**CHAPTER 05**

user interaction using controls

# Xamarin.Forms Views

Một số views cơ bản của Xamarin.Forms

* *Picker*: Hộp thoại phục vụ cho việc chọn một giá trị từ một danh sách đơn giản.
* *DataPicker:* Hộp thoại cho việc chọn thời gian ngày, tháng và năm.
* *TimePicker:* Hộp thoại cho việc chọn thời gian giờ, phút và AM/PM.
* *Stepper:* Nút tăng và giảm giá trị để xác định giá trị cho những giá trị rời rạc.
* *Slider:* Xác định đầu vào bằng dạng trượt cho các giá trị liên tục.
* *Switch:* Điều khiển với giá trị On/Off.

Xamarin.Forms views cung cấp một tập hợp các điều khiển được cài đặt và mở rộng giống hệt như những điều khiển tương ứng trong iOS và Android.

Xamarin.Forms views cung cấp giá trị được chọn ở 2 nơi: Một ở thuộc tính sự kiện cung cấp giá trị gần hiện tại nhất, và một ở thuộc tính dùng chung trên view cung cấp các giá trị được chọn để sử dụng xuyên suốt các trang. Bạn sẽ tạo 2 labels để thể hiện 2 loại giá trị trên : EventValue và PageValue.

<StackLayout HorizontalOptions="Center">

<Label x:Name="EventValue" Text="Value in Event" />

<Label x:Name="PageValue" Text="Value in Page" />

</StackLayout>

## Picker

Picker view cung cấp một hộp thoại sổ xuống để chọn một giá trị từ một danh sách đơn giản.

**Notes :** *Picker* ***View***  *được sử dụng cho* trường hợp lựa chọn đơn giản với các từ ngắn và cụm từ hoặc số. Những danh sách phức tạp với sự kết hợp của nhiều ô chứa nhiều chuỗi và hình ảnh được giới thiệu trong những chương sau.

Đầu tiên, Tạo picker và gán cho nó tiêu đề, “Options,” để xác định cái mà người dùng cần chọn:

<Picker x:Name="ThePicker" Title="Options" SelectedIndexChanged="Picker SelectedIndexChanged">

</Picker>

Sau đó, Thiết lập *ItemsSource của*picker với các options:

<Picker x:Name="ThePicker" Title="Options" SelectedIndexChanged="Picker SelectedIndexChanged">

###### <Picker.ItemsSource>

**<x:Array Type="{x:Type x:String}">**

**<x:String>First</x:String>**

**<x:String>Second</x:String>**

**<x:String>Third</x:String>**

**<x:String>Fourth</x:String>**

**</x:Array>**

**</Picker.ItemsSource>**

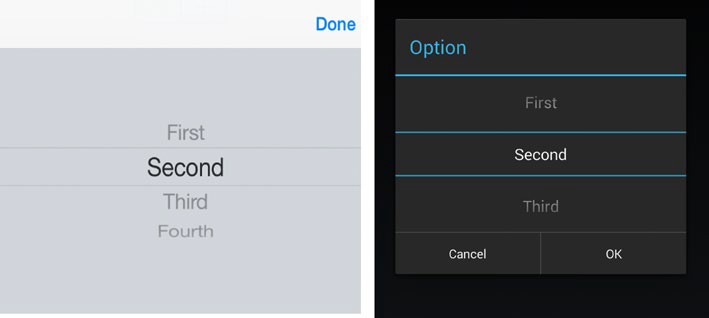
</Picker>

Các tên Option được đặt vào một danh sách và sau đó thêm vào bộ sưu tập của *ItemsSource* trong picker.

Một *Picker*đầu tiên sẽ hiển thị như một *Entry* Control, hiện thị giá trị của thuộc tính *Title* như hình dưới đây:



Khi chạm vào trường *Entry* ***(****Option)*  này, một hộp thoại xuất hiện chứa danh sách các items được định nghĩa trong phần ItemsSource của Picker như hình dưới đây:



**Tip :** Chỉ số giá trị được chọn trong *ThePicker.SelectedIndex* Property là một số nguyên bắt đầu từ không. Nếu *Cancel* được chọn, *SelectedIndex* sẽ được duy trì không thay đổi giá trị đã được chọn trước đó.

