


🔗 branch: master ▾

smx-bootstraps / README.md

☰ 📄

 kevinearls on Jan 14 Initial commits for 6.1 release3 contributors 

📄 file 152 lines (105 sloc) 10.141 kb

Edit

Raw

Blame

History

Delete

## TODO change fuse versions

This projects contains a set of OSGi bundles that bootstrap the use of JBoss Fuse 6.1.0. It is intended as a starting point for the creation of additional bundles, and as a guide to using Fuse feaures.

The Ping Pong bootstrap is used to show inter-bundle request-response communication using ActiveMQ.

1. A Pinger defines a Camel route which periodically triggers a message to be sent and prints out the response.
2. A Ponger listens for ping messages on a known queue, and responds by invoking an OSGi Blueprint service defined in an implementation bundle to generate the response.

## Project layout

The Maven projects contained within are as follows:

- `smxb-features` - Contains an XML features file used to install the rest of the bundles.
- `smxb-pinger` - Contains a bundle that periodically sends a ping message over the embedded ActiveMQ instance. Also exposes a RESTful web service on port 9090 that allows you you ping the service manually.
- `smxb-ponger` - Listens on the queue and replies after invoking an OSGi Blueprint service to generate a response. Contains an additional NMR-based route as an alternative mechanism for chatting with the pinger in the same Fuse instance.
- `smxb-ponger-service` - Contains the implementation of that service.

There is also an additional parent project `came1-bundle` that simplifies the Maven project configuration.

## Prerequisites

Set up JBoss Fuse by downloading the latest version from [Red Hat](#).

Ensure that Maven is set up on your system.

## Installation

Download this project and run

```
smx-bootstraps> mvn clean install
```

For this example, we need to add some credentials to the `$_JBoss_FUSE_HOME/etc/users.properties` file. To add a user with username `admin`, password `admin` and role `admin` we append or uncomment the following line:

```
admin=admin,admin
```

Start up JBoss Fuse

```
$JBoss_FUSE_HOME> bin/fuse

      _ _ _ _ _
     | | _ \       | | _ _ _ |
     | | |_) | _ _ _ _ _ | | _ _ _ _ _
 _   | | _ < / _ \ _ _ / _ _ | | _ _ / _ _ / _ \
| | _ | | |_) | ( _ \ _ \ _ | | | | _ \ _ \ _ /
 \ _ / | _ / \ _ / | _ / _ / | | _ \ _ | _ \ _ |

JBoss Fuse (6.1.0.redhat-312)
http://www.redhat.com/products/jbossenterprisemiddleware/fuse/

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
```

You may need to wait for a while as Fuse starts the bundles it needs to run fully. Running the `list` command should show around 200+ bundles all Active.

```
JBossFuse:karaf@root> list
```

Install the `smxb-features` features file. This gets pulled out from your local Maven repository, and defines which bundles you mean to install for the bootstrap project.

```
JBossFuse:karaf@root> features:addurl mvn:com.fusesource.examples/smxb-features/1.0-SNAPSHOT/xml/features
```

You can now check that the features defined in that file are available for installation:

```
JBossFuse:karaf@root> features:list | grep smxb
[uninstalled] [1.0] ] smxb-ping-pong smxb-features-1.0-SNAPSHOT
[uninstalled] [1.0] ] smxb-ping smxb-features-1.0-SNAPSHOT
[uninstalled] [1.0] ] smxb-pong smxb-features-1.0-SNAPSHOT
```

*NP: It's often a good idea to prefix all of your features and bundles with a known string, such as `smxb` in this case, so you can easily find them via the `grep` command in the various listings.*

Install all of the necessary OSGi bundles by installing the `smxb-ping-pong` feature

```
JBossFuse:karaf@root> features:install smxb-ping-pong
JBossFuse:karaf@root> list | grep smxb
[ 236] [Active] ] [Created] [ ] [ 60] smxb-pinger (1.0.0.SNAPSHOT)
[ 237] [Active] ] [Created] [ ] [ 60] smxb-ponger (1.0.0.SNAPSHOT)
[ 238] [Active] ] [Created] [ ] [ 60] smxb-ponger-service (1.0.0.SNAPSHOT)
```

Every 15 seconds, the 'smxb-pinger' bundle will send a message. The log should now contain output from your bundles

```
JBossFuse:karaf@root> log:display -n 5
2013-03-21 14:24:43,139 | INFO | sConsumer[pings] | pong | ?
2013-03-21 14:24:43,140 | INFO | lyManager[pings] | toActiveMQ | ?
2013-03-21 14:24:58,136 | INFO | #0 - timer://foo | pingFromTimer | ?
2013-03-21 14:24:58,139 | INFO | sConsumer[pings] | pong | ?
2013-03-21 14:24:58,169 | INFO | lyManager[pings] | toActiveMQ | ?
```

Check that the web service is running by hitting the following from your web browser:

```
http://localhost:9090/ping?msg=Ping
```

Congratulations. You have just deployed and run a bootstrapped integration project!

