1.1案例需求

1.1.1需求的概述

在CRM的系统中，有用户登录的功能，如果访问者知道了后台的访问页面的路径，有可能没 有进行登录直接就进入到CRM系统中。那么对于这类的账号需要进行拦截。

首先需要有登录的功能的实现，如果对于没有登录的用户直接访问后台的Action,可以实施拦 截。

1.2 Struts2 的拦截器

1.2.1 拦截器概述

拦截器，在AOP (Aspect-OrientedProgramming)中用于在某个方法或字段被访问之前，进行拦 截然后在之前或之后加入某些操作。拦截是AOP的一种实现策略。

在Webwork的中文文档的解释为——拦截器是动态拦截Action调用的对象。它提供了一种机制 可以使开发者可以定义在一个action执行的前后执行的代码，也可以在一个action执行前阻止其执 行。同时也是提供了一种可以提取action中可重用的部分的方式。

谈到拦截器，还有一个词大家应该知道 拦截器链(Interceptor Chain,在Struts 2中称为拦 截器栈Interceptor Stack)。拦截器链就是将拦截器按一定的顺序联结成一条链。在访问被拦截的方 法或字段时，拦截器链中的拦截器就会按其之前定义的顺序被调用。

1.2.1.1 什么是拦截器

- Interceptor：拦截器，起到拦截Action的作用。
  - Filter：过滤器，过滤从客户端向服务器发送的请求。
  - Interceptor：拦截器，拦截是客户端对Action的访问。更细粒度化的拦截。（拦截Action中的具体的方法）。
- Struts2框架核心的功能都是依赖拦截器实现。

