

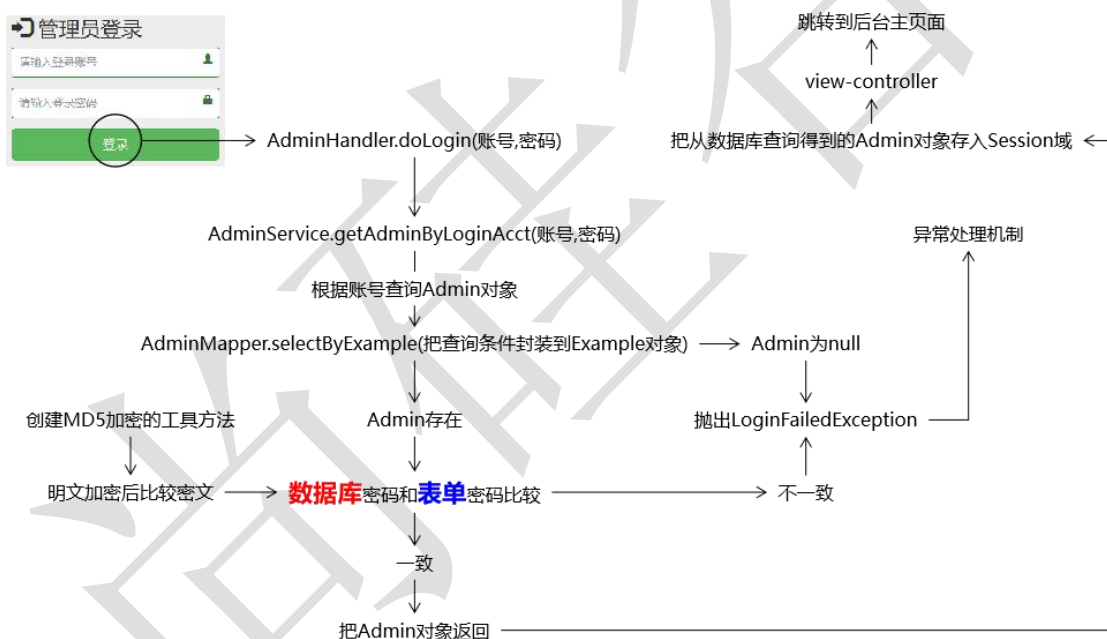
尚筹网

[04-后台管理系统-管理员登录]

1 目标

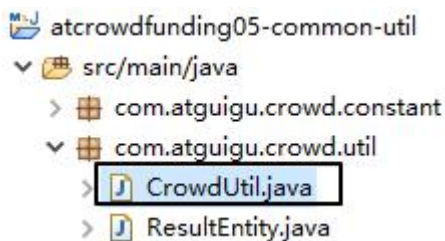
识别操作系统的人的身份，控制他的行为。

2 思路



3 代码

3.1 创建工具方法执行 MD5 加密



```
/**
 * 对明文字符串进行 MD5 加密
 * @param source 传入的明文字符串
 * @return 加密结果
 */
public static String md5(String source) {

    // 1.判断 source 是否有效
    if(source == null || source.length() == 0) {

        // 2.如果不是有效的字符串抛出异常
        throw new RuntimeException(CrowdConstant.MESSAGE_STRING_INVALIDATE);
    }

    try {
        // 3.获取 MessageDigest 对象
        String algorithm = "md5";

        MessageDigest messageDigest = MessageDigest.getInstance(algorithm);

        // 4.获取明文字符串对应的字节数组
        byte[] input = source.getBytes();

        // 5.执行加密
        byte[] output = messageDigest.digest(input);

        // 6.创建 BigInteger 对象
        int signum = 1;
        BigInteger bigInteger = new BigInteger(signum, output);

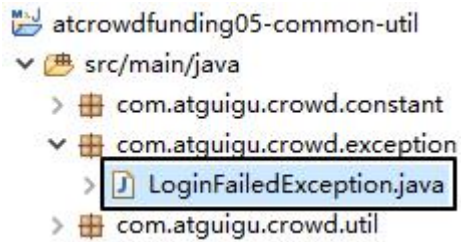
        // 7.按照 16 进制将 bigInteger 的值转换为字符串
        int radix = 16;
        String encoded = bigInteger.toString(radix).toUpperCase();

        return encoded;

    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }

    return null;
}
```

