Scriptable Struts Actions Using BSF

1. Overview

This project allows Struts Actions to be written in the scripting language of one's choice rather than as Java classes. It uses the Beans Scripting Framework to allow scripts to be written in any language BSF supports like Perl, Python, Ruby, JavaScript, BeanShell, and I believe even VBScript.

As fate would have it, as I prepared the struts-example demo, idle googling found another project doing the same thing in pretty much the same way: IBM Alphaworks Struts Scripting. Unfortunately, I don't believe IBM's license is exactly open source, so perhaps there is still a need for this project. If anyone knows anything further, please, do share.

2. Features

- Implement Actions with JavaScript, BeanShell, etc.
- Supports all BSF languages
- Scripts cached in memory on first use
- Ability to pass parameters to script through Struts config

3. What's New

3.1. 0.3

- Reorganized how Struts objects are presented
- Added a configurable filter system to easily manipulate the context
- Added a filter that adds request parameters as variables
- Added ability to pass parameters to scripts in the action definition
- Added and updated Javadocs, code cleanups, documentation

3.2. 0.2

- Added the ability to configure other BSF engines

3.3. 0.1

- Working Action implementation with hardcoded BeanShell support
- Rewrote struts-example to demonstrate usage

4. Usage Notes

To determine what script will be executed, the "parameter" attribute of the action mapping should contain the name of the script relative to the web application root directory (i.e. http://server/app). In the place of the traditional Action implementation, use the value `org.twdata.struts.ScriptAction`. For example:

```
    <action    path="/logoff"
               type="org.twdata.struts.ScriptAction"
               parameter="/WEB-INF/scripts/Logoff.bsh">
      <forward name="success"              path="/index.jsp"/>
    </action>
```

In addition to specifying the script file, the "parameter" attribute can also contain parameters that will be passed to the script in the form of pre-defined variables. The format follows the URL format:

```
    parameter="/path/file.bsh?PARAMETER_NAME=PARAMETER_VALUE[&amp;PARAMETER_NAME=PARAME
```

One example of how passed parameters can be used is have one script file handle multiple actions.

Before the script completes, the next ActionForward needs to be specified. This can be done one of two ways:

1. Set `struts.forwardName` to the name of the forward
2. Set `struts.forward` to the actual ActionForward object

A number of pre-defined variables are available to the script:

- `request` - The HTTP request
- `response` - The HTTP response
- `session` - The session
- `application` - The servlet context
- `struts` - A grouping of all Struts-related objects
- `log` - A logging instance

You can add your own variables by creating a BSFManagerFilter and configuring it in struts-bsf.properties:

- `struts-bsf.filters.FILTER_NAME.class=FILTER_CLASS` - The class implementing BSFManagerFilter where FILTER_NAME is the name you are calling the filter.
- `struts-bsf.filters.FILTER_NAME.PROPERTY_NAME=PROPERTY_VALUE` - A property to be used by the filter.

To use other scripting engines other than BeanShell, create a file called

`struts-bsf.properties` and add two properties for each engine:

- `struts-bsf.engine.ENGINE_NAME.class` - The class of the BSF engine where ENGINE_NAME is the name you are calling the engine.
- `struts-bsf.engine.ENGINE_NAME.extensions` - A comma-delimited list of file extensions that will be used to identify the engine to use to execute the script.

5. Contact

Please contact Don Brown with comments, and suggestions.