Best Practices for Naming Integration Tests

Integration tests focus on verifying the interactions between different components of the system. The naming conventions for integration tests are similar to those of unit tests but with added emphasis on interactions between components and external dependencies.

Naming Conventions for Integration Tests

1. MethodUnderTest_WithComponentA_AndComponentB_ShouldReturnExpectedOutcome

For integration tests involving multiple components, the name should indicate the components and the expected outcome.

Example:

```
public void
GetUserDetails_WhenCalledWithValidId_ShouldFetchDataFromServiceAndDatabase()
```

- **Method Under Test**: `GetUserDetails`
- **Components Involved**: `Service` and `Database`
- **Expected Behavior**: `ShouldFetchData`

---

2. ComponentA_Integration_WithComponentB_ShouldPerformExpectedBehavior

When testing the integration of specific components, reflect the names of those components in the test name.

Example:

```
public void
OrderService_Integration_WithPaymentGateway_ShouldProcessOrderSuccessfully()
```

- **Components**: `OrderService` and `PaymentGateway`
- **Expected Behavior**: `ShouldProcessOrderSuccessfully`

---

3. ApiEndpoint_WhenCalled_ShouldReturnExpectedResponse

For API integration tests, it's useful to reference the endpoint being tested and the expected response.

Example:

```
public void CreateOrderApi_WhenCalledWithValidData_ShouldReturnSuccessStatus()
```

- **API Endpoint**: `CreateOrderApi`
- **Expected Behavior**: `ShouldReturnSuccessStatus`

---

4. WhenDependencyIsUnavailable_ShouldHandleErrorGracefully

For tests that check how the system handles failure of external dependencies, the test name should reflect that failure scenario.

Example:

```
public void
PaymentService_WhenDatabaseIsUnavailable_ShouldThrowDatabaseException()
```

- **Service**: `PaymentService`
- **State**: `WhenDatabaseIsUnavailable`
- **Expected Behavior**: `ShouldThrowDatabaseException`

Additional Best Practices

External Dependencies

Integration tests often involve interactions with external systems like databases, file systems, or external APIs. It's helpful to highlight these dependencies in the test name.

Example:

```
public void GetUserDetails_WhenCalledWithValidId_ShouldRetrieveDataFromDatabase()
```

Multiple Components

When integration tests involve multiple components, make sure to reflect those interactions in the test name.

Example:

```
public void
OrderService_Integration_WithInventoryAndPaymentServices_ShouldCompleteOrderSucces
sfully()
```

Failure Scenarios

Integration tests should cover both success and failure cases. When dealing with unreliable systems or services, test names should indicate the failure scenarios being tested.

Example:

```
public void UserRegistration_WhenEmailServiceFails_ShouldLogError()
```

Summary for Integration Tests Naming:

- **[MethodUnderTest]*[ComponentOrDependency]*[ExpectedOutcome]**
- **[ComponentA]*[IntegrationWithComponentB]*[ExpectedOutcome]**

This structure emphasizes both the components being tested and the behavior expected from their interaction.