# Pact Testing Course

## Generate contract file from consumer and share it with provider

1. Execute the contract file from the provider side (in pact unit test)

2. Check what the consumer is expecting

## A diagram of a process AI-generated content may be incorrect.

## Create Provider Unit Test

### Benefits:

- Allows us to find issues early on in the test phase  
- Allows us to run pact tests independently  
- Provides fast feedback  
- Pact tests are stable  
- Pact testing gives us confidence in the release  
- Build smarter test cases

### Pre-requisite:

- Need to deploy consumer service & provider service to start testingA diagram of a data flow

AI-generated content may be incorrect.

- Consumer microservice (library) runs on port 9000 -> Method: getProducts()  
- Provider microservice (source) runs on port 81  
- We are using Pact JVM library + JUnit 5 to build contract tests  
- The contract tests should exist in the same place as the unit tests

A diagram of a computer program

AI-generated content may be incorrect.

When we prepare the unit test for getOfferedPrices() which internally calls getAllCourseDetails() we need to mock getAllCourseDetails() which is an external dependency i.e. you should not test with external integrated products. It needs to be tested in isolation.

Step 1

* Write test from the consumer side
* Later we will execute this contract on the provider side.

We need to override the base URL to hit the pact mock server instead of the real secondary endpoint.

Add the line of code

* libraryController.setBaseUrl(url);
* it will start on your "localhost"

@Test

@PactTestFor(pactMethod = "PactConsumerMethodName", port = "9999")

public void testAllProducts(MockServer mockServer) {

libraryController.setBaseUrl(mockServer.getUrl());

}

* After the pact test run, pact contract JSON schema will be generated in the target directory.
* The pact broker is a central management system. It routes the file from the consumer microservice to the provider microservice.
* If done manually, you place the file in the provider under src/main/pacts
* The correct pact maven dependencies need to be present in both the consumer & provider projects
* The pact schema verifies that the JSON contract schema is correct in terms of structure, fields (mandatory) & data types are correct, but it won't validate the values
* Pact-driven consumer tests are executed on the provider.
* In the pact JSON contract, there is a section called interactions [] array which contains entry to represent each from consumer to provider.
* On the consumer side you do not need to start the pact server.
* On the provider side you need to start the real server as a prerequisite. Note that this is the real server and not the pact server.
* On the consumer side we start the pact server as a prerequisite.

@SpringBootTest (webEnvironment.RANDOM\_PORT) (starts real server on a random port)  
@Provider("CourseCatalogue") // must match name of provider in the contract  
@PactFolder("pacts") // where pact contracts live, default is src/main/pacts  
public class PactProviderTest {

* consumer pact tests execute on provider side via pact broker to the mocked API
* provider pact tests
* Setup/Teardown is necessary to ensure data is present in the db needed for the consumer
  + Pre-requisite: insert a record in the provider side via the pact broker to the mocked Api
  + Delete the record that was inserted

## A Negative Pact test:

A diagram of a product

AI-generated content may be incorrect.