Custom Resource Type How-To

Table of contents

1 Decide if you really need a new resource type 2
2 Choose a name 2
3 Define the resource type ID 2
4 Create the schema 3
5 Resource type matcher 4
6 Sample File 5
7 Presentation 5
8 Menus 6
9 Using an existing menu 6
10 using a custom menu 7
11 Workflow 7

This tutorial explains how to add a custom resource type to the Default publication in order to mix free page editing (as in XHTML) and constrained editing (as you get with a custom resource type).

1. Decide if you really need a new resource type

Introducing a new resource type where you don't need one is a time consuming and often unnecessary excercise. Therefore you should ask yourself first if you really need a new resource type for what you are trying to achieve or if you should be looking for other means of solving your problem.

As a rule of thumb, you should use a new resource type if you are going to render documents which are fundamentally different from XHTML. A resource type is all about Lenya turning an arbitraty XML document into XHTML which then will be put into the content page of your publication's page.

It might make sense for example to introduce resource types for well known XML schemas such as:

- RSS
- DocBook (http://docbook.sourceforge.net/)
- NewsML, SportsML, NITF (http://nitf.org/) or IPTC (http://www.iptc.org)
- FOAF (http://xmlns.com/foaf/0.1/)

It may also make sense to introduce resource types for custom XML schemas such as

- the XML format you use for a catalog item in your company

In contrast, it is not a good idea to introduce a custom resource type if you want to render documents that are basically XHTML but just contain some non-XHTML tags such as

- <xi:include ...>
- <ft:...> (Cocoon Form tags)

In this case you should rather add some extra transformers to the pipeline that renders these tags into XHTML.

Also resource types are not XHTML templates.

2. Choose a name

Choose a name for the resource type ID: *myresourcetype*

3. Define the resource type ID

Define the resource type ID in *mypub/config/doctypes/doctypes.xconf* Note that resource types were historically called doctypes, thus the names of the config files.

```
<doctypes>
  <!-- Here is the document ID -->
```

```
  <doc type="myresourcetype">
    <!-- if the document is allowed to have children, define what resource types they can be -->
    <children>
      <!-- this resource type is allowed to have xhtml and/or myresourcetype documents as children -->

      <doc type="xhtml"/>
      <doc type="myresourcetype"/>
    </children>
    <!-- This is the class that creates a new instance of this resource type -->
    <creator src="org.apache.lenya.cms.authoring.DefaultBranchCreator">
      <!-- this is the sample file that is used when creating a new instance of this resource type. Can
be found in config/doctypes/samples -->

      <sample-name>myresourcetype.xml</sample-name>
    </creator>
    <!-- These tasks here refer to ant tasks that should be called for this resource type. For instance,
you can have a special publish task. -->
    <!-- tasks are defined in mypub/config/tasks/tasks.xconf -->
    <tasks>

      <task id="publish"/>
    </tasks>
    <!-- Specify the workflow definition for this resource type. You can define a seperate workflow for
each resource type, for instance with one or two levels of approval. -->
    <!-- workflow process is defined in mypub/config/workflow/workflow.xml (see below) -->
    <workflow src="workflow.xml"/>
  </doc>

</doctypes>
```

4. Create the schema

Create relax NG schema: *mypub/config/doctypes/schemas/myresourcetype.rng*

Note: This is optional, but important if you want to use the form editors. If you get your XML files that you try to render as a custom resource type from somewhere else you can omit this step. You can also omit it if you want to create a sample document manually first to see all the rest working and come back to this point later.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--+
    | myresourcetype resource type
    +-->

<grammar ns="http://www.w3.org/1999/xhtml"
         xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:lenya="http://apache.org/cocoon/lenya/page-envelope/1.0"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:dcterms="http://purl.org/dc/terms/"
         xmlns:xhtml="http://www.w3.org/1999/xhtml"
         >

<!-- include the general lenya schema -->
<!-- you should include the lenya schema if you want to use meta data or assets -->
<include href="lenya.rng"/>

<!-- include original XHTML transitional schema -->
<include href="xhtml/xhtml-basic.rng">

  <define name="html">

    <element name="html">
      <ref name="html.attlist"/>
      <ref name="lenya.meta"/> <!-- this is the Lenya meta data wrapper -->
      <ref name="head"/>
      <ref name="body"/>

    </element>
  </define>
```

```
  <define name="html.attlist">
    <ref name="XHTML.version.attrib"/>
    <ref name="I18n.attrib"/>
    <ref name="dummy.attlist"/> <!-- this is deprecated -->

  </define>

</include>


<!-- additional block elements -->
<define name="Block.class" combine="choice">
  <choice>
    <ref name="lenya.asset"/>

  </choice>
</define>


</grammar>
```

Note that you can generate Relax NG schemas from sample files by using Trang (http://www.thaiopensource.com/relaxng/trang.html) or Sun RELAX NG Converter (http://wwws.sun.com/software/xml/developers/relaxngconverter/)

5. Resource type matcher

Add a sourcetype matcher in *mypub/parameter-doctype.xmap*

```
<!-- This file is map:mounted from mypub/publication-sitemap.xmap -->

<map:action name="sourcetype" src="org.apache.cocoon.acting.sourcetype.SourceTypeAction">
  <sourcetype name="myresourcetype">