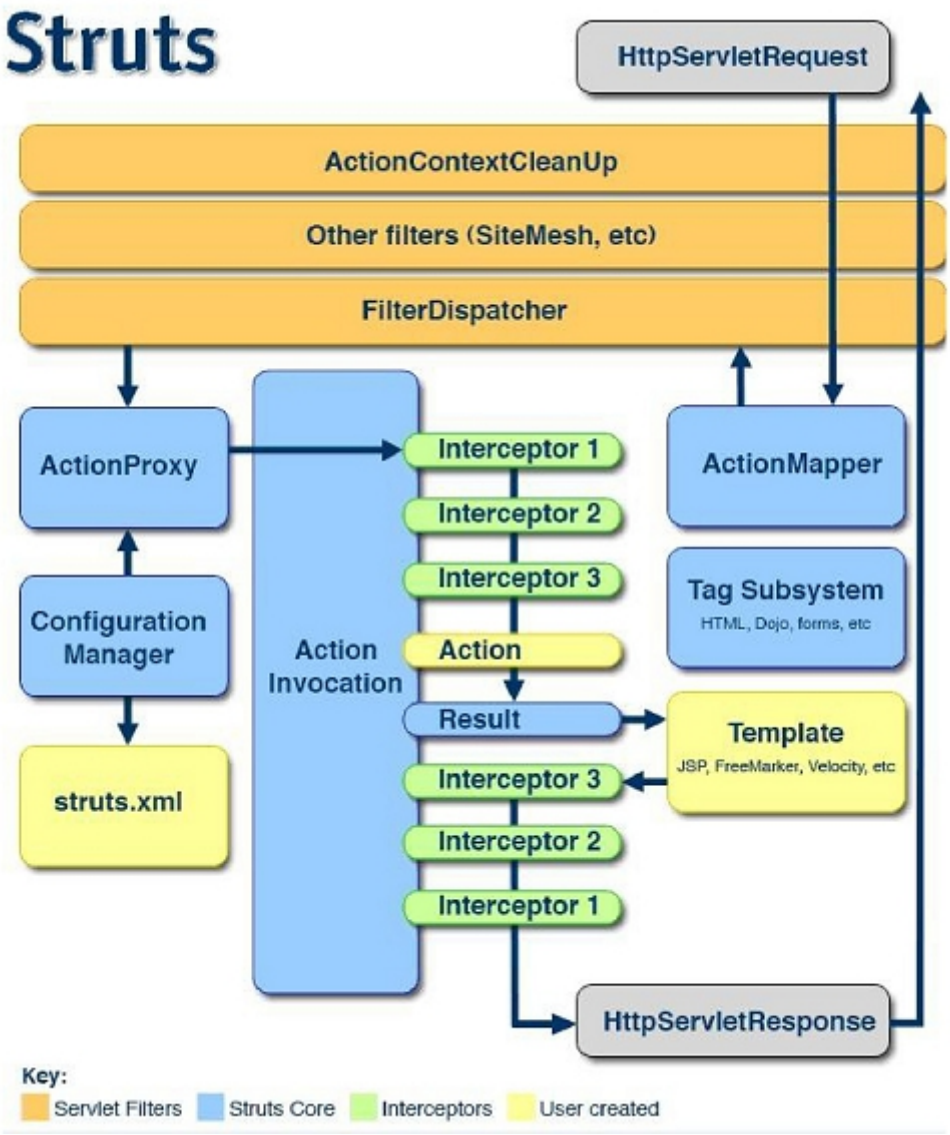
1.2.1.2 拦截器的实现原理

大部分时候，拦截器方法都是通过代理的方式来调用的。Struts 2的拦截器实现相对简单。当请 求到达Struts 2的ServletDispatcher时，Struts 2会查找配置文件，并根据其配置实例化相对的拦截器 对象，然后串成一个列表，最后一个一个地调用列表中的拦截器。

Struts2拦截器是可插拔的，拦截器是AOP的一种实现。Struts2拦截器栈就是将拦截器按一定的 顺序联结成一条链。在访问被拦截的方法或字段时，Struts2拦截器链中的拦截器就会按其之前定义 的顺序被调用。

1.2.1.3 Struts2 的执行流程

客户端向服务器发送一个Action的请求，执行核心过滤器（doFilter）方法。在这个方法中，调用executeAction()方法，在这个方法内部调用dispatcher.serviceAction();在这个方法内部创建一个Action代理，最终执行的是Action代理中的execute(),在代理中执行的execute方法中调用ActionInvocation的invoke方法。在这个方法内部递归执行一组拦截器（完成部分功能），如果没有下一个拦截器，就会执行目标Action，根据Action的返回的结果进行页面跳转。



1.2.2 拦截器入门

在程序开发过程中，如果需要开发自己的拦截器类，就需要直接或间接的实现 com.opensymphony.xwork2.interceptor.Interceptor 接口。其定义的代码如下：

```
public interface Interceptor extends Serializable (
    void init();
    void destroy();
    String intercept(Actioninvocation invocation) throws Exception;
}
```

该接口提供了三个方法，其具体介绍如下。

- void init：该方法在拦截器被创建后会立即被调用，它在拦截器的生命周期内只被调用一 次.可以在该方法中对相关资源进行必要的初始化。

- void destroy()：该方法与init方法相对应，在拦截器实例被销毁之前，将调用该方法来释放 和拦截器相关的资源。它在拦截器的生命周期内，也只被调用一次。
- String intercept(ActionInvocation invocation) throws Exception：该方法是拦截器的核心方法，用来添加真正执行拦截工作的代码，实现具体的拦截操作。它返回一个字符串作为逻辑视 图，系统根据返回的字符串跳转到对应的视图资源。每拦截一个动作请求，该方法就会被 调用一次。该方法的Actioninvocation参数包含了被拦截的Action的引用，可以通过该参 数的invoke。方法，将控制权转给下一个拦截器或者转给Action的execute()方法。

如果需要自定义拦截器，只需要实现Interceptor接口的三个方法即可。然而在实际开发过程中， 除了实现Interceptor接口可以自定义拦截器外，更常用的一种方式是继承抽象拦截器类AbstractInterceptero该类实现了 Interceptor接口，并且提供了 init()方法和destroy()方法的空实现。使用时，可以直接继承该抽象类，而不用实现那些不必要的方法。拦截器类Abstractinterceptor中定 义的方法如下所示：

```
public abstract class AbstractInterceptor implements Interceptor (
    public void init() {}
    public void destroy() {}
    public abstract String intercept(Actioninvocation invocation) throws
Exception;
)
```

从上述代码中可以看出，Abstractinterceptor类已经实现了 Interceptor接口的所有方法，一般情况下，只需继承AbstractInterceptor类，实现interceptor。方法就可以创建自定义拦截器。

只有当自定义的拦截器需要打开系统资源时，才需要覆盖Abstractinterceptor类的init()方法和destroy()方法。与实现Interceptor接口相比，继承Abstractinterceptor类的方法更为简单。

