PREV Table of Contents NEXT

The Struts Configuration File Tags

Table 16-1 lists and describes each of the tags used to configure the Struts configuration file.

Table 16-1: Struts Configuration File Tags

Tag	Description
action	Maps an application URL either to an **Action** object that will be executed when the specified URL is requested or to another URL that will be forwarded to.
action-mappings	Encapsulates the set of actions the application will have.
controller	Defines several global configuration settings for a Struts application.
data-source	Defines a Java data source that an application can use to access a database or similar resource.
data-sources	Encapsulates the set of data sources the application will have.
exception	Defines an exception handler to process a specific exception thrown by an **Action**.
form-bean	Defines a Form Bean and assigns a logical name to it.
form-beans	Encapsulates the set of Form Beans the application will have.
form-property	Defines a form property for dynamic Form Beans.
forward	Defines a logical name for a URL, thus allowing code to reference the logical name and not the URL itself.
global-exceptions	Encapsulates a set of exception handlers, defined by **exception** tags, which are global to the application.
global-forwards	Encapsulates a set of forwards, defined by **forward** tags, which are global to the application.
message-resources	Defines a resource bundle that Struts will use when looking up externalized strings, messages, and labels.
plug-in	Defines a plugin that Struts loads at application startup and unloads at application shutdown.
set-property	Defines a property and its value.
struts-config	Is the root tag for the Struts configuration file and thus encapsulates all other tags in the file.

The remainder of this chapter discusses each tag in detail, including a complete description of the tag, the tag's DTD definition, a table that lists each of the tag's attributes (if the tag has attributes), and a usage example for the tag. In the tables that describe each tag's attributes, pay special attention to the Required column, which denotes whether or not the given attribute is required when using the tag. If an attribute is required and you do not specify it when using the tag, the Struts framework will not be properly configured and, consequently, will not function properly.

The action Tag

The **action** tag is used to map an application URL to an **Action** object that will be executed when the specified URL is requested or to map the URL to another URL to forward the request to.

There are three ways you can use the **action** tag, as listed here and shown later in the "Example Usage" section.

- You can use the **type** attribute to map an **org.apache.struts.action.Action** subclass to the application URL specified by the **path** attribute.
- You can use the **forward** attribute to specify a URL to forward to with a call to **RequestDispatcher.forward( )** when the URL specified by the **path** attribute is matched.
- You can use the **include** attribute to specify a URL to forward to with a call to **RequestDispatcher.include( )** when the URL specified by the **path** attribute is matched.

DTD Definition

Following is the definition for the **action** tag from the Struts configuration file DTD:

```
<!ELEMENT action (icon?, display-name?, description?, set-property*, exception*,
forward*)>
```

