Assembly.Load Method (System.Reflection)

By dotnet-bot

Archived: 2026-04-05 17:21:50 UTC

Source:
Assembly.CoreCLR.cs

Source:
Assembly.CoreCLR.cs

Source:
Assembly.CoreCLR.cs

Source:
Assembly.CoreCLR.cs

Source:
Assembly.CoreCLR.cs

Loads an assembly with the specified name.

```
public static System.Reflection.Assembly Load(string assemblyString);
```

Parameters

assemblyString
String

The long or short form of the assembly name.

Returns

The loaded assembly.

Exceptions

`assemblyString` is a zero-length string.

`assemblyString` is not found.

A file that was found could not be loaded.

`assemblyString` is not a valid assembly for the currently loaded runtime.

Examples

The following example loads an assembly given its fully qualified name, and lists all the types contained in the specified assembly. For information about how to obtain the fully qualified assembly name, see Assembly Names.

```
 using System;
using System.Reflection;

class Class1
{
    public static void Main()
    {
        // You must supply a valid fully qualified assembly name.
        Assembly SampleAssembly = Assembly.Load
            ("SampleAssembly, Version=1.0.2004.0, Culture=neutral, PublicKeyToken=8744b20f8da049e3");
        // Display all the types contained in the specified assembly.
        foreach (Type oType in SampleAssembly.GetTypes()) {
            Console.WriteLine(oType.Name);
        }
    }
}
```

Remarks

In .NET Core/.NET 5+, the target assembly will be loaded into the current AssemblyLoadContext or into the AssemblyLoadContext.CurrentContextualReflectionContext context if it's set. For more information on assembly loading, see Managed assembly loading algorithm.

To load the correct assembly, it's recommended to call the `Load` method by passing the long form of the assembly name. The long form of an assembly name consists of its simple name (such as "System" for the System.dll assembly) along with its version, culture, public key token, and optionally its processor architecture. It corresponds to the assembly's FullName property. The following example illustrates the use of a long name to load the System.dll assembly for .NET Framework 4:

```
 using System;
using System.Reflection;

public class Example
{
   public static void Main()
   {
      string longName = "system, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089";
      Assembly assem = Assembly.Load(longName);
      if (assem == null)
         Console.WriteLine("Unable to load assembly...");
```

```
      else
         Console.WriteLine(assem.FullName);
   }
}
// The example displays the following output:
//        system, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
```

FileLoadException is thrown if `assemblyString` specifies the full assembly name, and the first assembly that matches the simple name has a different version, culture, or public key token. The loader does not continue probing for other assemblies that match the simple name.

In the .NET Framework version 2.0, processor architecture is added to assembly identity, and can be specified as part of assembly name strings. For example, "ProcessorArchitecture=msil". However, the recommended way to specify an assembly name is to create an AssemblyName object and pass it to an appropriate overload of the Load method. See AssemblyName.ProcessorArchitecture.

See also

- AssemblyName
- LoadFrom(String)
- How the Runtime Locates Assemblies

Applies to

▶ .NET 11 and other versions
