# Lab 2

## gateways_routers

# Lab Goals

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

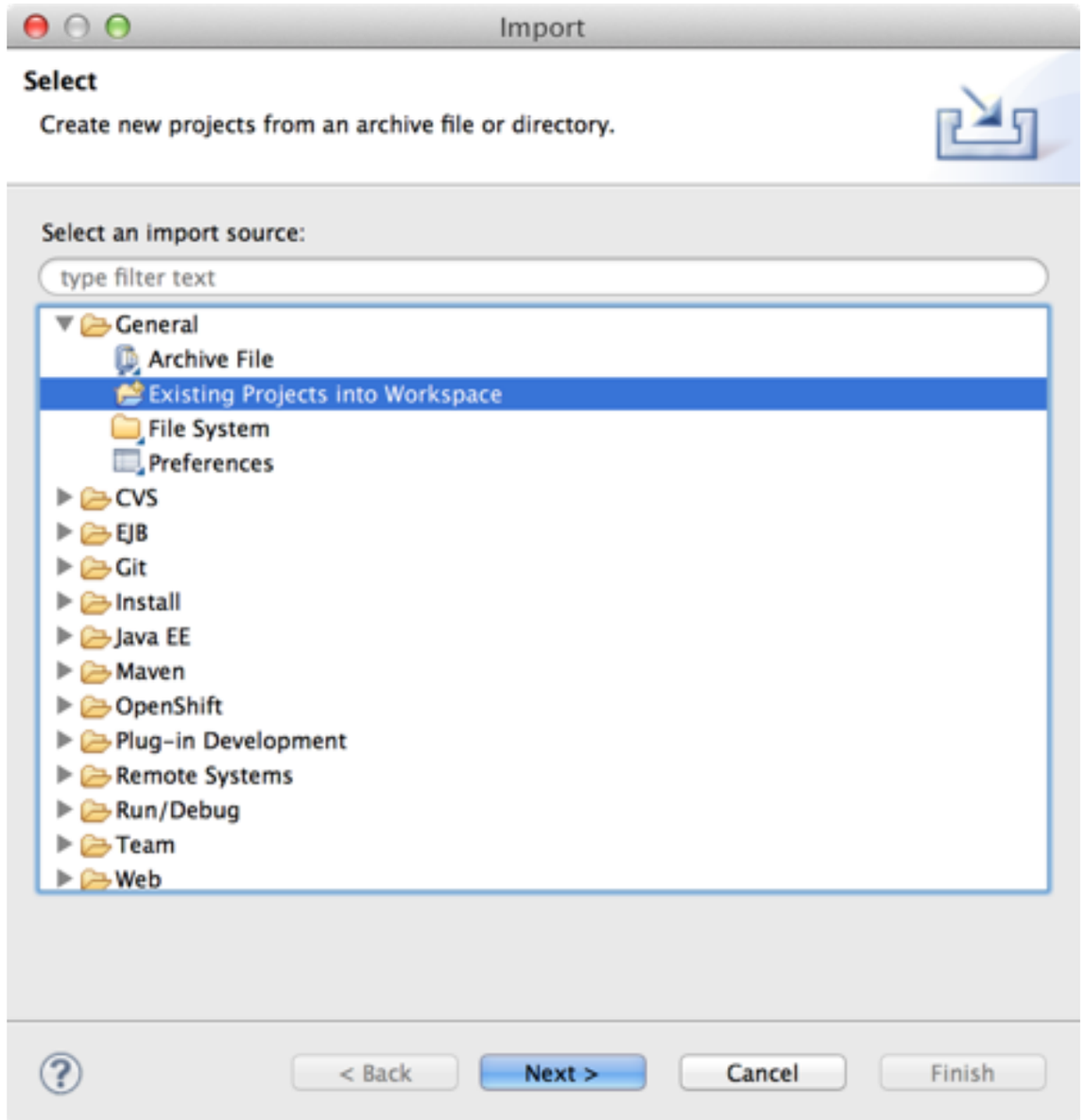
# Importing SOA 5 Gateways/ Routers

## TODO

1. File -> Import ... from the JBDS menu.

2. Select General -> Existing Projects into Workspace

3. Click Next



# Importing SOA 5 Gateways/ Routers

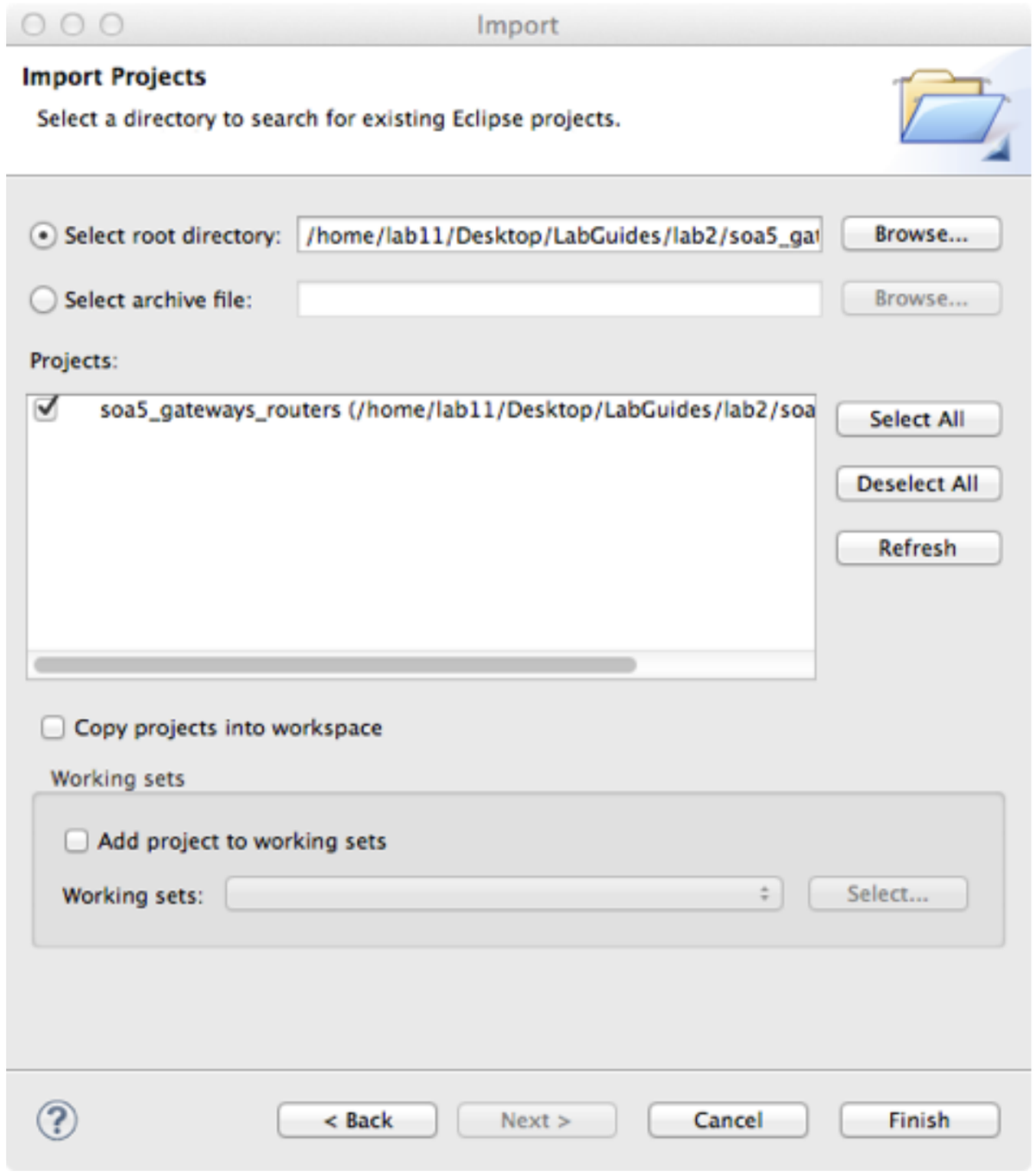
## TODO

1. Click Browse ... and navigate to:

`/home/lab11/Desktop/LabGuides/lab2/soa5_gateways_routers`

2. Make sure the soa5_gateways_routers project is checked

3. Click Finish

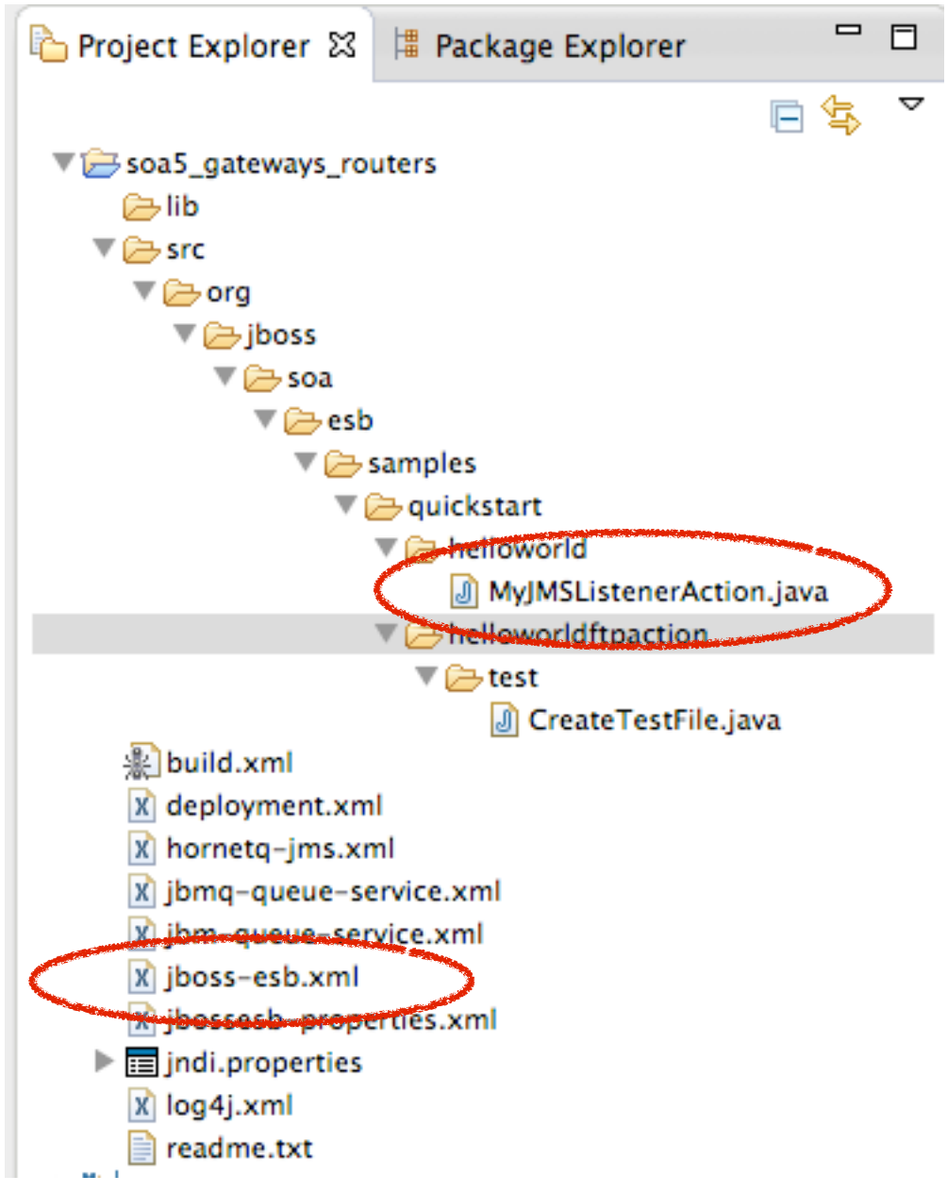


# Files To Note

## 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

jboss-esb.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<jbossesb xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/p
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://anonsvn.labs.jboss.com/labs/jbossesb/
    parameterReloadSecs="5">
    <providers>
          <ftp-provider name="FTPprovider" hostname="localhost">
            <ftp-bus busid="helloFTPChannel" >
                <ftp-message-filter
                    username="ftpboy"
                    password="camel123."
                    passive="false"
                    directory="/tmp/ftpinput"
                    input-suffix="ftpin.dat"
                    work-suffix=".esbWorking"
                    post-delete="false"
                    post-rename="true"
                    post-suffix=".COMPLETE"
                    error-delete="false"
                    error-suffix=".HAS_ERROR"
                />
            </ftp-bus>
          </ftp-provider>

