OptaPlanner

OptaPlanner

Available as of Camel 2.13

The **optaplanner:** component solves the planning problem contained in a message with OptaPlanner.
For example: feed it an unsolved Vehicle Routing problem and it solves it.

The component supports consumer as BestSolutionChangedEvent listener and producer for processing Solution and ProblemFactChange

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

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

URI format

```
optaplanner:solverConfig[?options]
```

The **solverConfig** is the classpath-local URI of the solverConfig, for example `/org/foo/barSolverConfig.xml`.

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

URI Options (since v 2.16)

Name	Default Value	Type	Context	Description
solverId	DEFAULT_SOLVER	String	Shared	The endpoint keeps a map of Solver instances. solverId specifies a the map key to use.
threadPoolSize	10	int	Producer	Specifies the size the of thread pool to use for processing async Solution messages.
async	false	Boolean	Producer	Specify whether to use another thread for submitting Solution instances rather than blocking the current thread.

Message Headers (since v 2.16)

Name	Default Value	Type	Context	Description
CamelOptaPlannerSolverId	null	String	Shared	Specifies the solverId to use
CamelOptaPlannerIsAsync	PUT	String	Producer	Specify whether to use another thread for submitting Solution instances rather than blocking the current thread.

Message Body

Camel takes the planning problem for the IN body, solves it and returns it on the OUT body.
(since v 2.16) The IN body object supports the following use cases:

- If the body is instance of Solution, then it will be solved using the solver identified by solverId and either synchronously or asynchronously.
- If the body is instance of ProblemFactChange, then it will trigger addProblemFactChange. If the processing is asynchronously, then it will wait till isEveryProblemFactChangeProcessed before returning result.
- If the body is none of the above types, then the producer will return the best result from the solver identified by solverId

Termination

The solving will take as long as specified in the `solverConfig`.

```
<solver>
  ...
  <termination>
    <!-- Terminate after 10 seconds, unless it's not feasible by then yet -->
    <terminationCompositionStyle>AND</terminationCompositionStyle>
    <secondsSpentLimit>10</secondsSpentLimit>
    <bestScoreLimit>-1hard/0soft</bestScoreLimit>
  </termination>
  ...
<solver>
```

Samples

Solve an planning problem that's on the ActiveMQ queue with OptaPlanner:

```
from("activemq:My.Queue").
  .to("optaplanner:/org/foo/barSolverConfig.xml");
```

Expose OptaPlanner as a REST service:

```
from("cxfrs:bean:rsServer?bindingStyle=SimpleConsumer")
  .to("optaplanner:/org/foo/barSolverConfig.xml");
```

See Also

- Configuring Camel
- Component
- Endpoint
- Getting Started