

Lab I

hello_world

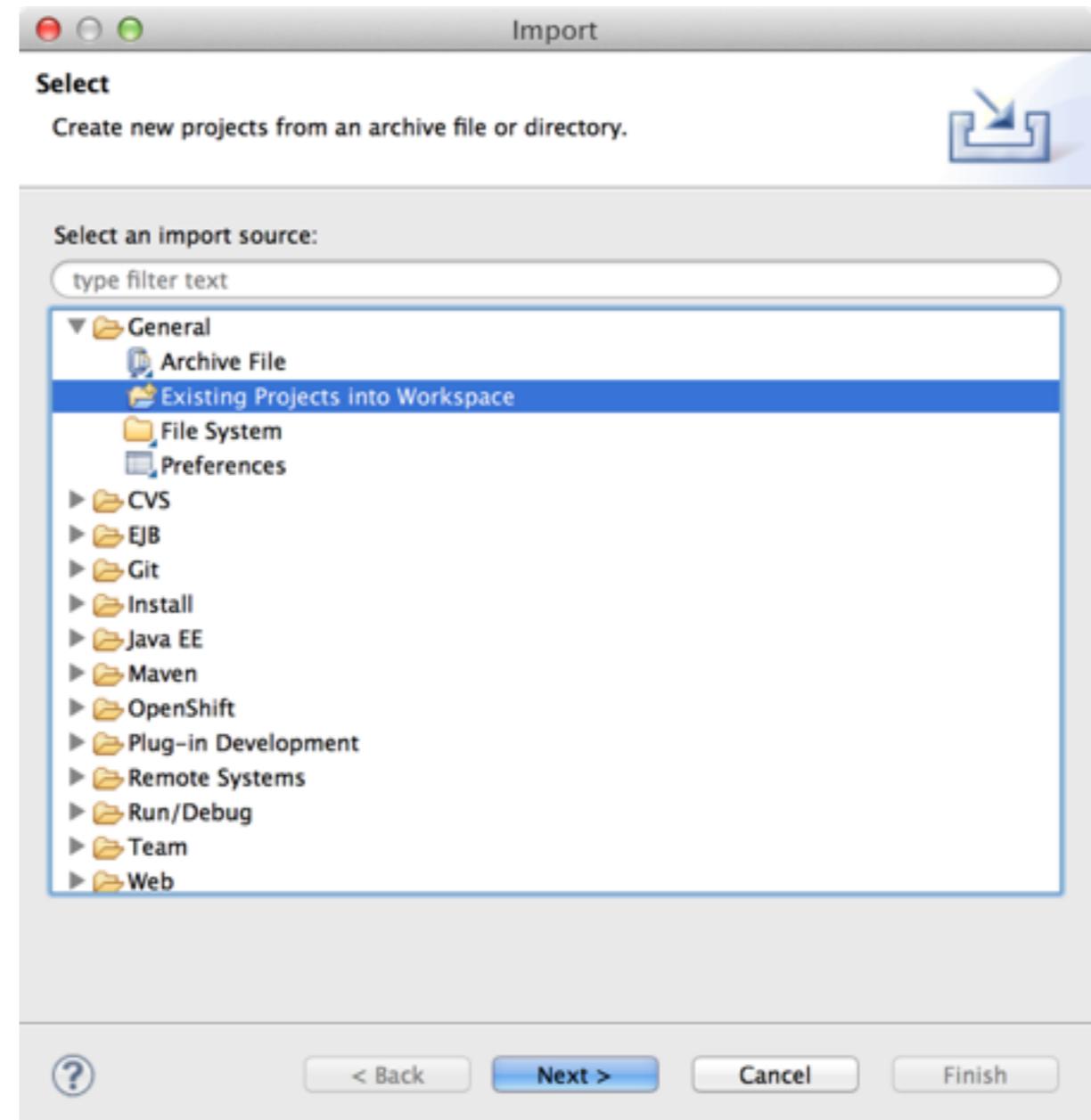
Lab Goals

- Introduction to hello_world in SOA 5
- Introduction to hello_world in SOA 6
- Step-by-step migration using Windup rules
- Deploy and test application in SOA 6

