.net core
&
asp.net core

Shalabayev Yerassyl Vladimirovich

Books

- Freeman A. - Pro ASP.NET Core MVC 2, Seventh Edition
- Jeffrey Richter - CLR via CSharp 4th Edition
- Robert Martin - Clean Code : A Handbook of Agile Software Craftsmanship

Overview

- Chars
- Strings
- Enums
- Arrays
- Delegates
- Nullable types
- ASP.NET CORE getting started

Chars, Strings, and Working with Text

- characters are always represented in 16-bit Unicode code values
- Char type also offers several static methods, such as **IsDigit**, **IsLetter**, **IsWhiteSpace**, **IsUpper**, **IsLower**, **IsPunctuation**, **IsLetterOrDigit**, **IsControl**, **IsNumber**, **IsSeparator**, **IsSurrogate**, **IsLowSurrogate**, **IsHighSurrogate**, and **IsSymbol**
- you can convert a single character to its lowercase or uppercase equivalent in a culture-agnostic way by calling the static **ToLowerInvariant** or **ToUpperInvariant** method

Strings

- String is primitive type
- String is reference type
- A String represents an **immutable** sequence of characters.
- Strings are reference types that behave in many ways like a value type. Assignment , comparison
- The String type also implements several interfaces (IComparable/IComparable<String>, ICloneable, IConvertible, IEnumerable/IEnumerable<Char>, and IEquatable<String>).

Strings

```
string s = "Original";

s = "Not " + s; // Not Original

if(s == "Not Original") // true

foreach(char character in s)
{
    Console.WriteLine(character);
}

Change(s);
```

Creates new string in a heap so there 2 strings

Concatenation

BAD

```
string s = "Original";

for (int i = 0; i < 100; i++)
{
    s = s + " one more";
}
```

GOOD

```
StringBuilder s =  new StringBuilder("Original");

for (int i = 0; i < 100; i++)
{
    s.Append( " one more");
}
```

Strings

Knows about the string at build time so creates ones and interns it in runtime

Returns new string and not changes **s**

```
string s = "Original"
s.ToLower();
if (object.ReferenceEquals(s, "Original"))
    Console.WriteLine("They have the same reference");
else
    Console.WriteLine("They do not have the same referenc

s = s.ToLower();
if (object.ReferenceEquals(s, "original"))
    Console.WriteLine("They have the same reference");
else
    Console.WriteLine("They do not have the same reference");


 if(s == "original")
     Console.WriteLine("They are equal");
 else
     Console.WriteLine("They are not equal");
```

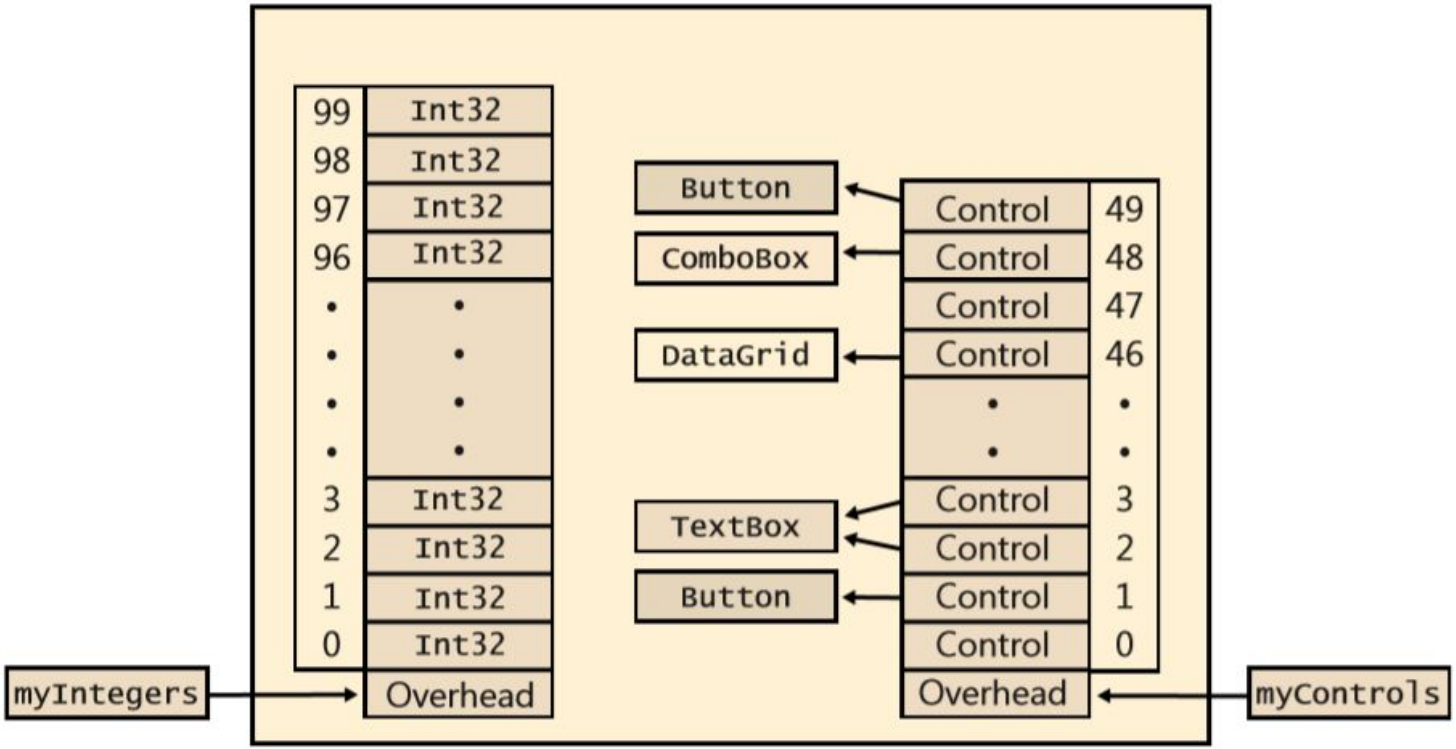
```
They have the same reference
They do not have the same reference
They are equal
```

