Bitflux Editor (BXE)

Table of contents

1. Overview

BXE (http://bxe.oscom.org) is a validating WYSIWYG XML editor for Mozilla-based browsers. It uses RELAX NG (http://www.relaxng.org/) for validation and CSS for layout. Integration with Lenya is done with a series of use case steps that are defined in a sitemap inside your publication: `usecase-bxeng.xmap`. If you understand the purpose of the various use case steps, you will have a good grasp of how to integration is done, and will be able to adjust it to your needs as necessary.

2. The open step

The open step is called when you first open BXE from the Lenya menu. It first checks if BXE is installed, and displays a warning page if it is not. It then attempts to check out the page to be edited (to protect it from being edited by other users at the same time), and displays an error page if this fails for some reason. It then loads the BXE start page that contains references to the BXE configuration to be used for this page, such as the location of the RELAX NG schema, CSS files, and more. The BXE start page is a dynamically generated page that is being aggregated from a configuration pipeline (the config step), a file that contains all the namespaces that may occur in the document to be edited (`content-namespaces.xml`), and the skeleton BXE start page (`index.xhtml`). The pipeline looks like this:

```
          <map:match type="step" pattern="open">
           <!-- Check for BXENG -->
           <map:act type="resource-exists" src="../../resources/bxeng/bxeLoader.js">
            <map:act type="reserved-checkout">
              <map:generate type="serverpages"
src="../../content/rc/{exception}.xsp">
                <map:parameter name="user" value="{user}"/>
                <map:parameter name="filename" value="{filename}"/>
                <map:parameter name="date" value="{date}"/>
                <map:parameter name="message" value="{message}"/>
              </map:generate>
              <map:transform src="../../xslt/rc/rco-exception.xsl"/>
              <map:call resource="style-cms-page"/>
            </map:act>
            <map:aggregate element="bxeng">
              <map:part src="../../resources/misc/bxeng/index.xhtml"/>
              <map:part src="../../resources/misc/bxeng/content-namespaces.xml"/>
            </map:aggregate>
            <map:transform src="../../xslt/bxeng/aggregate.xsl"/>
            <map:transform src="../../xslt/bxeng/index-xhtml.xsl">
              <map:parameter name="configfile"
value="{request:requestURI}?lenya.usecase=bxeng&amp;lenya.step=config"/>
              <map:parameter name="context" value="{request:contextPath}"/>
            </map:transform>
            <map:serialize type="xhtml"/>
            </map:act>
            <map:generate src="../../resources/misc/bxeng/download.xhtml"/>
            <map:call resource="style-cms-page"/>
            <map:serialize type="html"/>
          </map:match>
```

3. The config step

The config step generates the BXE config file by transforming a template file (`config.xml`) and passing in values for the following parameters:

- BX_exitdestination: URL of the location BXE should redirect to upon exit
- BX_validationfile: URL of the RELAX NG schema for the XML to be edited
- BX_xhtmlfile: URL of the XHTML page to load the XML content into for editing
- BX_xmlfile: URL of the XML for the page to be edited
- BX_xslfile: URL of the XSL stylesheet to use for formatting the XML in BXE (BXE 1.1 only)
- contextmenufile: URL of the file that defines the BXE context menu
- css: URL of the CSS file used for styling the XML to be edited in BXE
- defaultlanguage: default language of the publication, used for insert popups

If you want to customize BXE, the config step is your starting point. Make sure to pass in the right URL for these parameters. The default publication uses additional pipelines for these URL to provide more flexibility (such as automatically loading the correct RELAX NG schema based on the resource type of the current page). More information about the format of `config.xml` is available at the BXE wiki (http://wiki.bitfluxeditor.org/Config.xml) . The config step pipeline looks as follows in the default publication:

```
        <map:match type="step" pattern="config">
          <map:generate src="../../resources/misc/bxeng/inc/config.xml"/>
          <map:transform src="../../xslt/bxeng/config-xml.xsl">
            <map:parameter name="BX_xmlfile"
value="{request:requestURI}?lenya.usecase=bxeng&amp;lenya.step=xml"/>
            <map:parameter name="defaultlanguage"
value="{page-envelope:default-language}"/>

  <!--      Instead of an xsl we use the xhtml file to provide the basic layout
            <map:parameter name="BX_xslfile" value="{2}.xsl"/>
  -->
            <map:parameter name="BX_xhtmlfile" value="{../2}.bxe.html"/>
            <map:parameter name="BX_validationfile"
value="{request:contextPath}/{page-envelope:publication-id}/{page-envelope:area}/{page-envelo
            <map:parameter name="css"
value="{request:contextPath}/{page-envelope:publication-id}/{page-envelope:area}/css/{page-en
  <!--       The document is checked in when we exit from bx (in case of save&exit
and in case of exit), so we use the usecase
             for the checkin while we redirect to the document
  -->
            <map:parameter name="BX_exitdestination"
value="{request:requestURI}?lenya.usecase=checkin&amp;lenya.step=checkin&amp;backup=true"/>
            <map:parameter name="contextmenufile"
value="../../resources/misc/bxeng/contextmenu.xml"/>
          </map:transform>
          <map:transform type="cinclude"/>
          <map:serialize type="xml"/>
        </map:match>
```