1.2.2.1 搭建 Struts2 的环境

- 编写 Action

```
/**
* @Title: ActionDemo1.java
* @Package com.admiral.web.action
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.web.action;

import com.opensymphony.xwork2.ActionSupport;

public class ActionDemo1 extends ActionSupport {

    @Override
    public String execute() throws Exception {
        System.out.println("ActionDemo1 执行了....");
        return super.execute();
    }
}
```

- 配置 struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <constant name="struts.ognl.allowStaticMethodAccess" value="true">
</constant>
    <include file="com/admiral/struts2/valuestack/struts_demo1.xml"></include>

    <package name="demo1" extends="struts-default" namespace="/">
        <action name="actionDemo1" class="com.admiral.web.action.ActionDemo1">
            <result>/demo1/demo1.jsp</result>
        </action>
    </package>

</struts>
```

- 编写 demo1/demo1.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%
        System.out.println("demo1.jsp 执行了....");
    %>
</body>
</html>
```

1.2.2.2 编写拦截器类

- InterceptorDemo1.java

```
/**
* @Title: InterceptorDemo1.java
* @Package com.admiral.web.interceptor
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.web.interceptor;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;

public class InterceptorDemo1 extends AbstractInterceptor {

    @Override
```

```
    public String intercept(ActionInvocation invocation) throws Exception {
        System.out.println("InterceptorDemo1 执行了....");
        String invoke = invocation.invoke();
        System.out.println("InterceptorDemo1 执行结束了....");
        return invoke;
    }

}
```

- InterceptorDemo2.java

```
/**
* @Title: InterceptorDemo1.java
* @Package com.admiral.web.interceptor
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.web.interceptor;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;

public class InterceptorDemo2 extends AbstractInterceptor {

    @Override
    public String intercept(ActionInvocation invocation) throws Exception {
        System.out.println("InterceptorDemo2 执行了....");
        String invoke = invocation.invoke();
        System.out.println("InterceptorDemo2 执行结束了....");
        return invoke;
    }
}
```

1.2.2.3 对拦截器进行配置

1. 拦截器

要想让拦截器起作用，首先要对它进行配置。拦截器的配置是在struts.xml文件中完成的，它通 常以<interceptor>标签开头，以</interceptor> 标签结束。定义拦截器的语法格式如下：

```
<interceptor name="interceptorNameH" class="interceptorClass">
    <param name="paramName">paramValue</param>
</interceptor>
```

上述语法格式中，name属性用来指定拦截器的名称，class属性用于指定拦截器的实现类。有时，在定义拦截器时需要传入参数，这时需要使用<param>标签，其中name属性用来指定参数的名称，paramValue表示参数的值。

2. 拦截器栈
3. 在实际开发中，经常需要在Action执行前同时执行多个拦截动作，如：用户登录检查、登 录日志记录以及权限检查等，这时，可以把多个拦截器组成一个拦截器栈。在使用时，可以将栈内 的多个拦截器当成一个整体来引用。当拦截器栈被附加到一个Action 上时，在执行Action之前必须 先执行拦截器栈中的每一个拦截器。

定义拦截器栈使用<interceptors>元素和<mterceptor-stack>子元素，当配置多个拦截器时，需要使 用<interceptor-ref>元素来指定多个拦截器，配置语法如下：

```
<interceptors>
    <interceptor-stack name="interceptorStackName">
        <interceptor-ref name="interceptorName" />
        ....
    </interceptor-stack>
</interceptors>
```

在上述语法中，interceptorStackName值表示配置的拦截器栈的名称；interceptorName值表示拦截器的名称。除此之外，在一个拦截器栈中还可以包含另一个拦截器栈，示例代码如下：

```
<package name="default" namespace="/" extends="struts -default">

    <!-声明拦截器-->
    <interceptors>

        <interceptor name="interceptor1" class="interceptorClass"/>
        <interceptor name="interceptor2" class="interceptorClass"/>


        <!-定义一个拦截器栈myStack,该拦截器栈中包含两个拦截器和一个拦截器栈
        <interceptor-stack name=ninyStackn>

            <interceptor-ref name="defaultStack" / >
            <interceptor-ref name="interceptor1" / >
            <interceptor-ref name="interceptor2" />

        </interceptor-stack>
    </interceptors>
</package〉
```

在上述代码中，定义的拦截器栈是myStack,在myStack栈中，除了引用了两个自定义的拦截 器interceptor 1和mterceptor2外，还引用了一个内置拦截器栈defaultstack,这个拦截器是必须要引入的。

