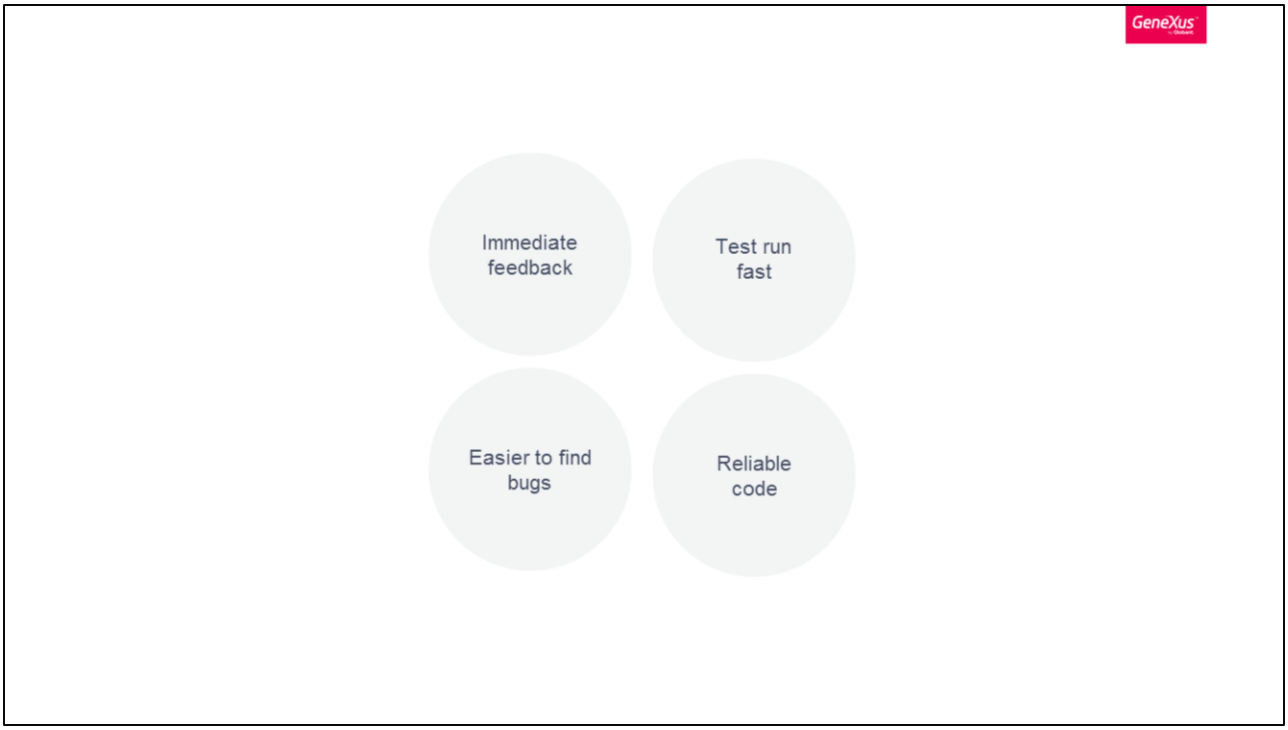

GeneXus™
by Globant

ユニットテスト機能

GeneXus by Globant

ユニットテストの有用性

GeneXus by Globant



GeneXusの開発者は、コードやルールを書いた後、何らかの方法でプロシージャを実行し、デバッグする必要があります。
従来は、自作のウェブパネルやUIを使用して入出力をテストしました。この時、異なる入力を使用して実行し、出力を評価します。
ビジネスロジックを検証するための開発者の努力を再利用可能にするUnit Testオブジェクトを使用することで、このすべての労力を削減（および自動化）することができます。

ユニットテストは、プロシージャが変更されビルドされるたびに、GeneXusが開発環境で自動的にテストを実行し、フィードバックを開発者に提供します。
そのため、バグのある変更をコミットし、デプロイ環境で確認する必要がなくなります。

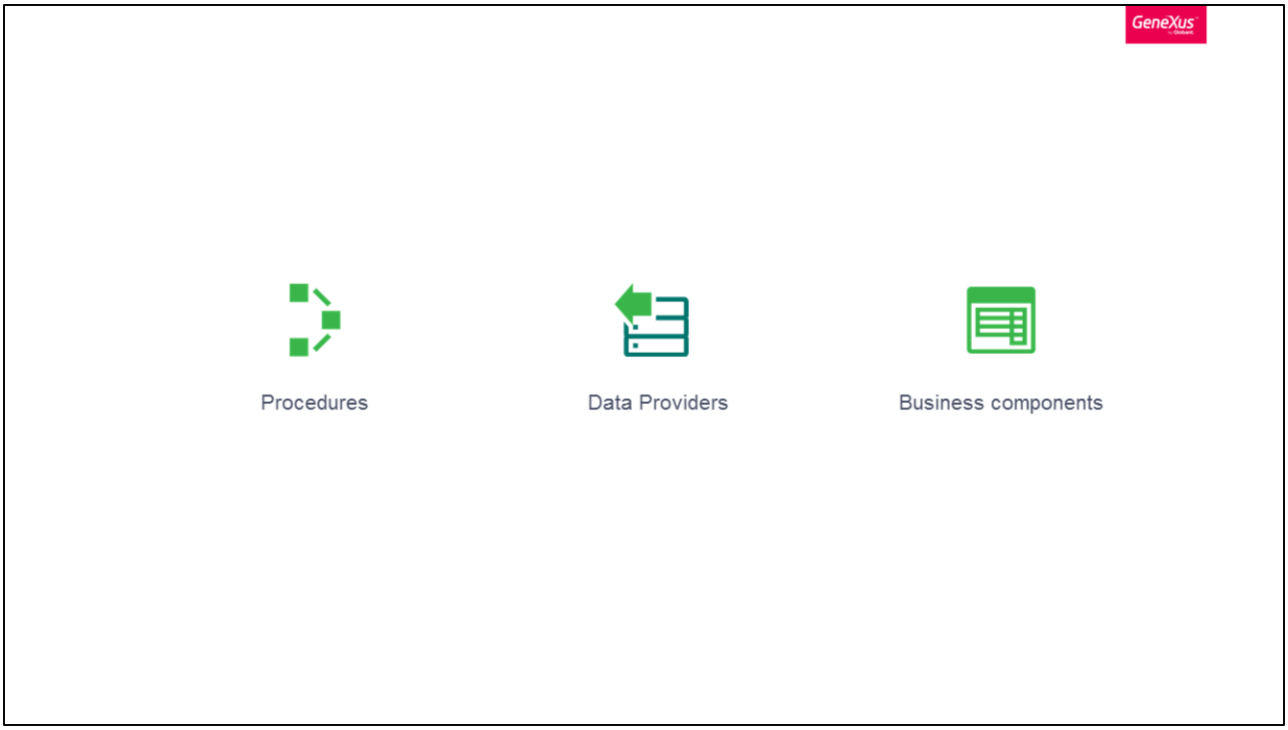
これにより、環境を構築し、ソリューションをデプロイすることなく、GeneXusコードのバグを検出する機会を得ることができます。
ユニットテストは、ソフトウェアの他の部分に影響を与えることなく、将来的に多くのコストとバグを防ぐために、開発プロセスの早い段階で問題を検出するのに役立ちます。

さらに、バグがシステム全体ではなく、コードの単位で発見された場合、バグの検出と修正が容易になります。
ユニットテストは非常に高速に実行され、GeneXusServerを通じて共有することができますが、本番環境には決してデプロイしてはいけません。

