User Guide for CamelBee WebGL Application

Contents

Introduction 2
Main Scene 3
Debugger Scene 4
Debugger Scene – Route Caller Panel 6
    Key Features: 6
    Parameter Adjustment: 6
    Flexibility in Tool Usage: 6
Debugger Scene – Message Panel 9
Debugger Scene – Health Panel 11
Debugger Scene – Search Panel 12
Metrics Scene 13
Settings Scene 15

Introduction

This guide provides an overview of how to use the different scenes in the CamelBee WebGL application, each designed for a specific function. The following sections will explain these scenes in detail.

To enable your Camel microservices to interact with the CamelBee WebGL application, the necessary steps for each framework are outlined in the core Java libraries of CamelBee at the following URLs:

For Spring Boot:

https://github.com/egekaraosmanoglu/camelbee/tree/main/core/springboot-core

For Quarkus:

https://github.com/egekaraosmanoglu/camelbee/tree/main/core/quarkus-core

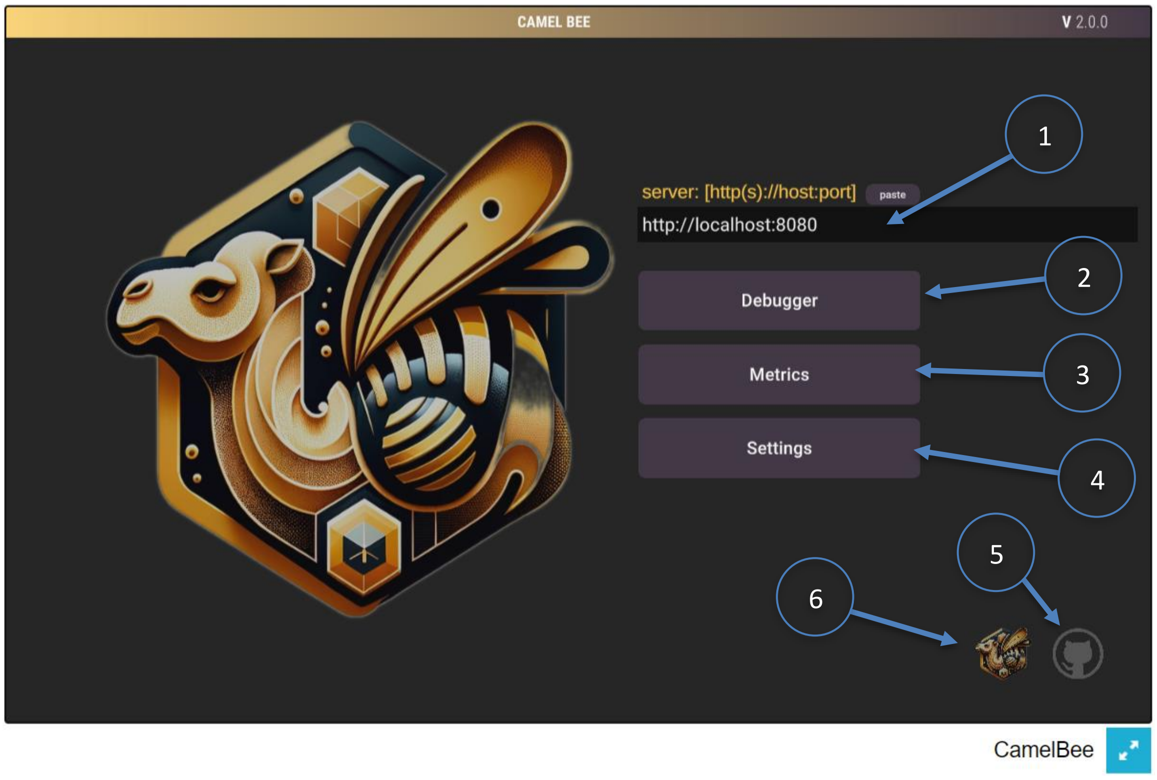
Once you have completed the steps outlined in these links, the CamelBee WebGL application will be able to interact with your microservices.

You can access the CamelBee WebGL application by clicking the **"Launch CamelBee"** link on the website **https://www.camelbee.io** or through the local Docker version accessed at **http://localhost:8083** by executing the following command:

```
docker run -d -p 8083:80 camelbee/webgl
```

