# POJO Messaging Example

## POJO Messaging Example

### Introduction

This example shows that you don't need to learn Camel's super cool DSLs if you don't want to. Camel has a set of annotations that allow you to produce, consume or route messages to endpoints.

### Requirements

The example is shipped with Camel 2.0, but only requires Camel 1.5 to run. It also depends on the camel-jms component and Apache ActiveMQ. Of course, since we are using Maven these dependencies will be downloaded automatically.

### Running the example

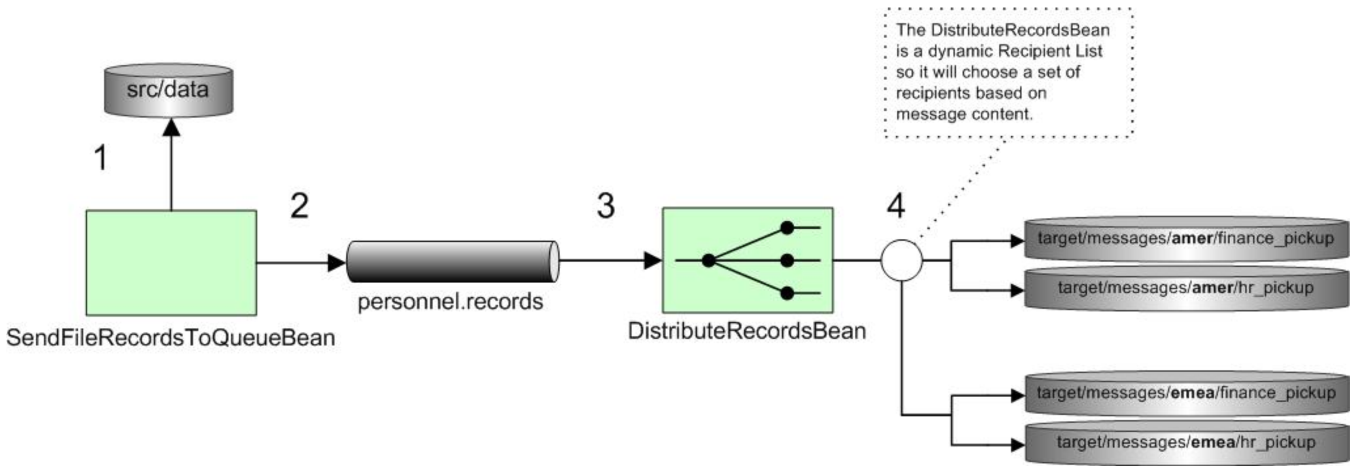
The `README.txt` states how to run the example from either Ant or Maven.

Here's how to run it with Maven:

mvn compile camel:run

### Whats happening?

When you start the example up you'll see a whole bunch of logs that won't really mean anything to you 🙂 The interesting stuff is happening in the background. Here's a diagram of whats going on.


The DistributeRecordsBean is a dynamic Recipient List so it will choose a set of recipients based on message content.
src/data
1
2
3
4
SendFileRecordsToQueueBean
personnel.records
DistributeRecordsBean
target/messages/**amer**/finance_pickup
target/messages/**amer**/hr_pickup
target/messages/**emea**/finance_pickup
target/messages/**emea**/hr_pickup

At step 1 the `SendFileRecordsToQueueBean` polls the `./src/data` directory for new files. There are 3 files in this directory so 3 messages will be created. As shown below, the `@Consume` annotation will cause any new messages coming from the `file:src/data endpoint` to be sent to the `onFileSendToQueue` method.

{snippet:id=ex|lang=java|url=camel/trunk/examples/camel-example-pojo-messaging/src/main/java/org/apache/camel/example/pojo_messaging/SendFileRecordsToQueueBean.java}

At step 2 the `SendFileRecordsToQueueBean` then sends the contents of the `File` message as a `String` to the `personnel.records` JMS queue, which is backed by an embedded instance of Apache ActiveMQ. The conversion from String to JMS message is automatic. The `@Produce` annotation is used to access the ActiveMQ endpoint.

At step 3 the `DistributeRecordsBean` (shown below) consumes the JMS message from the `personnel.records` queue. Again the `@Consume` annotation is used to get messages from the ActiveMQ endpoint.

{snippet:id=ex|lang=java|url=camel/trunk/examples/camel-example-pojo-messaging/src/main/java/org/apache/camel/example/pojo_messaging/DistributeRecordsBean.java}

You will notice an additional `@RecipientList` annotation on the route method. This turns the method into a Recipient List EIP where the return value is a list of URIs for the recipients (can be String[], List<String>, URI[], etc). This annotation is great for creating custom dynamic Recipient Lists. In this case at step 4 we peek at the city field in the message (using the `@XPath` annotation) and provide a set of recipients based on that. For folk from London, their files will be sent to file locations for the EMEA region (`file:target/messages/emea/...`). Others fall into the AMER region (`file:target/messages/amer/...`).

If you have messages that are not XML, don't fret! Camel has ways to get information out of arbitrary message payloads. For instance, you can try using the @Bean annotation to peek at the message using your own Java bean.

@Consume(uri = "activemq:personnel.records") @RecipientList public String[] route(@Bean("cityExtractorBean") String city) { if (city.equals("London")) {

Check out Parameter Binding Annotations for more information on this.

After running the example, browse to the `target/messages` directory to see where the messages were saved.

## See also

- Bean Integration
- POJO Producing
- POJO Consuming
- RecipientList Annotation
- Recipient List
- Parameter Binding Annotations
- Examples
- Tutorials
- User Guide