stonehenge

cross-platform desktop applications

written in pure C#

Frank Pfattheicher - .NET User Group Karlsruhe 24.10.2019

History of Desktop User Interface Techniques

- **1992** - MFC (Microsoft Foundation Classes)
  - C++
  - Modified Model-View-Controller-Architecture
- **2002** - WinForms
  - Comming with .NET Framework
  - Code Behind
- **2006** - WPF
  - Comming with .NET 3.0
  - MVVM (Model-View-ViewModel)

Other UI Techniques

Desktop
- Qt
- GTK

Web
- HTML5
- SVG

Mobile
- Xamarin-Forms
- Cordova PhoneGap

Other Projects

XAML

- Avalonia - http://avaloniaui.net/
- OmniGUI - https://github.com/OmniGUI/OmniGUI
- Uno Platform - https://platform.uno/

HTML

- Electron - https://electronjs.org/
- Electron.NET - https://github.com/ElectronNET/Electron.NET
- Blazor - https://dotnet.microsoft.com/apps/aspnet/web-apps/client

Electron.NET

Tooling

- dotnet tool install ElectronNET.CLI -g
- node.js v8.6.0
- npm install electron-builder –global

stonehenge Design Goals

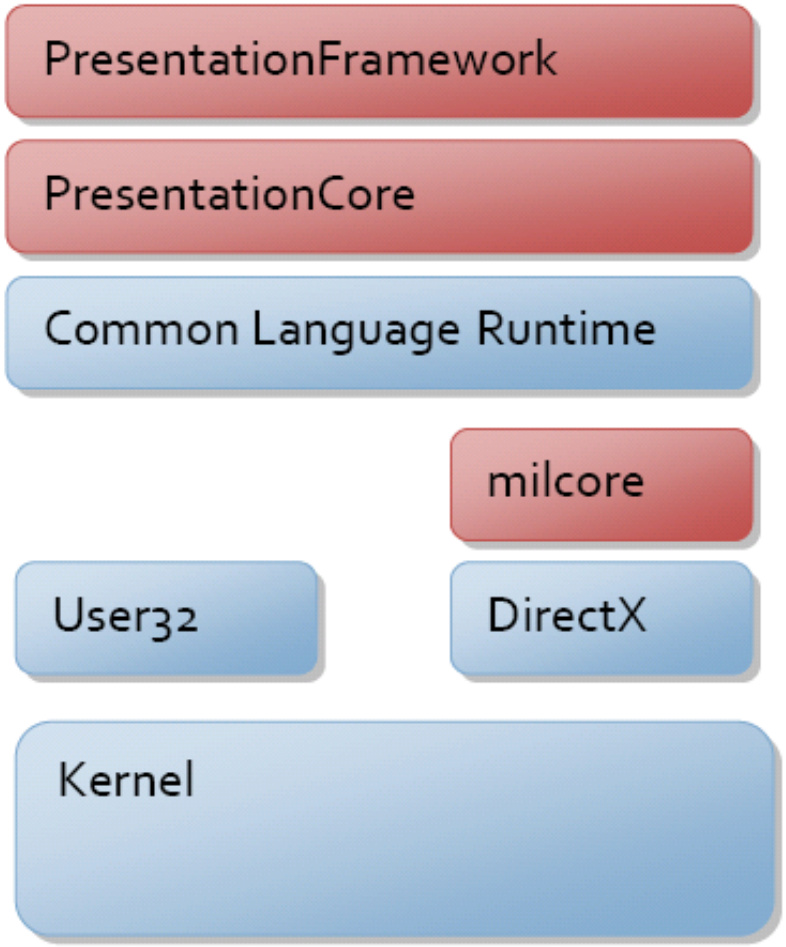
- All platforms supporting .NET Core
- Minimal dependencies
- No JavaScript tooling (Node) required
- No GRUNT, NPM, …
- No JavaScript usage for standard problems