Attributes

Attribute	Description	Required
attribute	Specifies the name of the request- or session-scope attribute under which the Form Bean associated with this action is stored. Normally the name specified with the **name** attribute is used to look up the Form Bean; however, if this attribute is used, it will be used instead.This attribute is only valid if the **name** attribute is specified.	No
className	Specifies the fully qualified class name of the configuration object to instantiate for this **Action** definition.Defaults to **org.apache.struts.config.ActionConfig**.	No
forward	Specifies a module-relative URL to forward to when this **Action** mappings path is matched. Uses **RequestDispatcher.forward( )** to perform forward.Only this attribute or one of the **include** or **type** attributes can be specified at a time.	No
include	Specifies a module-relative URL to forward to when this **Action** mappings path is matched. Uses **RequestDispatcher.include( )** to perform forward.Only this attribute or one of the **forward** or **type** attributes can be specified at a time.	No
input	Specifies a module-relative URL to which control will be forwarded if the Form Bean associated with this **Action** is set to be validated and the validation process fails.This attribute is only valid if the **name** attribute is specified.	No
name	Specifies the logical name of a Form Bean, defined with a **form-bean** tag, which will be associated with this action.	No
path	Specifies the module-relative URL to map to.	Yes
parameter	Specifies a value that will be passed as a general-purpose configuration parameter to the **Action** object defined by the **type** attribute upon each execution of the action.	No
prefix	Specifies the prefix to add to request parameter names when populating this **Action**'s associated Form Bean. Thus, a request parameter named "username" coupled with this attribute set to "search" would try to call a method called **setSearchUsername( )** on the Form Bean.This attribute is only valid if the **name** attribute is specified.	No
roles	Specifies a comma-delimited list of security roles that are allowed to access this **Action**.	No
scope	Specifies the scope ("request" or "session") that will be used to access the Form Bean associated with this action with the **name** attribute.This attribute is only valid if the **name** attribute is specified.	No
suffix	Specifies the suffix to add to request parameter names when populating this **Action**'s associated Form Bean. Thus, a request parameter named "username" coupled with this attribute set to "search" would try to call a method called **setUsernameSearch( )** on the Form Bean.This attribute is only valid if the **name** attribute is specified.	No
type	Specifies the fully qualified class name for the **org.apache.struts.action.Action** subclass to associate with this **Action** mapping.Only this attribute or one of the **forward** or **include** attributes can be specified at a time.	No
unknown	Accepts "true" or "false" to denote whether or not this action will be set as the default action for the application. If set to "true", any application URLs that don't match another mapping will be handled by this action definition.Only one **action** definition per module configuration should have this set to true.Defaults to "false".	No

validate	Accepts “true” or “false” to denote whether or not the Form Bean specified by the **name** attribute will have its **validate( )** method invoked before this **Action** is executed.This attribute is only valid if the **name** attribute is specified.Defaults to “true”.	No

Example Usage

As mentioned, there are three ways you can use the **action** tag. The first way, shown here, defines a global exception handler:

```
<action-mappings>
  <action path="/search"
          type="com.xyzcorp.app.SearchAction"/>
</action-mappings>
```

This example uses the **type** attribute to specify an **Action** object that will be executed when the specified path is matched.

The second way to use the **action** tag is shown here:

```
<action-mappings>
  <action path="/search"
       forward="/search.jsp"/>
</action-mappings>
```

This example uses the **forward** attribute to specify that the “/search.jsp” URL will be forwarded by using **RequestDispatcher.forward( )** when the specified path is matched.

The following snippet shows the third general way to use the **action** tag:

```
<action-mappings>
  <action path="/search"
       include="/search.jsp"/>
</action-mappings>
```

This example uses the **include** attribute to specify that the “/search.jsp” URL will be forwarded by using **RequestDispatcher.include( )** when the specified path is matched.

The action-mappings Tag

The **action-mappings** tag is used to encapsulate the set of actions the application will have. This tag is simply a container for **action** tags.

DTD Definition

Following is the definition for the **action-mappings** tag from the Struts configuration file DTD:

```
<!ELEMENT action-mappings (action*)>
```

Attribute

Attribute	Description	Required
type	DeprecatedUse the **action** tag’s **className** attribute instead.	No

Example Usage

The following example illustrates how to use the **action-mappings** tag:

```
<action-mappings>
  <action path="/search"
          type="com.xyzcorp.app.SearchAction"/>
</action-mappings>
```

The controller Tag

The **controller** tag is used to define several global configuration settings for your Struts application. In earlier versions of Struts, many of these settings were configured via the Web deployment descriptor **web.xml** by specifying initialization parameters for Struts’ **ActionServlet**. However, when Struts added support for application modules in version 1.1, these settings were moved to the Struts configuration file so that they could be configured on a per-module basis.

