Configuring Coplets (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. Overview

This document gives an overview over configuring coplets. The sample portal that comes with the Cocoon distribution contains several samples.

The configuration of a coplet is done in several steps that are outlined in the next chapters. We use the provided sample as a base. This sample stores all profiles in XML documents that are all stored in the samples/portal directory. So all file and directory names are relative to this directory. It's not required that you store the configuration in XML files, you can also store them in a database, an LDAP system etc. This is also configurable and customizable.

2. Configuring Coplets

Configuring coplets is like defining a class and creating their instances. So in fact, you define the available coplets (= classes) and each portal view gets some instances of these coplets.

In addition to the classes and the instances, we have coplet types which can be seen as a classification of the coplet classes. There are different coplet types, uri based coplets, JSR-168 coplets, application coplets etc. The coplet type defines, how the coplet works or how the coplet gets it's content. For example, the uri based coplet triggers a uri to get the coplet - this can be an http request or an internal cocoon pipeline request.

Another exsample is the JSR-168 coplet type. This type allows to use JSR-168 compliant portlets for fetching the content. So let's start with the coplet types:

2.1. Available Coplet Types

Before you can define your available coplets, you have to define the available coplet types, or the so called coplet base data. The current exsample contains an XML document for this (the file is in the profiles/copletbasedata directory and is called portlal.xml). This is an excerpt from the file:

... <coplets> <coplet-base-data id="URICoplet"> <coplet-adapter>uri</coplet-adapter> </coplet-base-data> </coplets> ...

In the example above, we define one coplet type, the *URICoplet*, that uses the *uri coplet adapter*. By this we define a type, that uses URIs to get the content of a coplet. A uri can either be targetted at a different server (using http, ftp etc.) or it can be an internal cocoon pipeline (using cocoon:).

You can add different coplet types with additional configuration here, but rarely have to touch this file as most is preconfigured already.

2.2. Available Coplets

Based on the coplet types, you can define the available coplets in your portal application (= classes). In the example portal an own configuration file contains these so called coplet datas (it's in the profiles/copletdata directory and has the filename portal.xml). Here is an excerpt:

... <coplets> <coplet-data id="CZ Weblog" name="standard"> <title>CZ's Weblog</title> <coplet-base-data>URICoplet</coplet-base-data> <attribute> <name>uri</name> <value xsi:type="java:java.lang.String"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">cocoon:/news/liverss?feed=http://radio.we
</attribute> <attribute> <name>buffer</name> <value xsi:type="java:java.lang.Boolean"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</value> </attribute>
<attribute> <name>error-uri</name> <value xsi:type="java:java.lang.String"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">cocoon:/news/CZ_weblog.rss</value>
</attribute> </coplet-data> </coplets> ...

Each coplet data contains a unique id and additional configuration. A required configuration is the underlying coplet base data. In the example above, the *URICoplet* is used that we have configured in the previous chapter.

The above configured coplet data requires some configuration. This configuration depends on the coplet type. In this case we use the URICoplet and this needs of course the URI to fetch the content from for this coplet. This configuration is passed in the different attributes you see above. Each attribute has a name and value.

The set of coplet datas defines the set of available coplets a user can choose from. If a user chooses to view a coplet, an instance of this coplet data is created. If, e.g. the user chooses the same coplet twice, two instances are created. This is useful for configurable coplets where the user can choose the same coplet with different configurations.

2.3. Selected Coplets

The selected coplets are described by the set of coplet instance datas This is again another XML document (that is stored in profiles/copletinstancedata and has the name portal.xml). Here is an excerpt:

... <coplets> <coplet-instance-data id="CZ Weblog-1" name="standard"> <coplet-data>CZ Weblog</coplet-data> </coplet-instance-data> </coplets> ...

The coplet instance data refers to its coplet data (the class) by specifying the unique ID. The instance itself has a unique ID as well that is referenced from the portal view. This id of the instance has only the be unique within the scope of a user. Different users can have different instances with the same id.

In addition, a coplet instance data could have own configuration information. This configuration depends on the coplet type and on the coplet (class).

3. Common Coplet Configuration

As outlined above, a coplet can be configured at various places. In general, configuration values are most often configured at the coplet type (coplet base data) and may be overwritten by the coplet class (coplet data).

