StreamingMedia

This page gives an outline of what you need to do in order to deliver streaming media (for example RealMedia or SMIL) via Cocoon.

There are a number of steps to doing this:

- Adding media serializers to the sitemap
- Adding media matchers to the sitemap
- Testing delivery of static content
- Creating SMIL on the fly
- Creating RealText on the fly

Because of the "flakiness" of many current media players, it's a good idea to take an incremental approach - making sure that each step works successfully - as problems can often be hard to track down and are frequently caused by bugs or quirks in the players.

Players

The state of the SMIL player market is disastrous - there are no really solid SMIL players, no cross-platform standardisation, and there are severe inconsistencies in what works where. As a minimum, the contents of this page should work with RealPlayer 8 on Windows and Linux, but you will probably need to make adjustments if your target player is something else (for example Apple's QuickTime or Real's RealOne).

Adding media serializers to the sitemap

The first step is to add serializers to the sitemap to make sure that the media files (and associated files) we'll be streaming will have the right MIME types, etc.

In the `serializers` section of the sitemap, add the following:

```
<map:serializer name="smil" mime-type="application/smil" logger="sitemap.serializer.smil"
        src="org.apache.cocoon.serialization.XMLSerializer"/>

<map:serializer name="rt" mime-type="text/vnd.rn-realtext" logger="sitemap.serializer.rt"
        src="org.apache.cocoon.serialization.HTMLSerializer"/>

<map:serializer name="ram" mime-type="audio/x-pn-realaudio" logger="sitemap.serializer.ram"
        src="org.apache.cocoon.serialization.TextSerializer"/>

<map:serializer name="rm" mime-type="audio/x-pn-realaudio" logger="sitemap.serializer.rm"
        src="org.apache.cocoon.serialization.XMLSerializer"/>

<map:serializer name="rp" mime-type="application/x-pn-realmedia"
        logger="sitemap.serializer.ram" src="org.apache.cocoon.serialization.XMLSerializer"/>
```

Rationale

Here's why the various serializers:

- SMIL is an XML-based language, so it makes sense to use the XMLSerializer
- RealText (rt) is a stripped-down (ugly) variant of HTML, but is not valid XML, so the HTMLSerializer makes sense.
- RAM files are plain text files containing just a URL, so the TextSerializer must be used for these.
- RM and RP files both contain XML.

Adding media matchers to the sitemap

This step is simply to make sure you are able to deliver binary media files to the players from within Cocoon. This is useful for testing. In a production environment, you will probably want to bypass Cocoon for delivering these files, as they tend to be fairly large.

In the `map:pipelines` section of the sitemap, add the following:

```
<!-- streaming media -->
<map:pipeline>
        <map:match pattern="**.rt">
                <map:read src="resources/realtext/{1}.rt" mime-type="text/vnd.rn-realtext"/>
        </map:match>

        <map:match pattern="**.smil">
                <map:read src="resources/smil/{1}.smil" mime-type="application/smil" />
        </map:match>

        <map:match pattern="**.ram">
                <map:read src="resources/realmedia/{1}.ram" mime-type="audio/x-pn-realaudio"/>
        </map:match>

        <map:match pattern="**.rm">
                <map:read src="resources/realmedia/{1}.rm" mime-type="audio/x-pn-realaudio"/>
        </map:match>

        <map:match pattern="**.rp">
                <map:read src="resources/realmedia/{1}.rp" mime-type="application/x-pn-realmedia"/>
        </map:match>
</map:pipeline>
```

This covers the basic types of file that you may be delivering to RealPlayer / RealOne. If you are using other media players (such as Windows Media Player or QuickTime) you may like to add matchers for other file formats (eg .wma, .mov).

Testing delivery of static content

When you have made the modifications listed above, it's a good idea to test they are working by trying to view some streaming media on your site. First, add some content to the realtext and realmedia directories.

TODO: Add links to example rt, rm, smil files on Real site.

Creating SMIL on the fly

Some browser/player combinations don't work terribly well with `http://localhost/foo.smil` URLs, but they will work fine with `http://localhost/foo.ram`. However, we want to output SMIL. To get round this, we create ram files that simply bounce the user's player to the SMIL.