3.2 创建登录失败异常



```
/**
 * 登录失败后抛出的异常
 * @author Lenovo
 *
 */
public class LoginFailedException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public LoginFailedException() {
        super();
    }

    public LoginFailedException(String message, Throwable cause, boolean enableSuppression,
        boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }

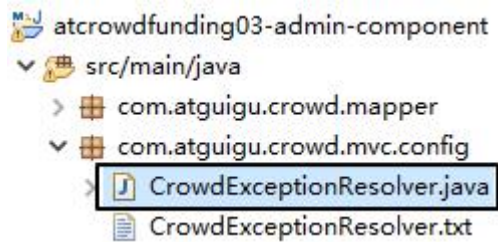
    public LoginFailedException(String message, Throwable cause) {
        super(message, cause);
    }

    public LoginFailedException(String message) {
        super(message);
    }

    public LoginFailedException(Throwable cause) {
        super(cause);
    }

}
```

3.3 在异常处理器类中增加登录失败异常的处理

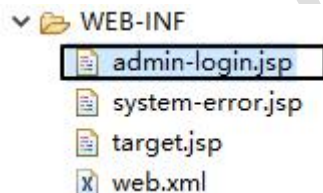


```
@ExceptionHandler(value = LoginFailedException.class)
public ModelAndView resolveLoginFailedException(
    LoginFailedException exception,
    HttpServletRequest request,
    HttpServletResponse response
) throws IOException {

    String viewName = "admin-login";

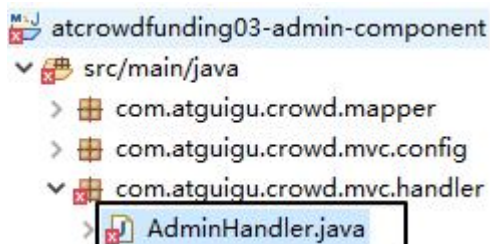
    return commonResolve(viewName, exception, request, response);
}
```

3.4 在登录页面显示异常消息



```
<h2 class="form-signin-heading">
    <i class="glyphicon glyphicon-log-in"></i> 管理员登录
</h2>
<p>${requestScope.exception.message}</p>
```

3.5 handler 方法



```
@RequestMapping("/admin/do/login.html")
```

```
public String doLogin(  
    @RequestParam("loginAcct") String loginAcct,  
    @RequestParam("userPswd") String userPswd,  
    HttpSession session  
) {  
  
    // 调用 Service 方法执行登录检查  
    // 这个方法如果能够返回 admin 对象说明登录成功，如果账号、密码不正确则会抛出异常  
    Admin admin = adminService.getAdminByLoginAcct(loginAcct, userPswd);  
  
    // 将登录成功返回的 admin 对象存入 Session 域  
    session.setAttribute(CrowdConstant.ATTR_NAME_LOGIN_ADMIN, admin);  
  
    return "admin-main";  
}
```

