31.8.–3.9.2015
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

RamphastosUI

Eine HTML5-Bibliothek für C#

Alexander S. Timmermans

matchmycolor GmbH

RamphastosUI

Eine HTML5-Bibliothek für C#

Entwickelt von der matchmycolor GmbH

Hinweis

Es wird ein Ergebnis aus Forschung und Entwicklung vorgestellt. Bei der präsentierten Programmbibliothek handelt es sich um einen Prototyp.

Zusammenfassung

RamphastosUI ermöglicht es in .Net Applikationen HTML5 als User Interface-Technologie zu verwenden.

RamphastosUI setzt HTML5 an die Stelle von UI Technologien wie WinForms, GTK, QT oder WPF.

RamphastosUI-Applikationen laufen sowohl lokal als native Anwendungen als auch als Web-Applikationen.

Ein Statement

“HTML ist die wahrscheinlich fortschrittlichste User Interface-Technologie in unserem Sonnensystem.”

Inhalt

1. Motivation
2. Theorie
3. Konzepte
4. Demo
5. Retrospektive
6. Aktueller Status
7. Vergleichbare Technologien
8. Ausblick

Motivation

Herbst 2012:

“Wie wäre es, das User Interface mit HTML5 zu entwickeln, sodass die Applikation sowohl als Rich-Client Desktop aber auch als Web-Applikation im Browser ausgeführt werden kann.”

- Natives und Web Frontend

  einmal entwickeln und zweimal verwenden
- Einfaches Skinning mit CSS

  einfacher als mit WPF, WinForms, QT, etc.
- Multi-Plattform Unterstützung ermöglichen