One way to do this is as follows: create two matchers in the sitemap, one for the ram file, one for the SMIL file:

```
<map:match pattern="foo.ram">
        <map:generate type="serverpages" src="xsp/ramhurl.xsp"/>
        <map:transform src="stylesheets/ramhurl.xsl"/>
        <map:serialize type="ram"/>
</map:match>

<map:match pattern="foo.smil">
        <map:generate type="serverpages" src="xsp/smil.xsp"/>
        <map:transform src="stylesheets/smil.xsl"/>
        <map:serialize type="smil"/>
</map:match>
```

Then, create `ramhurl.xsp` to return a URL to the SMIL file:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ==== RAMHURL.XSP ==== -->

<!--
        Dynamically creates a SMIL bounce
-->
<xsp:page language="java"
        xmlns:xsp="http://apache.org/xsp"
        xmlns:xsp-request="http://apache.org/xsp/request/2.0">

        <document>
                http://<xsp-request:get-server-name/>/path/to/foo.smil
        </document>
</xsp:page>
```

... and this simple XSL file makes sure that the only thing output by the XSP is the URL. Some players (particularly RealPlayer 8 on Linux) will break horribly if the format of the ram file is not precise.

```
<?xml version="1.0"?>

<!-- ==== RAMHURL.XSL ==== -->

<!--
        Dynamically creates a SMIL bounce
 -->
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:output indent="no" method="text"/>

        <xsl:template match="document">
                <xsl:value-of select="normalize-space(.)"/>
        </xsl:template>
</xsl:stylesheet>
```

The `smil.xsp` file can be used to assemble content dynamically from a database or some other source:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ==== SMIL.XSP ==== -->

<!--
        Dynamically creates SMIL files
-->
<xsp:page language="java"
        xmlns:xsp="http://apache.org/xsp"
        xmlns:xsp-request="http://apache.org/xsp/request/2.0"
        xmlns:esql="http://apache.org/cocoon/SQL/v2">

        <document>
                <esql:connection>
                        <esql:pool>foo</esql:pool>

                        <esql:execute-query>
                                <esql:query>
                                </esql:query>
                                <esql:results>
                                        <esql:row-results>
                                                <!-- Fill in the blanks here! -->
                                        </esql:row-results>
                                </esql:results>
                        </esql:execute-query>
                </esql:connection>
        </document>
</xsp:page>
```

... and the `smil.xsl` file converts it into valid SMIL:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ==== SMIL.XSL ==== -->

<!--
        Dynamically creates a SMIL file
 -->
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:template match="document">
                <smil>
                        <head>
                                <meta name="copyright" content="Apache Software Foundation" />
                                <layout>
                                        <root-layout width="360" height="400" background-color="white" />
                                        <region id="audio" left="5" top="80" z-index="1" fit="fill" />
                                        <region id="text" left="5" top="350" z-index="2" fit="fill" />
                                </layout>
                        </head>

                        <body>
                                <!-- media and synchronization tags -->
                                <par>
                                        <audio alt="Digital Resource" region="audio">
                                                <xsl:attribute name="src">/<xsl:value-of select="digital_object"
/>.rm</xsl:attribute>
                                        </audio>
                                        <textstream region="text" fill="freeze">
                                                <xsl:attribute name="src"><xsl:value-of select="digital_object"
/>.rt</xsl:attribute>
                                        </textstream>
                                </par>
                        </body>
                </smil>
        </xsl:template>
</xsl:stylesheet>
```

Creating RealText on the fly

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ==== RT.XSP ==== -->

<!--
        Dynamically creates realtext
-->
<xsp:page language="java"
        xmlns:xsp="http://apache.org/xsp"
        xmlns:xsp-request="http://apache.org/xsp/request/2.0"
        xmlns:esql="http://apache.org/cocoon/SQL/v2">

        <document>
                <esql:connection>
                        <esql:pool>foox</esql:pool>

                        <esql:execute-query>
                                <esql:query>
                                </esql:query>
                                <esql:results>
                                        <esql:row-results>
                                        </esql:row-results>
                                </esql:results>
                        </esql:execute-query>
                </esql:connection>

        </document>
</xsp:page>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ==== RT.XSL ==== -->

<!--
        Dynamically creates realtext.
 -->
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:output indent="no" method="text"/>

        <xsl:template match="document">
                <window type="generic" width="350" height="50" bgcolor="white" wordwrap="true">
                        <center>
                                <font size="-1" face="Arial" color="#666666">
                                        <xsl:value-of select="foo"/>
                                </font>
                        </center>
                </window>
        </xsl:template>
</xsl:stylesheet>
```