Exercise 2 - solution

Consumer

Step by step

- Create a contract `src/test/resources/contracts/order/prepared/publish-order-prepared-event.groovy` in the `barista` module

```
package contracts.order.prepared
org.springframework.cloud.contract.spec.Contract.make {
    description 'Barista publishes OrderPreparedEvent'
    label 'orderPreparedEvent'
    input {
        triggeredBy('publishOrderPreparedEvent()')
    }
    outputMessage {
        sentTo('orders-prepared')
        body('''{ "orderId" : "1" }''')
        headers {
            header('barista', 'Jane Doe')
        }
    }
}
```

- In the `barista` module run `mvn clean install -DskipTests` to generate the stubs and install them to your local maven repository
- Create a test `de.fabiankrueger.scc.customer.CustomerReceivesPreparedOrderTest` in the `src/test/java` dir of the `customer` module
- Annotate the test class with `@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.NONE)`
- Annotate the test class with `@AutoConfigureStubRunner(ids = "de.fabiankrueger.scc:barista:+:stubs", stubsMode = StubRunnerProperties.StubsMode.LOCAL)`
- Inject a mocked `CustomerService` bean by defining a member `customerService` of type `CustomerService` and annotate it with `@MockBean`
- Inject a `StubFinder` into the test using `@Autowired`. The `StubFinder` is provided by SCC and is used to trigger the stub to send a message as defined in the contract
- Create a test method which triggers the sending of an inbound `OrderPreparedEvent` as defined in the contract using `stubFinder.trigger("orderPreparedEvent")`
- Verify that the `customerService.onOrderPrepared(OrderPrepapredEvent, String)` method gets called when the message is received. Use Mockito's `ArgumentCaptor` to capture the parameters passed into the method
- Assert that the `orderId` of the `OrderPreparedEvent` passed into the `onOrderPrepared(…)` method matches the value defined in the contract

- Assert that the `barista` header passed into the `onOrderPrepared(⋯)` method matches the value defined in the contract
- Run the test and verify that it passes

Producer

Step by step

- Create an abstract base class `de.fabiankrueger.scc.barista.baseclasses.OrderPreparedBase` in `src/test/java` of the `barista` module.
- Configure the `spring-cloud-contract-maven-plugin` plugin in `barista` module
  - `de.fabiankrueger.scc.barista.baseclasses` should be used as package for base classes. Use `<packageWithBaseClasses>` to achieve this
  - The tests generated by SCC should have `de.fabiankrueger.scc.barista` as base package for tests. Use `<basePackageForTests>` to achieve this
- Annotate the base class with `@AutoConfigureMessageVerifier`
- Annotate the base class with `@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.NONE)`
- Inject existing `de.fabiankrueger.scc.barista.OrderPreparedOutboundAdapter` to the base class using `@Autowired`
- Create a method `publishOrderPreparedEvent` in the base class. This is the method defined in the contract which triggers the sending of the message
- The method should create a `OrderPreparedEvent` as defined in the contract and send it to Kafka using the `OrderPreparedOutboundAdapter.publish(⋯)` method