Dependency-Graph.Test

Details

Version: 2.2.0.0
Publisher: ANJ

App to test Dependency-Graph

Objects

Unknown

- DepGraphTest_ANJ (ID 99990)

  permissionset DepGraphTest_ANJ (ID 99990).

Codeunit

- DependencyGraphFacade_ANJ (ID 80810)

  Codeunit "DependencyGraphFacade_ANJ" (ID 80810).

- ExpectecValues_ANJ (ID 99994)
- Library Assert (ID 130002)

  This module provides functions for easy verification of expected values and error handling in test code.

- NumberSequenceTest_ANJ (ID 99990)

  Codeunit NumberSequenceTest_ANJ (ID 99990).

- GenerateFiguresTest_ANJ (ID 99991)

  Codeunit "GenerateFiguresTest_ANJ" (ID 99991).

- FillingProTablesMock_ANJ (ID 99993)

  Codeunit "FillingProTablesMock_ANJ" (ID 99993).

- TemporaryTablesTest_ANJ (ID 99992)

  Codeunit "TemporaryTablesTest_ANJ" (ID 99992).

- ExpectedValues_ANJ (ID 99994)

  Codeunit ExpectedValues_ANJ (ID 99994).

Table

- DependencyGraphSetup_ANJ (ID 80800)

  Table DependencyGraphSetup_ANJ (ID 80800).

- Extensions_ANJ (ID 80801)

  Table "Extensions_ANJ" (ID 80801).

- Relations_ANJ (ID 80802)

Table Relations_ANJ (ID 80802).

Enum

- GeometricFigure_ANJ (ID 80800)
  Enum GeometricFigure_ANJ (ID 80800) implements Interface FigureInGraph_ANJ.

EnumExtension

- FillingProcessingTablesMock_ANJ (ID 99990)

  EnumExtension FillingProcessingTablesMock_ANJ (ID 99990) extends Record FillingProcessingTables_ANJ.

- FillingProcessTablesMock_ANJ (ID 99990)

  EnumExtension FillingProcessingTablesMock_ANJ (ID 99990) extends Record FillingProcessingTables_ANJ.

Interface

- FillingProcessingTables_ANJ
  Interface FillingProcessingTables_ANJ.

Dependencies

Library Assert by Microsoft
Version: 22.0.0.0
Tests-TestLibraries by Microsoft
Version: 22.0.0.0
Any by Microsoft
Version: 22.0.0.0
Dependency-Graph by ANJ
Version: 3.6.0.0

FillingProcessingTables_ANJ

Interface FillingProcessingTables_ANJ.

Properties

Property	Value
Object Type	Interface
Object Subtype	Normal
Accessibility Level	Public

Procedures

`GetExtensions()`

GetExtensions.

Syntax

```
[Text] := GetExtensions()
```

Return

Text

Return value of type Text.

`GetRelations()`

GetRelations.

Syntax

```
[Text] := GetRelations()
```

Return

JsonText: Text

Return value of type Text.

DependencyGraphSetup_ANJ

Table DependencyGraphSetup_ANJ (ID 80800).

Properties

Property	Value
Object Type	Table
Object Subtype	Normal
Object ID	80800
Accessibility Level	Public

Procedures

`SetMarkdown()`

SetMarkdown.

Syntax

```
SetMarkdown(AuxText: Text, FieldNo: Integer)
```

Parameters

AuxText
  Type: Text

Text.

FieldNo
  Type: Integer

Integer.

`GetInstance()`

GetInstance.

Syntax

```
GetInstance()
```

DependencyGraphFacade_ANJ

Codeunit "DependencyGraphFacade_ANJ" (ID 80810).

Properties

Property	Value
Object Type	Codeunit
Object Subtype	Normal
Object ID	80810
Accessibility Level	Public

Procedures

`InitializeNumberSequence()`

InitializeNumberSequence

Syntax

```
InitializeNumberSequence()
```

`GetNextNumberSequence()`

GetNextNumberSequence.

Syntax

```
[Text] := GetNextNumberSequence()
```

Return

Text

Return value of type Text.

`GenerateFigures()`

GenerateFigures.

Syntax

```
[Text] := GenerateFigures(ExtensionScope: Enum ExtensionScope_ANJ, Identity: Text, AppName: Text)
```

Parameters

ExtensionScope
  Type: Enum ExtensionScope_ANJ

Enum ExtensionScope_ANJ.

Identity
  Type: Text

Text.

AppName
  Type: Text

Text.

