SavingFilesToFileSystem

Saving PDF files to the server file system

Introduction

Suppose we need to save a PDF file to the file system instead of sending it to the user's browser. I will explain later why we would need this.

The Source{{`Writing}}`Transformer could be used for this, but pipeline components having side effects aren't considered to be a good practice. So we describe here a more "modern" and correct solution using the *cocoon.processPipelineTo* method from flowscript, to save the result of a pipeline to a file.

Process

- Open the "sitemap.xmap" for "blablabla" directory and create a usual pipeline that creates PDF files from XML files. You can use any XML file and any XSL-FO stylesheet from Cocoon package. Bind it to some abstract extension, for example "*.do".

```
<map:pipeline internal-only="false">
  <map:match pattern="**/*.do">
    <map:generate src="{1}/{2}.xml"/>
    <map:transform src="convert.xsl" label="content"/>
    <map:serialize type="fo2pdf"/>
  </map:match>
</map:pipeline>
```

- Test the pipeline using your browser: i.e. type `http://<host>/blablabla/test.do`. Cocoon should return a PDF file to the browser as expected.
- Add the following to the top of the sitemap.xmap file:

```
<map:flow language="javascript">
  <map:script src="pdf.flow"/>
</map:flow>
```

pdf.flow - is our future flowscript file. It's extension can be anything (.js, .fff or .flow). I use ".flow" just to distinguish it from usual JavaScript files with ".js" extension.

- Add the following match to the pipeline.

```
<map:match pattern="**/*.pdf">
  <map:call function="makepdf">
    <map:parameter name="folder" value="{realpath:/}"/>
  </map:call>
</map:match>
```

As you see this match is linked to "*.pdf" extension. This match will call the "makepdf()" function from "pdf.flow" script file (which in its turn will call the .do match).

- Create the "pdf.flow" text-file and add the following code to it:

```
function makepdf() {
   var xml_file;
   var pdf_file;
   var outstreamPDF;

   // the parameter passed to the script from the pipeline.
   // This is the real path to the application context
   var folder = Packages.java.lang.String(cocoon.parameters["folder"]);

   try {
      // creating links to files
      pdf_file = Packages.java.io.File( folder + "/target_folder/book.pdf");
      xml_file = Packages.java.io.File( folder + "/source_folder/book.xml");

      // creating outputstream to dump the results of conversion to the file
      outstreamPDF = new Packages.java.io.FileOutputStream( pdf_file );

      // calling cocoon pipeline using processPipelineTo() method and dumping the results to the outputstream
      cocoon.processPipelineTo( "blablabla/book.do", null, outstreamPDF );

      // do not forget to close the outputstream
      outstreamPDF.close();

      // since the result is dumped to the filesystem we need to send smth. to the browser
      // to make it happy. So let's send just a usual .txt file with OK message
      cocoon.sendPage("yes.txt", null);

   } catch( ex ) {
      cocoon.log.error(ex);
      // Smth. went wrong. Sending a error.txt file to the browser
      cocoon.sendPage("error.txt", null);
   }
}
```

This function is calling the "*.do" match in the pipeline that we created before. Test how it works by typing `http://<host>/blablabla/test.pdf` in your browser. You should get a OK text message and file should be dumped to the folder specified in parameters.

- After debugging the script and the pipeline, make the pipeline invisible to the browser by setting:

```
<map:pipeline internal-only="true">
```

Points of further improvement

- The script can check if the target file is already created; and if it does exist, compare its lastModified() with the lastModified() of the source file, so the script will run conversions only when the source file is changed or when the new source file is added.
- Use cron build into Cocoon to make conversions at a predefined time. Add another match to the pipeline for cron program:

```
<map:match pattern="justdo.it">
  <map:call function="makepdf">
   <map:parameter name="folder" value="{realpath:/}"/>
  </map:call>
</map:match>
```

This match will not take the name of the file to convert and it is not supposed to be called from browser (though you can type http://<host>/blablabla/justdo.it and run the flow script). Instead this match will be called from cron.

- Instead of converting just one XML file try to improve the script, so it would convert ALL xml files from source directory.
- You can connect to the database from flowscript and use the data from the DB to make conditional conversions.
- Consider rewriting the script to use the Source API instead of File. Then it would work on any ModifiableSource.