dreamforce 20 Years

GROUNDSWELL CLOUD SOLUTIONS
a GyanSys company

GS Apex Mocking Framework

Improving Your Apex Unit Tests

Leonardo Berardino, Principal Developer
leonardo@gscloudsolutions.com

Leonardo Berardino

Principal Developer
Groundswell Cloud Solutions Inc
(a GyanSys Company)

Why Did We Build This?

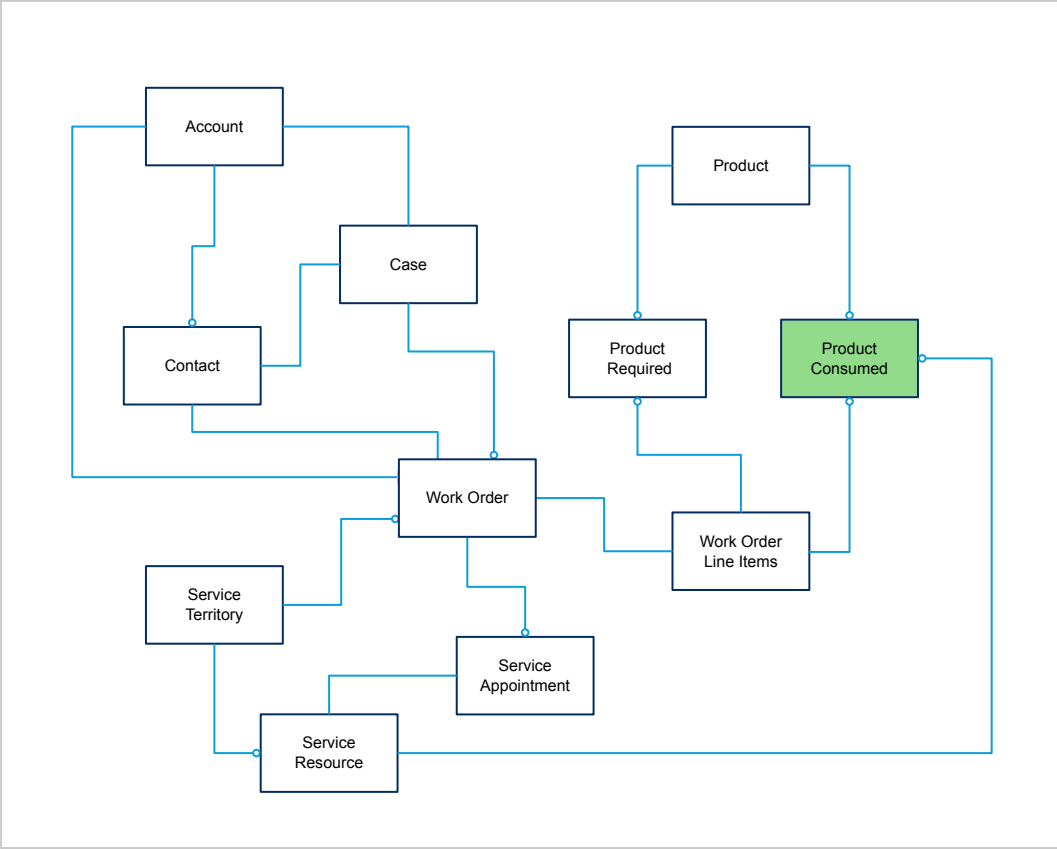
Common Issues with Apex Unit Testing

Long Deployment Cycles!!

Common Issues with Apex Unit Testing

Complex Test Setup

- Complex Data Models
- Every Test Seems To Be An Integration Test



Integration Tests

For integrated modules and functional testing.

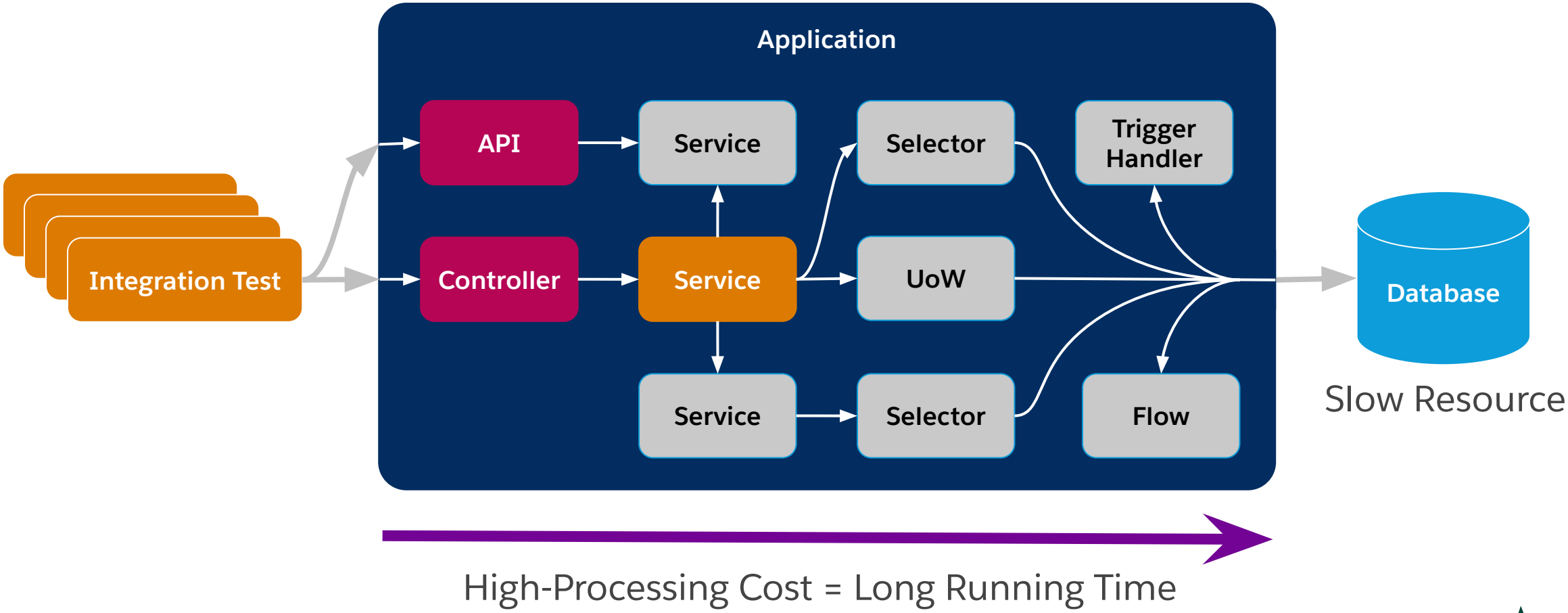
Integration Tests

What Are They?

