Session Contexts (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 chapter describes the concept of the session context provided by the session framework (session-fw block).

The session framework provides the concept of so called (session) contexts. A session context is a data storage in the session that can contain arbitrary data in XML format.

You can define your own contexts for your application. For example, if you build a web shop, you can create a context called *shop* and store the basket of the user in it.

The *session transformer* is the main component used for storing and retrieving information from such a context.

The chapter "Special Contexts" explains some special contexts which do not require a session. They are available everytime. These special contexts are the request context and the temporary context.

2. session Transformer

The *session transformer* is responsible for interpreting the tags and performing the actions required. The session transformer is configured in the sitemap and can be used inside a pipeline. All tags must be prefixed with the session namespace. So the actual tag used in the document will read *<session:xxxx>*. The current namespace URI for the session transformer is *"http://apache.org/cocoon/session/1.0"*.

2.1. Context-Tags

A context is basically an application specific block of XML data in the users session. Each context has a unique name.

The command *createcontext* is used to create a new context. A *name* attribute must be given to the context in order to identify it. The following names may not be used: request, response, session, context, temp, portal or authentication. If a context of the same name already exists then this command will have no effect.

<createcontext name="mycontext"/>

In order to delete a context the command *deletecontext* is provided. Again a *name* attribute must be provided.

<deletecontext name="mycontext"/>

Once created, XML data can then be stored inside or read from a given context.

2.2. Accessing context data

The data in a context can be dynamically accessed using the following commands.

getxml allows data to be read out of a context. A *path* attribute describes the source of the data inside the context and consists of an XPath expression. All *path* values must be absolute and only nodes and

attributes can be accessed.

An optional default value can also be provided to allow for the nonexistence of the requested data.

<getxml context="mycontext" path="/User/Name"/>

<getxml context="mycontext" path="/User/Name">Matthew</getxml>

Attributes can also be accessed.

<getxml context="mycontext" path="/User/Name/@Age"/>

Data can be written into a context using the *setxml* command. It is possible to set values or nodes as the following examples show.

<setxml context="mycontext" path="/User/Name"/>Carsten</setxml>

<setxml context="mycontext" path="/User/"/><Name>Carsten</Name></setxml>

Using the *setxml* command causes all the nodes below the target node to be deleted. If you wish to maintain the nodes and manipulate individual branches of the XML tree - then the session transformer offers the *mergexml* command.

Use the *removexml* command to remove nodes from the context.

<removexml context="mycontext" path="/User/"/>

2.2.1. Example

The following example shows the use of several commands and in particular how the *mergexml* command can be used to manipulate context data.

```
<resource xmlns:session="http://apache.org/cocoon/session/1.0"> <session:createcontext
name="trackdemo"/> <!-- build context data --> <session:setxml context="trackdemo"
path="/"> <context> <users> <user id="1"> <name>Carsten</name> </user> </users>
</context> </session:setxml> <session:mergexml context="trackdemo" path="/context">
<users> <user id="1"> <name>Ziegeler</name> <developer>true</developer> </user> <user
id="2"> <name>Walter</name> </user> </users> </session:mergexml> <session:getxml
context="trackdemo" path="/"/> </resource>
```

In the above example, a context for storing data is added. Using the *setxml* command data is then stored into the context. The following *mergexml* command then changes the name of user-1 and adds a further tag. As there is no original user-2 the complete subtree is then added to the context.

2.3. Reading and writing contexts

Aside from the described means of accessing data in a context, the session transformer also provides for the reading and writing of contexts. This can be used to write an application context out to a database or to read an application context in from a file.

The session transformer offers a very flexible way of defining the source of the context data. It is possible to specify a resource (defined in the sitemap) or a Java class. Using a resource allows for example the context data to be read from a database using the SQL Transformer. As this source is a Cocoon pipeline, the data can be generated and transformed before passing into the context.

When a context is created, it can get additional save and load URIs which are used for loading/saving to/from the context:

<createcontext name="mycontext" load="cocoon://load-from-db" save="cocoon://save-to-db"/>

These URIs can then be used inside a document to load data into a context:

<loadxml context="mycontext"/>

This example would then load the context data via the resource *load-from-db* which must be defined in the sitemap.

Parameters can be passed to and interpreted by the uri or the Java class. This allows the context data to be read dependent on say the current user.

<loadxml context="mycontext"><user>ben</user></loadxml>

The resource addressed by the uri will receive the parameters as request-parameters. In addition the name of the context will always be passed as *contextname*.

