一、 struts2配置文件

1.struts2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts2 PUBLIC
    "-//Apache Software Foundation//DTD struts2 Configuration 2.5//EN"
    "http://struts2.apache.org/dtds/struts2-2.5.dtd">
<!--配置常量-->
<constant name="struts2.i18n.encoding" value="UTF-8"/>
<!--引入包-->
<include file="com/lu/struts2_login.xml"></include>
<!--配置包-->
<package name="package1" extends="struts2-default" abstract="true" namespace="/">
    <!--当开启动态方法访问或者通配符的方式，需要配置允许被访问的方法-->
    <global-allowed-methods>execute2,execute3</global-allowed-methods>
    <!--配置action-->
    <action name="login" class="com.lu.login">
        <!--根据返回值配置返回地址-->
        <result name="success" type="">success.jsp</result>
        <result name="error" type="">error.jsp</result>
    </action>
</package>
```

1.1 package的属性

属性名	内容	模板
name	package名称，名字任意写，只要不与其他项目的名称相同	name=”package01“
extends	package继承其他包的内容，默认为struts2-default	extends="struts2-default"
namespace	配置命名空间（默认namespace="/"）如果为其他的，如/aaa，则地址为/aaa/action	namespace="/"
abstract	true为可以被继承，（默认）false为不可以被继承	abstract="true"

1.2 action的属性

属性	内容	模板
name	action的映射名字	name="loginAction"
class	action类的全路径	class="com.lu.action"
method	调整访问action的方法,默认为（execute）	method="execute"
converter	配置自定义的类型转换器	

1.2.1 action方法访问

1.2.1.1method访问

第一步：配置访问路径和方法

```
<action name="loginAction" class="com.lu.action.LoginAction" method="a">
<action name="loginAction2" class="com.lu.action.LoginAction" method="b">
```

第二步：填写地址

```
action="${pageContext.request.contextPath }/loginAction.action"
action="${pageContext.request.contextPath }/loginAction2.action"
```

1.2.1.2通配符访问

第一步：首先配置访问通配路径

```
<action name="loginAction2_*" class="com.lu.action.LoginAction" method="{1}">
```

第二步：添加被允许访问的方法

```
<global-allowed-methods>execute2,execute3</global-allowed-methods>
```

第三步：填写地址

```
action="${pageContext.request.contextPath }/loginAction2_execute2.action"
```

1.2.1.3 动态方法访问

第一步:首先配置常量，开启动态方法访问

```
<constant name="struts2.enable.DynamicMethodInvocation" value="true"/>
```

第二步：添加被允许访问的方法

```
<global-allowed-methods>execute2,execute3</global-allowed-methods>
```

第三步：地址填写

```
action="${pageContext.request.contextPath }/loginAction2!execute2.action"
```

1.3 result的属性

属性	内容
name	与action的返回值对应，设置跳转网页(逻辑视图的名称)，默认值success
type	控制发送类型(**redirect dispatcher**{action to Jsp} **chain redirectAction**{action to action} **stream**{提供文件下载的功能})

2.struts2的常用常量

struts22有大量的常量，在default.properties配置文件中

常量名	介绍
struts2.i18n.encoding	设置post的字符编码
struts2.multipart.saveDir	文件上传地址
struts2.devMode	开发者模式(显示完整的错误信息)
struts2.i18n.reload	开启后每次访问前重新加载xml
struts2.configuration.xml.reload	开启后每次修改被配置文件后重新加载配置文件
struts2.multipart.maxSize	上传文件大小限制
struts2.action.extension	请求默认的扩展名
struts2.enable.DynamicMethodInvocation	动态方法访问

2.1 常量使用方式(三种)(constant)

2.1.1 第一种 在struts2.xml中修改

```
<constant name="struts2.i18n.encoding" value="UTF-8"/>
```

2.1.2 第二种 在web.xml中的过滤器配置中修改

```
<init-param>
    <param-name>struts2.i18n.encoding</param-name>
    <param-value>UTF-8</param-value>
</init-param>
```

2.1.3 第三种 在struts2.properties中修改（该方式只能修改常量，不能作为配置文件）

