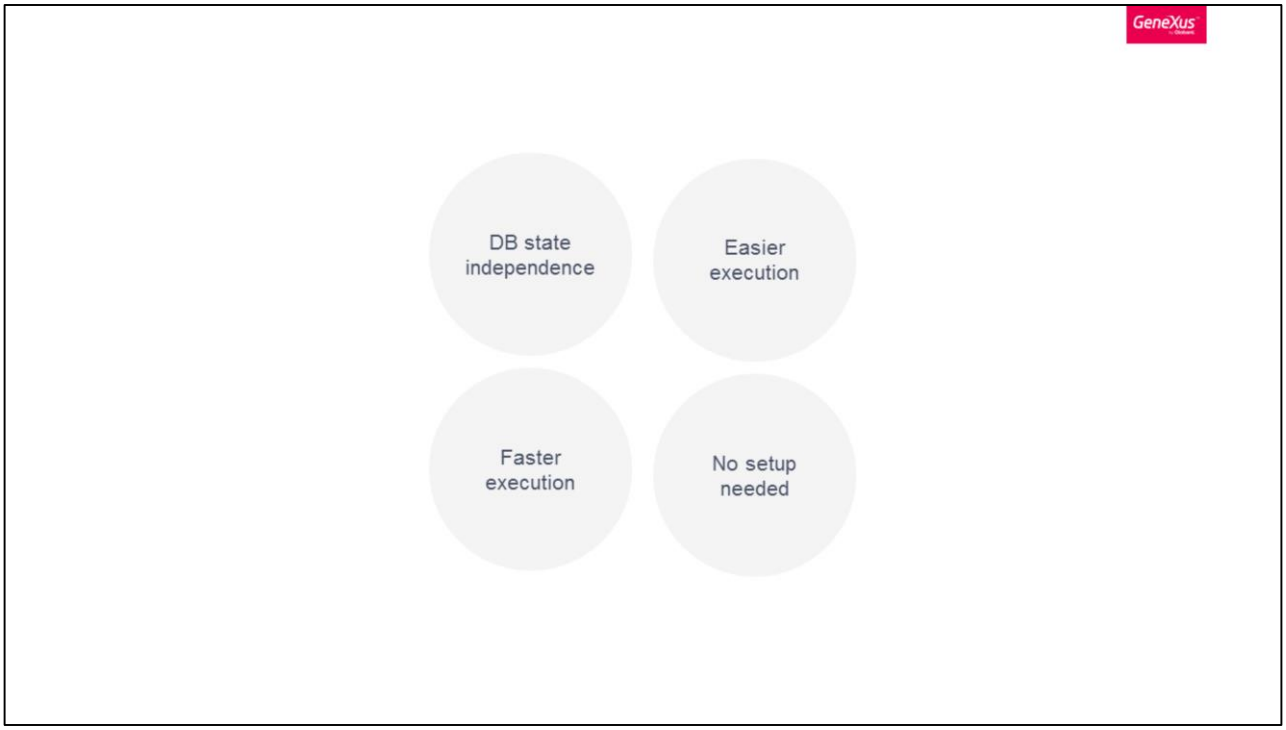

GeneXus™
by Globant


データベースモッキング
GeneXus

データベースモッキングの有用性

GeneXus by Globant



データベースモッキングは、データベースの状態に関係なく、ユニットテストに必要なデータセットをモックファイルとして保存して、そのデータセットをテストの実行ごとに使用することができます。

この場合、テストに必要なデータはすべてモックファイルに保存されているため、事前にデータベースにデータ登録を行う必要ありません。
モックを使ったテストでは、実際のデータベースにアクセスすることなく、モックファイルから読み込まれるインメモリデータから直接データを取得するため、実行時間が最も早くなります。

データベースモックを利用しない場合、データが必要なテストでは、実行する前にデータベースへデータの登録を実行する必要があります。
この場合、テストを実行するために、より多くの工数がかかり、テストの実施は遅くなります。

GeneXusでのデータベースモッキング

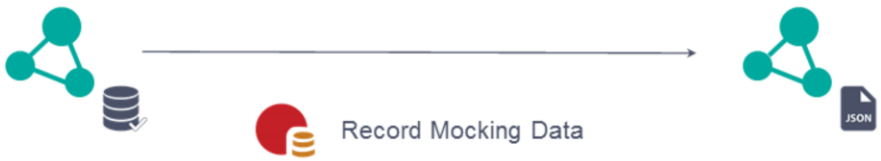
GeneXus by Globant

データベースモッキングは、Unit Testオブジェクトで利用可能な機能で、今後のユニットテスト実行のために特定のデータセットを準備できます。
このデータセットは、テストの中で対象となるテーブルのデータベース状態をキャプチャし、作成されます。

そのため、データベースモッキングの最初のステップとしては、ユニットテストが適切に実行されるために必要なすべてのデータを適切に登録することです。
つまり、データが存在しないこと、あるいは、口座に残高があることを確認するテストが必要ならば、
このテストを実行するために、データベースはすべての必要な情報を持っていなければなりません。

もし、データベースモッキング機能がなければ、テストを実行する前に毎回必要な情報がそろっているか確認を行う必要があります。

この機能を使えば、ユニットテストのデータセットを一度だけ実際のデータベースで準備するだけで、その後は異なるユニットテストのフェーズでそのデータを使うことができます。
つまり、データベースモッキングとは、レコード数の少ないデータベースのシミュレーションといえます。



モッキングは、開いたUnit Testオブジェクトのタブを右クリックし、"モックデータの記録"オプションをクリックし、全てのSQL文と結果を記録し、テストに使用したデータ（SQL文/結果）を保存することで機能します。
GeneXusでは、データベースのSQL文は、モッキングデータとしてjsonファイルに保存されます。

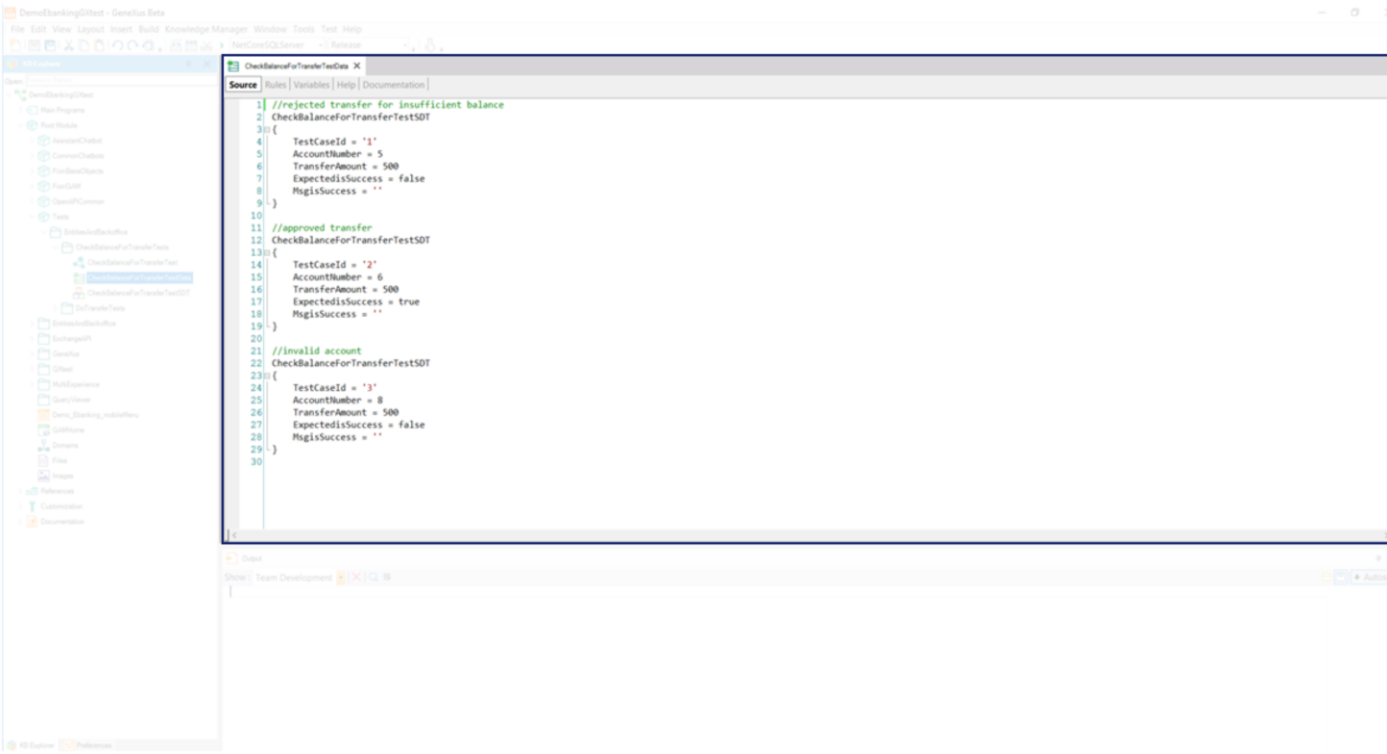
各ユニットテストで使用されたデータセットを保存し、チーム全員で共有することができます。

