Lab Exercise

Adapters - Lab 2

In the last lab, you explored channels that connect endpoints. Endpoints are Spring Integration (SI) components that put messages into a channel or consume messages from a channel. Adapters are endpoints that connect a message channel to an external system or technology. Adapters take content from outside of SI and bring it in as messages into SI channels or take SI messages and deposit their content to an external system.

Specifically, in this lab you will:

- Create and test an inbound file adapter.
- Create and test an outbound file adapter.

Lab solution folder: ExpressSpringIntegration\lab2\lab2-adapters-solution

Scenario

In this lab, you explore adapters. In fact, you have already used adapters in the first lab. You used Standard Input and Standard Output adapters to bring String data into the message channel and print it to the Console view.

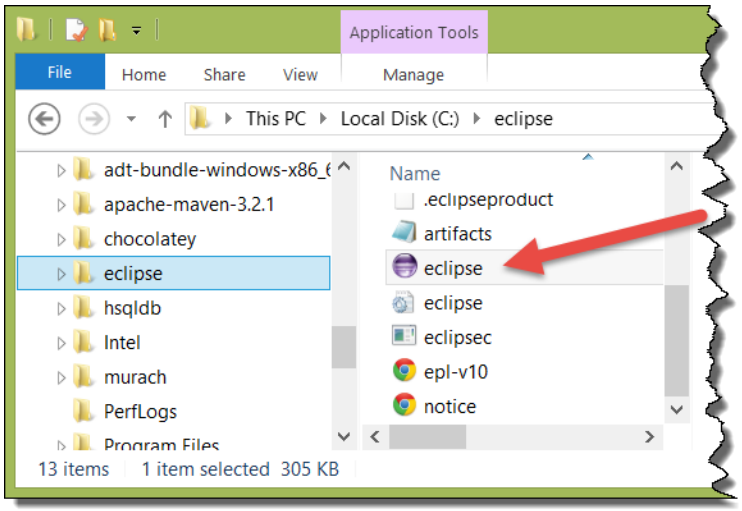
Step 1: Import the Maven Project

A Maven Project containing a slightly modified version of Lab 1 has already been created for you. You will use this project to begin your exploration of adapters.

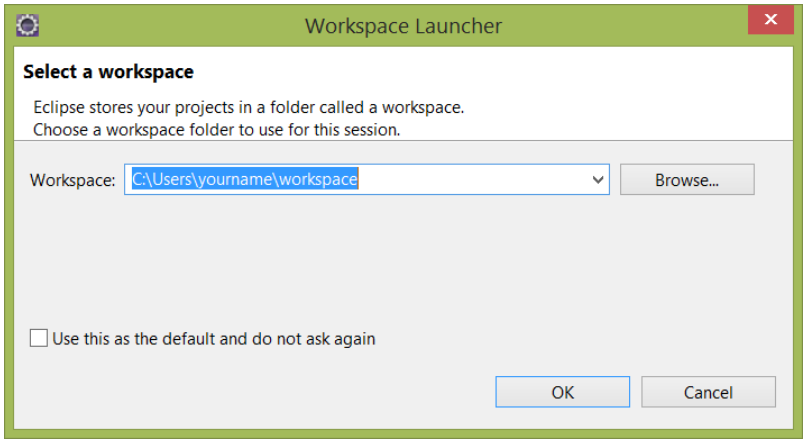
1.1 Start the Eclipse-based IDE. Locate and start Eclipse (or Eclipse-based) IDE.

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).