- The integration tests are responsible for assessing if different modules from the application work well together.
- And for evaluating the system's compliance with the functional requirements.

Integration Tests

What Do They Test?



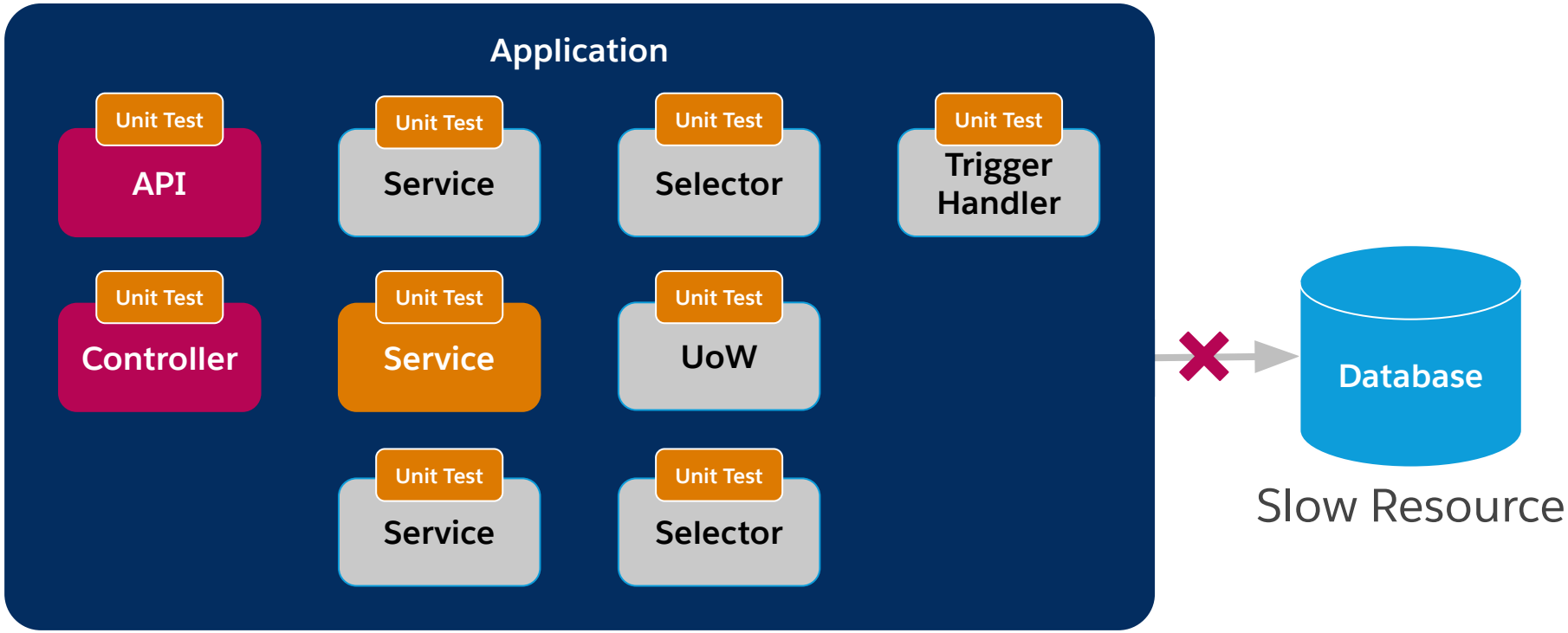
Unit Tests

For individual modules testing.

Unit Tests

What Are They?

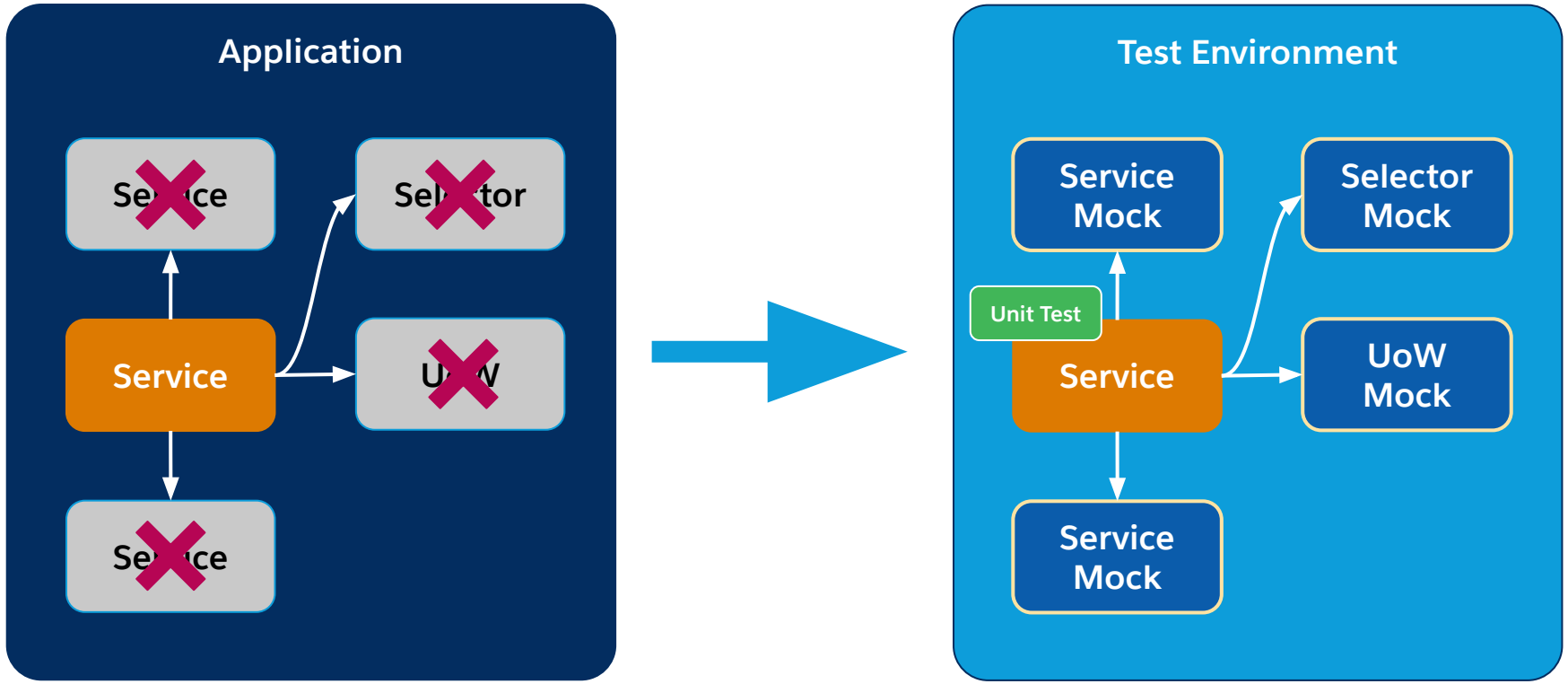
- The unit tests are low-level and close to the application source code.
- They consist of testing individual components isolating them from the other components.



