**CHAPTER 7: NAVIGATION**

Navigation cung cấp cho người dùng những gì họ cần để có được một ứng dụng nhanh chóng, di chuyển từ màn hình để tự tin và dễ dàng sàng lọc. Điều này có thể bao gồm các menu, biểu tượng có thể gõ, nút, các tab và liệt kê các item, cũng như nhiều loại màn hình nhạy cảm với cử chỉ để hiển thị dữ liệu, thông tin và các tùy chọn cho người dùng. Có những cách tiêu chuẩn công nghiệp để làm điều này gọi là mẫu navigation. Mỗi mẫu đó cung cấp một mẫu cho giao diện người dùng và tương tác. Những mẫu này có thể được kết hợp và xếp chồng lên nhau và kết nối như Legos để tạo thành bộ xương của toàn bộ ứng dụng. Khi chúng tôi có người dùng di chuyển giữa các màn hình, sau đó có sự xem xét về trạng thái hoặc cách dữ liệu được truyền qua lại giữa trang. Trong chương này, chúng ta sẽ đi sâu vào các mẫu navigation và cách Managing trạng thái navigation.

**Navigation Patterns**

Navigation patterns là các mẫu tiêu chuẩn ngành để buộc các màn hình của ứng dụng lại với nhau một cách thanh lịch và có thể sử dụng. Hai mẫu thiết kế hình ảnh phổ biến nhất trong điện thoại di động navigation là phân cấp và phương thức. Dưới đây là danh sách đầy đủ các mẫu navigation UI di động phổ biến nhất được sử dụng trong phát triển Xamarin:

• Hierarchical: Mẫu navigation dựa trên ngăn xếp cho phép người dùng di chuyển sâu hơn vào một hệ thống phân cấp màn hình và sau đó quay trở lại, một màn hình tại một thời điểm, bằng cách sử dụng các nút Lên hoặc Quay lại.

• Modal: Một màn hình làm gián đoạn navigation phân cấp, thường là một cửa sổ bật lên lên màn hình với một cảnh báo hoặc menu mà người dùng có thể hoàn thành hoặc hủy bỏ.

• Drill-down list: Danh sách các item có thể ghép được chọn để hiển thị chi tiết item.

• Navigation drawer: Một menu navigation trượt từ bên trái bên cạnh một biểu tượng, thường là ba đường ngang được gọi là “hamburger” ở góc trên bên trái của màn hình.

• Tabs : Một thanh chứa một số nút giống như thư item ở trên cùng hoặc dưới cùng của màn hình, mỗi biểu tượng có thể gõ hoặc văn bản gọi các trang mới.

• Springboard : Còn được gọi là bảng điều khiển , đây là một mạng lưới có thể khai thác biểu tượng gọi các trang mới.

• Carousel : Các tấm có kích thước màn hình trượt theo chiều ngang và đôi khi chứa hình ảnh lớn.

**Hierarchical**

Hierarchical là một mẫu dựa trên ngăn xếp cho phép người dùng di chuyển xuống một chồng màn hình và sau đó bật ra một lần nữa, mỗi lần một màn hình. Mẫu này thường sử dụng một thanh công cụ ở phía trên màn hình để hiển thị nút Lên (Hình 7-1 ) ở phía trên bên trái bất cứ khi nào một trang được chọn hoặc được khoan xuống thành phương tiện bằng bất kỳ phương tiện nào. Khi người dùng khoan sâu hơn vào cấu trúc menu, một ngăn xếp được duy trì, với mỗi trang được đẩy lên trên nó.



Hai nút tựa vào nút Back và nút Up được sử dụng song song để navigation lạc hậu, bật các trang ra khỏi ngăn xếp. Nút Quay lại là biểu tượng mũi tên cong tại phía dưới màn hình trên Android. Mạnh hơn nút Lên, nút Back lại có thể đưa người dùng ra khỏi ứng dụng. iOS không có. Nút Up là biểu tượng nhỏ hơn ở góc trên bên trái (Hình 7-2).



Hierarchical navigation sâu có thể được duyệt theo cách này, với yêu cầu chọn trang việc sử dụng các mẫu navigation UI bổ sung như ngăn kéo navigation, xem chi tiết danh sách, hoặc menu bật lên.

**Modal**

Một modal là một cửa sổ bật lên hoặc màn hình gián đoạn duy nhất, có hai hương vị. Nhất loại phổ biến nổi trên đầu trang chính và thường là một cảnh báo, hộp thoại hoặc menu người dùng có thể phản hồi hoặc hủy bỏ. Navigation trở lại trang gốc khi phương thức bị loại bỏ. Một phương thức thông báo cho người dùng về một sự kiện quan trọng, chẳng hạn như một bản ghi đã lưu hoặc cung cấp cho họ cơ hội cung cấp đầu vào hoặc hướng, chẳng hạn như menu hoặc liệu để cam kết hoặc hủy bỏ một giao dịch. Loại thứ hai, ít phổ biến hơn, phương thức thay thế hoàn toàn trang chính, làm gián đoạn ngăn xếp navigation phân cấp.

Hai menu phương thức phổ biến nhất trong giao diện người dùng di động là ngăn navigation và menu hành động. Các hướng ngăn kéo thường trượt vào từ rời và được kích hoạt bằng cách chạm vào một biểu tượng (thường là icon hamburger ) ở góc trên bên trái màn hình và hiển thị danh sách các trang để navigation đến. Các trình đơn hành động thường trượt hoặc bật lên ở bên phải màn hình và được gọi bằng cách nhấn vào một biểu tượng (thường là ba dọc dấu chấm) ở góc trên bên phải màn hình và chứa hầu hết các thao tác (ví dụ: Yêu thích Điều này), mặc dù ít thường xuyên hơn một số trang navigation là tốt. Để theo UI được thiết lập này mẫu, hãy nhớ quy tắc này: Nav bên trái, Hành động bên phải (Hình 7- 3 ).



Các mẫu navigation UI phân cấp và phương thức thường được sử dụng làm bổ sung kỹ thuật, với hệ thống phân cấp cung cấp bộ xương của cấu trúc navigation và phương thức cho người dùng lựa chọn cho những gì họ muốn làm và nơi họ muốn đi trong ứng dụng cũng như cập nhật thông tin trên đường đi.

Trước khi bạn đi sâu vào tất cả các mẫu, một chủ đề navigation xuyên suốt cần phải được giải quyết: Managing state. Khi người dùng navigation qua một ứng dụng, các màn hình riêng biệt phải xuất hiện để trở thành một phần của toàn bộ ứng dụng, mặc dù mỗi màn hình là một UI riêng với một bộ điều khiển riêng.

**State Management**

State giúp chúng tôi duy trì ảo tưởng về tính nhất quán và liên tục trong khi người dùng navigation giữa các màn hình, thông qua việc chia sẻ dữ liệu trên các màn hình đó. Chúng tôi không lâu hơn trong thế giới web của chuỗi truy vấn, cookie và biến Phiên, nhưng chúng ta phải vẫn duy trì trạng thái trong các ứng dụng di động. Hầu hết các biến được đặt trong một màn hình cụ thể, vì vậy Managing state thường liên quan đến việc truyền dữ liệu qua lại rõ ràng giữa các màn hình. Tham số truyền giữa các màn hình là phương pháp trạng thái được khuyến khích Managing trên tất cả các nền tảng di động, để giảm thiểu rủi ro lạm dụng bộ nhớ và tối đa hóa hiệu suất ứng dụng.

Xamarin.Forms cho phép chúng ta truyền tham số vào hàm tạo ContentPage. Android sử dụng một lớp có tên Bundle, đây là một từ điển chứa các giá trị được thông qua, nằm trong một lớp gọi là Intent, mà chúng ta sử dụng để gọi các hoạt động mới. Nhà phát triển iOS ưu tiên các thuộc tính công cộng trên bộ điều khiển xem đích, nhưng iOS hỗ trợ chuyển tham số vào hàm tạo của trang đích.

