Advanced Control Flow (2.1 legacy document)

Table of contents

1 Comments.....18

Table of contents

1 Flowscript....5
2 Flow Object Model....5
2.1 Cocoon Object....5
2.1.1 request....5
2.1.2 response....5
2.1.3 session....5
2.1.4 context....5
2.1.5 log....6
2.1.6 parameters....6
2.1.7 sendPage....6
2.1.8 sendPageAndWait....6
2.1.9 sendStatus....6
2.1.10 createPageLocal....7
2.1.11 processPipelineTo....7
2.1.12 redirectTo....7
2.1.13 createWebContinuation....7
2.1.14 load....7
2.1.15 getComponent....8
2.1.16 releaseComponent....8
2.1.17 createObject....8
2.1.18 disposeObject....8
2.2 Request Object....8
2.2.1 get....8
2.2.2 getAttribute....8
2.2.3 getAttributeNames....8
2.2.4 setAttribute....8
2.2.5 removeAttribute....8
2.2.6 getCharacterEncoding....9
2.2.7 setCharacterEncoding....9
2.2.8 getContentLength....9
2.2.9 getContentType....9
2.2.10 getParameter....9
2.2.11 getParameterValues....9
2.2.12 getParameterNames....9
2.2.13 getAuthType....9

2.2.14 getProtocol....9
2.2.15 getScheme....10
2.2.16 getMethod....10
2.2.17 getServerName....10
2.2.18 getServerPort....10
2.2.19 getRemoteAddr....10
2.2.20 isSecure....10
2.2.21 getLocale....10
2.2.22 getLocales....10
2.2.23 getCookies....10
2.2.24 getHeader....11
2.2.25 getHeaders....11
2.2.26 getHeaderNames....11
2.2.27 getRemoteUser....11
2.2.28 getUserPrincipal....11
2.2.29 isUserInRole....11
2.2.30 Properties....11
2.3 Response Object....11
2.3.1 createCookie....11
2.3.2 addCookie....12
2.3.3 containsHeader....12
2.3.4 setHeader....12
2.3.5 addHeader....12
2.3.6 setStatus....12
2.4 Session Object....12
2.4.1 getAttribute....12
2.4.2 setAttribute....12
2.4.3 removeAttribute....12
2.4.4 invalidate....13
2.4.5 isNew....13
2.4.6 getId....13
2.4.7 getCreationTime....13
2.4.8 getLastAccessedTime....13
2.4.9 setMaxInactiveInterval....13
2.4.10 getMaxInactiveInterval....13
2.4.11 Properties....13
2.5 Context Object....14

2.5.1 getAttribute....14
2.5.2 setAttribute....14
2.5.3 removeAttribute....14
2.5.4 getInitParameter....14
2.5.5 Properties....14
2.6 Cookie Object....14
2.6.1 getName....14
2.6.2 getVersion....14
2.6.3 setVersion....14
2.6.4 getValue....15
2.6.5 setValue....15
2.6.6 getComment....15
2.6.7 setComment....15
2.6.8 getDomain....15
2.6.9 setDomain....15
2.6.10 getPath....15
2.6.11 setPath....15
2.6.12 getSecure....15
2.6.13 setSecure....16
2.7 Log Object....16
2.7.1 error....16
2.7.2 debug....16
2.7.3 warn....16
2.7.4 info....16
2.7.5 isErrorEnabled....16
2.7.6 isDebugEnabled....16
2.7.7 isWarnEnabled....16
2.7.8 isInfoEnabled....16
2.8 WebContinuation....17
2.8.1 id....17
2.8.2 continuation....17
2.8.3 previousBookmark....17
2.8.4 isBookmark....17
2.8.5 getParent....17
2.8.6 getChildren....17
2.8.7 invalidate....17

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. Flowscript

Cocoon Flowscript is a JavaScript API to manage control flow based on an extended version of the Mozilla Rhino JavaScript interpreter that supports continuations.

2. Flow Object Model

Cocoon provides a set of system objects for use by Flowscripts. We call this set of objects the *Flow Object Model* (FOM). The Flow Object Model consists of the following objects:

- Cocoon
- Request
- Response
- Session
- Context
- Cookie
- Log
- WebContinuation

2.1. Cocoon Object

The Cocoon object represents the current Cocoon Sitemap. This is the entry point into the FOM. There is one instance of Cocoon which you may access in your scripts as the global variable cocoon, for example like this:

```
var value = cocoon.request.getAttribute("blah");
```

The Cocoon object supports the following properties and functions:

2.1.1. request

The current Cocoon request:

Property [Request] request

2.1.2. response

The current Cocoon response:

Property [Response] response

2.1.3. session

The current Cocoon session:

Property [Session] session

2.1.4. context

The current Cocoon application context:

Property [Context] context

2.1.5. log

A reference to the current logger:

Property [Log] log

2.1.6. parameters

Any parameters passed to the script by the Cocoon Sitemap

Property [Object] parameters

2.1.7. sendPage

Function sendPage([String] uri, [Object] bean)

Passes control to the Cocoon sitemap to generate the output page.

uri is the sitemap URI of the page to be sent back to the client. If the URI starts with a slash, it is resolved starting at the root sitemap, otherwise it is resolved relative to the current sitemap. The URI should not contain a scheme (such as cocoon:).

bean is a context object which can be accessed inside this page to extract various values and place them in the generated page.

