Microsoft

Microsoft Virtual Academy

Mastering Xamarin Forms Development, Part 1

Pages, Layout, and Navigation

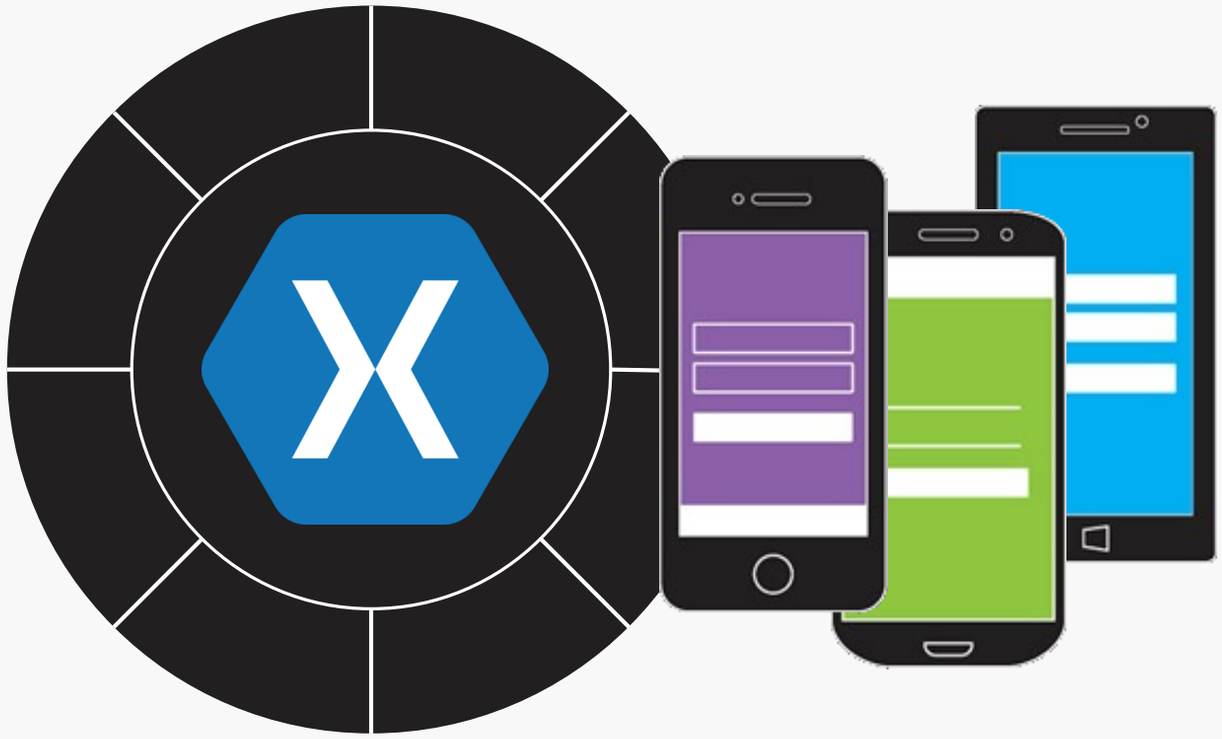
Scott J. Peterson

Produced by Wintellect NOW

Xamarin Forms

Layout and navigation

App structure, page layout, and navigation are the foundation of an app's user experience. Xamarin Forms includes advanced gesture and navigation support, layouts, buttons, labels, lists, and other common controls, easily connected via shared backend code, to create fully native iOS, Android, and Windows app experiences.