3.6 service 方法



```
@Override  
public Admin getAdminByLoginAcct(String loginAcct, String userPswd) {  
  
    // 1.根据登录账号查询 Admin 对象  
    // ①创建 AdminExample 对象  
    AdminExample adminExample = new AdminExample();  
  
    // ②创建 Criteria 对象  
    Criteria criteria = adminExample.createCriteria();  
  
    // ③在 Criteria 对象中封装查询条件  
    criteria.andLoginAcctEqualTo(loginAcct);  
  
    // ④调用 AdminMapper 的方法执行查询
```

```
List<Admin> list = adminMapper.selectByExample(adminExample);

// 2.判断 Admin 对象是否为 null
if(list == null || list.size() == 0) {
    throw new LoginFailedException(CrowdConstant.MESSAGE_LOGIN_FAILED);
}

if(list.size() > 1) {
    throw
new RuntimeException(CrowdConstant.MESSAGE_SYSTEM_ERROR_LOGIN_NOT_UNIQUE);
}

Admin admin = list.get(0);

// 3.如果 Admin 对象为 null 则抛出异常
if(admin == null) {
    throw new LoginFailedException(CrowdConstant.MESSAGE_LOGIN_FAILED);
}

// 4.如果 Admin 对象不为 null 则将数据库密码从 Admin 对象中取出
String userPswdDB = admin.getUserPswd();

// 5.将表单提交的明文密码进行加密
String userPswdForm = CrowdUtil.md5(userPswd);

// 6.对密码进行比较
if(!Objects.equals(userPswdDB, userPswdForm)) {
    // 7.如果比较结果是不一致则抛出异常
    throw new LoginFailedException(CrowdConstant.MESSAGE_LOGIN_FAILED);
}

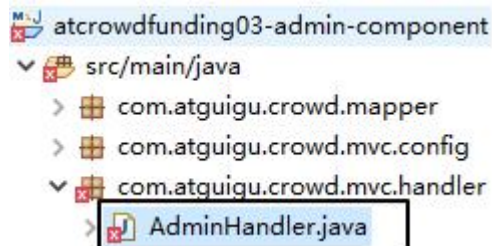
// 8.如果一致则返回 Admin 对象
return admin;
}
```

3.7 补充

3.7.1 前往后台主页面的方式调整

为了避免跳转到后台主页面再刷新浏览器导致重复提交登录表单,重定向到目标页面。

所以 handler 方法需要做相应修改

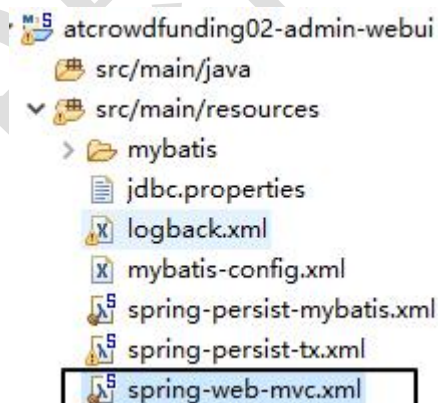


```
@RequestMapping("/admin/do/login.html")
public String doLogin(
    @RequestParam("loginAcct") String loginAcct,
    @RequestParam("userPswd") String userPswd,
    HttpSession session
){
    // 调用 Service 方法执行登录检查
    // 这个方法如果能够返回 admin 对象说明登录成功，如果账号、密码不正确则会抛出异常
    Admin admin = adminService.getAdminByLoginAcct(loginAcct, userPswd);

    // 将登录成功返回的 admin 对象存入 Session 域
    session.setAttribute(CrowdConstant.ATTR_NAME_LOGIN_ADMIN, admin);

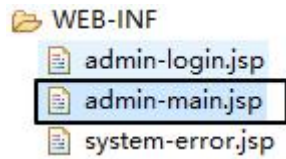
    return "redirect:/admin/to/main/page.html";
}
```

需要给目标地址配置 view-controller

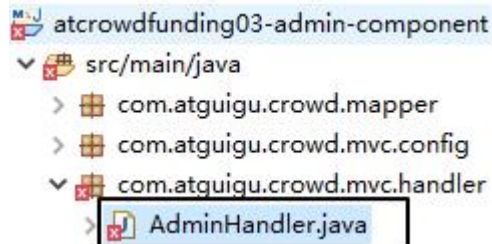


```
<mvc:view-controller path="/admin/to/main/page.html" view-name="admin-main"/>
```

3.7.2 退出登录



```
<li><a href="admin/do/logout.html"><i class="glyphicon glyphicon-off"></i> 退出系统</a></li>
```



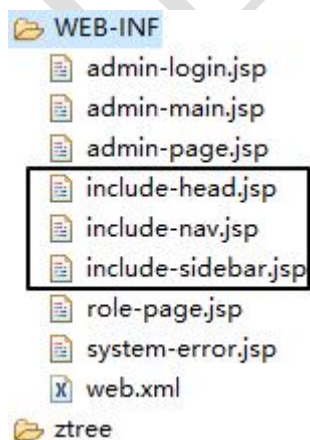
```
@RequestMapping("/admin/do/logout.html")
public String doLogout(HttpSession session) {

    // 强制 Session 失效
    session.invalidate();

    return "redirect:/admin/to/login/page.html";
}
```