Unit Tests

How Do They Deal With Dependencies?

- We can replace the real dependencies by **Mocked Classes**.
- But how can we create **Mock Classes**?



Unit Tests x Integration Tests

- Unit Tests don’t replace Integration Tests
- Both are different tools to ensure the high quality of our code

Unit Tests

Integration Tests

Apex Stub API

Native Salesforce Apex Stub API

Allow us:

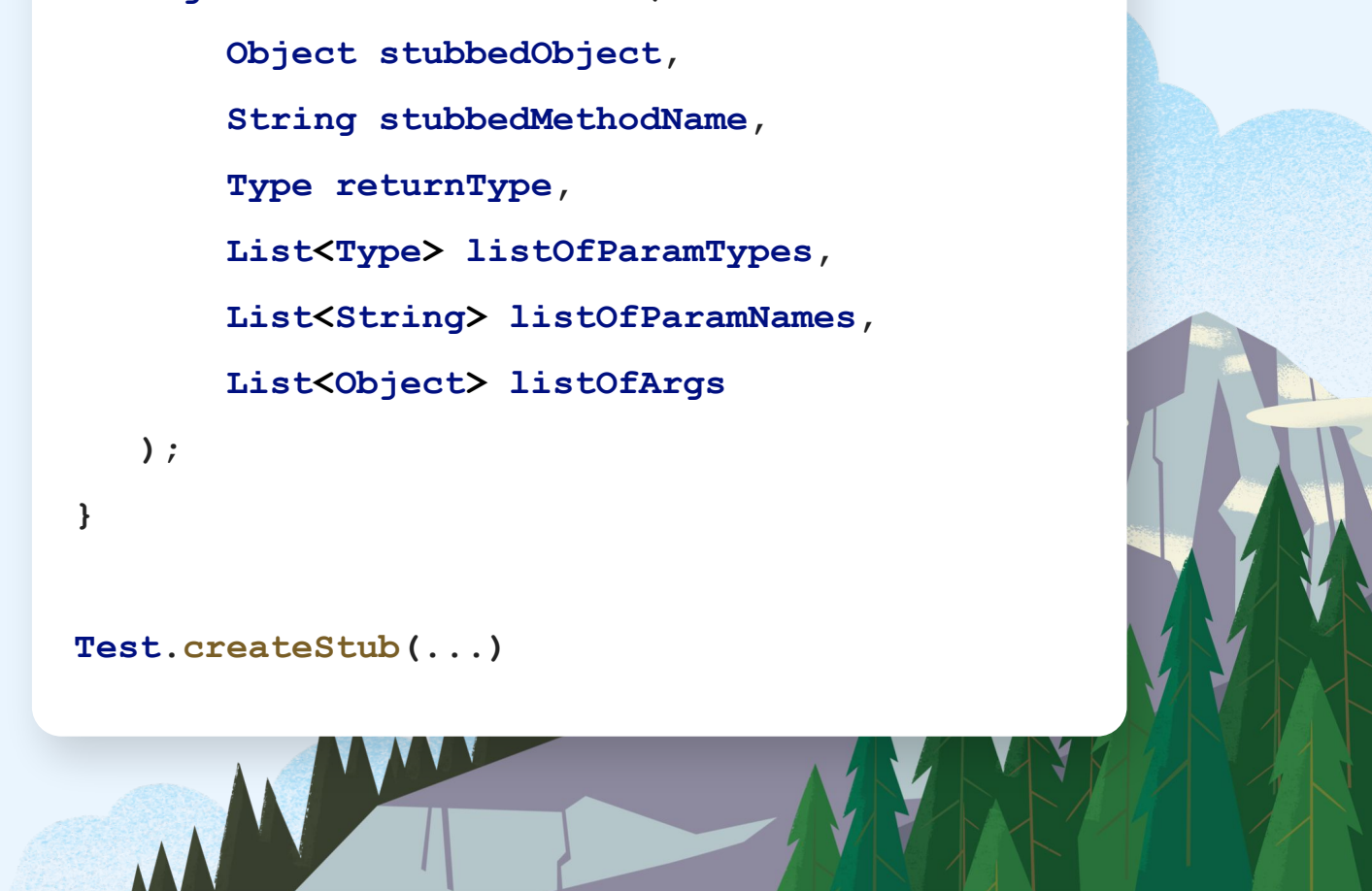
- To create mocked objects
- To handle method calls
- To read the method arguments
- To define the method return value

However:

- It is a low-level API
- It is necessary to implement the Stub Provider interface
- It is not easy to understand and use

```
public interface System.StubProvider {
    Object handleMethodCall(
        Object stubbedObject,
        String stubbedMethodName,
        Type returnType,
        List<Type> listOfParamTypes,
        List<String> listOfParamNames,
        List<Object> listOfArgs
    );
}

Test.createStub(...)
```

