

Injecting and Resolving Dependencies



Steve Gordon

MICROSOFT DEVELOPER TECHNOLOGIES MVP

@stevejgordon www.stevejgordon.co.uk



Overview



Constructor injection

Action injection

Middleware injection

View injection

Advanced scenarios for
resolving services



Service Resolution Mechanisms



Constructor Injection



Constructor Injection



Controllers

Razor page models

ViewComponents

TagHelpers

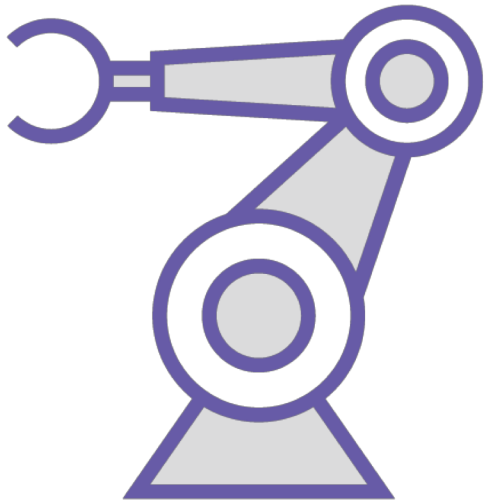
Filters

Middleware

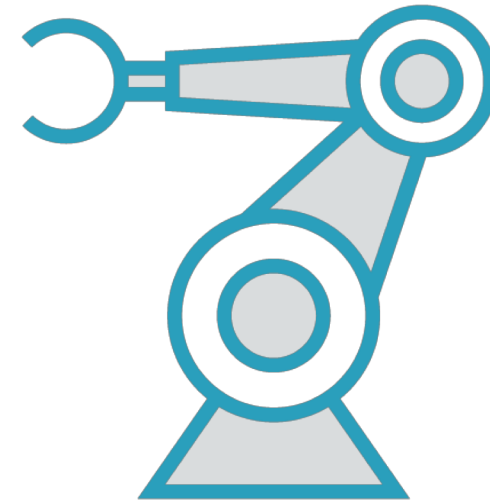
Custom classes



Service Resolution Mechanisms



IServiceProvider



ActivatorUtilities

Constructor Rules



Assign default values for arguments not provided by the container



When services are resolved, a public constructor is required



Only a single applicable constructor can exist for services resolved via ActivatorUtilities (framework components such as controllers)

IServiceProvider Constructor Selection

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<AService>();
    services.AddSingleton<AnotherService>();
}

...

public class MyService
{
    public MyService(AService aService)
    {
    }

    public MyService(AService aService, AnotherService anotherService)
    {
    }
}
```



IServiceProvider Constructor Selection

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<AService>();
    services.AddSingleton<AnotherService>();
}
```

...

```
public class MyService
{
    public MyService(AService aService)
    {
    }

    public MyService(AService aService, AnotherService anotherService)
    {
    }
}
```



IServiceProvider Constructor Selection

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<AService>();
    services.AddSingleton<AnotherService>();
}
```

...

```
public class MyService
```

```
{
    public MyService(AService aService)
    {
    }

    public MyService(AService aService, AnotherService anotherService)
    {
    }
}
```



IServiceProvider Constructor Selection

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<AService>();
    services.AddSingleton<AnotherService>();
}
```

...

```
public class MyService
{
    public MyService(AService aService)
    {
    }

    public MyService(AService aService, AnotherService anotherService)
    {
    }
}
```



IServiceProvider Constructor Selection

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<AService>();
    services.AddSingleton<AnotherService>();
}
```

...

```
public class MyService
```

```
{
    public MyService(AService aService)
    {
    }
}
```

```
public MyService(AService aService, AnotherService anotherService)
{
}
}
```



IServiceProvider Constructor Selection

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<AService>();
}
```

...

```
public class MyService
{
    public MyService(AService aService)
    {
    }

    public MyService(AService aService, AnotherService anotherService)
    {
    }
}
```



IServiceProvider Constructor Selection

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<AService>();
}
```

...

```
public class MyService
{
```

```
    public MyService(AService aService)
    {
    }
```

```
    public MyService(AService aService, AnotherService anotherService)
    {
    }
```

```
}
```



Demo



When to use action injection

Refactoring to use action injection



Demo



Injecting services into middleware

Choosing the correct point of injection
for your services



Comparing Middleware Dependency Injection

Constructor

Runs once for the lifetime of the application

Supports only singleton services

Scoped or transient services will be captured and may not behave correctly

Invoke/InvokeAsync

Runs once per request

Services are resolved from the request scope

Supports all service lifetimes



Factory-based middleware
is constructed once
per request.



Demo



Injecting services into Razor views

Populating lookup data



Creation and Disposal of Resolved Services



Demo



Manually creating a scope

Resolving services from a custom scope



Review



Rules for constructor injection

Using action injection in complex controllers

Choosing the correct point of injection for middleware dependencies

Injecting services into Razor views

Manually creating and using scopes

