

## Lab Exercise

### Routers - Lab 5

Routers direct messages to an appropriate message channel typically based on what is in the payload or header of a message. Routers do not alter the message like a transformer. Routers don't remove messages from the system like a filter can. They simply provide forks in the road of message channels in a Spring Integration (SI) application. In this lab, you explore a few commonly used SI routers.

#### Specifically, in this lab you will:

- Configure and use an XPath router to route XML payload messages.
- Define a recipient router to send the same message to a collection of channels.



***Lab solution folder: ExpressSpringIntegration\lab5\lab5-router-solution***

## Scenario – Route using XPath expression

In Lab 3, you used XPath expressions in filters to remove or discard some XML messages from the system. In a similar fashion, as will be demonstrated in this part of the lab, an XPath router uses an XPath expression to determine which message channel receives a message containing XML in its payload.

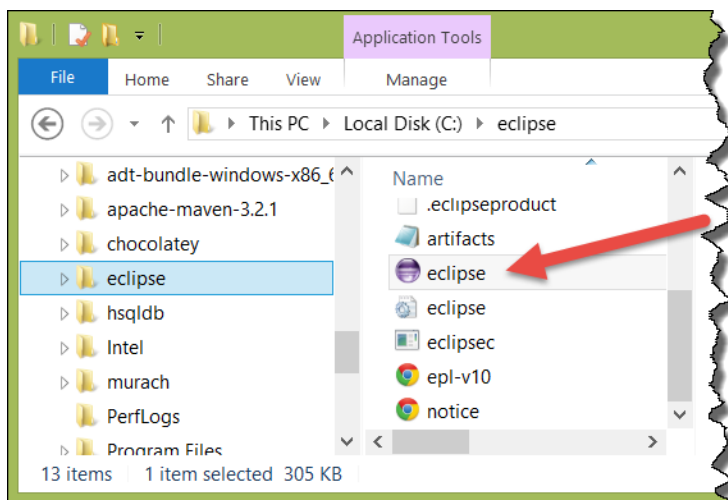
### Step 1: Import the Maven Project

A Maven Project containing the base code for an XPath transformer application has already been created for you. You will use this project to begin your exploration of routers.

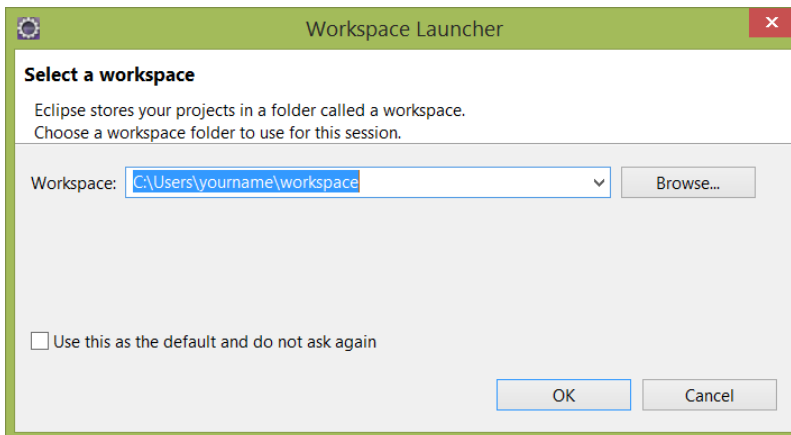
**1.1** Start the Eclipse-based IDE. Locate and start Eclipse (or Eclipse-based) IDE.

