



An Overview of Microsoft .NET and C#

*Hanspeter Mössenböck
University of Linz (Austria)*



What is .NET?

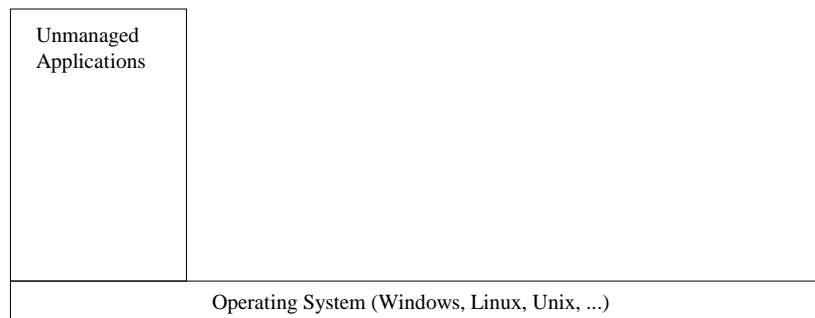


A new software platform for the PC and the Web

What is .NET?



A new software platform for the PC and the Web

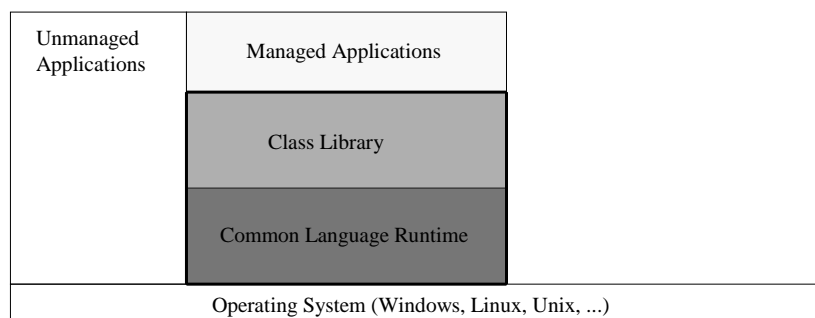


3

What is .NET?



A new software platform for the PC and the Web



CLR Interoperability, security, garbage collection, ...

4

What is .NET?



A new software platform for the PC and the Web

Unmanaged Applications	Managed Applications	Web Applications
	Class Library	Web Forms Web Services ASP.NET
	Common Language Runtime	Web Server (IIS)
Operating System (Windows, Linux, Unix, ...)		

ASP.NET, Web Forms Web-GUI (object-oriented, event-based, browser-independent)
 Web Services distributed services via RPC (SOAP, HTTP)

5

Goals and Benefits of .NET



Interoperability C#, C++, Eiffel, Fortran, Java, ML, Pascal, Perl, Python, Smalltalk, Visual Basic, ...

Uniformity uniform object model
 uniform type system
 web programming = desktop programming

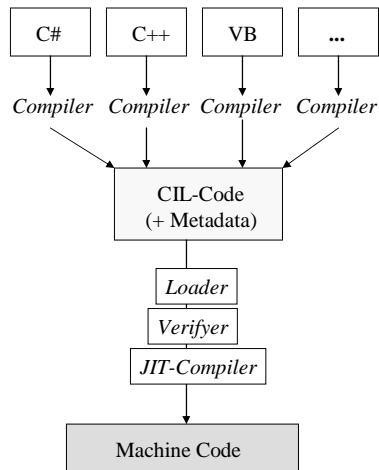
Security strict typing
 garbage collection
 versioning
 code-based and role-based access rights

Easy deployment end of "DLL Hell"

Web-orientation Web Services, Web Forms

6

Interoperability



```

    C#
    if (a > b) max = a; else max = b;

    CIL
    IL_0004: ldloc.0
    IL_0005: ldloc.1
    IL_0006: ble.s   IL_000c
    IL_0008: ldloc.0
    IL_0009: stloc.2
    IL_000a: br.s   IL_000e
    IL_000c: ldloc.1
    IL_000d: stloc.2

    Intel Code
    mov ebx,[-4]
    mov edx,[-8]
    cmp ebx,edx
    jle 17
    mov ebx,[-4]
    mov [-12],ebx
    ...
  
```

Interoperability



Binary compatibility between more than 20 programming languages.

