



UNIVERSITA' DEGLI STUDI DI BERGAMO
Facoltà di Ingegneria

Informatica Industriale

Prof. Davide Brugali

2.3 – Microsoft .NET

Microsoft .NET

- The essence of the Microsoft proposal is the possibility to compile existing code and new code written in the programmer's favorite language.
- The resulting applications can interoperate with class libraries and components written in different languages using the .NET runtime environment.

Microsoft .NET

- ❑ The *Common Type System (CTS)* is an object model that extends the previous COM e DCOM models with the goal of supporting multiple languages software development.
- ❑ The *Intermediate Language (IL)* is an object-oriented language that conforms to the CTS. Various Microsoft language compilers (for C++, Java, etc.) generate code in the IL language (e.g. Microsoft VisualStudio.NET).

Microsoft .NET

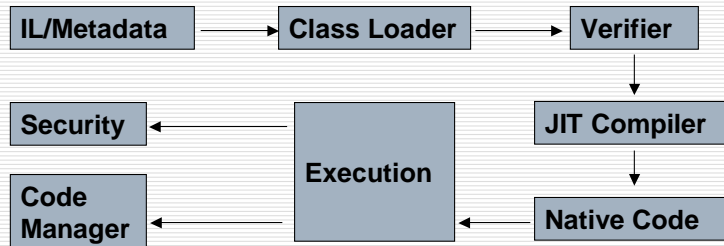
- ❑ The *Common Language Infrastructure (CLI)* is a runtime environment that executes code compiled in the IL language.
- ❑ The *.NET Software Development Kit (SDK)* is an object-oriented collection of reusable components. It provides runtime hosts for the CLI for a variety of execution platforms. Internet Explorer is an example of a runtime host.

.NET Execution Path

Compile Time



Run Time



Execution System

- ☐ Loads and verifies IL code
- ☐ JIT compiles IL to native code before first use
- ☐ Provides memory management
- ☐ Performs security checks
- ☐ Supports exception handling
- ☐ Provides interoperation between managed and unmanaged code

Class Loading

- ❑ When loading a class, the ES verifies that the class conforms to any known details about that class supplied by other classes.
- ❑ The ES check that other types referenced by this type are available.

Intermediate Language

- ❑ IL (Intermediate Language)
- ❑ .Net compilers produce IL code (rather than native code)
- ❑ IL is an assembly language for an abstract machine that:
 - Is stack based
 - Supports an OO programming model

HelloWorld in C#

```
using System;
class HelloWorld
{
    public void SayHello()
    {
        Console.WriteLine("Hello World!");
    }
    public static void Main()
    {
        HelloWorld h = new HelloWorld();
        h.SayHello();
    }
}
```

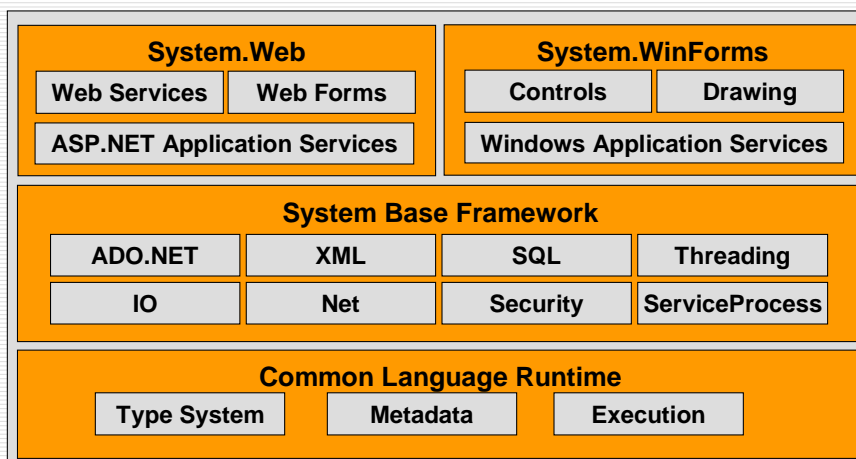
IL Code

```
.class auto ansi HelloWorld extends
['mscorlib']System.Object
{
    . . .
} // end of class 'HelloWorld'
```

IL Code

```
.method public hidebysig instance void  
    SayHello() il managed  
{  
    // Code size      11 (0xb)  
    .maxstack 8  
    IL_0000: ldstr      "Hello World!"  
    IL_0005: call       void  
        ['mscorlib']System.Console::WriteLine(  
            class System.String)  
    IL_000a: ret  
} // end of method 'HelloWorld::SayHello'
```

.NET Framework Architecture



System

- ❑ Provides basic and fundamental .NET types and services, such as:
 - System.Object
 - System.Type
- ❑ Most applications will be created using higher-level abstractions in the Framework.
- ❑ However, all functionality provided by the Framework is built on top of this functionality; for example, much of the .NET functionality depends on Reflection.