"モックデータの記録"オプションが終了すると、モッキングデータはKB内のFileオブジェクトとして格納されます。
このファイルへの参照は、Unit TestオブジェクトのMock Dataプロパティがtrueの時に表示される Fileというプロパティを通じて追加されます。

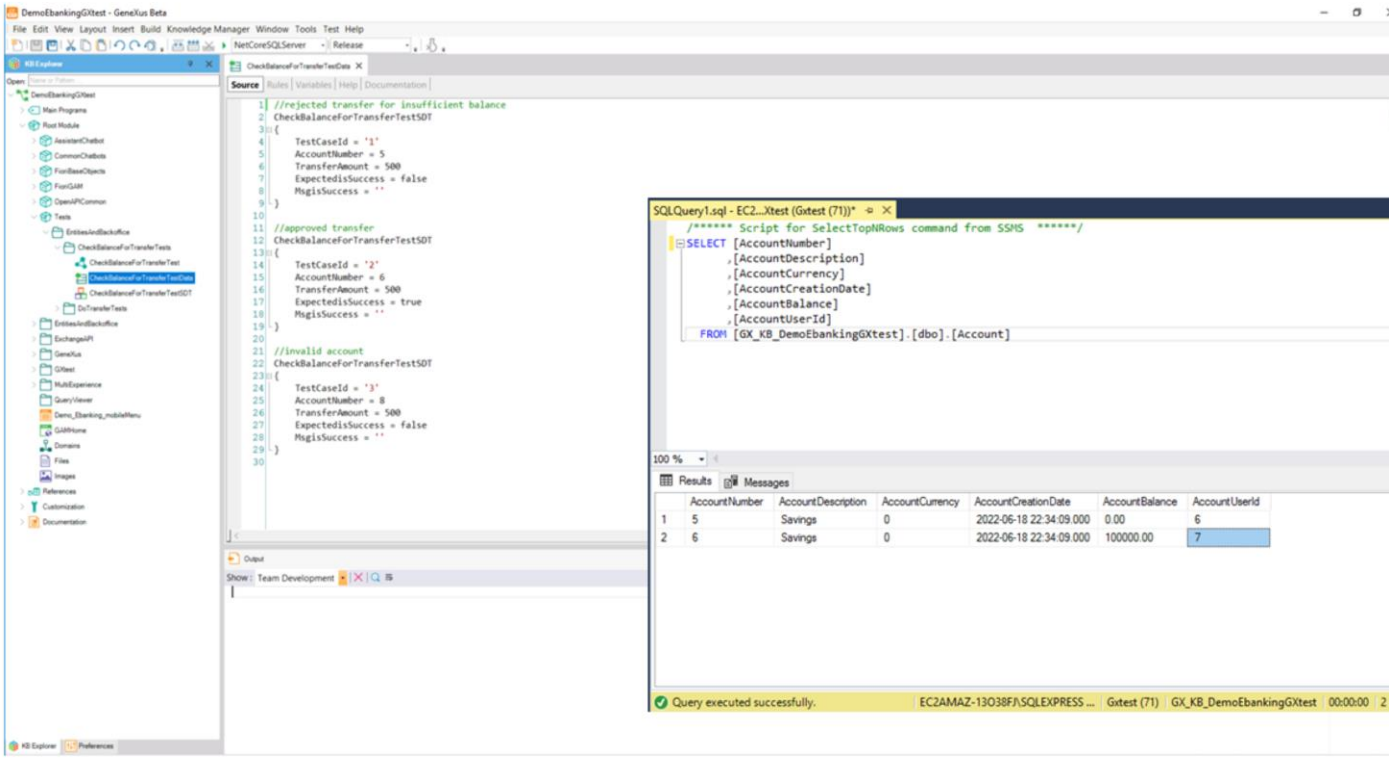
また、Mock DataプロパティはTrueに設定されています。
このプロパティを利用することにより、モックファイルへの参照を削除することなく、モックデータの利用を有効／無効にすることができます。

データベースモッキングのサンプル

GeneXus by Globant



CheckBalanceForTransferUnitTestのデータベースモックファイルを生成する方法を説明します。
前述の章で、このプロシージャでは、残高不足による振込拒否、振込承認、無効な口座の三つのテストケースでテストを行いました。