  HTML5 läuft auf allen wichtigen Plattformen

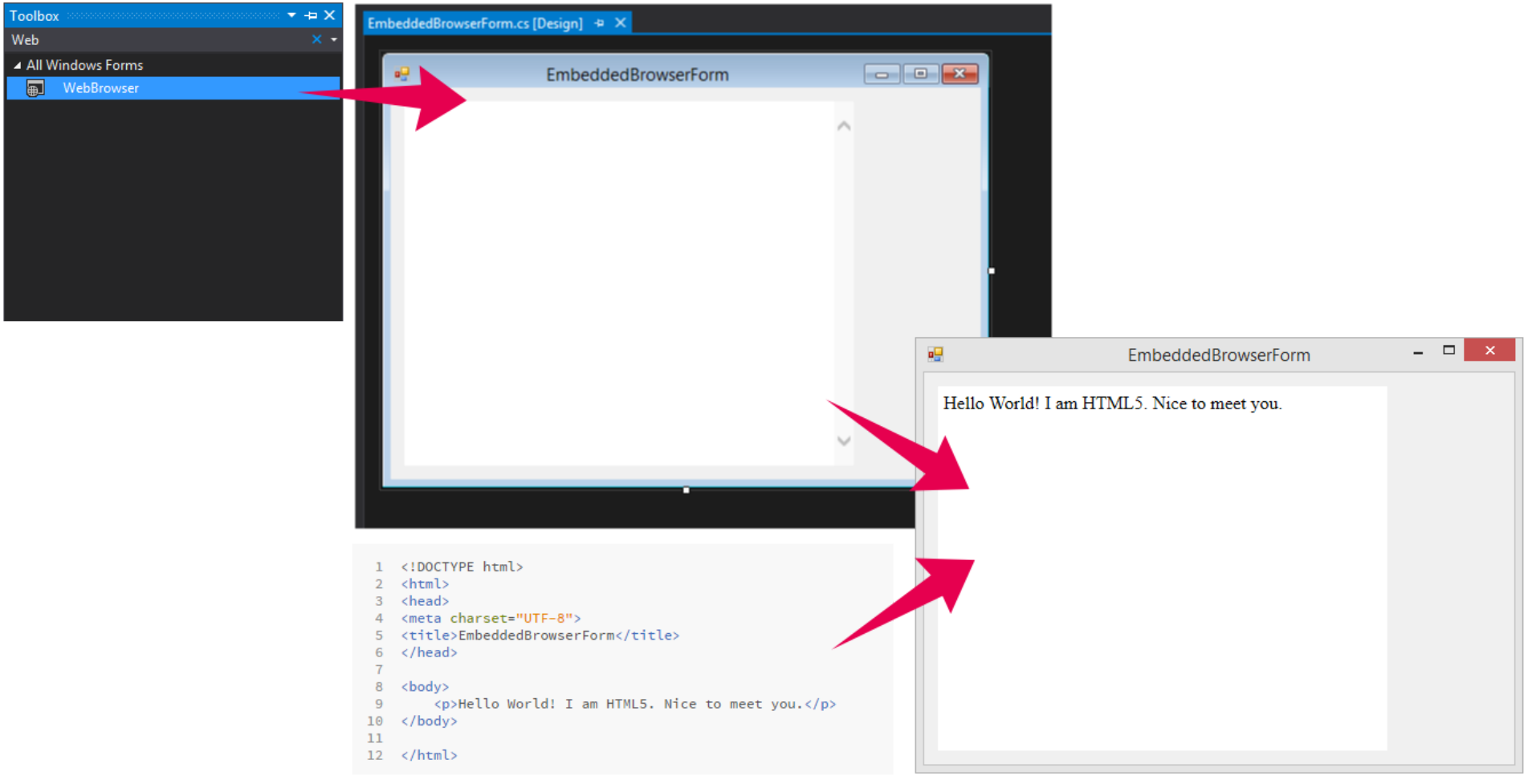
Theorie

1. Embedded Browser
2. MVC Pattern
3. MVVM Pattern
4. View Driven MVVM
5. View-Model Driven MVVM

Embedded Browser

Man spricht von Embedded Browser (eingebetteter Browser) wenn eine *Web Browser Engine* als Control bzw. Ui-Element in eine Applikation eingebettet wird. In diesem Control kann nun HTML dargestellt werden. Ein Beispiel dafür ist der WebView Control von WinForms.

Embedded Browser



Beispiel eines WebBrowser Controls

Embedded Browser

Embedded Browser-Technologien:

- WebBrowser Control
- Awesomium
- CEF (Chromium Embedded Framework)
  - CefGlue
  - CefSharp
  - ChromiumFX
- GeckoFX

Embedded Browser

CEF - Chromium Embedded Framework

Open Source Framework für eingebettete Web Browser basierend auf Chromium.

Eine unvollständige Liste von Applikation welche auf CEF aufbauen:

Spotify
Adobe Dreamweaver CC
Adobe Brackets
Amazon Music
Evernote

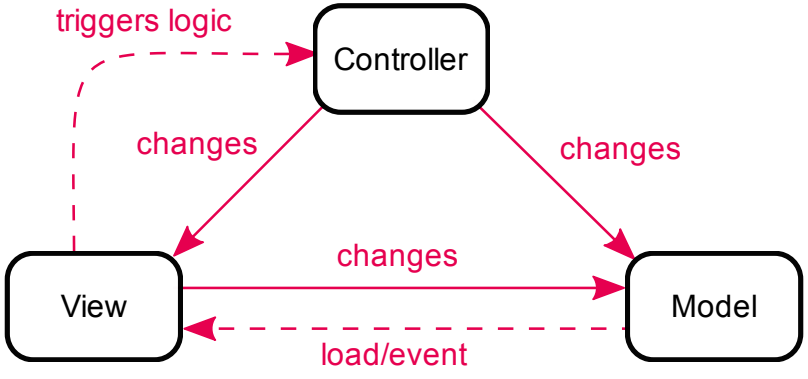
MVC Pattern

MVC (Model-View-Controller) ist ein Architekturmuster zur Strukturierung von Software. Die Ziele sind Flexibilität und Entkopplung.

Das Model enthält die Daten
Die View enthält Präsentationslogik
Der Controller verwaltet die View und verarbeitet Benutzeraktionen.

Es ist gibt keine allgemeingültige Definition wo die Geschäftslogik untergebracht sein sollte. Heute ist sie meist im Model implementiert. Früher war sie oft im Controller zu finden.

