

EXPLORE

Deliver running, tested,
accepted stories

RESOURCES

http://www.jamesshore.com/Agile-Book/iteration_planning.html

EXPLORE PRACTICES

Iteration Planning and Monitoring

Technical Practices

- Test-Driven Development
- Pair Programming
- Refactoring
- Continuous Integration

Iteration Show and Tell

Iteration Retrospective

WHY ITERATIONS

Iterations are the heartbeat of execution for a project

Iterations are a critical risk management practice

From James Shore Iteration Planning Article —

Programming schedules die in inches. At first you're on schedule: "I'll be done once I finish this test." Then you're limping. "I'll be done as soon as I fix this bug." Then gasping. "I'll be done as soon as I research this API flaw... no, really." Before you know it, two days have gone by and your task has taken twice as long as you estimated.

Death by inches sneaks up on a team. Each delay is only a few hours, so it doesn't feel like a delay, but they multiply across the thousands of tasks in a project. The cumulative effects are devastating.

ITERATION PLANNING & MONITORING

Theme

- A statement of the what the iteration is expected to accomplish
- Provides a focus for the iteration

Iteration Planning

- Iteration Length
- Estimating and Task Size
 - Stories Rule of Thumb – 2-5 Days of Effort
 - Tasks – less than 8 hours

Participants

- The Entire Team

How Long is the meeting?

- 1-2 Hours per of iteration

ITERATION PLANNING MEETING PREPARATION

Product Owner Responsibilities	Development Team Responsibilities
Review the current iteration results and determine impacts to the next iteration	Review the highest ranked items in the backlog and prepare questions
Review backlog items and re-rank as necessary	Understand and communicate the teams expected velocity for the upcoming iteration

RESULTS

The iteration theme

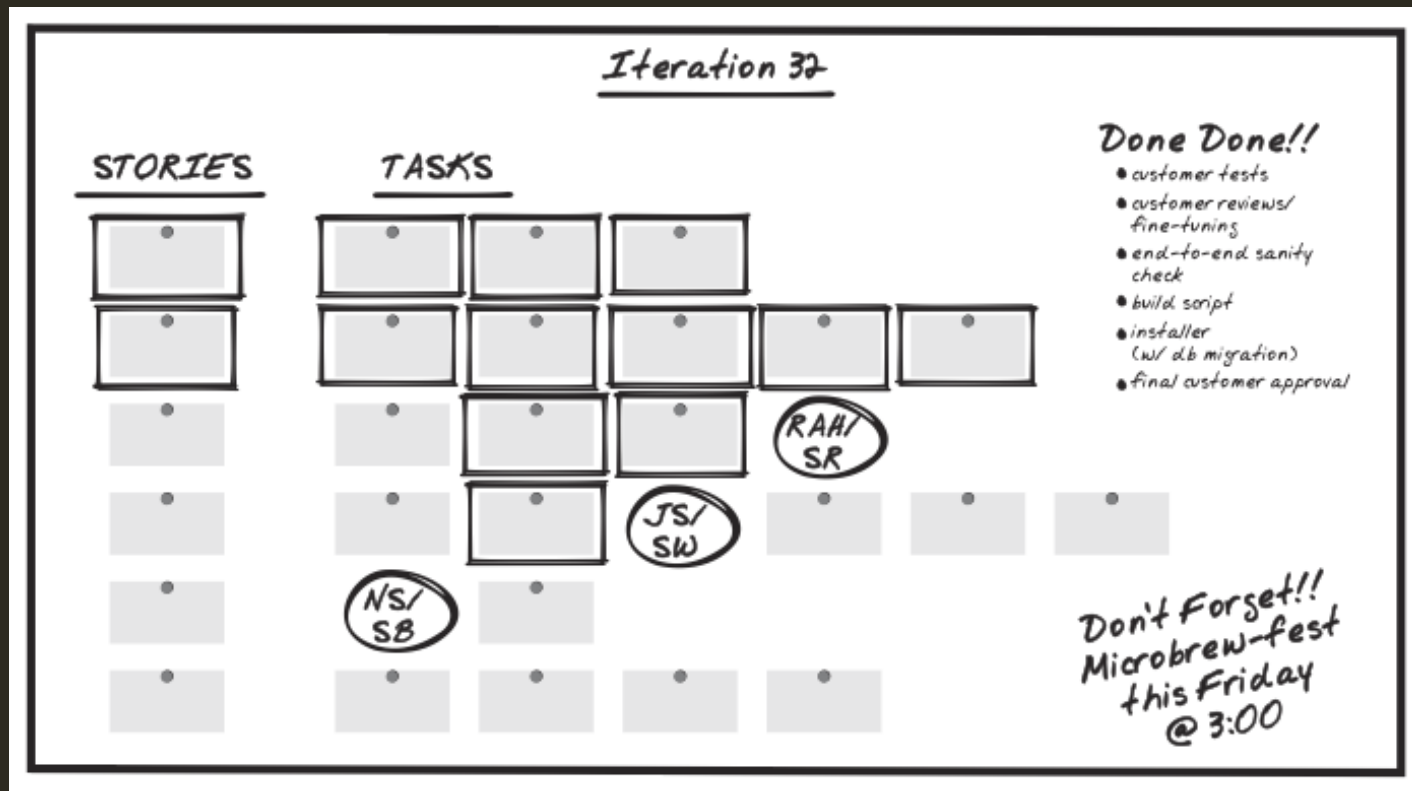
A stack-ranked list of stories to work on for the iteration