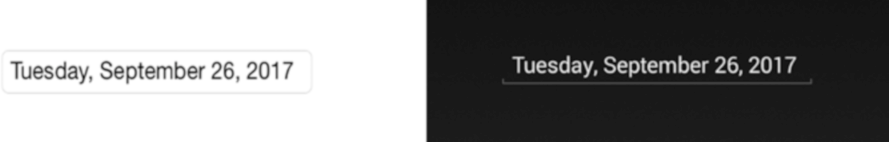
## DatePicker

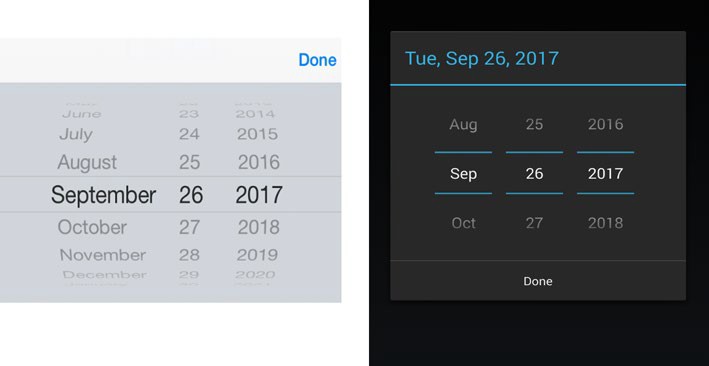
*DatePicker View* tạo ra một hộp thoại sổ xuống để chọn ngày, tháng, và năm. Tạo một *DatePicker* **như thế này :**

<DatePicker x:Name="TheDatePicker" Format="D" VerticalOptions="Center AndExpand" DateSelected="DatePickerDateSelected">

</DatePicker>

*Format* property thiết lập D cho định dạng đầy đủ tháng/ngày/năm. Nhiều định dạng hơn được cung cấp sau này trong phần này.

 *DatePicker View* bắt đầu như một trường nhập dữ liệu, giống như *Xamarin.Forms.Entry**hiển*thị giá trị của thuộc tính*Date.*

Khi trường *Date* được chạm, một hộp thoại sẽ xuất hiện.

Mỗi cột sẽ được cuộn độc lập khi xoay con lăn hoặc phím di chuyển, và các giá trị được làm nổi bật trở thành giá trị được chọn. Khi *Done*  được chạm, giá trị *Date* được chọn sẽ tự động gán vào trường Entry ban đầu. The *DateSelected* event sẽ được gán đến phương thức xử lý sự kiện *DatePickerSelectedIndexChanged.*

<DatePicker x:Name="TheDatePicker" Format="D" VerticalOptions="CenterAnd Expand" **DateSelected="DatePickerDateSelected"**>

Chọn một giá trị bằng cách Picker gọi phương thức *DatePickerDateSelected:*

void DatePickerDateSelected (object sender, DateChangedEventArgs e)

{

EventValue.Text = e.NewDate.ToString(); PageValue.Text = TheDatePicker.Date.ToString();

}

Các thuộc tính *e.NewDate* và *e.OldDate* thì có sẵn trong sự kiện để cung cấp giá trị thời gian cũ và mới. Trong các trường hợp thông thường, tuy nhiên, giá trị được nhập bới người dùng được lưu trữ trong thuộc tính *Date.* Tất cả thông tin sử dụng loại *DateTime.*

Định dạng của trường *Date* có thể thay đổi với thuộc tính *Format – Ví dụ:*

<DatePicker.Format>MM-dd-yyyy</DatePicker.Format>

Các giá trị định dạng khác theo sau đây:

* D: Đầy đủ tháng,ngày, và năm. (Thursday, October 31, 2019)
* d: Tháng, ngày, và năm. (31/10/2019)
* M: Tháng và ngày (October 31).
* Y: Tháng và năm (October 2019).
* yy: 2 chữ số cuối của năm (19).
* yyyy: Đầy đủ 4 chữ số của năm(2019).
* MM: 2 chữ số của tháng(10).
* MMMM: Tên của tháng(October).
* dd: ngày theo định dạng 2 chữ số (31).
* ddd: Viết tắt của ngày trong tuần (Thu).
* dddd: Đầy đủ ngày trong tuần (Thursday).

Bạn thiết lập khoảng thời gian cho sự lựa chọn bằng cách sử dụng *MaximumDate và MinimumDate:*

**Tip :** Trên Android, thuộc tính *Format và MaximumDate/MinimumDate* ảnh hưởng đến thời gian trong trường *DatePicker Entry* nhưng không ảnh hưởng đến thời gian được hiện thị trên hộp thoại lựa chọn thời gian.

