Unit test

Meghívjuk az eljárást, és megnézzük, hogy azt adja-e vissza, amit szeretnénk

Egységek és a tesztjeik

- Függvények, objektumok
- Az alkalmazás építőkövei
- Ha jól működnek, jól működik az alkalmazás
- Jó kipróbálni, látni működés közben
- Lehet írni gyorsan egy kis programot, vagy bekötni a meglévőbe
- Ezután feledésbe merül a következő hibáig vagy átszervezésig

- Hogy lehetne ebből később is hasznot húzni?

Keretrendszerek

Segítségükkel könnyebb a:

- Rendszerezés
- Futtatás
- Kimenet ellenőrzés
- Jelentéskészítés

Megfogalmazhatjuk, hogy adott bemenet esetén milyen kimenetre számítunk és a válasz vagy PIROS vagy ZÖLD.

GoogleTest

GoogleTest

https://code.google.com/p/googletest/

Vannak **testcase**-ek: Logikailag összefüggő **test**ek.

testek: Egy eljárás. Mindenféle **assert** és **expect** van benne

assert/expect: Ellenőriz valamit

gtest/gmock buildelése

```
cd 4. eloadas/external/lib
cmake -GNinja ..
ninja
```

Ez létrehoz majd egy **libgmock_main.a**-t és egy **libgmock.a**-t. Ezek kellenek a linkeléshez.

Egyszerű teszt

```
#include <gtest/gtest.h>

TEST(SimpleTestCase, FailingTest)
{
   FAIL(); // <-- Miért?
}

[----------] 1 test from SimpleTestCase
[ RUN      ] SimpleTestCase.FailingTest
../src/simpleTest.cpp:5: Failure
Failed
[  FAILED  ] SimpleTestCase.FailingTest (0 ms)
```

Options - parancssori beállítások

Bemenet: egy karakterlánc

Kimenet: egy kulcs-érték párokat tartalmazó objektum

```
Options ops("-source /usr -overwrite");
ops.GetOption("source") == "/usr";
ops.GetOption("overwrite") == "true";
```

Options - gtest

```
TEST(OptionTest, OneBooleanOption)
{
   Options ops("-overwrite");

   EXPECT_EQ(1, ops.GetOptions().size());
   EXPECT_EQ("true", ops.GetOption("overwrite"));
}
```

Options - gtest

```
TEST(OptionTest, OneBooleanOption)
{
   Options ops("-overwrite");

   EXPECT_EQ(1, ops.GetOptions().size());
   EXPECT_EQ("true", ops.GetOption("overwrite"));
}
```

OptionTest - Beszédes csoport (testcase) név

OneBooleanOption - Beszédes teszt név

Options - gtest

```
TEST(OptionTest, OneBooleanOption)
{
   Options ops("-overwrite");

   EXPECT_EQ(1, ops.GetOptions().size());
   EXPECT_EQ("true", ops.GetOption("overwrite"));
}
```

Tesztelendő objektum és bemenet összeállítása

Options - gtest

```
TEST(OptionTest, OneBooleanOption)
{
   Options ops("-overwrite");

   EXPECT_EQ(1, ops.GetOptions().size());
   EXPECT_EQ("true", ops.GetOption("overwrite"));
}
```

A várt érték

Options - gtest

```
TEST(OptionTest, OneBooleanOption)
{
   Options ops("-overwrite");

   EXPECT_EQ(1, ops.GetOptions().size());
   EXPECT_EQ("true", ops.GetOption("overwrite"));
}
```

A tesztelendő függvény

Options - gtest

```
TEST(OptionTest, OneBooleanOption)
{
   Options ops("-overwrite");

   EXPECT_EQ(1, ops.GetOptions().size());
   EXPECT_EQ("true", ops.GetOption("overwrite"));
}
```

Elvárjuk, hogy a két paraméter egyenlő legyen

gtest - további elvárások

Feltétel vizsgálat (EXPECT_TRUE, EXPECT_FALSE)

```
EXPECT_TRUE("true" == ops.GetOption("overwrite")); //!!!
```

Összehasonlítás

- _EQ, _NE, _LT, _LE, _GT, _GE

C string összehasonlítás

- _STREQ, _STRNE
- _STRCASEEQ, _STRCASENE

ASSERT_*() - ha nem teljesül, nem ellenőrzi tovább

Új metódus

Options& Options::Add(const Options&)

Hozzáadja a paraméterben tárolt kulcsérték párokat az objektumhoz. Ha már létezett felülírja.

ops1.Add(ops2) != ops2.Add(ops1)

Két teszt. Hol hozzuk létre az ops1 és ops2 objektumokat?

