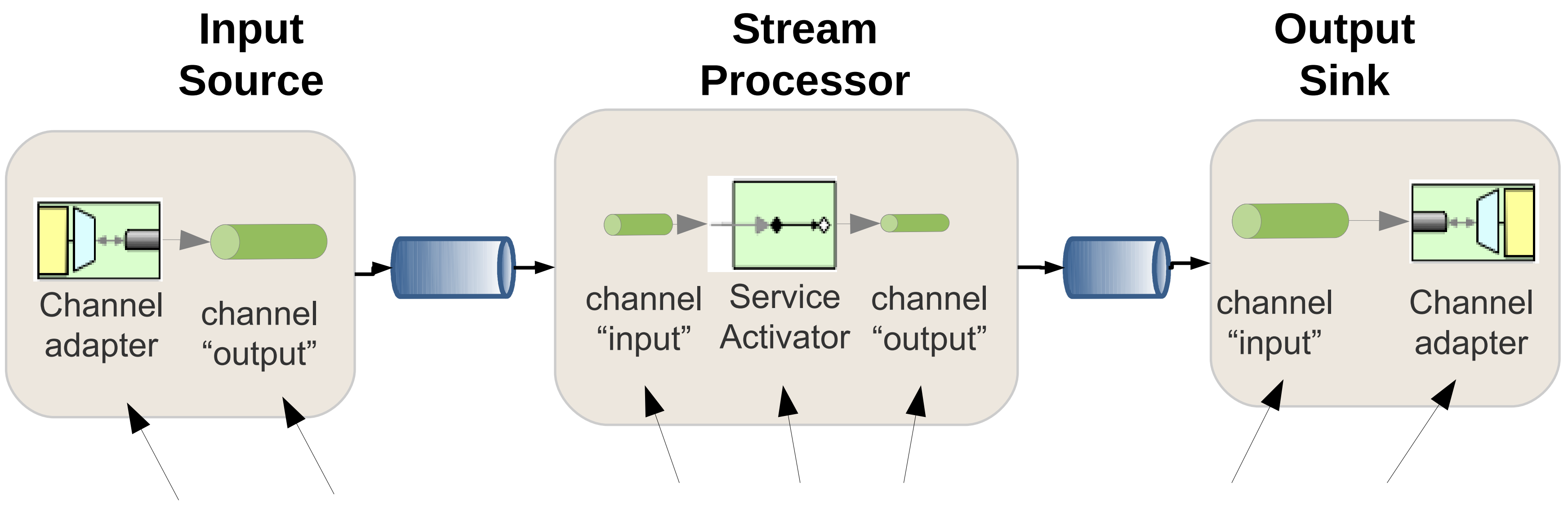
Spring XD

Creating Your Own Module

Data Ingestion and Spring XD

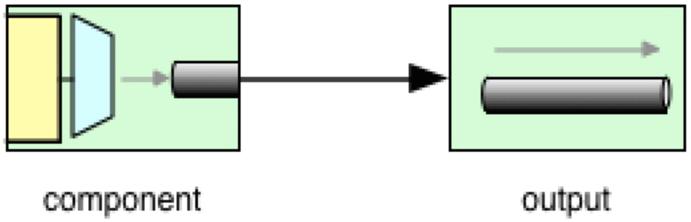
Spring XD Modules – Behind the Scenes



- Spring Integration provides the internal components used within Spring XD Modules

Spring XD Source Modules

- The first module in a stream is always a **source**
- Source is responsible for producing messages to an underlying direct channel named **output**
  - Message then consumed by the downstream modules in the stream.
  - Typically built using an inbound channel adapter, configured with a poller or triggered by an event



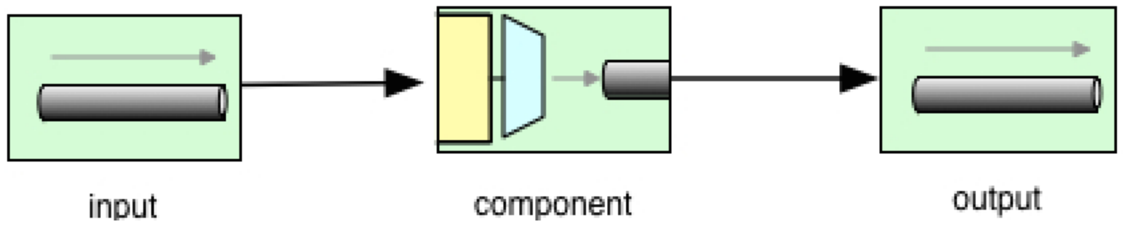
Spring XD JMS Source Module Example

```
<int-jms:message-driven-channel-adapter id="jmsSource"
    auto-startup="false"
    destination-name="${destination}"
    pub-sub-domain="${pubSub}"
    subscription-durable="${durableSubscription}"
    durable-subscription-name="${subscriptionName:#{null}}"
    client-id="${clientId:#{null}}"
    channel="output"
    connection-factory="connectionFactory"/>

<int:channel id="output"/>
```