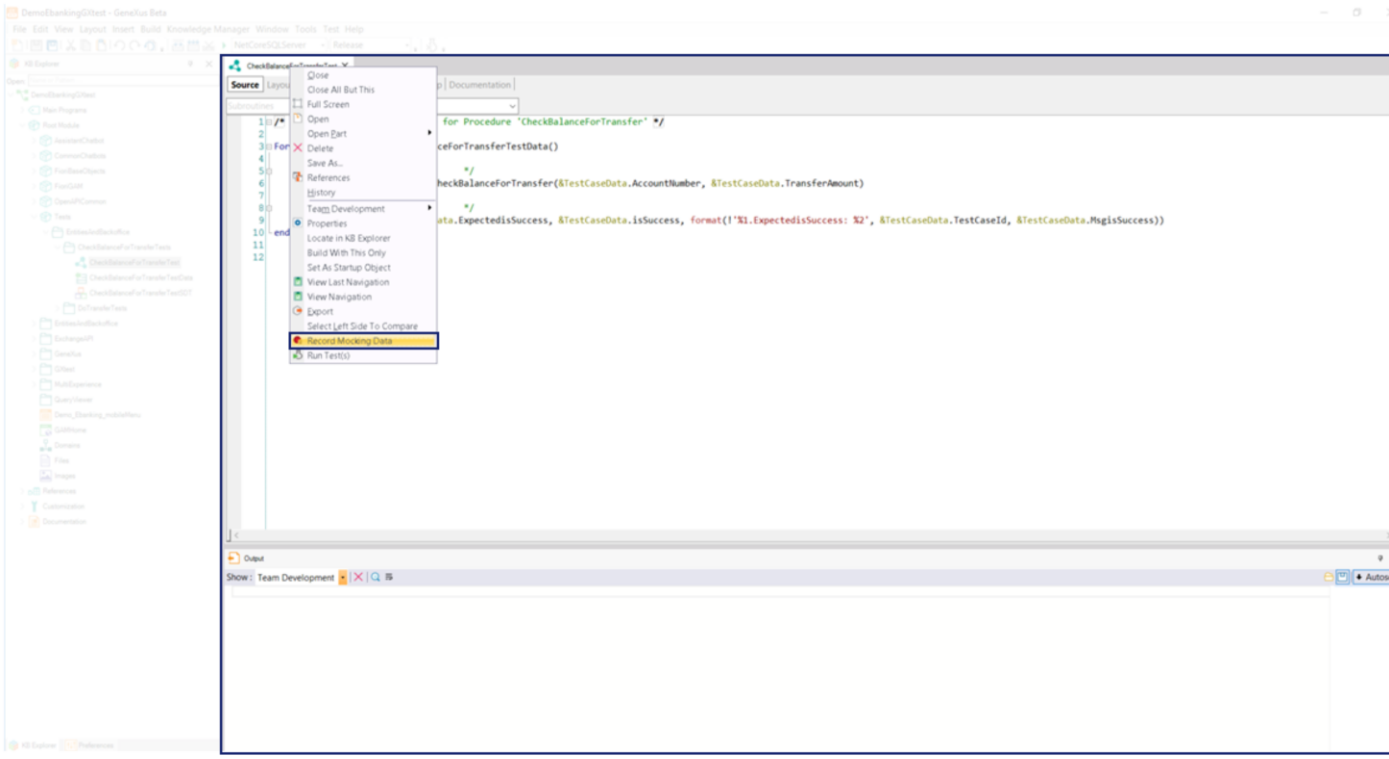


そこで、これらのテストケースを使用するために、データベースに必要なデータを準備する必要があります。

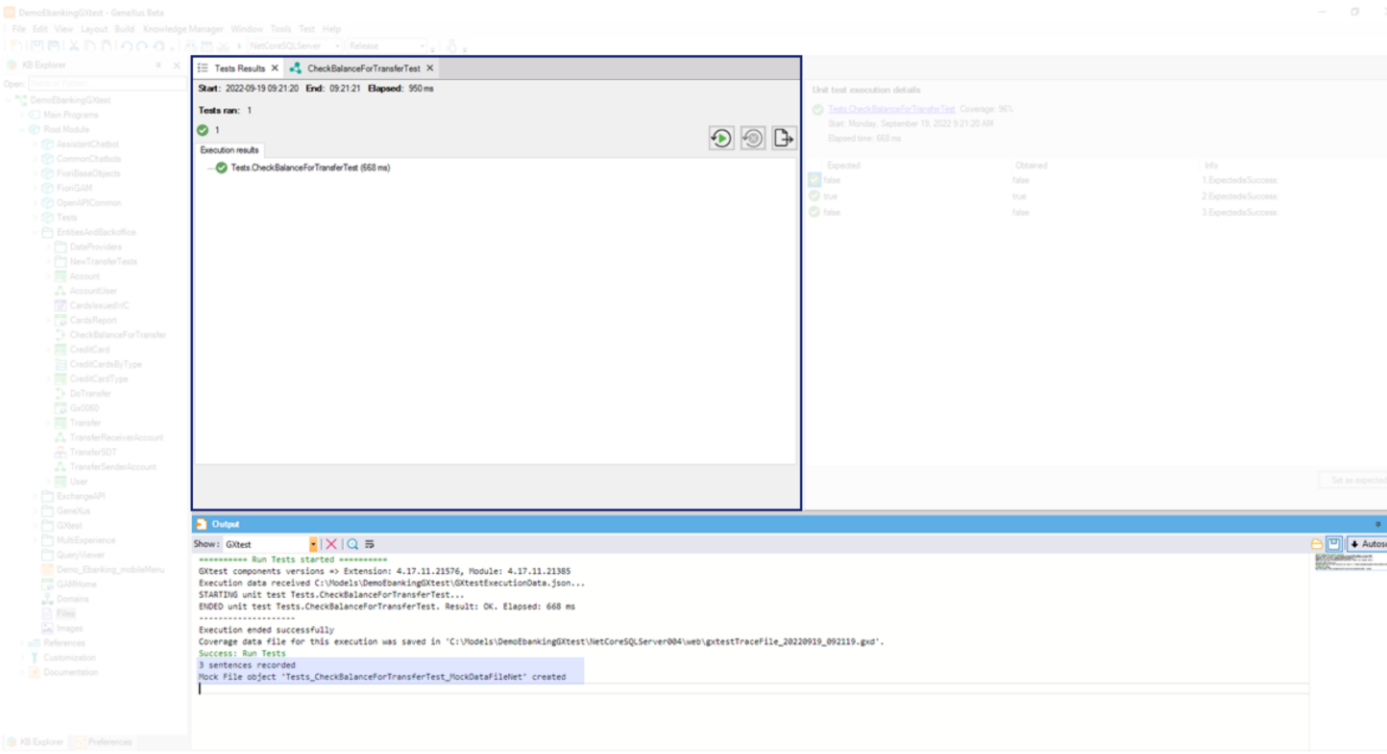
この場合、データとして次の三つの条件を満たすデータが必要です。

1. 残高0のAccountNumber5
2. 500ドルを送金するのに十分な残高のあるAccountNumber6
3. AccountNumber8が現状のデータベースに存在しない

