Xamarin Evolve 2014

Xamarin.UITest

Chris van Wyk
chris.vanwyk@xamarin.com

Xamarin University

Agenda

Xamarin.UITest

- What is Xamarin.UITest?
- Creating a query
- Using the REPL

Best Practices

Xamarin University

Xamarin.UITest

Xamarin University

What is Xamarin.UITest?

- Xamarin.UITest is an automated UI Acceptance testing framework that checks the behavior and functionality of the UI.

Xamarin University

Running UITest

- Has Support for operations on the UI
- Can Query, Tap, Type, Gesture, Wait
- Uses NUnit as its core framework

Xamarin University

Working with code

Xamarin.UITest allows you to take control of what an app does

```
var app = Xamarin.UITest.ConfigureApp.iOS
  .AppBundle ("path/to/my.app").StartApp ();

app.Tap (c => c.Marked ("Add"));
app.EnterText(c => c.Class("UITextField").Index(0),
    "Get Milk");

app.Tap (c => c.Marked ("Save"));
```

Xamarin University

Following Unit Testing Principals

Step 1 - Arrange

```
var app = Xamarin.UITest.ConfigureApp.iOS
  .AppBundle ("path/to/my.app").StartApp ();

app.Tap (c => c.Marked ("Add"));
app.EnterText(c => c.Class("UITextField").Index(0),
    "Get Milk");

app.Tap (c => c.Marked ("Save"));
```


Following Unit Testing Principals

Step 2 - Act

```
var app = Xamarin.UITest.ConfigureApp.iOS
  .AppBundle ("path/to/my.app").StartApp ();

app.Tap (c => c.Marked ("Add"));
app.EnterText(c => c.Class("UITextField").Index(0),
    "Get Milk");

app.Tap (c => c.Marked ("Save"));
```

Following Unit Testing Principals

Step 3 - Assert

```
var app = Xamarin.UITest.ConfigureApp.iOS
  .AppBundle ("path/to/my.app").StartApp ();

app.Tap (c => c.Marked ("Add"));
app.EnterText(c => c.Class("UITextField").Index(0),
    "Get Milk");

app.Tap (c => c.Marked ("Save"));
```


Using with NUnit

Fits into the existing NUnit TestFixture layout

```
[TestFixture]
public class TestTasks
{
    [Test]
    public void TestAddingMilk()
    {
        IApp app = ConfigureApp.iOS.InstalledApp
        ("xamu.tasky").StartApp ();
        app.Tap (c => c.Marked ("Add"));
        app.EnterText(c => c.Class("UITextField").Index(0),
            "Get Milk");
        app.Tap (c => c.Marked ("Save"));
    }
}
```

Query Commands

- Query
- WaitFor
- WaitForElement
- WaitForNoElement
- Flash

Xamarin University

Control Commands

- Tap
- EnterText
- PressEnter
- SwipeLeft
- SwipeRight
- ScrollUp
- ScrollDown
- TouchAndHold

Xamarin University

Other Mechanisms

- Screenshot
- Device.SetLocation

Xamarin University

Query Objects

Xamarin University

Query Objects

UITest operations are performed using a query that returns UI elements

```
AppResult[] items = app.Query (
    c => c.Button ().Marked("Save")
);
```

Xamarin University

Common Queries

Querying and basic selectors

```
app.Query();

app.Tap (c => c.Id ("MyButton"));

app.Tap (c => c.Marked ("Click me"));

app.Flash (c => c.Button ());
```

Xamarin University

Common Queries

By class, non-visible controls and looking at Css elements

```
app.Query (c => c.Class("UILabel"));
app.Query (c => c.Button ("Click me"));
app.Query (c => c.All());
app.Query (c => c.Id ("MyWeb").Css("input"));
```

Xamarin University

The REPL Tool

- REPL: Read-Eval-Print-Loop
- Allows you to perform inspection and operations on the testing app
- Far more productive than using the normal write-code/test cycle

Xamarin University

Demonstration

Xamarin University

Best Practices

1. Cross Platform Tests
2. Running on Devices
3. Advanced Operations

Xamarin University

Cross Platform Tests

- Define an interface to abstract our higher functions for testing
- Provide implementations of the interface that are used to access the unique UI for each platform
- Platforms may mean iPhone/ iPad and Android

