XcodeGuide
A guide for using the Google Testing Framework with Xcode on Mac OS X
Phase-Deploy

Updated Aug 19, 2010 by w...@google.com

This guide will explain how to use the Google Testing Framework in your Xcode projects on Mac OS X. This tutorial begins by quickly explaining what to do for experienced users. After the quick start, the guide goes provides additional explanation about each step.

Quick Start

Here is the quick guide for using Google Test in your Xcode project.

1. Download the source from the website using this command: `svn checkout http://googletest.googlecode.com/svn/trunk/ googletest-read-only`
2. Open up the `gtest.xcodeproj` in the `googletest-read-only/xcode/` directory and build the gtest.framework.
3. Create a new "Shell Tool" target in your Xcode project called something like "UnitTests"
4. Add the gtest.framework to your project and add it to the "Link Binary with Libraries" build phase of "UnitTests"
5. Add your unit test source code to the "Compile Sources" build phase of "UnitTests"
6. Edit the "UnitTests" executable and add an environment variable named "DYLD_FRAMEWORK_PATH" with a value equal to the path to the framework containing the gtest.framework relative to the compiled executable.
7. Build and Go

The following sections further explain each of the steps listed above in depth, describing in more detail how to complete it including some variations.

Get the Source

Currently, the gtest.framework discussed here isn't available in a tagged release of Google Test, it is only available in the trunk. As explained at the Google Test site, you can get the code from anonymous SVN with this command:

```
svn checkout http://googletest.googlecode.com/svn/trunk/ googletest-read-only
```

Alternatively, if you are working with Subversion in your own code base, you can add Google Test as an external dependency to your own Subversion repository. By following this approach, everyone that checks out your svn repository will also receive a copy of Google Test (a specific version, if you wish) without having to check it out explicitly. This makes the set up of your project simpler and reduces the copied code in the repository.

To use `svn:externals`, decide where you would like to have the external source reside. You might choose to put the external source inside the trunk, because you want it to be part of the branch when you make a release. However, keeping it outside the trunk in a version-tagged directory called something like `third-party/googletest/1.0.1`, is another option. Once the location is established, use `svn propedit svn:externals _directory_` to set the svn:externals property on a directory in your repository. This directory won't contain the code, but be its versioned parent directory.

The command `svn propedit` will bring up your Subversion editor, making editing the long, (potentially multi-line) property simpler. This same method can be used to check out a tagged branch, by using the appropriate URL (e.g. `http://googletest.googlecode.com/svn/tags/release-1.0.1`). Additionally, the svn:externals property allows the specification of a particular revision of the trunk with the `-r_##_` option (e.g. `externals/src/googletest -r60 http://googletest.googlecode.com/svn/trunk`).

Here is an example of using the svn:externals properties on a trunk (read via `svn propget`) of a project. This value checks out a copy of Google Test into the `trunk/externals/src/googletest/` directory.

```
[Computer:svn] user$ svn propget svn:externals trunk
externals/src/googletest http://googletest.googlecode.com/svn/trunk
```

Add the Framework to Your Project

The next step is to build and add the gtest.framework to your own project. This guide describes two common ways below.

- **Option 1** --- The simplest way to add Google Test to your own project, is to open gtest.xcodeproj (found in the xcode/ directory of the Google Test trunk) and build the framework manually. Then, add the built framework into your project using the "Add->Existing Framework..." from the context menu or "Project->Add..." from the main menu. The gtest.framework is relocatable and contains the headers and object code that you'll need to make tests. This method requires rebuilding every time you upgrade Google Test in your project.
- **Option 2** --- If you are going to be living off the trunk of Google Test, incorporating its latest features into your unit tests (or are a Google Test developer yourself). You'll want to rebuild the framework every time the source updates. to do this, you'll need to add the gtest.xcodeproj file, not the framework itself, to your own Xcode project. Then, from the build products that are revealed by the project's disclosure triangle, you can find the gtest.framework, which can be added to your targets (discussed below).

Make a Test Target

To start writing tests, make a new "Shell Tool" target. This target template is available under BSD, Cocoa, or Carbon. Add your unit test source code to the "Compile Sources" build phase of the target.

Next, you'll want to add gtest.framework in two different ways, depending upon which option you chose above.

- **Option 1** --- During compilation, Xcode will need to know that you are linking against the gtest.framework. Add the gtest.framework to the "Link Binary with Libraries" build phase of your test target. This will include the Google Test headers in your header search path, and will tell the linker where to find the library.
- **Option 2** --- If your working out of the trunk, you'll also want to add gtest.framework to your "Link Binary with Libraries" build phase of your test target. In addition, you'll want to add the gtest.framework as a dependency to your unit test target. This way, Xcode will make sure that gtest.framework is up to date, every time your build your target. Finally, if you don't share build directories with Google Test, you'll have to copy the gtest.framework into your own build products directory using a "Run Script" build phase.

Set Up the Executable Run Environment

Since the unit test executable is a shell tool, it doesn't have a bundle with a `Contents/Frameworks` directory, in which to place gtest.framework. Instead, the dynamic linker must be told at runtime to search for the framework in another location. This can be accomplished by setting the "DYLD_FRAMEWORK_PATH" environment variable in the "Edit Active Executable ..." Arguments tab, under "Variables to be set in the environment:". The path for this value is the path (relative or absolute) of the directory containing the gtest.framework.

If you haven't set up the DYLD_FRAMEWORK_PATH, correctly, you might get a message like this:

```
[Session started at 2008-08-15 06:23:57 -0600.]
  dyld: Library not loaded: @loader_path/../Frameworks/gtest.framework/Versions/A/gtest
    Referenced from: /Users/username/Documents/Sandbox/gtestSample/build/Debug/WidgetFrameworkTest
    Reason: image not found
```

To correct this problem, got to the directory containing the executable named in "Referenced from:" value in the error message above. Then, with the terminal in this location, find the relative path to the directory containing the gtest.framework. That is the value you'll need to set as the DYLD_FRAMEWORK_PATH.

Build and Go

Now, when you click "Build and Go", the test will be executed. Dumping out something like this:

```
[Session started at 2008-08-06 06:36:13 -0600.]
[==========] Running 2 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 2 tests from WidgetInitializerTest
[ RUN      ] WidgetInitializerTest.TestConstructor
[       OK ] WidgetInitializerTest.TestConstructor
[ RUN      ] WidgetInitializerTest.TestConversion
[       OK ] WidgetInitializerTest.TestConversion
[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran.
[  PASSED  ] 2 tests.

The Debugger has exited with status 0.
```

Summary

Unit testing is a valuable way to ensure your data model stays valid even during rapid development or refactoring. The Google Testing

Unit testing is a valuable way to ensure your data model stays valid even during rapid development or refactoring. The Google Testing Framework is a great unit testing framework for C and C++ which integrates well with an Xcode development environment.