**1.1.1** Locate the Eclipse folder.

**1.1.2** Start Eclipse by double clicking on the eclipse.exe icon (as highlighted in the image below).



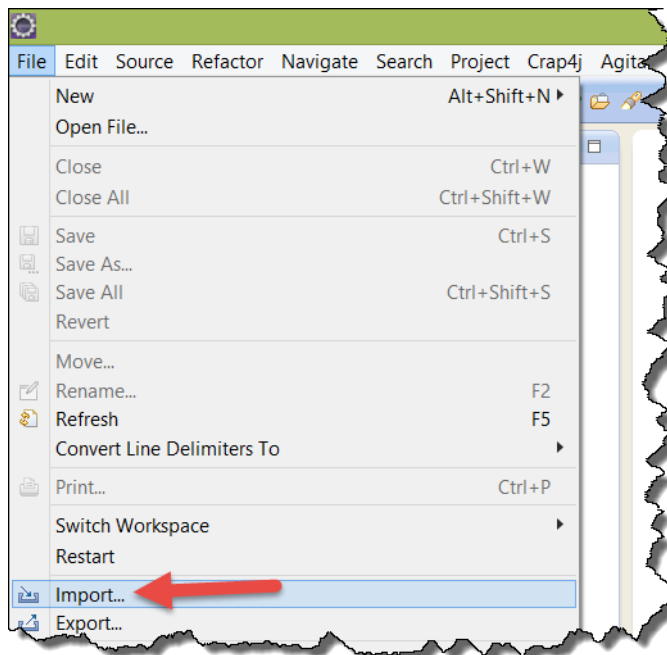
**1.1.3 Open your workspace. Type in `C:\Users\<your username >\workspace` and click the **OK** button.**



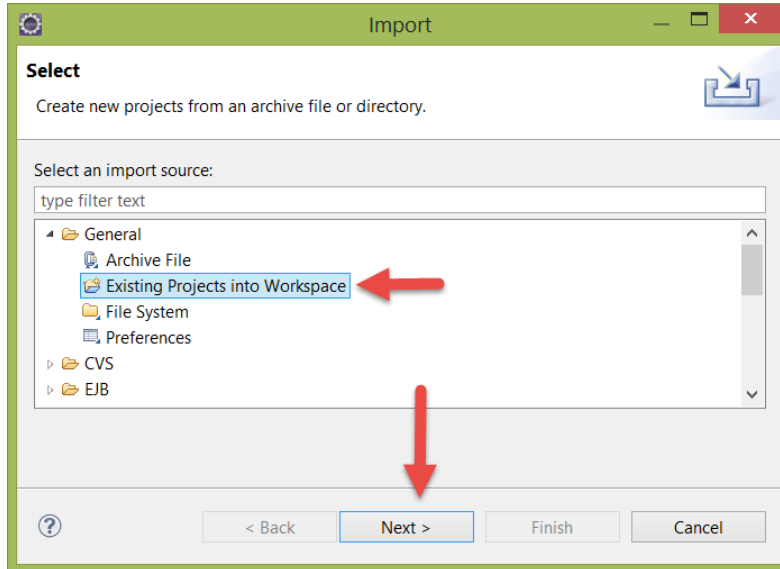
Note: As noted in the past labs, you may use an alternate location for your workspace, but the labs will always reference this location (`c:\users\[your username]\workspace`) as the default workspace location.