Lớp toàn cầu tĩnh là một triển khai C # của mẫu singleton. Nó có sẵn trên tất cả các nền tảng nhưng phải được sử dụng một cách thận trọng; chú ý bộ nhớ thiết bị di động hạn chế. Sự tồn tại của đĩa được tích hợp vào Xamarin.Forms bằng cách sử dụng Ứng dụng Các đối tượng 'Thuộc tính, một từ điển sử dụng các cặp ID / đối tượng.

Bây giờ bạn có cách chuyển các giá trị giữa các trang, hãy bắt đầu với Xamarin. Hình thức navigation.

**Xamarin.Forms Navigation**

Navigation trong Xamarin.Forms dựa trên hai mẫu navigation chính: phân cấp và phương thức.

Hierarchical pattern cho phép người dùng di chuyển xuống qua một chồng trang và sau đó bật lại thông qua chúng bằng cách sử dụng nút Lên hoặc Quay lại. Điều này đôi khi được gọi là khoan xuống hoặc navigation bánh mì .

Modal pattern là một màn hình gián đoạn đòi hỏi một hành động cụ thể từ người dùng nhưng thường có thể được loại bỏ bằng nút Hủy. Ví dụ bao gồm thông báo, cảnh báo, hộp thoại và chỉnh sửa hoặc trang ghi mới.

Trong chương này, bạn sẽ khám phá thứ bậc, phương thức và phần còn lại của navigation hoa văn. Xamarin.Forms cung cấp hầu hết trong số họ:

• Navigation phân cấp bằng NavigationPage

• Phương thức sử dụng NavigationPage, cảnh báo và ActionSheets

• Xem chi tiết danh sách bằng NavigationPage, ListView và TableView

• Navigation drawer bằng MasterDetailPage

• Tabs sử dụng TabbedPage

• Springboard sử dụng hình ảnh với nhận dạng cử chỉ

• Carousel sử dụng CarouselPage

Thành phần navigation Xamarin.Forms phổ biến nhất là NavigationPage, trong đó được dựa trên mô hình phân cấp nhưng cũng cung cấp chức năng phương thức

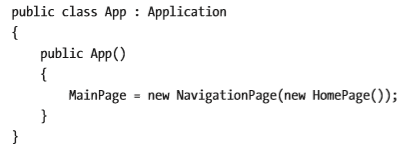
**Hierarchical Navigation Using NavigationPage**

NavigationPage tạo ra một chồng trang đầu tiên / cuối cùng. Các trang có thể được đẩy lên ngăn xếp và sau đó bật ra để trở về trang trước. NavigationPage thường kết thúc chính, hoặc trang chủ, trang. Nó có thể cung cấp một thanh navigation ở phía trên màn hình cung cấp tiêu đề trang, biểu tượng hiện tại và nút Lên (<). Nhân vật 7-4 hiển thị thanh navigation ở đầu màn hình cho iOS và Android.

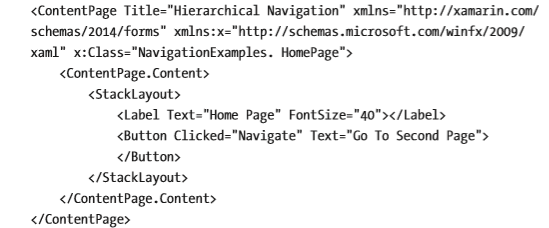


Lưu ý Lên và Quay lại là các nút navigation khác nhau. Lên là mũi tên ít hơn trong góc trên cùng bên trái của trang navigation và nút Quay lại ở dưới cùng thanh navigation được cung cấp bởi oS (mặc dù không có trong ioS).

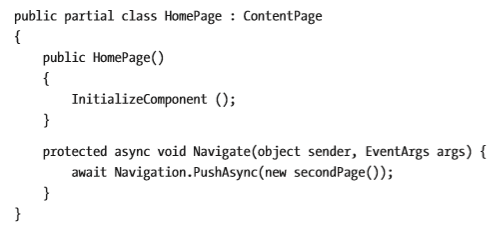
Để sử dụng NavigationPage, trong hàm tạo của lớp Ứng dụng của bạn, hãy khởi tạo một Đối tượng NavigationPage, truyền vào ContentPage tại nhà làm tham số và gán cho nó.



Như thể hiện trong Liệt kê 7-1 , tạo một trang XAML mới có tên HomePage có nhãn tự nhận mình là Trang chủ Trang chủ và một nút.



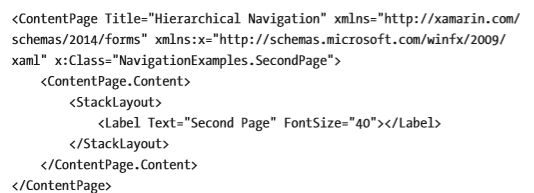
Kết nối sự kiện Nhấp chuột của nút có tên Navigation để đưa chúng tôi đến trang thứ hai thông qua phương thức Navigation.PushAsync như trong Liệt kê 7-2.



Trang chủ là một trang đơn giản với nhãn và nút của bạn, đang chờ để đưa bạn đến trang thứ hai, như trong hình 7-5.



Danh sách 7-3 chứa một trang XAML đơn giản có tên là SecondPage tự dán nhãn Thứ hai Trang. Không có thay đổi là cần thiết trong mã C # phía sau.



Bây giờ một trang được đẩy lên ngăn xếp navigation, thanh navigation sẽ trở thành có thể nhìn thấy, như trong Hình 7-6 .



**Pushing and Popping Screens on the Navigation Stack**

Ba phương pháp được sử dụng để di chuyển giữa các trang theo thứ bậc:

• PushAsync đẩy một trang lên ngăn xếp và đến đó: Navigation.PushAsync (new nextPage ()); Một tham số thứ hai có thể được thêm vào để xác định xem navigation là hoạt hình: Navigation.PushAsync (new nextPage (), bool animated);

• PopAsync bật một trang khỏi ngăn xếp và chuyển đến trang trước:

Navigation.PopAsync ();

• PopToRootAsync bật tất cả các trang khỏi ngăn xếp và đi đến thư item gốc trang:

Navigation.PopToRootAsync();

**Setting the Page Title**

Thuộc tính ContentPage.Title hiển thị một tiêu đề trong thanh navigation. Điều này nên có thể cài đặt trong thuộc tính ContentPage.Title trong XAML.



Tuy nhiên, kết quả có thể được trộn lẫn. Nếu bạn cần kết quả tốt hơn, hãy đặt Tiêu đề trong C # trong Trình xây dựng ContentPage:



Nếu điều đó không hiệu quả, hãy thử điều này trong XAML:



**Customizing the Navigation Bar**

NavigationPage có một số thuộc tính có thể truy cập từ bất kỳ trang con nào, tất cả đều cung cấp truy cập vào các yếu tố của thanh navigation. Thuộc tính thanh navigation như Tiêu đề và Biểu tượng được đặt trong trang con chứ không phải trong trang đã khởi tạo NavigationPage. Đây là phù hợp với kiến trúc nền tảng bản địa.

Trong hầu hết các ví dụ navigation Xamarin.Forms, tệp icon.png đã được được thay thế bằng một hình ảnh trống, vì vậy không có biểu tượng nào được nhìn thấy. Đây là một cái nhìn nạc và hiện đại. Tệp icon.png cũng có thể được thay thế bằng một hình ảnh phù hợp được sử dụng để phản ánh ứng dụng, như trong hình 7-7.



Biểu tượng navigation cũng có thể được đặt động để phản ánh bối cảnh của trang hoặc người dùng, bởi sử dụng SetTitleIcon và thuộc tính Page.Icon:

var image = "icon.png";

NavigationPage.SetTitleIcon (this, image);

Tùy chỉnh thanh navigation liên quan đến nhiều hơn một thay đổi nên xem xét việc sử dụng thuộc tính TitleView mới hơn. TitleView là một khung nhìn được xác định trong Trang navigation trong XAML (<NavigationPage.TitleView />) hoặc dưới dạng một thuộc tính trong lớp trang C #, NavigationPage.TitleView. Đó là một khung nhìn có thể chứa bố cục và các quan điểm khác, chẳng hạn như tiêu đề và biểu tượng.

