SERVLET

Servlet Component

The **servlet:** component provides HTTP based endpoints for consuming HTTP requests that arrive at a HTTP endpoint that is bound to a published Servlet.

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-servlet</artifactId> <version>x.x.x</version> <\!-\- use the same version as your Camel core version \--> </dependency> Stream
Servlet is stream based, which means the input it receives is submitted to Camel as a stream. That means you will only be able to read the content of the stream **once**. If you find a situation where the message body appears to be empty or you need to access the data multiple times (eg: doing multicasting, or redelivery error handling) you should use Stream caching or convert the message body to a `String` which is safe to be read multiple times.

URI format

servlet://relative_path[?options]

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

Options

confluenceTableSmall

Name	Default Value	Description
`httpBindingRef`	`null`	Reference to an `org.apache.camel.component.http.HttpBinding` in the Registry. A `HttpBinding` implementation can be used to customize how to write a response.
`httpBinding`	`null`	**Camel 2.16:** Reference to an `org.apache.camel.component.http.HttpBinding` in the Registry. A `HttpBinding` implementation can be used to customize how to write a response.
`matchOnUriPrefix`	`false`	Whether or not the `CamelServlet` should try to find a target consumer by matching the URI prefix, if no exact match is found.
`servletName`	`CamelServlet`	Specifies the servlet name that the servlet endpoint will bind to. This name should match the name you define in `web.xml` file.
`httpMethodRestrict`	`null`	**Camel 2.11: Consumer only**: Used to only allow consuming if the HttpMethod matches, such as GET/POST/PUT etc. From **Camel 2.15** onwards multiple methods can be specified separated by comma.

Message Headers

Camel will apply the same Message Headers as the HTTP component.

Camel will also populate **all** `request.parameter` and `request.headers`. For example, if a client request has the URL, `http://myserver/myserver?orderid=123`, the exchange will contain a header named `orderid` with the value 123.

Usage

You can consume only from endpoints generated by the Servlet component. Therefore, it should be used only as input into your Camel routes. To issue HTTP requests against other HTTP endpoints, use the HTTP Component

Using Servlet 3.0 Async Mode

Available as of Camel 2.18

You can configure the servlet with an init-param to turn on async mode when using a Servlet 3.x container. There is a sample XML configuration below:

xml <servlet> <servlet-name>CamelServlet</servlet-name> <display-name>Camel Http Transport Servlet</display-name> <servlet-class>org.apache.camel.component.servlet.CamelHttpTransportServlet</servlet-class> <init-param> <param-name>async</param-name> <param-value>true</param-value> </init-param> <load-on-startup>1</load-on-startup> <async-supported>true</async-supported> </servlet>

Putting Camel JARs in the app server boot classpath

If you put the Camel JARs such as `camel-core`, `camel-servlet`, etc. in the boot classpath of your application server (eg usually in its lib directory), then mind that the servlet mapping list is now shared between multiple deployed Camel application in the app server.

Mind that putting Camel JARs in the boot classpath of the application server is generally not best practice!

So in those situations you **must** define a custom and unique servlet name in each of your Camel application, eg in the `web.xml` define:

xml<servlet> <servlet-name>MyServlet</servlet-name> <servlet-class>org.apache.camel.component.servlet.CamelHttpTransportServlet</servlet-class> <load-on-startup>1</load-on-startup> </servlet> <servlet-mapping> <servlet-name>MyServlet</servlet-name> <url-pattern>/*</url-pattern> </servlet-mapping>

And in your Camel endpoints then include the servlet name as well

xml<route> <from uri="servlet://foo?servletName=MyServlet"/> ... </route>

From **Camel 2.11** onwards Camel will detect this duplicate and fail to start the application. You can control to ignore this duplicate by setting the servlet init-parameter ignoreDuplicateServletName to true as follows:

xml <servlet> <servlet-name>CamelServlet</servlet-name> <display-name>Camel Http Transport Servlet</display-name> <servlet-class>org.apache.camel.component.servlet.CamelHttpTransportServlet</servlet-class> <init-param> <param-name>ignoreDuplicateServletName</param-name> <param-value>true</param-value> </init-param> </servlet>

But its **strongly advised** to use unique servlet-name for each Camel application to avoid this duplication clash, as well any unforeseen side-effects.

Sample

From Camel 2.7 onwards it's easier to use Servlet in Spring web applications. See Servlet Tomcat Example for details.

In this sample, we define a route that exposes a HTTP service at `http://localhost:8080/camel/services/hello`.
First, you need to publish the CamelHttpTransportServlet through the normal Web Container, or OSGi Service.
Use the `web.xml` file to publish the CamelHttpTransportServlet as follows:{snippet:id=web|lang=xml|url=camel/trunk/components/camel-servlet/src/test/resources/org/apache/camel/component/servlet/web.xml}Then you can define your route as follows:{snippet:id=route|lang=java|url=camel/trunk/components/camel-servlet/src/test/java/org/apache/camel/component/servlet/HttpClientRouteTest.java}

Specify the relative path for camel-servlet endpoint
Since we are binding the Http transport with a published servlet, and we don't know the servlet's application context path, the `camel-servlet` endpoint uses the relative path to specify the endpoint's URL. A client can access the `camel-servlet` endpoint through the servlet publish address: `("http://localhost:8080/camel/services") + RELATIVE_PATH("/hello")`.

Sample when using Spring 3.x

See Servlet Tomcat Example

Sample when using Spring 2.x

When using the Servlet component in a Camel/Spring application it's often required to load the Spring ApplicationContext *after* the Servlet component has started. This can be accomplished by using Spring's `ContextLoaderServlet` instead of `ContextLoaderListener`. In that case you'll need to start `ContextLoaderServlet` after CamelHttpTransportServlet like this:

xml <web-app> <servlet> <servlet-name>CamelServlet</servlet-name> <servlet-class> org.apache.camel.component.servlet.CamelHttpTransportServlet </servlet-class> <load-on-startup>1</load-on-startup> </servlet> <servlet> <servlet-name>SpringApplicationContext</servlet-name> <servlet-class> org.springframework.web.context.ContextLoaderServlet </servlet-class> <load-on-startup>2</load-on-startup> </servlet> <web-app>

Sample when using OSGi

From **Camel 2.6.0**, you can publish the CamelHttpTransportServlet as an OSGi service with help of SpringDM like this.{snippet:id=service|lang=xml|url=camel/trunk/tests/camel-itest-osgi/src/test/resources/org/apache/camel/itest/osgi/servlet/ServletServiceContext.xml}Then use this service in your camel route like this:{snippet:id=camelContext|lang=xml|url=camel/trunk/tests/camel-itest-osgi/src/test/resources/org/apache/camel/itest/osgi/servlet/CamelServletWithServletServiceContext.xml}For versions prior to Camel 2.6 you can use an `Activator` to publish the CamelHttpTransportServlet on the OSGi platform{snippet:id=activator|lang=java|url=camel/trunk/tests/camel-itest-osgi/src/test/java/org/apache/camel/itest/osgi/servlet/support/ServletActivator.java}Endpoint See Also

- Servlet Tomcat Example
- Servlet Tomcat No Spring Example
- HTTP
- Jetty