

**DOKUZ EYLÜL UNIVERSITY**  
**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

**GENERATING A DYNAMIC USERCONTROL  
LIBRARY FOR WINDOWS PHONE 8  
DEVELOPMENT**



**by**  
**İlker SATMAZ**

**October, 2015**

**İZMİR**

# **GENERATING A DYNAMIC USERCONTROL LIBRARY FOR WINDOWS PHONE 8 DEVELOPMENT**

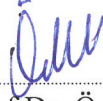
**A Thesis Submitted to the  
Graduate School of Natural and Applied Sciences of Dokuz Eylül University  
In Partial Fulfillment of the Requirements for the Master of Science in  
Computer Engineering**

**by  
İlker SATMAZ**

**October, 2015  
İZMİR**

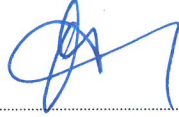
## M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **“GENERATING A DYNAMIC USERCONTROL LIBRARY FOR WINDOWS PHONE 8 DEVELOPMENT”** completed by **ILKER SATMAZ** under supervision of **ASST. PROF. DR. ÖZLEM AKTAŞ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Asst.Prof.Dr. Özlem AKTAŞ

Supervisor




Yrd. Doç. Dr. Dilek or Tamer

(Jury Member)



Yrd. Doç. Dr. Kâmil Ulaş BİRKANT

(Jury Member)



Prof.Dr. Ayşe OKUR  
Director

Graduate School of Natural and Applied Sciences

## ACKNOWLEDGEMENTS

I would like to thank my supervisor Asst.Prof.Dr. Özlem AKTAŞ for all the supports and encouragement that make me reach success at the end of this influential process.

Finally, I would like to offer my special thanks to my family for their support, patience, and help. I would not have been able to complete this thesis without their support and help.



İlker SATMAZ

# **GENERATING A DYNAMIC USERCONTROL LIBRARY FOR WINDOWS PHONE 8 DEVELOPMENT**

## **ABSTRACT**

In recent years, mobile software development has become an indispensable part of software development thanks to having increased on usage smart phones and tablets. Porting sites -that already developed- on mobile platforms or developing a new project that able to run the project on both sites and mobile platforms result in searching new development platforms.

Nowadays, even percentage of using devices which has operating systems like Android and IOS that are the most popular ones, developed mobile software must be referred to all customers. Due to the fact that applications must be suitable on devices that use Windows operation systems too.

Delivering application to a customer on delivery time and delivering it without any issue are the most two facts on software development. On any project, even the time schedule and resources are planned correctly, most of the companies require to postpone delivery time considering crossing some problems during development.

The idea of this project is basically planned when trying to fix a lack of a platform that used on mobile banking project. This project may be used on applications that use Windows Phone 8 platform for lacks that required on the project or spend so many times to develop. Also, for being able to use in most of the projects, it will be generated dynamically.

On this thesis, significant properties and detailed structure of project will be clarified. Also, information about problems which occurred on this development and used technologies will be explained. Lastly, cause of usage of controls that

developed on this project will be explained in detail, this thesis will be as a user guide for future works.

**Keywords:** Windows phone development, user controls for windows phone, custom slider, dynamic style for user control



# **WINDOWS PHONE 8 GELİŞTİRMESİ İÇİN DİNAMİK KULLANICI KONTROL KÜTÜPHANESİ OLUŞTURMA**

## **ÖZ**

Son zamanlarda akıllı telefonların ve tabletlerin kullanımının artması ile birlikte mobil yazılım sektörünün vazgeçilmez bir parçası haline gelmiştir. Daha önceden yapılan sitelerin mobil ortama taşınması ya da yeni yapılacak işlerin hem mobil hem de web tabanlı yapılacak şekilde kararlaştırılması, yazılımcıları yeni platformları tanımaya itmektedir.

Günümüzde Android, IOS gibi işletim sistemlerinin yaygın olduğu cihazların kullanım oranı her ne kadar fazla olsa da, yapılan mobil yazılımın müşteri kitlesinin hepsine birden hitap edebilmesi gerekir. Bunun için uygulamaların Windows işletim sistemlerini kullanan cihazlar da uyumlu yazılması gerekmektedir.

Bir uygulamanın müşteriye belirlenen zamanda ve sorunsuz teslim edilmesi yazılım sektörünün en önemli iki unsurudur. Bir projede zaman ve kaynak planlaması başlangıç aşamasında doğru yapılmazsa da çoğu firma proje yapım aşamasında yaşadığı sıkıntılardan dolayı uygulamanın teslim tarihini uzatma seçeneğine gitmek zorunda kalıyor.

Bu proje aslında temel olarak bir mobil bankacılık projesi aşamasında kullanılan platformun temel eksiklerinin giderilmesi yönünde ortaya çıkmıştır. Bu proje Windows Phone 8 platformunda yapılacak uygulamalarda yazılımcıların sıklıkla ihtiyaç duyacağı ve efor kaybettirecek eksiklikleri içerir. Ayrıca tüm uygulamalara uyacak bir şekilde geliştirilmiştir.

Bu tezde projenin kayda değer özelliklerinden ve detaylı yapısından bahsedilecektir. Ayrıca bu süreçte karşılaşılan sorunlar ve geliştirme boyunca kullanılan teknolojiler hakkında detaylı olarak bilgi verilecektir. Son olarak tez

yapılan kontrollerin kullanımını da detaylı içereceđi için ileriki projelerde kullanılabilmesi için bir kullanım kılavuzu niteliđine sahip olacaktır.

**Anahtar kelimeler:** Windows mobil geliřtirmesi, Windows mobil için kullanıcı kontrolleri, özel sürgü, kullanıcı kontrolü için dinamik stiller





## CONTENTS

	Pages
M.Sc THESIS EXAMINATION RESULT FORM.....	ii
ACKNOWLEDGEMENTS .....	iii
ABSTRACT.....	iv
ÖZ .....	vi
LIST OF FIGURES .....	x
LIST OF TABLES .....	xii
 <b>CHAPTER ONE-INTRODUCTION .....</b>	 <b>1</b>
1.1 Introduction.....	1
1.2 Motivation.....	2
1.3 Windows Phone 8 .....	3
 <b>CHAPTER TWO- DEVELOPMENT.....</b>	 <b>4</b>
2.1 Development Technologies .....	4
2.1.1 Enable Virtualization.....	4
2.1.2 Install Phone SDK .....	6
 <b>CHAPTER THREE-IMPLEMENTATION.....</b>	 <b>7</b>
3.1 Basic Components .....	7
3.1.1 CustomTextBox.....	7
3.1.1.1 Implementation of CustomTextBox .....	7
3.1.1.2 Input Scope .....	7
3.1.1.2.1 New InputScope Types .....	8
3.1.1.2.2 Validate Input .....	9
3.1.1.2.3 Validation Methods .....	10
3.1.1.3 TextBox Title.....	11

3.1.2	CustomPasswordBox.....	11
3.1.2.1	Input Scope .....	11
3.1.2.1.1	New InputScope Types .....	12
3.1.2.1.2	Validate Input .....	12
3.1.1.2.3	Validation Methods .....	13
3.1.2.2	Password Box Title .....	13
3.1.3	Custom Slider .....	14
3.1.4	Custom Map Control .....	15
3.1.5	Custom Pin Control .....	16
3.2	Dependency Property.....	17
3.2.1	Implementation.....	18
3.3	Binding Property.....	19
3.3.1	Binding Example.....	20
3.4	Converter .....	21
3.4.1	Converter Methods .....	22
3.4.2	Converter Example.....	23
3.5	MVVM Pattern .....	25
3.5.1	Pattern Properties .....	25
3.3.2	Delegate Command .....	26
3.5.3	Login Example Using MVVM Pattern .....	28
<b>CHAPTER FOUR-USAGE OF LIBRARY AND USERCONTROLS.....</b>		<b>31</b>
4.1	Including Library to Specific Project.....	31
4.2	Usage of CustomTextBox .....	34
4.3	Usage of PasswordTextBox .....	35
4.4	Usage of Slider .....	36
4.5	Usage of MapControl.....	38
<b>CHAPTER FIVE- CONCLUSION AND FUTURE WORKS.....</b>		<b>40</b>
<b>REFERENCES.....</b>		<b>42</b>

## LIST OF FIGURES

	<b>Pages</b>
Figure 1.1 Windows phone 8 emulator types.....	3
Figure 2.1 Enable virtualization in bios .....	5
Figure 2.2 Enable virtualization in windows features.....	5
Figure 3.1 Input scope keyboards .....	8
Figure 3.2 Keydown event for textbox .....	9
Figure 3.3 Custom textbox title.....	11
Figure 3.4 Custom textbox title.....	14
Figure 3.5 Custom pin control .....	17
Figure 3.6 Registering dependency property .....	18
Figure 3.7 Using defined property .....	18
Figure 3.8 Login model with property change event .....	20
Figure 3.9 Initializing view & model and binding properties.....	21
Figure 3.10 Result of two way binding operation.....	21
Figure 3.11 IValueConverter example.....	23
Figure 3.12 Using converter in page .....	24
Figure 3.13 Result of implemented converter.....	25
Figure 3.14 Delegate command and its usage.....	27
Figure 3.15 Defining model class for mvvm pattern .....	28
Figure 3.16 Defining viewmodel class for mvvm pattern .....	29
Figure 3.17 Including viewmodel to the page.....	29
Figure 3.18 Binding model items and command .....	30
Figure 3.19 Result of using mvvm pattern.....	30
Figure 4.1 Add reference the library to project.....	31
Figure 4.2 Adding new tap to toolbox .....	32
Figure 4.3 Choosing item to new tab .....	32
Figure 4.4 Adding controls to the new tab.....	33
Figure 4.5 New controls added from library.....	33
Figure 4.6 CustomTextBox example .....	34
Figure 4.7 CustomPasswordBox example .....	35

Figure 4.8 Style properties for custom slider .....	37
Figure 4.9 CustomSlider example.....	37
Figure 4.10 CustomMapControl example.....	39



**LIST OF TABLES**

	<b>Pages</b>
Table 3.1 Input scope types .....	8
Table 3.2 Input scope types .....	12
Table 3.3 Properties of custom slider .....	14
Table 3.4 Properties of dependency property .....	18



# **CHAPTER ONE**

## **INTRODUCTION**

### **1.1 Introduction**

Nowadays, there are some development tools for a multi-cross platform of mobile applications. You are developing the applications once in desire development language and then you may create packages for Android, IOS, and Windows Phone.