- 配置我们的拦截器方式一:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
```

```
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <constant name="struts.ognl.allowStaticMethodAccess" value="true">
</constant>
    <include file="com/admiral/struts2/valuestack/struts_demo1.xml"></include>

    <package name="demo1" extends="struts-default" namespace="/">
        <!-- 定义拦截器 -->
        <interceptors>
            <interceptor name="interceptorDemo1"
class="com.admiral.web.interceptor.InterceptorDemo1" />
            <interceptor name="interceptorDemo2"
class="com.admiral.web.interceptor.InterceptorDemo2" />
        </interceptors>

        <action name="actionDemo1" class="com.admiral.web.action.ActionDemo1">
            <result>/demo1/demo1.jsp</result>

            <!-- 引入拦截器(一旦引入自定义拦截器,默认的拦截器就不执行了) -->
            <interceptor-ref name="defaultStack" />
            <interceptor-ref name="interceptorDemo1" />
            <interceptor-ref name="interceptorDemo2" />
        </action>
    </package>

</struts>
```

```
信息: Starting ProtocolHandler ["ajp-bio-8009"]
九月 29, 2020 10:34:24 下午 org.apache.catalina.startup.Catalina start
信息: Server startup in 4051 ms
InterceptorDemo1 执行了....
InterceptorDemo2 执行了....
ActionDemo1 执行了....
demo1.jsp 执行了....
InterceptorDemo2 执行结束了....
InterceptorDemo1 执行结束了....
```

- 配置拦截器方式二:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <constant name="struts.ognl.allowStaticMethodAccess" value="true">
</constant>
    <include file="com/admiral/struts2/valuestack/struts_demo1.xml"></include>

    <package name="demo1" extends="struts-default" namespace="/">
        <!-- 定义拦截器 -->
        <interceptors>
```

```
            <interceptor name="interceptorDemo1"
class="com.admiral.web.interceptor.InterceptorDemo1" />
            <interceptor name="interceptorDemo2"
class="com.admiral.web.interceptor.InterceptorDemo2" />

            <!-- 自定义拦截器栈 -->
            <interceptor-stack name="myStack">
                <interceptor-ref name="defaultStack" />
                <interceptor-ref name="interceptorDemo1" />
                <interceptor-ref name="interceptorDemo2" />
            </interceptor-stack>
        </interceptors>

        <action name="actionDemo1" class="com.admiral.web.action.ActionDemo1">
            <result>/demo1/demo1.jsp</result>

            <!-- 引入自定义拦截器栈 -->
            <interceptor-ref name="myStack"></interceptor-ref>
        </action>
    </package>

</struts>
```

```
信息: Starting ProtocolHandler ["ajp-bio-8009"]
九月 29, 2020 10:36:46 下午 org.apache.catalina.startup.Catalina start
信息: Server startup in 4021 ms
InterceptorDemo1 执行了....
InterceptorDemo2 执行了....
ActionDemo1 执行了....
demo1.jsp 执行了....
InterceptorDemo2 执行结束了....
InterceptorDemo1 执行结束了....
```

1.3 CRM 的权限拦截器

1.3.1 实现用户登录的功能

1.3.1.1 创建表和实体

```
CREATE TABLE `sys_user` (
  `user_id` bigint(32) NOT NULL AUTO_INCREMENT COMMENT '用户id',
  `user_code` varchar(32) NOT NULL COMMENT '用户账号',
  `user_name` varchar(64) NOT NULL COMMENT '用户名称',
  `user_password` varchar(32) NOT NULL COMMENT '用户密码',
  `user_state` char(1) NOT NULL COMMENT '1:正常,0:暂停',
  PRIMARY KEY (`user_id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

```
/**
* @Title: User.java
* @Package com.admiral.domain
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
```

```
*/
package com.admiral.domain;

/**
 * CREATE TABLE `sys_user` ( `user_id` bigint(32) NOT NULL AUTO_INCREMENT
 * COMMENT '用户id', `user_code` varchar(32) NOT NULL COMMENT '用户账号',
`user_name`
 * varchar(64) NOT NULL COMMENT '用户名称', `user_password` varchar(32) NOT NULL
 * COMMENT '用户密码', `user_state` char(1) NOT NULL COMMENT '1:正常,0:暂停',
PRIMARY
 * KEY (`user_id`) ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
 */
public class User {
    private Long user_id;
    private String user_code;
    private String user_name;
    private String user_password;
    private String user_state;

    public User() {
        super();
    }

    public User(Long user_id, String user_code, String user_name, String
user_password, String user_state) {
        super();
        this.user_id = user_id;
        this.user_code = user_code;
        this.user_name = user_name;
        this.user_password = user_password;
        this.user_state = user_state;
    }

