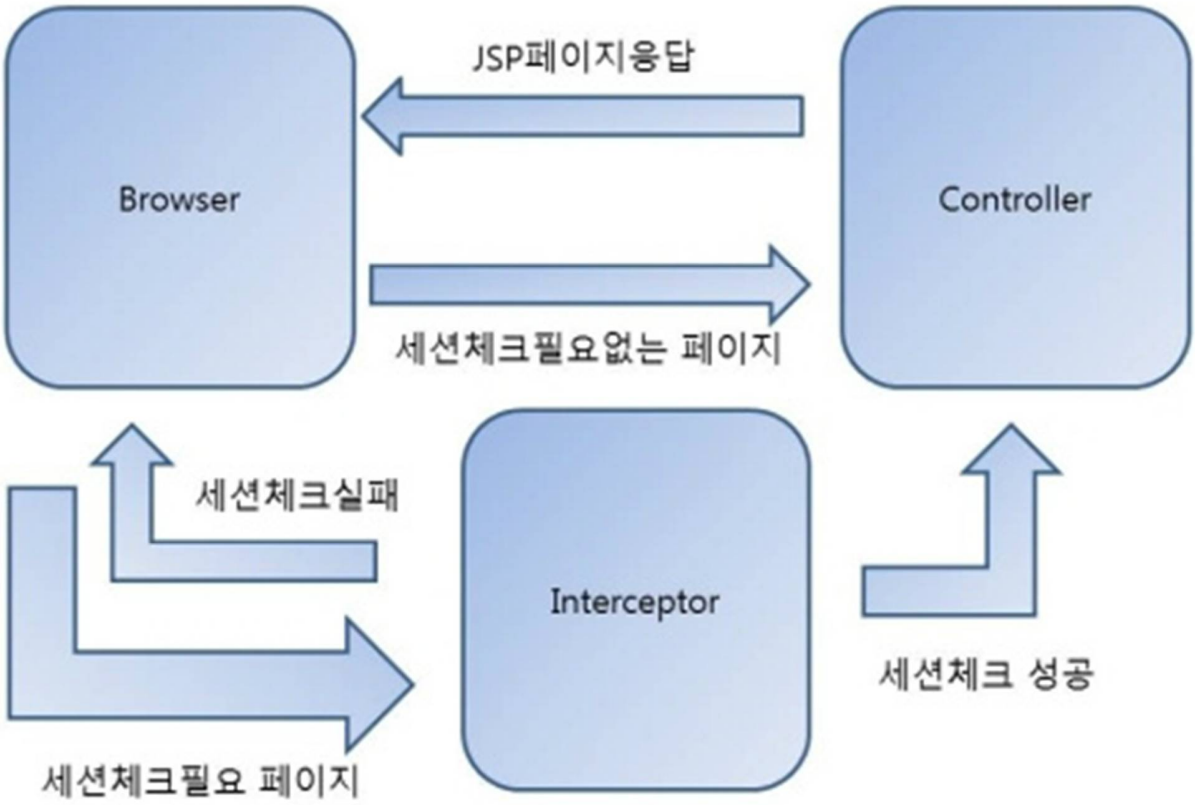
2.1 코드를 이용한 로그인 체크

인터셉터를 사용하지 않고 코드를 이용한 로그인 체크를 작성한다고 가정해보도록 하겠습니다



위처럼 로그인이 필수인 컨트롤러를 작성할 때 마다 로그인 체크 로직을 CTRL + C/V 해주어야 합니다. 코드도 길어지고 로그인 체크 일부가 변경되면 모두 변경해주어야 하는 번거로움이 있습니다.

2.2 인터셉터를 이용한 로그인 체크



● 위처럼 인터셉터를 거쳐서 통과 시 컨트롤러 호출이 진행됩니다.
세션체크가 필요한 페이지는 XML 설정에 의해 정의해줌으로써 각 페이지마다 로그인 체크 해주는 코드를 작성해줄 필요가 없습니다.

3. 인터셉터 설정

servlet-context.xml 에 코드 추가

```
<!-- 인터셉터 설정 -->
<interceptors>
    <interceptor>
        <!-- 여러 개 컨트롤러 추가 가능
             <mapping path="/admin_write"/>
             <mapping path="/admin_update"/>
             <mapping path="/admin_delete"/>
        -->
        <mapping path="/admin_main" />

        <!-- 로그인 체크 인터셉터 클래스 -->
        <beans:bean class="com.spring61.intercepter.Intercepter" />
    </interceptor>
```

```
</interceptors>
```

✓ servlet-context.xml 에는 여러 개의 인터셉터를 정의할 수 있습니다.

