**Spring Cloud Contract**

**with**

**Java Rest Service**

*© Copyright 2005, FedEx Corporation. All rights reserved.  
No portion of this document may be copied, displayed, reproduced, or used   
without the express written consent of FedEx*

1. **Spring Cloud Contract**

[Spring Cloud Contract](http://cloud.spring.io/spring-cloud-contract/spring-cloud-contract.html) is a framework for Consumer Driven Contract (CDC) testing. Spring Cloud Contract project provides support for Consumer Driven Contracts and service schemas in Spring applications (for both HTTP and message-based interactions), covering a range of options for writing tests, publishing them as assets, and asserting that a contract is kept by producers and consumers. This ensures the contract between a *Producer* and a *Consumer*, in a distributed system for both HTTP-based and message-based interactions.

Spring Cloud Contract provides us with JVM-based libraries, allowing us to generate Groovy contracts, package them as a jar.

The Spring Cloud Contract project itself consists of two main components:

* Spring Cloud Contract Verifier
* Spring Cloud Contract WireMock

1. **Consumer Driven Contract (CDC) Testing**

[Consumer Driven Contract](https://www.atlassian.com/company/events/summit-us/watch-sessions/2016/build-deploy/verifying-microservice-integrations-with-contract-testing) approach is nothing more than an agreement between the Consumer and Producer about the format of data that they transfer between each other. Normally, the format of the contract is defined by the Consumer and shared with the corresponding Producer. An interesting consequence behind the idea of CDC is that it actually promotes the concept of Test Driven Development on the architectural level.

Consumer-driven contract tests are actually integration tests that are targeting your API, whether it’s REST-based or messaging-based. Imagine you’re working on an application that exposes its data using a REST API. Another team is using your exposed data for some functionality that they are providing. In order to guarantee that the functionality of the other team their application doesn’t break if we make changes to our API, we create a contract between the two teams.

Key in this setup is that the contracts are defined by the consumer of your API instead of the developer that wrote the implementation of a certain functionality. With this approach we can generate tests by using those consumer-driven contracts, and verify whether we’re going to break any of our consumers’ applications.

Afterwards, tests are being implemented in order to verify that the contract is being kept. One of the prerequisites of CDC Testing is the possibility to have a good, at best case close communication with the Producer service team (for example when you are the owner of the Consumer and Producer). Sharing those contracts and communication on test results is important part of implementing proper CDC tests.

1. **Spring Cloud Contract Verifier**

Spring Cloud Contract Verifier enables Consumer Driven Contract (CDC) development of JVM-based applications. Spring Cloud Contract Verifier ships with Contract Definition Language (CDL).

Contract definitions are used to produce the following resources:

* JSON stub definitions to be used by WireMock when doing integration testing on the client code (client tests). Test code must still be written by hand, and test data is produced by Spring Cloud Contract Verifier.
* Acceptance tests (in JUnit) are used to verify if server-side implementation of the API is compliant with the contract (server tests). A full test is generated by Spring Cloud Contract Verifier.

The main purposes of Spring Cloud Contract Verifier with Stub Runner are:

* To ensure that WireMock/Messaging stubs (used when developing the client) do exactly what the actual server-side implementation does.
* To provide a way to publish changes in contracts that are immediately visible on both sides.
* To generate boilerplate test code to be used on the server side.

**Note:** Spring Cloud Contract Verifier’s purpose is NOT to start writing business features in the contracts. Assume that we have a business use case of fraud check. If a user can be a fraud for 100 different reasons, we would assume that you would create 2 contracts, one for the positive case and one for the negative case. Contract tests are used to test contracts between applications and not to simulate full behavior.

1. **Spring Cloud Contract WireMock**

WireMock is a library for stubbing and mocking web services. It constructs a HTTP server that we could connect to as we would to an actual web service. When a [WireMock](http://wiremock.org/) server is in action, we can set up expectations, call the service, and then verify its behaviors.

WireMock is a simulator for HTTP-based APIs. Some might consider it a service virtualization tool or a mock server.  
  
It enables you to stay productive when an API you depend on doesn't exist or isn't complete. It supports testing of edge cases and failure modes that the real API won't reliably produce. And because it's fast it can reduce your build time from hours down to minutes.

1. **Template Revision History Spring Cloud Contract Stub Runner**

Stub Runner allows you to automatically download the stubs of the provided dependencies (or pick those from the classpath), start WireMock servers for them and feed them with proper stub definitions. Treating stubs as contracts of services allows to use stub-runner as an implementation of [Consumer Driven Contracts](http://martinfowler.com/articles/consumerDrivenContracts.html).

* It contains the logic on how to download / register stubs in Http Servers.
* Knows how to register stubs in real service discovery implementations but also how to stub them out in integration tests.
* It consists of a boot server to run stubs from a standalone application.
* It contains a Junit rule that can be used to download / register stubs.

1. **Spring Cloud Contract Starters**

It contains starters that we can use in projects to add all necessary dependencies.

* *spring-cloud-starter-contract-verifier* always add this dependency at *producer* side.
* *spring-cloud-contract-stub-runner* always add this dependency at *consumer* side.

1. **Steps to create a sample application using all above points:**

**Create contract producer service:**

First we need to create the service which produces the contract. This is a regular spring

Boot application providing a very simple REST service. The rest service simply returns

|  |
| --- |
| */spring-cloud-contract-rest-producer/src/main/java/com/spring/cloud/contract/rest/ContractRestProducerApplication.java* |

Person object in JSON format.

|  |
| --- |
| **package** com.spring.cloud.contract.rest;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  **import** org.springframework.context.annotation.ComponentScan;  @SpringBootApplication  @ComponentScan(basePackages = "com.contract")  **public** **class** ContractRestProducerApplication {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(ContractRestProducerApplication.**class**, args);  }  } |

**Create the contract of the Rest Service:**

The contract of the REST service can be defined as a *.groovy* script. This contract specifies that if there is a GET request to url “/person/1”, sample data id: 1, name: “Hello” and surname: “Consumer” representing a Person entity will be returned in the response body of content-type application/json .

**Write the contract specification (Groovy DSL):**

The contract is written in Groovy DSL and consists of two main parts - defining the request and expected response. By specification, there is a minimum of elements that have to be defined.

For the Request part those are: method and url.

For the response part: status code.

|  |  |
| --- | --- |
| The default location for specifying a contract is as follows.  */spring-cloud-contract-rest-producer/****src/test/resources/contracts****/find\_person\_by\_id.groovy* | |
| import org.springframework.cloud.contract.spec.Contract  Contract.make {  description "should return person by id=1"  request {  url "/person/1"  method GET()  }  response {  status 200  headers {  contentType applicationJson()  }  body (  id: 1,  name: "Hello",  surname: "Consumer"  )  }  } |

**Generate automated acceptance tests on Producer side:**

When you are on the Producer side, you can use the possibilities of *spring-cloud-contract-verifier* to automatically generate a test based on this contract that verifies the Producer’s implementation.