また、コードをより堅牢で、信頼性の高い、安定したものにする機会にもなります。

GeneXusでのユニットテストについて

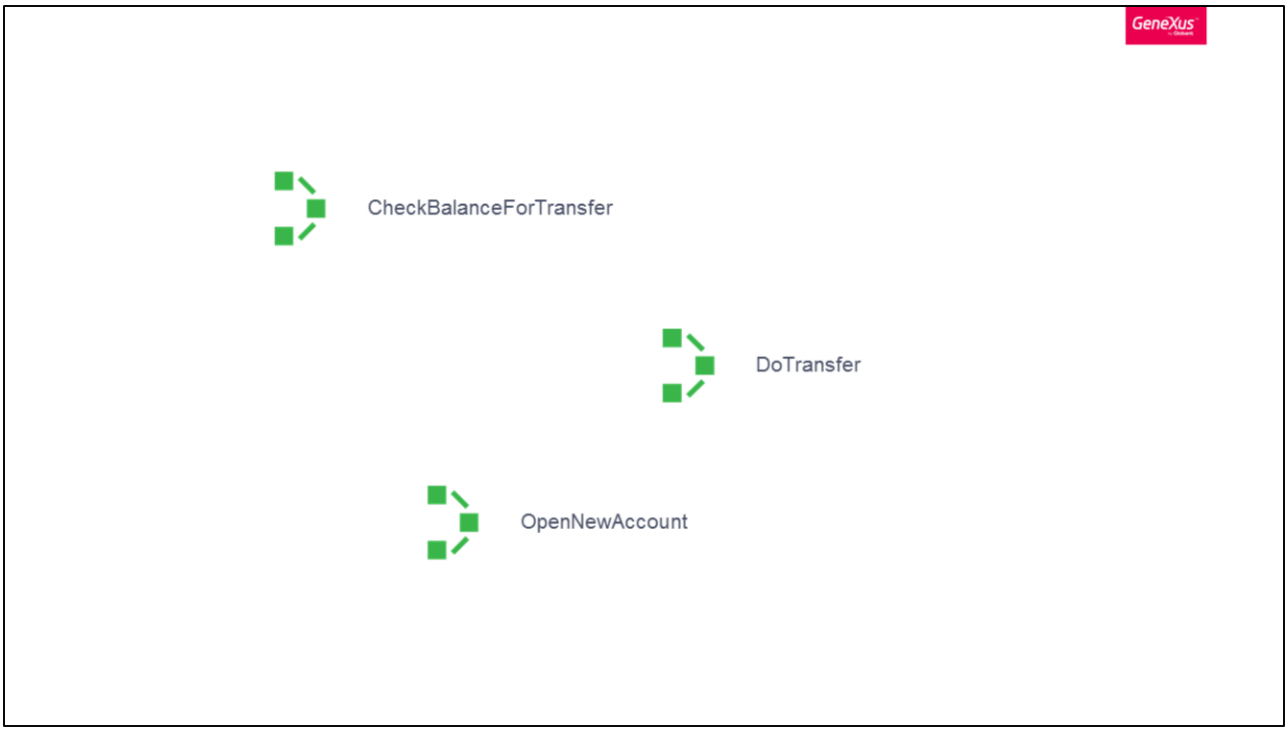
GeneXus by Globant



GeneXusでは、ビジネスロジックのコードをカプセル化し、異なるパネルで再利用する方法として、プロシージャを使用します。
そのため、ビジネスロジックをテストするためには、プロシージャにカプセル化する必要があります。
この開発習慣がない場合、アプリケーションをリファクタリングし、プログラミングの方法を変更する必要があります。

その上、アプリケーションロジックの一部は、データプロバイダーやビジネスコンポーネントに含まれています。
これらのオブジェクトの振る舞いを検証するためのテストを作成することができます。

GeneXus IDEでは、これらの非インターフェイスオブジェクトに対して、簡単にユニットテストを生成することができます。
また、異なるプロセス間の統合を検証するために、ユニットテストを作成することも可能です。

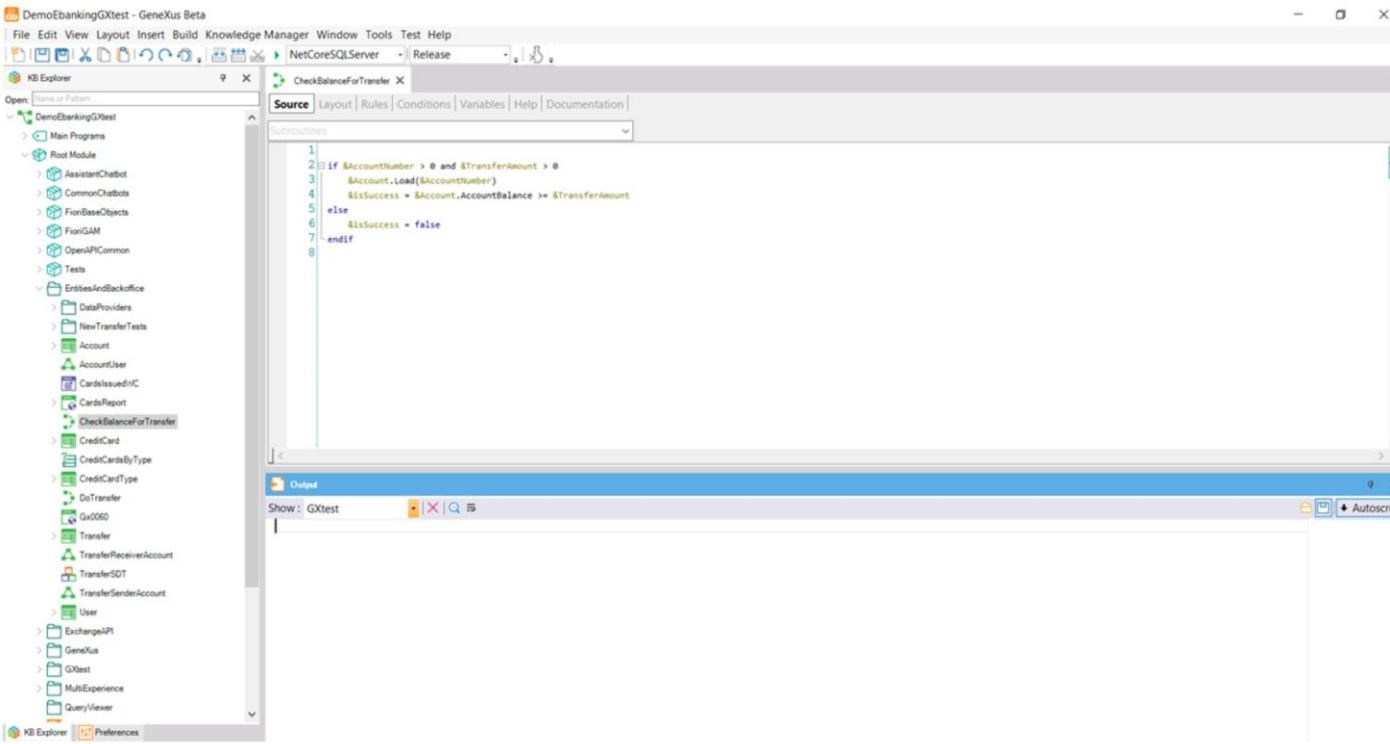


KBのユニットテストの中で最も重要なのは、プロシージャです。

KBのロジックを構成するすべてのプロシージャに対して、それがどのように動作するかを（プロシージャー一つ一つについて）原子的に検証するユニットテストを作成します。
必要であれば、プロシージャ/機能の統合をテストするために、複数のプロシージャを呼び出すユニットテストを作成します。

