Implementation of Dependency Injection Pattern in C#

(/mentors/shailendra-chauhan)

Shailendra Chauhan (/mentors/shailendra-chauhan)

4 min read **Print**

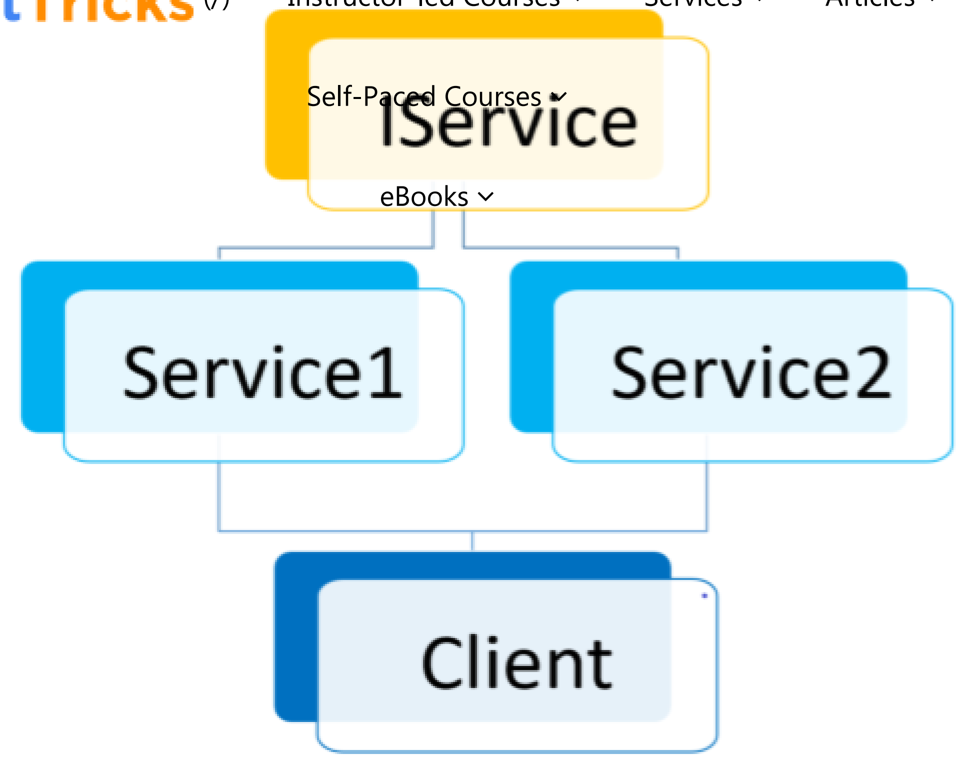
12 Apr 2013 29 Oct 2018

Intermediate 408K

Dependency Injection (DI) is a software design pattern that allows us to develop loosely coupled code. DI is a great way to reduce tight coupling between software components. DI also enables us to better manage future changes and other complexity in our software. The purpose of DI is to make code maintainable.

The Dependency Injection pattern uses a builder object to initialize objects and provide the required dependencies to the object means it allows you to "inject" a dependency from outside the class.

For example, Suppose your `Client` class needs to use two service classes, then the best you can do is to make your `Client` class aware of abstraction i.e. `IService` interface rather than implementation i.e. `Service1` and `Service2` classes. In this way, you can change the implementation of the `IService` interface at any time (and for how many times you want) without changing the client class code.



We can modify this code by following the Dependency Injection implementation (https://www.dotnettricks.com/learn/dependencyinjection) ways. We have following different ways to implement DI :

Constructor Injection

01. This is the widely used way to implement DI.

02. Dependency Injection is done by supplying the DEPENDENCY through the class's constructor when creating the instance of that class.

03. Injected component can be used anywhere within the class.

04. Recommended to use when the injected dependency, you are using across the class methods.

05. It addresses the most common scenario where a class requires one or more dependencies.

```
public interface IService {
void Serve();
}
public class Service1 : IService {
 public void Serve() { Console.WriteLine("Service1 Called"); }
}
public class Service2 : IService {
 public void Serve() { Console.WriteLine("Service2 Called"); }
}
public class Client {
 private IService _service;
 public Client(IService service) {
 this._service = service;
 }
 public ServeMethod() { this._service.Serve(); }
}

class Program
{
 static void Main(string[] args)
 {
 //creating object
 Service1 s1 = new Service1();
 //passing dependency
 Client c1 = new Client(s1);
 //TO DO:

 Service2 s2 = new Service2();
 //passing dependency
 c1 = new Client(s2);
 //TO DO:
 }
}
```


The Injection happens in the constructor, by passing the Service that implements the IService Interface. The dependencies are assembled by a "Builder" and Builder responsibilities are as follows:

(tel:+911133034100)
(/)
Instructor-led Courses Services Articles Work With Us
Self-Paced Courses
Reg
eBooks

01. Know the types of each IService
details)

02. According to the request, feed the abstract IService to the Client

Property/Setter Injection

01. Recommeded to use when a class has optional dependencies, or where the implementations may need to be swapped.

02. Different logger implementations could be used in this way.

03. Does not required the creation of new object or modifying the existing one. Without chaning the object state, it could work.

```
public interface IService {
 void Serve();
}
public class Service1 : IService {
 public void Serve() { Console.WriteLine("Service1 Called"); }
}
public class Service2 : IService {
 public void Serve() { Console.WriteLine("Service2 Called"); }
}
public class Client {
 private IService _service;
 public IService Service {
 set { this._service = value; }
 }
 public ServeMethod() { this._service.Serve(); }
}
```

