ASP.NET Core's

Built-in Dependency Injection Framework

Jonathan "J." Tower

Hi, I'm J.

Jonathan "J." Tower
Principal Consultant & Partner
Trailhead Technology Partners

TRAILHEAD
TECHNOLOGY PARTNERS

trailheadtechnology.com

- Microsoft MVP in ASP.NET
- Telerik/Progress Developer Expert
- Organizer of Beer City Code

- jtower@trailheadtechnology.com
- trailheadtechnology.com/blog
- jtowermi

Summary

- Quick Recap
  - Inversion of Control (IoC)
  - Dependency Inject (DI)
  - Benefits of DI
- DI in ASP.NET
  - Lifetime management
  - Startup Configuration
  - Using DI in Middleware, Controllers, and Views
- Real-world example in ASP.NET Core
- Using Third-Party DI Frameworks

Quick Recap

What are DI and IoC?

What is Dependency Injection?

It’s a way to do Inversion of Control (IoC)

OK, So What's Inversion of Control?

I'm glad you asked…

IoC By Example

```
public class TextEditor
{
    private SpellChecker _checker;
    public TextEditor()
    {
        _checker = new SpellChecker();
    }
}

// calling code
TextEditor textEditor = new TextEditor();
```

```
public class TextEditor
{
    private ISpellChecker _checker;
    public TextEditor(ISpellChecker checker)
    {
        _checker = checker;
    }
}

// calling code
SpellChecker sc = new SpellChecker();
TextEditor textEditor = new TextEditor(sc);
```

IoC By Example

```
public class TextEditor
{
    private SpellChecker _checker;
    public TextEditor()
    {
        _checker = new SpellChecker();
    }
}

// calling code
TextEditor textEditor = new TextEditor();
```

```
public class TextEditor
{
    private ISpellChecker _checker;
    public TextEditor(ISpellChecker checker)
    {
        _checker = checker;
    }
}

// calling code
SpellChecker sc = new SpellChecker();
TextEditor textEditor = new TextEditor(sc);
```

What's the Big Deal?

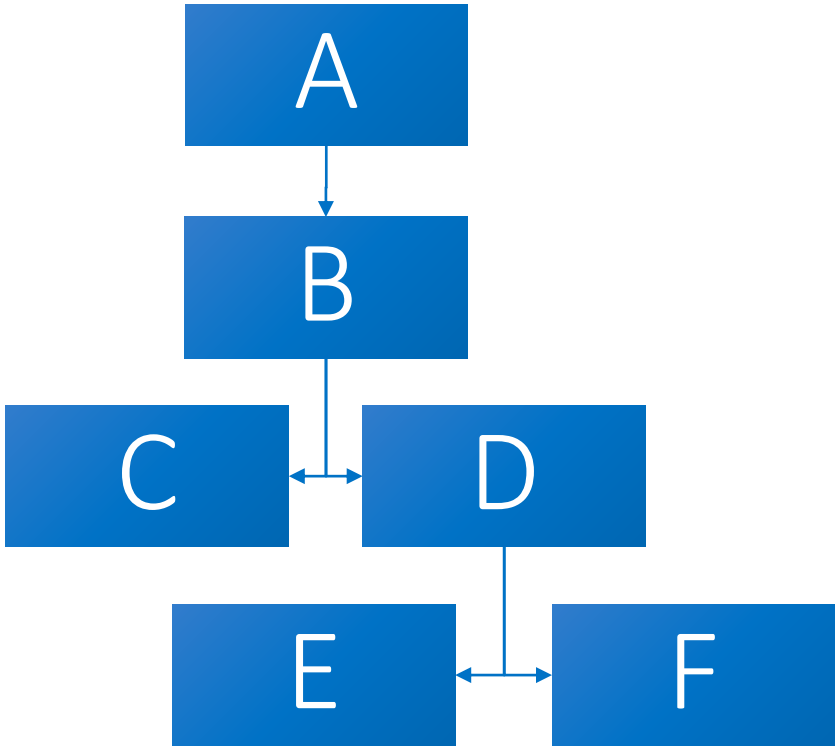
"New Is Glue" - Steve Smith (aka @ardalis)

Your classes should depend on the services they need, not specific implementations of those services

Dependency on implementations = hard-coded

In OO, service contracts = interfaces

Isn't This Going to Get Messy?




```
// this isn't cool
var f = new F();
var e = new E();
var d = new D(e, f);
var c = new C();
var b = new B(c, d);
var a = new A(b);

// this isn't much better
var a = new A(new B(new C(), new D(new E(), new F())));

// phew, finally doing something
a.DoSomething();
```