Tùy chỉnh thêm thanh navigation ở phía trên màn hình được thực hiện bằng cách sử dụng các phương thức NavigationPage này:

• SetTitleView hiển thị TitleView chứa toàn bộ chế độ xem trong thanh navigation để tùy chỉnh.

• SetHasNavlationBar hiển thị / ẩn thanh navigation trên hiện tại trang:

• SetTitleIcon thay đổi biểu tượng tiêu đề (thuộc tính Page.Icon).

• SetHasBackButton hiển thị / ẩn nút Quay lại.

• SetBackButtonTitle thay đổi tiêu đề navigation (Trang.Title thuộc tính được đặt trên trang gọi).

• BarBackgroundColor thay đổi màu của thanh navigation.

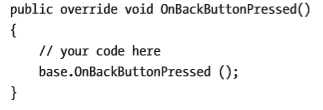
• BarTextColor thay đổi màu văn bản của thanh navigation.

Lưu ý Tại sao chúng tôi sử dụng C # thay vì XaML cho NavigationPage? Bởi vì NavigationPage không có thuộc tính cho trang gốc và XamarinCách tiếp cận được đề xuất là sử dụng C # cho NavigationPage.

**Handling the Back Button**

Có thể thực hiện các trang bật ra khỏi ngăn xếp bằng nút Lên hoặc Quay lại. Lên là biểu tượng nhỏ hơn (<) ở góc trên bên trái của trang navigation và Quay lại nút nằm trên thanh navigation phía dưới.

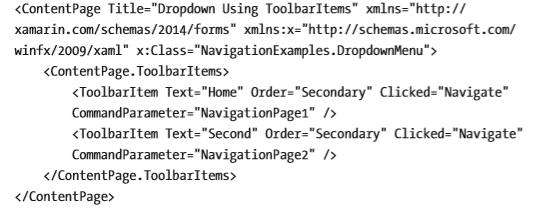
Sự kiện nhấp vào nút Quay lại có thể được xử lý rõ ràng bằng cách ghi đè trang Phương thức OnBackButtonPress:



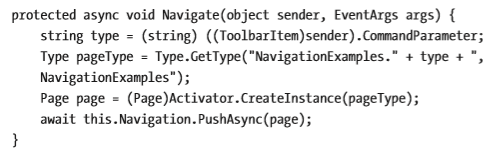
**Creating a Drop-Down Menu**

Một lớp menu thả xuống có tên ToolBarItems được tích hợp vào lớp Trang và hiển thị khi sử dụng NavigationPage.

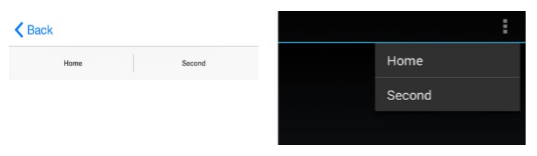
Khởi tạo NavigationPage để gọi thanh công cụ ContentPage, như thể hiện trong Danh sách 7-4.



Đây là phương pháp Navigation trong DropdownMothy.cs là cách triển khai trong số các sự kiện được nhấp của Thanh công cụ. Lấy loại trang từ CommandParameter và khởi tạo một trang mới bằng thủ thuật .NET cũ: (Trang) Activator. CreatInstance (pageType). Sau đó navigation đến đó với Navigation.PushAsync.



Điều này tạo ra một menu thả xuống / thanh công cụ hoặc menu tab với các item Home và Thứ hai. Nhấp vào một trong hai navigation đến trang tương ứng (xem Hình 7-8). Trên iOS, nó trông giống như một menu tab.



**Modal**

Xamarin.Forms cung cấp ba tùy chọn để navigation phương thức:

• NavigationPage cho các chế độ toàn trang

• Alerts for user notifications

• Action sheets for pop-up menus

**Full-Page Modal Using NavigationPage**

Các trang toàn màn hình có thể được tạo để phá vỡ mô hình phân cấp. Khi phương thức các trang được nâng lên, hệ thống phân cấp bị gián đoạn và thanh navigation biến mất. Các thanh navigation quay trở lại khi phương thức được bật ra khỏi ngăn xếp. Hai phương pháp này được sử dụng để di chuyển giữa các trang theo phương thức:

• PushModalAsync đẩy một trang trên ngăn xếp và đến đó:

Navigation.PushModalAsync (new nextPage ());

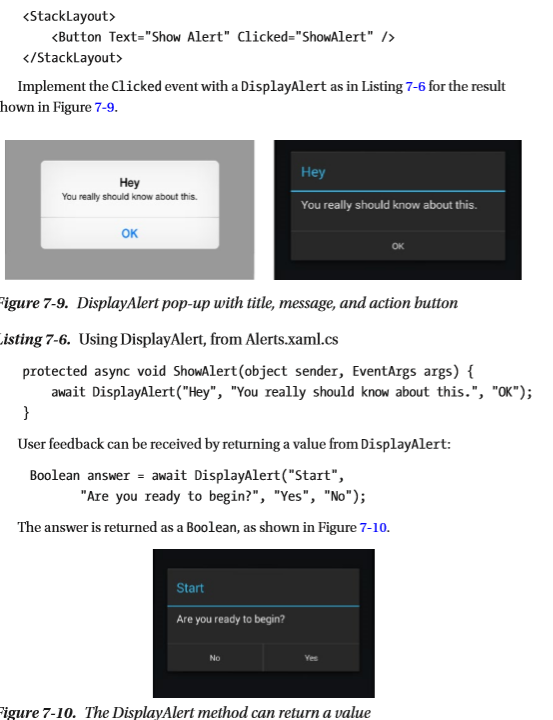
• PopModalAsync bật một trang khỏi ngăn xếp và đi đến trang trước:

Navigation.PopModalAsync ();

**User Notification Using Alerts**

Phương thức DisplayAlert trong ContentPage hiển thị cảnh báo bật lên, như được hiển thị trong Nhân vật 7-6. Điều này thường được sử dụng với async / await vì vậy việc thực thi sẽ tạm dừng cho đến khi cửa sổ bật lên bị xóa (Liệt kê 7-6 ).

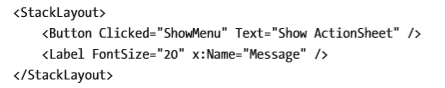
Tạo một trang XAML mới và đặt một nút vào StackLayout. Thêm một sự kiện đã nhấp có chứa các Show Showertert.



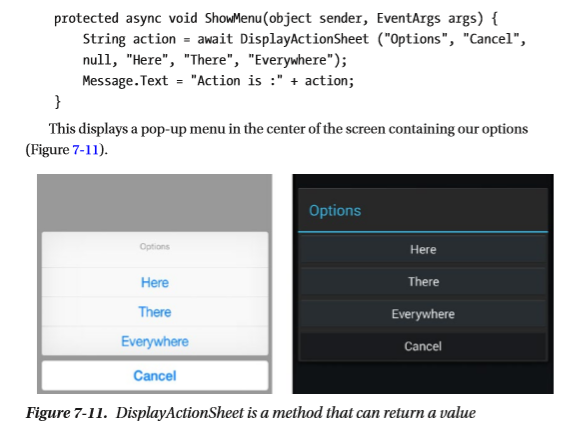
**Pop-Up Menu Using Action Sheets**

ActionSheet cung cấp một menu các tùy chọn trong cửa sổ bật lên và trả về một chuỗi.

Sử dụng DisplayActionSheet, tạo một bảng hành động được kích hoạt bằng một nút bấm gán kết quả cho nhãn, như trong Liệt kê 7-7 .



Triển khai sự kiện Nhấp chuột bằng phương thức ShowMothy có chứa DisplayActionSheet với các tùy chọn có thể lựa chọn. Thuộc tính Message.Text sẽ hiển thị tùy chọn / hành động được người dùng chọn (Liệt kê 7-8 ).



**Managing State**

Managing state là việc xử lý và chuyển dữ liệu giữa các trang với tư cách là người dùng navigation thông qua các ứng dụng. Có bốn cách tiếp cận chính: truyền trực tiếp giá trị dữ liệu vào hàm tạo của trang, sử dụng từ điển Thuộc tính tĩnh trên Ứng dụng đối tượng để duy trì các cặp khóa / giá trị vào đĩa, một thể hiện dữ liệu tĩnh ( toàn cầu ) có sẵn cho tất cả trang và thuộc tính tĩnh trên đối tượng Ứng dụng. Cả toàn cầu và Ứng dụng kỹ thuật đối tượng sử dụng mẫu singleton và hữu ích cho các lớp toàn ứng dụng như truy cập dữ liệu hoặc đối tượng kinh doanh.

