Modules

for beginners

Status of this document is draft.

There are three types of Modules: InputModules, OutputModules and DatabaseModules. Modules are introduced and configured in the cocoon.xconf. The documentation for the latter two is not easy to understand, so here's some newbie information on those.

While developing any Cocoon application, set log level to debug in web.xml to see all relevant messages. Otherwise it can be very frustrating to develop without any output from Cocoon.

Please insert any comments you have on this subject or move useful parts of this document to the other more relevant docs.

Input Modules

Input modules can be used in various places, including the sitemap. These are well documented in InputModules and Cocoon documentation. There are good examples in the Cocoon distribution (samples > Input Modules).

When using input modules, the execution order of different components needs to be taken in to account. The following was written using information in an email from Christian Haul. The sitemap setup is done before execution of the pipeline and the input modules are processed in this phase - after this they cannot be altered from sitemap's viewpoint (with XSP or anything). Action are processed before XSP scripts, so no XSP script can alter i.e. request attributes so that they could be used inside an action (i.e. modular database actions, see below).

Writing your own actions is not a simple task, but there's a TestAction.java on the scratchpad in 2.1.2 that can be used as a sample. It uses modules for input and output. To avoid making your own actions, you can execute XSP-code with some restrictions inside an action with XSPAction. This worked fine in my case, but you can only use xsp:structure and xsp:logic parts of the XSP. I used XSP Actions to pre-process form input from user's browser and to set request attributes to be used inside a modular database action.

A WhatIsFlow based approach could be more sophisticated solution. Other alternatives could include writing a custom input module (that does the necessary logic) or combination of several modules.

Output Modules

The name would suggest that you could use these modules as easily as you can use input modules. This is not the case however as you can't use these directly from the sitemap. You need to have an action or some other Avalon component to achieve this.

The most common use case for these seems to be to return values trough attributes to the sitemap from ModularDatabaseActions as they can be read easily elsewhere with Input Modules. There's also an example in the Cocoon distribution in samples > blocks > databases.

There are also some other actions where you can use these, see this email for more information.

Database Modules (AutoIncrementModules)

Database modules are used in ModularDatabaseActions, and are not to be used directly. These modules actually take care of autoincrement columns for different database systems and are named therefore AutoIncrementModules. You need to select the correct module for database or write a new one for your database system. See Modules for more information.

The following example from cocoon.xconf specifies HSQL-compatible AutoIncrementModule:

```
   <!-- =============== Database Modules ==================== -->
        <autoincrement-modules>
                <component-instance
class="org.apache.cocoon.components.modules.database.HsqlIdentityAutoIncrementModule"
logger="core.modules.auto" name="auto"/>
<!--
Choose the one suitable for your DBMS. You *can* have more than one at a time,
but they need to have different names. You then need to specify explicitly,
which one to use in your descriptor file.
-->
      <component-instance logger="core.modules.auto"
name="auto" class="org.apache.cocoon.components.modules.database.ManualAutoIncrementModule"/>
      <component-instance logger="core.modules.auto"
name="auto" class="org.apache.cocoon.components.modules.database.IfxSerialAutoIncrementModule"/>
      <component-instance logger="core.modules.auto"
name="auto" class="org.apache.cocoon.components.modules.database.MysqlAutoIncrementModule"/>
      <component-instance logger="core.modules.auto"
name="auto" class="org.apache.cocoon.components.modules.database.PgsqlAutoIncrementModule"/>
-->
        </autoincrement-modules>
```