The tasks and their detailed estimates

A commitment by the team

- Let's discuss...

BIG VISIBLE CHART



The Art of Agile Development – James Shore

GOOD PRACTICES

Avoid using the iteration planning meeting for extensive release planning.

Don't start iterations on Monday

Don't finish iterations on Friday

Keep Iterations short, but not too short

- Ideal 1-2 weeks
- 30 Days – too long
- Monthly – Terrible idea
 - Months vary in length

Keep tasks less than 1 days of activity

EXERCISE: ITERATION PLANNING

Goal: Have the development team complete an iteration plan for the following stack-ranked backlog

1. Read and understand a detailed, 10-page article of agile software development in IEEE Magazine
2. Upgrade your current version of Microsoft Office to Microsoft Office 2013
3. Create a 60-minute presentation about the Speculate phase of the Agile Delivery Framework
4. Change the hard drive on your computer
5. Change the brakes on your car

Setup: 4-5 people

Rules:

- No tasks greater than 8 hours
- Assume that you have to break-down all the stories into tasks
- Summarize the estimates to indicate the estimated effort needed to complete the first feasible deployment
- Create a big visible chart to track progress

Duration: 30 Minutes

PAIR PROGRAMMING

Work Together

PAIR PROGRAMMING — A DEMONSTRATION

Origami (折り紙, from ori meaning "folding", and kami meaning "paper" (kami changes to gami due to rendaku) is the traditional Japanese art of paper folding, which started in the 17th century AD at the latest and was popularized outside of Japan in the mid-1900s.

The number of basic origami folds is small, but they can be combined in a variety of ways to make intricate designs.

