Applying Test Driven Development (TDD) with Integration Tests

Steve Gordon
MICROSOFT DEVELOPER TECHNOLOGIES MVP

@stevejgordon www.stevejgordon.co.uk

Overview

Test Driven Development (TDD)

Applying TDD with integration tests

Defining external boundaries

Creating fakes of external services

Replacing services with fakes by customizing the test client

Test Driven Development

Retrospective Tests

Tests written after implementation code

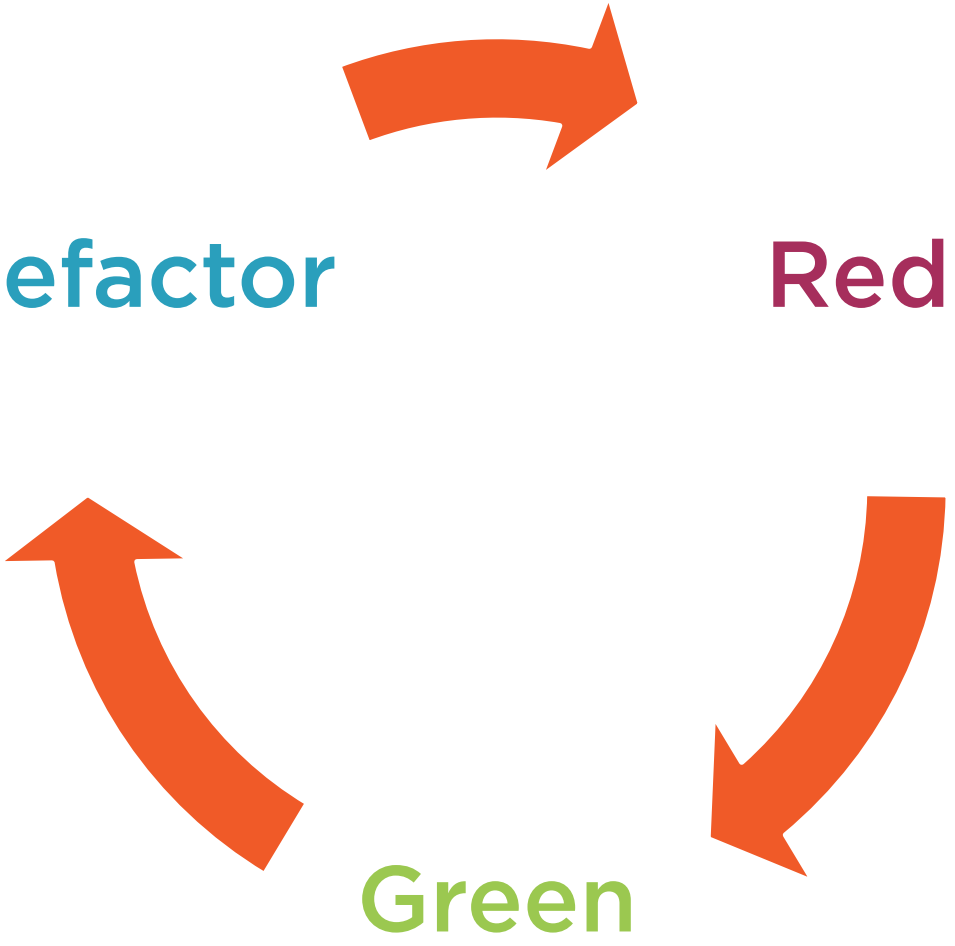
Possible to miss test cases

Possible to assert on incorrect output

Test Driven Development

- Tests come before implementation code
- Test coverage aligned to requirements
- Simplified implementation code
- Promotes refactoring to clean code

Test Driven Development Cycle



TDD Paradigm

Mental shift from retrospective testing

Requires perseverance to become productive

Test-driven Development: The Big Picture

by Jason Olson

Test-driven development is a development practice that helps runaway maintenance costs, and enables developers to build higher quality software. This course shows you why test-driven development is important and what problems it can solve.

Start Course | Bookmark | Add to Channel | Download Course

This course is part of: DevOps on Oracle Cloud: The Big Picture Path | Understanding DevOps Path

Expand All

The trademarks and trade names of third parties mentioned in this course are the property of their respective owners, and Pluralsight is not affiliated with or endorsed by these parties.

Course author

Jason Olson

Jason Olson is a software engineer passionate about distributed computing and cloud-based technology. He is a full stack developer at Concur, and formerly a Technical Evangelist and Program Manager...

