

# Lab 3

invm\_transport

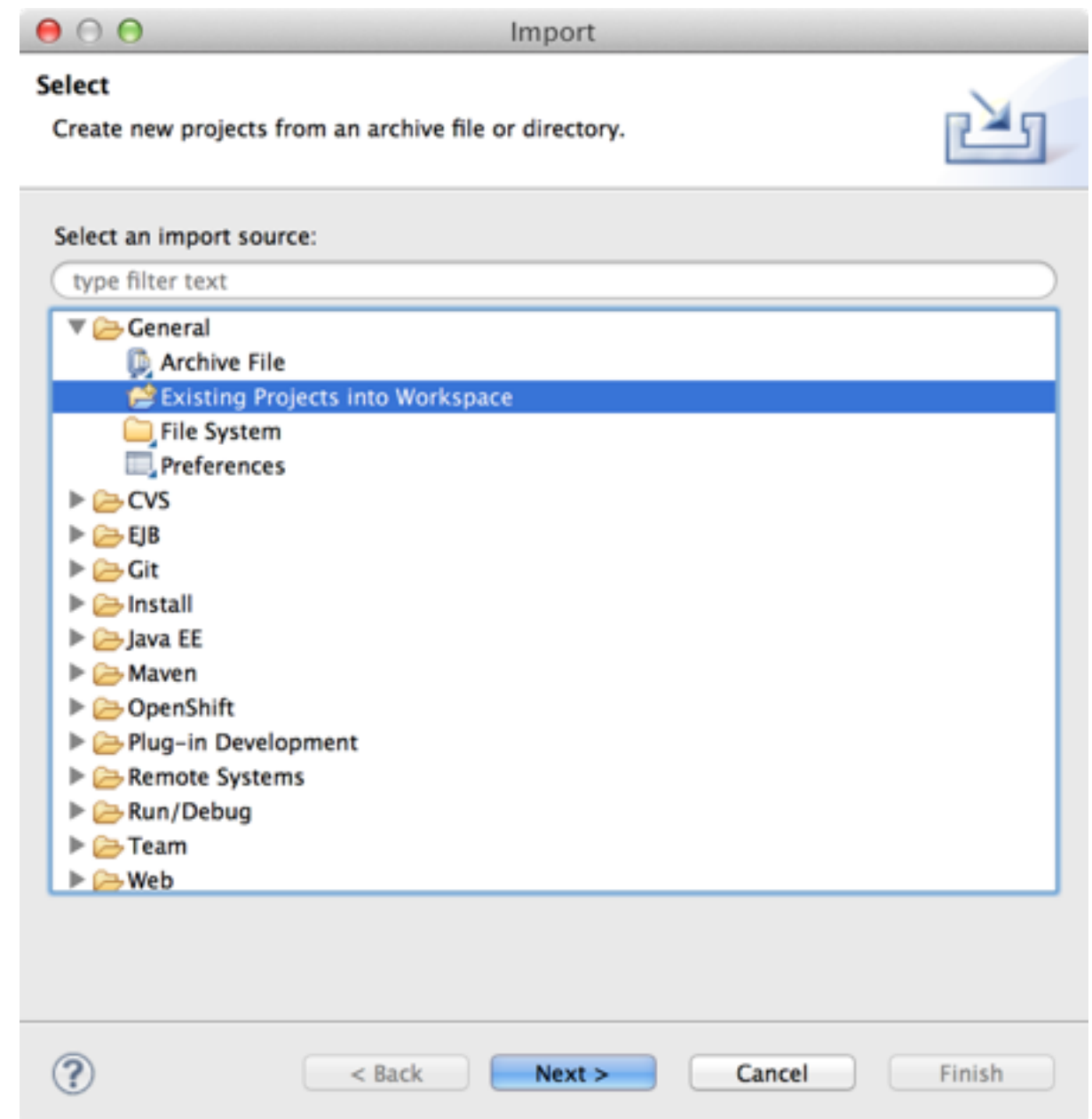
# Lab Goals

- Introduction to invm\_transport in SOA-P 5
- Introduction to invm\_transport in SOA-P 6
- Step-by-step migration using Windup rules
- Deploy and test application in SOA-P 6

# Importing SOA 5 InVM Transport

## TODO

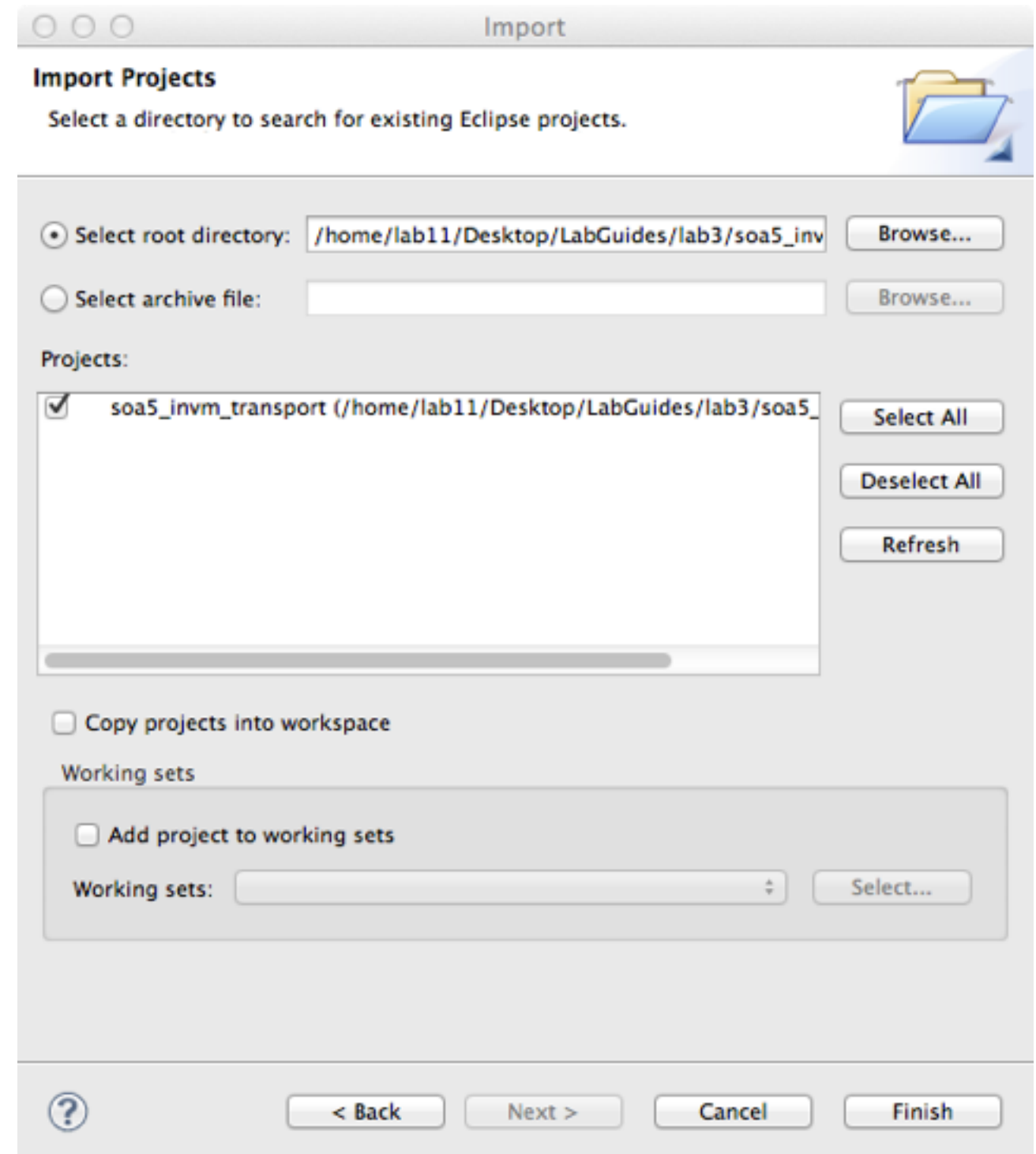
1. File -> Import ... from the JBDS menu.
2. Select General -> Existing Projects into Workspace
3. Click Next



