

Expanding the Product Importer



Henry Been

Independent DevOps & Azure Architect

@henry_been www.henrybeen.nl



Overview



Organizing DI registrations

Specific situations

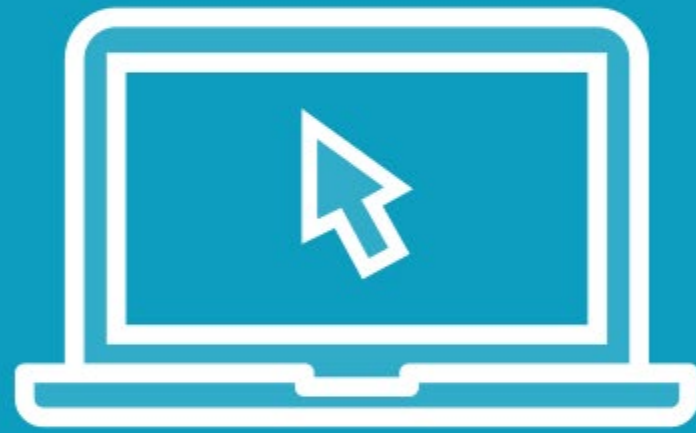
- Application configuration
- HttpClient

Service Locator pattern

DI and testability



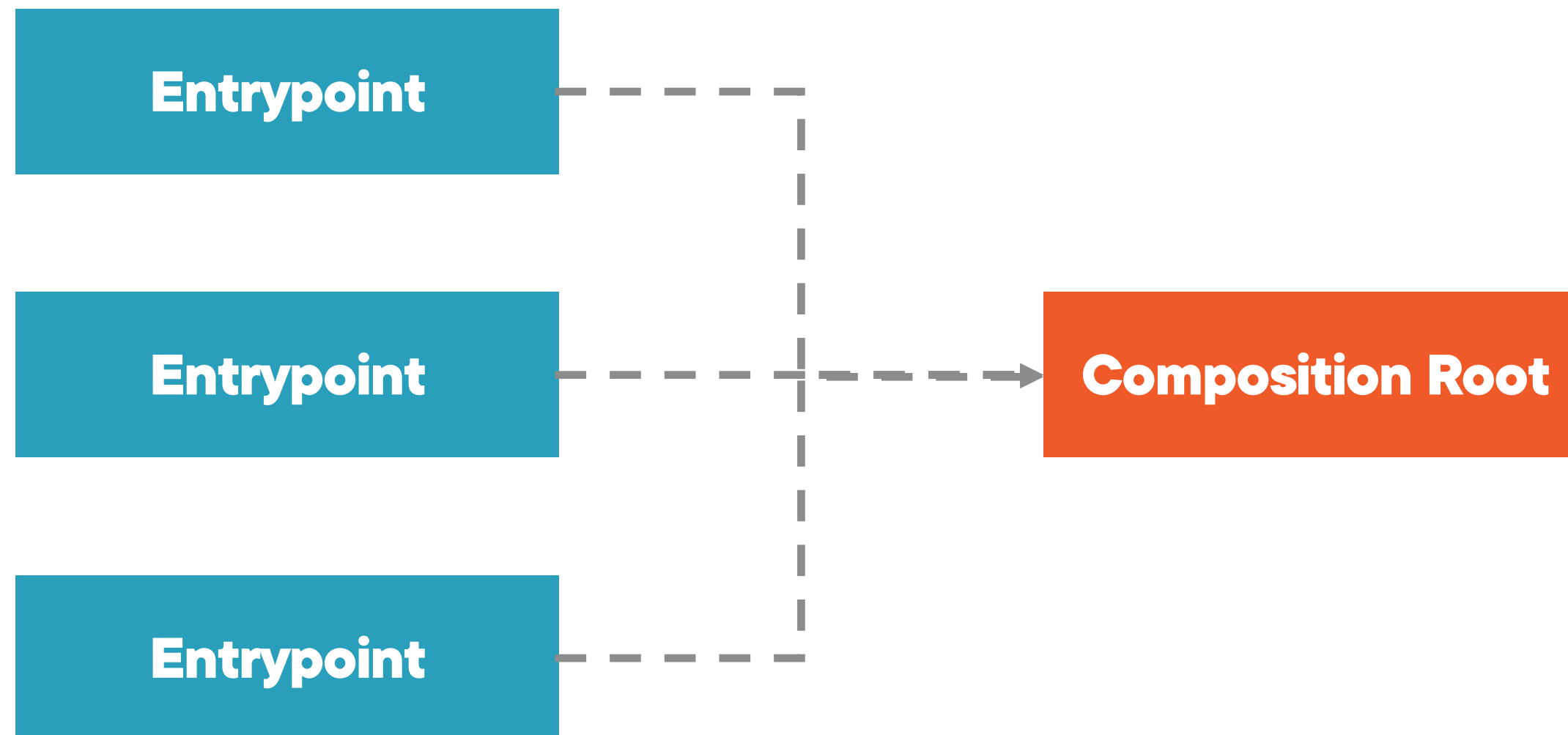
Demo



Introducing a composition root



Composition Root and Entrypoints



Add vs. TryAdd

Add{Lifetime}

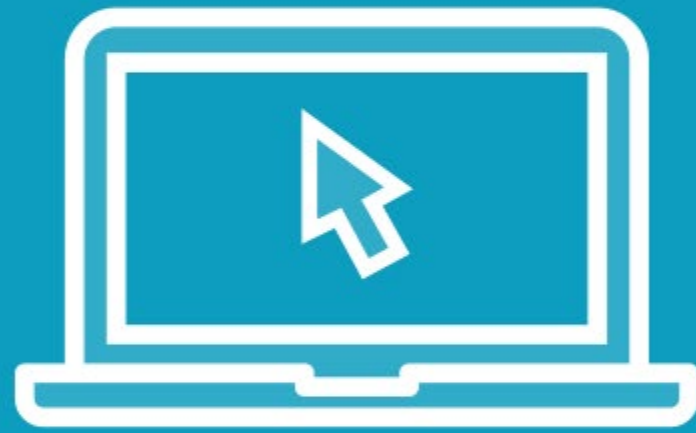
If there is an existing registration for the type, this call will overwrite it

