5장 인터셉터(Interceptor)를 활용하는 로그인 처리

5.1 Spring MVC의 인터셉터(Interceptor)

5.1.1 Filter와 인터셉터의 공통점과 차이점

- Servlet 기술의 Filter와 Spring MVC의 HandlerInterceptor는 특정 URI에 접근할 때 제어하는 용도로 사용된다는 공통점을 가지고 있다.

- Filter의 경우 웹 애플리케이션내에서 동작하므로, 스프링의 Context를 접근하기 어렵다. Interceptor의 경우 스프링에서 관리되기 때문에 스프링 내의 모든 객체(빈)에 접근이 가능하여 활용할 수 있다.

5.1.2 Spring AOP 기능과 HandlerInterceptor의 차이

- AOP Advice의 경우 JoinPoint나 ProceedingJoinPoint 등을 활용해서 호출 대상이 되는 메소드의 파라미터 등을 처리하는 방식이다.

- 반면에 HandlerInterceptor는 Filter와 유사하게 HttpServletRequest, HttpServletResponse를 파라미터로 받는 구조이다.

(1) HandlerInterceptorAdapter 클래스

- HandlerInterceptor는 인터페이스로 정의되어 있지만, HandlerInterceptorAdaptor는 HandlerInterceptor를 쉽게 사용하기 위해서 인터페이스의 메소드를 미리 구현한 클래스이다.

5.1.3 예제를 위한 프로젝트의 생성

- 예제 프로젝트는 'ex05Lab' 이름으로 생성한다.

(1) 패키지의 생성과 SampleInterceptor

- HandlerInterceptorAdaptor를 상속해서 'org.zerock.interceptor.SampleInterceptor' 클래스를 작성한다.

```
package org.zerock.interceptor;
import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
public class SampleInterceptor extends HandlerInterceptorAdapter {
}
```

(2) servlet-context.xml의 인터셉터 설정

```
[ex05/src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml]
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
		xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:beans="http://www.springframework.org/schema/beans"
		xmlns:context="http://www.springframework.org/schema/context"
		xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
			http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
			http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

		<!-- DispatcherServlet Context: defines this servlet's request-processing
			infrastructure -->

		<!-- Enables the Spring MVC @Controller programming model -->
		<annotation-driven />

		<!-- Handles HTTP GET requests for /resources/** by efficiently serving
			up static resources in the ${webappRoot}/resources directory -->
		<resources mapping="/resources/**" location="/resources/" />

		<!-- Resolves views selected for rendering by @Controllers to .jsp resources
			in the /WEB-INF/views directory -->
		<beans:bean
			class="org.springframework.web.servlet.view.InternalResourceViewResolver">
			<beans:property name="prefix" value="/WEB-INF/views/" />
			<beans:property name="suffix" value=".jsp" />
		</beans:bean>

		<context:component-scan base-package="org.zerock.controller" />

		<beans:bean id="multipartResolver"
			class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
			<beans:property name="maxUploadSize" value="10485760"></beans:property>
		</beans:bean>

		<beans:bean id="uploadPath" class="java.lang.String">
			<beans:constructor-arg value="C:\\zzz\\upload">
			</beans:constructor-arg>
		</beans:bean>

		<beans:bean id="sampleInterceptor"
class="org.zerock.interceptor.SampleInterceptor"></beans:bean>

		<beans:bean id="authInterceptor" class="org.zerock.interceptor.AuthInterceptor"></beans:bean>

		<beans:bean id="loginInterceptor"
class="org.zerock.interceptor.LoginInterceptor"></beans:bean>

		<interceptors>

			<interceptor>
				<mapping path="/user/loginPost" />
				<beans:ref bean="loginInterceptor" />
			</interceptor>

			<interceptor>
				<mapping path="/sboard/register" />
				<mapping path="/sboard/modifyPage" />
				<mapping path="/sboard/removePage" />
				<beans:ref bean="authInterceptor" />
			</interceptor>

			<interceptor>
```

```
66                                  <mapping path="/doA" />
67                                  <mapping path="/doB" />
68                                  <beans:ref bean="sampleInterceptor" />
69                          </interceptor>
70
71              </interceptors>
72
73
74      </beans:beans>
```

(3) HomeController의 수정

[ex05/src/main/java/org/zerock/controller/HomeController.java]

```
01  package org.zerock.controller;
02
03  import java.text.DateFormat;
04  import java.util.Date;
05  import java.util.Locale;
06
07  import org.slf4j.Logger;
08  import org.slf4j.LoggerFactory;
09  import org.springframework.stereotype.Controller;
10  import org.springframework.ui.Model;
11  import org.springframework.web.bind.annotation.RequestMapping;
12  import org.springframework.web.bind.annotation.RequestMethod;
13
14  /**
15   * Handles requests for the application home page.
16   */
17  @Controller
18  public class HomeController {
19
20    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
21
22    /**
23     * Simply selects the home view to render by returning its name.
24     */
25    @RequestMapping(value = "/", method = RequestMethod.GET)
26    public String home(Locale locale, Model model) {
27      logger.info("Welcome home! The client locale is {}.", locale);
28
29      Date date = new Date();
30      DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);
31
32      String formattedDate = dateFormat.format(date);
33
34      model.addAttribute("serverTime", formattedDate);
35
36      return "home";
37    }
38
39    @RequestMapping(value = "/doA", method = RequestMethod.GET)
40    public String doA(Locale locale, Model model) {
41
42
43      System.out.println("doA.....................");
44
45      return "home";
46    }
47
48    @RequestMapping(value = "/doB", method = RequestMethod.GET)
```

```
49      public String doB(Locale locale, Model model) {
50
51
52        System.out.println("doB.....................");
53
54        model.addAttribute("result", "DOB RESULT");
55
56        return "home";
57      }
58
59
60      @RequestMapping(value = "/test", method = RequestMethod.GET)
61      public void ajaxTest() {
62
63      }
64
65    }
```

