

# JAVA EE WEB应用开发基础

《Java EE Web应用开发基础》（电子工业出版社）

杭州电子科技大学 俞东进 任祖杰

dodge2000@163.com

dbsi.hdu.edu.cn

# 第7章：STRUTS 入门

- MVC简介
- Struts体系结构
- Struts配置
- 编写Action
- 配置Action
- Struts应用举例
- 思考题



随着实际 Web 应用的使用越来越广泛，Web 应用的规模也越来越大，开发人员发现动态 Web 应用的维护成本越来越大，即使只需要修改该页面的一个简单按钮文本，或者一段静态的文本内容，也不得不打开混杂的动态脚本的页面源文件进行修改——这是一种很大的风险，完全有可能引入新的错误。



- Java阵营发布了一套完整的企业开发规范 J2EE(JAVA EE),微软也发布了ASP.NET.它们本质都是分层思想, 解决Web应用维护困难的问题。



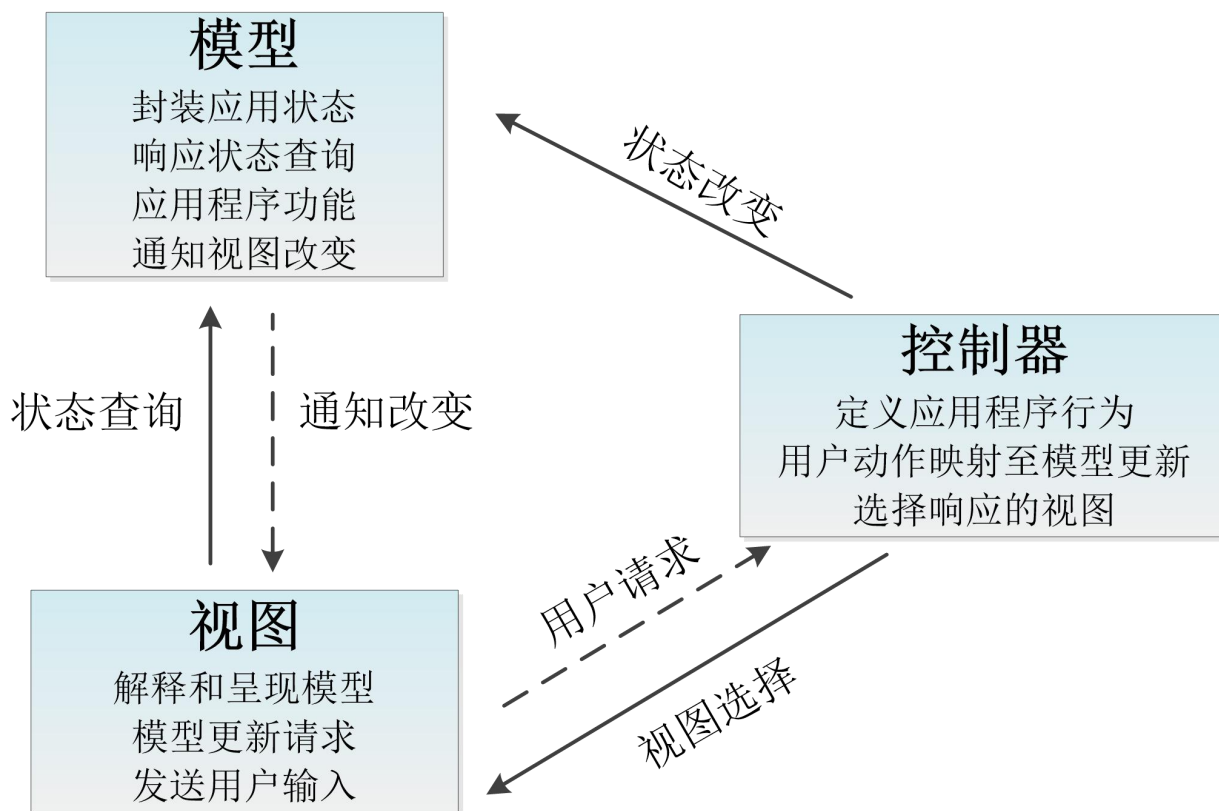
# MVC简介

- MVC是Model（模型）—View（视图）—Controller（控制器）的缩写，是一种非常经典的软件设计模式
- MVC框架的优点
  - 结构更加直观
  - 各个模块的代码按各自功能实现高效的分离



