Guice JMS Example

Guice JMS Example

Available as of Camel 1.5 onwards

The Guice JMS example is functionally similar to both the first example and the Spring Example but using Guice as the Dependency Injection framework.

In this example we just write RouteBuilder implementations, then we write a Guice module MyModule to create the CamelContext, bind any RouteBuilder instances and configure any components and endpoints, then we create a guicejndi.properties file to bootstrap Guice and Camel.

To run the example we currently use the maven exec plugin. For example from the source or binary distribution the following should work

```
cd examples/camel-example-guice-jms
mvn compile exec:java
```

What this does is boot up the Guice based JNDI provider from guicejndi.properties file on the classpath. This then bootstraps the Guice injector and loads whatever Guice modules are defined in the guicejndi.properties file - then injects the remaining properties in the file.

Configuring Components

If you see the **jms()** method of the Guice MyModule you will see it is annotated with **@Provides** to indicate to Guice that it is a provider and it is annotated with **@JndiBind("jms")** to bind it to the JNDI name **jms** when it is created.

This method then configures the component. The provider method is parameterized by the **@Named("activemq.brokerURL")** property which is injected from the guicejndi.properties file. This lets you define the properties which should be environment specific (such as URLs, machine names, usernames /passwords and so forth) while leaving all of the other configuration which does not change in different environments in Java code.

So you can use Guice to dependency inject whatever objects you need to create, be it an Endpoint, Component, RouteBuilder or arbitrary bean used within a route. Then you can inject any properties from the jndi.properties file easily - so that most of your configuration is all in Java code which is typesafe and easily refactorable - then leaving some properties to be environment specific (the guicejndi.properties file) which you can then change based on development, testing, production etc.