# Next steps

## Working with bundles

Change the Camel route defined in `smxb-pinger/target/classes/OSGI-INF/blueprint/blueprint.xml`. Run

```
smx-bootstraps> mvn clean install
```

In ServiceMix, update the `smxb-pinger` bundle

```
JBossFuse:karaf@root> list | grep smxb-pinger
[ 236] [Active   ] [Created   ] [      ] [ 60] smxb-pinger (1.0.0.SNAPSHOT)
JBossFuse:karaf@root> update 236
```

Your changes should now be visible.

## Configuration

The `smxb-pinger` and `smxb-ponger` projects make use of OSGi Blueprints config services. From the same `blueprints.xml`:

```
<cm:property-placeholder persistent-id="com.fusesource.examples.pinger"
  update-strategy="reload">
  <cm:default-properties>
    <cm:property name="broker.url" value="failover:(tcp://127.0.0.1:61616)" />
  </cm:default-properties>
</cm:property-placeholder>
...
<bean id="activemq" class="org.apache.activemq.camel.component.ActiveMQComponent">
  <property name="brokerURL" value="${broker.url}" />
  <property name="userName" value="admin"/>
  <property name="password" value="admin"/>
</bean>
```

You can dynamically change the default configuration by creating a properties file in the `$(JBOSS_FUSE_HOME/etc)` directory called `{persistent-id}.cfg`, so in this case `com.fusesource.examples.pinger.cfg`. This is a standard properties file, within which you can override any of the properties used in your `blueprints.xml`.

Adding the following line to the file and saving it will stop the bundles that listen on that `persistent-id` and start them again.

```
broker.url=failover:(tcp://127.0.0.1:61616,tcp://127.0.0.1:61617)
```

This assumes that the `cm:property-placheholders` are configured with `update-strategy="reload"`, if not the bundles need to be stopped and started manually.

The log should show the affected Camel routes stopping and starting:

```
14:31:58,764 | INFO | Thread-55 | BlueprintCamelContext | 130 - org.apache.camel.camel-core - 2.10.0.r
14:31:58,765 | INFO | Thread-55 | DefaultShutdownStrategy | 130 - org.apache.camel.camel-core - 2.10.0.r
14:31:58,767 | INFO | 6 - ShutdownTask | DefaultShutdownStrategy | 130 - org.apache.camel.camel-core - 2.10.0.r
14:31:58,767 | INFO | 6 - ShutdownTask | DefaultShutdownStrategy | 130 - org.apache.camel.camel-core - 2.10.0.r
14:31:58,845 | INFO | 6 - ShutdownTask | DefaultShutdownStrategy | 130 - org.apache.camel.camel-core - 2.10.0.r
14:31:58,845 | INFO | 6 - ShutdownTask | DefaultShutdownStrategy | 130 - org.apache.camel.camel-core - 2.10.0.r
14:31:58,845 | INFO | Thread-55 | DefaultShutdownStrategy | 130 - org.apache.camel.camel-core - 2.10.0.r
14:31:59,181 | INFO | Thread-55 | DefaultTypeConverter | 130 - org.apache.camel.camel-core - 2.10.0.r
14:31:59,183 | INFO | Thread-55 | BlueprintCamelContext | 130 - org.apache.camel.camel-core - 2.10.0.r
14:31:59,205 | INFO | NAPSHOT-thread-2 | ManagementStrategyFactory | 130 - org.apache.camel.camel-core - 2.10.0.r
14:31:59,212 | INFO | NAPSHOT-thread-2 | BlueprintCamelContext | 130 - org.apache.camel.camel-core - 2.10.0.r
```

Using this mechanism you can externalise any environment-specific config. If you like, define some properties to echo a different pong message and have a go at modifying them externally.