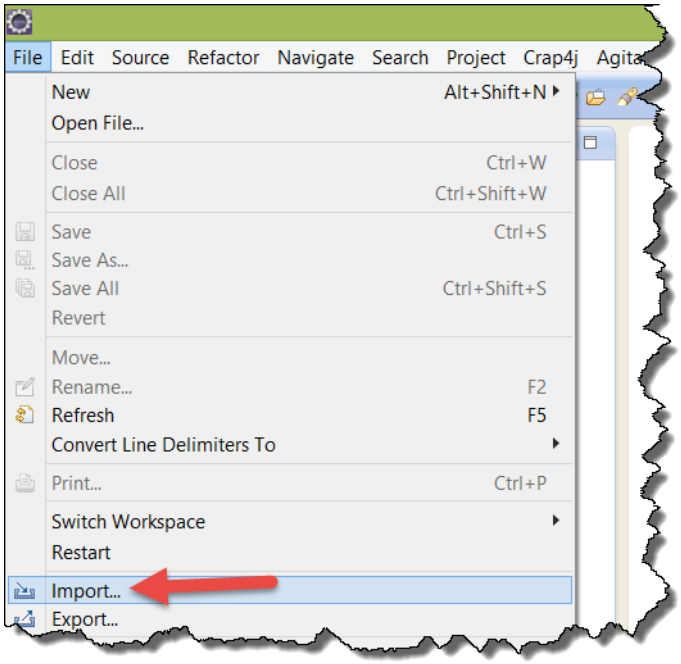
1.1.3 Open your workspace. Type in *C:\Users\<your username >\workspace* and click the *OK* button.



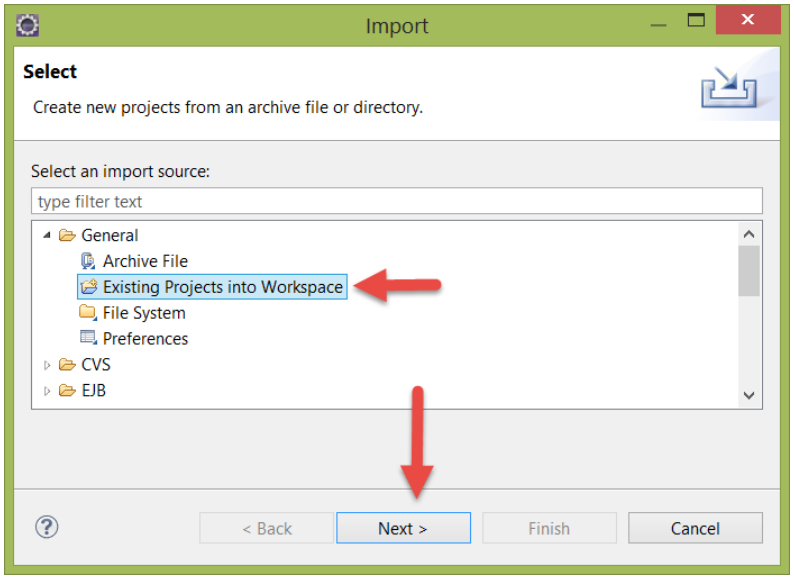
Note: As noted in the last lab, you may use an alternate location for your workspace, but the labs will always reference this location (c:\users\[your username]\workspace) as the default workspace location.

1.2 Import the new Maven Project.

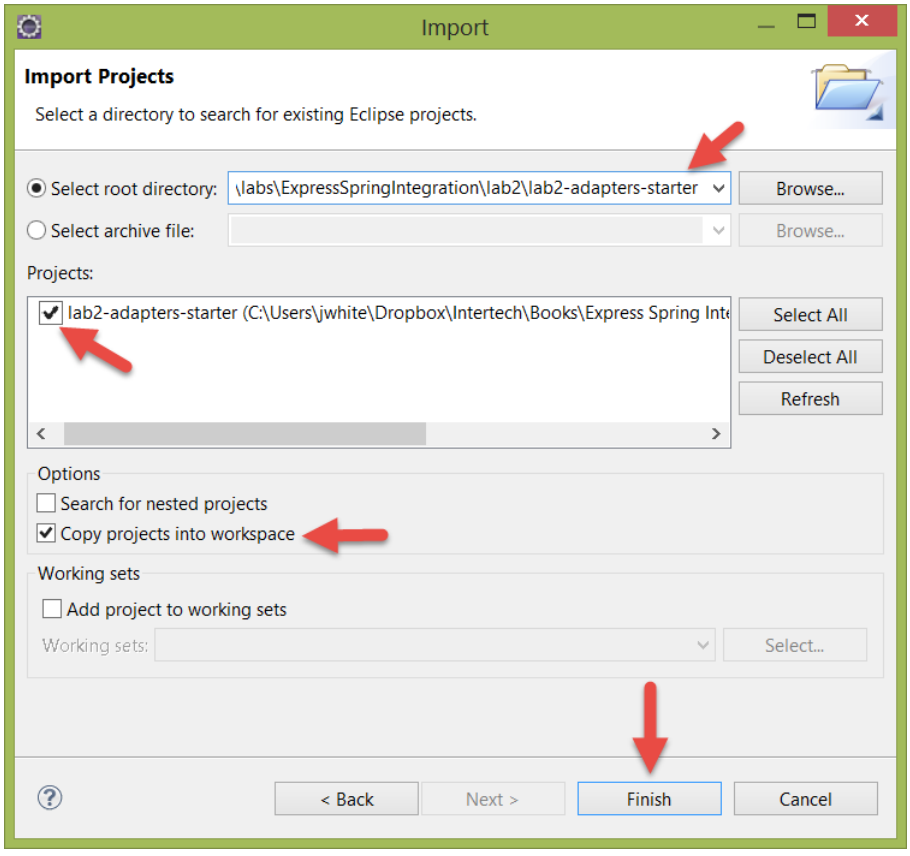
1.2.1 Select File>Import... from the Eclipse menu bar.