✓ interceptor 태그들을 interceptors 내에 인터셉터 여러 개를 정의할 수 있습니다.

ex) 관리자 로그인, 사용자 로그인 등...

✓ mapping 태그들을 interceptor 태그 내에 여러 개 정의하여 하나의 인터셉터로 여러 개의 컨트롤러를 제어 할 수 있습니다.

ex) 관리자 로그인 후 글쓰기페이지,글 수정페이지,삭제기능 등....

4. Interceptor 클래스 생성

```
public class Intercepter extends
org.springframework.web.servlet.handler.HandlerInterceptorAdapter {

    @Override
    public boolean preHandle(HttpServletRequest request,
            HttpServletResponse response, Object handler) throws Exception {
        // return super.preHandle(request, response, handler);

        try {
            //admin 이라는 세션 key 를 가진 정보가 null 일 경우 로그인 페이지로 이동
            if(request.getSession().getAttribute("admin") == null ){
                    response.sendRedirect("/login");
                    return false;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        //admin 세션 key 존재 시 main 페이지 이동
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest request,
            HttpServletResponse response, Object handler,
            ModelAndView modelAndView) throws Exception {
        // TODO Auto-generated method stub
        super.postHandle(request, response, handler, modelAndView);
    }
```

```
    @Override
    public void afterCompletion(HttpServletRequest request,
            HttpServletResponse response, Object handler, Exception ex)
            throws Exception {
        // TODO Auto-generated method stub
        super.afterCompletion(request, response, handler, ex);
    }

}
```

위에 보시면 3 가지의 인터셉터에 대한 override 속성이 존재합니다.

1. preHandle - controller 이벤트 호출 전
2. postHandle - controller 호출 후 view 페이지 출력 전
3. afterCompletion - controller + view 페이지 모두 출력 후

5. 컨트롤러에 코드 추가

```
@Controller
public class HomeController {

      @RequestMapping(value = "/", method = RequestMethod.GET)
      public String home(Model model) {

            return "redirect:/admin_main";
      }

    @RequestMapping(value = "/login", method = RequestMethod.GET)
    public String login(Model model) {

        model.addAttribute("userid"   , "admin" );
        model.addAttribute("password", "1234" );

        return "login";
    }

    @RequestMapping(value = "/login", method = RequestMethod.POST)
    public String login( HttpSession session
            , @RequestParam(value="userid"   , required = true) String userid
            , @RequestParam(value="password", required = true) String password
            , @RequestParam(value="usertype", required = true) String usertype ){

        String returnURL = "";
```

```
        if(userid.equals("admin") && password.equals("1234") && usertype.equals("admin")){

            // 아이디,패스워드 일치 시 admin 세션 key 생성
            Map<String, Object> map = new HashMap<String,Object>();
            map.put("admin_id", "admin");
            map.put("admin_name", "관리자");
            session.setAttribute("admin", map);

            returnURL = "redirect:/admin_main";
        }
        else {
            //일치하지 않으면 로그인체이지 재 이동
            returnURL = "redirect:/";
        }

        return returnURL;
    }

    @RequestMapping(value = "/admin_main", method = RequestMethod.GET)
    public String admin_main(){
        return "admin_main";
    }

}
```

HomeController 에 메서드를 생성하였는데

"/login" 메서드는 임의로 로그인 체크를 위하여 생성한 메서드이고

"/admin_main" 메서드는 로그인 이후 세션이 생성된 시점에 접근이 가능한 관리자 메서드입니다.

6. JSP 파일 추가

6.1 WEB-INF/view/login.jsp 추가

```
<%@ page session="false" language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>인터셉터</title>
</head>
<body>

    <div class="login">
        <form action="/login" method="post">
            <label> 이름을 입력하세요 : </label>
            <input type="text" name="userid" value="${userid}"/>  <br/>
            <label> 패스워드를 입력하세요 :</label>
            <input type="password"  name="password" value="${password}" />  <br/>
            <input type="checkbox"  name="usertype" value="user" checked="checked"
>사용자</input>
            <input type="checkbox"  name="usertype" value="admin" >관리자</input>  <br/>
            <input type="submit" id="submit" value="전송"/>
        </form>
    </div>

</body>
</html>
```