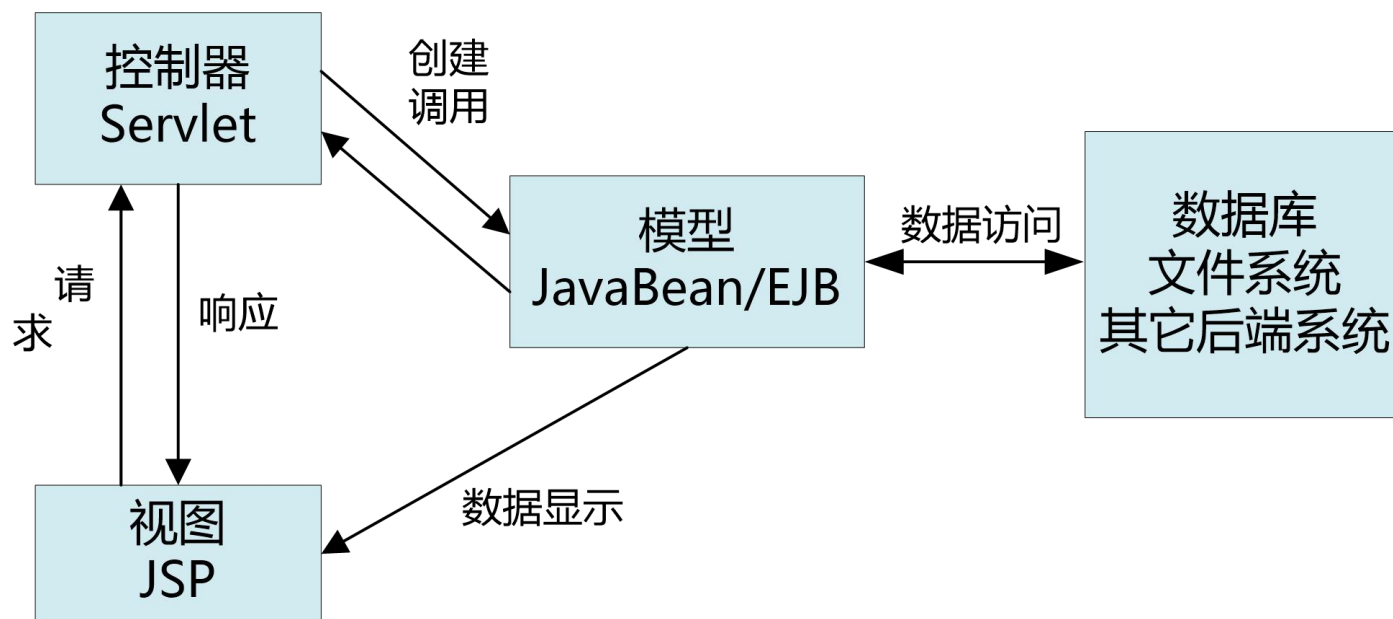
# MVC简介

## ○ MVC模式框架



# MVC简介

## ○ 基于MVC的Java EE Web开发过程



# STRUTS体系结构

- Struts 2实现了MVC的各项特性，是一个非常典型的MVC框架
- 与Struts 2紧密相关的两个概念
  - **Action**: Action是由开发人员编写的类，负责Web应用程序中实现页面跳转的具体逻辑
  - **Interceptor**: 拦截器 (Interceptor) 是动态拦截Action时调用的对象
- Struts 2使用多个拦截器来处理用户的请求，实现用户的业务逻辑代码与Servlet API分离





# 搭建STRUTS2开发环境的步骤

搭建Struts2环境时，我们一般需要做以下几个步骤的工作：

- 1》创建javaweb工程
- 2》找到开发Struts2应用需要使用到的jar文件.
- 3》在web.xml中加入Struts2 MVC框架启动配置
- 4》创建action文件
- 5》编写Struts2的配置文件
- 6》创建jsp文件



# 1 创建JAVAWEB工程

创建struts2工程



## 2 开发STRUTS2需要的JAR文件

文件名	说 明
<b>struts2-core-2.3.4.1.jar</b>	<b>Struts 2</b> 框架的核心类库
<b>Xwork-core-2.3.4.1.jar</b>	<b>XWork</b> 类库， <b>Struts 2</b> 的构建基础
<b>Ognl-3.0.5.jar</b>	<b>Struts 2</b> 使用的一种表达式语言类库
<b>freemarker-2.3.19.jar</b>	<b>Struts 2</b> 的标签模板使用类库
<b>commons-fileupload-1.2.2.jar</b>	<b>Struts 2</b> 文件上传依赖包
<b>javassist-3.11.0.GA.jar</b>	代码生成工具包
<b>commons-lang3-3.1.jar</b>	<b>Apache</b> 语言包，是 <b>java.lang</b> 包的扩展
<b>commons-io-2.0.1.jar</b>	<b>Apache IO</b> 包



# 3 STRUTS2的启动配置

配置web.xml如下:

```
<filter>
  <filter-name>StrutsPrepareAndExecuteFilter</filter-name>
  <filter-
class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-
class>
  <!-- 自从Struts 2.1.3以后, 下面的FilterDispatcher已经标注为过时
  <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class> -->
</filter>
<filter-mapping>
  <filter-name>StrutsPrepareAndExecuteFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

将全部请求定位到指定的  
**Struts 2**过滤器中

在StrutsPrepareAndExecuteFilter的init()方法中将会读取类路径下默认的配置文件的struts.xml完成初始化操作。注意: struts2读取到struts.xml的内容后, 是将内容封装进javabean对象然后存放在内存中, 以后用户的每次请求处理将使用内存中的数据, 而不是每次请求都读取struts.xml文件

## 4 创建ACTION文件

```
import com.opensymphony.xwork2.Action;  
public class HelloWorldAction implements Action{  
    public String execute() throws Exception {  
        System.out.println("helloWorld");  
        //转到成功页面  
        return "success";  
    }  
}
```

注:struts2中action要实现action的接口



# 5 编写STRUTS2的配置文件

Struts2默认的配置文件的为struts.xml，该文件需要存放在WEB-INF/classes下，也就是当前工程的src下,该文件的配置模版如下：

## •配置 package 元素

Struts2 把各种 Action 分门别类地组织成不同的包. 可以把包想象为一个模块. 一个典型的 struts.xml 文件可以有一个或多个包

每个 package 元素都必须有一个 name 属性

namespace 属性是可选的, 如果它没有给出, 则以 “/” 为默认值. 若 namespace 有一个非默认值, 则要想调用这个包里的Action, 就必须把这个属性所定义的命名空间添加到有关的 URI 字符串里

```
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>
  <package name="primer" namespace="/primer" extends="struts-default">
    <action name="helloWorldAction" class="cn.itcast.primer.HelloWorldAction">
      <result name="success"/>/success.jsp</result>
    </action>
  </package>
</struts>
```

package 元素通常要对 struts-default.xml 文件里定义的 struts-default 包进行扩展. 这么做了以后, 包里的所有动作就可以使用在 struts-default.xml 文件里的结果类型和拦截器了.

# STRUTS2的配置文件说明

- 配置 action 元素

action 元素嵌套在 package 元素内部, 它表示一个 Struts请求.