Writing context data works in the same manner.

<savexml context="mycontext"/>

Both commands can use the optional *path* attribute:

<loadxml context="mycontext" path="/user"/>

<savexml context="mycotnext" path="/user"/>

The first command will read xml from the uri and store it in the context under the node *user*, the second one saves only the xml subtree under the node *user* to the uri. The resource addressed by the uri will be passed in addition to the *contextname* parameter the *path* parameter with the corresponding value. If the *path* attribute is not present the *path* parameter will get the value *"/"*.

3. Special Contexts

Cocoon creates and maintains special contexts that allow the applications to access the environment. This allows the read-only access to such things as the current request using the same XPath commands previously described. These context do not require any session, they are always available and change on every request.

3.1. The Request Context - Accessing the Environment, Part One

The request context is an XML description of the current (HTTP) request. This context is a special read only context that can be accessed with the usual commands:

<getxml context="request" path="/parameter"/>

For example, if you want to get the value of a parameter with the name *username* you can include the following command in your XML and it will be replaced with the value of the parameter:

<getxml context="request" path="/parameter/username"/>

If you wish to obtain the complete querystring as it was passed into Cocoon - without converting the data to XML - then you can use the "/querystring" path:

<getxml context="request" path="/querystring"/>

The result will be a string in the format "?param=aaa&...".

The complete context you can access via these commands has the following XML format:

<parameter> <!-- All parameters: parameter names build the elements with the value of the first parameter with this name as text node child --> <firstparameter>value of parameter</firstparameter> <secondparameter>value of parameter</secondparameter> </parameter> <querystring>the querystring with a leading '?' or empty<querystring> (The querystring contains only parameters send by the GET method) <parametervalues> <!-- All parameters. The tags are all inside the cinclude transformer namespace. The generated XML can be used without modification for the cinclude:includexml command. --> <cinclude:parameters> <cinclude:parameter> <cinclude:name>1st parameter name</cinclude:name> <cinclude:value>1st parameter value</cinclude:value> </cinclude:parameter> ... <cinclude:parameter> <cinclude:name>2nd parameter name</cinclude:name> <cinclude:value>2nd parameter value</cinclude:value> </cinclude:parameter> </cinclude:parameters> <!-- If a parameter has more than one value, for each value a <session:param> block is generated. --> </parametervalues> <attributes> <!-- lists all attributes, attribute names build the elements with the values as text node childs --> </attributes> <headers> <!-- lists all headers, header names build the elements with the values as text node childs --> </headers> <cookies> <!-- lists all cookies --> <cookie name="..."> <value>the cookie value</value> <name>the name of the cookie</name> <comment>value</comment> <domain>value</domain> <path>value</path> <maxAge>value</maxAge> <secure>value</secure> <version>value</version> </cookie> </cookies> <characterEncoding>value</characterEncoding> <contentLength>value</contentLength> <contentType>value</contentType> <protocol>value</protocol> <remoteAddress>value</remoteAddress> <remoteHost>value</remoteHost> <scheme>value</scheme> <serverName>value</serverName> <serverPort>value</serverPort> <method>value</method> <contextPath>value</contextPath> <pathInfo>value</pathInfo> <pathTranslated>value</pathTranslated> <remoteUser>value</remoteUser> <requestedSessionId>value</requestedSessionId> <requestURI>value</requestURI> <servletPath>value</servletPath> <isRequestedSessionIdFromCookie>value</isRequestedSessionIdFromCookie> <isRequestedSessionIdFromCookie>value</isRequestedSessionIdFromCookie> <isRequestedSessionIdValid>value</isRequestedSessionIdValid>

3.2. The Temporary Context

The temporary context with the name *temporary* is available on each request. It is independent from the session and has no content when a new request starts. It can be used like any other context except that the content is lost/deleted when the current response is finished.

Using the tempory context it is possible to store any XML information for processing the current request.

4. Session-Context Input Module

In addition to the *session transformer*, the *session-context input module* can be used directly in the sitemap to get information out of a context.

The information for the input module consists of two parts, the first one being the context name and the second one the path inside the context. Let's have a look at an example:

<map:generate src="{session-context:request/method}.xml"/>

In the example above, either *GET.xml* or *POST.xml* is read by the *file generator*, depending on the value from the context *request* using the xpath *method*.

1. Comments

add your comments