InjectingNonServiceFactoryObtainedObjects

Q. Injecting Non-Service, Factory Obtained, Objects

(DanielFeist 21/06/2004)

(Maybe this is a ChangeProposal rather than a question. That depends on the answer ... lets wait and see and possibly move it is it doesn't belong here) Is there anyway to inject a service, using the BuilderFactory, with an object obtained by invoking a method of another service thus allowing non-service objects to be obtained from a POJOF (plain old java object factory) for injection? I can think of many cases were this would be very useful but it doesn't seem to be possible. Concrete examples are as follows:

Example 1: I have a Mailer service which depends on, and uses, a datasource to obtain email address. This datasource is to be obtained via JNDI so i decide to use the NameLookup service to obtain the datasource. As the datasource is not a service but rather the result from executing a lookup method of a service it seems i am limited from injecting it into my Mailer service through declarative configuration in the hivemodule.sdl.

Example 2: I have a Mailer service which uses a Properties to obtain email address. The backing store of the properties has to be flexible and configurable so i decide to create a PropertyFactory service that returns a Properties object for a given locator and whose backing store is defined and configured through a configuration point. This is a good solution to my problem but as in example 1 I cannot inject this properties into my Mailer Service.

On a side note: If this type of injection is possible then a clean solution to my change proposal ModuleResourcesProposal is very simple. Resources would be created using a factory specifically designed to do this (much like the BeanFactory) which would also manage backing-store type and synchronization issues. These resources could then be injected into services by using ResourceFactory's get method to obtain the resource.

A.

HowardLewisShip: Seems that BuilderFactory could do this, by changing the nested `set-service` element to have an additional attribute, `read-property` that would be a property of the service to read as the value to assign to the service implementation under construction. Perhaps `set-service` should be left alone, and there should be a `read-service-property` element added instead.

DanielFeist:
What we are actually want to set here (in order to accomplish what is explained in the examples above) is not as you mentioned the property of another service (i.e. allowing the copying properties between servcies) BUT rather the value obtained from invoking a method of another service. Something like the following would be needed as a nested BuilderFactory element in order to set a property of the service in construction with an object (in this case a datasource) obtained from another factory/lookup service:

```
set-service-method-return-value (property=datasource service-id=hivemind.lib.NameLookup method=lookup){
  argument{
    java:/jdbc/myDS
  }
  argument (translator=class)
    javax.sql.DataSource
  }
}
```

.

Obviosly the name i have put 'set-service-method-return-value' is rather long and would need to be improved. I've put that name in the example so it is clear what in fact the functionality is.

HowardLewisShip: Going down this path, you quickly get to the point where you want to use OGNL for this ... or a full scripting language. I don't have an objection to that as an add-on, but don't think it is right for the core module. The fact that you can add this (by creating your own service implementation factory) is good enough.

DanielFeist: What about including just the basics of this functionality in the core module? I am keen on seeing this available to be able to do such simple things such as inject an object obtained from the BeanFactory or similair without the complexity of having to implement and define your own service implementation factory. For more complex situations, where more than one String attribute is required yes its fair enough to have extend the BuilderFactory.

```
set-service-method-value (property=myBean service-id=my.lib.MyFactory method=get argument=myBean)}
```

The addition of this functionality to the BulderFactory doesn't actually require changes to the core module implementation but rather just the addition of an element in the BuilderFactory schema definition and a translator. Is there anyway of extending BuilderFactory's schema defined in core hivemind module?

HowardLewisShip: I think there may be a proposal about making it easier to reuse and extend portions of a schema. BuilderFactory's parameter's schema is massively complex.

An OGNL builder parameter element might look something like:

```
set-expression (property=datasource expression=<<
  service["hivemind.lib.NameLookup"].lookup("java:/jdbc/myDS")
>>)
```

This would imply a magic `service` property attached to `Module` that could invoke `getService(String,Class)`. I may end up building something like this for Tapestry, since Tapestry already has an OGNL dependency.

DanielFeist: What then, in terms of HiveMind 1.0 (assuming no OGNL) is the solution to the simple problem put forward in the two initial examples? Implement and define in sdl my own CustomBuilderFactory (the implementation of builder factory would be identical but with a slightly more complex schema)? There's no simpler solution? I think considering OGNL or something similar to make builder paramenters simpler and more flexible for the next version is a good move.

HowardLewisShip: A completely reasonable and testable solution is to assign the service (i.e. `hivemind.lib.NameLookup` to a property of your service, and use the `initialize-method` attribute to have your service initialize. In Java code, you can access the methods of the collaborating services.

DanielFeist: Yes that's a good workable solution the downside it that its not 100% IOC i.e. the DataSource is not being injected but rather work has to be done in the services init method to obtian the datasource. In HiveMinds current state that is the best solution I suppose. I do believe though that the ability to be able to inject other objects that are not services or configurations (obtained from a lookup/factory service) is an important functionality that should certainly be considered as not everything we might want to use and inject can or should de represented as a service.

DanielFeist: I've just been looking at the BeanFactory and translator and it would seem that you have actually started implementation of something that allows the us to obtain objects from a factory. The PiplineFactoryService, for example uses, in its configuration contributions, beans obtained from the Bean Factory. This is good and works well for the needs of the PipelineFactoryService configuration but it would be far more flexible to be able to obtain an object from any POJOF (plain old java object facotry) and not just from a BeanFactory. Is there anyway of doing the same sort of thing but allowing the use of arbitary user implemented factories (not neccessarly implementing BeanFactory)? Maybe it would be possible to extend the ObjectTranslator or BeanTranslator to do this.