          <camel-provider name="provider1">
              <camel-bus busid="bus1">
                  <from uri="file:///tmp/input?delete=true"/>
              </camel-bus>
          </camel-provider>
```

Tree Source

# MyJMSListenerAction.java

```java
package org.jboss.soa.esb.samples.quickstart.helloworld;

import org.jboss.soa.esb.actions.AbstractActionLifecycle;

public class MyJMSListenerAction extends AbstractActionLifecycle
{

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

    public Message displayMessage(Message message) {
        logHeader();
        System.out.println("Body: " + message.getBody().get());
        logFooter();
        return message;
    }

    public Message playWithMessage(Message message) throws Exception {
        Body msgBody = message.getBody();
        String contents = msgBody.get().toString();
        StringBuffer sb = new StringBuffer();
        sb.append("[Changed] ");
        sb.append(contents);
        sb.trimToSize();
        msgBody.add(sb.toString());
        return message;
    }

    private void logHeader() {
        System.out.println("\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&");
    }

    private void logFooter() {
        System.out.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&\n");
    }
}
```

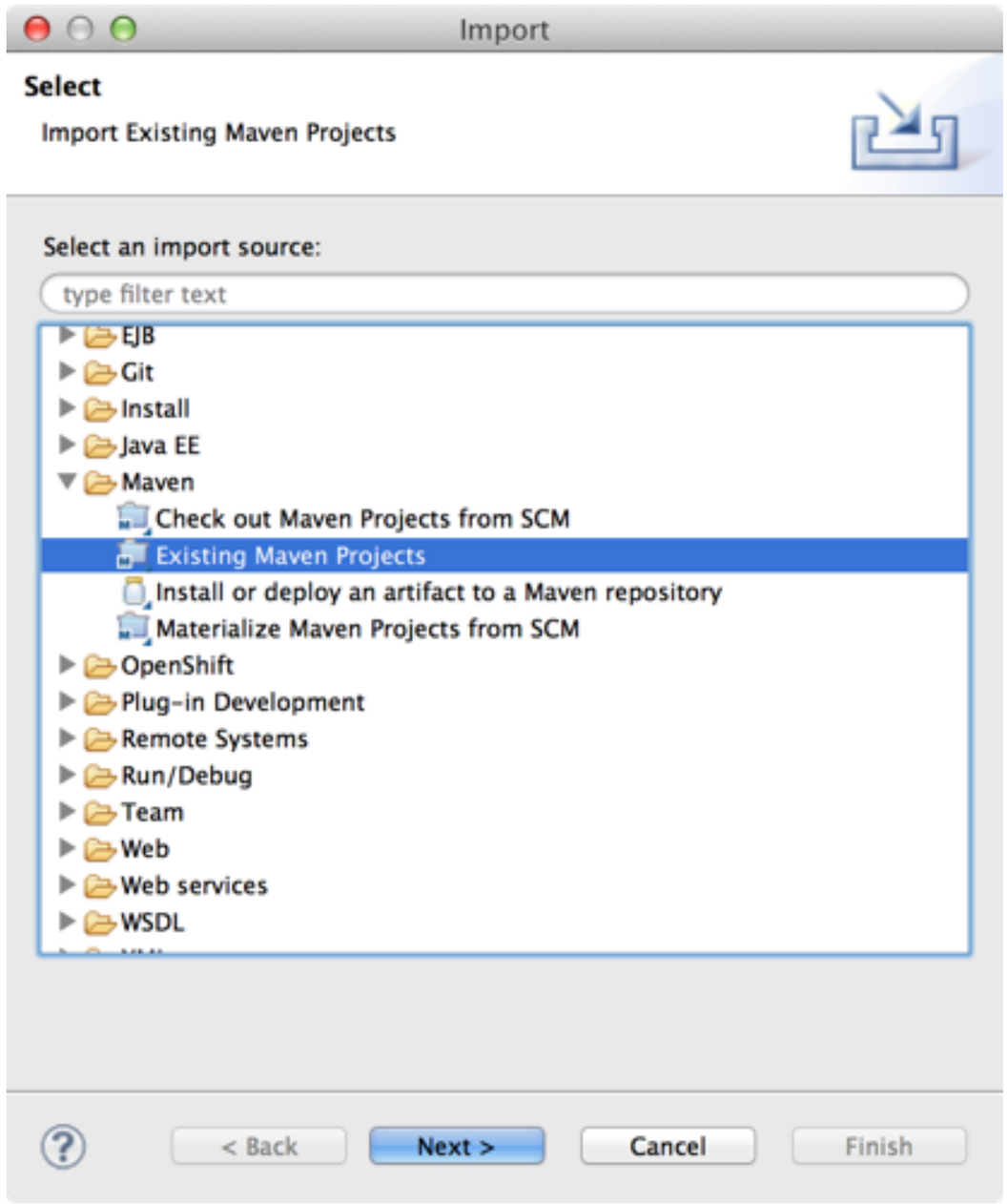
# Importing SOA 6 Gateways/ Routers

## TODO

1. File -> Import ... from the JBDS menu.

2. Select Maven -> Existing Maven Projects

3. Click Next



# Importing SOA 6 Hello World

## TODO

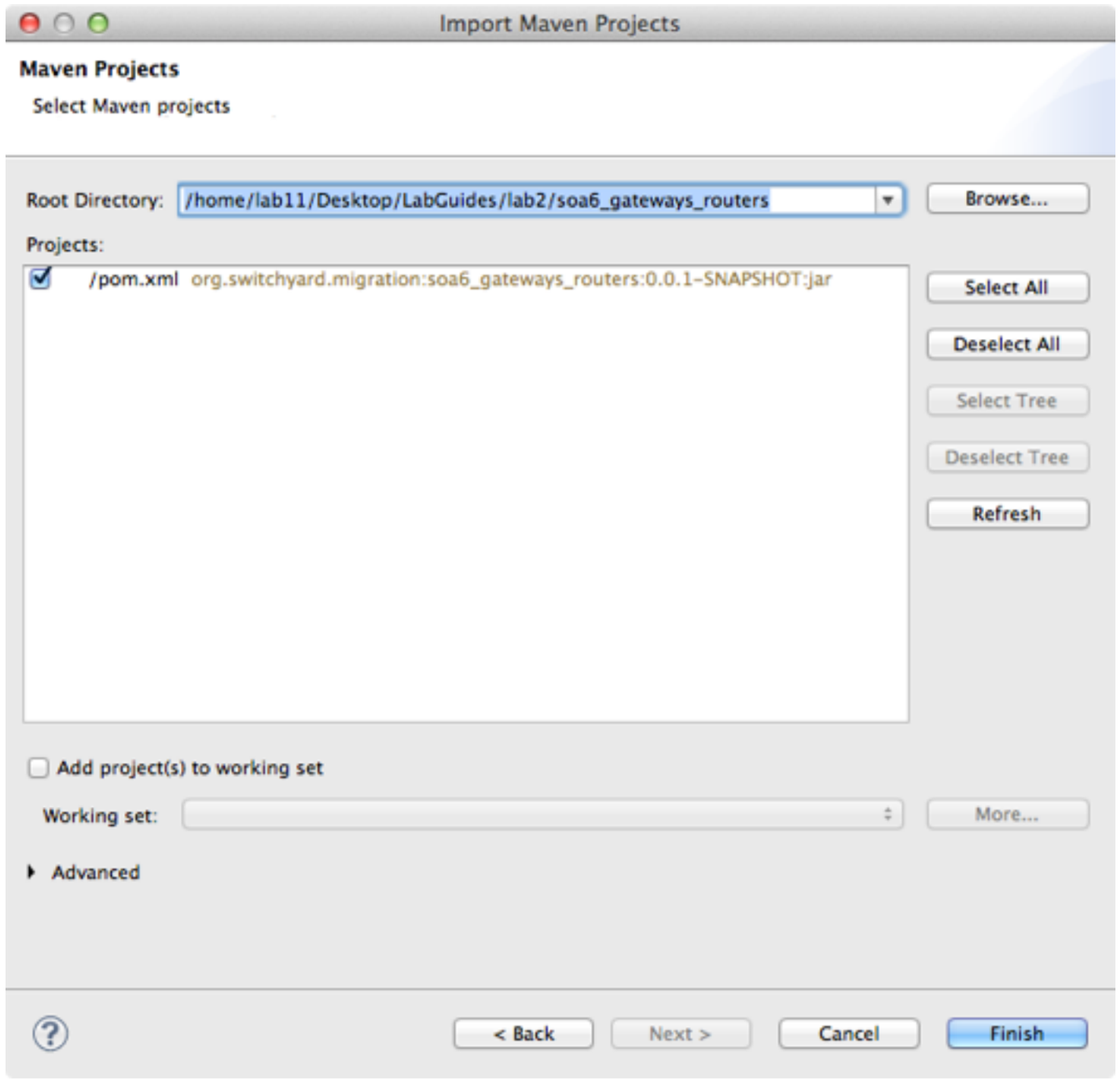
1. Click Browse ... and navigate to:

`/home/lab11/Desktop/LabGuides/lab2/soa6_gateways_routers`

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

`org.switchyard.migration:soa6_gateways_routers`

3. Click Finish

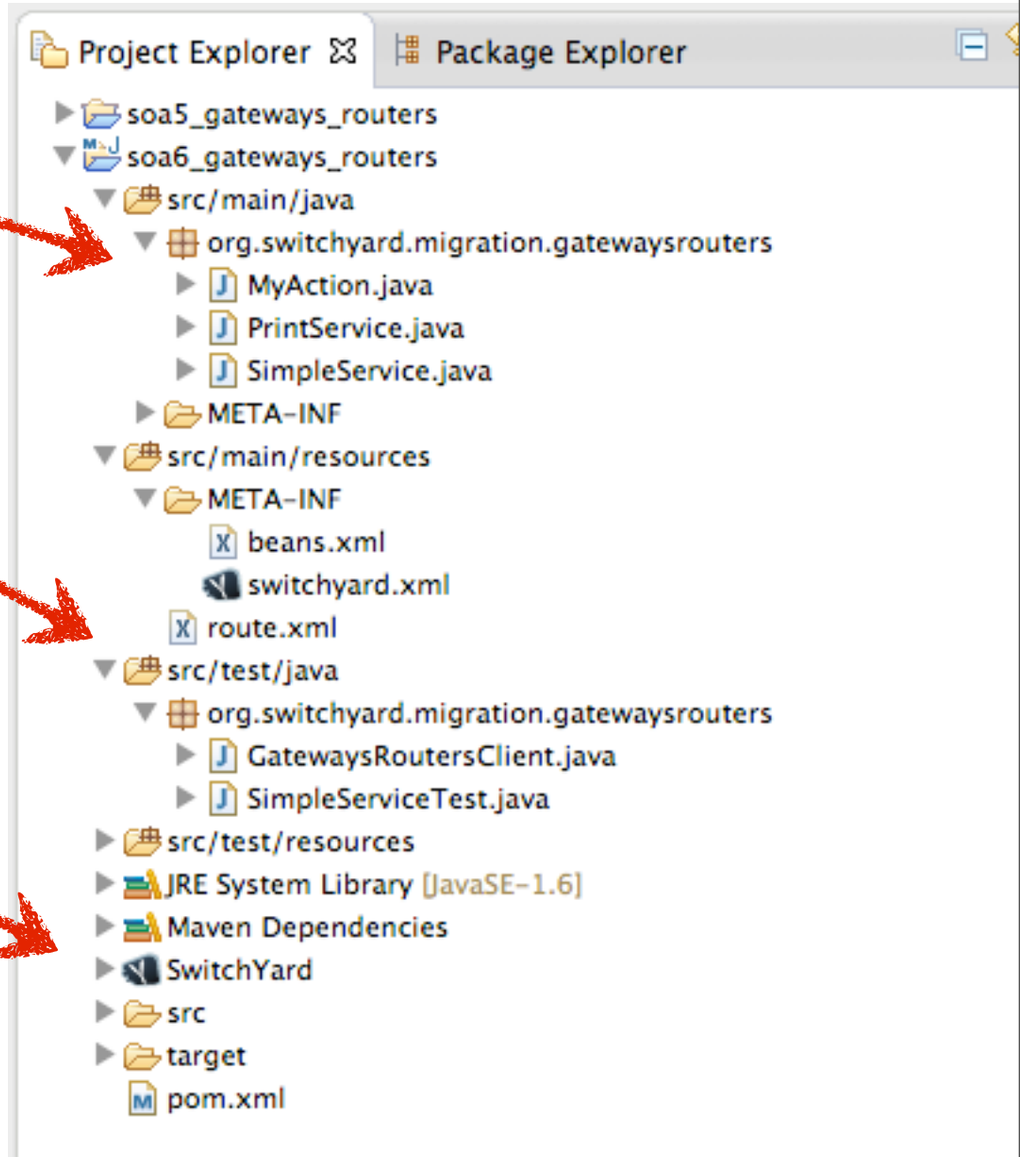


# Files To Note

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 `gateways_routers`

- 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 1

## Converting Action Class to Camel Service

**Notification**

- Action : service binding configuration in ftp-bus: busid="helloFTPChannel"
- Action : service binding configuration in ftp-bus: busid="bus1"
- Action : service binding configuration in jms-bus: busid="quickstartSimpleEsbChannel"
- Action : service binding configuration in ftp-bus: scheduleid="cron-schedule"
- Action : composite service required for service: name="SimpleService"
- Action : composite service binding required for listener: name="FtpGateway"
- Action : composite service binding required for listener: busidref="bus1"
- Action : create component service for action processing pipeline
- Action : convert SystemPrintln: class="org.jboss.soa.esb.actions.SystemPrintln"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert SystemPrintln: class="org.jboss.soa.esb.actions.SystemPrintln"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert StaticRouter to Camel routing: class="org.jboss.soa.esb.actions.StaticRouter"
- Action : convert JMSRouter: class="org.jboss.soa.esb.actions.routing.JMSRouter"

# 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.*