# Importing SOA 5 InVM Transport

## TODO

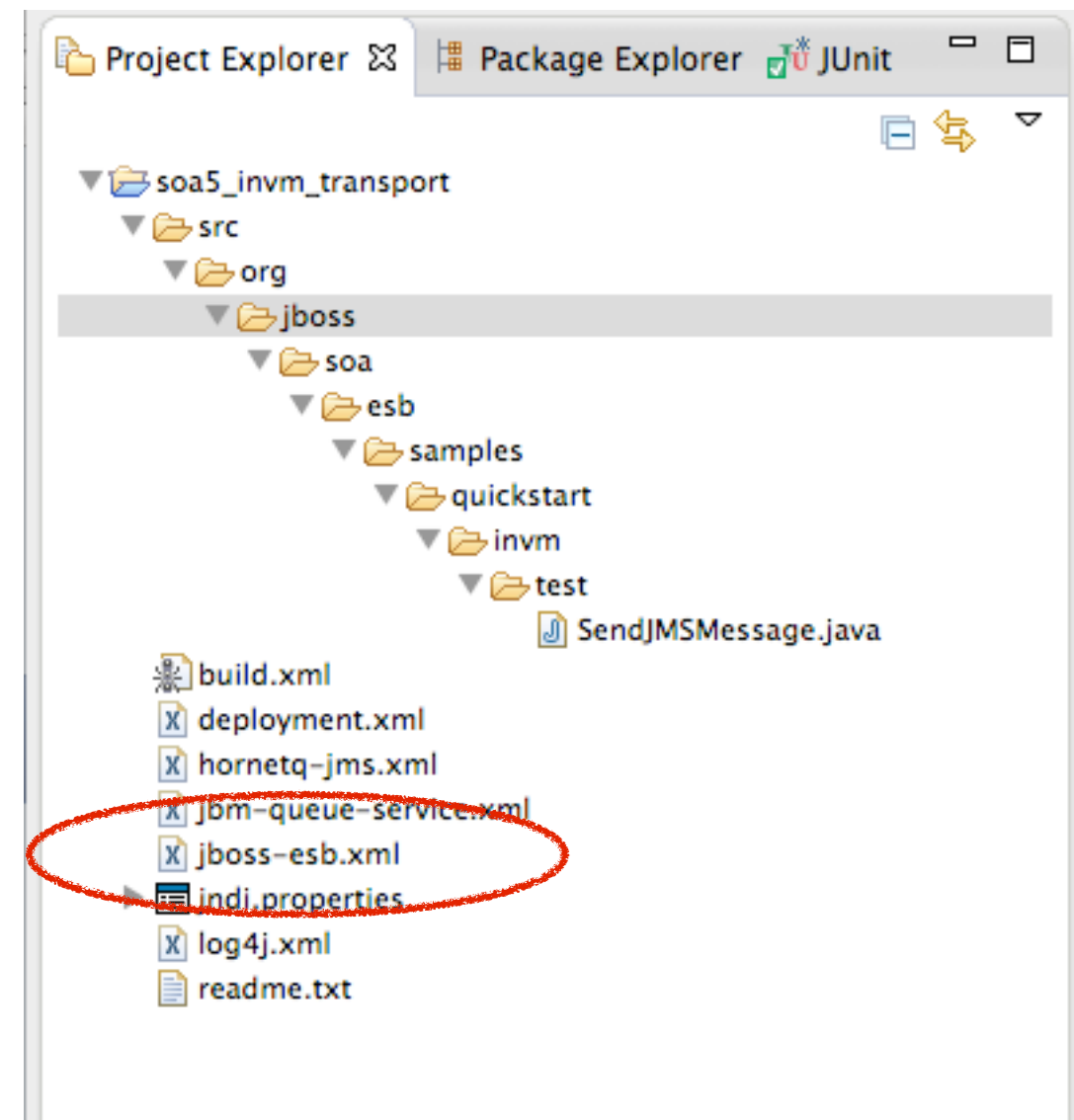
1. Click Browse ... and navigate to:  
`/home/lab13/Desktop/LabGuides/lab1/soa5_invm_transport`
2. Make sure the soa5\_invm\_transport project is checked
3. Click Finish