2.1.8. sendPageAndWait

Function [WebContinuation] sendPageAndWait([String] uri, [Object] bean, [Function] postPipelineCode, [Number] timeToLive)

Passes control to the Cocoon sitemap to generate the output page.

The flow script is suspended after the page is generated and the whole execution stack saved in the WebContinuation object returned from this function.

uri is the relative URL of the page to be sent back to the client. If the URI starts with a slash, it is resolved starting at the root sitemap, otherwise it is resolved relative to the current sitemap. The URI should not contain a scheme (such as cocoon:).

bean is a context object which can be accessed inside this page to extract various values and place them in the generated page.

If provided, the postPipelineCode function will be executed after pipeline processing is complete but before the script is suspended. You can use this to release resources that are needed during the pipeline processing but should not become part of the continuation. For example:

```
function processPage() { var id = ... var bizData = ... var uri = ... var comp = cocoon.getComponent(id); // use comp ... cocoon.sendPageAndWait(uri, bizData, function() { cocoon.releaseComponent(comp); comp = null; }); }
```

timeToLive is the time to live in milliseconds for the continuation created.

The return value is the continuation object.

2.1.9. sendStatus

Function sendStatus([Number] sc)

Sends an empty response with the provided HTTP status code.

2.1.10. createPageLocal

Function [Object] createPageLocal()

Creates a *Page Local* object. The returned object behaves like a normal JavaScript object, but restores the property values it had when sendPageAndWait was originally called, each time a page is resubmitted (e.g. after hitting the back button). Such objects can be used to associate state with one particular page sent to the browser.

2.1.11. processPipelineTo

Function processPipelineTo([String] uri, [Object] bizData, [java.io.OutputStream] stream)

Call the Cocoon sitemap for the given URI, sending the output of the eventually matched pipeline to the specified OutputStream.

uri is the URI for which the request should be generated. If the URI starts with a slash, it is resolved starting at the root sitemap, otherwise it is resolved relative to the current sitemap. The URI should not contain a scheme (such as cocoon:).

bizData is the business data object to be made available to the forwarded pipeline

stream is an OutputStream where the output should be written to.

2.1.12. redirectTo

Function redirectTo([String] uri)

Send a client-side redirect to the browser. The uri argument is the URI to which the browser should be redirected.

2.1.13. createWebContinuation

Function [WebContinuation] createWebContinuation()

Creates a new "bookmark" WebContinuation object. When invoked it will cause the script to resume right after the call. By calling this function prior to sendPageAndWait() you can be create a "bookmark" which will cause the page to be redisplayed in the browser. Note: the WebContinuation associated with sendPageAndWait() doesn't do this. Rather it resumes execution after the the page is submitted.

For example:

```
function processPage() { var bkm = cocoon.createWebContinuation(); var biz = getBizData(); cocoon.sendPageAndWait("uri", {bookmark: bkm, biz: biz}, function() { releaseData(biz); }); }
```

2.1.14. load

Function load([String] uri)

Load the JavaScript script specified by uri. The Cocoon source resolver is used to resolve uri.

2.1.15. getComponent

Function Object getComponent([String] id)

Access an Avalon component.

2.1.16. releaseComponent

Function releaseComponent([Object] component)

Release a pooled Avalon component.

2.1.17. createObject

Function createObject([JavaClass] componentClass)

Create and setup an object so that it can access the information provided to regular components. This is done by calling the various Avalon lifecycle interfaces implemented by the object.

2.1.18. disposeObject

Function disposeObject([Object] object)

Dispose an object that has been created using createObject.

2.2. Request Object

The Request object represents the current Cocoon request. It provides the following functions and properties:

2.2.1. get

Function [String] get([String] name)

Get the request parameter or attribute with the specified name.

2.2.2. getAttribute

Function [String] getAttribute([String] name)

Get the request attribute with the specified name.

2.2.3. getAttributeNames

Function [java.util.Enumeration] getAttributeNames()

Get an enumeration of request attribute names.

2.2.4. setAttribute

Function setAttribute([String] name, [Object] value)

Set the value of a request attribute.

2.2.5. removeAttribute

Function removeAttribute([String] name)

Remove the attribute with the name name from this request.

