Components

Components are the basic units of work in the Cocoon architecture. All components are declared and configured in the Sitemap.

All Cocoon components have a name and are associated with a concrete implementation. The sitemap includes parameters that are passed to the component when it is instantiated. Parameters are component specific.

Because most of these components are interfaces, additional implementations can be created and configured in the sitemap without changing the Cocoon server.

This is the current list of components. There are generally several implementation of each components.

- Matchers
- Generators
- Transformers
- Readers
- Serializers
- Selectors
- Actions
- Pipelines

Selectors, Matchers and Actions are all CocoonConditionals

Component Declarations.

There is a canonical form for component declarations:

```
<map:component-types default="component-name">
 <map:component-type name="component-name" src="implementation">
   <!-- component-type specific parameters -->
 </map:component-type>
</map:component-types>
```

1. Where `component-types` is one of transformer, reader, etc.
2. Each component must have a unique name within its type
3. Each component must specify its implementation, there can be multiple declarations with the same impl.
4. Parameters are component specific.
5. A default component instance can be named

The detailed description about each type of component includes a section on how to declare them.

Pooling

Cocoon components can be pooled. The pool management is actually provided by the Avalon framework.

There are three attributes that can be added to component declarations to control the pooling parameters.

- pool-min – default is 2
- pool-grow – default is pool-min
- pool-max – default is 8

These attributes control the number of component instances pooled by Cocoon. Once pool-max is reached new instances are created, but are destroyed when returned.

Writing Cocoon Components

Cocoon relies on a number of Avalon interfaces.

- Composable
- Configurable – can take parameters in declaration; all can take parameters

from pipeline

- Poolable – marker interface, can be pooled, otherwise created from factory
- Disposable – needs to release resources