Exercise 1 - solution

Consumer

Step by step

1. Copy this contract

```
package contracts.rest.payment
org.springframework.cloud.contract.spec.Contract.make {
  description "should accept payment for order and return payment details."
  request {
      url "/order/1/payment"
      method POST()
      headers {
          contentType applicationJson()
      }
      body(amountGiven: 3)
  }
  response {
      status OK()
      headers {
          contentType applicationJson()
      }
      body (
          amountAsked: 2.86,
          amountGiven: 3,
          changeReturned: 0.14
      )
  }
}
```

to `cashier/src/test/resources/contracts/rest/payment/cashier-accepts-payment.groovy`

1. In the `cashier` module run `mvn clean install -DskipTests` to generate the stubs and install them in the local Maven repository.
2. In the `customer` module create a test `de.fabiankrueger.scc.customer.CustomerPaysOrderTest`
3. Annotate the test class with `@AutoConfigureStubRunner` annotation and set the required properties
4. Create a test method and use e.g. Spring's `RestTemplate` or (better [1]) `WebClient` to execute calls against the stubbed payment endpoint of the `cashier`
5. Verify the correct behaviour of the API using assertions
6. Run the test and verify that it passes

Producer

Step by step

1. In the `cashier` module create an abstract base class `de.fabiankrueger.scc.cashier.PaymentTestBase` in `src/test/java/`
2. Annotate the BaseClass with `@WebMvcTest(CashierController.class)` to initialize the Controller for integration test.
3. Annotate the BaseClass with `@AutoConfigureMockMvc` so Spring creates an instance of `MockMvc` for you.
4. Add a member of type `MockMvc` and add `@Autowired` to it to make Spring inject the configured `MockMvc` instance into the test.
5. Define a member of type `CashierService` and annotate it with `@MockBean` to make Spring inject a Mockito mock for the `CashierService`.
6. Create a `public void setup()` method and annotate it with `@BeforeEach`
7. In the setup method initialize RestAssured and pass the mockMvc instance to it `` `RestAssuredMockMvc.mockMvc(mockMvc) ``. RestAssured will be used in the generated SCC test to call the payment endpoint.
8. Record the expected behaviour to the `cashierService` using Mockito's `when(..).thenReturn(..)` syntax
9. Configure a new `<baseClassMapping>` in the SCC plugin defintion in `pom.xml` that maps the new BaseClass to the contract.
10. Let SCC generate the tests by running `mvn clean install -DskipTests` and have a look at the generated test in the `cashier`s target dir.
11. If everything looks good run the generated tests for the `cashier`, e.g. by running `mvn clean test`

[1] RestTemplate is in maintenance mode