Several types of research were done about pros and cons of multi-cross platform applications rather than native apps. And also, work on several projects on windows phone native app and mobile app developed by multi-cross platform (Sencha). Each mobile environment has specific properties and a multi-cross platform may not provide all properties in same code. That's why there are some shell applications for being able to use HTML environment in a mobile environment. You can find so many tutorials for Sencha or any multi-cross platform environment.

After preparing requirements of a mobile project, you can decide whether a native app or multi-cross platform app will be used. Actually, this document will provide you to be able to use windows phone 8 properly.

There are so many courses for windows phone development. One of them is prepared by Microsoft on channel9 (Channel 9, 2013). Also, there are so many e-books. Actually, before doing this project two e-books were commonly searched for being acknowledge about windows phone development and XAML. Jamie Clark prepared a guide for developers about XAML platform (Clark, 2010). For more details about XAML and c#, e-book that prepared by Bubby James and Lori Lalonde may be used (James & Lalonde, 2015)

## 1.2 Motivation

Because of doing a several project on Windows Platforms, having controls that already implemented is the most desired thing. Because, developers can estimate deadline of any task but there may be so many issues caused by developers or platforms.

The thing about a login operation is being done on android. There is an SMS OTP (one-time password) authentication and you need to implement getting OTP that sent to mobile as a message and set it to text box automatically. It seems easy for a developer and it takes just 2-3 Hours to implement. But if a customer requests you to do this on windows phone or IOS, there is no way to do that because both platforms do not allow you to read inbox of the device. Also, the first user control you will in implementation part Custom Text Box, in windows phone 8 there is no way to restrict user entering keys out of rule. Even input scope defines as Telephone Number, you are allowed to enter special characters. All details about input scope are listed in Microsoft software development network site (MSDN, n.d.b). Also, there are so many works about input scopes (StackOverFlow, 2012a)

Because of these problems, it decided to create a library that includes all basic components that may be used in any project.

Dynamic User Control Library may be used by all developers that developed any windows phone 8 projects. The aim of this library is providing controls like Custom Text Box, Custom Password Box, Custom Slider and many custom controls defined in this document. Lots of controls are already included in a native phone application. But there are some major missing properties, like being able to use a proper scope on text boxes.

### 1.3 Windows Phone 8

Windows Phone 8 is a software development environment that provides you creating applications for devices using Windows Phone operating system. Microsoft has a certificate with some devices for this operation system like Nokia. Cause of most common used operation system is Microsoft in computers, some users felt confidence with Microsoft in mobile devices too.

Windows Phone 8 applications are developed on Visual Studio Tools (2012 and later versions). After having Visual Studio, you need to install SDK, which will provide you to create apps and debug them on an emulator for Windows Phone.

Rather than being able to test app on different devices, you can use an emulator in desired resolution.

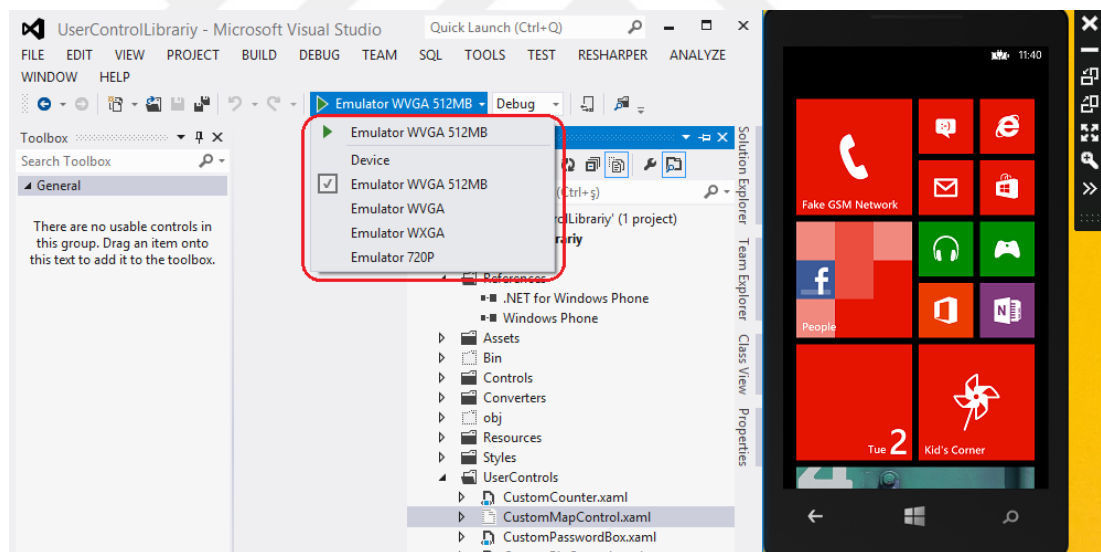


Figure 1.1 Windows phone 8 emulator types



## **CHAPTER TWO**

### **DEVELOPMENT**

#### **2.1 Development Technologies**

This project is a Windows Mobile Base project that is developed on Windows Phone 8 via Microsoft Visual Studio 2012 tool in Windows 8 platform. And please note that the minimum requirement of Phone 8 development is these specifications.

Another important factor in the development of this project, PC used for development needs to support virtualization. And also on some computers that support virtualization; the virtualization property is not enabled. You need to enable virtualization from Bios and then from IIS (Internet Information Services).

And last required tool for this development is Phone SDK. Without installing this tool on visual studio, you may not compile the existing any phone project and may not create new phone project. And also emulators are installed in a machine during installation of this SDK.

##### ***2.1.1 Enable Virtualization***

There are so many sources that guide you to enable virtualization property and open this feature on IIS.

Enabling virtualizing can be different according to your computer type. Because bios settings may be different for each computer. Just have a look at steps of enabling virtualization referred from technet blog (Enabling HYPER-V, 2014).

- Ensure that hardware virtualization support is turned on in the BIOS settings.



Figure 2.1 Enable virtualization in bios

- Save the BIOS settings and boot up the machine normally,
- At the Start Screen, swipe the right hand side of the screen and select the Search Charm.
- Type turn windows features on or off and select that item
- Select and enable Hyper-V

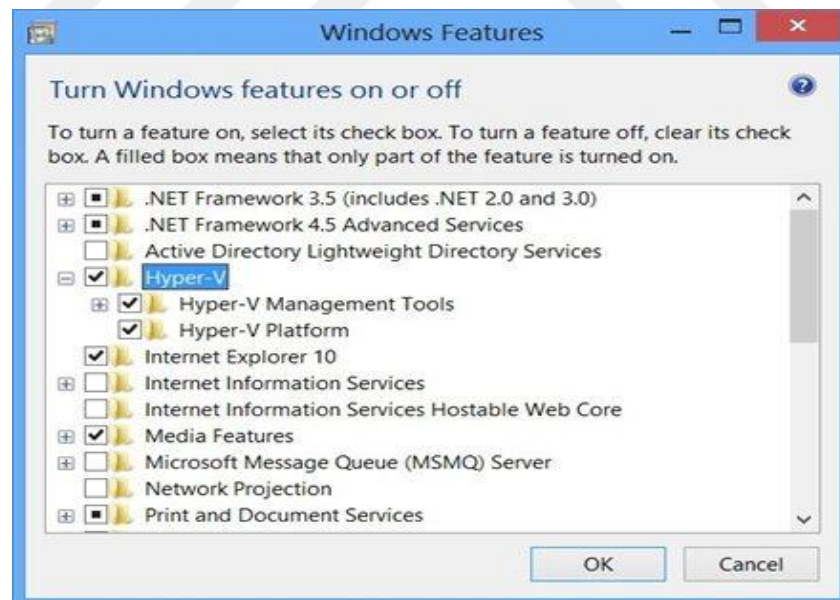


Figure 2.2 Enable virtualization in windows features

- If Hyper-V was not previously enabled, reboot the machine to apply the change

### ***2.1.2 Install Phone SDK***

After enabling virtualization and ensure that have requirement tools on your computer (Windows 8 and Visual Studio 2012 or later), phone SDK need to be installed. Required SDK and emulators may be downloaded from official Microsoft Development Center site (Windows Development, n.d.). You may visit for more info: <https://dev.windows.com/en-us/develop/download-phone-sdk>



## **CHAPTER THREE**

### **IMPLEMENTATION**

#### **3.1 Basic Components**

Dynamic user control library includes 8 user controls may be used any project in desired way. In this part of project, these controls are defined and usage of these controls included in example codes.

##### ***3.1.1 CustomTextBox***

Text Boxes may be the most used controls in any development technology. But in Phone 8 there are major missing properties. And the important one in these properties is Input Scope.

##### ***3.1.1.1 Implementation of CustomTextBox***

Cause of problem defined below, new property was added to text box. This property is an optional property. User may be select type of input for being able to validate inputs automatically.

##### ***3.1.1.2 Input Scope***

This default property is used for specify inputs and providing project more user-friendly. According to type of InputScope, keyboards will automatically change in devices (Number, TelephoneNumber, AlphaNumeric etc).