## 1.2 Import the new Maven Project.

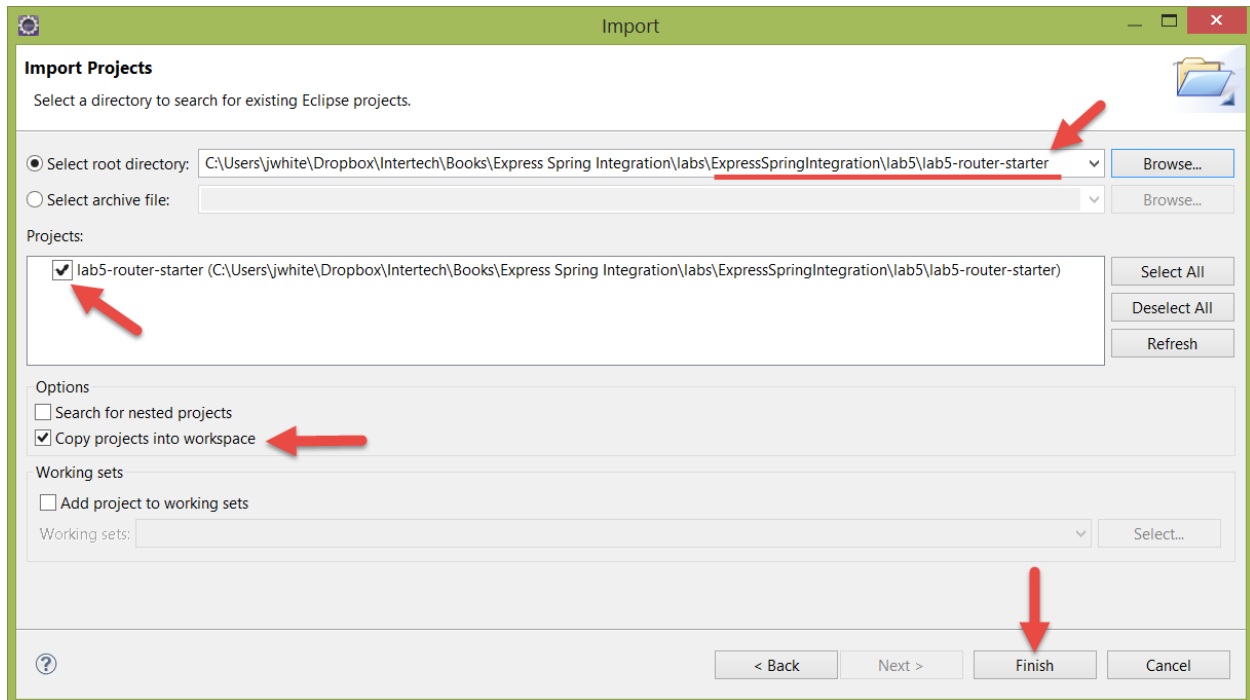
**1.2.1 Select `File>Import...` from the Eclipse menu bar.**



**1.2.2** Locate and open the General folder in the Import window, and then select *Existing Projects into Workspace* from the options. Now push the *Next>* button.

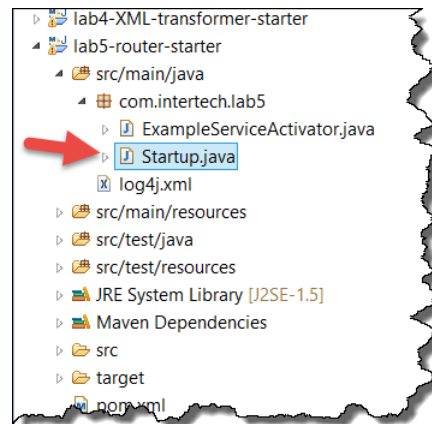


**1.2.3** In the “Import” window, use the *Browse...* button to locate the lab5-router-starter project folder located in ExpressSpringIntegration\lab5 (this folder is located in the lab downloads). Make sure the project is selected, and the “Copy projects into workspace” checkbox is also checked before you hit the *Finish* button (as shown below).

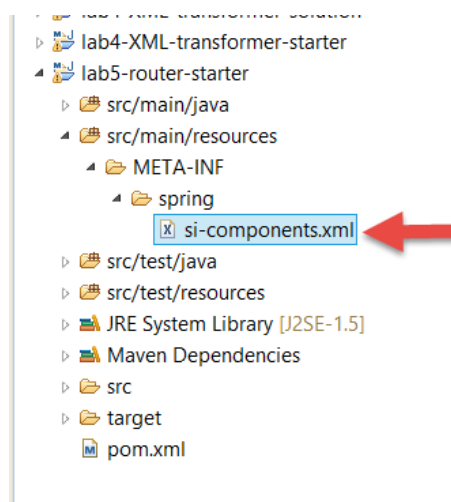


**1.3** Explore the project. Examine the project for the Spring Integration components that are already present.

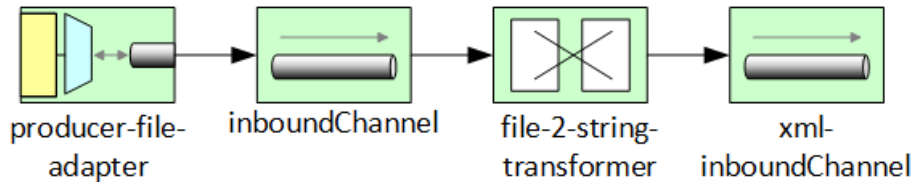
**1.3.1 Examine the Startup.java.** Expand the `src/main/java` folder in the Project Explorer view, and open the `Startup.java` file by double clicking on it. As in past labs, the `Startup.java` class is used to put the application in an infinite loop so that the SI components can do their work.



