|  |
| --- |
|  |

C:\Users\Frank\Documents\ICT Baden\git\stonehenge\IctBaden.Stonehenge2\App\icon32.png

***stonehenge***

***Version 2.0***

***13. Juni 2015***

**Document Conventions**

This document uses certain types of document conventions to help you distinguish source code from language elements, from keyboard sequences, and so on.

The following elements clarify the conventions used across the documentation.

** Example**

Indicating an example illustrating the above-described property or feature.

** Tip**

Indication of tips on how functionality described may be used.

** Operation**

Description of the internal functioning so that a better understanding is achieved.

** Prohibition**

Specification of prohibited or unsupported properties or functions.

** Question**

Questions and answers that have already been made ​​in the given context.

** Links**

References to Internet sites which contain more information.

 **Hint**

Note that has to be observed during operation.

 **Caution**

Important notice which needs to be strictly observed for error-free use of the functionality.

**Content**

[Introdution 4](#_Toc359234644)

[Features 5](#_Toc359234645)

[Application icon 5](#_Toc359234646)

[Supported HTML elements 6](#_Toc359234647)

[input text 6](#_Toc359234648)

[select 6](#_Toc359234649)

# Introdution

Die Software...

# Getting started

### Solution folders

 app  
  lib

 script  
 images

 ViewModels

## Create from scratch

Chreate new Windows Forms application.

Remove Form1

Open Program.cs

Add an static AppEngine App

Replace the contents of the Main method

static class Program

{

public static AppEngine App;

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

App = new AppEngine("MyApp", "start");

App.Run(true);

}

}

Create an application folder named “App”

# Features

## Application icon

To use an application icon place an 16x16 image named “icon.png” in the applications root folder.

/App/icon.png

## Data binding

### Basic functionality

### Options

[Bindable(true, BindingDirection.OneWay)]

## Command binding

## Supported Controls

### Button

### Calendar / DatePicker

### CheckBox

**HTML**

<input type="checkbox" data-bind="checked: Checked" />

**HTML (with update on change)**

The update on change works only if bound to a CheckedItem value (here AutoUpdate).

<label data-bind="with: AutoUpdate">

<input type="checkbox" data-bind="checked: Checked, valueUpdate: 'click',   
 event:{click: $root.OnAutoUpdateChanged}" />

  &nbsp;<span data-bind="text: Title"></span>

</label>

Using a label enables clicking on this too.

### ComboBox / ListBox

**HTML (as combobox)**

<select data-bind="options:Sites, selectedOptions:SelectedSiteIds" size="1" >

</select>

**HTML (as listbox)**

<select data-bind="options:Sites, selectedOptions:SelectedSiteIds" size="20" >

</select>

**HTML (with binding to complex element)**

<select data-bind="options:Sites, selectedOptions:SelectedSiteIds, optionsText:'Name', optionsValue:'Id'" size="20" >

</select>

**HTML (with update on change)**

<select data-bind="options:Sites, selectedOptions:SelectedSiteIds, optionsText:'Name', optionsValue:'Id', event:{change: OnSiteChanged}" size="20" >

</select>

### CheckedListBox

Items (here BitValues) are a list of CheckedItem values.

<div class="uneditable-input" style="width: 200px; height: 100px; overflow-y: scroll; cursor: default; " >

  <ul data-bind="foreach: BitValues" style="list-style-type: none;   
 margin: 0; padding: 0; overflow-x: hidden;">

    <li>

      <label>

        <input type="checkbox" data-bind="checked: Checked, valueUpdate: 'click',   
 event: {click: $root.OnBitsChanged}" />

        &nbsp;<span data-bind="text: Title"></span>

      </label>

    </li>

  </ul>

</div>

### Expander

### Image

### Label

### ListBox

See ComboBox.

### Menu

### PasswordBox

Use textbox type password.

### ProgressBar

### RadioButton

### ScrollBar / ScrollView

Use DIV with styles.

### TabControl

### TextBox

**HTML**

<input type="text" data-bind="value: Name" />

**HTML (with update on change)**

<input type="text" data-bind="value: Name, valueUpdate: 'afterkeydown', event:{keyup: OnNameChanged}" />

### TreeView

# Links

## Durandal

Single Page Apps Done Right <http://durandaljs.com/>

## Bootstrap

Sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development. <http://getbootstrap.com/>

## jQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

<http://jquery.com/>

## Knockout

Simplify dynamic JavaScript UIs by applying the Model-View-View Model (MVVM) pattern.

<http://knockoutjs.com/>