EXERCISE: ORIGAMI — BY YOURSELF

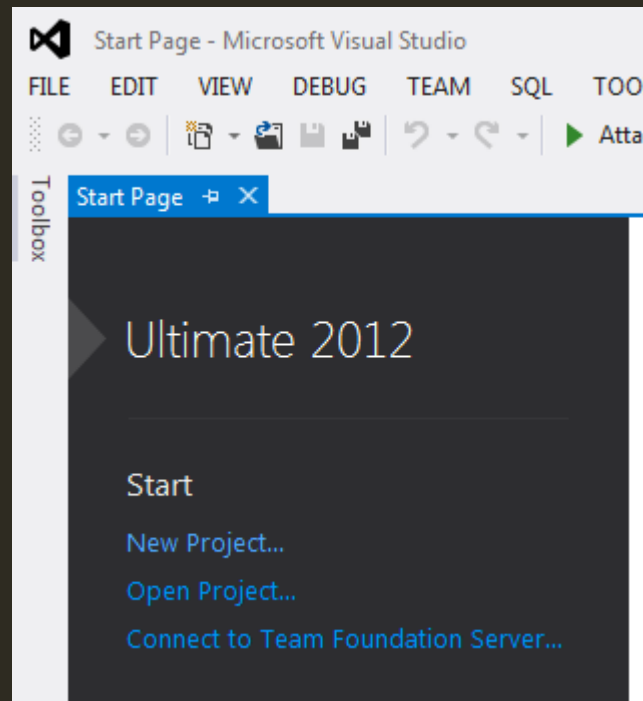
ORIGAMI - PAIRING

DEVELOPMENT TOOLS

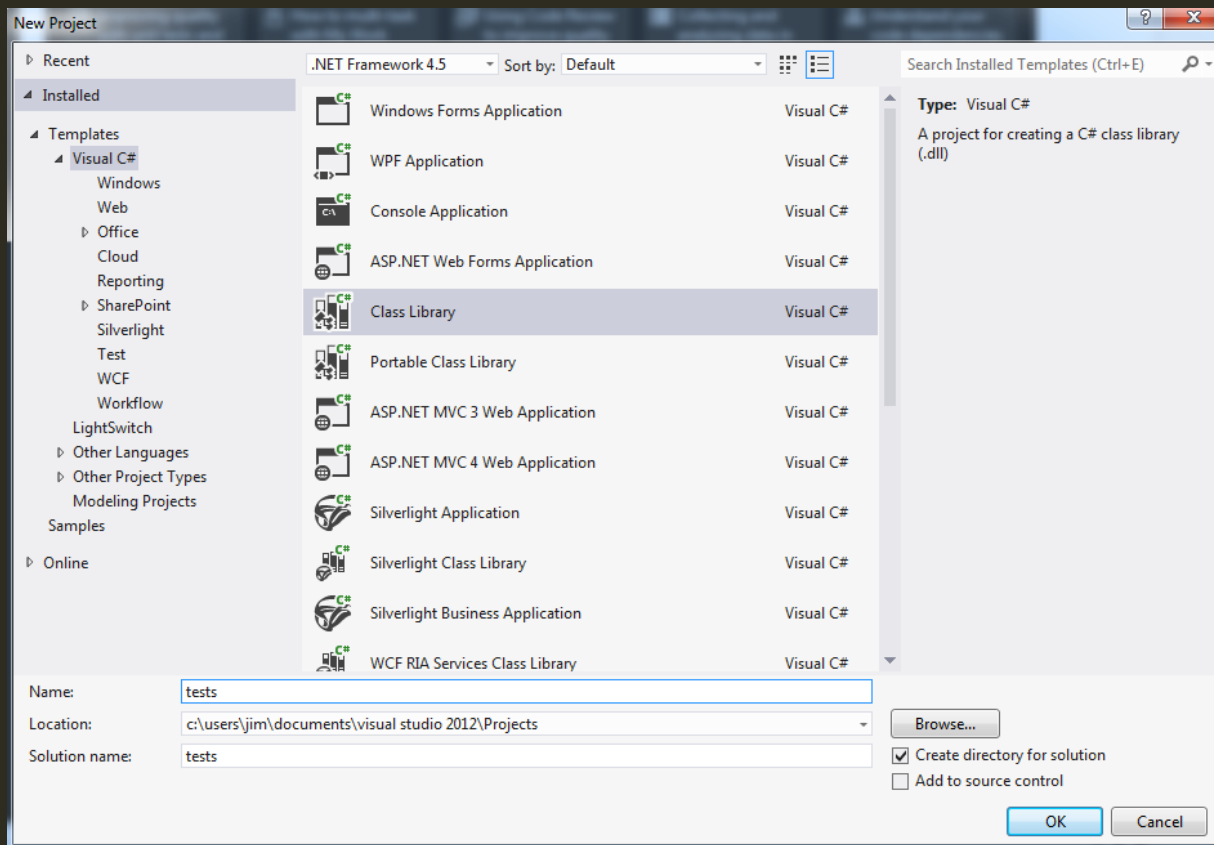
Visual Studio 2012 Trial

- <http://www.microsoft.com/visualstudio/eng/downloads>
- Download and install Visual Studio 2012 Ultimate