**1.3.2 Examine the SI configuration.** Expand the `src/main/resources/META-INF/spring` folder in the Project Explorer view, and open `si-components.xml` file by double clicking on it. The `si-components.xml` file contains the Spring configuration that includes the definitions for several SI components already.



**1.3.3** In particular, note that the application already contains an inbound file adapter and a file-to-string transformer to take XML files and get them to String form. Below is an EIP model representing what is currently defined in the si-components.xml file. The adapter gets XML files to the inboundChannel and then the transformer converts the file messages to string messages for the xml-inboundChannel.




At this point, there are no outbound adapters.

**Step 2: Create an XPath router**

The ship order XML messages will again be used in the last to experiment with routers. The XML ship order messages contain a <country> element nested in a <shipto> element which in turn is nested in the root <shiporder> element. In this step, you build an XPath router to direct those XML messages containing “Norway” as the <country> elements value to a norwayChannel and those containing “USA” to a usaChannel.

**2.1** Add two messages channels – one for USA ship orders and one for Norway ship orders.

```
<int:channel id="norwayFileChannel" />
<int:channel id="norwaySChannel" />
```

 Note: if you get stuck or feel like not typing in all the code yourself, you will find a working copy of the final si-component.xml file at ExpressSpringIntegration\lab5\lab5-router-solution.

**2.2** Add two outbound file adapters. Add two file adapters; one for USA ship orders and one for Norway ship orders. Each file adapter will place the messages it receives from the appropriate message channel in separate file system directories so that the routing can be seen to work.

**2.2.1** Add a Norway outbound file adapter, as shown below, to place its messages in a directory called `outboundNorway`.

```
<int-file:outbound-channel-adapter channel="norwayChannel"
  id="consumer-file-adapter1" directory="file:c://outboundNorway" />
```

**2.2.2** Add a USA outbound file adapter to place its messages in a directory called `outboundUSA`.

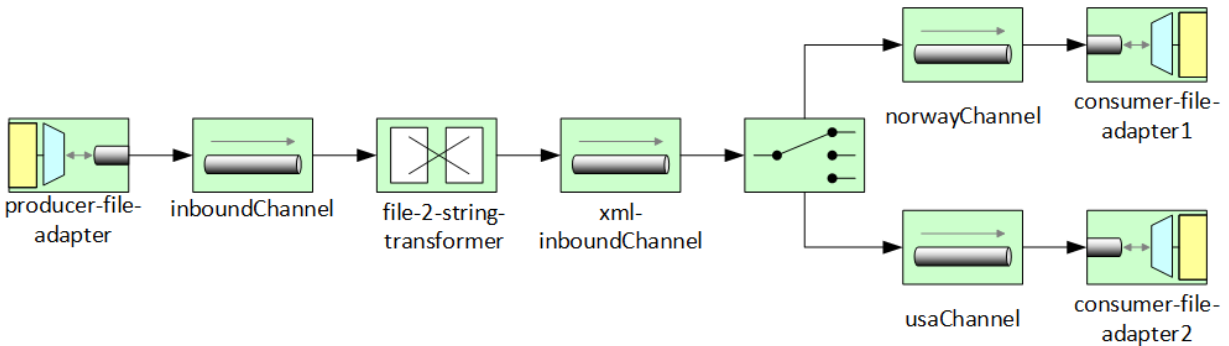
```
<int-file:outbound-channel-adapter channel="usaChannel"
  id="consumer-file-adapter2" directory="file:c://outboundUsa" />
```