```
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configur
  "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>
  <package name="primer" namespace="/primer" extends="struts-default">

    <action name="helloWorldAction" class="cn.itcast.primer.HelloWorldAction">

      <result name="success"/>success.jsp</result>

    </action>

  </package>
</struts>
```

`<a href="${pageContext.request.contextPath}/primer/helloWorldAction.action">helloWorld</a>`

action 元素的 class 属性是可选的. 如果没有配置 class 属性, Scom.opensymphony.xwork2.ActionSupport 作为其默认值. 如果 class 属性, 还可以使用 method 属性配置该类的一个动作方法性的默认值为 execute

每个 action 都必须有一个 name 属性, 该属性和用户请求 servletPath 之间存在着——对应关系



# STRUTS2的配置文件

## •配置 result 元素

result 元素:<action> 的一个子元素, 它告诉 struts 在完成动作后把控制权转交到哪里. result 元素(的name 属性)对应着 Action 方法的返回值. 因为动作方法在不同情况下可能返回不同的值, 所以同一个 action 元素可能会有多个 result 元素

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>
    <package name="primer" namespace="/primer" extends="struts-default">

        <action name="helloWorldAction" class="cn.itcast.primer.HelloWorldAction">

            <result name="success" type="dispatcher"/>success.jsp</result>

        </action>

    </package>
</struts>
```

result 元素的 type 属性负责指定结果类型. type 属性的须是在包含当前包或者是当前包的父包里注册过的结. type 属性的默认值为 dispatcher

result 元素的 name 属性建立  
<result> 和 Action 方法返回值之  
间的映射关系。

name 属性的默认值为 “success”

```
public class HelloWorldAction implements Action{

    public String execute() throws Exception {
        System.out.println("helloWorld");
        return "success";
    }
}
```



# STRUTS2的配置文件

```
<package name="default" namespace="/" extends="struts-default">
  <action name="HelloWorldAction" class="cn.itcast.action.HelloWorldAction"
    method="execute">
    <result name="success">/WEB-INF/page/hello.jsp</result>
  </action>
```

在struts2框架中使用包来管理Action，包的作用和java中的类包是非常类似的，它主要用于管理一组业务功能相关的action。在实际应用中，我们应该把一组业务功能相关的Action放在同一个包下。

配置包时必须指定name属性，如果其他包要继承该包，必须通过该属性进行引用，注意：name名称是唯一。包的namespace属性用于定义该包的命名空间。该属性可以不配置，对本例而言，如果不指定该属性，默认的命名空间为“/”

通常每个包都应该继承struts-default包，**struts-default包是由struts内置的，它定义了struts2内部的众多拦截器和Result类型**。而Struts2很多核心的功能都是通过这些内置的拦截器实现。如：从请求中把请求参数封装到action、文件上传和数据验证等等都是通过拦截器实现的。当包继承了struts-default包才能使用struts2为我们提供的这些功能。struts-default包是在struts2-core-2.x.x.jar文件中的struts-default.xml中定义。struts-default.xml也是Struts2默认配置文件。Struts2每次都会自动加载struts-default.xml文件。

包还可以通过abstract=“true”定义为抽象包，抽象包中不能包含action。



# STRUTS2的启动配置(底层代码)

```
public void init() {  
    if (configurationManager == null) {  
        configurationManager = new ConfigurationManager(BeanSelectionProvider.DEFAULT_BEAN_NAME);  
    }  
  
    try {  
        init_DefaultProperties(); // [1]  
        init_TraditionalXmlConfigurations(); // [2]  
        init_LegacyStrutsProperties(); // [3]  
        init_CustomConfigurationProviders(); // [5]  
        init_FilterInitParameters(); // [6]  
        init_AliasStandardObjects(); // [7]  
    }  
}
```

```
private void init_TraditionalXmlConfigurations() {  
    String configPaths = initParams.get("config");  
    if (configPaths == null) {  
        configPaths = DEFAULT_CONFIGURATION_PATHS;  
    }  
    String[] files = configPaths.split("\\s*[,]\\s*");  
    for (String file : files) {  
        if (file.endsWith(".xml")) {  
            if ("xwork.xml".equals(file)) {  
                configurationManager.addConfigurationProvider(new XmlConfigurationProvider(file, false));  
            } else {  
                configurationManager.addConfigurationProvider(new StrutsXmlConfigurationProvider(file, false, servletContext));  
            }  
        } else {  
            throw new IllegalArgumentException("Invalid configuration file name");  
        }  
    }  
}
```

```
private static final String DEFAULT_CONFIGURATION_PATHS = "struts-default.xml,struts-plugin.xml,struts.xml";
```

以上方法在org.apache.struts2.dispatcher.Dispatcher类中

# 6 创建JSP文件

定义test.jsp文件

增加如下连接:

```
<a href="${pageContext.request.contextPath}
/primer/HelloWorldAction.action">helloworld</a><br>
```

