Logicsheet

See XSP.

A logicsheet is basically just an XSLT transform. It contains templates to match specific custom elements (and attributes?) in the input and replace them with XSP logic and expression elements (i.e. *code embedding directives*). The contents of these elements is the actual code implementing the logic that these custom elements describe.

Logicsheets are the recommended way to integrate custom code with XSP pages.

Logicsheet elements should deal with both static attributes on the element, or nested parameter elements. They should perform parameter checking, and signal errors if necessary during page compilation.

Associating Pages with Logicsheets

Elements are associated with logicsheets by their namespace. Logicsheets are registered with Cocoon in the `cocoon.xconf` file. (See XindiceLogicsheet for an example of this configuration.)

Design Guidelines

In short: use helper classes.

Refactor as much code as possible into Java static helper methods. **[Q: where do I put the methods and how do I access them from the generated XSP?]** This has the following benefits:

- improves testability
- avoids variable name clashes (e.g. with those defined by another logicsheet's code)
- avoids possible side-effects of parameter values
- avoids ambiguous class references due to multiple imports of similarly named (but not qualified classes)
- localises errors to parameter values, not messy generated code
- reduces size of generated code (refactors common code to single place)

There are a number of BuiltInLogicsheets

Development hint (pseudo-protocol)

In most tutorials the registration of a custom logicsheet (in cocoon.xconf) uses the **resource:** pseudo-protocol. Logicsheets accessed via the **resource:** protocol are not reloaded when changed, so if you make a change, you must restart your servlet container to see the change take effect.

As described on this German page Cocoon Logicsheet Tutorial it is also possible to use the **context:** pseudo-protocol, so when you are just creating a straight forward XSLT logicsheet, it will be evaluated on each http request. This is much more convenient for development, although it would probably decrease performance in a production environment.

```
<builtin-logicsheet>
    <parameter name="prefix" value="mylogic"/>
    <parameter name="uri" value="http://hostname/mylogic/1.0"/>
    <!-- <parameter name="href" value="resource://mylogic/mylogic.xsl"/> -->
    <parameter name="href" value="context://taglib/mylogic.xsl"/>
</builtin-logicsheet>
```

so you can store your logicsheet at a place like COCOON_WEBAPP_HOME/taglib/mylogic.xsl