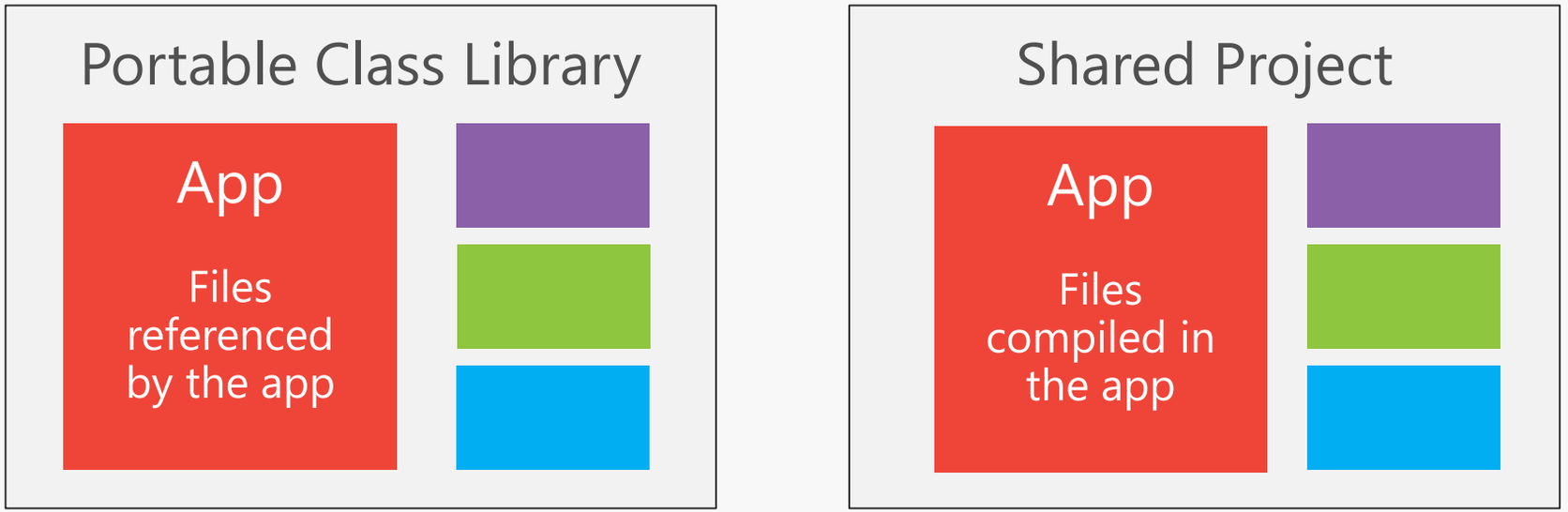
Agenda
In this session

- Code sharing concepts
- Basic UI concepts
- Page concepts
- Layout concepts
- Layout controls
- Demo: Using pages and layouts
- Advanced layouts
- Demo: Advanced layouts
- Navigation basics
- Demo: Basic app navigation
- Next steps

Code Sharing Concepts

Six and one-half dozen

The goal of a code-sharing strategy is to support an architecture where a single codebase can be utilized by multiple platforms. There are two methods for sharing code in Xamarin Forms:



Basic UI Concepts

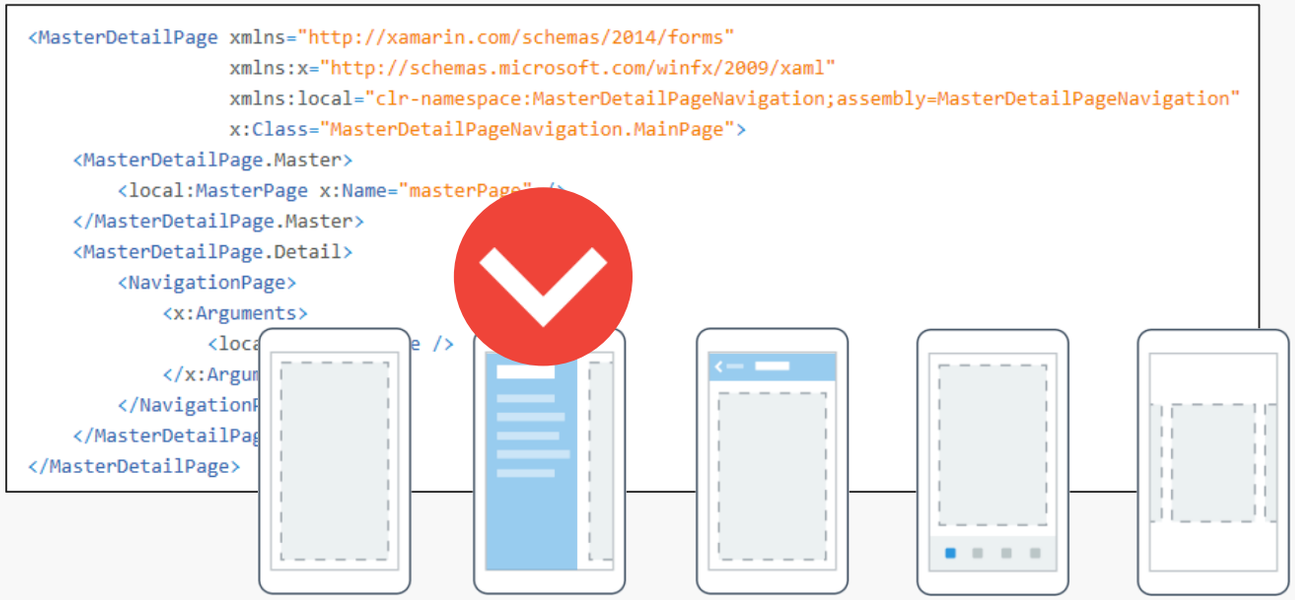
Cross platform magic

Xamarin Forms apps are not just interpreted code, but instead use native APIs, user interface controls, and native compliers, including platform-specific hardware acceleration.

Page Concepts

Can you be more specific?

In Xamarin Forms, a Page is a visual element that occupies most or all of the screen and contains a single child. A Page represents a **View Controller** in iOS, a **Page** in Windows, and acts a lot like an **Activity** on Android, but is not actually an Activity.



Page Concepts
What's available

- Content Page
- Navigation Page
- Master/Detail Page
- Tabbed Page
- Carousel Page
- Map Page
- List View Page