Find Courses, Jobs, Articles & More

```
class Program
{
    static void Main(string[] args)
    {
    //creating object
    Service1 s1 = new Service1();

    Client client = new Client();
    client.Service = s1; //passing dependency
    //TO DO:

    Service2 s2 = new Service2();
    client.Service = s2; //passing dependency
    //TO DO:
    }
}
```

Method Injection

01. Inject the dependency into a single method and generally for the use of that method.
02. It could be useful, where the whole class does not need the dependency, only one method having that dependency.
03. This is way is rarely used.

```
public interface IService {
 void Serve();
}
public class Service1 : IService {
 public void Serve() { Console.WriteLine("Service1 Called"); }
}
public class Service2 : IService {
 public void Serve() { Console.WriteLine("Service2 Called"); }
}
public class Client {
 private IService _service;
 public void Start(IService service) {
 service.Serve();
 }
}

class Program
{
 static void Main(string[] args)
 {
 //creating object
 Service1 s1 = new Service1();

 Client client = new Client();
 client.Start(s1); //passing dependency
 //TO DO:

 Service2 s2 = new Service2();
 client.Start(s2); //passing dependency
 }
}
```

Advantages of Dependency Injection

01. Reduces class coupling

02. Increases code reusability

03. Improves code maintainability

04. Make unit testing possible

DI Container

The recommended way to implement DI is, you should use DI containers. If you compose an application without a DI CONTAINER, it is like a `POOR MAN'S DI`. If you want to implemet DI within your ASP.NET MVC application using DI container, please do refer Dependency Injection in ASP.NET MVC using Unity IoC Container (https://www.dotnettricks.com/learn/dependencyinjection/dependency-injection-in-aspnet-mvc-4-using-unity-ioc-container).

What do you think?

I hope, you will enjoy the various ways of implementing DI pattern. I would like to have feedback from my blog readers. Your valuable feedback, question, or comments about this article are always welcome.

< Prev (/learn/dependencyinjection/what-is-ioc-container-or-di-container)

Next > (/learn/dependencyinjection/dependency-injection-in-aspnet-mvc-4-using-unity-ioc-container)

C# Questions and Answers

C# is an object-oriented programming language developed by Microsoft, which runs under .NET platform. Now, C# can be run on Mac, Linux/Unix and Windows using .NET Core. If you want to crack your C# interview, you'

Get This Book

(/books/csharp-interview-questions-and-answers-book-pdf)

ASP.NET Core Questions and Answers

ASP.NET Core is an open source, cross-platform framework for building web applications using C# and .NET. This book will teach you ASP.NET Core concepts from scratch to advance with the help of Interview Questions &

Get This Book

(/books/aspnet-core-interview-questions-and-answers-book-pdf)

Join Our Hands-on Training Programs

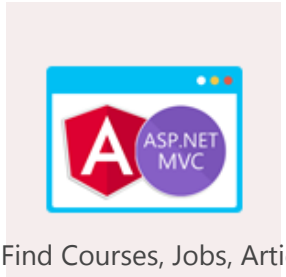
The Complete ASP.NET MVC5 with AngularJS

Learn how to build web applications using the ASP.NET MVC and AngularJS. In this course, you will learn about, how to create web pages, forms validations, custom validations, database operations, Web API, implementing

78 Lectures | 11+ Hours

View More

(/training/the-complete-aspnet-mvc-angularjs)



Project - ASP.NET MVC with Angular

Learn how to build an online e-commerce site using ASP.NET MVC5 and Angular5. The application will use an n-tier architecture with an extensible approach. We would be using the payment gateway to do online payment and

33 Lectures | 03+ Hours

View More

(/training/mvc-angular-project)

Microsoft Azure Developer Associate

The Microsoft Azure Developer Associate Certification Training Program makes you proficient in developing, planning, and scaling your web applications on Microsoft Azure. It includes training on Azure Infra, Azure Virtual Ma

View More

06
Courses

11
Skills

(/training/masters-program/microsoft-azure)

Learn. Build. Empower.

Master in-demand job skills with linear and project-based courses. Get access to world class Learning Platform to LEARN the tools and tricks to BUILD industry grade applications under our expert guidance. Our innovative hands-on training approach, will EMPOWER you to take any future assignment with confidence.

COMPANY

About Us (/about-us)
Founder (/about-shailendra-chauhan)
Terms and Conditions (/terms-and-conditions)
Privacy Policy (/privacy-policy)
Refund Policy (/refund-policy)
Verify Certificate (/certificate/verify)
Disclaimer
Sitemap (/sitemap)

PLATFORM

Mentors (/mentors)
Events (/events)
eBooks (/Books/Books)
Skill Challenge (/skill-challenge)
Professional Reviews (/reviews)
Corporate Training (/corporate-training)
Find Courses, Jobs, Articles & More
Software Consulting (/software-consulting-services)
Technical Recruiting (/technical-recruiting-service)

MORE

Contact Us (/contact-us)

(tel:+91113303410 0)
(/)
(/) Instructor-led Courses Services Articles Work With Us
details)
Student Scholarship (/scholarship-for-students)
Become An Instructor (/become-an-instructor)
Become An Author (/become-an-author)
Refer And Earn (/refer-and-earn)
Job Openings (/jobs)
Articles of the Month (/articles-of-the-month)

Self-Paced Courses Reg

eBooks

Made with ❤ in India.