Why performance matters

- You want to provide a solid experience for your app
- You want your users to expect the app to function at the same performance levels as other apps
- Bad performance will equal bad reviews

Xamarin University

Agenda

1. Standard tips
2. Garbage Collections
3. async/await
4. Using Instruments

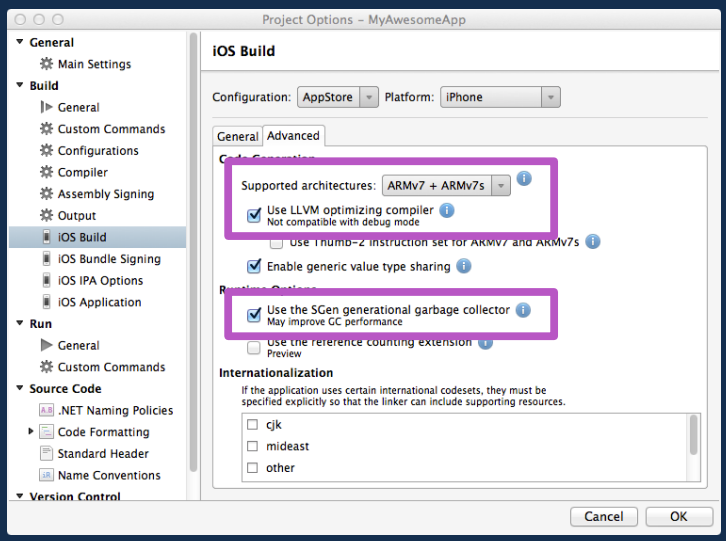
Xamarin University

Quick Fixes

Xamarin University

Use production build settings

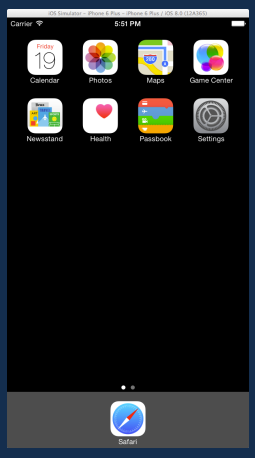
- Turn on the SGen garbage collector
- Use the LLVM compiler
- Build for all processors
- Turn off Debug options



Xamarin University

Run on real hardware

≠



Xamarin University

Xamarin University

Stress test with Maximum Datasets

- Use the larger datasets to find your apps performance weak spots

Xamarin University

Xamarin University

Collection controls should always reuse

- Controllers should always use re-use cells/items to limit memory
- Make drawing as efficient as possible
- Load as much into memory

Xamarin University

Xamarin University

Garbage Collection

Xamarin University

Garbage Collection (iOS)

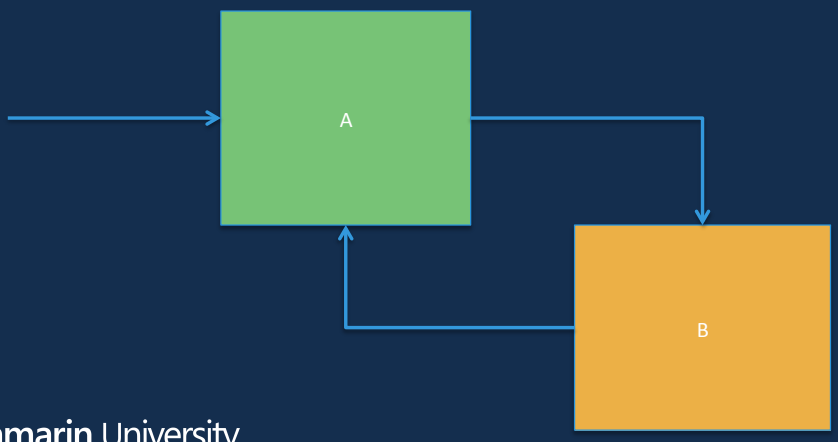
- Mobile devices are still fairly low capacity devices

iPhone 3GS
256MB RAM
833MHz CPU

iPhone 6
1GB RAM
1.4GHz ARMv8
Dual Core

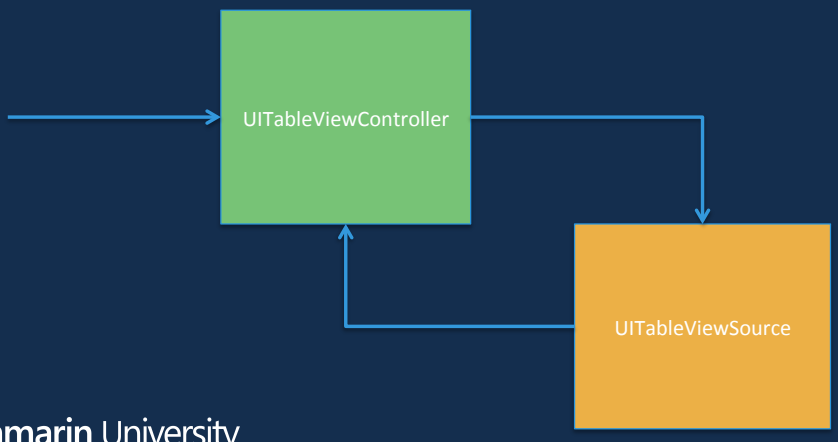
Xamarin University Xamarin University

GC Issue – Circular References



Xamarin University

GC Issue – Circular References



Xamarin University

GC Issue - Scoping

Scoping can catch you out when you least expect

```
public override void ViewDidLoad()
{
    base.ViewDidLoad ();

    var imageView = new UIImageView (
       new RectangleF (0, View.Bounds.Height / 2,
       View.Bounds.Width, View.Bounds.Height / 2));
    Add (imageView);

    AddButton (ref y, "Remove Image from Screen",
        (sender, e) => {
            imageView.RemoveFromSuperview ();
        });
}
```

Xamarin University

GC Issue - Scoping

Objects used with lamda's are moved to class scope

```
public override void ViewDidLoad()
{
    base.ViewDidLoad ();

    var imageView = new UIImageView (
       new RectangleF (0, View.Bounds.Height / 2,
       View.Bounds.Width, View.Bounds.Height / 2));
    Add (imageView);

    AddButton (ref y, "Remove Image from Screen",
        (sender, e) => {
            imageView.RemoveFromSuperview ();
        });
}
```

instances declared in methods used in closured will be moved to class scope...