Layout Concepts

Get into position

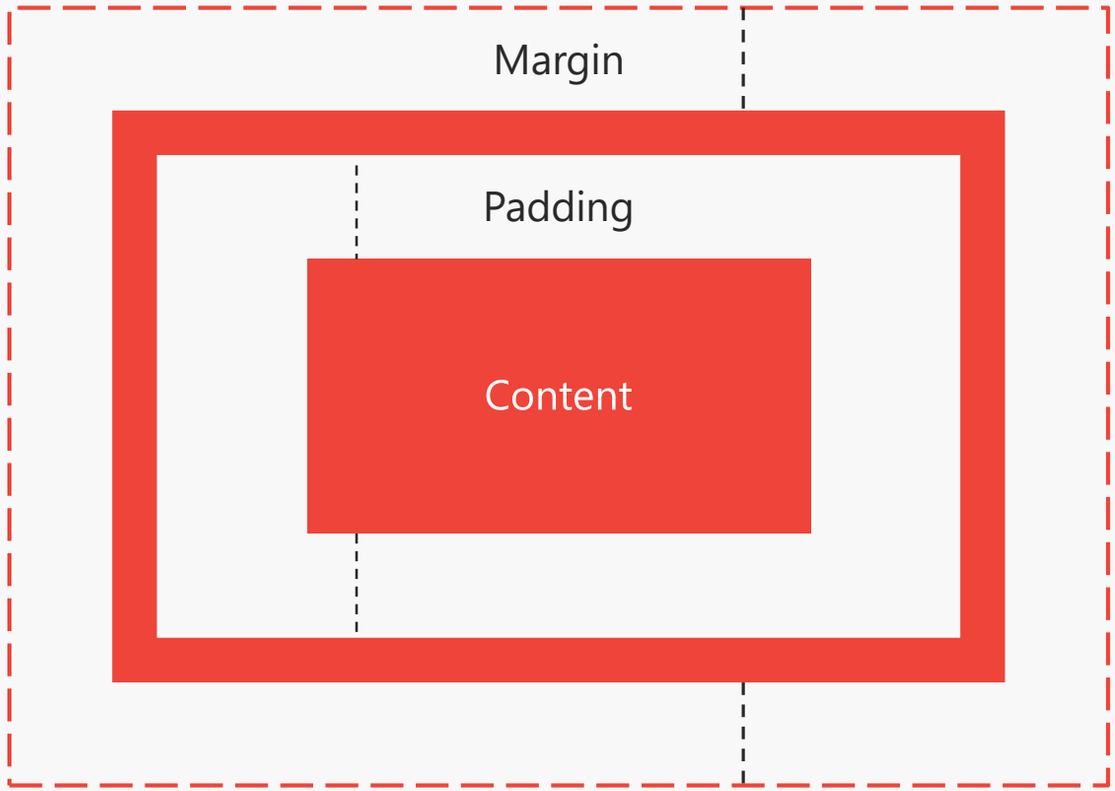
In Xamarin Forms, a Layout is a specialized subtype of a View, and ultimately acts as a container for other Layouts or Views. A Layout typically contains logic (either in XAML or code-behind) to set the position and size of child controls and elements in applications.

```
<AbsoluteLayout Padding="15">
  <Image AbsoluteLayout.LayoutFlags="PositionProportional" AbsoluteLayout.LayoutBounds="0.5, 0, 100, 100" Rota
    Source="bottom.png" />
    age AbsoluteLayout.LayoutFlags="PositionProportional" AbsoluteLayout.LayoutBounds="0.5, 0, 100, 100" Rota
    ource="middle.png" />
    mage AbsoluteLayout.LayoutFlags="PositionProportional" AbsoluteLayout.LayoutBounds="0.5, 0, 100, 100"
     Source="cover.png" />
</AbsoluteLayout>
```

Layout Controls

Fluid layout

The key to a responsive or "fluid" layout is the appropriate use of layout properties and panels to automatically and intelligently reposition, resize, and "reflow" content. With Xamarin Forms you can set a fixed size on an element, or use automatic sizing to allow a parent layout panel handle sizing and flow.



Layout Controls
Mix and match

- Stack Layout
- Absolute Layout
- Grid (Layout)
- Relative Layout
- Scroll View
- Custom Layouts