Missing on Phone 8; even if you are selecting InputScope as Number, user is able to enter special characters or change the keyboard type manually.



Figure 3.1 Input scope keyboards

There are two main keyboard types in phone 8 (Figure 3.1), one them (left one) includes all characters and the other only numeric and special (“,” “-” “.”) characters

*3.1.1.2.1 New InputScope Types.* There will be 6 Scope Types for this control. User may be select any of them and user inputs will be validate according to selected input:

Scope Type	Description
Default	<b>Default keyboard will be opened and user will be allowed to enter any input.</b>
Decimal	<b>User will be allowed to enter decimal inputs (15,500.75)</b> <b>[0-9] “,” “.”</b>
Float	<b>For only float inputs (15.75)</b> <b>[0-9] “,” “.”</b>
OnlyNumber	<b>Only numeric characters will be allowed</b> <b>[0,9]</b>
OnlyText	<b>Any special and numeric characters will not allowed</b> <b>[a-z],[A-Z]</b>
AlphaNumeric	<b>Just numeric and alphabetic characters are allowed</b> <b>[0-9], [a-z], [A-Z]</b>

Table 3.1 Input scope types

3.1.1.2.2 *Validate Input*. According to defined Scope Type, input must be validated and operations need to be done for being able to control this textbox.

- *KeyDown Event*: Input may be handled in this event. Pressed key is handled and you can disable this operation whether is not valid. According to pressed key and selected scope type, the key needs to be checked if it is valid or not. If the key is not valid, the system cancels this press event and in this case, keys that are not valid may be disabled for users.

```
private void CustomTextBoxControl_OnKeyDown(object sender, KeyEventArgs e)
{
    e.Handled = !IsValid(e.Key);
}
```

Figure 3.2 Keydown event for textBox

In Figure 7, you will see that according to `IsValid` method, `KeyEvent` will be handled or not.

- *KeyUp Event*: On two situations (decimal and float scope types), need to validate whole input as one text. Because user may be enter “,” and “.” characters. For validating the input, `KeyUp` Event is used. In this event, all input included new entered character is validated whether it is possible to parse it in selected scope type or not. If validation is not a success, last entered character is removed from entered text. See below example for this operation:

Scope Type:	Decimal
Current Text in TextBox:	“15.75”
New entered input:	“.”
KeyUp Event:	Validate (“15.75.”) (False)
Last Text in TextBox:	“15.75”

- *LostFocus Event*: On same situation used in `KeyUp` Event, input may be formatted on `LostFocus` Event. Formatting input according to

selected scope type is user-friendly because user see what is being typed at the time of typing completed.

Scope Type: Decimal

Current Text in TextBox: “15, 50.75”

Last Text after LostFocus: “1550.75”

*3.1.1.2.3 Validation Methods.* For being able to validate input according to selected input scope, some methods should be used. Validation may be done in Lost Focus, Key Up or Key Down events. But in Lost Focus and Key Up event if validated input is incorrect and removed from text box, it appears and disappears in a while. This does not make a good sense on users. Thanks to that Key Down event is used. This event is triggered when a user presses down to any key. If the validation is false, handling key may be canceled and user thinks that button is not active. It is a better approach for mobile development.

- **IsValid:** this method called on Key Down event and return if entrance key is valid or not. According to returned value key whether is handled or not.

```
private bool IsValid(Key key)
{
    switch (inputScope)
    {
        case ScopeType.AlphaNumeric:
            return ValidateAlphaNumeric(key.ToString());
        case ScopeType.OnlyText:
            return ValidateOnlyText(key.ToString());
        case ScopeType.OnlyNumber:
            return ValidateNumeric(key);
    }
    return true;
}
```

As you see in above code block, this method is used as methods for validations:

- **ValidateAlphaNumeric:** on the situation of selected scope is AlphaNumeric, this method will be used for validation. Regex was used for an exact validation.

```
private bool ValidateAlphaNumeric(string text)
{
    if (text.ToLower() == "unknown") return false;
    var regex = new Regex("[a-zA-Z0-9 ]+$");
    return regex.IsMatch(text);
}
```

- ValidateOnlyText: Another regex was used for this validation

```
private bool ValidateOnlyText(string text)
{
    var regex = new Regex("[a-zA-Z ]+$");
    return regex.IsMatch(text);
}
```

- ValidateNumeric: Cause of space problem on regex validate to only numbers, this validation was done with using key property.

```
private bool ValidateNumeric(Key key)
{
    //Valid keys for numeric option
    return (key == Key.D0 || key == Key.D1 || key == Key.D2 || key == Key.D3 || key == Key.D4 ||
            key == Key.D5 || key == Key.D6 || key == Key.D7 || key == Key.D8 || key == Key.D9);
}
```

### 3.1.1.3 TextBox Title

Lots of mobile application use a textbox with a title. Cause of this, title property is added to custom control. There is no need to do any control for adding textbox control. “**Title**” property is served to do this. Just set the title property as you desired, it will be generated.



Figure 3.3 Custom text box title

### 3.1.2 CustomPasswordBox

Phone 8 already has PasswordBox control in default. But there is no input scope property on this control. Cause of that new custom control is implemented on this library.



### 3.1.2.1 Input Scope

According to type of InputScope, keyboards will automatically change in devices (Number, TelephoneNumber, AlphaNumeric etc). Actually, implementation of this control is similar to CustomTextBox.

*3.1.2.1.1 New InputScope Types.* There will be 4 Scope Types for this control. User may select any of them and user inputs will be validated according to selected input:

Scope Type	Description
Default	<b>No validation, allows all characters: *</b>
OnlyNumber	<b>Only numeric characters will be allowed: [0,9]</b>
OnlyText	<b>Any special and numeric characters will not allowed: [a-z],[A-Z]</b>
AlphaNumeric	<b>Just numeric and alphabetic characters are allowed: [0-9], [a-z], [A-Z]</b>

Table 3.2 Input scope types

*3.1.2.1.2 Validate Input.* Cause of it is a password box, no need to format input just need to validate input according to selected scope type.

- KeyDown Event: Event Usage is same with CustomTextBox control. Input need to be validated and if it is valid need to be handled on text box
- KeyUp Event: Cause of using textbox instead of password box, input need to be masked. For providing this, each new input added to property that keeps all inputs. If the new input is back button last input is removed from the whole input.

*3.1.2.1.3 Validation Methods.* Methods that defined in Custom text Box control are used as the same functionality in this control: `IsValid`, `ValidateNumber`, `ValidateAlphaNumeric` and `ValidateText`.

Also, by cause of that there is mask operation, there should be Storing. entering pin and changing it through using mask character:

- `SetPinText`: Cause of entered character is checked if it is valid and changed with mask char. Only last textbox text characters may be get as unmasked. In each key up bottom this method will be called for storing pin.

```
private void SetPinText(TextBox textBox)
{
    if (Pin1.Length < textBox.Text.Length && !string.IsNullOrEmpty(textBox.Text))
        Pin1 += textBox.Text.Last();
    else if (!string.IsNullOrEmpty(Pin1))
    {
        Pin1 = Pin1.Substring(0, Pin1.Length - 1);
    }
}
```

- `ChangePinText`: After `SetPinText` method, this method will be called to change entered text with defined mask character. Change operation was done with using regex operation.

```
private void ChangePinText(TextBox textBox)
{
    textBox.Text = Regex.Replace(textBox.Text, @"[0-9]", maskChar);
    textBox.SelectionStart = textBox.Text.Length;
}
```

#### *3.1.2.2 Password Box Title*

Lots of mobile application use a textbox with a title. By cause of this, title property has been added to custom control. No need to do any control for adding password box control. “**Title**” property is served to do this. Just set the title property as you desire, it will be generated.

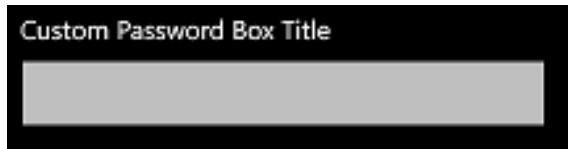


Figure 3.4 Custom text box title

### 3.1.3 Custom Slider

Another major implementation on this library is implementing a custom slider. There is a big problem on slider control. You can change slider value with touch event but also sliding page used touch event too. Cause of that, even the user only tries to slide page, if a finger touches on a slider, the value is automatically changed in a slider. And also customizing the slider takes too much time. Owing to these reasons, the slider is customized using Slider. For being able to create a custom slider, all properties of slider must be known (MSDN, 2014). Also, some developers give information about the custom slider in social MSDN blog (Social MSDN, 2013). And some works are already done about custom slider (StackOverFlow, 2012b)

#### 3.1.3.1 Properties

There are some major properties which are always used, and some of them mandatory for defining slider.

Name	Description
Amount	<b>Value of slider that changes in each slide</b>
Minimum	<b>Minimum value of slider</b>
Maximum	<b>Maximum value of slider</b>
LargeChange	<b>Large of changing value in each slide</b>
Prefix	<b>Used for defining currency (Like: 1,500.00 <b>EUR</b>)</b>
SliderType	<b>Used for formatting slider as amount or normal</b> <ul style="list-style-type: none"> <li>○ <b>Normal: 1500</b></li> <li>○ <b>Amount: 1,500.00</b></li> </ul>

Table 3.3 Properties of custom slider

### 3.1.3.2 Implementation

Manipulation Delta: This event is used for move operations. In this custom, only manipulation delta event is used for changing a value. According to a position of move operation slider value was increased or decreased. Changing value operation is done according to LargeChange property. Also, Minimum and Maximum properties are used for checking stopping criteria of changing operation. Last of all, a new value is formatted using IsInteger property as integer or decimal and bind to Amount.

