Struts 2开发技术

邹国兵，博士
副教授、博导/硕导
服务计算与数据挖掘实验室
https://scdm-shu.github.io

目录

Struts 2简介

- 简单地讲，Struts 2是一个MVC框架，是MVC模式一个实现。
- MVC体系结构的三个层次
  - 模型（Model）
  - 视图（View）
  - 控制器（Controller）
- 实际上Struts 2只实现了控制层（Controller），但是对其它层的技术有很好的支持。

Struts 2简介

- **框架，即Framework**
  - **功能：实现某种功能的半成品，供开发人员用来实现某一功能或行为。**
  - **实现技术：通常是一组组件，相互协作完成功能。**
  - **特点：框架一般是成熟的，不断升级的软件。**
  - **现状：在Java EE世界中有许多优秀的轻量级框架，供开发Java EE应用使用。如MVC框架、ORM框架、系统架构级的框架等。**

Struts 2简介

- **Struts 2**是一个非常优秀的、实用的**MVC**框架。
- 注意点：任何**MVC**框架其实都只实现了**C**（**Controller**）部分，没有实现其它部分，但是它负责用控制器调用业务逻辑组件，并负责将控制器与视图整合。

Struts 2框架的体系结构

- **Struts 2框架由4部分构成：**
  - 核心控制器
  - 业务控制器
  - **Struts**配置文件
  - 拦截器链

Struts 2框架的体系结构

核心控制器（**StrutsPrepareAndExecuteFilter**）

- **StrutsPrepareAndExecuteFilter**是**Struts 2**框架的核心控制器，该控制器作为一个**Filter**运行在**Web**应用服务器上，它负责拦截所有的用户请求。