```
61.          <action name="action2"
62.            class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
63.            process="displayMessage"/>
```

**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.*

28. **public class** MyJMSListenerAction **extends** AbstractActionLifecycle

**Extending class 'org.jboss.soa.esb.actions.AbstractActionLifecycle' at line 28**

Action classes in SOA 5 are simply CDI Beans in SOA 6. The extension of AbstractionActionLifecycle 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.

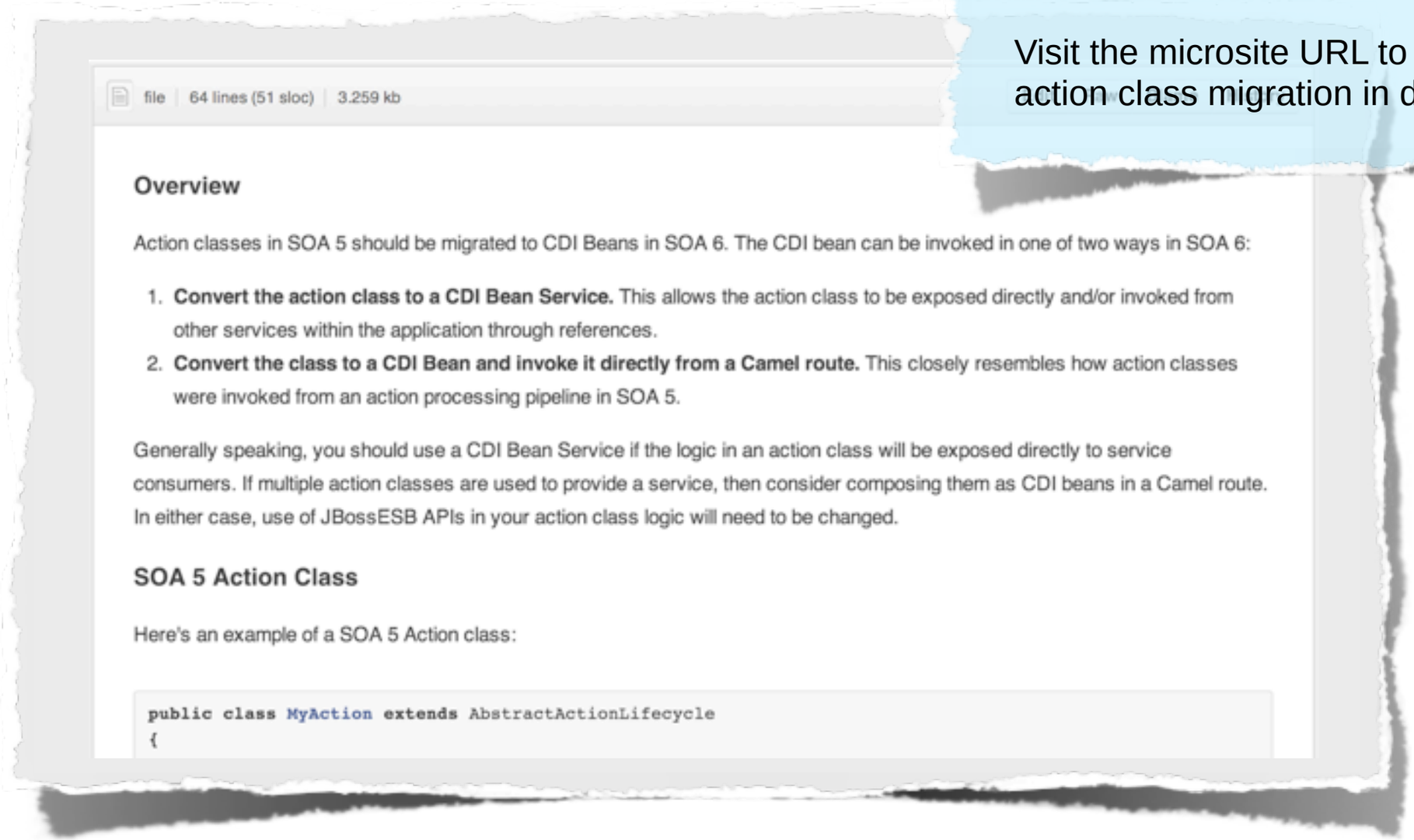
For additional information and tips, see the action class microsite.

# Microsite

action class

**TODO**

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



file | 64 lines (51 sloc) | 3.259 kb

## Overview

Action classes in SOA 5 should be migrated to CDI Beans in SOA 6. The CDI bean can be invoked in one of two ways in SOA 6:

1. **Convert the action class to a CDI Bean Service.** This allows the action class to be exposed directly and/or invoked from other services within the application through references.
2. **Convert the class to a CDI Bean and invoke it directly from a Camel route.** This closely resembles how action classes were invoked from an action processing pipeline in SOA 5.

Generally speaking, you should use a CDI Bean Service if the logic in an action class will be exposed directly to service consumers. If multiple action classes are used to provide a service, then consider composing them as CDI beans in a Camel route. In either case, use of JBossESB APIs in your action class logic will need to be changed.

### SOA 5 Action Class

Here's an example of a SOA 5 Action class:

```
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 PrintService in SOA 6. All services in SOA 6 have a contract.*

```
package org.switchyard.migration.gatewaysrouters;

public interface PrintService {

    void print(String message);
}
```

# Step 2

## Converting the Action Processing Pipeline

**Notification**

- Action : service binding configuration in ftp-bus: busid="helloFTPChannel"
- Action : service binding configuration in ftp-bus: busid="bus1"
- Action : service binding configuration in jms-bus: busid="quickstartSimpleEsbChannel"
- Action : service binding configuration in ftp-bus: scheduleid="cron-schedule"
- Action : composite service required for service: name="SimpleService"
- Action : composite service binding required for listener: name="FtpGateway"
- Action : composite service binding required for listener: busidref="bus1"
- Action : create component service for action processing pipeline
- Action : convert SystemPrintln: class="org.jboss.soa.esb.actions.SystemPrintln"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert SystemPrintln: class="org.jboss.soa.esb.actions.SystemPrintln"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert StaticRouter to Camel routing: class="org.jboss.soa.esb.actions.StaticRouter"
- Action : convert JMSRouter: class="org.jboss.soa.esb.actions.routing.JMSRouter"