6.2 WEB-INF/view/admin_main.jsp 추가

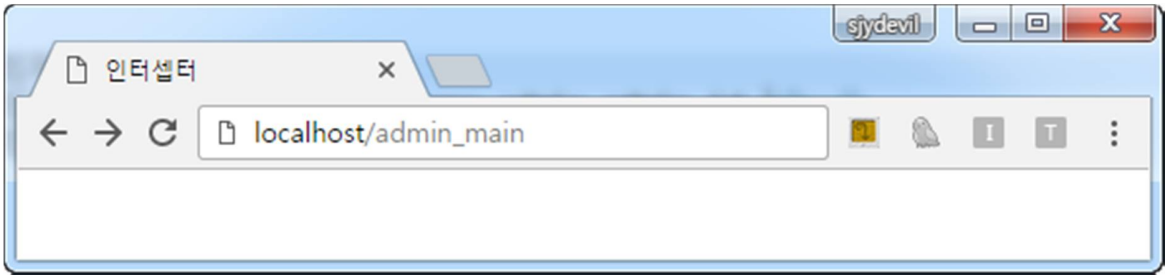
```
<%@ page session="true" language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<html>
<head>
      <title>Home</title>
</head>
<body>
```

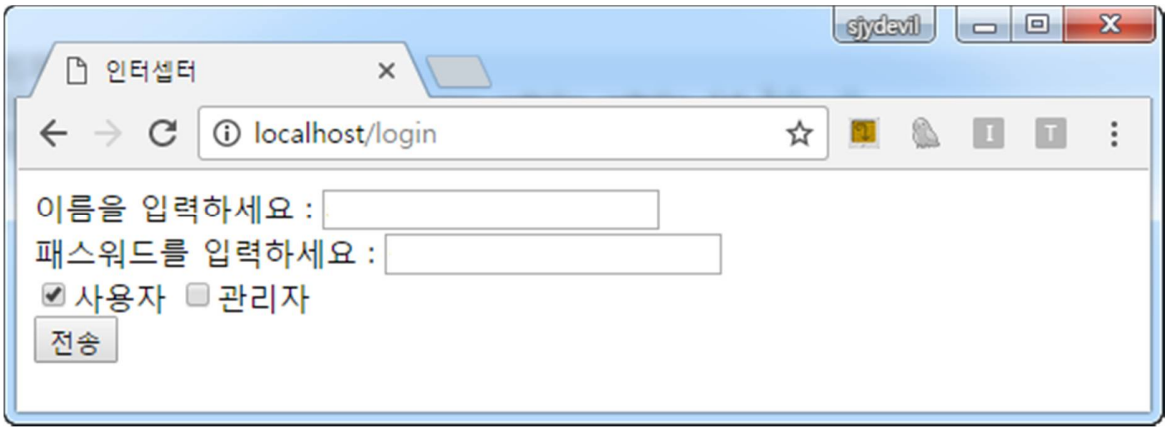
```
<h3>관리자 메인 페이지</h3>
    관리자 아이디 : ${sessionScope.admin.admin_id }<br/>
    관리자 이름: ${sessionScope.admin.admin_name }
</body>
</html>
```

위처럼 코드 작성을 해보았다면 실행을 해보도록 합니다.

