

Anyframe Login Plugin



Version 1.0.1

저작권 © 2007-2011 삼성SDS

본 문서의 저작권은 삼성SDS에 있으며 Anyframe 오픈소스 커뮤니티 활동의 목적하에서 자유로운 이용이 가능합니다. 본 문서를 복제, 배포할 경우에는 저작권자를 명시하여 주시기 바라며 본 문서를 변경하실 경우에는 원문과 변경된 내용을 표시하여 주시기 바랍니다. 원문과 변경된 문서에 대한 상업적 용도의 활용은 허용되지 않습니다. 본 문서에 오류가 있다고 판단될 경우 이슈로 등록해 주시면 적절한 조치를 취하도록 하겠습니다.

I. Introduction	1
II. Login	2
1. Authentication	3
1.1. Id, Password Authentication	3
1.2. 사용자 정보 조회	3
2. Session	4
2.1. 사용자 정보를 Session 저장	4
2.2. AuthenticationInterceptor	4
3. Biz. Service에서 사용자 정보 이용	6
3.1. ThreadLocalUtil	6
4. 사용자 Locale 변경	7
4.1. ChangeLocaleController	7

I.Introduction

어프리케이션에서 사용자가 로그인 했을 때, 사용자 정보를 세션에 저장하고 사용자 정보에 따른 로케일 설정, ThreadLocal을 이용해 서비스에서 사용자 정보를 사용하는 예제를 포함하고 있다.

Installation

Command 창에서 다음과 같이 명령어를 입력하여 login plugin을 설치한다.

```
mvn anyframe:install -Dname=login
```

installed(mvn anyframe:installed) 혹은 jetty:run(mvn clean jetty:run) command를 이용하여 설치 결과를 확인해볼 수 있다.

Dependent Plugins

Plugin Name	Version Range
core [http://dev.anyframejava.org/docs/anyframe/plugin/essential/core/1.0.2/reference/htmlsingle/core.html]	2.0.0 > *

II.Login

일반적인 Web Application 에서는 사용자 인증을 위해서는 Id, Password방식을 많이 사용한다.

Login Plugin은 아래와 같은 단계로 진행된다.



- 1. 사용자 인증
 - 2. 사용자 정보 조회
 - 3. 사용자 정보 세션 저장 및 로케일 설정
 - 4. Interceptor에서 ThreadLocal에 세션정보 저장
 - 5. Biz. Service에서 ThreadLocal활용
 - 6. User Locale 변경
-

1.Authentication

1.1.Id, Password Authentication

보통은 Web Application의 사용자 인증은 Id, Password 방식을 많이 사용한다. 아래는 Login Plugin의 로그인 화면이다. 사용자 Locale에 따라 두 명의 사용자가 등록되어 있다.



User : Password :

Locale	ID	Password
	anyframe_ko	anyframe0
	anyframe_en	anyframe1

로그인 화면에서 아이디, 패스워드를 입력한 후 Login 버튼을 클릭하면 AuthenticationController에서 사용자 입력값(id, password)을 이용해 사용자인증 절차를 거친다. Login Plugin에서는 AuthenticationService에서 사용자 인증을 한다. 만약 인증에 실패 했을 경우에는 아래와 같이 화면에 인증실패 메시지를 보여준다.

User : Password :

Ivaild id or password.

Locale	ID	Password
	anyframe_ko	anyframe0
	anyframe_en	anyframe1

1.2.사용자 정보 조회

인증에 성공하게 되면 Application에서 필요한 사용자 정보를 조회한다. 사용자 이름, 사용자 로케일 정보를 조회한다. Login Plugin에서는 UserService를 이용해 사용자 정보를 조회한다. 아래는 인증에 성공한 후 UserService의 API를 호출해 사용자 정보를 조회하는 샘플소스이다.

```
boolean certification = authenticationService.authenticate(loginInfo);

if(certification){
    UserInfo userInfo = userService.getUserInfo(id);
    ..중략
}else{
    ..중략
}
```

2.Session

2.1.사용자 정보를 Session 저장

Application에서 사용자 정보는 자주 사용되므로 Session에 저장해 필요할 때 Session에서 꺼내 쓰도록 한다. 그러기 위해서는 Session에 앞서 조회한 사용자 정보를 저장하는 로직이 필요하다.

아래는 사용자 인증, 정보조회, 세션 저장하는 AuthenticationController의 일부이다.

```
@RequestMapping("/login.do")
public String login(@RequestParam("id") String id,
    @RequestParam("password") String password, Model model,
    HttpSession session, HttpServletRequest request, HttpServletResponse response) throws
    Exception {

    //사용자 인증
    LoginInfo loginInfo = new LoginInfo();
    loginInfo.setId(id);
    loginInfo.setPassword(password);
    loginInfo.setIpAddress(request.getRemoteAddr());
    SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMddHHmmss");
    formatter.setTimeZone(TimeZone.getTimeZone("GMT"));
    loginInfo.setTime(formatter.format(new Date()));

    boolean certification = authenticationService.authenticate(loginInfo);

    if(certification){
        UserInfo userInfo = userService.getUserInfo(id);
        localeResolver.setLocale(request, response, userInfo.getLanguage());
        session.setAttribute("userInfo", userInfo);
        return "forward:/loginMovieFinder.do?method=list";
    }else{
        model.addAttribute("loginError", "login.error");
        return "/login/login";
    }
}
```

사용자 정보를 userInfo란 이름으로 Session에 저장한다.