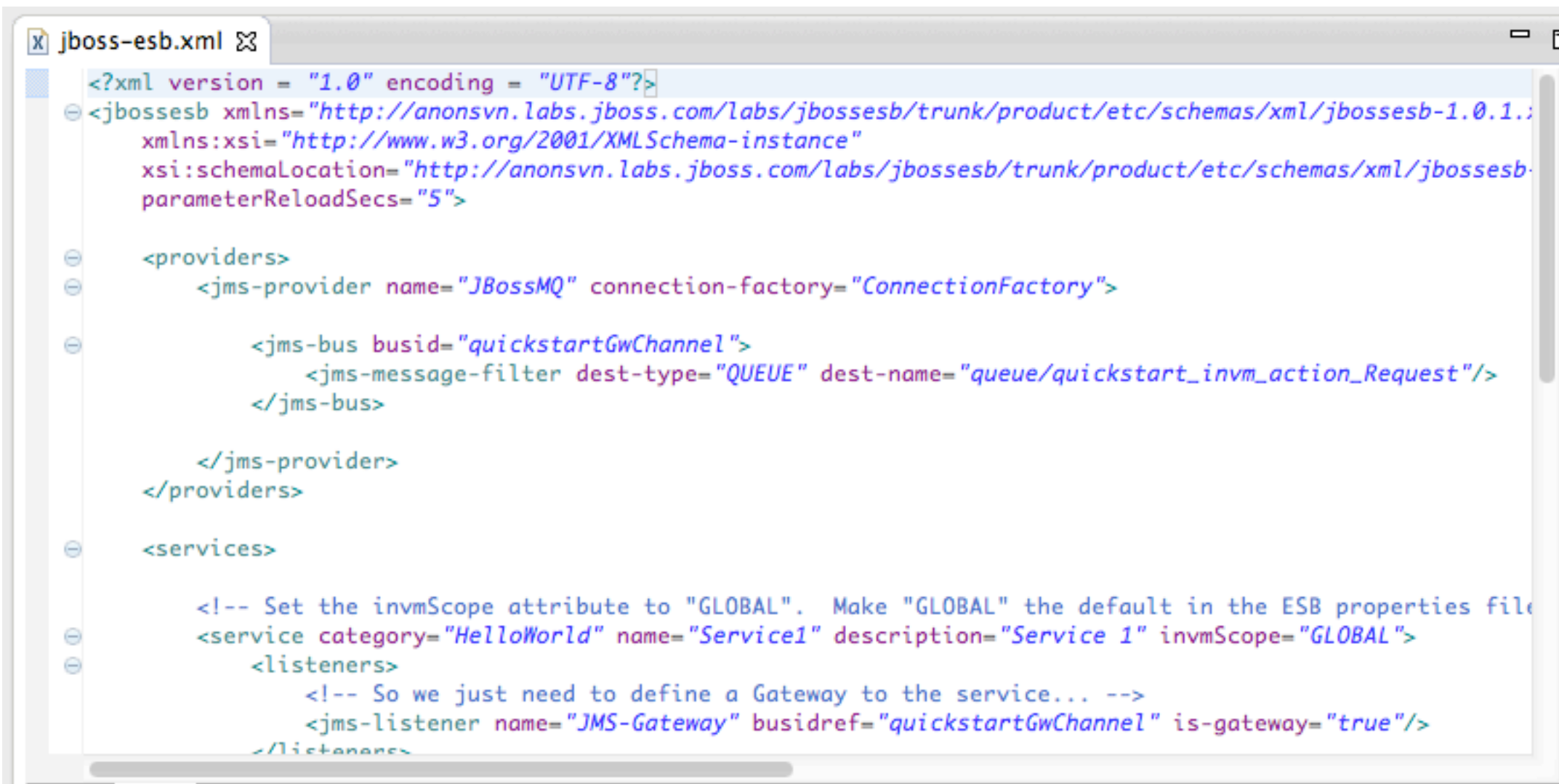
# Files To Note

## 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.



# jboss-esb.xml



```
<?xml version = "1.0" encoding = "UTF-8"?>
<jbossesb xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jbossesb-1.0.1.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jbossesb-1.0.1.xsd"
  parameterReloadSecs="5">

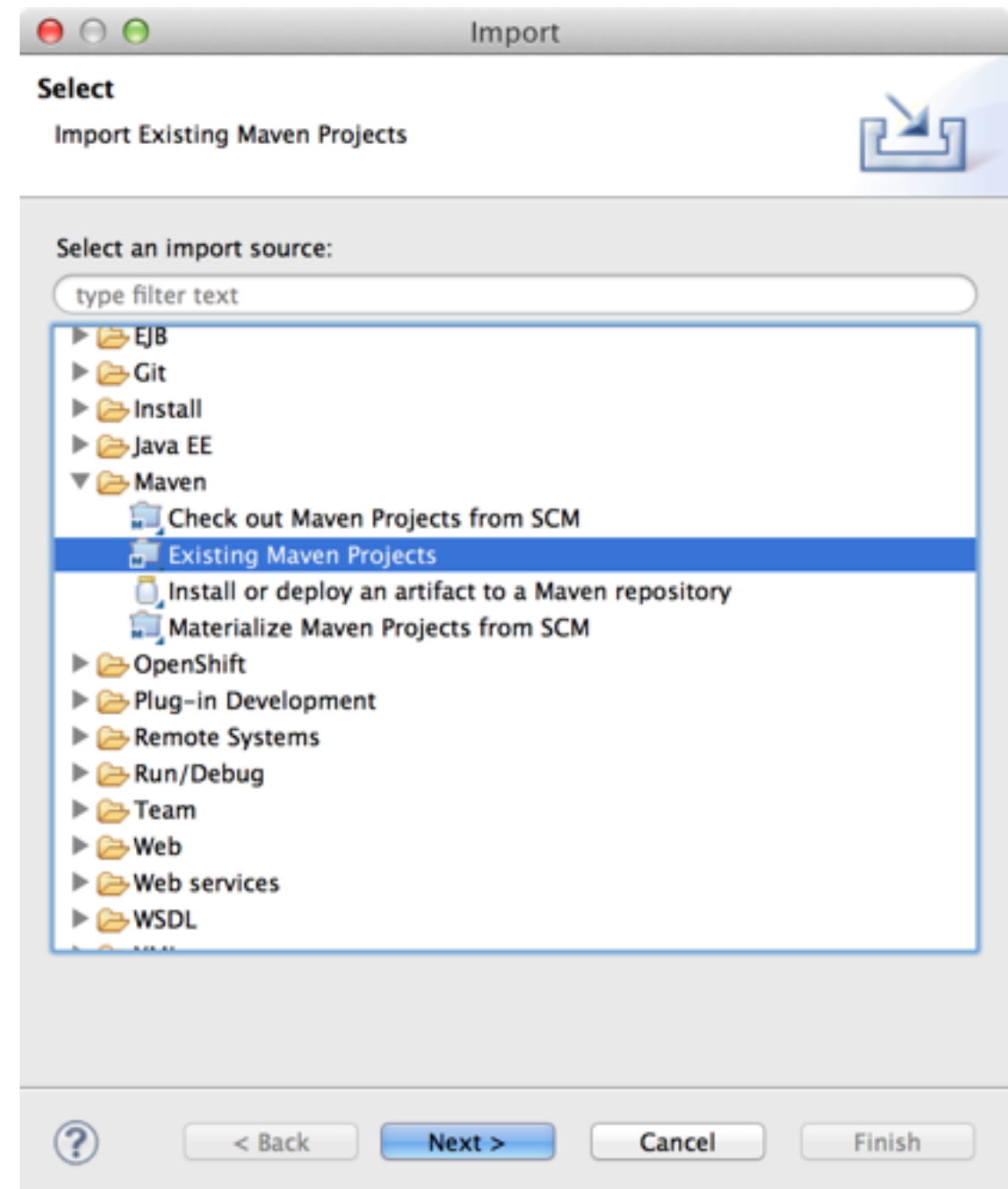
  <providers>
    <jms-provider name="JBossMQ" connection-factory="ConnectionFactory">
      <jms-bus busid="quickstartGwChannel">
        <jms-message-filter dest-type="QUEUE" dest-name="queue/quickstart_invm_action_Request"/>
      </jms-bus>
    </jms-provider>
  </providers>

  <services>
    <!-- Set the invmScope attribute to "GLOBAL". Make "GLOBAL" the default in the ESB properties file -->
    <service category="HelloWorld" name="Service1" description="Service 1" invmScope="GLOBAL">
      <listeners>
        <!-- So we just need to define a Gateway to the service... -->
        <jms-listener name="JMS-Gateway" busidref="quickstartGwChannel" is-gateway="true"/>
      </listeners>
    </service>
  </services>
</jbossesb>
```

