Google Test练习配置

TDD工作坊C++配套教程

2019.5

练习-环境准备

- 你的座位号码就是你本人的git repo
  - 例如：座位号g121，则你的git repo是 http://guest@129.211.134.67/Bonobo.Git.Server/g121.git
- 请先clone到本地：git clone http://guest@129.211.134.67/Bonobo.Git.Server/g121.git C:\TDD\g121
- 后续提交操作都可以通过命令行完成
  - git add -A #添加全部文件
  - git commit -m "本次提交的改动内容" #提交到本地
  - git push #推送到远程服务器

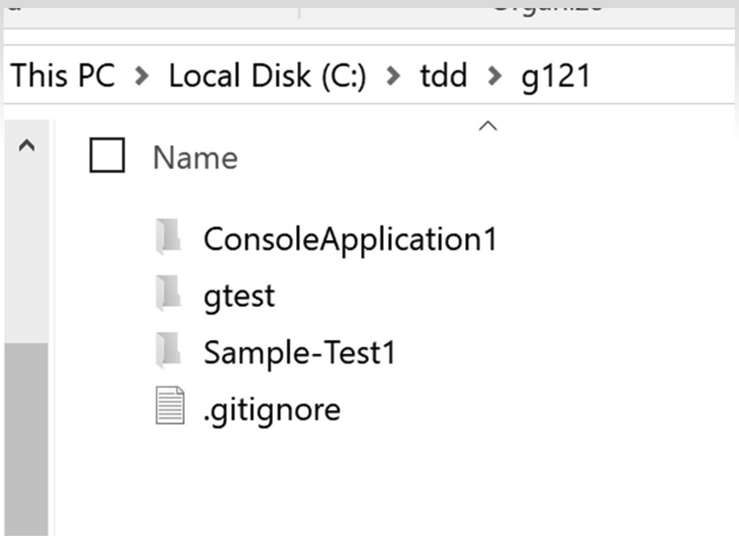
练习-下载练习代码

- ftp://129.211.134.67
- 用户名：Jason
- 密码: Jason2019

- 请下载文件
  - GoogleTest.pdf
  - Example.zip

练习-将样例代码解压到指定目录下

- 将Example.zip解压到C:\TDD\g121下，解压后文件目录结构应该是这样的：



This PC > Local Disk (C:) > tdd > g121

Name

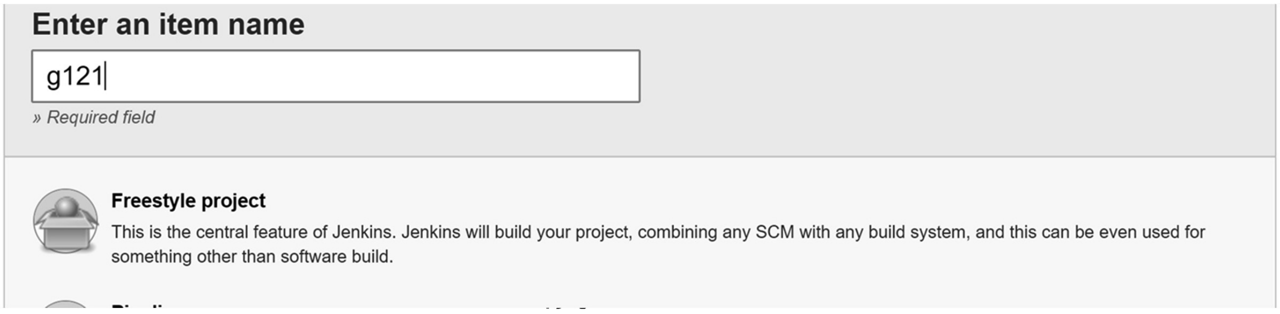
ConsoleApplication1
gtest
Sample-Test1
.gitignore

第一次提交到代码库

- git add -A #添加全部文件
- git commit -m “本次提交的改动内容” #提交到本地
- git push #推送到远程服务器

配置Jenkins Job （1）

- 配置Jenkins Job
  - 用户名：guest
  - 密码：guest2019
  - Jenkins：http://129.211.134.67:8080/
  - 创建个人job，使用你的座位号作为job名称，点击new item，填写座位号，示例如下
  - 选择Freestyle project