Before that you need to have a base test class implementation that all of the generated tests will extend from that base test class.

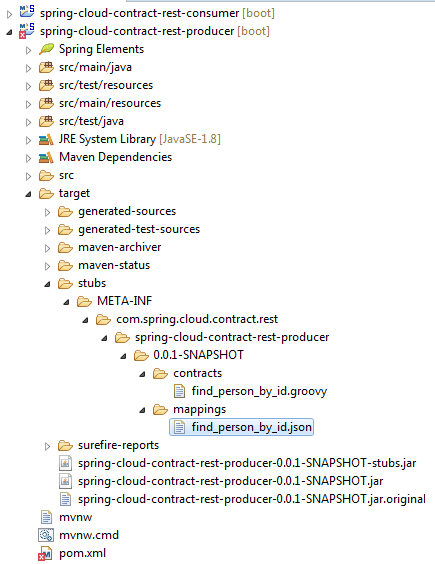
|  |
| --- |
| /spring-cloud-contract-rest-producer/src/test/java/com/spring/cloud/contract/rest/BaseTestClass.java |
| **package** com.spring.cloud.contract.rest;  **import** org.junit.Before;  **import** org.junit.runner.RunWith;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.boot.test.context.SpringBootTest;  **import** org.springframework.cloud.contract.verifier.messaging.boot.AutoConfigureMessageVerifier;  **import** org.springframework.test.annotation.DirtiesContext;  **import** org.springframework.test.context.junit4.SpringRunner;  **import** com.contract.rest.producer.controller.PersonRestController;  **import** io.restassured.module.mockmvc.RestAssuredMockMvc;  @RunWith(SpringRunner.**class**)  @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.***MOCK***)  @DirtiesContext  @AutoConfigureMessageVerifier  **public** **class** BaseTestClass {  @Autowired  PersonRestController personRestController;  @Before  **public** **void** setup() {  RestAssuredMockMvc.*standaloneSetup*(personRestController);  }  } |

|  |
| --- |
| After ***maven success build***, the auto-generated tests have to be located under:  */spring-cloud-contract-rest-producer/target/generated-test-sources/contracts/com/spring/cloud/contract/rest/ContractVerifierTest.java* |
| **package** com.spring.cloud.contract.rest;  **import** com.jayway.jsonpath.DocumentContext;  **import** com.jayway.jsonpath.JsonPath;  **import** com.spring.cloud.contract.rest.BaseTestClass;  **import** io.restassured.module.mockmvc.specification.MockMvcRequestSpecification;  **import** io.restassured.response.ResponseOptions;  **import** org.junit.Test;  **import** **static** com.toomuchcoding.jsonassert.JsonAssertion.*assertThatJson*;  **import** **static** io.restassured.module.mockmvc.RestAssuredMockMvc.\*;  **import** **static** org.springframework.cloud.contract.verifier.assertion.SpringCloudContractAssertions.*assertThat*;  **public** **class** ContractVerifierTest **extends** BaseTestClass {  @Test  **public** **void** validate\_find\_person\_by\_id() **throws** Exception {  // given:  MockMvcRequestSpecification request = *given*();  // when:  ResponseOptions response = *given*().spec(request)  .get("/person/1");  // then:  *assertThat*(response.statusCode()).isEqualTo(200);  *assertThat*(response.header("Content-Type")).matches("application/json.\*");  // and:  DocumentContext parsedJson = JsonPath.*parse*(response.getBody().asString());  *assertThatJson*(parsedJson).field("['id']").isEqualTo(1);  *assertThatJson*(parsedJson).field("['name']").isEqualTo("Hello");  *assertThatJson*(parsedJson).field("['surname']").isEqualTo("Consumer");  }  } |

At this step, when the tests are executed, test results should be GREEN indicating that the REST controller is aligned with the contract and you have a fully functioning service.

Automatically it will generate the json file based on grrovy file as follows under the location:

*/spring-cloud-contract-rest-producer/target/stubs/META-INF/com.spring.cloud.contract.rest/spring-cloud-contract-rest-producer/0.0.1-SNAPSHOT/mappings/find\_person\_by\_id.json*



**Configure Stub Runner on the Consumer side:**

On the Consumer side, we have to write our test and make configurations to execute it against the Producer Stub. Ideally, we will have the tests written before creating the contract with the Producer.

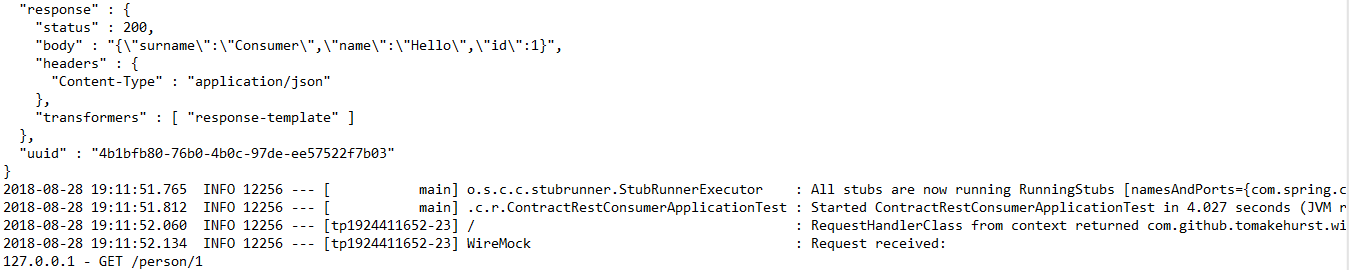
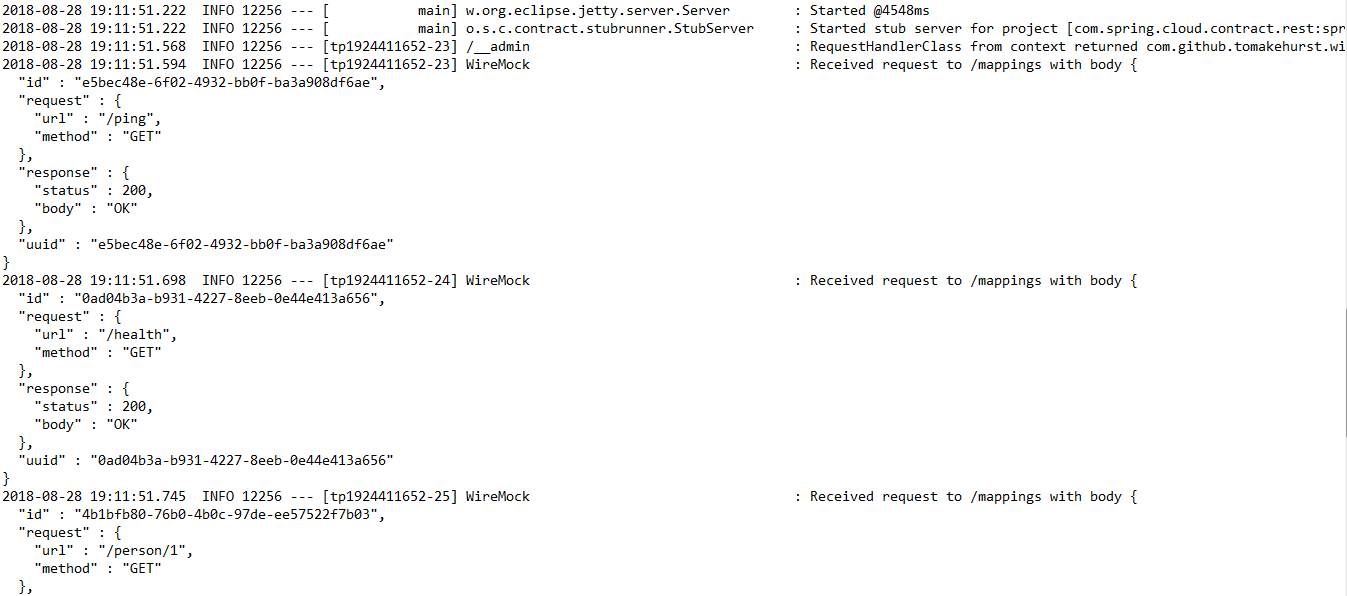
The configuration of spring-cloud-starter-contract-stub-runner, *StubsMode.LOCAL*