2.2.6. getCharacterEncoding

Function [String]getCharacterEncoding()

Return the character encoding used by this request.

2.2.7. setCharacterEncoding

Function setCharacterEncoding([String] value)

Set the character encoding used by this request.

2.2.8. getContentLength

Function [Number] getContentLength()

Get the content-length of this request

2.2.9. getContentType

Function [String] getContentType()

Get the content-type of this request

2.2.10. getParameter

Function [String] getParameter([String] name)

Get the request parameter with the specified name.

2.2.11. getParameterValues

Function [Array] getParameterValues([String] name)

Get an array of request parameters with the specified name.

2.2.12. getParameterNames

Function [java.util.Enumeration] getParameterNames()

Get an enumeration of the parameter names in this request.

2.2.13. getAuthType

Function [String] getAuthType()

Get the authorization type used in this request.

2.2.14. getProtocol

Function [String] getProtocol()

Get the protocol used in this request.

2.2.15. getScheme

Function [String] getScheme()

Get the scheme used in this request.

2.2.16. getMethod

Function [String] getMethod()

Get the method used in this request.

2.2.17. getServerName

Function [String] getServerName()

Get the server name of this request.

2.2.18. getServerPort

Function [Number] getServerPort()

Get the server port of this request.

2.2.19. getRemoteAddr

Function [String] getRemoteAddr()

Get the remote address of this request.

2.2.20. isSecure

Function [Boolean] isSecure()

Get the secure property of this request.

2.2.21. getLocale

Function [String] getLocale()

Get the locale of this request.

2.2.22. getLocales

Function [Array [String]] getLocales()

Get the locales of this request.

2.2.23. getCookies

Function [Array [Cookie]] getCookies()

Get the cookies associated with this request.

2.2.24. getHeader

Function [String] getHeader([String] name)

Get the header with name from this request.

2.2.25. getHeaders

Function [Array] getHeaders()

Get the headers associated with this request.

2.2.26. getHeaderNames

Function [java.util.Enumeration] getHeaderNames()

Get an enumeration of header names from this request.

2.2.27. getRemoteUser

Function [String] getRemoteUser()

Get the remote user associated with this request.

2.2.28. getUserPrincipal

Function [String] getUserPrincipal()

Get the user principal associated with this request.

2.2.29. isUserInRole

Function [Boolean] isUserInRole([String] role)

Returns whether the user associated with this request is in the specified role.

2.2.30. Properties

Request properties map to request parameters, i.e. request.blah is equivalent to request.getParameter("blah").

2.3. Response Object

The Response object represents the Cocoon response associated with the current request.

The response object contains hooks only to the cookies and to the response headers. Everything else will be controlled by the rest of the cocoon pipeline machinery (like output encoding, for example, which should *NOT* be determined by the flow).

It provides the following functions and properties:

2.3.1. createCookie

Function [Cookie] createCookie([String] name, [String] value)

Creates a new Cookie.

2.3.2. addCookie

Function addCookie([Cookie] cookie)

Adds cookie to the current response.

2.3.3. containsHeader

Function [Boolean] containsHeader([String] name)

Returns whether the current response contains a header with the specified name.

2.3.4. setHeader

Function setHeader([String] name, [String] value)

Replaces the value of the header with name with value.

2.3.5. addHeader

Function addHeader([String] name, [String] value)

Creates a new header in the current response with the specified name and value.

2.3.6. setStatus

Function setStatus([Number] sc)

Sets the status code for this response.

2.4. Session Object

The Session object represents the user session associated with the current Cocoon request.

It provides the following functions and properties:

2.4.1. getAttribute

Function [Object] getAttribute([String] name)

Get the value of the session attribute with the specified name.

2.4.2. setAttribute

Function setAttribute([String] name, [Object] value)

Set the value of the session attribute with the specified name to value.

2.4.3. removeAttribute

Function removeAttribute([String] name)

Remove the session attribute with the specified name.

2.4.4. invalidate

Function invalidate()

Invalidate this session, releasing all resources associated with it.

2.4.5. isNew

Function [Boolean] isNew()

Returns true if the client does not yet know about the session or if the client chooses not to join the session. For example, if the server used only cookie-based sessions, and the client had disabled the use of cookies, then a session would be new on each request.

2.4.6. getId

Function [String] getId()

Returns the unique id associated with this session.

2.4.7. getCreationTime

Function [Number] getCreationTime()

Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.

2.4.8. getLastAccessedTime

Function [Number] getLastAccessedTime()

Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.

2.4.9. setMaxInactiveInterval

Function setMaxInactiveInterval([Number] interval)

Specifies the time, in seconds, between client requests before the contextcontainer will invalidate this session. A negative time indicates the session should never timeout.