See: XD-HOME/libexec/xd/modules/source/jms/config/file.xml

Processor Modules

- Processors are optional modules within a Spring XD stream
  - Never the first or last module
- Processors are expected to:
  - Consuming messages from a direct channel named input
  - Producing messages on a direct channel named output



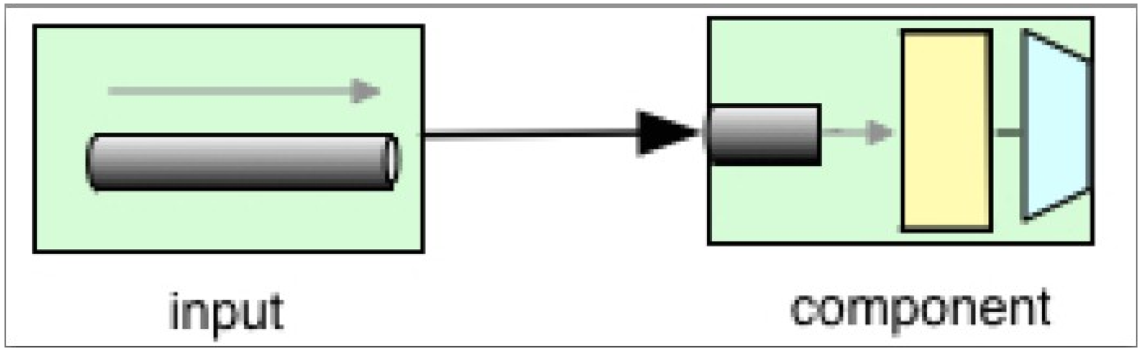
Spring XD Filter Processor Module Example

```
<channel id="input"/>

<filter id="invokeScript" input-channel="input"
  output-channel="to.script"
  discard-channel="to.spel" ... />

<filter input-channel="to.spel" ...
        output-channel="output"/>

<filter input-channel="to.script"
        output-channel="output"> ...
</filter>

<channel id="output"/>
```

See: XD-HOME/libexec/xd/modules/processor/filter/config/filter.xml

Sink Modules

- The last module in a stream is always a sink
- Sink is responsible for consuming messages from a direct channel named input
  - Message originated from upstream processor or source.
  - Message is typically sent to an outbound channel adapter
    - Which provides the message to an external resource
-



Spring XD File Sink Module Example

```
      <channel id="input"/>

      <logging-channel-adapter channel="input" level="${level}"
                   logger-name="xd.sink.${name}"
                  expression="${expression}" />
```

See: XD-HOME/libexec/xd/modules/sink/log/config/file.xml

Creating Spring XD Custom Modules

- Source modules
  - *Must* define a direct channel named **output**, typically connected to an inbound adapter
- Processor modules
  - *Must* have a direct input channel named **input** and a subscribe-able output channel named **output**
  - Typically performs transformations
- Sink modules
  - *Must* receive messages on a direct channel named **input**, and connect to an output adapter or service activator

What are we doing?

- The Scenario:
  - We are creating a source module that retrieves data via the Feed Adapter (RSS/ATOM) and transforms that data to JSON before passing it to the next module in the stream.
    - This project can be found: https://github.com/spring-projects/spring-xd-samples/tree/master/rss-feed-source
- Things to watch for:
  - Build Configuration - Notice in this scenario we have to include dependencies that are not included with XD
  - Where properties and xml files are located
  - How we setup module options.
  - How we upload the jar and name selection for the module.

Project Setup

```
project
    ├── src
    │   ├── java
    │   ├── resources
    │       ├── config
    ├build.gradle
    ├settings.gradle
  <OR>
    ├pom.xml
```

spring

Pivotal™

Gradle build.gradle

```
buildscript {
    repositories {
       ...
    }
    dependencies {
      classpath("org.springframework.xd:spring-xd-module-plugin:1.1.0.RELEASE")
    }
}

ext {
    springXdVersion = '1.1.0.RELEASE'
    springIntegrationVersion = '4.1.2.RELEASE'
}


group = 'com.acme'
version = '1.0.0.BUILD-SNAPSHOT'

description = "Spring XD source module for RSS Feed"


repositories {
     …
}

dependencies {
    compile "org.springframework.integration:spring-integration-feed:$springIntegrationVersion"
}
```

Include the XD Gradle plugin

Setup group

Add dependencies not already in XD

Gradle settings.gradle

```
rootProject.name = 'rss-feed-source'
```

Setup rootProject name (artifactId)