|  |
| --- |
| @AutoConfigureStubRunner(ids = "com.spring.cloud.contract.rest:spring-cloud-contract-rest-producer:+:stubs:8100", stubsMode = StubsMode.LOCAL) |

means that the stub runner will search for the Producer stub in the Maven local repository. With the value of ids, we say to use the latest installed stubs version from the Maven local repository.

The Stub runner comes with an embedded WireMock, thus when starting your tests the Stub runner will start a WireMock server internally and install on it the Producer stubs.

**Execute the Consumer test & check verification results:**After that it will execute the test logic and finally will do the assertions. If everything has been properly set, the test is expected to pass. Below you can see a short info from the test execution log that proves the stub installation and running process. If an error has been found then of course the team working on Consumer service has to communicate with the Producer’s service team.



#### **Check the simple Person query business logic:**

Model class *Person.java*

*/spring-cloud-contract-rest-producer/src/main/java/com/contract/rest/producer/model/Person.java*

|  |
| --- |
| **package** com.contract.rest.producer.model;  **public** **class** Person {  **public** Person(Long id, String name, String surname) {  **this**.id = id;  **this**.name = name;  **this**.surname = surname;  }  **private** Long id;  **private** String name;  **private** String surname;  **public** Long getId() {  **return** id;  }  **public** **void** setId(Long id) {  **this**.id = id;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getSurname() {  **return** surname;  }  **public** **void** setSurname(String surname) {  **this**.surname = surname;  }  } |

Service bean *PersonService.java* which just populates a few Person entity in memory and returns the one when asked.

/spring-cloud-contract-rest-producer/src/main/java/com/contract/rest/producer/service/PersonService.java

|  |
| --- |
| **package** com.contract.rest.producer.service;  **import** java.util.HashMap;  **import** java.util.Map;  **import** org.springframework.stereotype.Service;  **import** com.contract.rest.producer.model.Person;  @Service  **public** **class** PersonService {  **private** **final** Map<Long, Person> personMap;  **public** PersonService() {  personMap = **new** HashMap<>();  personMap.put(1L, **new** Person(1L, "Hello", "Consumer"));  personMap.put(2L, **new** Person(2L, "Cloud", "Driven Contract"));  personMap.put(3L, **new** Person(3L, "Spring Cloud", "Rest"));  }  **public** Person findPersonById(Long id) {  **return** personMap.get(id);  }  } |

Rest Controller bean *PersonRestController.java* which calls *PersonService* bean when a REST request is received for a person with the id.

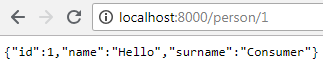
/spring-cloud-contract-rest-producer/src/main/java/com/contract/rest/producer/controller/PersonRestController.java

|  |
| --- |
| **package** com.contract.rest.producer.controller;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.web.bind.annotation.GetMapping;  **import** org.springframework.web.bind.annotation.PathVariable;  **import** org.springframework.web.bind.annotation.RestController;  **import** com.contract.rest.producer.model.Person;  **import** com.contract.rest.producer.service.PersonService;  @RestController  **public** **class** PersonRestController {  @Autowired  **private** PersonService personService;  @GetMapping("/person/{id}")  **public** Person findPersonById(@PathVariable("id") Long id) {  **return** personService.findPersonById(id);  }  } |

#### **Test the contract-rest-service application**

Run the *ContractRestProducerApplication.java* class as a Java Application or Spring Boot Application. The service should start at port 8000.

Visit the service in the browser [*http://localhost:8000/person/1*](http://localhost:8000/person/1), [*http://localhost:8000/person/2*](http://localhost:8000/person/2), etc.



#### **Create contract consumer service:**

With the contract producer service ready, now we need to create the client application which consumes the contract provided. This is a regular Spring Boot application providing a very simple REST service. The rest service simply returns a message with the queried Person’s name, e.g. Hello Consumer.

|  |
| --- |
| /spring-cloud-contract-rest-consumer/src/main/java/com/spring/cloud/contract/rest/ContractRestConsumerApplication.java |
| **package** com.spring.cloud.contract.rest;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  **import** org.springframework.context.annotation.ComponentScan;  @SpringBootApplication  @ComponentScan(basePackages = "com.contract")  **public** **class** ContractRestConsumerApplication {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(ContractRestConsumerApplication.**class**, args);  }  } | |

|  |
| --- |
| /spring-cloud-contract-rest-consumer/src/main/java/com/contract/rest/consumer/controller/ConsumerMessageRestController.java |
| **package** com.contract.rest.consumer.controller;  **import** org.springframework.boot.web.client.RestTemplateBuilder;  **import** org.springframework.web.bind.annotation.PathVariable;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RestController;  **import** org.springframework.web.client.RestTemplate;  **import** com.contract.rest.consumer.model.Person;  @RestController  **public** **class** ConsumerMessageRestController {  **private** **final** RestTemplate restTemplate;  ConsumerMessageRestController(RestTemplateBuilder restTemplateBuilder) {  **this**.restTemplate = restTemplateBuilder.build();  }  @RequestMapping("/message/{personId}")  String getMessage(@PathVariable("personId") Long personId) {  Person person = **this**.restTemplate.getForObject("http://localhost:8000/person/{personId}", Person.**class**, personId);  **return** "Hello " + person.getName();  }  } | |

Model class *Person.java*

*/spring-cloud-contract-rest-consumer/src/main/java/com/contract/rest/consumer/model/Person.java*

|  |
| --- |
| **package** com.contract.rest.consumer.model;  **public** **class** Person {  **public** Person() {  }  **public** Person(Long id, String name, String surname) {  **this**.id = id;  **this**.name = name;  **this**.surname = surname;  }  **private** Long id;  **private** String name;  **private** String surname;  **public** Long getId() {  **return** id;  }  **public** **void** setId(Long id) {  **this**.id = id;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getSurname() {  **return** surname;  }  **public** **void** setSurname(String surname) {  **this**.surname = surname;  }  } |

#### **Create the contract test**

The contract provided by the producer should be consumed as a simple spring test.