Enter an item name

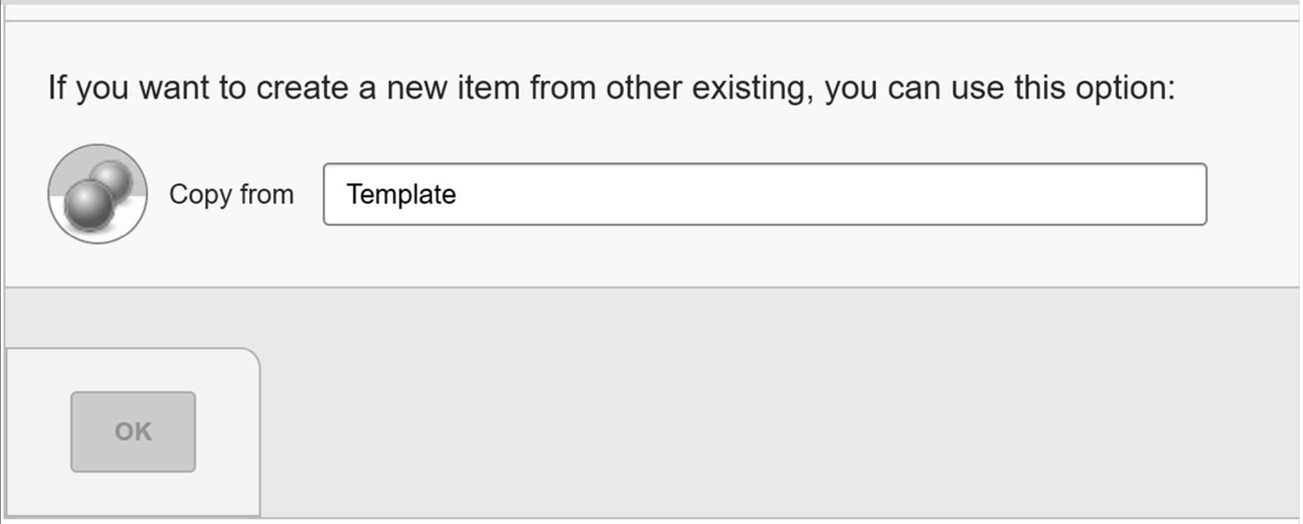
g121

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

配置Jenkins Job （2）复制模板配置

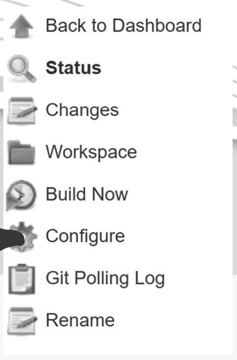


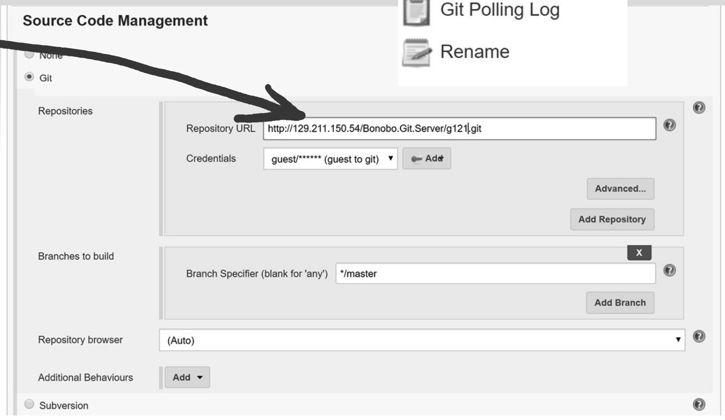
If you want to create a new item from other existing, you can use this option:

Copy from: Template

配置Jenkins Job （3）

- 选择Configure
- 选择Source Code Management -> Git
- 填写repo URL





Source Code Management

None

Git

Repositories

Repository URL: http://129.211.150.54/Bonobo.Git.Server/g121.git

Credentials: guest/****** (guest to git)