GotFocus: This event is used for refresh value of thumb. On this event method textbox that will use for input entrance, is cleared.

ValueChanged: Slider thumb should be automatically changed. On this event, a value will be kept, check if it is valid and then the new value will be set on a slider.

LostFocus: Lost focus event generally used for formatting value. On this event, the value is formatted according to a type of slider.

### 3.1.4 Custom Map Control

Implementing map control is another thing that takes so much time.. In mobile technology, maps are used for defining a location and displaying it. Properties of map control are listed in MSDN site (MSDN, n.d.c). Also works about maps listed in references (StackOverFlow, 2014a, 2014b).

#### 3.1.4.1 Properties

There are two major classes used in this control.

- **GeoCoordinate:** This class is defined in *System.Device.Location* library for being able to use this, you need to add this library in project.
- **LocationList:** This class was created for keeping location info. Properties of this class:

- ***LatitudeField***: keeps latitude of location (**string**)
- ***LongitudeField***: keeps longitude information of location (**string**)
- ***NameField***: keeps names of location (**string**)

Also, there are some properties that have been added in this control for using them. These properties are used for specifying map properties:

- ***SelectedCoordinate***: for setting selected coordinate of location (**GeoCoordinate**)
- ***IsSetPushpin***: used for specifying whether will be pin or not on map (**bool**)

#### 3.1.4.2 Methods

After defining properties, methods should be declared. Because the main aim on this control is to implement these methods:

- ***GetLocation***: After properties are set, this method will be called for getting location that gives by a developer.
- ***ResetMap***: is used for reset a map. Pins and locations are cleared. You will have a clear map for setting new locations and pins.
- ***AddPushpinToMap***: adding pushpin to a map may be done with using this method. This method has a parameter as;  
`ObservableCollection<LocationList>`  
 You may send many locations for adding pins to these.
- ***ReverseGeocoding***: This is a method that can be used as an alternative. Aim of this message is to get selected address. In the situation of address that is already set on properties will be sent to developer. There is a callback event on this method. Because getting address located on map needs async call. As a result, you need implement callback event for catching response.

#### 3.1.5 Custom Pin Control

Pin control is an alternative method that is included in this library. To do a mobile app, this control may be usable for some developers.

### 3.1.5.1 Implementation

There are four text boxes and one title in this control. Importance of this control is to focus on next text box on each input entrance and focus on back text box on backspace button.

Problems on password box may be occurred on this control to considering that textboxes are generated from CustomPasswordBox.

At the last textbox lost focus an event, pin was generated from 4 textboxes and stored as Pin property. For being able to use entrance pin;

```
string enteredPin=CustomPinControl.Pin;
```

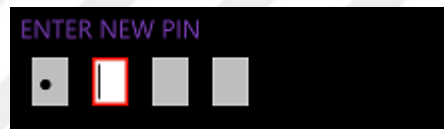


Figure 3.5 Custom pin control

## 3.2 Dependency Property

This property is the most important thing for dynamic structure. Managing controls on XAML page provides to developers. The thing about the most usable control the textbox, most of the developers set the **Text** property of text box on the design page. Setting this property on code behind is another alternate. But now, on the same control lets thing about **Style** property. There is no developer set the style on code behind because is not acceptable. Especially, when you are using MVVM structure, aim of this structure is moving all code pieces from code behind.

Also, the third part library like mine has no access to the code behind properties; you will not be able to set Title property.

For using properties like styles, titles, custom texts etc. dependency property is used.

### 3.2.1 Implementation

Dependency property has 4 properties (MSDN, n.d.a):

Property	Type	Description
Name	String	Defines names of property
PropertyType	Type	Use for type of property
OwnerType	Type	Use for type of control that property defines in it
TypeMetaData	PropertyMetaData	Defines certain behavior aspects of a dependency property as it is applied to a specific type, including conditions it was registered with

Table 3.4 Properties of dependency property

Now, have a look at the example defines for Title of control (Figure 3.6). There is TitleText Property defines for setting TextBlock text. This property is used on DependencyProperty registering. After that, you can reach to TitleText property on XAML (Figure 3.7).

```
public string TitleText { get { return TitleTextBlock.Text; } set { TitleTextBlock.Text = value; } }  
  
public static readonly DependencyProperty TitleTextProperty =  
    DependencyProperty.Register("TitleText",  
        typeof(string),  
        typeof(CustomCounter),  
        null);
```

Figure 3.6 Registering Dependency Property

```
<userControls:CustomCounter TitleText="Title" x:Name="CustomCounter" />
```

Figure 3.7 Using defined property

With using these implementation properties that be initialized by developers will be copied to design page. And also they are defined on code behind. So that, it will be optional for a developer. There are many examples that implemented by developers actually this development referenced from a post on code project side. (Code Project, 2009)

### **3.3 Binding Property**

There are actually two ways to set values of properties. The common way is setting values from code behind using a name of properties. But actually second way that called as binding should you used for code quality, reducing complexity and using exact models for mobile development.

Think about the basic toolbox item, the text box, a developer needs to define a name of a text box and then have to right set and get functions for being able to use it.

For a simple operation that uses a few controls this way may be sensible and easy. But when the number of controls increases that causes complexity and decrease readability of code. By cause of that using binding is more sensible then using this.

Binding operation has four properties that commonly used:

1. Value: Using for setting property
2. Mode: for defining mode of binding
  - a. OneWay: means operation is a one way operation. Property of item will be set but model that binds to item is never change
  - b. TwoWay: means item will be set from model and also when the item is changed, model will be changed too.
3. Converter: using for convert value to property type. Commonly used for getting picture according to item id.
4. ConvertParameter: provides to send any other parameter to converter for defining anything related with this parameter.



### 3.3.1 Binding Example

For being able to a good knowledge about binding operation, have a look at login operation done using “Binding”:

First of all login control includes username and password. On the grounds that LoginModel class was created and properties were defined in it. And please note that the property changed event must be initialized for all binding operations. If this property is not initialized and called on a set of property, the binding will not work at the time of any changing.

```
public class LoginModel
{
    private string userName;
    public string UserName
    {
        get { return userName; }
        set
        {
            userName = value;
            OnPropertyChanged("UserName");
        }
    }

    private string password;
    public string Password
    {
        get { return password; }
        set
        {
            password = value;
            OnPropertyChanged("Password");
        }
    }
}

public static event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged(string propertyName)
{
    if (PropertyChanged != null)
    {
        PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

Figure 3.8 Login model with propertychange event

Now, just need to initialize model and set it to related page or grid defined in page.

```

var model = new LoginModel {UserName = "Ilker"};
LayoutRoot.DataContext = model;

<Grid x:Name="LayoutRoot">

    <userControls:CustomTextBox TitleText="User Name" InputScope="Default"
        Text="{Binding UserName,Mode=TwoWay}" />

    <userControls:CustomPasswordBox TitleText="Password" InputScope="OnlyNumber" MaskChar="*"
        PasswordText="{Binding Password,Mode=TwoWay}" />

    <TextBlock Foreground="White" FontWeight="Bold">
        <Run Text="{Binding UserName,Mode=TwoWay}"></Run>
        <Run Text=" "></Run>
        <Run Text="{Binding Password,Mode=TwoWay}"></Run>
    </TextBlock>

</Grid>

```

Figure 3.9 Initializing view & model and binding properties

There is no button and event triggered, but by cause of property change event everything will be written to LoginModel and get from same class at run time. Also, on button event no need to get any property value, all items will be set in model.



Figure 3.10 Result of two way binding operation

As you see in figure (Figure 3.10), model property changes at the time the text is changed.

### 3.4 Converter

The converter is a property that may be called a binding operation. The aim of this property, processing binding element and return it to a type of item that used in. Binding operation checks types whether they are matched or not only in run time.

Cause of that any string property may be bound to textbox's Boolean property like `IsEnabled`. This issue will not cause any problem at the time of build, but when the page is rendered, operation throws an exception about that.

The problem may be solved checking binding elements with item property type. But in some situations, a model may not have exactly same type of property. And sometimes developers need to set properties according to any conditions.

Think about you have a `TextBlock` that displays available amount of account. If your amount is under your defined amount you need to display it in red color and higher then defined amount, an amount will be displayed in green color. In this situation, the exact way of doing this will be using a converter.

#### **3.4.1 Converter Methods**

Converter is a class that must be inherited from **`IValueConverter`** class. There is two methods which have to be overridden from main class in converter class:

- **Convert:** This method used as a main reason. Value will get in this class and converted value will be returned from this method. There is four properties that may be used from this method:
  - **Value:** keeps the value setted to element using binding operation.
  - **Parameter:** keeps the parameter that optionally set from developer for some reasons.
  - **TargetType:** defines type of data expected by the target dependency property
  - **Culture:** use for getting culture of page may be used also for cultural operations.
- **ConvertBack:** This method should be implemented for two-ways binding when the properties must be set in model property different than displayed type. Think about that `Decimal` is formatted using converter as Money-Currency format (1,000.00 USD). This field may be updated from user. But in model property, it is kept as decimal. Formatted value is not used for

setting model property because cause of converter is not decimal anyway. On this method formatted value will get and returned to decimal again. Method has the same properties with Convert method.

### 3.4.2 Converter Example

For adding a using a new converter to project,

1. Add new class to your project
2. Inherited it from IValueConverter
3. Implement Convert and ConvertBack method.
4. Define converter in project (should defined in Style folder)
5. Use defined converter in Binding operation

```
public class AmountForegroundConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter,
        System.Globalization.CultureInfo culture)
    {
        decimal availableAmount;
        decimal definedAmount;
        Decimal.TryParse(value.ToString(), out availableAmount);
        Decimal.TryParse(parameter.ToString(), out definedAmount);
        if (availableAmount < definedAmount)
        {
            var greenBrush=new SolidColorBrush(Colors.Green);
            return greenBrush;
        }
        var redBrush = new SolidColorBrush(Colors.Red);
        return redBrush;
    }

    public object ConvertBack(object value, Type targetType,
        object parameter, System.Globalization.CultureInfo culture)
    {
        return null;
    }
}
```

Figure 3.11 IValueConverter example

As you see in figure (Figure 3.11), AmountForegroundConvert class inherited from IValueConverter. Convert method used value as available amount and parameter as defined amount. According to result of comparing these values, Color property is returned.

For defining converter in project, use style page. Convert may be used in more than one page in project. Cause of that defining it in Style folder will be more useful.

