How Dependency Injection Makes Unit Testing Easier

Jeremy Clark
DEVELOPER BETTERER

@jeremybytes www.jeremybytes.com

Requests from the Boss

Different data sources

Client-side cache

Unit tests

DI &
Unit Tests

Testing the presentation logic

Property injection for tests

Unit Testing

Testing pieces of functionality in isolation

```
public class PeopleViewModel
{
  IPersonReader DataReader;

  IEnumerable<Person> People...

  public void RefreshPeople()
  {
    People =
      DataReader.GetPeople();
  }

  public void ClearPeople()
  {
    People = new List<Person>();
  } ...
}
```

- When RefreshPeople is called, People should be populated
- When ClearPeople is called, People should be empty

Unit Test without DI

```
[TestMethod]
public void People_OnRefreshPeople_IsPopulated()
{
    // Arrange
    var viewModel = new PeopleViewModel();

    ...
}
```

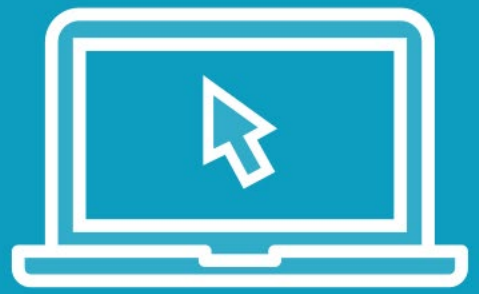
```
public PeopleViewModel()
{
    DataReader = new ServiceReader();
}
```

```
public class ServiceReader
{
  WebClient client = new WebClient();
  ...
}
```

Unit Test with DI

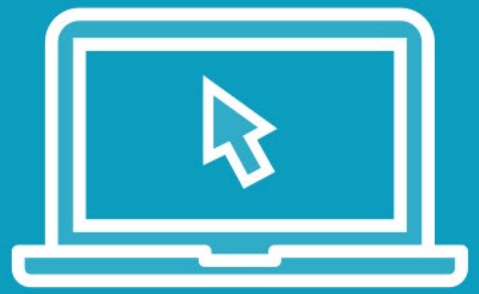
```
[TestMethod]
public void People_OnRefreshPeople_IsPopulated()
{
    // Arrange
    IPersonReader reader = GetFakeReader();
    var viewModel = new PeopleViewModel(reader);

    // Act
    viewModel.RefreshPeople();

    // Assert
    ...

}
```

Demo



Create a fake data reader

Demo



Create view model tests

Inject fake data reader with test data

Unit Testing the CSV Data Reader

```
public class CSVReader : IPersonReader
{
    public ICSVFileLoader FileLoader { get; set; }

    public CSVReader()
    {   ...
        FileLoader = new CSVFileLoader(filePath);
    }

    public IEnumerable<Person> GetPeople()
    {
        string fileData = FileLoader.LoadFile();
        IEnumerable<Person> people =
            ParseDataString(fileData);
        return people;
    } ...
}
```

CSV File Loader

```
public class CSVFileLoader : ICSVFileLoader
{
    private string _filePath;

    public CSVFileLoader(string filePath)
    {
        _filePath = filePath;
    }

    public string LoadFile()
    {
        using (var reader = new StreamReader(_filePath))
        {
            return reader.ReadToEnd();
        }
    }
}
```

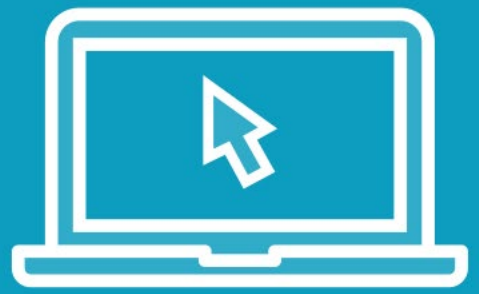
Accesses the file system

Unit Testing the CSV Data Reader

```
public class CSVReader : IPersonReader
{
    public ICSVFileLoader FileLoader { get; set; }

    public CSVReader()
    {   ...
        FileLoader = new CSVFileLoader(filePath);
    }

    public IEnumerable<Person> GetPeople()
    {
        string fileData = FileLoader.LoadFile();
        IEnumerable<Person> people =
            ParseDataString(fileData);
        return people;
    } ...
}
```

Injection point for test code

Demo



Create CSV data reader tests

Create fake file loader

Inject fake file loader with test data

Property Injection

Class property is initialized for standard behavior

By default, the standard behavior is used

Property can be set to provide alternate behavior

Property Injection

```
public class CSVReader : IPersonReader
{
    public ICSVFileLoader FileLoader { get; set; }

    public CSVReader()
    {   ...
        FileLoader = new CSVFileLoader(filePath);
    }

    public IEnumerable<Person> GetPeople()
    {
        string fileData = FileLoader.LoadFile();
        IEnumerable<Person> people =
            ParseDataString(fileData);
        return people;
    } ...
}
```

By default, uses the real file loader

Property Injection

```
[TestMethod]
public void GetPeople_WithGoodRecords_ReturnsAllRecords()
{
    var reader = new CSVReader();
    reader.FileLoader = new FakeFileLoader("Good");
```

Injection point to override default behavior for tests

```
    var result = reader.GetPeople();

    Assert.AreEqual(2, result.Count());

}
```

```
// Constructor Injection
public class PeopleViewModel
{
    protected IPersonReader DataReader;
    public PeopleViewModel(IPersonReader dataReader)
    {
        DataReader = dataReader;
    } ...
}
```

Required dependency

```
// Property Injection
public class CSVReader : IPersonReader
{
    public ICSVFileLoader FileLoader { get; set; }
    public CSVReader()
    {   ...
        FileLoader = new CSVFileLoader(filePath);
    } ...
}
```

Optional dependency

DI &
Unit Tests

Testing the presentation logic

Property injection for tests