Input Modules Reference (2.1 legacy document)

Table of contents

Table of contents

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. Introduction

This is meant to be a concise reference to Cocoon's InputModules, what they do, and how to use them. InputModules are available in Cocoon 2.0.4 and above. Many descriptions are taken directly from the respective modules' source code, or from trial and error experimentation.

See also: the InputModules Wiki Page.

2. Modules Reference

2.1. AbstractInputModule

AbstractInputModule gives you the infrastructure for easily deploying more InputModules.

2.2. AbstractJXPathModule

JXPathModule allows to access properties of any object in generic way. JXPath provides APIs for the traversal of graphs of JavaBeans, DOM and other types of objects using the XPath syntax. JXPathMetaModule is based on this class and duplicates the code since multiple inheritance is not possible. Please keep both classes in sync.

Configuration Example:

The following imports the class "String" as extension class to the JXPathContext using the prefix "str". Thus "str:length(xpath)" would apply the method "length" to the string object obtained from the xpath expression. Please note that the class needs to be fully qualified.

<function name="java.lang.String" prefix="str"/>

The following imports all classes in the package "java.util" as extension classes to the JXPathContext using the prefix "util". Thus "util:Date.new()" would create a new java.util.Date object.

<package name="java.util" prefix="util"/>

2.3. AbstractMetaModule

AbstractMetaModule gives you the infrastructure for easily deploying more "meta" InputModules i.e. InputModules that are composed of other InputModules. In order to get at the Logger, use getLogger().

2.4. CollectionMetaModule

Constructs an array of values suitable for a JDBC collection type from parameters obtained from another input module. Application is not limited to JDBC collections but can be used wherever similar named attributes shall be collected to an array of a given type. Currently, long, int, and string are known, more to come.

Global and Local Configuration:

TC: Finish the reference for this Module.

2.5. CookieModule

This module returns the value of the named HTTP cookie.

Example Pipeline Fragment:

<map:match pattern="foo.html"> <map:generate type="file" src="documents/{cookie:user-language}/foo.xml"/> <map:transform src="stylesheets/foo2html.xsl"/> <map:serialize/> </map:match>

2.6. DateInputModule

This module returns the current date, optionally formated as string. Format given through attribute "format" of configuration root node or nested <format/> tag on module declaration.

The 'format' attribute doesn't seem to work. Nested <format/> tags work fine.See also: Java Date Format.

2.7. GlobalInputModule

This module allows you to access "global" variables which are defined in a sitemap's pipelines definition.

cocoon.xconf usage:

<input-modules> ... <component-instance class="org.apache.cocoon.components.modules.input.GlobalInputModule" logger="core.modules.input" name="global"/> ... </input-modules>

Sitemap Usage:

<map:component-configurations> <global-variables> <doc>doc.xml</doc> </global-variables> </map:component-configurations>

Example Pipeline Fragment:

<map:match pattern="foo"> <map:generate type="file" src="documents/{global:doc}"/> <map:transform src="stylesheets/foo2html.xsl"/> <map:serialize/> </map:match>

2.8. RawRequestParameterModule

This module allows access to "raw" request parameters and their values.

Values returned are "URL Encoded", meaning if a parameter is submitted as "foo+bar", you will get the string "foo+bar".

For access to URL-Decoded request parameters, see the RequestParameterModule.

cocoon.xconf usage:

<input-modules> ... <component-instance class="org.apache.cocoon.components.modules.input.RawRequestParameterModule" logger="core.modules.input" name="raw-request-param"/> ... </input-modules>

Example Pipeline Fragment:

<map:match pattern="amazonProxy"> <map:generate type="file" src="http://localhost:8888/search?qry={raw-request-param:actor}"/> <map:serialize type="xml"/> </map:match>

This input module is useful when you are querying a remote service, and you need to be able to send a search string with spaces or other special characters intact. If you were to simply use {request-param:actor}, you would end up sending a space character to the remote service, which

would be an error, since RFC 1738 requires spaces and other special characters to be correctly encoded.

2.9. RequestParameterModule

This module allows access to request parameters. Values returned are "URL Decoded". That is, if a request parameter is submitted as foo+bar, you will get the string "foo bar".

For URL-Encoded request parameters, see the RawRequestParameterModule.

cocoon.xconf usage:

<input-modules> ... <component-instance class="org.apache.cocoon.components.modules.input.RequestParameterModule" logger="core.modules.input" name="request-param"/> ... </input-modules>

Example Pipeline Fragment:

<map:match pattern="bar"> <map:generate type="file" src="documents/{request-param:file}"/> <map:transform src="stylesheets/{request-param:stylesheet}"/> <map:serialize/> </map:match>

This pipeline will match a request for "bar" and pass the request parameters "file" and "stylesheet" to the generator and transformer, respectively. (e.g. http://localhost:8080/cocoon/bar?file=doc.xml&stylesheet=main.xsl).

Directly passing request parameters for file access can be dangerous. Remember to follow basic webapp safety rules when designing your pipelines, and when passing around request parameters.

2.10. RequestURIModule

Returns the URI of the request.

If you are familliar with Avalon and Cocoon's component architecture, the following will explain what is specifically returned:

String uri = ObjectModelHelper.getRequest(objectModel).getSitemapURI();

cocoon.xconf usage:

<input-modules> ... <component-instance class="org.apache.cocoon.components.modules.input.RequestURIModule" logger="core.modules.input" name="request-uri"/> ... </input-modules>

Example Sitemap Usage:

{request-uri:requestURI}

This is how you would use the module most of the time, since the module only returns one item, the Request URI.

The same effect can be achieved by passing the parameter name "sitemapURI" to the request module. Therefore this component is not declared in cocoon.xconf by default. You must manually add it.

1. Comments

add your comments