# Action Processing Pipeline

```
<actions mep="OneWay">
    <action name="action1" class="org.jboss.soa.esb.actions.SystemPrintln">
        <property name="printfull" value="false"/>
    </action>
    <action name="action2"
        class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
        process="displayMessage"/>
    <action name="action3" class="org.jboss.soa.esb.actions.SystemPrintln">
        <property name="printfull" value="false"/>
    </action>
    <action name="action4"
        class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
        process="playWithMessage"/>
    <action name="action5"
        class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
        process="displayMessage"/>

    <action name="routeAction"  class="org.jboss.soa.esb.actions.StaticRouter">
            <property name="destinations">
                    <route-to service-category="myCategory" service-name="PrintService" />
            </property>
    </action>

    <action name="routeToReplyQueue" class="org.jboss.soa.esb.actions.routing.JMSRouter">
        <property name="connection-factory" value="ConnectionFactory"/>
        <property name="jndiName" value="queue/quickstart_jms_router_routeTo"/>
        <property name="unwrap" value="true"/>
        <property name="security-principal" value="guest"/>
        <property name="security-credential" value="guest"/>

        <property name="jndi-prefixes" value="org.xyz."/>
        <property name="org.xyz.propertyName" value="propertyValue"/>

        <property name="java.naming.someproperty" value="propertyValue"/>
    </action>
</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.

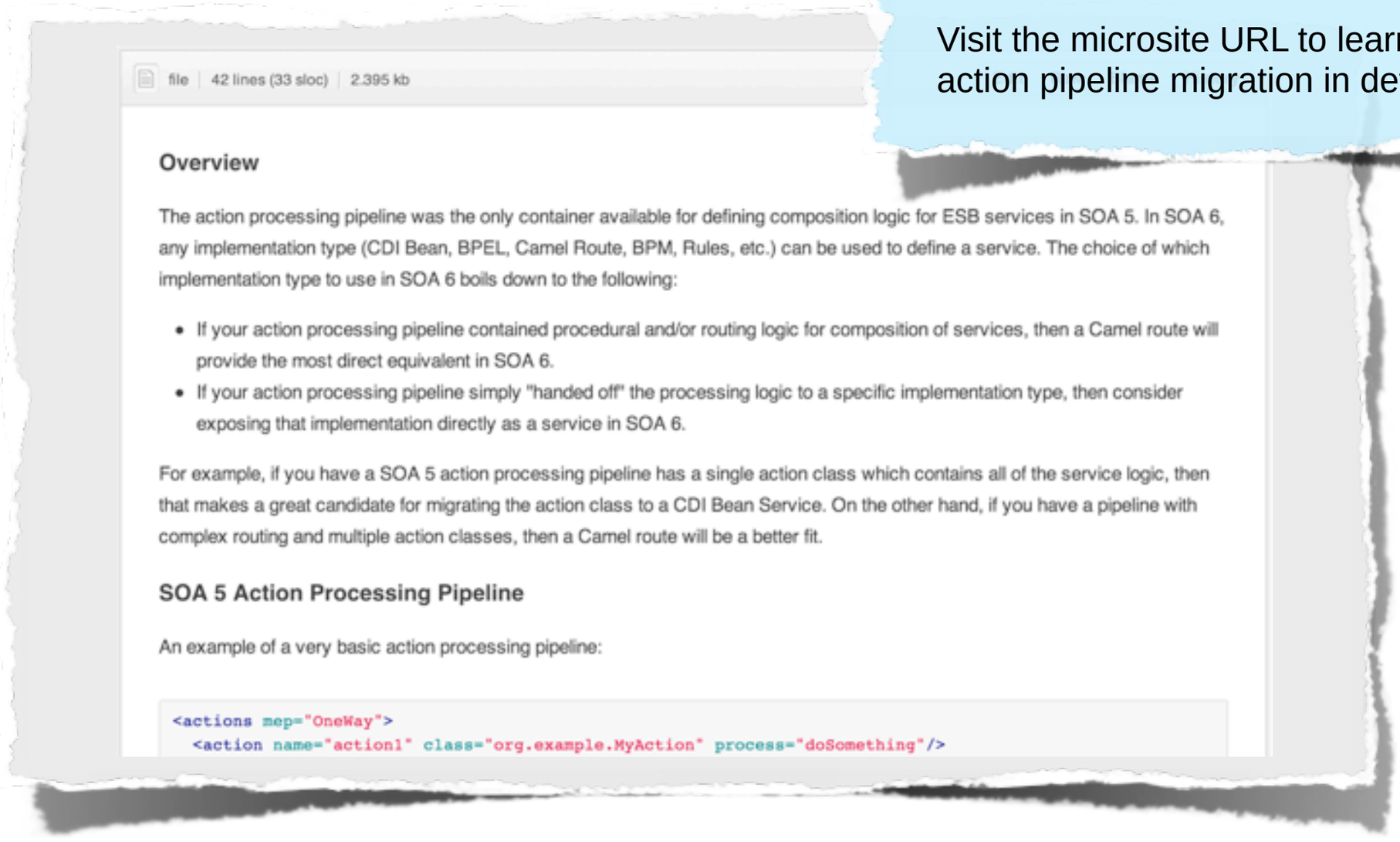
For additional information and tips, see the action pipline microsite.

# Microsite

action pipeline

## TODO

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


file | 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.

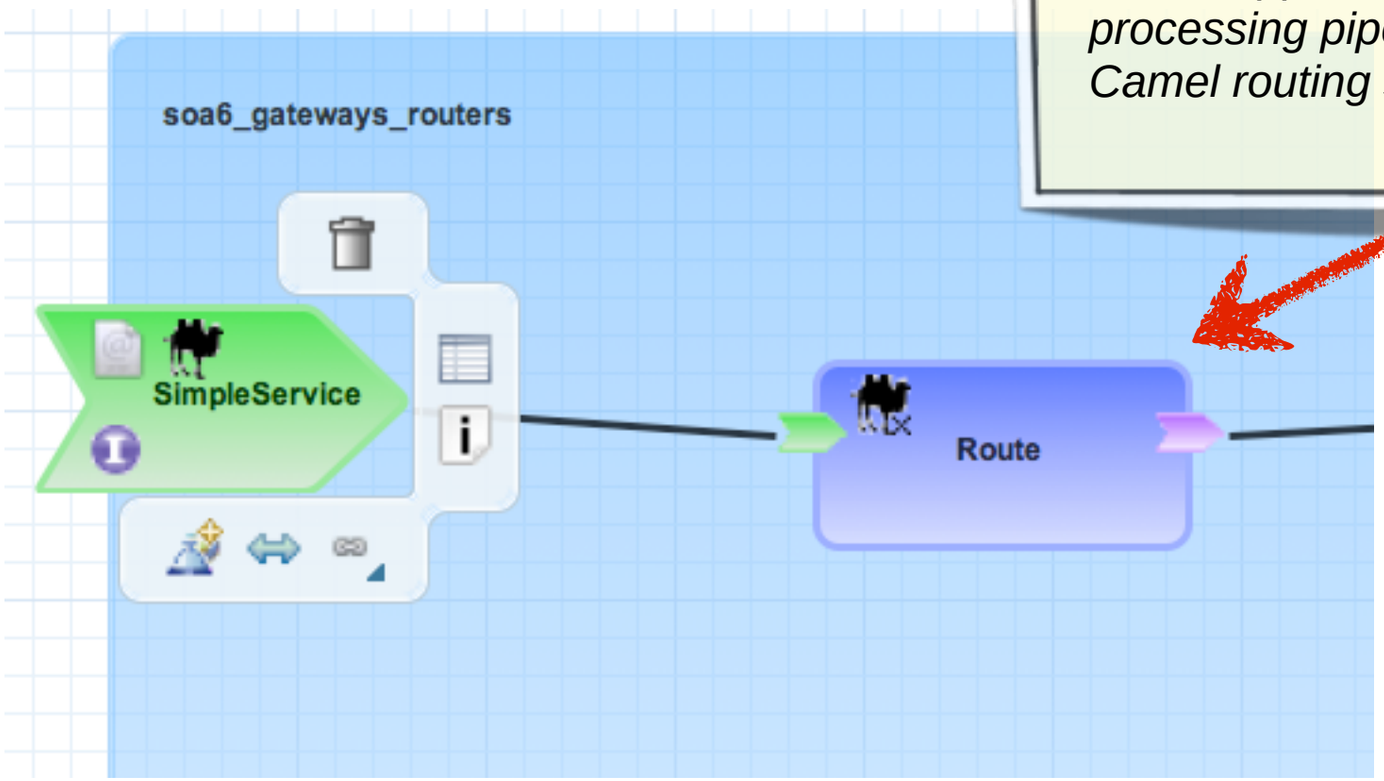
### SOA 5 Action Processing Pipeline

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




**FYI**

*This is the component service definition in our SOA 6 application. Note that the action processing pipeline has been replaced by a Camel routing service.*

# Step 3

## Creating Composite Service For ESB Services

**Notification**

- Action : service binding configuration in ftp-bus: busid="helloFTPChannel"
- Action : service binding configuration in ftp-bus: busid="bus1"
- Action : service binding configuration in jms-bus: busid="quickstartSimpleEsbChannel"
- Action : service binding configuration in ftp-bus: scheduleid="cron-schedule"
- Action : composite service required for service: name="SimpleService"
- Action : composite service binding required for listener: name="FtpGateway"
- Action : composite service binding required for listener: busidref="bus1"
- Action : create component service for action processing pipeline
- Action : convert SystemPrintln: class="org.jboss.soa.esb.actions.SystemPrintln"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert SystemPrintln: class="org.jboss.soa.esb.actions.SystemPrintln"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert StaticRouter to Camel routing: class="org.jboss.soa.esb.actions.StaticRouter"
- Action : convert JMSRouter: class="org.jboss.soa.esb.actions.routing.JMSRouter"

# Service Definition