MVC Pattern


triggers logic
Controller
changes
changes
changes
View
Model
load/event

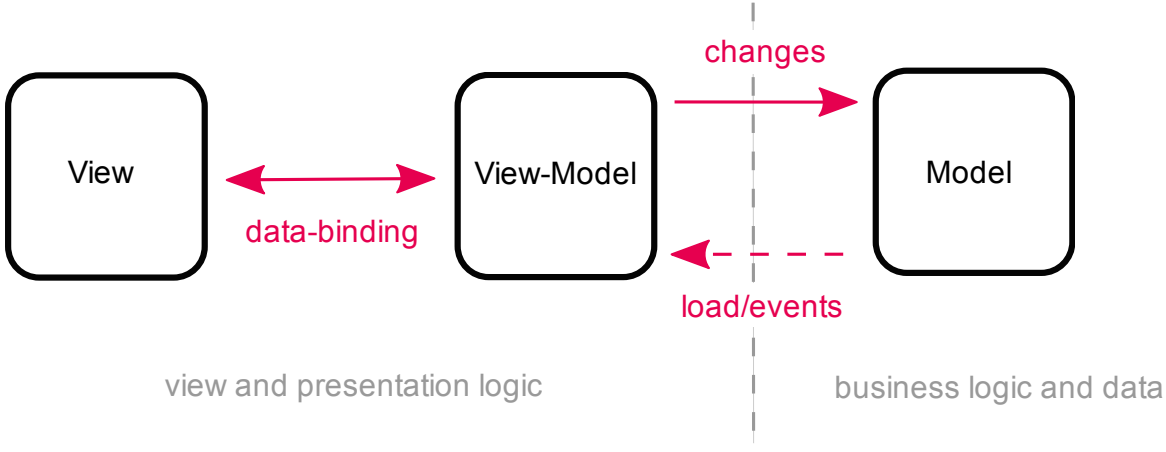
MVVM Pattern

MVVM (Mode-View-ViewModel) ist ein Architekturmuster entwickelt von Microsoft, um event-getriebene Programmierung zu vereinfachen. Es ist eine Variation des *Presentation Model* Design Patterns von Martin Fowler. Ein Kernprinzip sind *Data-Bindings*.

Verwendung: WPF, Sliverlight, Windows Store Apps, KnockoutJS, AngularJS

Das Model enthält die Daten
Die View enthält Präsentationslogik
Das View-Model ist eine Schnittstelle für die View, verbindet also View und Model.

MVVM Pattern



- Die View enthält nur View-spezifische Logik und das *Data Binding*
- Das View-Model stellt Daten bereit für das View-Model (z.B. Model Properties oder Farbe einer Schaltfläche). Enthält Triggers zum Ausführen von Applikationslogik (z.B. mit Commands).
- Das Model enthält die Businessdaten

MVVM Pattern

Regel: Es gibt keine direkte Verbindung zwischen Model und View.

Dadurch ist die Entwicklung von View und Applikationslogik separiert.

MVVM Pattern

Wie funktionieren event-getriebene Updates in C#? Mit INotifyPropertyChanged und INotifyCollectionChanged!

```
//Notifies that a property value has changed.
public interface INotifyPropertyChanged
{
    //Gets fired when a property value has changed
    event PropertyChangedEventHandler PropertyChanged;
}
```

```
//Notifies listeners of dynamic changes,
//such as when items get added and removed.
public interface INotifyCollectionChanged
{
    //Gets fired when the collection changes.
    event NotifyCollectionChangedEventHandler CollectionChanged;
}
```

Die Art der Änderung wird jeweils durch das Event Argument beschrieben.

View-Driven MVVM

Die Applikation wird *von der View getrieben*. Die Applikation startet mit einer View und ein dazugehöriges View-Model wird als Data-Source hinzugefügt. Um von einer View in eine andere zu wechseln wird eine neue View instanziiert und das korrespondierende View-Model geladen und hinzugefügt.

Die Navigation ist Teil der View

View-Model-Driven MVVM

Die Applikation startet mit einem Basis View-Model. Die korrespondierende View (definiert in einer DataTemplate-Registrierung) wird instanziiert und stellt das View-Model dar. Bei der Navigation zu anderen Views wird einfach ein anderes View-Model instanziiert wobei die dazugehörige View automatisch geladen wird.

Die Navigation ist Teil des View-Models

Vergleich View Driven / View-Model Driven

	View Driven	**View-Model Driven**
Navigation	Transition von Views (View lädt nächste View)	Transition von View-Models (View-Model lädt nächstes View-Model)
Kopplung (View / View-Model)	View (View lädt View-Model)	DataTemplate Registrierung
Unit-Test	ohne Navigation	mit Navigation (Übergang zwischen View-Models)

Referenzen

Introduction to Model/View/ViewModel pattern for building WPF apps - John Gossman