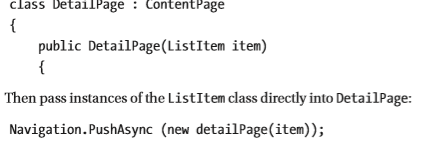
Truyền dữ liệu trực tiếp vào các trang bất cứ khi nào có thể để thu hẹp phạm vi của các biến và Managing bộ nhớ một cách thận trọng. Từ điển thuộc tính vẫn tồn tại khi ứng dụng của bạn nền và thậm chí sau khi ứng dụng của bạn đã khởi động lại!

Hãy bắt đầu với kỹ thuật đơn giản nhất, truyền dữ liệu trực tiếp vào một trang.

**Passing Data into Page Parameters**

Trạng thái thường được quản lý trong Xamarin.Forms bằng cách truyền dữ liệu trực tiếp vào Trang bằng cách sử dụng nhà xây dựng của nó. Cách tiếp cận này phạm vi các đối tượng dữ liệu đến một trang duy nhất, lý tưởng từ một kiến trúc và bộ nhớ sử dụng quan điểm.

Khi gọi một trang mới bằng phương thức Navigation, hãy chuyển vào bất kỳ biến nào đã được xác định trong hàm tạo của trang của bạn. Xác định trang chi tiết với lớp ListItem là tham số hàm tạo:



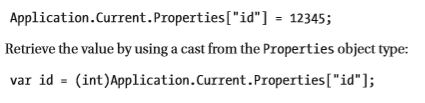
Thêm tất cả các tham số trong trình tạo trang của bạn cần để truyền dữ liệu từ dữ liệu khác trang. Thông tin chi tiết về ví dụ này có thể được tìm thấy trong Liệt kê 7-8 và7-10 .

Các yếu tố dữ liệu đôi khi được sử dụng trên nhiều trang trên toàn bộ ứng dụng và vượt qua chúng có thể trở nên cồng kềnh. Các yếu tố dữ liệu thường được sử dụng có thể được đặt vào một lớp toàn cầu tĩnh để chúng có sẵn trên toàn ứng dụng.

**Disk Persistence Using the Properties Dictionary**

Tính năng trạng thái bền bỉ nhất được tích hợp trong Xamarin.Forms là từ điển Thuộc tính. Các cặp tên / giá trị được lưu trữ dưới dạng đối tượng vào đĩa và được truy xuất theo yêu cầu từ bất cứ đâu trong ứng dụng, ngay cả sau khi ứng dụng đã khởi động lại. Thuộc tính hoạt động giống như cookie cho ứng dụng của bạn.

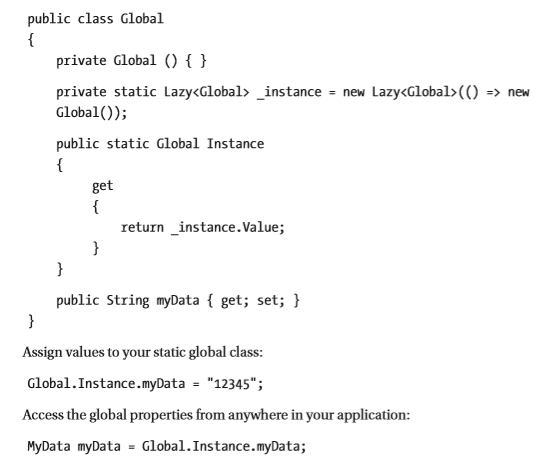
Lưu một giá trị vào từ điển Thuộc tính bằng cách sử dụng một giá trị khóa, chẳng hạn như id:



**Using a Static Global Class**

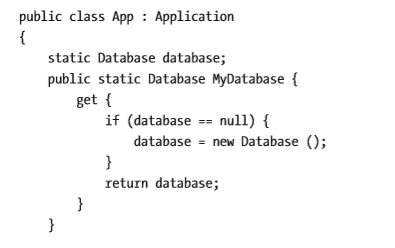
Một lớp toàn cầu tĩnh, một triển khai C # của mẫu đơn, có thể được sử dụng để lưu trữ dữ liệu trên toàn bộ ứng dụng.

Tạo một lớp tĩnh được gọi là Toàn cầu và đặt các thuộc tính trong đó mà bạn muốn sử dụng trên ứng dụng của bạn, như myData, như được liệt kê trong Liệt kê 7-9.



**Using a Static Property on the Application Object**

Một singleton có thể được tạo bằng cách sử dụng một thuộc tính tĩnh trên đối tượng Ứng dụng:



**Drill-Down Lists**

Một drill-down list là một danh sách các item tappable được lựa chọn để navigation đến một trang mới. Có nhiều cách để xây dựng chúng bằng Xamarin.Forms và các công thức sau đây bao gồm ba loại danh sách chi tiết phổ biến nhất: theo item, theo trang và được nhóm. Cái khoandanh sách xuống theo item có các hàng có thể được chọn để hiển thị thêm thông tin về mỗi item item: mô hình chi tiết tổng thể truyền thống. Danh sách chi tiết theo trang là một menu các trang có thể được chọn để navigation đến các ContentPages khác nhau. Cả hai công thức này đều sử dụng một ListView để liên kết với một mô hình dữ liệu để cung cấp một danh sách động các item có thể khai thác. Một nhóm danh sách chi tiết được xây dựng bằng TableView rất hữu ích để tạo các item menu tĩnh được phân loại.

ListView là một trong những công cụ linh hoạt nhất để tạo danh sách chi tiết. Danh sách ngắn tất nhiên, có thể được xây dựng bằng tay bằng cách sử dụng bất kỳ bố cục nào chứa đầy các nút hoặc nhãn kết hợp với nhận dạng cử chỉ để xử lý vòi. Danh sách dài hơn cho vay để liên kết dữ liệu bằng ListView.

Phân nhóm giống như trong Chương 6 sử dụng nhóm ListView. Cả hai mặt hàng và các trang có thể được nhóm lại bằng cách sử dụng IsGroupingEnables và GroupDisplayBinding thuộc tính của ListView.

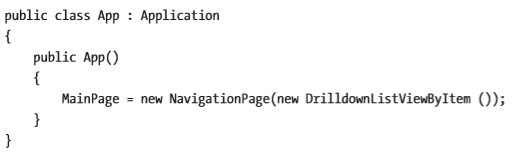
Danh sách các trang yêu cầu nhóm cũng có thể được xây dựng bằng cách sử dụng TableView. Điều này thay thế thủ công cho ListView sử dụng Lệnh TextCell và CommandParameter thuộc tính thay vì ràng buộc dữ liệu.

Chúng ta sẽ bắt đầu với các menu ListView ràng buộc dữ liệu.

**Using ListView by Item**

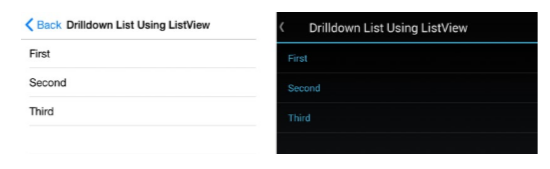
Nhiều danh sách chứa một loạt các item mà người dùng muốn truy sâu vào để tiếp cận chi tiết về từng item. Sử dụng ListView để hiển thị danh sách các item được liên kết dữ liệu với mô hình dữ liệu và sau đó hiển thị trang chi tiết bằng cách sử dụng PushAsync, tất cả được gói trong NavigationPage để người dùng có thể quay lại danh sách.

Bạn có thể tạo ListView của mình bằng cách sử dụng bất kỳ phương pháp tiếp cận nào được thảo luận trong Chương 6 . Việc triển khai này sử dụng lớp item danh sách của chúng tôi được gọi là DrilldownListViewByItem (xem phần Liệt kê đầy đủ 7-11 và7-12 ). Khởi tạo trang đó trong hàm tạo của lớp Ứng dụng được gói trong NavigationPage (xem danh sách đầy đủ 7-13).