Struts 2框架的体系结构

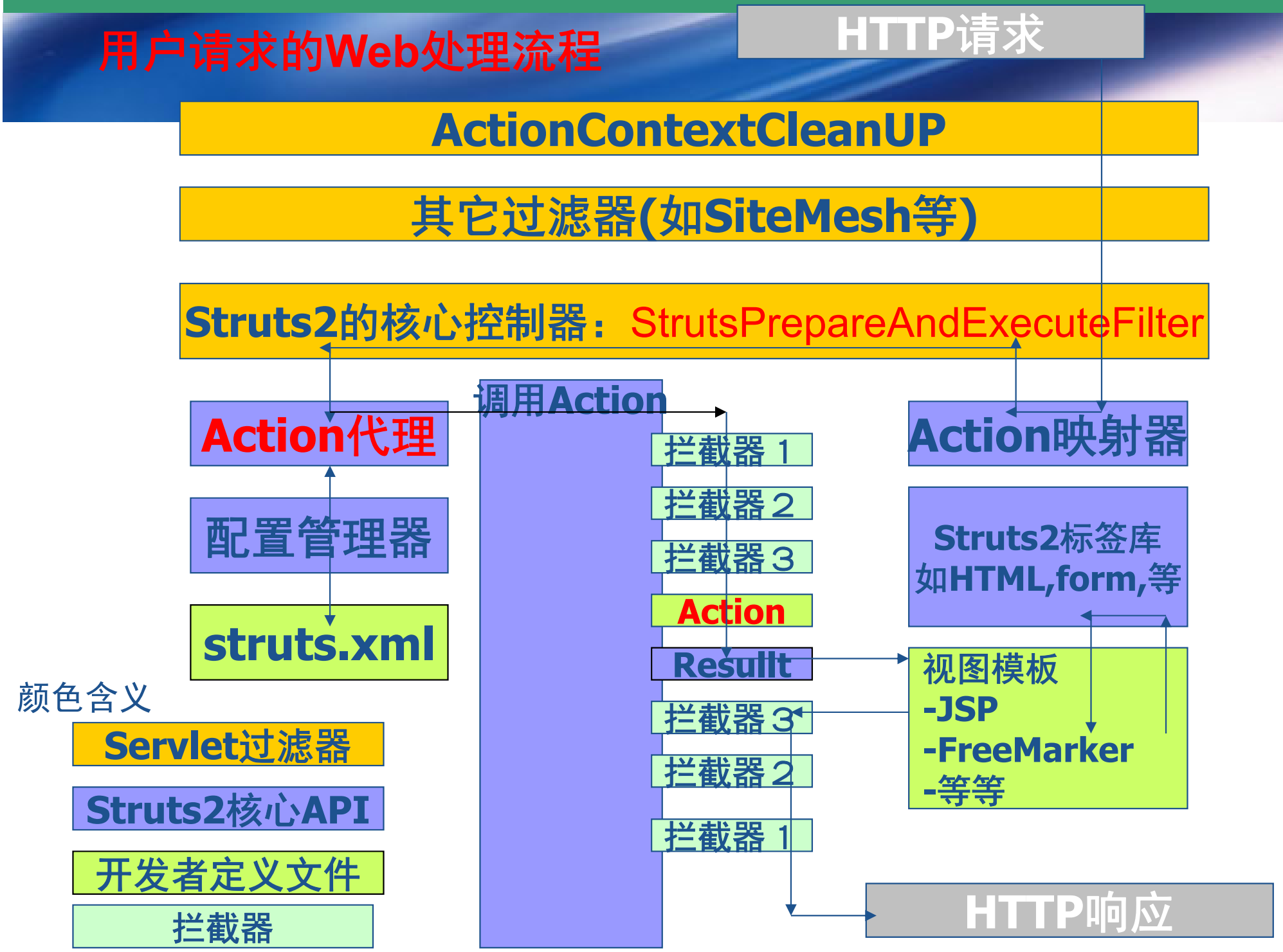
业务控制器（**Action**）

- 开发者需要实现的**Struts 2**核心组件是业务控制器。
- 业务控制器就是一个**action**，是**Struts 2**的基本程序单元。
- **Action**可以包含对用户请求的处理逻辑。
- 用户提交的所有请求，都是交给**Action**进行处理的，**Action**是实现企业应用的关键部分。

目录

- Struts 2体系架构
- Struts 2工作原理
- Struts 2组件功能分析
- Struts 2基本应用实例
- Action实现
- Struts 2配置文件
- Struts 2综合应用实例

用户请求的Web处理流程



用户请求的Web处理流程

- **Servlet过滤器**
  - 核心控制器StrutsPrepareAndExecuteFilter过滤器，在处理过程中使用ActionMapper和ActionProxy组件。

用户请求的Web处理流程

- **Action映射解析器—ActionMapper**

- 在**Struts 2**框架中，当核心控制器拦截到用户请求后，调用**ActionMapper**中定义的方法判断（不读**Struts**配置文件，而是靠系统实现中定义的一些规则进行匹配），用户请求是否与自己需要处理的请求匹配。
- 如果匹配，会继续调用**ActionProxy**组件。

用户请求的Web处理流程

- Action代理—ActionProxy组件
- 通过ActionMapper判断的请求，需要调用一个Action来处理，此时控制权交给ActionProxy。
- ActionProxy根据ActionMapper生成的URI和配置文件找到Action对象。
- 然后，ActionProxy创建代表Action对象的ActionInvocation对象，通过ActionInvocation（可认为是被各种拦截器处理装备过了的Action）执行相应的action方法。

Struts 2的处理流程

(1) 客户端初始化一个HTTP请求；

(2) 请求经过一系列的过滤器过滤处理；

(3) 核心控制器StrutsPrepareAndExecuteFilter被调用，StrutsPrepareAndExecuteFilter询问ActionMapper来决定该请求是否需要调用某个Action；

(4) 如果ActionMapper决定需要调用某个Action，StrutsPrepareAndExecuteFilter将请求的处理交给ActionProxy；

Struts 2的处理流程

(5) ActionProxy通过Configuration Manager询问框架的配置文件，查找需要调用的Action类；

(6) ActionProxy创建一个ActionInvocation实例；

(7) ActionInvocation实例使用命名模式来调用，回调Action的execute()方法，该方法先获取用户调用参数，再调用业务逻辑组件处理用户的逻辑。注意：在调用action的前后，会调用相关拦截器，自动对请求应用通用功能。

(8) 一旦Action执行完毕， ActionInvocation根据struts.xml中的相应配置找到相应的返回（Action或JSP）。

目录

核心控制器（StrutsPrepareAndExecuteFilter）

- 控制器负责拦截所有的客户端请求，然后 通过读取配置文件来确定交给哪个Action继续处理。
- StrutsPrepareAndExecuteFilter在web.xml文件中配置

```
<filter>
        <filter-name>struts2</filter-name>
        <filter-class>
            org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
       </filter-class>
  </filter>
  <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

Struts 2控制器

- 拦截器（Interceptor）

  拦截器能在action被调用之前和被调用之后执行一些“代码”。

  Struts2框架的大部分核心功能都是通过拦截器实现的，如类型转换、对象封装、数据校验、文件上传、页面预装载、国际化等，都是在拦截器的帮助下实现的。

Struts 2控制器

- 配置拦截器(struts.xml)
  1、定义拦截器
  2、使用拦截器

定义拦截器
name指定拦截器的名字
class指定拦截器对应的Java类型