MVVM in the Real World - Bryan Anderson

Presentation Model - Martin Fowler

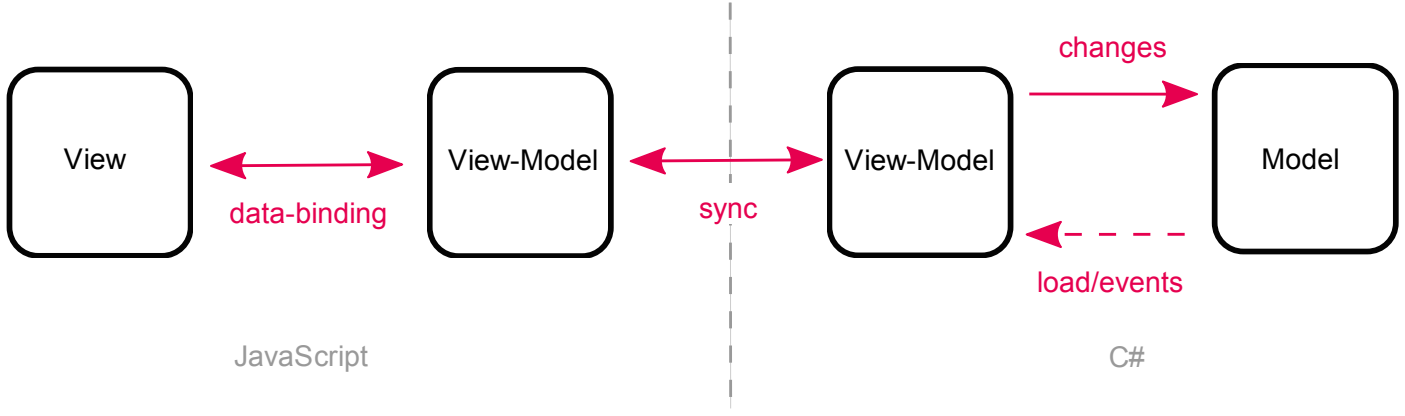
Konzepte

1. Design Pattern
2. Embedded Browser Technologie
3. Third Party Frameworks

Design Pattern

Gewählt wurde *View-Model Driven MVVM*, da sich auf diese Weise die Applikation komplett ohne View modellieren lässt. Unit Tests greifen somit auf einer höheren Ebene. Die Entwicklung vom User Interface ist separiert. Views können zentral ersetzt werden.