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

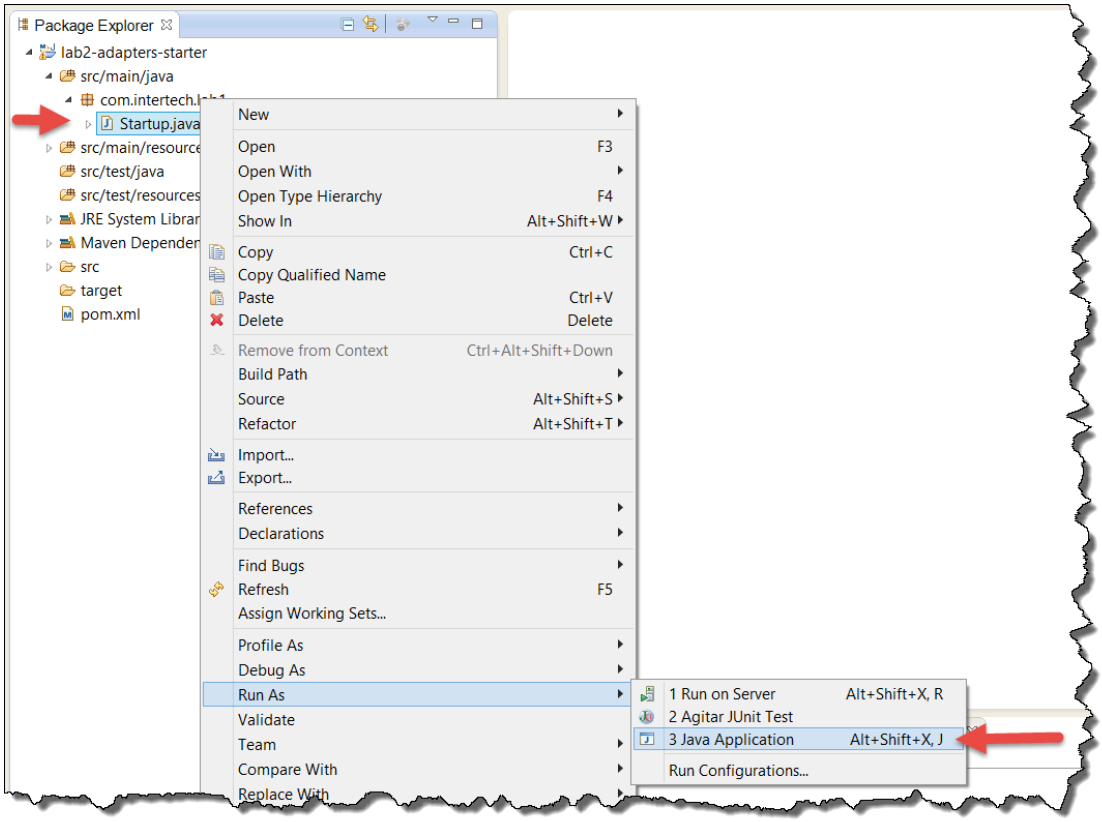


1.2.3 In the "Import" window, use the *Browse...* button to locate the lab2-adapters-starter project folder located in ExpressSpringIntegration\lab2 (this folder is located in the lab downloads). Make sure the project is selected, and the *"Copy projects into workspace"* checkbox is also checked before you hit the *Finish* button (as shown below).