ユニットテストを作成する

GeneXus by Globant

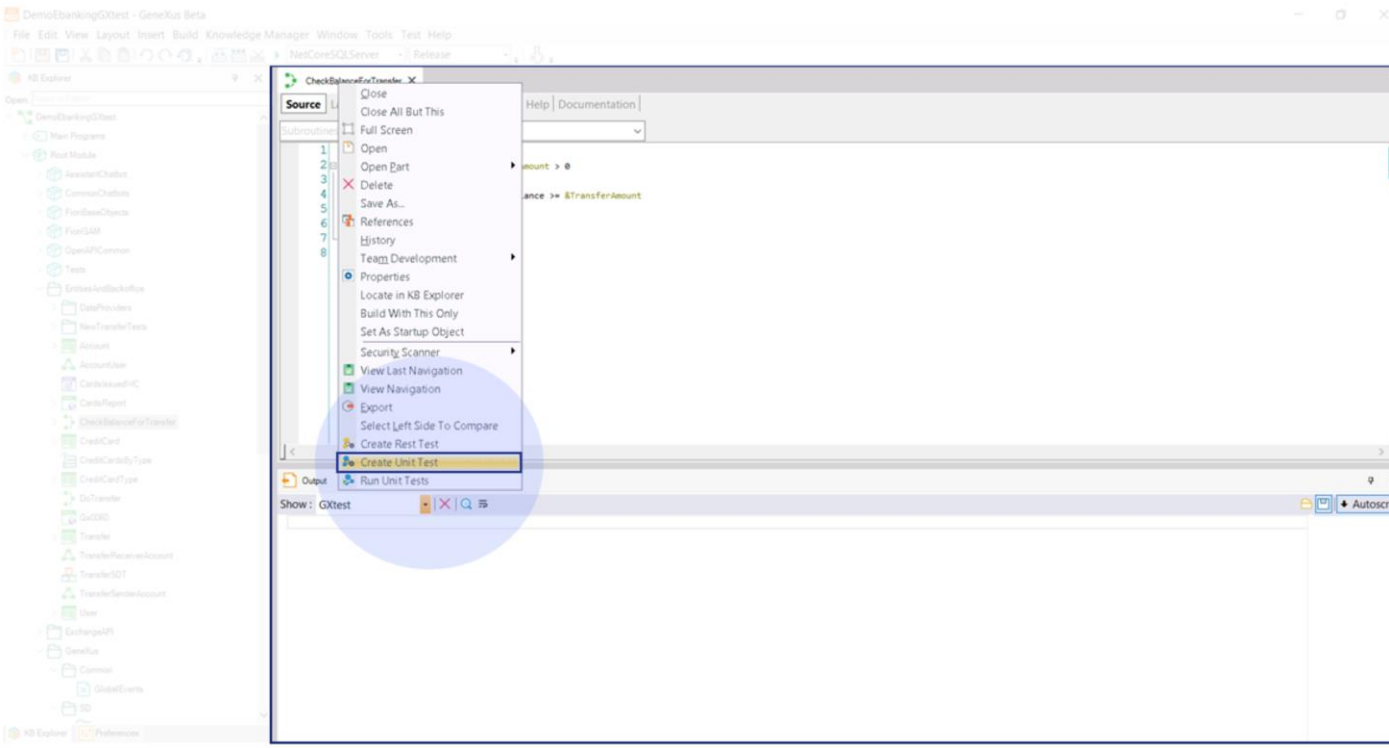


CheckBalanceForTransferというプロシージャがあるとします。
これは2つの入力パラメータ（口座番号と送金額）と、送金の承認/拒否を示すブール型変数を出力パラメータとして持っています。

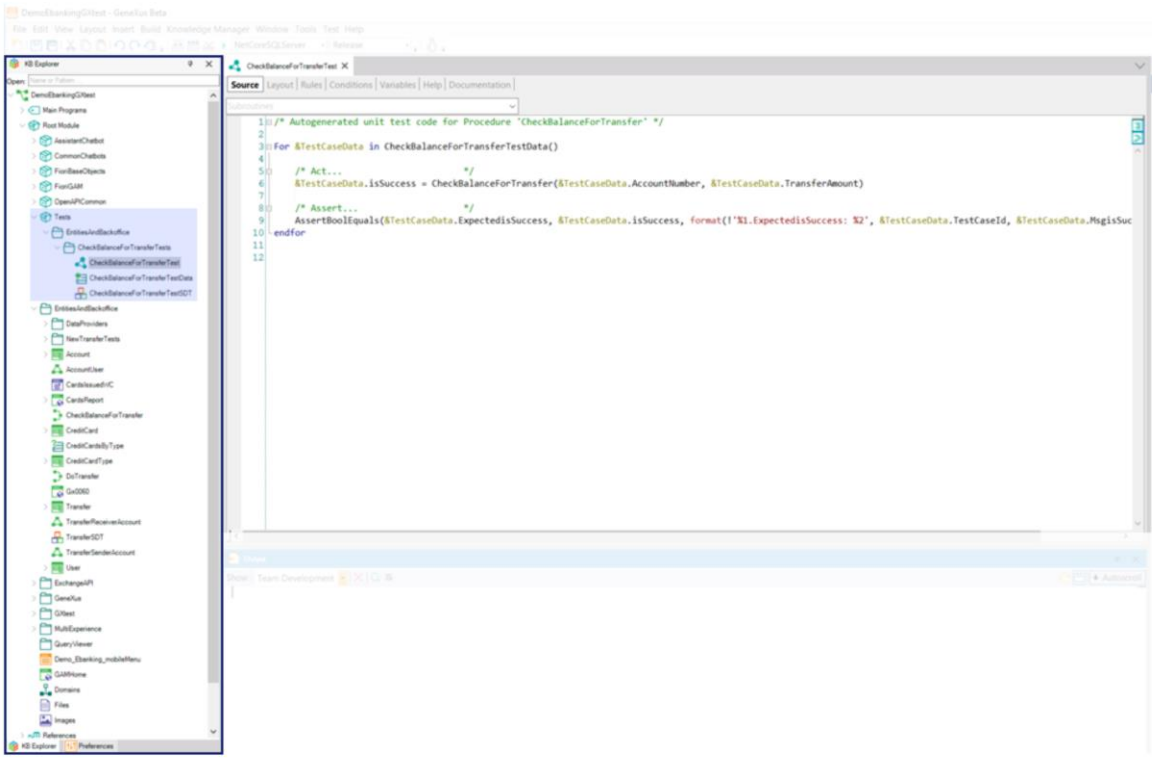
このプロシージャは、口座の有効性と残高に応じて、転送を承認または拒否します。

そのため、開発者として、プロシージャを呼び出すための補助パネル、ボタン、入力フィールドを使った手動テストを行う代わりに、開発者はテストケースを使ったユニットテストを作成する必要があります。