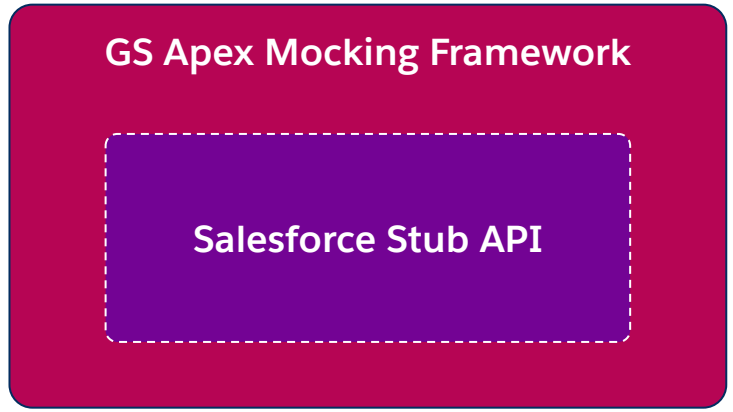


GS Apex Mocking Framework

An Open-Source Easy-To-Use Mocking Framework

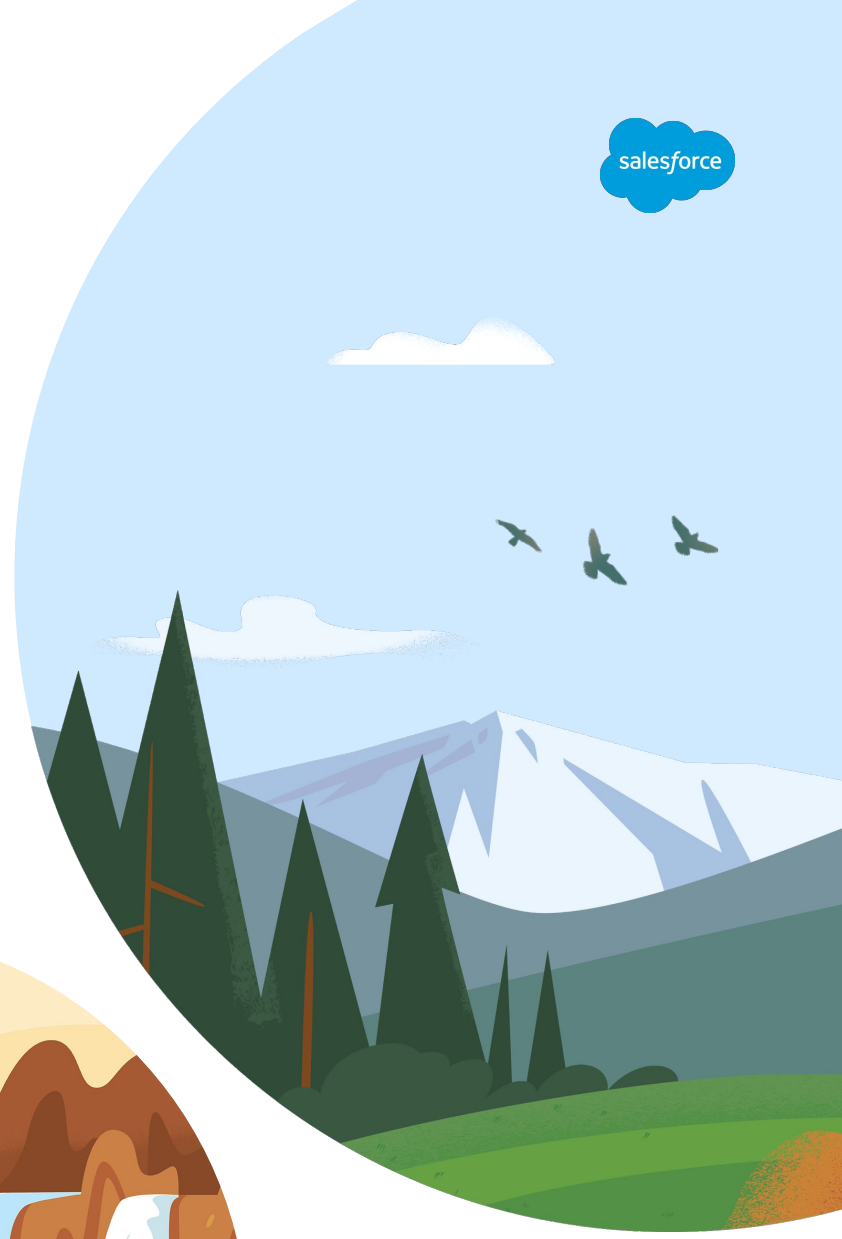
GS Apex Mocking Framework

- It encapsulates the STUB API complexity exposing an easy-to-use fluent interface.
- It records the expected mock behavior and replicates this behavior during the test execution phase.
- It also collects the execution statistics that can be asserted on the assertion phase.