Design Tradeoffs

- Server-side **state**
- Chatty
- No **IE11** support

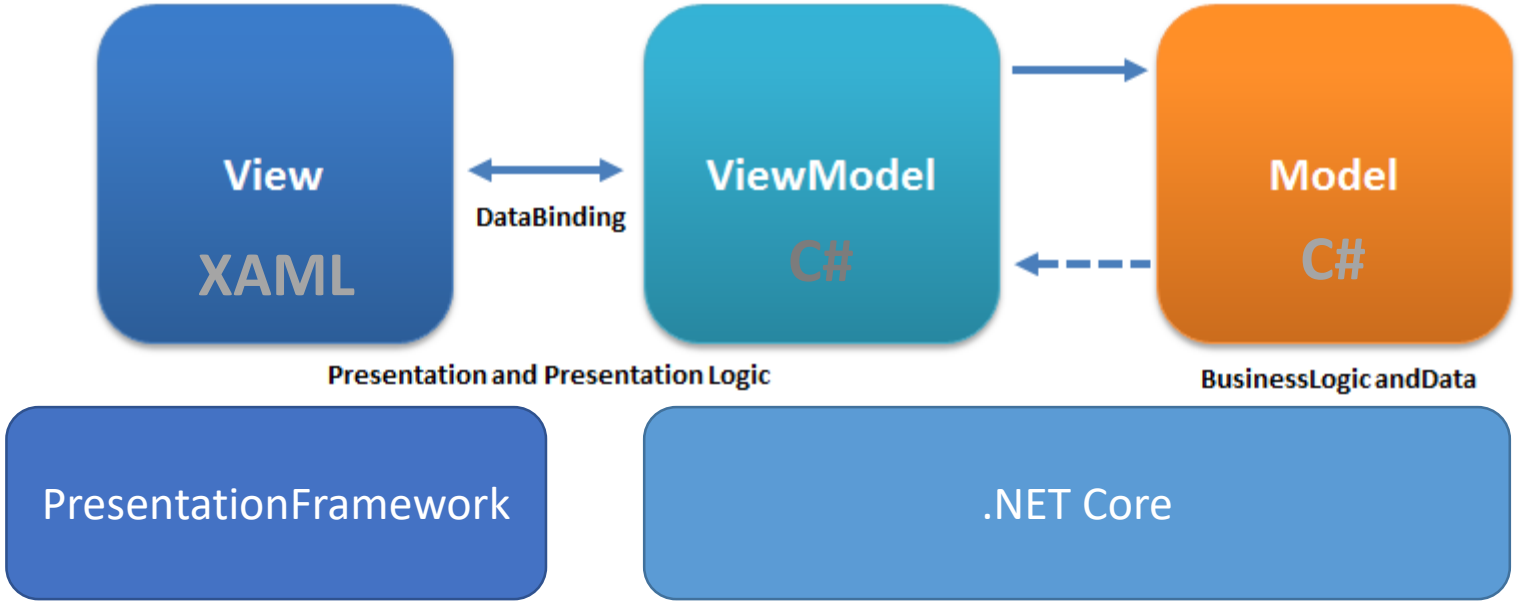
  (due to missing async/await support)