Course info

Level	Beginner
Rating	★★★★½ (194)
My rating	★★★★★
Duration	1h 50m
Released	14 Apr 2017

Share course

Demo

Apply test driven development

- Write a failing test (red)
- Make the test pass (green)
- Refactor the code

Add GET endpoint /api/stock/total

Return JSON (application/json) response

Total is count of all stock of all products

```
{
  "stockItemTotal": 100
}
```

Demo

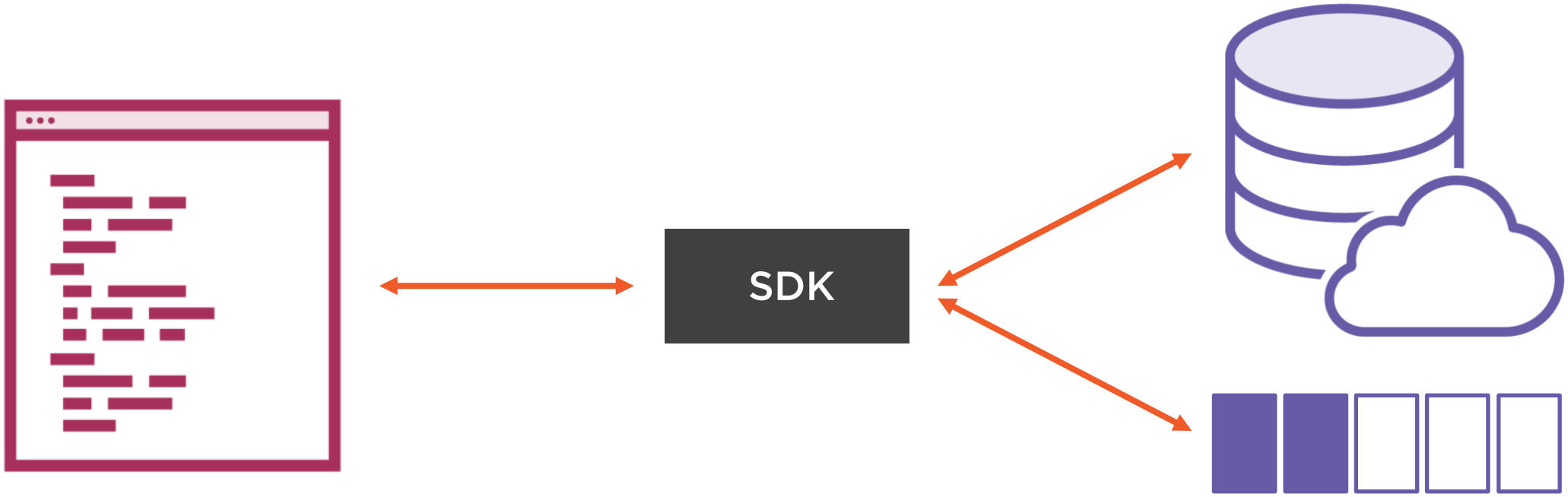
Continue development with TDD

- Test and implement a final requirement

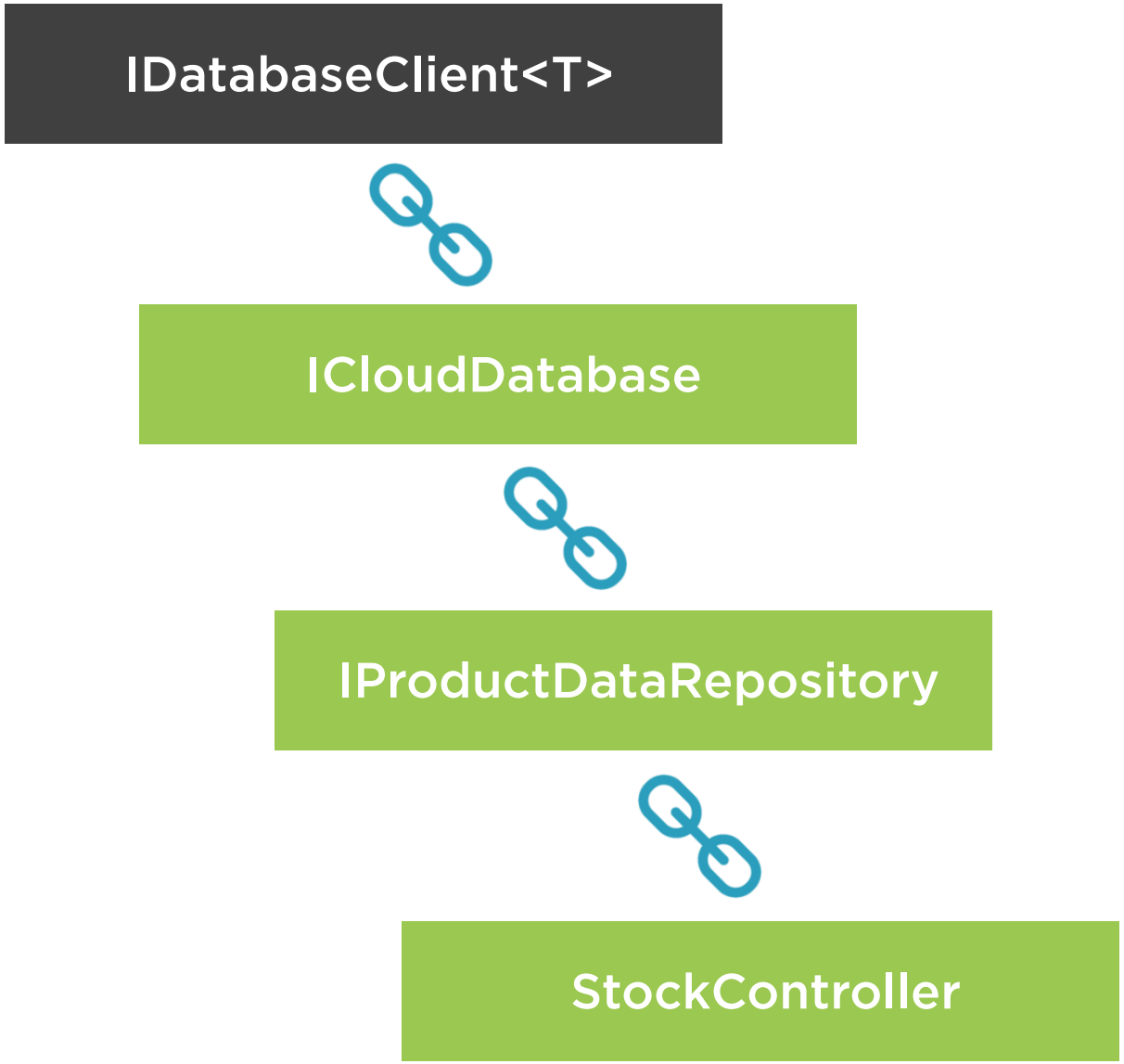
Define a boundary for integration tests

Create a fake of a dependency

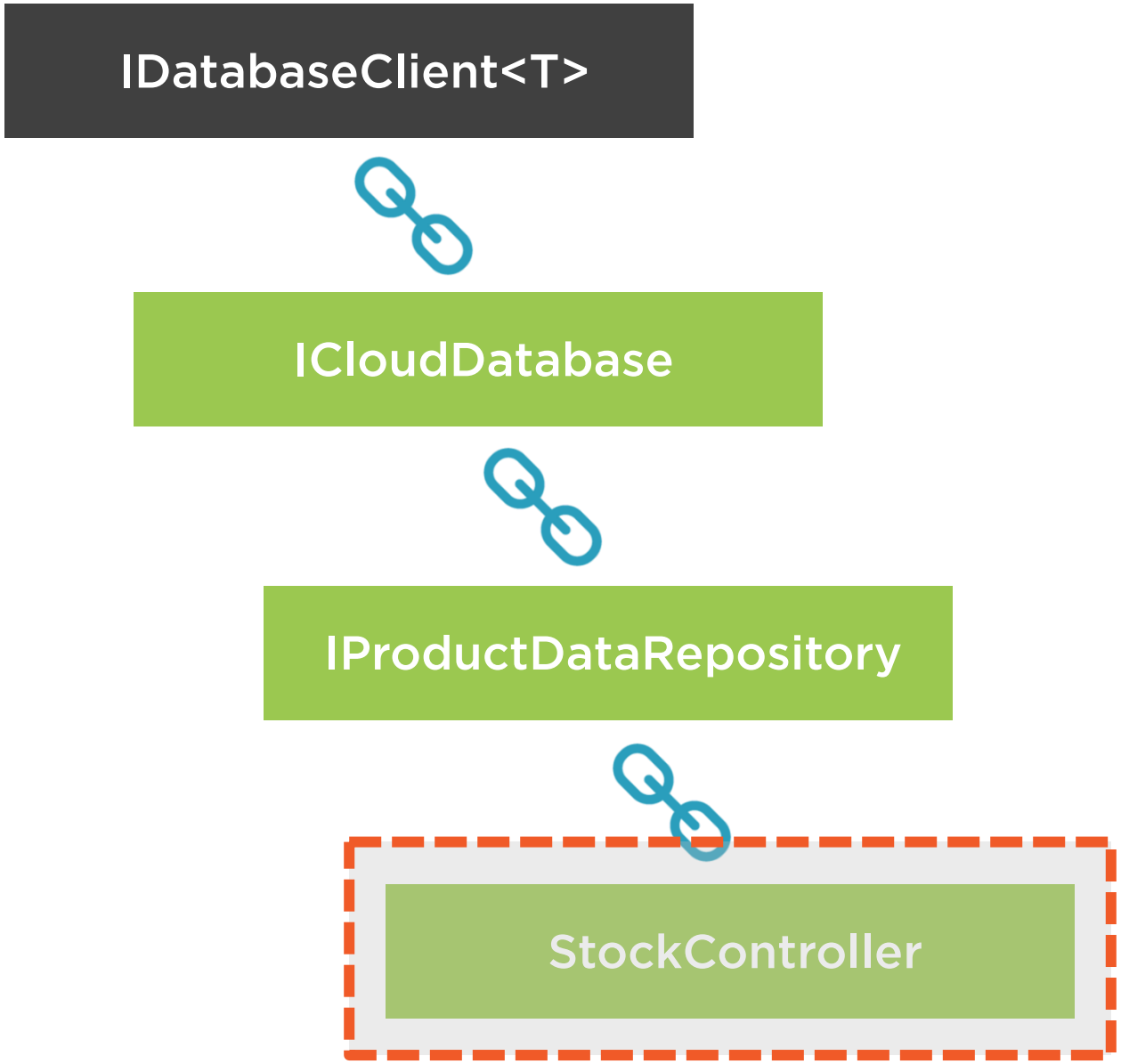
High-level Architecture



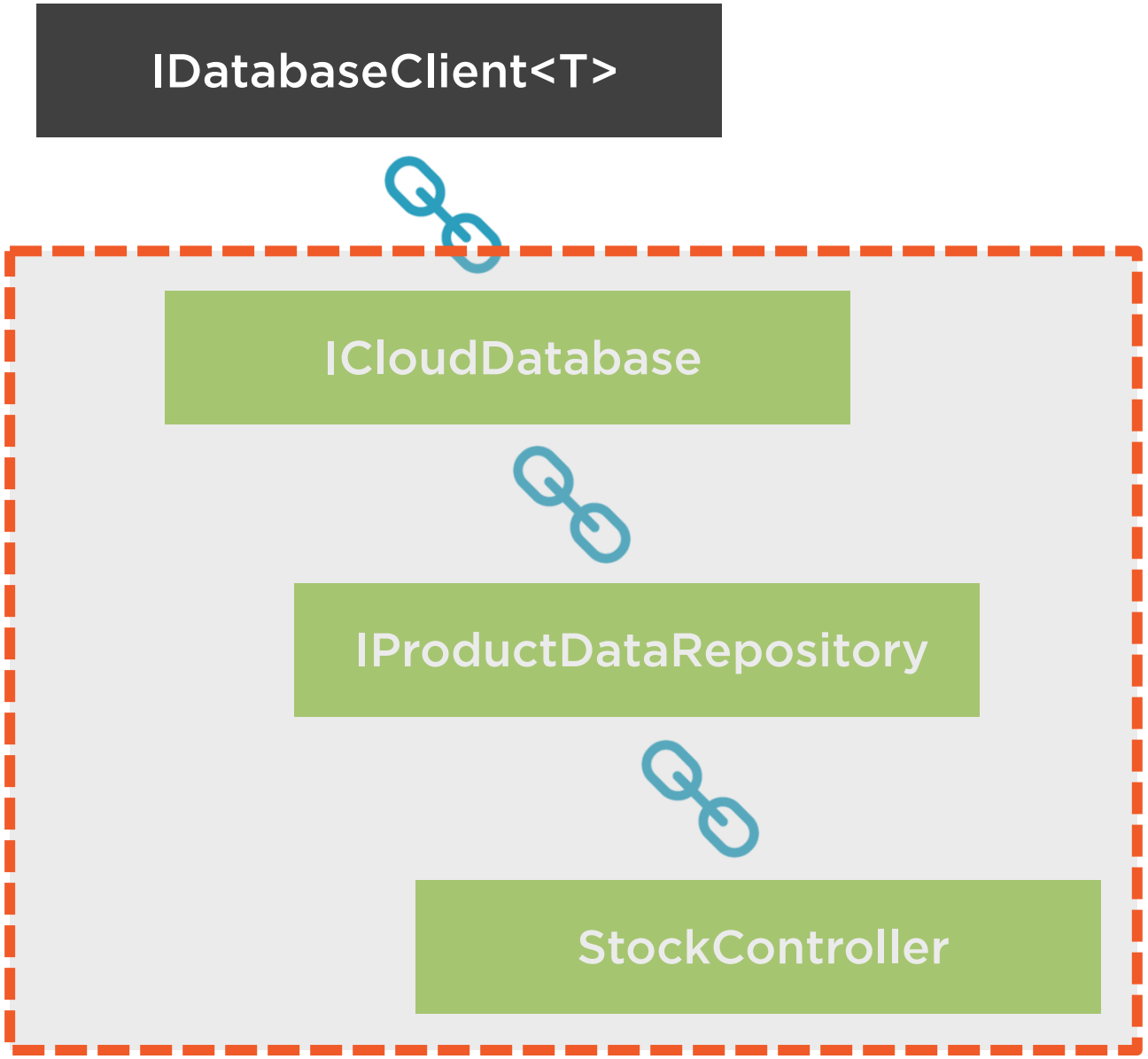
Code Dependency Chain



Code Dependency Chain