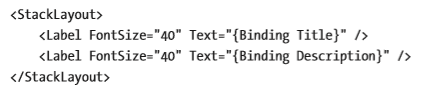


Lưu ý rằng ví dụ này phải khác một chút so với App.cs được tìm thấy trong mã có thể tải xuống, vì vậy hãy sử dụng những gì bạn thấy được viết ở đây trong ví dụ này.

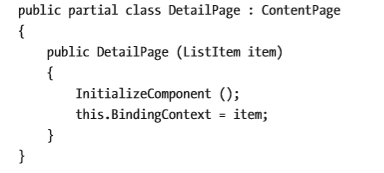
Điều này tạo ra danh sách hiển thị trong Hình 7-12, với một thanh navigation trên iOS và Android.



Đối với trang chi tiết đi sâu, hãy tạo một Chi tiết XAML hiển thị Tiêu đề và Mô tả.



Trong mã Chi tiết phía sau, tạo một hàm tạo lấy ListItem làm tham số (xem toàn bộ Liệt kê 7-15).



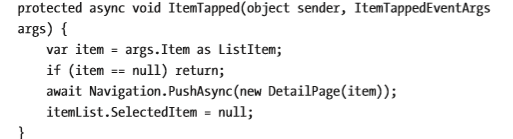
BindingContext được đặt thành tham số ListItem đến từ người dùng chạm vào ListView.

Quay lại trang danh sách, khi một hàng item được gõ, sự kiện ItemTapped sẽ kích hoạt. Trong này trong trường hợp, tên sự kiện xảy ra được đặt tên giống như thuộc tính ItemTapped: ItemTaps. Đây là tuyên bố ListView:

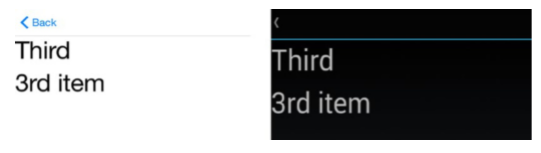


Phần còn lại của mã ListView nằm trong Liệt kê 7-11 , DrilldownListViewByItem.xaml.

Phương thức ItemTapped nhận ListItem và đẩy nó ra một hệ thống phân cấp mới Chi tiếtPage bằng phương pháp PushAsync của NavigationPage.

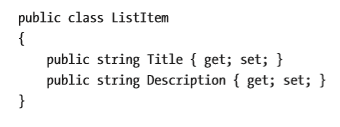


Trong cuộc gọi Navigation.PushAsync, ListItem của hàng đã khai thác được truyền vào Chi tiết sau đó được liên kết với BindingContext của trang, cho phép hiển thị Tiêu đề và Mô tả của ListItem, như trong Hình 7-13 .

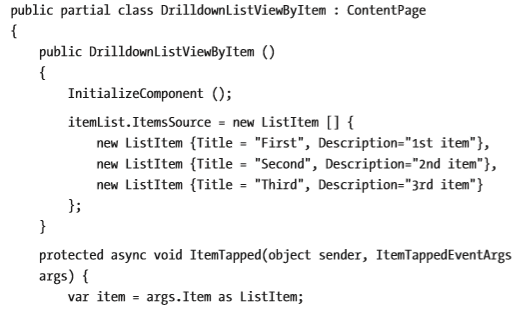
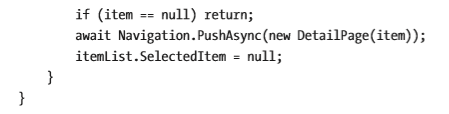


**CODE COMPLETE: Drill-Down List**

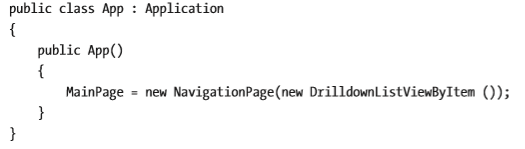
Đó là một bản tóm tắt nhanh chóng của một danh sách chi tiết theo item. Danh sách7-10 đến7-15 chương trình mã hoàn chỉnh của mẫu danh sách truy sâu xuống bằng NavigationPage.



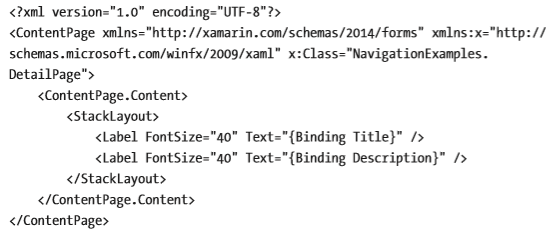
Listing 7-12. DrilldownListViewByItem.xaml.cs

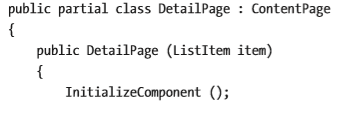
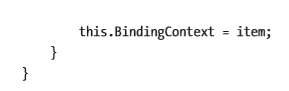
Listing 7-13. App Class for This Example (Differs from Code Download App.cs)



Listing 7-14. DetailPage.xaml.cs



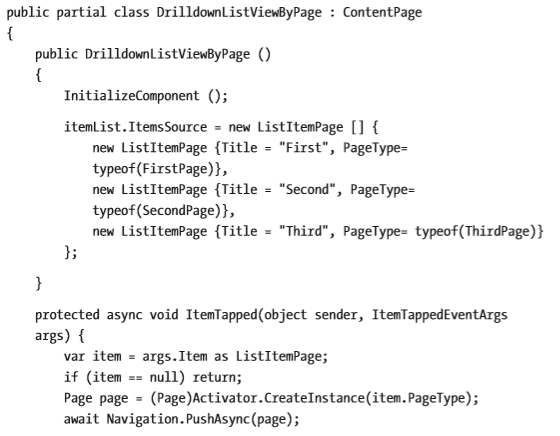
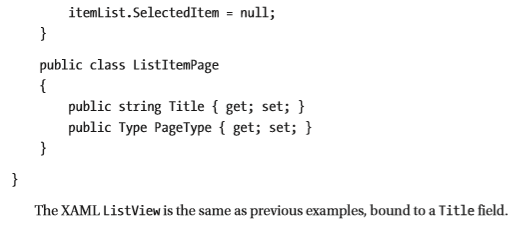
Listing 7-15. DetailPage.xaml.cs

**Using ListView by Page**

Navigation một danh sách các trang thật dễ dàng với ListView. Xây dựng một menu chứa một danh sách các trang riêng biệt (ví dụ: Thứ nhất, Thứ hai, Thứ ba). Liên kết dữ liệu ListView của bạn với một mô hình dữ liệu chứa các tiêu đề trang và các loại ContentPage. Xem chi tiết từng trang bằng cách sử dụng NavigationPage để cung cấp cho người dùng một cách để quay lại danh sách.

Kết quả của Liệt kê 7-16 trông giống như trên màn hình như Hình 7-9 nhưng navigation đến các loại ContentPage khác nhau (FirstPage, SecondPage, ThirdPage) thay vì chỉ một (Chi tiết trang).

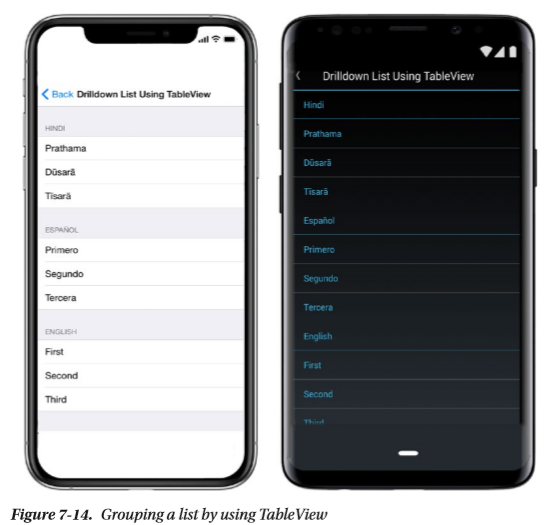
 



**Using TableView for Grouping Pages**

Hoàn hảo cho danh sách đa item của các item navigation, biến thể của danh sách truy sâu xuống mẫu hiển thị danh sách tĩnh bằng cách sử dụng chế độ xem được gọi là TableView. Khi một item được gõ, một màn hình chi tiết được hiển thị. Nó cũng sử dụng mô hình phân cấp, cung cấp một tùy chọn để sử dụng các nút Quay lại. Mẫu danh sách phân cấp / truy vấn ngược này được sử dụng trong nhiều các dự án mã có thể tải xuống trong suốt cuốn sách như trang chủ giải pháp, cho phép lựa chọn các ví dụ mã trong mỗi chương.