**2.3** Add an XPath message router. The message router should use an XPath expression to get the value in the `<shiporder><shipto><country>` element and map it to the appropriate message channel as defined in step 2.1 above.

```
<int-xml:xpath-router id="orderTypeRouter"
  input-channel="xml-inboundChannel">
  <int-xml:xpath-expression expression="/shiporder/shipto/country" />
  <int-xml:mapping value="Norway" channel="norwayChannel" />
  <int-xml:mapping value="USA" channel="usaChannel" />
</int-xml:xpath-router>
```

Note the use of mapping elements in the router to map the value discovered in the XPath expression to one of two message channels.

With the addition of this router component, you have provided a “fork” that distributes the inbound XML messages to the appropriate directory based on the content of the message.



The XPath router is a type of content router as it uses some part of the content of the message – in this case an element from the XML – to route the message through its intended path in the system.

**2.3.1 Save the configuration file and make sure there are no errors in the file.**

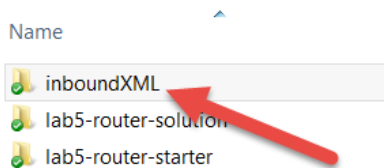
### ***Step 3: Test the Router***

#### **3.1 Add the files to the inbound message folder.**

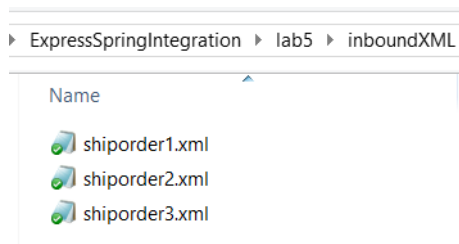
**3.1.1 In the si-components.xml, find the producer-file-adapter. Note the location of the directory. It is set, by default, to file:c://inboundXML. This is the location where the XML ship order messages will be taken into the application by the adapter. It is the same folder used in the last lab (lab 4). If it does not already exist, create this message folder - changing the location to suit your needs and your file system (change the producer-file-adapter to reflect your location).**



**3.1.2** Again, some sample XML shiporder messages have been provided to you. Find them in an inboundXML folder in ExpressSpringIntegration\lab5.



If you open the 3 messages in the folder, you will note that each is a shipment order with a country element. Two contain the value of “Norway” for the country while one has “USA” as the country. Copy the messages from this folder to the c:\\inboundXML folder (or whatever folder you created per 3.1.1. above).



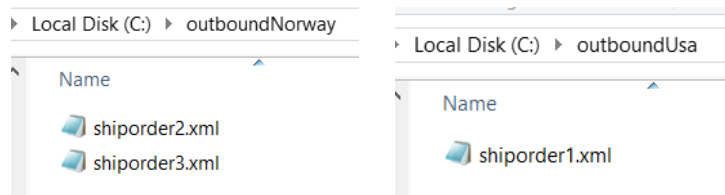
**3.2** Test the application. Test the application to see the Files in the inbound file folder are routed to the correct outbound folder based on the <country> element value.

**3.2.1** Locate the Startup.java file in the source folder. Right click on file and select Run As > Java Application from the resulting menu. Nothing should display in the Console view.



Note: depending on the speed of your system, you may have to give the application a couple of seconds to complete its work.

**3.2.2** Using a Windows Explorer, open the two folders specified as the output adapter’s directories (these should be c://outboundNorway and c://outboundUsa per configuration in step 2.2 above). See that the messages have been routed to the correct folder based on the <country> element in the message.



**3.3** Stop the application. Recall the application is running in an infinite loop to allow for the constant publishing and consuming of messages. Stop the application now.

**3.3.1** In the Console view, click on the red square to terminate the Startup application.

**3.3.2** The Console view should now indicate that the application is “<terminated>”.

#### **Step 4: Create a recipient list router**

Recipient list routers are meant to disburse received messages to all of its outbound channels without regard for message header or payload content. If you will, a recipient list router simply blasts incoming messages to all the channels listed as “recipients.”

