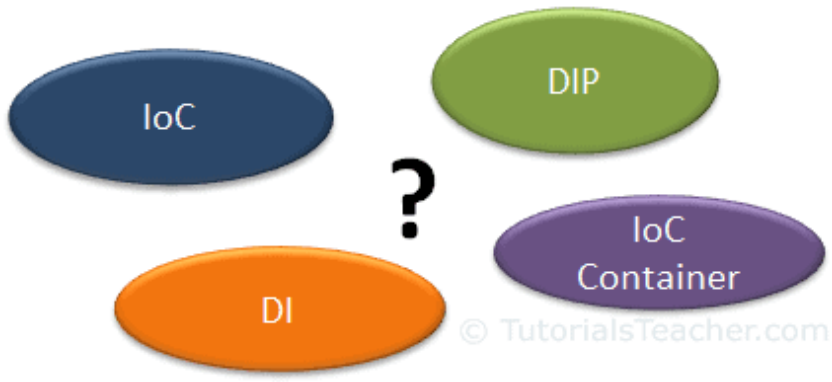
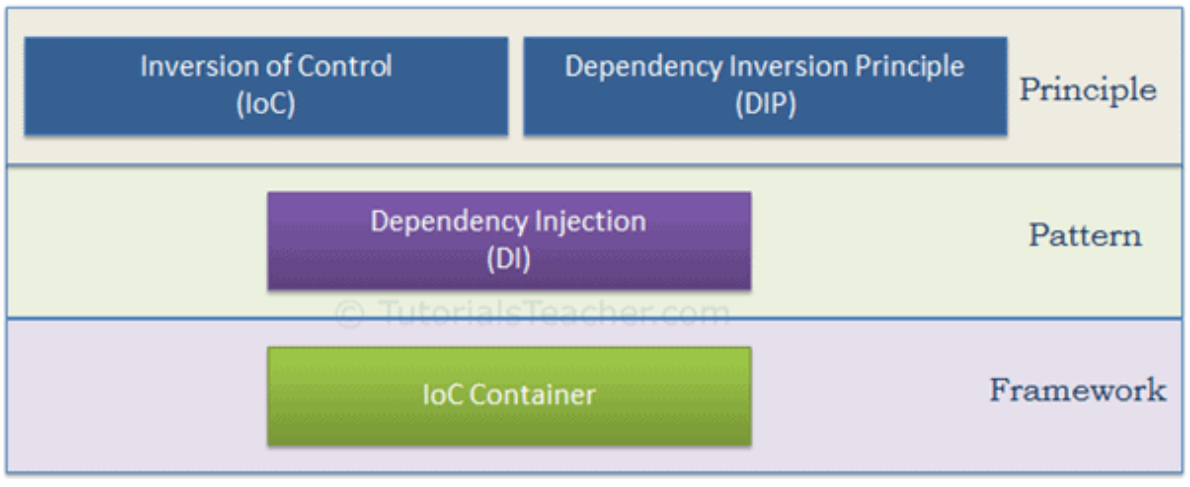
IoC Introduction

The terms Inversion of Control (IoC), Dependency Inversion Principle (DIP), Dependency Injection (DI), and IoC containers may be familiar. But are you clear about what each term means?



Here, you are going to learn about each term, using simple and real-world examples to clear your confusion. Before you go further, it is important to understand the difference between principle and pattern.

Now, let's understand the above buzz words. The following figure clarifies whether they are principles or patterns.



As illustrated in the above figure, IoC and DIP are high level design principles which should be used while designing application classes. As they are principles, they recommend certain best practices but do not provide any specific implementation details. Dependency Injection (DI) is a pattern and IoC container is a framework.

Let's have an overview of each term before going into details.

Inversion of Control

IoC is a design principle which recommends the inversion of different kinds of controls in object-oriented design to achieve loose coupling between application classes. In this case, control refers to any additional responsibilities a class has, other than its main responsibility, such as control over the flow of an application, or control over the dependent object creation and binding (Remember SRP - Single Responsibility Principle). If you want to do TDD (Test Driven Development), then you must use the IoC principle, without which TDD is not possible. Learn about IoC in detail in the next chapter.

