Injecting and Resolving Your Components

Miguel A. Castro
PRINCIPAL CONSULTANT

@miguelcastro67 www.melvicorp.com

Overview

Standard Injection

On-Demand Resolving

Injecting Into Views

Standard Injection

Constructor

- Components injected as constructor arguments
- Container will always be the one to resolve parent component

Property

- Components injected as public properties
- Controller would need to be registered with the trailing method "`PropertiesAutowired()`"

- In both cases, container resolves a dependency along with recursively resolving every dependency down its object graph

On-demand Resolve

Resolve As Needed

- Not all methods require all dependencies
- Resolve components when needed

"Creation" Component

- Itself is registered and injected
- Used to resolve other components
- Based on Abstract Factory pattern
- Variation of Service Locator

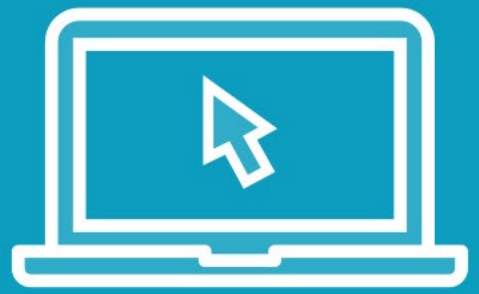
Service Locator Controversy

- Some consider it an anti-pattern
- Anti-pattern states a concrete locator embedded in your components
- This version is abstracted and registered
- It itself is injected as a dependency
- Host component remains fully testable

Injecting into Views

- View obtains **DependencyResolver**
- Resolves desired component
- Used standard Razor Syntax
- Bypasses controller-based injection
- Technique works with any DI Container
- Not specific to Autofac

Demo



Injecting into controllers using both techniques mentioned

Injecting into views

Summary

Couple of ways to inject into controllers

Container-agnostic way to inject into views