```
<converters:AmountForegroundConverter x:Key="AmountForegroundConverter"/>
```

Now, model may be bound to page and converter may be used for foreground property. For giving an example, AmountModel class has been created and LowAmount and HighAmount properties have been added to this class. After initializing model, it has been set to DataContext of page.

After that, converter was used in for binding foreground property. Also parameter was set in this binding.

```
public MainPage()
{
    InitializeComponent();
    var model = new AmountModel
    {
        LowAmount = 5000,
        HighAmount = 12000
    };
    DataContext = model;
}

public class AmountModel
{
    public decimal LowAmount
    { get; set; }
    public decimal HighAmount
    { get; set; }
}

<TextBlock FontSize="32" Text="{Binding LowAmount}"
    Foreground="{Binding LowAmount,
        Converter={StaticResource AmountForegroundConverter},
        ConverterParameter=10000}"></TextBlock>

<TextBlock FontSize="32" Text="{Binding HighAmount}"
    Foreground="{Binding HighAmount,
        Converter={StaticResource AmountForegroundConverter},
        ConverterParameter=10000}"></TextBlock>
```

Figure 3.12 Using converter in page

As you see in figure (Figure 3.13), it is in desired way. Converter is working for both amount fields.

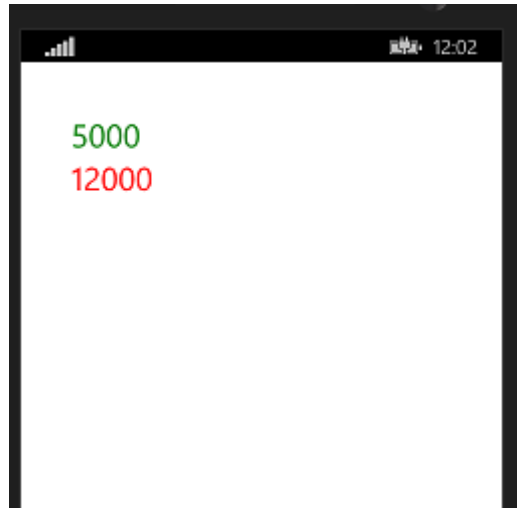


Figure 3.13 Result of implemented converter

### 3.5 MVVM Pattern

MVVM (Model-View-ViewModel) is an architectural pattern for software development. Especially, for development that used binding operation, it should use for whole projects.

For dynamic structure, readable code and high-performance product using any suitable pattern is a requirement. In windows phone development, lots of developers are used MVVM, because of doing two-way binding easily and creating a dynamic project.

In this part, MVVM pattern properties will be explained and Login Example will be implemented using this architecture.

#### 3.5.1 Pattern Properties

As defined in pattern name there are three main parts in this architecture:

- Model: is a class that used for storing data. All properties are defined in this class. Properties have to call PropertyChanged event on setter method for being able to do a two-way binding. Each model should be related with the

type of data. For Login Model, user properties should be initialized. Like username, password, email or whatever needs for login operation.

View: is a page that keeps display area. It is an XAML page for windows phone development. It takes data from a model and display it to a user with binding operations. As MVVM pattern, there should be no other operations in XAML page except display tools.

ViewModel: is a class which provides connect model and view to each other. Also, operations and validations are done in this class.

According to this pattern, ViewModel classes include more than one Model classes, but View must have just one ViewModel.

### ***3.3.2 Delegate Command***

According to MVVM pattern, operations should be done in ViewModel classes. But passing actions to view model is not allowed. Events for action tools like button, combo box, radio button are not used in binding. For doing this, Command property is used.

Command property is already initialized in action tools. But a type of this operation is ICommand and cause of this event that bounded have to be inherited from ICommand. For preventing rework, delegate command class should be created. Then, ICommand properties may be used to fire required event

Delegate Command Class must be inherited from ICommand Class and there are mandatory methods which need to implement cause of ICommand class:

- CanExecute: Validation for executing operation. According to this event Command will be executed or not.
- Execute: The main method that runs Command operation.
- CanExecuteChanged: According to execute situation, developers want to handle this method. It is fired when execute situation is changed.

- Action Property: there should be an action property that will be fired in execute method.

Usage of delegate command is so easy after defining class. Just create an ICommand property and call DelegateCommand in get method. As you see in figure (Figure 3.14) DelegateCommand will take only a method. This method will be called after any events that command binding done in.

```
public ICommand CommandExample
{
    get { return new DelegateCommand(DoOperation); }
}

private void DoOperation(object parameter)
{
}

public class DelegateCommand : ICommand
{
    private Action<object> executeAction;
    public DelegateCommand(Action<object> action)
    {
        executeAction = action;
    }

    //interface method, called when CanExecuteChanged event handler is fired
    public bool CanExecute(object parameter)
    {
        return true;
    }
    public event EventHandler CanExecuteChanged;

    //interface method
    public void Execute(object parameter)
    {
        executeAction(parameter);
    }
}
```

Figure 3.14 Delegate command and its usage



### 3.5.3 Login Example Using MVVM Pattern

Steps for doing Login Operation with using MVVM Pattern will be defined step by step.

- Defining Model Class: Model class will keep property of Login Page. These properties will be user name and password for this example.  
\*\*Never forget to raise NotifyPropertyChanged event.

```
public class LoginModel:INotifyPropertyChanged
{
    private string userName;
    public string UserName
    {
        get { return userName; }
        set
        {
            userName = value;
            NotifyPropertyChanged("UserName");
        }
    }

    private string password;
    public string Password
    {
        get { return password; }
        set
        {
            password = value;
            NotifyPropertyChanged("Password");
        }
    }
    public event PropertyChangedEventHandler PropertyChanged;
    internal void NotifyPropertyChanged(String info)
    {
        if (PropertyChanged != null)
        {
            PropertyChanged(this, new PropertyChangedEventArgs(info));
        }
    }
}
```

Figure 3.15 Defining model class for mvvm pattern

- Defining ViewModel Class: ViewModel class will include defining model and doing login operation. DelegateCommand will use binding button command. Details about DelegateCommand have been defined in previous part of this document.

```

public class LoginViewModel
{
    public LoginViewModel()
    {
        loginModel = new LoginModel { UserName = "ilker.satmaz" };
    }

    private LoginModel loginModel;
    public LoginModel LoginModel
    {
        get { return loginModel; }
        set
        {
            loginModel = value;
            NotifyPropertyChanged("LoginModel");
        }
    }

    public ICommand LoginCommand
    {
        get { return new DelegateCommand(DoOperation); }
    }
    private void DoOperation(object parameter)
    {
        MessageBox.Show(string.Format("User name:{0} \nPassword: {1}",
            LoginModel.UserName, LoginModel.Password));
    }
}

```

Figure 3.16 Defining viewmodel class for mvvm pattern

According to figure (Figure 3.16), LoginModel and LoginCommand are included in viewmodel. There is only last step for completing operation.

- Implementing View: ViewModel will be set as datacontext of page and binding operations will be done.

```

public partial class LoginPage
{
    // Constructor
    public LoginPage()
    {
        InitializeComponent();
        var loginViewModel = new LoginViewModel();
        DataContext = loginViewModel;
    }
}

```

Figure 3.17 Including viewmodel to the page

Recognized that there is no operation in LoginPage scope except setting viewmodel to datacontext. This operation also can be done in XAML page too. Choose whichever you want.