2.2.AuthenticationInterceptor

AuthenticationInterceptor는 사용자가 서버에 Request를 보낼 경우 Session에 저장된 사용자 정보를 꺼내 ThreadLocal에 저장하는 역할을 한다. ThreadLocal에 저장된 값들은 Request가 끝나는 시점까지 유지되기 때문에 Biz. Service Layer에서 언제든지 필요한 정보를 꺼낼 수 있다.

AuthenticationInterceptor 설정은 아래와 같다.

```
<mvc:interceptors>
  <bean class="org.anyframe.plugin.login.interceptor.AuthenticationInterceptor" />
</mvc:interceptors>
```

AuthenticationInterceptor의 preHandler메소드는 아래와 같다.

```
public boolean preHandle(HttpServletRequest request,
    HttpServletResponse response, Object handler) throws Exception {
    HttpSession session = request.getSession();
```

```
if ( session != null ){
    UserInfo userInfo = (UserInfo) session.getAttribute("userInfo");
    if ( userInfo != null ){
        ThreadLocalUtil.add("userInfo", (UserInfo) session.getAttribute("userInfo"));
    }
}
return super.preHandle(request, response, handler);
}
```

Session에 저장된 사용자 정보를 userInfo란 이름으로 ThreadLocal에 저장하고 있다. AuthenticationInterceptor의 afterCompletion메소드에서는 반드시 ThreadLocal에 저장된 값을 clearSharedInfo을 호출해 삭제하도록 한다.

```
public void afterCompletion(HttpServletRequest request,
    HttpServletResponse response, Object handler, Exception ex)
    throws Exception {
    ThreadLocalUtil.clearSharedInfo();
    super.afterCompletion(request, response, handler, ex);
}
```

3.Biz. Service에서 사용자 정보 이용

3.1.ThreadLocalUtil

ThreadLocalUtil은 ThreadLocal을 보다 쉽게 사용할 수 있도록 제공되는 클래스이다. Interceptor에서 Session에 저장된 사용자 정보를 ThreadLocalUtil의 add메소드를 이용해 ThreadLocal에 저장했다면 Biz. Service에서 언제든지 필요한 정보를 얻을 수 있다.

예를들어 각 Service Method에 AOP를 이용해 해당 메소드를 어떤 사용자가 호출했는지 로깅을 남길 수도 있다. 아래는 LoggingUserInfoAspect의 소스코드이다.

```
@Aspect
@Service
public class LoggingUserInfoAspect {

    @Pointcut("execution(* org.anyframe.plugin.login.*Impl.*(..))")
    public void loggingMethod() {
    }

    @Before("loggingMethod()")
    public void beforeLogging(JoinPoint thisJoinPoint) {
        Class<? extends Object> clazz = thisJoinPoint.getTarget().getClass();

        StringBuilder messageBuf = new StringBuilder();

        if( ThreadLocalUtil.isExist("userInfo")){
            UserInfo userInfo = (UserInfo)ThreadLocalUtil.get("userInfo");

            messageBuf

            .append("\n-----\n");
            messageBuf.append(" This method called by " + userInfo.getUserName() +"\n");
            messageBuf

            .append("-----");
        }

        Log logger = LoggerFactory.getLog(clazz);

        if (logger.isInfoEnabled()) {
            logger.info(messageBuf.toString());
        }
    }
}
```

Biz. Service의 메소드가 호출되면 어떤 User가 메소드를 실행 했는지 로그를 남긴다.

4. 사용자 Locale 변경

4.1. ChangeLocaleController

Spring Message 태그는 code(message key)값을 이용해 MessageSource에 있는 해당 문자를 사용자에게 보여준다. 이 때 LocaleResolver를 이용해 다국어 처리도 가능하다. ChangeLocaleController는 사용자가 다른 언어로 보고 싶을 때 SessionLocaleResolver의 언어를 바꿈으로써 다국어 처리를 지원한다.



The screenshot shows a web application interface with a search bar and a table of movie results. The search bar contains the text "타이틀:" followed by an input field. To the right of the search bar, there is a dropdown menu labeled "상영중:" with the value "Playing" and a search icon. The table has three columns: "배우" (Actor), "티켓가격" (Ticket Price), and "개봉일" (Release Date). The table contains three rows of data:

배우	티켓가격	개봉일
Johnny Depp	8000.0	2010-03-04
Sigourney Weaver	7000.0	2010-02-16
Yigal Naor	7000.0	2010-03-25

위 화면에서 보고 싶은 언어를 클릭하면 ChangeLocaleController의 changeLocale메소드가 호출되어 LocaleResolver의 로케일을 변경한다.

아래는 ChangeLocaleController의 소스코드이다.

```
@Controller("loginChangeLocaleController")
public class ChangeLocaleController {



    @Inject
    private LocaleResolver localeResolver;


    @RequestMapping("/loginChangeLocale.do")
    public String changeLocale(@RequestParam(value = "locale", defaultValue = "en") String
newLocale,
    Model model,
    HttpSession session, HttpServletRequest request, HttpServletResponse response) throws
Exception {

        LocaleEditor localeEditor = new LocaleEditor();
        localeEditor.setAsText(newLocale);
        localeResolver.setLocale(request, response, (Locale) localeEditor.getValue());

        return "forward:/loginMovieFinder.do?method=list";
    }
}
```

LocaleResolver의 값이 변경되면 화면에서 다음과 같이 다른 언어로 변경된 것을 확인 할 수 있다.

  [Logout](#)

Title: Now Playing: Playing ▼ 

	Actors	Ticket Price	Release Date
	Johnny Depp	8000.0	2010-03-04
	Sigourney Weaver	7000.0	2010-02-16
	Yigal Naor	7000.0	2010-03-25