Main Scene

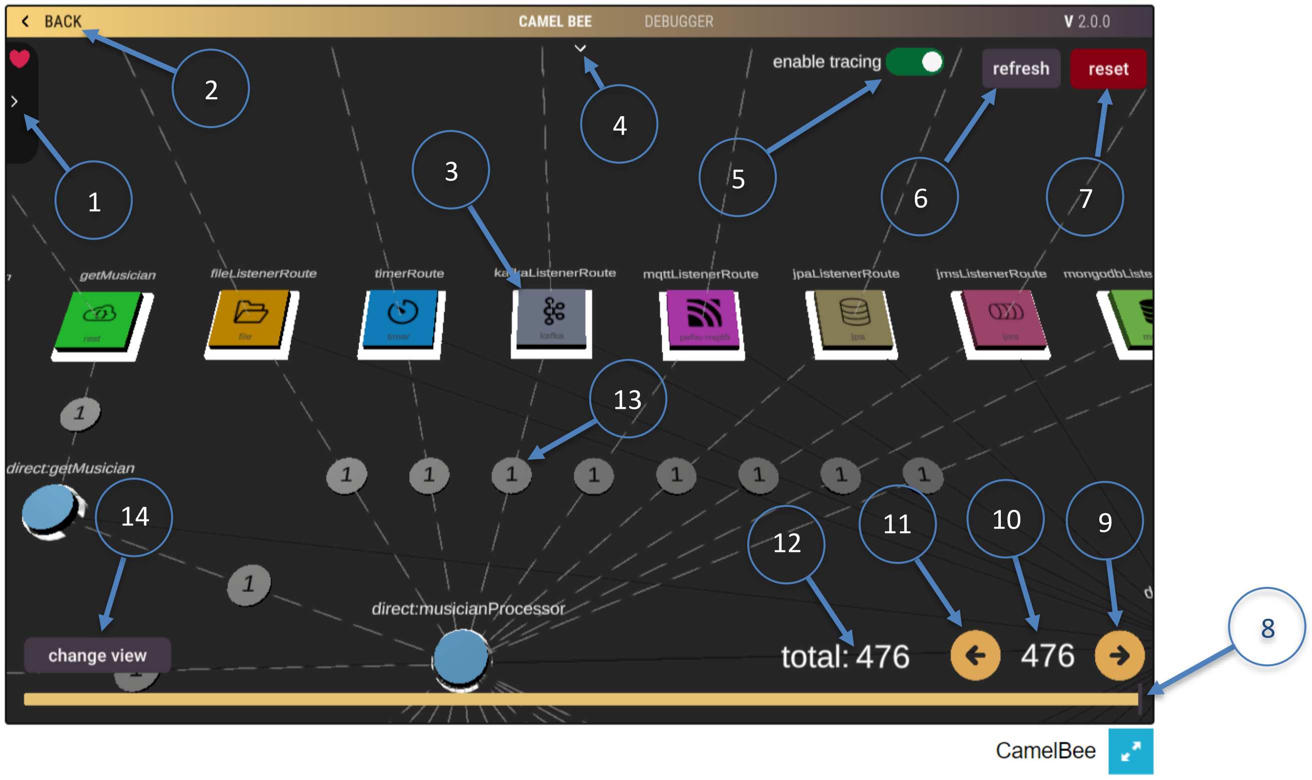
The initial scene is the home page of the CamelBee WebGL application, from which you can navigate to other scenes.



- 1. Text field where you need to enter the host and port of your microservice (http(s)://host:port)
- 2. "Debugger" button, which navigates to the Debugger Scene.
- 3. "Metrics" button, which takes you to the Metrics Scene to monitor your microservice.
- 4. "Settings" button, which takes you to the Settings Scene for updating configurations.
- 5. Button linking to the CamelBee GitHub page: “https://github.com/egekaraosmanoglu/camelbee”.
- 6. Button linking to the CamelBee website: “https://www.camelbee.io”.

Debugger Scene

The Debugger Scene allows you to visualize the 3D topology of your Camel routes, including their connections and the messages exchanged between them. Additionally, it provides the Route Caller panel to trigger consumer routes.



- 1. Health Panel expander button.
- 2. "BACK" button, which navigates to the Main Scene.
- 3. A consumer route that listens to a Kafka topic. You can click on this component to open the Route Caller Panel and trigger the consumer route.
- 4. Search Panel expander button.
- 5. "Enable tracing" toggle button: This allows you to turn message tracing between routes on or off. Once enabled, messages will start appearing in the WebGL application. Keep in mind that all messages exchanged between routes will be collected, so it is not recommended for PRODUCTION use, or at least proceed with caution.
- 6. The "Refresh" button is used to retrieve messages traced and collected by CamelBee tracers. It is recommended to press this button to ensure that you have the most up-to-date collected/traced messages..
- 7. "Reset" button, to delete all collected messages for a fresh start.
- 8. The slider to navigate to a specific point in time (message) within the collected messages.
- 9. "Next" button to move forward one message.

- 10. The current message in the sequence, displayed as the last message shown in the WebGL app.
- 11. "Prev" button to move back one message.
- 12. Total number of messages collected after enabling tracing.
- 13. A component that shows how many messages were exchanged in this specific connection. Clicking on it opens the Messages Panel, where you can view the details of each message within the connection.
- 14. "Change view" button to switch between First-Person View and Top-Level View.

Debugger Scene – Route Caller Panel

The Debugger Scene's RouteCaller Panel serves as a key tool for triggering consumer routes, which are essentially the entry points of your microservice. Normally, depending on the technology your microservice exposes (REST, SOAP, Messaging), you would use specialized tools such as Postman for REST, SoapUI for SOAP, or a Message Broker Client for queue or topic-based microservices.

However, the RouteCaller Panel simplifies this by enabling you to trigger your microservice's consumer routes without needing these external tools. This abstraction works regardless of the underlying interface technology you are using since they are all consumer Camel components. This is achieved because, in the background, the CamelBee core library utilizes the ProducerTemplate class to trigger the corresponding producer Camel component.

Key Features:

- When you click on any consumer route in your service topology, this panel is shown and displays the details of the selected consumer, including the parameters that are used by that route.
- By pressing the "Send" button, a REST resource in the CamelBee Core library (org.camelbee.debugger.controller.ProducerController) is invoked which uses the ProducerTemplate to trigger the associated producer component.