*/spring-cloud-contract-rest-consumer/src/test/java/com/spring/cloud/contract/rest/ContractRestConsumerApplicationTest.java*

|  |
| --- |
| **package** com.spring.cloud.contract.rest;  **import** org.assertj.core.api.BDDAssertions;  **import** org.junit.Test;  **import** org.junit.runner.RunWith;  **import** org.springframework.boot.test.autoconfigure.json.AutoConfigureJsonTesters;  **import** org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;  **import** org.springframework.boot.test.context.SpringBootTest;  **import** org.springframework.cloud.contract.stubrunner.spring.AutoConfigureStubRunner;  **import** org.springframework.cloud.contract.stubrunner.spring.StubRunnerProperties.StubsMode;  **import** org.springframework.http.ResponseEntity;  **import** org.springframework.test.context.junit4.SpringRunner;  **import** org.springframework.web.client.RestTemplate;  **import** com.contract.rest.consumer.model.Person;  @RunWith(SpringRunner.**class**)  @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.***MOCK***)  @AutoConfigureMockMvc  @AutoConfigureJsonTesters  @AutoConfigureStubRunner(ids = "com.spring.cloud.contract.rest:spring-cloud-contract-rest-producer:+:stubs:8100", stubsMode = StubsMode.***LOCAL***)  **public** **class** ContractRestConsumerApplicationTest {  @Test  **public** **void** get\_person\_from\_service\_contract() {  // given:  RestTemplate restTemplate = **new** RestTemplate();  // when:  ResponseEntity<Person> personResponseEntity = restTemplate.getForEntity("http://localhost:8100/person/1", Person.**class**);  // then:  BDDAssertions.*then*(personResponseEntity.getStatusCodeValue()).isEqualTo(200);  BDDAssertions.*then*(personResponseEntity.getBody().getId()).isEqualTo(1l);  BDDAssertions.*then*(personResponseEntity.getBody().getName()).isEqualTo("Hello");  BDDAssertions.*then*(personResponseEntity.getBody().getSurname()).isEqualTo("Consumer");  }  } |

This test class will load the stubs of the contract producer service and make sure that the integration to the service is aligned with the contract.

In case the communication is faulty between the consumer service’s test and the producer’s contract, tests will fail and the problem will need to be fixed before making a new change on production.

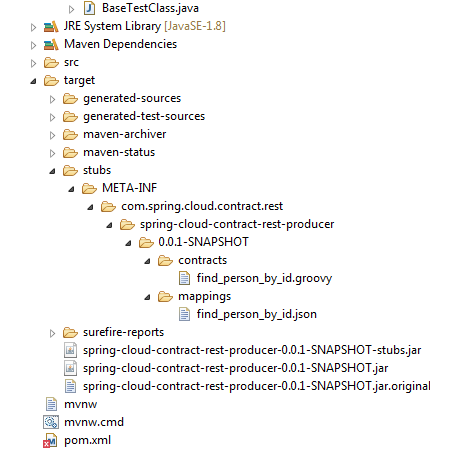
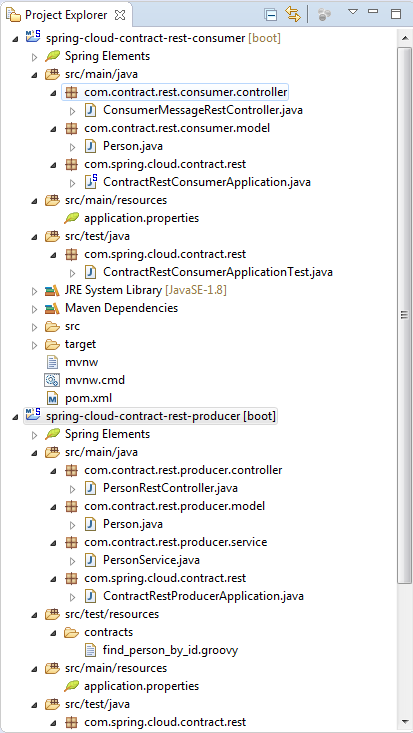
#### **Test the contract-rest-client application**

Run the *ContractRestConsumerApplication.java* class as a Java Application or Spring Boot Application. The service should start at port 9000.

Visit the service in the browser [*http://localhost:9000/message/1*](http://localhost:9000/message/1), [*http://localhost:9000/message/2*](http://localhost:9000/message/2), etc.

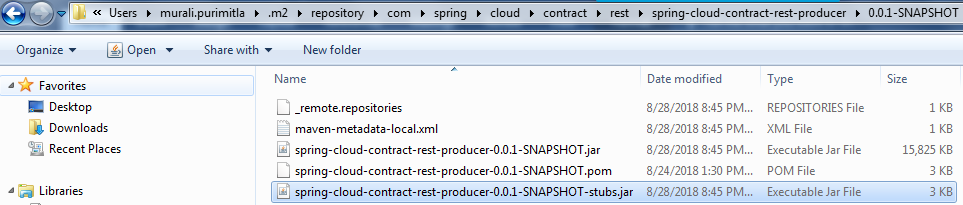


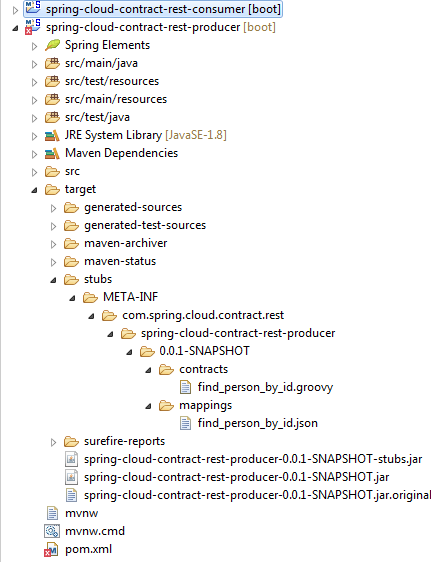
The project structure as follows:

