Struts Action Invocation Framework (SAIF)

1. Overview

The Struts Action Invocation Framework looks to improve how Struts handles Actions adding features like Action interceptors and Inversion of Control (IoC). Learning from other web frameworks like WebWork 2/XWork, SAIF hopes to be a testbed for the integration of the best features of other web frameworks into Struts. SAIF currently only supports the 1.1 version of Struts and is released under the BSD license.

2. Features

- Action interceptors
- WW2-style inversion of control
- Supports both regular Struts and Tiles
- Includes modified Struts example

3. What's New

3.1. 0.1 - September 18, 2003

- Initial release

4. Download

The distribution can be found here. The minimal release doesn't contain any Struts jars or the compiled struts example war.

5. Requirements

SAIF requres Struts 1.1 or greater and is not compatible with any other plugin uses a custom request processor.

6. Usage Notes

To use SAIF, you must configure it as a Struts plugin, configure the interceptor config file, and write any interceptors you need.

6.1. Struts Plug-In Configuration

SAIF needs to be configured as a plugin in the Struts configuration file. In the plug-in configuration, the location of the interceptor configuration file needs to be defined. It should look something like this:

```
<plug-in className="net.sf.struts.saif.SAIFPlugin">
  <set-property property="interceptor-config" value="/WEB-INF/interceptor-config.xml"
</plug-in>
```

6.2. Interceptor Configuration

All interceptors are defined in `interceptor-config.xml` (of course it could have any name). This file contains the interceptor definitions and how they should be applied. There are two ways to declare interceptors for Struts Actions: globally and by Action. When the Action is requested, first any global interceptors will be applied, then Action-specific interceptors.

The following interceptors are included in SAIF:

Class	Description
`net.sf.struts.saif.ComponentInterceptor`	Performs inversion of control functionality. Sets any components the Action has defined it needs.

Table 1: Included interceptors

This is an example of an interceptor configuration file:

```
<interceptor-config>
        <interceptor name="componentInterceptor" type="net.sf.struts.saif.ComponentInte
        <interceptor name="testInterceptor" type="net.sf.struts.saif.TestInterceptor"/>

        <default-interceptors>
                <interceptor name="componentInterceptor"/>
        </default-interceptors>

        <action type="org.apache.struts.webapp.example.EditRegistrationAction">
                <interceptor name="testInterceptor"/>
        </action>
</interceptor-config>
```

6.3. Interceptor Implementation

Interceptors can perform actions before and after a Struts Action is called. To write an interceptor, simple implement the `net.sf.struts.saif.ActionInterceptor` interface and implement the `beforeAction()` and `afterAction()` methods.

This is an example of an interceptor implementation:

```
public class TestInterceptor implements ActionInterceptor
{
        public void beforeAction(Action action, ActionMapping mapping, ActionForm form,
        {
                log.debug("beforeAction called");
        }

        public void afterAction(Action action, ActionMapping mapping, ActionForm form,
        {
                log.debug("afterAction called");
        }

        private Log log = LogFactory.getLog(TestInterceptor.class);
}
```

7. Contact

Please contact Lars Hoss or Don Brown with comments, bug reports, and suggestions.