In this step, you create a recipient router to route Norway ship orders to both a file adapter and a service activator that prints out information regarding the Norway ship order to the Console view.

**4.1** Add two new Norway message channels.

**4.1.1** There needs to be two new message channels to serve as recipients of the Norway messages. The first new channel will take messages that will ultimately be routed to the file system. The second new channel will receive messages that will ultimately be routed to the service activator.

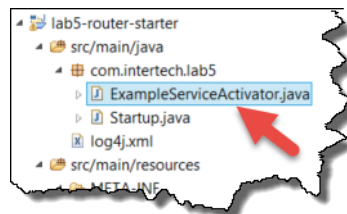
```
<int:channel id="norwayFileChannel" />
<int:channel id="norwaySChannel" />
```

**4.1.2** Change the File adapter for the Norway messages to now get its messages from the `norwayFileChannel` and not the `norwayChannel`.

```
<int-file:outbound-channel-adapter channel="norwayFileChannel"
  id="consumer-file-adapter1" directory="file:c://outboundNorway" />
```

**4.2** Add a service activator to display the Norway ship orders to the Console view.

**4.2.1** The service activator class (`ExampleServiceActivator` has already been created for you and is in the `com.intertech.lab5` package). Feel free to open and explore this class if you would like.



**4.2.2 Add the service activator SI component and associated bean in the si-components.xml file. Take the message from the norwaySChannel.**

```
<int:service-activator id="printing-service-activator"
  input-channel="norwaySChannel" ref="serviceActivator" />
<bean id="serviceActivator"
class="com.intertech.lab5.ExampleServiceActivator" />
```

**4.3 Add a recipient list router to distribute a copy of the Norway ship order messages from the norwayChannel to both the norwayFileChannel and norwaySChannel.**

```
<int:recipient-list-router input-channel="norwayChannel">
  <int:recipient channel="norwayFileChannel" />
  <int:recipient channel="norwaySChannel" />
</int:recipient-list-router>
```

Note how in a recipient list router, the nested <recipient> elements (versus <mapping> elements in the content routers) specified the receiving message channels and there is no expression or other router attribute to inspect the content of the message to determine which is the receiving channel. All recipients get a copy of the inbound message.

**4.4 Save the configuration file and make sure there are no errors in the file.**

**4.5 Retest the application (and see the recipient routing).**

**4.5.1 Use a Windows Explorer to remove the messages from both the outboundNorway and outboundUsa folders so that you will be able to see new messages arrive in these folders.**

**4.5.2 Again locate the Startup.java file in the source folder. Right click on file and select Run As > Java Application from the resulting menu.**



Note: you will probably have to give the application a few seconds (count to 10) to complete its work.

**4.5.3 Again using a Windows Explorer, open the two folders specified as the output adapter's directories (c://outboundNorway and c://outboundUsa). See that the messages have again been routed to the correct folder based on the <country> element in the message.**


**4.5.4 This time, the recipient router should have also caused the Norway ship order messages to be routed to the service activator that displays the message contents to the Console view (via System.out.println).**

