Creating and Using Strings

Gill Cleeren
CTO Xpirit Belgium

@gillcleeren | www.xpirit.com/gill

Agenda

Understanding strings in C#

Working with strings

Immutability of strings

Parsing from strings to other types

Understanding Strings in C#

h	e	l	l	o

```
string s1 = "Hello world";

string s2 = string.Empty;

var s3 = "I am a string too!";

string s4 = null;

string s5;
```

Creating Strings

Demo

Creating strings

Working with Strings

```
int l = myString.Length;

string upper = myString.ToUpper();

string lower = myString.ToLower();

bool b = myString.Contains("Hello");

string s = myString.Replace("a","b");

string sub = myString.Substring(1, 3);
```

◀ **Get the length of the string**

◀ **Set the string to uppercase**

◀ **Set the string to lowercase**

◀ **Check if a string contains “Hello”, return bool**

◀ **Replace “a” with “b” in the string**

◀ **Get a part of the string (zero-based)**

```
string s1 = "Learning C# ";//notice the extra space at the end

string s2 = "is awesome";

string s3 = s1 + s2;

//Output: "Learning C# is awesome"
```

Concatenating Multiple Strings

```
string s1 = "Learning C# ";//notice the extra space at the end

string s2 = "is awesome";

string s3 = String.Concat(s1, s2);

//Output: "Learning C# is awesome"
```

Using String.Concat

```
string employeeName = "Bethany";

int age = 34;

string greetingText = "Hello " + employeeName + ", you are " + age + " years";

//Output: Hello Bethany, you are 34 years
```

Less-readable String Concatenation

```
string employeeName = "Bethany";

int age = 34;

string greetingText =
    string.Format("Hello {0}, you are {1} years", employeeName, age);

//Output: Hello Bethany, you are 34 years
```

Using string.Format to Concatenate Strings

```
string employeeName = "Bethany";

int age = 34;

string greetingText = $"Hello {employeeName}, you are {age} years";

//Output: Hello Bethany, you are 34 years
```

String Interpolation

Often better and easier to read

Introduced with C# 6

Demo

Manipulating strings

Concatenating strings

Using string interpolation

```
Console.WriteLine("Here are the employee details:\nBethany\tSmith");
```

Adding Escape Characters

Always start with a \

```
string escapedFilePath = "C:\\Documents\\readme.txt";
```

Representing a File Path

```
string escapedFilePath = "C:\\Documents\\readme.txt";

string verbatimFilePath = @"C:\Documents\readme.txt";
```

Using Verbatim Strings

Used when text contains \ as part of the content

Improves readability

Demo

Escaping text

Using verbatim strings

Testing Strings for Equality

```
string firstName  = "Bethany";

bool b1 = firstName == "Bethany";//true

bool b2 = firstName == "bethany";//false

bool b3 = firstName.Equals("Bethany");//true
```

Comparing Two Strings

```
bool b = firstName.ToUpper() == anotherString.ToUpper();
```

Comparing Strings Case-insensitive

Demo

Comparing strings

The Immutability of Strings

```
string a = "Hello";

string b;

b = a;

b += " world";

Console.WriteLine(a);//Output: Hello
```

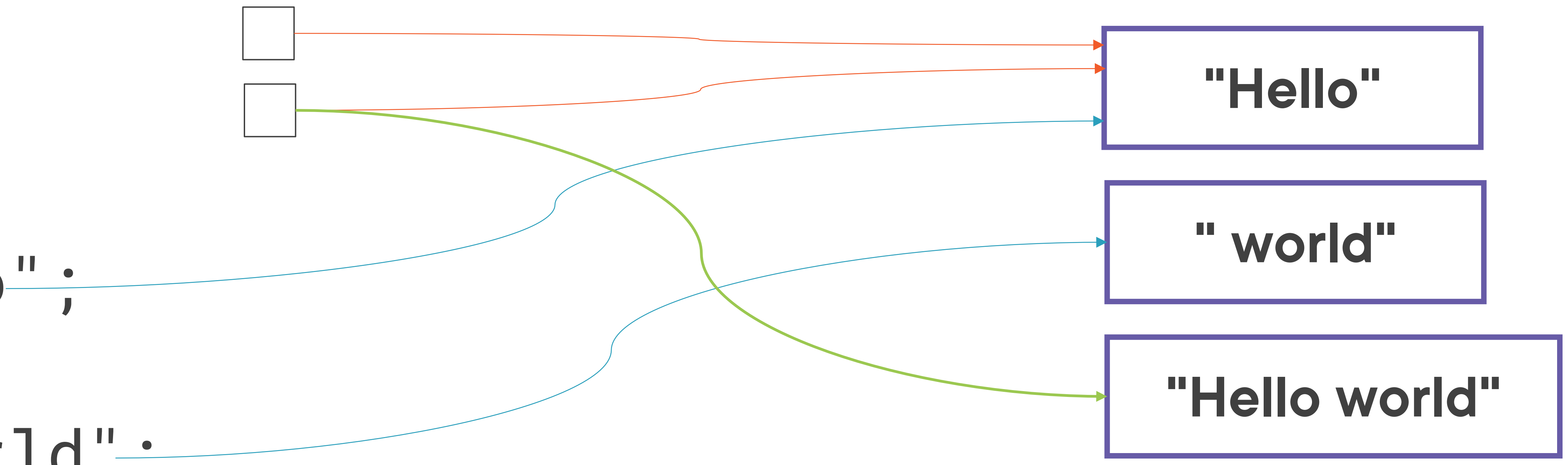
Let’s Append to an Existing String

Strings Are Reference Types

```
string a;
string b;

a = "Hello";
b = a;
b += " world";

Console.WriteLine("a = {0}", a);//a = Hello
Console.WriteLine("b = {0}", b);//b = Hello world
```



Strings are immutable.

String immutability can have a performance impact!

Loop actions or many concatenate actions can cause high memory use!

```
StringBuilder stringBuilder = new StringBuilder();

stringBuilder.Append("Employee list");

stringBuilder.AppendLine("Bethany Smith");

stringBuilder.AppendLine("George Jones");

stringBuilder.AppendLine("Gill Cleeren");

string list = stringBuilder.ToString();
```

Introducing the StringBuilder Class

Demo

Understanding string immutability

Using the StringBuilder

Parsing from Strings to Other Types

```
string w = Console.ReadLine();

double wage = double.Parse(w);


bool active = bool.Parse("true");
```

Use Parsing to Generate a Value from a String

Can cause issues though

```
string enteredText = "true";

if (bool.TryParse(enteredText, out bool b))

{

    Console.WriteLine($"The value is {b}");

}
```

Using TryParse

The out keyword will be covered in the next module

Demo

Parsing strings into other types

Using TryParse

Summary

Strings are a very important concept

Stored as references

Strings are immutable

- StringBuilder can be a solution

Up next:

Working with methods