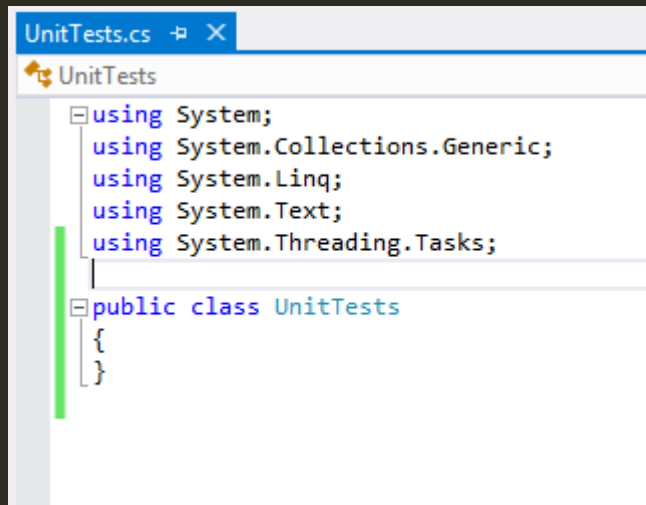
CLICK NEW PROJECT...



NEW PROJECT - CLASS LIBRARY — NAME IT TESTS



RENAME CLASS.CS TO UNITTESTS.CS

A screenshot of a code editor window titled 'UnitTests.cs'. The editor shows the following C# code:

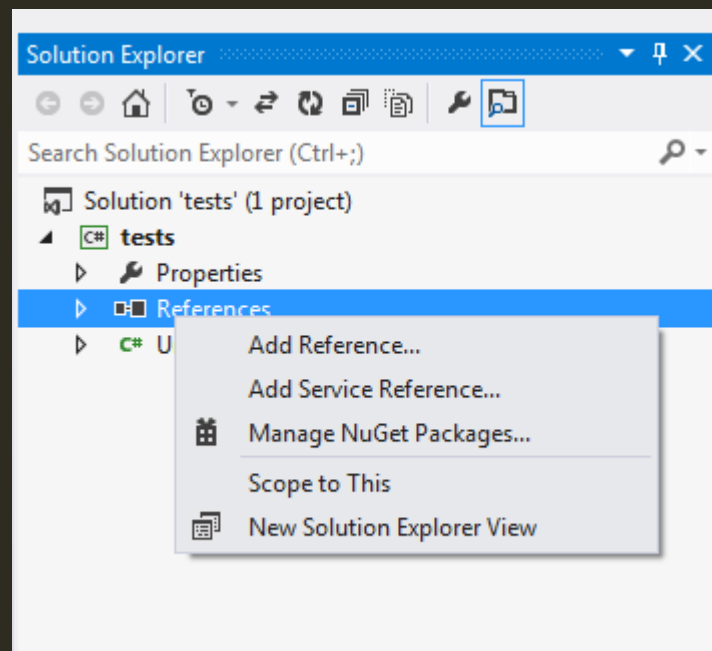
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

