BestPractices

Part of AvalonProjectPages.

Lifecycle circumvention

Tired of having to implement the lifecycle methods every time, not worried about performance, and already know which lifecycle stages apply to all of your components?

Note this is a "worst practice" 😃

Read:

http://marc.theaimsgroup.com/?l=avalon-users&m=104037997604436&w=2

Facades and the Lifecycle

Providing an avalon component wrapper around an existing non-avalonized application?

Read:

http://marc.theaimsgroup.com/?l=avalon-users&m=104694541504301&w=2

lookup stuff whenever it makes sense

you can call ServiceManager.lookup(ROLE) from inside the Servicable.service() method, the Initializable.initialize() method, or from any other place that makes sense. The same is true for context and configuration lookups.

It is often convenient to look things up as early as possible, which is what often happens.

Read:

http://marc.theaimsgroup.com/?t=104686717700005&r=1&w=2

Use AbstractLogEnabled

No mailing list thread to accompany this one, but if you have an avalon component implementation which doesn't extend AbstractLogEnabled, you're probably doing too much work. It is a safe bet you always need to do some logging 😃

Multiple providers of the same role

You can use a very convenient [] syntax in phoenix. In other containers, you should do the <Foo>Manager approach detailed below.

A <Foo>Manager

interface FooManager
{

```
  Foo aquireFoo( int param1, String param2 );
```

```
  void releaseFoo( Foo foo );
```

}

often make more sense than ServiceSelectors.

Read:

- http://marc.theaimsgroup.com/?t=104567450500007&r=1&w=2
- http://marc.theaimsgroup.com/?t=104626693600002&r=1&w=2