Description of the jx generator (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. JX Generator

(*JX* for Apache JXPath and Apache Jexl).

Uses the namespace http://apache.org/cocoon/templates/jx/1.0.

Provides a generic page template with embedded JSTL and XPath expression substitution to access data sent by Cocoon Flowscripts.

The embedded expression language allows a page author to access an object using a simplified syntax such as

```
<site signOn="${accountForm.signOn}">
```

Embedded JSTL expressions are contained in ${}.

Embedded XPath expressions are contained in #{}.

Note that since this generator uses Apache JXPath and Apache Jexl, the referenced objects may be Java Beans, DOM, JDOM, or JavaScript objects from a Flowscript. In addition the following implicit objects are available as both XPath and JSTL variables:

request (org.apache.cocoon.environment.Request)	The Cocoon current request
session (org.apache.cocoon.environment.Session)	The Cocoon session associated with the current request
context (org.apache.cocoon.environment.Context)	The Cocoon context associated with the current request
parameters (org.apache.avalon.framework.parameters.Param	A map of parameters passed to the generator in the pipeline

The current Web Continuation from the Flowscript is also available as a variable named continuation. You would typically access its id:

```
<form action="${continuation.id}">
```

You can also reach previous continuations by using the getContinuation() function:

```
<form action="${continuation.getContinuation(1).id}" >
```

The template tag defines a new template:

```
<template> body </template>
```

The import tag allows you to include another template within the current template. The content of the imported template is compiled and will be executed in place of the import tag:

```
<import uri="URI" [context="Expression"]/>
```

The Cocoon source resolver is used to resolve uri. If context is present, then its value is used as the context for evaluating the imported template, otherwise the current context is used.

The set tag creates a local alias of an object. The var attribute specifies the name of a variable to assign the object to. The value attribute specifies the object (defaults to body if not present):

```
<set var="Name" [value="Value"]> [body] </set>
```

If used within a macro definition (see below) variables created by set are only visible within the body of the macro.

The if tag allows the conditional execution of its body according to value of a test attribute:

<if test="Expression"> body </if>

The choose tag performs conditional block execution by the embedded when sub tags. It renders the body of the first when tag whose test condition evaluates to true. If none of the test conditions of nested when tags evaluate to true, then the body of an otherwise tag is evaluated, if present:

<choose> <when test="Expression"> body </when> <otherwise> body </otherwise> </choose>

The out tag evaluates an expression and outputs the result of the evaluation:

<out value="Expression"/>

The forEach tag allows you to iterate over a collection of objects:

<forEach [var="Name"] [items="Expression"] [begin="Number"] [end="Number"] [step="Number"]> body </forEach>

The items attribute specifies the list of items to iterate over. The var attribute specifies the name of a variable to hold the current item. The begin attribute specifies the element to start with (0 = first item, 1 = second item, ...). If unspecified it defaults to 0. The end attribute specifies the item to end with (0 = first item, 1 = second item, ...). If unspecified it defaults to the last item in the list. Every step items are processed (defaults to 1 if step is absent). Either items or both begin and end must be present.

The formatNumber tag is used to display numeric data, including currencies and percentages, in a locale-specific manner. The formatNumber action determines from the locale, for example, whether to use a period or a comma for delimiting the integer and decimal portions of a number. Here is its syntax:

<formatNumber value="Expression" [type="Type"] [pattern="Expression"] [currencyCode="Expression"] [currencySymbol="Expression"] [maxIntegerDigits="Expression"] [minIntegerDigits="Expression"] [maxFractionDigits="Expression"] [minFractionDigits="Expression"] [groupingUsed="Expression"] [var="Name"] [locale="Expression"]>

The formatDate tag provides facilities to format Date values:

<formatDate value="Expression" [dateStyle="Style"] [timeStyle="Style"] [pattern="Expression"] [type="Type"] [var="Name"] [locale="Expression"]>

The macro tag allows you define a new custom tag.

<macro name="Name" [targetNamespace="Namespace"]> <parameter name="Name" [optional="Boolean"] [default="Value"]/> body </macro>

For example:

<c:macro name="d"> <tr><td></td></tr> </c:macro>

The tag being defined in this example is <d> and it can be used like any other tag:

<d/>

However, when this tag is used it will be replaced with a row containing a single empty data cell.

When such a tag is used, the attributes and content of the tag become available as variables in the body of the macro's definition, for example:

<c:macro name="tablerows"> <c:parameter name="list"/> <c:parameter name="color"/>

<c:forEach var="item" items="${list}"> <tr><td bgcolor="${color}">${item}</td></tr> </c:forEach> </c:macro>

The parameter tags in the macro definition define formal parameters, which are replaced with the actual attribute values of the tag when it is used. The content of the tag is also available as a special variable ${content}.

Assuming you had this code in your flowscript:

var greatlakes = ["Superior", "Michigan", "Huron", "Erie", "Ontario"];

sendPage(uri, {greatlakes: greatlakes});

and a template like this:

<tablerows list="${greatlakes}" color="blue"/> </table>

When the tablerows tag is used in this situation the following output would be generated:

<table> <tr><td bgcolor="blue">Superior</td></tr> <tr><td bgcolor="blue">Michigan</td></tr> <tr><td bgcolor="blue">Huron</td></tr> <tr><td bgcolor="blue">Erie</td></tr> <tr><td bgcolor="blue">Ontario</td></tr> </table>

1. Comments

add your comments