DTD Definition

Following is the definition for the **controller** tag from the Struts configuration file DTD:

```
<!ELEMENT controller (set-property*)>
```

Attributes

Attribute	Description	Required

bufferSize	Specifies the input buffer size that will be used for file uploads.Defaults to 4096 bytes.	No
className	Specifies the fully qualified class name of the configuration object to instantiate for this controller definition.Defaults to **org.apache.struts.config.ControllerConfig**.	No
contentType	Specifies the content type (and optional character encoding) that will be set on each HTTP response. Note that this setting can be overridden by an **Action**, JSP, or similar resource that a request is forwarded to.Defaults to "text/html".	No
debug	DeprecatedConfigure your underlying logging library instead.	No
forwardPattern	Specifies the pattern for how the **path** attribute of **forward** tags is mapped to URLs.$M Replaced by this module's prefix.$P Replaced by the **path** attribute of the selected forward.$$ Causes a literal dollar sign ($) to be used.All other $*x* variables, where *x* is variable, are reserved for future use and will be silently ignored.Defaults to "MP".	No
inputForward	Accepts "true" or "false" to denote whether or not the **action** tag's **input** attribute will be treated as the name of a forward whose path will be used. If set to "false" (the default), the **action** tag's **input** attribute will be taken as the literal path.Defaults to "false".	No
locale	Accepts "true" or "false" to denote whether or not a **java.util.Locale** object will be stored in users' sessions.Defaults to "true".	No
maxFileSize	Specifies the maximum size, in bytes, for file uploads. Alternatively, if you add "K", "M", or "G" to the end of the value, it will be interpreted as kilobytes, megabytes, or gigabytes, respectively.Defaults to "250M".	No
memFileSize	Specifies the maximum size in bytes for file uploads that will be kept in memory. Alternatively, if you add "K", "M", or "G" to the end of the value, it will be interpreted as kilobytes, megabytes, or gigabytes, respectively. Files larger than this threshold will be written to disk.Defaults to "256K".	No
multipartClass	Specifies the fully qualified class name of the object to use for handling file uploads.Defaults to **org.apache.struts.upload.CommonsMultipartRequestHandler**.	No
nocache	Accepts "true" or "false" to denote whether or not HTTP headers will be added to each response to disable browser caching.Defaults to "false".	No
pagePattern	Specifies the pattern for how the **page** attribute of Struts' tag library tags is mapped to URLs.$M Replaced by this module's prefix.$P Replaced by the **page** attribute of the selected tag.$$ Causes a literal dollar sign ($) to be used.All other $*x* variables, where *x* is variable, are reserved for future use and will be silently ignored.Defaults to "MP".	No
processorClass	Specifies the fully qualified class name for the **RequestProcessor** subclass that will be used for this module.Defaults to **org.apache.struts.action.RequestProcessor**.	No
tempDir	Specifies a temporary directory to use to store data during file uploads.	No

Example Usage

There are many different ways that the **controller** tag can be used because it has several attributes, all of which are optional. Following is an example usage that specifies the maximum file size for file uploads:

```
<controller maxFileSize="3M"/>
```

This example sets the maximum size for file uploads to three megabytes.

The data-source Tag

The **data-source** tag is used to define a Java data source that your application can use to access a database or similar resource. The use of nested **set-property** tags configures the data source.

DTD Definition

Following is the definition for the **data-source** tag from the Struts configuration file DTD:

```
<!ELEMENT data-source (set-property*)>
```

Attributes

Attribute	Description	Required
className	Specifies the fully qualified class name for the configuration object to instantiate for this data source definition.Defaults to **org.apache.struts.config.DataSourceConfig**.	No
key	Specifies the servlet context attribute key under which this data source will be stored. If using application modules, the module prefix will be appended to the key (e.g., "${key}${prefix}").Defaults to the value specified by the constant **org.apache.struts.Globals.DATA_SOURCE_KEY**.	No
type	Specifies the fully qualified class name for the object that will be instantiated for this data source. Must implement **javax.sql.DataSource**.	No

Example Usage

The following example illustrates the basic usage of the **data-source** tag:

```
<data-sources>
  <data-source>
    <set-property property="driverClass"
                  value="org.postgresql.Driver"/>
    …
  </data-source>
</data-sources>
```

Typically, you will not have to set any attributes on the **data-source** tag itself unless you are defining multiple data sources. All you have to do is configure your data source with nested **set-property** tags. The **set-property** tags will be used to invoke setter methods on the data source when it is initialized.