# Importing SOA 6 InVM Transport

## TODO

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



# Importing SOA 6 Hello World

## TODO

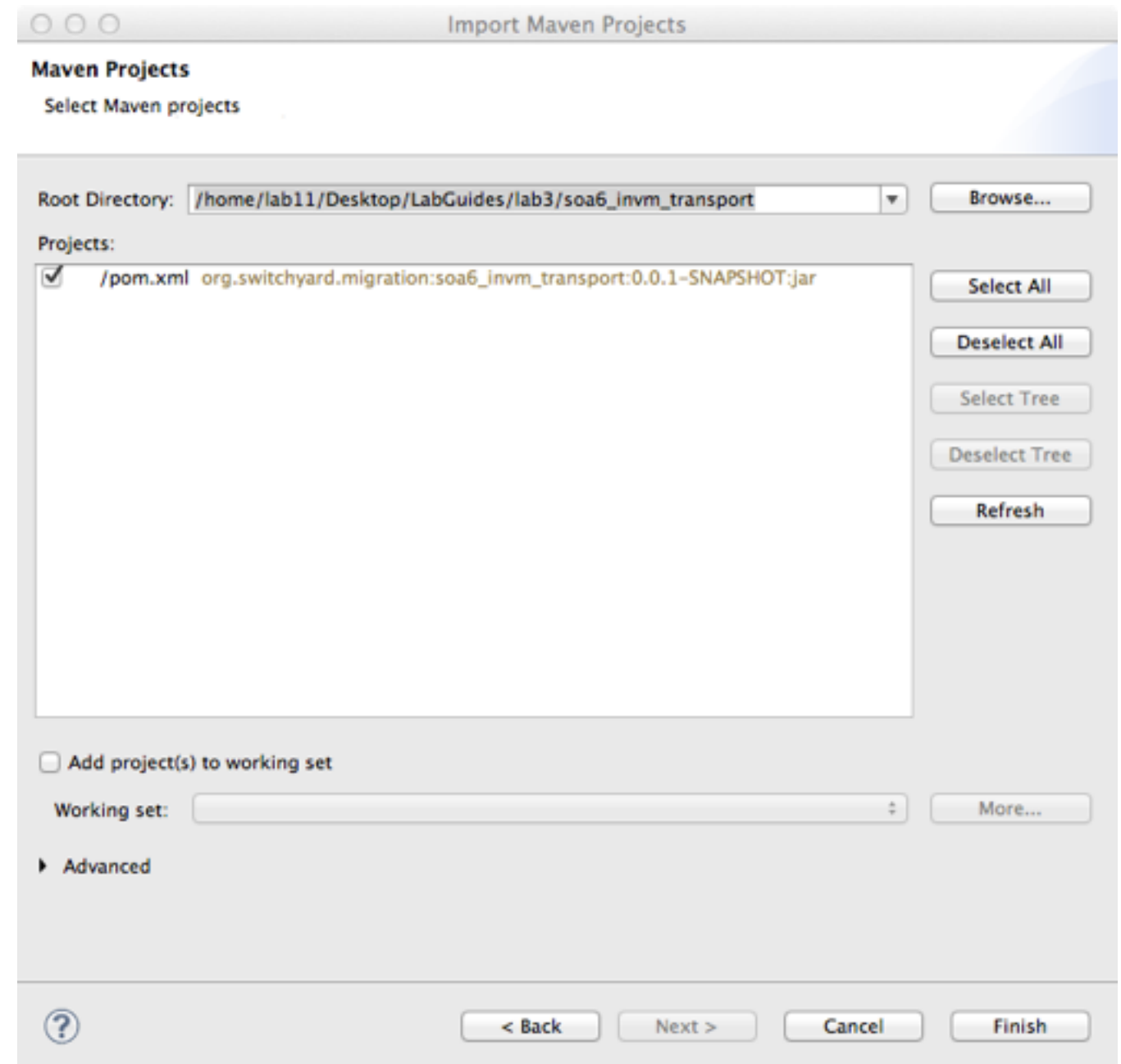
1. Click Browse ... and navigate to:

`/home/lab13/Desktop/LabGuides/lab1/  
soa6_invm_transport`

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

`org.switchyard.migration:soa6_invm_transport`

3. Click Finish






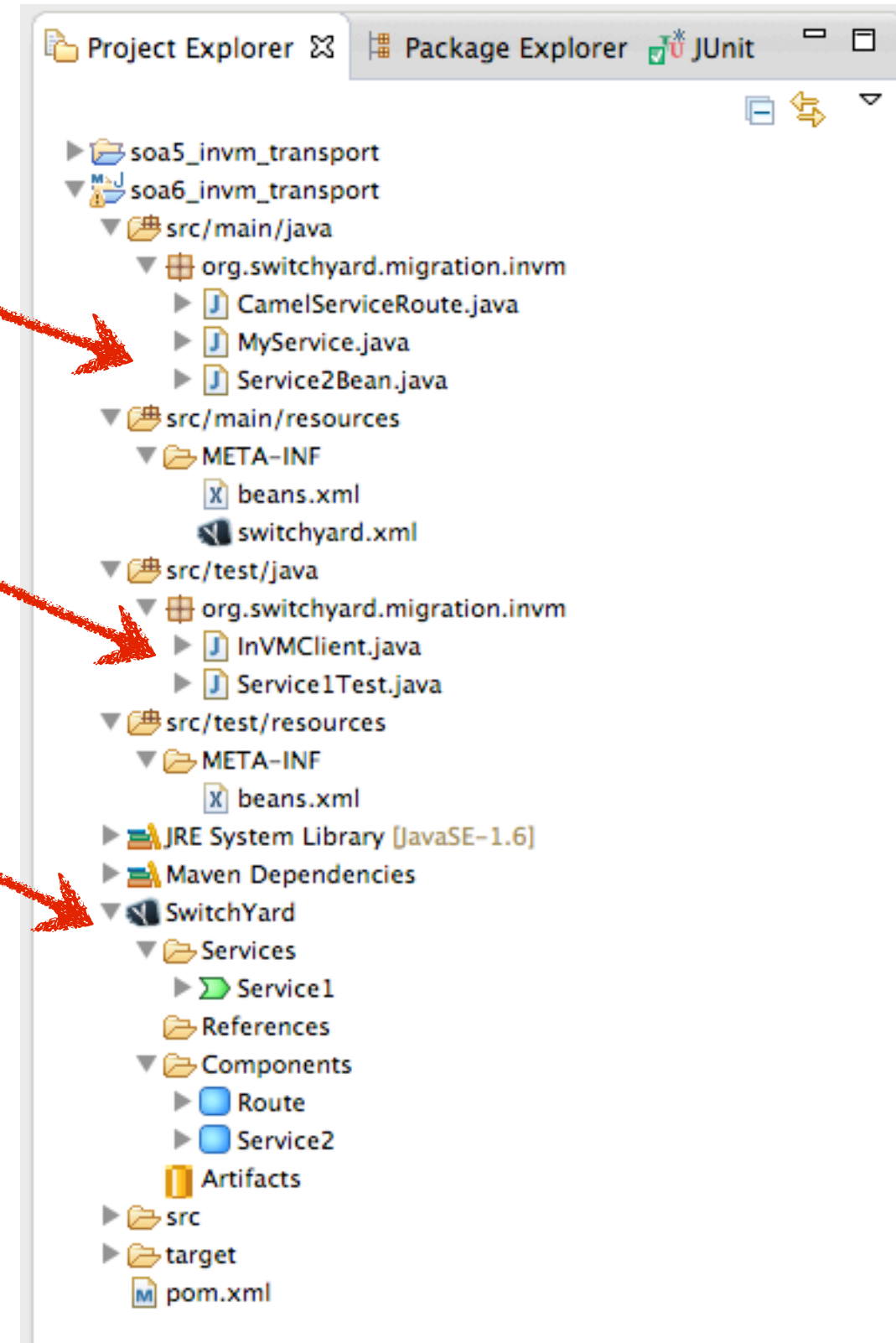
# Files To Note

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 `inv_m_transport`

- 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

## Creating a Bean Service

### Notification

- 
- ❗ [Action : service binding configuration in jms-bus: busid="quickstartGwChannel"](#)
  - ❗ [Action : composite service required for service: name="Service1"](#)
  - ❗ [Action : composite service binding required for listener: name="JMS-Gateway"](#)
  - ❗ [Action : create component service for action processing pipeline](#)
  - ❗ [Action : convert SystemPrintln: class="org.jboss.soa.esb.actions.SystemPrintln"](#)
  - ❗ [Action : convert StaticRouter to Camel routing: class="org.jboss.soa.esb.actions.StaticRouter"](#)
  - ❗ [Action : composite service required for service: name="Service2"](#)
  - ❗ [Action : create component service for action processing pipeline](#)
  - ❗ [Action : convert SystemPrintln: class="org.jboss.soa.esb.actions.SystemPrintln"](#)

# Service Contract

## FYI

*This is the service contract for Service1 in SOA 6. All services in SOA 6 have a contract.*

```
package org.switchyard.migration.invm;  
  
public interface MyService {  
    String process(String message);  
}
```

# Service Implementation

**FYI**

*This is the implementation of Service2 in SOA 6 using a CDI Bean Service.*

```
package org.switchyard.migration.invm;

import org.switchyard.component.bean.Service;

@Service(value = MyService.class, name = "Service2")
public class Service2Bean implements MyService {

    @Override
    public String process(String message) {
        return message + "\nHello from Service 2!";
    }

}
```

# 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="println" class="org.jboss.soa.esb.actions.SystemPrintln">
    <property name="message" value=" - > Service 1"/>
  </action>

  <!-- Route to the "Service 2" -->
  <action name="routeAction" class="org.jboss.soa.esb.actions.StaticRouter">
    <property name="destinations">
      <route-to service-category="HelloWorld" service-name="Service2"/>
    </property>
  </action>
</actions>
```

# Notifcation

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:

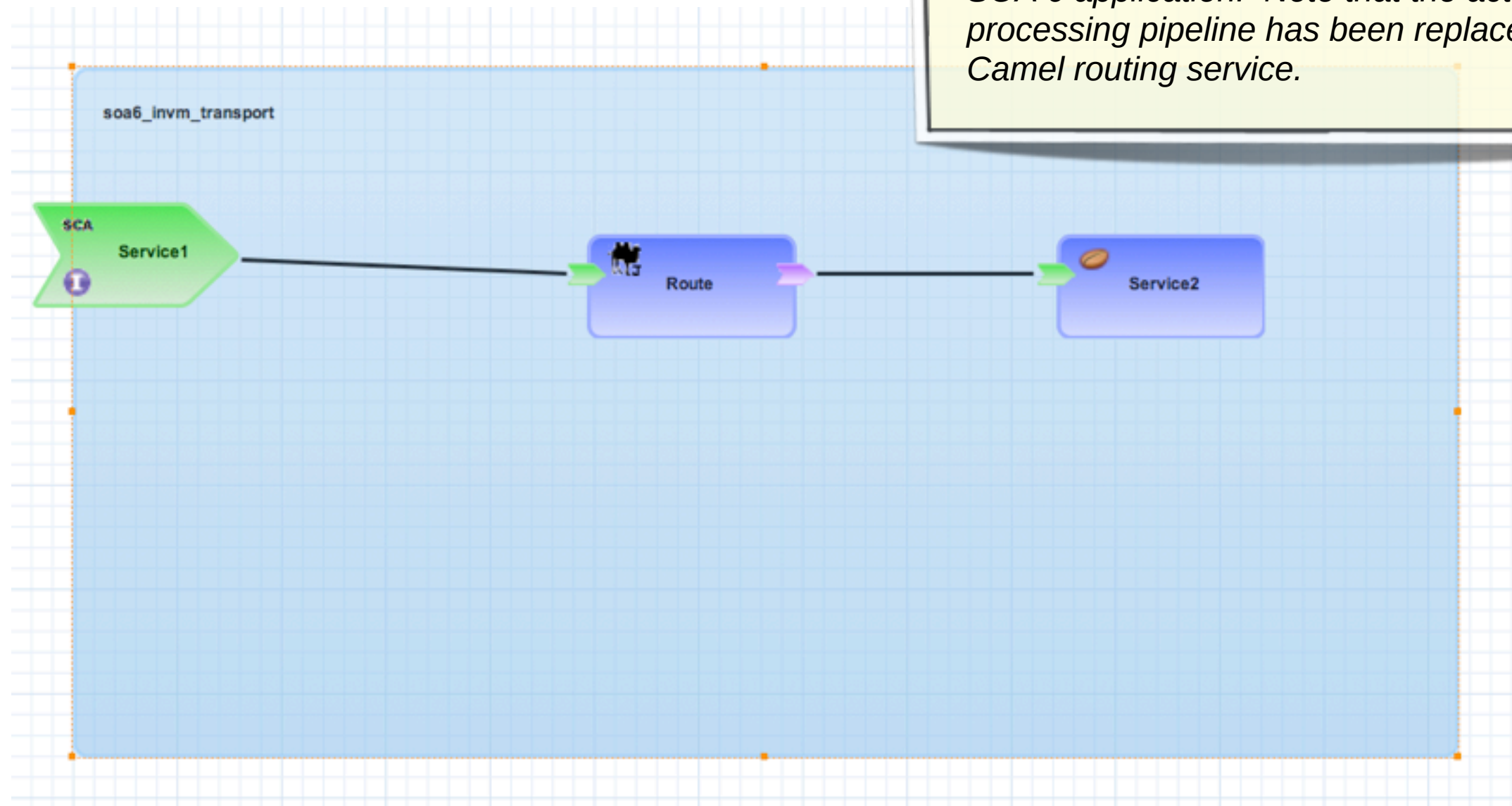
```
<actions mep="OneWay">
  <action name="action1" class="org.example.MyAction" process="doSomething" />
</actions>
```

<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 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 listener. 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.

file | 34 lines (26 sloc) | 1.309 kb

#### Overview

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