Code Dependency Chain



Advantages

Integration tests run fully in memory

Can be run often during development

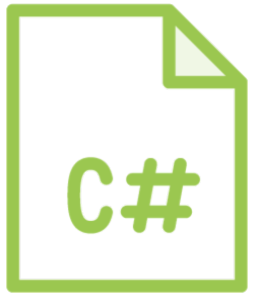
Easy to run during continuous integration

Demo

Customize the test client using WithWebHostBuilder

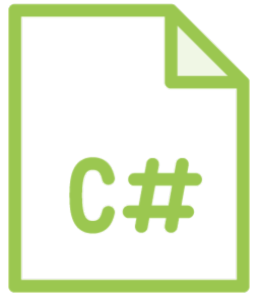
- Register a fake service

Service Registration Order



ConfigureServices

Startup.cs



ConfigureTestServices

StockControllerTests.cs

IServiceCollection
ICloudDatabase -> CloudDatabase
ISomeOtherService -> SomeOtherService
ICloudDatabase -> FakeCloudDatabase

Managing Registered Services

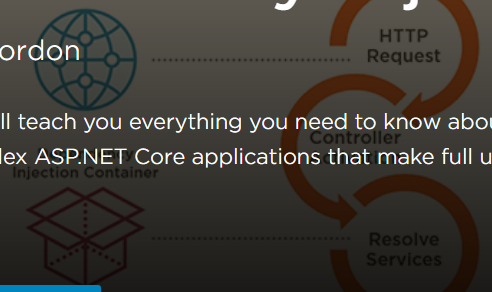
```
var client = _factory.WithWebHostBuilder(builder =>
    {
        builder.ConfigureTestServices(services =>
        {
            services.RemoveAll<IFoo>();