Fixture - állandó környezet

Ha több tesztünkhöz is ugyanazok a bemenő adatok szükségesek.

```
class OptionFixture : public ::testing::Test
{
public:
    OptionFixture();
    ~OptionFixture();

    virtual void SetUp();
    virtual void TearDown();
};
```

Fixture - folyamat

1. Új fixture objektum példányosítása
2. Az objektumon meghívódik a SetUp()
3. Lefut a teszt
4. Meghívódik a TearDown()
5. Törlődik a fixture objektum

Minden egyes teszt új objektumon dolgozik.
Minden korábbi fixture objektumon történt változtatás elvész.

Fixture - egy teszt

TEST_F(OptionFixture, WithoutUnite)

_F - F, mint Fixture

OptionFixture - használt fixture osztály neve

A teszt törzsében elérhetők az adattagok, metódusok a láthatósági módosítóknak megfelelően.

GoogleMock

GoogleMock

http://code.google.com/p/googlemock/

Előfordul, hogy a teszt kimenete:

- egy objektumon végrehajtott művelet (eredmény)
- függ egy objektum metódusától (vezérlés)

IFile, OptionsFileHandler

```
class IFile
public:
    virtual std::string ReadLine() const = 0;
    virtual bool WriteLine(const std::string& line) = 0;

class OptionsFileHandler
public:
    explicit OptionsFileHandler(IFile& file);
    Options Read() const;
    bool Write(const Options& options);
```

Kulcs-érték párok állományból

- Soronként egy kulcs-érték pár egyenlőségjellel elválasztva ("kulcs=érték")
- A OptionsFileHandler::Read kiolvass és feldolgoz minden sort, visszaad egy Options objektumot
- A OptionsFileHandler::Write kiírja a praméterben kapott Options objektumot és jelzi, hogy sikerült e
- Ha nincs több sor a IFile::ReadLine üres stringet ad vissza
- Ha hiba történt az íráskor a IFile::WriteLine visszatérési értéke false

Az OptionsFileHandler működését szeretnénk tesztelni.

Kell egy IFile objektum.

FakeFile

```
class FakeFile : public IFile
{
public:
    virtual std::string ReadLine() const
     { return "-overwrite"; }
    virtual bool WriteLine(const std::string& line)
     { return true; }
};
```

- Sok lehetőség esetén rugalmatlan, kényelmetlen.
- Bizonyos esetekben pont erre van szükség

MockFile - a Mock

```
#include <gmock/gmock.h>

#include "IFile.h"

class MockFile : public IFile
{
public:
    MOCK_CONST_METHOD0(ReadLine, std::string());
    MOCK_METHOD1(WriteLine, bool(const std::string&));
};
```

MockFile

```
#include <gmock/gmock.h>

#include "IFile.h"

class MockFile : public IFile
{
public:
    MOCK_CONST_METHOD0(ReadLine, std::string());
    MOCK_METHOD1(WriteLine, bool(const std::string&));
};
```

A mock objektum az interfészből származik.