Dependency Inversion Principle

The DIP principle also helps in achieving loose coupling between classes. It is highly recommended to use DIP and IoC together in order to achieve loose coupling.

DIP suggests that high-level modules should not depend on low level modules. Both should depend on abstraction.

The DIP principle was invented by Robert Martin (a.k.a. Uncle Bob). He is a founder of the SOLID principles. Learn more about DIP in the DIP chapter.

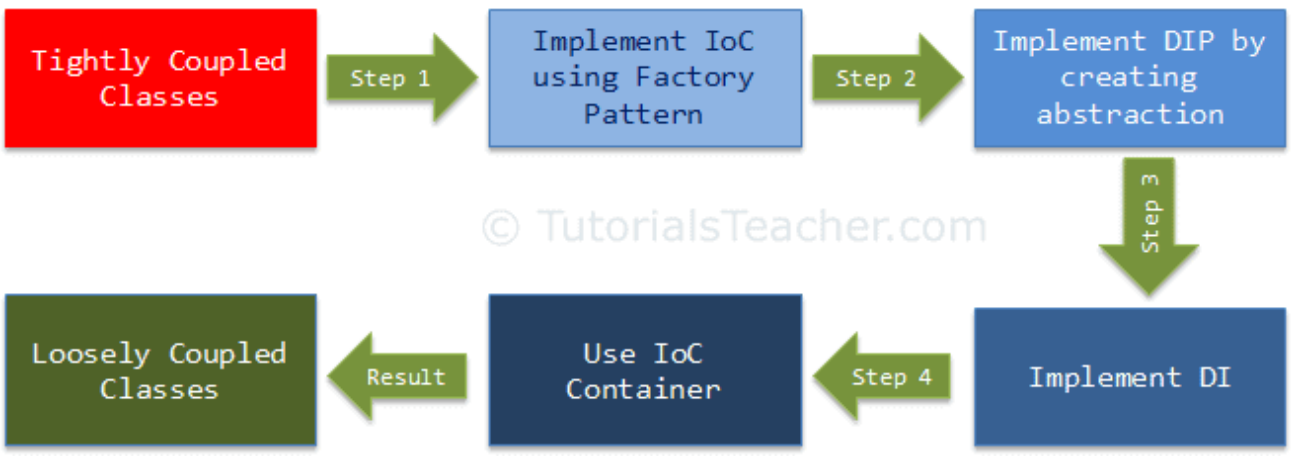
Dependency Injection

Dependency Injection (DI) is a design pattern which implements the IoC principle to invert the creation of dependent objects. We will learn about it in the DI chapter.

IoC Container

The IoC container is a framework used to manage automatic dependency injection throughout the application, so that we as programmers do not need to put more time and effort into it. There are various IoC Containers for .NET, such as Unity, Ninject, StructureMap, Autofac, etc. We will learn more about this in the IoC Container chapter.

We cannot achieve loosely coupled classes by using IoC alone. Along with IoC, we also need to use DIP, DI and IoC container. The following figure illustrates how we are going to achieve loosely coupled design step by step in the next few chapters.



Let's learn about each of the above steps, starting with IoC as the first step in the next chapter.

Want to check how much you know IoC?

Start IoC Test

Previous Next

TUTORIALSTEACHER.COM

TutorialsTeacher.com is optimized for learning web technologies step by step. Examples might be simplified to improve reading and basic understanding. While using this site, you agree to have read and accepted our terms of use and privacy policy.

feedback@tutorialsteacher.com

TUTORIALS

- ASP.NET Core
- ASP.NET MVC
- IoC
- Web API
- C#
- LINQ
- Entity Framework
- AngularJS 1
- Node.js
- D3.js
- JavaScript
- jQuery
- Sass
- Https

E-MAIL LIST

Subscribe to TutorialsTeacher email list and get latest updates, tips & tricks on C#, .Net, JavaScript, jQuery, AngularJS, Node.js to your inbox.

Email address GO

We respect your privacy.