October 8 2015

With Kurt:

1. First find the view\_count\_instance of the dicks sporting good

select \* from view\_count\_instances order by client\_code

1. Get the count instance id from the query and store number from the dart web to display the store id

select store\_id,\* from view\_stores where count\_instance\_id = 17 and store\_number = 66666

1. After you get the store id from the above query run this query to find data that should be displayed in that particular dashlet.

select \* from view\_totals\_summary where store\_id = 75

1. How to Update the data in Sql

begin transaction

update item\_xref set xref\_barcode = '123aBc' where item\_xref\_id = 1

commit

1. How to Get the item\_xref:

Select top 100 \* from item\_xref

1. **How to test of the getItemsxref is case sensitive or no :**

Put a breakpoint in line 231 in correction repository, run the program

Select the Fanzz in the selection

Click the store no 66666

Modify inventory

Enter the store number 1

Add another Barcode

And move away the mouce and it should hit the break point

Do F10 and go through the list and if the item is already in the db it will return 1 and if it is not in the db it will return 0

**October 12, 2015 with Kurt:**

**Comparison:**

If it is a sql comparison it is case insensitive but if it is a C# or Linq comparision, it is case sensitive.

For Sql use this syntax: join ix in this.ImportViewItemXrefs on i.xref\_barcode.ToLower() equals ix.xref\_barcode.ToLower() into ixJoin

For LInq Syntax:

var viewItemXref = viewItemXrefs.FirstOrDefault(c => c.xref\_barcode.Equals(xrefBarcode, StringComparison.OrdinalIgnoreCase));

**Source Tree:**

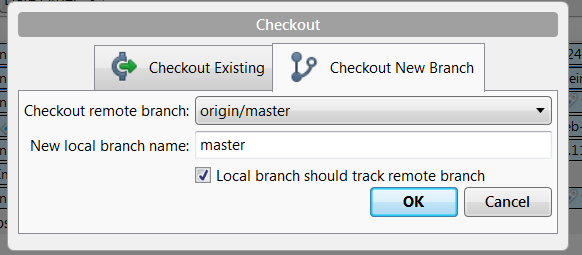
In order to create an feature you need to initialize it first:

**How to initialize the master branch:**

Expand the origin, under Remotes:

Right Click Master

Checkout origin/Master



Make sure it matches with the above settings

Click OK.

After this make sure to double click the develop branch and work on it.

**How to check out a branch?**

Right click the branch and click checkout. For eg. Right click release and click checkout.

**What to do if you started working but someone else already pushed changes and you are behind in git**

First commit your changes, then pull, which will merge others push with your commit

**How to pass command line parameters in a debug mode in Visual Studio?**

Right click the project and click properties

Click Build

Enter arguments in the “Command Line Arguments”

This way you do not have to enter any of these.

**How to make a C# Project to run on 32 bit or 64 bit solutions?**

Right click the project, properties and select the Build

Configuaration on the top left : select to All configuaration

Platform Target to 64.

Let’s say you have 5 different projects in your solutions, some are targeting 32 bit and others are targeting 64 bit, it will throw an error. You will need to change to all the projects .

**Stream:**

Read (byte[]buffer, int offset, int count)

Buffer: the array to store the read data

Offset: Buffer off set at which to start placing read data.

Count: represents the max no of bytes to read. Actually it may be less.

**How Reading works:**

The proper way to read data

First instantiate the data byte[] dataToLoad = new byte[stream.Length]

Int totalBytesRead = 0;

Int chunkbybytesread = 1;

While(totalBytesRead<dataToRead.Length && chunkBytesRead >0)

{

chunkBytesRead = stream.Read(dataToRead, totalBytesRead, dataToRead.Length-totalBytesRead);

totalBytesRead = totalBytesRead+chunkBytesRead;

return datToLoad;

}

**Using File Class to instantiate a stream : 3 ways**

**OpenRead:**

FileStream fs = File.openRead(@”c:myfiles\test.txt”

**If the file does not already exist:**

OpenWrite: Creates the file and returns a write-only stream

FileStream fs = File.OpenWrite(C:myfiles\data.txt”)

**If the file already exists:**

Create: truncates existing content

OpenWrite: leaves existing content and sets position to 0

Using Filestream constructors to instantiate the stream:

15 constructor overloads

FileStream fs = new FileStream(string path);

**FileMode:**

CreateNew - if a file already exists an exception is thrown

Create: creates a new file if a file already exists it is overwritten

Open : opens the existing file , if a file not found exception is thrown

OpenorCreate: Opens an existing file if the file does not exist, an exception is thrown

Truncate : open a file and deletes the file content. It is set to zero bytes

Append: Opens a file and sets the position to the end of the file, if a file does not exist, it is created.

