IntroductionToEmbedding

This is a little introduction to embedding Merlin. The descriptions starts off with an overview of the Avalon Repository bootstrapping system - which at the end of the day is the core embedding technology used within the merlin platform.

Simple Embedding Example

1. creating an initial context

```
File cache = new File( "my-cache" );
InitialContext context = new DefaultInitialContext( cache );
```

The above code defines a cache directory for the repository system to use when loading resources needed in your embedded application - and provides the directory as an argument when creating a new InitialContext object. The InitialContext is you hook into the repository system and the embedding machinery.

2. declare what you want to embed

```
String spec = "artifact:avalon-logging/avalon-logkit?version=1.0-SNAPSHOT"
Artifact artifact = Artifact.createArtifact( spec );
```

An artifact is a logical reference to a jar file (or other type of resource) that the repository can access. The avalon-repository system uses artifact references as the key to locating meta data about embedded classloaders. The classloader meta data is maintained as a properties file with the .meta extension. For example the above artifact meta address translates to:

```
[host]/avalon-logging/jars/avalon-logkit-impl-1.0-SNAPSHOT.jar.meta
```

The content of the meta file is automatically generated using the avalon-plugin avalon:artifact goal. Two real examples are attached (the logkit factory and the merlin factory meta).

The contents of the meta file includes:

- a ordered list of jar files that are required to construct a classloader for the embedded application
- the name of a factory class to be used as the embedded instance creator

3. create the factory

Using the initial context and the artifact you now have everything you need to create you embedded instance.

```
Builder builder = m_context.newBuilder( artifact );
Factory factory = builder.getFactory();
```

Behind the scenes the avalon-repository system has gone off, pulled down the meta data, downloaded and cached all of the classloader entries, constructed a new classloader, and instantiated the factory.

4. parameterizing the factory

The factory object is the central entry point of the embedded application - it is responsible for instantiation of the embedded instance based on a supplied criteria. The initial criteria (the defaults) are established by the factory in response to the following operation:

```
Map criteria = factory.createDefaultCriteria();
```

Based on the documentation about the facility your embedding you can update the criteria using application specific keys. All of the Merlin related criteria instances use the avalon-util Criteria as the map implementation. This provides support for key and type validation under the put operations and type coercion on get operations.

For example:

```
String key = "avalon.logging.configuration";
File file = new File( "logging.xml" );
criteria.put( key, file );
```

Parameterization of the criteria is typically different for each embedding scenario. A CLI handler will for example adapt to the command line operations and set criteria values accordingly. An application may set the criteria based on parameters declared in a web.xml file. Typically the embedding class acts as the adapter between the embedded context and the factory.

4. embedded instance creation

Creation of the embedded instance is now a simple one line operation:

```
      Object object = factory.create( criteria );
```

The object that is created is the embedded application. In this example its a logging manager that uses the logkit implementation. However, it could have been the Merlin kernel. The only difference between the Merlin scenario and the logging manager scenario is the initial artifact and the actions taken to parameterize the criteria.

Examples of Embedding

merlin/logging/logkit

Package contains an example of a logging manager that is established as part of merlin establishment. Key classes include the DefaultLoggingCriteria and DefaultLoggingFactory - the test package demonstrates the complete deployment of the logging manager with nothing more than avalon-repository-main in the project.xml dependencies.

Usage of this factory can be seen in the merlin/kernel/impl package - DefaultFactory class under the createLoggingManager operation.

merlin/kernel/impl

The kernel/impl package defines the factory that is used to create a new Merlin Kernel. Examples of embeddors that use this factory include:

- merlin/kernel/cli - Main class
- merlin/kernel/plugin - MerlinBean
- merlin/kernel/servlet - MerlinServlet
- merlin/kernel/unit - AbstractMerlinTestCase

Please note that there are some small differences between the 3.2 and 3.3 versions of the APIs. The above description of based on 3.3 (CVS HEAD).