```
<package name ="testInterceptor" extends="struts-default">
    <interceptors>
      <interceptor name ="beforeResult" class
                  ="tmq.interceptor.BeforeResultInterceptor"/>
      <interceptor name ="replace" class ="tmq.interceptor.MyInterceptor"/>
      ……
    </interceptors>
<action name="checkLogin" class="tmq.action.DefaultAction">
      <result name="success">/success.jsp</result>
      <result name="login">/login.jsp</result>
      <interceptor-ref name="checkLogin" />
      <interceptor-ref name="defaultStack"/>
</action>
```

使用拦截器

Struts 2控制器

- **业务控制器Action**

  Struts 2应用中起作用的业务控制器不是用户定义的Action，而是系统生成的Action代理，但该Action代理以用户定义的Action为目标。

  Action是一个普通的Java类，该类定义了一些属性，并为属性提供了对应的setter和getter方法。除此之外，Action 还提供了一个execute方法来处理。

Struts 2控制器

```
import com.opensymphony.xwork2.ActionSupport;

public class HelloWorld extends ActionSupport {
    private String message;
    public String getMessage() {
        return message;
    }
    public void setMessage(String message) {
        this.message = message;
    }
    public String execute() throws Exception {
        setMessage("你好！ Struts 2！ ");
        System.out.println(getText("HelloWorld.message"));
        return SUCCESS;
    }
}
```

Struts 2的配置文件

- Struts 2配置文件：
  - struts.xml：配置Action的文件

    定义了Struts 2的系列Action，指定Action的实现类，并定义Action处理结果与视图资源之间的映射关系。
  - struts.properties：定义Struts 2程序运行时所需的一些常量信息，它是Struts 2全局属性的文件。

Struts 2的配置文件

```
<struts>
   <package name="default" extends="struts-default">
      <action name="HelloWorld" class="example.HelloWorld">
         <result>/HelloWorld.jsp</result>
      </action>
      <action name="HelloPOJO" class="example.HelloPOJO">
         <result name="success " >/HelloPOJO.jsp</result>
      </action>
      <action name="Login" class="example.Login">
         <result name="input">/Login.jsp</result>
         <result name="success">/success.jsp</result>
      </action>
   </package>
</struts>
```

定义Action的实现类

result元素指定execute方法返回值和视图资源之间的映射关系
result元素可以有多个，其中name指定了execute方法执行后，返回的字符串。

Struts 2的配置文件

- **struts.properties：Struts 2全局属性的文件。**

  struts.properties文件的形式是系列的key、value对，它指定了Struts 2应用的全局属性。

```
#指定Struts 2处于开发状态
    struts.devMode = true
//指定当Struts 2配置文件改变后，Web框架是否重新加载Struts 2配置文件
    struts.configuration.xml.reload=true
```

目录

使用Struts 2实现登录

1 导入Struts 2类库

- Struts2-core-2.x.x.jar :Struts 2框架的核心类库
- xwork-core-2.x.x.jar :XWork类库，Struts 2在其上构建
- ognl-2.x.x.jar :对象图导航语言（Object Graph Navigation Language），struts 2框架通过其读写对象的属性
- freemarker-2.3.x.jar :Struts 2的UI标签的模板使用FreeMarker编写
- commons-logging-1.x.x.jar :ASF出品的日志包，Struts 2框架使用这个日志包来支持Log4J和JDK 1.4+的日志记录。
- commons-fileupload-1.2.1.jar 文件上传组件，2.1.6版本后必须加入此文件

使用Struts 2实现登录

2 在web.xml文件中配置ng.filter.StrutsPrepareAndExecuteFilter

```
<filter>
 <filter-name>struts2</filter-name>
 <filter-class>
org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
</filter-class>
</filter>
<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/* </url-pattern>
 </filter-mapping>
```

使用Struts 2实现登录

3 开发Action

提供属性参数及set/get方法
将页面的参数
注入到Action类中

