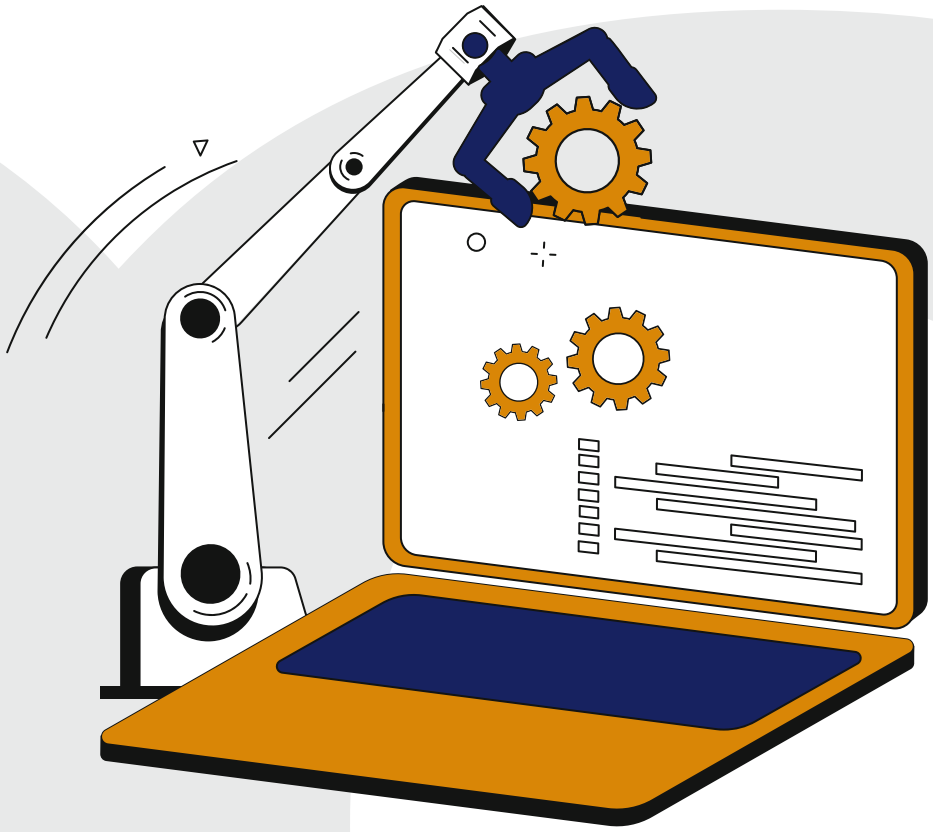
Quick Start for BECKHOFF



Automate Your Code

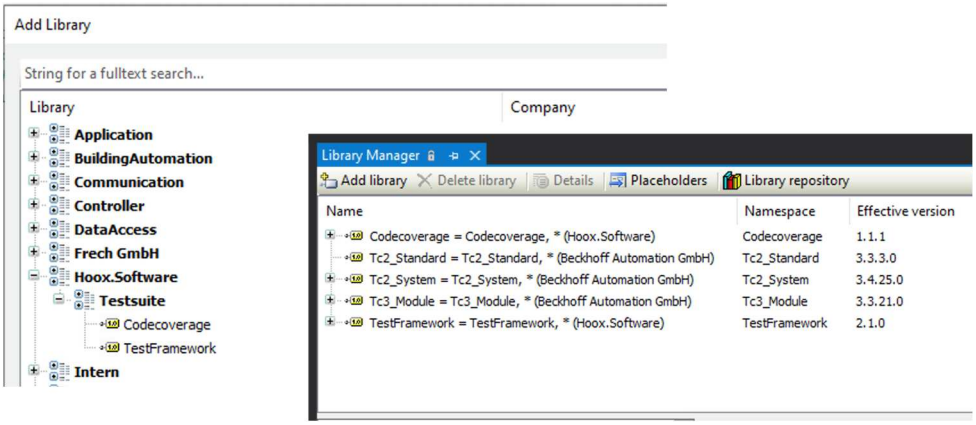
HOOX
[SOFTWARE]

HOOX Software
Patrick Dressel
Steinbühl 1
95233 Helmbrechts
M: +49 170 5260988
patrick@hoox.software

Testbibliothek

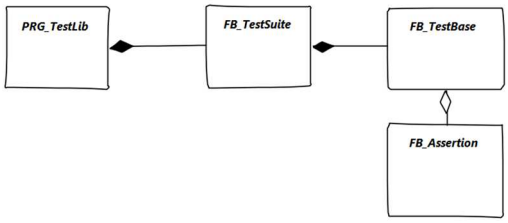
Bibliothek importieren

Die Verwaltung des Bibliotheksrepository öffnen und die neue Bibliothek hinzufügen. Danach in der Applikation die Bibliothek einbinden.



Allgemein

Die Bibliothek besteht im wesentlichen aus drei Teilen: PRG_Testlib, FB_Testsuite und FB_TestBase.