4. The xml step

The xml step is responsible for retrieving the XML of the page to be edited, and sending it back to the server for saving (via HTTP PUT). It first checks if the request method is a GET or a PUT, and goes on to either deliver the XML to BXE (for the GET case) or sending the XML to the server (for the PUT case). If the request method is a PUT, it then checks if the page is properly checked out, and invokes a flow script function (`editDocument`) to save the page. The `editDocument` function takes care of checking the page back in, triggering workflow transitions, and finally redirecting to the saved page. This is the xml pipeline from the default publication:

```
      <map:match type="step" pattern="xml">
        <map:select type="request-method">

          <map:when test="PUT">
            <!-- before we save, we must be sure that the document is well checked
out
            -->
                <map:act type="reserved-checkout-test">
                  <map:generate type="serverpages"
src="../../content/rc/{exception}.xsp">
                    <map:parameter name="user" value="{user}"/>
                    <map:parameter name="filename" value="{filename}"/>
                    <map:parameter name="date" value="{date}"/>
                  </map:generate>
                  <map:transform src="../../xslt/rc/rco-exception.xsl"/>
                  <map:call resource="style-cms-page"/>
                </map:act>

          <map:call function="editDocument">
            <map:parameter name="sourceUri" value="cocoon:/request2document"/>
            <map:parameter name="noCheckin" value="true"/>
          </map:call>
          </map:when>

          <map:otherwise> <!-- GET -->
            <map:generate src="content/authoring/{page-envelope:document-path}"/>
            <map:transform src="../../xslt/bxeng/change-object-path.xsl">
              <map:parameter name="documentid" value="{page-envelope:document-id}"/>
            </map:transform>
            <map:serialize type="xml"/>
          </map:otherwise>

        </map:select>
      </map:match>
```

5. The image-upload-show, link-show and asset-upload-show steps

BXE supports the notion of callbacks to allow Lenya to display a list of assets, images or links to be inserted into a page. These three steps generate the content of these popup windows, respectively. The `link-show` step is the most complex of these since it takes lots of parameters, such as the currently selected language, the position in the sitetree and the list of available languages. These parameters are necessary to recreate the sitetree visualization from the site area for the link insert popup.

```
      <map:match pattern="image-upload-show" type="step">
```

```
        <map:call resource="cms-screen">
           <map:parameter name="serverpage" value="info/assets.xsp"/>
           <map:parameter name="stylesheet" value="bxeng/image.xsl"/>
        </map:call>
      </map:match>

      <map:match pattern="asset-upload-show" type="step">
        <map:call resource="cms-screen">
           <map:parameter name="serverpage" value="info/assets.xsp"/>
           <map:parameter name="stylesheet" value="bxeng/asset.xsl"/>
        </map:call>
      </map:match>

      <map:match pattern="link-show" type="step">
           <!-- just a dummy xsp since we call the info area directly -->
          <map:generate type="serverpages" src="../../content/info/assets.xsp"/>
         <map:transform src="../../xslt/bxeng/link.xsl" label="content">
           <map:parameter name="infoarea" value="true"/>
           <map:parameter name="contextprefix" value="{request:contextPath}"/>
           <map:parameter name="publicationid"
value="{page-envelope:publication-id}"/>
           <map:parameter name="area" value="authoring"/>
           <map:parameter name="tab" value="en"/>
           <map:parameter name="chosenlanguage"
value="{page-envelope:document-language}"/>
           <map:parameter name="documentid" value="{page-envelope:document-id}"/>
           <map:parameter name="documenturl"
value="/{page-envelope:document-url}"/>
           <map:parameter name="documentextension"
value="{page-envelope:document-extension}"/>
           <map:parameter name="defaultlanguage"
value="{page-envelope:default-language}"/>
           <map:parameter name="languages"
value="{page-envelope:publication-languages-csv}"/>
         </map:transform>
        <map:call resource="style-cms-page"/>
      </map:match>
```

6. The image-upload and asset-upload steps

To enable the upload of new assets to Lenya directly from the BXE `insert Image` and `insert Asset` popup windows, there are two use case steps that use the upload action to process the uploaded file, and then redirect to the respective popup window.

```
         <map:match type="step" pattern="asset-upload">
           <map:act type="upload">
             <map:redirect-to
uri="{request:requestURI}?lenya.usecase=bxeng&amp;lenya.step=asset-upload-show"/>
           </map:act>
           <map:call resource="cms-screen">
             <map:parameter name="serverpage" value="info/assets.xsp"/>
             <map:parameter name="stylesheet" value="bxeng/asset.xsl"/>
           </map:call>
         </map:match>

         <map:match type="step" pattern="image-upload">
```

```
            <map:act type="upload">
              <map:redirect-to
uri="{request:requestURI}?lenya.usecase=bxeng&amp;lenya.step=image-upload-show"/>
            </map:act>
                <map:call resource="cms-screen">
                   <map:parameter name="serverpage" value="info/assets.xsp"/>
                   <map:parameter name="stylesheet" value="bxeng/image.xsl"/>
                </map:call>
               </map:match>
```

7. Further BXE configuration

BXE offers a lot of customization options beyond those outlined above. If you plan to make the most of BXE, you should familiarize yourself with the contents of the BXE wiki (http://wiki.bitfluxeditor.org/Main_Page) .