Return

Text

Return value of type Text.

`GenerateAllTables()`

GenerateAllTables.

Syntax

```
GenerateAllTables(HideDialog: Boolean)
```

Parameters

HideDialog
  Type: Boolean

Boolean.

`GenerateExtensionsTable()`

GenerateExtensionsTable.

Syntax

```
GenerateExtensionsTable()
```

`GenerateRelationTable()`

GenerateRelationsTable.

Syntax

```
GenerateRelationTable()
```

`CleanExtensionsTable()`

CleanExtensionsTable.

Syntax

```
CleanExtensionsTable()
```

`CleanRelationsTable()`

CleanRelationsTable.

Syntax

```
CleanRelationsTable()
```

`GenerateGraph()`

GenerateGraph.

Syntax

```
GenerateGraph()
```

`GetMarkdownText()`

GetMarkdownText.

Syntax

```
[Text] := GetMarkdownText(FieldNo: Integer)
```

Parameters

FieldNo
  Type: Integer

Integer.

Return

Text

Return value of type Text.

`GetInterfaceFillProcessingTables()`

GetInterfaceFillProcessingTables.

Syntax

```
GetInterfaceFillProcessingTables(var FillingProcessingTables: Interface
FillingProcessingTables_ANJ)
```

Parameters

FillingProcessingTables
  Type: Interface FillingProcessingTables_ANJ

VAR Interface FillingProcessingTables_ANJ.

`OnAfterGetFillingProcessingTables()`

Syntax

```
OnAfterGetFillingProcessingTables(var FillingProcessingTables: Interface
FillingProcessingTables_ANJ)
```

Parameters

FillingProcessingTables
  Type: Interface FillingProcessingTables_ANJ

ExpectecValues_ANJ

Properties

Property	Value
Object Type	Codeunit
Object Subtype	Normal
Object ID	99994
Accessibility Level	Internal

Extensions_ANJ

Table "Extensions_ANJ" (ID 80801).

Properties

Property	Value
Object Type	Table
Object Subtype	Normal
Object ID	80801
Accessibility Level	Public

Procedures

`UpdateFigure()`

UpdateFigure

Syntax

```
UpdateFigure()
```

`UpdateRelationTable()`

UpdateRelationTable.

Syntax

```
UpdateRelationTable()
```

Relations_ANJ

Table Relations_ANJ (ID 80802).

Properties

Property	Value
Object Type	Table
Object Subtype	Normal
Object ID	80802
Accessibility Level	Internal

GeometricFigure_ANJ

Enum GeometricFigure_ANJ (ID 80800) implements Interface FigureInGraph_ANJ.

Properties

Property	Value
Object Type	Enum
Object Subtype	Normal
Implementing	FigureInGraph_ANJ
Object ID	80800
Accessibility Level	Public

Library Assert

This module provides functions for easy verification of expected values and error handling in test code.

Properties

Property	Value
Object Type	Codeunit
Object Subtype	Normal
Object ID	130002
Accessibility Level	Public

Procedures

`IsTrue()`

Tests whether the specified condition is true and throws an exception if the condition is false.

Syntax

```
IsTrue(Condition: Boolean, Msg: Text)
```

Parameters

Condition
  Type: Boolean

The condition the test expects to be true.

Msg
  Type: Text

The message to include in the exception when condition is false. The message is shown in test results.

`IsFalse()`

Tests whether the specified condition is false and throws an exception if the condition is true.

Syntax

```
IsFalse(Condition: Boolean, Msg: Text)
```

Parameters

Condition
  Type: Boolean

The condition the test expects to be false.

Msg
Type: Text

The message to include in the exception when condition is true. The message is shown in test results.

`AreEqual()`

Tests whether the specified values are equal and throws an exception if the two values are not equal.

Syntax

```
AreEqual(ExpectedVariant: Variant, ActualVariant: Variant, Msg: Text)
```

Parameters

ExpectedVariant
Type: Variant

The first value to compare. This is the value the tests expects.

ActualVariant
Type: Variant

The second value to compare. This is the value produced by the code under test.

Msg
Type: Text

The message to include in the exception when actual is not equal to expected. The message is shown in test results.

`AreEqual()`

Tests whether the specified dictionaries are equal and throws an exception if the two dictionaries are not equal.

Syntax

```
AreEqual(Expected: Dictionary of [Text, Text], Actual: Dictionary of [Text, Text])
```

Parameters

Expected
Type: Dictionary of [Text, Text]

The first dicitonary to compare.