The data-sources Tag

The **data-sources** tag is used to encapsulate the set of Data Sources the application will have. This tag is simply a container for **data-source** tags.

DTD Definition

Following is the definition for the **data-sources** tag from the Struts configuration file DTD:

```
<!ELEMENT data-sources (data-source*)>
```

Example Usage

The following example illustrates how to use the **data-sources** tag:

```
<data-sources>
  <data-source>
    <set-property property="driverClass"
                  value="org.postgresql.Driver"/>
    …
  </data-source>
</data-sources>
```

The exception Tag

The **exception** tag is used to define an exception handler to process a specific exception thrown by an **Action**. This feature allows you to assign a different handler to each type of exception that is thrown by actions.

There are two ways you can use the **exception** tag, as listed here and shown later in the "Example Usage" section:

- You can define global exception handlers by placing the **exception** tags inside the **global-exceptions** tag. Global exception handlers apply to all actions.
- You can define action-specific exception handlers by nesting **exception** tags underneath an **action** tag. Action-specific exception handlers can only be "seen" by the enclosing action and will override any global exception handlers with the same target exception.

DTD Definition

Following is the definition for the **exception** tag from the Struts configuration file DTD:

```
<!ELEMENT exception (icon?, display-name?, description?, set-property*)>
```

Attributes

Attribute	Description	Required
bundle	Specifies the servlet context attribute key for a resource bundle, defined with the **message-resources** tag, which will be used to retrieve the message for the key specified by the **key** attribute.	No
className	Specifies the fully qualified class name for the configuration object to instantiate for this exception definition.Defaults to **org.apache.struts.config.ExceptionConfig**.	No
handler	Specifies the fully qualified class name for this exception handler.	No
key	Specifies the resource bundle message key to use with this handler.	Yes
path	Specifies the module-relative URL to redirect to if this exception handler is triggered.	No
scope	Specifies the scope ("request" or "session") that will be used to access the **org.apache.struts.action.ActionError** object for this exception.	No
type	Specifies the fully qualified class name of the exception class that this handler is for.	Yes

Example Usage

As mentioned, there are two ways you can use the **exception** tag. The first way, shown here, defines a global exception handler:

```
<global-exceptions>
  <exception type="com.xyzcorp.app.DateFormatException"
              key="errors.date.format"
             path="/error.jsp"/>
</global-exceptions>
```

The following is the second way to use the **exception** tag:

```
<action-mappings>
  <action path="/search"
          type="com.xyzcorp.app.SearchAction">
    <exception type="com.xyzcorp.app.DateFormatException"
                key="errors.date.format"
               path="/searchError.jsp"/>
  </action>
</action-mappings>
```

This example defines an action-specific exception handler. Only the enclosing action can "see" this exception handler and it will override a global exception handler targeted at the same exception.

The form-bean Tag

The **form-bean** tag is used to define a Form Bean and assign a logical name to it. Struts uses Form Beans to capture form data when a form is submitted and to populate forms before being displayed.

There are two ways you can use the **form-bean** tag, as listed here and shown later in the "Example Usage" section:

- You can define a concrete Form Bean by specifying its concrete class with the **type** attribute. This requires creating a class that subclasses **org.apache.struts.action.ActionForm** and creating all the getter and setter methods.
- You can define a dynamic Form Bean by using the **type** attribute to specify that its type is **org.apache.struts.action.DynaActionForm** or a subclass thereof. With this approach, you specify each of the Form Bean's fields in the configuration file with **form-property** tags.