Xamarin University

Cross Platform Tests

Defining abstractions for screens & devices

```
public interface EnterTaskScreen
{
        EnterTaskScreen SetName(string name);
        EnterTaskScreen SetNotes(string notes);
        EnterTaskScreen MarkAsDone();
        EnterTaskScreen Cancel();
        EnterTaskScreen Save();
}

MainTaskScreen
    .SetName("Get Milk")
    .SetNotes("Pick up standard and low fat milk")
    .Save();
```

Xamarin University

Marked Selector

- Marked selector works differently between platforms
- On Android, the marked selector looks for the Id or the name of the control
- On iOS it also looks at the Accessibility Id and Accessibility Label

Xamarin University

Testing on Devices

- Android and iOS require different setups before executing the applications on devices

Xamarin University

Running on Android

- Requires Internet Permissions
- Support the full range of processors
- Publish using a keystore to avoid resigning

Xamarin University

Running on Android

- Use the “adb devices” to list the devices
- May need to use the Device ID if multiple devices are selected

```
$ adb devices
List of devices attached
10.71.34.101:5555  device
```

Xamarin University

Running on an Android Device

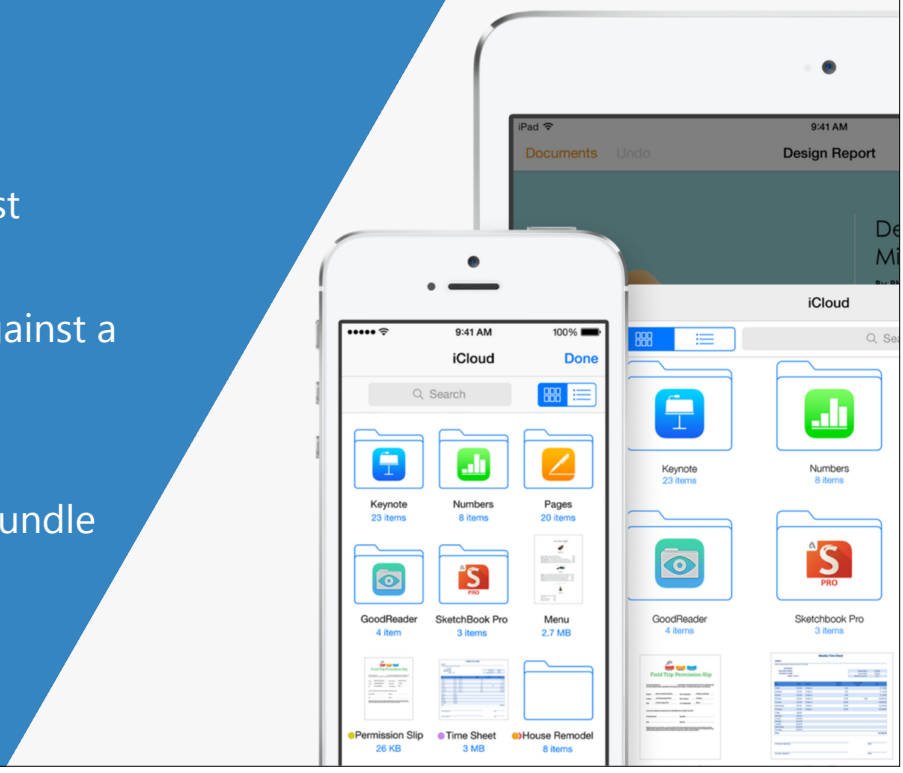
Connect to an device identifier

```
IApp _app = ConfigureApp.Android
    .ApkFile("/path/toapp.apk")
    .DeviceSerial("device").StartApp();
```

Xamarin University

Running on iOS

- Requires a embedded Test Cloud agent server
- Create an Adhoc build against a device
- Deploy to the device
- Use the IP Address and Bundle ID



Xamarin University

Running on an iOS Device

Connect to an IP address and installed app

```
IApp _app = ConfigureApp.iOS.DeviceIp("192.168.1.3")
    .InstalledApp("com.taskypro.iosapp").StartApp();
```

Xamarin University

Waiting

- Waiting for a fixed amount of time changes the way you would realistically wait between device differences
- Best to wait for elements to exist of not exist

Xamarin University

Hybrid Applications

- Many different applications have been developed as Hybrid apps where they are HTML pages embedded in a Native application, but the browser device capabilities differ significantly especially on Android

Xamarin University

Demonstration

Xamarin University

Summary

- Learning Xamarin.UITests
- Best Practices
- Running on Devices

Xamarin University