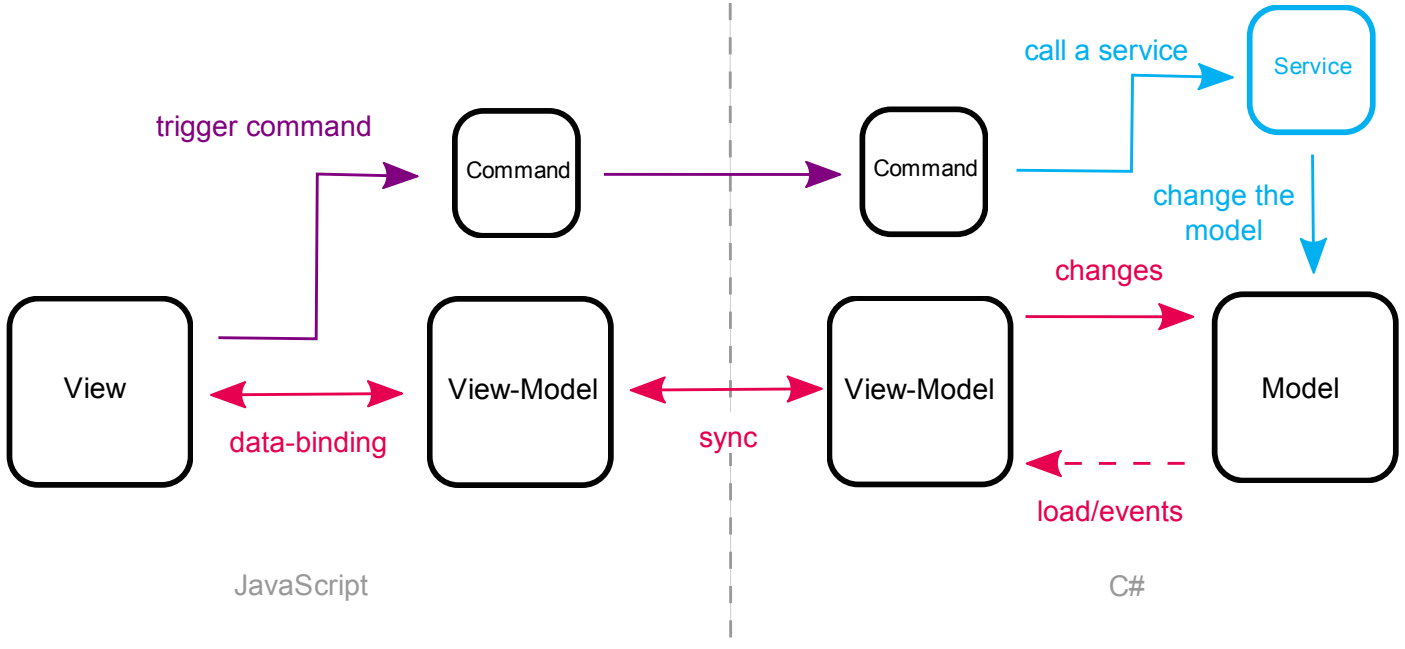
MVVM in RamphastosUI



View-Model wird zwischen JavaScript und C# synchronisiert.

MVVM in RamphastosUI

mit Commands



Commands existieren als *Repräsentanten* auf JavaScript-Seite als Trigger.

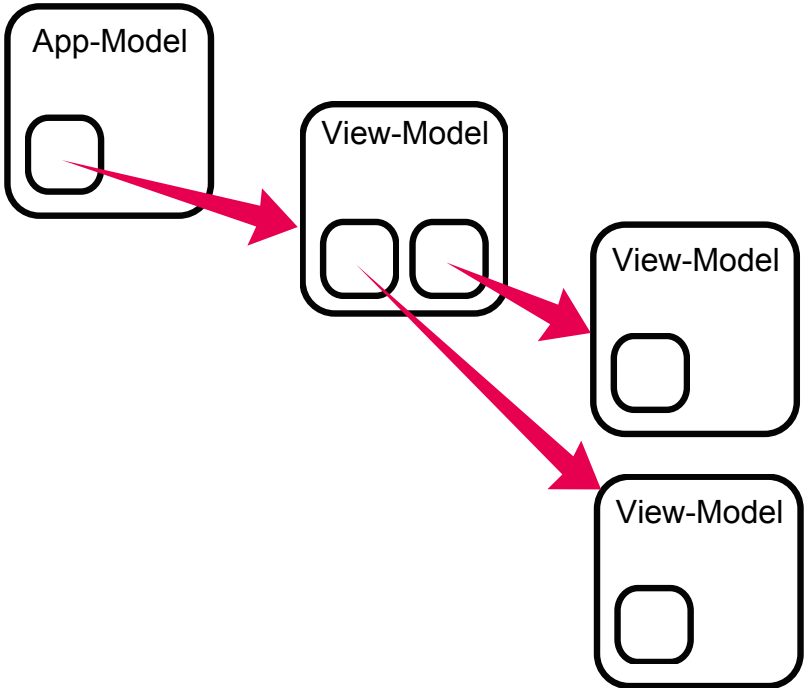
Synchronisation der View-Models

Lokal: C#/JavaScript Brücke

Remote: Web-Sockets

Verschachtelung

Jedes View-Model kann View-Models als Properties oder Elemente einer Collection haben. Eine Applikation besteht aus einer Hierarchie von View-Models.



Embedded Browser-Technologie

RamphastosUI ist flexibel in Bezug auf die Rendering Engine. Dies wird durch eine Abstraktionsebene erreicht. In erster Linie sollen *Awesomium* und *CEF (CefGlue)* unterstützt werden.

Third Party Frameworks

- AngularJS
- SignalR

AngularJS

Wichtig: Kurz erläutern was Direktiven sind.

MVC/MVVM Framework für JavaScript von Google.

Verwendung:

Die Client-Seite ist mit AngularJS (1.4) implementiert.
Views stehen als Direktiven zur Verfügung.

SignalR

Eine Bibliothek für Echtzeit-HTTP-Kommunikation mit Web Sockets von Microsoft. Sie bietet Fall-Back Mechanismen falls Web-Socket Unterstützung fehlt.

Verwendung:

Im Remote-Fall wird SignalR für die Kommunikation verwendet.

Demo

Retrospektive

Anfang 2013

Entwicklung eines ersten Protoptypen:

- Basierend auf Awesomium
- Jedes UI Element hatte eine Render-Funktion in JavaScript
- Das Entwickeln sollte sich ähnlich anfühlen wie mit WinForms
- Verwendung des MVC Patterns
- Reduktion von JavaScript auf ein Minimum
- Lokale Rich-Client-Applikation mit Awesomium und Remote Applikation basierend auf Asp.Net Webservices
- UI Events werden von JavaScript zu C# weitergeleitet:
  - Lokal: JavaScript/C# Bridge von Awesomium
  - Remote: Web-Services (Polling)

Frühling 2013

Projekt wird eingefroren. Stattdessen wird WPF evaluiert.