**File Access:**

By default, File Stream will open a file in read and write access mode. With exception to the FileMode.Append option.

Read: File can only be read

Write : can only be written to

Read/Write: both allowed

**FileShare:**

Files gets locked by a FileStream until the stream is closed

Fileshare Enumeration can change this behavior

None: Sharing is not allowed by Default

Delete: Subsequest streams can delete the file

Inheritable: File handle can be inherited by child processes

Read: Stream can open file for reading only

Write: Other Streams are allowed to open for writing only

ReadWrite: other streams can open for both read and write

Be Careful when you allow other streams to open the file , it is different from the multiple threading .

**How to pass parameter via command line:**

-i "C:\Source\Dart.FixtureMessageDateChanger\Dart.FixtureMessageDateChangerCLI\bin\Debug\V2PublishFixtureMessage.json" -d "10/19/2018 12:00" -o "C:\Source\Dart.FixtureMessageDateChanger\Dart.FixtureMessageDateChangerCLI\bin\Debug\WriteFile.json" –w

**Why Command Line Parser ?**

Command Line parser is used to read arguments that is being passed via command line. There is a nuget library called CommandLineParser.

Few tricks :

For Boolean: You do not pass true or false . If you need to pass true, pass it like –w , -w is the option from the options class. If you want to pass false, leave the parameter blank. Make sure to annotate this value “Required=False” in the options class.

For String: If you pass – in front of a string for eg : -i “-test.txt”. It won’t go inside the if statement. It does not throw an exception, you should log the error in the catch block as below

if (CommandLine.Parser.Default.ParseArguments(args, options))

{// do something

}else {

Log.logger.Error(“Invalid format or missing the parameter”);

}

**How to implement serilog and configure it to write to the seq server ?**

1. First download Serilog, SerilogMetrics, Serilog seq and Serilog console via nuget package. If they are not available in VS go to the Nuget.org and get download them via package manager console.
2. On the Web program or Exe or class with Main method. You need to copy and paste right after the main method:

var environment = ConfigurationManager.AppSettings["Environment"] ?? "Missing";

Log.Logger = new LoggerConfiguration().ReadFrom.AppSettings()

                 .Enrich.WithMachineName()

                 .Enrich.WithProcessId()

                 .Enrich.WithThreadId()

                 .Enrich.FromLogContext() // Required for Context Logging as described below.

                 .Enrich.WithProperty("Environment", environment)

                 .WriteTo.LiterateConsole()

                 .CreateLogger();

1. Go to the app.config file and paste this inside the configuration :

<appSettings>

<add key="Environment" value="Dev"/>

<add key="serilog:minimum-level" value="Debug"/>

<add key="serilog:using" value="Serilog.Sinks.Seq" />

<add key="serilog:write-to:Seq.serverUrl" value="http://192.168.90.12" />

<add key="serilog:write-to:Seq.apiKey" value="api from seq server" />

</appSettings>

1. For the ApiKey , go to the seq server as admin ( username admin and password is :scanner\*1. Go to settings, Api key, Add Api key

Enter the Title and leave everything blank. Click Save Changes.

Go back and under the applied properties

AppName = Name of the app and click Add

Key = copy the token from top and paste it here and click add

1. Configuration is done.

For the Library. You need to install the Liblog instead.

Download liblog via nuget in vs

And initialize it :

private ILog logger = LogProvider.GetCurrentClassLogger();

You can use it by

try

{

   // some code

}

catch (Exception ex)

{

   this.logger.ErrorException("Error saving store {store\_id}", ex, model.storeId);

}

**Try Parse Vs Parse**

DateTimeOffset date;

if (DateTimeOffset.TryParse(options.ChangedDate, out date)) {

// do something here

}

else

{

Log.Logger.Error("Invalid Date format, should be mm/dd/yy hh:mm");

}

You should try to use tryparse instead of parse method.

Parse : Throws an exception if it cannot parse the value.

TryParse: Does not just try/catch internally, whole point is that it implements without an exception so that it is fast.

When to pick Parse vs TryParse: If you are sure that the value will be valid, use parse otherwise use TryParse

**Notes on Serilog:**

Minium Level:

Verbose: way too much detail, rarely enabled in production.

Debug: Used for internal system events, mostly to log when a method is called, helps to determine how something happened.

Information: describe thinkg happening in the system

Warning: When service degraded or endangered.

Error: Functionality is unavailable or expectations broken

Fatal: Most critical level, demand immediate attention

**Sample logging:**

Log.Warning("Disk quota {Quota} MB exceeded by {User}", quota, user);

Disk quota 1024 MB exceeded by "nblumhardt"

Assuming quota is an integer and user as in string. String is displayed in the quotes.