現在のデータベースにおけるAccountテーブルのデータはスライドに表示された通りです。

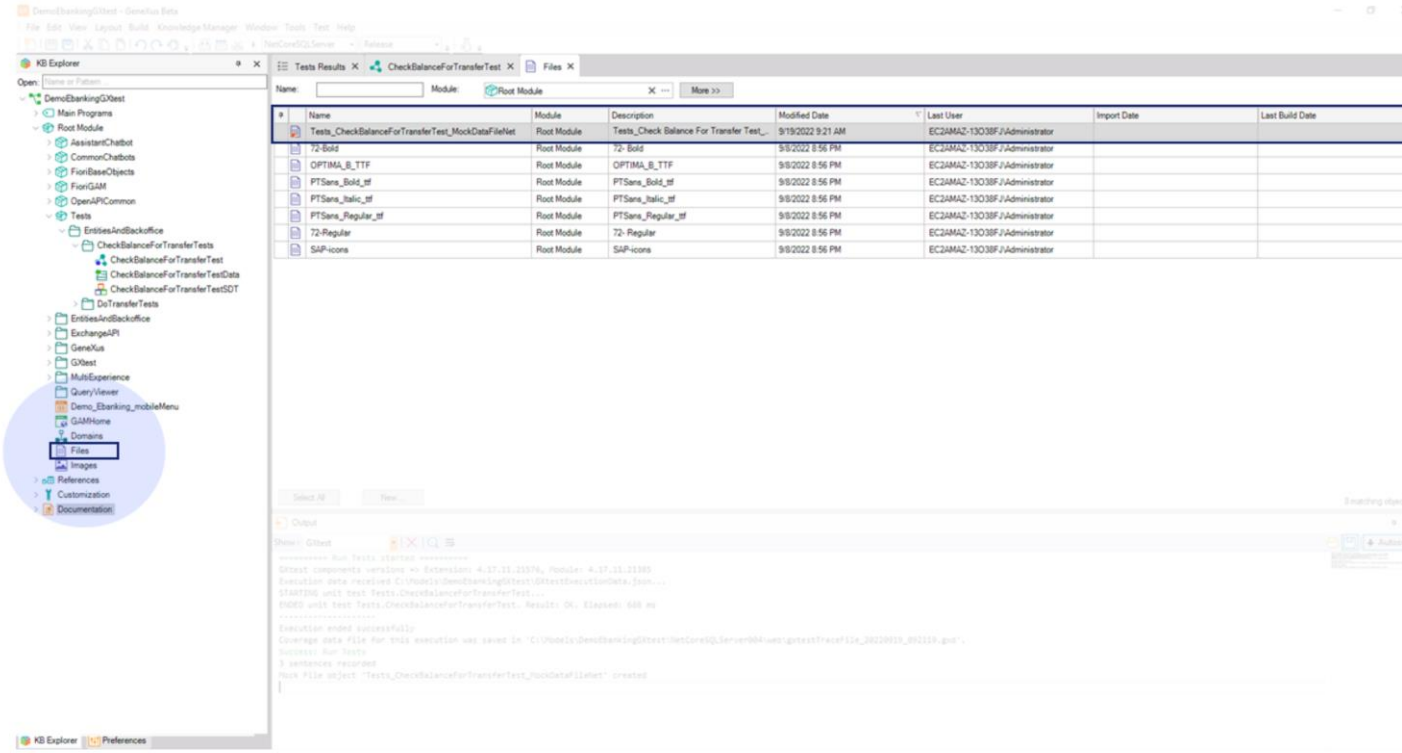


Unit Testオブジェクトのタブを右クリックし、"モックデータの記録"オプションをクリックします。

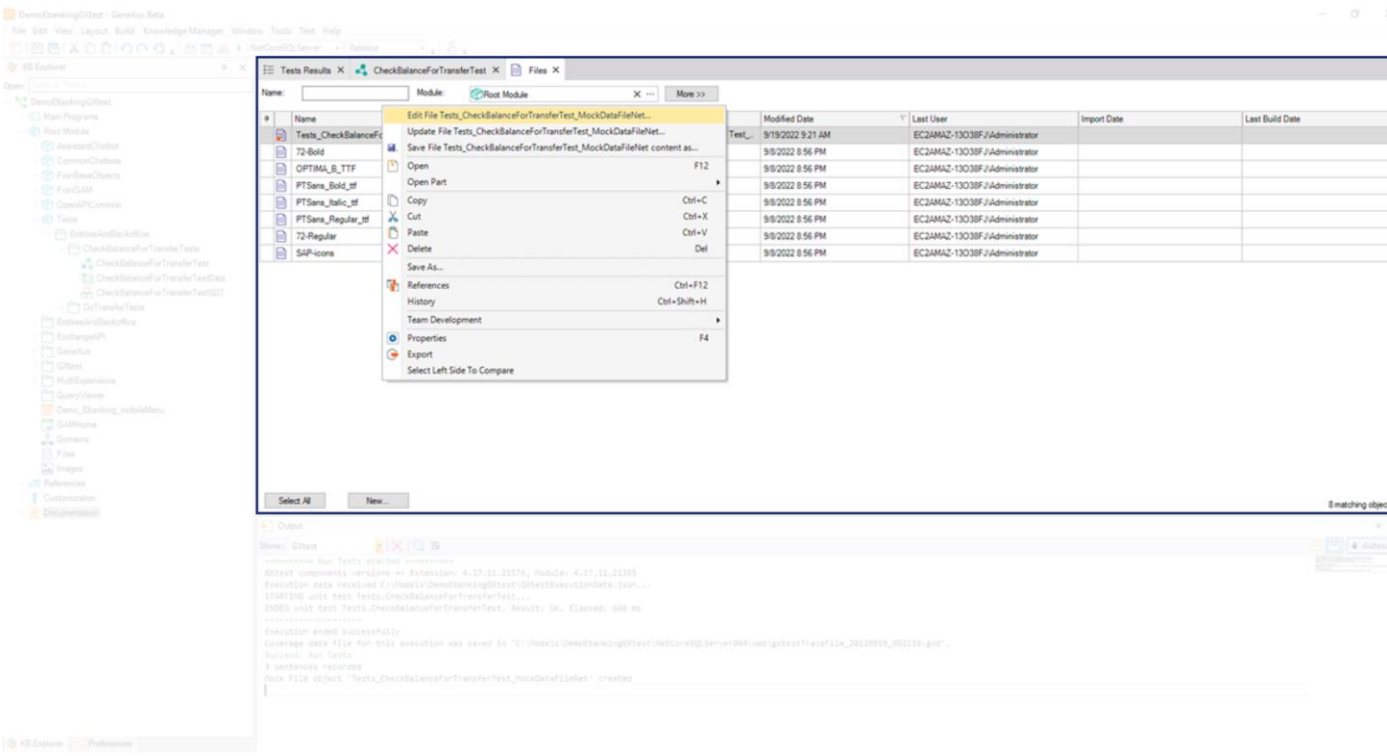


この操作によりテストが実行され、データベースに対するSQL文/結果がKB内のFileオブジェクトとして保存されます。
GeneXusの出力で表示コンボボックスをGXtestに変更することで、SQL文/結果とモックファイル名を確認することができます。

今後、ユニットテストを実行すると、データベースの代わりに記録されたデータを使って実行されます。
つまり、データベースが変更されても、モックファイルにはユニットテストの実行に必要なデータがすべて含まれているので、ユニットテストの結果には影響がありません。



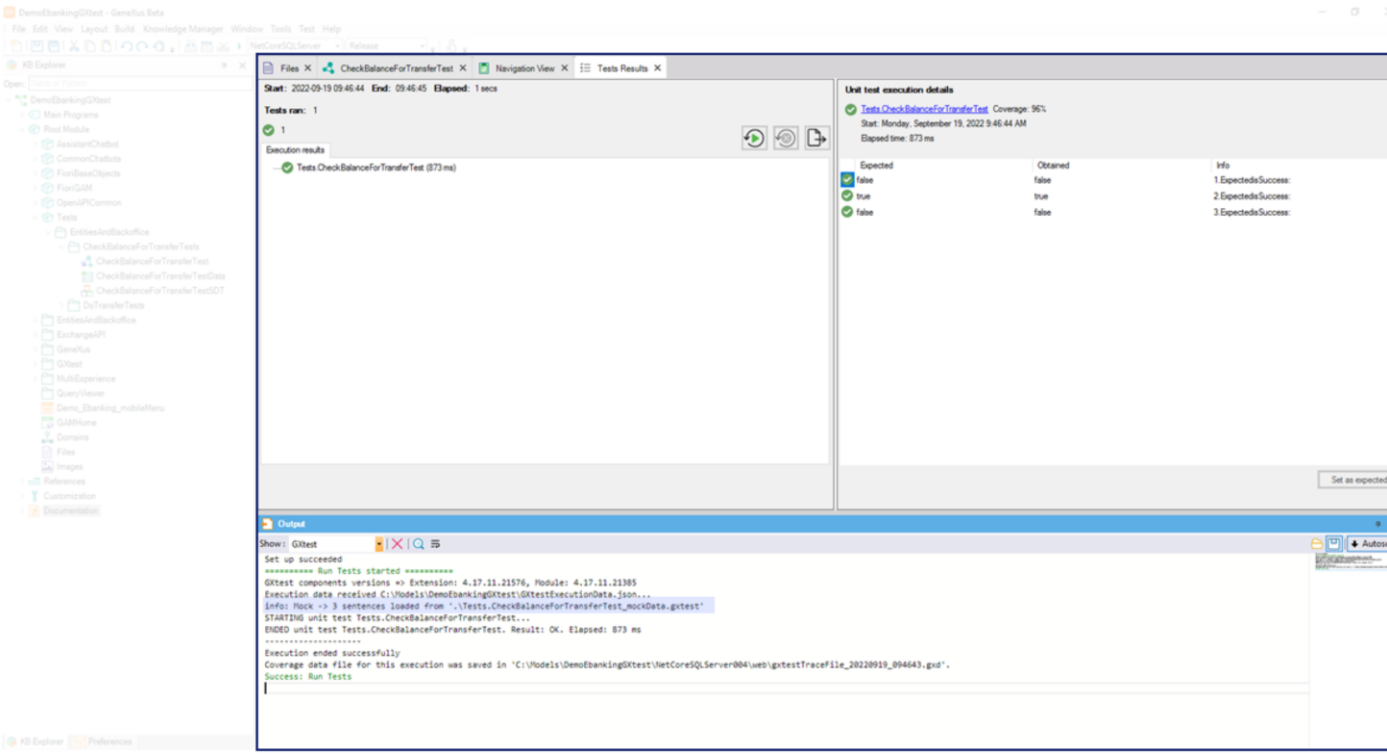
作成されたファイルは、KBエクスプローラーに表示される「ファイル」をダブルクリックし、表示される画面で確認できます。