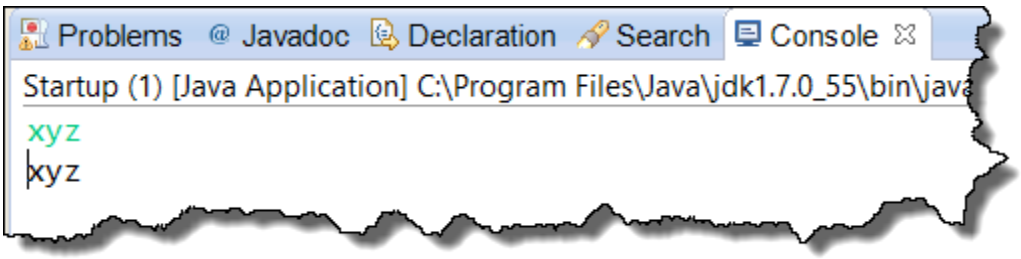


1.3 Test the application. Test the application to make sure the project works and to re-explore the Standard Input/Output adapters.

1.3.1 Locate the Startup.java file in the source folder. Right click on file and select Run As > Java Application from the resulting menu.

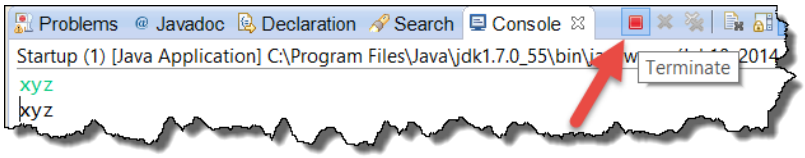


1.3.2 In the Console view, enter some text and then hit the *Enter* key (by default, your text will be displayed in green). As you saw in the last lab, a text message created from the text you enter into the Standard Input will immediately be marshalled into a text message and entered into the message channel and then delivered to the subscriber – the Standard Output channel - which displays the text back to the Console view (in black). In this lab, there is only a single Standard Output adapter so the text displays only one time.

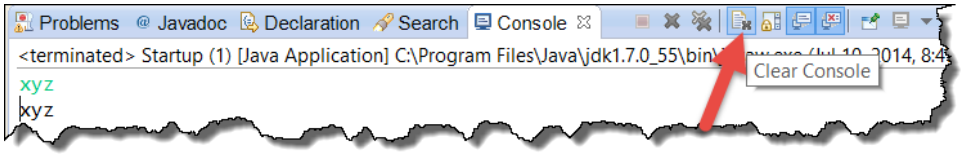


1.4 Stop the application. Recall the application is running in an infinite loop to allow for the constant publishing and consuming of messages. Stop the application now.

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



1.4.2 The Console view should now indicate that the application is "<terminated>". Clear the Console view by clicking on the Clear Console icon.