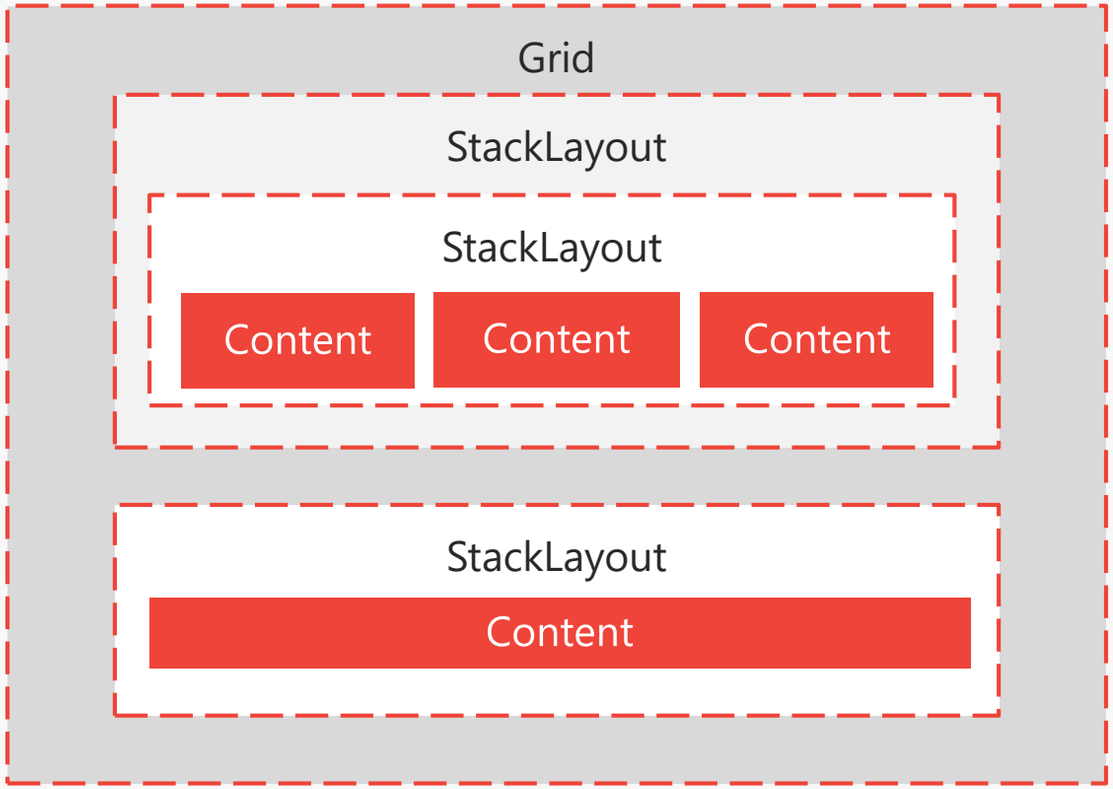
Demo

Using Pages and Layouts

Advanced Layouts

Keep it simple

In most apps layout scenarios, more than one layout choice can be used to create your intended design. It's common to have a number of reasonable layout choices, so consider which approach will be the easiest for your situation, and "nest" layouts as needed to create more complex designs.



Advanced Layouts
Respond gracefully

Although Xamarin Forms does not offer any native events for notifying your app of orientation changes in shared code, it's simple to design interfaces using the built-in layouts so that they transition gracefully when the device is rotated using a combination of the right layout with a little code.

```
private double width = 0;
private double height = 0;

protected override void OnSizeAlloca                    height)
{
    base.OnSizeAllocated(width, heig
    if (this.width != width || this.
    {
        this.width = width;
        this.height = height;
        //reconfigure layout
    }
}
```

Demo

Advanced Layouts

Navigation Basics

Getting around

Navigation in Xamarin Forms apps is based on a flexible model of navigation structures, navigation elements, and system-level features. Together, they enable a variety of intuitive user experiences for moving between apps, pages, and content.

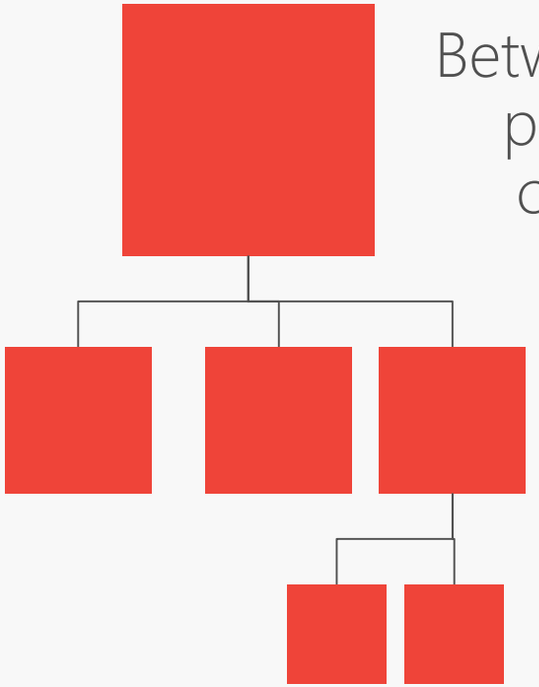
Navigation Basics

The right combination

As peers

Between pages in the same level of the same subtree.

