ActionMapper

32

Description

The ActionMapper interface provides a mapping between HTTP requests and action invocation requests and vice-versa.

When given an HttpServletRequest, the ActionMapper may return null if no action invocation request matches or it may return an {{ActionMapping}} that describes an action invocation for the framework to try.

The ActionMapper is not required to guarantee that the {{ActionMapping}} returned be a real action or otherwise ensure a valid request. Accordingly, most ActionMappers do not need to consult the Struts configuration just to determine if a request should be mapped.

Just as requests can be mapped from HTTP to an action invocation, the opposite is true as well. However, because HTTP requests (when shown in HTTP responses) must be in String form, a String is returned rather than an actual request object.

DefaultActionMapper

Default action mapper implementation, using the standard `*.[ext]` (where ext usually **action**) pattern. The extension is looked up from the Struts configuration key {<b>{struts.action.extension}}.

To help with dealing with buttons and other related requirements, this mapper (and other {{ActionMapper}}s, we hope) has the ability to name a button with some predefined prefix and have that button name alter the execution behaviour. The four prefixes are:

- Method prefix - `method:default`
- Action prefix - `action:dashboard`

In addition to these four prefixes, this mapper also understands the action naming pattern of `foo!bar` in either the extension form (eg: `foo!bar.action`) or in the prefix form (eg: `action:foo!bar`). This syntax tells this mapper to map to the action named `foo` and the method `bar`.

Method prefix

With method-prefix, instead of calling baz action's `execute()` method (by default if it isn't overridden in `struts.xml` to be something else), the baz action's `anotherMethod()` will be called. A very elegant way determine which button is clicked. Alternatively, one would have submit button set a particular value on the action when clicked, and the `execute()` method decides on what to do with the setted value depending on which button is clicked.

xml<!-- START SNIPPET: method-example --> <s:form action="baz"> <s:textfield label="Enter your name" name="person.name"/> <s:submit value=" Create person"/> <s:submit method="anotherMethod" value="Cancel"/> </s:form> <!-- END SNIPPET: method-example -->

Action prefix

With action-prefix, instead of executing baz action's `execute()` method (by default if it isn't overridden in struts.xml to be something else), the anotherAction action's `execute()` method (assuming again if it isn't overridden with something else in `struts.xml`) will be executed.

xml<!-- START SNIPPET: action-example --> <s:form action="baz"> <s:textfield label="Enter your name" name="person.name"/> <s:submit value="Create person"/> <s:submit action="anotherAction" value="Cancel"/> </s:form> <!-- END SNIPPET: action-example -->

Allowed action name RegEx

By default the mapper will check if extracted action name matches provided RegEx, i.e. `[a-zA-Z0-9._!/\-]*`. You redefine this RegEx by defining a constant in `struts.xml` named `struts.allowed.action.names`. If action name doesn't match the RegEx a default action name will be returned which is defined as `index`. You can also redefine this by specifying constant `struts.default.action.name` in `struts.xml`

Allowed method name RegEx

The same logic as above is used for extracted methods, the default RegEx (`[a-zA-Z_]*[0-9]*`) is used to check if method is allowed, you can change this by setting constant `struts.allowed.method.names` in `struts.xml`. If method doesn't match the RegEx a default method is returned, i.e. `execut`. This can be changed by defining constant `struts.default.method.name` in `struts.xml`.

Please note that this funcionallity only works when Dynamic Method Invocation is enabled.

Custom ActionMapper

You can define your own ActionMapper by implementing `org.apache.struts2.dispatcher.mapper.ActionMapper` then configuring Struts 2 to use the new class in struts.xml

xml<bean type="org.apache.struts2.dispatcher.mapper.ActionMapper" name="mymapper" class="com.mycompany.myapp.MyActionMapper" /> <constant name="struts.mapper.class" value="mymapper" />

Possible uses of the ActionMapper include defining your own, cleaner namespaces, such as URLs like **/person/1**, which would be similar to a request to **/getPerson.action?personID=1** using the DefaultActionMapper.

CompositeActionMapper

A composite action mapper that is capable of delegating to a series of ActionMapper if the former failed to obtained a valid ActionMapping or uri.

It is configured through struts.xml. For example, with the following entries in struts.xml

xml<constant name="struts.mapper.class" value="composite" /> <constant name="struts.mapper.composite" value="struts,restful,restful2" />

When `CompositeActionMapper#getMapping(HttpServletRequest, ConfigurationManager)` or `CompositeActionMapper#getUriFromActionMapping(ActionMapping)` is invoked, `CompositeActionMapper` would go through these ActionMappers in sequence starting from ActionMapper identified by `struts.mapper.composite.1`, followed by `struts.mapper.composite.2` and finally `struts.mapper.composite.3` (in this case) until either one of the ActionMapper return a valid result (not null) or it runs out of ActionMapper in which case it will just return null for both `CompositeActionMapper#getMapping(HttpServletRequest, ConfigurationManager)` and `CompositeActionMapper#getUriFromActionMapping(ActionMapping)` methods.

For example with the following in struts.xml:

xml<constant name="struts.mapper.class" value="composite" /> <constant name="struts.mapper.composite" value="struts,restful" />

`CompositeActionMapper` will be configured with 2 ActionMapper, namely "struts" which is `org.apache.struts2.dispatcher.mapper.DefaultActionMapper` and "restful" which is `org.apache.struts2.dispatcher.mapper.RestfulActionMapperRestfulActionMapper`. `CompositeActionMapper` would consult each of them in order described above.

PrefixBasedActionMapper

{snippet:id=description|javadoc=true|url=org.apache.struts2.dispatcher.mapper.PrefixBasedActionMapper}

PrefixBasedActionProxyFactory

{snippet:id=description|javadoc=true|url=org.apache.struts2.factory.PrefixBasedActionProxyFactory}

ActionMapper and ActionMapping objects

The ActionMapper fetches the ActionMapping object corresponding to a given request. Essentially, the ActionMapping is a data transfer object that collects together details such as the Action class and method to execute. The mapping is utilized by the Dispatcher and various user interface components. It is customizable through `struts.mapper.class` entry in `struts.properties` or `struts.xml`. Note that the value of this constant is the name of the bean of the new mapper.

Customize

Custom ActionMapper must implement ActionMapper interface and have a default constructor.
xml<bean type="org.apache.struts2.dispatcher.mapper.ActionMapper" name="mymapper" class="com.mycompany.myapp.MyActionMapper" /> <constant name="struts.mapper.class" value="mymapper" /> javapublic class MyCustomActionMapper implements ActionMapper { public ActionMapping getMapping (HttpServletRequest request, ConfigurationManager configManager) { .... } public String getUriFromActionMapping(ActionMapping mapping) { .... } }

💡 See also: RestfulActionMapper

Next: Action Proxy & ActionProxy Factory