Define a class in VB.NET

```

    Public Class A
      Public x As Integer
      Public Sub Foo() ...
    End Class
  
```

Define a subclass in C#

```

    class B : A {
      public string s;
      public void Bar() {...}
    }
  
```

Use it in Eiffel

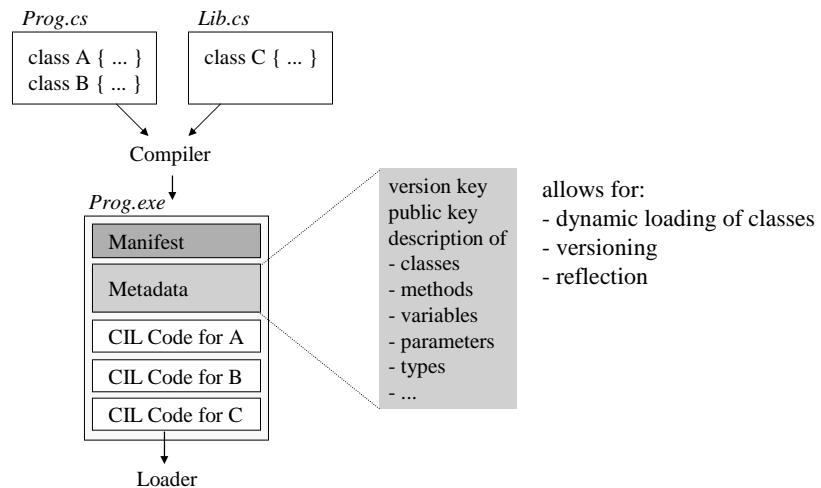
```

    class Client feature
      obj: B;
      ...
      create obj;
      obj.Bar;
      ...
    end
  
```

Assemblies



Self-contained units of deployment and versioning



9

ASP.NET -- Web Forms



Dynamic Web Pages with server-side scripting



Benefits

- All object-oriented
- Event-based
- Separation of HTML and script code
- Interactive Composition (RAD)
- Custom GUI elements
- Efficient (compiled server scripts)
- State management
- Authorisation / Authentication
- ...

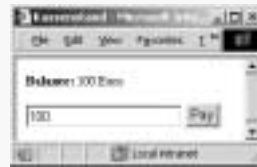
10

Sample Web Form



```
<%@ Page Language="C#" Inherits="AdderPage" Src="Adder.aspx.cs"%> Adder.aspx
<html>
<head><title>Cash Adder</title></head>
<body>
<form method="post" Runat="server">
<b>Balance:</b>
<asp:Label ID="total" Text="0" Runat="server"/> Euro<br><br>
<asp:TextBox ID="amount" Runat="server"/>
<asp:Button ID="ok" Text="Pay" OnClick="ButtonClick" Runat="server" />
</form>
</body>
</html>
```

```
using System; using System.Web.UI; using System.Web.UI.WebControls; Adder.aspx.cs
public class AdderPage : Page {
protected Label total;
protected TextBox amount;
protected Button ok;
public void ButtonClick (object sender, EventArgs e) {
int totalVal = Convert.ToInt32(total.Text);
int amountVal = Convert.ToInt32(amount.Text);
total.Text = (totalVal + amountVal).ToString();
}
}
```



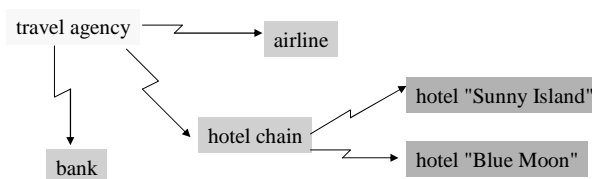
11

XML Web Services



Distributed services on the Internet

B2B applications without Web browser



Based on simple protocols

- HTTP
- SOAP (XML)
- Remote Procedure Call

12

Sample Web Service



```

<%@ WebService Language="C#" Class="BankService"%>
using System.Web.Services;

public class BankService: WebService {
    ...
    [WebMethod]
    public double GetRate(string currency) { // return exchange rates
        return table[currency];
    }
}
    
```

BankService.asmx

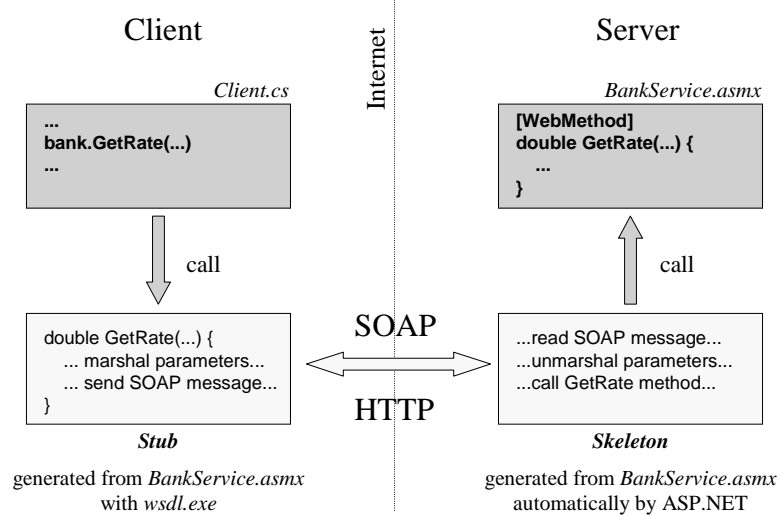
```

using System;
public class Client {
    public static void Main() {
        BankService bank = new BankService();
        Console.WriteLine(bank.GetRate("USD"));
    }
}
    
```

