Handling Strings

Deborah Kurata

@deborahkurata | blogs.msmvps.com/deborahk/

C# Strings

```
var name = "Frodo";
```

```
vendor.CompanyName = "ABC Corp";
```

```
var orderText = "Order from Acme, Inc" + Environment.NewLine +
                "Product: " + product.ProductCode + Environment.NewLine +
                "Quantity: " + quantity;
```

Strings Are Immutable

- The contents of a string cannot be changed after the string is created

```
 var orderText = "Order from Acme, Inc" + Environment.NewLine +
                 "Product: " + product.ProductCode + Environment.NewLine +
                              "Quantity: " + quantity;
if (deliverBy.HasValue)
{
   orderText += System.Environment.NewLine +
                "Deliver By: " + deliverBy.Value.ToString("d");
}
```

A String Is a Reference Type

Value Type

- Store their data directly
- Examples
  - int, decimal, bool, enum

Reference Type

- Store references to their data
- Examples
  - string, object

Value Type

Reference Type

Left on Main
South on 1st
Second house on the right

A String Is a Reference Type …

```
var name = "Frodo";          //name = Frodo
var hobbit = name;           //name = Frodo  hobbit = Frodo
hobbit = "Bilbo";            //name = Frodo  hobbit = Bilbo
```

…That Acts Like a Value Type

Module Overview

- Working with .NET String Methods
- Handling Nulls
- Defining Verbatim String Literals
- Formatting Strings
- Using String Interpolation
- Building Long Strings
- FAQ

.NET String Methods

```
var vendorInfo = "Vendor: ABC Corp.";

string result;
result = vendorInfo.ToLower();
result = vendorInfo.ToUpper();
result = vendorInfo.Replace("Vendor", "Supplier");

var length = vendorInfo.Length;
var index = vendorInfo.IndexOf(":");
var begins = vendorInfo.StartsWith("Vendor");
var ends = vendorInfo.EndsWith("Vendor");
```

.NET String Method Best Practices

Do:

Look at what .NET provides before writing your own

Use IntelliSense to view the list of available methods

Avoid:

Calling string methods on null strings

Handling Nulls

```
string vendorInfo = null;

string result;
result = vendorInfo.ToLower();
```

```
if (!String.IsNullOrWhiteSpace(vendorInfo))
{
    result = vendorInfo.ToLower();
}
```

```
result = vendorInfo?.ToLower();
```

Null String Handling Best Practices

Do:

Handle nulls

Write unit tests that cover null conditions

Use IsNullOrWhiteSpace when testing for null in a block of code

Use the null-conditional operator when checking for null in a single statement

Avoid:

Verbatim String Literals

```
var orderText = "Product: Tools-1\r\nQuantity: 12\r\nInstructions: standard delivery";
```

```
Product: Tools-1
Quantity: 12
Instructions: standard delivery
```

```
var directions = "Insert \r\n to define a new line";
```

```
Insert
 to define a new line
```

```
var directions = @"Insert \r\n to define a new line";
```

```
Insert \r\n to define a new line
```

Verbatim String Literal Best Practices

Do:

Use verbatim string literals when the string contains special characters such as backslashes

Use verbatim string literals to hold folder or file names

```
@"c:\mydir\myfile.txt";
```

Use two quotes to include quotes in a verbatim string literal

```
@"Say it with a long ""a"" sound";
```

Avoid:

Using verbatim string literals when there is no reason

```
@"Frodo";
```

Formatting Strings

```
var p = product.Category + "-" + product.SequenceNumber;
```

```
var p = String.Format("{0}-{1}",
                      product.Category,
                      product.SequenceNumber);
```

String.Format Best Practices

Do:

Use String.Format to insert the value of an expression into a string

Better technique with C# 6

Include a formatting string as needed

```
String.Format("Deliver by: {0:d}",
              deliveryBy);
```

Avoid:

Using String.Format when concatenating string literals

```
String.Format("Hello {0}",
              "World");
```

String Interpolation

```
var pc = String.Format("{0}-{1}",
                       product.Category,
                       product.SequenceNumber);
```

```
var pc = $"{product.Category}-{product.SequenceNumber}";
```

String Interpolation Best Practices

Do:

Use string interpolation instead of String.Format

Include a formatting string as needed

```
$"Deliver by: {deliverBy :d}";
```

Avoid:

If you are using Visual Studio 2013 or older

Building Long Strings

```
var orderText = "Order from Acme, Inc" + Environment.NewLine +
          "Product: " + productIdentifier + Environment.NewLine +
          "Quantity: " + quantity;
```

Building Long Strings

```
var orderText = "Order from Acme, Inc" + Environment.NewLine +
        "Product: " + productIdentifier + Environment.NewLine +
        "Quantity: " + quantity;
if (deliverBy.HasValue)
{
    orderText += System.Environment.NewLine +
                 "Deliver By: " + deliverBy.Value.ToString("d");
}
if (!String.IsNullOrWhiteSpace(instructions))
{
    orderText += System.Environment.NewLine +
                 "Instructions: " + instructions;
}
```

StringBuilder

- Conceptually a mutable string
- Allows string operations, such as concatenation, without creating a new string
- Provides methods for manipulating the mutable string
  - Append, Insert, Replace, etc
- Use ToString to convert to a string
- More efficient when working with strings that are
  - Built up with many separate concatenation operations
  - Changed a large number of times, such as within a loop

Long String Best Practices

Do:

Use StringBuilder when building up a string with numerous concatenation operations

Use StringBuilder when modifying a string numerous times

Such as in a loop

Consider readability

Avoid:

Using StringBuilder when only modifying a string a few times

Frequently Asked Questions

- What does it mean to say that C# strings are immutable?
  - It means that strings cannot be modified once they are created.
- Is a string a value type or a reference type?
  - A string is a reference type
  - That acts like a value type
- What is the best way to check for null strings?
  - It depends
  - Using String.IsNullOrWhiteSpace is great when checking nulls for a code block
  - Using the new C# 6 null-conditional operator is great for code statements

Frequently Asked Questions (cont)

- What are the benefits to using StringBuilder?
  - The .NET StringBuilder class is mutable, meaning that it can be readily changed.
  - Using StringBuilder is therefore more efficient when appending lots of strings.

This Module Covered

- Working with .NET String Methods
- Handling Nulls
- Defining Verbatim String Literals
- Formatting Strings
- Using String Interpolation
- Building Long Strings