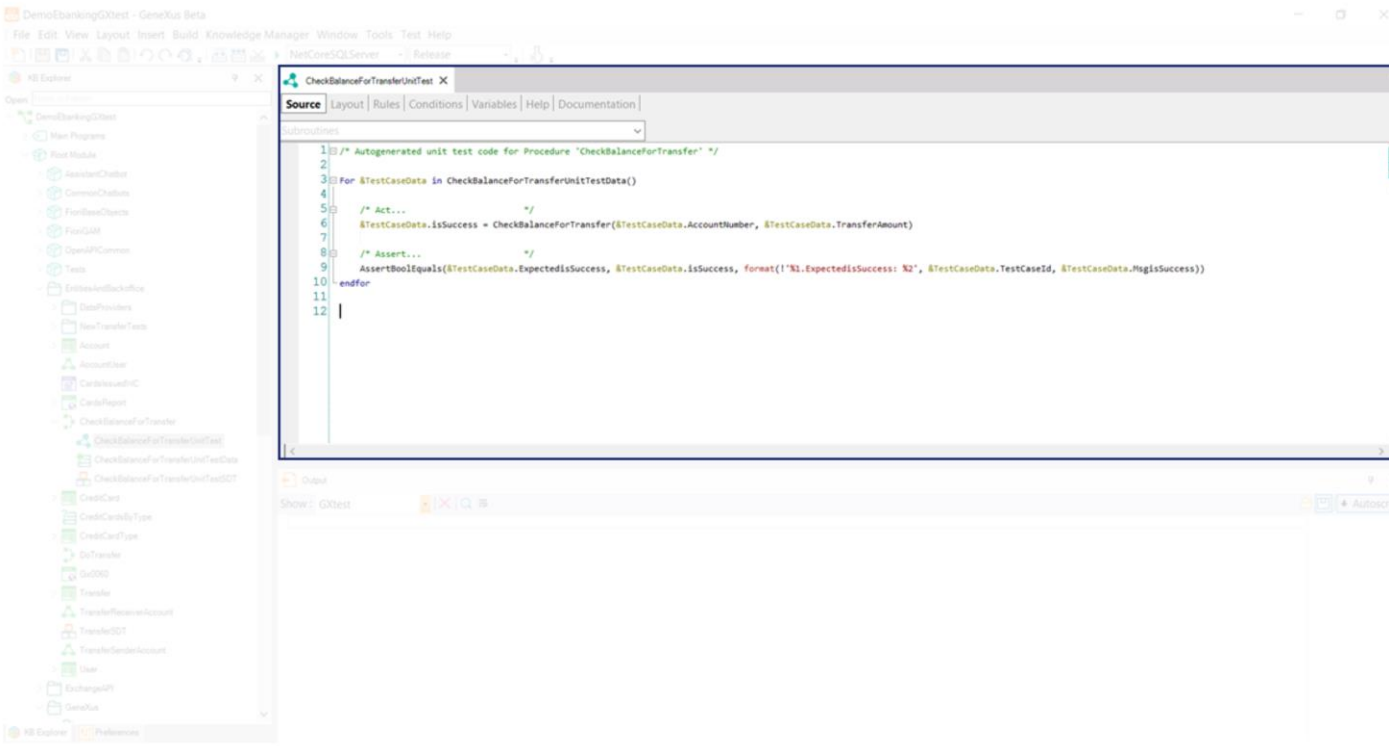
それでは、GeneXusで簡単にユニットテストを作成する方法をご紹介しましょう。



Unit Testオブジェクトを作成するには、対象のProcedureオブジェクトを開き、タブを右クリックするか、
またはKBエクスプローラで対象のProcedureオブジェクトを右クリックし、「ユニットテストの作成」オプションを選択するだけです。



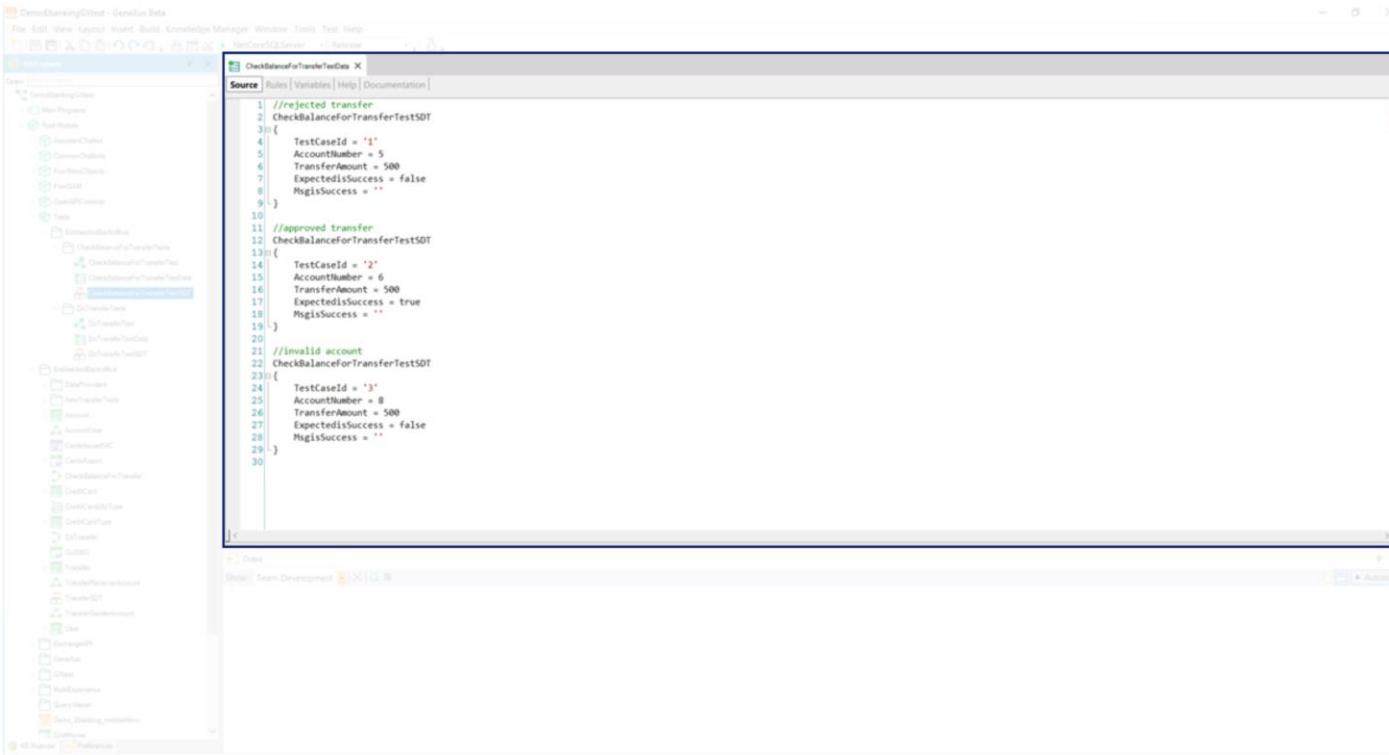
この機能によって、モジュールに含まれる形でテストオブジェクトを自動的に生成します。
自動生成されたオブジェクトは、Unit Testオブジェクト、テストケースを定義するデータプロバイダ、テストケース構造を持つSDTです。



今回の例では、生成されたユニットテストのテンプレートがテストケースコレクション CheckBalanceForTransferUnitTestDataを繰り返し参照し、その中に含まれるテストケースを利用していることがわかります。