MockFile

```
#include <gmock/gmock.h>

#include "IFile.h"

class MockFile : public IFile
{
public:
    MOCK_CONST_METHOD0(ReadLine, std::string());
    MOCK_METHOD1(WriteLine, bool(const std::string&));
};


virtual std::string ReadLine() const = 0;
```

MockFile

```
#include <gmock/gmock.h>

#include "IFile.h"

class MockFile : public IFile
{
public:
    MOCK_CONST_METHOD0(ReadLine, std::string());
    MOCK_METHOD1(WriteLine, bool(const std::string&));
};


virtual std::string ReadLine() const = 0;
```

MockFile

```
#include <gmock/gmock.h>

#include "IFile.h"

class MockFile : public IFile
{
public:
    MOCK_CONST_METHOD0(ReadLine, std::string());
    MOCK_METHOD1(WriteLine, bool(const std::string&));
};

virtual std::string ReadLine() const = 0;
```

MockFile

```
#include <gmock/gmock.h>

#include "IFile.h"

class MockFile : public IFile
{
public:
    MOCK_CONST_METHOD0(ReadLine, std::string());
    MOCK_METHOD1(WriteLine, bool(const std::string&));
};


virtual std::string ReadLine() const = 0;
```

OptionsFileHandlerTest - a Fixture

```
class OptionsFileHandlerTest : public ::testing::Test
{
public:
    OptionsFileHandlerTest()
        : m_sourceLine("source=/usr")
        , m_file()
        , m_fileHandler(m_file)  {}

    std::string m_sourceLine;
    MockFile    m_file;
    OptionsFileHandler m_fileHandler;
};
```

OptionsFileHandlerTest

```
class OptionsFileHandlerTest : public ::testing::Test
{
public:
    OptionsFileHandlerTest()
        : m_sourceLine("source=/usr")
        , m_file()
        , m_fileHandler(m_file)  {}

    std::string m_sourceLine;
    MockFile    m_file;
    OptionsFileHandler m_fileHandler;
};
```

A Mock objektum.

OptionsFileHandlerTest - a Fixture

```
class OptionsFileHandlerTest : public ::testing::Test
{
public:
    OptionsFileHandlerTest()
        : m_sourceLine("source=/usr")
        , m_file()
        , m_fileHandler(m_file)  {}

    std::string m_sourceLine;
    MockFile    m_file;
    OptionsFileHandler m_fileHandler;
};
```

A tesztelendő objektum.

OptionsFileHandlerTest - a Fixture

```
class OptionsFileHandlerTest : public ::testing::Test
{
public:
    OptionsFileHandlerTest()
        : m_sourceLine("source=/usr")
        , m_file()
        , m_fileHandler(m_file)  {}

    std::string m_sourceLine;
    MockFile    m_file;
    OptionsFileHandler m_fileHandler;
};
```

A tesztelendő objektum a Mockot használja.

EXPECT_CALL

```
EXPECT_CALL(mock_object, method
(matchers))
    .With(multi_argument_matcher)  ?
    .Times(cardinality)            ?
    .InSequence(sequences)             *
    .After(expectations)           *
    .WillOnce(action)              *
    .WillRepeatedly(action)        ?
    .RetiresOnSaturation();        ?
```

EXPECT_CALL

```
EXPECT_CALL(mock_object, method (matchers))
    .With(multi_argument_matcher)  ?
    .Times(cardinality)            ?
    .InSequence(sequences)           *
    .After(expectations)           *
    .WillOnce(action)              *
    .WillRepeatedly(action)        ?
    .RetiresOnSaturation();        ?
```

EXPECT_CALL - vezérlés

```
TEST_F(OptionsFileHandlerTest, ReadOneOption)
{
    EXPECT_CALL(m_file, ReadLine())
            .WillOnce(Return(m_overwriteFalse))
            .WillRepeatedly(Return(""));

    Options ops = m_fileHandler.Read();

    EXPECT_EQ(1, ops.GetOptions().size());
    EXPECT_EQ("false", ops.GetOption("overwrite"));
}
```

EXPECT_CALL - vezérlés

```
TEST_F(OptionsFileHandlerTest, ReadOneOption)
{
    EXPECT_CALL(m_file, ReadLine())
            .WillOnce(Return(m_overwriteFalse))
            .WillRepeatedly(Return(""));

    Options ops = m_fileHandler.Read();

    EXPECT_EQ(1, ops.GetOptions().size());
    EXPECT_EQ("false", ops.GetOption("overwrite"));
}
```

Az m_file objektum viselkedését álllítjuk be ...