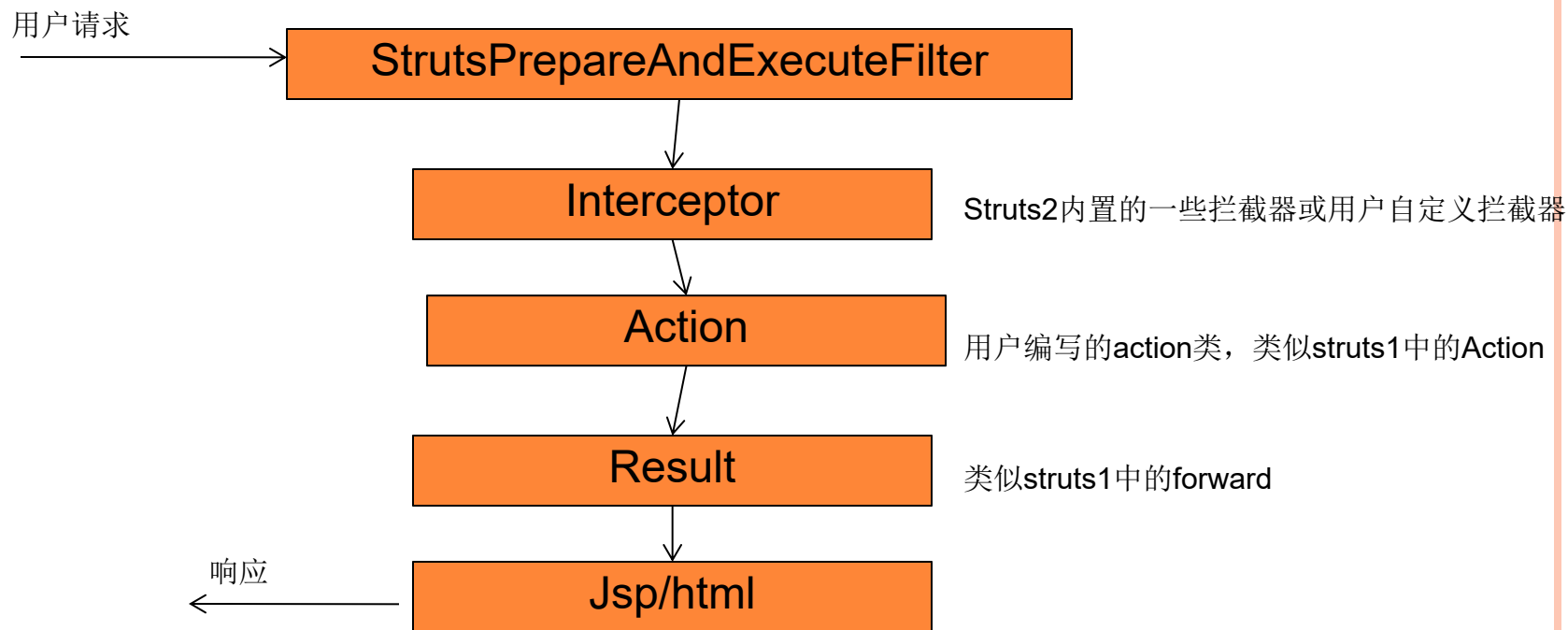


## 定义hello.jsp文件

```
<%@ 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></title>
</head>
<body>
hello ${message }</body>
</html>
```



