ServiceImplementationSwitch

Problem Description

AchimHuegen, August 9 2004

This proposal is about multiple implementations of the same service interface. The problem has already been described in this proposal: ConditionalContributionsProposal.

I dont't think that the example used there (pick implementation depending on jdk version and library availability) is a very common use case. For me the most important use cases are:

- Select most appropriate implementation for your application out of a set of possible candidates (e.g. security realms: JDBCRealm, JNDIRealm, MemoryRealm)
- Switch all services of a middleware connector to dummy implementations

Proposed Solution

I have one problem with the solution from ConditionalContributionsProposal:
The decision how to switch implementations technically is not made at the application level. E.g. the first provider of a service implementation decides to use a system property. Now all applications that use this service must provide this system property. But different providers means different methods and finally your application has to deal with inconsistent implementation selection.

So here is my approach:
Implementations get a name and via a contribution a mapping from service to an implementation is defined by referencing its name.

```
<service-point id="foo.service" interface="foo.MultipleService" />

<implementation service-id="foo.service" >
    <create-instance class="foo.serviceImplDefault" />
</implementation>

<implementation service-id="foo.service" >
    <create-instance name="implementation1" class="foo.serviceImpl1" />
</implementation>

<contribution configuration-id="hivemind.ServiceImplementationSwitch" >
    <switch service-id="foo.service" implementation="implementation1" />
</contribution>
```

This example shows a service point with two implementations. The first implementation is the default and has no name. The second one is named "implementation1". The contribution tells the registry to use the latter one when creating instances of this service.

Default implementation

If only one implementation exists it is the default. If multiple implementations exist and one is unnamed, then its the default. If no mapping is defined, then the default implementation is used.

Symbol sources

You can use symbols in the configuration immediately. E.g. a system property:

```
<contribution configuration-id="hivemind.ServiceImplementationSwitch" >
    <switch service-id="foo.service" implementation="{propertyFooService}" />
</contribution>
```

Overriding services

It's easy compared to http://jakarta.apache.org/hivemind/override.html. Assuming that a service point is already defined and a default implementation is provided you could add a new implementation and make it the default this way:

```
<implementation service-id="foo.service" >
    <create-instance name="implementation1" class="foo.serviceImpl1" />
</implementation>

<contribution configuration-id="hivemind.ServiceImplementationSwitch" >
    <switch service-id="foo.service" implementation="implementation1" />
</contribution>
```

Extensions

Wildcards

Switch all implementations matching a wildcard. More concrete definitions have higher priority:

```
<contribution configuration-id="hivemind.ServiceImplementationSwitch" >
    <switch service-id="foo.*" implementation="dummy" />
    <switch service-id="foo.connectors.*" implementation="local" />
</contribution>
```

Expression Language

The expression language from ConditionalContributionsProposal could be integrated.

```
<contribution configuration-id="hivemind.ServiceImplementationSwitch" >
    <switch service-id="foo.service" implementation="implJdk14"
        condition="jdk(1.4)" />
</contribution>
```