EXPECT_CALL - vezérlés

```
TEST_F(OptionsFileHandlerTest, ReadOneOption)
{
    EXPECT_CALL(m_file, ReadLine())
            .WillOnce(Return(m_overwriteFalse))
            .WillRepeatedly(Return(""));

    Options ops = m_fileHandler.Read();

    EXPECT_EQ(1, ops.GetOptions().size());
    EXPECT_EQ("false", ops.GetOption("overwrite"));
}
```

... amikor történik egy paraméter nélküli ReadLine hívás ...

EXPECT_CALL - vezérlés

```
TEST_F(OptionsFileHandlerTest, ReadOneOption)
{
    EXPECT_CALL(m_file, ReadLine())
            .WillOnce(Return(m_overwriteFalse))
            .WillRepeatedly(Return(""));

    Options ops = m_fileHandler.Read();

    EXPECT_EQ(1, ops.GetOptions().size());
    EXPECT_EQ("false", ops.GetOption("overwrite"));
}
```

... egyszer ...

EXPECT_CALL - vezérlés

```
TEST_F(OptionsFileHandlerTest, ReadOneOption)
{
    EXPECT_CALL(m_file, ReadLine())
            .WillOnce(Return(m_overwriteFalse))
            .WillRepeatedly(Return(""));

    Options ops = m_fileHandler.Read();

    EXPECT_EQ(1, ops.GetOptions().size());
    EXPECT_EQ("false", ops.GetOption("overwrite"));
}
```

... térjen vissza m_overwriteFalse értékkel ...

EXPECT_CALL - vezérlés

```
TEST_F(OptionsFileHandlerTest, ReadOneOption)
{
    EXPECT_CALL(m_file, ReadLine())
            .WillOnce(Return(m_overwriteFalse))
            .WillRepeatedly(Return(""));

    Options ops = m_fileHandler.Read();

    EXPECT_EQ(1, ops.GetOptions().size());
    EXPECT_EQ("false", ops.GetOption("overwrite"));
}
```

... majd minden további hívás üres stringgel.

EXPECT_CALL - ellenőrzés

```
TEST_F(OptionsFileHandlerTest, WriteTwoOption)
{
    EXPECT_CALL(m_file, WriteLine(m_overwriteFalse)).
Times(0);
    EXPECT_CALL(m_file, WriteLine(m_overwriteTrue)).
WillOnce(Return(true));
    EXPECT_CALL(m_file, WriteLine(m_sourceLine)).
WillOnce(Return(true));

    Options ops("-overwrite false -source /usr -overwrite");

    EXPECT_TRUE(m_fileHandler.Write(ops));
}
```

EXPECT_CALL - ellenőrzés

```
TEST_F(OptionsFileHandlerTest, WriteTwoOption)
{
    EXPECT_CALL(m_file, WriteLine(m_overwriteFalse)).
Times(0);
    EXPECT_CALL(m_file, WriteLine(m_overwriteTrue)).
WillOnce(Return(true));
    EXPECT_CALL(m_file, WriteLine(m_sourceLine)).
WillOnce(Return(true));

    Options ops("-overwrite false -source /usr -overwrite");

    EXPECT_TRUE(m_fileHandler.Write(ops));
}
```

Nem hívódik WriteLine m_overwriteFalse paraméterrel ...

EXPECT_CALL - ellenőrzés

```
TEST_F(OptionsFileHandlerTest, WriteTwoOption)
{
    EXPECT_CALL(m_file, WriteLine(m_overwriteFalse)).
Times(0);
    EXPECT_CALL(m_file, WriteLine(m_overwriteTrue)).
WillOnce(Return(true));
    EXPECT_CALL(m_file, WriteLine(m_sourceLine)).
WillOnce(Return(true));

    Options ops("-overwrite false -source /usr -overwrite");

    EXPECT_TRUE(m_fileHandler.Write(ops));
}
```

... de egyszer hívódik m_overwriteTrue-val és true-t ad ...