DTD Definition

Following is the definition for the **form-bean** tag from the Struts configuration file DTD:

```
<!ELEMENT form-bean (icon?, display-name?, description?, set-property*, form-
property*)>
```

Attributes

Attribute	Description	Required

className	Specifies the fully qualified class name of the configuration object to instantiate for this Form Bean definition.Defaults to **org.apache.struts.config.FormBeanConfig**.	No
dynamic	DeprecatedThis is now determined based on whether or not the class specified with the **type** attribute is a subclass of **org.apache.struts.action.DynaActionForm**.	No
name	Specifies the logical name for the Form Bean.	Yes
type	Specifies the fully qualified class name for the Form Bean class.	Yes

Example Usage

As mentioned, there are two ways you can use the **form-bean** tag. The first way, shown here, defines a concrete Form Bean:

```
<form-beans>
  <form-bean name="logonForm"
             type="com.xyzcorp.app.LogonForm"/>
</form-beans>
```

The following is the second way to use the **form-bean** tag:

```
<form-beans>
  <form-bean name="logonForm"
             type="org.apache.struts.action.DynaActionForm">
    <form-property name="username"
                   type="java.lang.String"/>
    <form-property name="password"
                   type="java.lang.String"/>
  </form-bean>
</form-beans>
```

This example defines a dynamic Form Bean whose properties are specified with the nested **form-property** tags. Notice that the **type** attribute is set to **org.apache.struts.action.DynaActionForm**. This informs Struts that the Form Bean's properties are defined in the configuration file.

The form-beans Tag

The **form-beans** tag is used to encapsulate the set of Form Beans the application will have. This tag is simply a container for **form-bean** tags.

DTD Definition

Following is the definition for the **form-beans** tag from the Struts configuration file DTD:

```
<!ELEMENT form-beans (form-bean*)>
```

Attribute

Attribute	Description	Required
type	DeprecatedUse the **form-bean** tag's **className** attribute instead.	No

Example Usage

The following example illustrates how to use the **form-beans** tag:

```
<form-beans>
  <form-bean name="searchForm"
             type="com.xyzcorp.app.SearchForm"/>
</form-beans>
```

The form-property Tag

The **form-property** tag is used to define form properties for dynamic Form Beans. Dynamic Form Beans allow you to define a form's properties in the Struts configuration file instead of in a concrete class. Use of this tag will be ignored if the enclosing **form-bean** tag's **type** attribute is not **org.apache.struts.action.DynaActionForm** or a subclass of it.

DTD Definition

Following is the definition for the **form-property** tag from the Struts configuration file DTD:

```
<!ELEMENT form-beans (form-bean*)>
```

Attributes

Attribute	Description	Required
className	Specifies the fully qualified class name of the **FormPropertyConfig** subclass to use for this property.Defaults to **org.apache.struts.config.FormPropertyConfig**.	No
initial	Specifies the initial value of the property. If not specified, primitives will be initialized to 0 and objects will be initialized with their default constructor (thus, **String**s will be initialized to "").	No
name	Specifies the name of the property.	Yes
size	Specifies the size of the array to create if the **type** attribute specifies an array and the **initial** attribute is omitted.	No
type	Specifies the fully qualified class name for the property's underlying field. Optionally, [] can be appended to the type declaration to denote that the field is indexed (e.g., **java.lang.String[]**).	Yes

Example Usage

The following example illustrates the basic usage of the **form-property** tag:

```
<form-beans>
  <form-bean name="logonForm"
             type="org.apache.struts.action.DynaActionForm">
    <form-property name="username"
                   type="java.lang.String"/>
    <form-property name="password"
                   type="java.lang.String"/>
  </form-bean>
</form-beans>
```

Instead of specifying a concrete class for your Form Bean definition, you set its type to **org.apache.struts.action.DynaActionForm** or a subclass and then list each of its properties with the **form-property** tag.

The forward Tag

The **forward** tag is used to define a logical name for a URL, thus allowing code and so on to reference the logical name and not the URL itself.

There are two ways you can use the **forward** tag, as listed here and shown later in the "Example Usage" section:

- You can define global forwards by placing the **forward** tags inside the **global-forwards** tag. Global forwards are accessible by any action.
- You can define action-specific forwards by nesting **forward** tags underneath an **action** tag. Action-specific forwards can only be "seen" by the enclosing action and will override any global forwards with the same logical name.