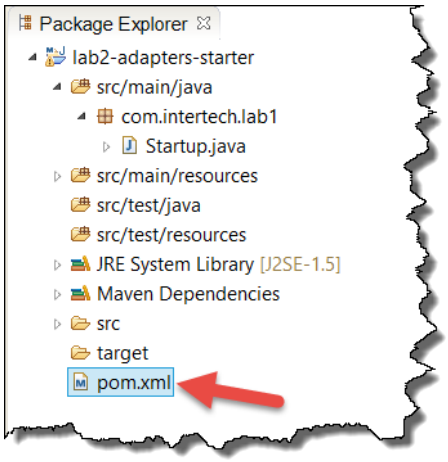


Step 2: Replace the Standard Output Adapter

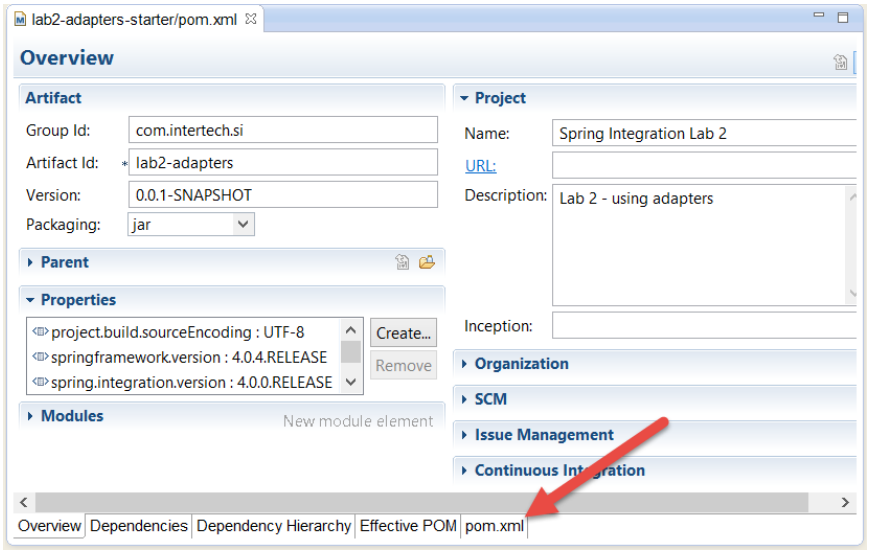
In this step, replace the Standard Output adapter with an outbound file adapter.

2.1 Add the Spring Integration File dependency to the pom.xml file. Most adapters are found in various SI modules. File adapters used to get files in and out of channels are found in SI's file module. You need to add this module to the project's pom.xml to include it in your project's build/runtime path.

2.1.1 Locate (in the root project folder) and open the pom.xml in an editor by double clicking on the file.



2.1.2 The pom.xml file will open in the "Overview" tab. Click on the pom.xml tab to see the raw xml source of the pom file.



2.1.3 Enter the SI file dependency as shown below. Add the dependency element inside of the <dependencies> </dependencies> tags.

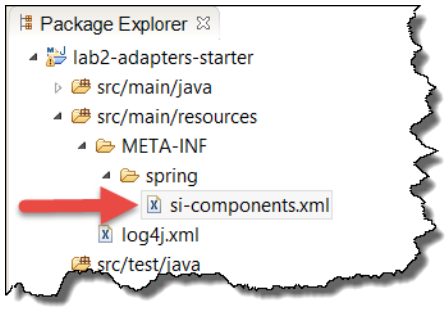
Note: if you get stuck or feel like not typing in all the code yourself, you will find a working copy of the final pom.xml file at ExpressSpringIntegration\lab2\lab2-adapters-solution\pom.xml

```
<dependency>
  <groupId>org.springframework.integration</groupId>
  <artifactId>spring-integration-file</artifactId>
  <version>${spring.integration.version}</version>
</dependency>
```

2.1.4 Save the pom.xml file and make sure there are no errors in the project.

2.2 Remove or comment out the Standard Output Channel.

2.2.1 Locate (in the src/main/resources/META-INF/spring folder) and open the si-components.xml in an editor by double clicking on the file.



2.2.2 Remove the stout-channel-adapter by placing XML comments around it (<!-- -->).

2.3 Add SI's file module namespace to the si-components.xml file. In order to use Spring Integration's file reading, writing, and transforming components, you need to add the SI file module namespace to the declarations at the top of the configuration file.

2.3.1 Add the file namespace to the <beans> root element.

Note: Again, if you do not wish to enter the code by hand, the final si-components.xml file for this lab can be found at ExpressSpringIntegration\lab2\lab2-adapters-solution\src\main\resources\META-INF\spring\si-components.xml.

```
<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-stream=
            "http://www.springframework.org/schema/integration/stream
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

2.3.1 Add the schema file location to the schemaLocation attribute.