public class UnitTests
{
}
```

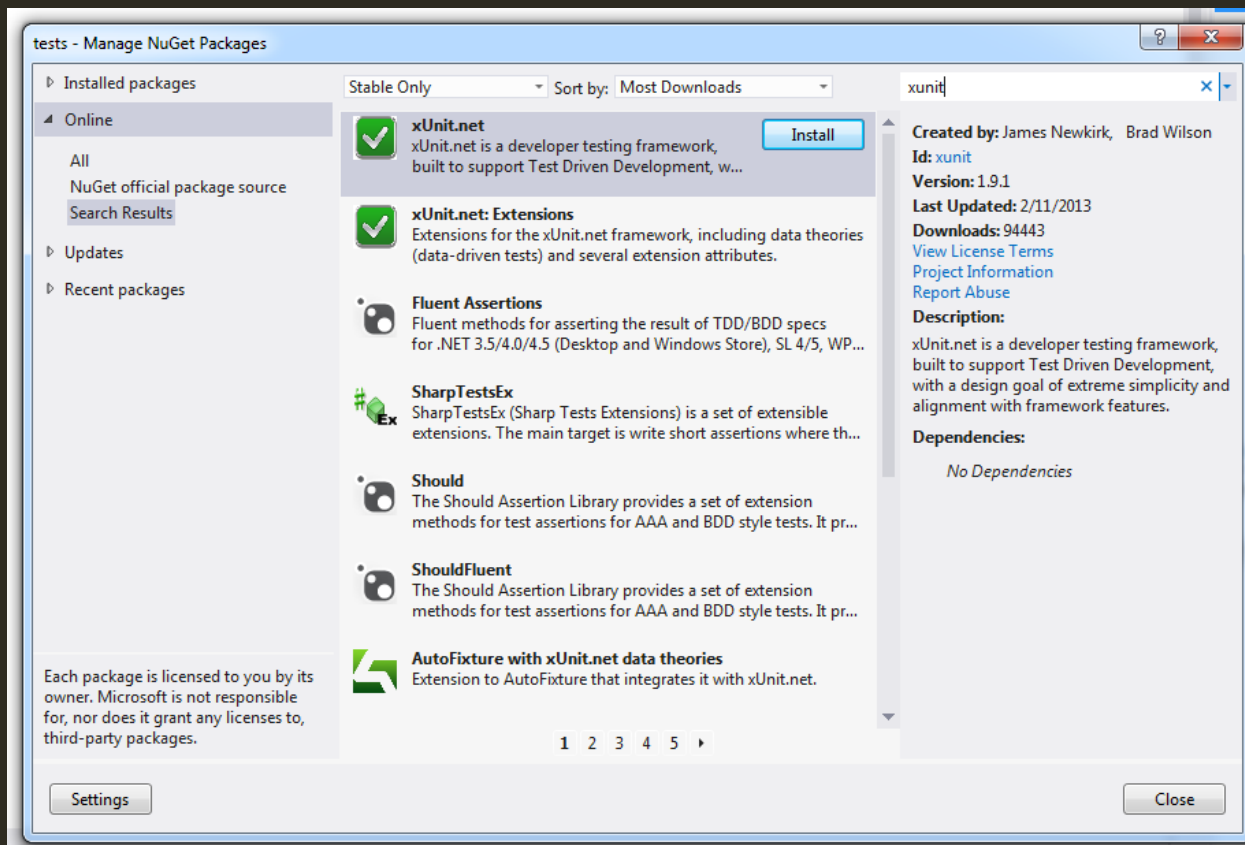
The code is formatted with syntax highlighting: 'using' is blue, 'System' and its namespaces are black, 'public class' is blue, 'UnitTests' is black, and curly braces are black. A green vertical line is