WPF



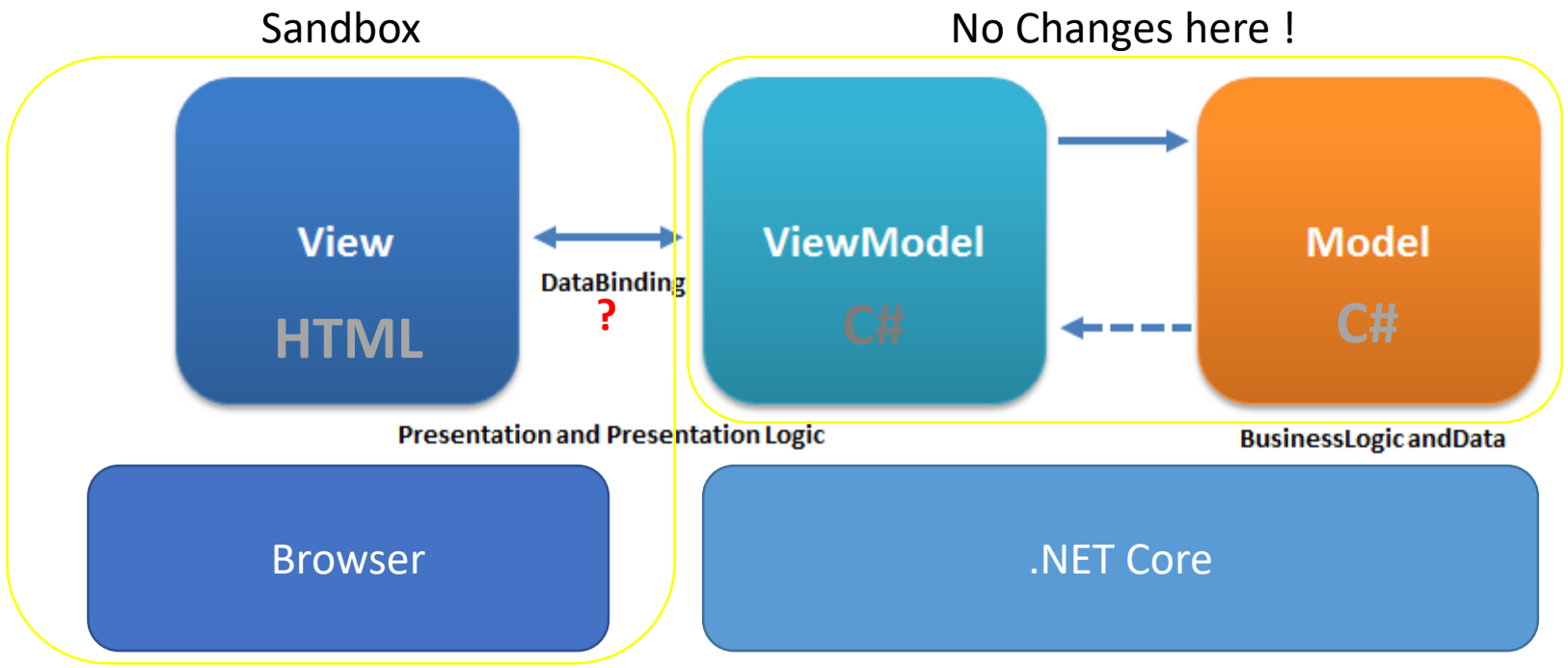
Source: Microsoft, 08.06.2019, https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/wpf-architecture