Client.cs

13

How Web Services Work



14



The Language C#

Similarities to Other Languages



Like in Java

- Object-oriented (single inheritance)
- Interfaces
- Exception handling
- Threads
- Namespaces (similar to packages)
- Strict typing
- Garbage collection
- Reflection
- Dynamic loading of classes
- ...

Like in C++

- Operator overloading
- Pointer arithmetic in unsafe code
- Some syntactical details

New Features in C#



Really new (vs. Java)

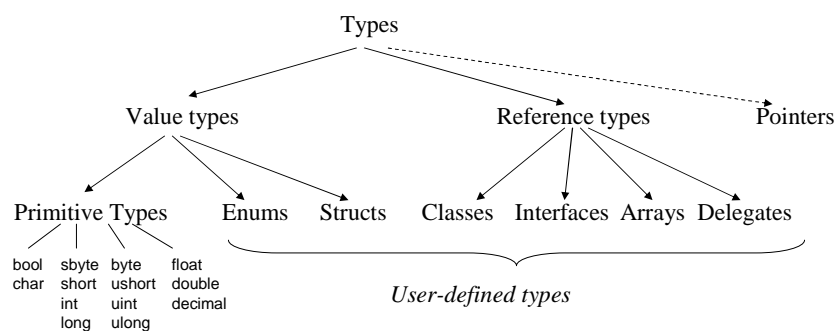
- Uniform type system
- Objects on the stack (structs)
- Enumeration types
- Call-by-reference parameters
- Block matrices
- Goto statement
- Attributes
- Low level programming possible
- Compatible with other .NET languages

"Syntactic Sugar"

- Properties
- Events
- Delegates
- Indexers
- Operator overloading
- foreach iterator
- Boxing/unboxing
- ...

17

Uniform Type System



All types are compatible with *object*

- can be assigned to *object* variables
- can perform *object* operations

18

Boxing and Unboxing



Value types (int, structs, enumerations) are also compatible with *object*!

Boxing

The assignment

```
object obj = 3;
```

automatically wraps up the value 3 in a heap object



Unboxing

The assignment

```
int x = (int) obj;
```

automatically unwraps the wrapped *int* value

19

Boxing and Unboxing



Useful for generic container types

```
class Queue {  
    ...  
    public void put(object x) {...}  
    public object get() {...}  
    ...  
}
```

This *Queue* can be used both for value types and for reference types

```
Queue q = new Queue();  
  
q.put(new Rectangle());  
q.put(3);  
  
Rectangle r = (Rectangle) q.get();  
int x = (int) q.get();
```

20

Classes and Inheritance



C#

```
class A {  
    private int x;  
    public A(int x) { this.x = x; }  
}  
  
class B : A, I1, I2 {  
    private int y;  
    public int Bar() { ...}  
    public B(int x, int y) : base(x) {  
        this.y = y;  
    }  
}
```

Java

```
class A {  
    private int x;  
    public A(int x) { this.x = x; }  
}  
  
class B extends A implements I1, I2 {  
    private int y;  
    public int Bar() { ...}  
    public B(int x, int y) {  
        super(x);  
        this.y = y;  
    }  
}
```

21

Overriding methods



C#

```
class A {  
    ...  
    public virtual void Foo() { ...}  
}  
  
class B : A {  
    ...  
    public override void Foo() {  
        base.Foo();  
        ...  
    }  
}
```

Java

```
class A {  
    ...  
    public void Foo() { ...}  
}  
  
class B extends A {  
    ...  
    public void Foo() {  
        super.Foo();  
        ...  
    }  
}
```

22

Properties



Special syntax for get/set methods

```
class Data {  
    FileStream s;  
  
    public string FileName {  
        set {  
            s = new FileStream(value, FileMode.Create);  
        }  
        get {  
            return s.Name;  
        }  
    }  
}
```

property type

property name

"input parameter" of the set method

Are used like "smart fields"

```
Data d = new Data();  
  
d.FileName = "myFile.txt"; // calls d.set("myFile.txt")  
string s = d.FileName;     // calls d.get()
```

Inlining makes get/set calls as efficient as field accesses.

