Action

A Cocoon Sitemap component that manipulates runtime parameters based on the request and application state. The results of the manipulation are available to other site map components.

Actions are useful for:

- performing simple work outside of an XSP (further separation of logic)
- setting up pipelines of reusable actions (e.g. session/form validation)
- dynamic navigation (named result params accessible to sitemap)
- basic flow control (returning null means content isn't evaluated, making actions one of the CocoonConditionals)

Also seems very similar to the Model part of the standard Model 2 MVC set-up. In fact Generators (particularly XSP pages) can be used to fill the Model role for a particular request. Generators fit the role better however.

For a more apt MVC design, check out Flow Script (Flow or WhatIsFlow).

Declaring Actions

Actions are declared as follows:

```
<map:actions>
<map:action name="add-employee"
             src="org.apache.cocoon.acting.DatabaseAddAction"/>
<map:action name="locale"
             src="org.apache.cocoon.acting.LocaleAction"/>
<map:action name="request"
             src="org.apache.cocoon.acting.RequestParamAction"/>
<map:action name="form-validator"
             src="org.apache.cocoon.acting.FormValidatorAction"/>
</map:actions>
```

Built-in Actions

Cocoon has several built-in Actions

- DatabaseActions : automatic insert, update, delete of data
- ResourceExistsAction : Check to see if a resource exists. Useful for conditional pipeline processing.
- The FormValidatorAction, see FormValidationUsingCocoon
- DatabaseAuthenticationAction : can automatically authenticate a user
- HttpHeaderAction: adds HTTP headers to the response
- RequestParamAction: request parameters are made available to sitemap
- SendMailAction: sends an email
- Session* : various session related actions
- XSPActions: allow write new Actions using XSP
- ...others