Restlet

Restlet Component

The **Restlet** component provides Restlet based endpoints for consuming and producing RESTful resources.

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

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

URI format

javarestlet:restletUrl[?options]

Format of restletUrl:

javaprotocol://hostname[:port][/resourcePattern]

Restlet promotes decoupling of protocol and application concerns. The reference implementation of Restlet Engine supports a number of protocols. However, we have tested the HTTP protocol only. The default port is port 80. We do not automatically switch default port based on the protocol yet.

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

It seems Restlet is case sensitive in understanding headers. For example to use content-type, use Content-Type, and for location use Location and so on. We have received a report about drop in performance in camel-restlet in Camel 2.14.0 and 2.14.1. We have reported this to the Restlet team in issue 996. To remedy the issue then from Camel 2.14.2 onwards you can set synchronous=true as option on the endpoint uris, Or set it on the RestletComponent as a global option so all endpoints inherit this option.

Options

Name	Default Value	Description
`headerFilterStrategy=#`*refName*	An instance of `RestletHeaderFilterStrategy`	Use the # notation (`headerFilterStrategy=#`*refName*) to reference a header filter strategy in the Camel Registry. The strategy will be plugged into the restlet binding if it is `HeaderFilterStrategyAware`.
`restletBinding=#`*refName*	An instance of `DefaultRestletBinding`	The bean ID of a `RestletBinding` object in the Camel Registry.
`restletMethod`	`GET`	On a producer endpoint, specifies the request method to use. On a consumer endpoint, specifies that the endpoint consumes only `restletMethod` requests. The string value is converted to org.restlet.data.Method by the `Method.valueOf(String)` method.
`restletMethods`	*None*	**Consumer only** Specify one or more methods separated by commas (e.g. `restletMethods=post,put`) to be serviced by a restlet consumer endpoint. If both `restletMethod` and `restletMethods` options are specified, the `restletMethod` setting is ignored.
`restletRealm=#`*refName*	`null`	The bean ID of the Realm Map in the Camel Registry.
`restletUriPatterns=#`*refName*	*None*	**Consumer only** Specify one ore more URI templates to be serviced by a restlet consumer endpoint, using the # notation to reference a `List<String>` in the Camel Registry. If a URI pattern has been defined in the endpoint URI, both the URI pattern defined in the endpoint and the `restletUriPatterns` option will be honored.

`throwExceptionOnFailure` (**2.6 or later**)	`true`	*Producer only * Throws exception on a producer failure.
`connectionTimeout`	300000	**Since Camel 2.12.3 Producer only** The Client will give up connection if the connection is timeout, 0 for unlimited wait.
`socketTimeout`	300000	**Since Camel 2.12.3 Producer only** The Client socket receive timeout, 0 for unlimited wait.
`disableStreamCache`	`false`	**Camel 2.14:** Determines whether or not the raw input stream from Jetty is cached or not (Camel will read the stream into a in memory/overflow to file, Stream caching) cache. By default Camel will cache the Jetty input stream to support reading it multiple times to ensure it Camel can retrieve all data from the stream. However you can set this option to `true` when you for example need to access the raw stream, such as streaming it directly to a file or other persistent store. DefaultRestletBinding will copy the request input stream into a stream cache and put it into message body if this option is `false` to support reading the stream multiple times.
`streamRepresentation`	`false`	**Camel 2.16.4/2.17.2: Producer only** Whether to support stream representation as response from calling a REST service using the restlet producer. If the response is streaming then this option can be enabled to use an java.io.InputStream as the message body on the Camel Message body. If using this option you may want to enable the autoCloseStream option as well to ensure the input stream is closed when the Camel Exchange is done being routed. However if you need to read the stream outside a Camel route, you may need to not auto close the stream.
`autoCloseStream`	`false`	**Camel 2.16.4/2.17.2: Producer only** Whether to auto close the stream representation as response from calling a REST service using the restlet producer. If the response is streaming and the option streamRepresentation is enabled then you may want to auto close the InputStream from the streaming response to ensure the input stream is closed when the Camel Exchange is done being routed. However if you need to read the stream outside a Camel route, you may need to not auto close the stream.
`cookieHandler`	`null`	**Camel 2.19: Producer only:** Configure a cookie handler to maintain a HTTP session

Component Options

The Restlet component can be configured with the following options. Notice these are **component** options and cannot be configured on the endpoint, see further below for an example.

Name	Default Value	Description
`controllerDaemon`	`true`	**Camel 2.10:** Indicates if the controller thread should be a daemon (not blocking JVM exit).
`controllerSleepTimeMs`	`100`	**Camel 2.10:** Time for the controller thread to sleep between each control.
`inboundBufferSize`	`8192`	**Camel 2.10:** The size of the buffer when reading messages.
`minThreads`	`1`	**Camel 2.10:** Minimum threads waiting to service requests.
`maxThreads`	`10`	**Camel 2.10:** Maximum threads that will service requests.
`lowThreads`	`8`	**Camel 2.13:** Number of worker threads determining when the connector is considered overloaded.
`maxQueued`	`0`	**Camel 2.13:** Maximum number of calls that can be queued if there aren't any worker thread available to service them. If the value is '0', then no queue is used and calls are rejected if no worker thread is immediately available. If the value is '-1', then an unbounded queue is used and calls are never rejected.
`maxConnectionsPerHost`	`-1`	**Camel 2.10:** Maximum number of concurrent connections per host (IP address).