Importing SOA 5 Hello World

TODO

1. File -> Import ... from the JBDS menu.
2. Select General -> Existing Projects into Workspace
3. Click Next



Importing SOA 5 Hello World

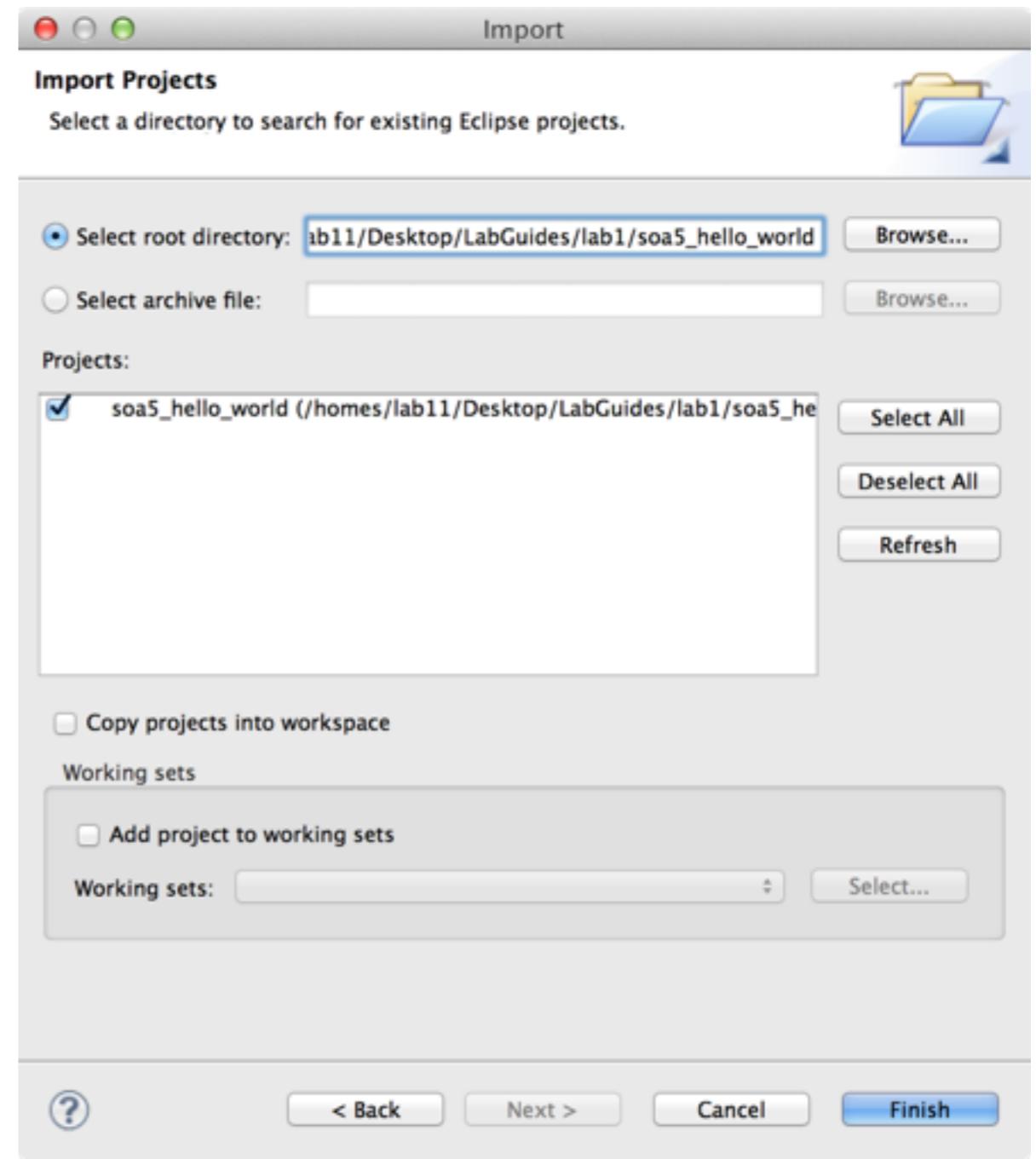
TODO

1. Click Browse ... and navigate to:

/home/lab11/Desktop/LabGuides/lab1/soa5_hello_world

2. Make sure the soa5_hello_world project is checked

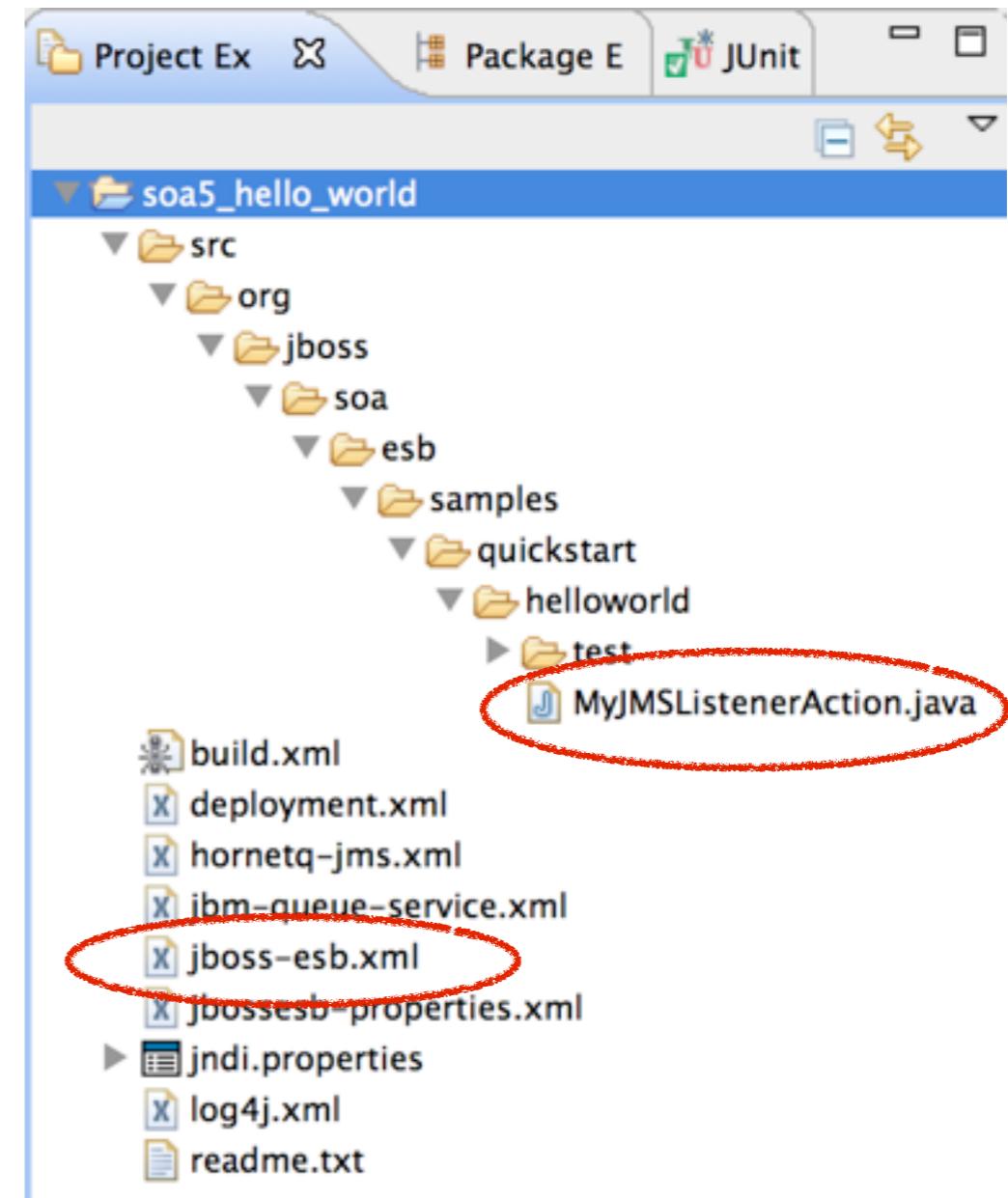
3. Click Finish



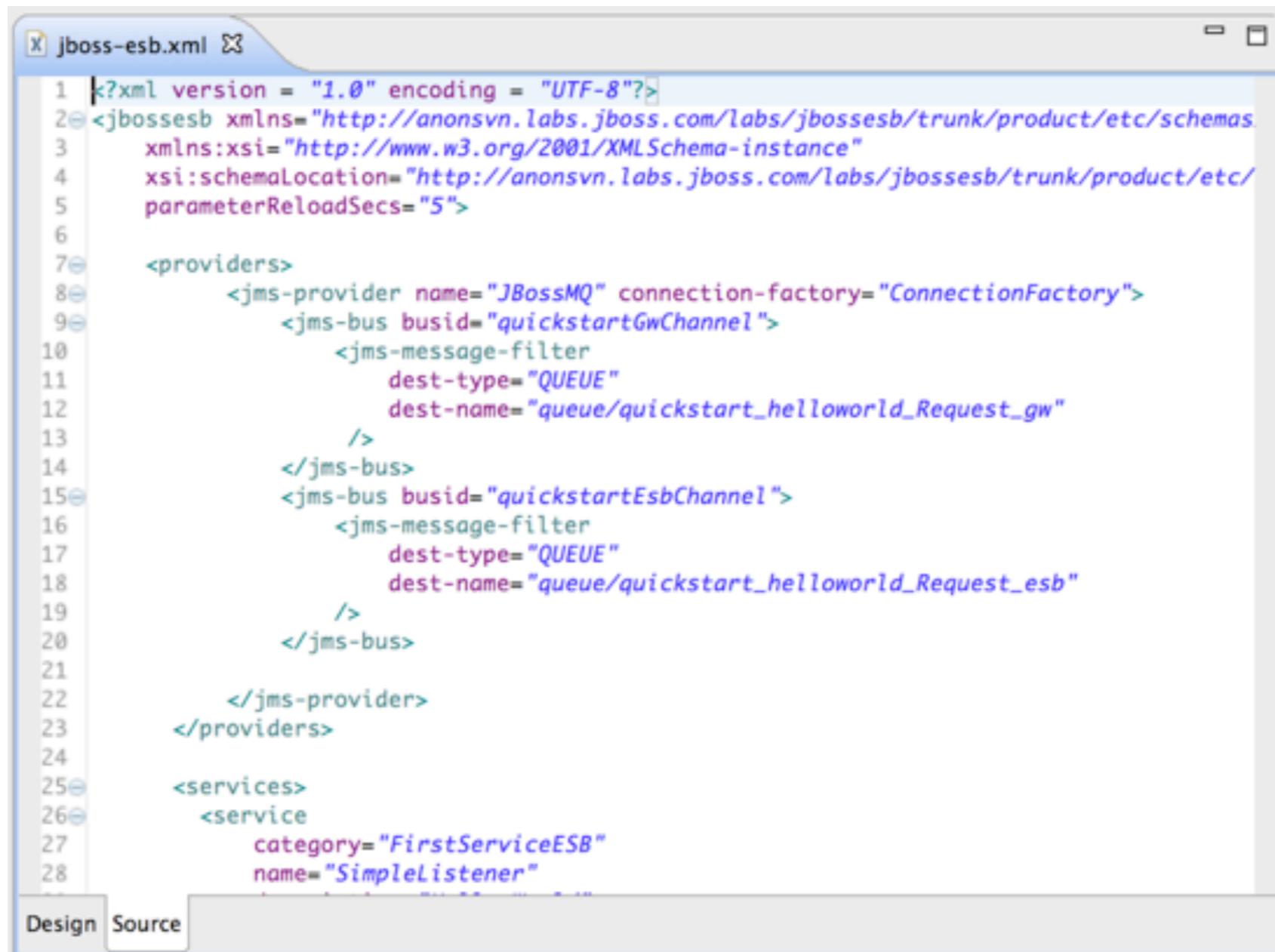
Files To Note

TODO

1. `jboss-esb.xml` contains the project's service definitions and configuration. Open the file by double-clicking on it in the Project Explorer.
2. `MyJMSListenerAction.java` contains the service logic for the application. Open the file by double-clicking on it in the Project Explorer.



jboss-esb.xml



The screenshot shows a code editor window titled "jboss-esb.xml" containing XML configuration for JBoss ESB. The XML defines providers and services, specifically setting up JBossMQ connection factories and JMS buses for quickstart channels.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<jbossesb xmlns="http://anonsvn.labs.jboss.com/labs/jbossejb/trunk/product/etc/schemas.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://anonsvn.labs.jboss.com/labs/jbossejb/trunk/product/etc/parameterReloadSecs=5">

<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-bus>
        <jms-bus busid="quickstartEsbChannel">
            <jms-message-filter dest-type="QUEUE" dest-name="queue/quickstart_helloworld_Request_esb" />
        </jms-bus>
    </jms-provider>
</providers>