EXPECT_CALL - ellenőrzés

```
TEST_F(OptionsFileHandlerTest, WriteTwoOption)
{
    EXPECT_CALL(m_file, WriteLine(m_overwriteFalse)).
Times(0);
    EXPECT_CALL(m_file, WriteLine(m_overwriteTrue)).
WillOnce(Return(true));
    EXPECT_CALL(m_file, WriteLine(m_sourceLine)).
WillOnce(Return(true));

    Options ops("-overwrite false -source /usr -overwrite");

    EXPECT_TRUE(m_fileHandler.Write(ops));
}
... m_sourceLine paraméterrel szintén ...
```

EXPECT_CALL - ellenőrzés

```
TEST_F(OptionsFileHandlerTest, WriteTwoOption)
{
    EXPECT_CALL(m_file, WriteLine(m_overwriteFalse)).
Times(0);
    EXPECT_CALL(m_file, WriteLine(m_overwriteTrue)).
WillOnce(Return(true));
    EXPECT_CALL(m_file, WriteLine(m_sourceLine)).
WillOnce(Return(true));

    Options ops("-overwrite false -source /usr -overwrite");

    EXPECT_TRUE(m_fileHandler.Write(ops));
}
```

A Write visszatérési értéke true lesz az ops paraméterrel.

EXPECT_CALL

Több EXPECT_CALL kiértékelése aluról felfelé történik és az első egyezés számít!!

```
   EXPECT_CALL(m_file, WriteLine(m_overwriteTrue)).
WillOnce(Return(false));
   EXPECT_CALL(m_file, WriteLine(_)).WillOnce(Return
(true));

Options ops("-overwrite -target /tmp");

m_fileHandler.Write(ops);
```

Jó unitteszt

- Beszédes a neve
- Azt teszteli, amit a neve sejtet
- Egyszerű a szerkezete
- Egy dolgot ellenőriz
- Dokumentálja a működést
- A kódváltoztatáskor íródik
- Nem függ előző tesztek eredményétől
- Nem függ külső erőforrásoktól

További gondolatok

- Változik az igény (többlet munka)
- Hogyan teszteljük a teszteket? (egyszerűség, kóddal)
- Hogyan teszteljünk régi kódot? (sehogy, jó gyakorlás, sok buktató, átírás)
- Tényleg csak a tesztek miatt át kell írni a meglévő működő kódot? (nem, de jobb szerkezet)
- Mi történik ha nem írunk valamire tesztet? (lehet hiba, majd arra)
- Hogyan unittesztelünk nagy komponenseket? (sehogy, nem unitteszt, gyakrabban változhat)
- Mikor ne írjunk unittesztet? (tökéletes munka)

TDD

- Test Driven Development
- Inkább tervezés, mint tesztelés
- Csak a szükséges működést írjuk meg és rögötön ellenőrizzük is a helyességét
- Ciklusokban dolgozunk:
  - írunk egy PIROS tesztet
  - megírjuk/javítjuk a kódot (akárhogy)
  - addig ismételjük még ZÖLD nem lesz a teszt
  - szépítjük a kódot (refactor)

BDD

BDD - Behaviour driven development

Normál, emberi mondatokat írunk.

A tesztkörnyezet ebből hajtja végre a tesztet.

Laikusok is meg tudják írni/érteni, sőt inkább nekik szól.

Cucumber

Eredetileg ruby, de sok portja van (C++ is)

Eredetileg angol mondatokat lehet írni, de le van fordítva egy csomó nyelvre (magyarra is).

"eljárásokat" írunk, ha azt akarjuk, hogy ne sikerüljön a teszteset exception-t kell dobni

http://cukes.info/

Példa - Reconnect disconnect

Feature: Reconnect
    In order to handle wifi and network disconnections
    As a user of the program
    I want the program to try reconnect after network disconnection happened

Scenario: Disconnect notification
    Given I have disabled the network
    When I wait 10 sec
    Then the error message should be network is down

Ezt .feature fájlba érdemes menteni

Példa - Reconnect disconnect

Feature: Reconnect
    In order to handle wifi and network disconnections
    As a user of the program
    I want the program to try reconnect after network disconnection happened

Scenario: Disconnect notification
    Given I have disabled the network
    When I wait 10 sec
    Then the error message should be network is down

Elnevezzük a feature-t. Nem igazán lényeges.