DTD Definition

Following is the definition for the **forward** tag from the Struts configuration file DTD:

```
<!ELEMENT forward (icon?, display-name?, description?, set-property*)>
```

Attributes

Attribute	Description	Required
className	Specifies the fully qualified class name for the configuration object to instantiate for this forward definition.Defaults to **org.apache.struts.config.ForwardConfig**.	No
contextRelative	Accepts "true" or "false" to denote whether or not the URL specified with the **path** attribute will be application-relative when using application modules.Defaults to "false".	No
name	Specifies the logical name for the forward.	Yes
path	Specifies the URL for this forward.	Yes
redirect	Accepts "true" or "false" to denote whether or not an HTTP redirect will be executed for this forward's URL.Defaults to "false".	No

Example Usage

As stated, there are two ways you can use the **forward** tag. The first way, shown here, defines a global forward:

```
<global-forwards>
  <forward name="success"
           path="/success.jsp"/>
</global-forwards>
```

The second way to use the **forward** tag is shown here:

```
<action-mappings>
  <action path="/search"
          type="com.xyzcorp.app.SearchAction">
    <forward name="success"
             path="/results.jsp"/>
  </action>
</action-mappings>
```

This example defines an action-specific forward. Only the enclosing action can "see" this forward and it will override a global forward with the same logical name if present.

The global-exceptions Tag

The **global-exceptions** tag is used to encapsulate a set of exception handlers, defined by **exception** tags, which are global to the application. This set of global exception handlers will be used to handle any exceptions being thrown from actions unless an action's corresponding **action** tag overrides one or more of the global exception handlers by nesting **exception** tags underneath it.

DTD Definition

Following is the definition for the **global-exceptions** tag from the Struts configuration file DTD:

```
<!ELEMENT global-exceptions (exception*)>
```

Example Usage

The following example illustrates how to use the **global-exceptions** tag:

```
<global-exceptions>
  <exception type="com.xyzcorp.app.DateFormatException"
             key="errors.date.format"
            path="/error.jsp"/>
</global-exceptions>
```

This tag simply encapsulates the set of exception handlers that are global to the application.

The global-forwards Tag

The **global-forwards** tag is used to encapsulate a set of forwards, defined by **forward** tags, which are global to the application. Unless an action defines a forward with a nested **forward** tag that overrides one or more of these global forwards, the global forward will be used to determine where to forward when an action finishes executing.

DTD Definition

Following is the definition for the **global-forwards** tag from the Struts configuration file DTD:

```
<!ELEMENT global-forwards (forward*)>
```

Attribute

Attribute	Description	Required
Type	DeprecatedUse the **forward** tag's **className** attribute instead.	No

Example Usage

The following example illustrates the basic usage of the **global-forwards** tag:

```
<global-forwards>
  <forward name="success" path="/success.jsp"/>
  <forward name="failure" path="/failure.jsp"/>
</global-forwards>
```

This tag simply encapsulates the set of forwards that will be global to the application.

The message-resources Tag

The **message-resources** tag is used to define a resource bundle that Struts will use when looking up externalized strings, messages, and labels.

DTD Definition

Following is the definition for the **message-resources** tag from the Struts configuration file DTD:

```
<!ELEMENT message-resources (set-property*)>
```

Attributes

Attribute	Description	Required
className	Specifies the fully qualified class name of the configuration object to instantiate for this message resource's definition.Defaults to **org.apache.struts.config.MessageResourcesConfig**.	No
factory	Specifies the fully qualified class name of the **MessagesResourcesFactory** subclass that will be used to create this message resource instance.Defaults to **org.apache.struts.util.PropertyMessageResourcesFactory**.	No
key	Specifies the servlet context attribute key under which this message resource instance will be stored. If using application modules, the module prefix will be appended to the key (e.g., "${key}${prefix}").Defaults to the value specified by the constant **org.apache.struts.Globals.MESSAGES_KEY**.	No
null	Accepts "true" or "false" to denote whether or not missing messages should return null.	No
parameter	Specifies a configuration parameter value that will be passed to the **createResources( )** method of the factory object specified by the **factory** attribute.	Yes

Example Usage

The following example illustrates the basic usage of the **message-resources** tag:

```
<message-resources
  parameter="com.xyzcorp.app.ApplicationResources"/>
```

This example specifies that Struts should use a file called **ApplicationResources .properties** from the **com.xyzcorp.app** package as its resource bundle. Notice that the ".properties" portion of the filename is not specified with the tag; Struts automatically appends that to the name of the file that you specify with the **parameter** attribute.