TryAdd{Lifetime}

If there is an existing registration for the type, this call will do nothing



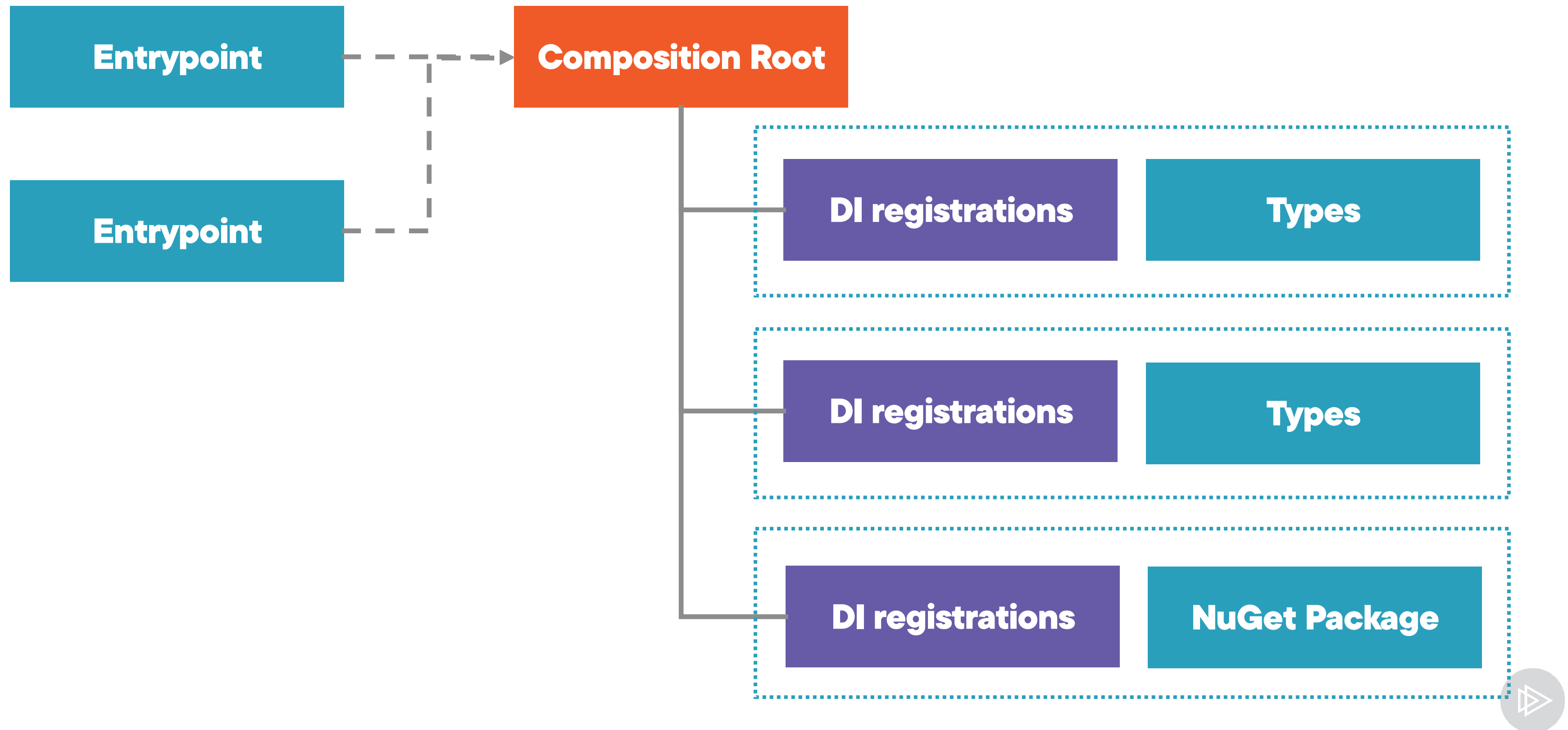
Demo



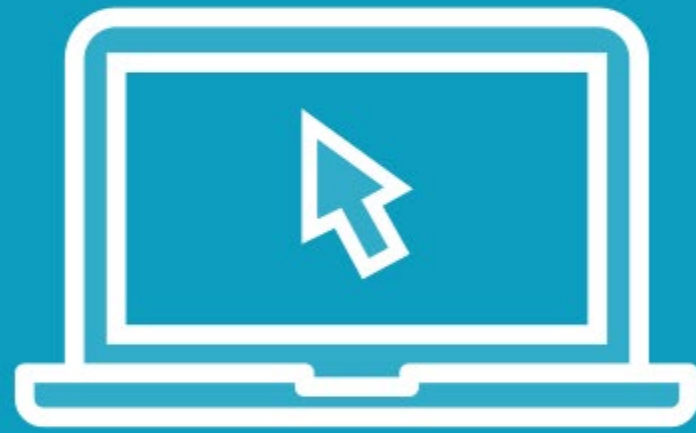
Organizing and grouping DI registrations



