JBossESBHelloworld

JBossESB - Helloworld Quickstart

-

Quickstart Description

The purpose of the helloworld quickstart sample is to prove that the ESB is properly configured and happy. As well as to demonstrate the needed minimal files to make a basic ESB component execute.

Implementation

An ant target called "runtest" sends a message to the ESB via JMS. A simple JMS listener on the ESB will invoke the MyJMSListenerAction and display the message to the console. You can modify build.xml to change the phrase "Hello World" to something else and re-run "ant runtest".

Configuration

Specify a JMS gateway. The JMS client will send messages to a JMS queue named *queue/quickstart_helloworld_Request_gw*. Every gateway listener has a corresponding internal ESB listener. In this example, we have a JMS queue called *queue/*

quickstart_helloworld_Request_esb that the ESB will use internally when making action calls.

```xml
<providers>
    <jms-provider name="JBossMQ" connection-factory="ConnectionFactory">
      <jms-bus busid="quickstartGwChannel">
        <jms-message-filter
            dest-type="QUEUE"
            dest-name="queue/quickstart_helloworld_Request_gw"
         ></jms-message-filter>
      </jms-bus>
      <jms-bus busid="quickstartEsbChannel">
        <jms-message-filter
            dest-type="QUEUE"
            dest-name="queue/quickstart_helloworld_Request_esb"
         ></jms-message-filter>
      </jms-bus>
    </jms-provider>
 </providers>
```

As you can see, the service defined uses both the "gateway listener" and its corresponding "esb listener".

```xml
<services>
  <service
      category="FirstServiceESB"
      name="SimpleListener"
      description="Hello World">
    <listeners>
      <jms-listener name="JMS-Gateway"
              busidref="quickstartGwChannel"
              maxThreads="1"
              is-gateway="true"
      ></jms-listener>
      <jms-listener name="helloWorld"
              busidref="quickstartEsbChannel"
              maxThreads="1"
      ></jms-listener>
    </listeners>
  ...
```

Each service will have a list of actions. In this example, we only have one action. And this action will just output the "helloworld" message.

```
<actions>
    <action name="action1"
        class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
        process="displayMessage"
    ></action>
</actions>
```

Attribute	Required	Description
name	yes	action name
class	yes	Java class that contains methods that have a signature like: *public Message doSomething(Message message)*
process	no	Action method you would like to execute. In the helloworld example, the method name is *displayMessage*. If not specified the *process* method will be called.

Last, but not least, here's our Action code. Notice the protected *_config* variable. The constructor of your action classes must accept a *org.jboss.soa.esb.helpers.ConfigTree* and set it to *_config*. Don't worry about this variable

```
public class MyJMSListenerAction extends AbstractActionLifecycle
{
  protected ConfigTree _config;

  public MyJMSListenerAction(ConfigTree config) { _config = config; }

  public Message displayMessage(Message message) throws Exception {

  System.out.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&");
```

```
    System.out.println("Body: " + new String(message.getBody().getByteArray()));
    System.out.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&");

    return message;
    }
 }
```

Quickstart Code Link