<services>
    <service category="FirstServiceESB" name="SimpleListener" />
</services>
```

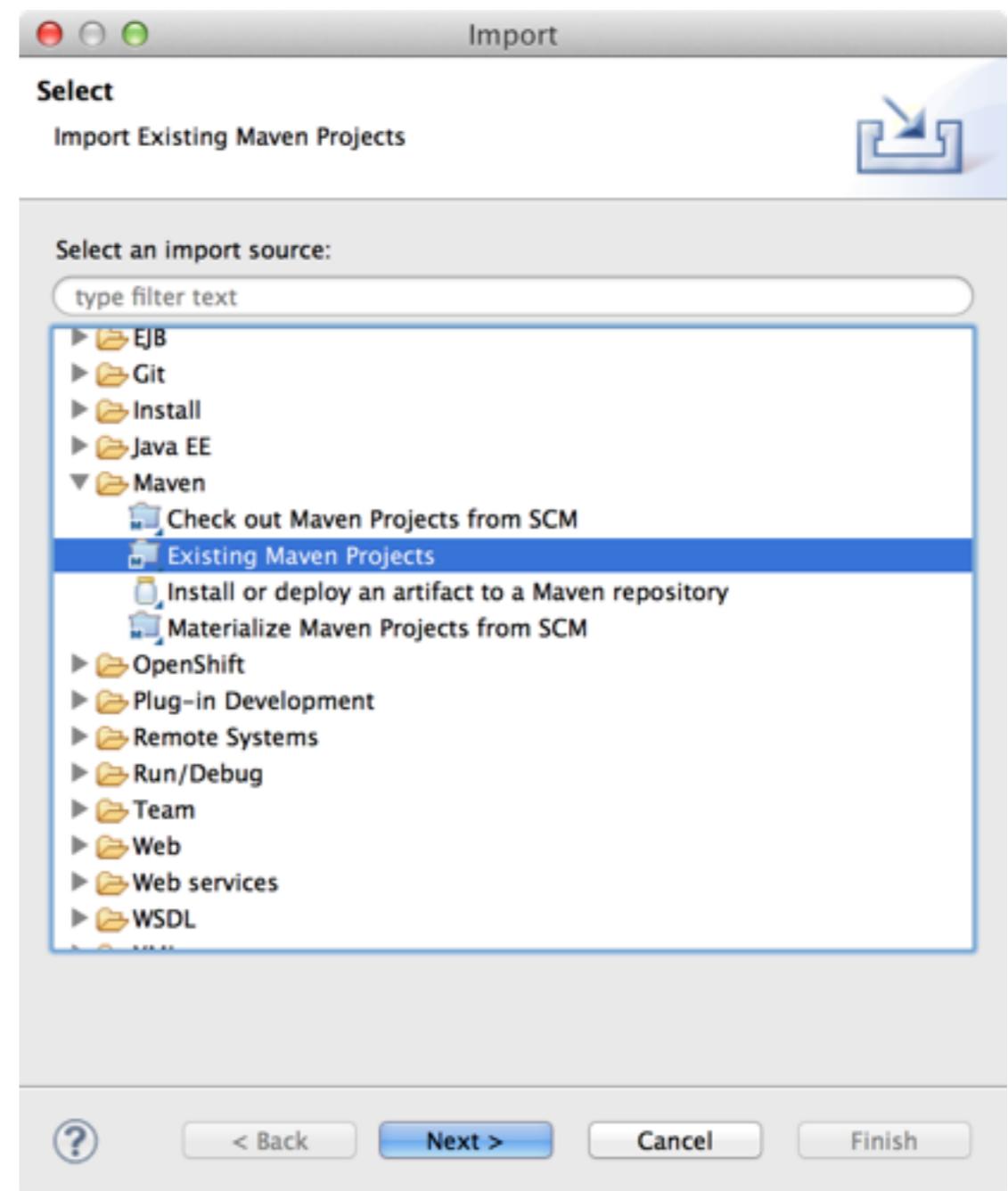
The code editor has tabs for "Design" and "Source", with "Source" currently selected.

MyJMSListenerAction.java

Importing SOA 6 Hello World

TODO

1. File -> Import ... from the JBDS menu.
2. Select Maven -> Existing Maven Projects
3. Click Next



Importing SOA 6 Hello World

TODO

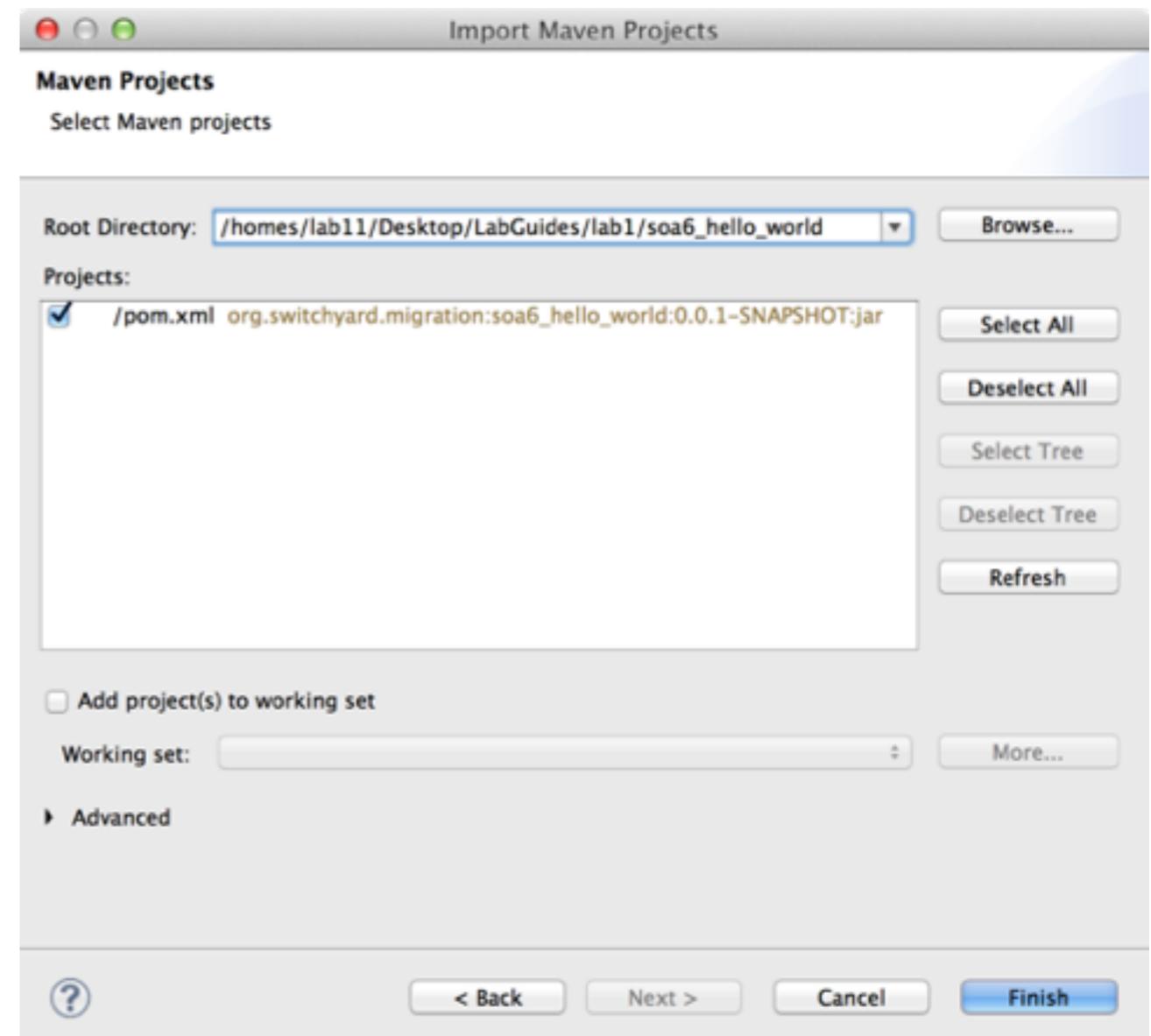
1. Click Browse ... and navigate to:

/home/lab11/Desktop/LabGuides/lab1/soa6_hello_world

2. Make sure the pom.xml is checked for:

org.switchyard.migration:soa6_hello_world

3. Click Finish



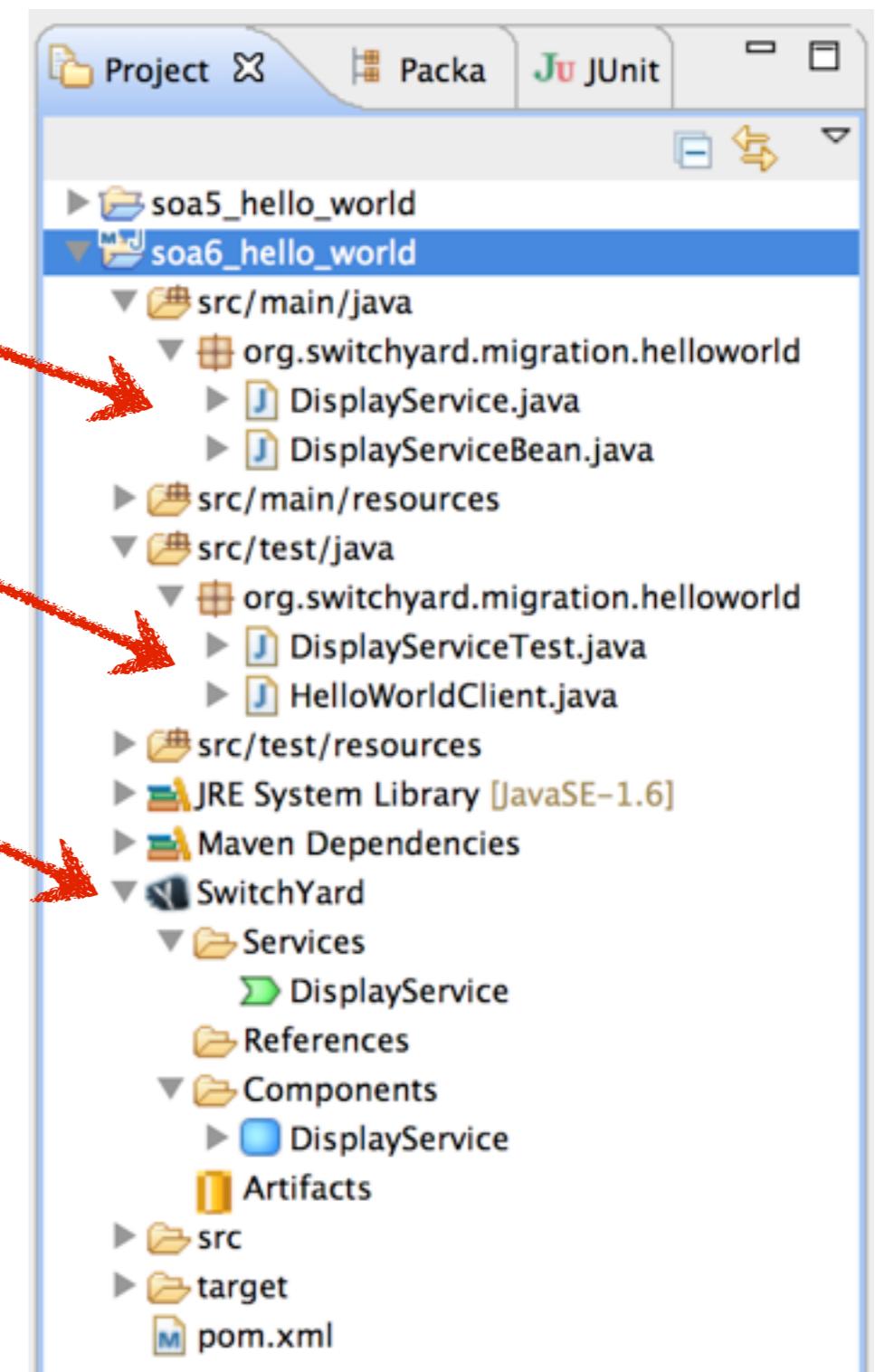
Files To Note

1. Java implementation classes for your application are stored in src/main/java

2. Test classes are stored in src/test/java

3. The  SwitchYard node in the Project Explorer can be used to open the visual editor for the SwitchYard Project. Try double-clicking the SwitchYard node to open the editor.

Open these files and familiarize yourself with the content



Migrating hello_world

- A step-by-step migration walkthrough follows
- Each step is driven by a notification in the windup report
- Read the notification and microsite details for each step
- Study the before and after for each step using the applications you've imported into JBDS

Step I

Converting Action Class to Bean Service

Notification

- ❶ [Action : service binding configuration in jms-bus: busid="quickstartGwChannel"](#)
- ❶ [Action : service binding configuration in jms-bus: busid="quickstartEsbChannel"](#)
- ❶ [Action : composite service required for service: name="SimpleListener"](#)
- ❶ [Action : composite service binding required for listener: name="JMS-Gateway"](#)
- ❶ [Action : create component service for action processing pipeline](#)
- ❶ [Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"](#)



Notification

jboss-esb.xml

FYI

This is the notification from jboss-esb.xml, noting that a custom action class is referenced in the action processing pipeline that needs to be migrated.