4 抽取后台主页面公共部分

4.1 创建公共部分 JSP



4.2 抽取后效果

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="zh-CN">
<%@include file="/WEB-INF/include-head.jsp" %>

<body>

    <%@ include file="/WEB-INF/include-nav.jsp" %>
    <div class="container-fluid">
        <div class="row">
            <%@ include file="/WEB-INF/include-sidebar.jsp" %>
            <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
                <h1 class="page-header">控制面板</h1>

                <div class="row placeholders">
                    .....
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

4.3 创建 JSP 模板

模板内容是：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="zh-CN">
<%@include file="/WEB-INF/include-head.jsp" %>
<body>

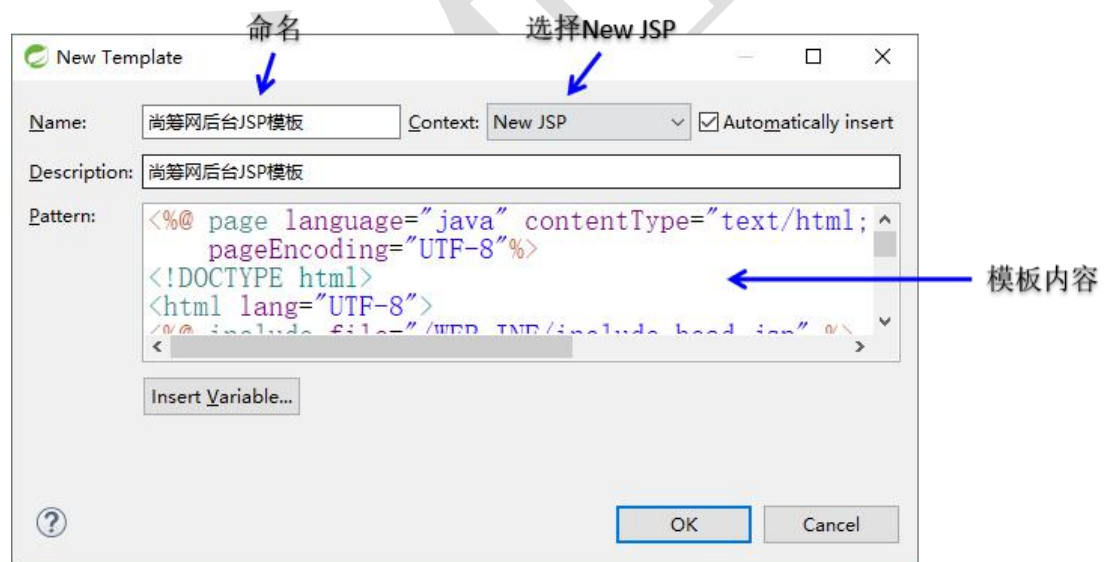
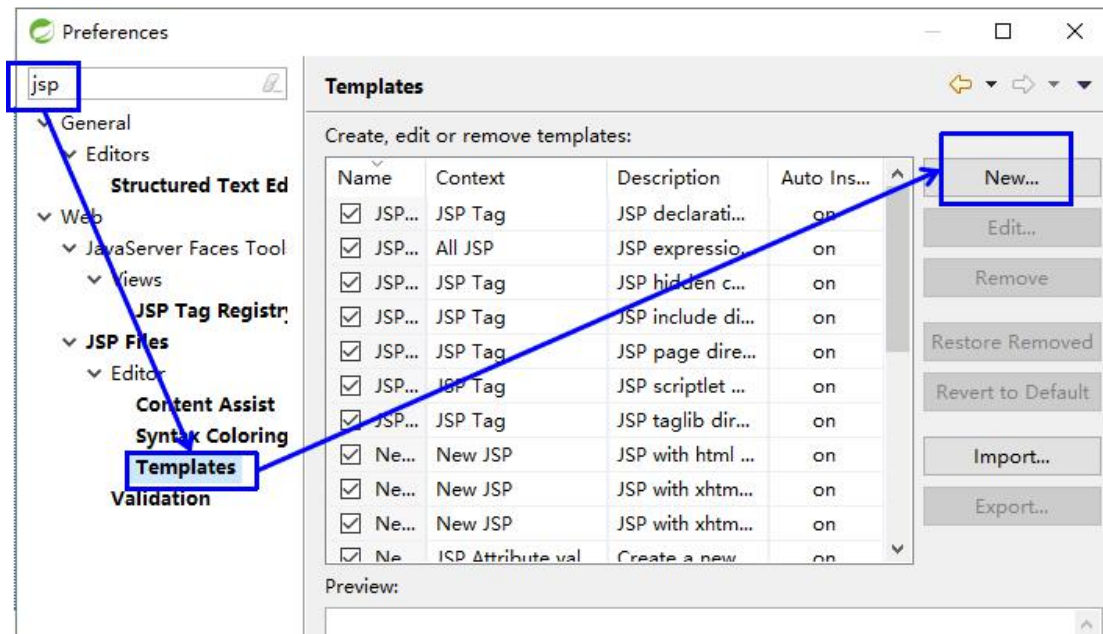
    <%@ include file="/WEB-INF/include-nav.jsp" %>
    <div class="container-fluid">
        <div class="row">
            <%@ include file="/WEB-INF/include-sidebar.jsp" %>
            <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">