(4) SampleInterceptor의 처리

[ex05/src/main/java/org/zerock/interceptor/SampleInterceptor.java]

```
01    package org.zerock.interceptor;
02
03    import java.lang.reflect.Method;
04
05    import javax.servlet.http.HttpServletRequest;
06    import javax.servlet.http.HttpServletResponse;
07
08    import org.springframework.web.method.HandlerMethod;
09    import org.springframework.web.servlet.ModelAndView;
10    import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
11
12    public class SampleInterceptor extends HandlerInterceptorAdapter {
13
14
15      @Override
16      public void postHandle(HttpServletRequest request,
17          HttpServletResponse response, Object handler,
18          ModelAndView modelAndView) throws Exception {
19
20        System.out.println("post handle.........................");
21
22        Object result = modelAndView.getModel().get("result");
23
24        if(result != null){
25          System.out.println("result exists");
26          request.getSession().setAttribute("result", result);
27          response.sendRedirect("/doA");
28        }
29
30      }
31
32
33
34      @Override
35      public boolean preHandle(HttpServletRequest request,
36          HttpServletResponse response, Object handler) throws Exception {
37
38        System.out.println("pre handle.........................");
39
40        HandlerMethod method = (HandlerMethod) handler;
```

```
41        Method methodObj = method.getMethod();
42
43        System.out.println("Bean: " + method.getBean());
44        System.out.println("Method: " + methodObj);
45
46        return true;
47
48      }
49
50    }
51
52
53    //@Override
54    //public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler,
55    //  ModelAndView modelAndView) throws Exception {
56    //
57    //System.out.println("post handle.........");
58    //
59    //}
60
61    // @Override
62    // public boolean preHandle(HttpServletRequest request,
63    // HttpServletResponse response, Object handler) throws Exception {
64    //
65    // System.out.println("pre handle..........");
66    //
67    // return true;
68    // }
```

(5) 실행 및 결과

```
// http://localhost:8080/doA
pre handle.........................
Bean: org.zerock.controller.HomeController@add3f4
Method: public java.lang.String
org.zerock.controller.HomeController.doA(java.util.Locale,org.springframework.ui.Model)
doA...................
post handle.......................

// http://localhost:8080/doB
pre handle.........................
Bean: org.zerock.controller.HomeController@add3f4
Method: public java.lang.String
org.zerock.controller.HomeController.doB(java.util.Locale,org.springframework.ui.Model)
doB...................
post handle.......................
result exists
pre handle.........................
Bean: org.zerock.controller.HomeController@add3f4
Method: public java.lang.String
org.zerock.controller.HomeController.doA(java.util.Locale,org.springframework.ui.Model)
doA...................
post handle.......................
```

5.1.4 인터셉터의 request, response 활용하기

(1) preHandle()의 Object 파라미터

- preHandle()의 경우 세 개의 파라미터를 사용하는데, 마지막의 Handler는 현재 실행하려는 메소드 자체를 의미한다.

```
// org.zerock.interceptor.SampleInterceptor.java의 일부

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

		System.out.println("pre handle.....................");

		HandlerMethod method = (HandlerMethod) handler;
		Method methodObj = method.getMethod();

		System.out.println("Bean: " + method.getBean());
		System.out.println("Method: " + methodObj);

		return true;
	}
```

(2) postHandle()을 이용해서 추가적인 작업하기

```
// org.zerock.interceptor.SampleInterceptor.java의 일부

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

		System.out.println("post handle.....................");

		Object result = modelAndView.getModel().get("result");

		if (result != null) {
			System.out.println("result exists");
			request.getSession().setAttribute("result", result);
			response.sendRedirect("/doA");
		}
	}
```

5.2 HttpSession을 이용하는 로그인 처리

- HttpSession의 동작은 실제로는 세션 쿠키(session cookie)를 통해서 이뤄지는데, 서버는 필요한경우 접속한 브라우저에게 고유한 세션 쿠키를 전달하고, 매번 브라우저에서 서버를 호출할 때 세션 쿠키를 같이 가지고 다니기 때문에, 이를 마치 열쇠처럼 사용해서 필요한 데이터를 보관한다.

- 세션 쿠키가 열쇠(key)라면 HttpSession은 열쇠가 필요한 잠금장치가 되어 있는 상자와 유사하다. 이 상자들이 모여 있는 공간을 '세션 저장소(Session Repository)'라고 하는데, 많은 세션이 존재하면 서버의 성능에 영향을 미치기 때문에, 서버에는 일정 시간 이상 사용되지 않는 상자들을 정리하는 기능이 있다. (web.xml을 이용해서 HttpSession의 timeout을 지정할 수 있음)