WPF - MVVM

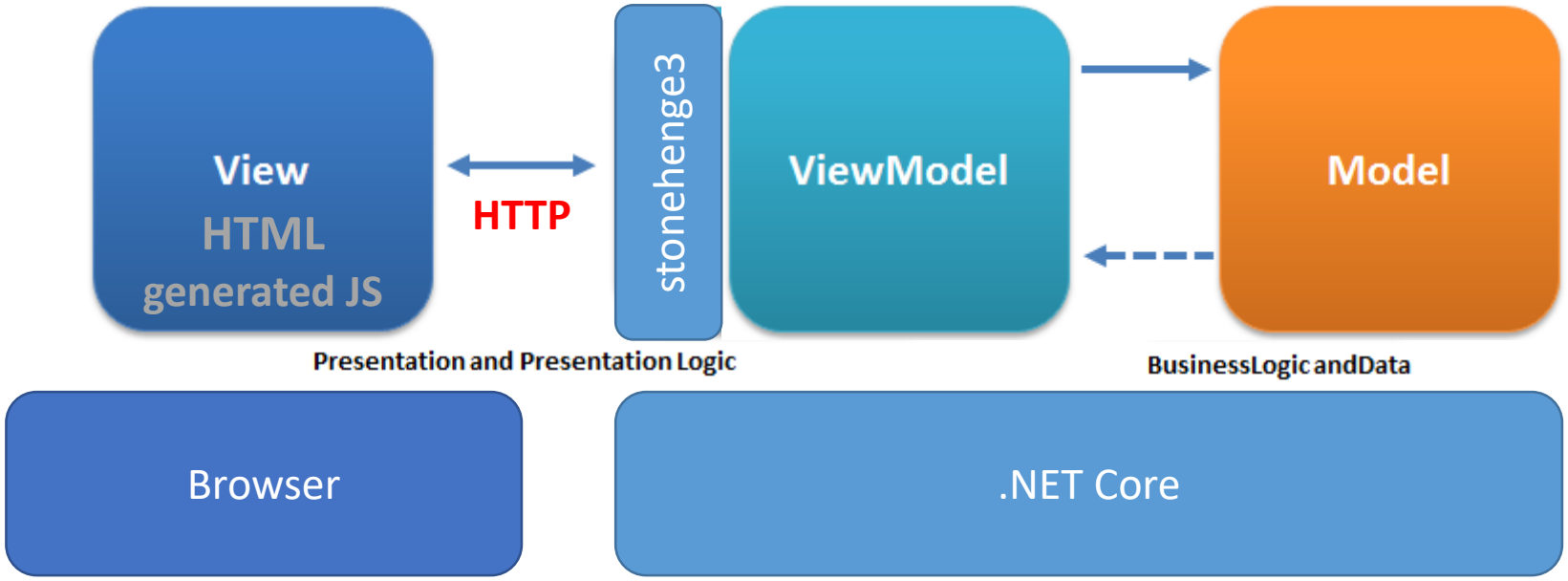


Source: Wikipedia, 08.06.2019, https://de.wikipedia.org/wiki/Model_View_ViewModel

Move to HTML


Sandbox
No Changes here !
View
HTML
DataBinding
?
ViewModel
C#
Model
C#
Presentation and Presentation Logic
BusinessLogic andData
Browser
.NET Core

Binding using client JS



Client - Vus.js, Bootstrap & Fontawesome

Vue.js (2.6)

- Used as client framework
- Client code is automatically generated from ViewModels
- Provides bindings and event handling
- Supports components
- Router support via vue-router

Bootstrap 4

Simple layout

Fontawesome 5

- Supports symbols

~~Theory~~

Practice

```
Console.WriteLine(@"Sample starting");
```

Getting Started

Create Project

- File - New Project - Visual C# - .NET Core - Console App (.NET Core)
- Choose .NET Core 2.x … 3.0 as target framework
- Add reference to **stonehenge3** Nuget package

```
Console.WriteLine(@"Sample starting");
```

Getting Started

Add Code to Main

```
var vue = new VueResourceProvider();
var provider = StonehengeResourceLoader.CreateDefaultLoader(vue);

var options = new StonehengeHostOptions { Title = "Demo", StartPage = "start" };

var host = new KestrelHost(provider, options);

host.Start("localhost", 32000);

Console.ReadLine();
```

```
public string ComputerName => Environment.MachineName;
```

Getting Started

Add content

Create a html file named **start.html** within the **app** solution folder

```
<div>
    <p>Hello Client from computer {{ComputerName}}!</p>
</div>
```

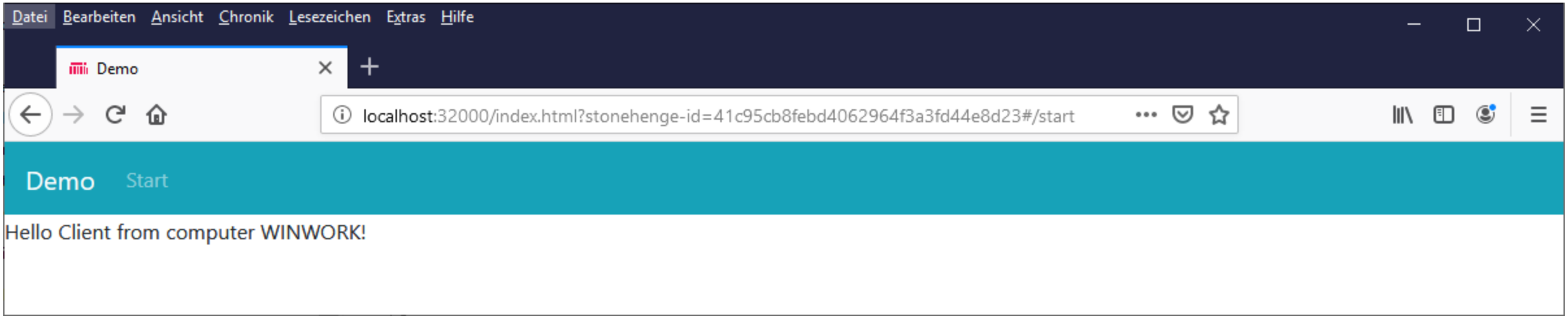
Mark start.html as **EmbeddedRessource**