```
45.        <services>
46.      <service category="myCategory" name="SimpleService"
47.            description="SOA 5 camel and ftp listening service" invmScope="GLOBAL">
```

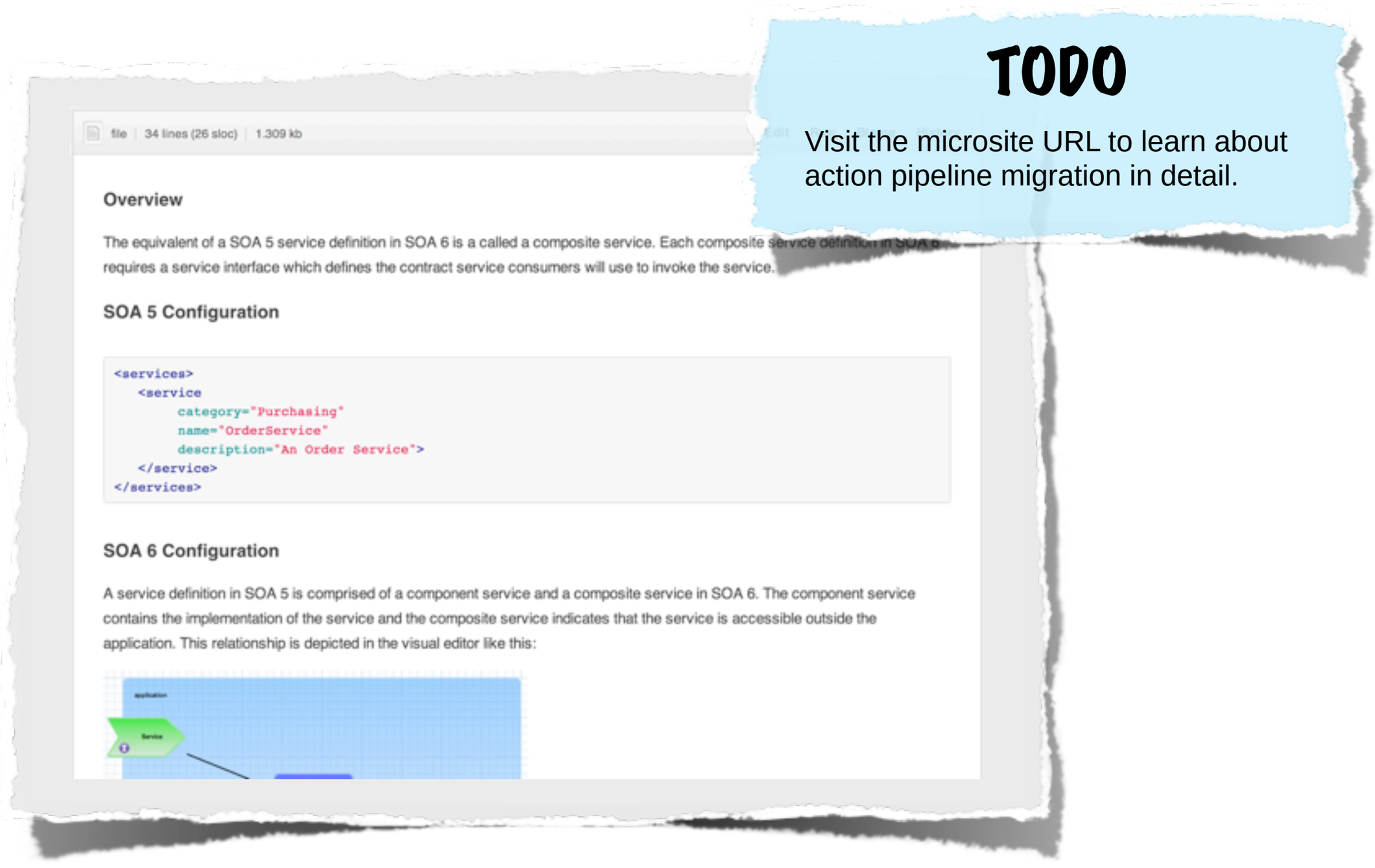
**Action : composite service required for service: name="SimpleService"**

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.

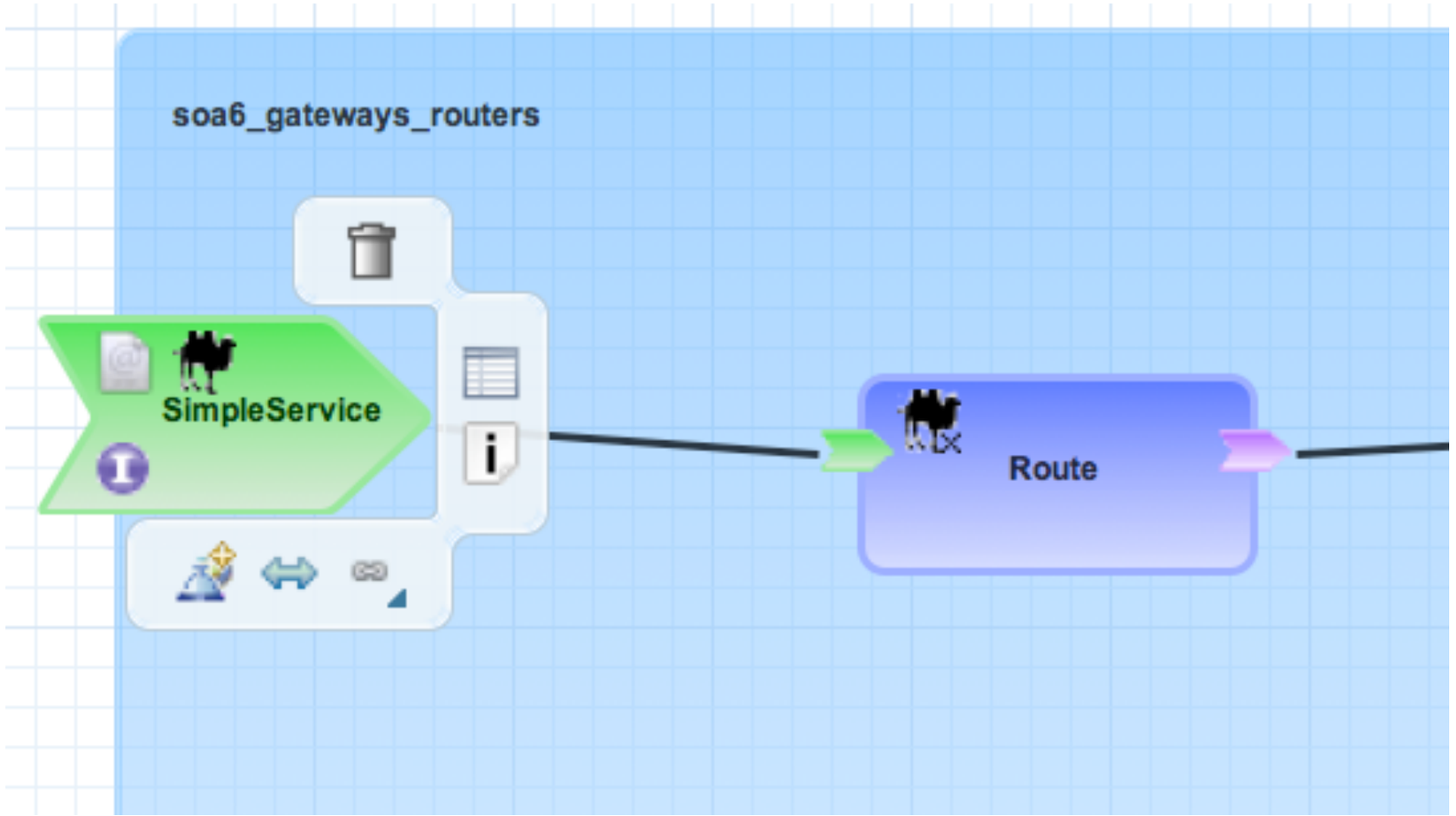
# Microsite

## service migration




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

# Composite Service



# Step 4

## Migrating FTP and Camel Gateway Listeners

**Notification**

- Action : service binding configuration in ftp-bus: busid="helloFTPChannel"
- Action : service binding configuration in ftp-bus: busid="bus1"
- Action : service binding configuration in jms-bus: busid="quickstartSimpleEsbChannel"
- Action : service binding configuration in ftp-bus: scheduleid="cron-schedule"
- Action : composite service required for service: name="SimpleService"
- Action : composite service binding required for listener: name="FtpGateway"
- Action : composite service binding required for listener: busidref="bus1"
- Action : create component service for action processing pipeline
- Action : convert SystemPrintln: class="org.jboss.soa.esb.actions.SystemPrintln"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert SystemPrintln: class="org.jboss.soa.esb.actions.SystemPrintln"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert action class: class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
- Action : convert StaticRouter to Camel routing: class="org.jboss.soa.esb.actions.StaticRouter"
- Action : convert JMSRouter: class="org.jboss.soa.esb.actions.routing.JMSRouter"

# Gateway Listener

```
49.     <ftp-listener name="FtpGateway"
50.         busidref="helloFTPChannel"
51.         is-gateway="true"
52.         scheduleidref="cron-schedule"/>
```

**Action : composite service binding required for listener: name="FtpGateway"**

This listener requires a composite service binding in SwitchYard. The configuration for a FTP binding can be found in the ftp-bus definition associated with this listener.

For additional information and tips, see the gateway listener microsite.

# Gateway Listener

53. `<camel-gateway name="gateway1" busidref="bus1"/>`

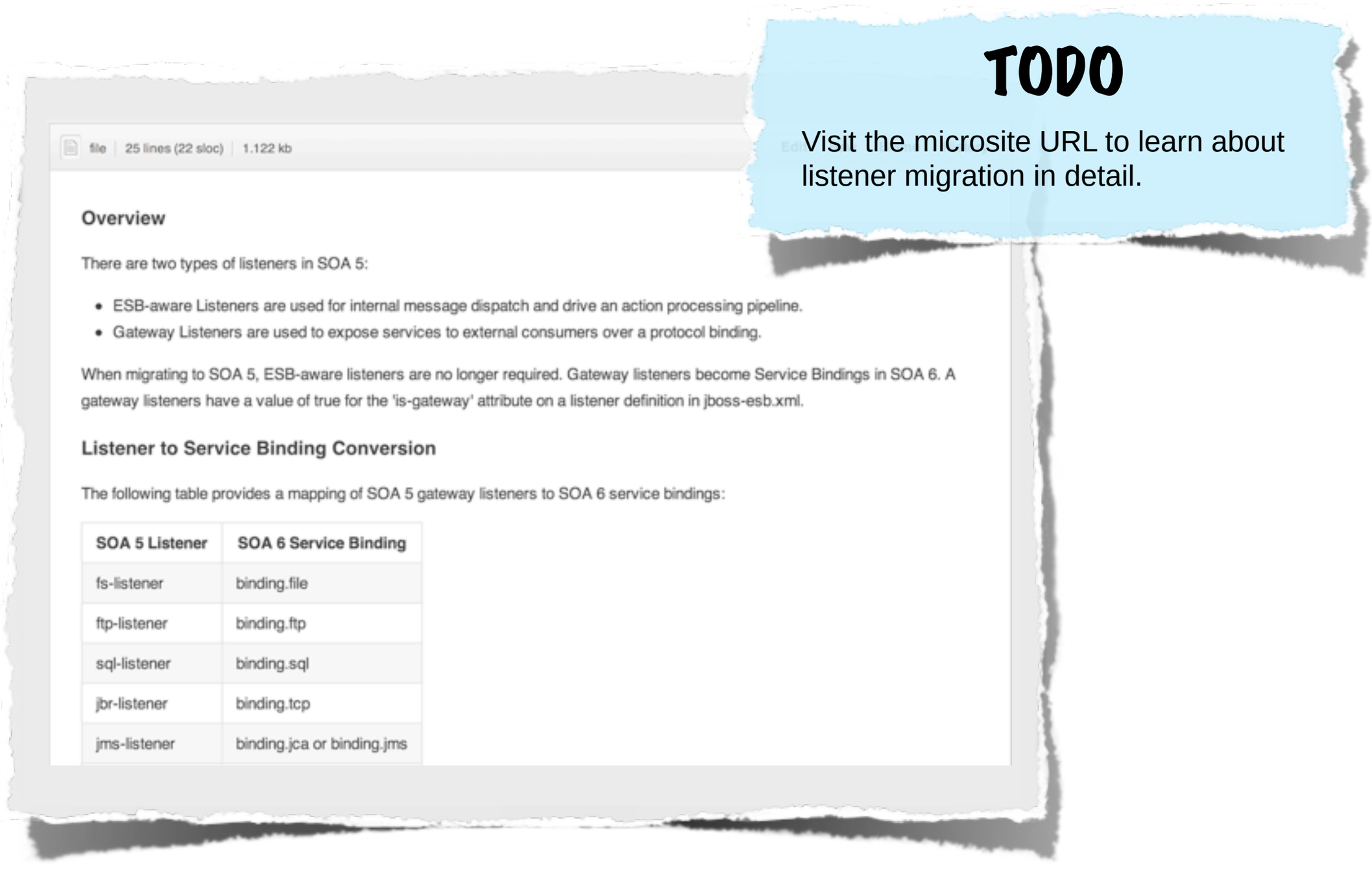
**Action : composite service binding required for listener: busidref="bus1"**

This gateway requires a composite service binding in SwitchYard. The configuration for a Camel binding can be found in the camel-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.

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

# Gateway Listener Config

```
26.          <camel-bus busid="bus1">
```

**Action : service binding configuration in ftp-bus: busid="bus1"**

A camel-bus definition can be converted to a Camel gateway binding on a composite service in SwitchYard.

For additional information and tips, see the camel-bus migration microsite.

```
27.              <from uri="file:///tmp/input?delete=true"/>
28.          </camel-bus>
```

# Gateway Listener Config

```
08.          <ftp-bus busid="helloFTPChannel" >
```

**Action : service binding configuration in ftp-bus: busid="helloFTPChannel"**

A ftp-bus definition can be converted to a FTP gateway binding on a composite service in SwitchYard.

For additional information and tips, see the ftp-bus migration microsite.

