WebDAVAuthentic

Authentic

Authentic (formerly known as XML Spy document editor browser plugin) is a ActiveX control (free of charge) which allows you to edit your XML documents in a WYSIWYG manner from within Internet Explorer.

This tutorial shows you how to extends the "step3" sample of Cocoon's WebDAV block with Authentic for WYSIWYG editing.

First copy the complete "step3" folder to "step6".

Go to the Altova site and download the Authentic 2004 Browser Edition.

Within the "step6" folder create another folder named "authentic" and copy the "AuthenticBrowserEditionUnicode.cab" into the "authentic" folder.

Within this folder create 2 additional files, content.xsd (the XML schema)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xs:element name="page">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="title"/>
      <xs:element ref="content"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="content">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="para" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="title" type="xs:string"/>
<xs:element name="para">
  <xs:complexType mixed="true">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="emphasis" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="image" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="link" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="emphasis">
  <xs:complexType mixed="true"/>
</xs:element>
<xs:element name="image">
  <xs:complexType>
    <xs:attribute name="href" type="xs:string" use="required"/>
    <xs:attribute name="height" type="xs:long"/>
    <xs:attribute name="width" type="xs:long"/>
  </xs:complexType>
</xs:element>
<xs:element name="link">
  <xs:complexType mixed="true">
    <xs:attribute name="href" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

and content.sps (a special stylesheet needed by Authentic)

```
```

```
<?xml version="1.0" encoding="UTF-8"?>
<structure version="2" schemafile="contentA.xsd" workingxmlfile="" templatexmlfile="">
<nspair prefix="xs" uri="http://www.w3.org/2001/XMLSchema"/>
<template>
  <match overwrittenxslmatch="/"/>
  <children>
    <template>
      <match match="page"/>
      <children>
        <template>
          <match match="title"/>
          <children>
            <xpath allchildren="1">
              <styles color="#0000A0" font-size="larger" font-weight="bold" text-decoration="underline"/>
            </xpath>
          </children>
        </template>
        <template>
          <match match="content"/>
          <children>
            <template>
              <match match="para"/>
              <children>
                <paragraph paragraphtag="p">
                  <children>
                    <xpath allchildren="1"/>
                  </children>
                </paragraph>
              </children>
            </template>
          </children>
        </template>
      </children>
    </template>
  </children>
</template>
<template>
  <match match="emphasis"/>
  <children>
    <xpath allchildren="1">
      <styles font-weight="bold"/>
    </xpath>
  </children>
</template>
<template>
  <match match="image"/>
  <children>
    <image>
      <properties border="0"/>
      <imagesource>
        <xpath value="@href"/>
      </imagesource>
    </image>
  </children>
</template>
<template>
  <match match="link"/>
  <children>
    <link>
      <hyperlink>
        <xpath value="@href"/>
      </hyperlink>
      <children>
        <xpath allchildren="1"/>
      </children>
    </link>
  </children>
</template>
</structure>
```

Add the following entry to the sitemap:

```
<map:match pattern="authentic/**">
  <map:read src="{global:staging}authentic/{1}"/>
</map:match>
```

Then open your "file2html.xsl" stylesheet and change it to look like this:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:param name="file"></xsl:param>
<xsl:param name="sitemapURI"></xsl:param>
<xsl:param name="requestURI"></xsl:param>
<xsl:param name="serverName"></xsl:param>
<xsl:param name="serverPort"></xsl:param>

<xsl:template match="/page">
  <html>
    <body>
<!--
HINT:
  THE_CLASSID and THE_CODEBASE_CAB_FILE_NAME depends on downloaded version.
  Form more information see: "The OBJECT element"
  in http://www.altova.com/manualDocEditorPlugIn/