Baustein	Beschreibung
PRG_TestLib	Verwaltet alle konfigurierten TestSuites
FB_TestSuite	Verwaltet die zugeordneten Testfälle
FB_TestBase	Basisklasse für einen Testfall
FB_StandardTextReport	Beispiel für einen Export des Testlaufs
FB_ReportBase	Basisklasse für einen Report der nach dem Testlauf aufgerufen wird.

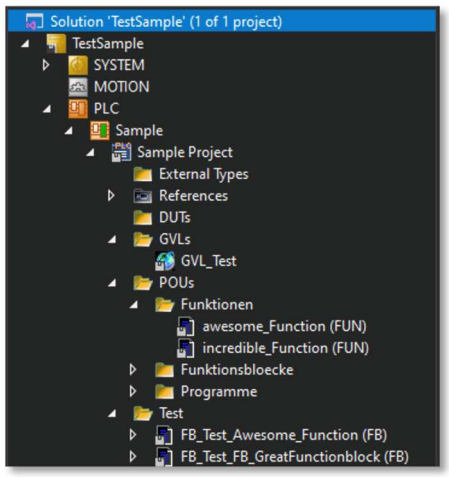
Der Baustein PRG_Testlib enthält standardmäßig den Teil zum exportieren der JUnit.xml Datei.

Eigene Reports können durch Veerbung von FB_ReportsBase erstellt werden.

Patrick Dressel
Steinbühl 1
95233 Helmbrechts
M: +49 170 5260988
patrick@hoox.software

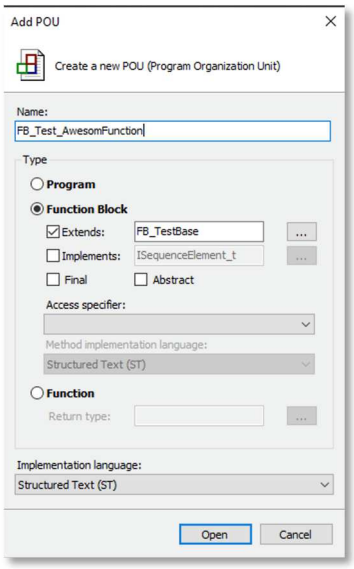
Beispiel

Es soll die Funktion *awesomefunction* getestet werden.

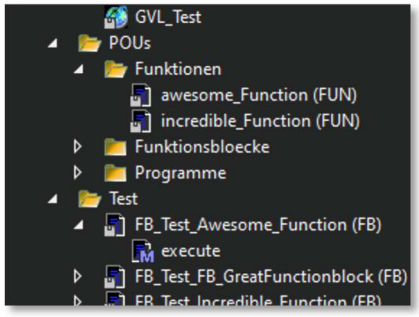


Vorgehensweise

1. Testbaustein anlegen
Dafür eine neue (Test)Klasse von der Klasse FB_TestBase ableiten.



2. Notwendige Methoden überschreiben



Um den Testbaustein zu verwenden muß die Methode `execute` *(TestAusführung)* überschrieben werden.

Im ersten Schritt reicht es aus `prepare` und `cleanup` jeweils `auszulassen`.
In diesen beiden Methoden kann Code eingetragen werden der vor (`prepare`) oder nach (`cleanup`) dem Testfall ausgeführt werden soll.

Patrick Dressel
Steinbühl 1
95233 Helmbrechts
M: +49 170 5260988
patrick@hoox.software

3. Test implementieren

Für die Einstufung des Testfalls ("passed" oder "failed") müssen sogenannte Asserts aufgerufen werden. Diese bestehen immer aus einem aktuellen und einem erwarteten Wert.

Ein Assert ist eine Behauptung das der aktuelle dem erwarteten Wert entpricht : `AssertTrue`, oder
nicht entpricht : `AssertFalse`.

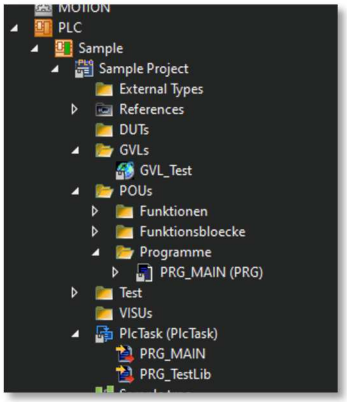
```
19  ********************************************************************
20  METHOD PUBLIC execute : ERetCode_t
21  VAR_INPUT
22  END_VAR

1
2       setMessage('Test 1/1 = 1');
3       s32Value    := awesome_Function(1, 1);
4       s32Expected := 2;
5       assert.EQUAL(s32Value,  s32Expected);
6
7       setMessage('Test 1/-1 = -1');
8       s32Value    := awesome_Function(1, -1);
9       s32Expected := -1;
10      assert.EQUAL(s32Value,  s32Expected);
11
12      setMessage('Test 1/0 = 0');
13      s32Value    := awesome_Function(1, 0);
14      s32Expected := 0;
15      assert.EQUAL_Fatal(s32Value, s32Expected);
16
17      execute := ERetCode_t.eRetDone;
```

Dieser Wert **muss** als Variable übergeben werden.
- Es kann jeder Datentyp verwendet werden,
- Es müssen für beide Parameter die gleichen Datentypen verwendet werden.