5.2.1 준비 작업

(1) 테이블 생성 및 객체 처리

```
create table tbl_user (
          uid varchar(50) NOT NULL,
          upw varchar(50) NOT NULL,
          uname varchar(100) NOT NULL,
          upoint int NOT NULL DEFAULT 0,
          primary key (uid)
);

create table tbl_message (
          mid int not null auto_increment,
          targetid varchar(50) not null,
          sender varchar(50) not null,
          message text not null,
          opendate timestamp,
          senddate timestamp not null default now(),
          primary key (mid)
);

alter table tbl_message add constraint fk_usertarget
foreign key (targetid) references tbl_user (uid);

alter table tbl_message add constraint fk_usersender
foreign key (targetid) references tbl_user (uid);

insert into tbl_user(uid,upw,uname) values ('user00','user00','IRON MAN');
insert into tbl_user(uid,upw,uname) values ('user01','user01','CAPTAIN');
insert into tbl_user(uid,upw,uname) values ('user02','user02','HULK');
insert into tbl_user(uid,upw,uname) values ('user03','user03','Thor');
insert into tbl_user(uid,upw,uname) values ('user10','user10','Quick Silver');

-- 댓글 카운트의 처리
alter table tbl_board add column replycnt int default 0;


-- 게시물 등록의 파일 업로드
create table tbl_attach (
          fullName varchar(150) not null,
          bno int not null,
          regdate timestamp default now(),
          primary key (fullName)
);

alter table tbl_attach add constraint fk_board_attach
foreign key (bno) references tbl_board (bno);


-- 자동 로그인과 쿠키
alter table tbl_user add column
sessionkey varchar(50) not null default 'none';

alter table tbl_user
add column sessionlimit timestamp;
```

[ex05/src/main/java/org/zerock/domain/UserVO.java]

```
01    package org.zerock.domain;
02
03    public class UserVO {
```

```
04
05      private String uid;
06      private String upw;
07      private String uname;
08      private int upoint;
09
10      public String getUid() {
11        return uid;
12      }
13
14      public void setUid(String uid) {
15        this.uid = uid;
16      }
17
18      public String getUpw() {
19        return upw;
20      }
21
22      public void setUpw(String upw) {
23        this.upw = upw;
24      }
25
26      public String getUname() {
27        return uname;
28      }
29
30      public void setUname(String uname) {
31        this.uname = uname;
32      }
33
34      public int getUpoint() {
35        return upoint;
36      }
37
38      public void setUpoint(int upoint) {
39        this.upoint = upoint;
40      }
41
42      @Override
43      public String toString() {
44        return "UserVO [uid=" + uid + ", upw=" + upw + ", uname=" + uname + ", upoint=" + upoint + "]";
45      }
46    }
```

[ex05/src/main/java/org/zerock/dto/LoginDTO.java]

```
01    package org.zerock.dto;
02
03    public class LoginDTO {
04
05              private String uid;
06              private String upw;
07              private boolean useCookie;
08
09              public String getUid() {
10                        return uid;
11              }
12              public void setUid(String uid) {
13                        this.uid = uid;
14              }
15              public String getUpw() {
16                        return upw;
17              }
18              public void setUpw(String upw) {
```

```
19                      this.upw = upw;
20              }
21              public boolean isUseCookie() {
22                      return useCookie;
23              }
24              public void setUseCookie(boolean useCookie) {
25                      this.useCookie = useCookie;
26              }
27
28              @Override
29              public String toString() {
30                      return "LoginDTO [uid=" + uid + ", upw=" + upw + ", useCookie="
31                                      + useCookie + "]";
32              }
33
34      }
```

[꿀팁] VO(Value Object)와 DTO(Data Transfer Object)

- DTO와 VO의 용도는 데이터의 수집과 전달에 사용할 수 있다는 공통점이 있다. 양쪽 모두 파라미터나 리턴 타입으로 사용하는 것이 가능하다. 다만 VO가 보다 데이터베이스와의 거리가 가깝다. 즉, VO는 테이블의 구조를 이용해서 작성되는 경우가 더 많다. DTO의 경우는 보다 화면과 가깝다. 화면에서 전달되는 데이터를 수집하는 용도로 사용하는 경우가 많다.
- 스프링 MVC를 이용하는 경우 DTO는 검증을 위한 처리가 들어간다. 스프링은 Controller에 전달되는 데이터에 대해서 검증하는 기능을 추가할 수 있는데, 이러한 상황에서는 별도의 DTO를 구성해서 사용한다.

(2) UserDAO의 생성 및 SQL 처리

```
// org.zerock.persistence.UserDAO
package org.zerock.persistence;

import java.util.Date;

import org.zerock.domain.UserVO;
import org.zerock.dto.LoginDTO;

public interface UserDAO {

          public UserVO login(LoginDTO dto)throws Exception;

  public void keepLogin(String uid, String sessionId, Date next);

  public UserVO checkUserWithSessionKey(String value);
}
```

```
// resources/mappers/UserMapper.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="org.zerock.mapper.UserMapper">

  <select id="login" resultType="UserVO">
```

```
  select uid, upw, uname from tbl_user where uid = #{uid} and upw = #{upw}

  </select>

  <update id="keepLogin" >
  update tbl_user set sessionKey = #{sessionId}, sessionLimit = #{next} where uid = #{uid}
  </update>

          <select id="checkUserWithSessionKey" resultType="UserVO">
          select * from tbl_user where sessionKey = #{value}  and sessionlimit > now()
          </select>

</mapper>
```

[ex05/src/main/java/org/zerock/persistence/UserDAOImpl.java]

```
01    package org.zerock.persistence;
02
03    import java.util.Date;
04    import java.util.HashMap;
05    import java.util.Map;
06
07    import javax.inject.Inject;
08
09    import org.apache.ibatis.session.SqlSession;
10    import org.springframework.stereotype.Repository;
11    import org.zerock.domain.UserVO;
12    import org.zerock.dto.LoginDTO;
13
14    @Repository
15    public class UserDAOImpl implements UserDAO {
16
17              @Inject
18              private SqlSession session;
19
20              private static String namespace ="org.zerock.mapper.UserMapper";
21
22              @Override
23              public UserVO login(LoginDTO dto) throws Exception {
24
25                        return session.selectOne(namespace +".login", dto);
26              }
27
28      @Override
29      public void keepLogin(String uid, String sessionId, Date next) {
30
31        Map<String, Object> paramMap = new HashMap<String, Object>();
32        paramMap.put("uid", uid);
33        paramMap.put("sessionId", sessionId);
34        paramMap.put("next", next);
35
36        session.update(namespace+".keepLogin", paramMap);
37
38      }
39
40      @Override
41      public UserVO checkUserWithSessionKey(String value) {
42
43        return session.selectOne(namespace +".checkUserWithSessionKey", value);
44      }
45    }
```