Gründe: Das Risiko mit dem Embedded Browser von Awesomium LLC wurde als zu hoch eingeschätzt. Ein Wechsel auf andere Embedded Web Browser-Technologien war noch keine Option.

Mai 2014

Entwicklung weiterer Prototypen:

- Verwendung des MVVM Design Patterns anstelle von MVC
- HTML Templates anstelle von Render-Funktionen
- Synchronisieren des View-Models zwischen JavaScript und C# anstelle der Übertragung von UI-Events
- Verwendung von Web-Sockets im Remote Fall
- Das Synchronisieren von View-Models wurde mit einer reinen REST-Implementierung verglichen
- Verwendung von CEF (CefGlue) parallel zu Awesomium

Juni 2014

Start eines Projektes zur Entwicklung eines HTML5 UI-Frameworks, später genannt 'RamphastosUI', zur weiteren Evaluation.

Juni 2015

RamphastosUI wird intern verwendet.

Aktueller Status

Windows (lokal):	Late Beta
Windows (remote):	Alpha
OS X (lokal):	Beta
Linux (lokal):	Prototyp

Vergleichbare Technologien

1. Electron
2. NW.js
3. Bracket-Shell
4. MVVM-for-awesomium
5. MVVM.CEF.Glue
6. Ionic Framework

Electron

Lizenz:	Open Source (MIT)
Rendering Engine:	Chromium
Plattformen	Windows, OS X, Linux
Back-End:	JavaScript (NodeJs)

Vormals bekannt als Atom-Shell, basiert auf io.js.
Verwendung: Atom Editor, Visual Studio Code.

NW.js

Lizenz:	Open Source (MIT)
Rendering Engine:	Chromium
Plattformen:	Windows, OS X, Linux
Back-End:	JavaScript (NodeJs)

Früher genannt *node-webkit*. Erlaubt es, node.js-Applikationen auf einem Desktop auszuführen. Entwickelt im Intel Open Source Technology Center.

Bracket-Shell

Lizenz:	Open Source (MIT)
Rendering Engine:	CEF
Plattformen:	Windows, OS X, Linux
Back-End:	C++

Entwickelt von Adobe für Brackets. Wurde aber schon für andere Projekte verwendet.

MVVM-for-awesomium

Lizenz:	Open Source (LGPL 3)
Rendering Engine:	Awesomium
Plattformen:	Windows (WPF)
Back-End:	C#

Interagiert sehr gut mit WPF. Bindings mit KnockoutJS.
Verwendet von CataMoeda.

MVVM.CEF.Glue

Lizenz:	Open Source (LGPL 3)
Rendering Engine:	CEF (CefGlue)
Plattformen:	Windows (WPF)
Back-End:	C#

Interagiert sehr gut mit WPF. Bindings mit KnockoutJS.
Nachfolger von MVVM-for-awesomium.

Ionic Framework

Lizenz:	MIT
Rendering Engine:	verfügbare Web View
Plattformen:	iOS, Android
Back-End:	JavaScript (Cordova)

Windows Phone und FirefoxOS geplant.

RamphastosUI

Lizenz:	Noch nicht festgelegt
Rendering Engine:	Awesomium, CEF (CefGlue)
Plattformen:	Windows, OS X, (Linux), Web
Back-End:	C#

Hierarchischer Aufbau von View-Models. Binding mit AngularJS.

Ausblick

- Fokussierung auf CEF
- Unterstützung für WindowsPhone, Android und iOS
- Unterstützung weiterer Embedded Browser Technologien (Bsp. Microsoft Edge)
- Migration zu Angular 2.0
- (Unterstützung anderer MVC Frameworks wie ReactJS, Aurelia, Ember, etc.)

Vision

- HTML als Alternative zu existierenden UI Technologien etablieren
- Eine echte Multi-Plattform UI Technologie für C# bieten

RamphastosUI

… eine Bibliothek zur Erstellung von C# Applikationen mit HTML5 als UI

Mit simultaner Unterstützung von Web- und Native-Applikationen

Fragen?

Feedback ist erwünscht!

Alexander S. Timmermans
alexander.timmermans@matchmycolor.com

31.8.–3.9.2015
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank

Alexander S. Timmermans

matchmycolor GmbH