Auf das Beispiel bezogen:
Die Funktion erwartet einen DINT Wert und addiert oder subtrahiert diesen Wert vom aktuellen Wert der Instanz.

Zusätzlich empfiehlt es sich vor dem Aufruf über setMessage(...) einen Hinweis zu speichern falls der Assert fehlschlägt um einen Hinweis zu erhalten an welcher Stelle genau der Fehler aufgetreten ist.



PRG_Testlib aktivieren und einem Task der Runtime zuweisen.
Im Beispiel der "PlcTask"

4. Test ausführen

```
GVL_Test
1   {attribute 'qualified_only'}
2   VAR_GLOBAL
3
4       bStart  : BOOL;
5       bReady  : BOOL;
6
7       // Register Testsuite
8        // Testuite connects itself to PRG_TestLib
9       fb_TestSuite1 : FB_Testsuite( i_strId     := 'TestSuite1',
10                                    i_strPath   := 'Testframework\Testframework\UnitTest\');
11
12      // Register Test in Testsuite1
13       // Testcase connect itself to given Testsuite interface
14       // set Execute to false to skip the test.
15      fb_Test   : FB_Test_Awesome_Function(  i_strId          := 'TF1.1',
16                                             i_strClassName   := 'awesomefunction.TcPOU',
17                                             i_strPath        := 'Source/POUs',
18                                             i_iTestSuite     := fb_TestSuite1,
19                                             i_bExecute       := TRUE);
20
21  END_VAR
```

Watch 1

Expression	Appl...	Type	Value	Prepared value	Exec
PRG_TestLib	TestS...	PRG_TESTLIB			Cyclic
i_bStart		BOOL	FALSE	TRUE	Cyclic
o_eState		ESTATE_T	eStateIdle		Cyclic
o_eResult		ERESULT_T	eStateIdle		Cyclic
o_sStatistic		SStatistic_t			Cyclic
bHwStart		BOOL	FALSE		Cyclic
bHwBusy		BOOL	FALSE		Cyclic
bHwReady		BOOL	FALSE		Cyclic

Um die Testfälle ausführen zu können muss das Kommando **i_bStart** in PRG_Testlib beschrieben werden.
Der Testlauf ist beendet wenn **o_eState** den Wert **eStateReady** ausgibt.

Patrick Dressel
Steinbühl 1
95233 Helmbrechts
M: +49 170 5260988
patrick@hoox.software

Testbausteine Instanzieren

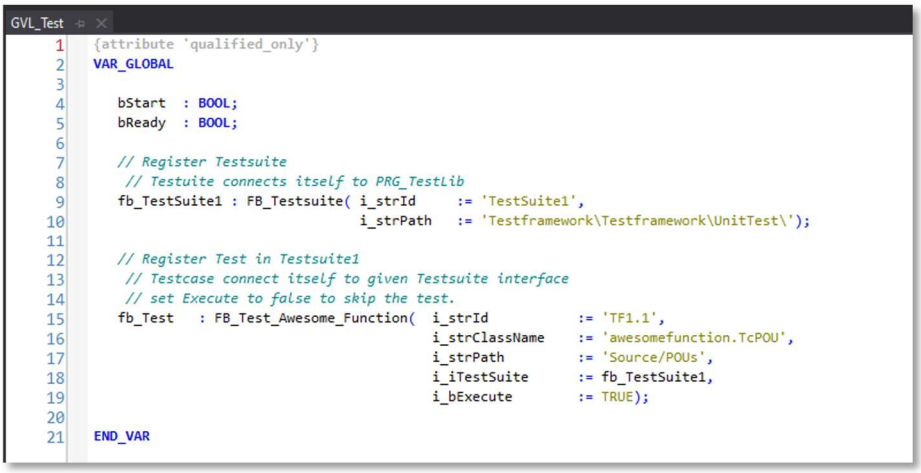
Die Testbausteine müssen innerhalb einer globalen Variablenliste instanziert werden. Für die Ausführung wird zur besseren Übersicht eine Testsuite benötigt die mehrere Tests verwaltet.
Für die spätere Auswertung müssen noch weitere Informationen der Instanz übergeben werden:

Testsuite

Parameter	Beschreibung
`i_strId`	Bezeichnung der Testsuite
`i_strPath`	Pfad zum Quellcode der enthaltenen Testfälle

Testfall

Parameter	Beschreibung
`i_strId`	eindeutige Id des Tests
`i_strClassName`	Klasse des Testobjekts
`i_strPath`	Pfad zum Quellcode der enthaltenen Testfälle
`i_iTestSuite`	Zugeordnete Testsuite
`i_bExecute`	Test ausführen oder nicht. Bei false wird dieser später als *SKIPPED* angezeigt.



Patrick Dressel
Steinbühl 1
95233 Helmbrechts
M: +49 170 5260988
patrick@hoox.software

HOOX

[SOFTWARE]

www.hoox.software