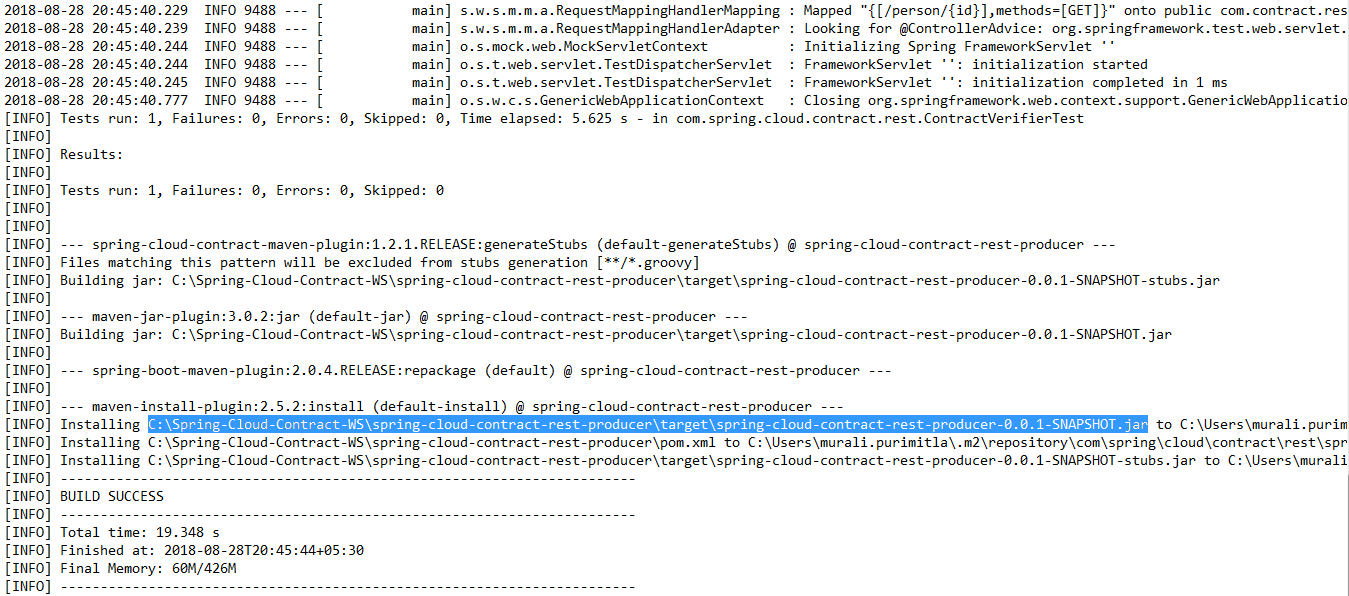


After Maven success build (Producer side), automatically it will create/install the stub folders & spring-cloud-contract-rest-producer-0.0.1-SNAPSHOT.jar in workspace under /target/stubs folder as well as copied into local maven (.m2) repository i.e.

*C:\Users\murali.purimitla\.m2\repository\com\spring\cloud\contract\rest\spring-cloud-contract-rest-producer*





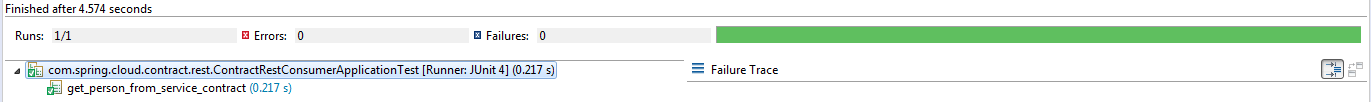


After Maven success build (Consumer side), automatically it will consume the stub folders & spring-cloud-contract-rest-producer-0.0.1-SNAPSHOT.jar from local maven (.m2) repository and executes all test cases successfully.