Branches to build

Branch Specifier (blank for 'any'): */master

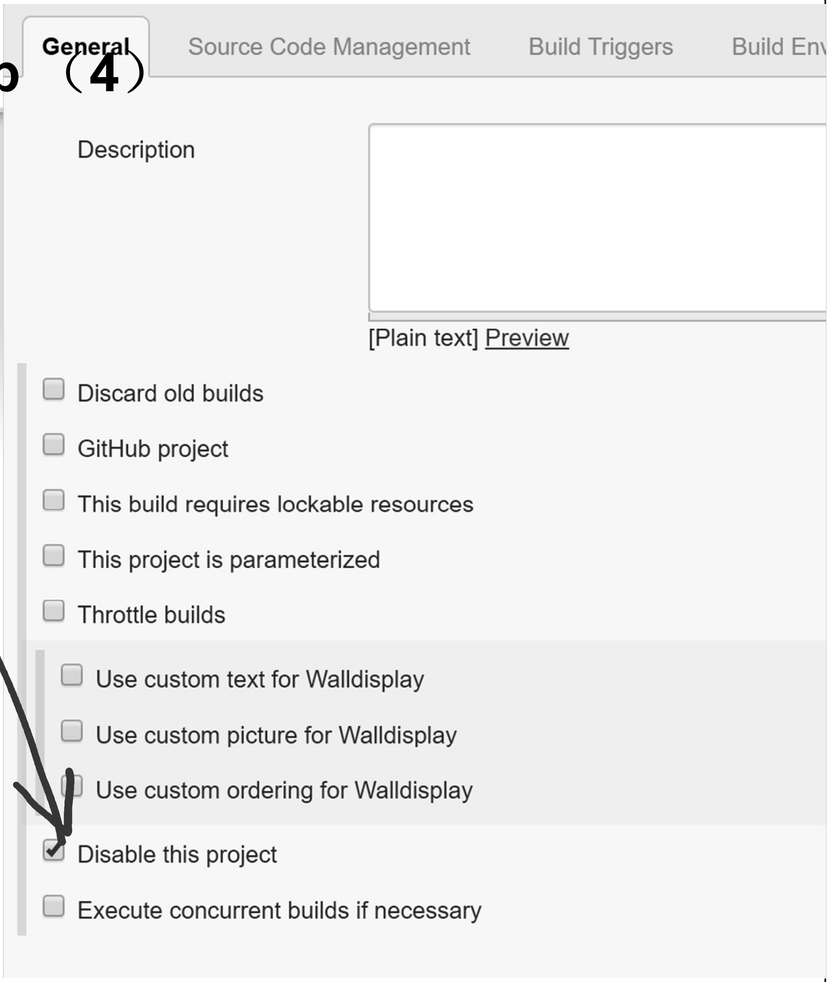
Repository browser: (Auto)

Additional Behaviours

Subversion

配置Jenkins Job （4）

- 反选“Disable this project”



General | Source Code Management | Build Triggers | Build Env

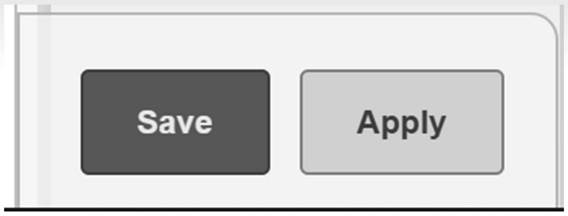
Description

[Plain text] Preview

- [ ] Discard old builds
- [ ] GitHub project
- [ ] This build requires lockable resources
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Use custom text for Walldisplay
- [ ] Use custom picture for Walldisplay
- [ ] Use custom ordering for Walldisplay
- [x] Disable this project
- [ ] Execute concurrent builds if necessary