```
<StackPanel Margin="50 50">
    <TextBlock Text="UserName:"/>
    <TextBox FontSize="32" Text="{Binding LoginModel.UserName,Mode=TwoWay}"/>
    <TextBlock Text="Password:"/>
    <PasswordBox FontSize="32" Password="{Binding LoginModel.Password,Mode=TwoWay}"/>
    <Button Content="Login" Command="{Binding LoginCommand,Mode=TwoWay}"/>
</StackPanel>
```

Figure 3.18 Binding Model Items and Command

As you see, there are only binding operations and toolbox items, more readable code and high performance on project. Also, there is no need to set items again on change or button event, model was updated cause of using two way binding

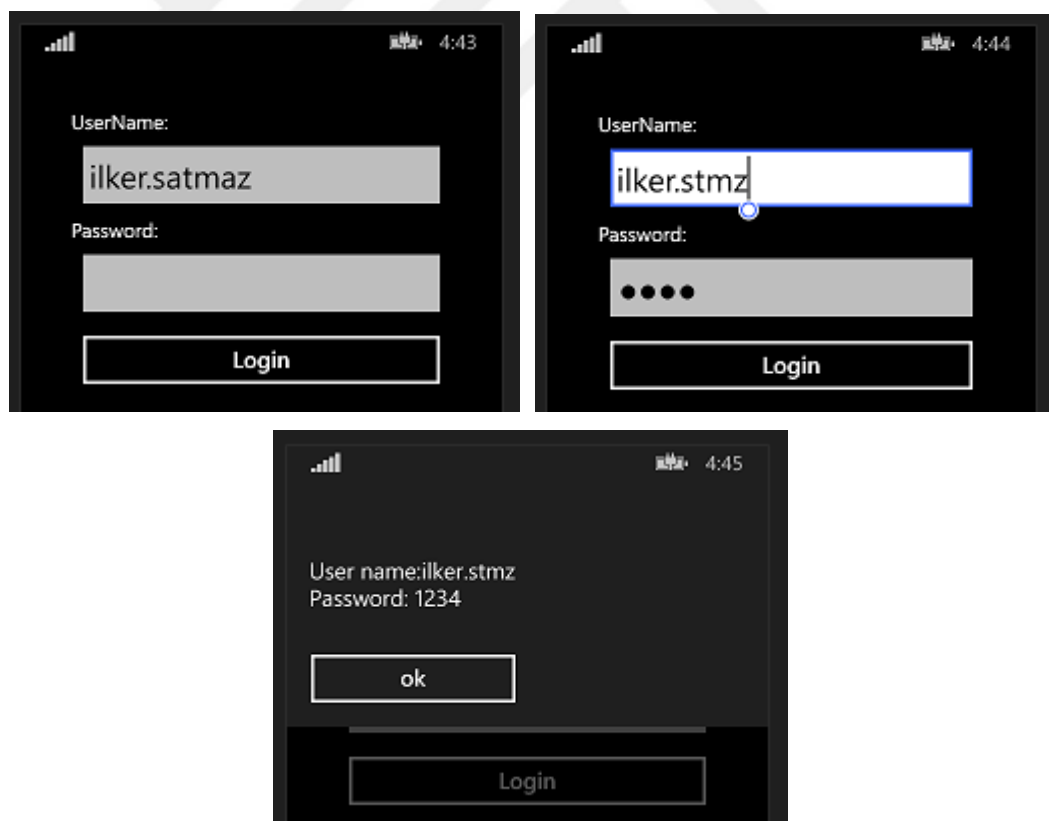


Figure 3.19 Result of using mvvm pattern

## CHAPTER FOUR

### USAGE OF LIBRARY AND USERCONTROLS

#### 4.1 Including Library to Specific Project

For being able to use library, you need to add a library in your project. You may include this library with adding as a reference in the project.

- Add reference of library dll file to the project

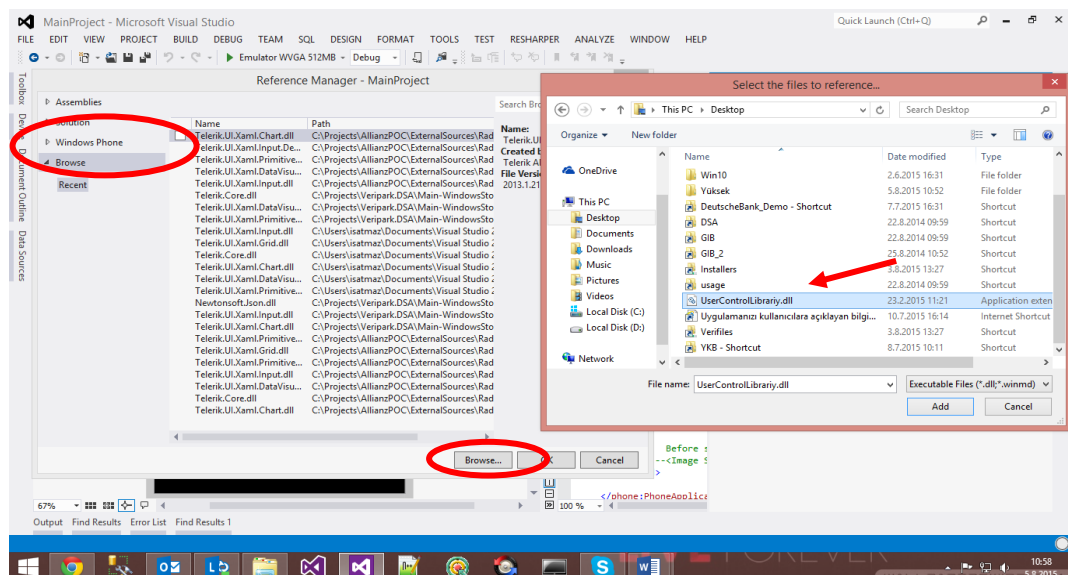
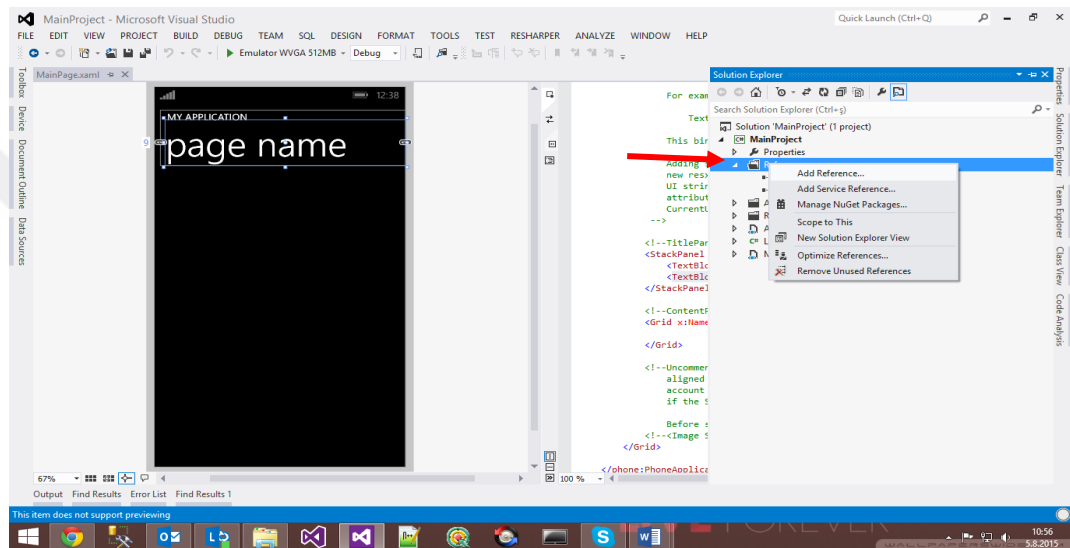


Figure 4.1 Add reference the library to project

- Open Toolbox, right click and then click Add Tab button. Name your tab that will include user controls

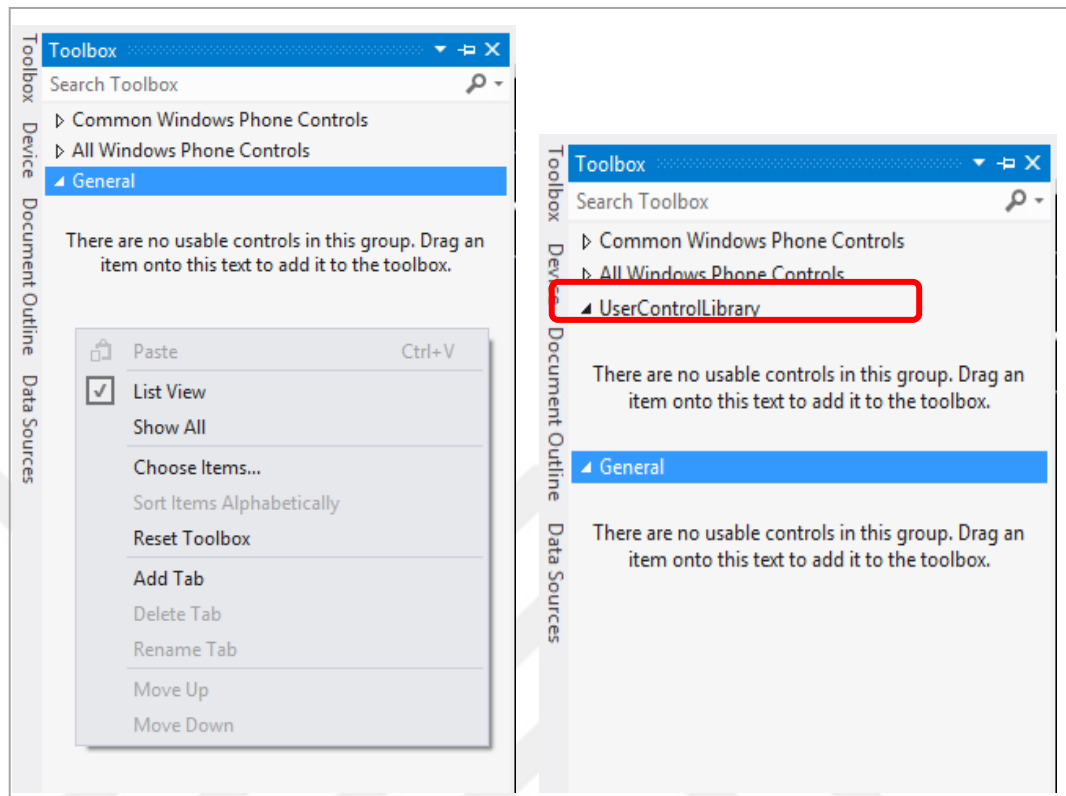


Figure 4.2 Adding new tap to toolbox

- Right click to new tab and click to “Choose items...”

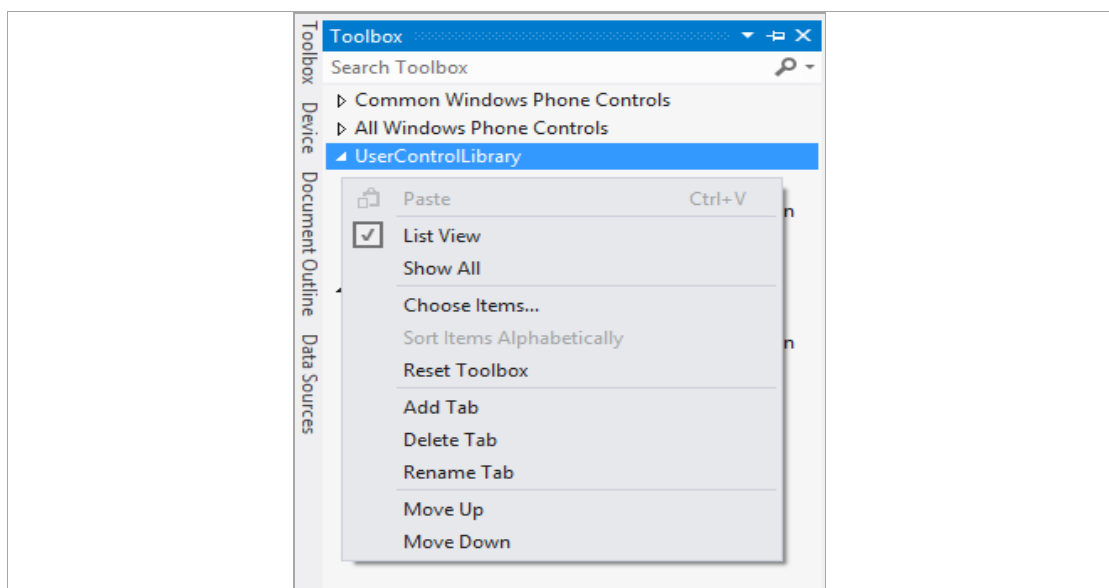


Figure 4.3 Choosing item to new tab

- Browse dll that already included in project. CustomControls will automatically be loaded and selected. You may select specific ones or include all.

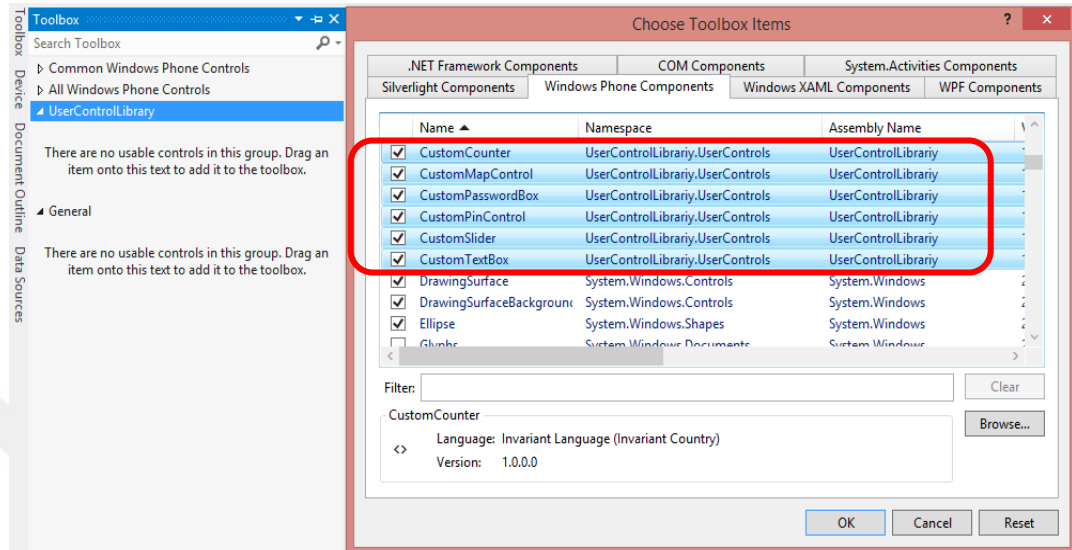


Figure 4.4 Adding controls to the new tab

- Now, you have controls implemented on library and you can use them in your project.

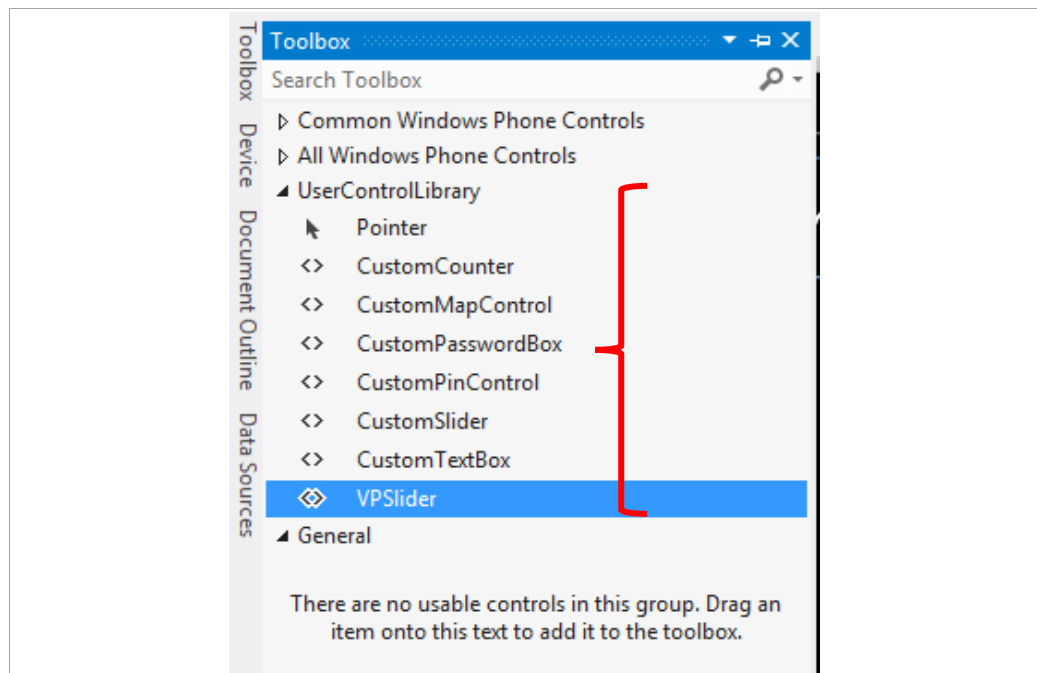


Figure 4.5 New controls added from library

## 4.2 Usage of CustomTextBox

For CustomTextBox there are six fields need to be initialize:

- InputScope: will be used for initializing type of keyboard
- MaxLength: For defining max length of input entrance will be blocked when length of input equal to maxlength
- TextBoxStyle: There is a style property for being able change style of textbox. You may use any style that defined for text box in your project or you can use default text box initialized in Library.
- TextBoxTitle: There is an optional field. For including a title to text box or not. You may have a title of text box with defining it in this property
- TitleStyle: For the condition of adding title to textbox you can change style of title with adding any style that defined for textblock
- IsMandatory: is a boolean property used for defining whether input is mandatory or not. There will be a red star “\*” included on end of title if it is mandatory