```
public class LoginAction {
    private String name;
    private String password;
    /**在此方法里实现业务逻辑处理*/
    public String execute() throws Exception {
        if(this.name.equals("sa") && this.password.equals("123"))
            return "success";
        else
            return "error";
    }
  setter/getter
}
```

4 配置struts.xml文件

在src目录下创建struts.xml文件，在文件中对Action类进行配置，并将action与结果页面关联在一起。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN"
    "http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
        <package name="default" namespace="/" extends="struts-default">
            <action name="login" class="cn.com.web.action.LoginAction">
                <result name="success">/success.jsp</result>
                <result name="error">/error.jsp</result>
             </action>
        </package>
</struts>
```

对Action类进行配置

使用Struts 2实现登录

5 编写login.jsp及相关结果页面

```
<form action="login.action">
        <input name="name"/>
        <input name="password"/>
        <input type="submit" value="提交"/>
</form>
```

Action中的url路径与struts.xml中保持一样

表单元素名称与Action类中的属性名称保持一致

MyEclipse演示

目录

Struts2.0中的Action

- **Struts 2.0的Action**

  有Struts 1.x经验的都知道Action是Struts的核心内容，当然Struts 2.0也不例外。不过，Struts 1.x与Struts 2.0的Action区别很大。

	Struts 1.x	Struts 2.0
接口	必须继承org.apache.struts.action.Action或者其子类	无须继承任何类型或实现任何接口
表单数据	表单数据封装在FormBean中	表单数据包含在Action中，通过Getter和Setter获取

Struts2.0中的Action

- Struts2.0中的Action几种方式

  虽然，理论上Struts 2.0的Action无须实现任何接口或继承任何类，但是，我们为了方便实现Action，大多数情况下会采用以下二种方式对Action进行处理：

  一. 继承类 com.opensymphony.xwork2.ActionSupport

  二. 实现接口 com.opensymphony.xwork2.Action

  默认情况下：重写(Override)其中execute()方法。

ActionSupport类的使用

- **ActionSupport类是辅助Action类能够更好地完成工作的基类，它实现了几个接口并包含了一组默认的实现。**
- **ActionSupport类所实现的接口主要有:**

  Action, LocaleProvider, TextProvider, Validateable, ValidationAware
- **在实际开发中通常要让Action类继承ActionSupport类，该类可以使Action方便进行数据验证、国际化等工作，只需在Action中完成这些功能的具体实现。**

Struts2.0中的Action

- 使用ActionSupport的方式

  下面我们针对前面的登录的例子进行改造，主要改造以下一个环节：

1 将LoginAction类继承ActionSupport

```
public class LoginAction extends ActionSupport {
        private String name;
        private String password;
        public String execute() throws Exception {
                if(this.name.equals("sa") && this.password.equals(“123”))
                        return “success”;
                else
                        return “error”;
        }
}
```

MyEclipse演示

Struts2.0中的Action

- 使用ActionSupport的方式

  下面我们针对前面的登录的例子进行改造，主要改造以下一个环节：

2 在struts.xml中进行配置

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN"
   "http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
            <package name="default" extends="struts-default">
               <action name="login"
            class="cn.com.web.action.LoginAction">
                  <result name="success">/success.jsp</result>
                  <result name="error">/error.jsp</result>
               </action>
            </package>
</struts>
```

MyEclipse演示

实现Action接口

Action接口：

```
public interface Action
    //下面定义了5个字符串常量
    public static final String SUCCESS = "success";
    public static final String NONE = "none";
    public static final String ERROR = "error";
    public static final String INPUT = "input";
    public static final String LOGIN = "login";
    //定义处理用户请求的execute抽象方法
    public String execute() throws Exception;
}
```

Action接口的使用

- 这5个常量就是在日常开发中业务逻辑方法中返回的字符串，可以简化和标准化execute方法的返回值。
- execute()方法则是Action接口定义的一个默认的业务逻辑方法，在自定义的Action类中只需重写该方法即可，Struts 2会默认调用execute方法。

Struts 2.0中的Action