配置Jenkins Job （5）

- 保存



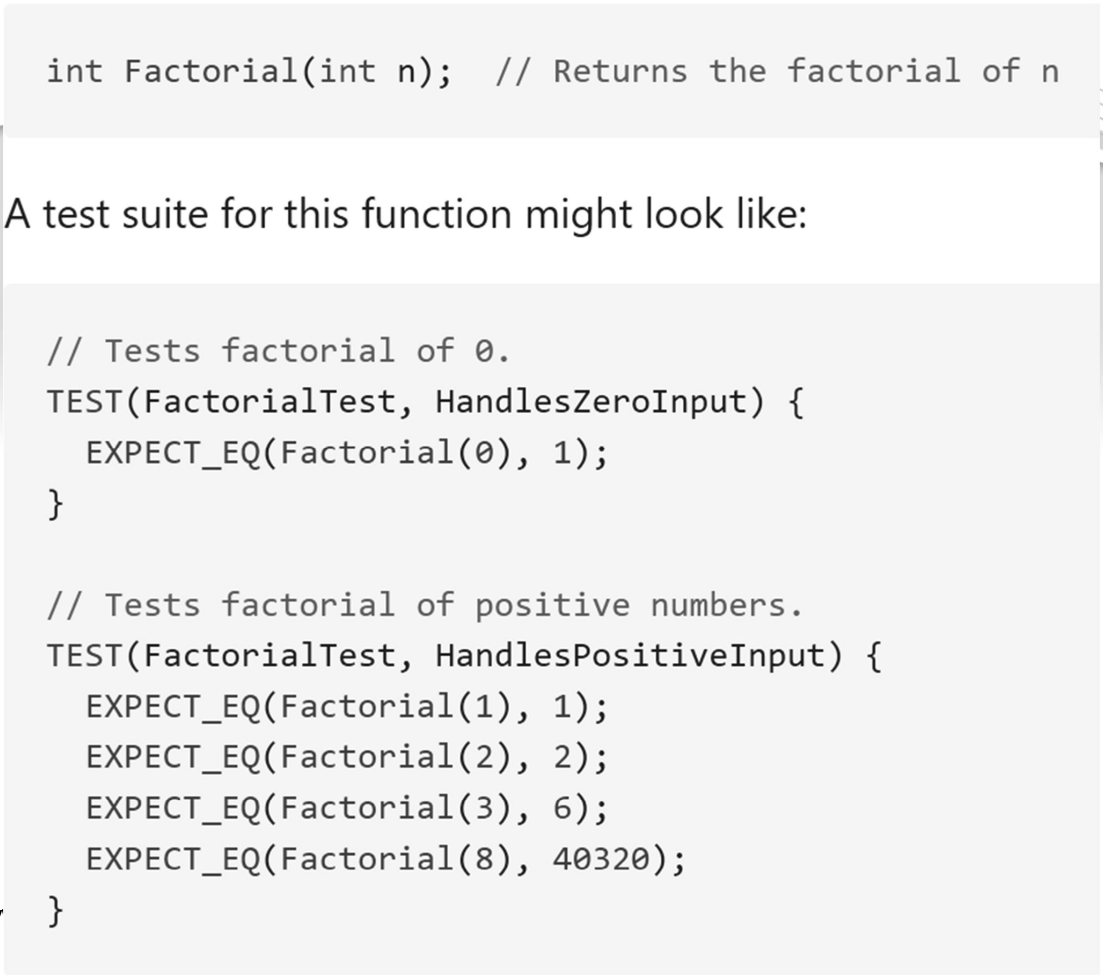
最常用的断言

```
ASSERT_EQ(x.size(), y.size()) << "Vectors x and y are of unequal length";

for (int i = 0; i < x.size(); ++i) {
  EXPECT_EQ(x[i], y[i]) << "Vectors x and y differ at index " << i;
}
```

Google Test断言

Fatal assertion	Nonfatal assertion	Verifies
`ASSERT_EQ(val1, val2);`	`EXPECT_EQ(val1, val2);`	`val1 == val2`
`ASSERT_NE(val1, val2);`	`EXPECT_NE(val1, val2);`	`val1 != val2`
`ASSERT_LT(val1, val2);`	`EXPECT_LT(val1, val2);`	`val1 < val2`
`ASSERT_LE(val1, val2);`	`EXPECT_LE(val1, val2);`	`val1 <= val2`
`ASSERT_GT(val1, val2);`	`EXPECT_GT(val1, val2);`	`val1 > val2`
`ASSERT_GE(val1, val2);`	`EXPECT_GE(val1, val2);`	`val1 >= val2`



```
int Factorial(int n);  // Returns the factorial of n
```

A test suite for this function might look like:

```
// Tests factorial of 0.
TEST(FactorialTest, HandlesZeroInput) {
  EXPECT_EQ(Factorial(0), 1);
}

// Tests factorial of positive numbers.
TEST(FactorialTest, HandlesPositiveInput) {
  EXPECT_EQ(Factorial(1), 1);
  EXPECT_EQ(Factorial(2), 2);
  EXPECT_EQ(Factorial(3), 6);
  EXPECT_EQ(Factorial(8), 40320);
}
```

异常处理

```
TEST(CRomanConverter, Exception_Expect) {
    CRomanConverter roman;
    EXPECT_THROW(roman.WillThrowException(), std::bad_exception);
}
```

Google Mock (1)

