Unsupported Groovy DSL Features on Web Console

Unsupported Groovy DSL Features on Web Console

Most of the DSL features have been supported on Web Console, so you can view and edit in an easy way. However, Web Console doesn't support all the features that you can get from an IDE. Here will list and explain the unsupported features, including the reason for giving up them, alternative solutions or some suggestion for those who indeed need to extend them.

Unsupported Features List

Creating New Classes

When writing a route in Java DSL, we often create new classes. For example, we may create a processor as follows:

```
from("direct:start").process(new Processor() {
     public void process(Exchange exchange) {
          Message in = exchange.getIn();
          in.setBody(in.getBody(String.class) + " World!");
     }
}).to("mock:result");
```

Here an anonymous inner class is created to process the message. But this feature is not supported on Web Console since it use a groovy class loader to parse the route content and you should use groovy grammar to build the processor.
Using a bundle of anonymous inner classes may cause you to lost the route definition after creating them because Web Console can't render them in the view/edit operations.

- happen to: aggregate, bean, process, etc.
- alternative solution: you should build the processors externally, register them on Camel Context and then you can use them by using beanRef or processRef DSL. For such an example, you should refer Content Based Routing on Camel.

Invoking Un-Imported Classse

This feature also appears frequently. In the try...catch DSL or onException DSL, we may use some external or custom exception classes. But current groovy renderer doesn't process the import packages, so you should manually add the import lines each time you edit it. Another problem is that groovy renderer can't return the instance name of that class. For example, you may define a route like:

```
import org.apache.camel.*;
import org.apache.camel.processor.*;
import org.apache.camel.language.groovy.GroovyRouteBuilder;
class GroovyRoute extends GroovyRouteBuilder {
  void configure() {
    MyValidator validator = new MyValidator();
    from("direct:start")
      .doTry().process(validator).to("mock:valid")
      .doCatch(ValidationException.class).to("mock:invalid")
  }
}
```

For this route, groovy renderer can't return the name: validator and it only know there is a processor instance here.

- happen to: doTry...doCatch(...doFinally), onException, throwException, pollEnrich, process, etc.
- improving suggestion: let groovy renderer add the import packages automatically, but I am afraid it can't process them at all times.

Filter and When DSL using Groovy Closure

The Web Console parses routes in groovy language by using the GroovyRouteBuilder in camel-groovy component. GroovyRouteBuilder uses the ConfigureCamel to dynamically add methods for the route configuration behaviors. Currently it has added closure for filter and when DSL. So you can use two method to defined a filter through Web Console:

```
from("direct:start").filter(header("foo").isEqualTo("bar")).to("mock:result")
```

or

```
from("direct:start").filter {e -> e.in.headers.foo == "bar"}.to("mock:result")
```

However, groovy renderer can only render the first filter because a closure is unreadable in Camel Context.

Other Features

Some expression and predicate have complex and refractory toString method, so the ExpressionRenderer and PredicateRenderer class are implemented a little hard.