

# 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配置详解

## ➤ Struts 2 的配置文件

Struts 2框架的配置文件分为内部使用和供开发人员使用二大类，内部配置文件由Struts 2框架自动加载，对其自身进行配置，其他的配置文件由开发人员使用，用于对Web应用进行配置。

文件	可选	位置(相对于webApp)	用途
web.xml	否	/WEB-INF/	Web部署描述文件
struts.xml	是	/WEB-INF/classes/	包含result映射、action映射、拦截器配置等
struts.properties	是	/WEB-INF/classes/	Struts框架中全局性变量的配置
struts-default.xml	是	/WEB-INF/lib/struts-core.jar	Struts 2提供的默认配置，由框架提供
struts-plugin.xml	是	/WEB-INF/lib/struts-xxx-plugin.jar	Struts 2框架插件所用的配置文件

# struts.properties

- **配置该文件可更改Struts 2框架定义的默认属性，开发者可以通过改变这些属性来满足应用的需求。**
- **struts.properties文件是一个标准的Properties文件，该文件包含了系列的key-value对象，每个key就是一个Struts属性，该key对应的value就是一个Struts 2属性值，如果一个key对应多个属性值时，用英文逗号分开。**

# struts.properties

- 开发struts2应用过程中，常用的常量配置信息有：
  - — **struts.i18n.encoding**：指定Web应用默认编码集。该属性对于处理中文请求参数非常有用。
  - — **struts.configuration.xml.reload**：该属性设置当struts.xml文件改变后，系统是否自动重新加载该文件。该属性的默认值是false，但在开发阶段建议设置为true，提高开发效率。