System

```
...
static int Main(string[] args)
{
    System.String s = "Hello world!";
    System.Type t = s.GetType();
    System.Console.Out.WriteLine(t.ToString());
    return 0;
}
```

System.Reflection

- Provides types that allows programs to dynamically:
 - Inspect the Type System
 - Extend the Type System (create new types at runtime)
 - Create instances and invoke methods on types

System.Reflection

```
using System;
using System.Reflection;
namespace Damien
{
    public class TypeInfo
    {
        public static void Main(String[] args)
        {
            ...
        }
    }
}
```


System.Reflection

```
public static void Main(String[] args)
{
    Type t = Type.GetType("System.String");
    Console.WriteLine(t.FullName);

    Console.WriteLine(t.BaseType);

    MethodInfo[] ma = t.GetMethods();
    for(int i = 0; i < ma.Length; i++)
        Console.WriteLine(ma[i]);
}
```

System.Reflection

```
Object[] a = new Object[1];
a[0] = "Hello World";
Object o = Activator.CreateInstance(t, a);

Type[] p = new Type[2];
p[0] = Type.GetType("System.Int32");
p[1] = Type.GetType("System.String");
```

System.Reflection

```
MethodInfo m = t.GetMethod("Insert",p);
Console.WriteLine("Invoking this method");
Console.WriteLine(m);

a = new Object[2];
a[0] = 0;
a[1] = "Hello World ";
Console.WriteLine(m.Invoke(o, a));
}
```

System.Reflection

```
using System;
using System.Reflection;
using System.Text;

[AttributeUsage(AttributeTargets.ClassMembers,
                AllowMultiple = true)]
public class AuthorAttribute : System.Attribute
{
    public String Author;
    ...
}
```

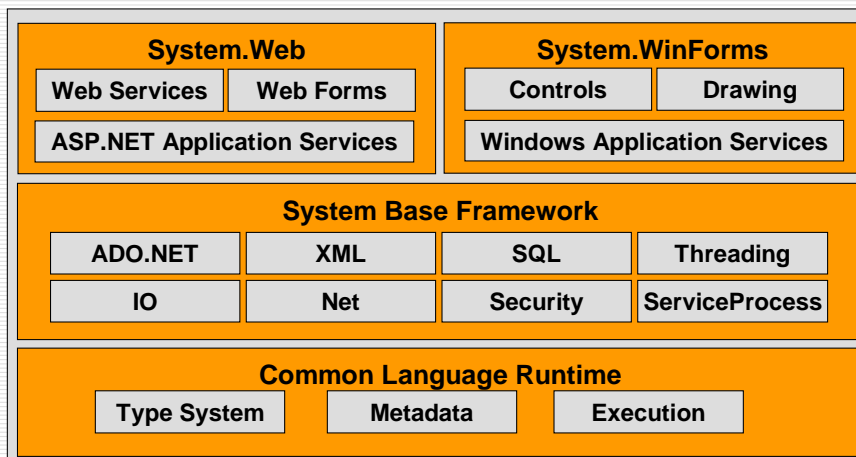
System.Collections

□ Provides basic collection types, such as:

- System.ArrayList
- System.Hashtable
- System.Stack

```
Hashtable ht = new Hashtable();
ht.Add( "Apple", "Green" );
ht.Add( "Orange", "Orange" );
ht.Add( "Grape", "Grape" );
Console.WriteLine("Number of items: " +
                  ht.Count);
IDictionaryEnumerator e = ht.GetEnumerator();
while(e.MoveNext())
    Console.WriteLine("Key: " + e.Key +
                      "value: " + e.Value );
```

Base Framework



Base Framework

- ❑ Contains a number of classes that generally provide an abstraction that covers an area of programming
- ❑ For example, `System.Threading` provides an abstraction of writing multi-threaded programs that can be used across languages

System.Threading

```
using System;
using System.Threading;
class ThreadClass
{
    public void StartHere()
    {
        Console.WriteLine("Thread starting");
        Thread.Sleep(0);
        Console.WriteLine("Thread ending");
    }
    ...
}
```

System.Threading

```
public static int Main()
{
    Console.WriteLine("Main Thread started");
    ThreadClass tc = new ThreadClass();
    Thread t = new Thread(
        new ThreadStart(tc.StartHere));
    t.Start();
    ...
}
```

System.Threading

```
    Console.WriteLine("t has been started");
    Thread.Sleep(0);
    while (
        (t.ThreadState & ThreadState.Unstarted) != 0);
    Console.WriteLine("Main Thread ending");
    return 0;
}
}
```

System.XML

```
public class Name
{
    public Name(){}
    public Name(string Family, string Given)
    {
        this.Family = Family;
        this.Given = Given;
    }
    public string Family;
    private string Given;
}
```

System.XML

```
public class Test
{
    public static int Main()
    {
        Name Damien = new Name("Watkins", "Damien");
        XmlSerializer XMLoutput =
            new XmlSerializer(Damien.GetType());
        StringWriter sw = new StringWriter();
        XMLoutput.Serialize(sw, Damien);
        Console.WriteLine(sw.ToString());
        return 0;
    }
}
```

System.Data.ADO (DB connection)