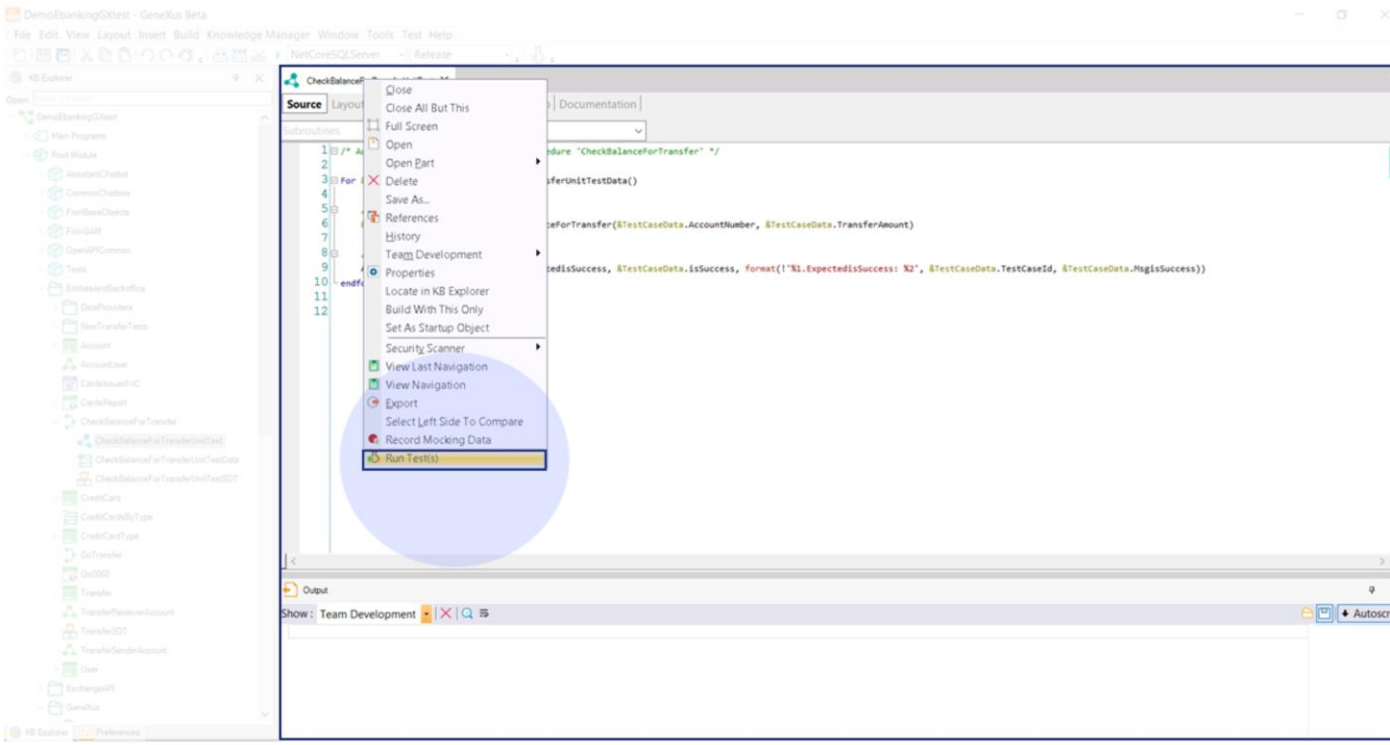
それぞれの反復において、ユニットテストはテストしたいプロシージャを呼び出し、アサーション関数で期待される値を検証します。アサーションとは、テストの合格・不合格を定義する仕組みのことです。
アサーション関数は、アサーションするパラメータの型 (ブール、数値、文字列) に応じてさまざまなものがあります。

この例では、アサーションは自動的に生成され、 プロシージャの出力パラメータを検証していることに注目してください。

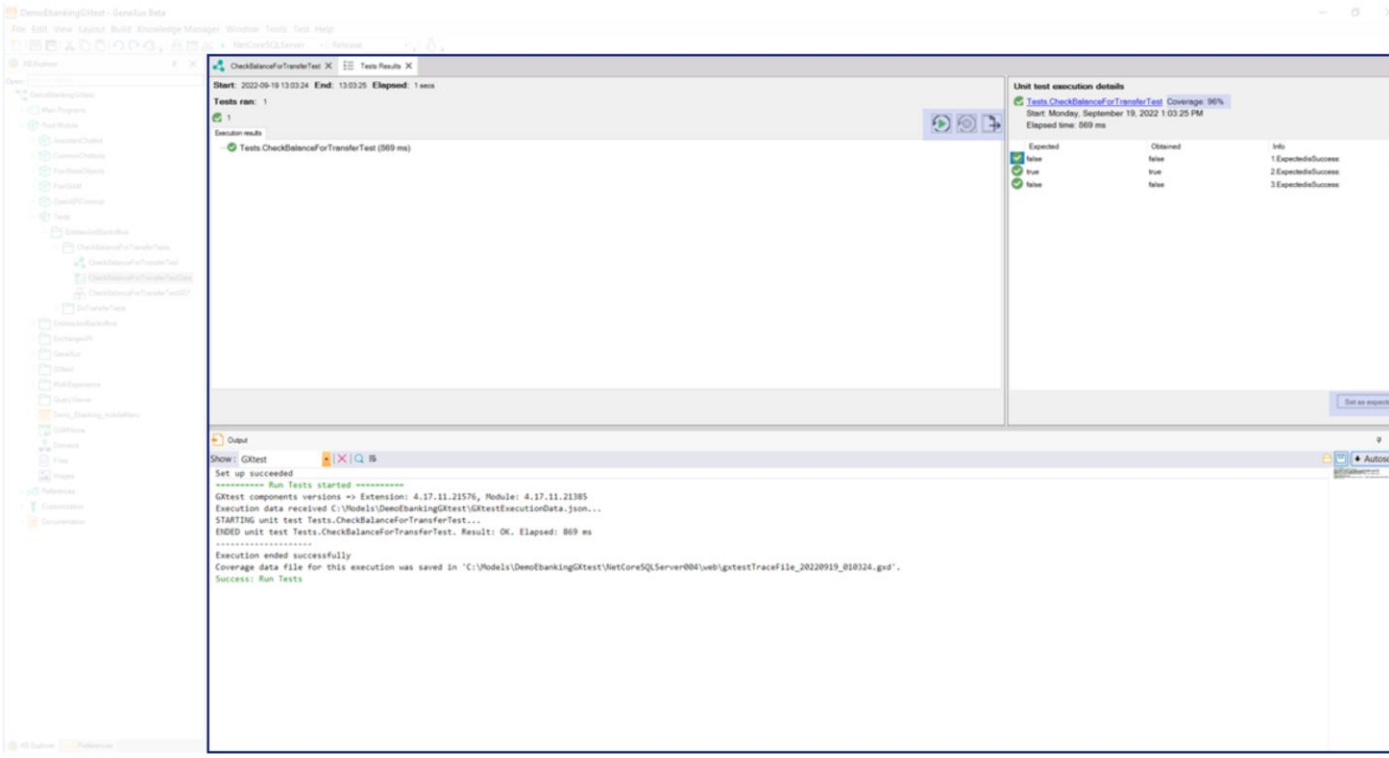


あとは、データプロバイダーにテストケースを追加するだけです。

この例では、口座番号5のテストケース1は、残高のない口座からのものなので、転送は拒否されなければなりません。
テストケース2は、残高のある有効な口座からのものなので、転送は承認されなければなりません。
そして最後に、口座番号8のテストケース3は、無効な口座番号なので、これも拒否されなければなりません。



テストを実行するには、開いたUnit Testオブジェクトのタブを右クリックし、"テストの実行"オプションを選択します。
ビルド処理が行われ、ビルドが成功するとテストが実行されます。