                </div>
        </div>
    </div>
```

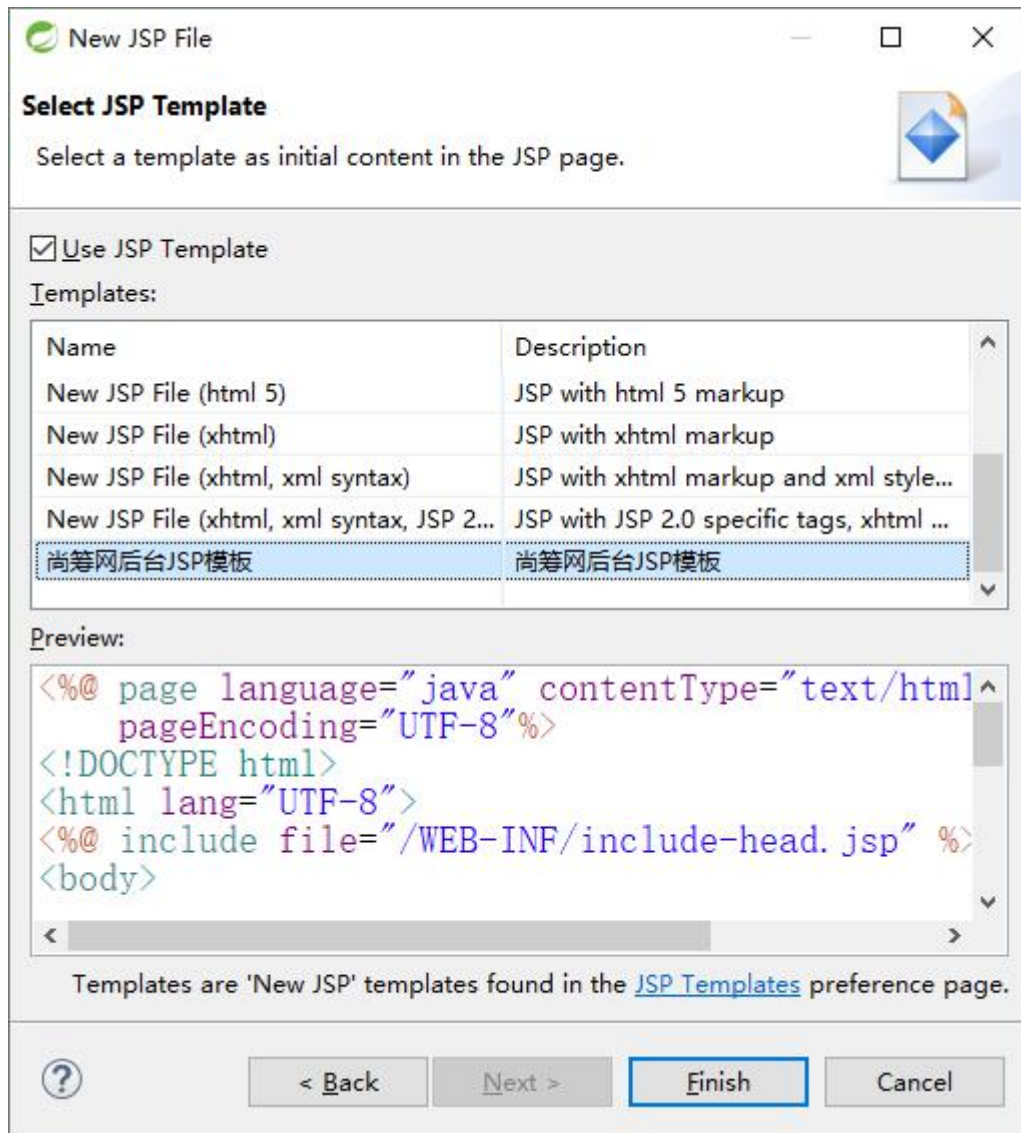
```

