.NET + C#

Burkhard Linke

Bielefeld, 23. Juni 2009

Overview

- What is .NET?
- Features of .NET
- Features of C#
- Summary

What is .NET

What is .NET?

- .NET is a
  - Implementation of a runtime environment
  - Created by Microsoft
  - Has become the de-facto environment on windows among Java
  - Available on other platforms, too (Mono)

What is .NET, cont

.NET Application	An application written in a Language supported by CLI
Common Language Infrastructure	An implementation of the CLI, e.g. by Microsoft
Operating System + System libraries	Provides services for the CLI

CLI: common language infrastructure, **definition** of the runtime environment

Available **implementations** of the CLI:

.NET	(Microsoft)
Mono	(open source)
dotGNU	(open source)

History of .NET

History of .NET

Old releases:

- .NET 1.0 (5.01.2002)
- .NET 2.0 (7.11.2005)

Current:

- .NET 3.5 SP1 (11.08.2008)

Announced:

- .NET 4.0 (Beta since 18.05.2009)

History of Mono

Old releases:

- 1.0(30.06.2004)

Current:

- 2.4 (30.03.2009)
  - Implements .NET 2.0 + many features of 3.5 on Unix platforms
  - Driven by Novell

Features of .NET

Features: Languages

- *Managed* code
  - Platform independent
  - Based on CLI
  - Object oriented
  - Garbage collected
- *Unmanaged* code
  - May perform unsafe operations
  - May access operating system services etc.

Features: Languages, cont

- CLS: *common language specification*
  - CIL: *common intermediate language*
    - Bytecode of compiled .NET applications
    - JIT-compiled at runtime
  - CTS: *common type system*
    - Definition of data types and their API

→ the same runtime for different languages

Features: Languages, cont.

About 50 languages available for the CLI:

- C#
- Visual Basic
- Java (IKVM, virtual machine on top of CLI)
- F#
- Python, Ruby, LUA, Nemerle, Scala, PHP....

Features: GAC

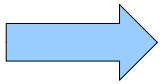
- *Assembly*: a library or executable
- Can be identified by combination of
  - Name
  - Version
  - „Culture“ (I18N)
  - Checksum (for signed assemblies)
- Similar to shared object/dll/jar file

Features: GAC

- GAC: *global assembly cache*
- System-wide cache of assemblies
- Similar to library cache of GNU ld
- Management of different versions of the same assembly
- No more *dll hell*

Features: Deployment

- Archive containing executable and assemblies
- Executable refers complete assembly definition
- Assemblies are loaded from working directory or GAC



- Extract to a directory
- Execute

Features of .NET

(presented as C# code)

C#: Attributes

- Meta data contained in assemblies
- May describe assembly, class, fields etc.
- Accessible at runtime during reflection
- Similar to *annotations* in Java

```
using System.Web.Services;
...
[WebService(Namespace = "http://biograph.cebitec.uni-bielefeld.de/",
Name="BioGraphInfo", Description="BioGraph web service information
interface")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class InfoService : System.Web.Services.WebService
{
    ....
    [WebMethod(Description = "returns true if this server requires
authentication")]
    public bool RequiresAuthentication()
    {
        ....
```

C#: Delegates + Events

- *Delegate:* function pointer, similar to C++

```
// definition of the delegate type
delegate void Greeter(string name);
class MainClass
{
  // static variable storing a delegate
  public static Greeter greet;
  public static void Main(string[] args)
  {
    // set a delegate
    greet = delegate(string name) {
      System.Console.WriteLine("Hello,{0}!"+name);
    };
    // invoke the delegate
    greet("World");
  }
}
```

C#: Delegates + Events, cont.

