JavaSpace

JavaSpace Component

Available as of Camel 2.1

The **javaspace** component is a transport for working with any JavaSpace compliant implementation and this component has been tested with both the Blitz implementation and the GigaSpace implementation .
This component can be used for sending and receiving any object inheriting from the Jini `net.jini.core.entry.Entry` class. It is also possible to pass the bean ID of a template that can be used for reading/taking the entries from the space.
This component can be used for sending/receiving any serializable object acting as a sort of generic transport. The JavaSpace component contains a special optimization for dealing with the `BeanExchange`. It can be used to invoke a POJO remotely, using a JavaSpace as a transport.
This latter feature can provide a simple implementation of the master/worker pattern, where a POJO provides the business logic for the worker.
Look at the test cases for examples of various use cases for this component.

Maven users will need to add the following dependency to their `pom.xml` for this component:

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-javaspace</artifactId>
    <version>x.x.x</version>
    <!-- use the same version as your Camel core version -->
</dependency>
```

URI format

```
javaspace:jini://host[?options]
```

You can append query options to the URI in the following format, `?option=value&option=value&...`

Options

Name	Default Value	Description
`spaceName`	`null`	Specifies the JavaSpace name.
`verb`	`take`	Specifies the verb for getting JavaSpace entries. The values can be: `take` or `read`.
`transactional`	`false`	If `true`, sending and receiving entries is performed within a transaction.
`transactionalTimeout`	`Long.MAX_VALUE`	Specifies the transaction timeout.
`concurrentConsumers`	`1`	Specifies the number of concurrent consumers getting entries from the JavaSpace.
`templateId`	`null`	If present, this option specifies the Spring bean ID of the template to use for reading/taking entries.

Examples

Sending and Receiving Entries

```
// sending route
from("direct:input")
    .to("javaspace:jini://localhost?spaceName=mySpace");

// receiving Route
from("javaspace:jini://localhost?spaceName=mySpace&templateId=template&verb=take&concurrentConsumers=1")
    .to("mock:foo");
```

In this case the payload can be any object that inherits from the Jini `Entry` type.

Sending and receiving serializable objects

Using the preceding routes, it is also possible to send and receive any serializable object. The JavaSpace component detects that the payload is not a Jini `Entry` and then it automatically wraps the payload with a Camel Jini `Entry`. In this way, a JavaSpace can be used as a generic transport mechanism.

Using JavaSpace as a remote invocation transport

The JavaSpace component has been tailored to work in combination with the Camel bean component. It is therefore possible to call a remote POJO using JavaSpace as the transport:

```
// client side
from("direct:input")
    .to("javaspace:jini://localhost?spaceName=mySpace");

// server side
from("javaspace:jini://localhost?concurrentConsumers=10&spaceName=mySpace")
    .to("mock:foo");
```

In the code there are two test cases showing how to use a POJO to realize the master/worker pattern. The idea is to use the POJO to provide the business logic and rely on Camel for sending/receiving requests/replies with the proper correlation.

See Also

- Configuring Camel
- Component
- Endpoint
- Getting Started