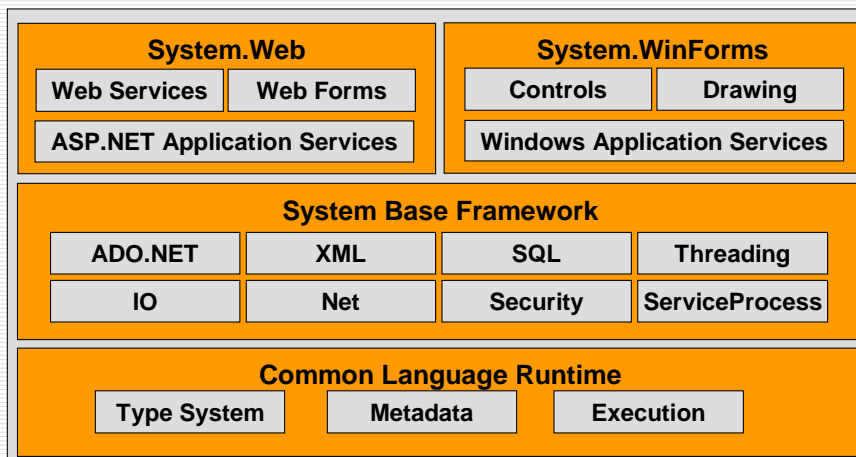
```
ADOConnection myConnection = new
    ADOConnection("Provider ...");

ADODataSetCommand myCommand = new
    ADODataSetCommand("SELECT ...", myConnection);

DataSet ds = new DataSet();
myCommand.FillDataSet(ds, "Subjects");

Subjects.DataSource = ds.Tables[0].DefaultView;
Subjects.DataBind();
```

ASP.NET



ASP.NET

- ❑ Provides a means of exposing the .NET Framework and its functionality to the WWW
- ❑ Contains a number of pre-built types that take input from .NET types and represents them in a form for the web (such as HTML)

ASP.NET

- ❑ In the following example, a DataList, named studentList:
 - Takes its input from a C# array object, which is located in a C# file
 - Displays it on a aspx page in HTML

ASP.NET

```
<asp:DataList runat=server
  id="studentList"
  RepeatColumns="2"
  RepeatDirection="Vertical"
  RepeatMode="Table"
  Width="100%">
  ...
</asp:DataList>
```

ASP.NET

```
<asp:DataList runat=server ...
<property name="AlternatingItemStyle">
  <asp:TableItemStyle BackColor="yellow"/>
</property>
<template name="ItemTemplate">
  <asp:Panel runat=server font-size="12pt"
    font-bold="true">
    <%# ((Student)Container.DataItem).Name %>
  </asp:Panel>
  <asp:Panel runat=server font-size="12pt">
    <%# ((Student)Container.DataItem).Number %>
  </asp:Panel> </template>
```

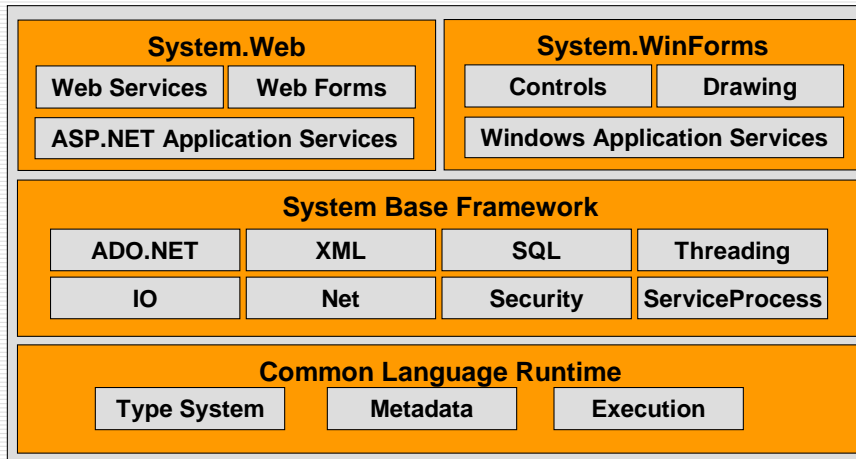
Web Services

- ❑ Web Services also provides a means to expose .NET functionality on the web but Web Services expose functionality via XML and SOAP

System.Web.Services

```
namespace WebServiceDemo
{
    ...
    using System.Web.Services;
    public class WebService1 :
        System.Web.Services.WebService
    {
        [WebMethod]
        public string HelloWorld()
        ...
    }
}
```

System.WinForms



2.3 - .NET

Informatica Industriale - Prof. Davide Brugali

37/38

System.WinForms

- ❑ Provides .NET types for building Windows Applications
- ❑ System.WinForms.Form

```
namespace Test {  
    using System;  
    ...  
    using System.WinForms.Forms;  
    public class Form1 :  
        System.WinForms.Forms.Form {  
        private System.Windows.Forms.Button b1;  
        ...  
    }  
}
```

2.3 - .NET

Informatica Industriale - Prof. Davide Brugali

38/38