Enumerated types

```
0 references
internal enum Color
{
    White,       // Assigned a value of 0
    Red,         // Assigned a value of 1
    Green,       // Assigned a value of 2
    Blue,        // Assigned a value of 3
    Orange       // Assigned a value of 4
}
```

Arrays

```
Int32[
myInte
...

Control[] myCo
myControls = n
```



All Arrays Are Implicitly Derived from System.Array

Clone, CopyTo, GetLength, GetLongLength, GetLowerBound, GetUpperBound, Length, Rank, and others

Static methods

AsReadOnly, BinarySearch, Clear, ConstrainedCopy, ConvertAll, Copy, Exists, Find, FindAll, FindIndex, FindLast, FindLastIndex, ForEach, IndexOf, LastIndexOf, Resize, Reverse, Sort, and TrueForAll

All Arrays Implicitly Implement IEnumerable, ICollection, and IList

Passing and Returning Arrays

Note that the Array.Copy method performs a shallow copy, and therefore, if the array’s elements are reference types, the new array refers to the already existing objects.

Delegates

In the same fashion that a `class` is a reference type that holds references to objects, `delegates` are also reference types, except they hold references to other `methods`.

Types of delegates

- The `Delegate` type
- The `Action` type
- The `Func` type

example

```
// Structure of a delegate

delegate <return.type> <name> (<type.parameter>)

// Example

delegate string Foo (int value);
```

```
0 references
void Main()
{
    Foo fooExample = First;
    fooExample += Second;

    string time = fooExample(DateTime.UtcNow);
    Console.WriteLine(time);
}

public delegate string Foo(DateTime time);
0 references
public string Bar(DateTime value)
{
    return value.ToString("t");
}

1 reference
public string First(DateTime value)
{
    return value.ToString("t");
}

1 reference
public string Second(DateTime value)
{
    return value.ToString("t");
}
```

Action and Func

The **Action** Delegate is a delegate which has a return type of void. The parameters of the `action` delegate are set using type parameters.

The **Func** Delegate is similar to the Action Delegate, the difference being that Func can never return `void`, it will always require at least one type argument. As mentioned earlier, the type argument specified last dictates the return type of the delegate.