実行後、テスト結果パネルで結果や経過時間の情報を見ることができます。
また、各テストについて、アサーションごとの期待値と実際の取得値を可視化することができます。

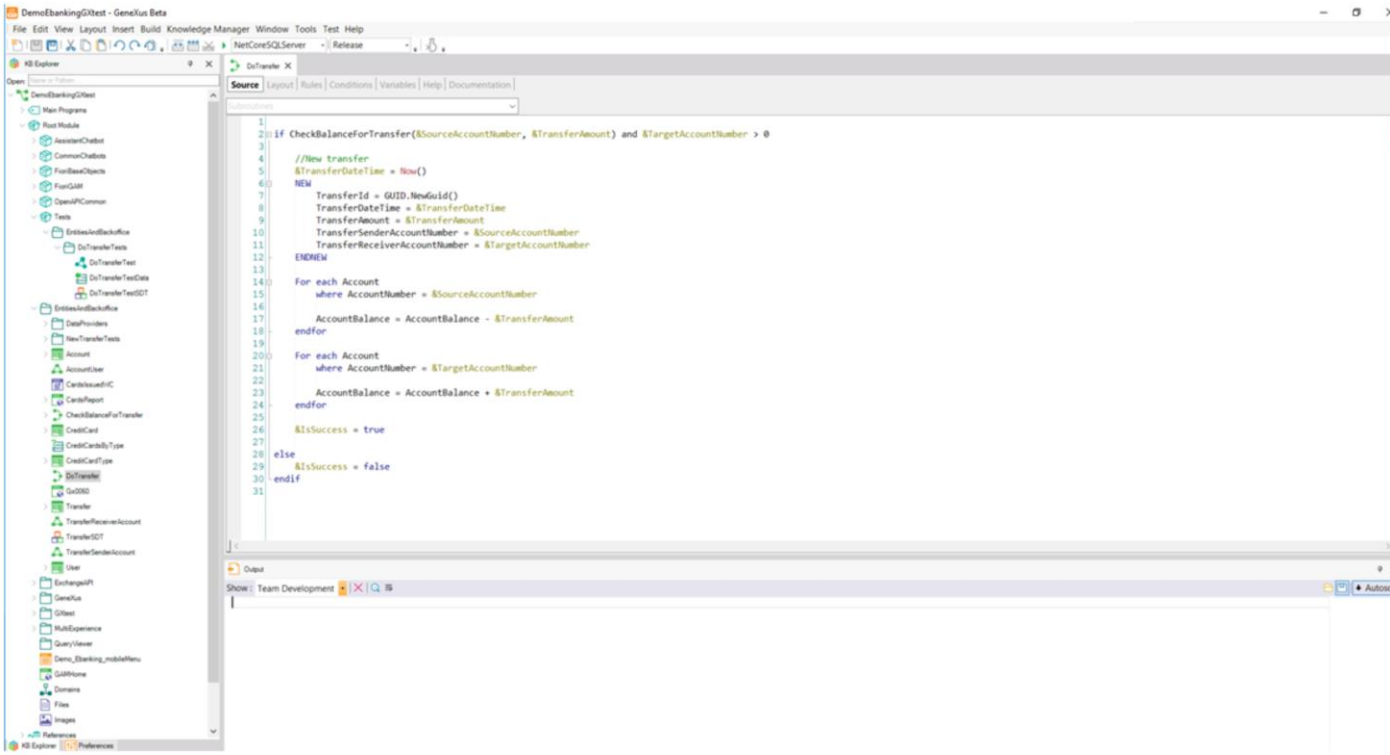
テスト結果パネルからは、「再実行」「失敗再実行」「実行結果をHTMLとしてエクスポート」などのアクションを実行することができます。