Xamarin University

GC Issue - Scoping

Objects used with lamda's are moved to class scope

```
UIImageView imageView;

void ImageViewRemoveFromSuperview(object sender, EventArgs e)
{
    imageView.RemoveFromSuperview ();
}

public override void ViewDidLoad()
{
    base.ViewDidLoad ();
    imageView = new UIImageView (
        new RectangleF (0, View.Bounds.Height / 2,
        View.Bounds.Width, View.Bounds.Height / 2));
    Add (imageView);
    AddButton (ref y, "Remove Image from Screen",
      ImageViewRemoveFromSuperview);
}
```

...as if you were declaring them like this

Xamarin University

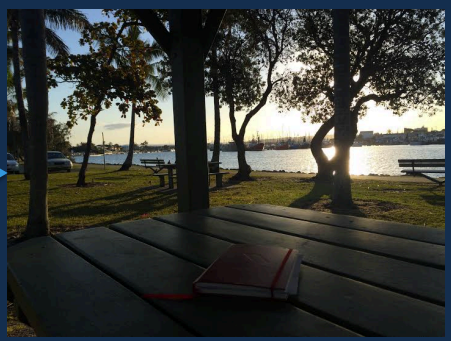
GC Issue – Native Resources

Native Resources should be released asap

```
var image =
   UIImage.FromBundle ("Lake.jpg");
```

Internal Memory: 80 bytes

Actual Image Memory Size: 890Kb



Xamarin University

Leave the UI Thread for the UI

- Using async/await lets you perform long running operations of the main thread so you apps can be more responsive

Xamarin University

Xamarin University

Make long running tasks asynchronous

Let other threads/tasks do the heavy lifting

```
public static List<Customer> GetCustomers()
{
    var client = new WebClient ();
    var data = client.DownloadString(restUrl);

    return JsonConvert.DeserializeObject<List<Customer>> (data);
}
```

Xamarin University

Make long running tasks asynchronous

Convert methods to their async equivilant

```
public static Task<List<Customer>> GetCustomersAsync()
{
    var client = new WebClient ();
    var data = await client.DownloadStringTaskAsync(restUrl);

    return JsonConvert.DeserializeObject<List<Customer>> (data);
}
```

Xamarin University

Make long running tasks asynchronous

Make sure processing is async too

```
public static Task<List<Customer>> GetCustomersAsync()
{
    var client = new WebClient ();
    var data = await client.DownloadStringTaskAsync(restUrl);

    return await Task.Run(() =>
        JsonConvert.DeserializeObject<List<Customer>>(data));
}
```

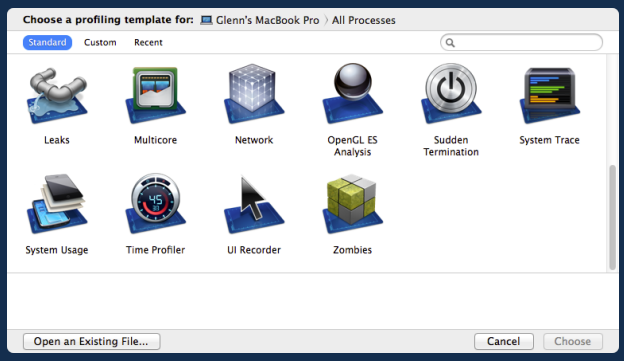
Xamarin University

Instruments

Xamarin University

Instruments

- Sometimes apps perform badly and you don't know where
- Instruments allows you to measure aspects of your application and find out what is happening and investigate using the information you gather

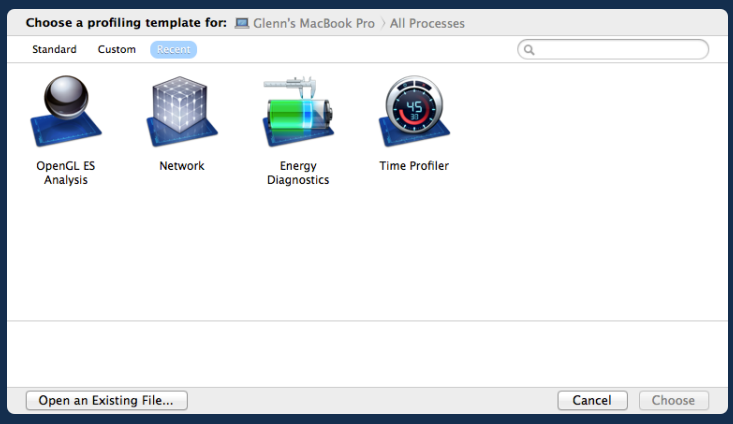


Xamarin University

Xamarin University

Types of Instruments

Some of the main ones to improve performance



Xamarin University

Demo

Xamarin University

Summary

- Optimize your build settings
- Enforce good garbage collection processing
- Use async where possible, especially for long running processes
- Use Instruments to diagnose and fix performance

Xamarin University