Actual
Type: Dictionary of [Text, Text]

The second dictionary to compare.

`AreEqualDateTime()`

Tests whether the specified DateTime values are equal and throws an exception if the two DateTime values are not equal. This function uses the high precision format type 1

Syntax

```
AreEqualDateTime(Expected: DateTime, Actual: DateTime, Msg: Text)
```

Parameters

Expected
  Type: DateTime

The first DateTime value to compare. This is the DateTime value the tests expects.

Actual
  Type: DateTime

The second DateTime value to compare. This is the DateTime value produced by the code under test.

Msg
  Type: Text

The message to include in the exception when actual is not equal to expected. The message is shown in test results.

`AreNotEqual()`

Tests whether the specified values are unequal and throws an exception if they are equal.

Syntax

```
AreNotEqual(ExpectedVariant: Variant, ActualVariant: Variant, Msg: Text)
```

Parameters

ExpectedVariant
  Type: Variant

The first value to compare. This is the value the test expects not to match actual.

ActualVariant
  Type: Variant

The second value to compare. This is the value produced by the code under test.

Msg
  Type: Text

The message to include in the exception when actual is not equal to expected. The message is shown in test results.

`AreNearlyEqual()`

Tests whether the specified decimals are equal and throws an exception if the they are not equal.

Syntax

```
AreNearlyEqual(Expected: Decimal, Actual: Decimal, Delta: Decimal, Msg: Text)
```

Parameters

Expected
  Type: Decimal

The first value to compare. This is the value the tests expects.

Actual
  Type: Decimal

The second value to compare. This is the value produced by the code under test.

Delta
  Type: Decimal

The required accuracy. An exception will be thrown only if actual is different than expected by more than delta.

Msg
  Type: Text

The message to include in the exception when actual is different than expected by more than delta. The message is shown in test results.

`AreNotNearlyEqual()`

Tests whether the specified decimals are unequal and throws an exception if the they are equal.

Syntax

```
AreNotNearlyEqual(Expected: Decimal, Actual: Decimal, Delta: Decimal, Msg: Text)
```

Parameters

Expected
  Type: Decimal

The first value to compare. This is the value the tests expects not to match actual.

Actual
  Type: Decimal

The second value to compare. This is the value produced by the code under test.

Delta
  Type: Decimal

The required accuracy. An exception will be thrown only if actual is different than Expected by at most delta.

Msg
  Type: Text

The message to include in the exception when actual is equal to Expected or different by less than delta. The message is shown in test results.

`Fail()`

Throws an exception.

Syntax

```
Fail(Msg: Text)
```

Parameters

Msg
Type: Text

The message to include in the exception. The message is shown in test results.

`RecordIsEmpty()`

Tests whether the specified record is non-empty and throws an exception if it is.

Syntax

```
RecordIsEmpty(RecVariant: Variant)
```

Parameters

RecVariant
Type: Variant

The record to be checked

`RecordIsNotEmpty()`

Tests whether the specified record is empty and throws an exception if it is.

Syntax

```
RecordIsNotEmpty(RecVariant: Variant)
```

Parameters

RecVariant
Type: Variant

The record to be checked

`TableIsEmpty()`

Tests whether the specified table is non-empty and throws an exception if it is.

Syntax

```
TableIsEmpty(TableNo: Integer)
```

Parameters

TableNo
Type: Integer

The id of table the test expects to be empty

`TableIsNotEmpty()`

Tests whether the specified table is empty and throws an exception if it is.

Syntax

```
TableIsNotEmpty(TableNo: Integer)
```

Parameters

TableNo
  Type: Integer

The id of table the test expects not to be empty

`RecRefIsEmpty()`

Syntax

```
RecRefIsEmpty(var RecordRef: RecordRef)
```

Parameters

RecordRef
  Type: RecordRef

`RecRefIsNotEmpty()`

Syntax

```
RecRefIsNotEmpty(var RecordRef: RecordRef)
```

Parameters

RecordRef
  Type: RecordRef

`RecordCount()`

Tests whether the Table holds the expected number of Records and throws an exception when the count is different.

Syntax

```
RecordCount(RecVariant: Variant, ExpectedCount: Integer)
```

Parameters

RecVariant
  Type: Variant

The table whos records will be counter

ExpectedCount
  Type: Integer

The expected number of records in the table

`KnownFailure()`

This function is used to indicate the test is known to fail with a certain error. If the last error thrown is the expected one, a known failure error is thrown. If the last error was a different error than an exception is thrown.

