TechDays Script

# Setup

* Make sure console font is nice and big
* Copy the ef diagram & hello tech days demos to desktop
* Preload user voice sites
* Install Visual Studio 2010 SP 1
* Install Visual Studio 2012
* Start up zoom it
* Make sure chrome is installed
* Set VS to start with chrome
* Hide the document outline window

# Risk less VS

Show the hello techdays demo in VS 2010 & run it – then open in VS 2012 and then back to VS 2010.

Remember to talk about WPF designer improvements

# Culture Demo

Cold start VS 2012 – talk amount speed & colours

New console app – demo the format

1. var amount = string.Format("{0:C}",234.123);
2. Console.WriteLine(amount);
3. Console.ReadLine();

Configure the application to use all code analysis settings, Run analyse to generate the issue with it.

Add in the culture info

1. var amount = string.Format(new CultureInfo("en-us"), "{0:C}",234.123);

Talk about the annoying-ness of having this all the time – solved

1. CultureInfo.DefaultThreadCurrentCulture = new CultureInfo("en-us");

# CallerMemberName Demo

New WPF application and add some WPF

1. <TextBlock Text="{Binding Counter,FallbackValue=0}"/>

Then use the GUI tool to create a style, and then user the property window to adjust the style. Talk about the fact it is the blend UI tooling. Head to code and add:

1. this.DataContext = new Data();
2. /\* elided \*/
3. public class Data : INotifyPropertyChanged
4. {
5. private int counter;
6. public int Counter
7. {
8. get { return counter; }
9. set
10. {
11. counter = value;
12. RaisePropChanged("Counter");
13. }
14. }
15. public Data()
16. {
17. var timer = new DispatcherTimer();
18. timer.Interval = TimeSpan.FromSeconds(0.5);
19. timer.Tick += timer\_Tick;
20. timer.Start();
21. }
22. void timer\_Tick(object sender, EventArgs e)
23. {
24. Counter++;
25. }
26. private void RaisePropChanged(string propertyName)
27. {
28. if (PropertyChanged != null)
29. {
30. PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
31. }
32. }
33. public event PropertyChangedEventHandler PropertyChanged;
34. }

Change the binding to count, and the property but not the magic string – demo and fail.

Now fix it with caller info

1. RaisePropChanged();
2. }
3. }
4. private void RaisePropChanged([CallerMemberName]string propertyName = "")
5. {
6. if (PropertyChanged != null)
7. {
8. PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
9. }
10. }

# Workflow

Start off with a simple workflow console application and – talk & remove statement machine

Drag on a Writeline – note the C#

Drag a second note the sequence magic

Add annotations to the sequence – show the pin inline awesome ness too

Do a Ctrl+F search for text BOOM!

Great time to show quick launch – then document outline

Slide about versioning

# Async Demo

New WPF application and setup the XAML

1. <ProgressBar IsIndeterminate="True" HorizontalAlignment="Stretch" Height="50" VerticalAlignment="Top"/>
2. <Button Content="Button" HorizontalAlignment="Left" Margin="74,102,0,0" VerticalAlignment="Top" Width="75" Click="Button\_Click\_1"/>
3. <TextBlock x:Name="resultTextBlock" HorizontalAlignment="Left" Margin="284,93,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" FontSize="48"/>

Code behind is

1. private void Button\_Click\_1(object sender, RoutedEventArgs e)
2. {
3. var result = 0;
4. for (int i = 0; i < 500; i++)
5. {
6. Thread.Sleep(10);
7. result += i;
8. }
9. resultTextBlock.Text = result.ToString();
10. }

Demo it and show the freezing – sad panda.

Let’s do the processing in a thread… fuck

1. ThreadPool.QueueUserWorkItem(s =>
2. {
3. var result = 0;
4. for (int i = 0; i < 500; i++)
5. {
6. Thread.Sleep(10);
7. result += i;
8. }
9. resultTextBlock.Text = result.ToString();
10. });

Okay move the bits outside – double fuck

1. var result = 0;
2. ThreadPool.QueueUserWorkItem(s =>
3. {
4. for (int i = 0; i < 500; i++)
5. {
6. Thread.Sleep(10);
7. result += i;
8. }
9. });
10. resultTextBlock.Text = result.ToString();

Add in events & dispatcher

1. public MainWindow()
2. {
3. InitializeComponent();
4. OnCalculated += MainWindow\_OnCalculated;
5. }
6. void MainWindow\_OnCalculated(int obj)
7. {
8. Dispatcher.BeginInvoke(new Action(() =>
9. {
10. resultTextBlock.Text = obj.ToString();
11. }));
12. }
13. public event Action<int> OnCalculated;
14. private void Button\_Click\_1(object sender, RoutedEventArgs e)
15. {
16. ThreadPool.QueueUserWorkItem(s =>
17. {
18. var result = 0;
19. for (int i = 0; i < 500; i++)
20. {
21. Thread.Sleep(10);
22. result += i;
23. }
24. OnCalculated(result);
25. });
26. }

Okay about .NET 4 tasks – bloody messy!