|  |
| --- |
| . \_\_\_\_ \_ \_\_ \_ \_  /\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \  ( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \  \\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )  ' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /  =========|\_|==============|\_\_\_/=/\_/\_/\_/  :: Spring Boot :: (v2.0.4.RELEASE)  2018-08-28 20:52:41.761 INFO 4732 --- [ main] .c.r.ContractRestConsumerApplicationTest : Starting ContractRestConsumerApplicationTest on WKSBAN18RBW6364 with PID 4732 (started by Murali.Purimitla in C:\Spring-Cloud-Contract-WS\spring-cloud-contract-rest-consumer)  2018-08-28 20:52:41.766 INFO 4732 --- [ main] .c.r.ContractRestConsumerApplicationTest : No active profile set, falling back to default profiles: default  2018-08-28 20:52:42.035 INFO 4732 --- [ main] o.s.w.c.s.GenericWebApplicationContext : Refreshing org.springframework.web.context.support.GenericWebApplicationContext@533bda92: startup date [Tue Aug 28 20:52:42 IST 2018]; root of context hierarchy  2018-08-28 20:52:42.782 INFO 4732 --- [ main] f.a.AutowiredAnnotationBeanPostProcessor : JSR-330 'javax.inject.Inject' annotation found and supported for autowiring  2018-08-28 20:52:44.127 INFO 4732 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/\*\*/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]  2018-08-28 20:52:44.217 INFO 4732 --- [ main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.web.context.support.GenericWebApplicationContext@533bda92: startup date [Tue Aug 28 20:52:42 IST 2018]; root of context hierarchy  2018-08-28 20:52:44.318 INFO 4732 --- [ main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/message/{personId}]}" onto java.lang.String com.contract.rest.consumer.controller.ConsumerMessageRestController.getMessage(java.lang.Long)  2018-08-28 20:52:44.323 INFO 4732 --- [ main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(javax.servlet.http.HttpServletRequest)  2018-08-28 20:52:44.324 INFO 4732 --- [ main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)  2018-08-28 20:52:44.354 INFO 4732 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]  2018-08-28 20:52:44.354 INFO 4732 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]  2018-08-28 20:52:44.534 INFO 4732 --- [ main] o.s.b.t.m.w.SpringBootMockServletContext : Initializing Spring FrameworkServlet ''  2018-08-28 20:52:44.534 INFO 4732 --- [ main] o.s.t.web.servlet.TestDispatcherServlet : FrameworkServlet '': initialization started  2018-08-28 20:52:44.547 INFO 4732 --- [ main] o.s.t.web.servlet.TestDispatcherServlet : FrameworkServlet '': initialization completed in 13 ms  2018-08-28 20:52:44.601 INFO 4732 --- [ main] o.s.c.c.s.AetherStubDownloaderBuilder : Will download stubs and contracts via Aether  2018-08-28 20:52:44.625 INFO 4732 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Remote repos not passed but the switch to work offline was set. Stubs will be used from your local Maven repository.  2018-08-28 20:52:44.875 INFO 4732 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Desired version is [+] - will try to resolve the latest version  2018-08-28 20:52:44.894 INFO 4732 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Resolved version is [0.0.1-SNAPSHOT]  2018-08-28 20:52:44.902 INFO 4732 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Resolved artifact [com.spring.cloud.contract.rest:spring-cloud-contract-rest-producer:jar:stubs:0.0.1-SNAPSHOT] to C:\Users\murali.purimitla\.m2\repository\com\spring\cloud\contract\rest\spring-cloud-contract-rest-producer\0.0.1-SNAPSHOT\spring-cloud-contract-rest-producer-0.0.1-SNAPSHOT-stubs.jar  2018-08-28 20:52:44.905 INFO 4732 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Unpacking stub from JAR [URI: file:/C:/Users/murali.purimitla/.m2/repository/com/spring/cloud/contract/rest/spring-cloud-contract-rest-producer/0.0.1-SNAPSHOT/spring-cloud-contract-rest-producer-0.0.1-SNAPSHOT-stubs.jar]  2018-08-28 20:52:44.913 INFO 4732 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Unpacked file to [C:\Users\MURALI~1.PUR\AppData\Local\Temp\contracts3604802965700825355]  2018-08-28 20:52:46.089 INFO 4732 --- [ main] wiremock.org.eclipse.jetty.util.log : Logging initialized @5476ms  2018-08-28 20:52:46.213 INFO 4732 --- [ main] w.org.eclipse.jetty.server.Server : jetty-9.2.z-SNAPSHOT  2018-08-28 20:52:46.230 INFO 4732 --- [ main] w.o.e.j.server.handler.ContextHandler : Started w.o.e.j.s.ServletContextHandler@2291d9a0{/\_\_admin,null,AVAILABLE}  2018-08-28 20:52:46.230 INFO 4732 --- [ main] w.o.e.j.server.handler.ContextHandler : Started w.o.e.j.s.ServletContextHandler@25c2a9e3{/,null,AVAILABLE}  2018-08-28 20:52:46.283 INFO 4732 --- [ main] w.o.e.j.s.NetworkTrafficServerConnector : Started NetworkTrafficServerConnector@2a02e34b{HTTP/1.1}{0.0.0.0:8100}  2018-08-28 20:52:46.284 INFO 4732 --- [ main] w.org.eclipse.jetty.server.Server : Started @5672ms  2018-08-28 20:52:46.284 INFO 4732 --- [ main] o.s.c.contract.stubrunner.StubServer : Started stub server for project [com.spring.cloud.contract.rest:spring-cloud-contract-rest-producer:0.0.1-SNAPSHOT:stubs] on port 8100  2018-08-28 20:52:46.763 INFO 4732 --- [qtp390010388-24] /\_\_admin : RequestHandlerClass from context returned