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0...>
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.acme</groupId>
   <artifactId>rss-feed-source</artifactId>
   <version>1.0.0.BUILD-SNAPSHOT</version>
   <parent>
      <groupId>org.springframework.xd</groupId>
      <artifactId>spring-xd-module-parent</artifactId>
      <version>1.1.0.RELEASE</version>
   </parent>

   <dependencies>
      <dependency>
         <groupId>org.springframework.integration</groupId>
         <artifactId>spring-integration-feed</artifactId>
         <version>4.1.2.RELEASE</version>
      </dependency>
   </dependencies>
   <repositories>
    ...
   </repositories>
</project>
```

Setup group and artifact id

Include the XD Starter parent

Add dependencies not already in XD

Setting up the module

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      ...
      xsi:schemaLocation="http://www.springframework.org/schema/beans
      ...
      http://www.springframework.org/schema/integration/feed/spring-integration-feed.xsd">

   <int-feed:inbound-channel-adapter id="xdFeed" channel="to.json" url="${url}" auto-startup="false">
      <int:poller fixed-rate="${fixedRate}" max-messages-per-poll="${maxMessagesPerPoll}" />
   </int-feed:inbound-channel-adapter>

   <int:transformer input-channel="to.json" output-channel="output">
      <bean class="com.acme.SyndEntryJsonTransformer"/>
   </int:transformer>

   <int:channel id="output"/>


</beans>
```

Retrieve RSS or ATOM data from a URL

Where do the props get defined?

XdFeed sends its data to a JSON Transformer

Once the data is translated it Is sent to the next module Via the output channel

Module Options

- Each module can expose metadata about the options it accepts.
- In the properties file a user specifies the module options required for the module in the following format.
  - options.*option_name*.*definition*
    - I.e options.fixedRate.default=5000
- Definitions include:
  - Description – a summary of the option displayed to the user when they hit module Info
  - Default - The default value. If not present the user must specify the value when stream is created.
  - Type – int, long, String, or other fully qualified class name

Module Options Example:

- In our example we have 3 properties
  - Url
  - FixedRate
  - maxMessagesPerPoll

```
options.url.description = the URL of the RSS feed
options.url.type = java.lang.String

options.fixedRate.description = the fixed rate polling interval
specified in milliseconds
options.fixedRate.default = 5000
options.fixedRate.type = int

options.maxMessagesPerPoll.description = the maximum number of
messages per poll
options.maxMessagesPerPoll.default = 100
options.maxMessagesPerPoll.type = int
```

The Translator

The SyndEntry sent from the Feed is converted to a Json String

```
package com.acme;
...

@JsonFilter("foreignMarkup filter")
public class SyndEntryJsonTransformer {
   private final ObjectMapper mapper;

   /**
    * Configure ObjectMapper to filter out fields causing serialization problems
    */
   public SyndEntryJsonTransformer() {
      mapper = new ObjectMapper();
      mapper.addMixInAnnotations(SyndEntry.class,this.getClass());
      FilterProvider filterProvider = new SimpleFilterProvider()
            .addFilter("foreignMarkup filter", SimpleBeanPropertyFilter.serializeAllExcept("foreignMarkup"));
      mapper.setFilters(filterProvider);
   }

   /**
    * Convert from SyndEntry to JSON string
    * @param entry the SyndEntry
    * @return JSON string
    */
   public String toJson(SyndEntry entry) {
      try {
         return mapper.writer().writeValueAsString(entry);
      }
      catch (JsonProcessingException e) {
         throw new RuntimeException(e.getMessage(),e);
      }
   }
}
```

Location of files

```
project
    ├── src
        ├── java
            ├── com.acme.SyndEntryJsonTransformer
        ├── resources
            ├── config
                ├── spring-module.properties
                ├── spring-module.xml
```

Note that properties and xml are in the config directory. If not they will not be found

Building your Module Jar

- Maven
  - Maven clean package
- Gradle
  - ./gradlew clean bootRepackage

Uploading the new jar

- From your shell execute

```
xd:>module upload --file /Users/glennrenfro/project/spring-xd-
samples/rss-feed-source/target/rss-feed-source-1.0.0.BUILD-
SNAPSHOT.jar --name myRssFeed --type source
Successfully uploaded module 'source:myRssFeed'
```

Type Declaration

Module Name

- Verify that it uploaded by executing

```
xd:>module info source:myRssFeed
Information about source module 'myRssFeed':

  Option Name         Description                                                  Default  Type
  ------------------  -----------------------------------------------------------  -------  --------
  fixedRate           the fixed rate polling interval specified in milliseconds    5000     int
  url                 the URL of the RSS feed                                      <none>   String
  maxMessagesPerPoll  the maximum number of messages per poll                      100      int
  outputType          how this module should emit messages it produces             <none>   MimeType
```