    @Override
    public String toString() {
        return "User [user_id=" + user_id + ", user_code=" + user_code + ",
user_name=" + user_name + ", user_password="
                + user_password + ", user_state=" + user_state + "]";
    }

}
```

User.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="com.admiral.domain.User" table="sys_user">
        <id name="user_id" column="user_id">
            <generator class="native" />
        </id>
```

```
        <property name="user_code" column="user_code" />
        <property name="user_name" column="user_name" />
        <property name="user_password" column="user_password" />
        <property name="user_state" column="user_state" />
    </class>
</hibernate-mapping>
```

1.3.1.2 提交数据到 Action

- 修改登陆页面

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3c.org/TR/1999/REC-html401-19991224/frameset.dtd">
<HTML xmlns="http://www.w3.org/1999/xhtml">
<HEAD>
<META http-equiv=Content-Type content="text/html; charset=utf-8">
<STYLE type=text/css>
BODY {
    FONT-SIZE: 12px; COLOR: #ffffff; FONT-FAMILY: 宋体
}
TD {
    FONT-SIZE: 12px; COLOR: #ffffff; FONT-FAMILY: 宋体
}
</STYLE>

<META content="MSHTML 6.00.6000.16809" name=GENERATOR></HEAD>
<BODY>
<FORM id=form1 name=form1 action="${ pageContext.request.contextPath }/user_login.action" method=post>

<DIV id=UpdatePanel1>
<DIV id=div1
style="LEFT: 0px; POSITION: absolute; TOP: 0px; BACKGROUND-COLOR: #0066ff"></DIV>
<DIV id=div2
style="LEFT: 0px; POSITION: absolute; TOP: 0px; BACKGROUND-COLOR: #0066ff"></DIV>


<DIV>   </DIV>
<DIV>
<TABLE cellSpacing=0 cellPadding=0 width=900 align=center border=0>
  <TBODY>
  <TR>
    <TD style="HEIGHT: 105px"><IMG src="images/login_1.gif"
  border=0></TD></TR>
  <TR>
    <TD background=images/login_2.jpg height=300>
      <TABLE height=300 cellPadding=0 width=900 border=0>
        <TBODY>
        <TR>
          <TD colSpan=2 height=35></TD></TR>
        <TR>
          <TD width=360></TD>
          <TD>
            <TABLE cellSpacing=0 cellPadding=2 border=0>
              <TBODY>
              <TR>
                <TD style="HEIGHT: 28px" width=80>登 录 名：</TD>
                <TD style="HEIGHT: 28px" width=150><INPUT id=txtName
                  style="WIDTH: 130px" name="user_code"></TD>
                <TD style="HEIGHT: 28px" width=370><SPAN
                  id=RequiredFieldValidator3
                  style="FONT-WEIGHT: bold; VISIBILITY: hidden; COLOR: white">请输入登录名</SPAN></TD></TR>
              <TR>
                <TD style="HEIGHT: 28px">登录密码：</TD>
                <TD style="HEIGHT: 28px"><INPUT id=txtPwd style="WIDTH: 130px"
                  type=password name="user_password"></TD>
                <TD style="HEIGHT: 28px"><SPAN id=RequiredFieldValidator4
                  style="FONT-WEIGHT: bold; VISIBILITY: hidden; COLOR: white">请输入密码</SPAN></TD></TR>
              <TR>
                <TD style="HEIGHT: 28px">验证码：</TD>
                <TD style="HEIGHT: 28px"><INPUT id=txtcode
                  style="WIDTH: 130px" name=txtcode></TD>
                <TD style="HEIGHT: 28px"> </TD></TR>
              <TR>
                <TD style="HEIGHT: 18px"></TD>
                <TD style="HEIGHT: 18px"></TD>
                <TD style="HEIGHT: 18px"></TD></TR>
              <TR>
                <TD></TD>
                <TD><INPUT id=btn
                  style="BORDER-TOP-WIDTH: 0px; BORDER-LEFT-WIDTH: 0px; BORDER-BOTTOM-WIDTH: 0px; BORDER-RIGHT-WIDTH: 0px"
                  onclick='javascript:WebForm_DoPostBackWithOptions(new WebForm_PostBackOptions("btn", "", true, "", "", fa
                  type=image src="images/login_button.gif" name=btn>
              </TD></TR></TBODY></TABLE></TD></TR></TBODY></TABLE></TD></TR>
 <TR>
   <TD><IMG src="images/login_3.jpg"
order=0></TD></TR></TBODY></TABLE></DIV></DIV>