com.github.tomakehurst.wiremock.http.AdminRequestHandler. Normalized mapped under returned 'null'  2018-08-28 20:52:46.815 INFO 4732 --- [qtp390010388-24] WireMock : Received request to /mappings with body {  "id" : "1d0eca8b-4b94-4d04-9cd0-757ed3d22f38",  "request" : {  "url" : "/ping",  "method" : "GET"  },  "response" : {  "status" : 200,  "body" : "OK"  },  "uuid" : "1d0eca8b-4b94-4d04-9cd0-757ed3d22f38"  }  2018-08-28 20:52:46.934 INFO 4732 --- [qtp390010388-25] WireMock : Received request to /mappings with body {  "id" : "d1758b13-0962-4002-9aa4-e5d60d17788e",  "request" : {  "url" : "/health",  "method" : "GET"  },  "response" : {  "status" : 200,  "body" : "OK"  },  "uuid" : "d1758b13-0962-4002-9aa4-e5d60d17788e"  }  2018-08-28 20:52:46.980 INFO 4732 --- [qtp390010388-26] WireMock : Received request to /mappings with body {  "id" : "55733786-d1fb-481b-9206-0fb468459cbb",  "request" : {  "url" : "/person/1",  "method" : "GET"  },  "response" : {  "status" : 200,  "body" : "{\"surname\":\"Consumer\",\"name\":\"Hello\",\"id\":1}",  "headers" : {  "Content-Type" : "application/json"  },  "transformers" : [ "response-template" ]  },  "uuid" : "55733786-d1fb-481b-9206-0fb468459cbb"  }  2018-08-28 20:52:46.996 INFO 4732 --- [ main] o.s.c.c.stubrunner.StubRunnerExecutor : All stubs are now running RunningStubs [namesAndPorts={com.spring.cloud.contract.rest:spring-cloud-contract-rest-producer:0.0.1-SNAPSHOT:stubs=8100}]  2018-08-28 20:52:47.037 INFO 4732 --- [ main] .c.r.ContractRestConsumerApplicationTest : Started ContractRestConsumerApplicationTest in 5.573 seconds (JVM running for 6.423)  2018-08-28 20:52:47.270 INFO 4732 --- [qtp390010388-24] / : RequestHandlerClass from context returned com.github.tomakehurst.wiremock.http.StubRequestHandler. Normalized mapped under returned 'null'  2018-08-28 20:52:47.413 INFO 4732 --- [qtp390010388-24] WireMock : Request received:  127.0.0.1 - GET /person/1  User-Agent: [Java/1.8.0\_141]  Connection: [keep-alive]  Host: [localhost:8100]  Accept: [application/json, application/\*+json]  Matched response definition:  {  "status" : 200,  "body" : "{\"surname\":\"Consumer\",\"name\":\"Hello\",\"id\":1}",  "headers" : {  "Content-Type" : "application/json"  },  "transformers" : [ "response-template" ]  }  Response:  HTTP/1.1 200  Content-Type: [application/json]  [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 6.608 s - in com.spring.cloud.contract.rest.ContractRestConsumerApplicationTest  2018-08-28 20:52:47.529 INFO 4732 --- [ Thread-15] o.s.w.c.s.GenericWebApplicationContext : Closing org.springframework.web.context.support.GenericWebApplicationContext@533bda92: startup date [Tue Aug 28 20:52:42 IST 2018]; root of context hierarchy  2018-08-28 20:52:47.552 INFO 4732 --- [ Thread-15] w.o.e.j.s.NetworkTrafficServerConnector : Stopped NetworkTrafficServerConnector@2a02e34b{HTTP/1.1}{0.0.0.0:8100}  2018-08-28 20:52:47.553 INFO 4732 --- [ Thread-15] w.o.e.j.server.handler.ContextHandler : Stopped w.o.e.j.s.ServletContextHandler@25c2a9e3{/,null,UNAVAILABLE}  2018-08-28 20:52:47.554 INFO 4732 --- [ Thread-15] w.o.e.j.server.handler.ContextHandler : Stopped w.o.e.j.s.ServletContextHandler@2291d9a0{/\_\_admin,null,UNAVAILABLE}  2018-08-28 20:52:47.555 WARN 4732 --- [ Thread-15] w.o.e.j.util.thread.QueuedThreadPool : qtp390010388{STOPPING,8<=8<=10,i=0,q=2} Couldn't stop Thread[qtp390010388-22-acceptor-0@1ed5aba7-NetworkTrafficServerConnector@2a02e34b{HTTP/1.1}{0.0.0.0:8100},5,main]  2018-08-28 20:52:47.555 WARN 4732 --- [ Thread-15] w.o.e.j.util.thread.QueuedThreadPool : qtp390010388{STOPPING,8<=8<=10,i=0,q=2} Couldn't stop Thread[qtp390010388-23-acceptor-1@a0eaf81-NetworkTrafficServerConnector@2a02e34b{HTTP/1.1}{0.0.0.0:8100},5,main]  [INFO]  [INFO] Results:  [INFO]  [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  [INFO]  [INFO]  [INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ spring-cloud-contract-rest-consumer ---  [INFO] Building jar: C:\Spring-Cloud-Contract-WS\spring-cloud-contract-rest-consumer\target\spring-cloud-contract-rest-consumer-0.0.1-SNAPSHOT.jar  [INFO]  [INFO] --- spring-boot-maven-plugin:2.0.4.RELEASE:repackage (default) @ spring-cloud-contract-rest-consumer ---  [INFO]  [INFO] --- maven-install-plugin:2.5.2:install (default-install) @ spring-cloud-contract-rest-consumer ---  [INFO] Installing C:\Spring-Cloud-Contract-WS\spring-cloud-contract-rest-consumer\target\spring-cloud-contract-rest-consumer-0.0.1-SNAPSHOT.jar to C:\Users\murali.purimitla\.m2\repository\com\spring\cloud\contract\rest\spring-cloud-contract-rest-consumer\0.0.1-SNAPSHOT\spring-cloud-contract-rest-consumer-0.0.1-SNAPSHOT.jar  [INFO] Installing C:\Spring-Cloud-Contract-WS\spring-cloud-contract-rest-consumer\pom.xml to C:\Users\murali.purimitla\.m2\repository\com\spring\cloud\contract\rest\spring-cloud-contract-rest-consumer\0.0.1-SNAPSHOT\spring-cloud-contract-rest-consumer-0.0.1-SNAPSHOT.pom  [INFO] ------------------------------------------------------------------------  [INFO] BUILD SUCCESS  [INFO] ------------------------------------------------------------------------  [INFO] Total time: 13.165 s  [INFO] Finished at: 2018-08-28T20:52:48+05:30  [INFO] Final Memory: 42M/298M  [INFO] ----------------------------------------------------------------------- |