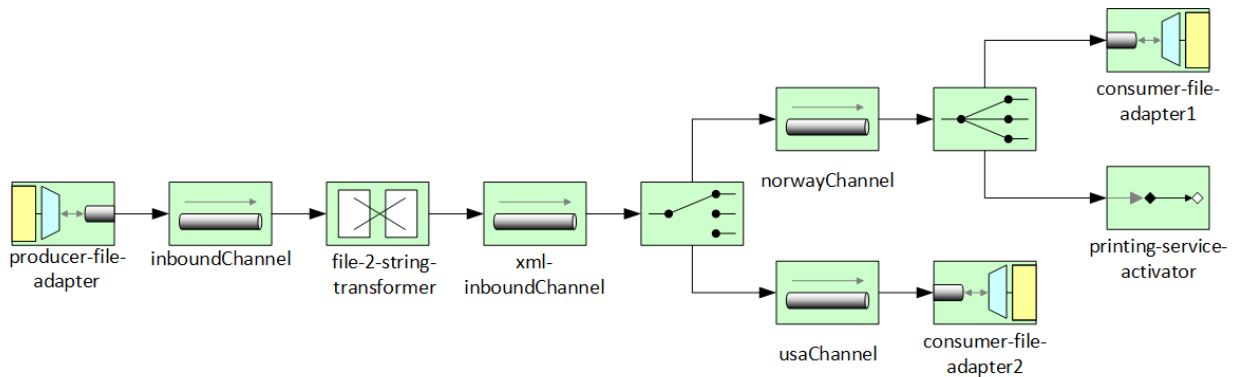
```

Startup (6) [Java Application] C:\Program Files\Java\jdk1.7.0_55\bin\javaw.exe (Aug 17, 2014, 8:00:17 PM)
<?xml version="1.0" encoding="UTF-8"?>
<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
<?xml version="1.0" encoding="UTF-8"?>
<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
  </item>
</shiporder>
  
```

**4.5.5 Terminate the application and clear Console view.**

**4.5.6 Below is the EIP model for your completed routing application.**

 Note: the Hohpe/Wolfe icon for a recipient router is a little different than the icon for a content router.



## Lab Solution

---

### si-components.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:int="http://www.springframework.org/schema/integration"
  xmlns:int-
file="http://www.springframework.org/schema/integration/file"
  xmlns:int-
mail="http://www.springframework.org/schema/integration/mail"
  xmlns:int-
xml="http://www.springframework.org/schema/integration/xml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:int-
stream="http://www.springframework.org/schema/integration/stream"
  xsi:schemaLocation="
  http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
  http://www.springframework.org/schema/integration
http://www.springframework.org/schema/integration/spring-
integration.xsd
  http://www.springframework.org/schema/integration/stream
http://www.springframework.org/schema/integration/stream/spring-
integration-stream.xsd
  http://www.springframework.org/schema/integration/file
http://www.springframework.org/schema/integration/file/spring-
integration-file.xsd
  http://www.springframework.org/schema/integration/xml
http://www.springframework.org/schema/integration/xml/spring-
integration-xml.xsd">

  <!-- Adapter for reading files -->

  <int-file:inbound-channel-adapter id="producer-file-adapter"
    channel="inboundChannel" directory="file:c://inboundXML"
    prevent-duplicates="true">
    <int:poller fixed-rate="5000" />
  </int-file:inbound-channel-adapter>

  <int:channel id="inboundChannel" />

  <int-file:file-to-string-transformer
    id="file-2-string-transformer" input-channel="inboundChannel"
    output-channel="xml-inboundChannel" charset="UTF-8" />

  <int:channel id="xml-inboundChannel" />

  <int-xml:xpath-router id="orderTypeRouter"
    input-channel="xml-inboundChannel">
    <int-xml:xpath-expression
      expression="/shiporder/shipto/country" />
    <int-xml:mapping value="Norway" channel="norwayChannel" />
    <int-xml:mapping value="USA" channel="usaChannel" />
  </int-xml:xpath-router>
```

```
<int:channel id="norwayChannel" />
<int:channel id="usaChannel" />

<int:recipient-list-router input-channel="norwayChannel">
  <int:recipient channel="norwayFileChannel" />
  <int:recipient channel="norwaySChannel" />
</int:recipient-list-router>

<int:channel id="norwayFileChannel" />
<int:channel id="norwaySChannel" />

<int-file:outbound-channel-adapter
  channel="norwayFileChannel" id="consumer-file-adapter1"
  directory="file:c://outboundNorway" />
<int-file:outbound-channel-adapter
  channel="usaChannel" id="consumer-file-adapter2"
  directory="file:c://outboundUsa" />

<int:service-activator id="printing-service-activator"
  input-channel="norwaySChannel" ref="serviceActivator" />
<bean id="serviceActivator"
class="com.intertech.lab5.ExampleServiceActivator" />

</beans>
```