Testing the Module in a stream

- From your shell create a stream to read from the ABC news feed

  -
    ```
    stream create abc --definition "myRssFeed
    --url=http://feeds.abcnews.com/abcnews/topstories|log" --deploy
    ```
- Let's view the log:

```
2015-02-16 17:16:33,937 1.1.0.SNAP  INFO xdbus.abc.0-1 sink.abc - {"uri":"http://abcnews.go.com/International/wireStory/greece-creditors-debt-talks-28994036","link":"http://abcnews.go.com/International/wireStory/greece-creditors-debt-talks-28994036","comments":null,"updatedDate":null,"title":"Eurozone Issues Greece an Ultimatum; Athens Hopeful of Deal","description":{"type":"text/html","value":"Amid signs of discord, eurozone meeting gives Greece rest of week to request bailout extension","mode":null,"interface":"com.rometools.rome.feed.synd.SyndContent"},"links":[],"contents":[],"modules":[{"uri":"http://purl.org/dc/elements/1.1/","title":null,"creator":null,"subject":null,"description":null,"publisher":null,"contributors":[],"date":1424124829000,"type":null,"format":null,"identifier":null,"source":null,"language":null,"relation":null,"coverage":null,"rights":null,"types":[],"sources":[],"identifiers":[],"formats":[],"subjects":[],"interface":"com.rometools.rome.feed.module.DCModule","creators":[],"titles":[],"descriptions":[],"publishers":[],"contributor":null,"dates":[1424124829000],"languages":[],"relations":[],"coverages":[],"rightsList":[]}],"enclosures":[],"authors":[],"contributors":[],"source":null,"wireEntry":null,"categories":[{"name":"International","interface":"com.rometools.rome.feed.synd.SyndCategory","taxonomyUri":null}],"publishedDate":1424124829000,"titleEx":{"type":null,"value":"Eurozone Issues Greece an Ultimatum; Athens Hopeful of Deal","mode":null,"interface":"com.rometools.rome.feed.synd.SyndContent"},"author":"","interface":"com.rometools.rome.feed.synd.SyndEntry"}
```

So what just happened

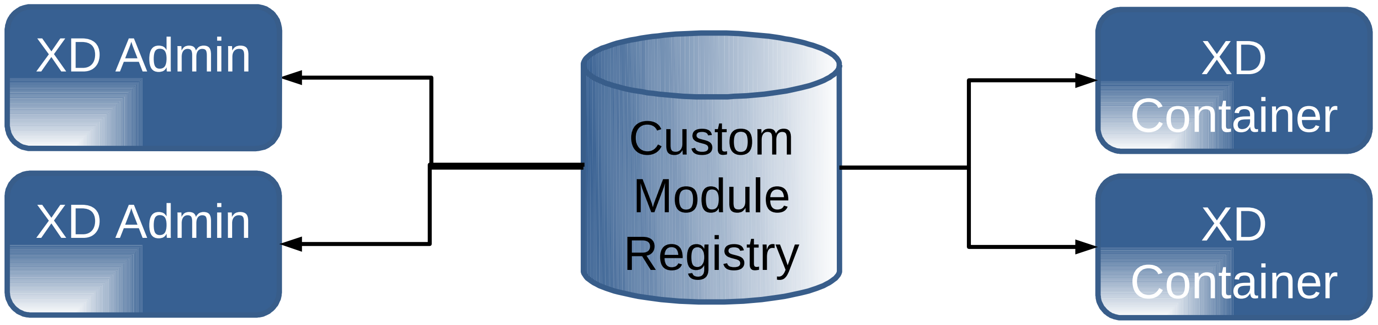
- We have registered the module with the Spring XD Module Registry
- The module uploaded the rss-feed-source-1.0.0.BUILD-SNAPSHOT.jar to the ${XD_HOME}/custom-modules/source directory
  - Convention is that the module name is normally a single word representing the purpose of the module

Module Registry

- Spring XD Module Registry is configured to search for modules in the following locations, in order:
  - File path specified in **`servers.yml`** by **`xd.module.home`**
    - Default is **`${xd.home}/modules`**
  - **`classpath:/modules/`**
    - Empty by default
  - File path specified in **`servers.yml`** by **`xd.customModule.home`**
    - Default is **`${xd.home}/custom-modules`**

Custom Module Registry Best Practices

- Override default **`xd.customModule.home`** setting to a shared directory outside of the Spring XD installation
  - Will prevent custom modules being overwritten by Spring XD product upgrades
- The location must be a network share so that the XD-Admin and all container servers access a single copy in a distributed deployment (DIRT)



Lab