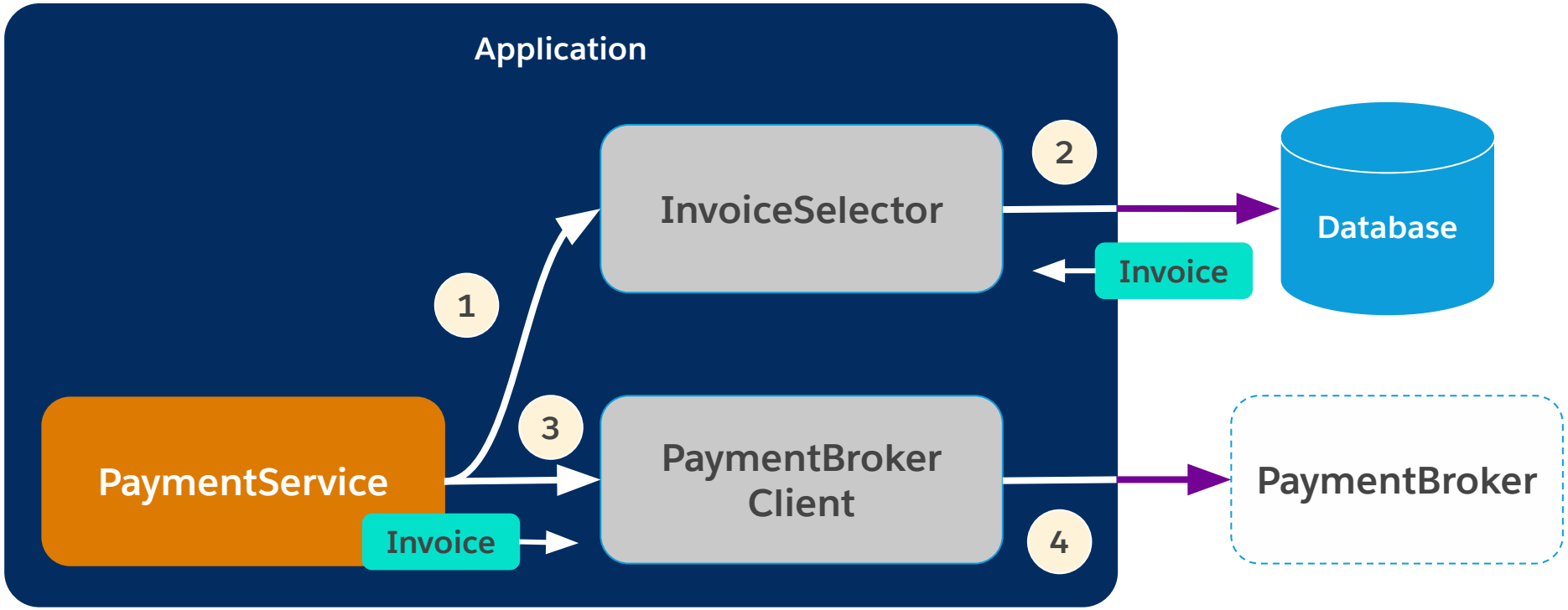


Creating An Unit Test

With the GS Apex Mocking Framework

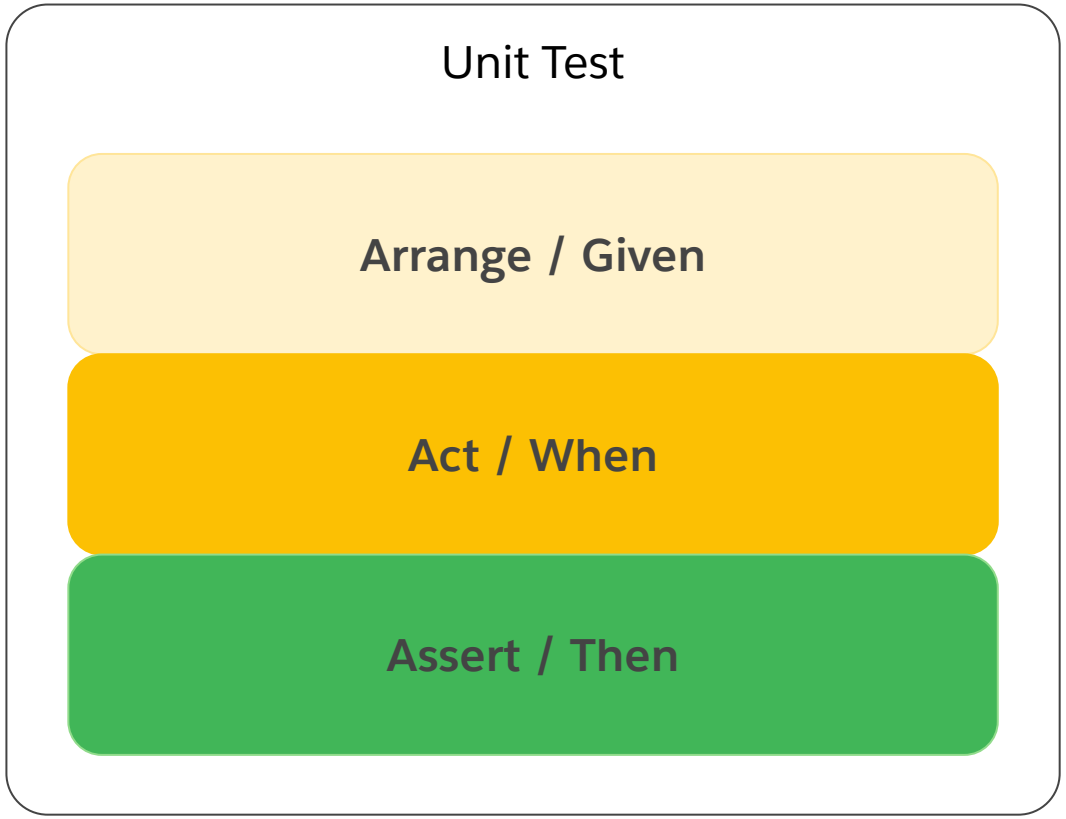


Unit Test Scenario

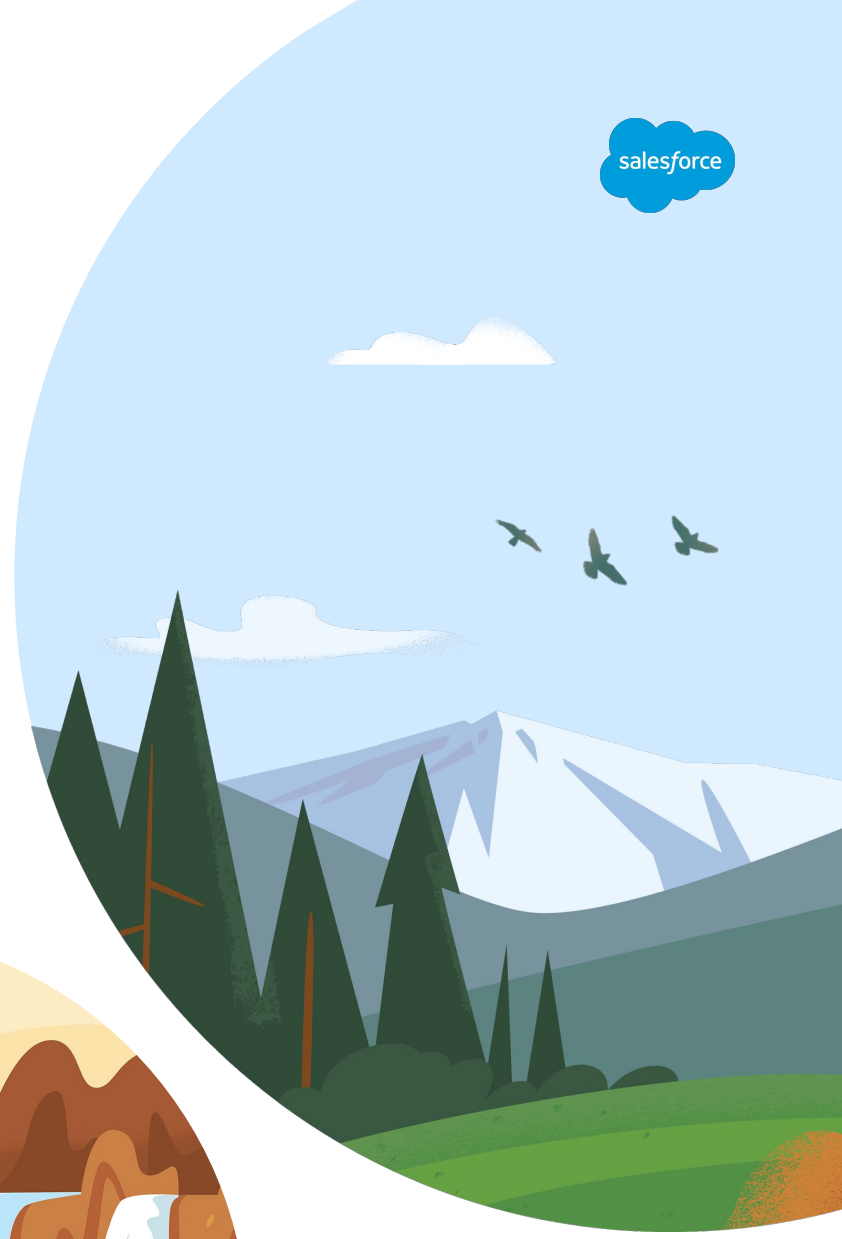


- Has the Payment Broker been called once?
- Has the Payment Broker received the correct Invoice?

