AvalonMusings

(this page is part of the wiki materials for ApacheAvalon; Avalon main page in the wiki is FrontPage)

Object Naming

Avalon has followed the principle that concrete objects should avoid directly interacting, but rather interfaces should be used liberally for maximum flexibility. The way the current AvalonNamingPolicy has been working, there are a lot of simple concrete classes that go by the name of Default(interface name). For example, DefaultConfiguration or DefaultServiceManager. Does the word 'Default' convey anything of value? It might be better to name the objects for what they really do. For example, the name MapServiceManager would convey the idea that it's really just a simple adaptor to a Map.

Auto-Generated Configuration

I know this is a pipe-dream, but here's an idea. If component uses the accessors on a Configuration object that supply defaults, a complete and consistant Configuration object could be built every time the component is initialized. What if the Configuration could be saved back to whatever medium it came from (XML or some other format)?

For example, a configuration file looks like this:

```
<components>
  <component name="Widget" class="com.WidgetClass">
    <color>green</color>
    <oldParamName>true</oldParamName>
  </component>
</compoenents>
```

You start the container. This code runs:

```
class WidgetClass implements Configurable
{
  public void configure(Configuration config) throws ConfigurationException
  {
    m_maxSize = config.getInt("mamimumWidgetSize", 5);

    m_color = config.getString("color", "red");

    // auto-upgrade
    if (config.getString("oldParamName") != null)
      config.rename("oldParamName", "newParamName");

    config.saveChanges();
  }
}
```

Suddently the container rewrites the configuration file:

```
<components>
  <component name="Widget" class="com.WidgetClass">
    <color>green</color>
    <maximumWidgetSize>5</maximumWidgetSize>
    <newParamName>true</newParamName>
  </component>
</compoenents>
```

I know, I'm dreaming. It'd require a special XML parser that didn't muck up the formatting or remove comments and it would require other weirdness. On the plus side it would mean that keeping XML descriptors and actually classes in sync would be easier. Anyway, just an idle thought.

Comments from Niclas:

1. XML Parsers doesn't remove comments or any whitespace. The serializers normally do. 2. Changing the structure of the configuration (i.e. rename) is IMO BAD! The same code won't run a second time. 3. Is *Configuration* related to a setup or is it related to state of a component? If it is setup, then it should not be changable, and state should be handled outside of Configuration.