## TimePicker

The *TimePicker* view tạo một hộp thoại sổ xuống để lựa chọn giờ, phút, và AM/PM. Tạo một *TimePicker View* như sau:

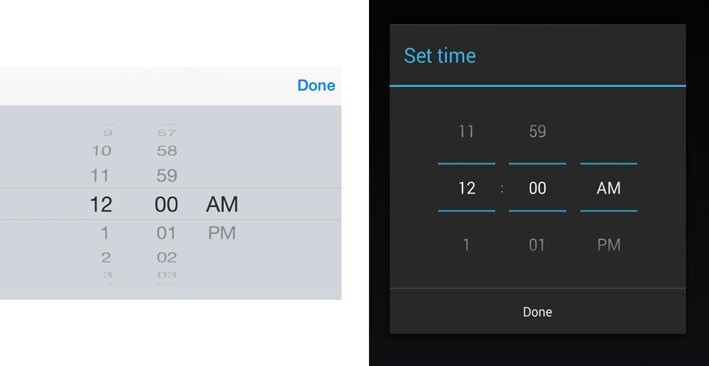
<TimePicker x:Name="TheTimePicker" Format="T" VerticalOptions="Center AndExpand" PropertyChanged="TimePickerPropertyChanged"></TimePicker>

Thuộc tính *Format* thiết lập T để hiển thị định dạng thời gian đầy đủ. Nhiều định dạng hơn sẽ được đề cập ở phần sau.

*TimePicker* view bắt đầu như một trường nhập dữ liệu giống như *Xamarin.Forms.Entry,* Hiển thị giá trị của thuộc tính *Time.*



Khi trường thời gian được chạm, một hộp thoại sẽ xuất hiện.



Mỗi cột cuộn độc lập khi xoay, và giá trị được làm nổi bật sẽ được chọn khi chạm vào nút *Done. Thời gian*  được tự động gán vào trường nhập dữ liệu ban đầu.

*Không có* sự kiện *TimeSelected* cái sẽ được gọi khi giá trị thay đổi mà thay vào đó là sự kiện *PropertyChanged* trong mô hình ràng buộc dữ liệu của Xamarin.Forms để theo dõi sự thay đổi của View này (TimePicker).

<TimePicker x:Name="TheTimePicker" Format="T" VerticalOptions="Center AndExpand" **PropertyChanged="TimePickerPropertyChanged"**>

Phương thức *TimePickerPropertyChanged* được gọi khi thời gian được chọn trong Picker:

void TimePickerPropertyChanged (object sender, PropertyChangedEventArgs e)

{

if (e.PropertyName == TimePicker.TimeProperty.PropertyName)

{

PageValue.Text = TheTimePicker.Time.ToString();

}

}

*Thuộc tính TimePicker.Time* được thiết lập với giá trị được chọn như loại *TimeSpan.*

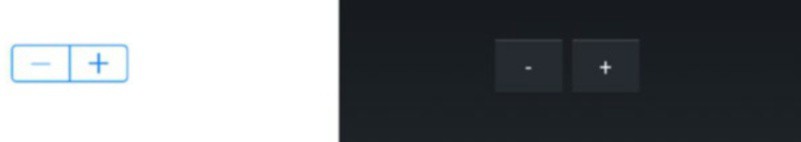
*Định dạng* của trường *Time* có thể được thay đổi với thuộc tính *Format*. Các loại định dạng khác có thể chỉ định:

* T: Thời gian đầy đủ với giờ, phút, giây và AM/PM(9:30:25 AM).
* t: Đầy đủ thời gian với giờ, phút, và AM/PM (9:30AM).
* hh: Chỉ có giờ ở định dạng 2 chữ số (09).
* mm: Chỉ có phút ở định dạng 2 chữ số (30).
* ss: Giây ở định dạng 2 chữ số (25) ; giây không thể chọn được trong hộp thoại chọn thời gian.
* tt: định dạng thời gian 12 tiếng AM/PM.

## Stepper

*Stepper* view tạo ra nút tăng và giảm để điều chỉnh giá trị rời rạc:

<Stepper x:Name="TheStepper" Minimum="0" Maximum="10" Increment="1" HorizontalOptions="Center" VerticalOptions="CenterAndExpand" ValueChanged="StepperValueChanged"></Stepper>

*Minimum, Maximum, và Increment* thiết lập cho giá trị của *Stepper.* ***Giá trị bắt đầu***  có thể được xác định bằng thuộc tính *Value.*

Chạm vào nút cộng hoặc trừ sẽ thay đổi giá trị và gọi sự kiện *ValueChanged, StepperValueChanged* được gọi trong trường hợp này.

void StepperValueChanged (object sender, ValueChangedEventArgs e)

{

EventValue.Text = String.Format("Stepper value is {0:F1}", e.NewValue);

PageValue.Text = TheStepper.Value.ToString();

}

Thuộc tính *e.OldValue và e.NewValue* thì có sẳn trong sự kiện để cung cấp giá trị được chọn cũ hoặc mới. Trong các trường hợp thông thường , giá trị được nhập bởi người dùng sẽ được lưu trữ trong thuộc tính *Value* của Stepper. Tất cả thuộc tính này đều có kiểu dữ liệu là *Double.*

