**What you need?**

JDK 10+

Apache Maven 3.6+

IntelliJ/Eclipse IDE

**Clone Example Projects:**

Clone the “consumer” [search-books] & “provider” [book-catalog] projects from the below GithHub repos and checkout “**cdc\_mod2\_day1\_begin**” branch.

<https://github.com/pradip79/search-books.git>

<https://github.com/pradip79/book-catalog.git>

**Consumer Side:**

**Step1**: Write a mocked integration test

1. Add Wiremock dependency in the pom.xml:

<**dependency**>  
 <**groupId**>com.github.tomakehurst</**groupId**>  
 <**artifactId**>wiremock-jre8</**artifactId**>  
 <**version**>2.28.0</**version**>  
 <**scope**>test</**scope**>  
</**dependency**>

1. Write the below integration test & see it pass

**import** com.github.tomakehurst.wiremock.WireMockServer;  
**import static** com.github.tomakehurst.wiremock.client.WireMock.\*;  
**import static** org.assertj.core.api.BDDAssertions.*then*;  
**import** org.junit.jupiter.api.AfterEach;  
**import** org.junit.jupiter.api.BeforeEach;  
**import** org.junit.jupiter.api.Test;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.boot.test.context.SpringBootTest;  
**import** org.springframework.http.ResponseEntity;  
  
@SpringBootTest  
**class** BookCatalogClientIntegrationTest {  
 @Autowired  
 **private** BookCatalogClient **bookCatalogClient**;  
 **private** WireMockServer **wireMockServer**;  
  
 @BeforeEach  
 **public void** setUp(){  
 **wireMockServer** = **new** WireMockServer(7071);  
 **wireMockServer**.start();  
  
 *configureFor*(7071);  
  
 *stubFor*(*get*(*urlEqualTo*(**"/book-catalog/book/101201"**))  
 .willReturn(*aResponse*()  
 .withStatus(200)  
 .withHeader(**"Content-Type"**, **"application/json"**)  
 .withBody(**"{\"isbn\":\"101201\",\"title\":\"The Alchemist\",\"author\":[\"Paolo Coelho\"]}"**)  
 )  
 );  
 }  
  
 @Test  
 **void** getBookByISBN\_shouldReturn\_singleBook() {  
 *//Given* String ISBN = **"101201"**;  
  
 *//When* ResponseEntity<Book> response = **bookCatalogClient**.getBookByIsbn(ISBN);  
  
 *//Then  
 then*(response.getStatusCode().value())  
 .as(**"Status code correct"**)  
 .isEqualTo(200);  
 *then*(response.getBody().getTitle())  
 .as(**"Title as expected"**)  
 .isEqualTo(**"The Alchemist"**);  
 *then*(response.getBody().getAuthor())  
 .as(**"Authors as expected"**)  
 .contains(**"Paolo Coelho"**);  
 }  
  
 @AfterEach  
 **public void** tearDown(){  
 **wireMockServer**.stop();  
 }  
}

**Step2**: Execute against the real provider endpoint

1. Change book-service.url to point to the “book-catalog” service:

**book-service.url** = **http://localhost:7070**

1. Run the test and see it fail – analyze why failed.

**Step3**: Write a pact contract test

1. Add “pact-consumer-junit5” dependency in the pom.xml

<**dependency**>  
 <**groupId**>au.com.dius.pact.consumer</**groupId**>  
 <**artifactId**>junit5</**artifactId**>  
 <**version**>4.1.6</**version**>  
</**dependency**>

1. Write the below contract test & see it fail (as happened with actual provider)

**import** au.com.dius.pact.consumer.MockServer;  
**import** au.com.dius.pact.consumer.dsl.PactDslJsonBody;  
**import** au.com.dius.pact.consumer.dsl.PactDslJsonRootValue;  
**import** au.com.dius.pact.consumer.dsl.PactDslWithProvider;  
**import** au.com.dius.pact.consumer.junit5.PactConsumerTestExt;  
**import** au.com.dius.pact.consumer.junit5.PactTestFor;  
**import** au.com.dius.pact.core.model.RequestResponsePact;  
**import** au.com.dius.pact.core.model.annotations.Pact;  
**import** org.junit.jupiter.api.Test;  
**import static** org.assertj.core.api.BDDAssertions.*then*;  
**import** org.junit.jupiter.api.extension.ExtendWith;  
**import** java.util.Arrays;  
**import** java.util.Map;  
  