(3) UserService와 UserServiceImpl 처리

[ex05/src/main/java/org/zerock/service/UserService.java]

```
01    package org.zerock.service;
02
03    import java.util.Date;
04
05    import org.zerock.domain.UserVO;
06    import org.zerock.dto.LoginDTO;
07
08    public interface UserService {
09
10      public UserVO login(LoginDTO dto) throws Exception;
11
12      public void keepLogin(String uid, String sessionId, Date next)throws Exception;
13
14      public UserVO checkLoginBefore(String value);
15    }
```

[ex05/src/main/java/org/zerock/service/UserServiceImpl.java]

```
01    package org.zerock.service;
02
03    import java.util.Date;
04
05    import javax.inject.Inject;
06
07    import org.springframework.stereotype.Service;
08    import org.zerock.domain.UserVO;
09    import org.zerock.dto.LoginDTO;
10    import org.zerock.persistence.UserDAO;
11
12    @Service
13    public class UserServiceImpl implements UserService {
14
15      @Inject
16      private UserDAO dao;
17
18      @Override
19      public UserVO login(LoginDTO dto) throws Exception {
20
21        return dao.login(dto);
22      }
23
24      @Override
25      public void keepLogin(String uid, String sessionId, Date next)
26          throws Exception {
27
28        dao.keepLogin(uid, sessionId, next);
29
30      }
31
32      @Override
33      public UserVO checkLoginBefore(String value) {
34
35        return dao.checkUserWithSessionKey(value);
36      }
37    }
```

5.2.2 컨트롤러의 처리

(1) UserController의 작성

[ex05/src/main/java/org/zerock/controller/UserController.java]

```
01  package org.zerock.controller;
02
03  import java.util.Date;
04
05  import javax.inject.Inject;
06  import javax.servlet.http.Cookie;
07  import javax.servlet.http.HttpServletRequest;
08  import javax.servlet.http.HttpServletResponse;
09  import javax.servlet.http.HttpSession;
10
11  import org.springframework.stereotype.Controller;
12  import org.springframework.ui.Model;
13  import org.springframework.web.bind.annotation.ModelAttribute;
14  import org.springframework.web.bind.annotation.RequestMapping;
15  import org.springframework.web.bind.annotation.RequestMethod;
16  import org.springframework.web.util.WebUtils;
17  import org.zerock.domain.UserVO;
18  import org.zerock.dto.LoginDTO;
19  import org.zerock.service.UserService;
20
21  @Controller
22  @RequestMapping("/user")
23  public class UserController {
24
25    @Inject
26    private UserService service;
27
28    @RequestMapping(value = "/login", method = RequestMethod.GET)
29    public void loginGET(@ModelAttribute("dto") LoginDTO dto) {
30
31    }
32
33    // @RequestMapping(value = "/loginPost", method = RequestMethod.POST)
34    // public void loginPOST(LoginDTO dto, HttpSession session, Model model)
35    // throws Exception {
36    //
37    // UserVO vo = service.login(dto);
38    //
39    // if (vo == null) {
40    // return;
41    // }
42    //
43    // model.addAttribute("userVO", vo);
44    //
45    // }
46
47    @RequestMapping(value = "/loginPost", method = RequestMethod.POST)
48    public void loginPOST(LoginDTO dto, HttpSession session, Model model) throws Exception {
49
50      UserVO vo = service.login(dto);
51
52      if (vo == null) {
53        return;
54      }
55
56      model.addAttribute("userVO", vo);
57
58      if (dto.isUseCookie()) {
```

```
59
60          int amount = 60 * 60 * 24 * 7;
61
62          Date sessionLimit = new Date(System.currentTimeMillis() + (1000 * amount));
63
64          service.keepLogin(vo.getUid(), session.getId(), sessionLimit);
65        }
66
67      }
68
69      @RequestMapping(value = "/logout", method = RequestMethod.GET)
70      public void logout(HttpServletRequest request,
71          HttpServletResponse response, HttpSession session) throws Exception {
72
73        Object obj = session.getAttribute("login");
74
75        if (obj != null) {
76          UserVO vo = (UserVO) obj;
77
78          session.removeAttribute("login");
79          session.invalidate();
80
81          Cookie loginCookie = WebUtils.getCookie(request, "loginCookie");
82
83          if (loginCookie != null) {
84            loginCookie.setPath("/");
85            loginCookie.setMaxAge(0);
86            response.addCookie(loginCookie);
87            service.keepLogin(vo.getUid(), session.getId(), new Date());
88          }
89        }
90      }
91
92    }
```