</div>
</div>
</body>
</html>

```



模板创建完成后，第一次创建 JSP 页面时需要选择一下我们创建的模板，以后就默认使用这个模板。



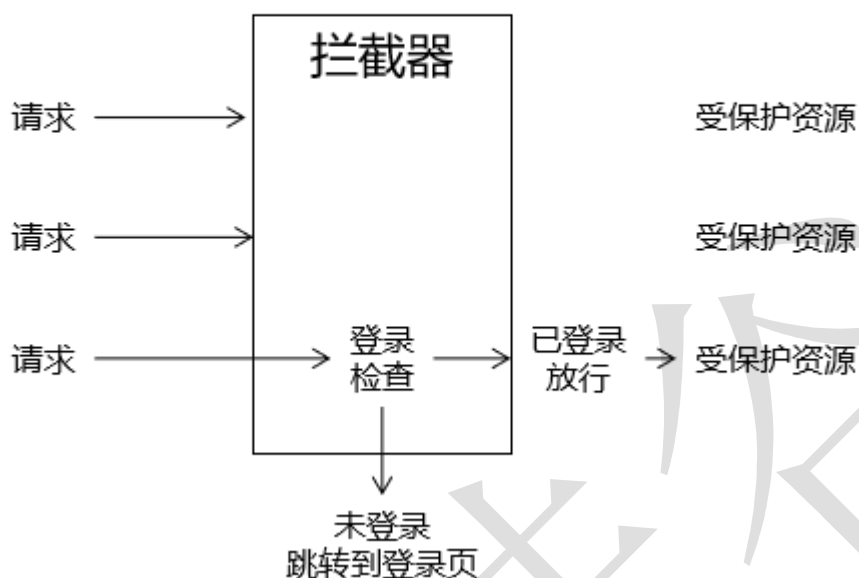
5 登录状态检查

5.1 目标

将部分资源保护起来，让没有登录的请求不能访问。

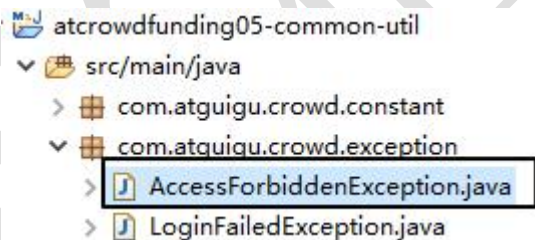
5.2 思路

请求 → 公共资源

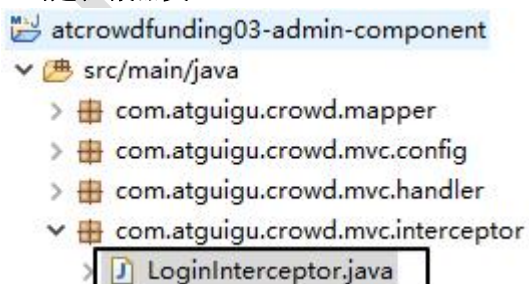


5.3 代码

5.3.1 创建自定义异常



5.3.2 创建拦截器类



```

public class LoginInterceptor extends HandlerInterceptorAdapter {

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
    
```

```

Object handler)
    throws Exception {

    // 1.通过 request 对象获取 Session 对象
    HttpSession session = request.getSession();