@PactTestFor(providerName = **"book-catalog"**, port = **"7071"**)  
@ExtendWith(PactConsumerTestExt.**class**)  
**public class** BookCatalogConsumerPactTest {  
 BookCatalogClient **bookCatalogClient**;  
  
 @Pact(consumer = **"search-books"**, provider = **"book-catalog"**)  
 RequestResponsePact getOneBook(PactDslWithProvider pactBuilder){  
 **return** pactBuilder.given(**"book with ISBN 101201 exists"**)  
 .uponReceiving(**"get one book"**)  
 .method(**"GET"**)  
 .path(**"/book-catalog/book/101201"**)  
 .willRespondWith()  
 .status(200)  
 .headers(Map.*of*(**"Content-Type"**, **"application/json"**))  
 .body(**new** PactDslJsonBody()  
 .stringType(**"isbn"**, **"101201"**)  
 .stringType(**"title"**, **"The Alchemist"**)  
 .minArrayLike(**"authors"**, 1, PactDslJsonRootValue.*stringType*(**"Paolo Coelho"**))  
 .stringType(**"publisher"**, **"HarperCollins"**)  
 .stringType(**"genre"**, **"Fiction"**))  
 .toPact();  
 }  
  
 @Test  
 @PactTestFor(pactMethod = **"getOneBook"**)  
 **void** getBookByISBN\_shouldReturn\_singleBook(MockServer mockServer){  
 *//Given* **bookCatalogClient** = **new** BookCatalogClient(mockServer.getUrl());  
 Book expectedBook = **new** Book(**"101201"**, **"The Alchemist"**, Arrays.*asList*(**"Paolo Coelho"**), **"Fiction"**, **"HarperCollins"**);  
  
 *//When* Book actualBook = **bookCatalogClient**.getBookByIsbn(**"101201"**).getBody();  
  
 *//Then  
 then*(expectedBook)  
 .as(**"Book returned as expected"**)  
 .isEqualTo(actualBook);  
 }  
  
}

1. Refactor Book.java to rename “author” to “authors” as defined in the contract, run the test again and make it pass.
2. Open the generated contract file from below directory and analyze.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Provider Side:**

**Step1:**

Add “pact-provider-junit5” dependency in the pom.xml

<**dependency**>  
 <**groupId**>au.com.dius.pact.provider</**groupId**>  
 <**artifactId**>junit5</**artifactId**>  
 <**version**>4.1.6</**version**>  
</**dependency**>

**Step2:**

Copy the consumer generated pact (.json contract file) file into “src/test/resources/pacts” folder.

**Step3**: Write the provider side contract verification test & run it

**import** au.com.dius.pact.provider.junit5.HttpTestTarget;  
**import** au.com.dius.pact.provider.junit5.PactVerificationContext;  
**import** au.com.dius.pact.provider.junit5.PactVerificationInvocationContextProvider;  
**import** au.com.dius.pact.provider.junitsupport.Provider;  
**import** au.com.dius.pact.provider.junitsupport.State;  
**import** au.com.dius.pact.provider.junitsupport.loader.PactFolder;  
**import** org.junit.jupiter.api.BeforeEach;  
**import** org.junit.jupiter.api.TestTemplate;  
**import** org.junit.jupiter.api.extension.ExtendWith;  
**import** org.springframework.boot.test.context.SpringBootTest;  
**import** org.springframework.boot.web.server.LocalServerPort;  
**import** org.springframework.test.context.junit.jupiter.SpringExtension;  
  
@Provider(**"book-catalog"**)  
@PactFolder(**"pacts"**)  
@ExtendWith(SpringExtension.**class**)  
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.***RANDOM\_PORT***)  
**public class** SearchBooksPactProviderTest {  
 @LocalServerPort  
 **int port**;  
  
 @BeforeEach  
 **void** setUp(PactVerificationContext context) {  
 context.setTarget(**new** HttpTestTarget(**"localhost"**, **port**));  
 }  
  
 @TestTemplate  
 @ExtendWith(PactVerificationInvocationContextProvider.**class**)  
 **void** verifyPact(PactVerificationContext context) {  
 context.verifyInteraction();  
 }  
  
 @State(**"book with ISBN 101201 exists"**)  
 **void** toBookWithIsbn101201ExistsState() {  
 System.***out***.println(**"book with ISBN 101201 exists"**);  
 }  
}