## Slider

*Slider View* là một điều khiển nhập liệu dạng trượt để xác định các giá trị liên tục (Khác với Stepper là giá trị rời rạc):

<Slider x:Name="TheSlider" Minimum="0" Maximum="100" Value="50" VerticalOptions="CenterAndExpand" ValueChanged="SliderValueChanged" WidthRequest="300"></Slider>

*Minimum và Maximum* được thiết lập để giới hạn khoảng giá trị có thể được chọn bởi người dùng. Giá trị bắt đầu của khoảng có thể được xác định bằng thuộc tính *Value. Giá trị thay đổi*  với khoảng cách là 0.1 mỗi khi nút trược di chuyển. Thuộc tính *WidthRequest* thiết lập chiều rộng cho Slider mà không làm thay đổi *Minimum hoặc Maximum của* Slider.



Thao tác trượt Slider sẽ thay đổi giá trị và gọi sự kiện *ValueChanged,* phương thức *SliderValueChanged* sẽ được gọi để xử lý sự kiện trong trường hợp này.

void SliderValueChanged (object sender, ValueChangedEventArgs e)

{

EventValue.Text = String.Format("Slider value is {0:F1}", e.NewValue);

PageValue.Text = TheSlider.Value.ToString();

}

Các thuộc tính *e.OldValue và e.NewValue* có sẳn trong biến sự kiện để cung cấp giá trị được chọn cũ và mới. Trong trường hợp thông thường, giá trị trượt được xác định sẽ được lưu trữ trong thuộc tính *Value*. Tất cả các thuộc tính này có kiểu dữ liệu *Double.*

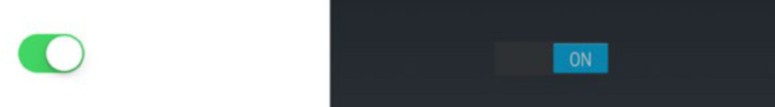
## Switch

*Switch View* là điều khiển cho phép xác định giá trị luận lý ON/OFF:

<Switch x:Name="TheSwitch" HorizontalOptions="Center" VerticalOptions=" CenterAndExpand" Toggled="SwitchToggled"></Switch>

Hình ảnh switch khi OFF



 Hình ảnh swithc khi ON

Chạm vào switch để chuyển đổi trạng thái của giá trị và gọi sự kiện *Toggled*, phương thức xử lý *SwitchToggled* sẽ được gọi để xử lý trong trường hợp này.

void SwitchToggled (object sender, ToggledEventArgs e)

{

EventValue.Text = String.Format("Switch is now {0}", e.Value ? "On" : "Off");

PageValue.Text = TheSwitch.IsToggled.ToString();

}

Thuộc tính e.Value có sẳn trong sự kiện để cung cấp giá trị được chọn mới. Trong các trường hợp thông thường, giá trị cũng được lưu trữ trong thuộc tính *IsToggled.*Tất cả thuộc tính trên có kiểu dữ liệu là *Boolean.*

## Scale, Rotation, Opacity, Visibility, and Focus

Bạn có thể thay đổi giao diện và hành vi của các Xamrin.Forms View bằng các sử dụng các thành viên của View superclass, *VisualElement.* Ở đây là một số thuộc tính khóa có thể được thiết lập trên View:

* *Scale:* Thay đổi kích thướn của một View mà không ảnh hưởng đến các View xung quanh nó. Giá trị mặc định là 1.0.
* *IsVisible:* Chuyển đổi giữa trạng thái ẩn và hiện của View. Mặc định False.
* *IsEnabled:* Vô hiệu hóa hoặc Kích hoạt một View. Mặc định False.
* *Opacity:* Làm phai màu, nhạt màu View. Mặc định là 1.0.
* *Rotation:*Xoay View , có thể xoay trên tất cả các trục bằng các sử dụng *Rotation, RotationX, và RotationY*. Những thuộc tính này xoay View xung quanh 1 điểm được xác định bởi *AnchorX và AnchorY.*

Sử dụng C#, Gán *focus* đến một View bằng cách sử dụng *Focus() method,* phương thức này trả về *true* nếu thành công.

var gotFocus = entry.Focus();

Ví dụ trên xác định *focus (sự chú ý)* trên một *Entry* View.

**Tip :** Sử dụng các hoạt ảnh mở rộng trên các điều khiển của Xamarin.Forms và các thành phần để xử lý fades, rotation, scaling, và translation. Các phương thức thực hiện các hoạt ảnh mở rộng bao gồm *TranslateTo, ScaleTo, RelScaleTo, RotateTo, RelRotateTo, RotateXTo, RotateYTo, và FadeTo. Ví dụ* : Làm mờ một hình ảnh trong 4 giây như sau: *await image.FadeTo(1,4000);*

# Custom Controls