5.2.3 LoginInterceptor의 작성 및 설정

(1) LoginInterceptor의 작성

[ex05/src/main/java/org/zerock/interceptor/LoginInterceptor.java]

```
01    package org.zerock.interceptor;
02
03    import javax.servlet.http.Cookie;
04    import javax.servlet.http.HttpServletRequest;
05    import javax.servlet.http.HttpServletResponse;
06    import javax.servlet.http.HttpSession;
07
08    import org.slf4j.Logger;
09    import org.slf4j.LoggerFactory;
10    import org.springframework.ui.ModelMap;
11    import org.springframework.web.servlet.ModelAndView;
12    import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
13
14    public class LoginInterceptor extends HandlerInterceptorAdapter {
15
16      private static final String LOGIN = "login";
17      private static final Logger logger = LoggerFactory.getLogger(LoginInterceptor.class);
18
19      @Override
20      public void postHandle(HttpServletRequest request,
21          HttpServletResponse response, Object handler,
```

```
22          ModelAndView modelAndView) throws Exception {
23
24        HttpSession session = request.getSession();
25
26        ModelMap modelMap = modelAndView.getModelMap();
27        Object userVO = modelMap.get("userVO");
28
29        if (userVO != null) {
30
31          logger.info("new login success");
32          session.setAttribute(LOGIN, userVO);
33
34          if (request.getParameter("useCookie") != null) {
35
36            logger.info("remember me.............");
37            Cookie loginCookie = new Cookie("loginCookie", session.getId());
38            loginCookie.setPath("/");
39            loginCookie.setMaxAge(60 * 60 * 24 * 7);
40            response.addCookie(loginCookie);
41          }
42          // response.sendRedirect("/");
43          Object dest = session.getAttribute("dest");
44
45          response.sendRedirect(dest != null ? (String) dest : "/");
46        }
47      }
48
49      // @Override
50      // public void postHandle(HttpServletRequest request,
51      // HttpServletResponse response, Object handler,
52      // ModelAndView modelAndView) throws Exception {
53      //
54      // HttpSession session = request.getSession();
55      //
56      // ModelMap modelMap = modelAndView.getModelMap();
57      // Object userVO = modelMap.get("userVO");
58      //
59      // if(userVO != null){
60      //
61      // logger.info("new login success");
62      // session.setAttribute(LOGIN, userVO);
63      // //response.sendRedirect("/");
64      //
65      // Object dest = session.getAttribute("dest");
66      //
67      // response.sendRedirect(dest != null ? (String)dest:"/");
68      // }
69      // }
70
71      @Override
72      public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
73    throws Exception {
74
75        HttpSession session = request.getSession();
76
77        if (session.getAttribute(LOGIN) != null) {
78          logger.info("clear login data before");
79          session.removeAttribute(LOGIN);
80        }
81
82        return true;
83      }
84    }
```

(2) LoginInterceptor의 설정

```
// /WEB-INF/spring/appServlet/servlet-context.xml의 일부
          <beans:bean id="loginInterceptor" class="org.zerock.interceptor.LoginInterceptor"></beans:bean>

          <interceptors>
                    <interceptor>
                              <mapping path="/user/loginPost" />
                              <beans:ref bean="loginInterceptor" />
                    </interceptor>
          </interceptors>
```

(3) 로그인의 화면 처리

[ex05/src/main/webapp/WEB-INF/views/user/login.jsp]

```
01    <%@ page language="java" contentType="text/html; charset=UTF-8"
02        pageEncoding="UTF-8"%>
03    <!DOCTYPE html>
04    <html>
05      <head>
06        <meta charset="UTF-8">
07        <title>AdminLTE 2 | Log in</title>
08        <meta content='width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no'
09    name='viewport'>
10        <!-- Bootstrap 3.3.4 -->
11        <link href="/resources/bootstrap/css/bootstrap.min.css" rel="stylesheet" type="text/css" />
12        <!-- Font Awesome Icons -->
13        <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.css"
14    rel="stylesheet" type="text/css" />
15        <!-- Theme style -->
16        <link href="/resources/dist/css/AdminLTE.min.css" rel="stylesheet" type="text/css" />
17        <!-- iCheck -->
18        <link href="/resources/plugins/iCheck/square/blue.css" rel="stylesheet" type="text/css" />
19
20        <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
21        <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
22        <!--[if lt IE 9]>
23            <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
24            <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
25        <![endif]-->
26      </head>
27      <body class="login-page">
28        <div class="login-box">
29          <div class="login-logo">
30            <a href="/resources/index2.html"><b>Zerock</b>Project</a>
31          </div><!-- /.login-logo -->
32          <div class="login-box-body">
33            <p class="login-box-msg">Sign in to start your session</p>
34
35    <form action="/user/loginPost" method="post">
36      <div class="form-group has-feedback">
37        <input type="text" name="uid" class="form-control" placeholder="USER ID"/>
38        <span class="glyphicon glyphicon-envelope form-control-feedback"></span>
39      </div>
40      <div class="form-group has-feedback">
41        <input type="password" name="upw" class="form-control" placeholder="Password"/>
42        <span class="glyphicon glyphicon-lock form-control-feedback"></span>
43      </div>
44      <div class="row">
```

```
45        <div class="col-xs-8">
46          <div class="checkbox icheck">
47            <label>
48              <input type="checkbox" name="useCookie"> Remember Me
49            </label>
50          </div>
51        </div><!-- /.col -->
52        <div class="col-xs-4">
53          <button type="submit" class="btn btn-primary btn-block btn-flat">Sign In</button>
54        </div><!-- /.col -->
55      </div>
56    </form>
57
58
59          <a href="#">I forgot my password</a><br>
60          <a href="register.html" class="text-center">Register a new membership</a>
61
62        </div><!-- /.login-box-body -->
63      </div><!-- /.login-box -->
64
65      <!-- jQuery 2.1.4 -->
66      <script src="/resources/plugins/jQuery/jQuery-2.1.4.min.js"></script>
67      <!-- Bootstrap 3.3.2 JS -->
68      <script src="/resources/bootstrap/js/bootstrap.min.js" type="text/javascript"></script>
69      <!-- iCheck -->
70      <script src="/resources/plugins/iCheck/icheck.min.js" type="text/javascript"></script>
71      <script>
72        $(function () {
73          $('input').iCheck({
74            checkboxClass: 'icheckbox_square-blue',
75            radioClass: 'iradio_square-blue',
76            increaseArea: '20%' // optional
77          });
78        });
79      </script>
80    </body>
81  </html>
```