Parameter Adjustment:

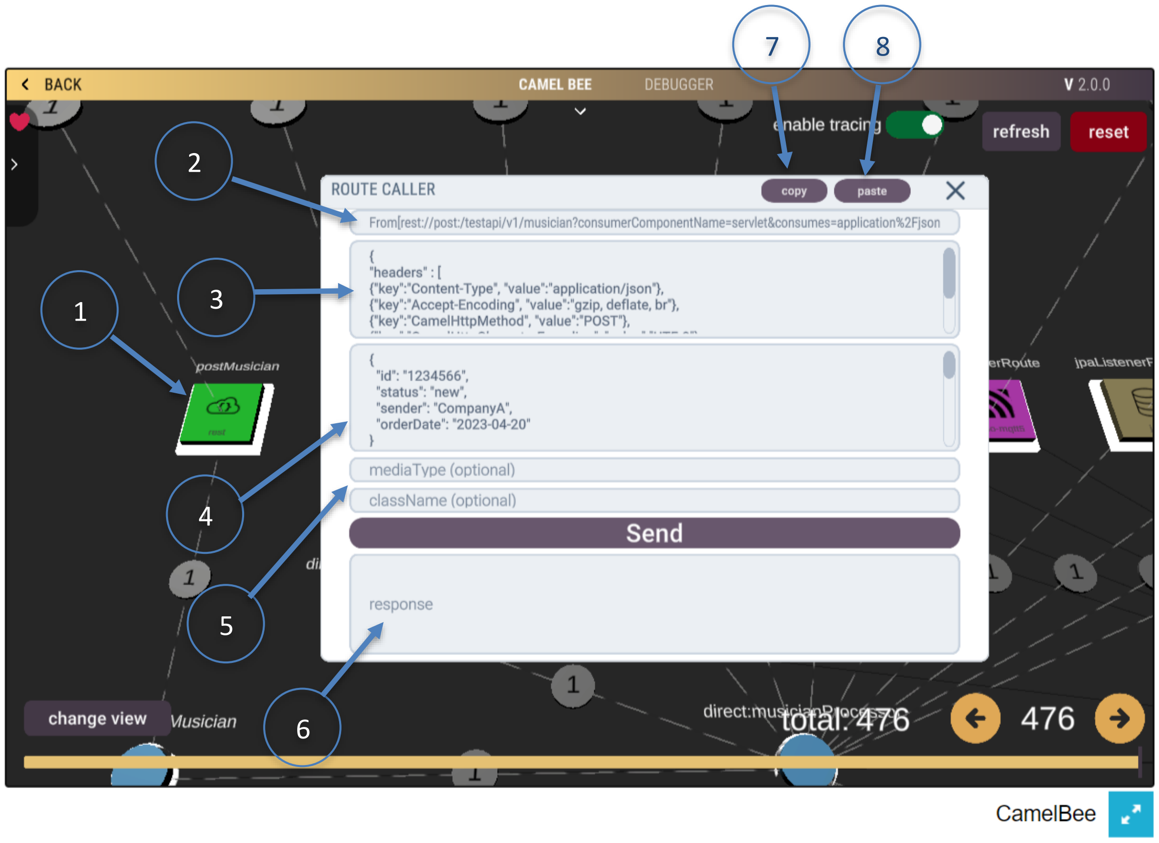
In some cases, the consumer parameters might differ from the ones expected by the producer. If this happens, you can modify or replace those parameters directly in the first text field of the **RouteCaller Panel** to suit the producer's requirements. This ensures smooth communication across the different interfaces and components, regardless of the specific consumer/producer combination.

This approach provides flexibility, allowing developers to test and trigger routes without needing to rely on external tools, streamlining the testing process during development.

Flexibility in Tool Usage:

While the RouteCaller Panel offers an integrated way to trigger consumer routes, you are still free to use your preferred tools like Postman, SoapUI, or Message Broker Clients to interact with your microservice. It doesn’t matter which tool is used, because CamelBee tracers will trace all message exchanges across your microservice, ensuring full visibility and traceability of the communication flow.

This flexibility allows developers to choose whichever tool they are comfortable with while ensuring that all message exchanges are still captured and traced by CamelBee's infrastructure.



```
From[rest://post:/testapi/v1/musician?consumerComponentName=servlet&consumes=application%2Fjson
```

```
{
"headers" : [
{"key":"Content-Type", "value":"application/json"},
{"key":"Accept-Encoding", "value":"gzip, deflate, br"},
{"key":"CamelHttpMethod", "value":"POST"},
```

```
{
 "id": "1234566",
 "status": "new",
 "sender": "CompanyA",
 "orderDate": "2023-04-20"
}
```

mediaType (optional)

className (optional)

Send

response

- 1. Click on any consumer (entrypoint) route and the "Route Caller" panel will appear, displaying the consumer parameters defined in the code for that route. You may need to adjust or remove certain consumer parameters, as they will be used when invoking that component as a producer endpoint.
- 2. Consumer definition for the selected entrypoint route will be shown.
- 3. “Headers” to be sent to the producer component are listed as Exchange Headers and must follow the JSON structure below:

```
{
"headers" : [
{"key":"Content-Type", "value":"application/json"},
{"key":"Accept-Encoding", "value":"gzip, deflate, br"},
{"key":"CamelHttpMethod", "value":"POST"},
{"key":"CamelHttpCharacterEncoding", "value":"UTF-8"},
{"key":"application", "value":"CamelBee"}
]
}
```