```
40. <action name="action1"  
41.   class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"  
42.   process="displayMessage"  
43. />
```

! Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"

Custom action classes should be migrated to CDI Beans in SOA 6. These beans can be defined as services or called directly from a Camel route.

For additional information and tips, see the [action class microsite](#).

Notification

MyJMSListenerAction.java

FYI

This is the notification from MyJMSListenerAction.java providing API level details on what needs to change in the action class implementation.

27. **public class** MyJMSListenerAction **extends** AbstractActionLifecycle

! Extending class 'org.jboss.soa.esb.actions.AbstractActionLifecycle' at line 27

Action classes in SOA 5 are simply CDI Beans in SOA 6. The extension of AbstractActionLifecycle is no longer necessary. An action class can become a standalone service in SOA 6 or it can be invoked as a bean from a Camel route.

For additional information and tips, see the [action class microsite](#).

Microsite

action class

TODO

Visit the microsite URL to learn about action class migration in detail.

The screenshot shows a web page with a header bar indicating the file is 64 lines (51 sloc) and 3.259 kb. The main content starts with an 'Overview' section. It discusses the migration of Action classes from SOA 5 to CDI Beans in SOA 6, mentioning two methods: converting the action class to a CDI Bean Service or converting the class to a CDI Bean and invoking it directly from a Camel route. It notes that generally, a CDI Bean Service is preferred if logic will be exposed directly to service consumers. Below this, there's a section titled 'SOA 5 Action Class' with sample code:

```
public class MyAction extends AbstractActionLifecycle
{
```

<https://github.com/windup/soa-migration/blob/master/advice/action-class-migration.md>

Service Contract

FYI

This is the service contract for DisplayService in SOA 6. All services in SOA 6 have a contract.

```
package org.switchyard.migration.helloworld;

public interface DisplayService {

    void display(String message);

}
```

Service Implementation

FYI

This is the implementation of DisplayService in SOA 6 using a CDI Bean Service.

Step 2

Converting the Action Processing Pipeline

Notification

- ! [Action : service binding configuration in jms-bus: busid="quickstartGwChannel"](#)
- ! [Action : service binding configuration in jms-bus: busid="quickstartEsbChannel"](#)
- ! [Action : composite service required for service: name="SimpleListener"](#)
- ! [Action : composite service binding required for listener: name="JMS-Gateway"](#)
- ! [Action : create component service for action processing pipeline](#)
- ! [Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"](#)



Action Processing Pipeline

```
<actions mep="OneWay">

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

    <action name="action2"
        class="org.jboss.soa.esb.actions.SystemPrintln">
        <property name="printfull" value="false"/>
    </action>

    <!-- The next action is for Continuous Integration testing -->
    <action name="testStore" class="org.jboss.soa.esb.actions.TestMessageStore"/>

</actions>
```

Notification

jboss-esb.xml

39.

<actions mep="OneWay">

! Action : create component service for action processing pipeline

The logic and execution flow of a service in SOA 5 is defined in an action processing pipeline. In SOA 6, this logic is contained within a service component definition and expressed using any of the available implementation types in SwitchYard.

For additional information and tips, see the [action pipeline microsite](#).

Microsite

action pipeline

TODO

Visit the microsite URL to learn about action pipeline migration in detail.

The screenshot shows a file editor window with the following details:

- File type: XML
- Size: 42 lines (33 sloc) | 2.395 kb

Overview

The action processing pipeline was the only container available for defining composition logic for ESB services in SOA 5. In SOA 6, any implementation type (CDI Bean, BPEL, Camel Route, BPM, Rules, etc.) can be used to define a service. The choice of which implementation type to use in SOA 6 boils down to the following:

- If your action processing pipeline contained procedural and/or routing logic for composition of services, then a Camel route will provide the most direct equivalent in SOA 6.
- If your action processing pipeline simply "handed off" the processing logic to a specific implementation type, then consider exposing that implementation directly as a service in SOA 6.

For example, if you have a SOA 5 action processing pipeline has a single action class which contains all of the service logic, then that makes a great candidate for migrating the action class to a CDI Bean Service. On the other hand, if you have a pipeline with complex routing and multiple action classes, then a Camel route will be a better fit.

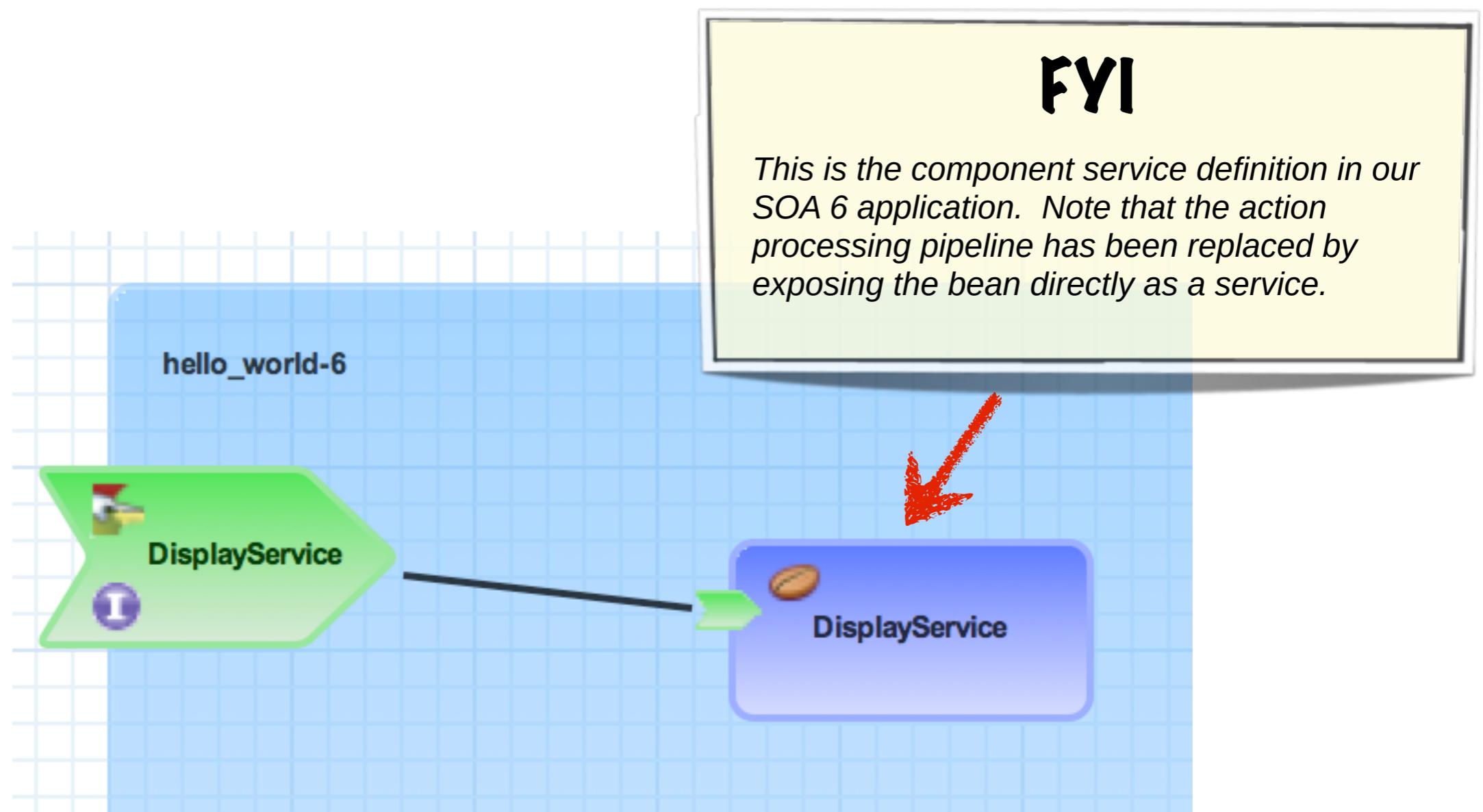
SOA 5 Action Processing Pipeline

An example of a very basic action processing pipeline:

```
<actions mep="OneWay">
  <action name="action1" class="org.example.MyAction" process="doSomething"/>
```

<https://github.com/windup/soa-migration/blob/master/advice/action-pipeline-migration.md>

Component Service



Step 3

Creating Composite Service For ESB Services

Notification

- 
- ! [Action : service binding configuration in jms-bus: busid="quickstartGwChannel"](#)
 - ! [Action : service binding configuration in jms-bus: busid="quickstartEsbChannel"](#)
 - ! [Action : composite service required for service: name="SimpleListener"](#)
 - ! [Action : composite service binding required for listener: name="JMS-Gateway"](#)
 - ! [Action : create component service for action processing pipeline](#)
 - ! [Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"](#)

Service Definition

```
25.    <services>
26.      <service
27.        category="FirstServiceESB"
28.        name="SimpleListener"
29.          description="Hello World">
```

! Action : composite service required for service: name="SimpleListener"

Each definition in SOA 5 represents a service which can be called from outside the application through an ESB listner. The equivalent definition in SOA 6 is a composite service.

For additional information and tips, see the [service migration microsite](#).

Microsite

service migration

TODO

Visit the microsite URL to learn about action pipeline migration in detail.

The screenshot shows a web-based interface for managing SOA configurations. At the top, there's a toolbar with a file icon, '34 lines (26 sloc)', and '1.309 kb'. Below the toolbar, the word 'Edit' is visible. The main content area is divided into sections:

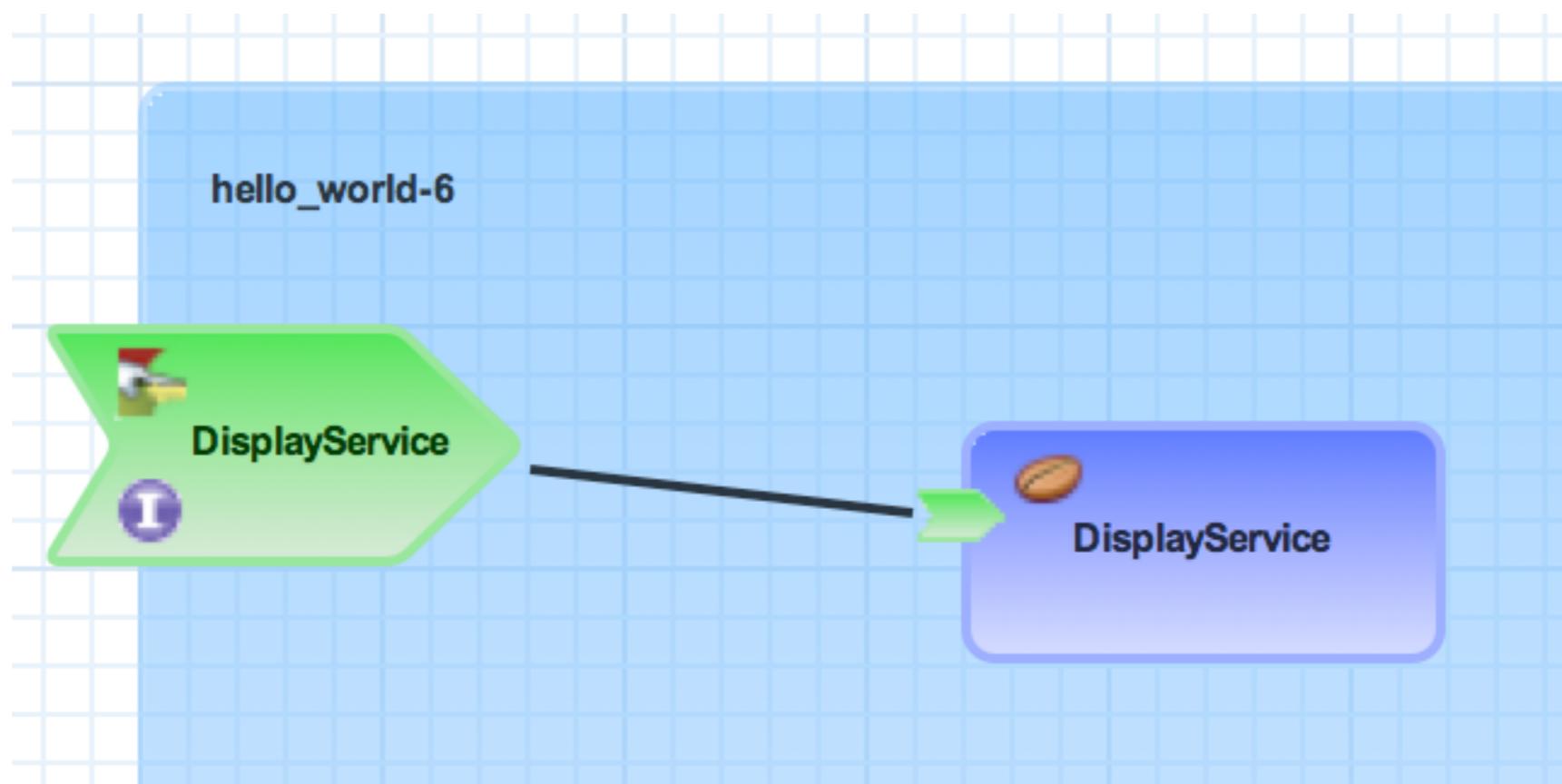
- Overview**: A brief explanation of composite services: "The equivalent of a SOA 5 service definition in SOA 6 is called a composite service. Each composite service definition in SOA 6 requires a service interface which defines the contract service consumers will use to invoke the service."
- SOA 5 Configuration**: Displays an XML snippet for defining a service:

```
<services>
  <service
    category="Purchasing"
    name="OrderService"
    description="An Order Service">
  </service>
</services>
```
- SOA 6 Configuration**: An explanatory text block stating: "A service definition in SOA 5 is comprised of a component service and a composite service in SOA 6. The component service contains the implementation of the service and the composite service indicates that the service is accessible outside the application. This relationship is depicted in the visual editor like this:" followed by a diagram.

The diagram illustrates the relationship between an application and a service. It features a large blue rectangular box labeled 'application' at the top. Inside the application box, there is a green arrow pointing to the right, labeled 'Service'. A black arrow points from the 'Service' label down towards the bottom of the application box, indicating the flow or dependency.

<https://github.com/windup/soa-migration/blob/master/advice/service-migration.md>

Composite Service



Step 4

Migrating JMS Gateway Listener

Notification

- 
- ! [Action : service binding configuration in jms-bus: busid="quickstartGwChannel"](#)
 - ! [Action : service binding configuration in jms-bus: busid="quickstartEsbChannel"](#)
 - ! [Action : composite service required for service: name="SimpleListener"](#)
 - ! [Action : composite service binding required for listener: name="JMS-Gateway"](#)
 - ! [Action : create component service for action processing pipeline](#)
 - ! [Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"](#)

Gateway Listener

```
30.      <listeners>
31.          <jms-listener name="JMS-Gateway"
32.              busidref="quickstartGwChannel"
33.              is-gateway="true"
34.      />
```

! Action : composite service binding required for listener: name="JMS-Gateway"

This listener requires a composite service binding in SwitchYard. The configuration for a JCA or JMS binding can be found in the jms-bus definition associated with this listener.

For additional information and tips, see the [gateway listener microsite](#).

Microsite

listener migration

TODO

Visit the microsite URL to learn about listener migration in detail.

The screenshot shows a web page with the following content:

Overview

There are two types of listeners in SOA 5:

- ESB-aware Listeners are used for internal message dispatch and drive an action processing pipeline.
- Gateway Listeners are used to expose services to external consumers over a protocol binding.

When migrating to SOA 5, ESB-aware listeners are no longer required. Gateway listeners become Service Bindings in SOA 6. A gateway listeners have a value of true for the 'is-gateway' attribute on a listener definition in jboss-esb.xml.

Listener to Service Binding Conversion

The following table provides a mapping of SOA 5 gateway listeners to SOA 6 service bindings:

SOA 5 Listener	SOA 6 Service Binding
fs-listener	binding.file
ftp-listener	binding.ftp
sql-listener	binding.sql
jbr-listener	binding.tcp
jms-listener	binding.jca or binding.jms

<https://github.com/windup/soa-migration/blob/master/advice/gateway-listener-migration.md>

Gateway Listener Config

09. `<jms-bus busid="quickstartGwChannel">`

! Action : service binding configuration in jms-bus: busid="quickstartGwChannel"

A jms-bus definition can be converted to a JMS or JCA gateway binding on a composite service in SwitchYard. If the jms-bus configuration is used for a non-gateway listener, it does not need to be migrated to SOA 6.

For additional information and tips, see the [jms-bus migration microsite](#).

10. `<jms-message-filter`

11. `dest-type="QUEUE"`

12. `dest-name="queue/quickstart_helloworld_Request_gw"`

13. `/>`

14. `</jms-bus>`

Microsite

jms-bus migration

TODO

Visit the microsite URL to learn about jms-bus migration in detail.

The screenshot shows a web page with a header bar indicating 'File | 61 lines (49 sloc) | 2.629 kb'. The main content area has a title 'Overview' and a paragraph explaining that the jms-bus element in jboss-esb.xml provides configuration for JMS listeners. It can be converted into one of two binding types: JCA (binding.jca) or JMS (binding.jms). Below this is a section titled 'SOA 5 Configuration' containing XML code:

```
<jms-bus busid="quickstartGwChannel">
  <jms-message-filter
    dest-type="QUEUE"
    dest-name="queue/quickstart_helloworld_Request_gw"/>
</jms-bus>
```

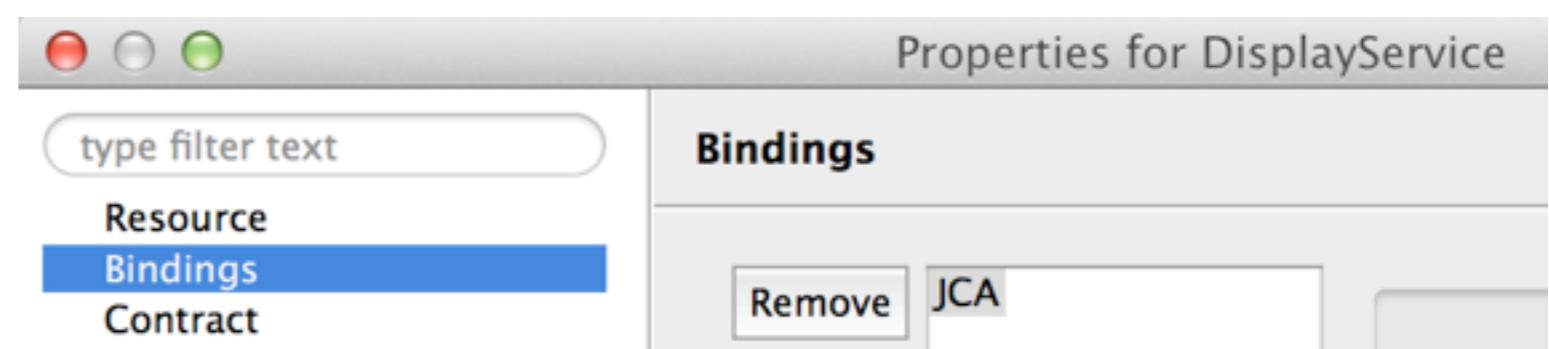
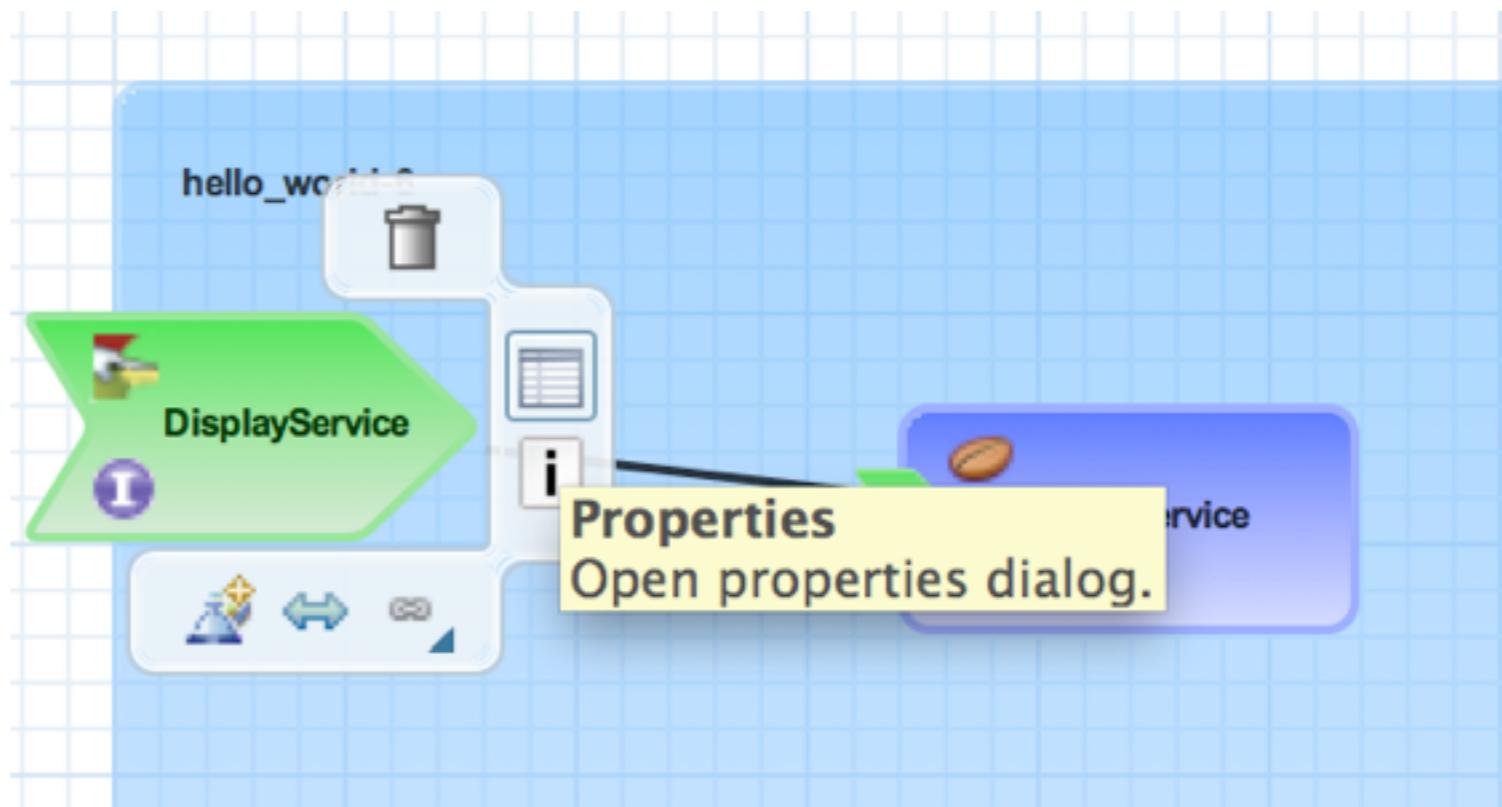
There is also a section titled 'SOA 6 Configuration' with a heading 'Converting to JCA' and a note about destination name requirements.

<https://github.com/windup/soa-migration/blob/master/advice/jms-bus-migration.md>

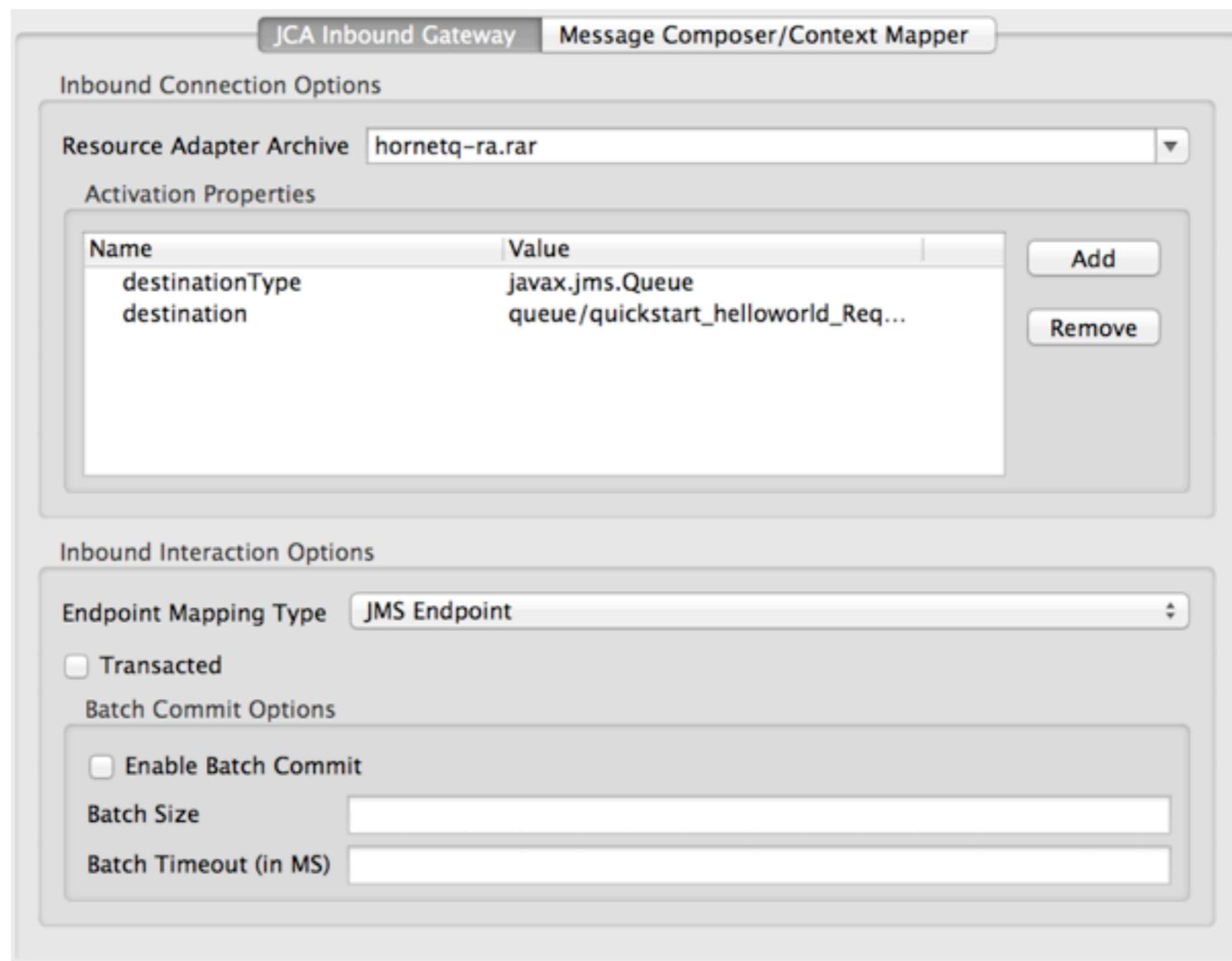
Service Bindings

TODO

1. Bring up the button bar for the composite service 'DisplayService' by hovering over the large green service icon on the left of the composite.
2. When the button bar appears, click on the table icon which brings up the service properties.
3. On the resulting property page, select the Bindings page in the left frame.