[ex05/src/main/webapp/WEB-INF/views/user/loginPost.jsp]

```
01  <%@ page language="java" contentType="text/html; charset=UTF-8"
02           pageEncoding="UTF-8"%>
03  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04  "http://www.w3.org/TR/html4/loose.dtd">
05  <html>
06  <head>
07  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08  <title>Insert title here</title>
09  </head>
10  <body>
11           <script type="text/javascript">
12                    self.location = "/sboard/list";
13           </script>
14  </body>
15  </html>
```

5.2.4 AuthInterceptor의 작성 및 설정

(1) AuthInterceptor의 작성

■ 로그인하지 않은 사용자는 로그인해야 하는 화면으로 이동하게 만든다.

[org.zerock.interceptor.AuthInterceptor.java]

```
01  package org.zerock.interceptor;
02
03  import javax.inject.Inject;
04  import javax.servlet.http.Cookie;
05  import javax.servlet.http.HttpServletRequest;
06  import javax.servlet.http.HttpServletResponse;
07  import javax.servlet.http.HttpSession;
08
09  import org.slf4j.Logger;
10  import org.slf4j.LoggerFactory;
11  import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
12  import org.springframework.web.util.WebUtils;
13  import org.zerock.domain.UserVO;
14  import org.zerock.service.UserService;
15
16  public class AuthInterceptor extends HandlerInterceptorAdapter {
17
18          private static final Logger logger = LoggerFactory.getLogger(AuthInterceptor.class);
19
20          @Inject
21          private UserService service;
22
23          @Override
24          public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object
25  handler)
26                          throws Exception {
27
28                  HttpSession session = request.getSession();
29
30                  if (session.getAttribute("login") == null) {
31
32                          logger.info("current user is not logined");
33
34                          saveDest(request);
35
36                          Cookie loginCookie = WebUtils.getCookie(request, "loginCookie");
37
38                          if (loginCookie != null) {
39
40                                  UserVO userVO =
41  service.checkLoginBefore(loginCookie.getValue());
42
43                                  logger.info("USERVO: " + userVO);
44
45                                  if (userVO != null) {
46                                          session.setAttribute("login", userVO);
47                                          return true;
48                                  }
49
50                          }
51
52                          response.sendRedirect("/user/login");
53                          return false;
54                  }
55                  return true;
56          }
57
58          private void saveDest(HttpServletRequest req) {
59
60                  String uri = req.getRequestURI();
61
```

```
62                      String query = req.getQueryString();
63
64                      if (query == null || query.equals("null")) {
65                              query = "";
66                      } else {
67                              query = "?" + query;
68                      }
69
70                      if (req.getMethod().equals("GET")) {
71                              logger.info("dest: " + (uri + query));
72                              req.getSession().setAttribute("dest", uri + query);
73                      }
74              }
75
76      }
```

(2) AuthInterceptor의 설정

■ AuthInterceptor는 특정 경로에 접근하는 경우 현재 사용자가 로그인한 상태의 사용자인지를 체크하는 역할을 처리하기 위해서 작성한다.

```
// /WEB-INF/spring/appServlet/servlet-context.xml의 일부

        <beans:bean id="loginInterceptor" class="org.zerock.interceptor.LoginInterceptor"></beans:bean>

        <interceptors>
                <interceptor>
                        <mapping path="/user/loginPost" />
                        <beans:ref bean="loginInterceptor" />
                </interceptor>
                <interceptor>
                        <mapping path="/sboard/list" />
                        <mapping path="/sboard/register" />
                        <mapping path="/sboard/modifyPage" />
                        <mapping path="/sboard/removePage" />
                        <beans:ref bean="authInterceptor" />
                </interceptor>

        </interceptors>
```

5.3 게시물의 세부 기능 적용

5.3.1 인터셉터 URI mapping

■ 지금까지 작성된 게시물 관리에 AuthInterceptor를 적용하는 규칙은 다음과 같다.

- 로그인한 사용자 (포인트 점수)
  - 게시물의 등록
  - 게시물의 수정/삭제
  - 댓글 추가/수정/삭제
- 일반 사용자 (로그인 여부)
  - 게시물의 목록
  - 게시물의 조회
  - 댓글 목록

■ servlet-context.xml에 다음과 같이 설정한다.

```
// /WEB-INF/spring/appServlet/servlet-context.xml의 일부

        <beans:bean id="loginInterceptor" class="org.zerock.interceptor.LoginInterceptor"></beans:bean>

        <interceptors>
                <interceptor>
                        <mapping path="/user/loginPost" />
                        <beans:ref bean="loginInterceptor" />
                </interceptor>
                <interceptor>
                        <mapping path="/sboard/list" />
                        <mapping path="/sboard/register" />
                        <mapping path="/sboard/modifyPage" />
                        <mapping path="/sboard/removePage" />
                        <beans:ref bean="authInterceptor" />
                </interceptor>

        </interceptors>
```