23

Properties (continued)



get or set can be missing

```
class Account {  
    long balance;  
  
    public long Balance {  
        get { return balance; }  
    }  
}
```

x = account.Balance; // ok
account.Balance = ...; // forbidden

What are properties good for?

- Allows read-only or write-only data.
- Allows validation during data access.
- The implementation of data can differ from the clients' view.

24

Indexers



Programmable operators for indexing a sequence (collection)

```
class File {  
    FileStream s;  
  
    public int this [int pos] {  
        get { s.Seek(pos, SeekOrigin.Begin);  
            return s.ReadByte();  
        }  
        set { s.Seek(pos, SeekOrigin.Begin);  
            s.WriteByte((byte)value);  
        }  
    }  
}
```

Annotations in the original image:

- type of the indexed expression: points to `FileStream s;`
- name (always *this*): points to `public int this [int pos]`
- type and name of the index: points to `[int pos]`

Usage

```
File f = ...;  
int x = f[10];           // calls f.get(10)  
f[10] = 'A';           // calls f.set(10, 'A')
```

25

Delegate = Method Type



Declaration of a delegate type

```
delegate void Notifier (string sender); // normal method signature  
                                           // with the keyword delegate
```

Declaration of a delegate variable

```
Notifier notify;
```

Assigning a method to a delegate variable

```
void SayHello(string sender) {  
    Console.WriteLine("Hello from " + sender);  
}
```

```
notify = new Notifier(SayHello);
```

Calling the delegate variable

```
notify("Alice"); // calls SayHello("Alice") => "Hello from Alice"
```

26

Assigning Different Methods



Any compatible method can be assigned to a delegate variable

```
void SayGoodBye(string sender) {  
    Console.WriteLine("Good bye from " + sender);  
}  
  
Notifier notify = new Notifier(myObj.SayGoodBye);  
  
notify("Bob");    // calls myObj.SayGoodBye("Bob") => "Good bye from Bob"
```

A delegate variable actually stores a a method and its receiver object

27

Multicast Delegates



A delegate variable can hold multiple methods

```
Notifier notify;  
notify = new Notifier(SayHello);  
notify += new Notifier(SayGoodBye);
```

```
notify("Bob");    // "Hello from Bob"  
                  // "Good bye from Bob"
```

```
notify -= new Notifier(SayHello);
```

```
notify("Bob");    // "Good bye from Bob"
```

28

Threads



C#

```
void P() {  
    ... thread actions ...  
}  
  
Thread t = new Thread(new ThreadStart(P));  
t.Start();
```

Java

```
class MyThread extends Thread {  
    public void run() {  
        ... thread actions ...  
    }  
}  
  
Thread t = new MyThread();  
t.start();
```

- Any method can be started as a thread
- No subclass of *Thread* necessary
- Thread actions must be in a *run* method
- User thread must be a subclass of *Thread*

Thread synchronisation possible (similar to Java)

29

Attributes



User-defined information about program elements

- Can be attached to types, members, assemblies, etc.
- Stored as Metadata.
- Can be queried at run time.
- Implemented as classed derived from *System.Attribute*.

Example

```
[Serializable]  
class C {...} // makes C serializable
```

Multiple attributes possible

```
[Serializable] [Obsolete]  
class C {...}
```

```
[Serializable, Obsolete]  
class C {...}
```

30

Example: [Conditional] Attribute



For conditional method calls

```
#define debug // preprocessor command

class C {
    [Conditional("debug")]
    static void Assert (bool ok, string errorMsg) {
        if (!ok) {
            Console.WriteLine(errorMsg);
            System.Environment.Exit(0);
        }
    }

    static void Main (string[] arg) {
        Assert(arg.Length > 0, "no arguments specified");
        Assert(arg[0] == "...", "invalid argument");
        ...
    }
}
```

Assert is only called, if *debug* is defined.

31

Summary



- .NET is the way where Microsoft goes
- .NET makes things easier
 - Interoperability between languages
 - Type safety, security, versioning
 - Easy component deployment
 - Easy development of dynamic Web pages (ASP.NET)
 - Easy development of distributed services (Web Services)
- .NET is similar to Java but in some aspects ahead of it
 - Versioning
 - ASP.NET, Web Services
 - C#

32