4. View the configuration details of the JCA binding definition in SOA 6.



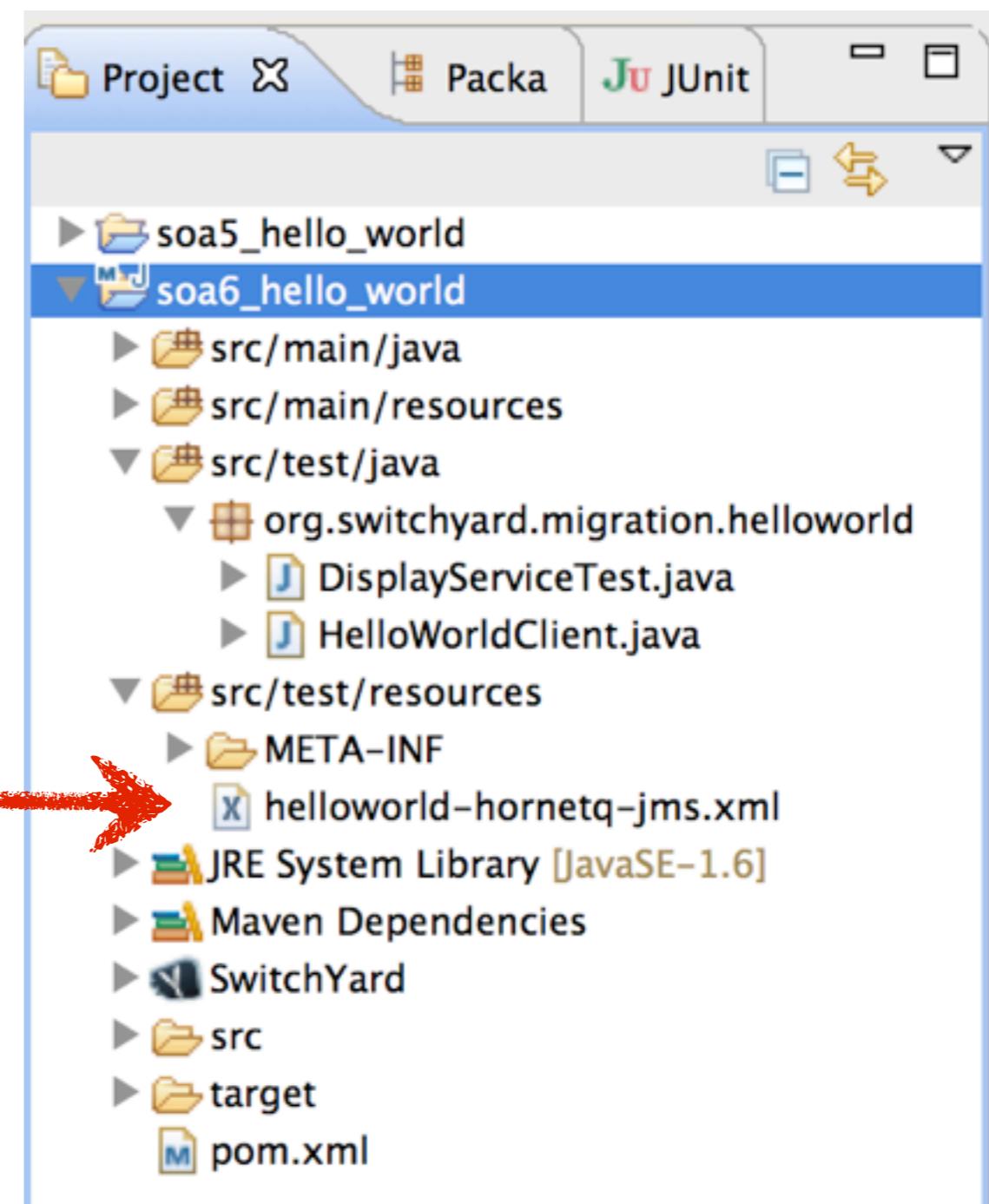
Binding Configuration



Deploy Queue

TODO

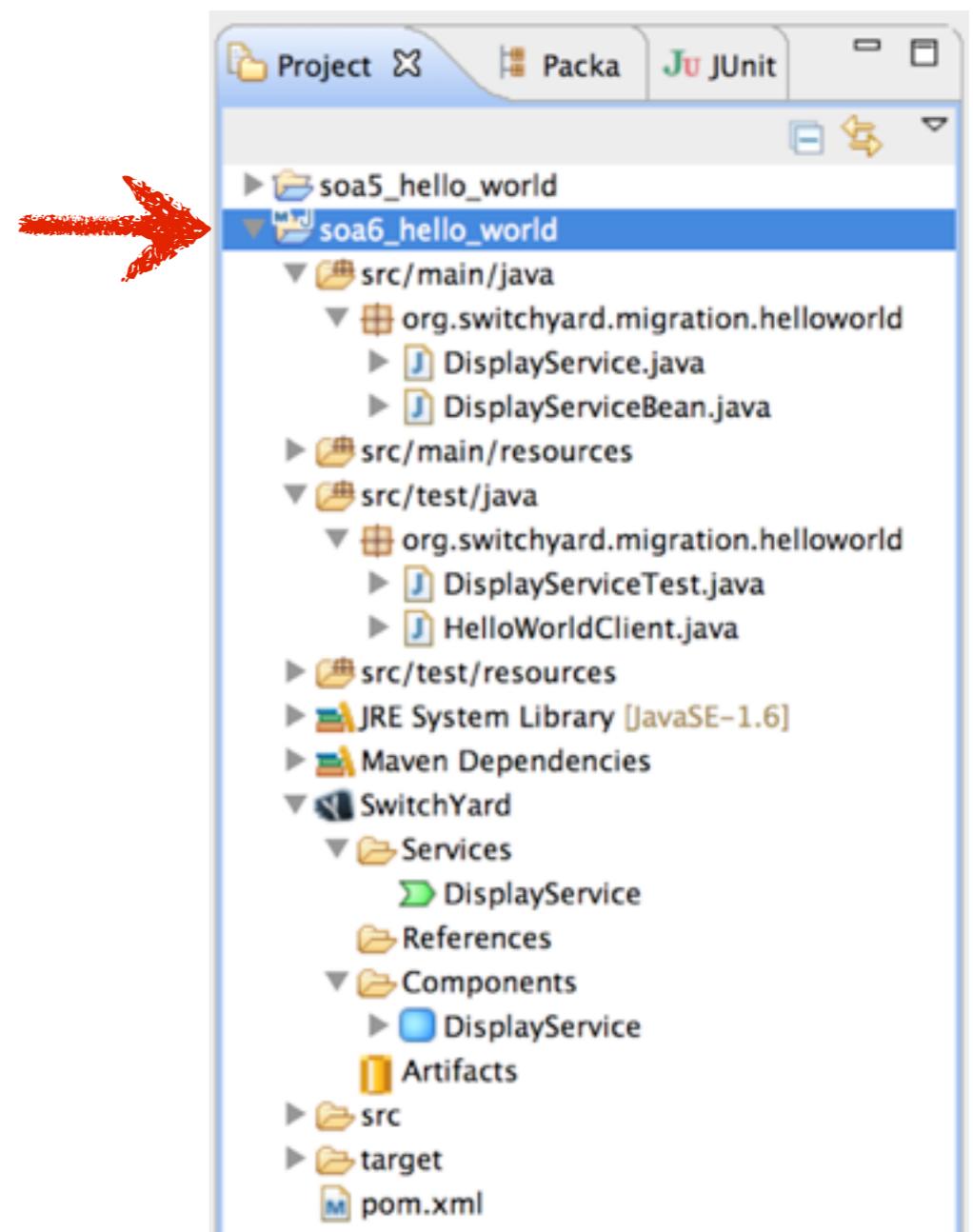
1. Right-click on helloworld-hornetq-jms.xml and select 'Mark as Deployable'.
2. When prompted with "Really mark these as deployable?", click OK.



Deploy Application

TODO

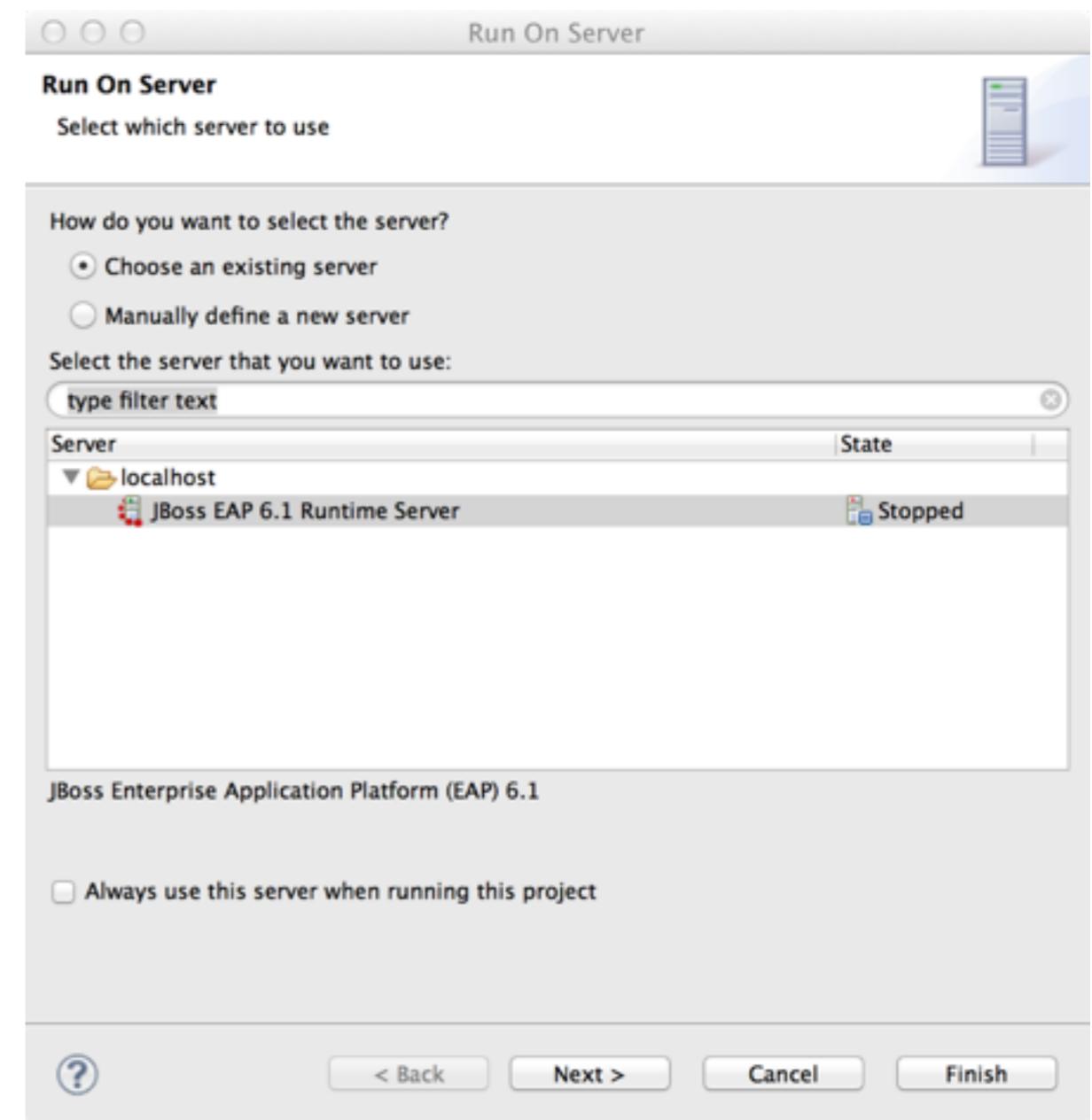
1. Right-click on the soa6_hello_world project and select Run As ... Run on Server



Select Server

TODO

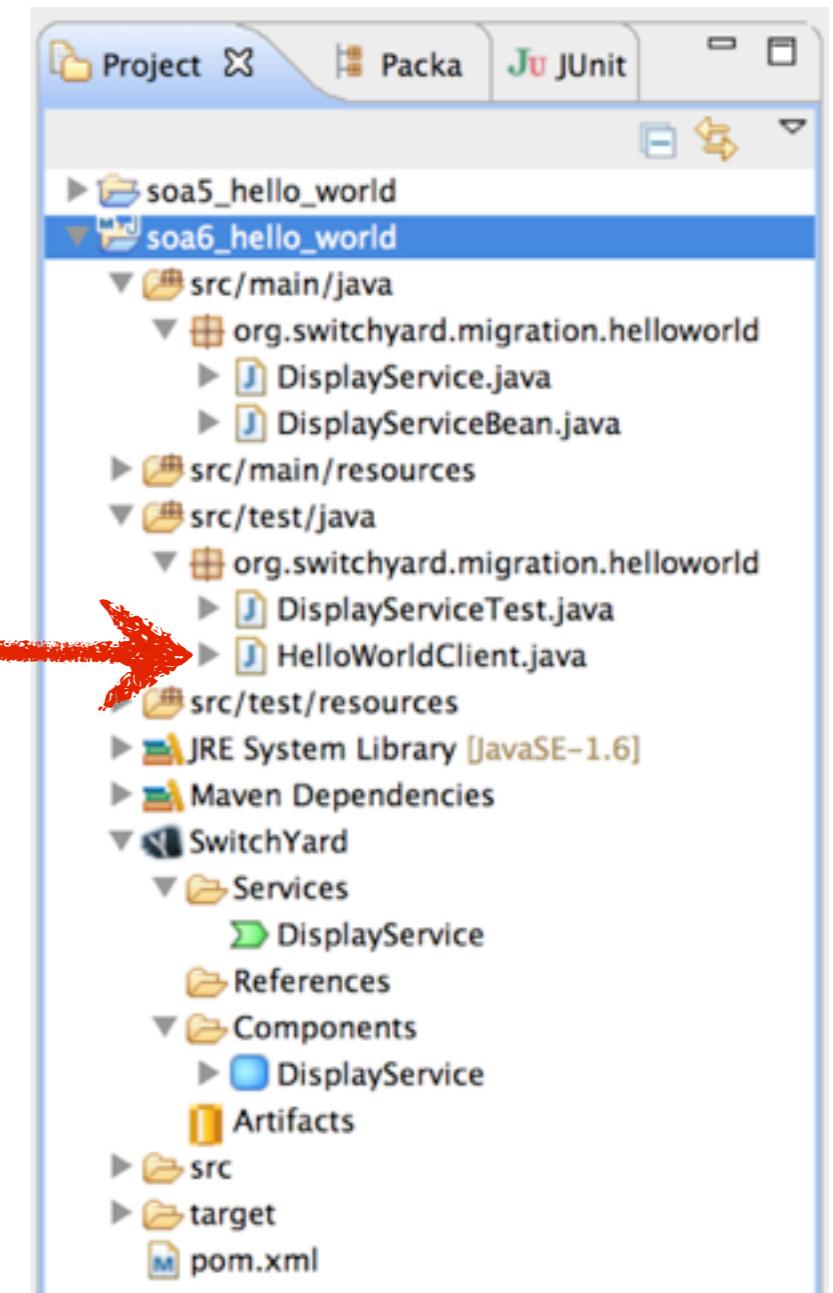
1. Select the JBoss EAP 6.1 Runtime Server
2. Click Finish



Run Test Client

TODO

1. Open HelloWorldClient.java from the Project Explorer view.
2. Go to the Run menu in the main menu bar and select 'Run As -> Java Application'



Verify Output

TODO

1. Click here to swap between application output and server output.