Grouping DI Registrations



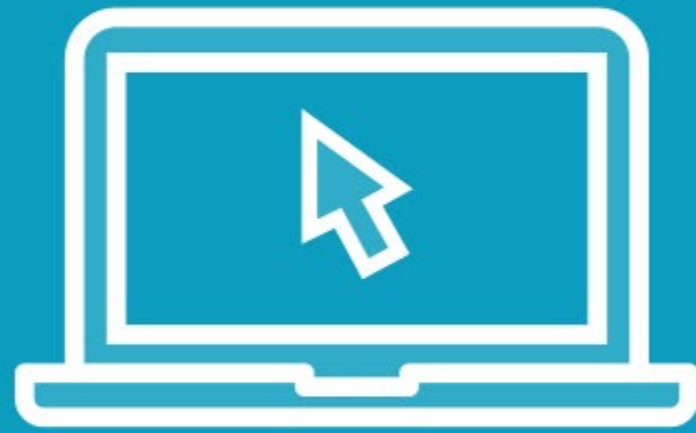
Demo



Working with application configuration
using the Options **pattern**



Demo



**Injecting multiple implementing types
under the same service type**



Multiple Registrations

Resolving directly

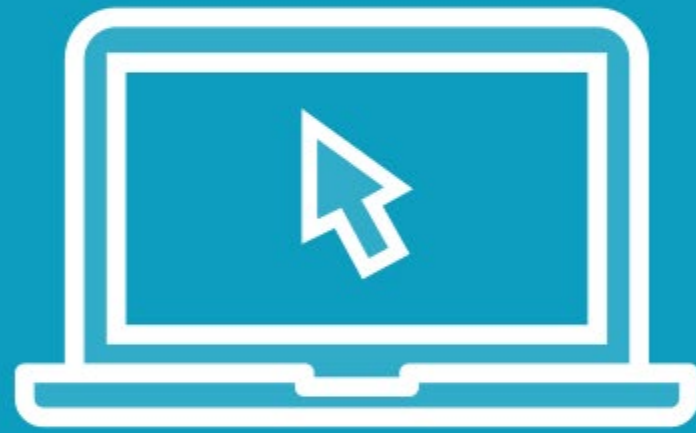
**The last registration is
returned**