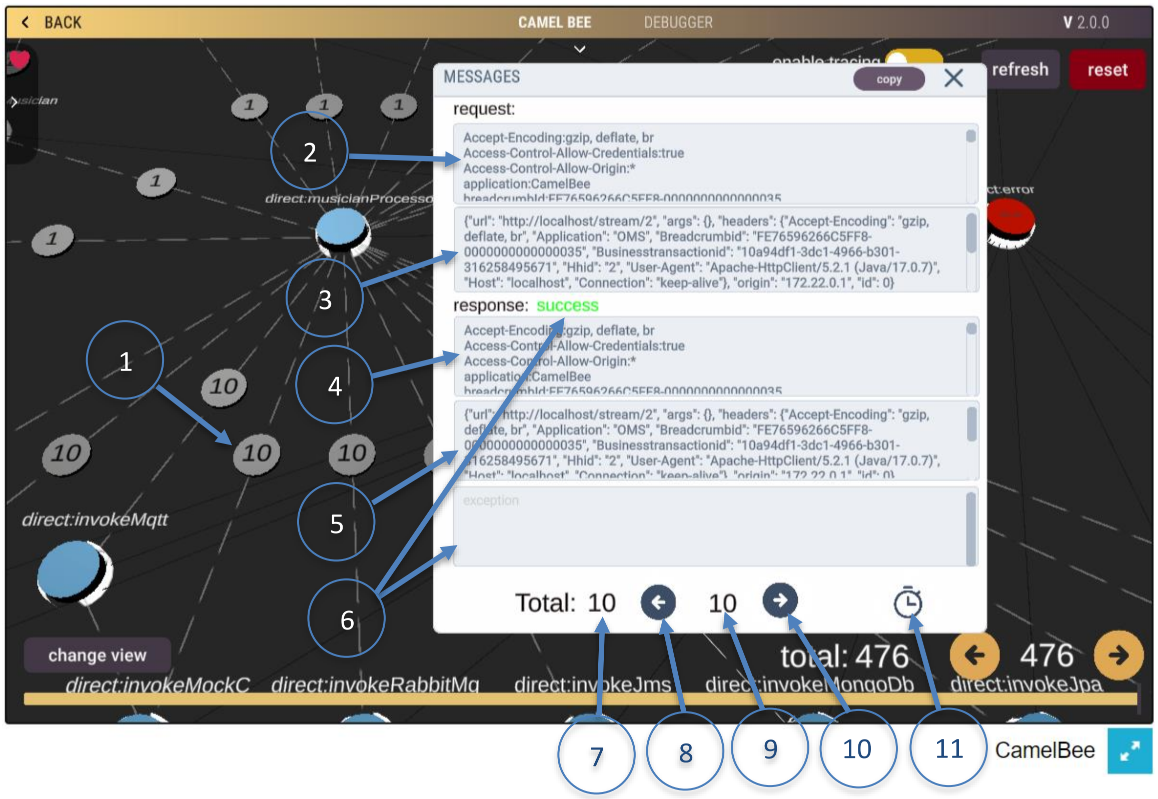
- 4. “Body” to be sent to the entrypoint route can be in JSON, XML, CSV, or plain text format.

- 5. MediaType and ClassName can be specified if you wish to send a Java class instance to the consumer component. In such cases, the body should be a JSON payload, MediaType should be set to "json", and ClassName should include the fully qualified name of the class.
- 6. The response from the REST resource in the CamelBee Core library (org.camelbee.debugger.controller.ProducerController) will be returned after invoking the corresponding producer component via the ProducerTemplate.
- 7. "Copy" button allows you to copy any selected text from the Route Caller panel's text fields. (This feature is provided because the default copy/paste behavior is restricted by browsers for WebGL applications.)
- 8. "Paste" button allows you to paste clipboard data into a selected text field within the Route Caller panel. (This feature is also necessary due to the default browser restrictions on copy/paste actions in WebGL applications.)

Debugger Scene – Message Panel

Clicking on any coin-shaped object in the scene with numbers on them will open the Messages Panel, allowing you to view the details of each message within the connection that the coin-shaped object represents.

In the Messages Panel, you can navigate through messages using the Prev and Next buttons. Additionally, pressing the "timer" icon in the bottom-right corner of the panel will adjust the timeline to make that specific message the last one, and the main slider at the bottom will automatically move to that point in time.



```
Accept-Encoding:gzip, deflate, br
Access-Control-Allow-Credentials:true
Access-Control-Allow-Origin:*
application:CamelBee
breadcrumbId:FE76596266C5FF8-0000000000000035
```

```
{"url": "http://localhost/stream/2", "args": {}, "headers": {"Accept-Encoding": "gzip, deflate, br", "Application": "OMS", "Breadcrumbid": "FE76596266C5FF8-0000000000000035", "Businesstransactionid": "10a94df1-3dc1-4966-b301-316258495671", "Hhid": "2", "User-Agent": "Apache-HttpClient/5.2.1 (Java/17.0.7)", "Host": "localhost", "Connection": "keep-alive"}, "origin": "172.22.0.1", "id": 0}
```