Add corresponding server side ViewModel

Create class **StartVm** in **ViewModels** solution folder.

```
public class StartVm : ActiveViewModel
{
    public string ComputerName => Environment.MachineName;
}
```

Go!


Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
Demo
localhost:32000/index.html?stonehenge-id=41c95cb8febd4062964f3a3fd44e8d23#/start
Demo
Start
Hello Client from computer WINWORK!

```
Console.WriteLine(@"Sample starting");
```

Hosting Options

Create Host Window

```
var wnd = new HostWindow(host);
wnd.Open();
```

- Selects installed Browser
  (Chrome, Firefox, others)
- Start Browser in Kiosk Mode
  (no address line and toolbars)
- Navigates to App content

How does it work 1/3?

Startup

- Search all html files in /app folder => this are the pages
- Parse html for **Title** comment
  `<!–Title:PageTitle:PageSortIndex-->`
- Parse html for **ViewModel** comment
  `<!–ViewModel:ViewModelClassName-->`
  Default ViewModel class name is UpperCamelCase of filename + „Vm“
  Example: start.html => StartVm
- Parse html for **CustomElement** comment
  `<!–CustomElement:ListOfProperies-->`
  This is a component view with the given properties to bind data to
- Build list of routes for vue-router
- Register vue-components in application

How does it work 2/3?

Delivering index.html

- Search all css files in /styles and /themes folders
  Insert them at the placeholder of index.html
  `<!–stonehengeUserStylesheets-->`
- Search all js files in /scripts folder
  Insert them at the placeholder of index.html
  `<!–stonehengeUserScripts->`
- Create app.js including information collected at startup
  - Title
  - Routes

How does it work 3/3?

Delivering Page

- Deliver *page*.html
- Create corresponding Vue component client script *page*.js
- Check for existing *page*_user.js
  If found, add content at the end of generated *page*.js
- Serialize ViewModel data as JSON
  and send this data on request

```
GET /ViewModel/ViewModelClassName
```

Vue Bindings

Content

```
<p>Message: "{{ MessageText }}"</p>
```

Raw HTML

```
<div>{{{ RawHtml }}}</div>
```

Attributes

```
<div v-bind:class="ClassName"></div>
<a v-bind:href="TargetUrl"> ... </a>
```

Conditional

```
<h1 v-if="awesome">Vue is awesome!</h1>
```

Lists

```
<ul id="example-1">
    <li v-for="item in Items"> {{ item.message }} </li>
</ul>
```

Source: Vue.js, 26.06.2019, https://v1.vuejs.org/guide/syntax.html

Vue Advanced Bindings

Input

```
<input v-model="message1" placeholder="edit me">
<textarea v-model="message2" placeholder="add multiple lines"></textarea>
```

Checkbox

```
<input type="checkbox" id="checkbox" v-model="checked">
<label for="checkbox">{{ checked }}</label>
```

Radio

```
<input type="radio" id="one" value="One" v-model="picked"> <label for="one">One</label> <br>
<input type="radio" id="two" value="Two" v-model="picked"> <label for="two">Two</label> <br>
<span>Picked: {{ picked }}</span>
```

Select

```
<select v-model="selected">
        <option>A</option>
        <option>B</option>
</select>

<span>Selected: {{ selected }}</span>
```