1. Task.Factory.StartNew(() =>
2. {
3. var result = 0;
4. for (int i = 0; i < 500; i++)
5. {
6. Thread.Sleep(10);
7. result += i;
8. }
10. return result;
11. }).ContinueWith(t =>
12. {
13. resultTextBlock.Text = t.Result.ToString();
14. }, TaskScheduler.FromCurrentSynchronizationContext());

Async

1. var result = 0;
2. await Task.Factory.StartNew(() =>
3. {
4. for (int i = 0; i < 500; i++)
5. {
6. Thread.Sleep(10);
7. result += i;
8. }
9. });
11. resultTextBlock.Text = result.ToString();

Let’s see how this will get better with the new Task.Delay timeout

1. var result = 0;
2. var t = Task.Factory.StartNew(() =>
3. {
4. for (int i = 0; i < 500; i++)
5. {
6. Thread.Sleep(10);
7. result += i;
8. }
9. });
10. var r = await Task.WhenAny(t, Task.Delay(1000));
11. if (r == t)
12. {
13. resultTextBlock.Text = result.ToString();
14. }
15. else
16. {
17. resultTextBlock.Text = "time out";
18. }

Cancellation options – add a cancel button

1. private CancellationTokenSource cancelSource;
2. private async void Button\_Click\_1(object sender, RoutedEventArgs e)
3. {
4. this.cancelSource = new CancellationTokenSource();
5. var result = 0;
6. var t = Task.Factory.StartNew(() =>
7. {
8. for (int i = 0; i < 500; i++)
9. {
10. Thread.Sleep(10);
11. result += i;
12. }
13. });
14. var r = await Task.WhenAny(t, Task.Delay(1000, cancelSource.Token));
15. if (r == t)
16. {
17. resultTextBlock.Text = result.ToString();
18. }
19. else
20. {
21. if (r.Status == TaskStatus.Canceled)
22. {
23. resultTextBlock.Text = "cancelled";
24. }
25. else
26. {
27. resultTextBlock.Text = "time out";
28. }
29. }
30. }
31. private void Button\_Click\_2(object sender, RoutedEventArgs e)
32. {
33. cancelSource.Cancel();
34. }

Real world time – lets show the IO/Stream API async – add button & drop in prebuilt “old” way.

Change to 64bit mode

1. var stream = new MemoryStream();
2. using (var fileStream = new FileStream(@"C:\Users\Robert\Desktop\hello.txt", FileMode.Open, FileAccess.Read))
3. {
4. await fileStream.CopyToAsync(stream);
5. }
6. resultTextBlock.Text = stream.Length.ToString();

# WCF

New WCF Service – drop in service interface code. Implement it

1. public void Start()
2. {
3. var client = OperationContext.Current.GetCallbackChannel<ISocketServiceCallback>();
4. var random = new Random();
5. Task.Factory.StartNew(() =>
6. {
7. var initialPrice = 29d;
8. while (true)
9. {
10. client.Update(string.Format("MSFT: {0}", initialPrice));
11. Thread.Sleep(500);
12. initialPrice += random.Next(-1, 1) + random.NextDouble();
13. }
14. });
15. }

Right click config – tool is just there! switch to config, take about how clean it is (.NET 4), show new intellisense and add

1. <protocolMapping>
2. <clear/>
3. <add binding="netHttpBinding" scheme="http"/>
4. <add binding="netHttpsBinding" scheme="https" />
5. </protocolMapping>

Talk about web sockets and the new ntHttpBinding

Paste as XML demo

Run show the new singlewsdl bit.

Add console app, add reference, show it is web sockets in config, note how clean the config is– implement it

1. static void Main(string[] args)
2. {
3. var callback = new Callback();
4. var instance = new InstanceContext(callback);
5. var client = new ServiceReference1.SocketServiceClient(instance);
6. client.Start();
7. Console.ReadLine();
8. }
9. class Callback : ServiceReference1.ISocketServiceCallback
10. {
11. public void Update(string message)
12. {
13. Console.WriteLine(message);
14. }
15. }

# EF.

New console app

Nuget EF 5

Add code:

1. public class Person
2. {
3. public int Id { get; set; }
4. public string Name { get; set; }
5. public Gender Gender { get; set; }
6. public int Age { get; set; }
7. }
8. public enum Gender
9. {
10. Male
11. }
12. class MyDB : DbContext
13. {
14. public DbSet<Person> Persons { get; set; }
15. }
16. class Program
17. {
18. static void Main(string[] args)
19. {
20. using (var db = new MyDB())
21. {
22. db.Persons.Add(new Person()
23. {
24. Age = 30,
25. Gender = Gender.Male,
26. Name = "Robert"
27. });
28. db.SaveChanges();
29. }
30. Console.WriteLine("Done");
31. Console.ReadLine();
32. }
33. }

Run it works – show the DB in server explorer! Talk about localdb!