5.3.2 각 JSP 별 로그인 처리

(1) 게시물의 등록 페이지

■ 게시물의 등록 화면에서는 로그인한 사용자의 아이디 정보가 '작성자'로 처리되도록 한다.

```
// /WEB-INF/views/sboard/register.jsp 의 일부

                <div class="form-group">
                        <label for="exampleInputEmail1">Writer</label>
                        <input type="text" name="writer"
                          class="form-control" value='${login.uid }' readonly>
                </div>
```

(2) 게시물의 조회 페이지

■ 게시물의 수정은 게시물을 작성한 사용자만이 수정할 수 있도록 한다.

```
// /WEB-INF/views/sboard/readPage.jsp 의 일부

        <div class="box-footer">

                <div>
                        <hr>
                </div>

                <ul class="mailbox-attachments clearfix uploadedList">
                </ul>
                <c:if test="${login.uid == boardVO.writer}">
                        <button type="submit" class="btn btn-warning" id="modifyBtn">Modify</button>
                        <button type="submit" class="btn btn-danger" id="removeBtn">REMOVE</button>
                </c:if>
                <button type="submit" class="btn btn-primary" id="goListBtn">GO
                        LIST</button>
```

```
        </div>
```

5.4 자동 로그인과 쿠키

■ 자동 로그인은 브라우저가 서버에 접속할 때 특정한 쿠키를 같이 전송하고, 이를 이용해서 로그인을 처리하는 방식이다.

5.4.1 쿠키를 이용하는 자동 로그인 방식

(1) LoginInterceptor에서의 쿠키 생성하기

```
// org.zerock.interceptor.LoginInterceptor.java의 일부

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

    HttpSession session = request.getSession();

    ModelMap modelMap = modelAndView.getModelMap();
    Object userVO = modelMap.get("userVO");

    if (userVO != null) {

      logger.info("new login success");
      session.setAttribute(LOGIN, userVO);

      if (request.getParameter("useCookie") != null) {

        logger.info("remember me.............");
        Cookie loginCookie = new Cookie("loginCookie", session.getId());
        loginCookie.setPath("/");
        loginCookie.setMaxAge(60 * 60 * 24 * 7);
        response.addCookie(loginCookie);
      }
      // response.sendRedirect("/");
      Object dest = session.getAttribute("dest");

      response.sendRedirect(dest != null ? (String) dest : "/");
    }
  }
```

(2) 브라우저 종료 후 다시 접속하기

■ 세션 쿠키는 브라우저가 종료될 때 같이 종료되었기 때문에, 매번 브라우저를 새로 실행하고 접속하면 새롭게 만들어집니다. 반면에 loginCookie의 경우 로그할 때 브라우저를 이용해서 보관된다.

5.4.2 자동 로그인의 구현

(1) 데이터베이스의 변경

```
-- 테이블 변경

alter table tbl_user add column
sessionkey varchar(50) not null default 'none';

alter table tbl_user
add column sessionlimit timestamp;
```

(2) 코드의 변경

- UserDAO의 변경: 로그인한 사용자의 sessionKey와 sessionLimit를 업데이트하는 기능과 loginCookie에 기록된 값으로 사용자의 정보를 조회하는 기능을 추가한다.

```
// org.zerock.persistence.UserDAO.java의 일부

public interface UserDAO {

          public UserVO login(LoginDTO dto) throws Exception;

          public void keepLogin(String uid, String sessionId, Date next);

          public UserVO checkUserWithSessionKey(String value);
}
```

```
// /resources/mappers/userMapper.xml의 일부

          <update id="keepLogin">
                    update tbl_user set sessionKey = #{sessionId}, sessionLimit = #{next} where
                    uid = #{uid}
          </update>

          <select id="checkUserWithSessionKey" resultType="UserVO">
                    select * from tbl_user where sessionKey = #{value} and sessionlimit > now()
          </select>
```

```
// org.zerock.persistence.UserDAOImpl.java의 일부

  @Override
  public void keepLogin(String uid, String sessionId, Date next) {

    Map<String, Object> paramMap = new HashMap<String, Object>();
    paramMap.put("uid", uid);
    paramMap.put("sessionId", sessionId);
    paramMap.put("next", next);

    session.update(namespace+".keepLogin", paramMap);

  }

  @Override
  public UserVO checkUserWithSessionKey(String value) {

    return session.selectOne(namespace +".checkUserWithSessionKey", value);
  }
```

- UserService의 변경: 로그인 정보를 유지하는 keepLogin과 과거에 접속한 사용자인지를 확인하는 기능을 작성한다.

```
// org.zerock.service.UserService.java의 일부

public interface UserService {

  public UserVO login(LoginDTO dto) throws Exception;

  public void keepLogin(String uid, String sessionId, Date next)throws Exception;

  public UserVO checkLoginBefore(String value);
}
```

```
// org.zerock.service.UserServiceImpl.java의 일부

  @Override
  public void keepLogin(String uid, String sessionId, Date next)
      throws Exception {

    dao.keepLogin(uid, sessionId, next);

  }

  @Override
  public UserVO checkLoginBefore(String value) {

    return dao.checkUserWithSessionKey(value);
  }
```

- UserController의 변경: 사용자가 '자동 로그인'을 선택한 경우 필요한 기능을 추가한다.

```
// org.zerock.controller.UserController.java의 일부

@RequestMapping(value = "/loginPost", method = RequestMethod.POST)
public void loginPOST(LoginDTO dto, HttpSession session, Model model) throws Exception {

    UserVO vo = service.login(dto);

    if (vo == null) {
      return;
    }

    model.addAttribute("userVO", vo);

    if (dto.isUseCookie()) {

      int amount = 60 * 60 * 24 * 7;

      Date sessionLimit = new Date(System.currentTimeMillis() + (1000 * amount));

      service.keepLogin(vo.getUid(), session.getId(), sessionLimit);
    }

}
```