The xsl:attribute elements have linebreak and whitespace added to look good on the wiki.
==> Remove this in your editor before testing (xsl:attribute only on one line).
-->
      <object id="objPlugIn" style="width:100%; height:90%"
              classid="<THE_CLASSID>" >
        <xsl:attribute name="codebase">
          http://<xsl:value-of select="$serverName"/>:<xsl:value-of select="$serverPort"/>
          <xsl:value-of select="substring-before($requestURI, $sitemapURI)"/>
          authentic/<THE_CODEBASE_CAB_FILE_NAME></xsl:attribute>
        <param name="XMLDataURL">
          <xsl:attribute name="value">
            http://<xsl:value-of select="$serverName"/>:<xsl:value-of select="$serverPort"/>
            <xsl:value-of select="substring-before($requestURI, $sitemapURI)"/>
            raw/<xsl:value-of select="$file"/></xsl:attribute>
        </param>
        <param name="XMLDataSaveURL">
          <xsl:attribute name="value">
            http://<xsl:value-of select="$serverName"/>:<xsl:value-of select="$serverPort"/>
            <xsl:value-of select="substring-before($requestURI, $sitemapURI)"/>
            write/<xsl:value-of select="$file"/>
          </xsl:attribute>
        </param>
        <param name="SPSDataURL">
          <xsl:attribute name="value">
            http://<xsl:value-of select="$serverName"/>:<xsl:value-of select="$serverPort"/>
            <xsl:value-of select="substring-before($requestURI, $sitemapURI)"/>
            authentic/content.sps</xsl:attribute>
        </param>
        <param name="SchemaDataURL">
          <xsl:attribute name="value">
            http://<xsl:value-of select="$serverName"/>:<xsl:value-of select="$serverPort"/>
            <xsl:value-of select="substring-before($requestURI, $sitemapURI)"/>
            authentic/content.xsd</xsl:attribute>
        </param>
        <param name="EntryHelpersEnabled" value="TRUE"/>
        <param name="EntryHelperWindows" value="3"/>
      </object>
      <br/>
      <input type="button" name="submit1" value="Save" onClick="objPlugIn.SavePOST()"/>
    </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Modify the pipeline for reading content to look like this:

```
<map:match pattern="repo/**">
  <map:generate src="{global:staging}repo/{1}"/>
  <map:transform src="{global:staging}styles/file2html.xsl">
    <map:parameter name="file" value="{1}"/>
    <map:parameter name="requestURI" value="{request:requestURI}"/>
    <map:parameter name="sitemapURI" value="{request:sitemapURI}"/>
    <map:parameter name="serverName" value="{request:serverName}"/>
    <map:parameter name="serverPort" value="{request:serverPort}"/>
  </map:transform>
  <map:serialize type="html"/>
</map:match>
```

and add a pipeline for reading the raw xml content (this could be just a view on the above as well):

```
<map:match pattern="raw/**">
  <map:generate src="{global:staging}repo/{1}"/>
  <map:serialize type="xml"/>
</map:match>
```

Then change the pipeline for writing the content to look like this:

```
<map:match pattern="write/**">
  <map:generate type="stream">
    <map:parameter name="defaultContentType" value="text/xml"/>
  </map:generate>
  <map:transform src="{global:staging}styles/stream2write.xsl">
    <map:parameter name="file" value="{global:staging}repo/{1}"/>
  </map:transform>
  <map:transform type="write-source"/>
  <map:serialize type="xml"/>
</map:match>
```

with "stream2write.xsl" being:

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

<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:req="http://apache.org/cocoon/request/2.0"
                xmlns:source="http://apache.org/cocoon/source/1.0">

<xsl:param name="file"></xsl:param>

<xsl:template match="/">
<page>
  <source:write create="true">
    <source:source><xsl:value-of select="$file"/></source:source>
    <source:fragment>
      <xsl:copy-of select="node()"/>
    </source:fragment>
  </source:write>
</page>
</xsl:template>

</xsl:stylesheet>
```

This works nice, but how am I supposed to edit my meta data you might ask.

Well, for once you can put a second form above or below the editing area.

But Authentic can do WebDAVAuthenticMetadata for you as well.