Examples

Are there any more "advanced" complete examples about HiveMind than the Calculator example? It would be nice to have, for example:

- How to fetch java.sql.Connection using a HiveMind service.
- More interceptor docs?

SteveGibson :
The cool thing about HiveMind is that using services is no more complex than the simple calculator example. (Oh wow, I just realized the example is a whole calculator now, not just an Adder!) Configuration schema examples are something that would probably be more beneficial.

As for the interceptor - here is an ugly example of an interceptor for the Divider (off the top of my head):

```
public class DividerInterceptor extends AbstractServiceInterceptorFactory
{
    protected void addServiceMethodImplementation(
      ClassFab classFab,
      String methodName,
      Class returnType,
      Class[] parameterTypes,
      Class[] exceptionTypes
      )
    {
        if (methodName.equals("divide"))
        {
            BodyBuilder builder = new BodyBuilder();

             builder.begin();
             builder.addln("if ($1 == 0)");
             builder.addln("{");
             builder.addln("throw new ArithmeticException(\"Division by zero\");");
             builder.addln("}");
             builder.addln("return ($r) _inner." + methodName + "($$);" );
             builder.end();
             classFab.addMethod(Modifier.PUBLIC,
               methodName,
               returnType,
               parameterTypes,
               exceptionTypes,
               builder.toString()
             );
        }
    }
}
```

MarkoKocic:
Ok, here's how to fetch Cayenne DataContext. Is there a better way?

```
hivemodule.sdl
module (id=markko.hivemind version="1.0.0") {
        service-point(id=CayenneService interface=markko.hivemind.CayenneService) {
                create-instance(class=markko.hivemind.CayenneServiceImpl)
                interceptor (service-id=hivemind.LoggingInterceptor)
        }

        service-point(id=DataService interface=markko.hivemind.DataService) {
                invoke-factory (service-id=hivemind.BuilderFactory)                {
                        construct (class=markko.hivemind.DataServiceImpl)                          {
                                set-service (property=cayenneService service-id=CayenneService)
                        }
                }
                interceptor (service-id=hivemind.LoggingInterceptor)
        }
}

public interface CayenneService {
        public DataContext getDataContext ();
}

public class CayenneServiceImpl implements CayenneService {
        static {
                Configuration.bootstrapSharedConfiguration(CayenneService.class);
        }

        public DataContext getDataContext() {
                return Configuration.getSharedConfiguration().getDomain().createDataContext();
        }
}

public interface DataService {
        public Korisnik authenticate(String username, String password);
}

public class DataServiceImpl implements DataService {

        private CayenneService _cs;
        public void setCayenneService(CayenneService cs) {
                _cs = cs;
        }
        private DataContext getDataContext() {
                return _cs.getDataContext();
        }

        public Korisnik authenticate(String username, String password) {
                DataContext ctx = getDataContext();

                Map params = new HashMap();
                params.put("username", username);
                params.put("password", password);
                Expression qual = Expression.fromString("username = $username and password = $password").
expWithParameters(params);

                SelectQuery query = new SelectQuery(Korisnik.class, qual);

                List list = ctx.performQuery(query);
                log.debug("size=" + list.size());
                if (list.size() == 1)
                        return (Korisnik) list.get(0);
                else
                        return null;
        }
}
```

SteveGibson: Well, you could always EagerLoad your CayenneService, and do the work in an init method. You could then change your hivemodule.sdl:

```
service-point(id=CayenneService interface=markko.hivemind.CayenneService) {
  invoke-factory(service-id=hivemind.BuilderFactory) {
    construct (class=markko.hivemind.CayenneServiceImpl) {
      initialize-method (method=init)
    }
  }
  interceptor (service-id=hivemind.LoggingInterceptor)
}
```

Excuse any sdl errors - I am using the xml format (*fixed*). I assume your LoggingInterceptors are there for testing? My take on the logging interceptor is that you can swap it in for debugging (and I want to investigate doing this at runtime) and leave it out when your code is in production, as your production code still contains the appropriate logging code.

Maybe someone with more knowledge with Cayenne could help you out with this. We use Hibernate and have another service which sits on top. This is how our code was pre-HiveMind (I am looking at persisting configuration before we go full steam on using HiveMind in this part of the application.) We are using Hibernate and OSCache, so there are 3 properties/xml files that form parts of the configuration before you factor in application specifics.

ChristianMittendorf: Here is an idea for a HiveMind service providing CayenneIntegration.