Syntax

```
KnownFailure(Expected: Text, WorkItemNo: Integer)
```

Parameters

Expected
  Type: Text

The expected error

WorkItemNo
  Type: Integer

The Id of the workitem to fix the know test defect

`ExpectedError()`

Verifies that the last error thrown is the expected error. If a different error was thrown, an exception is thrown.

Syntax

```
ExpectedError(Expected: Text)
```

Parameters

Expected
  Type: Text

The expected error

`ExpectedErrorCode()`

Verifies that the last error code thrown is the expected error code. If a different error code was thrown, an exception is thrown.

Syntax

```
ExpectedErrorCode(Expected: Text)
```

Parameters

Expected
  Type: Text

The expected error code

`ExpectedMessage()`

Tests that the Expected message matches the Actual message

Syntax

```
ExpectedMessage(Expected: Text, Actual: Text)
```

Parameters

Expected
  Type: Text

The first value to compare. This is the value the tests expects not to match actual.

Actual
  Type: Text

The second value to compare. This is the value produced by the code under test.

`AssertRecordNotFound()`

Verifies that the last error code thrown was the Record Not Found error code.

Syntax

```
AssertRecordNotFound()
```

`AssertRecordAlreadyExists()`

Verifies that the last error code thrown was the Record Already Exists error code.

Syntax

```
AssertRecordAlreadyExists()
```

`AssertNothingInsideFilter()`

Verifies that the last error code thrown was the Nothing Inside Filter error code.

Syntax

```
AssertNothingInsideFilter()
```

`AssertNoFilter()`

Verifies that the last error code thrown was the No Filter error code.

Syntax

```
AssertNoFilter()
```

`AssertPrimRecordNotFound()`

Verifies that the last error code thrown was the Primary Record Not Found error code.

Syntax

```
AssertPrimRecordNotFound()
```

`TypeOf()`

Syntax

```
[Integer] := TypeOf(ValueVariant: Variant)
```

Parameters

ValueVariant
  Type: Variant

`TypeNameOf()`

Syntax

```
[Text] := TypeNameOf(ValueVariant: Variant)
```

Parameters

ValueVariant
  Type: Variant

`UnsupportedTypeName()`

Syntax

```
[Text] := UnsupportedTypeName(ValueVariant: Variant)
```

Parameters

ValueVariant
  Type: Variant

`Equal()`

Syntax

```
[Boolean] := Equal(LeftVariant: Variant, RightVariant: Variant)
```

Parameters

LeftVariant
  Type: Variant

RightVariant
  Type: Variant

`IsNumber()`

Syntax

```
[Boolean] := IsNumber(ValueVariant: Variant)
```

Parameters

ValueVariant
  Type: Variant

`EqualNumbers()`

Syntax

```
[Boolean] := EqualNumbers(Left: Decimal, Right: Decimal)
```

Parameters

Left
  Type: Decimal

Right
  Type: Decimal

`VerifyFailure()`

Syntax

```
VerifyFailure(ErrorCodeExpected: Text, FailureText: Text)
```

Parameters

ErrorCodeExpected
  Type: Text

FailureText
  Type: Text

FillingProcessingTablesMock_ANJ

EnumExtension FillingProcessingTablesMock_ANJ (ID 99990) extends Record FillingProcessingTables_ANJ.

Properties

Property	Value
Object Type	EnumExtension
Object Subtype	Normal
Extending	FillingProcessingTables_ANJ
Object ID	99990
Accessibility Level	Public

FillingProcessTablesMock_ANJ

EnumExtension FillingProcessingTablesMock_ANJ (ID 99990) extends Record FillingProcessingTables_ANJ.

Properties

Property	Value
Object Type	EnumExtension
Object Subtype	Normal
Extending	FillingProcessingTables_ANJ
Object ID	99990
Accessibility Level	Public

NumberSequenceTest_ANJ

Codeunit NumberSequenceTest_ANJ (ID 99990).

Properties

Property	Value
Object Type	Codeunit
Object Subtype	Test
Object ID	99990
Accessibility Level	Public

Procedures

`TestNumberSequence()`

TestNumberSequence.

Syntax

```
TestNumberSequence()
```

`CheckInitializeAndCreateSomeNumberSeries()`

CheckInitializeAndCreateSomeNumberSeries.

Syntax

```
CheckInitializeAndCreateSomeNumberSeries(var FirstRequest: Text, var SecondRequest: Text, var
ThirdRequest: Text)
```

Parameters

FirstRequest
  Type: Text

VAR Text.