```
<userControls:CustomTextBox
    InputScope="Decimal"
    MaxLength="10"
    IsMandatory="True"
    TitleText="Custom TextBox"
    TitleBoxStyle="{StaticResource TextBlockStyle}"
    TextBoxStyle="{StaticResource TextBoxStyle}"/>

<Style x:Name="TextBlockStyle" TargetType="TextBlock">
    <Setter Property="Foreground" Value="#7039A7"></Setter>
    <Setter Property="FontSize" Value="14"/>
</Style>
<Style x:Name="TextBoxStyle" TargetType="TextBox">
    <Setter Property="Foreground" Value="#7039A7"></Setter>
    <Setter Property="FontSize" Value="14"/>
</Style>
```



Figure 4.6 CustomTextBox example

### 4.3 Usage of PasswordTextBox

For this control, there are seven fields which need to be initialized:

- InputScope: will be used for initializing type of keyboard
- MaxLength: For defining max length of input entrance will be blocked when length of input equal to maxlength
- TextBoxStyle: There is a style property for being able change style of textbox. You may use any style that defined for text box in your project or you can use default text box initialized in Library.
- TextBoxTitle: There is an optional field. For including a title to text box or not. You may have a title of text box with defining it in this property
- TitleStyle: For the condition of adding title to textbox you can change style of title with adding any style that defined for textblock
- IsMandatory: is a Boolean property used for defining whether input is mandatory or not. There will be a red star “\*” included on end of title if it is mandatory
- MaskChar: Defines mask character for displaying password to user. Like “\*\*\*\*\*” or “@@@@@” etc.

```
<userControls:CustomPasswordBox
    InputScope="OnlyNumber"
    MaxLength="10"
    IsMandatory="False"
    TitleText="Custom PasswordBox"
    TitleBoxStyle="{StaticResource TextBlockStyle}"
    TextBoxStyle="{StaticResource TextBoxStyle}"
    MaskChar="*" />

<Style x:Name="TextBlockStyle" TargetType="TextBlock">
    <Setter Property="Foreground" Value="#7039A7"></Setter>
    <Setter Property="FontSize" Value="18"/>
</Style>
<Style x:Name="TextBoxStyle" TargetType="TextBox">
    <Setter Property="Foreground" Value="#7039A7"></Setter>
    <Setter Property="FontSize" Value="18"/>
</Style>
```



Figure 4.7 CustomPasswordBox example



## 4.4 Usage of Slider

Slider has seven fields which need to be initialized:

- **SliderType:** Defines whether a slider used for amount values or normal values:
  - **Amount:** used for formatting values as an amount format
  - **Normal:** used for using values as user entrance
- **IsInteger:** Defines to format values as integer or float
  - **True:** In situation of selecting “true” for this value output will be “15.00”
  - **False:** In this option output will be “15”
- **Minimum:** Used for defining minimum value of slider
- **Maximum:** Used for defining maximum value
- **Amount:** This property used for keeping selected value of a slider. Also, predefined condition thumb will be moved to selected value on a slider
- **Prefix:** Especially for an amount sliders. This property may be used for adding currency to end of amounts.
- **LargeChange:** This property used to define changes of value in each slide. It may be useful when developers want to prevent specific values. Think about large change is 50. That means that use can be select values that multiplied of 50
- There are two ways on a slider according to thumb. Left of thumb means this part is already selected and right part means you can slide to a slider on this way. So designs need to verify these two parts on different design.
  - **TrackStyle:** This property used to change style of right part of thumb
  - **FillStyle:** Defines style of left part of thumb
  - **ThumbStyle:** This property defines thumb style

Have a look below screen shots for having idea about how style properties can be defined. Track and Fill Properties are inherited from Rectangle and Thumb inherited from Border. As a result of that you need to define like this:

```

<Style x:Key="HorizontalTrackStyle" TargetType="Rectangle">
  <Setter Property="Fill">
    <Setter.Value>
      <ImageBrush Stretch="Fill" ImageSource="/Assets/Backgrounds/slider-track.png"/>
    </Setter.Value>
  </Setter>
</Style>

<Style x:Key="HorizontalFillStyle" TargetType="Rectangle">
  <Setter Property="Fill">
    <Setter.Value>
      <ImageBrush Stretch="Fill" ImageSource="/Assets/Backgrounds/slider-back.png"/>
    </Setter.Value>
  </Setter>
</Style>

<Style x:Key="HorizontalCenterElementStyle" TargetType="Border" >
  <Setter Property="Background">
    <Setter.Value>
      <ImageBrush Stretch="Fill" ImageSource="/Assets/Backgrounds/thumb.png"/>
    </Setter.Value>
  </Setter>
</Style>

```

Figure 4.8 Style properties for custom slider

```

<userControls:CustomSlider
  Minimum="100"
  Maximum="1500"
  Prefix="USD"
  Amount="200"
  SliderType="Amount"
  LargeChange="75"
  IsInteger="False"
  TrackStyle="{StaticResource HorizontalTrackStyle}"
  FillStyle="{StaticResource HorizontalFillStyle}"
  ThumbStyle="{StaticResource HorizontalCenterElementStyle}"/>

```

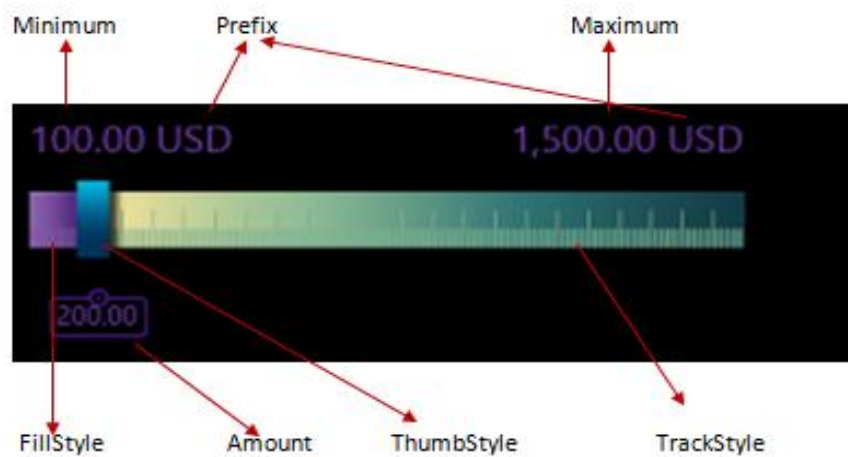


Figure 4.9 CustomSlider example

## 4.5 Usage of MapControl

There are just three properties that can be set on Xaml.

- **IsTrackTrace:** Used for being able to create trace on a map with given coordinates.
  - Yes: given coordinate will be combined as a line on a map
  - No: you will be just add push pin to given coordinates
- **PushpinStyle:** Defines style of push pin will be used on a map
- **SelectedCoordinate:** Used for defining initial coordinate of a map

Actually, methods that have been mentioned on implementation part are important for this control. We will use `GetCoordinate` method for giving an example. For more information you can check implementation part. Methods are defined clearly there.

```

<userControls:CustomMapControl x:Name="CustomMapControl"
    IsTrackLocation="True"
    PushpinStyle="{StaticResource PushpinStyle}" />

<Style TargetType="toolkit:Pushpin" x:Key="PushpinStyle">
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="toolkit:Pushpin">
                <Image Source="../Assets/Icons/Pin.png" Stretch="None"/>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

CustomMapControl.SelectedCoordinate=new GeoCoordinate
{
    Longitude = 27.13696779999998,
    Latitude = 38.4307233
};
CustomMapControl.GetLocation();

```



Figure 4.10 CustomMapControl example

## **CHAPTER FIVE**

### **CONCLUSION AND FUTURE WORKS**

After having a look at implementation and usage of this library, you will be sure the project was prepared on a big time schedule. Any developer that wants to decrease his development time may use this library. As you see it is a dynamic library especially for using it according to your specific design. Font size, font style, pictures, backgrounds and other properties of each control can be set by you.

This control developed for Windows Phone 8. After Microsoft released Windows 8.1. Phone SDKs are updated too. Properties like adding a title to password box and text box are added to controls by default. But still other problems are missing on latest SDK. Library was tested on both Phone 8 and Phone 8.1 development. There is no problem using it in both platforms.

Also, there was a big lack on windows development. You need to develop tablet and phone project separately. But on windows 8.1, Microsoft announced the Universal Application, you will be designed just one project for both environment.

Projects may be defined separately or common pages may be placed in Shared project. For the reason that this project is about Windows Phone, you can safely reference it to Windows Phone 8.1 project and use it in separate pages. There are so many user guides for Windows 8.1 Universal App. Conditional compile may be used for common projects for being able to write condition each project. So that you can use this library in common pages with using Conditional compile.

There are so many confusions about using Windows 8.1 Universal App. But then Microsoft release Windows 10 and give a big deal with Universal App. Now you can develop one and run in multiple platforms like Windows Phone, Windows Tablet, Desktop, Xbox, and hololens. It is a new structure called as UWP (Universal Windows Platform).

Now, upgrading the library to Windows 10 and updating each controls using adaptive layout will be next. Finally, another library will be provided that may be used on Windows 10 Universal App. Also, after developing Universal App any control that have an issue or spend so many times to be developed will be included on my library.



## REFERENCES

- Channel9 (2013). *Windows Phone 8 development course*. Retrieved January 23, 2014, from <https://channel9.msdn.com/Series/Windows-Phone-8-Development-for-Absolute-Beginners>
- Clark, J. (2010). *Introduction to XAML with WPF*. Retrieved October 22, 2014, from <http://www.jeremybytes.com/downloads/introductontoxaml.pdf>
- Code Project (2009). *Dependency Property implementation*. Retrieved September 8, 2009, from <http://www.codeproject.com/Articles/42203/How-to-Implement-a-DependencyProperty>
- Enabling HYPER-V* (2014). Retrieved January 30, 2015, from <http://blogs.technet.com/b/canitpro/archive/2014/03/11/step-by-step-enabling-hyper-v-for-use-on-windows-8-1.aspx>
- James, B. & Lalonde, L. (2015). *Pro XAML with C#*, Retrieved October 22, 2014, from [http://philkildea.co.uk/james/books/Pro\\_WPF\\_CSharp\\_2010\\_dotNET\\_4](http://philkildea.co.uk/james/books/Pro_WPF_CSharp_2010_dotNET_4)
- Microsoft Developer Network (MSDN) (n.d.a). *Dependency Property properties*. Retrieved April 28, 2014, from [https://msdn.microsoft.com/en-us/library/windows/apps/cc221408\(v=vs.105\).aspx](https://msdn.microsoft.com/en-us/library/windows/apps/cc221408(v=vs.105).aspx)
- Microsoft Developer Network (MSDN) (n.d.b). *Input scope index for Windows Phone 8*. Retrieved March 20, 2015, from [http://msdn.microsoft.com/en-us/library/windows/apps/hh393998\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windows/apps/hh393998(v=vs.105).aspx)
- Microsoft Developer Network (MSDN) (n.d.c). *Map Control description*. Retrieved April 09, 2015, from [https://msdn.microsoft.com/en-us/library/windows/apps/jj207037\(v=vs.105\).aspx](https://msdn.microsoft.com/en-us/library/windows/apps/jj207037(v=vs.105).aspx)

Microsoft Developer Network (MSDN) (2014). *Slider control design*. Retrieved January 14, 2015, from [http://msdn.microsoft.com/en-us/library/windows/apps/hh202877\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windows/apps/hh202877(v=vs.105).aspx)

Social Microsoft Developer Network (MSDN) (2013). *Custom slider*. Retrieved January 15, 2015, from <https://social.msdn.microsoft.com/forums/windowsapps/en-us/b3befbf9-2f75-4126-8ebf-f5a38734a988/custom-slider-under-wp8>

StackOverflow (2012a). *Open numeric keyboard in Windows Phone*. Retrieved March 20, 2015, from <http://stackoverflow.com/questions/9690708/open-numeric-only-keyboard-in-windows-phone>

StackOverflow (2012b). *Making custom slider*. Retrieved March 21, 2015, from <http://stackoverflow.com/questions/26892998/windows-phone-8-how-to-make-slider-thumb-invisible-or-disable-it>

StackOverflow (2014a). *Adding pushpin control*. Retrieved April 23, 2015, from <http://stackoverflow.com/questions/22836551/windows-phone-maps-control-how-to-add-layers-and-pushpin>

StackOverflow (2014b). *Pushpin operation on map control*. Retrieved April 23, 2015, from <http://stackoverflow.com/questions/21295870/image-as-pushpin-on-maps-windows-phone-8>

Windows Development (n.d). *Download phone SDK*. Retrieved February 15, 2015, from <https://dev.windows.com/en-us/develop/download-phone-sdk>