/FORM></BODY></HTML>
```

- 编写 UserAction

```
/**
* @Title: UserAction.java
* @Package com.admiral.web.action
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
```

```java
*/
package com.admiral.web.action;

import com.admiral.domain.User;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class UserAction extends ActionSupport implements ModelDriven<User>{

    private User user = new User();

    @Override
    public User getModel() {
        return user;
    }

    public String login() {
        System.out.println(user);
        return NONE;
    }


}
```

- 配置 struts.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>

    <package name="crm" extends="struts-default" namespace="/">
        <action name="customer_*" class="com.admiral.web.action.CustomerAction"
method="{1}">
            <result name="findSuccess">/jsp/customer/list.jsp</result>
            <result name="saveUI">/jsp/customer/add.jsp</result>
            <result name="saveSuccess"
type="redirectAction">customer_find.action</result>
        </action>

        <action name="user_*" class="com.admiral.web.action.UserAction" method="
{1}">

        </action>
    </package>

</struts>
```

1.3.1.3 Action-->Service-->Dao

```
/**
* @Title: UserAction.java
* @Package com.admiral.web.action
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.web.action;

import org.apache.struts2.ServletActionContext;

import com.admiral.domain.User;
import com.admiral.service.UserService;
import com.admiral.service.impl.UserServiceImpl;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class UserAction extends ActionSupport implements ModelDriven<User>{

    private User user = new User();

    @Override
    public User getModel() {
        return user;
    }

    public String login() {
        //获取参数
        System.out.println(user);

        UserService userService = new UserServiceImpl();
        User existUser = userService.login(user);

        // 根据结果页面跳转
        if(existUser==null) {
            //登陆失败
            this.addActionError("用户名或密码错误!");
            return LOGIN;
        }else {
            //登陆成功
//          ActionContext.getContext().getSession().put("existUser", existUser);

ServletActionContext.getRequest().getSession().setAttribute("existUser",
existUser);
            return SUCCESS;
        }

    }


}
```

```java
/**
* @Title: UserServiceImpl.java
* @Package com.admiral.service.impl
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.service.impl;

import com.admiral.dao.UserDao;
import com.admiral.dao.impl.UserDaoImpl;
import com.admiral.domain.User;
import com.admiral.service.UserService;

public class UserServiceImpl implements UserService {

    @Override
    public User login(User user) {
        UserDao userDao = new UserDaoImpl();
        return userDao.login(user);
    }

}
```

```java
/**
* @Title: UserDaoImpl.java
* @Package com.admiral.dao.impl
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.dao.impl;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import com.admiral.dao.UserDao;
import com.admiral.domain.User;
import com.admiral.utils.HibernateUtils;

public class UserDaoImpl implements UserDao {
```

```
    @Override
    public User login(User user) {
        Session session = HibernateUtils.getCurrentSession();
        Transaction transaction = session.beginTransaction();

        Query query = session.createQuery("from User where user_code=? and
user_password=?");

        query.setParameter(0, user.getUser_code());
        query.setParameter(1, user.getUser_password());

        List<User> list = query.list();

        if(list.size()>0) {
            return list.get(0);
        }

        transaction.commit();

        return null;
    }

}
```

1.3.1.4 根据结果进行页面跳转

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>

    <package name="crm" extends="struts-default" namespace="/">
        <action name="customer_*" class="com.admiral.web.action.CustomerAction"
method="{1}">
            <result name="findSuccess">/jsp/customer/list.jsp</result>
            <result name="saveUI">/jsp/customer/add.jsp</result>
            <result name="saveSuccess"
type="redirectAction">customer_find.action</result>
        </action>

        <action name="user_*" class="com.admiral.web.action.UserAction" method="
{1}">
            <result name="login">/login.jsp</result>
            <result name="success" type="redirect">/index.jsp</result>
        </action>
    </package>
```

