JAVA资格认证:JSP中Action属性的功能浅析Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/559/2021_2022_JAVA_E8_B5_84_E6_A0_BC_c104_559608.htm 1. 完整的action action path="/aFullAction"type="somePackage.someActionClass"name="someForm"input="someJSP.jsp"forward name="successful" path="someJSP.jsp"/forward name="failed" path="someOtherJSP.jsp"//action 首先，Struts的ActionServlet接收到一个请求，然后根据struts-config.XML的配置定位到相应的mapping（映射）；接下来假如form的范围是request或在定义的范围中很难找到这个form，创建一个新的form实例；取得form实例以后，调用其 reset ()方法，然后将表单中的参数放入form，假如validate属性不为false，调用validate()方法；假如validate()返回非空的ActionErrors，将会被转到input属性指定的URI，假如返回空的ActionErrors，那么执行Action的execute() 方法，根据返回的ActionForward确定目标URI。这样做的效果是：execute()仅当validate()成功以后才执行；input属性指定的是个URI。 2. 仅有Form的action action path="/aFormOnlyAction"type="org.apache.struts.actions.ForwardAction"name="someForm"input="someJSP.jsp"parameter="someOtherJSP.jsp"/&gt. 首先，Struts会在定义的scope搜寻someForm，假如找到则重用，假如很难找到则新建一个实例；取得form实例以后，调用其 reset()方法，然后将表单中的参数放入form，假如validate属性不为false，调用validate()方法；假如validate()返回非空的 ActionErrors，将会被转到input属性指定的URI，假如返回空的ActionErrors，那么转到parameter属性

指定的目标 URI。 这样做的效果是：没有action类能够存放我们的业务逻辑，所以任何需要写入的逻辑都只能写到form的reset()或 validate()方法中。validate()的作用是验证和访问业务层。因为这里的action映射不包括forward（也没有意义），所以不能重定向，只能用默认的那个forward。这种仅有form的action能够用来处理数据获取并forward到另一个JSP来显示。 3. 仅有Action的action action path="/anActionOnlyAction"type="somePackage.someActionClass "input="someJSP.jsp"forward name="successful" path="someJSP.jsp"/forward name="failed" path="someOtherJSP.jsp"//action 首先，ActionServlet接收到请求后，取得action类实例，调用execute()方法；然后根据返回的ActionForward在配置中找forward，forward到指定的URI或action。 这样做的效果是：没有form实例被传入execute()方法，于是execute()必须自己从请求中获取参数。Action能够被forward或重定向。这种action不能处理通过HTML FORM提交的请求，只能处理链接式的请求。 4. 仅有JSP的action action path="/aJSPOnlyAction"type="org.apache.struts.actions.ForwardAction"parameter="someOtherJSP.jsp"/ 首先，ActionServlet接到请求后调用ForwardAction的execute()方法，execute()根据配置的parameter属性值来forward到那个URI。 这样做的效果是：没有任何form被实例化，比较现实的情形可能是form在request更高级别的范围中定义；或这个action被用作在应用程式编译好后充当系统参数，只需要更改这个配置文档而无需重新编译系统。 5. 两个action对应一个form action path="/anAction"type="somePackage.someActionClass"name="so

meForm"input="someJSP.jsp"forward name="successful" path="/anotherAction.do"//actionaction path="/anotherAction"type="somePackage.someOtherActionClass" name="someForm"input="someOtherJSP.jsp"forward name="successful" path="someResultJSP.jsp"//action 就每个单独的action来讲，处理上并没有和完整的action有什么实质的区分。这个组合模式能够被用来传递命令对象（form）。需要注意的是在后一个action中同样会调用form的reset()和validate()方法，因此我们必须确保form中的信息不被重写。 处理的方式大致分为两种： a) 在request中放入一个指示器表明前一个action有意向后一个action传递form，从而在后一个action能够保留那个form中的值，这一方式只能在使用forward时使用。b) 当使用redirect而不是forward时，能够把指示器放在session或更高的级别，在命令链的最后一环将这个指示器清除。 action path="/anAction"type="somePackage.someActionClass"name="someForm"input="someJSP.jsp"forward name="successful" path="/anotherAction.do" redirect="true"//actionaction path="/anotherAction"type="somePackage.someOtherActionClass" "name="someOtherForm"input="someOtherJSP.jsp"forward name="successful" path="someResultJSP.jsp"//action 这个组合方式跟前一种在流程上没有太大区分，只是我们现在对于两个action分别提供了form，于是代码看上去更加清楚。于是我们能够分别处理Web 应用程式的输入和输出。值得注意的是，后一个action同样会尝试往form中写入那些参数，但是我们能够这样处理： a） 在后一个form中使用另一套属性名； b）

只提供getter而不提供setter. 大致的处理是这样： 前一个action接收输入、验证、然后将数据写入业务层或持久层，重定向到后一个action，后一个action手动的从业务层/持久层取出数据，写入form（通过其他方式），交给前台JSP显示。 这样做的好处是不必保留输入form中的值，因此能够使用redirect而不是forward.这样就降低了两个action之间的耦合度，同时也避免了不必要的重复提交。 