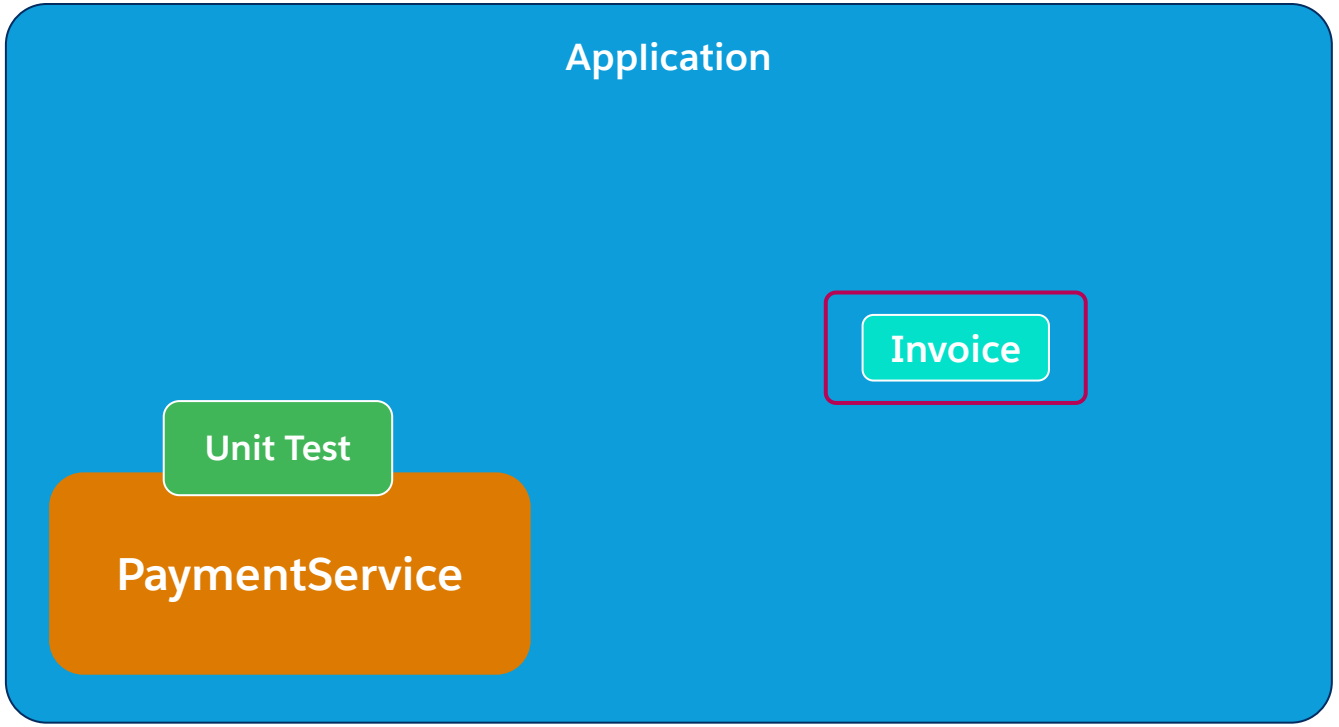
Unit Testing Pattern



Unit Test - Arrange



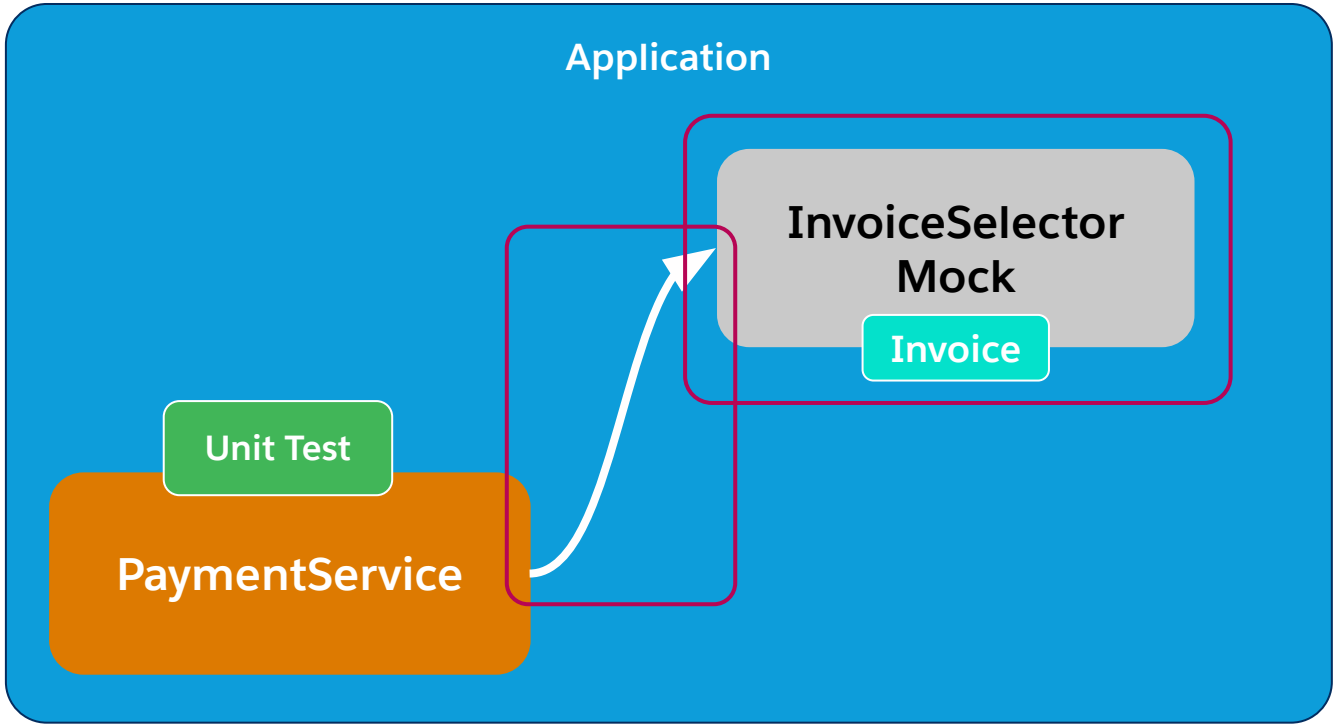
Creating The Invoice In Memory



Creating The Invoice In Memory

```
@IsTest
public class PaymentServiceTest {
    @IsTest
    static void processPaymentShouldCallBrokerWhenInvoiceIsValid() {
        // Generating a fake Invoice Id and creating the Invoice in memory
        Id invoiceId = MockerUtils.generateId(Invoice__c.SObjectType);
        // Creating the Invoice in Memory
        Invoice__c invoice = new Invoice__c(
            Id = invoiceId,
            State__c = 'Open',
            TotalAmount__c = 1,000.00,
            ... );
```