- *Event:* List of delegates

```
// definition of the delegate type
delegate void Greeter(string name);
class MainClass
{
  // an event of greeting people
  static event Greeter greet;
  public static void Main(string[] args)
  {
    // set a delegate
    greet += delegate(string name) {
      System.Console.WriteLine("Hello,{0}!",name);
    };
    // and add another delegate
    greet += delegate(string name) {
      System.Console.WriteLine("Guten Tag, {0}",name);
    };
    // invoke the delegate
    greet("World");
  }
}
```

C#: Delegates + Event

- Delegates can be invoked asynchronously on worker thread
- Events and delegates are base for GUI and Web development in .NET
- Similar to XXXListener in Java Swing
- May be written in any supported language

Features of C#, based on .NET 3.5

(presented as C# code, compiler feature)

C#: Extensions

- APIs cannot be changed after compilation
- Extension allow additions to the API:

```
static class StringHelper
{
  public static string UCFirst(this string input) {
  if (input.Length > 1)
    return input.Substring(0,1).ToUpper()+input.Substring(1);
  else if (input.Length == 1)
    return input.ToUpper();
  else
    return "";
  }
}


String foo = "bar";
String upperFoo = foo.UCFirst();
```

C#: functional programming

C# offers certain functional features

- Higher order functions
- Expression trees
- Lambda functions (think Haskell...)
- Type inference

Delegates, revisited:

```
// set a delegate
greet += s => {System.Console.WriteLine("Hello, "+ s);};
```

C#: LINQ

- LINQ: *language integrated query*
- Queries various data sources
  - Objects, XML files, SQL databases
- Queries are C# statements
  - Type checking by compiler
  - Optimization by compiler
- Mostly lazy evaluation

C#: Example

Example: convert string to camel case

```
#!/usr/bin/env perl

sub camelCase {
    return join("", map {ucfirst}
                    grep { length($_) > 0}
                    map { s/[^\w]//g; $_ }
                    split(/\s+/,$_[0]));
}

print camelCase("just a test for camel cases.")."\n"
```

„JustATestForCamelCases“

C#: Example

- Start with splitting the string

```
public static void Main(string[] args)
{
  String test = "Just a test for camel cases!";
  var result = test.Split(new char[]{ },
                          StringSplitOptions.RemoveEmptyEntries).


  .....


  System.Console.WriteLine(result);
}
```

Result: implicitly typed variable

C#: Example

- Remove unwanted characters

```
public static void Main(string[] args)
{
  String test = "Just a test for camel cases!";
  var result = test.Split(new char[]{ },
                          StringSplitOptions.RemoveEmptyEntries).
               Select((s) => Regex.Replace(s, @"\W", "")).

  .....


  System.Console.WriteLine(result);
}
```

Select(): applies a delegate to each element in a list

C#: Example

- Exclude empty strings

```
public static void Main(string[] args)
{
  String test = "Just a test for camel cases!";
  var result = test.Split(new char[]{ },
                          StringSplitOptions.RemoveEmptyEntries).
               Select((s) => Regex.Replace(s, @"\W", "")).
               Where((s) => s.Length > 0).
  .....


  System.Console.WriteLine(result);
}
```

Where(): filters list based on a predicate

C#: Example

- Convert using UCFirst and concatenate

```
public static void Main(string[] args)
{
  String test = "Just a test for camel cases!";
  var result = test.Split(new char[]{ },
                          StringSplitOptions.RemoveEmptyEntries).
               Select((s) => Regex.Replace(s, @"\W", "")).
               Where((s) => s.Length > 0).
               Select((s) => s.UCFirst()).
               Aggregate((first, second) => first + second);

  System.Console.WriteLine(result);
}
```

Aggregate(): aggregates all elements of a list and returns the result

Features beyond this talk

- ASP.NET – Webserver
- ADO.NET – Database access layer
- Windows.Form / WPF – GUI applications
- Remoting / WCF
- ...

Summary

- .NET offers a modern environment
- CLI + C# standardized by ECMA
- Platform independent (→ Mono)
- Comparable to Java SDK + Java EE
- Very modern approaches in C#
- Fun to work with C#

  (if mono and libgc would be stable...)