Ace Editor Integration for Template Editing

The goal of this project is to create the JavaScript components necessary to enhance a server generated HTML page to provide editing of XHTML, XSLT, JavaScript, and CSS files. These files will be listed in the server generated page. The editing component will be the Ace editor. The project will integrate the Ace editor with a basic HTML structure, and server side functionality for retrieving and saving the file contents.

For example, starting with this HTML:

```
<div class="container">

   <div class="file-list">
     <ul>
          <li><a data-
type="xslt" href="../xslt/styleSheetOne.xslt">styleSheetOne</a></li>
          <li><a data-type="html" href="../templates/master.html>master</a></li>
          …
    </ul>
  </div>

  <div class="editor">
    <div id="toolBar"><button>Save</button><button>Reset</button>….. search and
replace...</div>
    <div id="editor-container"></div>
  </div>
  <script type="text/javascript">
     // initialization code goes here…
    $(function(){
       var editor = new namespace.MarkupEditor( {
          editorEL : '#editor-container',
          fileListEL : 'div.file-list'
          urls : [{'type':'xslt', 'url' : '../request/saveXSLT'}, {} … ]
       });
    });
  </script>
</div>
```

The server application will generate this markup, to include the list of files, and required CSS and JavaScript includes for the entire page. Note that each file link caries a "data-type" attribute to determine whether it is an XSLT , XHTML, JavaScript, or CSS file.

The editor will be based on the Ace JavaScript editing component, http://ace.c9.io/#nav=about. We suggest following the basic instructions from

that site, especially http://ace.c9.io/#nav=embedding, which includes using a pre-packaged build. Also reference, https://github.com/ajaxorg/ace-builds/blob/master/editor.html

The JavaScript component that needs to be created is to manage the interaction between the user, the file list, the editor, and the server.

Each individual file can be loaded from the URL provided in the href attribute of the anchor tag. A "click" on one of the links in the file list should load that file into the editor asynchronously, provide an indication that the file is loading, then allow the user to edit the contents.

The component will accomplish the following use cases:

1. Open the file, in the editor, as determined by the hyperlink clicked on by the user.

2. Save the modified contents of the editor. Notify the user that a save is in progress, and notify when the save is completed. The server components at present to not provide much error handling so a simple notification is sufficient if it fails to save. To "save" an edited file will involve an HTTP POST to a configurable, server side, URL. The POST will be an HTML form including the file contents from the editor, the original file URL (from the hyperlink), and the "name" which is the text value of the hyperlink tag, and any HTML5 "data-" attributes included in the "a" tag.

Provision should be made so that there is a set of URL's configurable based on file type — for example, an XHTML file may be POST to a different server URL than an XSLT file. These URL's will be provided to the component as part of an object passed to the initializing (constructor) function.

3. Reset the contents of the editor to it's original value, abandoning any changes. This may simply re-request the original file, and reload the editor.

4. Perform a search, and replace, of a string within the editor. This will involve some additional user interface as well for entering the term, and performing the action.

The following JavaScript libraries/frameworks will be available in the page:

- jQuery
- Bootstrap (V3)
- Underscore
- Backbone
- The page will also contain a script tag to include ace.js. A JavaScript module system, such as Require.js is not in use in the underlying product.

The component should also provide a method to obtain a reference to the ace editor object,

after initialization.

The goal is not to create a complete, fully featured, IDE style editor, but to take advantage of the basic capabilities of the Ace editor to provide a better browser based editing experience than is available with traditional HTML forms and simple textareas.

The deliverable is a JavaScript file containing the integration module, and a copy of the ace.js file used for development to ensure we will deploy with the correct version.