visible on the left side of the editor, indicating the current line or a selection.

Change the code to look exactly like the above code

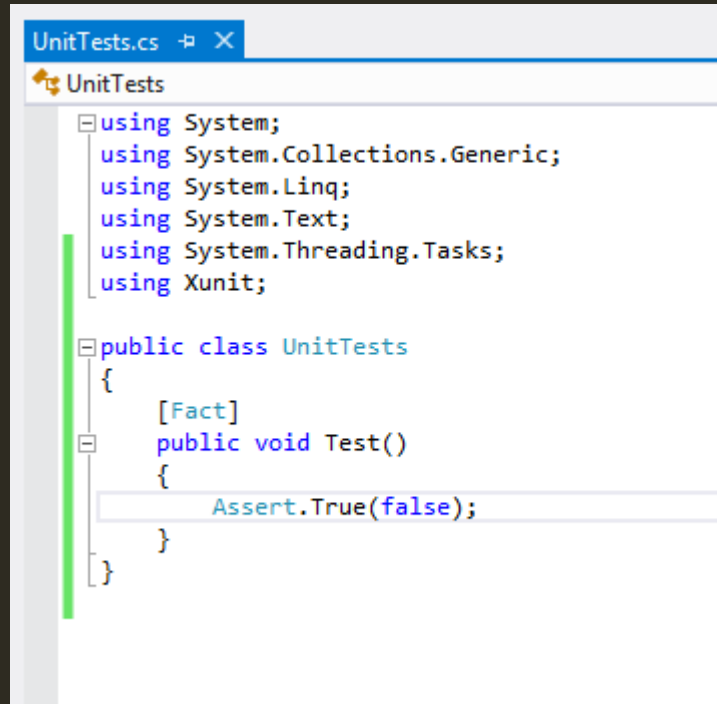
ADD A REFERENCE USING NUGET



INSTALL XUNIT.NET AND EXTENSIONS



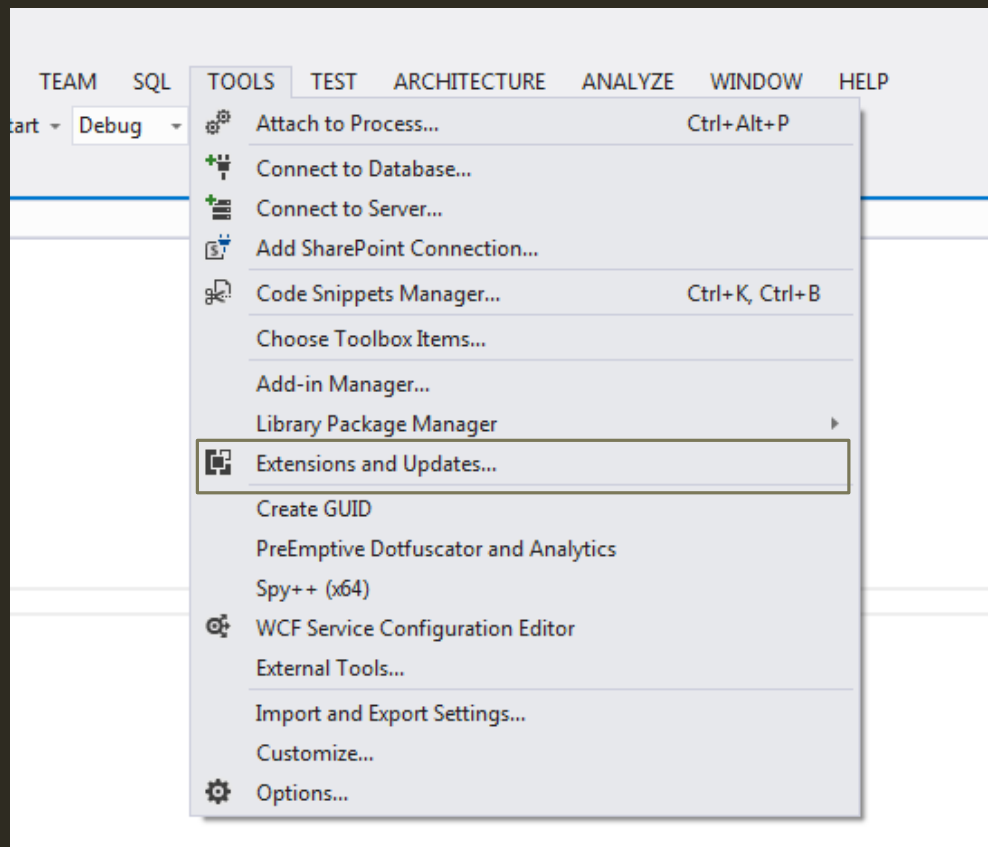