ユニットテストの実行詳細では、「テストカバレッジ」の割合や、「期待値として設定」することで、得られた値を可視化することもできます。

「テストカバレッジ」と「期待値として設定」機能については、後ほどご紹介します。

データベース検証

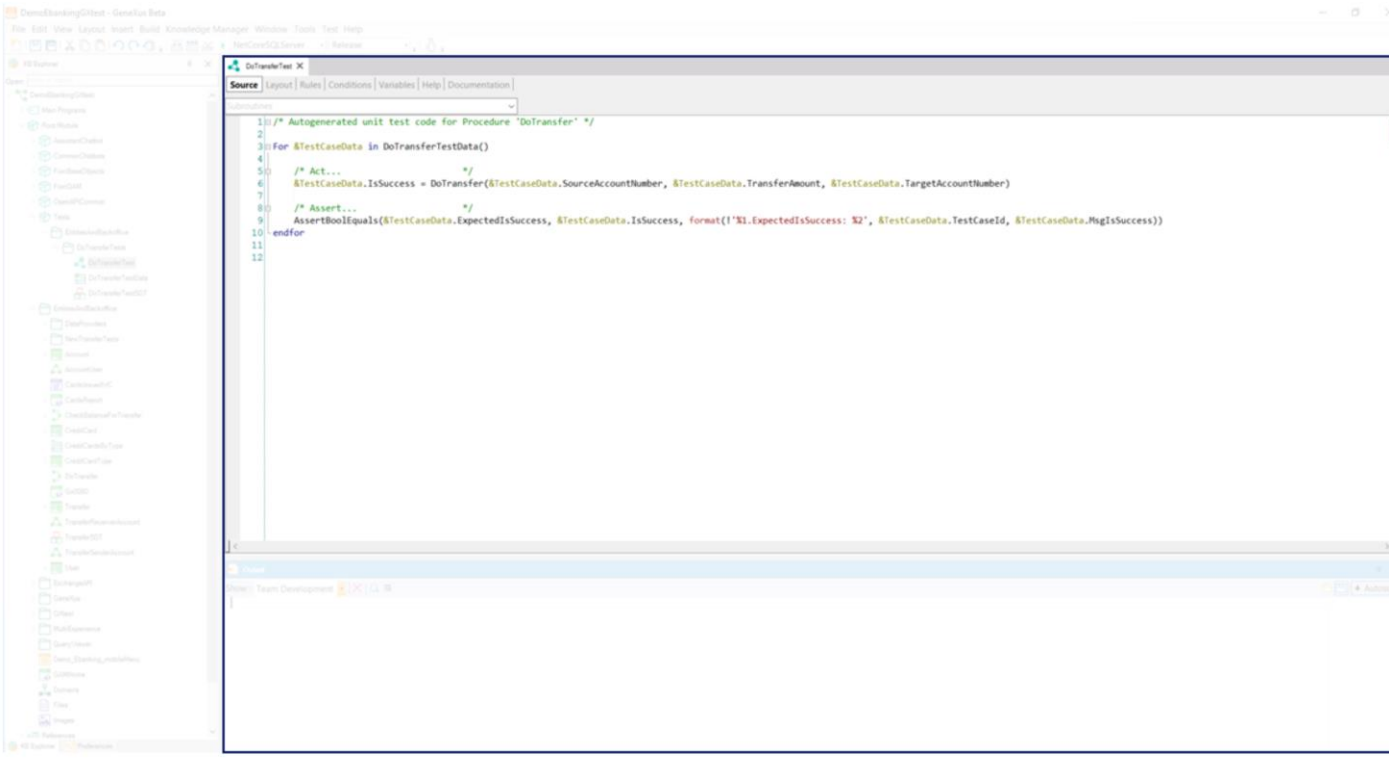
GeneXus by Globant



ユニットテストの考え方がわかったところで、データベースの検証を追加する方法を見てみましょう。

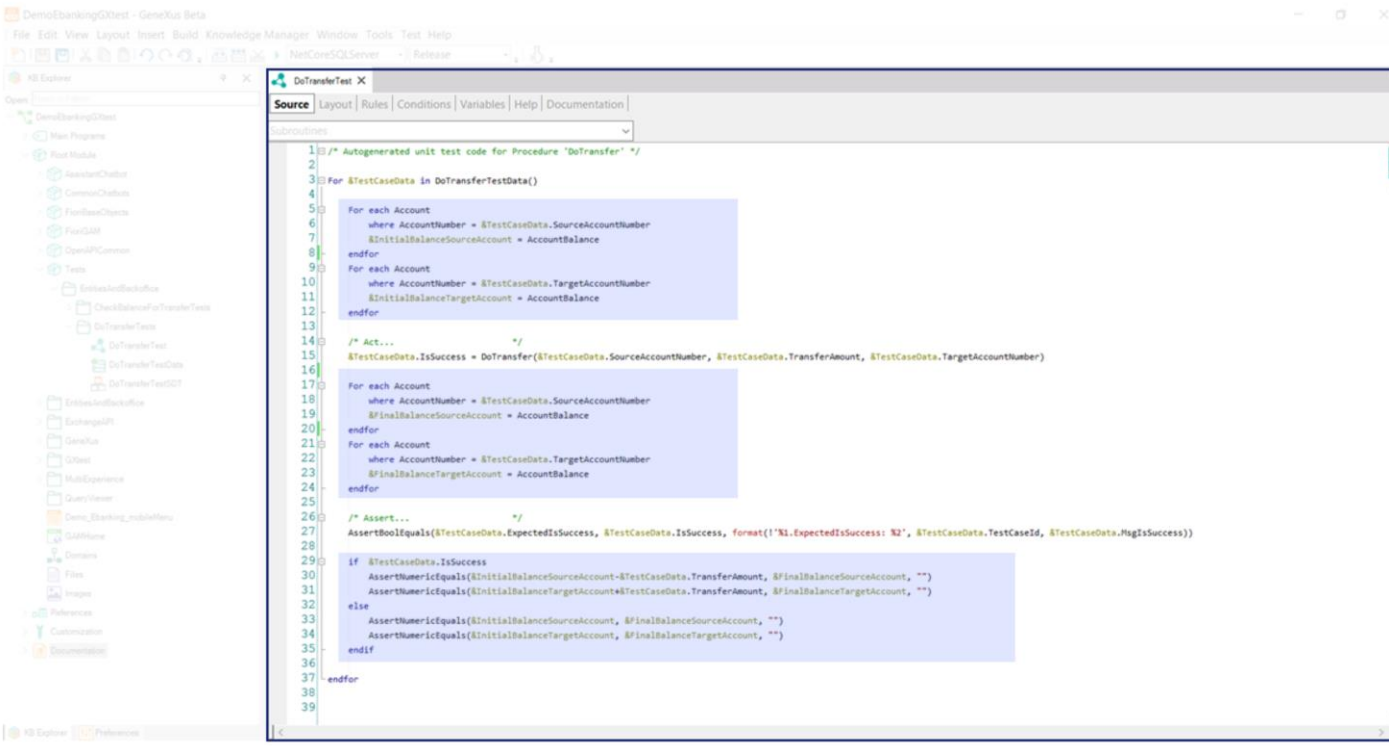
DoTransferというプロシージャがあるとします。
このプロシージャは入力パラメータとしてSourceAccountNumber, TransferAmount, TargetAccountNumber変数を持ち、出力変数としてIsSuccessを持ちます。

このプロシージャは口座の残高をチェックし、十分であれば、Transferを作成し、残高を更新します。
最後に、変数IsSuccessをTrueに設定します。



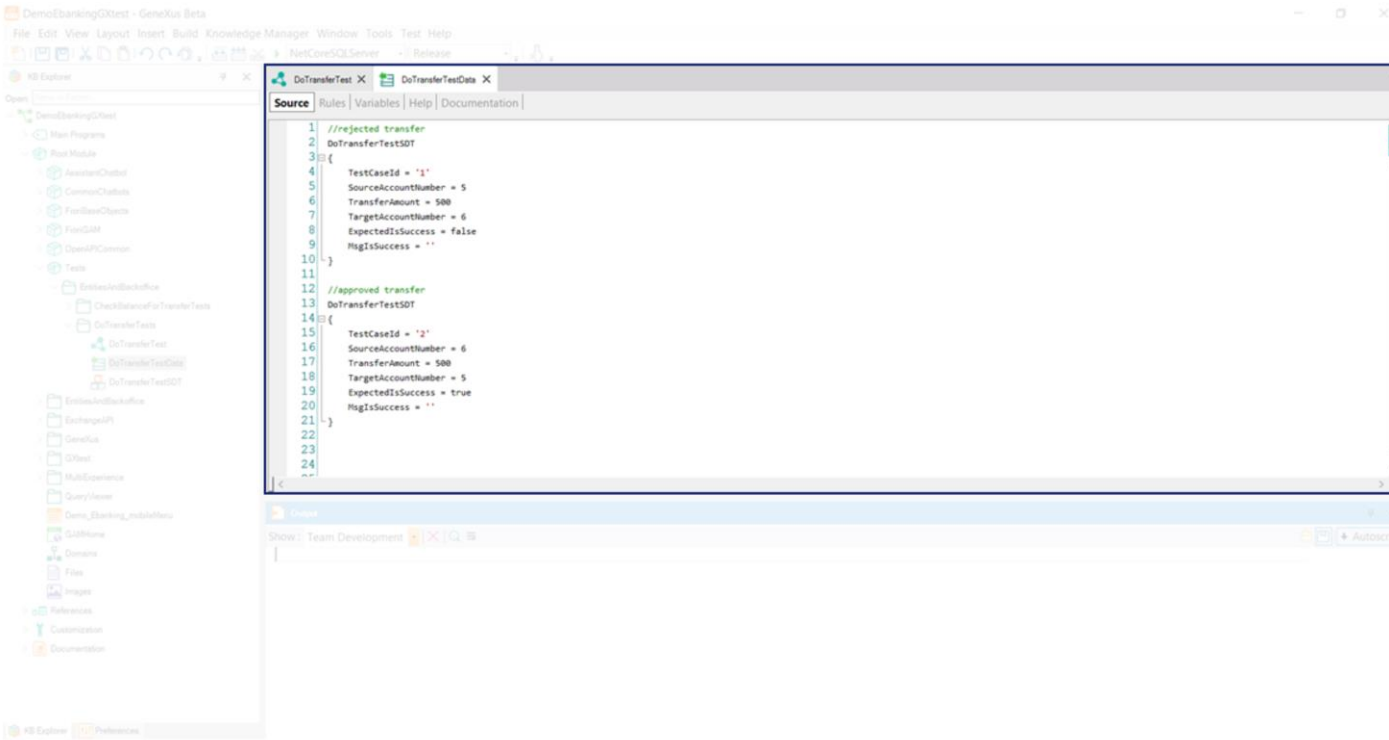
GeneXusがユニットテストを生成する際、出力パラメータのアサーションを含むテンプレートが自動生成されます。
開発者は、ユニットテスト内にデータベース検証をGeneXusコードとして追加する必要があります。