            services.AddSingleton<IFoo,FakeFoo>();
        });
}).CreateClient();
```

ASP.NET Core and Dependency Injection

Dependency Injection in ASP.NET Core

by Steve Gordon

This course will teach you everything you need to know about using dependency injection in ASP.NET Core. The skills you will learn will help you to build complex ASP.NET Core applications that make full use of dependency injection.



Resume Course | Bookmarked | Add to Channel | Download Course

Table of contents | Description | Transcript | Exercise files | Discussion | Learning Check | Related Courses

This course is part of: ASP.NET Core Path

Expand All

The trademarks and trade names of third parties mentioned in this course are the property of their respective owners, and Pluralsight is not affiliated with or endorsed by these parties.

Course author

Steve Gordon

Steve Gordon is a Microsoft MVP, senior developer and community lead based in Brighton, UK. He works for Madgex developing and supporting their data products portfolio, built using .NET Core...

Course info

Level	Intermediate
Rating	★★★★½ (183)
My rating	★★★★★
Duration	1h 56m
Released	6 Mar 2019

Share course

Summary

Learned about Test Driven Development

Applied the TDD cycle

- Wrote a failing test
- Implemented with minimal code
- Confirmed the test passed
- Refactored where necessary

Defined external boundary for testing

Created a fake for a service

Customized the test client to use the fake

Up Next:
Writing Integration Tests for ASP.NET Core Web APIs: Part 2