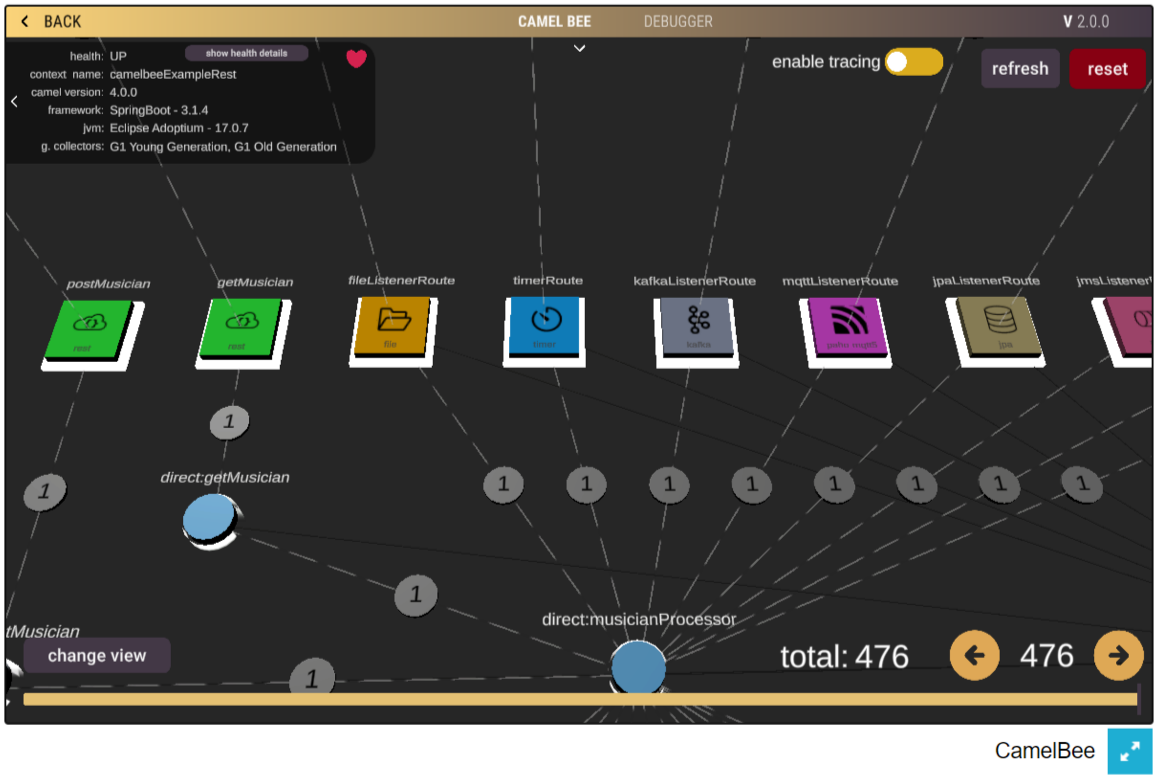
```
Accept-Encoding:gzip, deflate, br
Access-Control-Allow-Credentials:true
Access-Control-Allow-Origin:*
application:CamelBee
breadcrumbId:FE76596266C5FF8-0000000000000035
```

```
{"url": "http://localhost/stream/2", "args": {}, "headers": {"Accept-Encoding": "gzip, deflate, br", "Application": "OMS", "Breadcrumbid": "FE76596266C5FF8-0000000000000035", "Businesstransactionid": "10a94df1-3dc1-4966-b301-316258495671", "Hhid": "2", "User-Agent": "Apache-HttpClient/5.2.1 (Java/17.0.7)", "Host": "localhost", "Connection": "keep-alive"}, "origin": "172.22.0.1", "id": 0}
```

- 1. Clicking on any coin-shaped object in the scene with numbers on it will open the Messages Panel, showing the list of messages for the specific interconnection between Camel routes or endpoints..
- 2. Displays the Request Exchange Headers sent for the interaction within that connection.
- 3. Displays the Request Exchange Body sent for the interaction within that connection.
- 4. Displays the Response Exchange Headers received for the interaction within that connection.
- 5. Displays the Response Exchange Body received for the interaction within that connection.

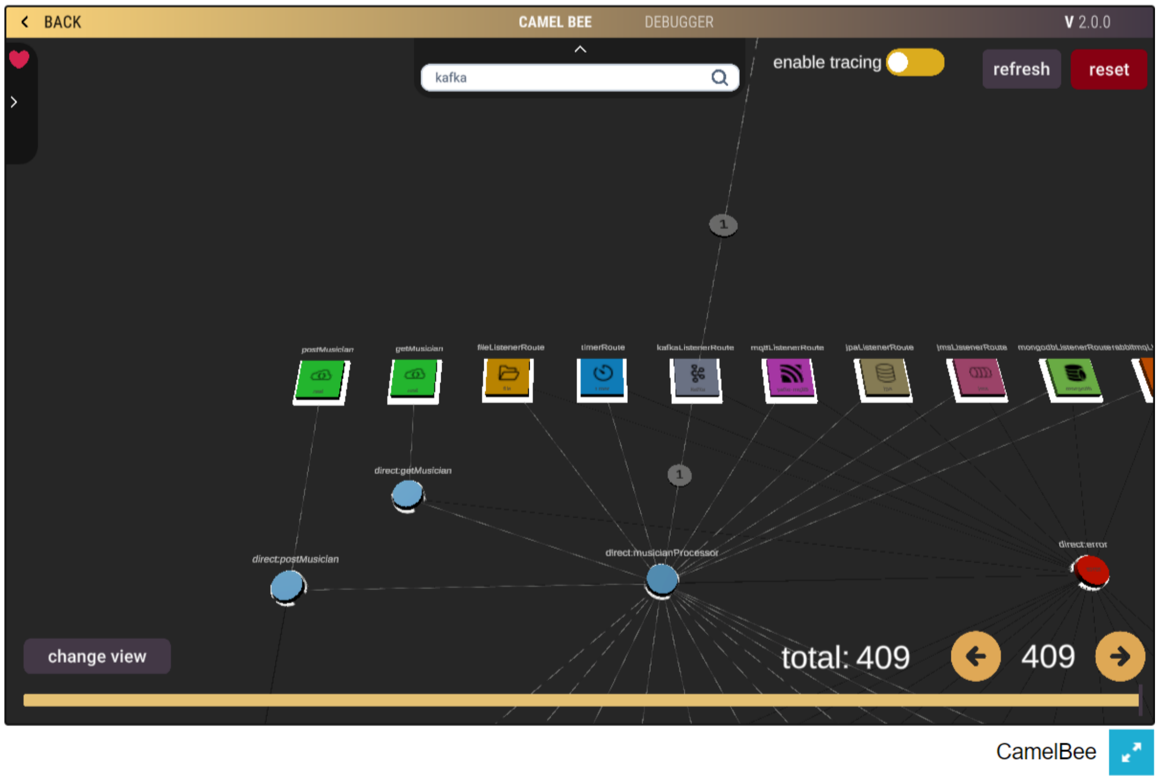
- 6. If an exception occurs, the error message will be displayed here, and 'error' will appear next to the 'response' text in place of 'success'.
- 7. Shows the total number of interactions that occurred within this connection.
- 8. "Prev" button to navigate back to the previous message.
- 9. Displays the current message being viewed in the Messages Panel.
- 10. "Next" button to navigate forward to the next message.
- 11. The "Timer" button lets you jump to the specific time when this interaction occurred on the global timeline.