右クリックして、“File Tests_CheckBalanceForTransferTest_MockDataNet… を編集”オプションをクリックすると、
GeneXusを起動している端末にインストールされているテキストエディタでファイルを開くことができ、保存されたSQL文とその結果を確認することができます。

このファイルを開くと、SQL文と結果がjson構造で格納されています。
記録中、SQL文は順番に保存されますので、テストを再実行した場合では記録された順番と同じ順番でSQL文が実行されることが期待されます。
＊この順番は非常に重要になります。

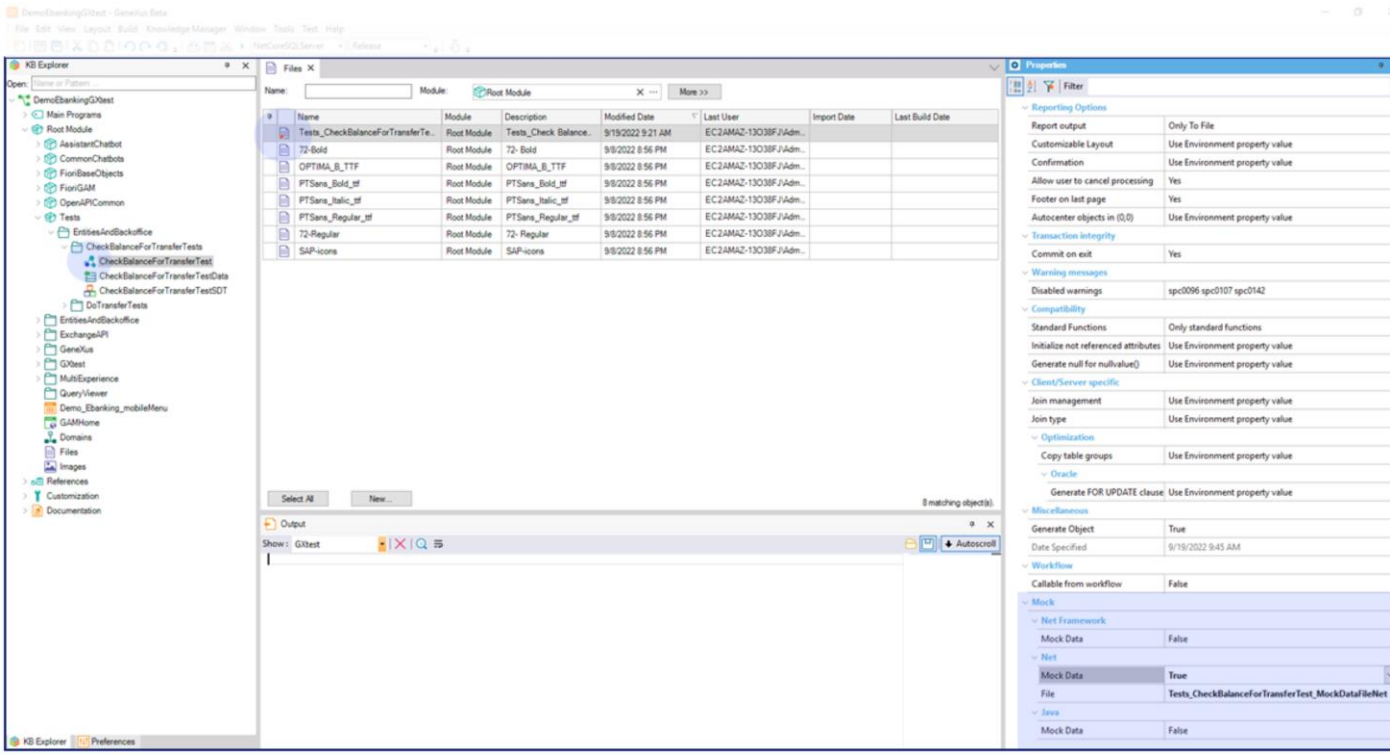
もし、開発が進むにつれてテーブルの参照が変更された場合や、データベース構造に変更が行われた場合は、モックデータを再度記録する必要があります。



モッキングで記録した後にユニットテストを実行すると、モックファイルから読み込まれた文章数を示すメッセージが出力に表示されます。

モッキングのための記録時に記録されなかった項目をテストが必要とする場合、そのクエリは結果を得るために実際のデータベース/データソースを使用します。
つまり、実際のデータソース/データベースを使用して選択したテストを実行します。

前述の通りモックファイルはGeneXusServerにコミットして共有できるため、チームで活用できるように、テスト対象として必要な状態のデータベースに対する記録であることを確認してください。