In a hierarchy

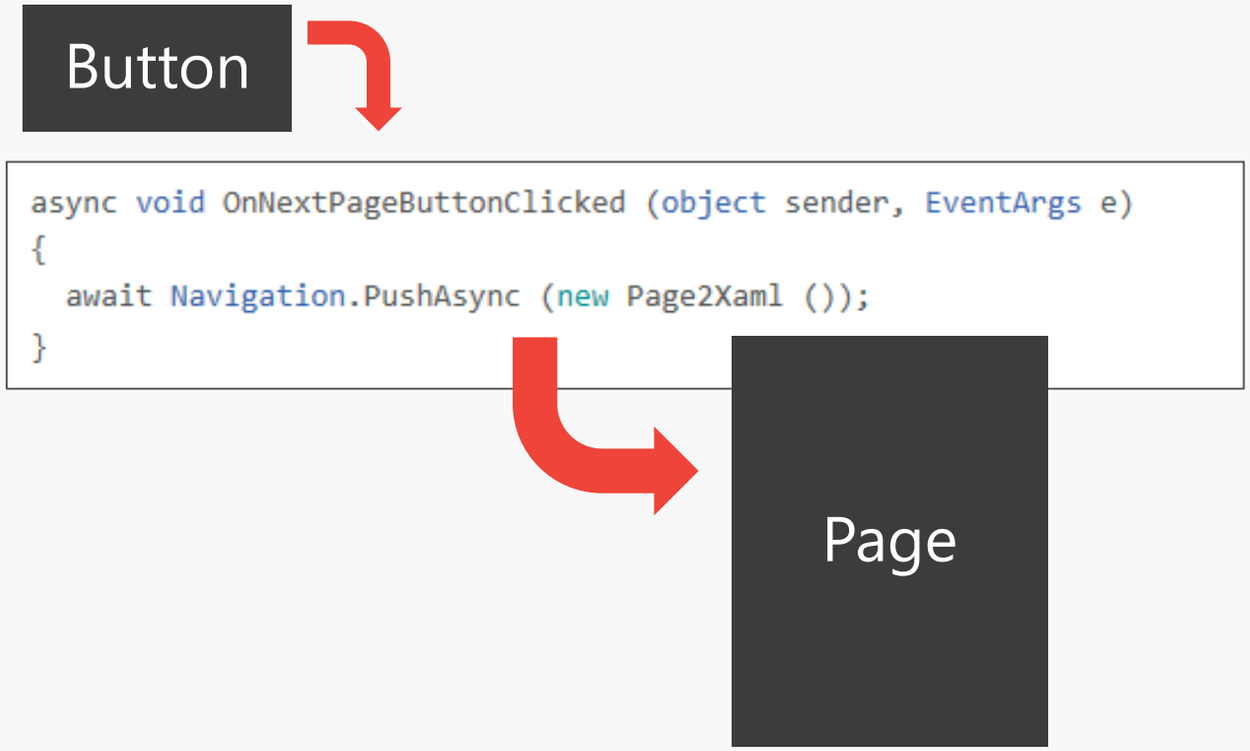


Between a parent page and its child pages

Navigation Targets

Push and pop

Navigating content typically occurs against a **Navigation**, object. By default the app has a “root” page created when the app is initialized (launched) to serve as the “grandparent” for initial navigation, which can be created as a “navigation” page as needed.



Demo

Basic App Navigation

Next Steps

Review device layouts

Select and evaluate layout and navigation changes across a larger number of devices, especially phones vs. tablets and desktops.

Check for accessibility

Change accessibility settings like font size and magnification to review page and layout rendering on various platforms.

Migrate to using commands

Move click event based navigation to a command navigation model where events can easily be centralized and reused.

Mastering Xamarin Forms Development
Pages, Layouts, and Navigation

Scott J. Peterson, MCSD, MCPSB, MCT
Chief Solution Architect
scott@liquiddaffodil.com

Wintellect NOW