Source: Vue.js, 26.06.2019, https://v1.vuejs.org/guide/syntax.html

Vue Event Bindings

Click

```
<button v-on:click="AddOne()">Add 1</button>
```

Keyboard

```
<input v-on:keyup.page-down="OnPageDown()">
```

Source: Vue.js, 26.06.2019, https://v1.vuejs.org/guide/syntax.html

Style Options

Index.html could be replaced

Basic bootstrap style is incliuded with stonehenge

Client UI Choices

General change in UI is done by customizing **index.html**

Default

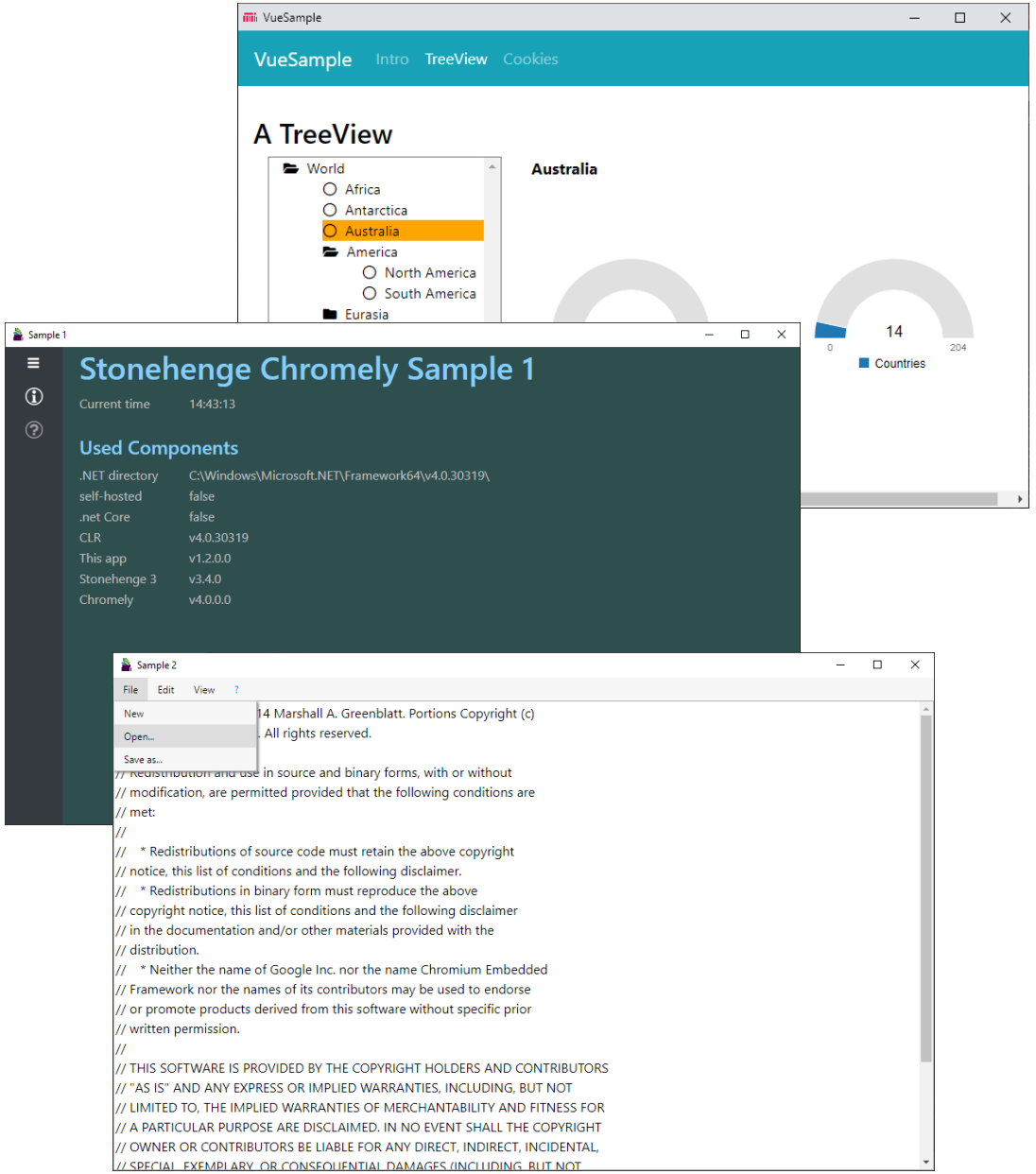
- Bootstrap SPA menu

VsCode Style

- Alternate menu style

Classic Desktop

- Classic Menu support using CSS



Still a Browser required

Default Support

- Automatic browser detection
- Start in kiosk mode

Missing Functionality

- System Dialogs (File Open/Save, MessageBox)

Pause

Questions?