Create The Invoice Selector Mock



Create The Invoice Selector Mock

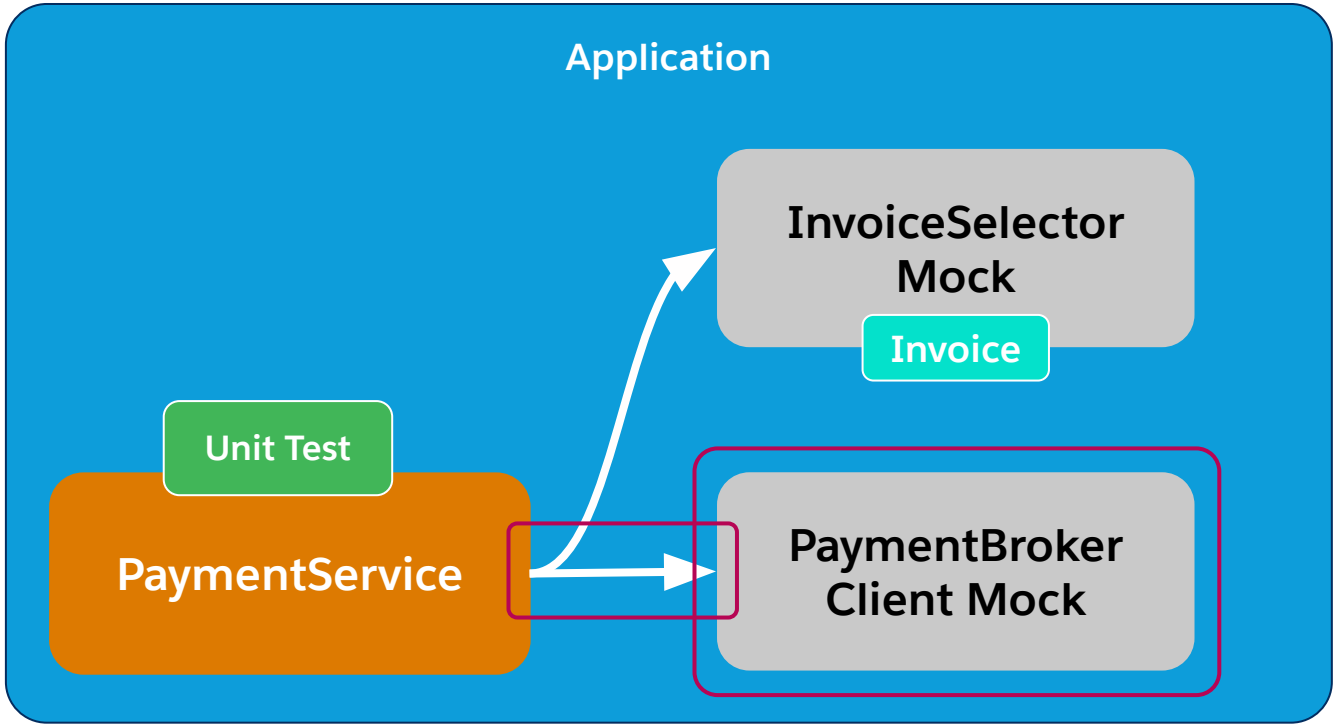
```
// Starting the recording phase
Mocker mocker = Mocker.startStubbing();
```

```
// Creating the Invoice Selector Mock
InvoiceSelector selectorMock = (InvoiceSelector) mocker.mock(InvoiceSelector.class);
```

```
// Recording the return value for the getInvoice method
mocker.when(selectorMock.getInvoice(invoiceId)).thenReturn(invoice);
```

```
// Injecting the Invoice Selector Mock into the Payment Service
PaymentService.invoiceSelector = selectorMock;
```

Create The Payment Broker Client Mock



Create The Payment Broker Client Mock

```
// Creating the Payment Broker Cliner mock
PaymentBrokerClient brokerClientMock = mocker.mock(PaymentBrokerClient.class);
```