2.4.10. getMaxInactiveInterval

Function [Number] getMaxInactiveInterval()

Returns the maximum time interval, in seconds, that the context container will keep this session open between client accesses. After this interval, the context container will invalidate the session. The maximum time interval can be set with the setMaxInactiveInterval method. A negative time indicates the session should never timeout.

2.4.11. Properties

Session properties map to session attributes, i.e. session.blah is equivalent to session.getAttribute("blah").

2.5. Context Object

The Context object represents the client context associated with the current Cocoon request.

It provides the following functions and properties:

2.5.1. getAttribute

Function [Object] getAttribute([String] name)

Get the value of the context attribute with the specified name.

2.5.2. setAttribute

Function setAttribute([String] name, [Object] value)

Set the value of the context attribute with the specified name to value.

2.5.3. removeAttribute

Function removeAttribute([String] name)

Remove the context attribute with the specified name.

2.5.4. getInitParameter

Function getInitParameter([String] name)

Get the value of the context initialization parameter with the specified name.

2.5.5. Properties

Context properties map to context attributes, i.e. context.blah is equivalent to context.getAttribute("blah").

2.6. Cookie Object

Cookie provides the following functions and properties:

2.6.1. getName

Function [String] getName()

Get the name of this cookie.

2.6.2. getVersion

Function [String] getVersion()

Get the version of this cookie.

2.6.3. setVersion

Function setVersion([String] version)

Set the version of this cookie.

2.6.4. getValue

Function [String] getValue()

Get the value of this cookie.

2.6.5. setValue

Function setValue([String] value)

Set the value of this cookie.

2.6.6. getComment

Function [String] getComment()

Get the comment of this cookie.

2.6.7. setComment

Function setComment([String] comment)

Set the comment of this cookie.

2.6.8. getDomain

Function [String] getDomain()

Get the domain of this cookie.

2.6.9. setDomain

Function setDomain([String] domain)

Set the domain of this cookie.

2.6.10. getPath

Function [String] getPath()

Get the path of this cookie.

2.6.11. setPath

Function setPath([String] path)

Set the path of this cookie.

2.6.12. getSecure

Function [Boolean] getSecure()

Get the secure property of this cookie.

2.6.13. setSecure

Function setSecure([Boolean] value)

Set the secure property of this cookie.

2.7. Log Object

The Log object provides an interface to the Cocoon logging system.

It supports the following functions:

2.7.1. error

Function error([String] message, [java.lang.Throwable] exception)

Log an error message. If exception is provided its stack trace will also be logged.

2.7.2. debug

Function debug([String] message, [java.lang.Throwable] exception)

Log a debug message. If exception is provided its stack trace will also be logged.

2.7.3. warn

Function warn([String] message, [java.lang.Throwable] exception)

Log a warning message. If exception is provided its stack trace will also be logged.

2.7.4. info

Function info([String] message, [java.lang.Throwable] exception)

Log an information message. If exception is provided its stack trace will also be logged.

2.7.5. isErrorEnabled

Function [Boolean] isErrorEnabled()

Returns whether error message logging is enabled.

2.7.6. isDebugEnabled

Function [Boolean] isDebugEnabled()

Returns whether debug message logging is enabled.

2.7.7. isWarnEnabled

Function [Boolean] isWarnEnabled()

Returns whether warning message logging is enabled.

2.7.8. isInfoEnabled

Function [Boolean] isInfoEnabled()

Returns whether information message logging is enabled.

2.8. WebContinuation

A WebContinuation represents a continuation of a Flowscript. Because a user may click on the back button in the browser and restart a saved computation in a continuation, each WebContinuation becomes the parent of a subtree of continuations.

If there is no parent WebContinuation, the created continuation becomes the root of a tree of WebContinuations.

WebContinuation objects support the following functions and properties:

2.8.1. id

Property [String] id

Returns the unique string identifier of this Web Continuation.

2.8.2. continuation

Property [Continuation] continuation

Returns the JavaScript continuation associated with this Web Continuation.

2.8.3. previousBookmark

Property [WebContinuation] previousBookmark

Returns a reference to the first bookmark continuation among the pages preceding the one associated with this object, or null if no such bookmark continuation exists. The returned object is the continuation you would invoke to implement a "Back" button.

2.8.4. isBookmark

Function [Boolean] isBookmark()

Returns true if this continuation was *not* created by sendPageAndWait.

2.8.5. getParent

Function [WebContinuation] getParent()

Get the parent continuation of this continuation.

2.8.6. getChildren

Function [Array [WebContinuation]] getChildren()

Get the child continuations of this continuation.

2.8.7. invalidate

Function invalidate()

Invalidates a WebContinuation. This effectively means that the continuation object associated with it will no longer be accessible from Web pages. Invalidating a WebContinuation invalidates all the WebContinuations which are children of it.

1. Comments

add your comments