Công thức này có vẻ tốt nhất khi có nhiều loại mặt hàng để lựa chọn, bởi vì ít nhất một TableSection là bắt buộc, ngay cả khi không cần nhiều danh item, như thể hiện trong hình 7-14 .



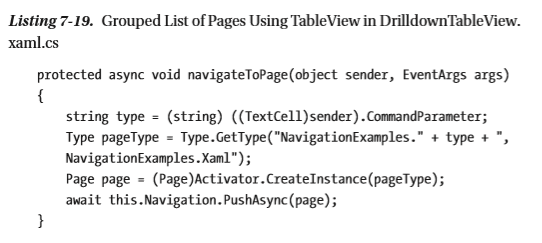
Nếu bạn không muốn sử dụng các danh item, bạn có thể để trống tiêu đề TableSection, nhưng để lại một khoảng trống khá lớn ở đầu danh sách. Nếu bạn không cần danh item, hãy xem xét sử dụng ListView theo trang, như được mô tả trong phần trước.

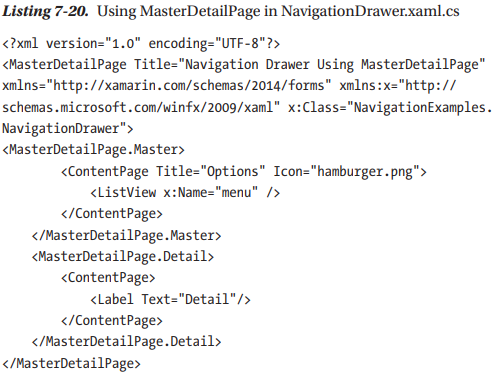
TableView về mặt kỹ thuật không phải là một bố cục nhưng hoạt động giống như một. Quan điểm này được tạo thành từ TextCells được sắp xếp trong các phần, chẳng hạn như tiếng Hindi, Español và tiếng Anh. Mỗi bảng biểu thị một danh item khác nhau, như được liệt kê trong Liệt kê 7-18 . Đặt trang nào cho nên navigation đến khi khai thác bằng cách đặt các loại ContentPage trong CommandParameter của mỗi TextCell. Đặt thuộc tính Taps cho sự kiện xử lý vòi của TextCell.





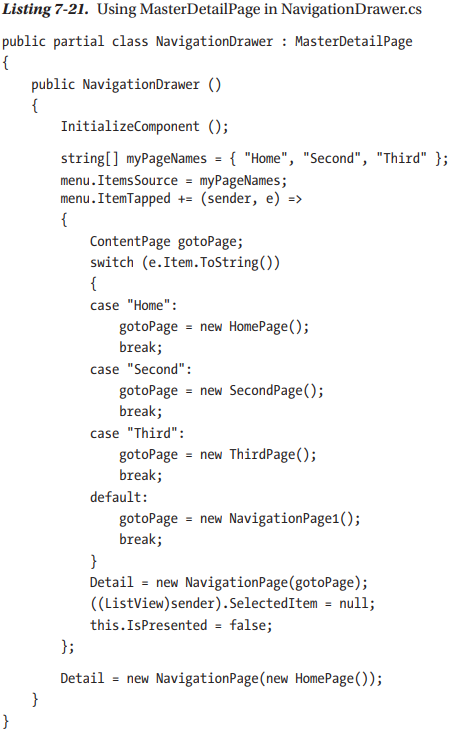
Khi khai thác TextCell, phương thức navigationToPage được gọi trong mã C # phía sau. Ở đó, CommandParameter chứa loại ContentPage được phân tích cú pháp và loại được khởi tạo vào Trang bằng Activator.CreateInstance. Sử dụng PushAsync để navigation đến trang mới này, trong trường hợp này sẽ là FirstPage, SecondPage hoặc Thứ ba.





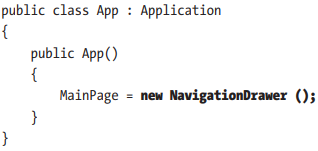
|  |
| --- |
| Tip: Thuộc tính Title của MasterDetailPage là bắt buộc. |

Khi một mục menu được chọn, sự kiện ItemTapped sẽ kích hoạt và đặt thuộc tính Detail cho trang đích. Detail được đặt thành HomePage ban đầu.

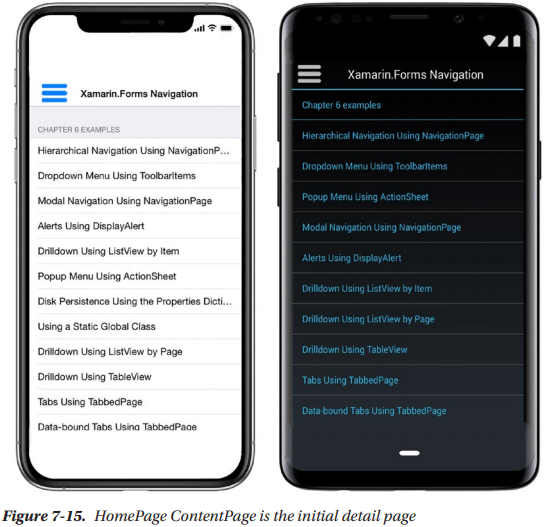


Một vài điều xảy ra sau khi trang Detail được điều hướng đến. SelectedItem được đặt thành null để xóa phần tô sáng trên hàng đã chọn và IsPresented được đặt thành false để xóa menu.

Vì navigation drawer đã khởi tạo các trang điều hướng của riêng nó, nên bạn không cần phải tạo một NavigationPage khác khi bạn gọi nó. Đây là cách triển khai nav drawer của bạn trong App.cs, tại thư mục gốc của điều hướng của bạn.

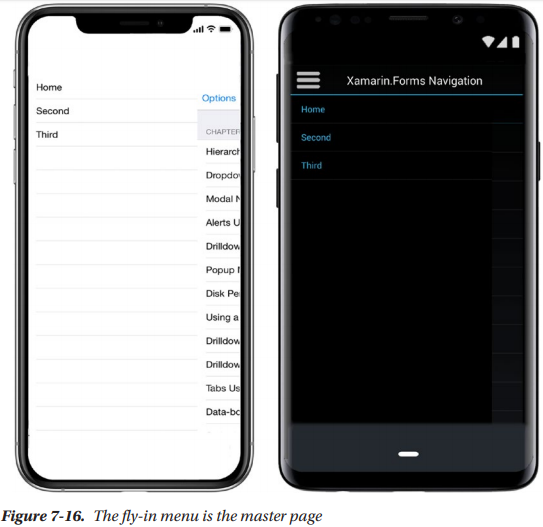


Ví dụ này bắt đầu khi trang chủ được đặt làm trang chi tiết, như trong Hình 7-15. Nó mã hóa các trang ngăn kéo điều hướng thay vì làm cho chúng động.



|  |
| --- |
| Lưu ý quan trọng HomePage.cs chứa HomePage ContentPage được hiển thị trong Hình 7-12 và nằm trong code có thể tải xuống nhưng không được liệt kê ở đây. Hãy chắc chắn tải xuống code cho chương này và kiểm tra nó, bởi vì nó chứa một số ví dụ sẵn sàng nhất cho dự án trong cuốn sách này. |

Khi nhấp vào biểu tượng, trang chính được hiển thị, chứa menu và biểu tượng menu ở góc trên bên trái, như trong Hình 7-16.