Sometimes it's useful or necessary to have more than one resource bundle. You can accomplish that by using multiple **message-resources** tags in your configuration file, as shown here:

```
<message-resources
  parameter="com.xyzcorp.app.ApplicationResources"/>

<message-resources
  parameter="com.xyzcorp.app.AlternateApplicationResources"
        key="alternate"/>
```

Each instance of the **message-resources** tag must specify a unique key with the **key** attribute to identify it, unless it's for the default resource bundle, which does not require an explicit key.

The plug-in Tag

The **plug-in** tag is used to define a plugin that Struts loads at application startup and unloads at application shutdown. Among other things, plugins are useful for loading persistent resources at application startup. Each plugin class must implement Struts' **org.apache.struts.action.PlugIn** interface. Upon application startup, the plugin's **init( )** method will be called. Upon application shutdown, the plugin's **destroy( )** method will be called.

DTD Definition

Following is the definition for the **plug-in** tag from the Struts configuration file DTD:

```
<!ELEMENT plug-in (set-property*)>
```

Attribute

Attribute	Description	Required
className	Specifies the fully qualified class name for the plugin. This class must implement the **org.apache.struts.action.PlugIn** interface.	Yes

Example Usage

The following example illustrates the usage of the **plug-in** tag:

```
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
  <set-property property="pathnames"
                  value="/WEB-INF/validator-rules.xml,
                         /WEB-INF/validation.xml"/>
</plug-in>
```

The **plug-in** tag is quite simple because it has only one attribute. In most cases, there will be nested **set-property** tags to dynamically configure the plugin. Each plugin defines its own set of properties that can be configured via **set-property** tags.

The set-property Tag

The **set-property** tag is used to define a property and its value. Several of the other Struts configuration file tags allow instances of this tag to be nested inside them to specify properties that will be set when the tags' corresponding configuration objects are instantiated. Struts uses reflection to look up and invoke a setter method (e.g., **setMyProp( )** for a property called **myProp**) on the configuration object based on the name specified with this tag's **property** attribute.

DTD Definition

Following is the definition for the **set-property** tag from the Struts configuration file DTD:

```
<!ELEMENT set-property EMPTY>
```

Attributes

Attribute	Description	Required
property	Specifies the name of the property.	Yes
value	Specifies the value of the property.	Yes

Example Usage

The following example illustrates the basic usage of the **set-property** tag:

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
  <set-property property="definitions-config"
                  value="/WEB-INF/tiles-defs.xml"/>
  <set-property property="moduleAware" value="true"/>
</plug-in>
```

At run time, when Struts parses a configuration file with a definition similar to this, it will use reflection to look up and invoke the **setDefinitionsConfig( )** and **setModuleAware( )** methods of the class specified by the **plug-in** tag's **className** attribute, passing them the defined values.

The struts-config Tag

The **struts-config** tag is the root tag for the Struts configuration file and thus encapsulates all other tags in the file. This tag has no other use than to denote the beginning and end of configuration data.

DTD Definition

Following is the definition for the **struts-config** tag from the Struts configuration file DTD:

```
<!ELEMENT struts-config (data-sources?, form-beans?, global-exceptions?, global-
forwards?, action-mappings?, controller?, message-resources*, plug-in*)>
```

Example Usage

The following snippet illustrates how to use the **struts-config** tag:

```
<struts-config>
  <form-beans>
    …
  </form-beans>
  <action-mappings>
    …
  </action-mappings>
</struts-config>
```