Chromely

cross-platform desktop applications

written in pure C#

Frank Pfattheicher - .NET User Group Karlsruhe 24.10.2019

Browser replaced by CEF

- Uses Chromium Embedded Framework (CEF)
  A simple framework for embedding Chromium-based browsers in other applications.
- Lightweight alternative to Electron.NET, Electron for .NET Core
- Based on Xilium.CefGlue, CefSharp implementations of embedded Chromium (CEF) **without WinForms** or **WPF**
- Uses Windows and Linux native GUI API as "thin" chromium hosts

Source: Chromium Embedded Framework, 26.06.2019, https://bitbucket.org/chromiumembedded/cef
Source: chromelyapps, 26.06.2019, https://github.com/chromelyapps/Chromely

Supported Platforms

Platform	CefGlue.Winapi	CefGlue.Gtk	CefSharp.Winapi
Windows[1] 32-bit	**net461, netstandard2.0**	net461, netstandard2.0	net461
Windows[1] 64-bit	**net461, netstandard2.0**	---	---
Linux[2] 32-bit	---	**netstandard2.0**	---
Linux[2] 64-bit	---	**netstandard2.0**	---
MacOSX[3] 64-bit	---	---	---
Linux ARM[4]	---	**netstandard2.0**	---

[1] Windows 7, Service Pack 1 and newer
[2] Ubuntu 16.04 and newer (Mono currently not working, window resizing not working)
[3] Work in progress...
[4] i.e. Raspberry Pi 3+ (manual download of CEF builds for ARM available on http://chromely.org/cefbuilds/index.html)
For more info/documentation, please check Chromely wiki.

Source: chromelyapps, 24.10.2019, https://github.com/chromelyapps/Chromely

Simple to Use

- Add Nuget Packages
  - Chromely.Core
  - Chromely.CefGlue
  - Chromely.CefGlue.Winapi
  - Chromely.CefGlue.Gtk

- Add some Hosting Code…

Simple to Use

```
class Program
{
    static void Main(string[] args)
    {
        var startUrl = "https://google.com";
        var config = ChromelyConfiguration .Create()
                .WithHostMode(WindowState.Normal)
                .WithHostTitle("chromely")
                .WithHostIconFile("chromely.ico")
                .WithAppArgs(args)
                .WithHostBounds(1000, 600)
                .WithStartUrl(startUrl);
        using var window = ChromelyWindow.Create(config);
        window.Run(args);
}
```

Source: chromelyapps, 26.06.2019, https://github.com/chromelyapps/Chromely

Bringing it togeter

```
class Program
{
   static void Main(string[] args)
   {
      // stonehenge Backend
      var provider = StonehengeResourceLoader .CreateDefaultLoader(new VueResourceProvider());
      var options = new StonehengeHostOptions { Title = "Demo", StartPage = "start", };
      var host = new KestrelHost(provider, options);

      if (!host.Start("localhost", 32000))
      {
         Console.WriteLine(@"Failed to start server on: " + server.BaseUrl);
         Environment.Exit(1);
      }


      // Chromely Frontend
       var config = ChromelyConfiguration .Create()
                    .WithHostMode(WindowState.Normal)
                    .WithHostTitle("chromely")
                    .WithHostIconFile("chromely.ico")
                    .WithAppArgs(args)
                    .WithHostBounds(1000, 600)
                    .WithStartUrl(host.BaseUrl);

       using var window = ChromelyWindow.Create(config);
       window.Run(args);
}
```