Resolving IEnumerable

All registrations are returned



Demo



Injecting HttpClient from the DI container



Service Locator Pattern



Dependency Injection vs. Service Locator

Dependency Injection

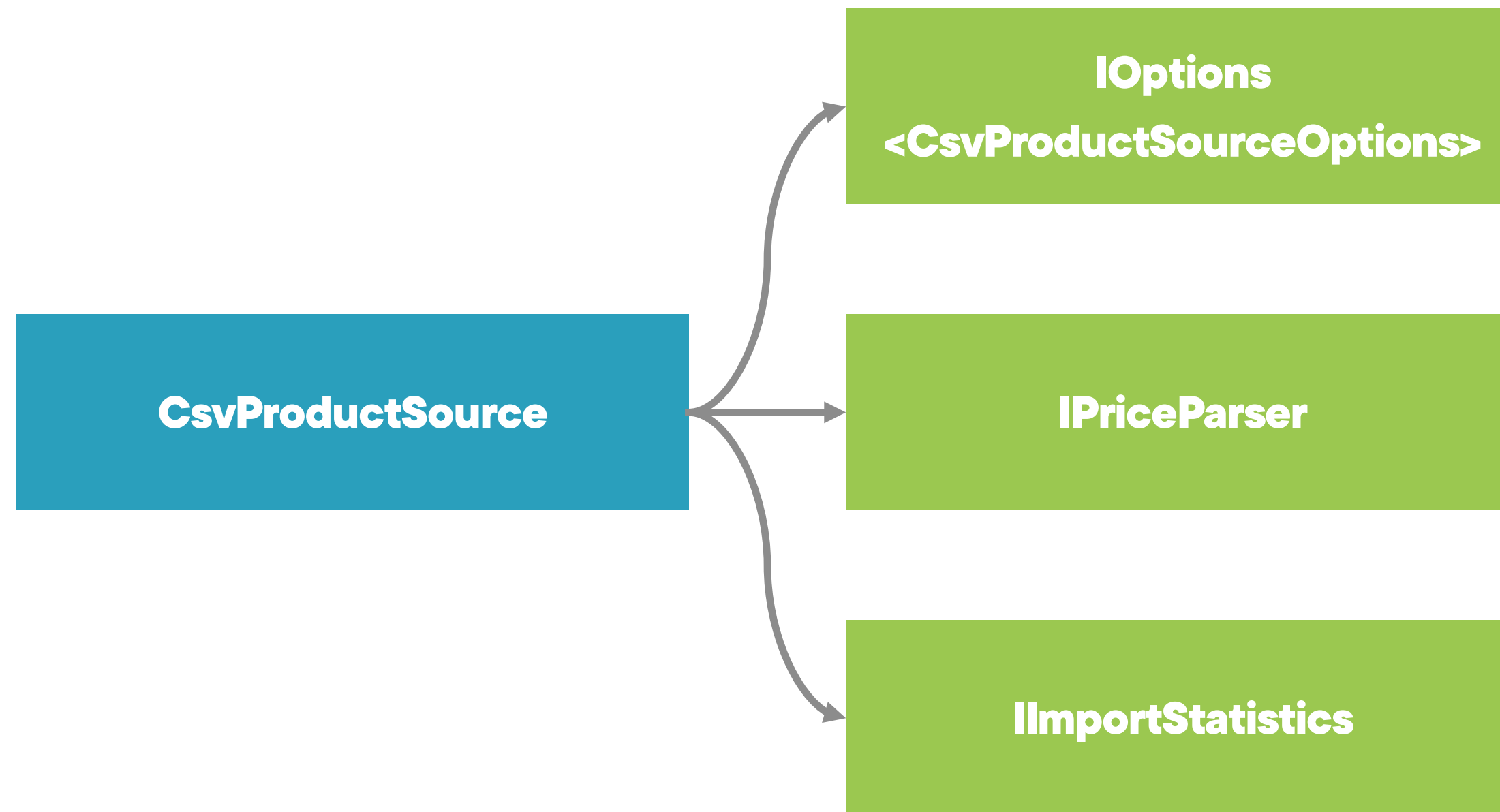
A class declares its dependencies, and they get provided somehow. A DI container is one way.