```
public class LoginAction implements Action {
        private String name;
        private String password;
        public String execute() throws Exception {
                if(this.name.equals("sa") && this.password.equals(“123”))
                        return SUCCESS;//这里采用的是一个常量值
                else
                        return ERROR;
        }//通过此种方式，可以简化使用。
}
```

接收用户数据问题

- 开发Web应用程序，首先应会遇到对用户输入数据的接收，传统Web应用程序是由开发人员调用HttpServletRequest的getparameter(String name)方法从请求中获取数据，Web框架提供了**数据绑定机制**。
- Struts 2框架通过拦截器负责提取请求参数，并将请求数据封装到相应的Action实例的属性或专门模型的属性。

如何在Action中接收用户输入

- **接收用户输入的几种方式**

  在Struts2.0中有如下几种方式可以获得用户输入信息，具体方式如下：

  一. 使用action类的属性接收用户输入
  二. 使用领域对象接收用户输入
  三. 使用ModelDriven的方式接收用户输入

如何在Action中接收用户输入

- 使用action类的属性接收用户输入

在struts 2中,可以直接使用action的属性来接收用户的输入，比如前面在登录中的案例就是采用这种方式。使用此方式注意两点：

一．在action类中提供对应的属性及set/get方法

二．在页面表单中表单元素的name属性与action中的属性名称一样

使用action属性接收用户输入

```
public class LoginAction {
        private String name;
        private String password;
        /**在此方法里实现业务逻辑处理*/
        public String execute() throws Exception {
                if(this.name.equals("sa") && this.password.equals(“123”))
                        return "success";
                else
                        return "error";
        }
    setter/getter
}
```

使用领域对象接收用户输入

- 在Action类中使用POJO类接收用户输入参数

  在Struts 2中支持直接使用领域对象来接收用户输入的数据。

```
public class LoginAction implements Action {
    private User user;
    public String execute() throws Exception {
        //业务处理
    }

    Set/Get…
```

在Action类中
直接使用POJO对象

使用领域对象接收用户输入

- 在Action类中使用POJO类接收用户输入参数

  在Struts 2中支持直接使用领域对象来接收用户输入的数据。

```
<form action="login.action">
        <input name="user.username"/>
        <input name="user.password"/>
        <input type="submit" value="提交"/>
</form>
```

在表单元素中，使用领域对象所包含的属性名表示用户输入信息

举例login.jsp

```
<%@ page contentType="text/html;charset=GBK" %>
<html>
    <head>
        <title>登录页面</title>
    </head>
    <body>
        <form action="login.action" method="post">
            <table>
                <tr>
                    <td>用户名：</td>
                    <td><input type="text" name="user.username"></td>
                </tr>
                <tr>
                    <td>密码：</td>
                    <td><input type="password" name="user.password"></td>
                </tr>
                <tr>
                    <td><input type="reset" value="重填"></td>
                    <td><input type="submit" value="登录"></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

编写User类

```
public class User implements Serializable
{
    private String username;
    private String password;

    public String getUsername()
    {
        return username;
    }

    public void setUsername(String username)
    {
        this.username = username;
    }

    public String getPassword()
    {
        return password;
    }

    public void setPassword(String password)
    {
        this.password = password;
    }
}
```

编写LoginAction类

```
public class LoginAction implements Action
{
    private User user;

    @Override
    public String execute() throws Exception
        if("zhangsan".equals(user.getUsername())
            && "1234".equals(user.getPassword()))
            return SUCCESS;
        else
            return ERROR;
    }

    public User getUser()
    {
        return user;
    }

    public void setUser(User user)
    {
        this.user = user;
    }
}
```

struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
    <package name="default" extends="struts-default">
      <action name="login" class="org.sunxin.struts2.ch03.action.LoginAction">
          <result>/success.jsp</result>
          <result name="error">/error.jsp</result>
       </action>
    </package>
</struts>
```

MyEclipse演示

使用ModelDriven Action

- 通过implements ModelDriven使用领域对象

  在Struts 2中，可以让Action类实现（implements）ModelDriven(com.opensymphony.xwork2.ModelDriven）接口，来直接操作应用程序中的领域对象，允许在Web层和业务逻辑层使用相同的对象。

  ModelDriven接口中只有一个方法，该方法返回一个用于接收用户输入数据的模型对象，如下：

```
public getModel()
```

使用ModelDriven Action

- 通过implements ModelDriven使用领域对象

  在Struts 2中，可以让Action类实现（implements）ModelDriven(com.opensymphony.xwork2.ModelDriven)接口，来直接操作应用程序中的领域对象，允许在Web层和业务逻辑层使用相同的对象。

  ModelDriven接口中只有一个方法，该方法返回一个用于接收用户输入数据的模型对象，如下：