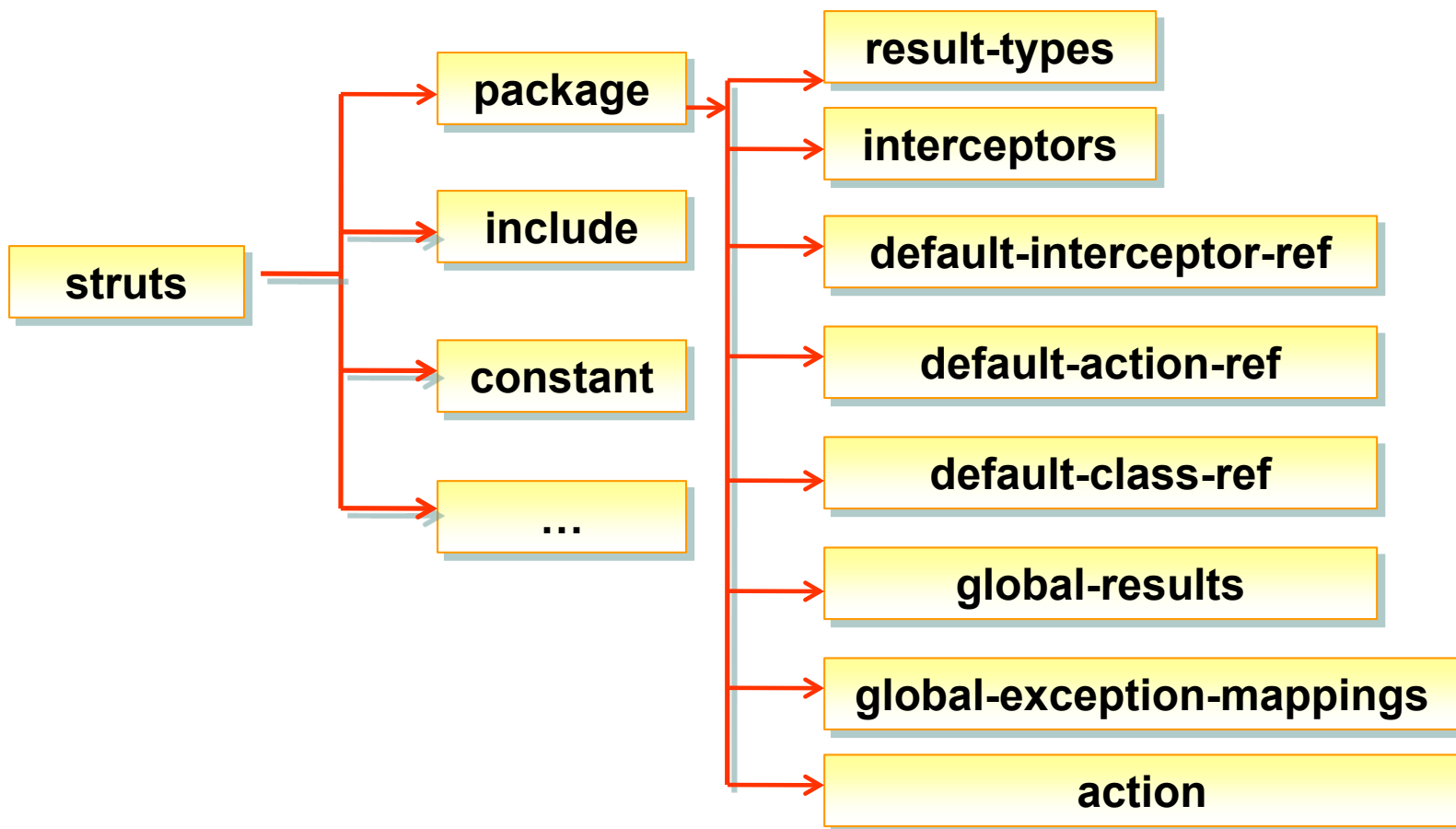
# struts.properties

- 有些时候，开发者不喜欢使用额外的struts.properties文件。所有在struts.properties文件中定义的属性，都可以在struts.xml中通过`<constant> </constant>`标签配置，或者在web.xml中通过`<init-param></init-param>`标签进行配置。
- 如struts.xml文件指定Struts 2 Web应用的默认编码集：  
`<constant name="struts.i18n.encoding" value="GBK" />`

# Struts 2配置详解

## ➤ struts.xml

**struts.xml是struts 2框架的核心配置文件，主要用于配置和管理开发人员编写的Action。在这个配置文件中，开发人员可以配置作用于action的拦截器、action和result的映射等。struts.xml由框架自动加载。**



# struts.xml配置详解

## ➤ 常量(Constant)的配置

通过常量的配置，可以改变struts2框架和插件的行为，从而满足不同Web应用的需要。默认地，Struts2框架按照下列文件的顺序搜索常量，越靠后的文件优先级越高，也就是说，顺序靠后的文件中的常量设置可以覆盖顺序靠前的文件中的常量设置。

- ① struts-default.xml
- ② struts-plugin.xml
- ③ struts.xml
- ④ struts.properties
- ⑤ web.xml

在struts.xml文件中配置常量，constant元素的属性如下表：

属性	是否必需	说明
name	是	常量的名字
value	是	常量的值

# struts.xml配置详解

//下面我们以struts.devMode属性设置为例，此属性是在struts.properties文件中定义的，可以通过web.xml(也可以通过指定init-param)加载，作用主要是控制当前项目所采用的模式，为true时意味着开发模式，为false时意味着部署模式。(默认为false)

```
<struts>
```

```
  <constant name="struts.devMode" value="true"/>
```

```
  ....
```

```
</struts>
```

```
<filter>
```

```
  <filter-name>struts2</filter-name>
```

```
  <filter-class>
```

```
    org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
```

```
  </filter-class>
```

```
  <init-param>
```

```
    <param-name>struts.devMode</param-name>
```

```
    <param-value>true</param-value>
```

```
  </init-param>
```

```
</filter>
```



## 常用的常量介绍 (续)

<!-- 与spring集成时, 指定由spring负责action对象创建 -->

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

<!--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-->

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

<!--上传文件的大小限制-->

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

<!-- 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.configuration.xml.reload" value="true"/>`

