Developer’s Notes

ItsImportServlet utilizes Okapi’s parser to help traverse the imported document. During the import process, the external rules document will always be imported first because a File object needs to be created for the external rules document before the parser can run. The rules from the external rules document will be stored into the Jackrabbit repository before the mark up document is parsed.

As the document is being traversed, each element will be stored as its own node in the jackrabbit repository, while each attribute will be stored as a property. It should be duly noted that text content is saved as an element rather than as a property under the parent node/element. This is due to text content’s sibling could also be an element. For example, you have a paragraph of text with various keywords in bold. The text before the first bold element will be treated as one Text node and it is considered a sibling to the bold element. There is currently no way to determine when certain elements should be treated as a string instead of an element object. So the easiest solution is to simply treat text content as an element and create a node to store the metadata. An added bonus to this solution is that order is preserved for nodes, while properties are not.

Elements with custom prefixes will require special handling. Jackrabbit has the ability to register namespaces on the fly for the current workspace; however, two issues arise from this approach. Documents are allowed to have multiple sets of global rules, so each set uses a different prefix. For example, the head of the document contains a set of global rules with the prefix of “its” and the body of the document contains a set of global rules with the prefix of “i". Since both of these prefixes uses the same namespace URI, the first set of global rules will no longer have the prefix of “its”, but rather “i" once all the nodes are generated in jackrabbit. Another issue is the instance where we import two different documents with the same prefix, but different namespace URIs. The second document’s namespace URI will override the first document’s URI for that particular prefix.

Another approach to elements with prefixes is to register the namespace declarations in the Maven pom file. The flaw to this logic is that there could be custom prefixes on each document and those prefixes will end up throwing exceptions during the import process.

The solution is to simply make sure the node name does not contain a prefix and store the prefix as a property (the node-prefix property). So when we export the document, each element can check if such property exists before generating the element name.

This solution also ties into the problem of namespace declarations. A namespace needs to be declared before the prefix can be used. Jackrabbit sees the namespace declaration as an attribute with a prefix and will throw an error. Using the same solution for element names with prefixes, namespace declaration will need to be saved as another property. Every time an attribute contains “xmlns:” in its node name, the indicator that attribute is a namespace declaration, the prefix is saved as the property name and the prefix is also stored into the “namespace-declaration” property. So during the export process, the property name can be matched to the list of prefixes stored in the “namespace-declaration”. If the property name matches to the list, “xmlns:” will be appended to element name.

ItsServlet contains the logic to traverse the jackrabbit repository and build out the xml/html document.