Run the Junit test consumer side (*ContractRestConsumerApplicationTest.java*) class explicitly:

|  |
| --- |
| . \_\_\_\_ \_ \_\_ \_ \_  /\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \  ( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \  \\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )  ' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /  =========|\_|==============|\_\_\_/=/\_/\_/\_/  :: Spring Boot :: (v2.0.4.RELEASE)  2018-08-28 21:15:39.055 INFO 2588 --- [ main] .c.r.ContractRestConsumerApplicationTest : Starting ContractRestConsumerApplicationTest on WKSBAN18RBW6364 with PID 2588 (started by Murali.Purimitla in C:\Spring-Cloud-Contract-WS\spring-cloud-contract-rest-consumer)  2018-08-28 21:15:39.057 INFO 2588 --- [ main] .c.r.ContractRestConsumerApplicationTest : No active profile set, falling back to default profiles: default  2018-08-28 21:15:39.260 INFO 2588 --- [ main] o.s.w.c.s.GenericWebApplicationContext : Refreshing org.springframework.web.context.support.GenericWebApplicationContext@6fdbe764: startup date [Tue Aug 28 21:15:39 IST 2018]; root of context hierarchy  2018-08-28 21:15:39.906 INFO 2588 --- [ main] f.a.AutowiredAnnotationBeanPostProcessor : JSR-330 'javax.inject.Inject' annotation found and supported for autowiring  2018-08-28 21:15:40.797 INFO 2588 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/\*\*/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]  2018-08-28 21:15:40.953 INFO 2588 --- [ main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.web.context.support.GenericWebApplicationContext@6fdbe764: startup date [Tue Aug 28 21:15:39 IST 2018]; root of context hierarchy  2018-08-28 21:15:41.023 INFO 2588 --- [ main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/message/{personId}]}" onto java.lang.String com.contract.rest.consumer.controller.ConsumerMessageRestController.getMessage(java.lang.Long)  2018-08-28 21:15:41.025 INFO 2588 --- [ main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(javax.servlet.http.HttpServletRequest)  2018-08-28 21:15:41.026 INFO 2588 --- [ main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)  2018-08-28 21:15:41.054 INFO 2588 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]  2018-08-28 21:15:41.055 INFO 2588 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]  2018-08-28 21:15:41.233 INFO 2588 --- [ main] o.s.b.t.m.w.SpringBootMockServletContext : Initializing Spring FrameworkServlet ''  2018-08-28 21:15:41.233 INFO 2588 --- [ main] o.s.t.web.servlet.TestDispatcherServlet : FrameworkServlet '': initialization started  2018-08-28 21:15:41.253 INFO 2588 --- [ main] o.s.t.web.servlet.TestDispatcherServlet : FrameworkServlet '': initialization completed in 20 ms  2018-08-28 21:15:41.299 INFO 2588 --- [ main] o.s.c.c.s.AetherStubDownloaderBuilder : Will download stubs and contracts via Aether  2018-08-28 21:15:41.301 INFO 2588 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Remote repos not passed but the switch to work offline was set. Stubs will be used from your local Maven repository.  2018-08-28 21:15:41.437 INFO 2588 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Desired version is [+] - will try to resolve the latest version  2018-08-28 21:15:41.452 INFO 2588 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Resolved version is [0.0.1-SNAPSHOT]  2018-08-28 21:15:41.460 INFO 2588 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Resolved artifact [com.spring.cloud.contract.rest:spring-cloud-contract-rest-producer:jar:stubs:0.0.1-SNAPSHOT] to C:\Users\murali.purimitla\.m2\repository\com\spring\cloud\contract\rest\spring-cloud-contract-rest-producer\0.0.1-SNAPSHOT\spring-cloud-contract-rest-producer-0.0.1-SNAPSHOT-stubs.jar  2018-08-28 21:15:41.462 INFO 2588 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Unpacking stub from JAR [URI: file:/C:/Users/murali.purimitla/.m2/repository/com/spring/cloud/contract/rest/spring-cloud-contract-rest-producer/0.0.1-SNAPSHOT/spring-cloud-contract-rest-producer-0.0.1-SNAPSHOT-stubs.jar]  2018-08-28 21:15:41.472 INFO 2588 --- [ main] o.s.c.c.stubrunner.AetherStubDownloader : Unpacked file to [C:\Users\MURALI~1.PUR\AppData\Local\Temp\contracts2970851436936762782]  2018-08-28 21:15:42.215 INFO 2588 --- [ main] wiremock.org.eclipse.jetty.util.log : Logging initialized @4414ms  2018-08-28 21:15:42.281 INFO 2588 --- [ main] w.org.eclipse.jetty.server.Server : jetty-9.2.z-SNAPSHOT  2018-08-28 21:15:42.296 INFO 2588 --- [ main] w.o.e.j.server.handler.ContextHandler : Started w.o.e.j.s.ServletContextHandler@3fc7c734{/\_\_admin,null,AVAILABLE}  2018-08-28 21:15:42.296 INFO 2588 --- [ main] w.o.e.j.server.handler.ContextHandler : Started w.o.e.j.s.ServletContextHandler@1cf336fd{/,null,AVAILABLE}  2018-08-28 21:15:42.309 INFO 2588 --- [ main] w.o.e.j.s.NetworkTrafficServerConnector : Started NetworkTrafficServerConnector@2c829dbc{HTTP/1.1}{0.0.0.0:8100}  2018-08-28 21:15:42.310 INFO 2588 --- [ main] w.org.eclipse.jetty.server.Server : Started @4510ms  2018-08-28 21:15:42.310 INFO 2588 --- [ main] o.s.c.contract.stubrunner.StubServer : Started stub server for project [com.spring.cloud.contract.rest:spring-cloud-contract-rest-producer:0.0.1-SNAPSHOT:stubs] on port 8100  2018-08-28 21:15:42.596 INFO 2588 --- [qtp600554759-23] /\_\_admin : RequestHandlerClass from context returned com.github.tomakehurst.wiremock.http.AdminRequestHandler. Normalized mapped under returned 'null'  2018-08-28 21:15:42.625 INFO 2588 --- [qtp600554759-23] WireMock : Received request to /mappings with body {  "id" : "7f51a2e9-bcaa-442a-9dcd-9b7f19f39b85",  "request" : {  "url" : "/ping",  "method" : "GET"  },  "response" : {  "status" : 200,  "body" : "OK"  },  "uuid" : "7f51a2e9-bcaa-442a-9dcd-9b7f19f39b85"  }  2018-08-28 21:15:42.726 INFO 2588 --- [qtp600554759-24] WireMock : Received request to /mappings with body {  "id" : "7c353b39-4ebe-42e3-a877-23e4eabc90c9",  "request" : {  "url" : "/health",  "method" : "GET"  },  "response" : {  "status" : 200,  "body" : "OK"  },  "uuid" : "7c353b39-4ebe-42e3-a877-23e4eabc90c9"  }  2018-08-28 21:15:42.804 INFO 2588 --- [qtp600554759-25] WireMock : Received request to /mappings with body {  "id" : "55733786-d1fb-481b-9206-0fb468459cbb",  "request" : {  "url" : "/person/1",  "method" : "GET"  },  "response" : {  "status" : 200,  "body" : "{\"surname\":\"Consumer\",\"name\":\"Hello\",\"id\":1}",  "headers" : {  "Content-Type" : "application/json"  },  "transformers" : [ "response-template" ]  },  "uuid" : "55733786-d1fb-481b-9206-0fb468459cbb"  }  2018-08-28 21:15:42.825 INFO 2588 --- [ main] o.s.c.c.stubrunner.StubRunnerExecutor : All stubs are now running RunningStubs [namesAndPorts={com.spring.cloud.contract.rest:spring-cloud-contract-rest-producer:0.0.1-SNAPSHOT:stubs=8100}]  2018-08-28 21:15:42.889 INFO 2588 --- [ main] .c.r.ContractRestConsumerApplicationTest : Started ContractRestConsumerApplicationTest in 4.14 seconds (JVM running for 5.089)  2018-08-28 21:15:43.080 INFO 2588 --- [qtp600554759-23] / : RequestHandlerClass from context returned com.github.tomakehurst.wiremock.http.StubRequestHandler. Normalized mapped under returned 'null'  2018-08-28 21:15:43.187 INFO 2588 --- [qtp600554759-23] WireMock : Request received:  127.0.0.1 - GET /person/1  User-Agent: [Java/1.8.0\_141]  Connection: [keep-alive]  Host: [localhost:8100]  Accept: [application/json, application/\*+json]  Matched response definition:  {  "status" : 200,  "body" : "{\"surname\":\"Consumer\",\"name\":\"Hello\",\"id\":1}",  "headers" : {  "Content-Type" : "application/json"  },  "transformers" : [ "response-template" ]  }  Response:  HTTP/1.1 200  Content-Type: [application/json]  2018-08-28 21:15:43.275 INFO 2588 --- [ Thread-15] o.s.w.c.s.GenericWebApplicationContext : Closing org.springframework.web.context.support.GenericWebApplicationContext@6fdbe764: startup date [Tue Aug 28 21:15:39 IST 2018]; root of context hierarchy  2018-08-28 21:15:43.391 INFO 2588 --- [ Thread-15] w.o.e.j.s.NetworkTrafficServerConnector : Stopped NetworkTrafficServerConnector@2c829dbc{HTTP/1.1}{0.0.0.0:8100}  2018-08-28 21:15:43.394 INFO 2588 --- [ Thread-15] w.o.e.j.server.handler.ContextHandler : Stopped w.o.e.j.s.ServletContextHandler@1cf336fd{/,null,UNAVAILABLE}  2018-08-28 21:15:43.467 INFO 2588 --- [ Thread-15] w.o.e.j.server.handler.ContextHandler : Stopped w.o.e.j.s.ServletContextHandler@3fc7c734{/\_\_admin,null,UNAVAILABLE}  2018-08-28 21:15:43.469 WARN 2588 --- [ Thread-15] w.o.e.j.util.thread.QueuedThreadPool : qtp600554759{STOPPING,8<=8<=10,i=8,q=8} Couldn't stop Thread[qtp600554759-26,5,main]  2018-08-28 21:15:43.469 WARN 2588 --- [ Thread-15] w.o.e.j.util.thread.QueuedThreadPool : qtp600554759{STOPPING,8<=8<=10,i=8,q=8} Couldn't stop Thread[qtp600554759-25,5,main]  2018-08-28 21:15:43.469 WARN 2588 --- [ Thread-15] w.o.e.j.util.thread.QueuedThreadPool : qtp600554759{STOPPING,8<=8<=10,i=8,q=8} Couldn't stop Thread[qtp600554759-24,5,main]  2018-08-28 21:15:43.469 WARN 2588 --- [ Thread-15] w.o.e.j.util.thread.QueuedThreadPool : qtp600554759{STOPPING,8<=8<=10,i=8,q=8} Couldn't stop Thread[qtp600554759-30,5,main]  2018-08-28 21:15:43.469 WARN 2588 --- [ Thread-15] w.o.e.j.util.thread.QueuedThreadPool : qtp600554759{STOPPING,8<=8<=10,i=8,q=8} Couldn't stop Thread[qtp600554759-23,5,main]  2018-08-28 21:15:43.469 WARN 2588 --- [ Thread-15] w.o.e.j.util.thread.QueuedThreadPool : qtp600554759{STOPPING,8<=8<=10,i=8,q=8} Couldn't stop Thread[qtp600554759-20,5,main]  2018-08-28 21:15:43.469 WARN 2588 --- [ Thread-15] w.o.e.j.util.thread.QueuedThreadPool : qtp600554759{STOPPING,8<=8<=10,i=8,q=8} Couldn't stop Thread[qtp600554759-19,5,main]  2018-08-28 21:15:43.469 WARN 2588 --- [ Thread-15] w.o.e.j.util.thread.QueuedThreadPool : qtp600554759{STOPPING,8<=8<=10,i=8,q=8} Couldn't stop Thread[qtp600554759-31,5,main] |



------------------------------------------------------ END -----------------------------------------------------------