WRITE YOUR FIRST XUNIT.NET TEST AND COMPILE



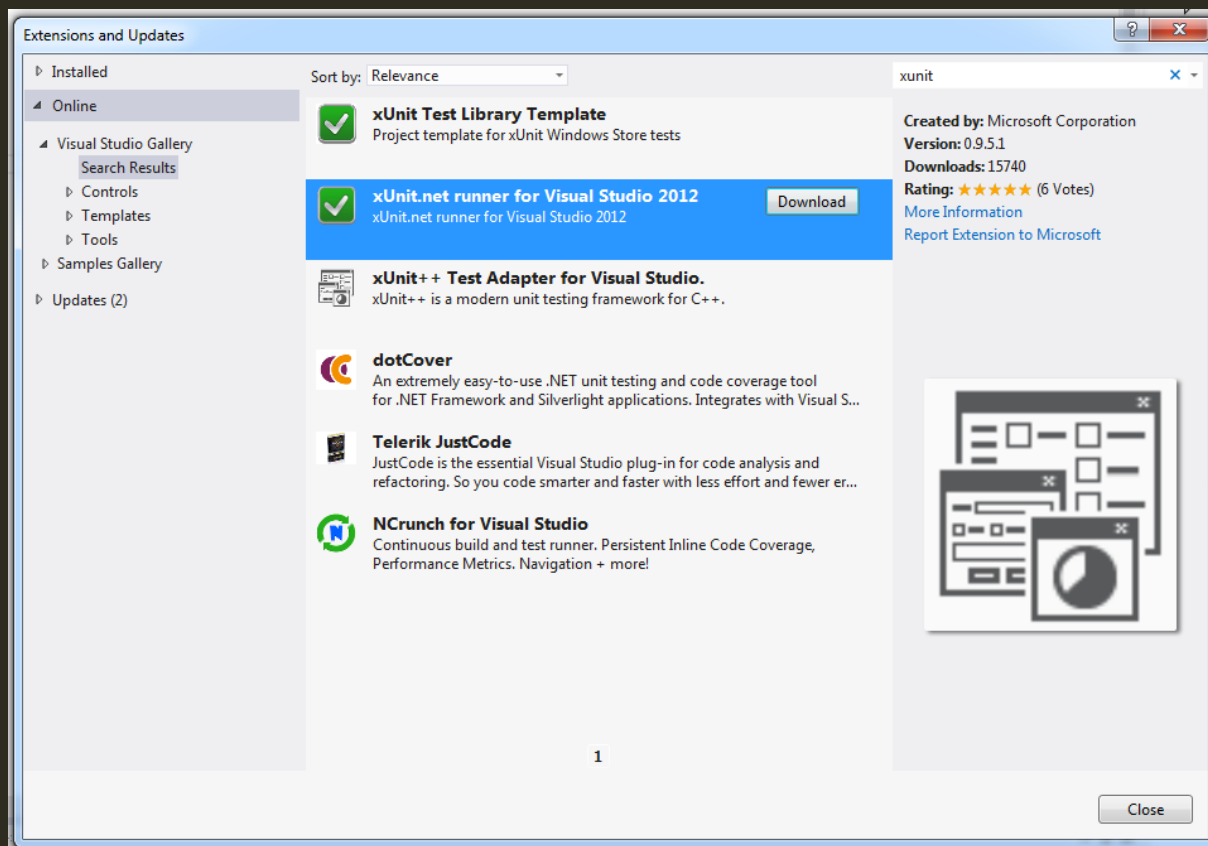
```
UnitTests.cs  ▢ ✕
UnitTests
- using System;
- using System.Collections.Generic;
- using System.Linq;
- using System.Text;
- using System.Threading.Tasks;
- using Xunit;