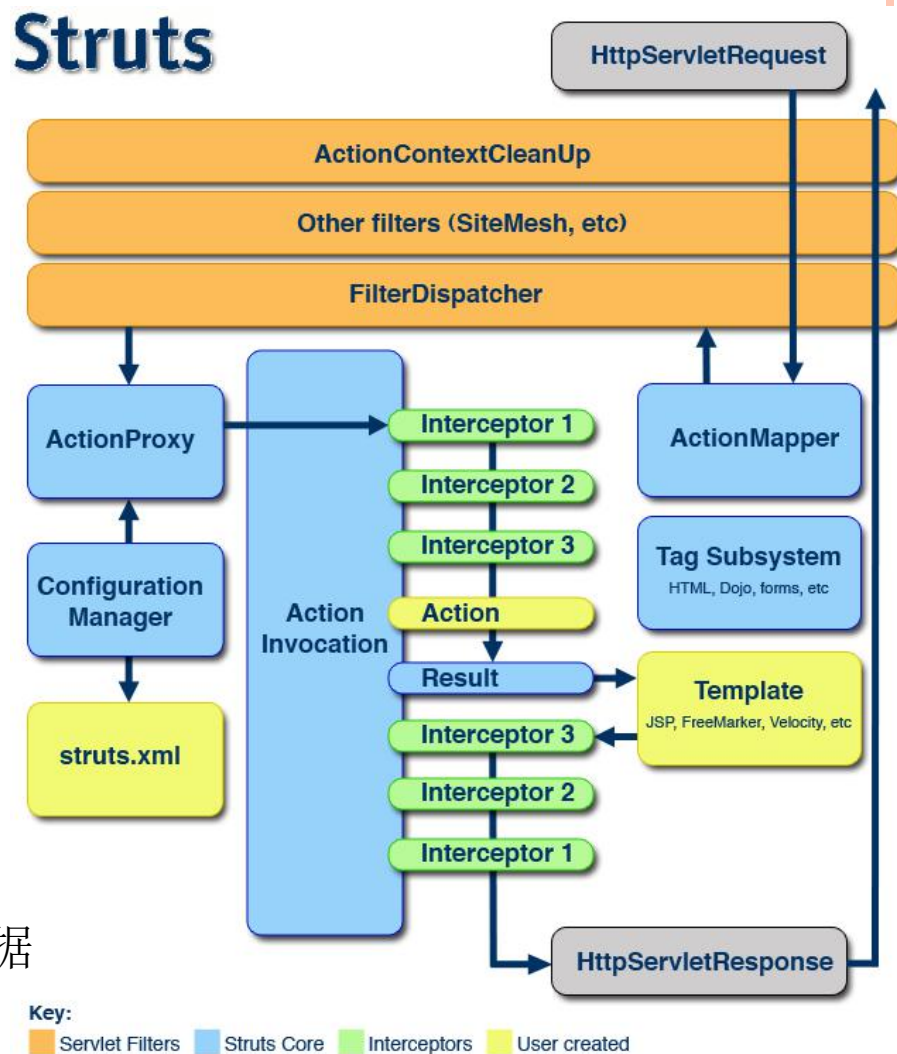
# STRUTS2的处理流程



# Struts 2体系架构

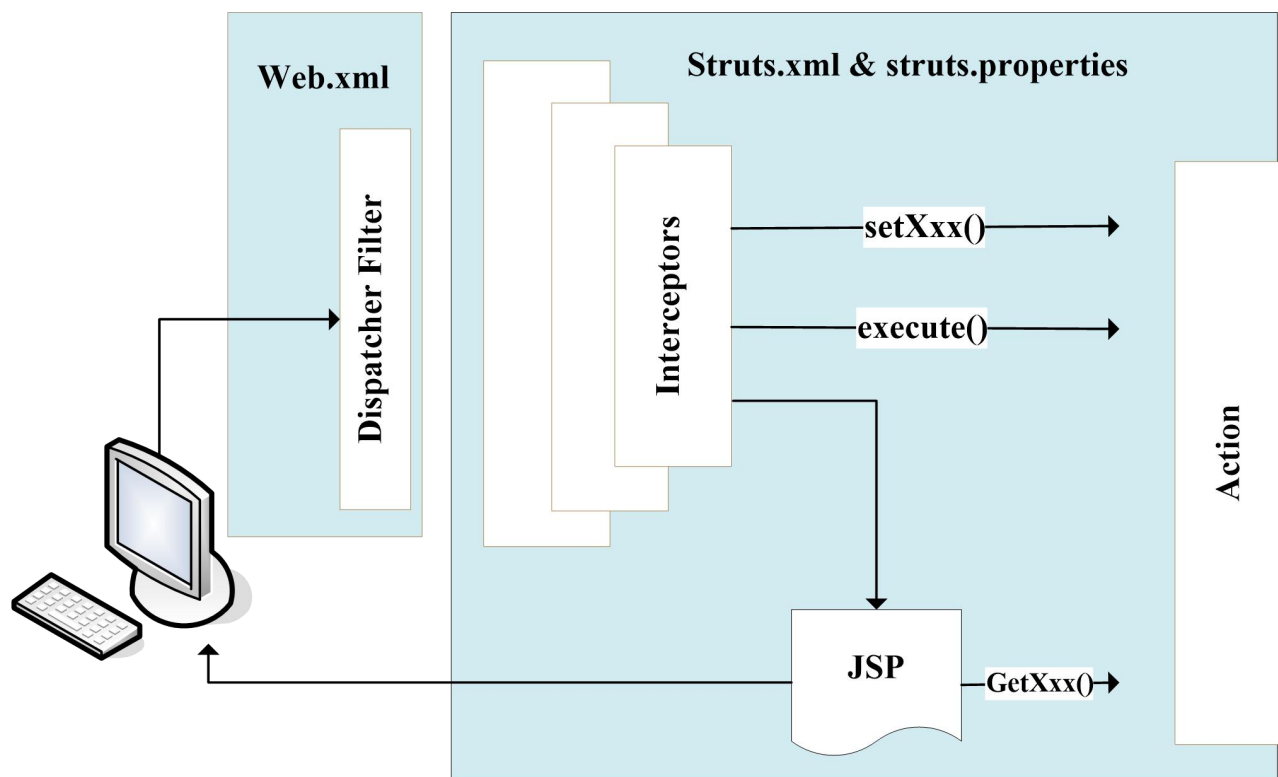
- 分发器  
FilterDispatcher
- 拦截器Interceptor
- 转换器Converter
- 业务控制器Action
- 视图模板Template

(1) FilterDispatcher询问ActionMapper决定请求是否调用某个Action  
(2) ActionProxy通用Configuration Manager询问框架配置文件找到需要的Action类  
(3) ActionProxy创建ActionInvocation,根据Struts.xml配置找到对应的返回结果。



# STRUTS体系结构

## ○ Struts2的基本体系结构



# 访问HELLOWORLD应用的路径的设置

在struts2中，访问struts2中action的URL路径由两部份组成：

**包的命名空间+action的名称**

例如：访问本例子HelloWorldAction的URL路径为：**/helloWorldAction.action** (注意：完整路径为：**http://localhost:端口/内容路径/helloWorldAction.action**)。另外我们也可以加上.action后缀访问此Action。

```
<package name="default" namespace="/" extends="struts-default">  
  <action name="helloWorldAction" class="cn.it.action.HelloWorldAction">  
    <result name="success" type="dispatcher">/hello.jsp</result>  
  </action>  
</package>
```