Dependency Injection Saves the Day

DI is a way of doing IoC

One object supplies the dependencies of another object

Usually done for you by a Framework

Terminology:

- “Service” - the dependency
- “Client” - the dependent class
- “Injector” – where all the magic happens

DI by Example

```
// service contract
public interface INowLogger {
  void Log();
}

// implementation
public class ConsoleNowLogger : INowLogger {
  public void Log() {
    Console.Log(DateTime.Now);
  }
}
```

```
// some sort of initialization
DILib.Map<INowLogger>().To<ConsoleNowLogger>();

// using it somewhere
public class SomeClass {
  private INowLogger _logger;

  public SomeClass(INowLogger logger) {
    _logger = logger;
  }
}
```

Benefits of Dependency Injection

Lowers coupling between modules

SOLID design principles

- **S** ingle Responsibility Principle
- **O** pen/Closed Principle
- **L** iskov Substitution Principle
- **I** nterface Segregation Principle
- **D** ependency Inversion Principle

More easily testable

Swapable modularity

DI in ASP.NET Core

Lifetime Management

1. Singleton - Only **one** instance total
2. Scoped - Single one for an **HTTP request**
3. Transient - New one **every time** you ask for it

Startup Configuration

ConfigureServices (optional, 1st)
- Add “services” to services collection (read: DI config)
- Configure all services

Configure – Configure HTTP pipeline (middleware)

Startup Configuration

```
public class StartUp
{
    public void Configure(IApplicationBuilder app) { ... }

    public void ConfigureServices(IServiceCollection services)
    {
        services.AddTransient<IEmailSender, EmailSender>();
        services.AddTransient<ISmsSender, SmsSender>();
    }
}
```

Injecting Into Controllers

```
public class HomeController : Controller
{

    public HomeController()
    {

    }

    public IActionResult Index()
    {
        // do something custom here
    }
}
```

Injecting Into Controllers

```
public class HomeController : Controller
{
    private ISomeDependency _dep;
    public HomeController(ISomeDependency dep)
    {
        _dep = dep;
    }

    public IActionResult Index()
    {
        _dep.UseItHere();
    }
}
```

Injecting Into Middleware

```
public class MyMiddleware
{
    private readonly RequestDelegate _next;

    public MyMiddleware(RequestDelegate next)
    {
        _next = next;

    }

    public async Task Invoke(HttpContext context)
    {
        // do something custom here
        await _next(context);
    }
}
```

Injecting Into Middleware

```
public class MyMiddleware
{
    private readonly RequestDelegate _next;
    private ISomeDependency _dep;
    public MyMiddleware(RequestDelegate next, ISomeDependency dep)
    {
        _next = next;
        _dep = dep;
    }

    public async Task Invoke(HttpContext context)
    {
        _dep.DoSomethingCustom();
        await _next(context);
    }
}
```

Injecting Into Views

```
@model ToDoItem

<div>
  <span>Name:          @Model.TaskName</span>
  <span>Assigned To:   @Model.AssignedTo</span>
  <span>Related Items: <!-- Related Item Count --></span>
</div>
```

Injecting Into Views

```
@model ToDoItem
@inject IToDoItemHelper Helper
<div>
  <span>Name:          @Model.TaskName</span>
  <span>Assigned To:   @Model.AssignedTo</span>
  <span>Related Items: @Helper.FindRelated(Model).Count</span>
</div>
```

Let’s Bring it All Together

DEMO

Third Party DI Libraries

Using Autofac Instead

DEMO

Using StructureMap Instead

DEMO

What About Ninject?

Ninject beta 4.0.0-beta-0134

Work-arounds do exist

github.com/dotnetjunkie/Missing-Core-DI-Extensions

DEMO

Summary

- Quick Recap
  - Inversion of Control (IoC)
  - Dependency Inject (DI)
  - Benefits of DI
- DI in ASP.NET
  - Lifetime management
  - Startup Configuration
  - Using DI in Middleware, Controllers, and Views
- Real-world example in ASP.NET Core
- Using Third-Party DI Frameworks

Thank You! Questions?

Jonathan "J." Tower
Principal Consultant & Partner
Trailhead Technology Partners

TRAILHEAD
TECHNOLOGY PARTNERS

trailheadtechnology.com

- Microsoft MVP in ASP.NET
- Telerik/Progress Developer Expert
- Organizer of Beer City Code

- jtower@trailheadtechnology.com
- trailheadtechnology.com/blog
- jtowermi

github.com/jonathantower/aspnet-core-di