7. 실행

7.1 "http://localhost/admin_main" 페이지 접속

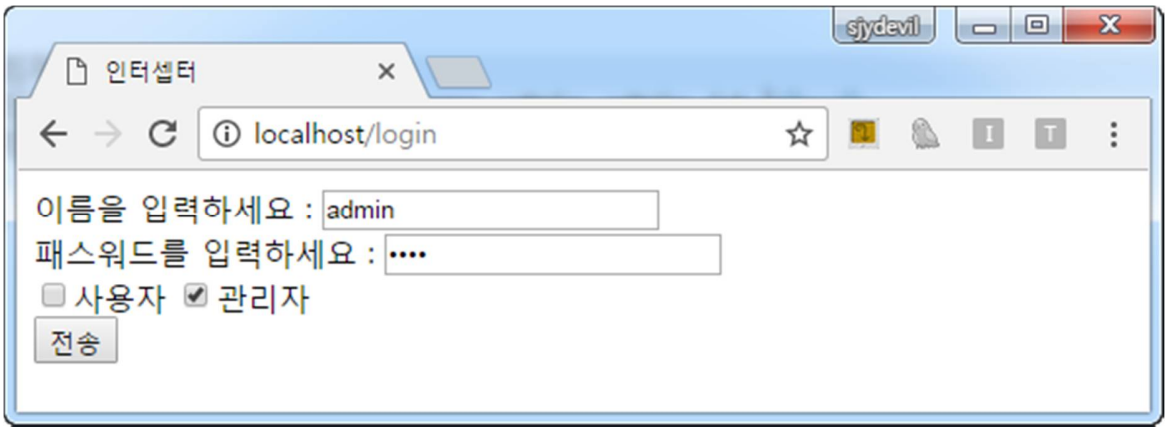


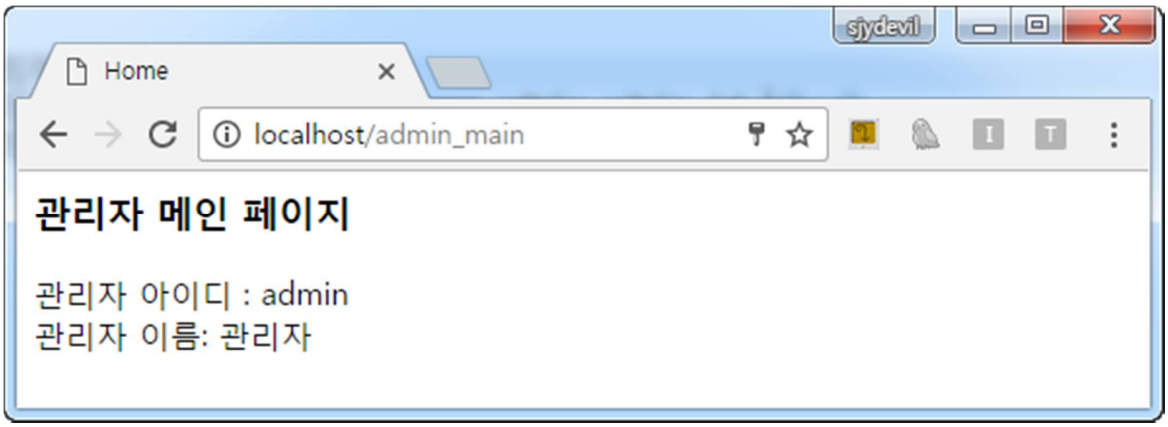


실행결과 메인 URL 로 이동되어 로그인 페이지가 출력되었습니다. 왜냐하면 세션이 존재 하지 않기 때문에 자동으로 인터셉터에서 튕겨준 것입니다.

7.2 로그인 테스트

admin/1234 를 입력하여 전송버튼을 클릭해주도록 합니다.





그럼 위처럼 관리자 아이디와 이름을 확인할 수 있는 "/admin_main" 페이지로 성공적으로 이동이 되었습니다.

새로고침해도 역시 세션이 존재하고 있어서 페이지가 유지되게 있습니다.

위처럼 인터셉터를 이용하여 로그인 세션체크를 해주시면 되겠습니다.

8. 인터셉터, 필터, AOP 비교

8.1 인터셉터나 필터, AOP 의 사용 용도 : 전후 처리기 역할

스프링에서 interceptor, filter, aop 셋 다 무슨 행동을 하기 전에 먼저 실행하거나, 실행 한 후에 추가적인 행동을 할 때 사용 되는 기능들입니다.