`maxTotalConnections`	`-1`	**Camel 2.10:** Maximum number of concurrent connections in total.
`outboundBufferSize`	`8192`	**Camel 2.10:** The size of the buffer when writing messages.
`persistingConnections`	`true`	**Camel 2.10:** Indicates if connections should be kept alive after a call.
`pipeliningConnections`	`false`	**Camel 2.10:** Indicates if pipelining connections are supported.
`threadMaxIdleTimeMs`	`60000`	**Camel 2.10:** Time for an idle thread to wait for an operation before being collected.
`useForwardedForHeader`	`false`	**Camel 2.10:** Lookup the "X-Forwarded-For" header supported by popular proxies and caches and uses it to populate the Request.getClientAddresses() method result. This information is only safe for intermediary components within your local network. Other addresses could easily be changed by setting a fake header and should not be trusted for serious security checks.
`reuseAddress`	`true`	**Camel 2.10.5/2.11.1:** Enable/disable the SO_REUSEADDR socket option. See java.io.ServerSocket#reuseAddress property for additional details.
`disableStreamCache`	`false`	**Camel 2.14:** Determines whether or not the raw input stream from Jetty is cached or not (Camel will read the stream into a in memory/overflow to file, Stream caching) cache. By default Camel will cache the Jetty input stream to support reading it multiple times to ensure it Camel can retrieve all data from the stream. However you can set this option to `true` when you for example need to access the raw stream, such as streaming it directly to a file or other persistent store. DefaultRestletBinding will copy the request input stream into a stream cache and put it into message body if this option is `false` to support reading the stream multiple times.
enabledConverters	null	**Camel 2.18:** By default, Restlet engine loads all the extension it finds at run-time and this option filter out all the converters except those explicit listed using full qualified class name or simple class name. i.e. by setting enabledConverters=JacksonConverter, GsonConverter the RestletComponent will remove all the converters loaded by the Restlet engine except Jackson and Gson. Note that you still need to add the extensions as dependency.

Message Headers

confluenceTableSmall

Name	Type	Description
`Content-Type`	`String`	Specifies the content type, which can be set on the OUT message by the application/processor. The value is the `content-type` of the response message. If this header is not set, the content type is based on the object type of the OUT message body. In Camel 2.3 onward, if the Content-Type header is specified in the Camel IN message, the value of the header determine the content type for the Restlet request message. Otherwise, it is defaulted to "application/x-www-form-urlencoded'. Prior to release 2.3, it is not possible to change the request content type default.
`CamelAcceptContentType`	`String`	**Since Camel 2.9.3, 2.10.0:** The HTTP Accept request header.
`CamelHttpMethod`	`String`	The HTTP request method. This is set in the IN message header.
`CamelHttpQuery`	`String`	The query string of the request URI. It is set on the IN message by `DefaultRestletBinding` when the restlet component receives a request.
`CamelHttpResponseCode`	`String` or `Integer`	The response code can be set on the OUT message by the application/processor. The value is the response code of the response message. If this header is not set, the response code is set by the restlet runtime engine.
`CamelHttpUri`	`String`	The HTTP request URI. This is set in the IN message header.
`CamelRestletLogin`	`String`	Login name for basic authentication. It is set on the IN message by the application and gets filtered before the restlet request header by Camel.

`CamelRestletPassword`	`String`	Password name for basic authentication. It is set on the IN message by the application and gets filtered before the restlet request header by Camel.
`CamelRestletRequest`	`Request`	**Camel 2.8:** The `org.restlet.Request` object which holds all request details.
`CamelRestletResponse`	`Response`	**Camel 2.8:** The `org.restlet.Response` object. You can use this to create responses using the API from Restlet. See examples below.
`org.restlet.*`		Attributes of a Restlet message that get propagated to Camel IN headers.
`cache-control`	`String` or `List<CacheDirective>`	**Camel 2.11:** User can set the cache-control with the String value or the List of CacheDirective of Restlet from the camel message header.

Message Body

Camel will store the restlet response from the external server on the OUT body. All headers from the IN message will be copied to the OUT message, so that headers are preserved during routing.