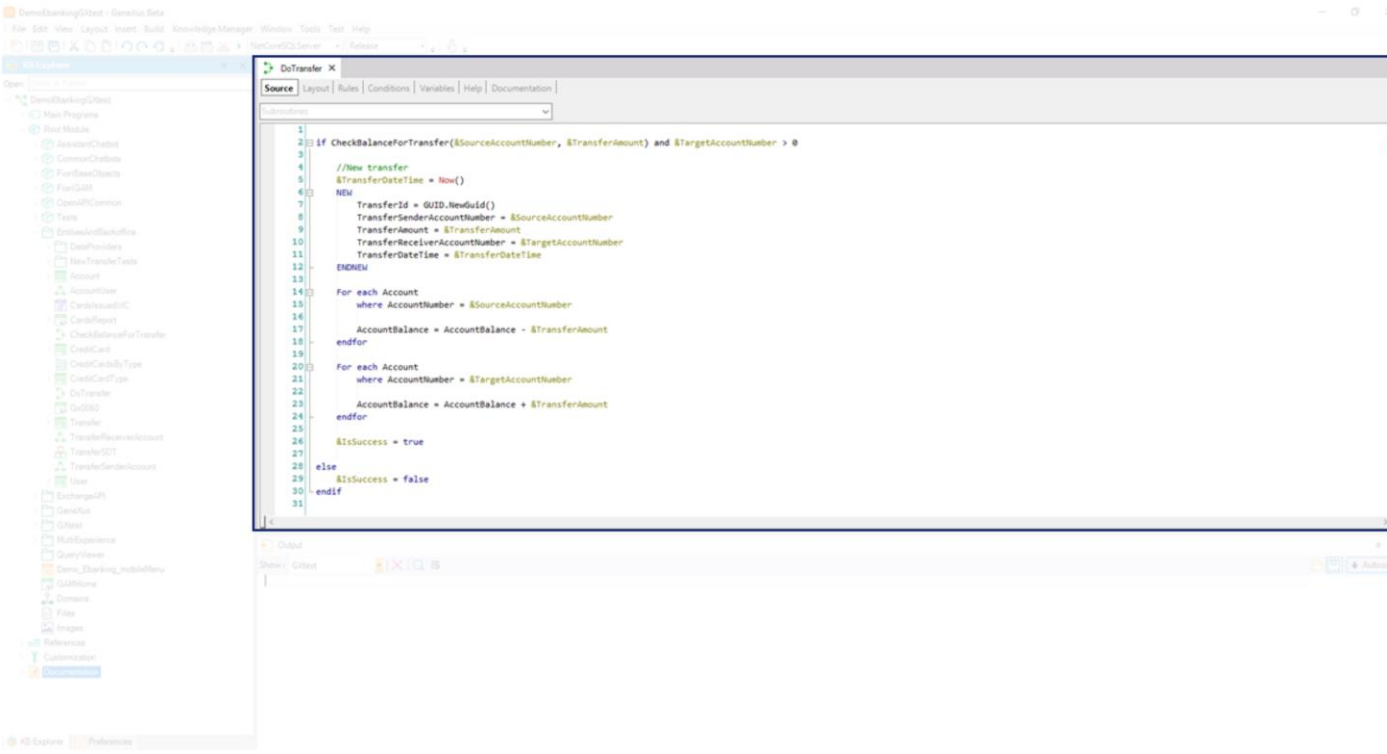


Unit Testオブジェクトのプロパティが更新され、「Mock Data」プロパティがTrueになっていることに注目してください。
これで、ユニットテストとモックファイルをコミットして、他の環境でもそのファイルを使って実行することができ、テストが必要なプロシージャに対し、テストを実行するために、毎回データベースを設定する必要がなくなります。

