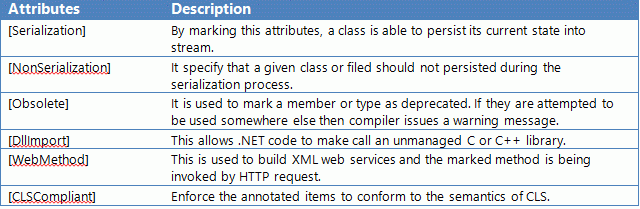
Reflection is the process of describing the metadata of types, methods and fields in a code. The namespace **System.Reflection** enables you to obtain data about the loaded assemblies, the elements within them like classes, methods and value types. Some of the commonly used classes of *System.Reflection* are:

|  |  |
| --- | --- |
| Class | Description |
| Assembly | describes an assembly which is a reusable, versionable, and self-describing building block of a common language runtime application |
| AssemblyName | Identifies an assembly ith a unique name |
| ConstructorInfo | Describes a class constructor and gives access to the metadata |
| MethodInfo | Describes the class method and gives access to its metadata |
| ParameterInfo | Describes the parameters of a method and gives access to its metadata |
| EventInfo | Describes the event info and gives accessto its metadata |
| PropertyInfo | Discovers the attributes of a property and provides access to property metadata |
| MemberInfo | Obtains information about the attributes of a member and provides access to member metadata |

Attributes

Attributes are like adjectives, which are used for metadata annotation that can be applied to a given type, assembly, module, method and so on. The .NET framework stipulates two types of attribute implementations, which are Predefined Attributes and Custom Attributes.  
  
Attributes are types derived from the System.Attribute class. This is an abstract class defining the required services of any attribute. The following is the syntax of an attribute;  
  
[**type:** attributeName(parameter1, parameter2,.........n)]  
  
The attribute name is the class name of the attribute. Attributes can have zero or more parameters. The following code sample states the attributes implementation in which we are declaring a method as deprecated using the obsolete attribute:

Predefined Attributes

The predefined attributes have been defined by Microsoft as a part of .NET FCL, and many of them receive special support from the C# compiler. Which implies that for those specific attributes, the compiler could customize the compilation process in a specific way.  
  
The System.Attribute base class library provides a number of attributes in various namespaces. The following table gives a snapshot of some predefined attributes.  
  


### Custom Attributes

Custom attributes can be created in C# for attaching declarative information to methods, assemblies, properties, types, etc. in any way required. This increases the extensibility of the .NET framework.

Steps for creating Custom Attributes:

* Define a custom attribute class that is derived from System.Attribute class.
* The custom attribute class name should have the suffix **Attribute**.
* Use the attribute AttributeUsage to specify the usage of the custom attribute class created.
* Create the constructor and the accessible properties of the custom attribute class.