Példa - Reconnect disconnect

Feature: Reconnect
    In order to handle wifi and network disconnections
    As a user of the program
    I want the program to try reconnect after network disconnection happened

Scenario: Disconnect notification
    Given I have disabled the network
    When I wait 10 sec
    Then the error message should be network is down

Leírás. Bármi lehet, nem ellenőrzi a program. Ki is lehet hagyni.

Példa - Reconnect disconnect

Feature: Reconnect
    In order to handle wifi and network disconnections
    As a user of the program
    I want the program to try reconnect after network disconnection happened

Scenario: Disconnect notification
    Given I have disabled the network
    When I wait 10 sec
    Then the error message should be network is down

Így kezdődnek a különböző tesztesetek

Példa - Reconnect disconnect

Feature: Reconnect
    In order to handle wifi and network disconnections
    As a user of the program
    I want the program to try reconnect after network disconnection happened

Scenario: Disconnect notification
    Given I have disabled the network
    When I wait 10 sec
    Then the error message should be network is down

Különböző lépések. Végrehajtja őket.

Futtassuk!

```
You can implement step definitions for undefined steps with these snippets:

Given /^I have disabled the network$/ do
  pending # express the regexp above with the code you wish you had
end

When /^I wait (\d+) sec$/ do |arg1|
  pending # express the regexp above with the code you wish you had
end

Then /^the error message should be network is down$/ do
  pending # express the regexp above with the code you wish you had
end
```

Leírja nekünk, hogy mit kell megvalósítani

Implementáció

```
Given /^I have disabled the network$/ do
  system('rfkill block all')
end

When /^I wait (\d+) sec$/ do |sec|
  sleep(sec.to_i)
end

Then /^the error message should be (.*)$/ do
|msg|
 realStatus =  File.read('status.txt')
 if not msg.strip == realStatus.strip
    raise msg
 end
```

Implementáció

```
Given /^I have disabled the network$/ do
  system('rfkill block all')
end

When /^I wait (\d+) sec$/ do |sec|
  sleep(sec.to_i)
end

Then /^the error message should be (.*)$/ do
|msg|
 realStatus =  File.read('status.txt')
 if not msg.strip == realStatus.strip
    raise msg
 end
```

Implementáció

```
Given /^I have disabled the network$/ do
  system('rfkill block all')
end

When /^I wait (\d+) sec$/ do |sec|
  sleep(sec.to_i)
end

Then /^the error message should be (.*)$/ do
|msg|
 realStatus =  File.read('status.txt')
 if not msg.strip == realStatus.strip
    raise msg
 end
```

Implementáció

```
Given /^I have disabled the network$/ do
  system('rfkill block all')
end

When /^I wait (\d+) sec$/ do |sec|
  sleep(sec.to_i)
end

Then /^the error message should be (.*)$/ do
|msg|
 realStatus =  File.read('status.txt')
 if not msg.strip == realStatus.strip
    raise msg
 end
```

Valamilyen alkönyvtárba, valamilyen .rb fájlba érdemes menteni.

A futtatókörnyezet automatikusan megkeresi.

Eredmény

```
  In order to handle wifi and network disconnections
  As a user of the program
  I want the program to try reconnect after network disconnection happened

  Scenario: Reconnect notification
    Given I have disabled the network
    When I wait 10 sec
    Then the error message should be network is down

1 scenario (1 passed)
3 steps (3 passed)
0m10.010s
```

Könnyen bővíthető

Új scenario:

Scenario: Reconnect notification
  Given I have enabled the network
  When I wait 10 sec
  Then the error message should be network is up

Új step:

```
Given /^I have enabled the network$/ do
  system('rfkill unblock all')
end
```

Ami kimaradt

- Táblázatok - Különböző adatok miatt ne kelljen az az eseteket lemásolni
- Before, after - setup, teardown
- 'But' - mi nem történt
- ...

