Writing Interceptors

See the Interceptors page for an overview of how interceptors work.

Interceptor interface

Interceptors must implement the com.opensymphony.xwork2.interceptor.Interceptor interface.

Interceptor.java

```
public interface Interceptor extends Serializable {

    void destroy();

    void init();

    String intercept(ActionInvocation invocation) throws Exception;
}
```

The *init* method is called the after interceptor is instantiated and before calling *intercept*. This is the place to allocate any resources used by the interceptor.

The *intercept* method is where the interceptor code is written. Just like an action method, *intercept* returns a result used by Struts to forward the request to another web resource. Calling *invoke* on the parameter of type ActionInvocation will execute the action (if this is the last interceptor on the stack) or another interceptor.

Keep in mind that *invoke* will return **after** the result has been called (eg. after you JSP has been rendered), making it perfect for things like open-session-in-view patterns. If you want to do something before the result gets called, you should implement a PreResultListener.

Overwrite *destroy* to release resources on application shutdown.

Thread Safety

Interceptors must be thread-safe!

A Struts 2 Action instance is created for every request and do not need to be thread-safe. Conversely, Interceptors are shared between requests and must be thread-safe.

AbstractInterceptor

The AbstractInterceptor class provides an empty implementation of *init* and *destroy*, and can be used if these methods are not going to be implemented.

Mapping

Interceptors are declared using the *interceptor* element, nested inside the *interceptors* element. Example from struts-default.xml:

```
<struts>
   ...

   <package name="struts-default">
      <interceptors>
         <interceptor name="alias" class="com.opensymphony.xwork2.interceptor.AliasInterceptor"/>
         <interceptor name="autowiring" class="com.opensymphony.xwork2.spring.interceptor.
ActionAutowiringInterceptor"/>
         ...
      </interceptors>
   </package>

   ...
</struts>
```

Example

Assuming there is an action of type "MyAction", with a setDate(Date) method, this simple interceptor will set the date of the action to the current date:

Interceptor Example

```
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;

public class SimpleInterceptor extends AbstractInterceptor {

    public String intercept(ActionInvocation invocation) throws Exception {
       MyAction action = (MyAction)invocation.getAction();
       action.setDate(new Date());
       return invocation.invoke();
    }
}
```

Next: Action Chaining