```
09.             <ftp-message-filter
10.                username="ftpboy"
11.                password="camel123."
12.                passive="false"
13.                directory="/tmp/ftpinput"
14.                input-suffix="ftpin.dat"
15.                work-suffix=".esbWorking"
16.                post-delete="false"
17.                post-rename="true"
18.                post-suffix=".COMPLETE"
19.                error-delete="false"
20.                error-suffix=".HAS_ERROR"
21.             />
22.          </ftp-bus>
```

# Microsite

## ftp-bus migration


**TODO**

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

https://github.com/windup/soa-migration/blob/master/advice/ftp-bus-migration.md

# Microsite

## camel-bus migration

**TODO**

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

file | 37 lines (23 sloc) | 1.091 kb

### Overview

The camel-bus element in jboss-esb.xml provides configuration for the Camel gateway. The configuration found in a camel-bus definition can be converted into a Camel binding in SwitchYard:

### SOA 5 Configuration

```
<camel-bus busid="bus1">
    <from uri="file:///tmp/input?delete=true"/>
</camel-bus>
```

### SOA 6

Converting to Camel binding

You will need the following pieces of information to migrate a camel-bus definition to a Camel binding:

- from URI - available in camel-bus

The relevant bits of the application descriptor are included for reference:

# Service Bindings

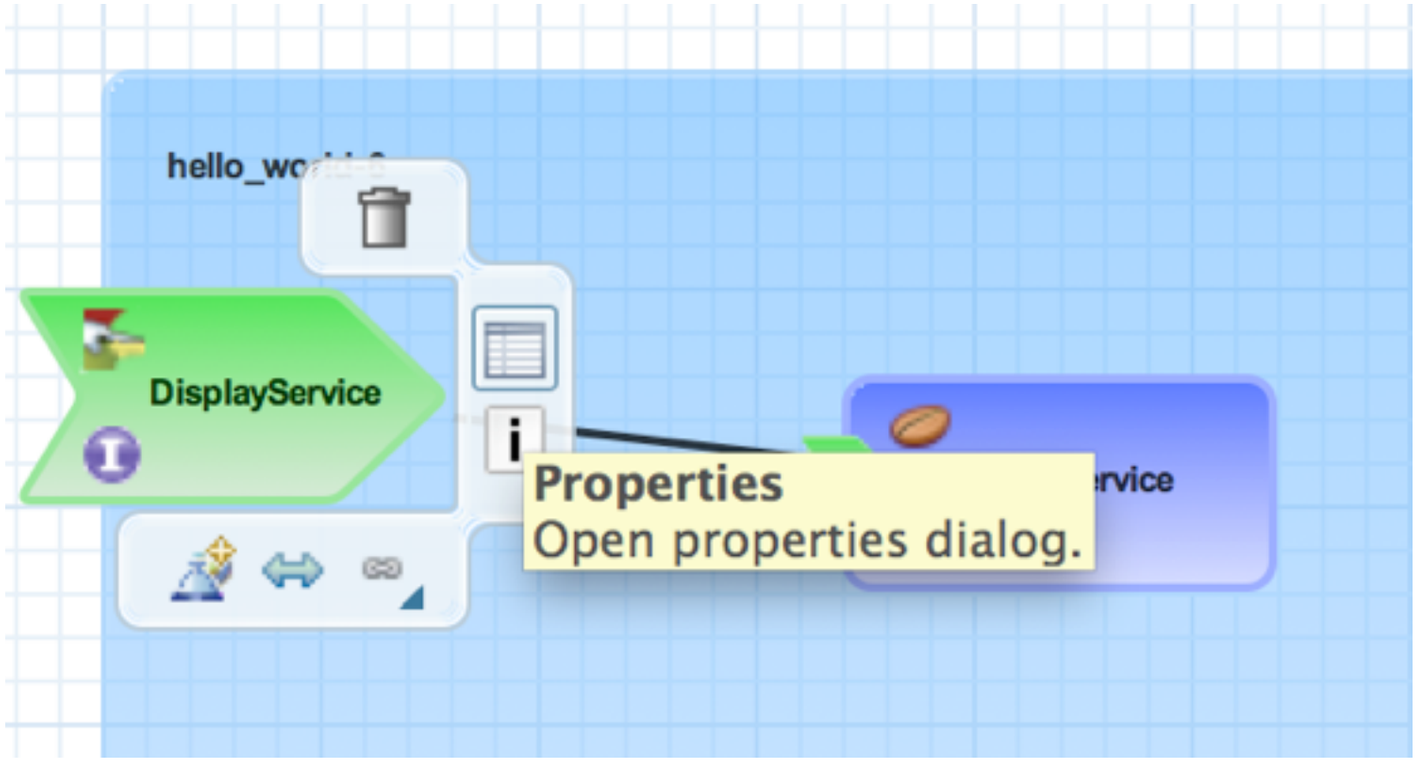
## TODO

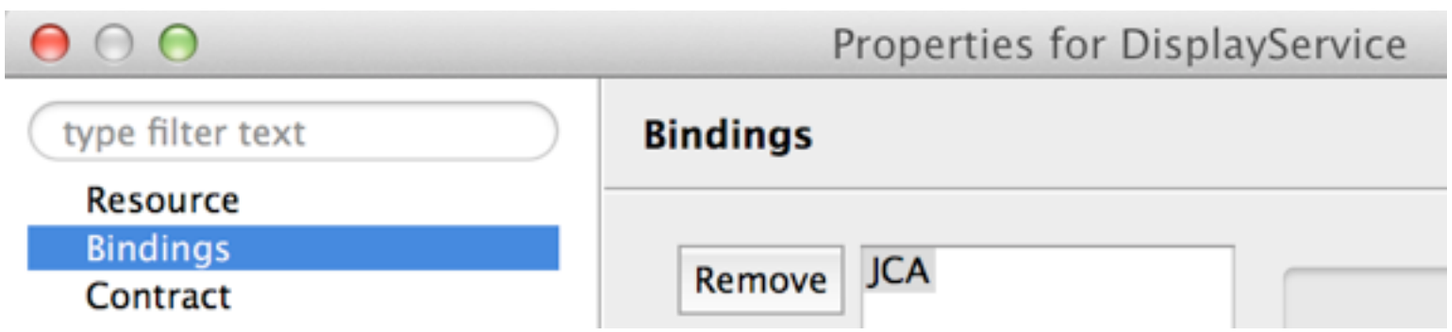
1. Bring up the button bar for the composite service ‘DisplayService’ by hovering over the large green service icon on the left of the composite.

2. When the button bar appears, click on the table icon which brings up the service properties.

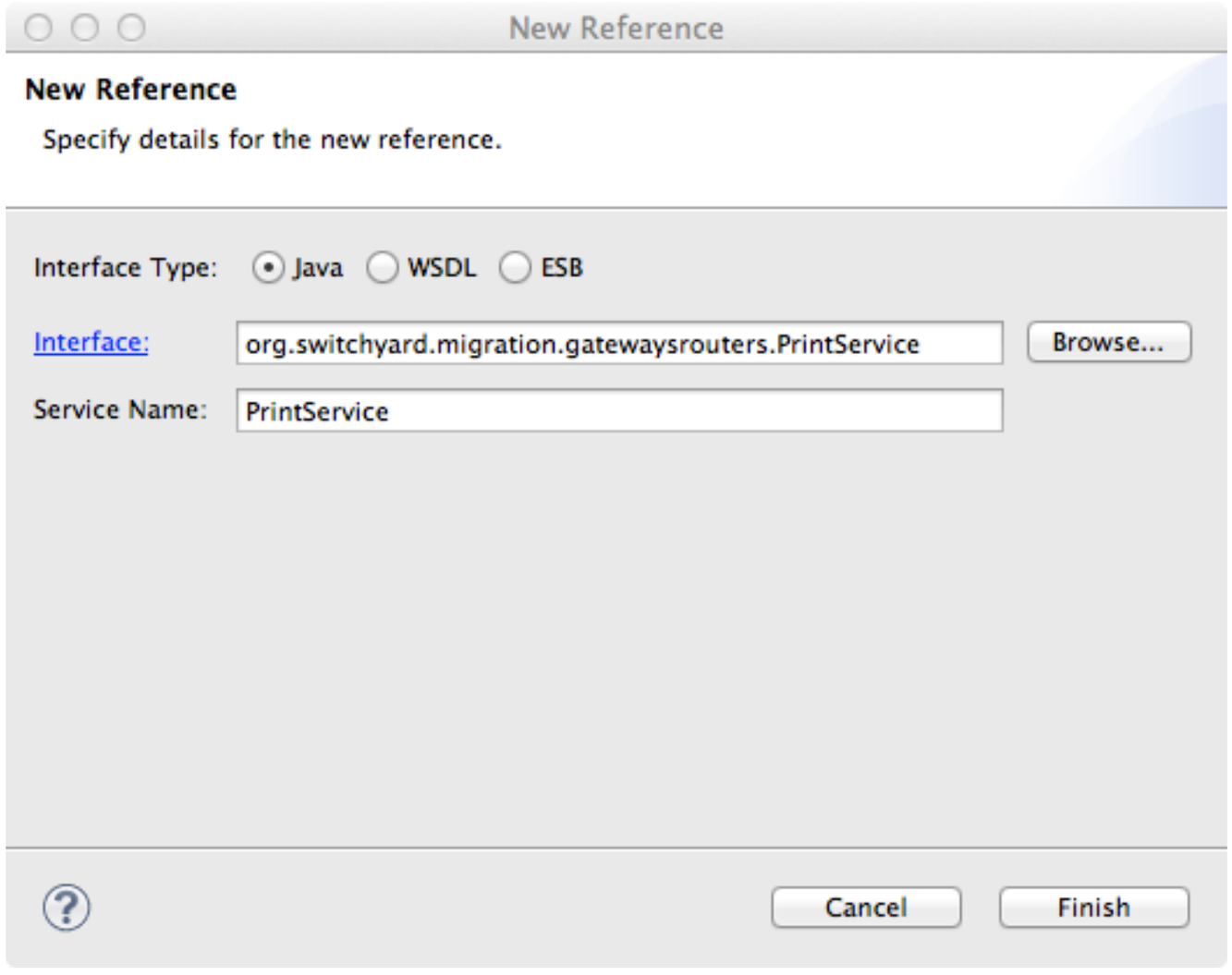
3. On the resulting property page, select the Bindings page in the left frame.

4. View the configuration details of the FTP binding definition in SOA 6.

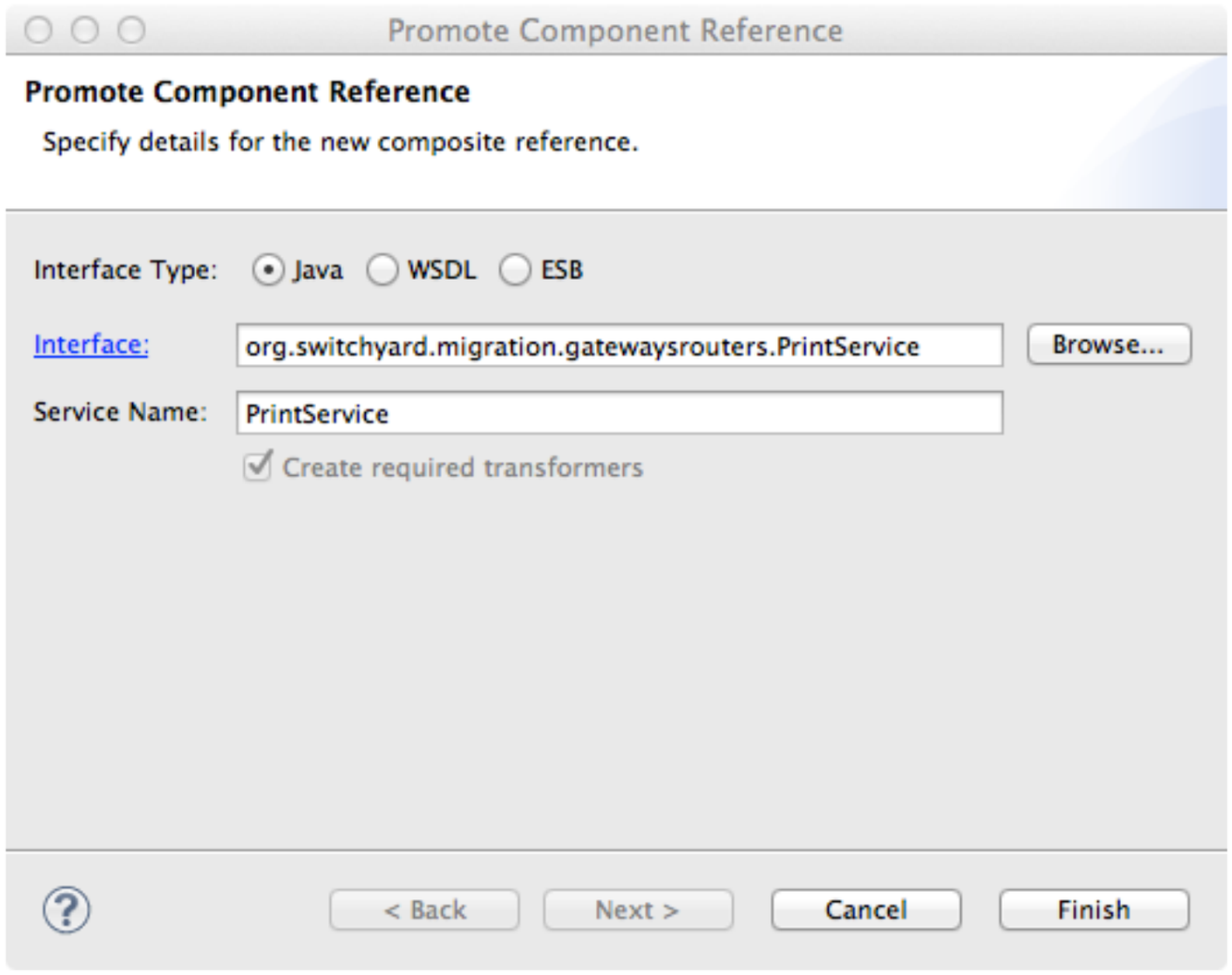


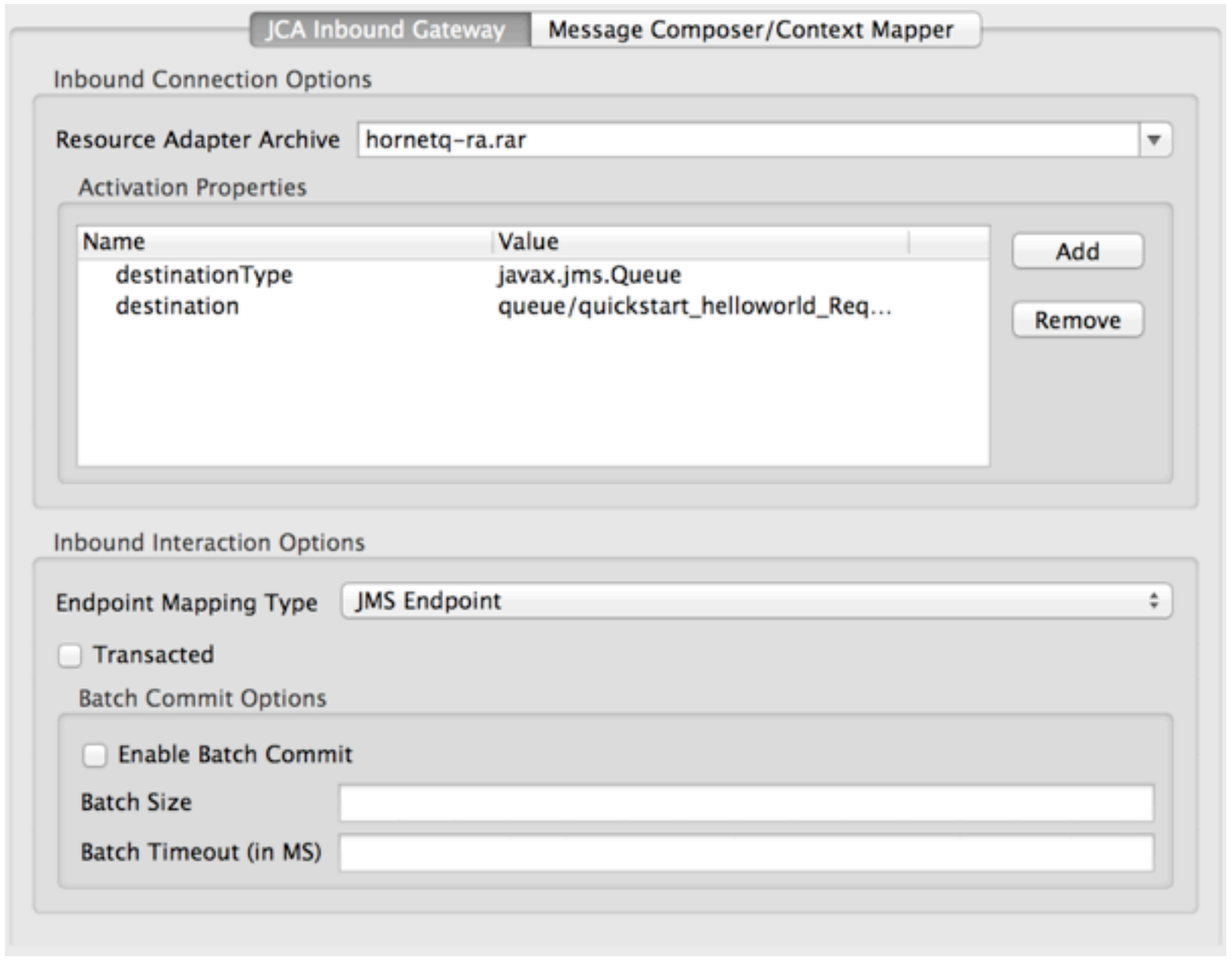
# References


New Reference
New Reference
