Building Your Own Interceptor

Building your own Interceptor

If none of the above interceptors suit your particular need, you will have to implement your own interceptor. Fortunately, this is an easy task to accomplish. Suppose we need an interceptor that places a greeting in the Session according to the time of the day (morning, afternoon or evening). Here's how we could implement it:

GreetingInterceptor.java:

```
package cookbook;

import java.util.Calendar;
import com.opensymphony.xwork.interceptor.Interceptor;
import org.apache.struts2.ActionInvocation;

public class GreetingInterceptor implements Interceptor {
  public void init() { }
  public void destroy() { }
  public String intercept(ActionInvocation invocation) throws Exception {
    Calendar calendar = Calendar.getInstance();
    int hour = calendar.get(Calendar.HOUR_OF_DAY);
    String greeting = (hour < 6) ? "Good evening" :
        ((hour < 12) ? "Good morning":
        ((hour < 18) ? "Good afternoon": "Good evening"));
        invocation.getInvocationContext().getSession().put("Greeting", greeting);
        String result = invocation.invoke();
        return result;
    }
}
```

struts.xml

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
        <!-- Include defaults (from Struts JAR). -->
        <include file="struts-default.xml" />

        <!-- Configuration for the default package. -->
        <package name="default" extends="webwork-default">
            <interceptors>
                <interceptor name="Greeting" class="cookbook.GreetingInterceptor" />
              </interceptors>

                <!-- Greeting  -->
            <action name="Greeting">
              <result type="velocity">Greeting.vm</result>
                <interceptor-ref name="Greeting" />
            </action>
        </package>
</struts>
```

Greeting.vm:

```
<html>
<head>
<title>Cookbook - Building Your Own Interceptor</title>
</head>
<body>

#set ($ses = $req.getSession())
<p><b>${ses.getAttribute('Greeting')}!</b></p>

</body>
</html>
```

Let's take a look at our interceptor class first. As explained before, the interceptor must implement `com.opensymphony.xwork.interceptor.Interceptor`'s methods: `init()`, called during interceptor initialization, `destroy()`, called during destruction, and most importantly, `intercept(ActionInvocation invocation)`, which is where we place the code that does the work.

Notice that our interceptor returns the result from `invocation.invoke()` which is the method responsible for executing the next interceptor in the stack or, if this is the last one, the action. This means that the interceptor has the power of short-circuiting the action invocation and return a result string without executing the action at all! Use this with caution, though.

One other thing that interceptors can do is execute code after the action has executed. To do that, just place code after the invocation.invoke() call.

The `struts.xml` configuration and the result page are pretty straightforward and require no further explanation. A custom Action is not neede for this example, so we omit the class reference, and use the default ActionSupport class.