AvalonBrainStorming

http://jakarta.apache.org/avalon/images/header.gif

(this page is part of the wiki materials for ApacheAvalon; avalon main page in the wiki is AvalonProjectPages)

Component Scripting Language

By providing a terse and very simplified "near english" scripting language, we can interact with our existing Components in high level manner. The language interpreter/compiler would hide the details of looking up and releasing the components that are used. The scripting language would work best in the context of a "Command" component where a function or command was issued by user interaction of some sort.

The biggest benefit would be to help the suites (you know, the business experts) to define how the general logic will work. If there needs to be some tweeking at a lower level, the work can always be handed off to a developer to implement the hard logic. As components can use other components, the scripting language can also be used as a quick way to write web services...

Here is an example of how it might look:

```
use "org.d_haven.business.Command" as myCommand.
```

```
result is from myCommand:execute with "context:foo", "context:bar"
```

Keywords would be something like this:

- use "${role}" – find a component with the role specified in quotes
- as ${name} – assign the result to the name supplied
- ${name} is – assign the result to the name supplied
- from ${component}:${method} with ${arglist}

Meta Information Based Component Discovery

Component querying is a very powerful way to interact with a container and get the best match for your component possible. The problem with a traditional "query" language is that it places too much power in the hands of the component. In essence you are providing a mechanism to wrest control from the container.

By using a declarative way to describe component dependencies, we can find the best match for a component that a container can provide without explicitly stating as much. Even better, we can place the validation and verification of the component matching at assembly time instead of at runtime.

By declaring a set of desired attributes we can find the one component out of several that might exist. It is also the best way to get rid of the need for the "ServiceSelector". It allows us to provide enough assembly time information to the container so that it can verify the system before we set it up.

LSD says: use Jython as a base 😃

NicolaKen says: use Jython as a base 😃

Noel says: Use BSF as the base, then stick in whatever standard or ad hoc language is desired.