Specify details for the new reference.
Interface Type:
Java
WSDL
ESB
Interface:
org.switchyard.migration.gatewaysrouters.PrintService
Browse...
Service Name:
PrintService
Cancel
Finish

# Promote Reference



# Binding Configuration

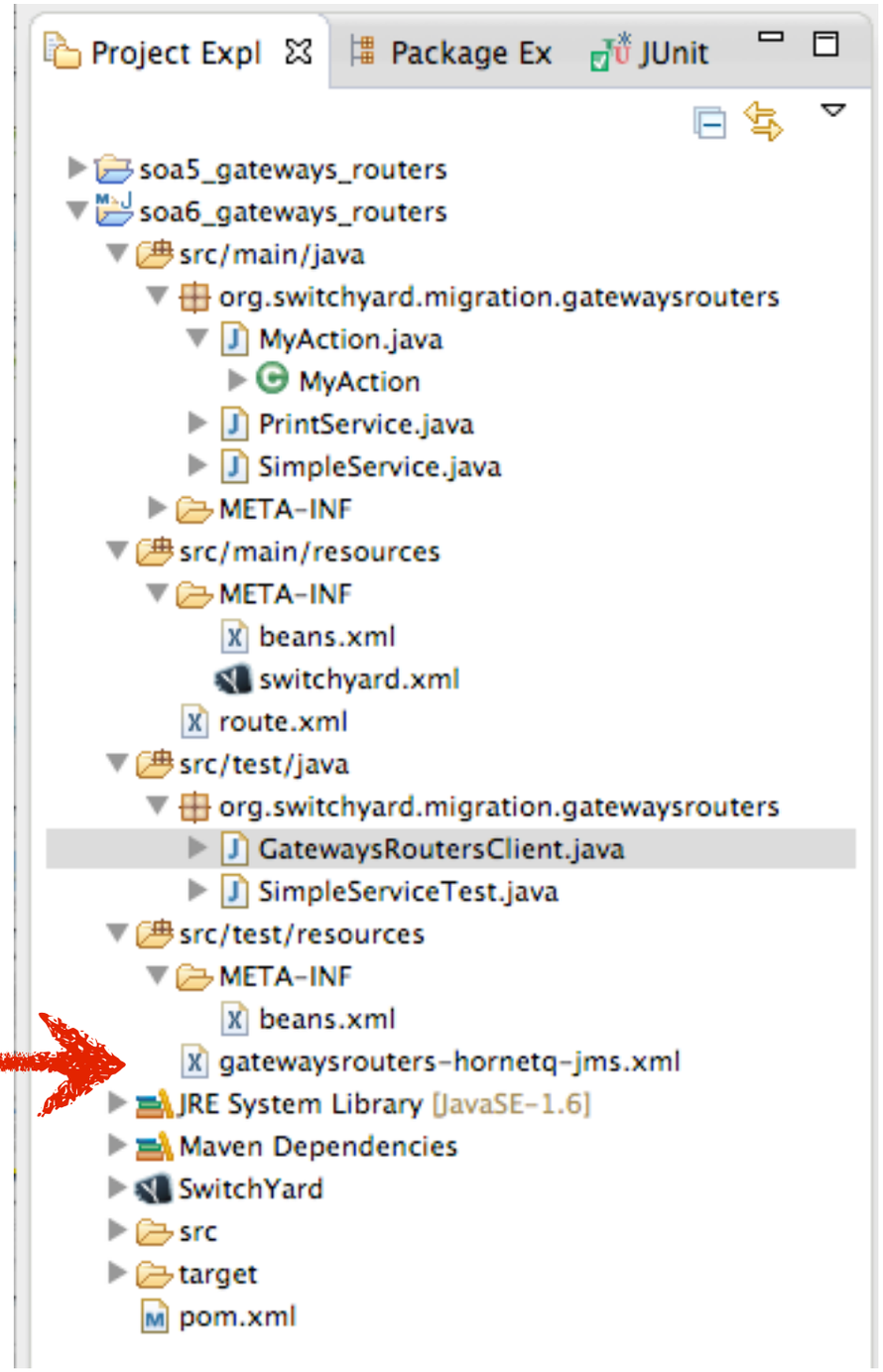


# Deploy Queue

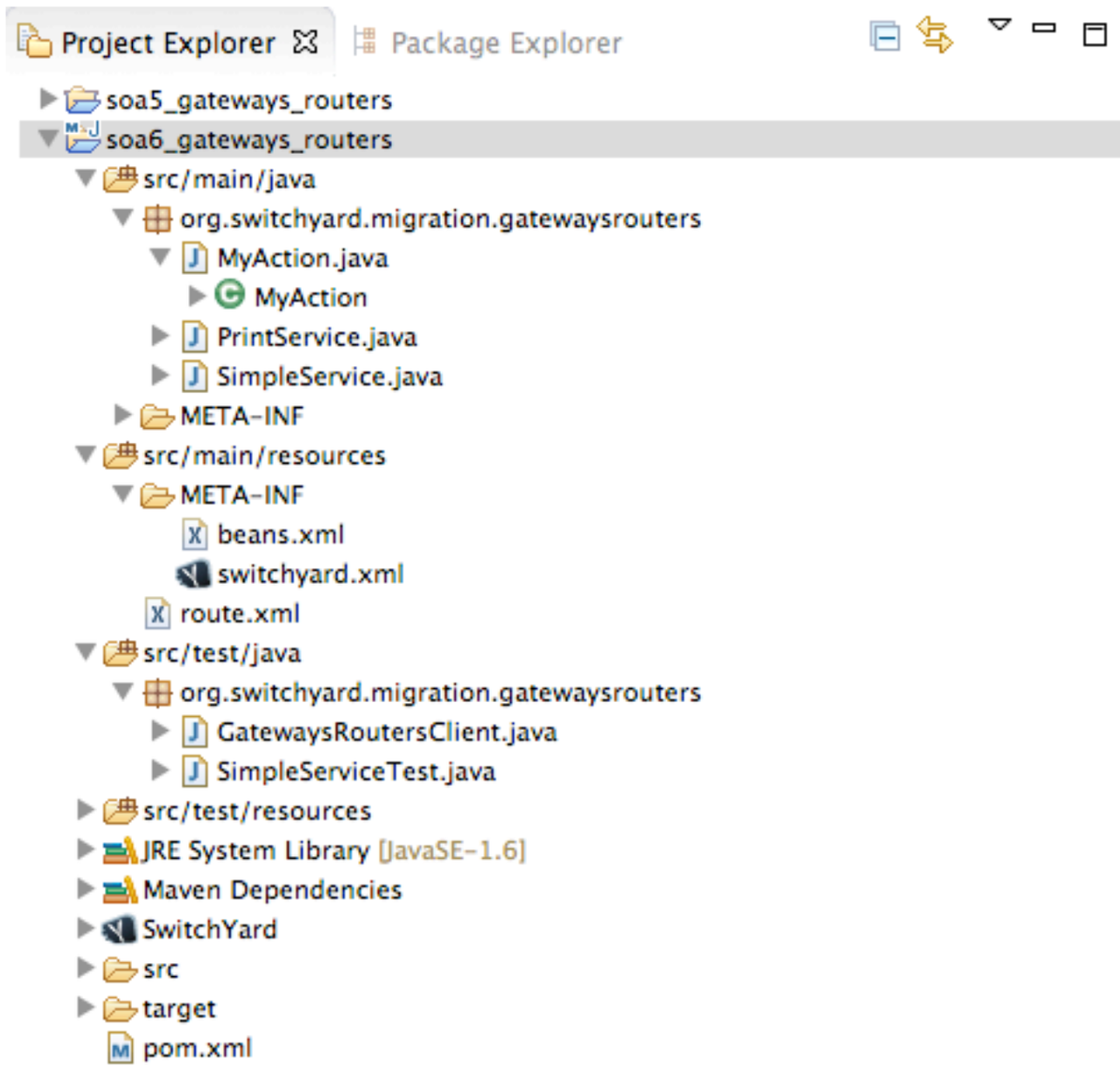
## TODO

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

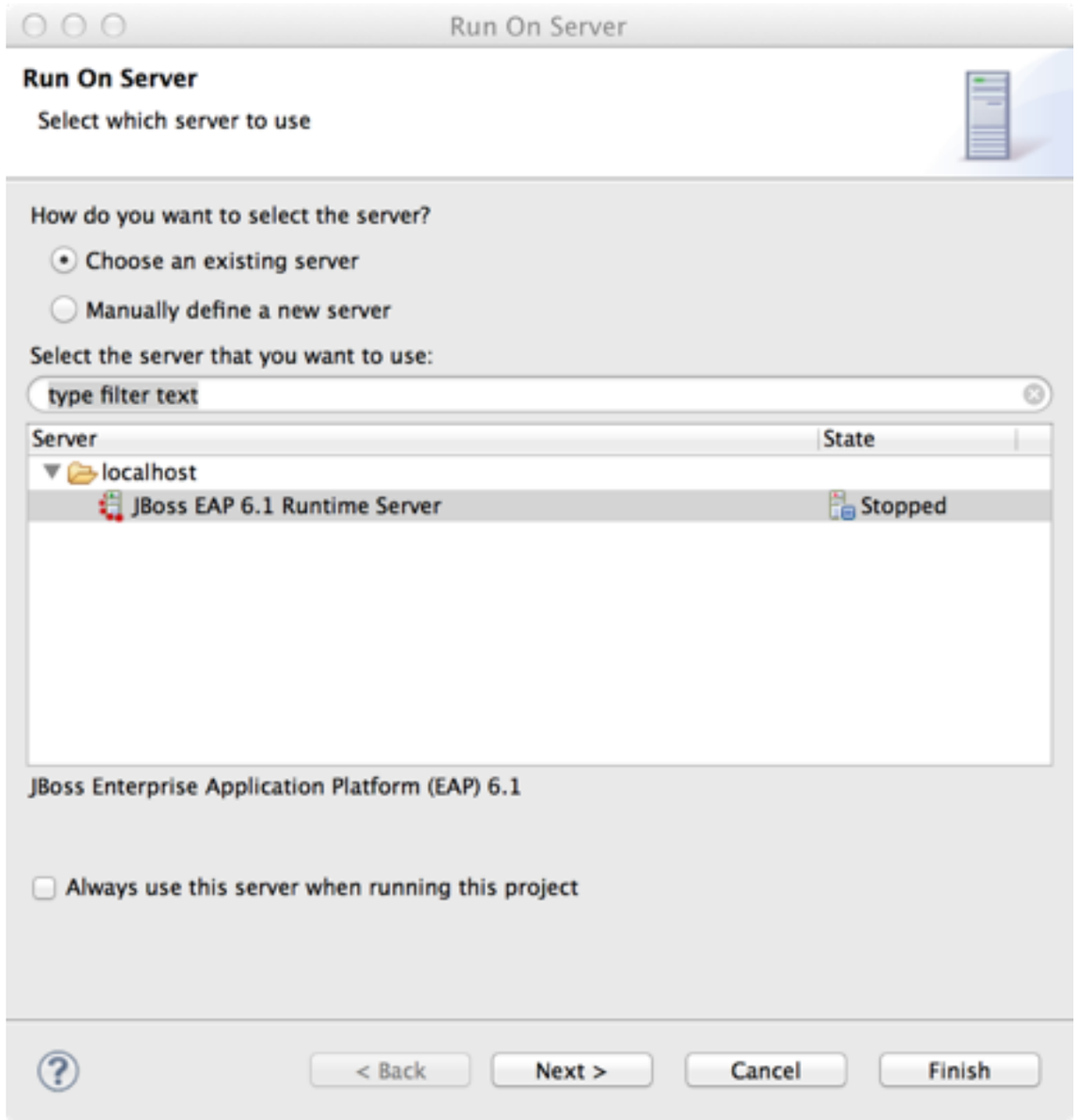
1. Right-click on the soa6_gateways_routers project and select Run As ... Run on Server

# Select Server

## TODO

1. Select the JBoss EAP 6.1 Runtime Server

2. Click Finish



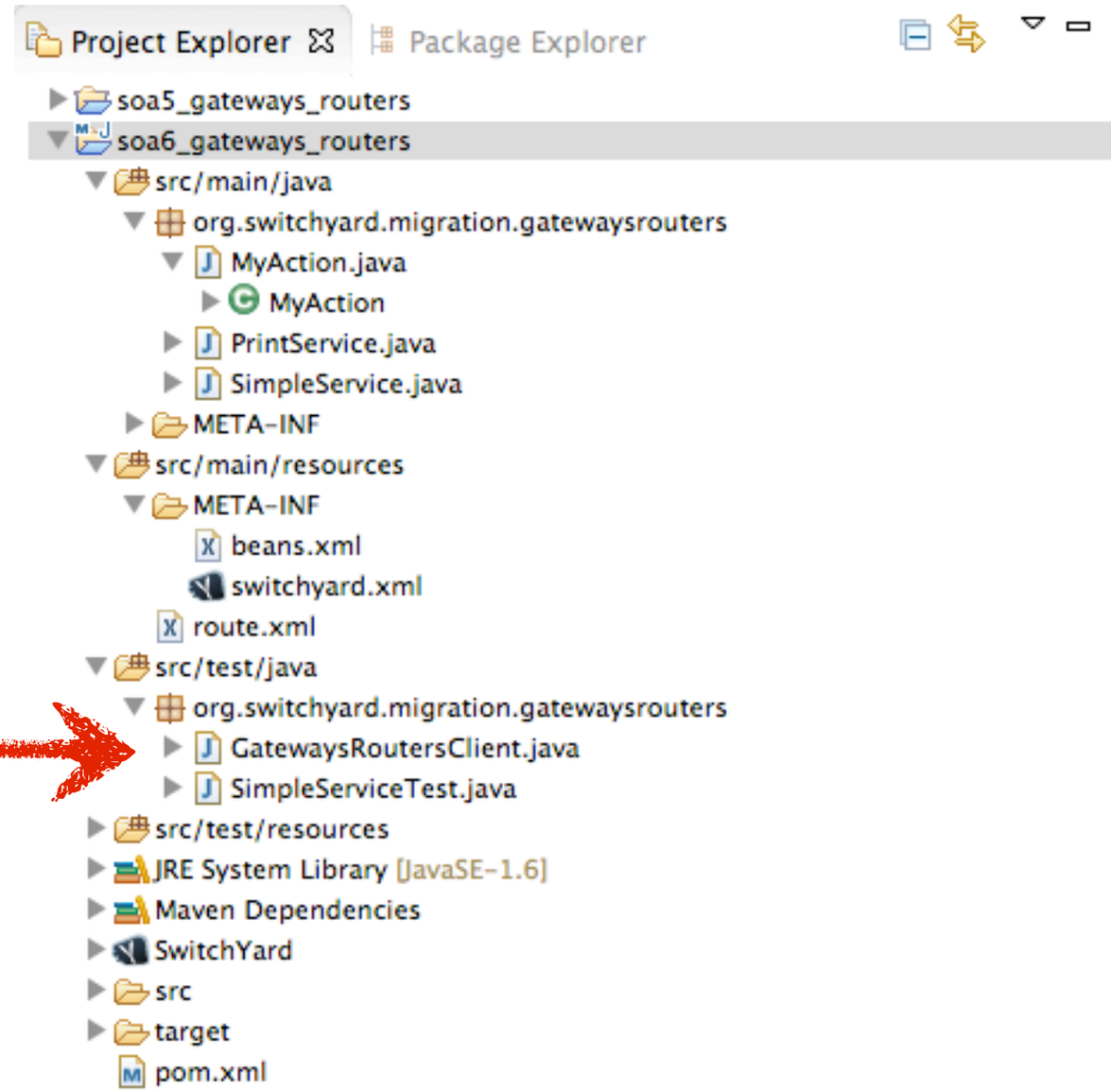
# Run Test Client

## TODO

1. Create a file in either /tmp/input/filein.dat or /tmp/input/ftpin.dat

2. Open GatewaysRoutersClient.java from the Project Explorer view.

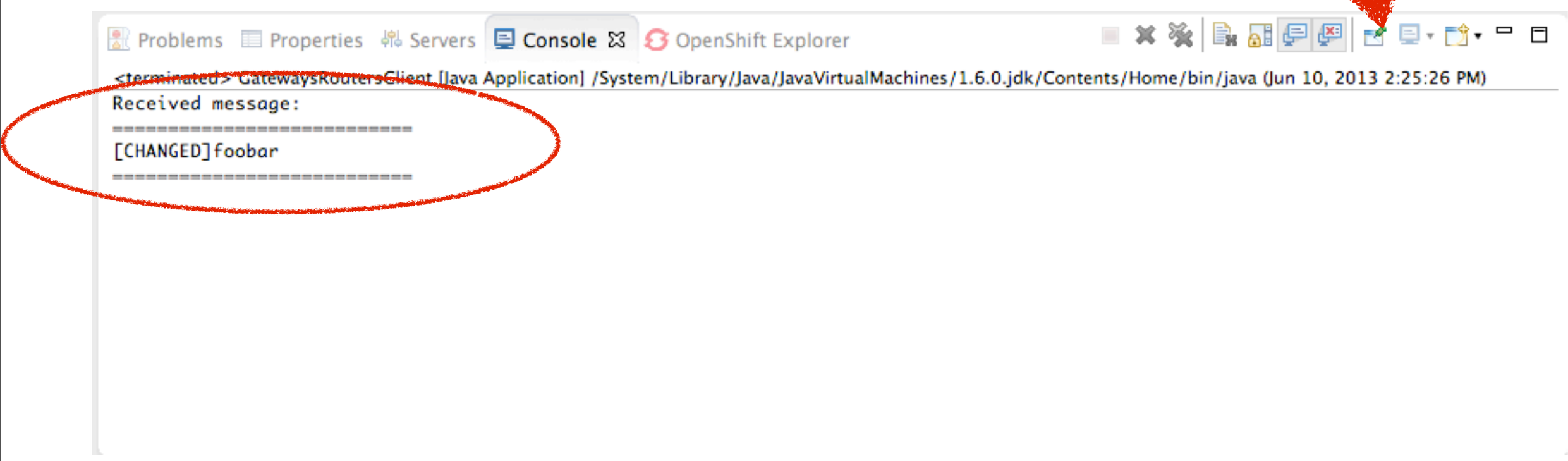
3. Go to the Run menu in the main menu bar and select ‘Run As -> Java Application’



# Verify Output

**TODO**

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



# Lab 2 Complete!