```
JBoss EAP 6.1 Runtime Server [JBoss Application Server Startup Configuration] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (May 28, 2013 5:14:40 PM)
[0m[0m17:14:42,640 INFO  [org.switchyard.common.camel.SwitchYardCamelContext] (MSC service thread 1-1) Apache Camel 2.0.1.Final (CamelContext)
[0m[0m17:14:42,649 INFO  [org.hornetq.ra] (MSC service thread 1-1) HornetQ resource adaptor started
[0m[0m17:14:42,649 INFO  [org.jboss.as.connector.services.resourceadapters.ResourceAdapterActivatorService$ResourceAdapterActivator] (MSC service thread 1-1) JBAS010401: Bound JCA ConnectionFactory [java:/JmsXA]
[0m[0m17:14:42,651 INFO  [org.jboss.as.connector.deployment] (MSC service thread 1-1) JBAS010401: Bound JCA ConnectionFactory [java:/JmsXA]
[0m[0m17:14:42,754 INFO  [org.apache.camel.impl.converter.DefaultTypeConverter] (MSC service thread 1-12) Loaded 179 type converters
[0m[0m17:14:43,462 INFO  [org.switchyard.common.camel.SwitchYardCamelContext] (MSC service thread 1-12) Route: direct:{urn:org.switchyard.mis
[0m[0m17:14:43,519 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 29) JBAS018559: Deployed "soa6_hello_world.jar" (runtime-name :
[0m[0m17:14:43,520 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 29) JBAS018559: Deployed "helloworld-hornetq-jms.xml" (runtime-n
[0m[0m17:14:43,525 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015961: Http management interface listening on http://127.0.0.1:9990/mar
[0m[0m17:14:43,525 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015951: Admin console listening on http://127.0.0.1:9990
[0m[0m17:14:43,525 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015874: JBoss SOA 6.0.0.Alpha-redhat-1 (AS 7.2.0.Final-redhat-4) started
[0m[0m17:15:14,645 INFO  [stdout] (Thread-0 (HornetQ-client-global-threads-1191295788)) &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
[0m[0m17:15:14,645 INFO  [stdout] (Thread-0 (HornetQ-client-global-threads-1191295788)) Body: Hello SwitchYard!
[0m[0m17:15:14,645 INFO  [stdout] (Thread-0 (HornetQ-client-global-threads-1191295788)) &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

Lab I Complete!