Change person to:

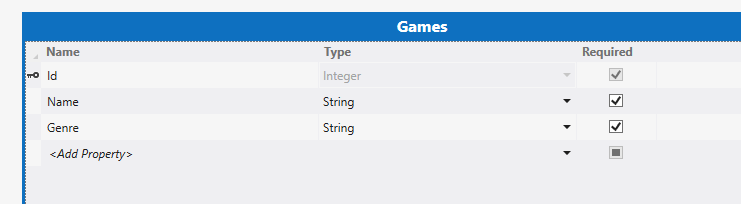
1. public class Person
2. {
3. public int PersonId { get; set; }
4. [Required]
5. public string Name { get; set; }
6. public Gender Gender { get; set; }
7. public DateTime DateOfBirth { get; set; }
8. }

Then Package manager enable migrations, run add-migration then update-database in package command windows

## Demo the EF diagram app

# Lightswitch

Create new table name it: games. Add a string for name, an int for agerestriction and a string for genre – make genre a choice list



Add a list & details screen for it – show it and add a game or two.

Show the new publish option

Show off the odata service

<http://localhost:5687/ApplicationData.svc/>

# ASP.NET MVC Time

NEW VISUAL STUDIO

New ASP MVC 4 project – select internet application. Take about the launcher – chome, ie, page inspector etc… Run in chrome and do the resize demo – responsive UI.

Switch to page inspector and talk about that – good point to show off the temp file in VS.

Show Nuget – installed, talk about ASP.NET is fully Nuget now, talk about updates, switch to online – include prerelease and pull down datajs!

In index.aspx kill the content and use the p snippet then use the tag match to change to table and add an id

1. <asp:Content ID="indexContent" ContentPlaceHolderID="MainContent" runat="server">
2. <table id="gameList">
4. </table>
5. </asp:Content>

Go to the script folder and right click, add javascript: games.js – drag on jquery and data. Show off javascript snippets, intellisense etc…

1. /// <reference path="jquery-1.7.1.js" />
2. /// <reference path="datajs-1.1.0-beta.js" />
3. function addRow(row, list) {
4. /// <summary>Adds a row</summary>
5. /// <param name="row" value="{Name:'',Genre:''}">Row from the data</param>
6. var tableRow = document.createElement("tr");
7. var titleCol = document.createElement("td");
8. var genreCol = document.createElement("td");
9. titleCol.innerText = row.Name;
10. genreCol.innerText = row.Genre;
11. tableRow.appendChild(titleCol);
12. tableRow.appendChild(genreCol);
13. list.append(tableRow);
14. }
15. $(document).ready(function () {
16. OData.read("http://localhost:26461/applicationdata.svc/GamesSet", function (data) {
17. $.each(data.results, function (index, row) {
18. addRow(row, $("#gameList"));
19. });
20. });
21. });

Finally back in index.aspx add the scripts block and demo in IE - ugly colours

1. <asp:Content ContentPlaceHolderID="ScriptsSection" runat="server">
2. <script src="../../Scripts/datajs-1.1.0-beta.js"></script>
3. <script src="../../Scripts/games.js"></script>
4. </asp:Content>

Open up the site.css – add a new style. Show off colour picker, show off support for both RGB & # use border-radius snippet tp add in radius. demo in IE. Also show network view (F12) and show that each script is being pulled down and talk about file size and HTTP overhead.

1. #gameList
2. {
3. background-color: #333;
4. color: rgb(234, 234, 234);
5. -moz-border-radius: 5px;
6. -webkit-border-radius: 5px;
7. border-radius: 5px;
8. }

Edit the BundleConfig.cs in App\_Start

1. public static void RegisterBundles(BundleCollection bundles)
2. {
3. BundleTable.EnableOptimizations = true;
4. bundles.Add(new ScriptBundle("~/bundles/index").Include(
5. "~/Scripts/datajs-1.1.0-beta.js",
6. "~/Scripts/games.js"));

Change the code to

1. <asp:Content ContentPlaceHolderID="ScriptsSection" runat="server">
2. <script src="/bundles/index"></script>
3. </asp:Content>

Demo and then show perf! SLIDES!

Add new controller – show we have MVC & API controllers and add a games controller

1. public IEnumerable<Game> Get()
2. {
3. return new[]
4. {
5. new Game(){
6. Name = "RPS",
7. Genre = "Lame",
8. },
9. new Game(){
10. Name = "XCom",
11. Genre = "Awesome"
12. }
13. };
14. }

The change the game.js code to

1. $(document).ready(function () {
2. $.getJSON("http://localhost:47328/api/games", function (data) {
3. $.each(data, function (index, row) {
4. addRow(row, $("#gameList"));
5. });
6. });