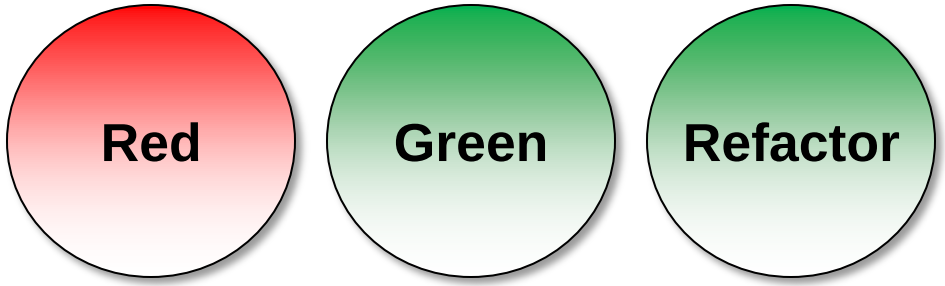
TDD with ASP.NET MVC



pluralsight
see what you can learn

Overview

- **An Overview of TDD**
  - It's not about testing …
- **Test first approach with MVC**
  - Testing controllers
  - Testing routes
- **Test doubles**
- **Refactoring tools**

TDD

- **TDD is …**
  - Creating an executable specification
  - Iterative design
  - Like using a white board with real code
  - Removing the fear of change
  - A skill acquired with practice
- **TDD is not …**
  - Just writing unit tests
  - 100 % code coverage
  - A replacement for QA and integration tests

TDD Steps

1. Write a failing test
2. Make the test pass (with the simplest possible solution)
3. Refactor and cleanup

TDD Tips

- **Keep the tests clean**
  - Readable
  - Maintainable
- **One logical assertion per test**
- **Test qualities**
  - Fast
  - Independent
  - Repeatable

TDD Oriented Tools

- **Frameworks**
  - MSTest (Visual Studio)
  - xUnit.net (http://www.codeplex.com/xunit)
  - NUnit (http://www.nunit.org/)
  - MbUnit (http://www.mbunit.com/)
- **Add-ins**
  - CodeRush Xpress (http://www.devexpress.com/crx)
  - ReSharper (http://www.jetbrains.com/resharper/)
  - TestDriven.Net (http://www.testdriven.net/)

Testing Controllers

- **What to test**
  - Did the controller return the proper ActionResult?
  - Did the controller build the proper model?
  - Did the controller produce the right side-effects (like saving data)?

```
[TestClass]
public class when_movie_controller_index_action_executes
{
    [TestMethod]
    public void it_renders_the_conventional_view()
    {
        var controller = new MovieController();

        var result = controller.Index();

        Assert.AreEqual("", result.ViewName);
    }
}
```

Test Doubles

- **A test double replaces a hard to test dependency**
  - Avoids slow tests (network and database)
  - Avoids unreliable tests (3rd party components)
  - Can make difficult scenarios easy (time outs and failues)
- **Isolation**
- **Types of test doubles**
  - Fakes (simple implementation, like an in-memory database)
  - Stubs (hand written, provides “just enough” implementation)
  - Mocks (like stubs, but generated)
- **Mock frameworks**
  - Rhino (http://ayende.com/projects/rhino-mocks.aspx)
  - Moq (http://code.google.com/p/moq/)

Testing with Mocks

- **Hide dependencies behind an abstraction**
  - Controller doesn't know the concrete implementation
- **Inject dependency into controller**
  - Consider using an inversion of control container for DI work

```
var repository = new Mock<IMovieRepository>();
var controller = new MovieController(repository.Object);
repository.Setup(r => r.FindAll())
          .Returns(movies);

var result = controller.Index();


var model = result.ViewData.Model as IQueryable<Movie>;
Assert.IsTrue(model.SequenceEqual(movies));
```

Testing Routes

- **Tedious without help**
- **MVCContrib includes a routing test helper**
  - Built for nUnit and RhinoMocks
  - Not hard to port

```
[TestInitialize]
public void Initialize()
{
    MyApplication.RegisterRoutes(RouteTable.Routes);
}

[TestMethod]
public void request_routes_to_movie_controller_index_action()
{
    "~/Movie".ShouldMapTo<MovieController>(c => c.Index());
}
```

Summary

- **MVC Framework is designed for testing**
  - Controllers are easy to test
  - Routes are testable, too
  - For views, use an automation tool
- **TDD can take some practice**
  - Remember it's about design
  - Red, green, refactor
  - Small steps
- **Use tools and frameworks to make life easier**