Samples

Restlet Endpoint with Authentication

The following route starts a `restlet` consumer endpoint that listens for `POST` requests on http://localhost:8080. The processor creates a response that echoes the request body and the value of the `id` header.{snippet:id=consumer_route|lang=java|url=camel/trunk/components/camel-restlet/src/test/java/org/apache/camel/component/restlet/route/TestRouteBuilder.java}The `restletRealm` setting in the URI query is used to look up a Realm Map in the registry. If this option is specified, the restlet consumer uses the information to authenticate user logins. Only *authenticated* requests can access the resources. In this sample, we create a Spring application context that serves as a registry. The bean ID of the Realm Map should match the *restletRealmRef*.{snippet:id=realm|lang=xml|url=camel/trunk/components/camel-restlet/src/test/resources/org/apache/camel/component/restlet/camel-context.xml}The following sample starts a `direct` endpoint that sends requests to the server on http://localhost:8080 (that is, our restlet consumer endpoint).{snippet:id=producer_route|lang=java|url=camel/trunk/components/camel-restlet/src/test/java/org/apache/camel/component/restlet/route/TestRouteBuilder.java}That is all we need. We are ready to send a request and try out the restlet component:{snippet:id=auth_request|lang=java|url=camel/trunk/components/camel-restlet/src/test/java/org/apache/camel/component/restlet/RestletRouteBuilderAuthTest.java}The sample client sends a request to the `direct:start-auth` endpoint with the following headers:

- `CamelRestletLogin` (used internally by Camel)
- `CamelRestletPassword` (used internally by Camel)
- `id` (application header)

Note
`org.apache.camel.restlet.auth.login` and `org.apache.camel.restlet.auth.password` will not be propagated as Restlet header.

The sample client gets a response like the following:

textreceived [<order foo='1'/>] as an order id = 89531

Single restlet endpoint to service multiple methods and URI templates

It is possible to create a single route to service multiple HTTP methods using the `restletMethods` option. This snippet also shows how to retrieve the request method from the header:{snippet:id=routeDefinition|lang=java|url=camel/trunk/components/camel-restlet/src/test/java/org/apache/camel/component/restlet/RestletMultiMethodsEndpointTest.java}In addition to servicing multiple methods, the next snippet shows how to create an endpoint that supports multiple URI templates using the `restletUriPatterns` option. The request URI is available in the header of the IN message as well. If a URI pattern has been defined in the endpoint URI (which is not the case in this sample), both the URI pattern defined in the endpoint and the `restletUriPatterns` option will be honored.{snippet:id=routeDefinition|lang=java|url=camel/trunk/components/camel-restlet/src/test/java/org/apache/camel/component/restlet/RestletMultiUriTemplatesEndpointTest.java}The `restletUriPatterns=#uriTemplates` option references the `List<String>` bean defined in the Spring XML configuration.

xml<util:list id="uriTemplates"> <value>/users/{username}</value> <value>/atom/collection/{id}/component/{cid}</value> </util:list>

Using Restlet API to populate response

Available as of Camel 2.8

You may want to use the `org.restlet.Response` API to populate the response. This gives you full access to the Restlet API and fine grained control of the response. See the route snippet below where we generate the response from an inlined Camel Processor:{snippet:id=e1|title=Generating response using Restlet Response API|lang=java|url=camel/trunk/components/camel-restlet/src/test/java/org/apache/camel/component/restlet/RestletRequestAndResponseAPITest.java}

Configuring max threads on component

To configure the max threads options you must do this on the component, such as:

xml<bean id="restlet" class="org.apache.camel.component.restlet.RestletComponent"> <property name="maxThreads" value="100"/> </bean>

Using the Restlet servlet within a webapp

Available as of Camel 2.8
There are three possible ways to configure a Restlet application within a servlet container and using the subclassed SpringServerServlet enables configuration within Camel by injecting the Restlet Component.

Use of the Restlet servlet within a servlet container enables routes to be configured with relative paths in URIs (removing the restrictions of hard-coded absolute URIs) and for the hosting servlet container to handle incoming requests (rather than have to spawn a separate server process on a new port).

To configure, add the following to your camel-context.xml;

xml<camelContext> <route id="RS_RestletDemo"> <from uri="restlet:/demo/{id}" /> <transform> <simple>Request type : ${header.CamelHttpMethod} and ID : ${header.id}</simple> </transform> </route> </camelContext> <bean id="RestletComponent" class="org.restlet.Component" /> <bean id="RestletComponentService" class="org.apache.camel.component.restlet.RestletComponent"> <constructor-arg index="0"> <ref bean="RestletComponent" /> </constructor-arg> </bean>

And add this to your web.xml;

xml<!-- Restlet Servlet --> <servlet> <servlet-name>RestletServlet</servlet-name> <servlet-class>org.restlet.ext.spring.SpringServerServlet</servlet-class> <init-param> <param-name>org.restlet.component</param-name> <param-value>RestletComponent</param-value> </init-param> </servlet> <servlet-mapping> <servlet-name>RestletServlet</servlet-name> <url-pattern>/rs/*</url-pattern> </servlet-mapping>

You will then be able to access the deployed route at http://localhost:8080/mywebapp/rs/demo/1234 where;

localhost:8080 is the server and port of your servlet container
mywebapp is the name of your deployed webapp
Your browser will then show the following content;

"Request type : GET and ID : 1234"

You will need to add dependency on the Spring extension to restlet which you can do in your Maven pom.xml file:

xml<dependency> <groupId>org.restlet.jee</groupId> <artifactId>org.restlet.ext.spring</artifactId> <version>${restlet-version}</version> </dependency>

And you would need to add dependency on the restlet maven repository as well:

xml<repository> <id>maven-restlet</id> <name>Public online Restlet repository</name> <url>http://maven.restlet.org</url> </repository>

Endpoint See Also