動的データの管理

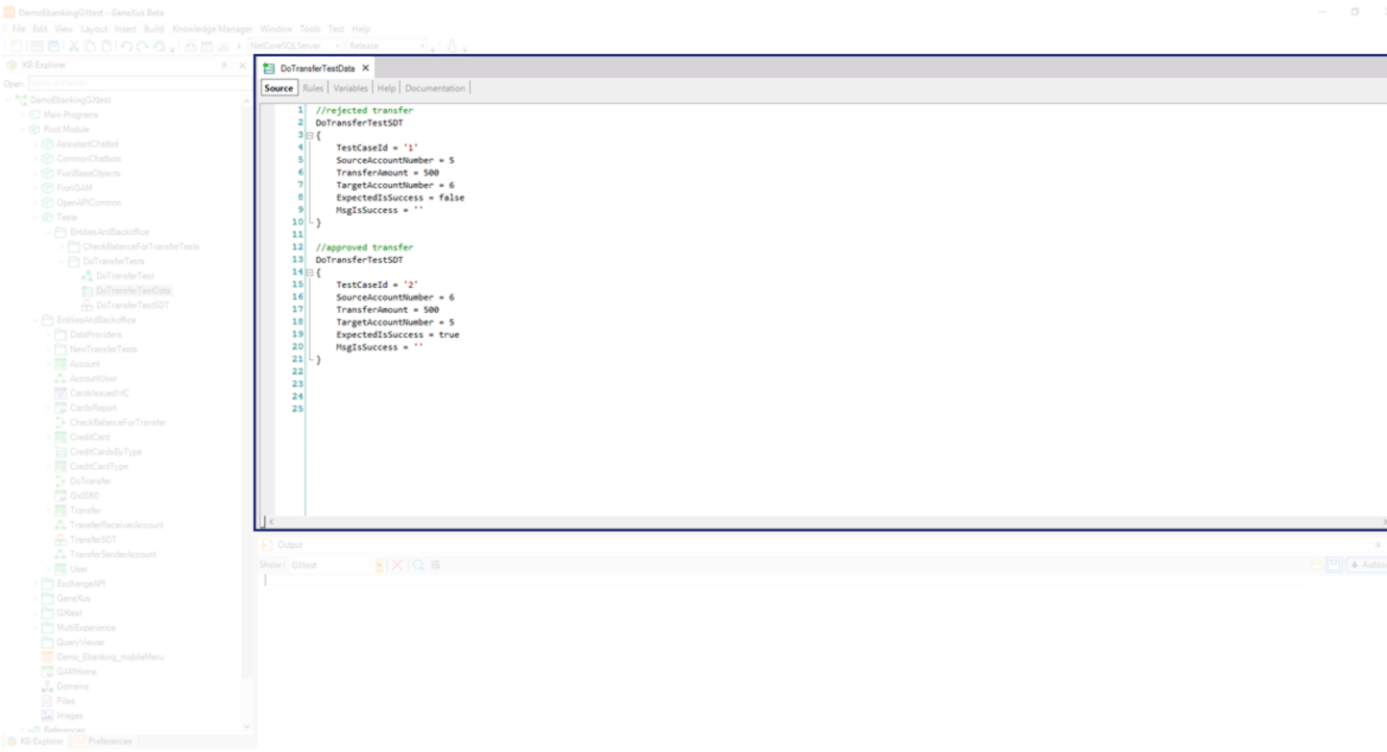
GeneXus by Globant

記録されたモックファイルの動的なデータをどのように管理するか見てみましょう。

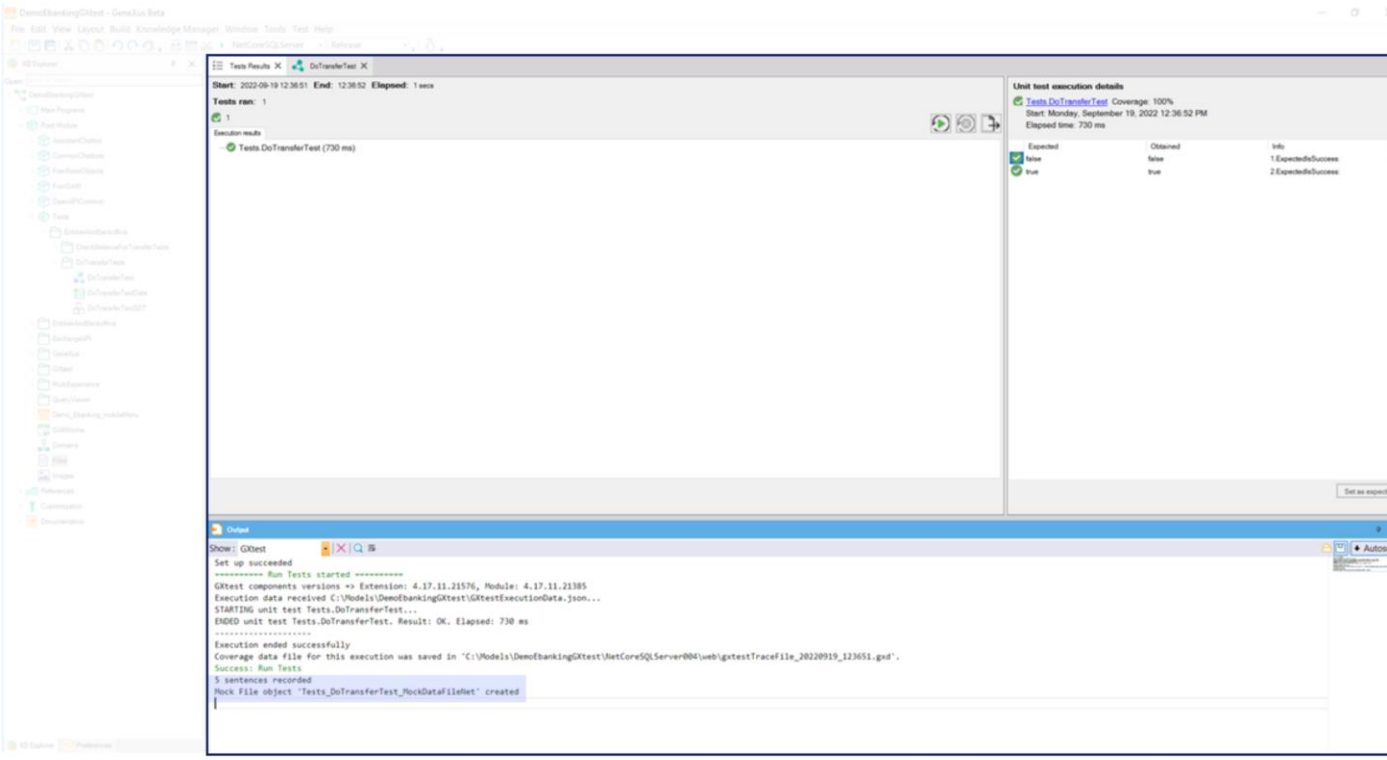


データベースモッキングの概念がわかったところで、動的なデータで記録する例を見てみましょう。
ここではDoTransferTestユニットテストをモックします。

これは新しい転送をTransferテーブルに保存し、TrasferDateTimeとTransferIdを動的データとして保存します。
まず、承認されたTransfarと拒否されたTransfarをデータベースに用意します。
モッキングファイルを生成するためにUnit Testオブジェクトのタブを右クリックし、"モックデータの記録"オプションをクリックします。

