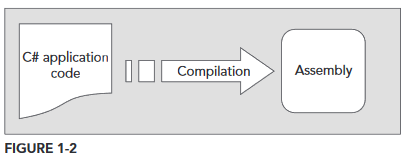
# C# Chapter 1 Introduction

Garbage Collection

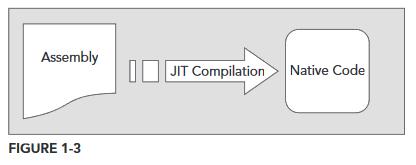
One of the most important features of managed code is the concept of garbage collection. This is the .Net method of making sure that the memory used by an application is freed up completely when the application is no longer in use. .Net garbage collection works by periodically inspecting the memory of the computer and removing anything that is no longer needed.

## Steps required to create a .Net application

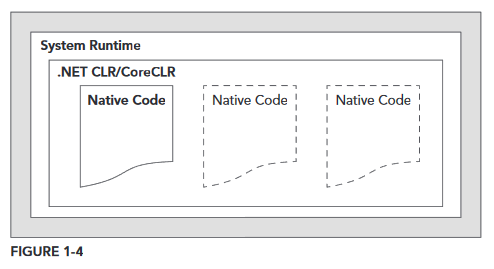
1. Application code is written using a .Net-compatible language such as C#.
2. That code is compiled into CIL, which is stored in an assembly. (Figure 1-2)



1. When this code is executed (either in its own right if it is an executable or when it is used from other code), it must first be compiled into native code using a JIT compiler. (Figure 1-3)



1. The native code is executed in the context of the managed CLR/CoreCLR, along with any other running applications or processes, Figure 1-4.



## Linking

**Note** **:** The C# code that compiles into the CIL doesn’t need to be contained in one single file. It is possible to split application code across multiple source code files, which are compiled together into a single assembly. This is known as **linking**. It is required because it is much easier to work with smaller files than one enormous one.

## What is C#?

C# is a slightly more verbose language than C++. This is a consequence of C# being a **typesafe** language (unlike C++). This means that once some data has been assigned to a **type**, it cannot subsequently transform itself into another **unrelated type**. Consequently, strict rules must be adhered to when converting between types, this means that you will often need to write more code to carry out the same task in C# that you might write in C++.

However, there are benefits to this – the code is more robust, debugging is simpler, and .Net can always track the type of a piece of data at any time.

## Types of applications you can write in C#:

**\*\*For Console & ASP.NET applications you can only do so using .Net Core\*\***

* **Desktop Application**
  + You can use the WPF (Windows Presentation Foundation) module of .Net Framework (library of controls e.g. buttons, toolbars, menus and so on) to build a Windows UI.
* **Windows Store Application**
  + The Windows Store Application is used primarily for touch devices, and it is usually run full-screen, with minimum of clutter, and an emphasis on simplicity.
* **Cloud / Web Application**
  + For generating web content dynamically, .Net Framework and .Net Core use the powerful ASP.NET system. You can host these platforms in the Cloud e.g. Azure.
* **Web APIs**
  + An ideal framework for building RESTful HTTP services that support a broad variety of clients, including mobile devices and browsers.
* **WCF Services**
  + A way to create versatile distribute applications. Using WCF you can exchange data over local networks or the internet.

In addition, some of these types might also require database access. This can be achieved using the ADO.NET (Active Data Objects .NET) section of the .NET framework, through the Entity framework, or through the LINQ (Language Integrated Query) capabilities of C#.

|  |  |
| --- | --- |
| **TOPIC** | **KEY CONCEPTS** |
| **.NET Framework and .NET Core Fundamentals** | Currently in Version 4.7. it includes a common type system (CTS) and common language runtime (CLR/CoreCLR). Noth .NET Framework and .NET Core applications are written using OOP methodology, and usually contain managed code. Memory management of managed code is handled by the .NET runtime; this includes the garbage collection. |
| **.NET Framework applications** | Applications using the .Net Framework are first compiled into CIL. When an application is executed, the CLR uses a JIT to compile this CIL into native code for the OS. Applications are compiled and different parts are linked together into assemblies that contain the CIL. |
| **.NET Core Applications** | Work similar to .NET Framework applications however instead of using the CLR is uses CoreCLR. |
| **.NET Standard** | Provides a unified class library which can be targeted from multiple .NET platforms like the .NET Framework, .NET Core, and Xamarin. |
| **Integrated Development Environments (IDEs)** | Can use Visual Studio to write any type of .NET application using C#. |