**Message Template Syntax:**

1. Property names are written between { and } brackets
2. Brackets can be escaped by doubling them {{ will be rendered as {
3. Formats that use numeric property like {0} and {1}, will be matched with the log methods parameters by treating the property name as indexes this is identical to string.format()’s behavior.

Don’t:

Log.Information(“The time is : “ + DateTime.Now);

Do:

Log.Information(“The time is {Now}”, DateTime.Now);

**Git Bash:**

First create a repo in github, go to your folder and right click and run git bash window

Git init

Git add .

Git commit –m “First Commit”

**(Only for the first time otherwise skip to push)**

Git remote add origin “https://github.com/padam10/test.git --

Git push –u origin master

*Sometime you get an error because the master may be ahead of you, if it is the case you need to pull first and push again.*

Git pull origin master – to pull

………………………………………………………..

Git diff - see what was changed

**GitFlow:**

First create a new feature

Make any changes to the code

Commit

Close the feature

Push to develop branch – be careful here.

**Asp.net Web API**

**Representation State Transfer(REST)**

Rest is a way of creating client server communication that is stateless.

**Difference between Model vs Viewmodel**

Check this link: <http://stackoverflow.com/questions/11064316/what-is-viewmodel-in-mvc>

A view model represents only the data that you want to display on your view/page, whether it be used for static text or for input values (like textboxes and dropdowns).

For example: You have an employee class

public class Employee : IEntity

{

public int Id { get; set; } // Employee's unique identifier

public string FirstName { get; set; } // Employee's first name

public string LastName { get; set; } // Employee's last name

public DateTime DateCreated { get; set; } // Date when employee was created

}

View models differs from domain models in that view models only contain the data that you are going display in the view. For example, you do not want the id to set from the view, it is probably auto generated by the Employee table. Also DateTime may be set in the store procedure.

public class CreateEmployeeViewModel

{

public string FirstName { get; set; }

public string LastName { get; set; }

}

**DbContext and Dbset:**

Normally DbContext corresponds to the Database and Dbset corresponds to the table.

You will be using DbContext object to get access to your tables and views(which will be represented by DbSet’s) and you will be using your DbSet’s to get access , create, update, delete and modify your table data.

Let’s say you have 10 tables in your database and your application works with 5 of them. Let’s call them Table1 to Table5. It makes sense to access it using a MyAppContext object where MyAppContext class is defined as below;

public class MyAppContext : DbContext

{

public MyAppContext () : ;

public DbSet<Table1> Table1 { get; set; }

public DbSet<Table2> Table2 { get; set; }

public DbSet<Table3> Table3 { get; set; }

public DbSet<Table4> Table4 { get; set; }

public DbSet<Table5> Table5 { get; set; }

}

public class Table1

{

public int Id {get; set;}

public string AStringField {get; set;}

//etc.

}

**Http Fundamentals:**

Uniform Resource Locator : <http://news.google.com> , <http://news.yahoo.com>

<http://food.com/receipe/grilled-cauliflower>

**http**: is the url scheme

**food.com**: host – tells which computer Ip address ( DNS server)

**/receipe/grilled-cauliflower**: Url path. It is like the file path – grilled cauliflower is more useful for search engine than it is for the file system.

IIS : Internet information Server: Web server that runs on windows machine

**Types of URLs:**

<http://food.com:80/receipes/squash>

80 is the port number, it is a default and so you do not need to include the port number in the url. If the server is listening other than the port 80, then you would need to include the port number at the url. Most commercial do not want to include the port number because it is hard to remember and long.

<http://bing.com/search?q=apple>

http: url scheme, bing.com: host, search : url path, q=apple : is a query or **query string.**

Web site can take this information and need to interpret. It is mostly name value pair.

<http://wikipedia.org/wiki/jabuticaba#Description>

http: url scheme, Wikipedia.org = host, wiki/jabutica: url path, **Description**: Fragment

Fragment is not processed by the server, it only used in the client.

**URL Encoding:**

Unsafe character should not show up in a url. For e.g. # or space. They may create error.

**Content Types:**

**Common MIME types**

**Type/Subtype Description**

Application/atom+xml Atom Feed

Application/json JSON Data

Image/gif GIF image

Image/png PNG image

Text/xml XML

Text/html HTML

Text/plain Just text

**Http Messages:**

Two types of Messages:

Http Request and Http Response

Standards are defined on what is sent and what is received so that the both client and server understand and full the request.

**HTTP Request :** Primarily Get and Post is used for HTML.

**Get – for Retrieve**

**Post : Update a resource -**

Put: Store a resource

Delete: Remove a resource

Head: Retrieve the header for a resource.

**Get :** is considered as safe method because you only retrieve the data from the server

**Post**: is considered as unsafe method for eg like cc process, send order etc. It may change the state.

Due to the nature of their request web server treats Gets and Post differently.