SecondRequest
  Type: Text

VAR Text.

ThirdRequest
  Type: Text

VAR Text.

GenerateFiguresTest_ANJ

Codeunit "GenerateFiguresTest_ANJ" (ID 99991).

Properties

Property	Value
Object Type	Codeunit
Object Subtype	Test
Object ID	99991
Accessibility Level	Public

Procedures

`GenerateFigures()`

GenerateFigures.

Syntax

```
GenerateFigures()
```

`InitializeDependencyGraphSetup()`

InitializeDependencyGraphSetup

Syntax

```
InitializeDependencyGraphSetup()
```

`GetFigureText()`

GetFigureText.

Syntax

```
GetFigureText(var ScopeDevFigure: Text, var ScopeGlobalFigure: Text, var ScopePTEFigure: Text)
```

Parameters

ScopeDevFigure
  Type: Text

VAR Text.

ScopeGlobalFigure
  Type: Text

VAR Text.

ScopePTEFigure

Type: Text

VAR Text.

FillingProTablesMock_ANJ

Codeunit "FillingProTablesMock_ANJ" (ID 99993).

Properties

Property	Value
Object Type	Codeunit
Object Subtype	Normal
Implementing	FillingProcessingTables_ANJ
Object ID	99993
Accessibility Level	Internal

Procedures

`GetExtensions()`

GetExtensions.

Syntax

```
[Text] := GetExtensions()
```

Return

Text

Return value of type Text.

`GetRelations()`

GetRelations.

Syntax

```
[Text] := GetRelations()
```

Return

Text

Return value of type Text.

`AddNewExtensionToJsonArry()`

AddNewExtensionToJsonArry.

Syntax

```
AddNewExtensionToJsonArry(var ExtensionArry: JsonArray, PackageId: Text, DisplayName: Text,
Publisher: Text, PublishedAs: Text)
```

Parameters

ExtensionArry
Type: JsonArray

VAR JsonArray.

PackageId
Type: Text

Text.

DisplayName
Type: Text

Text.

Publisher
Type: Text

Text.

PublishedAs
Type: Text

Text.

`AddNewRelationToJsonArry()`

AddNewRelationToJsonArry. ///

Syntax

```
AddNewRelationToJsonArry(var RelationsArry: JsonArray, SourceAppID: Guid, DestinationAppID: Guid)
```

Parameters

RelationsArry
Type: JsonArray

SourceAppID
Type: Guid

DestinationAppID
Type: Guid

TemporaryTablesTest_ANJ

Codeunit "TemporaryTablesTest_ANJ" (ID 99992).

Properties

Property	Value
Object Type	Codeunit
Object Subtype	Test
Object ID	99992
Accessibility Level	Public

Procedures

`GenerateFigures()`

Syntax

```
GenerateFigures()
```

`InitializeDependencyGraphSetup()`

InitializeDependencyGraphSetup

Syntax

```
InitializeDependencyGraphSetup()
```

`GetExtensionRecords()`

GetExtensionRecords.

Syntax

```
[Integer] := GetExtensionRecords()
```

Return

Integer

Return value of type Integer.

`GetRelationsRecords()`

GetRelationsRecords.

Syntax

```
[Integer] := GetRelationsRecords()
```

Return

Integer

Return value of type Integer.

`GetMarkdownTexts()`

GetMarkdownTexts.

Syntax

```
GetMarkdownTexts(var MarkdownText: Text, var MarkdownMermaidText: Text)
```

Parameters

MarkdownText
  Type: Text

VAR Text.

MarkdownMermaidText
  Type: Text

VAR Text.

ExpectedValues_ANJ

Codeunit ExpectedValues_ANJ (ID 99994).

Properties

Property	Value
Object Type	Codeunit
Object Subtype	Normal
Object ID	99994
Accessibility Level	Internal

Procedures

`GetExpectedMarkdownText()`

GetExpectedMarkdownText.

Syntax

```
[Text] := GetExpectedMarkdownText()
```

Return

Text

Return value of type Text.

`GetExpectedMarkdownMermaidText()`

GetExpectedMarkdownMermaidText.

Syntax

```
[Text] := GetExpectedMarkdownMermaidText()
```

Return

Text

Return value of type Text.

DepGraphTest_ANJ

permissionset DepGraphTest_ANJ (ID 99990).

Properties

Property	Value
Object Type	Unknown
Object Subtype	Normal
Object ID	99990
Accessibility Level	Public