```
</struts>
```

- 在登陆页显示错误信息

```
<TABLE height=300 cellPadding=0 width=900 border=0>
  <TBODY>
  <TR>
    <TD colSpan=2 height=35></TD></TR>
  <TR>
    <TD width=360></TD>
    <TD>
      <s:actionerror/>
      <TABLE cellSpacing=0 cellPadding=2 border=0>
        <TBODY>
        <TR>
          <TD style="HEIGHT: 28px" width=80>登 录 名：</TD>
          <TD style="HEIGHT: 28px" width=150><INPUT id=txtName
            style="WIDTH: 130px" name="user_code"></TD>
          <TD style="HEIGHT: 28px" width=370><SPAN
            id=RequiredFieldValidator3
            style="FONT-WEIGHT: bold; VISIBILITY: hidden; COLOR: white">请输入登录名</SPAN></
        <TR>
          <TD style="HEIGHT: 28px">登录密码：</TD>
```

在首页显示当前用户　top.jsp

```
</TR>
<TR>
    <TD height=35 align="right">
        当前用户：<s:property value="#session.existUser.user_name"/>
            
        <A href="#" target=_top><FONT color=red>修改密码</FONT></A>
            
        <A href="#" target=_top><FONT color=red>安全退出</FONT></A>
    </TD>
</TR>
</TBODY>
```

1.3.2 实现权限拦截器

1.3.2.1 编写权限拦截器

```
/**
* @Title: PrivilegeInterceptor.java
* @Package com.admiral.web.interceptor
* @Description:
* @author 白世鑫
* @date 2020-9-30
* @version V1.0
*/
package com.admiral.web.interceptor;

import org.apache.struts2.ServletActionContext;

import com.admiral.domain.User;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.interceptor.MethodFilterInterceptor;

public class PrivilegeInterceptor extends MethodFilterInterceptor{

    @Override
    protected String doIntercept(ActionInvocation invocation) throws Exception {

        //检查 Session 是否有对象
        User existUser = (User)
ServletActionContext.getRequest().getSession().getAttribute("existUser");

        //判断从 session 中获取的用户是否为空
        if(existUser == null) {
            //没有登陆
            //给出提示信息
            ActionSupport actionSupport = (ActionSupport)
invocation.getAction();
            actionSupport.addActionError("没有登陆,没有权限操作!");
            return actionSupport.LOGIN;
        }else {
            //已经登陆
            return invocation.invoke();
        }

    }

}
```

1.3.2.2 配置拦截器

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>

    <package name="crm" extends="struts-default" namespace="/">
        <!-- 定义拦截器 -->
        <interceptors>
            <interceptor name="privilegeInterceptor"
class="com.admiral.web.interceptor.PrivilegeInterceptor"/>
        </interceptors>

        <global-results>
            <result name="login">/login.jsp</result>
        </global-results>

        <action name="customer_*" class="com.admiral.web.action.CustomerAction"
method="{1}">
            <result name="findSuccess">/jsp/customer/list.jsp</result>
            <result name="saveUI">/jsp/customer/add.jsp</result>
            <result name="saveSuccess"
type="redirectAction">customer_find.action</result>

            <interceptor-ref name="defaultStack"/>
            <interceptor-ref name="privilegeInterceptor" />
        </action>

        <action name="user_*" class="com.admiral.web.action.UserAction" method="
{1}">

            <result name="success" type="redirect">/index.jsp</result>

            <interceptor-ref name="defaultStack"/>
            <interceptor-ref name="privilegeInterceptor" >
                <param name="excludeMethods">login</param>
            </interceptor-ref>
        </action>
    </package>

</struts>
```

login.jsp

```
    FONT-SIZE: 12px; COLOR: #ffffff; FONT-FAMILY: 宋体
}
</STYLE>

<META content="MSHTML 6.00.6000.16809" name=GENERATOR></HEAD>
<BODY>
<FORM id=form1 name=form1 action="${ pageContext.request.contextPath }/user_login.action" method=post target="_parent">

<DIV id=UpdatePanel1>
<DIV id=div1
style="LEFT: 0px; POSITION: absolute; TOP: 0px; BACKGROUND-COLOR: #0066ff"></DIV>
<DIV id=div2
style="LEFT: 0px; POSITION: absolute; TOP: 0px; BACKGROUND-COLOR: #0066ff"></DIV>
```

1.4 Struts2 标签库

1.4.1 通用标签库

1.4.1.1 判断标签

1.4.2 UI标签库