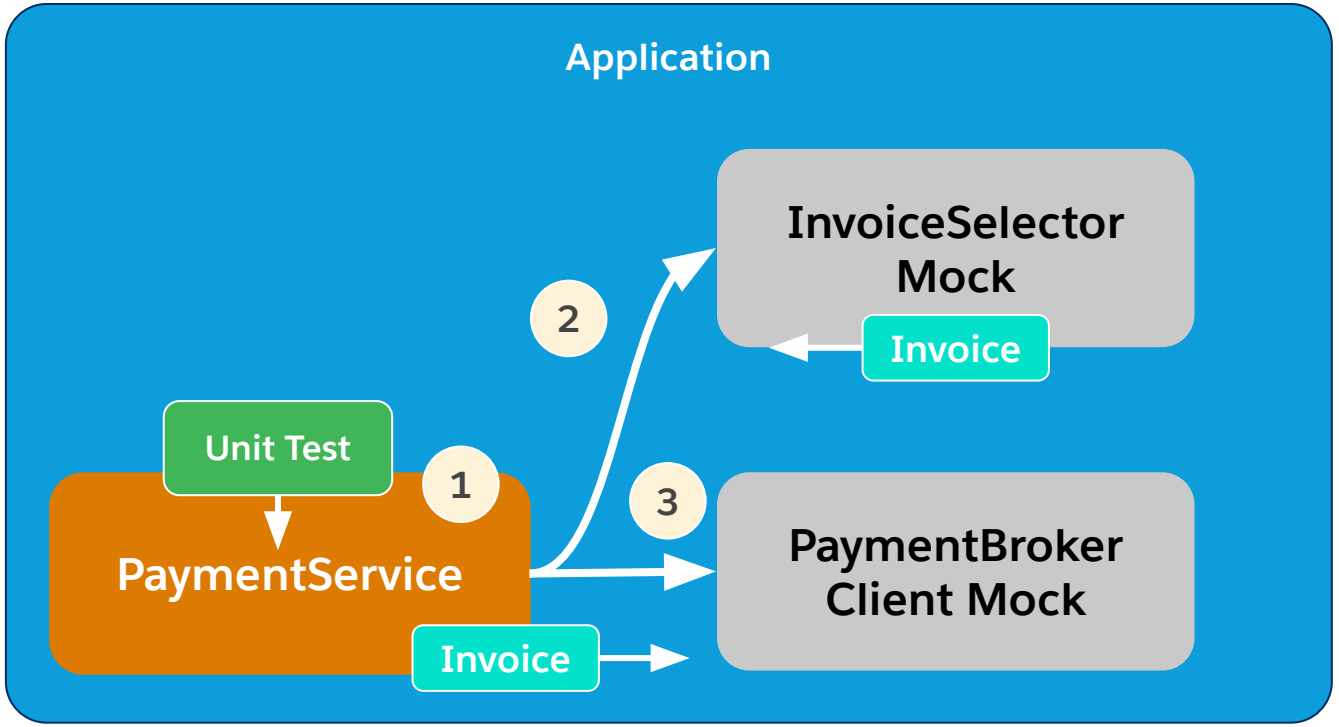
```
// Recording the method behavior
brokerClientMock.processPayment(invoiceId);
Mocker.MethodRecorder processPaymentRec = mocker.when().getMethodRecorder();
```

```
// Injecting the Payment Broker Client Mock into the Payment Service
PaymentService.brokerClient = brokerClientMock;
```

```
// Stopping the recording phase and going to the execution phase
mocker.stopStubbing();
```

Unit Test - Act

Creating Invoice In Memory



Calling The Method Unter Test

```
// Calling The Method Unter Test
Test.startTest();
new PaymentService().processPayment(invoiceId);
Test.stopTest();
```

Unit Test - Assert

Asserting The Results

- Has the Payment Broker been called once?

```
// Asserting how many calls the method
// PaymentBrokerClient.processPayment(Invoice__c invoice) has received
System.assertEquals(1, processPaymentRec.getCallsCount());
```

- Has the Payment Broker received the correct Invoice?

```
// Asserting the argument invoice received on the
// PaymentBrokerClient.processPayment(Invoice__c invoice) call
System.assertEquals(
    invoice, processPaymentRec.getCallRecording(1).getArgument('invoice')
);
```

GS Apex Mocking Framework

Other Features

GS Apex Mocking Framework

Other features

```
// It can have different return values depending on the argument value
mocker.when(selectorMock.getInvoice('0I1000000111AAA'))
      .thenReturn(new Invoice__c(Amount__c = 1000.0));


mocker.when(selectorMock.getInvoice('0I1000000222AAA'))
      .thenReturn(new Invoice__c(Amount__c = 2000.0));
```

GS Apex Mocking Framework

Other features

```
// It can ignore the argument value
mocker.when(selectorMock.getInvoice(null))
      .withAnyValues()   // <====
      .thenReturn(new Invoice__c(Amount__c = 5000.0));


// It can also simulate an exception
mocker.when(selectorMock.getInvoice('0I1000000333AAA'))
      .thenThrow(new DmlException('Record not found'));
```

GS Apex Mocking Framework

Other features

```
// We can define expected behaviors
mocker.when(selectorMock.getInvoice('0I1000000444AAA'))
      .thenReturn(new Invoice__c(Amount__c = 4000.0))
      .shouldBeCalledOnce();

mocker.when(selectorMock.getInvoice('0I1000000555AAA'))
      .shouldNeverBeCalled();
```

Open Source GS Apex Mocking Framework

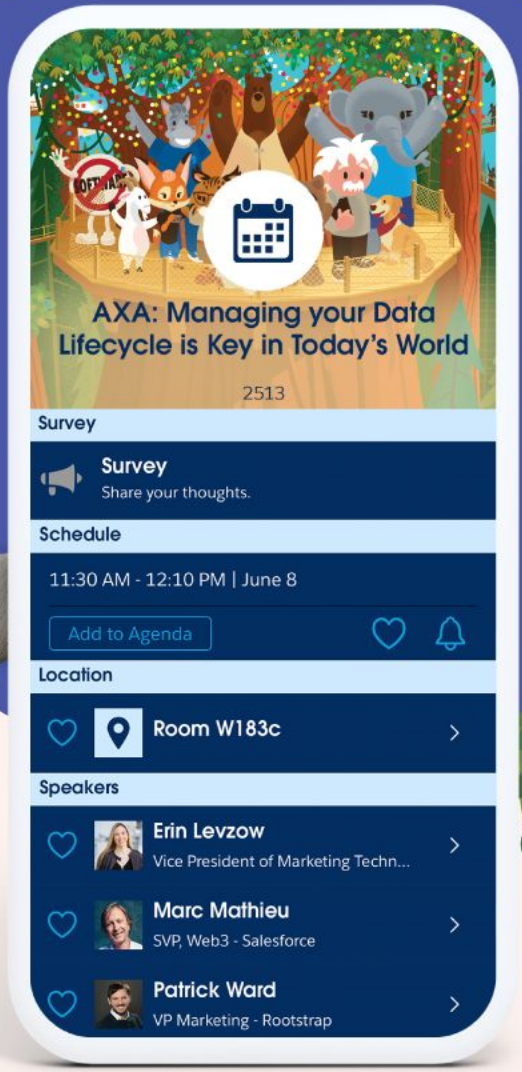
Blog Post

https://gscloudsolutions.com/tips-tricks/gs-apex-mocking-framework

dreamforce 20 Years

Share your feedback.

Provide your feedback on this session in the Salesforce Events mobile app and help make our content even better.



#DF22

Groundsweller's at Dreamforce

We're all wearing "**G**" pins so come say hello!

Savio Jose

Product Practice Manager

Presenting: ***Best Practices for Data Security in Experience Cloud***

Cameron Reid

Emerging Technologies Lead

Presented: ***Diagramming for the Admin***

Pei Huang

CTO

Presenting: ***Named Credentials: Securing & Simplifying API Callouts***

3 pm today!

Gerauld Rivera

Marketing Cloud Product Lead

Colin Hamilton

Field Service Product Lead

dreamforce

Thank you