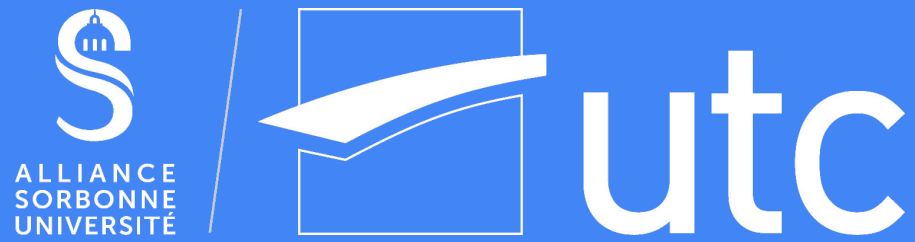


GTest

Présentation et introduction du logiciel de test **open source développé par Google**

Arman Soltani
Guillaume Nibert

AI03

Introduction

Pourquoi ce logiciel ?
Architecture xUnit

Introduction

Pourquoi ce logiciel de test ?
Architecture xUnit

Open Source

Léger

Performant

Architecture xUnit

- Setup
- Instructions
- Teardown

Test fixtures

Présentation de Google Test

Description
Popularité
Recommandation/philosophie de Google concernant les tests

Présentation de Google Test

Description - Popularité - Philosophie

Équipe “Testing Technology”

Multiplateforme

C/C++

Recommandations aux développeurs

Mises en gardes concernant les termes utilisés

Utilisation : Chromium, OpenCV, LLVM, création de guides

Nombre de commits au cours du temps

Installation & Exemple de tests unitaires

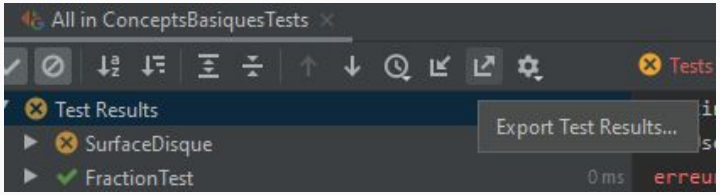
Démo live : Les bases de GTest.
Les codes de la démo sont disponible à l'adresse:
https://gitlab.utc.fr/asoltani/demo_ai03

Pour un exemple plus complet voir notre cas d'étude :
https://gitlab.utc.fr/asoltani/ai03_gtest

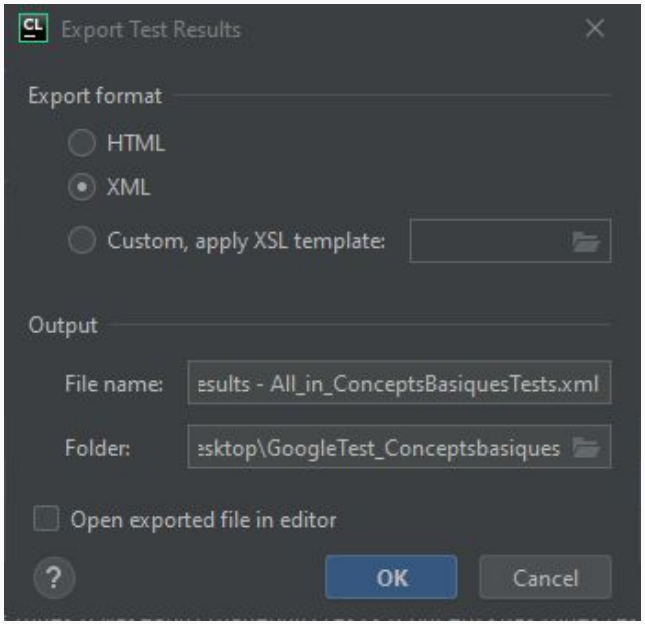
Génération d’un rapport XML

Intégration dans l’EDI CLion

1



2



3

```
<?xml version="1.0" encoding="UTF-8"?><testrun duration="2" footerText="Generated by CLion on 11/01/2020 13:37" name
    <count name="total" value="6"/>
    <count name="failed" value="1"/>
    <count name="passed" value="5"/>
    <config nameIsGenerated="true" PASS_PARENT_ENVS_2="true" PROJECT_NAME="ConceptsBasiques" TARGET_NAME="ConceptsBa
    <suite duration="2" locationUrl="gtest://suite:SurfaceDisque" name="SurfaceDisque" status="failed">
        <test duration="2" locationUrl="gtest://suite:SurfaceDisque/test:Zero" name="Zero" status="passed">
            <output type="stdout">La surface vaut 0m².
</output>
        </test>
        <test duration="0" locationUrl="gtest://suite:SurfaceDisque/test:Positif" name="Positif" status="failed">
            <output type="stdout">La surface vaut 3.14m².
La surface vaut 6.28m².
C:\Users\Guillaume\Desktop\GoogleTest_Conceptsbasiques\Tests\ConceptsBasiquesTests.cpp:14: Failure
Expected equality of these values:
  12.56
  surface(2)
```