Known Problems

- Debugging
  - .NET Core 2.x does not build an EXE by default – publish required
    - No debugging possible
- GTK Window Handling (Linux)
  - window resizing does not work
  - tooltips at wrong position

Missing for Desktop Applications

- System Dialogs
  - MessageBox
  - FileOpen
  - FileSave
  - SelectFolder

UI Components

- HTML5 provides simple controls only
  - label
  - input
  - select / option / optgroup
  - textarea
  - button
  - datalist
    (drop-down list of the pre-defined options as they input data)

- Often this is not enough…

UI Components (Vue)

- Vue requires to register a component with
  - Tag-name
  - List of properties (data)
  - Template (HTML presentation)

Vue Components - View

Define a View as „tree-view.html“, define rootnodes as parameter

```
<!--CustomElement:rootnodes-->
<div>
   <ul style="…" >
        <tree-node v-for="child of rootnodes" :node="child" :key="child.Id„
                   @select="$emit('select', $event)"
                   @toggle="$emit('toggle', $event)"
   </ul>
</div>
```

The file name is used as custom HTML5 tag, so you can now use **tree-view** as a element.

- **The filename defines the HTML tag-name**
- **The CustomElement comment is the hint for stonehenge to create a Vue component.**
- **Names following the colon are the names of variables the component should expose.**
  (comma sepearated list)

Vue Components - View

Define a second View as „tree-node.html“, define node as parameter

```
<!--CustomElement:node-->
<div>
   <li v-if="node.IsVisible" >
      <div :class="node.Class" >

         <span v-if="node.HasChildren" :class="node.Icon" style="cursor: pointer;"
                  @click="$emit('toggle', node.Id)"></span>

         <span v-else class="far fa-circle" @click="$emit('toggle', node.Id)"></span>

         <span style="cursor: pointer;" @click="$emit('select', node.Id)"> {{node.Name}}</span>

      </div>

      <ul style="list-style-type: none;">

          <tree-node v-for="child of node.Children" :node="child" :key="child.Id"
                     @select="$emit('select', $event)" @toggle="$emit('toggle', $event)"/>

      </ul>

    </li>
</div>
```

Vue Components – Model

Define the ViewModel as „TreeNode.cs“

```
public class TreeNode
{
    public string Id { get; }
    public string Name { get; }

    public List<TreeNode> Children { get; set; }

    public bool IsVisible => _parent?.IsExpanded ?? true;

    public bool IsExpanded { get; set; }
    public bool IsSelected { get; set; }
    public bool HasChildren => Children.Count > 0;

    public string Icon => IsExpanded ? "fa fa-folder-open" : "fa fa-folder";
    public string Class => IsSelected ? "tree-selected" : "";

    private readonly TreeNode _parent;

    public TreeNode(TreeNode parentNode, string name)
    {
        Id = Guid.NewGuid().ToString("N");
        Name = name;
        Children = new List<TreeNode>();
        _parent = parentNode;
    }

    public IEnumerable<TreeNode> AllNodes()
    {
        yield return this;
        foreach (var node in Children.SelectMany(child => child.AllNodes()))
        {
            yield return node;
        }
    }

    public TreeNode FindNodeById(string id)
    {
        return AllNodes().FirstOrDefault(node => node.Id == id);
    }
}
```

Vue Components - Usage

In the View simply use the „tree-view“ tag and bind rootnodes.

```
<div>
        <h3>A TreeView</h3>
        <tree-view v-bind:rootnodes="RootNodes"
                   v-on:select="TreeSelect($event)"
                   v-on:toggle="TreeToggle($event)"/>
</div>
```

In the ViewModel provide RootNodes and handle select and toggle.

```
public List<TreeNode> RootNodes => new List<TreeNode>();

[ActionMethod]
public void TreeToggle(string nodeId)
{
    var node = RootNodes[0].FindNodeById(nodeId);
    if (node == null) return;
    node.IsExpanded = !node.IsExpanded;
}
[ActionMethod]
public void TreeSelect(string nodeId)
{ … }
```

Done.

Questions?