# ACTION名称的搜索顺序

1. 获得请求路径的URI，例如url是：  
`http://server/struts2/path1/path2/path3/test.action`
2. 首先寻找namespace为/path1/path2/path3的package，  
如果存在这个package，则在这个package中寻找名字为test的action，  
如果不存在这个package则转步骤3；
3. 寻找namespace为/path1/path2的package，  
如果存在这个package，则在这个package中寻找名字为test的action，  
如果不存在这个package，则转步骤4；
4. 寻找namespace为/path1的package，  
如果存在这个package，则在这个package中寻找名字为test的action，  
如果仍然不存在这个package，就去默认的namespace的package下面去找名字为test的action（默认的命名空间为空字符串“/”），  
如果还是找不到，页面提示找不到action。



# ACTION配置中的各项默认值

问题:如果没有为action指定class, 默认是  
**com.opensymphony.xwork2.ActionSupport**  
执行**ActionSupport**中的**execute**方法

由**struts-default.xml**文件

```
<default-class-ref class="com.opensymphony.xwork2.ActionSupport" />
```

决定

```
<package name="primer" namespace="/" extends="struts-default">  
  <action name="helloWorldAction" class="cn.it.action.HelloWorldAction">  
    <result name="success" type="dispatcher">/hello.jsp</result>  
  </action>
```

```
<action name="actionNoClass">  
  <result>/success.jsp</result>  
</action>
```

```
</package>
```

- 1>如果没有为action指定class, 默认是ActionSupport。
- 2>如果没有为action指定method, 默认执行action中的execute() 方法。  
ActionSupport的execute方法里面就一句话**return "success";**
- 3>如果没有指定result的name属性, 默认值为success。



# ACTION配置中的各项默认值

问题:如果请求的路径查找不到**action**的情况下,程序运行会抛出异常,可以通过配置当找不到**action**的情况下,会执行默认的**action**

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

**<!--指定默认的action引用,如果该包下没有对应action配置,则启用该配置-->**

```
<default-action-ref name="helloWorldAction"></default-action-ref>
```

```
<action name="helloWorldAction" class="cn.it.action.HelloWorldAction">
```

```
    <result name="success" type="dispatcher">/hello.jsp</result>
```

```
</action>
```

```
<action name="actionNoClass">
```

```
    <result>/success.jsp</result>
```

```
</action>
```

```
</package>
```



# STRUTS 2处理的请求后缀

**StrutsPrepareAndExecuteFilter**是Struts 2框架的核心控制器，它负责拦截由<url-pattern>/\*</url-pattern>指定的所有用户请求，当用户请求到达时，该Filter会过滤用户的请求。默认情况下，如果用户请求的路径不带后缀或者后缀以.action结尾，这时请求将被转入Struts 2框架处理，否则Struts 2框架将略过该请求的处理。

根据配置文件:struts2-core-2.1.8.1.jar包下的  
org.apache.struts2/default.properties文件定义的常量决定  
struts.action.extension=action,,

默认处理的后缀是可以通过常量“struts.action.extension”进行修改的，如下面配置Struts 2只处理以.do为后缀的请求路径：

```
<struts>  
  <constant name="struts.action.extension" value="do"/>  
</struts>
```

如果用户需要指定多个请求后缀，则多个后缀之间以英文逗号（,）隔开。  
如：

```
<constant name="struts.action.extension" value="do,go"/>
```



# 细说常量定义

常量可以在struts.xml或struts.properties中配置，建议在struts.xml中配置，两种配置方式如下：

在**struts.xml**文件中配置常量

```
<struts>
```

```
  <constant name="struts.action.extension" value="do"/>
```

```
</struts>
```

在**struts.properties**中配置常量, (**struts.properties**文件放置在src下)

```
struts.action.extension=do
```

因为常量可以在多个配置文件中定义，所以我们需要了解下**struts2**加载常量的搜索顺序：

struts-default.xml

struts-plugin.xml

struts.xml

struts.properties

web.xml

如果在多个文件中配置了同一个常量，则后一个文件中配置的常量值会覆盖前面文件中配置的常量值。



# 常用的常量介绍

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

指定默认编码集,作用于`HttpServletRequest`的`setCharacterEncoding`方法 和 `freemarker`、`velocity`的输出

- `<constant name="struts.action.extension" value="do"/>`

该属性指定需要**Struts 2**处理的请求后缀,该属性的默认值是**action**,即所有匹配**\*.action**的请求都由**Struts2**处理。如果用户需要指定多个请求后缀,则多个后缀之间以英文逗号(,)隔开

- `<constant name="struts.serve.static.browserCache" value="false"/>`

设置浏览器是否缓存静态内容,默认值为**true**(生产环境下使用),开发阶段最好关闭

- `<constant name="struts.configuration.xml.reload" value="true"/>`

当**struts**的配置文件修改后,系统是否自动重新加载该文件,默认值为**false**(生产环境下使用),开发阶段最好打开

- `<constant name="struts.objectFactory" value="spring" />`

与**spring**集成时,指定由**spring**负责**action**对象的创建

- `<constant name="struts.enable.DynamicMethodInvocation" value="false"/>`

该属性设置**Struts 2**是否支持动态方法调用,该属性的默认值是**true**。如果需要关闭动态方法调用,则可设置该属性为 **false**

- `<constant name="struts.multipart.maxSize" value="10701096"/>`

上传文件的大小限制



# STRUTS 2再体验4-2

- 第一步：加载Struts2 类库
- 第二步：配置web.xml
- 第三步：开发视图层页面
  - 登录页面login.jsp
  - 登录成功页面success.jsp
  - 登录失败页面fail.jsp

```
<h1>登录成功</h1>
```

```
<div>
```

```
    欢迎您, ${username}!
```

```
</div>
```

```
<h1>登录失败</h1>
```

```
<div>
```

```
    用户名为空, 或用户名密码不匹配
```

```
</div>
```

```
<form action="login.action">
```

```
    <div>
```

```
        用户名: <input name="username" type="text" value="" />
```

```
    </div>
```

```
    <div>
```

```
        密码: <input name="password" type="password" value="" />
```

```
    </div>
```

```
    <input type="submit" value="提交" />
```

```
</form>
```

