Getting Started

See the Video Tutorials At : http://www.youtube.com/playlist?list=PL0C32F89E8BBB5368

Installation
All you need to do to use ApprovalTests is simply include the ApprovalTests.dll in your project. Then use it with your favorite Testing Framework (currently supports MSTest, Nunit, MBUnit, Xunit).

Parts of a Test
All tests (unit and otherwise) contain 2 parts:

Do
Verify

ApprovalTests is a way to handle the second part: Verification.
All call's will look about the same. Approvals.Verify(objectToBeVerified)

Strings

Let's say you wanted to test if a string was being built correctly.

Do Verify	->	create a string with "Approval" append "Tests" to it Verify the resulting string	->	string s = "Approval"; s += "Tests"; Approvals.*Verify*(s);

If you see "ApprovalTests" as the result, simply Approve The Result (see below).

Objects

Let's say you wanted to test a customized StringBuilder was creating text correctly.

Do
Verify
	->
	
		create my String Builder
		append "Approval" to it
		append "Tests" to it
		Verify the object
			->
			
				var s = **new** MyStringBuilder();
				s.append("Approval");
				s.append("Tests");
				Approvals.*Verify*(s.toString());

If you see "ApprovalTests" as the result, simply Approve The Result (see below).
It's important to notice that you will need to create a *useful* instance of the toString() Method for objects you want to use.

Arrays
Let's say you wanted to test an array of Strings

Do Verify	->	create a String Array set 1st index to “Approval” set 2nd index to “Tests” Verify the array	- >	var s = **new** string[2]; s[0] = "Approval"; s[1] = "Tests"; Approvals.*VerifyAll*(s, "Text");

Note the use of the Label “Text”. This is needed for
and the resulting approval file will contain them:

Text[0] = Approval
Text[1] = Tests

Again, simply Approve The Result (see below).

WinForms / WPF
Let’s say you wanted to test you’ve created a winform correctly.

Do Verify	->	create a TvGuide select show for 3pm Verify the TvGuide	- >	TvGuide tv = **new** TvGuide(); tv.SelectTime("3pm"); WpfApprovals.*Verify*(tv);

First, I want to note that even though there is a UI and a select box for times, I’m not “poking” it to select the time. Just because we are looking at the UI at the end, doesn’t mean I need to manipulate it directly. We are programmers, and are not limited by the constraints of the UI. I simple expose a selectTime(String value) function.

Second, this will produce a screen shot of the form as a result. Simply Approve The Result when it’s ready(see below).

Approving The Result

When you run a test with an approval, it will generate a file named “YourTestClass.YourTestMethod.received.txt” (or png, html, etc) and place it in the same directory as your test.

For the test to pass, this file must match:
YourTestClass.YourTestMethod.approved.txt

There are many ways to do this. But basically you are going to just copy/move the .received file to the .approved file
1) Rename the .received file to .approved
   OR
2) run the “move” command that is displayed (also added to your clipboard) in the command line
   OR

3) Use “use whole file” on a diff reporter

It’s doesn’t matter how you do it.
Note: if the files match, then the received file will be deleted.
Note: you must include the approved files in your source control, you will never need to check in the received files.

Reporters

If an approval fails, then a reporter will be called that will show both the “.received” and “.approved” files. There are many reporters, and you can create your own.

Each approve() call has a default reporter, however you can change which reporter is used.
The simplest way to use a different reporter is to use the Attribute
[UseReporter(**typeof**(Reporter))]
You can also use multiple reporters at the same time
[UseReporter(**t**(**typeof**(FileLuncherReporter), (**typeof**(ClipboardReporter),)]

you can annotate at either the method or class level. You can also add this at the Assembly Level by adding the following to the AssemblyInfo.cs file:
[assembly: UseReporter(**typeof**(Reporter))]

Here are some common Reporters and uses

DiffReporter	Launches an instance of the best matching DiffReporter on your system
ClipboardReporter	Copies the command line move statement needed to approve the result to you Clipboard.
FileLauncherReporter	Opens the .received file
ImageReporter	Launches an instance of TortoiseSvnImageDiff
NunitReporter	Text only, displays the contents of the files as a AssertEquals failure
MsTestReporter	Text only, displays the contents of the files as a AssertEquals failure
NotepadLancher	Opens the .received file in notepad

QuietReporter	outputs the move command to the console. Great for build systems
BeyondCompareReporter	Uses the Beyond Compare Diff Tool
WinMergeReporter	Uses the Win Merge Diff Tool