Nullable value types

```
Int32? x = 5;
Int32? y = null;

if (y.HasValue)
{
    var z = y.Value;
}

var a = y.GetValueOrDefault();
```

Null-Coalescing Operator

C# has an operator called the null-coalescing operator (??), which takes two operands. If the operand on the left is not null, the operand's value is returned. If the operand on the left is null, the value of the right operand is returned

```
var z = y ?? 10;
```

ASP.NET CORE

Entry point

```
0 references
public class Program
{
    0 references | 0 exceptions
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    1 reference | 0 exceptions
    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>();
}
```

CreateDefaultBuilder

- Configures Kestrel server as the web server using the app's hosting configuration providers. For the Kestrel server's default options.
- Sets the content root to the path returned by Directory.GetCurrentDirectory.
- Loads host configuration from:
    - Environment variables prefixed with `ASPNETCORE_` (for example, `ASPNETCORE_ENVIRONMENT`).
    - Command-line arguments.
- Loads app configuration in the following order from:
    - *appsettings.json*.
    - *appsettings.{Environment}.json*.
    - Secret Manager when the app runs in the `Development` environment using the entry assembly.
    - Environment variables.
    - Command-line arguments.

CreateDefaultBuilder

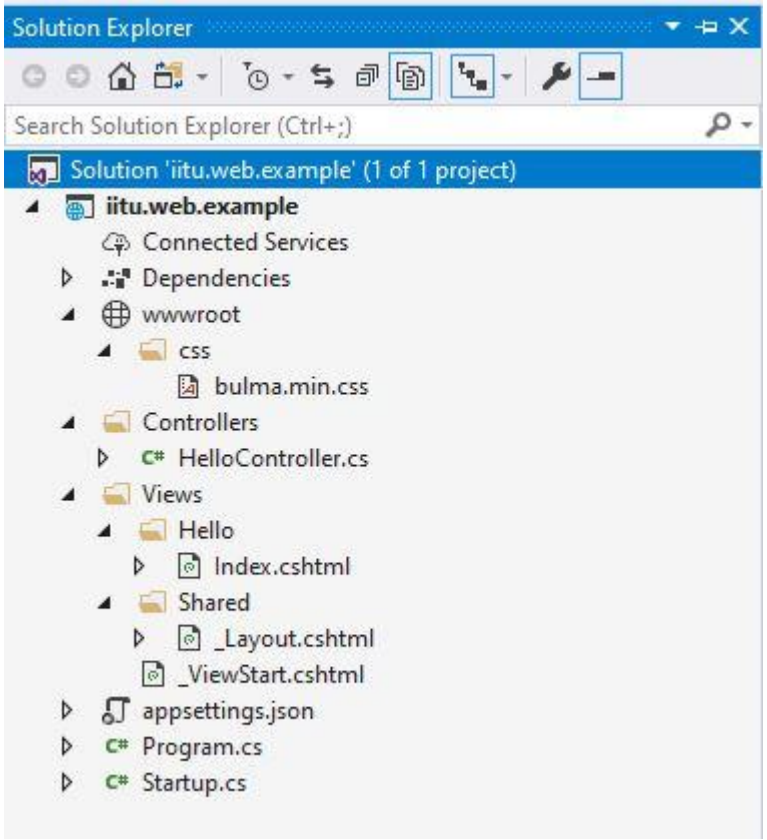
- Configures logging for console and debug output. Logging includes log filtering rules specified in a Logging configuration section of an *appsettings.json* or *appsettings.{Environment}.json* file.
- When running behind IIS with the ASP.NET Core Module, `CreateDefaultBuilder`enables IIS Integration, which configures the app's base address and port. IIS Integration also configures the app to capture startup errors. For the IIS default options, see Host ASP.NET Core on Windows with IIS.
- Sets ServiceProviderOptions.ValidateScopes to `true` if the app's environment is Development. For more information, see Scope validation.

Startup

The `Startup` class configures services and the app's request pipeline.

