REST Swagger Component

Available from Camel 2.19

The **rest-swagger** configures rest producers from Swagger (Open API) specification document and delegates to a component implementing the *RestProducerFactory* interface. Currently known working components are:

- HTTP component
- HTTP4 component
- Netty4 component
- Restlet component
- Jetty component
- Undertow 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-rest-swagger</artifactId>
    <version>x.x.x</version>
    <!-- use the same version as your Camel core version -->
</dependency>
```

URI format

```
rest-swagger:[specificationPath#]operationId
```

Where **operationId** is the ID of the operation in the Swagger specification, and **specificationPath** is the path to the specification.
If the `specificationPath` is not specified it defaults to **swagger.json**. The lookup mechanism uses Camel's *ResourceHelper* to load the resource, which means that you can use CLASSPATH resources (*classpath:my-specification.json*), files (*file:/some/path.json*), the web (*http://api.example.com/swagger.json)* or reference a bean (*ref:nameOfBean*) or use a method of a bean (*bean:nameOfBean.methodName*) to get the specification resource, failing that Swagger's own resource loading support.

This component does not act as a HTTP client, it delegates that to another component mentioned above. The lookup mechanism searches for a single component that implements the *RestProducerFactory* interface and uses that. If the CLASSPATH contains more than one, then the property **componentName** should be set to indicate which component to delegate to.

Most of the configuration is taken from the Swagger specification but the option exists to override those by specifying them on the component or on the endpoint. Typically you would just need to override the **host** or **basePath** if those differ from the specification.

NOTE: The **host** parameter should contain the absolute URI containing scheme, hostname and port number, for instance *https://api.example.com*.

With **componentName** you specify what component is used to perform the requests, this named component needs to be present in the Camel context and implement the required *RestProducerFactory* interface - as do the components listed at the top.

If you do not specify the **componentName** at either component or endpoint level, CLASSPATH is searched for a suitable delegate. There should be only one component present on the CLASSPATH that implements the *RestProducerFactory* interface for this to work.

Name	Default Value	Description
basePath	""	API basePath for example /v2. Default is unset if set overrides the value present in Swagger specification.
componentName	null	Name of the Camel component that will perform the requests. The compnent must be present in Camel registry and it must implement RestProducerFactory service provider interface. If not set CLASSPATH is searched for single component that implements RestProducerFactory SPI. Can be overriden in endpoint configuration.
consumes	null	What payload type this component capable of consuming. Could be one type like application/json or multiple types as application /json application/xml; q=0.5 according to the RFC7231. This equates to the value of Accept HTTP header. If set overrides any value found in the Swagger specification. Can be overriden in endpoint configuration
host	null	Scheme hostname and port to direct the HTTP requests to in the form of http[s]://hostname:port. Can be configured at the endpoint component or in the correspoding REST configuration in the Camel Context. If you give this component a name (e.g. petstore) that REST configuration is consulted first rest-swagger next and global configuration last. If set overrides any value found in the Swagger specification RestConfiguration. Can be overriden in endpoint configuration.

produces	null	What payload type this component is producing. For example application/json according to the RFC7231. This equates to the value of Content-Type HTTP header. If set overrides any value present in the Swagger specification. Can be overriden in endpoint configuration.
specificationUri	"swagger.json"	Path to the Swagger specification file. The scheme host base path are taken from this specification but these can be overriden with properties on the component or endpoint level. If not given the component tries to load swagger.json resource. Can be overriden in endpoint configuration.
operationId		**Endpoint only**, the ID of the operation from the Swagger specification.

Example: PetStore

Checkout the example in the `camel-example-rest-swagger` project in the `examples` directory.

For example if you wanted to use the PetStore provided REST API simply reference the specification URI and desired operation id from the Swagger specification or download the specification and store it as `swagger.json` (in the root) of CLASSPATH that way it will be automaticaly used. Let's use the Undertow component to perform all the requests and Camels excelent support for Spring Boot.

Here are our dependencies defined in Maven POM file:

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-undertow-starter</artifactId>
</dependency>

<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-rest-swagger-starter</artifactId>
</dependency>
```

Start by defining the *Undertow* component and the *RestSwaggerComponent*:

```
@Bean
public Component petstore(CamelContext camelContext, UndertowComponent undertow) {
    RestSwaggerComponent petstore = new RestSwaggerComponent(camelContext);
    petstore.setSpecificationUri("http://petstore.swagger.io/v2/swagger.json");
    petstore.setDelegate(undertow);

    return petstore;
}
```

NOTE: Support in Camel for Spring Boot will auto create the `UndertowComponent` Spring bean, and you can configure it using `application.properties` (or `application.yml`) using prefix `camel.component.undertow.`. We are defining the `petstore` component here in order to have a named component in the Camel context that we can use to interact with the PetStore REST API, if this is the only `rest-swagger` component used we might configure it in the same manner (using `application.properties`).

Now in our application we can simply use the `ProducerTemplate` to invoke PetStore REST methods:

```
@Autowired
ProducerTemplate template;

String getPetJsonById(int petId) {
    return template.requestBodyAndHeaders("petstore:getPetById", null, "petId", petId);
}
```