To avoid multiple post, Post/Redirect/Get is used extensively now a days.

@Request.QueryString[.

Searching is inheritably a safe operation

HttpPost is done through message so it is not so safe operation.

**Request Messages:**

[method] [URL] [Version]

[headers] – Generally contain server information. Like language acceptance etc. Date header. Header is optional but when it appears it must follow the protocol.

[body]

**Common Request Headers:**

**Header Description**

Refer The URL of the referring Page

User-Agent Information of the browser

Accept Preferred media types

Accept-Language Preferred Language

Cookie Cookie information

If-modified-Since Date of last retrievel

Date Create timestamp for the message

**Response Message:**

[version][status][reason]

[headers]

[body]

Status Code Categories:

Range Category

100 -199 Information;

200 -299 Successful

300-309 Redirections

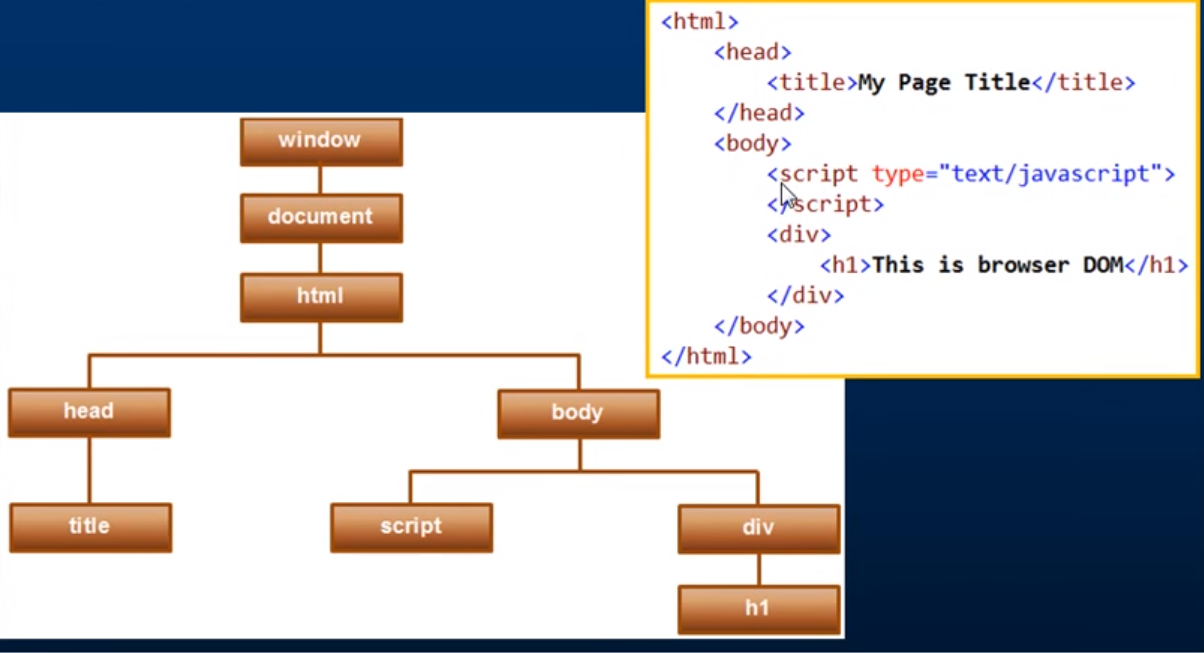
400-499 Client Error

500-599 ServerError.

Popular Status Codes

|  |  |  |
| --- | --- | --- |
| Code | Reason | Description |
| 200 | OK | Success |
| 301 | Moved Permanently | Resource moved don’t check again |
| 302 | Moved Temporarily | Resource moved check here again |
| 304 | Not modified | Resource has not changed since last retrieval |
| 400 | Bad Request | Bad Syntax |
| 403 | Forbidden | Refusal access |
| 404 | Not found | Resource do not exist |
| 500 | Internal Server Error | Something went wrong during processing |
| 503 | Service Unavailable | Server will not service the request, can not handle any more request |

DOM – Document Object Model



Javascript == vs ===

=== no type conversion is done

== type conversion is done.

4==”4” will return true

4===”4” will return false

**Nullables in C#:**

C# provides a special data types, the **Nullable** types to which you can assign normal range ofvalues as well as null values.

int? num1 = null;

int? num2 = 45;

Console.Writeline(“Values are {0}, {1}”,num1, num2);

Out put is : ,45.

**?? in C#(The Null Coalescing Operator)**

It is used with the nullable value types and reference types.

int? num1 = null;

int? num2 = 45;

int num3;

num3 =num1?? 5;

console.writeline(num3);

num3=num2?? 5

Output would be :

**Default Value Types in C#:**

bool false

int 0

byte 0