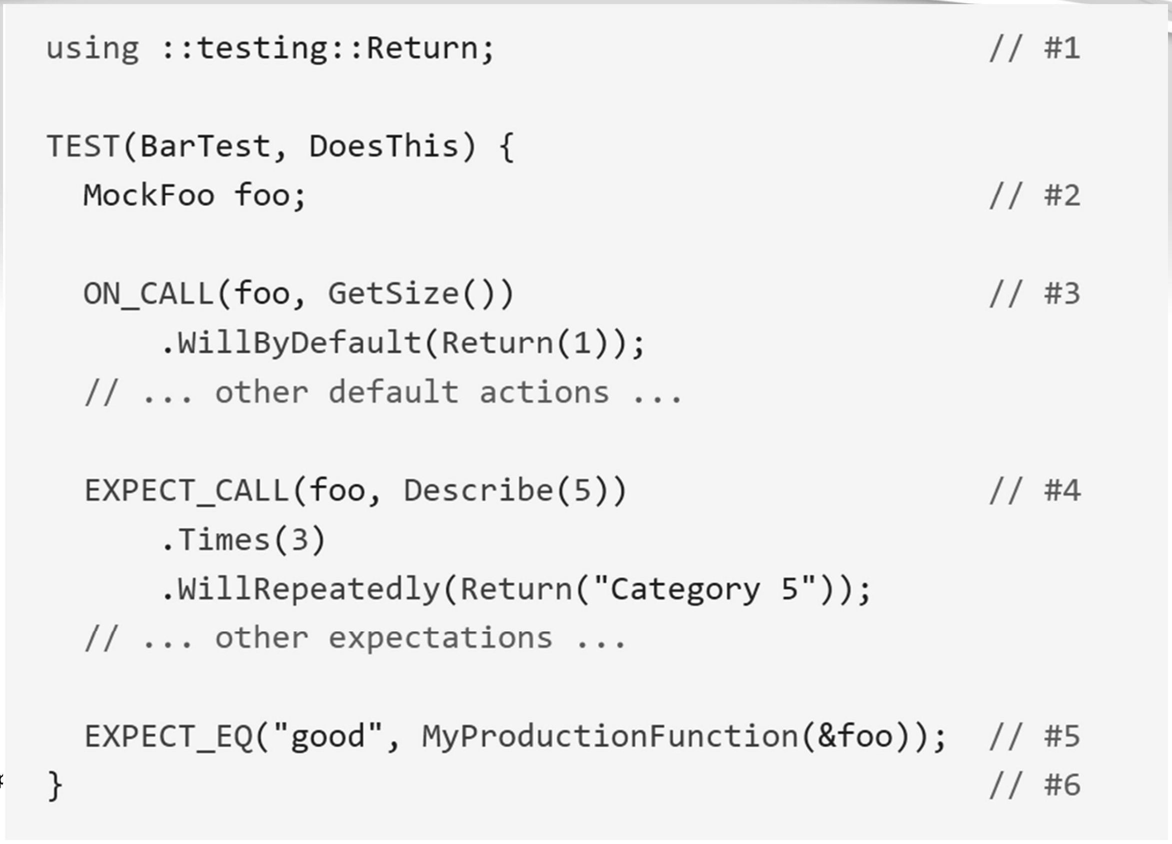
```
class Foo {
  ...
  virtual ~Foo();
  virtual int GetSize() const = 0;
  virtual string Describe(const char* name) = 0;
  virtual string Describe(int type) = 0;
  virtual bool Process(Bar elem, int count) = 0;
};
```

Google Mock (2)

```
#include "gmock/gmock.h"

class MockFoo : public Foo {
  ...
  MOCK_METHOD(int, GetSize, (), (const, override));
  MOCK_METHOD(string, Describe, (const char* name), (override));
  MOCK_METHOD(string, Describe, (int type), (override));
  MOCK_METHOD(bool, Process, (Bar elem, int count), (override));
};
```

Google Mock (3)



```
using ::testing::Return;                            // #1

TEST(BarTest, DoesThis) {
  MockFoo foo;                                      // #2

  ON_CALL(foo, GetSize())                           // #3
      .WillByDefault(Return(1));
  // ... other default actions ...

  EXPECT_CALL(foo, Describe(5))                     // #4
      .Times(3)
      .WillRepeatedly(Return("Category 5"));
  // ... other expectations ...

  EXPECT_EQ("good", MyProductionFunction(&foo));  // #5
}                                                   // #6
```

http