The .NET Framework is a technology that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

* To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
* To provide a code-execution environment that minimizes software deployment and versioning conflicts.
* To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
* To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
* To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
* To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

Manage Code - Code which is executing in CLR

Managed Execution Environment – Choosing a compiler, Convert HLL to MSIL,IL to Machine code using JIT, Running Code

Dotnet Framework = FCL+CLR

The **Common Language Infrastructure** (**CLI**) is an open [specification](https://en.wikipedia.org/wiki/Specification) (technical standard) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) and standardized by [ISO](https://en.wikipedia.org/wiki/International_Organization_for_Standardization)[[1]](https://en.wikipedia.org/wiki/Common_Language_Infrastructure#cite_note-iso_iec_23271_2012-1) and [ECMA](https://en.wikipedia.org/wiki/Ecma_International" \o "Ecma International)[[2]](https://en.wikipedia.org/wiki/Common_Language_Infrastructure#cite_note-ecma_355_2012-2) that describes executable code and a runtime environment that allows multiple high-level languages to be used on different computer platforms without being rewritten for specific architectures.

The [.NET Framework](https://en.wikipedia.org/wiki/.NET_Framework) and the [free and open source](https://en.wikipedia.org/wiki/Free_and_open_source_software) [Mono](https://en.wikipedia.org/wiki/Mono_(software)) and [Portable.NET](https://en.wikipedia.org/wiki/Portable.NET" \o "Portable.NET) are implementations of the CLI.

CLI Consist of :

1. **The**[**Common Type System**](https://en.wikipedia.org/wiki/Common_Type_System)**(CTS)**

A set of [data types](https://en.wikipedia.org/wiki/Data_type) and operations that are shared by all CTS-compliant [programming languages](https://en.wikipedia.org/wiki/Programming_language).

1. **The**[**Metadata**](https://en.wikipedia.org/wiki/Metadata_(CLI))

Information about program structure is [language-agnostic](https://en.wikipedia.org/wiki/Language-independent_specification), so that it can be referenced between languages and tools, making it easy to work with code written in a language one's not using.

1. **The Common Language Specification (CLS)**

A set of base rules to which any language targeting the CLI should conform in order to interoperate with other CLS-compliant languages. The CLS rules define a subset of the Common Type System.

1. **The**[**Virtual Execution System**](https://en.wikipedia.org/wiki/Virtual_Execution_System)**(VES)**

The VES loads and executes CLI-compatible programs, using the metadata to combine separately generated pieces of code at runtime.

MetaData refers to certain data structures embedded within the [Common Intermediate Language](https://en.wikipedia.org/wiki/Common_Intermediate_Language) (CIL) code that describes the high-level structure of the code. Metadata describes all [classes](https://en.wikipedia.org/wiki/Class_(computer_science)) and class members that are defined in the assembly, and the classes and class [members](https://en.wikipedia.org/wiki/Method_(computer_science)) that the current assembly will call from another assembly. The metadata for a method contains the complete description of the method, including the class (and the assembly that contains the class), the [return type](https://en.wikipedia.org/wiki/Return_type) and all of the method [parameters](https://en.wikipedia.org/wiki/Parameter_(computer_science)).

The **Virtual Execution System** (**VES**) is a [run-time system](https://en.wikipedia.org/wiki/Run-time_system) of the [Common Language Infrastructure](https://en.wikipedia.org/wiki/Common_Language_Infrastructure) CLI which provides an environment for executing [managed code](https://en.wikipedia.org/wiki/Managed_code)

**Common Type System** (**CTS**) defines how types are declared, used, and managed in the common language runtime, and is also an important part of the runtime's support for cross-language integration. The common type system performs the following functions:

* Establishes a framework that helps enable cross-language integration, type safety, and high-performance code execution.
* Provides an object-oriented model that supports the complete implementation of many programming languages.
* Defines rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.
* Provides a library that contains the primitive data types (such as [Boolean](https://msdn.microsoft.com/en-us/library/system.boolean(v=vs.100).aspx), [Byte](https://msdn.microsoft.com/en-us/library/system.byte(v=vs.100).aspx), [Char](https://msdn.microsoft.com/en-us/library/system.char(v=vs.100).aspx), [Int32](https://msdn.microsoft.com/en-us/library/system.int32(v=vs.100).aspx), and [UInt64](https://msdn.microsoft.com/en-us/library/system.uint64(v=vs.100).aspx)) used in application development.

**Common Language Specification** (CLS), which is a set of basic language features needed by many applications, has been defined. The CLS rules define a subset of the [Common Type System](https://msdn.microsoft.com/en-us/library/zcx1eb1e(v=vs.100).aspx); that is, all the rules that apply to the common type system apply to the CLS, except where stricter rules are defined in the CLS

The CLR provides the following benefits:

* Performance improvements.
* The ability to easily use components developed in other languages.
* Extensible types provided by a class library.
* Language features such as inheritance, interfaces, and overloading for object-oriented programming.
* Support for explicit free threading that allows creation of multithreaded, scalable applications.
* Support for structured exception handling.
* Support for custom attributes.
* Garbage collection.
* Use of delegates instead of function pointers for increased type safety and security.

**Dotnet Framework 4.6.2 Features**

1. Debugging improvements -The *unmanaged debugging API* has been enhanced in the .NET Framework 4.6.2 to perform additional analysis when a [NullReferenceException](https://msdn.microsoft.com/en-us/library/system.nullreferenceexception(v=vs.110).aspx) is thrown so that it is possible to determine which variable in a single line of source code is null.

### 2. Converting Windows Forms and WPF apps to UWP apps

Windows now offers capabilities to bring existing Windows desktop apps, including WPF and Windows Forms apps, to the Universal Windows Platform (UWP).

### 3. Character categories

Characters in the .NET Framework 4.6.2 are classified based on the [Unicode Standard, Version 8.0.0](http://www.unicode.org/versions/Unicode8.0.0/).

**4. Improved support for localized error messages in data annotation validators**  
Data annotation validators enable you to perform validation by adding one or more attributes to a class property. The attribute's [ValidationAttribute.ErrorMessage](https://msdn.microsoft.com/en-us/library/system.componentmodel.dataannotations.validationattribute.errormessage(v=vs.110).aspx) element defines the text of the error message if validation fails.

public class RatingInfo

{

[Required(ErrorMessage = "The rating must be between 1 and 10.")]

[Display(Name = "Your Rating")]

public int Rating { get; set; }

}