# Dependency Injection (DI) vs. Inversion of Control (IOC)

==========================================================================

**[Shivprasad koirala](https://www.codeproject.com/script/Membership/View.aspx?mid=1335831)**, 1 Jan 2014

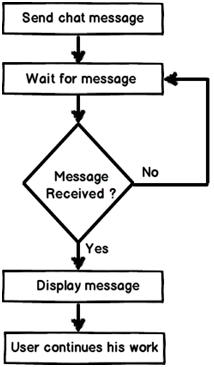
|  |  |
| --- | --- |
| https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-fill-md.png  https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-empty-md.png | 4.84 (68 votes) |
|  | |  |  |  | | --- | --- | --- | | Rate this: | [vote 1vote 2vote 3vote 4vote 5](https://www.codeproject.com/Articles/592372/Dependency-Injection-DI-vs-Inversion-of-Control-IO) |  | |

Dependency Injection (DI) vs. Inversion of Control (IOC).

The main goal of Inversion of control and Dependency Injection is to remove dependencies of an application. This makes the system more decoupled and maintainable.

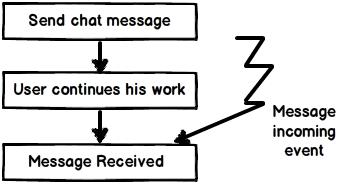
First let’s try to understand IOC (Inversion of control). If you go back to old computer programming days, program flow used to run in its own control. For instance let’s consider a simple chat application flow as shown in the below flow diagram.

1. End user sends chat message.
2. Application waits for the message from the other end.
3. If no message is found it goes to Step 2 or else moves to Step 4.
4. Displays the message.
5. User continues with his work ahead.

[](http://3.bp.blogspot.com/-Sb34Uxuvjmk/UZCCb_05h1I/AAAAAAAAEgA/qOSoSUw04e0/s1600/1st+image.jpg)

Now if you analyze the program flow closely, it’s sequential. The program is in control of himself. Inversion of control means the program delegates control to someone else who will drive the flow. For instance if we make the chat application event based then the flow of the program will go something as below:-

1. End user sends chat message.
2. User continues with his work ahead.
3. Application listens to events. If a message arrives event is activated and message is received and displayed.

[](http://3.bp.blogspot.com/-4Y4krJiAn0U/UZCCxNPNDjI/AAAAAAAAEgI/p1bqKzyZPw4/s1600/2nd+image.jpg)

If you see the program flow it’s not sequential, its event based. So now the control is inverted. So rather than the internal program controlling the flow, events drive the program flow. Event flow approach is more flexible as their no direct invocation which leads to more flexibility.

A word of caution here, do not conclude that IOC are implemented by only events. You can delegate the control flow by callback delegates, observer pattern, events, DI (Dependency injection) and lot of other ways.

IOC (Inversion of control) is a general parent term while DI (Dependency injection) is a subset of IOC. IOC is a concept where the flow of application is inverted. So for example rather than the caller calling the method.

Hide   Copy Code

SomeObject.Call();

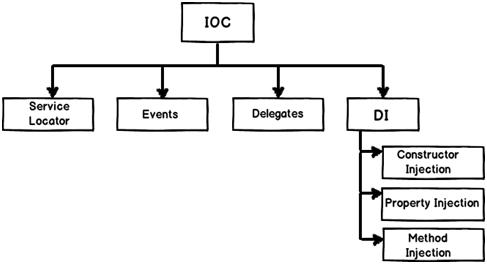
Will get replaced with an event based approach as shown below.

Hide   Copy Code

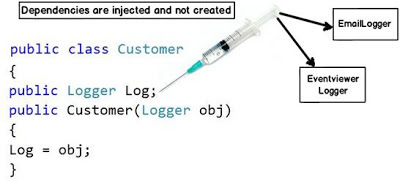
SomeObject.WhenEvent += Call();

In the above code the caller is exposing an event and when that event occurs he is taking action. It’s based on the Hollywood principle “Don’t call us we will call you”. In Hollywood when artists used to give auditions the judges would say them “Don’t call us we will call you”.

The above approach makes code more flexible as the caller is not aware of the object methods and the object is not aware of caller program flow.

[](http://1.bp.blogspot.com/-2JkcEGnJVrY/UZCDAoPjk8I/AAAAAAAAEgQ/ilz1zM5fc78/s1600/3rd+image.jpg)

DI provides objects that an object needs. So rather than the dependencies construct themselves they are injected by some external means. For instance let’s say we have the following below class “Customer” who uses a “Logger” class to log errors. So rather than creating the “Logger” from within the class, you can inject the same via a constructor as shown in the below code snippet.

[](http://4.bp.blogspot.com/-vUOcNsQcrUk/UZCDROepyWI/AAAAAAAAEgY/8NbtFggPfh4/s1600/4th+image.jpg)

The biggest benefit achieved by the above approach is “Decoupling”. You can now invoke the customer object and pass any kind of “Logger” object as shown in the below code.

Hide   Copy Code

Customer obj = new Customer(new EmailLogger());

Customer obj1 = new Customer(new EventViewerLogger());

So summarizing the differences.

**Inversion of control** :- It’s a generic term and implemented in several ways (events, delegates etc).

**Dependency injection** :- DI is a subtype of IOC and is implemented by constructor injection, setter injection or method injection.