ユニットテストは特別なGeneXusプロシージャであるため、テストをより堅牢にするために必要なコードを追加することができることを忘れないでください。

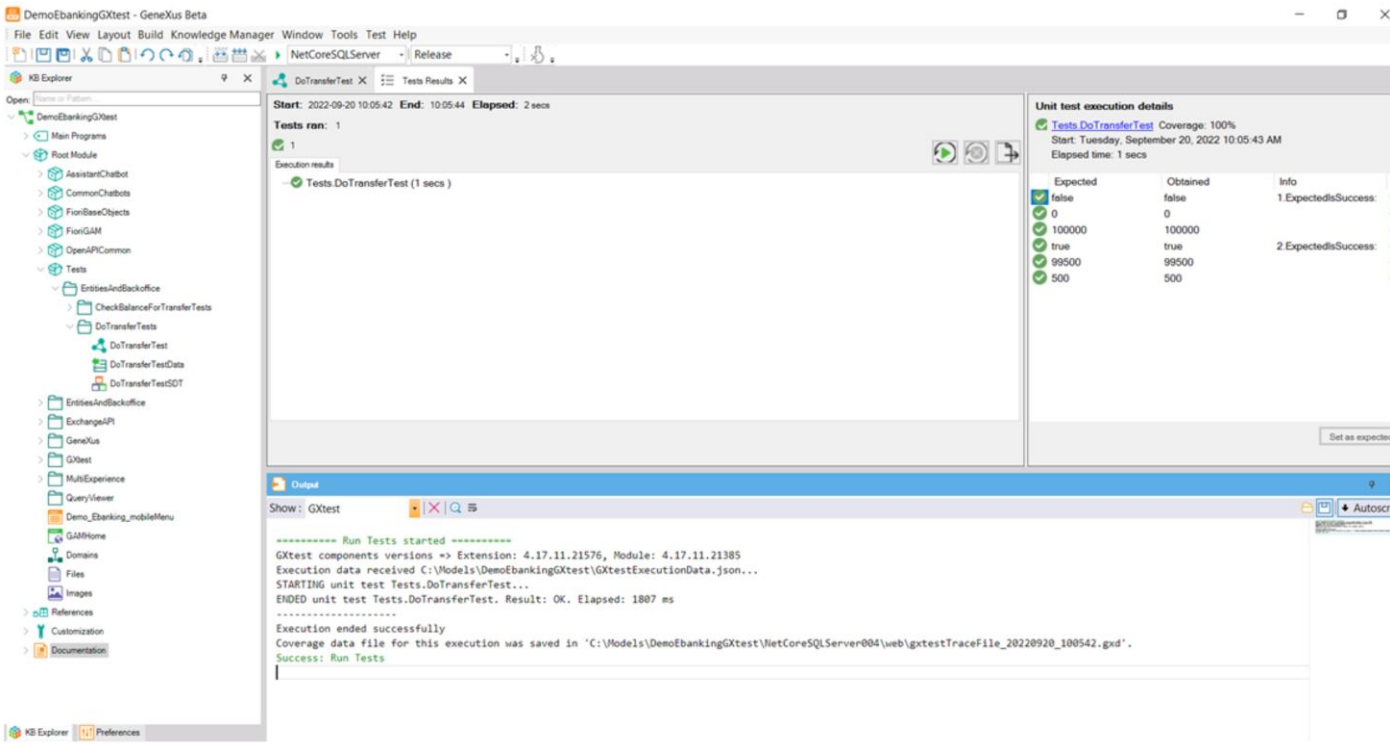


この例では、DoTransferプロシージャを実行した後、
AccountBalanceの値をアサーション関数によって検証するコードを
追加しています。

たとえば、Transferテーブルに新しい転送が作成された場合など、
もっと多くの検証を追加することができます。



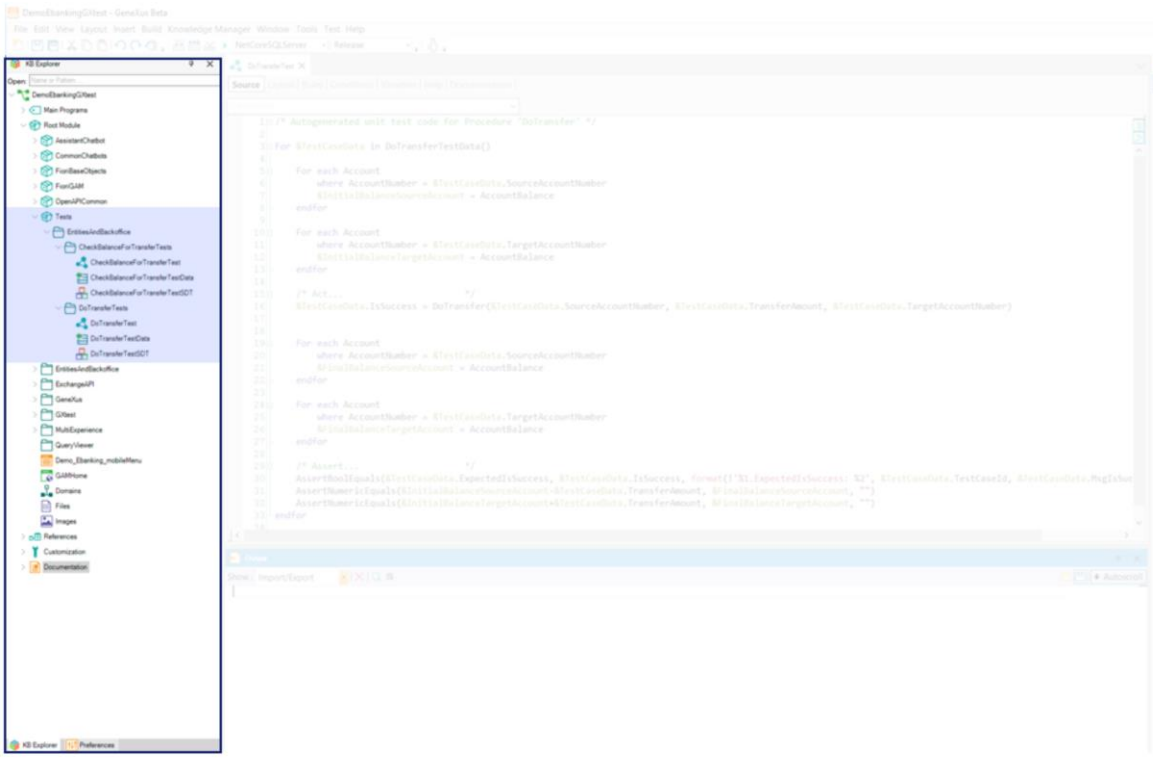
最後に、DoTransferDataProviderに、転送が拒否された場合と成功した場合のテストケースを設定します。



すべての設定が完了したら、ユニットテストを実行します。

テスト結果に、実行されたすべてのアサーションが表示されます。
各テストケースについて、出力パラメータ・AccountBalanceの値を検証したアサーションが表示されます。

[情報]列には、アサーションの三番目のパラメータが表示されます。
これは通常、アサーションを識別するためと、 今後の修正のためのコンテキストとして使用されます。



また、テストは“Tests”モジュールに格納され、各プロシージャのテストはProcedureオブジェクトにを足した名前でフォルダが作成されることに注目してください。

前に記載したように、ユニットテストでは、データプロバイダのテスト（例えば、外部のサービスやプロシージャがデータプロバイダからデータを取得する場合など）やビジネスコンポーネントのテストを実行することも可能です。
そして、異なる機能間の統合を検証するために、スタンドアロンのユニットテストを作成することが可能/必要です。


GeneXus™
by Globant
training.genexus.com
wiki.genexus.com