Test fixture

Syntaxe

```
class TestFixture : public testing::Test { // cette classe dérive de testing::Test
    protected: // pour que les membres soient accessibles par d'éventuelles
               // sous-classes.
    void SetUp() override {
        // Appelé avant l'exécution de chaque test, c'est ici qu'on initialise le
        // variables utilisées par chaque test.
    }
    void TearDown() override {
        // Appelé après l'exécution de chaque test, c'est ici qu'on effectue le
        // nettoyage (ex : delete un pointeur si initialisé dans SetUp().
    }

    // Déclaration des variables que l'on souhaite utiliser dans nos tests de
    // TestSuite.
};
```

Test fixture

Syntaxe

```
TEST_F(nom_du_text_fixture, nom_du_test) {
    Assertions // Instructions qui vérifient si une condition est vraie.
}
```

GTest est un outil **simple** et **puissant**.

Merci de votre attention !

ANNEXES

Environnement de travail type

```
Root_Projet/
├── googletest/
├── Tests/
│   ├── CMakeLists.txt
│   └── test.cpp
├── CMakeLists.txt
└── Src
    ├── code.cpp
    └── code.hpp
```

ANNEXES

Listes non exhaustive des assertions les plus utilisées dans GTest

	Non fatale	Fatale	Description
Assertions basiques	`EXPECT_TRUE(condition);`	`ASSERT_TRUE(condition);`	Vérifie que condition est **true**
	`EXPECT_FALSE(condition);`	`ASSERT_FALSE(condition);`	Vérifie que condition est **false**
Comparaison binaire	`EXPECT_EQ(val1, val2);`	`ASSERT_EQ(val1, val2);`	Vérifie que **val1 == val2**
	`EXPECT_NE(val1, val2);`	`ASSERT_NE(val1, val2);`	Vérifie que **val1 != val2**
	`EXPECT_LT(val1, val2);`	`ASSERT_LT(val1, val2);`	Vérifie que **val1 < val2**
	`EXPECT_LE(val1, val2);`	`ASSERT_LE(val1, val2);`	Vérifie que **val1 <= val2**
	`EXPECT_GT(val1, val2);`	`ASSERT_GT(val1, val2);`	Vérifie que **val1 > val2**
	`EXPECT_GE(val1, val2);`	`ASSERT_GE(val1, val2);`	Vérifie que **val1 >= val2**
Comparaison de chaînes de caractères	`EXPECT_STREQ(ch1,ch2);`	`ASSERT_STREQ(ch1,ch2);`	Vérifie que les chaînes **ch1** et **ch2** sont identiques.
	`EXPECT_STRNE(ch1,ch2);`	`ASSERT_STRNE(ch1,ch2);`	Vérifie que les chaînes **ch1** et **ch2** ne sont pas identiques.
	`EXPECT_STRCASEEQ(ch1, ch2);`	`ASSERT_STRCASEEQ(ch1, ch2);`	Vérifie que les chaînes **ch1** et **ch2** sont identiques en ignorant la casse.
	`EXPECT_STRCASENE(ch1, ch2);`	`ASSERT_STRCASENE(ch1, ch2);`	Vérifie que les chaînes **ch1** et **ch2** ne sont pas identiques en ignorant la casse.

ANNEXES

Termes

Terme Google	Terme équivalent ISTQB	Définition
Test	Test Case	À partir d'un jeu de données / de valeurs d'entrées prédéfinies, vérifier la cohérence / validité des résultats.
Test Case	Test Suite	Il s'agit d'une collection de Test Case (au sens de l'ISTQB) qui permet de tester un comportement spécifique du programme à tester.

ANNEXES

Points forts / points faibles

Points forts	Points faibles
• Détection automatique des tests • Multiplateforme • Support un grand nombre d'assertions différentes fatales et non fatales. • Tests indépendants et parallélisables, grandes performances. • Les résultats des tests sont exportables en XML. • Équipe de développement très active • Intégration dans les EDI modernes	• Aucune interface utilisateur autre que le terminal. • Nécessite un niveau minimum requis en C/C++ et quelques connaissances en POO.