```
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"
```


2.4 Add a new File output adapter. Like the Standard Input and Output adapters, you will usually find that adapters are either in or out adapters. The <int-file:outbound-channel-adapter> defines a SI File adapter that writes messages to a file system directory you specify.

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

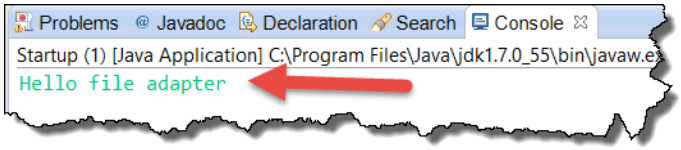
Note: The directory can be a location of your choosing. Make sure the directory is accessible (from a security standpoint) to Java.

2.5 Save the si-components.xml file and make sure there are no errors in the project.

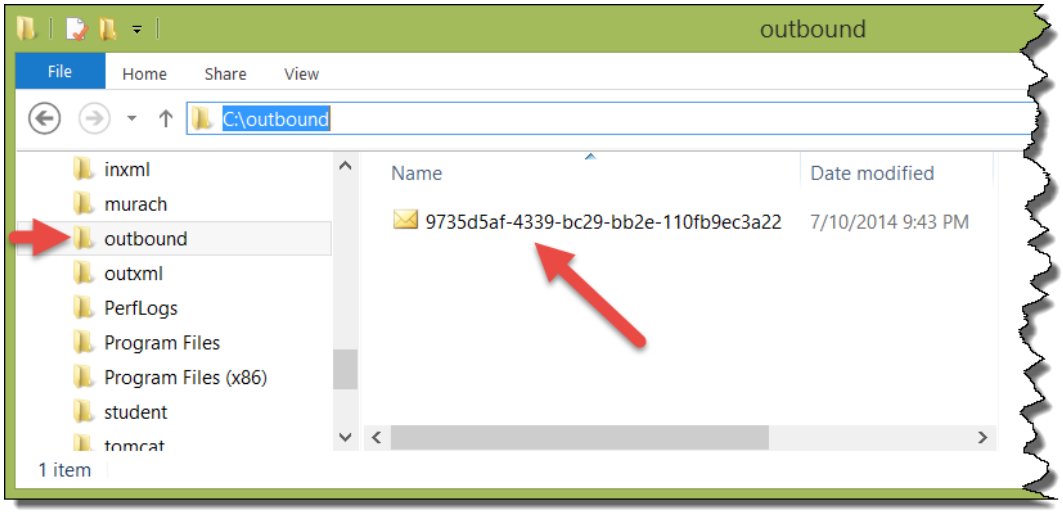
2.6 Retest the application. Test the application to see the text you enter into the Standard Input makes its way to the file system.

2.6.1 Locate the Startup.java file in the source folder. Right click on file and select Run As > Java Application from the resulting menu.

2.6.2 In the Console view, enter some text and then hit the *Enter* key. This time, the text you enter will not be redisplayed in the Console view by the Standard Output adapter (which was remove).

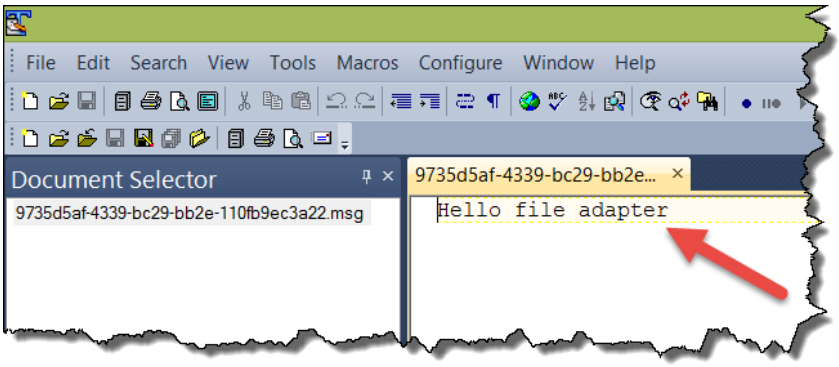


2.6.3 Using a Windows Explorer, open the folder specified as the File output adapter's directory (in step 2.4 above).



Note: The file has an automatically generated file name. SI allows you to develop a bean that generates the names of your choosing.

2.6.4 Right click on the file in Windows Explorer and open it with a text editor to see the text you entered into the Standard Input.



2.7 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.

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

2.7.2 The Console view should now indicate that the application is "<terminated>". Clear the Console view by clicking on the Clear Console icon.

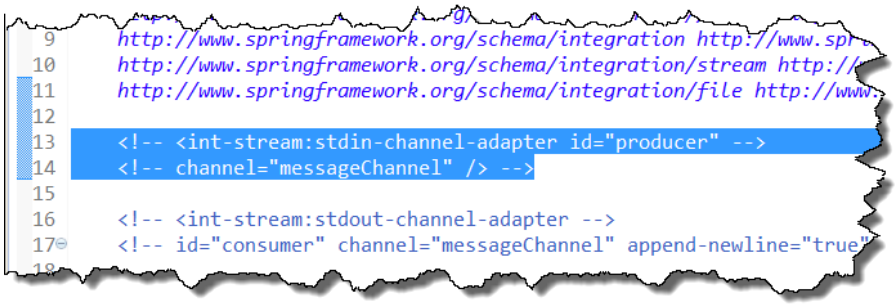
Step 3: Replace the Standard Input Adapter

In this step, replace the Standard Input adapter with an inbound file adapter.

3.1 Remove or comment out the Standard Input Channel.

3.1.1 Again, locate (in the src/main/resources/META-INF/spring folder) and open the si-components.xml in an editor by double clicking on the file.