```
public class LoginAction implements Action,ModelDriven<User> {

private User user=new User();//注意实例化POJO对象

public String execute() throws Exception {
   //业务逻辑处理
}

public User getModel() {
   return user;
}
```

实现了Action与ModelDriven接口
注意用到了范型，所以要求jdk为1.5以上

使用ModelDriven Action

- 通过implements ModelDriven使用领域对象

  在Struts 2中，可以让Action类实现（implements）ModelDriven(com.opensymphony.xwork2.ModelDriven)接口，来直接操作应用程序中的领域对象，允许在Web层和业务逻辑层使用相同的对象。

  ModelDriven接口中只有一个方法，该方法返回一个用于接收用户输入数据的模型对象，如下：

```
<form action="login.action">
        <input name="name"/>
        <input name="password"/>
        <input type="submit" value="提交"/>
</form>
```

MyEclipse演示

在页面中，模型对象中的属性可以直接通过属性名来访问，而不需要象user.name这种方式

在Action中访问Servlet API

- **Struts 2中的Action并没有和任何Servlet API耦合，这样框架更具灵活性，更易测试。**
- **但是，有时候我们必须访问Servlet API，例如要使用request对象，或者跟踪HTTP Session用户状态等。Struts2框架提供了两种方式来访问Servlet API。**

访问Servlet API对象

- 如何访问request、 session、 application等

如果我要取得Servlet API中的一些对象，
如request、response或session等，应该怎么做？

在Struts 2.0可以有两种途径获得这些对象：非IoC（控制反转Inversion of Control）途径和IoC途径。

一. 非IoC途径

要获得上述对象，通过使用com.opensymphony.xwork2.ActionContext类。

二. IoC途径

要使用IoC途径，我们首先要告诉IoC容器想取得某个对象的意愿，通过实现相应的接口做到这点。

解耦：Servlet API的访问方式

- 非IOC途径

  为了避免与Servlet API耦合在一起，方便Action类做单元测试，Struts2对HttpServletRequest、HttpSession和ServletContext进行了封装，构造了三个Map对象来替代这三种对象。

  在Action中，使用HttpServletRequest、HttpSession、ServletContext对应的Map对象保存和读取数据。获得这三个对象，用com.opensymphony.xwork2.ActionContext。

解耦：Servlet API的访问方式

- **public Object get(Object key)**

  ActionContext类没有提供类似getRequest()这样的方法来获取封装了的HttpServletRequest的Map对象。要得到request对象的Map对象，需要为get()方法传递参数“request”

- **public Map getSession()**

  获得封装了HttpSession的Map对象

- **public Map getApplication()**

  获取封装了ServletContext的Map对象

```
public class LoginAction extends ActionSupport {
        private String username;
        ActionContext context;
        Map request;
        Map session;
        Map application;
        public String execute() throws Exception {
                context=ActionContext.getContext();
                request=(Map) context.get("request");
                session=context.getSession();
                application=context.getApplication();

                request.put("req", "requst属性");
                session.put("ses", "sesion属性");
                application.put("app", "application属性");
                return SUCCESS;
        }...省略username的get/set方法
}
```

struts.xml配置如下：

```
<struts>
    <package name="scope" extends="struts-default">
        <action name="login" class="com.asm.LoginAction">
            <result>/loginSuc.jsp</result>
        </action>
    </package>
</struts>
```

login.jsp内容如下：

- <form action="login.action" method="post">
- 用户名:
- <input type="text" name="username">
- <input type="submit" value="login">
- </form>

loginSuc.jsp的主要内容如下：

MyEclipse演示

- <h4>以下使用scope.getAttribute的形式来接受</h4>
- request：<%=request.getAttribute("req") %>
- session：<%=session.getAttribute("ses") %>
- application:<%=application.getAttribute("app") %>

分析：

- 我们通过request.getAttribute这样方式可以获取对象值，这说明了这些Map request对象实际是存储在对应范围内的对象。

解耦：Servlet API的访问方式

- IOC途径（控制反转）

  除了利用ActionContext来获取request、session、application对象这种方式外，还可以采用在Action类中实现某些特定的接口的方式，让Struts2框架在运行时向Action实例注入request、session和application对象，与之有关的三个接口和它们的方法如下所示：

- **org.apache.struts2.interceptor.RequestAware**

  框架利用该接口，向Action实例注入request Map对象。该接口方法：

  - **void setRequest(Map request)**
- **org.apache.struts2.interceptor.SessionAware**

  框架利用该接口，向Action实例注入session Map对象。该接口方法：

  - **void setSession(Map session)**
- **org.apache.struts2.interceptor.ApplicationAware**

  框架利用该接口，向Action实例注入application Map对象。该接口方法：

  - **void setApplication(Map application)**