■ AuthInterceptor의 변경: 현재 사용자의 세션에 login이 존재하지 않지만, 쿠키 중에서 loginCookie가 존재할 때 처리가 진행할 수 있도록 한다.

```
// org.zerock.interceptor.AuthInterceptor.java의 일부

public class AuthInterceptor extends HandlerInterceptorAdapter {

	private static final Logger logger = LoggerFactory.getLogger(AuthInterceptor.class);

	@Inject
	private UserService service;

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

		HttpSession session = request.getSession();

		if (session.getAttribute("login") == null) {

			logger.info("current user is not logined");

			saveDest(request);

			Cookie loginCookie = WebUtils.getCookie(request, "loginCookie");

			if (loginCookie != null) {

				UserVO userVO = service.checkLoginBefore(loginCookie.getValue());

				logger.info("USERVO: " + userVO);

				if (userVO != null) {
					session.setAttribute("login", userVO);
					return true;
				}

			}

			response.sendRedirect("/user/login");
			return false;
		}
		return true;
	}
```

(3) 자동 로그인 테스트

■ 위의 코드가 정상적으로 동작하는지 알아보기 위해서는 다음과 같이 진행한다.

- 로그인하지 않은 사용자가 로그인이 필요한 URI에 접근하는 경우: 로그인 페이지로 이동, 예) http://localhost:8080/sboard/list
- 로그인할 때 'Remember Me'를 체크하지 않은 상태에서 로그인 실행, 이후 로그인이 필요한 페이지로 이동: 정상적인 이동
- 브라우저를 종료하고, 2)번 테스트를 다시 진행: HttpSession이 변경됐으므로, 로그인 페이지로 이동
- 로그인 화면에서 'Remember Me'를 선택하고 로그인
- 브라우저를 종료하고 다시 실행한 후 로그인이 필요한 URI 접속: 정상적인 이동

5.4.3 로그아웃 처리

```
// org.zerock.controller.UserController.java의 일부

  @RequestMapping(value = "/logout", method = RequestMethod.GET)
  public String logout(HttpServletRequest request,
      HttpServletResponse response, HttpSession session) throws Exception {

    Object obj = session.getAttribute("login");

    if (obj != null) {
      UserVO vo = (UserVO) obj;

      session.removeAttribute("login");
      session.invalidate();

      Cookie loginCookie = WebUtils.getCookie(request, "loginCookie");

      if (loginCookie != null) {
        loginCookie.setPath("/");
        loginCookie.setMaxAge(0);
        response.addCookie(loginCookie);
        service.keepLogin(vo.getUid(), session.getId(), new Date());
      }
    }

    return "user/logout";
  }
```

```
// /WEB-INF/views/user/logout.jsp의 일부

<%@ page language="java" contentType="text/html; charset=UTF-8"
          pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
          <script type="text/javascript">
                    self.location = "/user/login";
          </script>
</body>
</html>
```

5.4.4 보완이 필요한 부분

- 예제에서는 자동 로그인이 어떤 식으로 구현이 가능한지를 설명하였지만, 실제로는 반드시 암호화에 대한 처리가 이뤄져야만 한다.

- 암호화에 대한 처리와 더불어 고민해 봐야 하는 것은 '스프링 시큐리티'를 적용하는 것이다. 만일 업무에 대한 권한의 구분이 복잡하다면, 인터셉터보다는 차리리 처음부터 스프링 시큐리티를 고려하는 것이 좋다.

5.5 실습 예제 요약

```
// 로그인 경우
1. http://localhost:8080/sboard/list

2. authInterceptor
          if (session.getAttribute("login") == null) {
                    Cookie loginCookie = WebUtils.getCookie(request, "loginCookie");
                    if (loginCookie != null) {
                              return true;
                         }
                      response.sendRedirect("/user/login")
            }
3. http://localhost:8080/user/login

4. /user/loginPost

5. loginInterceptor
          preHandle
                    if (session.getAttribute(LOGIN) != null) {
                              session.removeAttribute(LOGIN);
                    }
6. UserController
          @RequestMapping(value = "/loginPost",
          public void loginPOST() {
                    UserVO vo = service.login(dto);
                    if (vo == null) {
                              return;
                    }
                    model.addAttribute("userVO", vo);
                    // remember me를 선택하면
                     if (dto.isUseCookie()) {
                              // db에서 sessionid, sessionLimit(1주일)을 저장한다.
                              service.keepLogin(vo.getUid(), session.getId(), sessionLimit);
                     }
          }

7. /views/user/loginPost.jsp
          self.location = "/sboard/list";

8. loginInterceptor
          postHandle
                    Object userVO = modelMap.get("userVO");
                    if (userVO != null) {
                              session.setAttribute(LOGIN, userVO);
                              // "remember me" 체크하면 쿠키 생성한다.
                              Cookie loginCookie = new Cookie("loginCookie", session.getId());
                              response.addCookie(loginCookie);
                    }
                    Object dest = session.getAttribute("dest")   // "http://localhost:8080/sboard/list"
                    response.sendRedirect(dest != null ? (String) dest : "/sboard/list");

9. authInterceptor
          preHandle
                    if (session.getAttribute("login") == null) {
                              if (loginCookie != null) {
                                        // 통과
                                        session.setAttribute("login", userVO);
                              }
                              response.sendRedirect("/user/login");
                    }
```

```
10. SearchBoardController
            @RequestMapping(value = "/list", method = RequestMethod.GET)
            public void listPage {
            }

11. /views/sboard/list.jsp


// 로그아웃 경우
```