```
<services>
  <service
    category="Purchasing"
    name="OrderService"
    description="An Order Service">
  </service>
</services>
```

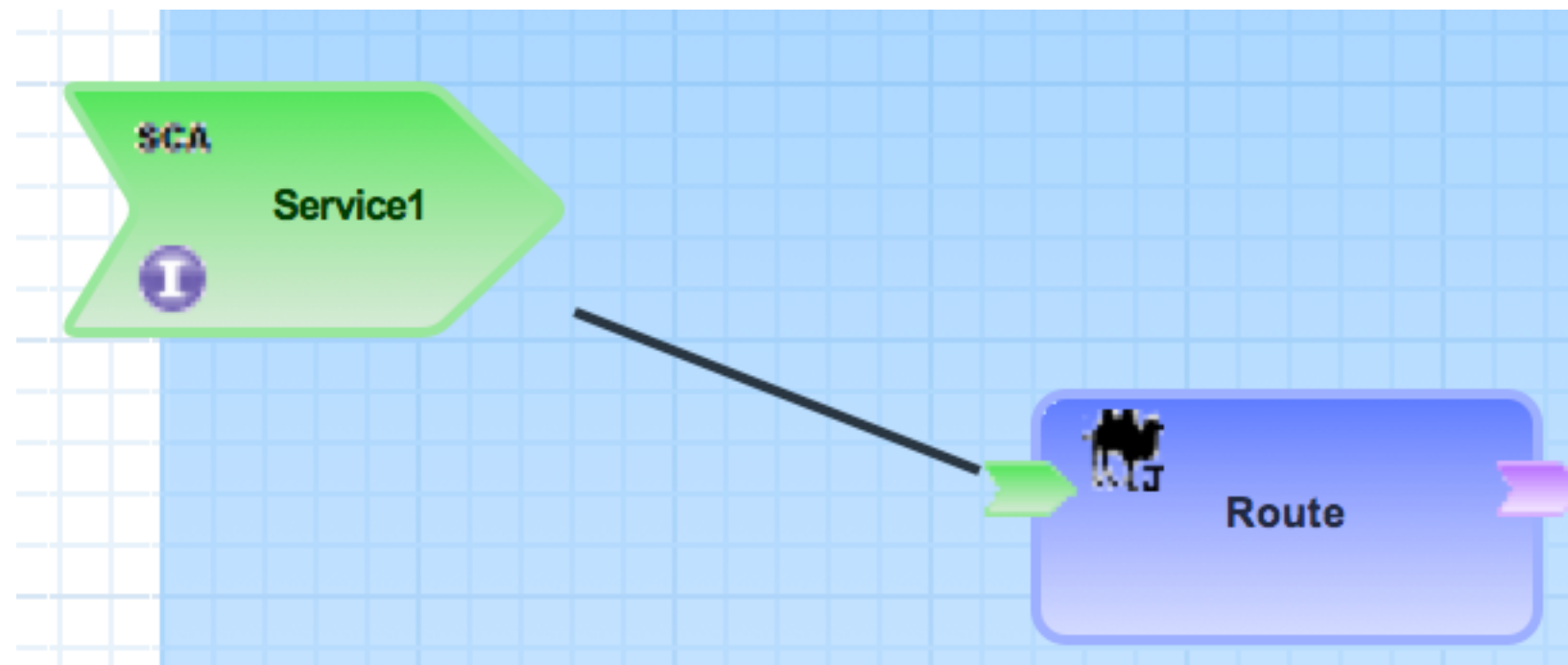
#### SOA 6 Configuration

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:



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

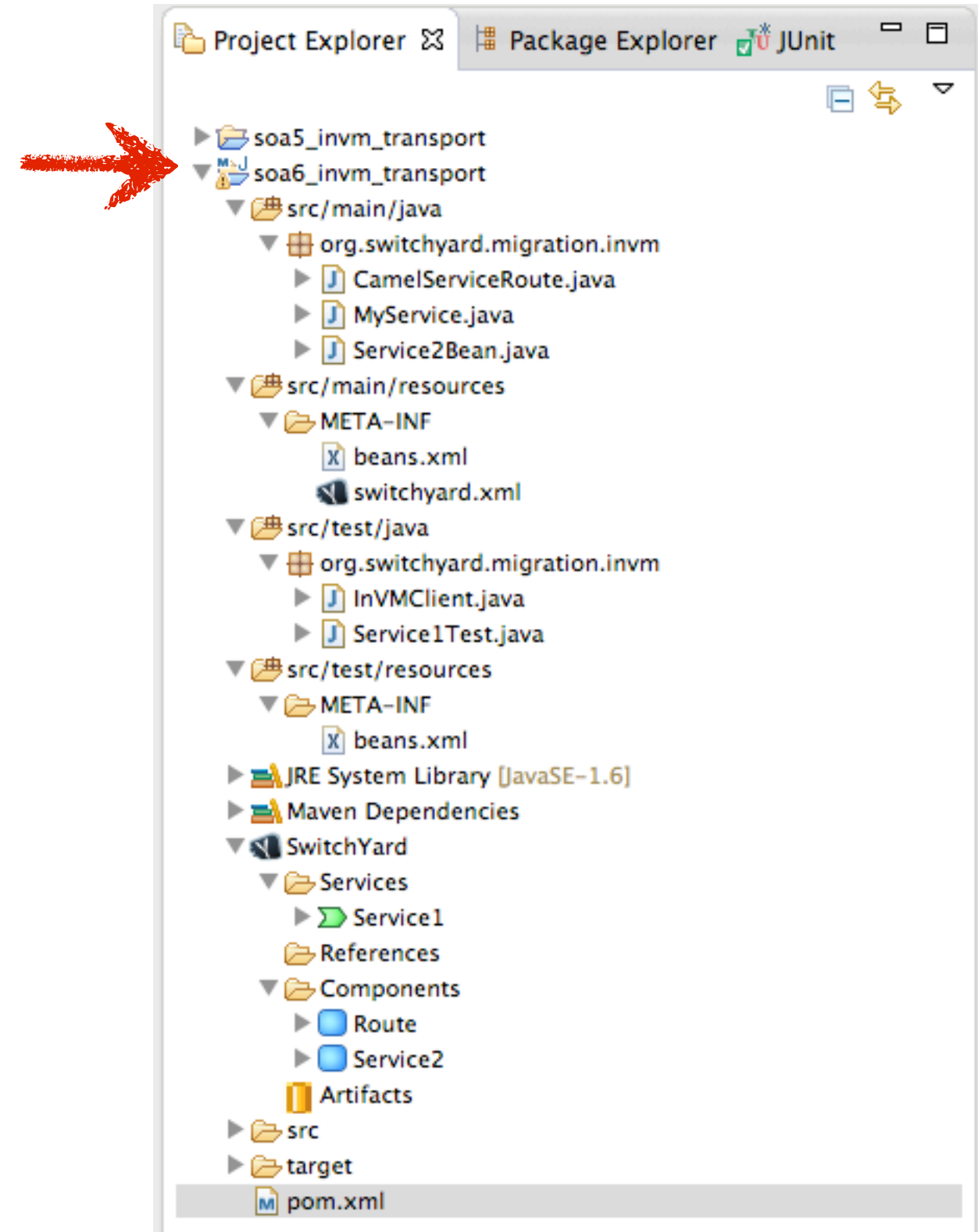
# Composite Service



# Deploy Application

## TODO

1. Right-click on the soa6\_invm\_transport project and select Run As ... Run on Server

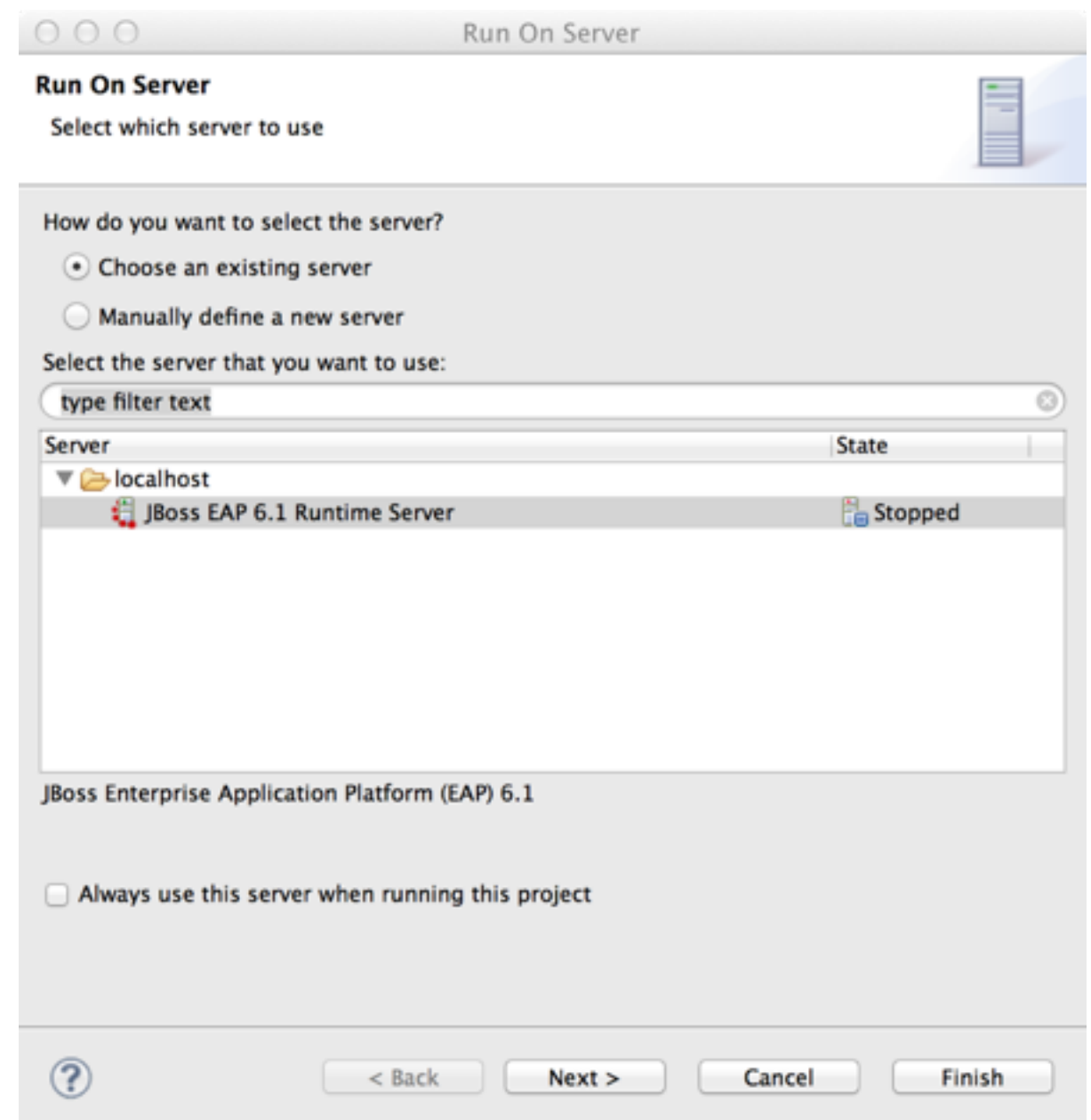




# Select Server

## TODO

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

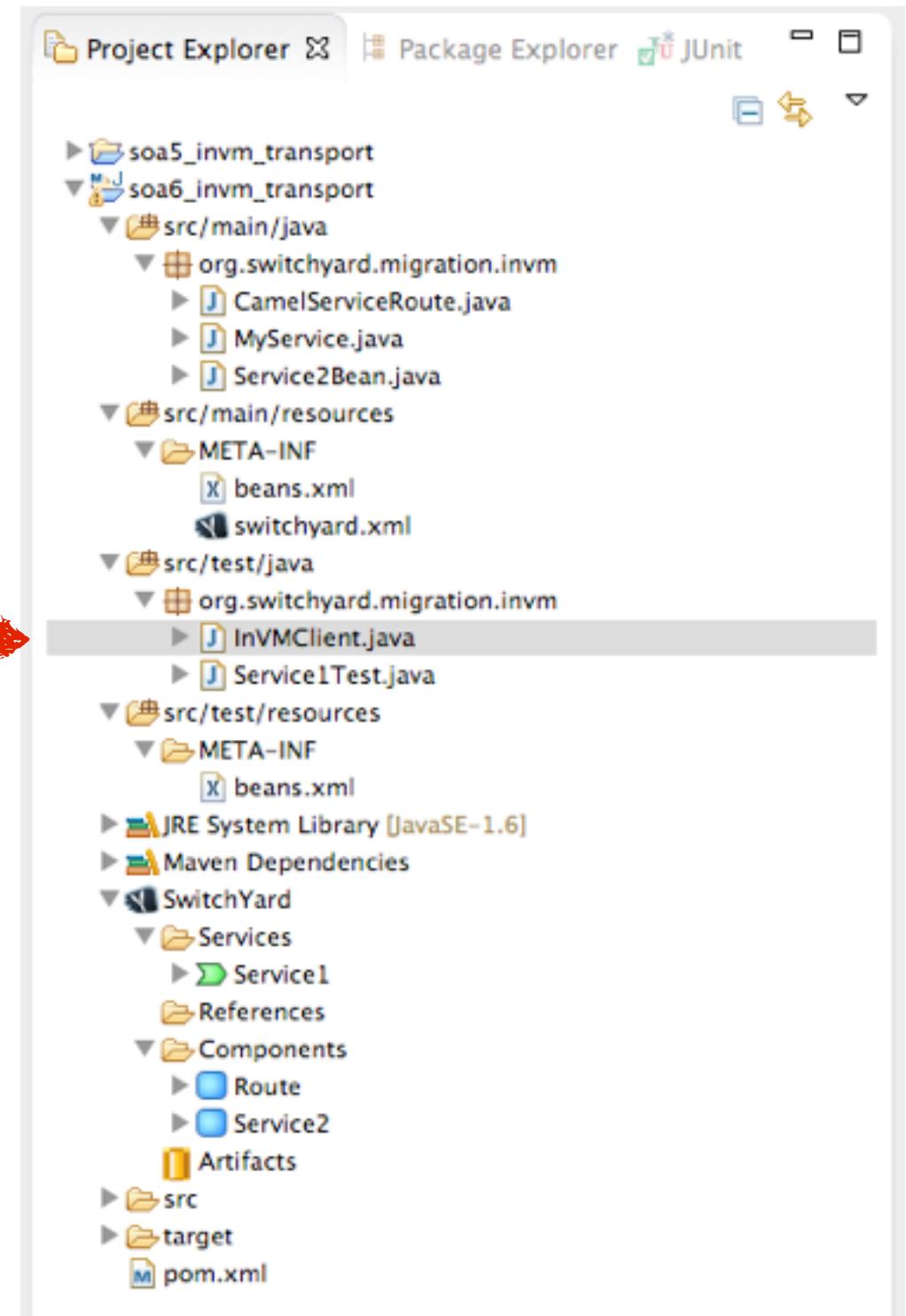




# Run Test Client

## TODO

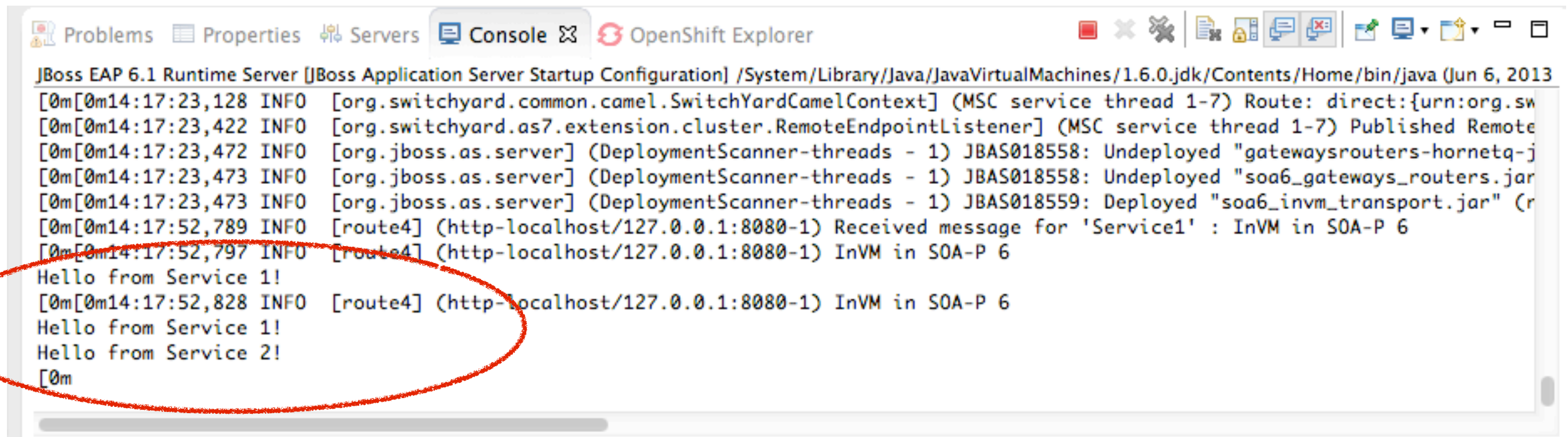
1. Open InVMClient.java from the Project Explorer view.
2. Go to the Run menu in the main menu bar and select 'Run As -> Java Application'