Service Locator

A class references a central services repository and requests the services it needs from there.



Dependency Injection



```
public CsvProductSource(  
    IOptions<CsvProductSourceOptions> productSourceOptions,  
    IPriceParser priceParser,  
    IImportStatistics importStatistics)  
{  
    ...  
}
```

Dependency Injection

A class declares its dependencies, and they get provided somehow. A DI container is one way.

Service Locator




```
public CsvProductSource()  
{  
    _productSourceOptions = Locator.getService<Ioptions<CsvProductSourceOptions>>;  
    _priceParser = Locator.getService<IPriceParser>;  
    _importStatistics = Locator.getService<IImportStatistics>;  
}
```

Service Locator

A class references a central services repository and requests the services it needs from there.

```
public CsvProductSource(IServiceProvider serviceProvider)
{
    _productSourceOptions = serviceProvider.GetService<Ioptions<...>>;
    _priceParser = serviceProvider.GetService<IPriceParser>;
    _importStatistics = serviceProvider.GetService<IImportStatistics>;
}
```

Service Locator Using a DI Container

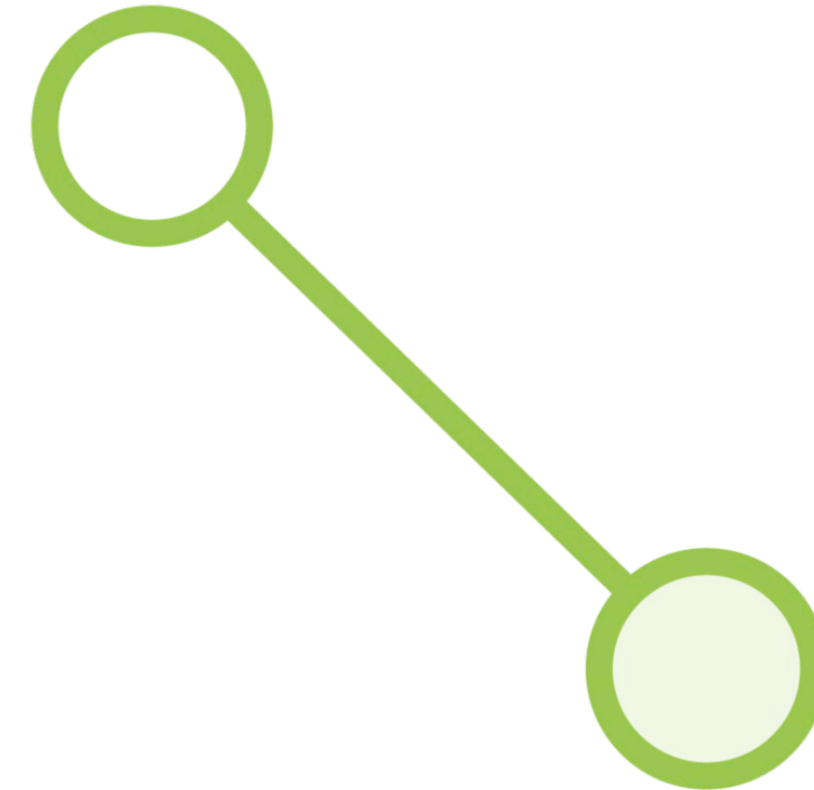
You misuse the DI container as a Service Locator.

Service Locator Is Discouraged



Testability

Classes that use a Service Locator are harder to test



Implicit dependencies

Dependencies are not clearly advertised, but implicit

The less you reference the
container, the better!