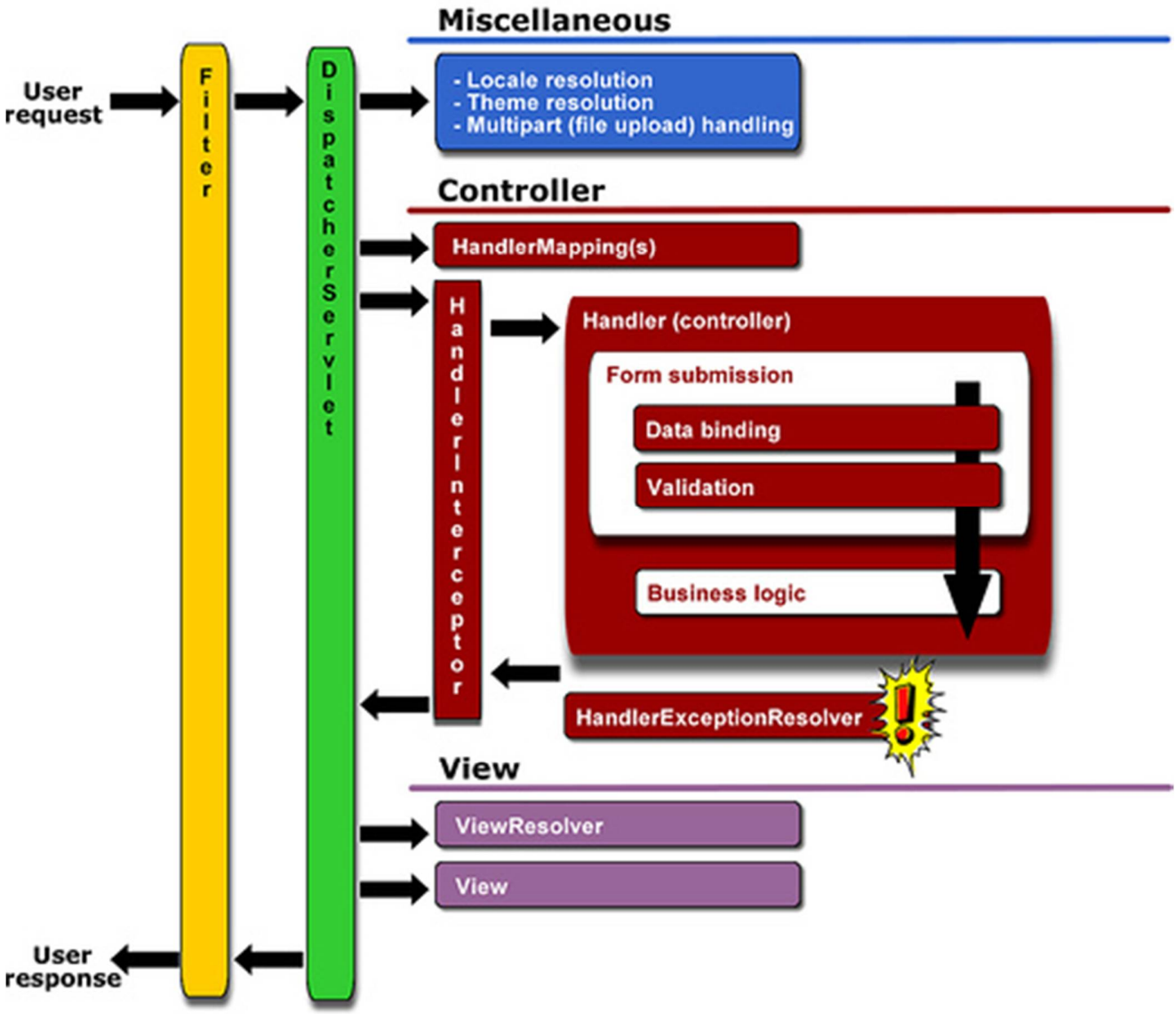
특히 인터셉터와 필터만을 놓고 보면 이 두 개는 무척 비슷해 보인다. filter 로 해야 되는 일들도 interceptor 로 해결 할 수 있는 듯하다.

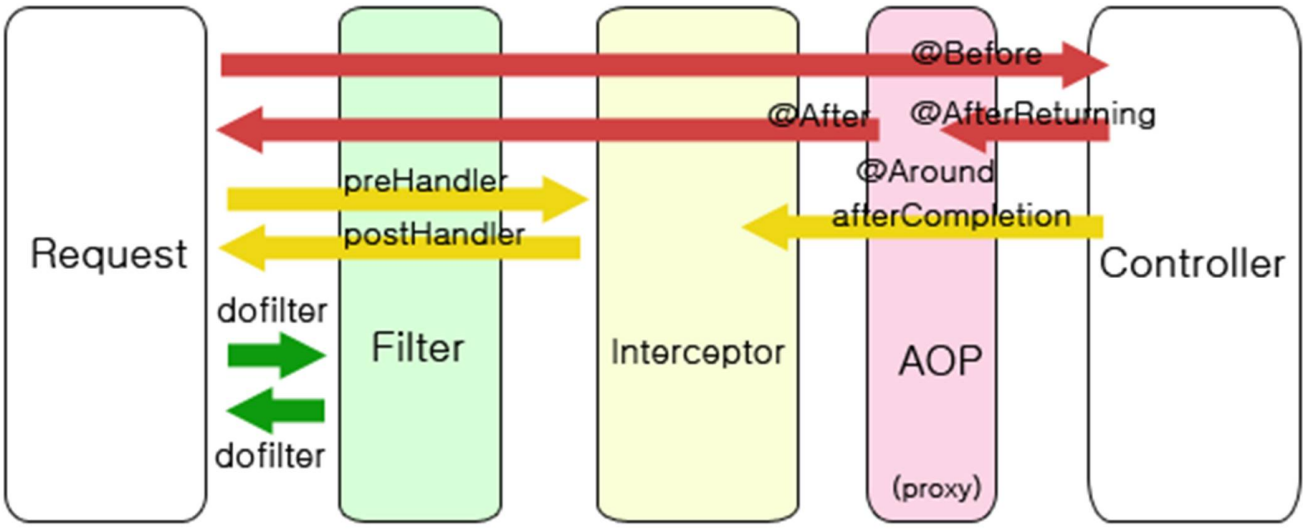
	Filter	Interceptor	AOP
실행 위치	서블릿	서블릿	메소드
실행 순서	1(제일 먼저와 제일 나중)	2	3
설정 위치	web.xml	xml or java	xml or java
실행 메소드	init dofilter destroy	preHandler postHandler afterCompletion	@after @before @around