Debugger Scene – Health Panel


health: UP
context name: camelbeeExampleRest
camel version: 4.0.0
framework: SpringBoot - 3.1.4
jvm: Eclipse Adoptium - 17.0.7
g. collectors: G1 Young Generation, G1 Old Generation

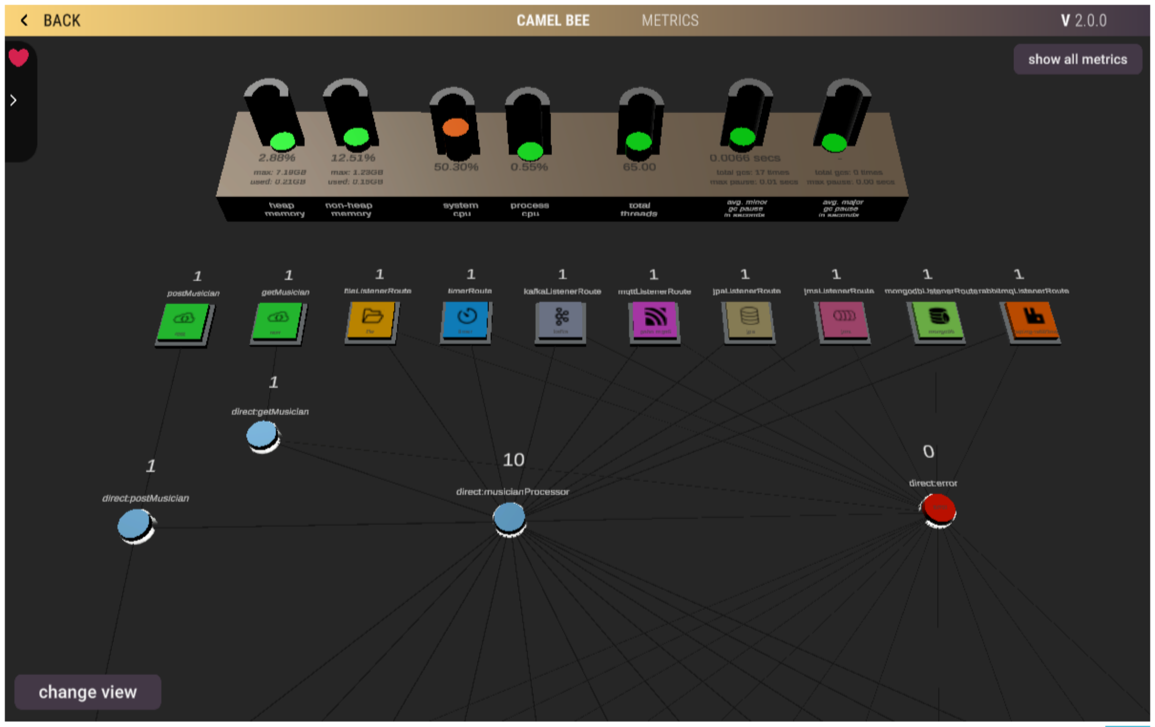
- Coming soon!

Debugger Scene – Search Panel

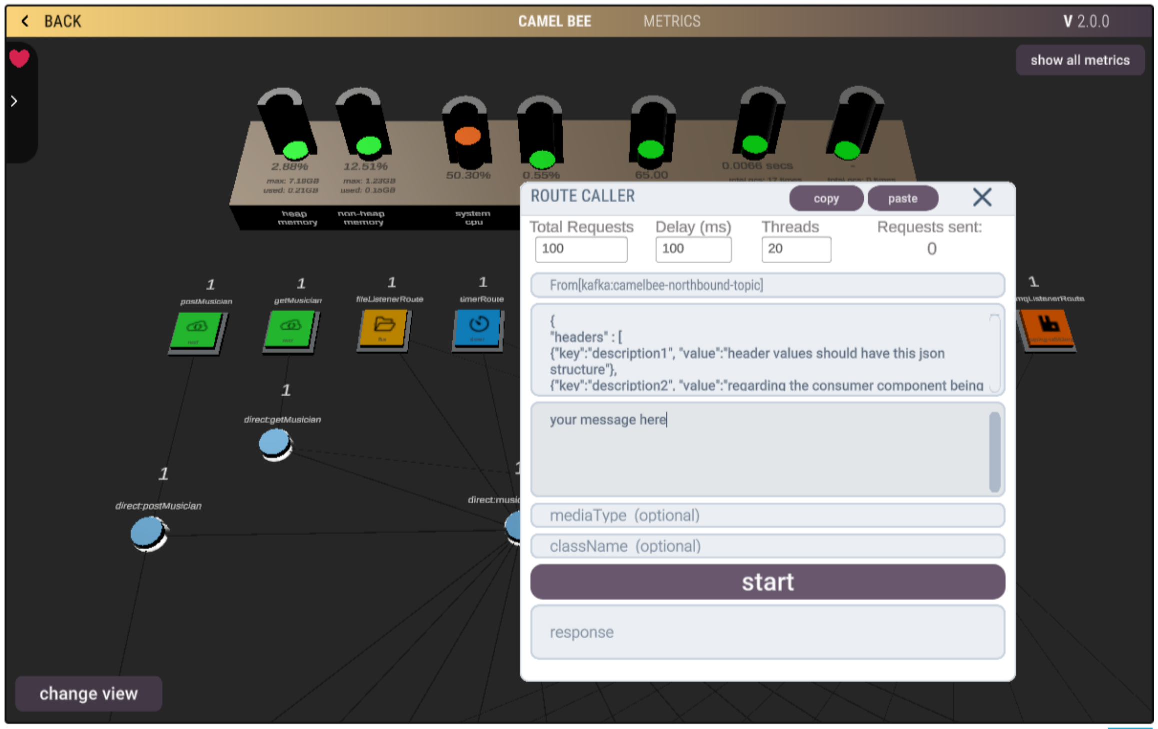


- Coming soon!