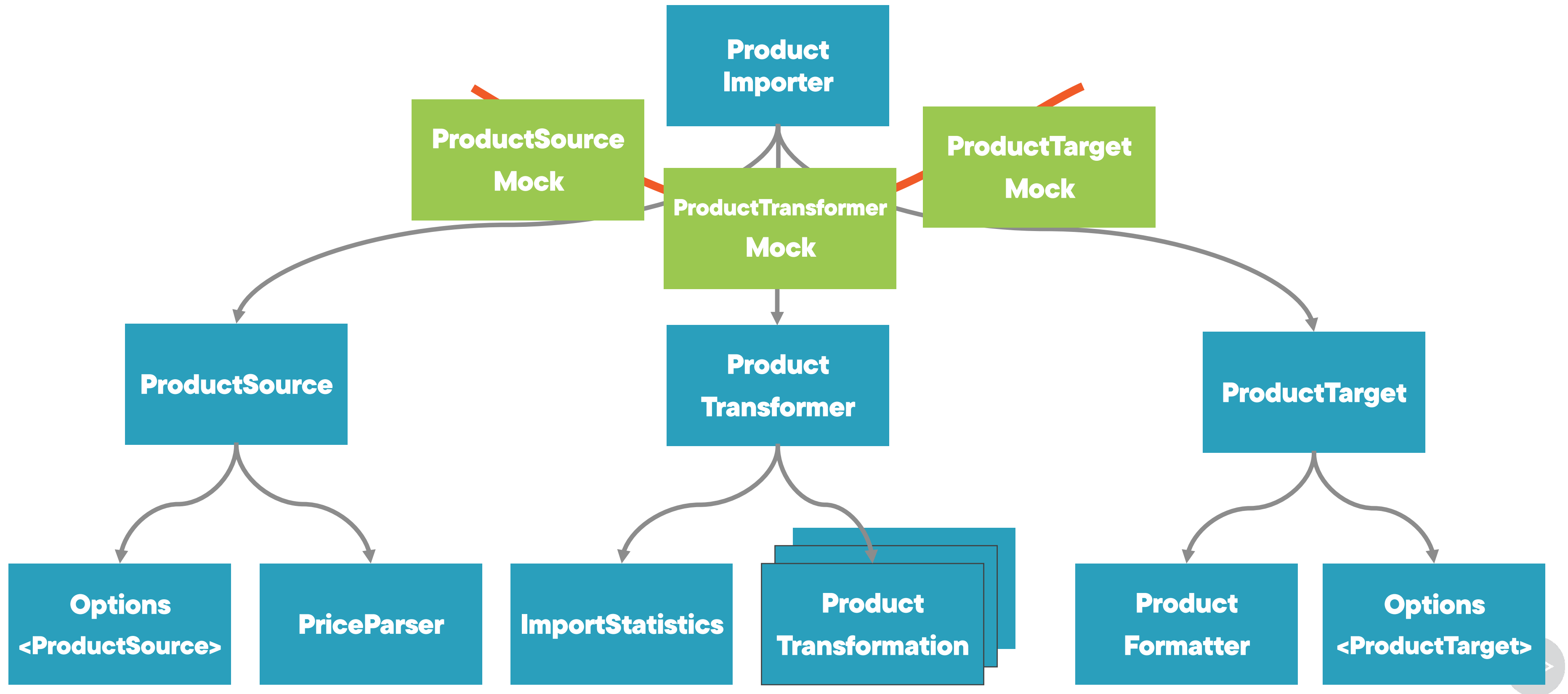
Demo



DI enables unit testing



Dependencies, Dependencies, Dependencies





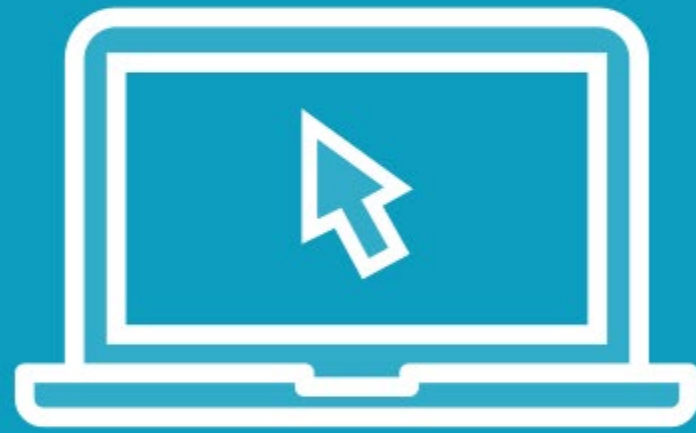
More Information

Testing .NET Code with xUnit.net 2: Getting Started

Jason Roberts



Demo



Optional dependencies



Summary



Organizing DI registrations

- Using a composition root
- Organizing and reusing DI registrations

Specific situations

- Injecting configuration using `Options<T>`
- Injecting `HttpClient`
- Injecting multiple implementations

Service Locator pattern

- Injecting the container

Dependency Injection and unit testing



Up Next:
Common Pitfalls and Challenges