실행 위치는 interceptor 와 filter 는 서블릿 단위에서 실행됩니다. 반면에 AOP 는 메서드 단위에서 실행됩니다.

실행 순서는 스프링이 구동될 때 필터의 init 이 실행되고, 그 후 dofilter 가 실행됩니다. 그 후 컨트롤러에 들어가기 전에 preHandler 가 실행되고, aop 가 실행된 후에 컨트롤러에서 나와 postHandler, after Completion, dofilter 순서대로 진행되고, 서블릿 종료시 destory 가 실행 될 것입니다.

그림으로 표현해보자면 이렇게 될 것 같네요.


Miscellaneous
User request
Filter
DispatcherServlet
- Locale resolution
- Theme resolution
- Multipart (file upload) handling
Controller
HandlerMapping(s)
HandlerInterceptor
Handler (controller)
Form submission
Data binding
Validation
Business logic
HandlerExceptionResolver
View
ViewResolver
View
User response


Request
dofilter
dofilter
Filter
preHandler
postHandler
Interceptor
@Before
@After
@AfterReturning
@Around
afterCompletion
AOP
(proxy)
Controller

- 인터셉터 설정: servlet.xml

```
<interceptors>
    <interceptor>
        <mapping path="/api1/*" />
        <mapping path="/api2/*" />
        <mapping path="/api3/*" />
        <beans:bean class="인터셉터클래스" />
    </interceptor>
</interceptors>
```

이 뜻은 http://사이트주소/api1, http:// 사이트주소/api2, http://사이트주소/api3 로 접속하면 먼저 인터셉터를 먼저 호출하겠다는 것이다.

- 필터 설정 : web.xml

```
<filter>
    <filter-name>필터이름</filter-name>
    <filter-class>com.changpd.test.filter.필터클래스</filter-class>
</filter>
<filter-mapping>
    <filter-name>필터이름</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

이것도 해석해보면 어떤게 호출되든지 간에 (/*) '필터이름' 부터 먼저 호출하겠다.

인터셉터나 필터나 자신이 전후 처리기역할을 수행하려는 의도이다.

8.2 차이점

8.2.1 일단 호출되는 시점이 다르다.

보통 인터셉터나 필터나 컨트롤러 들어가기 전에 작업을 처리 하기 위해 사용하므로

별반 차이 없어 보일 수 있으나 위의 라이프사이클 그림을 보면 호출되는 시점이 다르다는 걸 알 수 있다.

8.2.2 설정 파일이 다르다

필터의 경우는 설정 정보를 web.xml 에 하는데 반해 인터셉터의 경우는 설정 정보를 servlet.xml 에 한다.

8.2.3 멤버함수의 용도가 다르다.

인터셉터	preHandle()	컨트롤러 들어가기 전
	postHanle()	컨트롤러 들어갔다 나온 후 뷰로 보내기 전
	afterCompletion()	뷰까지 끝나고 나서
필터	init()	필터 인스턴스 초기화
	doFilter()	전/후 처리
	destroy()	필터 인스턴스 종료

doFilter 함수는 보통 아래처럼 작성된다. 필터는 doFilter 함수에서 전후 처리를 모두 담당하는데 doFilter 가 요청전과 후, 두 번 호출되는 방식이다.

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
throws IOException, ServletException {

    // 전 처리 코드
    ...

    chain.doFilter(request, response);

    // 후 처리 코드
    ...
}
```

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 있습니다.

인터셉터와 필터의 기능은 비슷하지만, 필터의 경우 호출 시점이 자유롭지 못하다
보니 사용성에서는 인터셉터보다 떨어진다.

9. Reference

http://addio3305.tistory.com/43
http://hellogk.tistory.com/90
http://changpd.blogspot.kr/2013/03/spring.html
http://blog.naver.com/platinasnow/220035316135