```
struts2.i18n.encoding=UTF-8
```

3.分模块开发的配置(include)

引入其他的struts2配置文件

```
<include file="com/lu/struts2_demo01.xml/>
```

4.web.xml配置

添加过滤器

```
  <filter>
    <filter-name>struts22</filter-name>
    <filter-
class>org.apache.struts22.dispatcher.filter.struts2PrepareAndExecuteFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts22</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

二、 struts2 Action将数据向前台发数据

1. 将数据存入request

1.1 struts2原生的方式

```
ActionContext context=ActionContext.getContext();
Map map=context.get("request");
//向request中存值
context.set("name","lugege");
//向session存值
context.getSession().put("name2","lugeO2");
//向application中存值
context.getApplication().put("name3","lugeO3");
```

1.2 通过ServletActionContext获取request对象

```
HttpServletRequest req=ServletActionContext.getRequest();
//向ruquest存值
req.setAttribute("name","lugege");
//向session存值
req.getSession.setAttribute("name2","lugeO2");
```

1.3 通过接口注入的方式获取request对象（IOC机制）

```
public class LoginAction extends ActionSupport implements ServletRequestAware{

    private HttpServletRequest req;
    public void setServletRequest(HttpServletRequest arg0) {
        // TODO Auto-generated method stub
        req=arg0;
    }
```

三、struts2的数据封装

1. 单个数据封装

1. 1 属性驱动：在action中提供表的对应的属性和set方法

JSP

```
<form action="${pageContext.request.contextPath }/loginAction2_test.action" method="post">
    姓名：<input type="text" name="name"> <br/>
    密码：<input type="text" name="password"><br/>
    <input type="submit" value="提交">
</form>
```

Action

```java
private String name;
private String password;
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getPassword() {
    return password;
}

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

1.2 属性驱动：在页面中使用表达式的方法，在action用domain对象

JSP

```jsp
<form action="${pageContext.request.contextPath }/loginAction.action" method="post">
    姓名：<input type="text" name="user.name"> <br/>
    密码：<input type="text" name="user.password"><br/>
    <input type="submit" value="提交">
</form>
```

Action

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

1.3 模型驱动 实现接口ModelDrivern<?>接口

JSP

```
<form action="loginAction3!execute3.action" method="post">
    姓名：<input type="text" name="name"> <br/>
    密码：<input type="text" name="password"><br/>
    <input type="submit" value="提交">
</form>
```

Action

```
public class LoginAction extends ActionSupport implements ServletRequestAware,ModelDriven<User>{

    private User user=new User();
    @Override
    public User getModel() {
        // TODO Auto-generated method stub
        return user;
    }
```

2.多个数据封装

2.1 使用属性驱动（List集合）

Jsp

```
<form action="${pageContext.request.contextPath }/loginAction2_execute4.action" method="post">
    姓名：<input type="text" name="list[0].name"> <br/>
    密码：<input type="text" name="list[0].password"><br/>
    姓名：<input type="text" name="list[1].name"> <br/>
    密码：<input type="text" name="list[1].password"><br/>
    <input type="submit" value="提交">
</form>
```

Action

```
List<User> list=new ArrayList<User>();
public List<User> getList() {
    return list;
}
public void setList(List<User> list) {
    this.list = list;
}
public String execute4() throws Exception {
    for(User user:list) {
        System.out.println(user.toString());
    }
    return SUCCESS;
}
```

2.2 使用属性驱动（Map键值对）

Jsp

```
<form action="${pageContext.request.contextPath }/loginAction2_execute4.action" method="post">
    姓名：<input type="text" name="map['one'].name"> <br/>
    密码：<input type="text" name="map['one'].password"><br/>
    姓名：<input type="text" name="map['two'].name"> <br/>
    密码：<input type="text" name="map['two'].password"><br/>
    <input type="submit" value="提交">
```

Action

```
Map<String,User> map=new HashMap<String,User>();
public Map<String, User> getMap() {
    return map;
}
public void setMap(Map<String, User> map) {
    this.map = map;
}
public String execute4() throws Exception {
    Set                t = map.keySet();
    for                ySet) {
        System.out.println(key+"  "+map.get(key));
    }
    return SUCCESS;
}
```

Open Declaration
Open Implementation