Action Chaining

The framework provides the ability to chain multiple actions into a defined sequence or workflow. This feature works by applying a Chain Result to a given Action, and intercepting its target Action's invocation with a Chaining Interceptor.

Don't Try This at Home
As a rule, Action Chaining is not recommended. First explore other options, such as the Redirect After Post technique.

Chain Result

The Chain Result is a result type that invokes an Action with its own Interceptor Stack and Result. This Interceptor allows an Action to forward requests to a target Action, while propagating the state of the source Action. Below is an example of how to define this sequence.

{snippet:id=example|lang=xml|javadoc=true|url=com.opensymphony.xwork2.ActionChainResult}

Another action mapping in the same namespace (or the default "" namespace) can be executed after this action mapping (see Configuration Files). An optional "namespace" parameter may also be added to specify an action in a different namespace.

Chaining Interceptor

If you need to copy the properties from your previous Actions in the chain to the current action, you should apply the Chaining Interceptor. The Interceptor will copy the original parameters from the request, and the ValueStack is passed in to the target Action. The source Action is remembered by the ValueStack, allowing the target Action to access the properties of the preceding Action(s) using the ValueStack, and also makes these properties available to the final result of the chain, such as the JSP or Velocity page.

One common use of Action chaining is to provide lookup lists (like for a dropdown list of states). Since these Actions get put on the ValueStack, their properties will be available in the view. This functionality can also be done using the ActionTag to execute an Action from the display page. You may also use the Redirect Action Result to accomplish this.

Use with care

Experience shows that chaining should be used with care. If chaining is overused, an application can turn into "spaghetti code". Actions should be treated as a Transaction Script, rather than as methods in a Business Facade. Be sure to ask yourself why you need to chain from one Action to another. Is a navigational issue, or could the logic in Action2 be pushed back to a support class or business facade so that Action1 can call it too?

Ideally, Action classes should be as short as possible. All the core logic should be pushed back to a support class or a business facade, so that Actions only call methods. Actions are best used as adapters, rather than as a class where coding logic is defined.

Next: Result Types