Metrics Scene



CamelBee



ROUTE CALLER
copy
paste

Total Requests	Delay (ms)	Threads	Requests sent:
100	100	20	0

From[kafka:camelbee-northbound-topic]

```
{
"headers" : [
{"key":"description1", "value":"header values should have this json structure"},
{"key":"description2", "value":"regarding the consumer component being
```

your message here

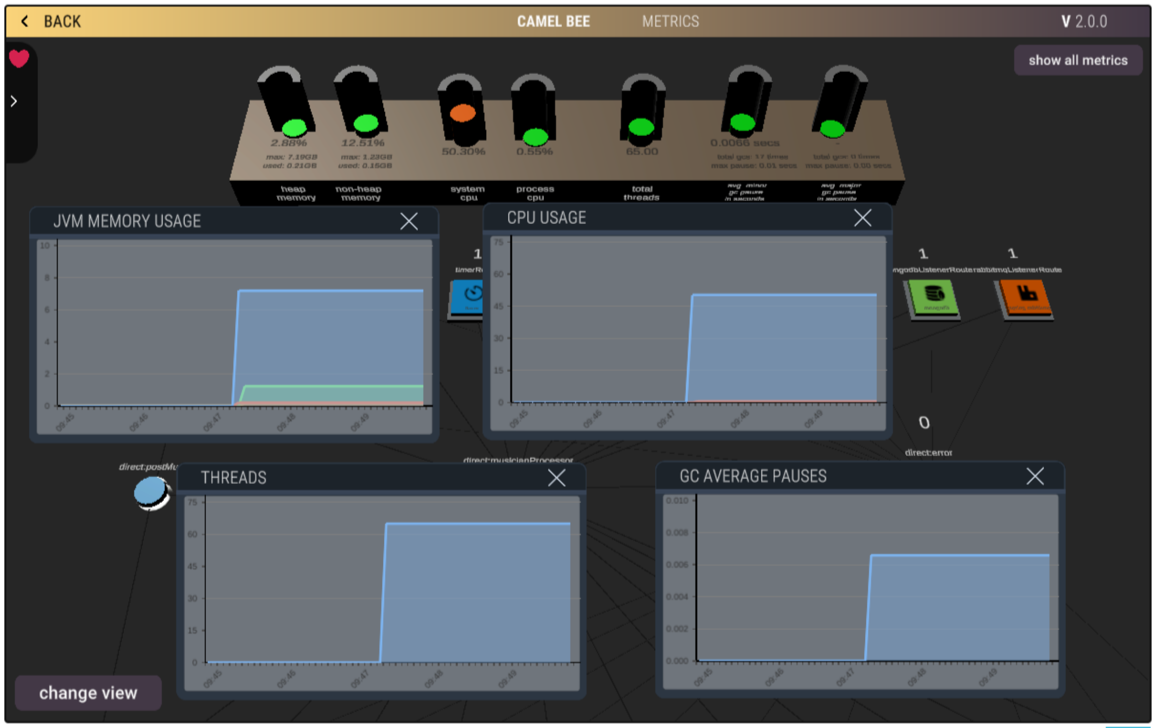
mediaType (optional)

className (optional)