The coplet base data takes the information as config entries whereas the coplet data uses attributes. Apart from that the keys and the data types are the same. Let's have a look at an example:

... <coplet-base-data id="URICoplet"> <coplet-adapter>uri</coplet-adapter> <configuration> <name>buffer</name> <value xsi:type="java:java.lang.Boolean" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</value> </configuration> </coplet-base-data> ... <coplet-data id="CZ Weblog" name="standard"> <title>CZ's Weblog</title> <coplet-base-data>URICoplet</coplet-base-data> <attribute> <name>buffer</name> <value xsi:type="java:java.lang.Boolean" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">false</value> </attribute> </coplet-data> ...

As you can see in the exsample above, there is no real difference in configuring a coplet type (base

data) and a coplet class (coplet data) apart that the first one uses the element name *configuration* and the last one uses *attribute*.

3.1. Output Buffering

Each coplet can be configured with the boolean value *buffer* (default is false) that defines if the xml data stream from the coplet is buffered before it is streamed in the main portal pipeline.

If the stream is not buffered and an exception occurs during the streaming of the one coplet, then the whole portal might be rendered invalid. Therefore you should turn on buffering whenever you can't guarantee that the output of a coplet is always valid XML.

3.2. Delivery Timeout

Usually the portal waits forever(!) for a coplet to deliver it's content. If you want to fine-tune this behaviour, you can configure a *timeout* value. This integer value (default is endless) defines the maximum time (in seconds) the coplet has to deliver it's content.

If the timeout is reached the content is assumed as not gettable. If you set a timeout, the content is automatically buffered and the *buffer* configuration is ignored.

4. The URICoplet

This section describes the URICoplet and the different possibilities to configure it. For general configuration see the previous chapter.

4.1. The Content Location and Parameter Handling

A URICoplet uses a uri to fetch the coplet. This configuration is obviously a configuration of the coplet class (and not of the coplet type) as each coplet uses a different location.

The attribute *uri* takes the complete uri (as a string) to fetch the content from:

```
... <coplet-data id="CZ Weblog" name="standard"> <title>CZ's Weblog</title>
<coplet-base-data>URICoplet</coplet-base-data> <attribute> <name>uri</name> <value
xsi:type="java:java.lang.String" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
cocoon:/news/liverss?feed=http://radio.weblogs.com/0107211/rss.xml </value> </attribute>
</coplet-data> ...
```

If you're using the cocoon protocol to call internal pipelines, you can also configure the boolean attribute *handleParameters* (default is false), that defines if the coplet handles request parameters itself. This should be used for coplets that process request parameters (have forms for example). In that case, the parameters from the request are only forwarded to the coplet, if they are meant for this coplet. All other coplets that use request parameters don't get them in this request-response-cycle.

4.2. Error Handling

Usually, in the case of an error during rendering the content of a coplet, a message is included in the coplet window instead of the coplet content. If you want to customize this, you can specifiy the attribute *error-uri* that should contain a valid uri that deliveres the content for a better error message.

5. Coplet Rendering

Each coplet can be configured for supporting different features, like minimizing or removing. These are aspect configurations that are configured similar to attributes.

5.1. Sizing

By default, the window size of a coplet can be changed by the user. The user can choose between maximized and minimized.

This is configurable by the boolean aspect *sizable* on the coplet data (class). The default is true:

... <coplet-data id="CZ Weblog" name="standard"> <title>CZ's Weblog</title> <coplet-base-data>URICoplet</coplet-base-data> <aspect> <name>sizable</name> <value xsi:type="java:java.lang.Boolean" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> false </value> </aspect> </coplet-data> ...

In addition the size of a coplet (minimized/maximized) can be preconfigured as well by the integer aspect value *size*. The value *0* means minimized and *1* is maximized (with 1 as the default).

... <coplet-data id="CZ Weblog" name="standard"> <title>CZ's Weblog</title> <coplet-base-data>URICoplet</coplet-base-data> <aspect> <name>size</name> <value xsi:type="java:java.lang.Integer" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> 0 </value> </aspect> </coplet-data> ...

5.2. Mandatory Coplets

By default, the user can delete any coplet from his portal view. However important coplets can be configured as *mandatory*. In this case, the remove icon is not displayed, and the coplet can never be removed by the user.

This is configurable by the boolean aspect *mandatory* on the coplet data (class). The default is false:

... <coplet-data id="CZ Weblog" name="standard"> <title>CZ's Weblog</title> <coplet-base-data>URICoplet</coplet-base-data> <aspect> <name>mandatory</name> <value xsi:type="java:java.lang.Boolean" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> true </value> </aspect> </coplet-data> ...

1. Comments

add your comments