Magyarul

Jellemző: Osztás

Azért, hogy elkerüljem a buta hibákat

a számológépeknek tudniuk kell osztani.

Forgatókönyv: Egyszerű számok

Amennyiben beütök a számológépbe egy 3-ast

És beütök a számológépbe egy 2-est

Ha megnyomom a divide gombot

Akkor eredményül 1.5-öt kell kapnom

Sikuli
UI tesztelés

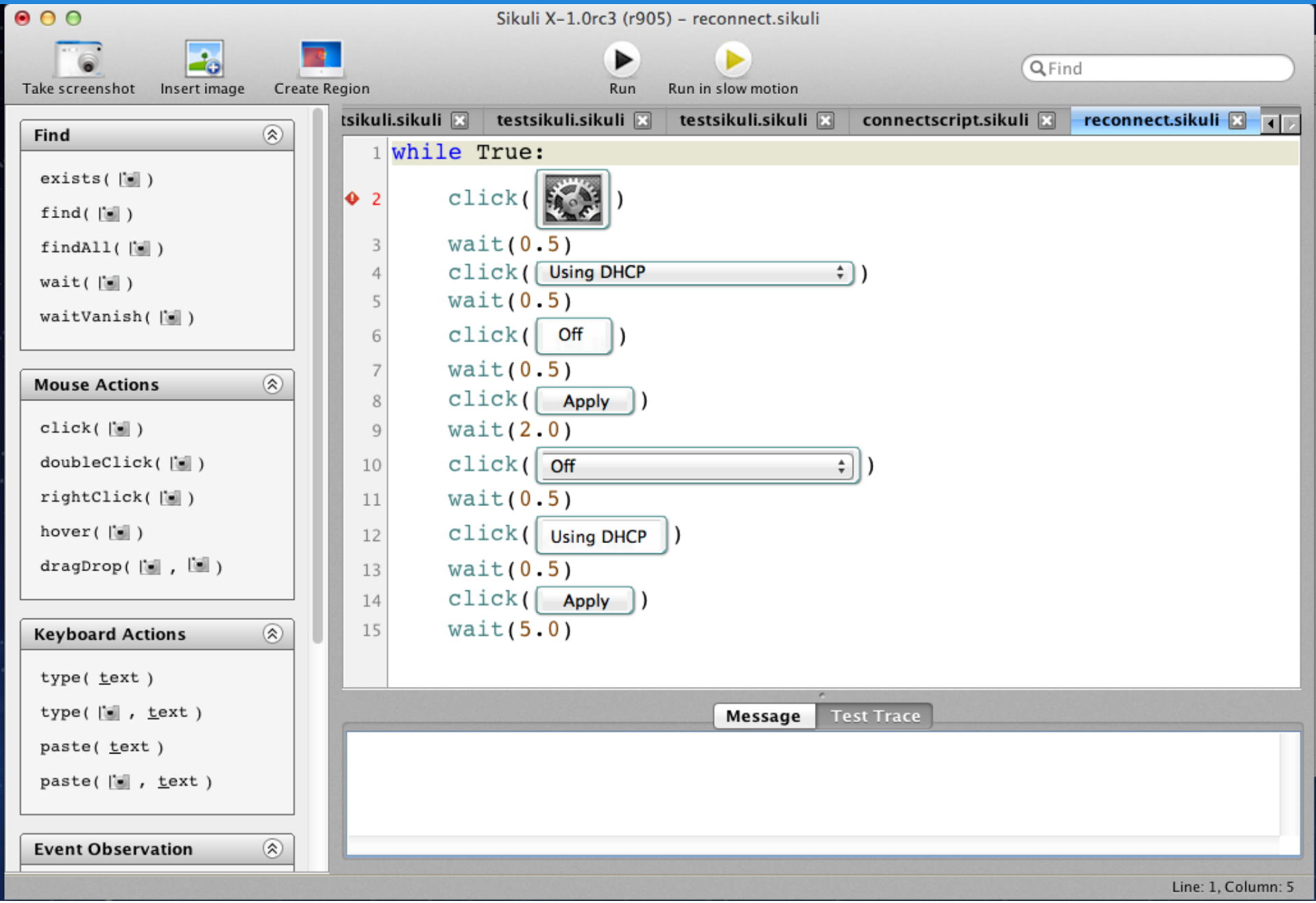
http://www.sikuli.org/

Igen jó dolog

Olyan script, amiben képeket lehet literálként használni.