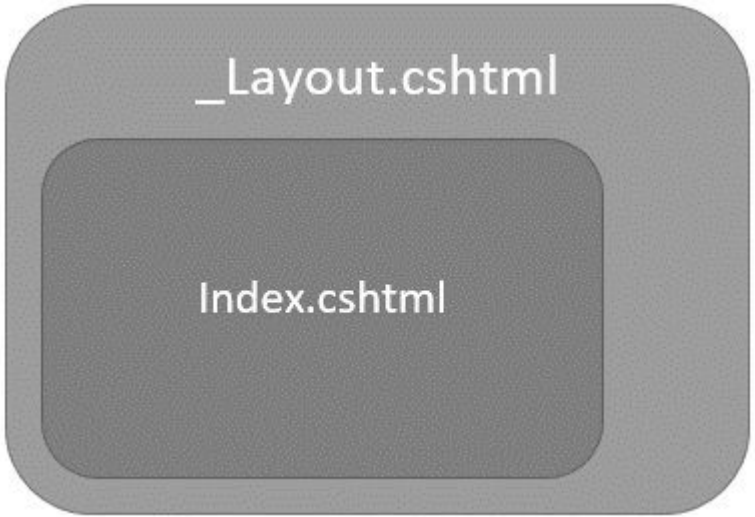
```
1 reference
public class Startup
{
    // This method gets called by the runtime. Use this method to add services to the container.
    // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
    0 references | 0 exceptions
    public void ConfigureServices(IServiceCollection services)
    {
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    0 references | 0 exceptions
    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        app.Run(async (context) =>
        {
            await context.Response.WriteAsync("Hello World!");
        });
    }
}
```

Basic MVC


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'iitu.web.example' (1 of 1 project)
iitu.web.example
Connected Services
Dependencies
wwwroot
css
bulma.min.css
Controllers
HelloController.cs
Views
Hello
Index.cshtml
Shared
_Layout.cshtml
_ViewStart.cshtml
appsettings.json
Program.cs
Startup.cs

Layout

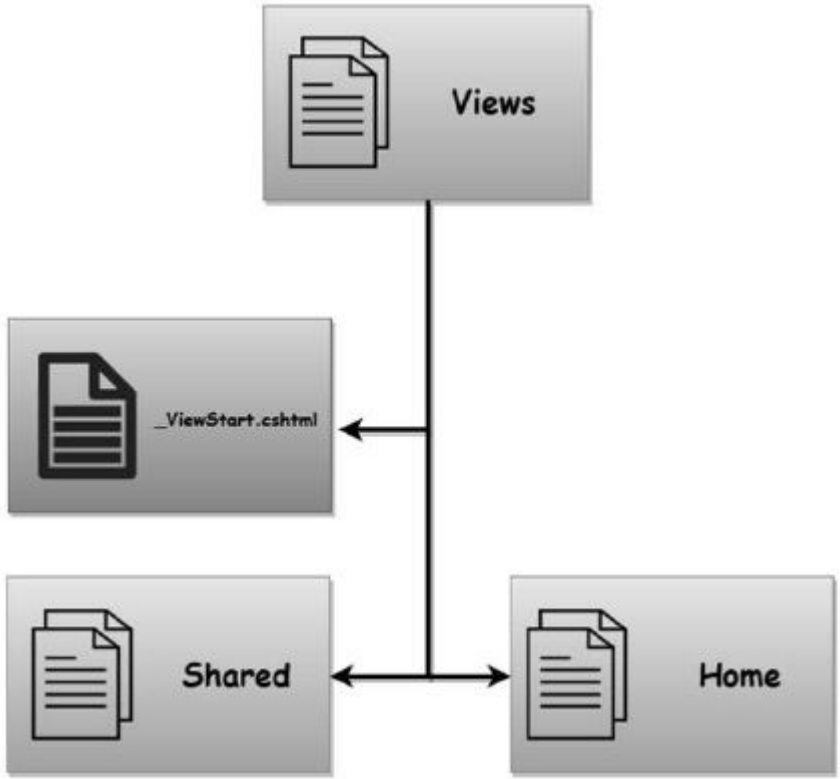
- You typically have a top area on every page where you display a logo and a navigational menu.
- You might also have a sidebar with additional links and information and probably a footer at the bottom of the page with some content.
- Every page of the application will want to have these common factors. Here, we make use of the Layout view to avoid duplication of factors in every page that we write.



_ViewStart

The Razor view engine in MVC has a convention where it will look for any file with the name _ViewStart.cshtml and execute the code inside this file. before executing the code inside an individual view.

- The code inside the ViewStart file cannot render into the HTML output of a page, but it can be used to remove duplicate code from the code blocks inside the individual views.
- In our example, if we want every view to use the Layout view that we have created in the last chapter, we could put the code to set the Layout view inside a ViewStart instead of having the code inside every view.



_ViewImports

In addition to the ViewS
there is also a ViewImp
that the MVC framework
for when rendering any vie

Like the ViewStart file,
drop ViewImports.cshtml
folder, and the ViewImp
can influence all the view
folder hierarchy