    <document-element local-name="myroottag"/>
    <!-- this matches the root tag -->
  </sourcetype>
</map:action>
```

The parameter-doctype.xmap allows Lenya to determine the resource type of any URI. the mapping provided in here allows to mix and match resource types freely. It makes use of the Source Type action (http://forrest.apache.org/docs/cap.html) . There are several ways to identify a resource type, such as by its root tag, the namespace of its root element or its schema. Please note that the name of the resource type you specify here is used throughout Lenya as a naming convention, for instance for the resourcetype2xhtml.xsl.

The root-tag is just the doctype within your publication (i.e., 'homepage', if you have a homepage.rng doctype definition). For more background, see: http://forrest.apache.org/docs/cap.html (http://forrest.apache.org/docs/cap.html)

For great examples of complex custom types, please also go to http://www.wyona.org (http://www.wyona.org) and do a Subversion update to download the example publication (University of Zurich and others).

Many problems with resource types not rendering correctly result from this matcher not recognizing the resource type and therefore using the default. So pay special attention to this. Check the overview page in the site area where it will tell you what resource type the document has.

6. Sample File

Add a sample file: *mypub/config/doctypes/samples/myresourcetype.xml*

The sample file will be used as a template. If you create a new document with this resource type a copy of the sample file will be placed as a starting point. Therefore it makes sense to set some good defaults here.

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:xhtml="http://www.w3.org/1999/xhtml"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:dcterms="http://purl.org/dc/terms/"
      xmlns:lenya="http://apache.org/cocoon/lenya/page-envelope/1.0"
      xhtml:dummy="FIXME:keepNamespace" dc:dummy="FIXME:keepNamespace"
      lenya:dummy="FIXME:keepNamespace" dcterms:dummy="FIXME:keepNamespace">
    <lenya:meta>
        <dc:title>dctitle</dc:title>

        <dc:creator>dccreator</dc:creator>
        <dc:subject>dcsubject</dc:subject>
        <dc:description>An empty page for a new resource type</dc:description>
        <dc:publisher/>

        <dc:contributor/>
        <dc:date>2004-4-6</dc:date>
        <dc:type/>
        <dc:format/>
        <dc:identifier/>

        <dc:source/>
        <dc:language>en</dc:language>
        <dc:relation/>
        <dc:coverage/>
        <dc:rights>dcrights</dc:rights>

    </lenya:meta>
  <head>
    <title>Sample new resource type</title>
  </head>
  <body>

    <h1>New resource type sample</h1>
    <p>This sample page is the basis for pages based on this resource type</p>
  </body>
</html>
```

7. Presentation

- Create presentation XSLT and CSS. If the XSLT and CSS are specific to the publication, you may want to store them in *mypub/xslt/myresourcetype2xhtml.xsl* and *mypub/resources/shared/css/myresourcetype.css*, respectively. Take care of the naming conventions for the XSLT files.

- Copy *mypub/xslt/page2xhtml-xhtml.xsl* to *mypub/xslt/page2xhtml-myresourcetype.xsl*. This will style your new document type like your old documents. If you wish to change how the new documents look, create your own stylesheet at *mypub/xslt/page2xhtml-myresourcetype.xsl*

- If you wish to edit your new documents using BXE (Kupu works only with XHTML), you should place a CSS file for styling your documents in *mypub/resources/misc/myresourcetype-bxeng.css*

- Add presentation pipelines in *mypub/sitemap.xmap* if you need special pipelines for your resource type

Make sure you understand that for your new resource type you don't need a replacement for page2xhtml.xsl but for xhtml2xhtml. The reason is this:

Transforming a document to a page in the browser is always a two-step process,

1. Using {resourcetype}2xhtml.xsl the document (which might be any arbritary XML such as the link list, Docbook, a RSS feed, etc.) is turned into an XHTML presentation.
2. This XHTML is then processed to contain nagivation, menus and the like using page2xhtml.
3. The stylesheet {resourcetype}2xhtml.xsl must return a document of the form * ... *.

If you have 5 resource types in your publication, for example:

- rss
- linklist
- gallery
- docbook
- xhtml

the concept would be

```
document -> rss2xhtml.xsl       --+
document -> linklist2xhtml.xsl  --+
document -> gallery2xhtml.xsl   --+---> page2xhtml.xsl -> Browser
document -> docbook2xhtml.xsl   --+
document -> xhtml2xhtml.xsl     --+
```

8. Menus

Configure the menus (either 1. or 2.)

9. Using an existing menu

Use an existing menu and add necessary menu items

- Edit *mypub/config/menus/generic.xsp* and add an entry for the new resource type
-

```
...
    <menus>
      <menu i18n:attr="name" name="File" label="File">
        <block>
          <xsp:logic>
```

```
            {
        if (Publication.ARCHIVE_AREA.equals(area) || Publication.TRASH_AREA.equals(area)) {
          <item><i18n:text>New Document</i18n:text></item>

        }
        else {
          <item uc:usecase="create" uc:step="showscreen" href="?doctype=xhtml"><i18n:text>New
Document</i18n:text></item>
          <item uc:usecase="create" uc:step="showscreen" href="?doctype=myresourcetype">New "Doc
type" Document</item>
        }
            }
          </xsp:logic>

        </block>
...
```

10. using a custom menu

or Create a custom menu for this resource type

- edit *mypub/menus.xmap*
- create a menu file *mypub/config/menus/myresourcetype.xsp*

11. Workflow

(optional) define a specific workflow

- add the workflow schema to *mypub/config/workflow/*
- assign the workflow schema to the resource type in *mypub/config/doctypes/doctypes.xconf*