Thay đổi biểu tượng menu thành hamburger bằng cách sử dụng thuộc tính Trang chủ Icon. Tệp biểu tượng được lấy từ thư mục hình ảnh cục bộ cho mỗi nền tảng.

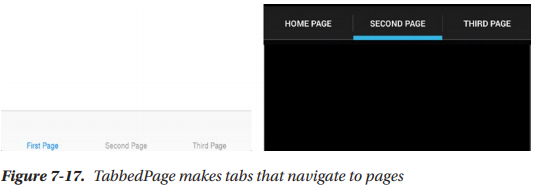


Nhấp vào một mục menu sẽ đưa bạn đến trang chi tiết mới được chỉ định.

|  |
| --- |
| Lưu ý Trải nghiệm máy tính bảng của MasterDetailPage hoàn toàn khác với trải nghiệm điện thoại và mang hơi thở của không khí trong lành. Danh sách chính hiển thị ở bên trái và trang chi tiết hiển thị ở phần bên phải của màn hình. Điều này cung cấp hiển thị thời gian thực của chi tiết danh sách khi người dùng chạm vào các mục khác nhau trong danh sách. Có menu và các chi tiết trên màn hình cùng một lúc sẽ tận dụng tối đa lợi thế của máy tính bảng. |

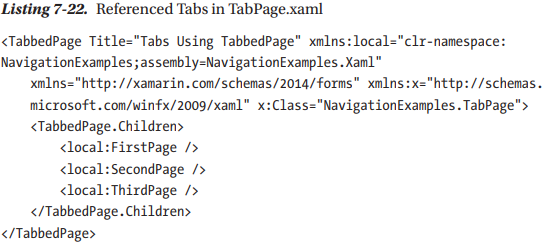
**Tabs Using TabbedPage**

Có các tab giống như thư mục có thể nhấp ở trên cùng hoặc dưới cùng của màn hình là một mẫu điều hướng phổ biến, được thực hiện bởi TabbedPage, như trong Hình 7-17. Các tab có thể được khai báo trong XAML dưới dạng tham chiếu hoặc nội tuyến. Hãy bắt đầu với các tài liệu tham khảo.



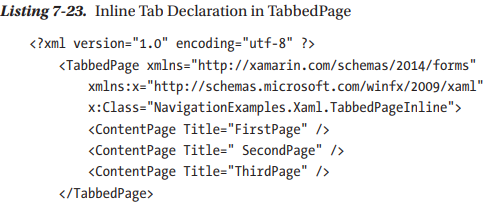
|  |
| --- |
| Mẹo các tab iOS nằm ở cuối màn hình và các tab Android ở trên cùng. |

Tạo một TabbedPage, tạo XAML và mã phía sau. Các trang tab tham chiếu trong thuộc tính TabbedPage.Children, như trong Ví dụ 7-22.



Thuộc tính Title của mỗi trang con là nơi tiêu đề tab xuất phát. Hãy nhớ chỉ định nó bất cứ nơi nào bạn tạo trang con. Tất cả các hành động xảy ra trong XAML, không cần mã hóa trong mã phía sau ngoài hàm tạo mặc định. Biên dịch và chạy ví dụ này để xem kết quả trong Hình 7-14.

Các tab tham chiếu nội tuyến bằng cách khai báo chúng trực tiếp trong thẻ TabbedPage XAML như trong Ví dụ 7-23.

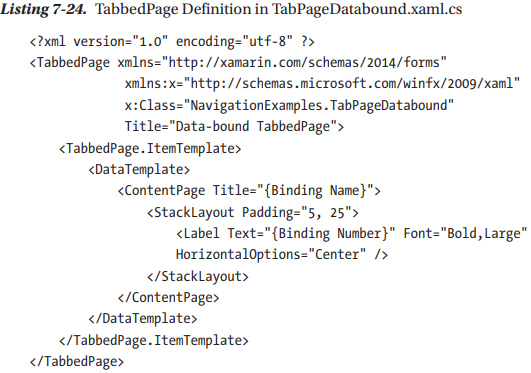


|  |
| --- |
| Mẹo Trong iOS, bạn có thể đặt biểu tượng trên các tab bằng cách sử dụng thuộc tính Icon của trang con. Không như vậy trên Android. |

**Creating Data-Bound Tabs**

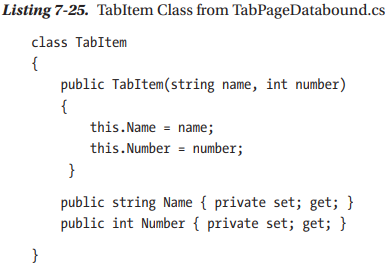
TabbedPage có thể được liên kết với một nguồn dữ liệu. Sử dụng các thuộc tính TabbedPage ItemSource và ItemTemplate để đạt được một menu được gắn thẻ dữ liệu.

Tạo một Tabbed Page, cái mà tạo XAML và mã phía sau. Trong XAML, tạo trang DataTemplate bên trong thuộc tính TabbedPage.ItemTemplate. Đây là trang bạn thấy khi một tab được chọn, hiển thị các thuộc tính Bind Name và Number, như trong Ví dụ 7-24.

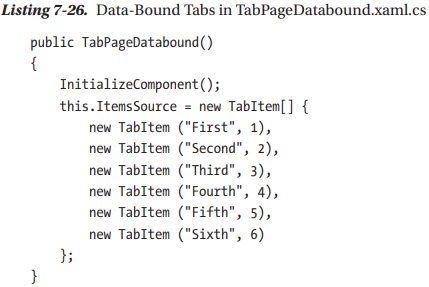


|  |
| --- |
| Mẹo DataTemplate thường được sử dụng cho các lớp liên kết dữ liệu như ListView và TableView. Đọc thêm về nó trong Chương 6. |

Tạo một danh sách các tab bằng cách khai báo một lớp TabItem và xây dựng một mảng của chúng. Tên và số là các thuộc tính của TabItem, như trong Ví dụ 7-25.



Ràng buộc trong Ví dụ 7-24 liên kết thuộc tính TabItem.Name với Title của trang và thuộc tính TabItem.Number với thuộc tính Text của label. Xây dựng mảng TabItem và gán cho thuộc tính ItemSource của TabbedPage (Ví dụ 7-26).



ItemSource hiển thị các thuộc tính của ràng buộc mảng TabItem trong ItemTemplate, TabItem.Name là ContentPage.Title và TabItem.Number là Label.Text.

|  |
| --- |
| Mẹo Ghi nhớ Khởi InitializeComponent(); trong hàm tạo TabbedPage của bạn. Lỗi mật mã có thể xảy ra khác. |

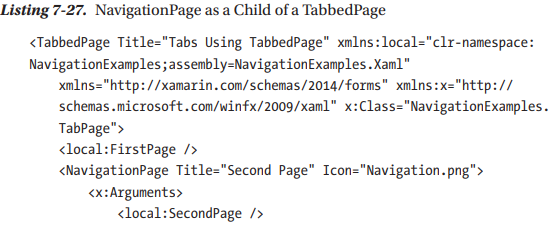


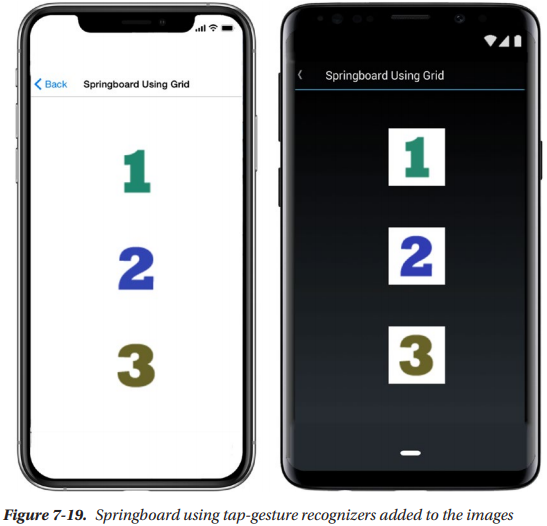
Cuộn thanh tab theo chiều dọc trên Android hoặc nhấn vào “More” trên iOS để xem tab thứ sáu.

Khi một tab được nhấn, NumberPage tương ứng được tạo và điều hướng đến, hiển thị Số bị ràng buộc.