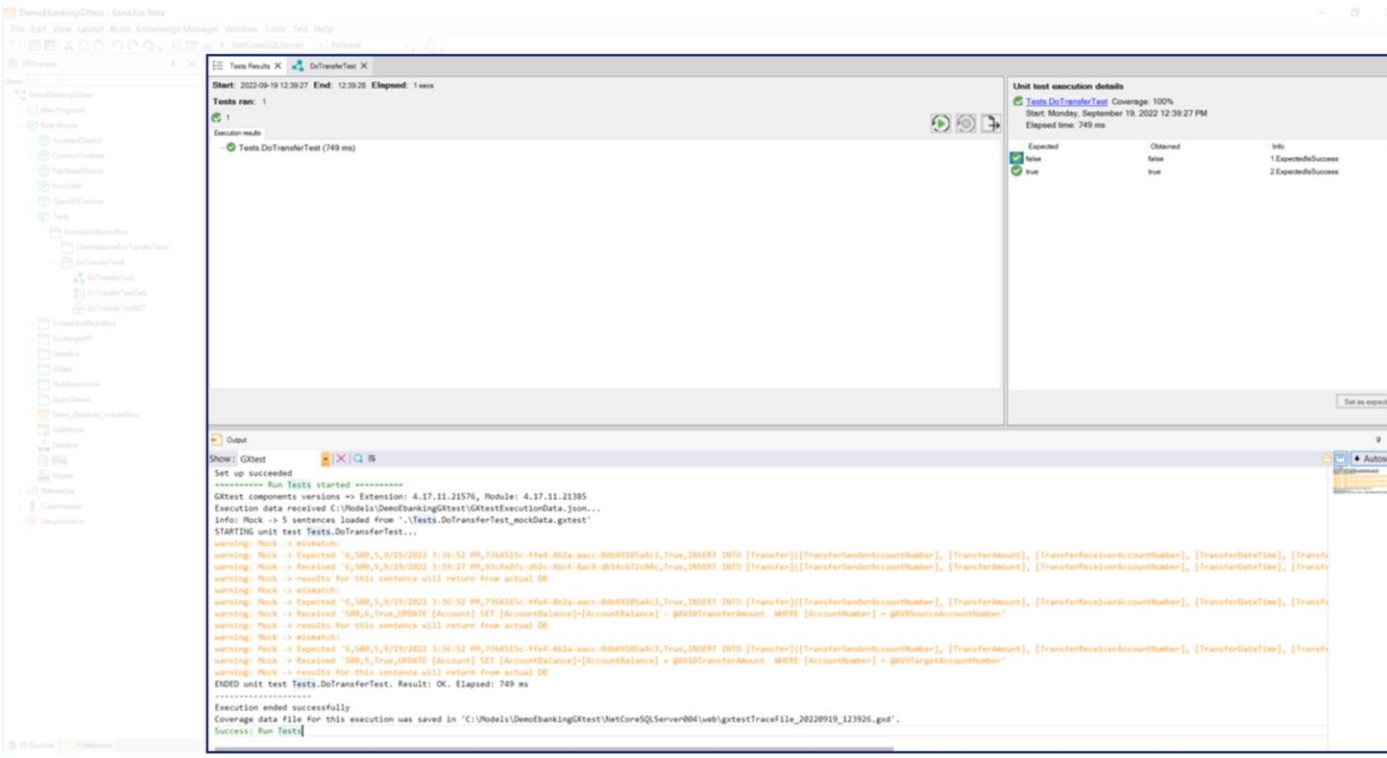


テストケースには、承認されるTransfarと拒否されるTransfarが含まれます。



"モックデータの記録"オプションをクリックすると、ご覧のように、5件のSQL文/結果が記録されています。

では、記録したばかりのモックファイルをそのまま使って、ユニットテストを実行してみましょう。

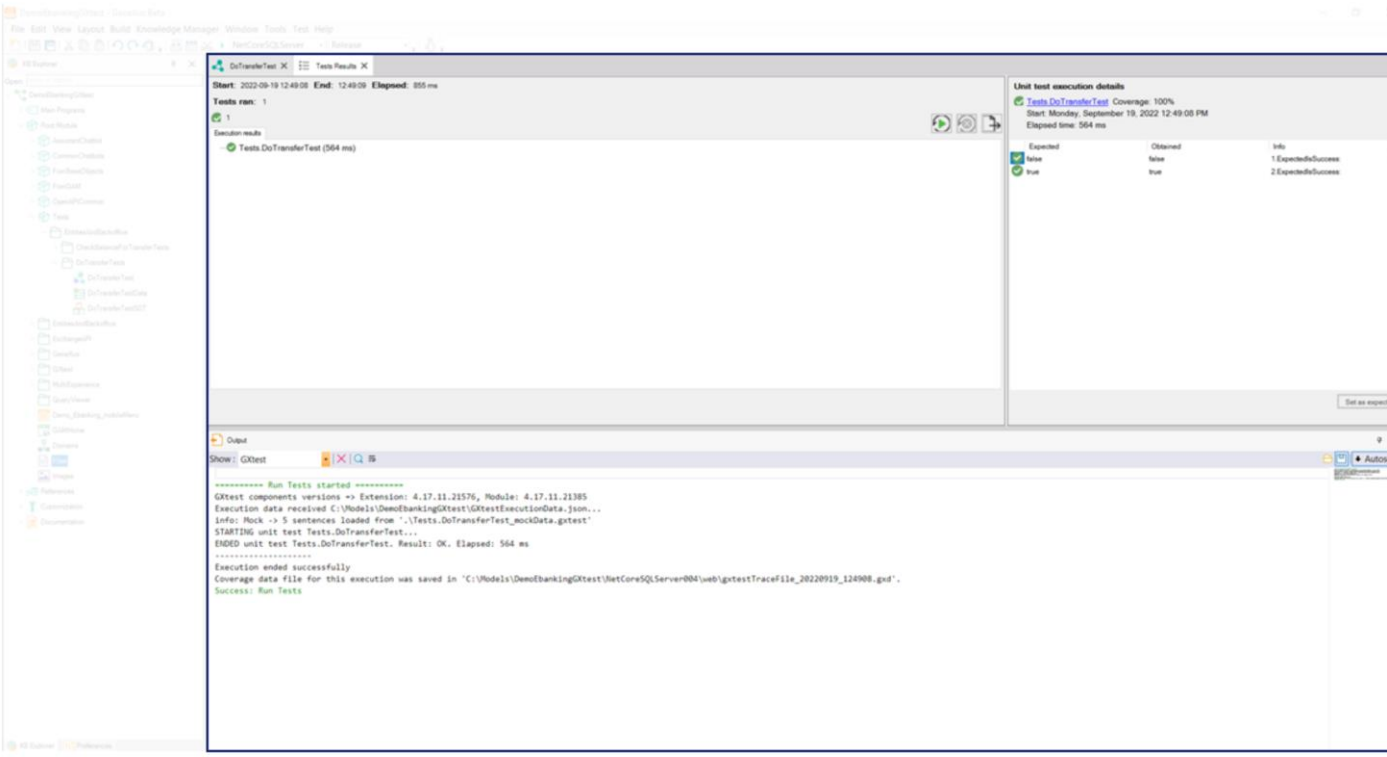


ここでOutputに警告が表示されるのは、モックファイルに記録されたデータとテスト実行中に要求されたデータとの間にミスマッチが発生したためです。
動的なデータはデータベース操作に関わるため、実行中に変化し、モックファイルに保存されているデータと一致しません。
これは、SQL文/結果に動的なデータ、例えば、自動的に設定される主キーや日付時刻がある場合に起こります。

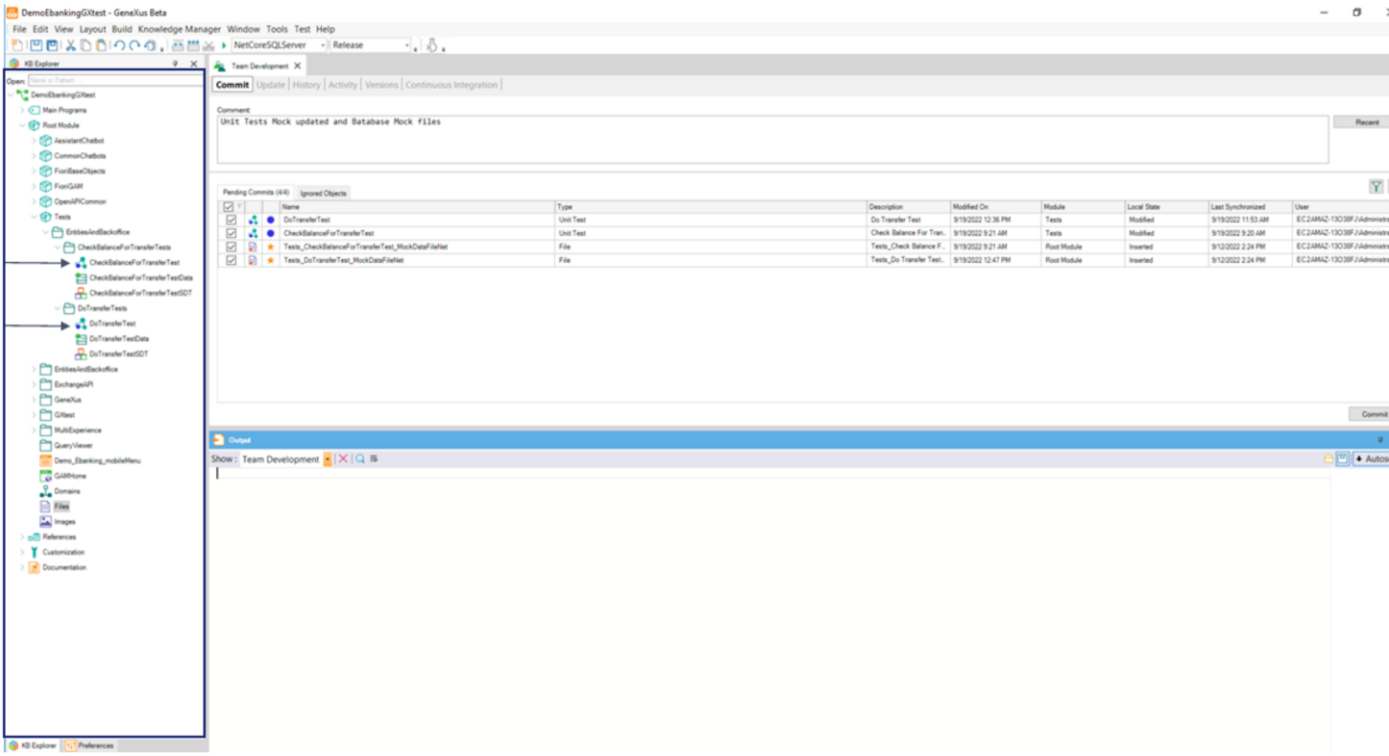
このような場合、警告にあるように、SQL文に対する結果が実際のデータベースから返されます。
この結果をモックファイルから取り出すには、モックファイルの編集時にKeyPatternというフィールドを設定して、マッチングのタイプを変更する必要があります。

今回の例では、Transferテーブルの動的なDateTime属性とのミスマッチが発生しています。
そこで、正規表現を追加してモックファイルを修正しましょう。

モックファイルのKeyPatternフィールドには、テスト実行時にミスマッチが起きないように正規表現を埋め込むことが可能です。
KeyPatternフィールドの値にKeyフィールドの値をコピー＆ペーストし、記録されたDateTimeとGUIDを期待する値にマッチする正規表現に置き換えることをお勧めします。



もう一度テストを実行すると、モックファイルから読み込んだ文章が正常に実行されます。



繰り返しとなりますが、モックデータを用意した場合、Mockプロパティをtrueにしたユニットテストとモックファイルを GeneXusServerにコミットすれば、
他の開発者が使用するデータベースの状態にテストが依存することはありません。

そして、さらに重要なこととしては、その結果が継続的インテグレーションパイプラインで使用される異なる環境にも依存しないことです。
ローカル環境でテストが成功すれば、他のどの環境でも (オブジェクトのバージョンが同じであれば) 同じように動作するということです。

データベースのアサーションを行わないテストでは、データベースのモックを使用することに注意しましょう。
実際のデータベースへの影響をチェックするようなテストでは、 データベースモッキングは使用したくありませんよね？
そのようなシナリオでは、代わりにデータベース初期化スクリプトを用意することが望ましいでしょう。


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