public class UnitTests
{
    [Fact]
    public void Test()
    {
        Assert.True(false);
    }
}
```

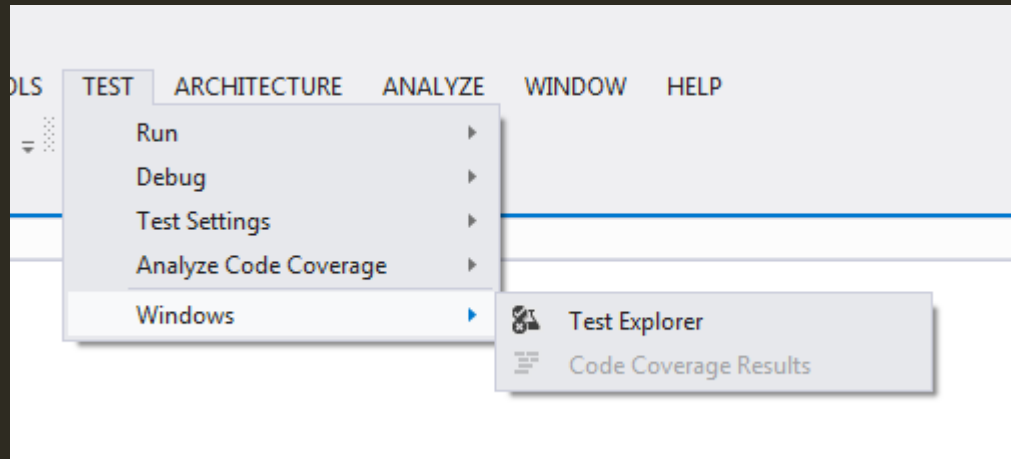
ADD THE VS2012 XUNIT.NET EXTENSION



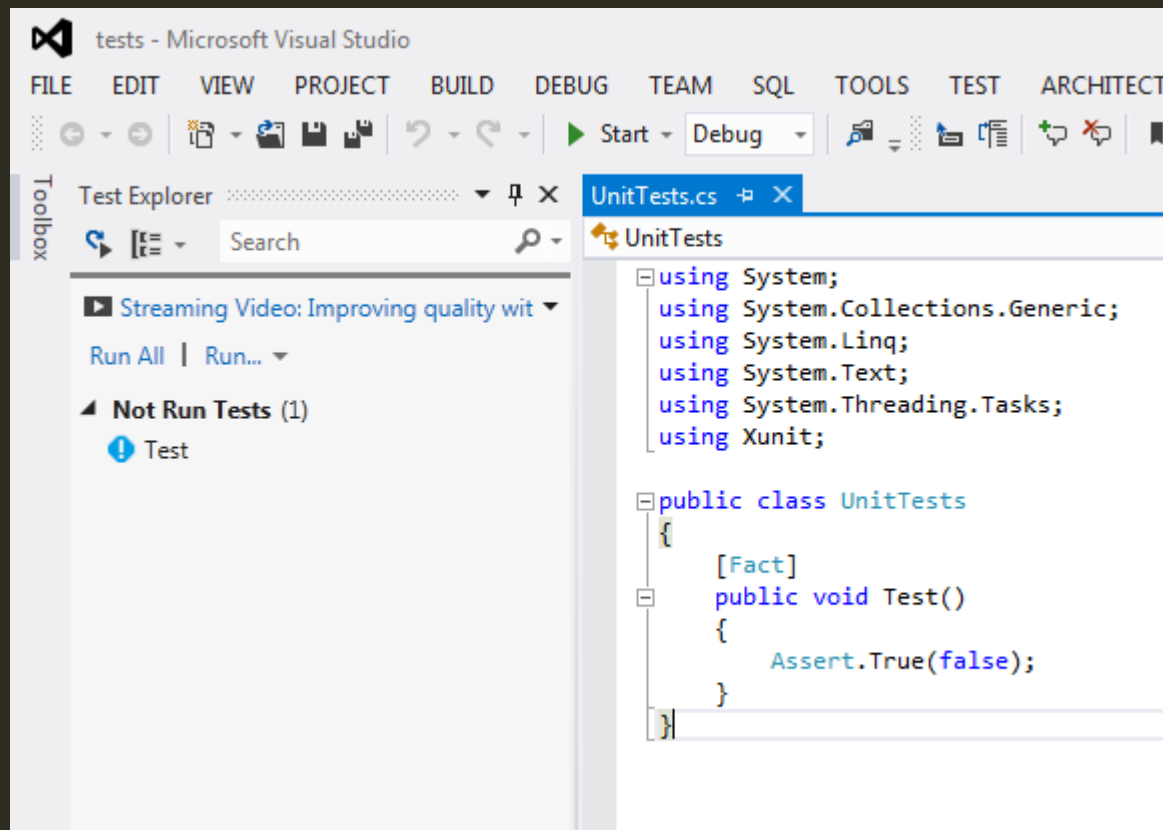
DOWNLOAD AND XUNIT.NET RUNNER



VIEW TEST EXPLORER



VIEW TEST EXPLORER



RUN THE TESTS