**Putting NavigationPages Inside a TabbedPage**

Các trang điều hướng được sử dụng trong một trang được gắn thẻ bằng cách gán chúng là con, tạo một thanh điều hướng khi tab được chọn (Liệt kê 7-27). Hãy nhớ gán Tiêu đề cho NavigationPage để chỉ định tên của tab.







Ví dụ này sẽ hiển thị hai tab, Trang đầu tiên và Trang thứ hai. Trang thứ hai được gói trong NavigationPage.

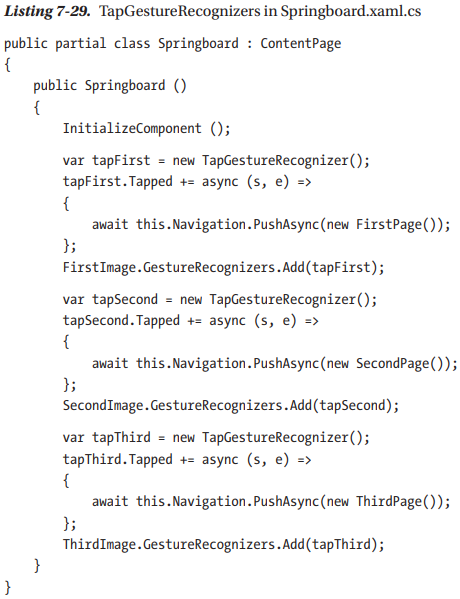
**Springboard**

Một springboard là một lưới các hình ảnh có thể ghép được trên menu màn hình chính, đôi khi được gọi là bảng điều khiển, như trong Hình 7-19.

Springboard này được triển khai với một Grid layout, như trong Ví dụ 7-28. Code này đặt ba hình ảnh trên Grid.

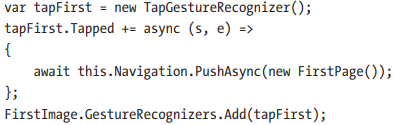


Mỗi trong số ba Hình ảnh được cung cấp với các nhận dạng cử chỉ nhấn, như trong Ví dụ 7-29. Sự kiện Taps của mỗi trình xử lý cử chỉ chứa PushAsync đến trang được yêu cầu. Tôi sẽ nói nhiều hơn về TapGestureRecognologists ngay.

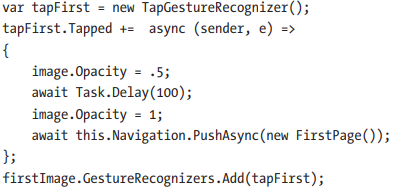


**Making Icons Tappable by Using Gesture Recognizers**

Các trình nhận dạng cử chỉ được thêm vào mỗi hình ảnh trong Ví dụ 7-28 xử lý các taps trong sự kiện TapGestureRecognizer Taps, sử dụng PushAsync để đẩy trang được chỉ định vào ngăn xếp điều hướng. Ở đây, một trình nhận dạng cử chỉ nhấn được thêm vào FirstImage:



Hình ảnh có thể chạm được nên phản ứng trực quan khi chạm vào. Sử dụng thủ thuật độ mờ được trình bày lại trong Chương 1 (Liệt kê 1-6):

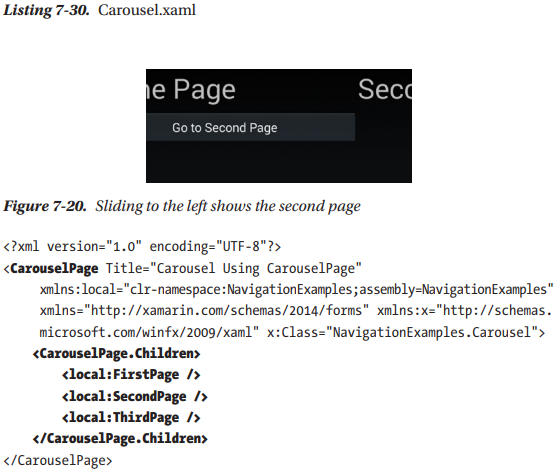


Điều này làm mờ hình ảnh một chút ngay lập tức khi chạm vào để cung cấp phản hồi của người dùng và cho họ biết rằng cử chỉ của họ đã làm gì đó. Khi sử dụng lớp Task, hãy nhớ thêm câu lệnh using:

using System.Threading.Tasks;

**Carousel Using CarouselPage**

Carousel pages cuộn ra khỏi màn hình để hiển thị một trang khác khi người dùng trượt sang trái hoặc phải. Tạo một carousel page và thêm các trang con, như trong Ví dụ 7-30.



Điều này cho phép cuộn ngang giữa các trang con. Hình 7-20 cho thấy trang chủ, khi trượt sang trái, sẽ hiển thị trang thứ hai.

Khi sử dụng CarouselPage làm trang chi tiết trong MasterDetailPage, hãy đặt MasterDetailPage.IsGestureEnables thành false để ngăn xung đột cử chỉ giữa CarouselPage và MasterDetailPage.

|  |
| --- |
| Mẹo Dự án cộng đồng CarouselView được coi là cập nhật hơn so với CarouselPage bởi một số người. https://github.com/alexrainman/CarouselView |

Bây giờ bạn đã được trang bị để xây dựng phác thảo điều hướng cho bất kỳ ứng dụng Xamarin.Forms nào bạn có thể tưởng tượng!

**Summary**

Điều hướng là một chủ đề quan trọng trong việc tạo ra các ứng dụng di động. Trong các ứng dụng web và máy tính để bàn, các màn hình đơn rất lớn và chiếm quá nhiều quy trình làm việc của người dùng nên việc điều hướng thường là một phần nhỏ trong trải nghiệm người dùng và thậm chí đôi khi được thêm vào như một suy nghĩ sau. Do tính kinh tế của màn hình bất động sản trong các ứng dụng di động, chúng tôi phải cho phép người dùng dễ dàng truy cập vào một ứng dụng trong thời gian ngắn nhất có thể. Ứng dụng khách hàng có thể thu hút người dùng trong một thời gian dài, nhưng thành công trong ứng dụng kinh doanh không được đo bằng lượng thời gian người dùng dành cho ứng dụng mà là câu trả lời cho câu hỏi này: Họ có tìm thấy thông tin họ đang tìm kiếm không? Menu có thể là một bãi đậu xe thả xuống ở đầu trang hoặc (cũng tệ như vậy) một ngăn kéo điều hướng bùng nổ với các tính năng vô tổ chức.

Sự quan trọng của điều hướng trong các ứng dụng di động đưa chúng ta đến nguyên lý này: Điều hướng di động phải phù hợp chặt chẽ với quy trình làm việc của người dùng. Nếu không, chúng ta có nguy cơ làm người dùng bối rối và thất vọng.

Trong hầu hết các ứng dụng của chúng tôi, đặc biệt là các ứng dụng kinh doanh, rất hữu ích để thử và khớp các câu chuyện của người dùng và các trường hợp sử dụng với các mẫu điều hướng chính này: hierarchical, modal, drill-down list, navigation drawer, và tabs.

Hy vọng, chương này cung cấp cho bạn các ý tưởng và mô hình để vạch ra bộ xương của bất kỳ ứng dụng nào bạn có thể tưởng tượng. Các mẫu mã có thể tải xuống có thể được trộn lẫn và khớp để giúp bạn phác thảo ứng dụng của mình.

Trong chương tiếp theo, chúng ta sẽ trở lại giao diện người dùng di động cho chủ đề cuối cùng và xúc tác trong phát triển Xamarin.Forms: tùy chỉnh bằng cách sử dụng hiệu ứng, native controls và custom renderers. Custom renderers cho phép bạn sử dụng hầu hết tất cả các chức năng UI nền tảng Xamarin, trong bối cảnh của ứng dụng Xamarin.Forms. Bạn có thể bao gồm mã dành riêng cho nền tảng, sử dụng Xamarin.iOS và Xamarin.Android, trong các trang Xamarin.Forms của bạn bằng cách sử dụng các hiệu ứng, native views và custom renderers.

Hãy đến chương tiếp theo!