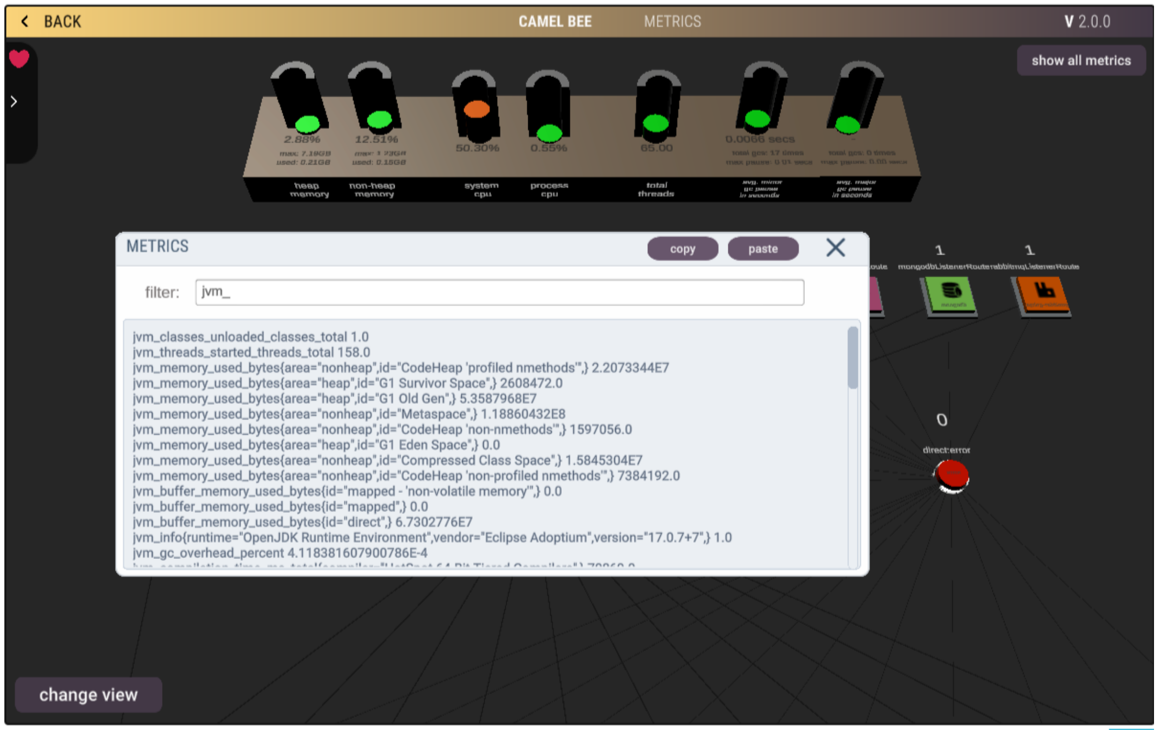
start

response

CamelBee



CamelBee

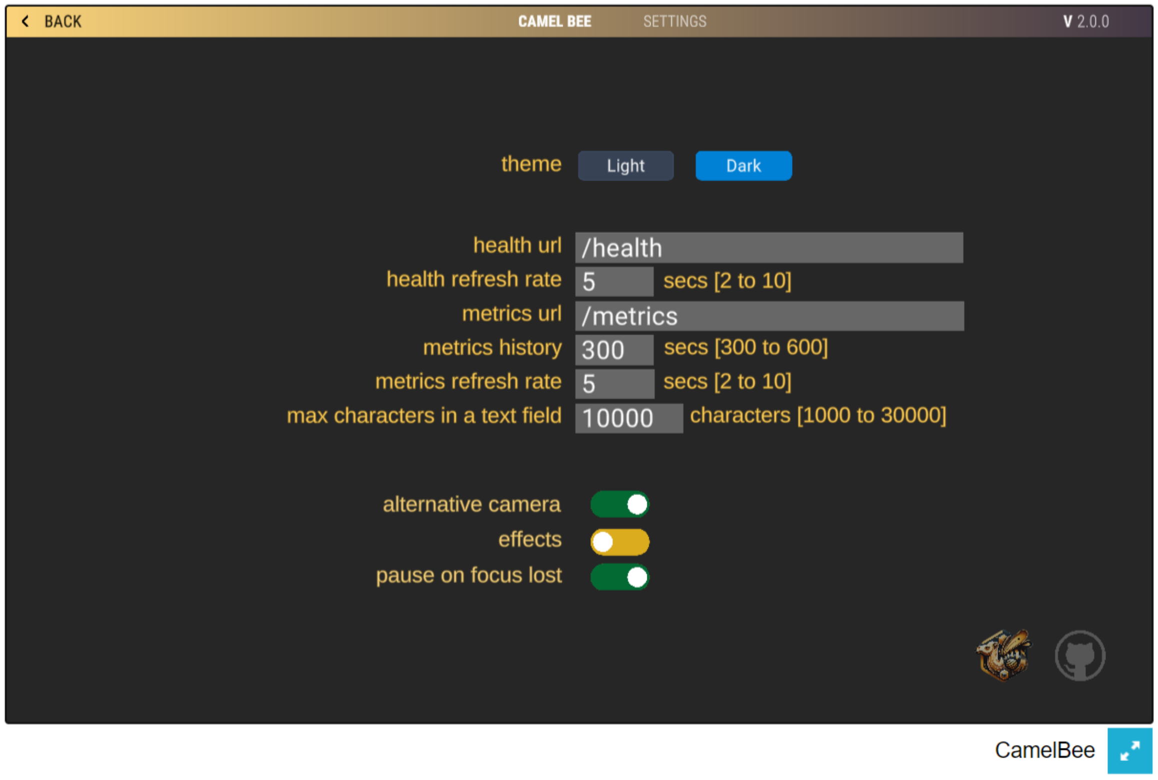


```
jvm_classes_unloaded_classes_total 1.0
jvm_threads_started_threads_total 158.0
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'profiled nmethods'",} 2.2073344E7
jvm_memory_used_bytes{area="heap",id="G1 Survivor Space",} 2608472.0
jvm_memory_used_bytes{area="heap",id="G1 Old Gen",} 5.3587968E7
jvm_memory_used_bytes{area="nonheap",id="Metaspace",} 1.18860432E8
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'non-nmethods'",} 1597056.0
jvm_memory_used_bytes{area="heap",id="G1 Eden Space",} 0.0
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",} 1.5845304E7
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'non-profiled nmethods'",} 7384192.0
jvm_buffer_memory_used_bytes{id="mapped - 'non-volatile memory'",} 0.0
jvm_buffer_memory_used_bytes{id="mapped",} 0.0
jvm_buffer_memory_used_bytes{id="direct",} 6.7302776E7
jvm_info{runtime="OpenJDK Runtime Environment",vendor="Eclipse Adoptium",version="17.0.7+7",} 1.0
jvm_gc_overhead_percent 4.118381607900786E-4
```

CamelBee

- Coming soon!

Settings Scene



theme Light [✓ Dark]

health url /health
health refresh rate 5 secs [2 to 10]
metrics url /metrics
metrics history 300 secs [300 to 600]
metrics refresh rate 5 secs [2 to 10]
max characters in a text field 10000 characters [1000 to 30000]

alternative camera [✓ on]
effects [off]
pause on focus lost [✓ on]

- Coming soon!