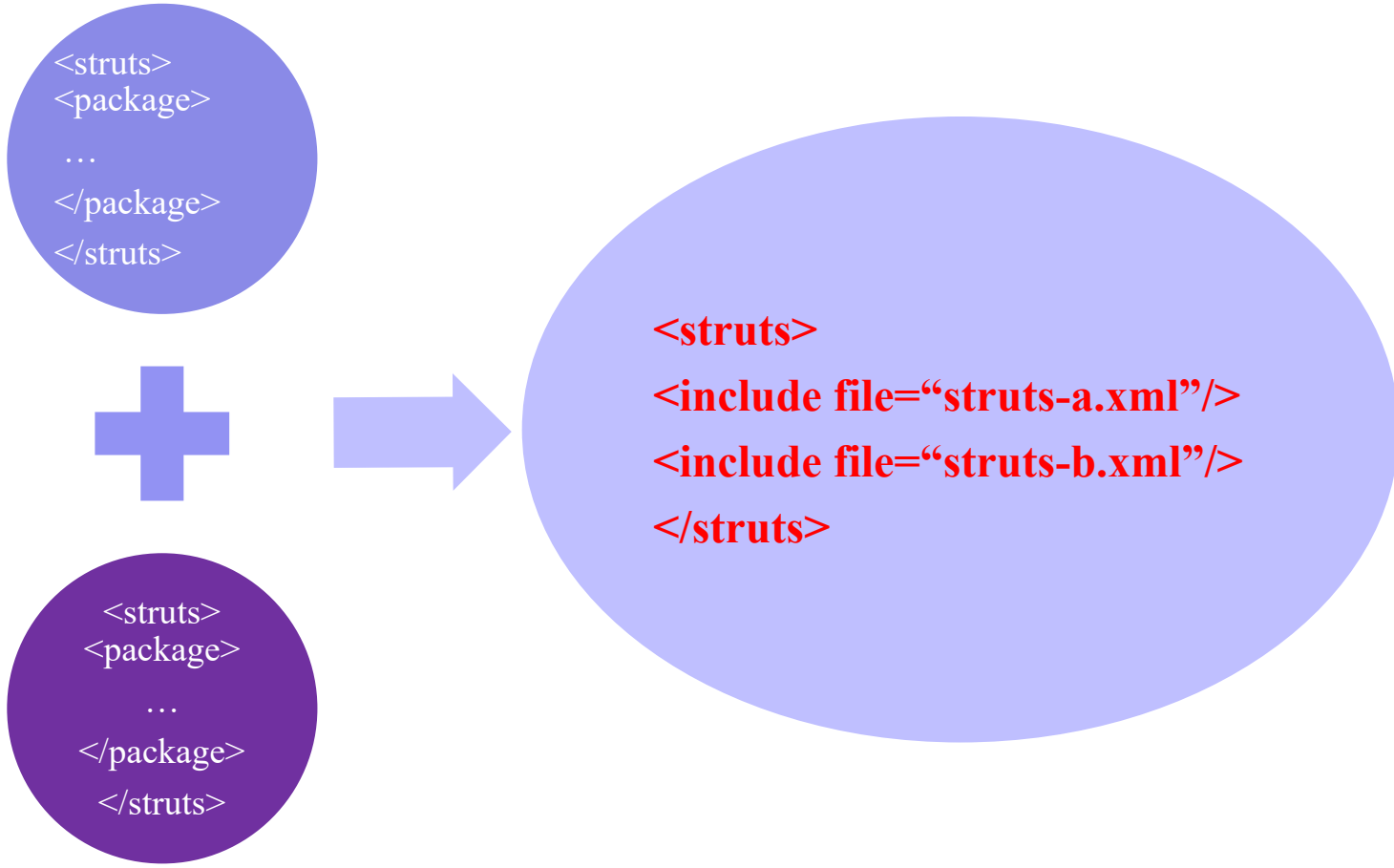
# Struts.xml中include配置

## ➤ 包含(include)配置

在Web项目中，为了降低项目的复杂度，便于团队建设成员分工协作，通常会将项目划分为多个较小的模块，每个模块单独开发和管理。

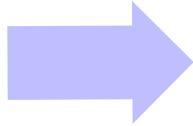
对于这种“分为治之”策略，struts2也提供了支持，我们可以采用每个模块单独一个配置文件，对其进行配置，然后在struts.xml中使用include元素来包含其他的配置文件。include只有一个必需的属性file,指定被包含文件的文件名。

