FrequentlyAskedQuestions

Frequently Asked Questions

This page will eventually become the HiveMind FAQ. The idea is that people ask questions and that other HiveMind users and developers respond and comment until a "accepted solution" is reached. Eventually we will be able to compile a list of questions and accepted solutions into a more official FAQ for the HiveMind website.

Question 1): InjectingNonServiceFactoryObtainedObjects (DanielFeist 21/06/2004)

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?

Question 2): UsingJNDIToObtainHiveMindServices (DanielFeist 24/06/2004)

I have been experimenting with the use of JNDI as a facade to HiveMind by implementing a simple JNDI SPI. The *lookup* method of a JNDI *Context* takes just a *Name*, yet to obtain a HiveMind service i need to be able to specifiy the interface expected. What i have found is that i can, in place of the interface expected put `Object.class` and everything works ok. *Object* is not an interface so i'm not quite sure why this works. Is this intentional? Will HiveMind continue to support this?

Accepted Answer:

HowardLewisShip: Yes, passing `Object.class` will continue work but it is not recommended in *normal* cirmcumstaces. The idea of passing in the expected (assignable) type is to allow HiveMind to do a check that the service object or proxy returned is assignable. Better a good message from inside HiveMind than a bad ClassCastException. Using java.lang.Object is acceptible if you don't care about that test.

Question 3): PuttingHivemindConfigurationFilesOutsideAnEAROrWAR (TomDavies 24/09/2004)

I have configuration data which may change reasonably often, and which I'd like an administrator to be able to change without rebuilding an ear/war file. What's the best way to make this available to HiveMind?

Answer

Michael Frericks: You could add files that reside outside the classpath as modules this way:

```
ClassResolver resolver = new DefaultClassResolver();
RegistryBuilder builder = new RegistryBuilder();

// process classpath modules
builder.processModules(resolver);


// process modules located anywhere else
File[] modules = getHiveModules(confPaths);

for (int i = 0; i < modules.length; i++)
{
    File module = modules[i];
    builder.processModule(resolver,
        new URLResource(module.toURL()));
}

Registry registry = builder.constructRegistry(Locale.getDefault());
```

Question 4): How can I use HiveMind to lookup my EJB remote component interfaces? (aka ServiceLocator pattern) (DavidKarlsen 03/10/2004)

I'd like to lookup my EJB beans and receive the narrowed remote component interface via HiveMind.

Answer

David Karlsen: Use the hivemind.lib.EJBProxyFactory:

```
<module id="com.mypackage" version="1.0.0">
         <service-point id="SessionFacade" interface="com.mypackage.SessionFacade">
         <invoke-factory service-id="hivemind.lib.EJBProxyFactory">
                 <construct
                 home-interface="com.mypackage.SessionFacadeHome"
                 jndi-name="ejb/com/mypackage/SessionFacadeHome"/>
         </invoke-factory>
         </service-point>
</module>

Then in your code:
//initialize Registry registry some way
SessionFacade sf = (SessionFacade) registry.getService(SessionFacade.class);
```

Question 5): How does one explicitly enable logging within Hivemind? (mferris 03/29/2004)

When I run, I get the spew below, which I see comes from the guts of the RegistryBuilder. I specify an interceptor class, but the problem is that Hivemind is not instantiating my service, hence I get no interception. Another service point in the same hivemodule.xml DOES get instantiated. I get a runtime error of: Unable to construct service tradeworkflow.TradeLifecycleService: Failure invoking constructor for class trade.workflow.service.TradeLifecycleService (at META-INF/hivemodule.xml, line 9, column 86)

The hivemodule line and column referenced contains the > symbol (the end of the <construct class line)

L O G G I N G C O N F I G U R A T I O N E R R O R

Logging is not enabled for org.apache.hivemind.impl.RegistryBuilder.
Errors during HiveMind module descriptor parsing and validation may not be
logged. This may result in difficult-to-trace runtime exceptions, if there
are errors in any of your module descriptors. You should enable error
logging for the org.apache.hivemind and hivemind loggers.

I attempted to pass in -Dorg.apache.commons.logging.Log=org.apache.commons.logging.impl.Jdk14Logger -Dorg.apache.hivemind=SEVERE at the command line, but this has no effect.