# STRUTS 2再体验4-3

- 第四步：开发控制层Action-LoginAction

```
public class LoginAction {  
    private String username = "";  
    private String password = "";  
    public String execute() {  
        if("jbit".equals(username) && "bdqn".equals(password)) {  
            return "success";  
        } else {  
            return "fail";  
        }  
    }  
    ... //省略setter和getter方法  
}
```





# STRUTS 2再体验4-4

- 第五步：配置Struts 2配置文件（struts.xml）

```
...  
<struts>  
  <package name="default" namespace="/" extends="struts-  
default">  
    <action name="login" class="cn.itcast.action.LoginAction">  
      <result name="success">/WEB-  
INF/page/success.jsp</result>  
      <result name="fail">/WEB-INF/page/fail.jsp</result>  
    </action>  
  </package>  
</struts>
```



演示示例：使用**Struts 2**实现用户登录

# 详细流程

- 1、浏览器初始化一个指向Servlet容器的请求
- 2、web.xml指定Struts2的核心控制器  
StrutsPrepareAndExecuteFilter接收用户请求，如果符合要求转入Struts2框架处理，进行请求分发，以及调用指定的Action操作
- 3、Struts2框架获得请求后，根据url中的前面部分调用相应的业务逻辑部分Action。  
Action处理过程中，会通过get set方法得到表单中相应的数据



- 4、拦截器对请求对应的相应的功能，调用Action中的execute()方法，后者获得用户请求参数，在调研相应的业务逻辑组件以处理用户的请求。
- 5、当Action 处理用户请求结束后，会返回一个String类型的处理结果。根据这个result值，根据配置文件struts.xml找到相应的关联的也没，并跳转到管理的关联的页面。并会调用set 方法填充数据



# STRUTS配置

- Struts 2的体系结构的各个部分都依赖于它的配置文件：
  - **web.xml**: 包含所有必须的框架组件的Web部署描述符。如要使用Struts2, 则必须在该文件中定义Struts 2的核心控制器FilterDispatcher以及过滤规则
  - **struts.xml**: 主要负责管理应用中的Action映射关系, 以及Action包含的Result定义等
  - **struts.properties**: 定义了Struts 2框架的全局属性



## WEB.XML

- 位置：WEB-INF目录下
- 配置加载Struts2的核心控制器（版本有区别）
- **StrutsPrepareAndExecuteFilter**是Struts 2框架的核心控制器，它负责拦截由<url-pattern>/\*</url-pattern>指定的所有用户请求，当用户请求到达时，该Filter会过滤用户的请求。默认情况下，如果用户请求的路径不带后缀或者后缀以.action结尾，这时请求将被转入Struts 2框架处理，否则Struts 2框架将略过该请求的处理。



## STRUTS.XML :

- 位置：src目录下
- 主要负责管理应用中的Action映射关系，以及Action包含的Result定义等



# 编写ACTION

- Action是Struts 2的核心，开发人员需要根据业务逻辑实现特定的Action代码，并在struts.xml中配置Action
- 每个Action都有一个execute()方法，实现对用户请求的处理逻辑
- Action中execute()方法会返回一个String类型的处理结果。该String值用于决定页面需要跳转到哪个视图或者另一个Action



# 编写ACTION

## ○ Action类型

- Action定义为普通Java类，实现一个execute()方法即可，该方法返回的结果决定了页面跳转的方向，例如：

```
public class LoginAction {  
    public String execute() {  
        return "success";  
    }  
}
```





# 编写ACTION

## ○ Action类型 ( 续 )

- 实现com.opensymphony.xwork2.Action接口，这个接口中定义了一些常量，如SUCCESS，ERROR，以及一个execute()方法，例如：

```
import com.opensymphony.xwork2.Action;
public class LoginAction implements Action {
    public String execute() {
        //return "success";
        return this.SUCCESS;
        //SUCCESS常量 值为: "success"
    }
}
```



# 编写ACTION

## ○ Action类型（续）

- 继承com.opensymphony.xwork2.ActionSupport类。这个ActionSupport类又实现了Action接口，所以只需要重写execute()方法就可以了例如：

```
import com.opensymphony.xwork2.ActionSupport;
public class LoginActionSimpleAction3 extends
    ActionSupport {
    public String execute() {
        //return "success";
        return this.SUCCESS
        //SUCCESS是ActionSupport的String常静态数据成员，
        值为: "success"
    }
}
```



# 编写ACTION

## ○ 在Action中访问Servlet API

- Struts2提供用于Action访问Servlet API的类：
  - com.opensymphony.webwork.ActionContext
  - com.opensymphony.webwork.ServletActionContext
- ServletActionContext类直接继承了ActionContext类。通过ServletActionContext提供的一些静态方法：
  - javax.servlet.http.HttpServletRequest: HTTPservlet请求对象
  - javax.servlet.http.HttpServletResponse: HTTPservlet响应对象
  - javax.servlet.ServletContext: Servlet上下文信息
  - javax.servlet.ServletConfig: Servlet配置对象
  - javax.servlet.jsp.PageContext: Http页面上下文



# 配置ACTION

- Struts 2使用Package来配置一个Action，在<package>元素下的<action>子元素中配置Action
- Package的配置元素
  - **name**: package的名字，用于其他package引用该package时唯一标识该package的关键字。
  - **extends**: 定义package的继承源。package可以继承其他package，继承其他package中的Action定义以及拦截器定义。
  - **namespace**: 为解决命名冲突，package可以设置一个命名空间。
  - **abstract**: 当abstract=" true" 时，表示该package为抽象包。抽象包中意味着该package不能定义Action.