# Struts.xml中include配置



The diagram illustrates the process of including external configuration files into a main Struts.xml file. On the left, two separate configuration files are shown as circles. The top circle is light blue and contains the XML snippet: `<struts>`, `<package>`, an ellipsis, `</package>`, and `</struts>`. The bottom circle is dark blue and contains the same XML snippet. A large blue plus sign is positioned between these two circles. A large blue arrow points from the plus sign to a large light blue oval on the right. This oval represents the final merged configuration, containing the XML snippet: `<struts>`, `<include file="struts-a.xml"/>`, `<include file="struts-b.xml"/>`, and `</struts>`. The included file names are in red text.

```
<struts>
<package>
...
</package>
</struts>
```



```
<struts>
<package>
...
</package>
</struts>
```

```
<struts>
<include file="struts-a.xml"/>
<include file="struts-b.xml"/>
</struts>
```

# struts.xml

**例如：某网上购物程序，用户配置、商品配置、订单配置分别放在3个配置文件user.xml, goods.xml和order.xml中，然后在struts.xml中将此3文件引入。**

```
<!-- 下面是Struts 2配置文件的根元素 -->
```

```
<struts>
```

```
    <!-- 通过include元素导入其他配置文件 -->
```

```
    <include file="user.xml" />
```

```
    <include file="goods.xml" />
```

```
    <include file="order.xml" />
```

```
    ...
```

```
</struts>
```

## ➤ 包(package)配置

Struts2中的包类似于Java中的包，提供了将action、result、拦截器和拦截器栈组织为一个逻辑单元的一种方式。与Java中的包不同，Struts 2中的包可以扩展另外的包，从而“继承”原有的包的所有定义，并可以添加自己包特有的配置，以及修改原有包部分配置。在struts.xml中使用**package**来定义包。

属性	是否必需	说明
name	是	被其他包引用的键(key)
extends	否	指定要扩展的包(如果要扩展多个包，以“,”隔开)
namespace	否	指定名称空间
abstract	否	声明包为抽象的(这样在包中不能有action的定义，比如在前面使用的struts-default包就是抽象包，可以使用抽象包定义一些默认配置，其它包选择继承，从而减少工作量)

## ➤ 包(package)配置

```
<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>  
  
<package name="test" extends="default" namespace="/accp">  
</package>
```

当发起/accp/login.action的路径请求时，会先在test包中查找是否有合适的action,而在此时是在test包存在LoginAction处理请求，因为test包继承了default包，某种意义上包的extends就相当于java中类的extends

# Struts.xml配置中的包介绍

- namespace属性:
  - 可以根据namespace的不同向服务器提交不同的package的action的请求，通常存在如下三种方式：
  - “/” 表示根namespace，所有直接在应用程序上下文环境下的请求都在这个package中查找。
  - “” 表示默认namespace，当所有的namespace中都找不到的时候就在这个namespace中寻找。
  - “/\*” 指定命名空间，\*为要设置的命名空间名称。

# Struts.xml配置说明

## ➤ 命名空间(NameSpace)配置

当Struts2接收到一个请求的时候，它会将请求URL分为namespace和action名字两个部分，然后Struts2就会从struts.xml中查询namespace/action这个命名对，如果没有找到，就会在默认的名称空间中搜索相应的action名。**默认的名称空间用空字符串(“”)来表示，当你在定义包时没有使用namespace那就指定了默认的名称空间。**

Struts2还支持以”/”命名的根名称空间，如果直接请求WEB应用上下文路径下(也就是工程根目录下)的action,那么框架就会在根名称空间查找对应的action,如果没有找到就在默认的名称空间中查找。