    // 2.尝试从 Session 域中获取 Admin 对象
    Admin admin = session.getAttribute(CrowdConstant.ATTR_NAME_LOGIN_ADMIN);

    // 3.判断 admin 对象是否为空
    if(admin == null) {

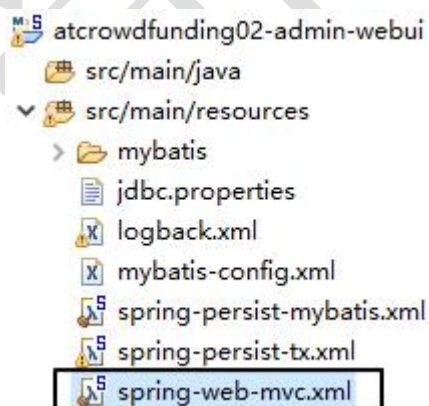
        // 4.抛出异常
        throw new AccessForbiddenException(CrowdConstant.MESSAGE_ACCESS_FORBIDDEN);

    }

    // 5.如果 Admin 对象不为 null，则返回 true 放行
    return true;

}
}
    
```

5.3.3 注册拦截器类



```

<!-- 注册拦截器 -->
<mvc:interceptors>
    <mvc:interceptor>
        <!-- mvc:mapping 配置要拦截的资源 -->
        <!-- /*对应一层路径，比如：/aaa -->
    
```

```

<!-- /**对应多层路径，比如： /aaa/bbb 或/aaa/bbb/ccc 或/aaa/bbb/ccc/ddd -->
<mvc:mapping path="/**"/>

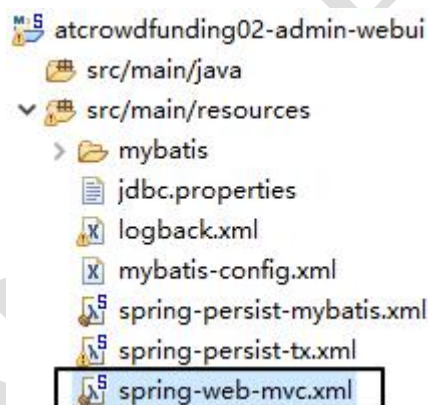
<!-- mvc:exclude-mapping 配置不拦截的资源 -->
<mvc:exclude-mapping path="/admin/to/login/page.html"/>
<mvc:exclude-mapping path="/admin/do/login.html"/>
<mvc:exclude-mapping path="/admin/do/logout.html"/>

<!-- 配置拦截器类 -->
<bean class="com.atguigu.crowd.mvc.interceptor.LoginInterceptor"/>
</mvc:interceptor>
</mvc:interceptors>

```

5.4 完善配套的异常映射

5.4.1 基于 XML 的异常映射



```

<!-- 配置基于 XML 的异常映射 -->
<bean
                                id="simpleMappingExceptionHandler"
class="org.springframework.web.servlet.handler.SimpleMappingExceptionHandler">
    <!-- 配置异常类型和具体视图页面的对应关系 -->
    <property name="exceptionMappings">
        <props>
            <!-- key 属性指定异常全类名 -->
            <!-- 标签体中写对应的视图（这个值要拼前后缀得到具体路径） -->
            <prop key="java.lang.Exception">system-error</prop>
            <prop
key="com.atguigu.crowd.exception.AccessForbiddenException">admin-login</prop>
        </props>
    </property>
</bean>

```

5.4.2 基于注解的异常映射



```
atcrowdfunding03-admin-component
├── src/main/java
│   ├── com.atguigu.crowd.mapper
│   └── com.atguigu.crowd.mvc.config
│       ├── CrowdExceptionHandler.java
│       └── CrowdExceptionHandler.txt
```

```
@ExceptionHandler(value = AccessForbiddenException.class)
public ModelAndView resolveAccessForbiddenException(
    AccessForbiddenException exception,
    HttpServletRequest request,
    HttpServletResponse response
) throws IOException {

    String viewName = "admin-login";

    return commonResolve(viewName, exception, request, response);
}
```