Hozzá vannak kötve képfeldolgozó algoritmusok. Képes a képernyőn képeket keresni.

Tud szöveget begépelni, egeret mozgatni, kattintani, stb.


Sikuli X-1.0rc3 (r905) - reconnect.sikuli
Take screenshot
Insert image
Create Region
Run
Run in slow motion
Find
Find
exists( )
find( )
findAll( )
wait( )
waitVanish( )
Mouse Actions
click( )
doubleClick( )
rightClick( )
hover( )
dragDrop(,)
Keyboard Actions
type(text)
type(, text)
paste(text)
paste(, text)
Event Observation
tsikuli.sikuli
testsikuli.sikuli
testsikuli.sikuli
connectscript.sikuli
reconnect.sikuli
1 while True:
2 click( )
3 wait(0.5)
4 click(Using DHCP)
5 wait(0.5)
6 click(Off)
7 wait(0.5)
8 click(Apply)
9 wait(2.0)
10 click(Off)
11 wait(0.5)
12 click(Using DHCP)
13 wait(0.5)
14 click(Apply)
15 wait(5.0)
Message
Test Trace
Line: 1, Column: 5

Python programot írunk.

JVM van alatta, javában is lehet írni.

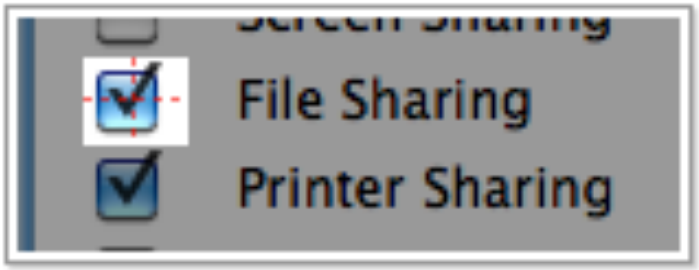
OpenCV-t használ a képfelismeréshez.

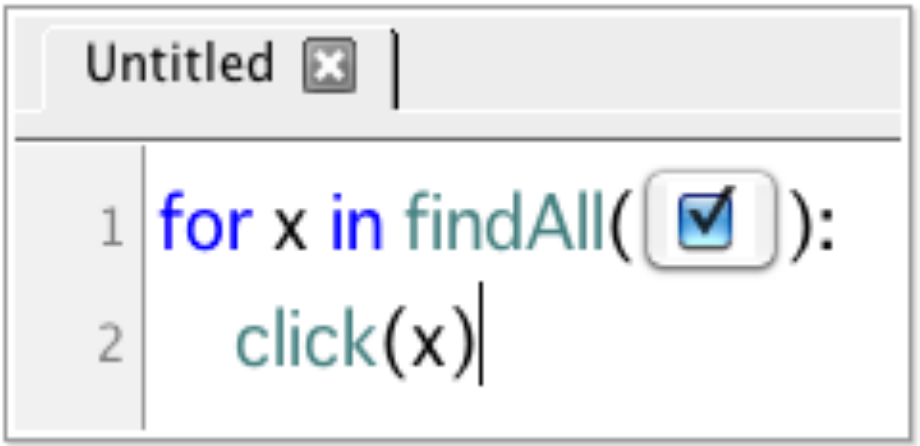
A képek rendes literálok, lehet változóknak is értékül adni

Be lehet állítani a képfelismerési érzékenységet

Kicsit összetettebb példa

Kattintsuk ki az összes checkboxot a képen





Forrás

http://simpleprogrammer.com/2010/10/15/the-purpose-of-unit-testing/

http://code.google.com/p/googletest/

http://code.google.com/p/googlemock/