3.1.2 Remove the stdin-channel-adapter by placing XML comments around it (<!-- -->).



3.2 Add a new File input adapter. The <int-file:inbound-channel-adapter> defines a SI File adapter that writes messages to a file system directory you specify.

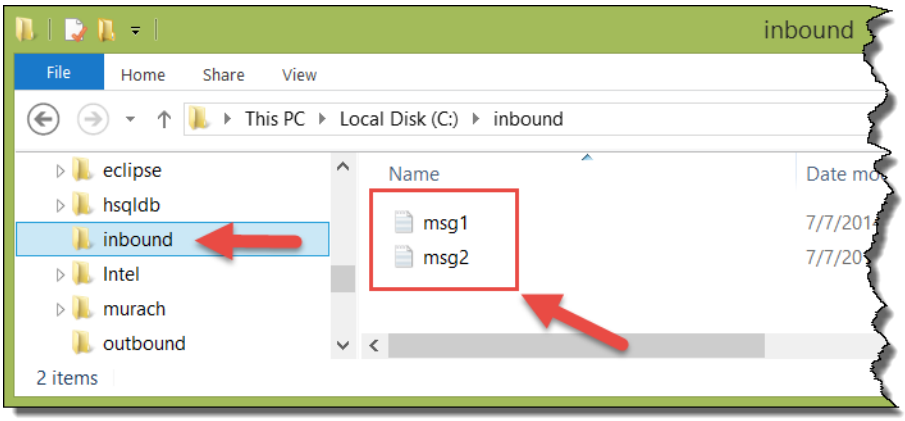
```
<int-file:inbound-channel-adapter id="producer-file-adapter"
  channel="messageChannel" directory="file:c://inbound"
  prevent-duplicates="true">
    <int:poller fixed-rate="5000" />
</int-file:inbound-channel-adapter>
```

Note: The directory can be a location of your choosing. Make sure the directory is accessible (from a security standpoint) to Java.

3.3 Save the si-components.xml file and make sure there are no errors in the project.

3.4 Retest the application. Test the application to see the file moved from your inbound directory to the outbound directory in the file system.

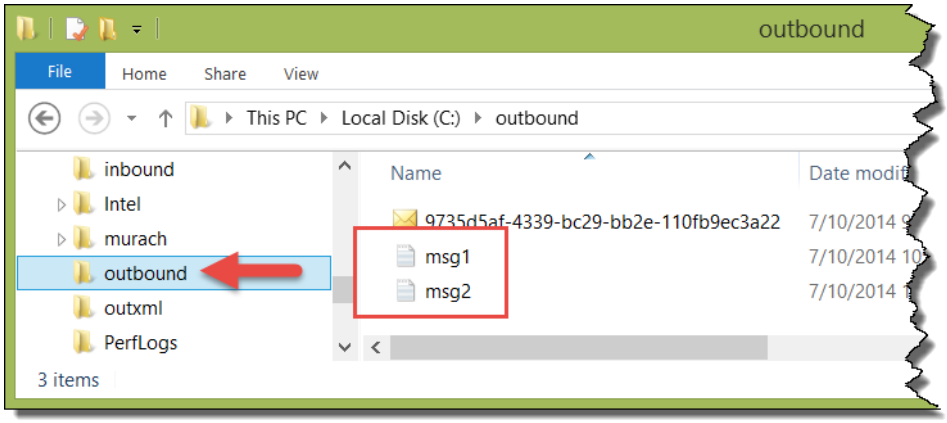
3.4.1 Use a text editor to create a file in the inbound directory (c:/inbound is the inbound directory used in step 3.2). The content of the file does not matter; just the existence of the file is important.



Note: Feel free to create as many files as you would like. All the files in the directory will be moved by SI to the outbound directory.

3.4.2 Now start the application. Locate the Startup.java file in the source folder. Right click on file and select Run As > Java Application from the resulting menu.

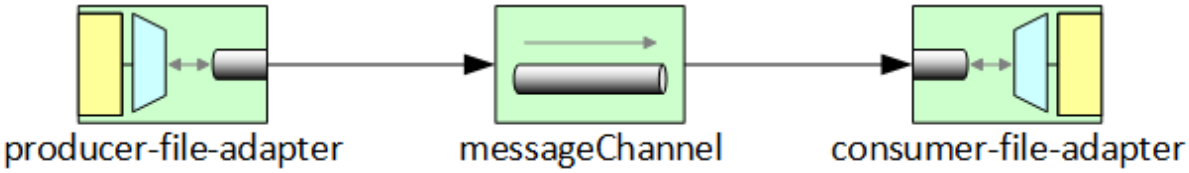
3.4.3 Nothing will display in the Console view in Eclipse. Using a Windows Explorer, open the folder specified as the File input adapter's directory.



Note: If you did add content to the message files, you can use a text editor to see that the content remains the same.

3.5 Terminate the application and clear Console view.

3.6 Here is the EIP diagram for this application.



Adapters are an important component to every SI project. They allow the input and output of messages to SI pipelines. There are many types of adapters. You have seen two over the past two labs: Standard Input/Output adapters and File adapters. There are other adapters to get/put messages to JMS messages queues, to select/insert data into relational databases, send/receive mail messages from an SMTP mail system, etc.

Lab Solution