# 配置ACTION

- Action的配置元素：
  - Action的**name**，即**用户请求所指向的URL**。
  - Action所对应的**class**元素，对应Action类的全限定类名。
  - 指定**result**逻辑名称和实际资源的定位关系。



# 配置ACTION

## ○ Action映射

- Action映射就是将一个请求URL（即Action的名字）映射到一个Action类，当一个请求匹配某个Action的名字时，Struts 2框架就使用这个映射来确定由该Action处理请求

## ○ Action映射的简单配置：

属性	是否必须	说明
name	是	Action的名字，用于匹配URL
class	否	Action实现类的全限定类名
method	否	执行Action类时调用的方法
convert	否	应用于action的类型转换的全限定类名



# 配置ACTION

- Action映射的配置形式

- 配置直接转发的请求

- 只定义name属性来表示要匹配的映射地址，并在子元素<result>中配置要转发的页面。如下：对于URL类似为“http://localhost:8080/StrutsDemo/index.action”的请求，页面将跳转到welcome.jsp

```
<action name="index">
```

```
  <result>welcome.jsp</result>
```

```
</action>
```



# 配置ACTION

## ○ Action映射的配置形式（续）

- 配置指定处理的Action类

- 可以使用class属性来指定要使用的Action类名，如下：定义了处理请求URL类路径为UserAction，调用其execute()方法。根据execute()方法返回的值，决定页面的跳转。Action配置可以添加多个<result>，这表示Action类可能会有多个返回结果，不同的返回结果跳转到不同的JSP页面。

```
<action name="user"  
  class="edu.hdu.javaee.struts.UserAction" >  
  <result name="success">/user.jsp</result>  
  <result name=" error" >/error.jsp</result>  
</action>
```





# 配置ACTION

## ○ Action映射的配置形式（续）

- 指定method

- 可以为同一个Action类配置不同的别名，并使用method属性指定Action调用的方法（而不是默认的execute()方法） Struts 2根据action元素的method属性查找对应请求的执行方法的过程如下：

- 1) 首先查找与method属性值完全一致的方法
- 2) 如果没有找到完全一致的方法，则查找doMethod()形式的方法
- 3) 如果仍然没有找到，则Struts 2抛出无法找到方法的异常



# 配置ACTION

## ○ Action映射的配置形式（续）

- 动态方法调用

- Struts 2中的动态方法调用，指的是无需配置method属性就可以直接调用Action中的非execute方法的方式。这种调用方法在action的名字中使用感叹号 (!) 来标识要调用的方法，如下：当请求/user!delete.action时，就会自动调用UserAction中的delete()方法

```
<action name="user"  
  class="edu.hdu.javaee.struts.UserAction">  
  <result name="success">/Methods/list.jsp</result>  
</action>
```



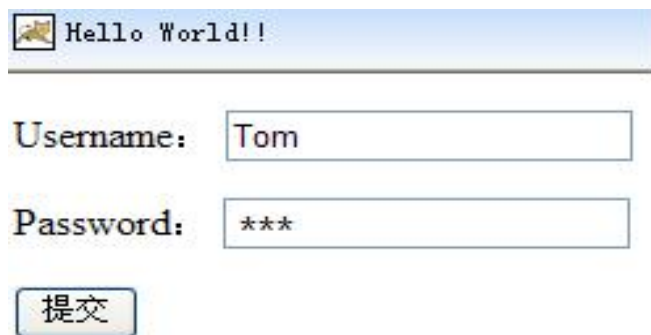
# STRUTS应用实例---升级作业

- 应用实例： 用户登录界面的示例，当用户输入用户名和密码匹配时，跳转到success页面，否则跳转到error页面。
- 处理过程：
  - 第一步：新建一个动态Web 项目并导入Struts 2的相关类库
  - 第二步：配置web.xml，在web.xml中配置Struts 2过滤器，用于拦截用户请求，启动Struts 2相应处理
  - 第三步：配置struts.xml文件
  - 第四步：编写Action
  - 第五步：编写JSP页面



# STRUTS应用实例

- 程序运行结果：



A screenshot of a web application interface. At the top, there is a blue header bar with a small icon and the text "Hello World!!". Below the header, there is a login form. The form consists of two input fields: "Username:" with the value "Tom" and "Password:" with the value "\*\*\*". Below the password field, there is a button labeled "提交" (Submit).

- 提交信息后的返回页面：



A screenshot of the return page after submission. It features a blue header bar with a small icon and the text "返回界面" (Return Interface). Below the header, the text "Welcome! Tom" is displayed.

# 思考题

- MVC的三个组成部分各自的任务是怎么样？为什么大量的WEB框架采用MVC的设计模式？
- Web容器是如何将用户的请求转给Struts 2框架来处理的？Struts 2处理请求的基本流程包括哪几步？
- 拦截器的作用是什么？
- 如何配置一个Action，以及如何设定根据Action运行结果实现页面跳转？