# 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 (Jun 6, 2013)
[0m[0m14:17:23,128 INFO [org.switchyard.common.camel.SwitchYardCamelContext] (MSC service thread 1-7) Route: direct:{urn:org.sw
[0m[0m14:17:23,422 INFO [org.switchyard.as7.extension.cluster.RemoteEndpointListener] (MSC service thread 1-7) Published Remote
[0m[0m14:17:23,472 INFO [org.jboss.as.server] (DeploymentScanner-threads - 1) JBAS018558: Undeployed "gatewaysrouters-hornetq-j
[0m[0m14:17:23,473 INFO [org.jboss.as.server] (DeploymentScanner-threads - 1) JBAS018558: Undeployed "soa6_gateways_routers.jar
[0m[0m14:17:23,473 INFO [org.jboss.as.server] (DeploymentScanner-threads - 1) JBAS018559: Deployed "soa6_invm_transport.jar" (r
[0m[0m14:17:52,789 INFO [route4] (http-localhost/127.0.0.1:8080-1) Received message for 'Service1' : InVM in SOA-P 6
[0m[0m14:17:52,797 INFO [route4] (http-localhost/127.0.0.1:8080-1) InVM in SOA-P 6
Hello from Service 1!
[0m[0m14:17:52,828 INFO [route4] (http-localhost/127.0.0.1:8080-1) InVM in SOA-P 6
Hello from Service 1!
Hello from Service 2!
[0m
```

# Verify Output

## TODO

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

A screenshot of an IDE console window. The window has a title bar with tabs for 'Problems', 'Properties', 'Servers', 'Console', and 'OpenShift Explorer'. The 'Console' tab is active. The console output shows a terminated Java application with the following text: 

```
<terminated> InVMClient [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 6, 2013 2:17:51 PM)
=====
InVM in SOA-P 6
Hello from Service 1!
Hello from Service 2!
=====
```

 A red oval is drawn around the application output text.

# Lab 3 Complete!