# Struts.xml

当发起/moo.action请求时，框架会在根名称空间("/")中查找moo action,如果没找到再到默认名称空间查找。在此例中，mypackage1中存在moo action，因此它被执行，结果转向到moo.jsp页面

**<!--default包在默认的**

```
<package name="default" extends="struts-default">
  <action name="foo" class="cn.com.web.action.LoginAction">
    <result name="success">/foo.jsp</result>
  </action>
  <action name="bar" class="cn.com.web.action.LoginAction"></action>
</package>
```

**<!--mypackage1包在根名称空间中-->**

```
<package name="mypackage1" namespace="/">
  <action name="moo" class="cn.com.web.action.LoginActionTwo">
    <result name="success">/moo.jsp</result>
  </action>
</package>
```

**<!--mypackage2包在/accp名称空间中-->**

```
<package name="mypackage2" namespace="/accp">
  <action name="foo" class="cn.com.web.action.LoginActionThree">
    <result name="success">/foo2.jsp</result>
  </action>
</package>
```

如果发起/accp/foo.action结果会导向到哪里？  
如果发起/accp/bar.action结果又是怎么样？

## ➤ Action映射

**Struts.xml文件中的action元素的通常包含的属性：**

属性	是否必需	说明
name	是	action的名字，用于匹配请求URL
class	否	Action实现类的完整类名
method	否	执行Action调用的方法

# Action配置中的各项默认值

```
<package name="itcast" namespace="/test" extends="struts-  
    default">  
    <action name="helloworld"  
        class="cn.itcast.action.HelloWorldAction"  
        method="execute" >  
        <result name="success">/WEB-INF/page/hello.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。
- 3、如果没有指定result的name属性，默认值为success。

# struts.xml中各项默认值

## ➤ <default-class-ref>

- 如果没有为某个Action指定具体的class值时，系统自动引用<default-class-ref>所指的类，默认为ActionSupport。
- 手动指定默认的class后，Struts2原来默认class被覆盖。指定类必须包含execute方法，否则出错。

## ➤ <default-action-ref>

- 通常情况下，如果请求的Action不存在，Struts2框架会返回一个Error画面：“404 - Page not found”，有些时候或许我们不想出现一个控制之外的错误画面，我们可以指定一个默认的Action，在请求的Action不存在的情况下，调用默认Action。

# Struts.xml配置

## ➤ 使用method属性

在前面我们讲述了在struts2中可以使用一个普通的java类来作为action类，而且在此类中的方法没有特别要求，对于此点struts2中是如何处理的呢？

例如：在一个新闻发布系统中，对新闻有四种操作：添加，修改，删除，查询。在具体实现中，为了节省action类的数量，通常是在一个action类中编写4个方法来实现CRUD操作。

```
public class CrudAction{  
    public String addNews(){return SUCCESS;}  
    public String deleteNews(){return SUCCESS; }  
    public String updateNews(){return SUCCESS;}  
    public String selectNews(){return SUCCESS;}  
    public String execute(){return SUCCESS;}  
}
```

现在问题是，我们如何才能让框架在不同的请求到来时，去调用CrudAction中的相应方法呢！在执行actoin时，默认调用的方法是execute()方法

# Struts.xml配置

```
<package name="default" extends="struts-default">
    <!-- 请求/list时, 调用CrudAction类上的selectNews方法-->
    <action name="list" class="cn.com.CrudAction" method="selectNews">
        <result>/list.jsp</result>
    </action>
    <action name="create" class="cn.com.CrudAction" method="addNews">
        <result>/create.jsp</result>
    </action>
    <action name="edit" class="cn.com.CrudAction" method="updateNews">
        <result>/edit.jsp</result>
    </action>
    <action name="delete" class="cn.com.CrudAction" method="deleteNews">
        <result>/delete.jsp</result>
    </action>
    <!--请求/other时, 会调用什么方法?-->
    <action name="other" class="cn.com.CrudAction">
        <result>/edit.jsp</result>
    </action>
</package>
```