JCR

JCR Component

The **`jcr`** component allows you to add/read nodes to/from a JCR compliant content repository (for example, Apache Jackrabbit) with its producer, or register an EventListener with the consumer.

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

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

URI format

```
jcr://user:password@repository/path/to/node
```

> **Consumer added**
>
> From **Camel 2.10** onwards you can use consumer as an EventListener in JCR or a producer to read a node by identifier.

Usage

The `repository` element of the URI is used to look up the JCR `Repository` object in the Camel context registry.

Producer

Name	Default Value	Description
`CamelJcrOperation`	`CamelJcrInsert`	CamelJcrInsert or CamelJcrGetById operation to use
`CamelJcrNodeName`	`null`	Used to determine the node name to use.
`CamelJcrNodeType`	`null`	**Camel 2.16:** To use a specify primary node type when creating adding a new node.

When a message is sent to a JCR producer endpoint:

- If the operation is CamelJcrInsert: A new node is created in the content repository, all the message headers of the IN message are transformed to `javax.jcr.Value` instances and added to the new node and the node's UUID is returned in the OUT message.
- If the operation is CamelJcrGetById: A new node is retrieved from the repository using the message body as node identifier.

> Please note that the JCR Producer used message properties instead of message headers in Camel versions earlier than 2.12.3. See https://issues.apache.org/jira/browse/CAMEL-7067 for more details.

Consumer

The consumer will connect to JCR periodically and return a List<javax.jcr.observation.Event> in the message body.

Name	Default Value	Description
`eventTypes`	`0`	A combination of one or more event types encoded as a bit mask value such as javax.jcr.observation.Event.NODE_ADDED, javax.jcr.observation.Event.NODE_REMOVED, etc.
`deep`	`false`	When it is true, events whose associated parent node is at current path or within its subgraph are received.
`uuids`	`null`	Only events whose associated parent node has one of the identifiers in the comma separated uuid list will be received.

`nodeTypeNames`	`null`	Only events whose associated parent node has one of the node types (or a subtype of one of the node types) in this list will be received.
`noLocal`	`false`	If `noLocal` is `true`, then events generated by the session through which the listener was registered are ignored. Otherwise, they are not ignored.
`sessionLiveCheckInterval`	`60000`	Interval in milliseconds to wait before each session live checking.
`sessionLiveCheckIntervalOnStart`	`3000`	Interval in milliseconds to wait before the first session live checking.
`username`		**Camel 2.15:** Allows to specify the username as a uri parameter instead of in the authority section of the uri
`password`		**Camel 2.15:** Allows to specify the password as a uri parameter instead of in the authority section of the uri
`workspaceName`	`null`	**Camel 2.16:** Allows to specify a workspace different from default

Example

The snippet below creates a node named `node` under the `/home/test` node in the content repository. One additional property is added to the node as well: `my.contents.property` which will contain the body of the message being sent.

```
from("direct:a").setHeader(JcrConstants.JCR_NODE_NAME, constant("node"))
    .setHeader("my.contents.property", body())
    .to("jcr://user:pass@repository/home/test");
```

The following code will register an EventListener under the path import-application/inbox for Event.NODE_ADDED and Event.NODE_REMOVED events (event types 1 and 2, both masked as 3) and listening deep for all the children.

```
<route>
    <from uri="jcr://user:pass@repository/import-application/inbox?eventTypes=3&deep=true" />
    <to uri="direct:execute-import-application" />
</route>
```

See Also

- Configuring Camel
- Component
- Endpoint
- Getting Started