```
public class LoginAction implements Action, RequestAware,
SessionAware, ApplicationAware {
    public Map request;//在这里将通过ioc的方式注入，注意是Map类型
    public Map session;
    public Map application;
    public User user;
    public String execute() throws Exception {
        request.put("user",user);//将user对象放到request作用域中
        session.put("user",user);//将user对象放到session作用域中
        application.put("user",user);//将user对象放到application作用域中
    }

    //user对象对应的Set/get方法

    public void setRequest(Map<String, Object> request) {
        this.request=request;
    }
    public void setSession(Map<String, Object> session) {
        this.session=session;
    } //省略了setApplication方法实现。
}
```

MyEclipse演示

解耦：Servlet API的访问方式小结

- 依赖注入与控制反转：所谓依赖注入就是一个对象自己本身的初始化是依赖其它对象。比如这里Map request这些对象会依赖struts2来给其初始化，称为依赖注入。
- 而依赖注入就表示，这些对象的控制权不再由此类本身掌握，而是交给了别的对象，即是控制权反转了。
- 强调：方式二是开发中主要方式，重点掌握。

耦合：Servlet API的访问方式

- 非IOC途径

  前面采用的用Map对象来封装Servlet API。如果想要在action类中直接使用HttpServletRequest、HttpServletResponse、ServletContext这些对象，在struts2中又能以什么的方式提供支持(此种方式的不足就是与Servlet API耦合，在测试时需要Servlet容器)?

  在Struts2中可以直接获取HttpServletRequest和ServletContext对象，可以使用
  org.apache.struts2.ServletA
  ActionContext的子类，在这个[illegible]定义了三个静态方法：

思考：在这里如何获得HttpSession对象？

- public static HttpServletRequest getRequest()
  得到HttpServletRequest对象
- public static HttpServletResponse getResponse()
  得到HttpServletResponse对象
- public static ServletContext getServletContext()
  得到ServletContext对象

耦合：Servlet API的访问方式

- 非IOC途径

  前面采用的用Map对象来封装Servlet API。如果想要在action类中直接使用HttpServletRequest、HttpServletResponse、ServletContext这些对象，在

```
public class LoginAction implements Action {
//在这里展示了，如何将用户登录对象写入到request,session,application作用域中
public User user;

public String execute() throws Exception {
   HttpServletRequest request=ServletActionContext.getRequest();
   HttpSession session=request.getSession();
   ServletContext application=ServletActionContext.getServletContext();
    request.setAttribute("user",user); //将user对象放到request作用域中
   session.setAttribute("user",user); //将user对象放到session作用域中
   application.setAttribute("user",user); //将user对象放到application作用域
}
```

耦合：Servlet API的访问方式

- IOC途径

  除了利用ServletActionContext来获取HttpServletRequest对象和ServletContext对象这种方式外，Action类还可以实现ServletRequestAware和ServletContextAware接口，由struts2框架向Action实例注入HttpServletRequest和ServletContext对象。

```
public class LoginAction implements Action, ServletRequestAware,
   ServletContextAware {
public HttpServletRequest request;  //在这里将通过ioc的方式注入
public ServletContext application;
public User user;
public String execute() throws Exception {
    request.setAttribute("user",user); //将user对象放到request作用域中
    application. setAttribute("user",user); //将user对象放到application作用域中
}

//实现ServletRequestAware, ServletContextAware接口方法
```

Action实现小结

- **Struts 2接收用户输入数据的三种方式：**
  - **使用action属性接收用户输入**
  - **使用领域对象接收用户输入**
  - **使用ModelDriven接口接受用户输入**

- **Struts 2在Action中如何访问request, session, application对象：**
  - **解耦：可以使用struts2中提供的Map对象来访问HttpServletRequest,HttpSession和ServletContext对象**
  - **耦合：也可以直接访问Servlet环境中的HttpServletRequest, HttpSession和ServletContext对象**
  - **每种访问方式都提供了两种方法：非IOC和IOC（控制反向）**