

# Document Publishing

---

This document will describe the process of how a document gets published by Daisy. What follows below are some first rough notes.

The publishing process is split between a "Publisher" extension component running in the repository server and the frontend running on Cocoon.

The goal of the Publisher component is to gather all information needed to publish a document and return the result to Cocoon as an XML document. This avoids a lot of relatively expensive remote calls that would otherwise need to happen. More specifically, here's what the Publisher component does:

- It starts from the document XML (containing the data of the requested version: the live version, the last version, or some other version)
- For fields and parts, the label of the fields and parts is added to the XML, according to the user's locale. Field values are also formatted using the user's locale.
- For "Daisy HTML" parts, the content of those parts is retrieved, parsed and inserted into the XML
- Queries and query-includes embedded in the document are processed
- Includes are processed, and for included documents, the same operations happen again: merging parts, processing queries and includes, ... It is also here that detection of recursive includes happens.
- Links and images having "daisy:" URLs are augmented with the name of the document.
- The navigation tree is also added to the output

Included documents are not directly inserted at the location of inclusion, rather they will be added "side by side" to the output. This is to allow to execute different XSLs on them on the Cocoon side.

All this processing is done streaming (SAX). (with the exception of some parsing results that are first pushed into buffers -- SaxBuffer instances -- to gracefully handle parsing exceptions)

Cocoon then performs the following tasks:

- The XML retrieved from the Publisher is split into multiple parts: one for general information, and then one for each document (the main document and each of the included documents, if any). These different pieces of XML are stored in temporary buffers (SaxBuffers).
- The main document and each of the included documents are taken through a pipeline using a document type specific XSLT, or a default one if no document type specific XSLT is available, and the result is again stored in temporary buffers (replacing the before-XSLT buffers). Note that each XSLT only gets the XML of one document as input, so the input trees for the XSLTs remain relatively small.
- The XML containing the "general information" (ie general metadata about the document and document version, the navigation tree) is then used to feed the main pipeline which:
  - first applies an XSLT to do the general page layout (with the navigation tree). This XSLT leaves a special tag, `<insertDocumentContent/>` at the place where the actual document should come. Note again that the input for this XSLT is rather small, ie it doesn't include the actual document content and all its included documents etc.
  - a special transformer, the `IncludePreparedDocumentsTransformer`, will insert the styled main document at the location of the `<insertDocumentContent/>` element, and insert the (styled) included documents at the location of inclusion (recursively).
  - the next transformer then processes the inclusion of non-"daisy:" includes (ie includes of cocoon:/ URLs)
  - the next transformer translates the "daisy:" links on `<a>` and `<img >` elements to public URLs.
  - and finally the result is serialized as HTML.

With exception of the XSLTs (for which we keep the input as small as possible) and the temporary buffers containing the (styled) documents, all processing again happens in a SAX-streaming manner.