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.intertech.si</groupId>
 <artifactId>lab2-adapters</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 <name>Spring Integration Lab 2</name>
 <description>Lab 2 - using adapters</description>
 <properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <springframework.version>4.0.4.RELEASE</springframework.version>

<spring.integration.version>4.0.0.RELEASE</spring.integration.version>

<spring.integration.xml.version>4.0.0.RELEASE</spring.integration.xml.
version>
  <log4j.version>1.2.17</log4j.version>
 </properties>

 <dependencies>

  <dependency>
   <groupId>org.springframework.integration</groupId>
   <artifactId>spring-integration-core</artifactId>
   <version>${spring.integration.version}</version>
  </dependency>

  <dependency>
   <groupId>org.springframework.integration</groupId>
   <artifactId>spring-integration-stream</artifactId>
   <version>${spring.integration.version}</version>
  </dependency>

  <dependency>
   <groupId>org.springframework.integration</groupId>
   <artifactId>spring-integration-file</artifactId>
   <version>${spring.integration.version}</version>
  </dependency>

  <dependency>
   <groupId>log4j</groupId>
   <artifactId>log4j</artifactId>
   <version>${log4j.version}</version>
  </dependency>

 </dependencies>
</project>
```


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: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">

 <!-- <int-stream:stdin-channel-adapter id="producer" -->
 <!-- channel="messageChannel" /> -->

 <!-- Adapter for reading files -->
 <int-file:inbound-channel-adapter id="producer-file-adapter"
  channel="messageChannel" directory="file:c://inbound"
  prevent-duplicates="true">
  <int:poller fixed-rate="5000" />
 </int-file:inbound-channel-adapter>

 <!-- <int-stream:stdout-channel-adapter -->
 <!-- id="consumer" channel="messageChannel" append-newline="true" />
-->

 <!-- Adapter for writing files -->
 <int-file:outbound-channel-adapter
  channel="messageChannel" id="consumer-file-adapter"
  directory="file:c://outbound" />

 <int:poller id="defaultPoller" default="true"
  max-messages-per-poll="5" fixed-rate="200" />

 <!-- a direct channel -->
 <int:channel id="messageChannel" />
</beans>
```