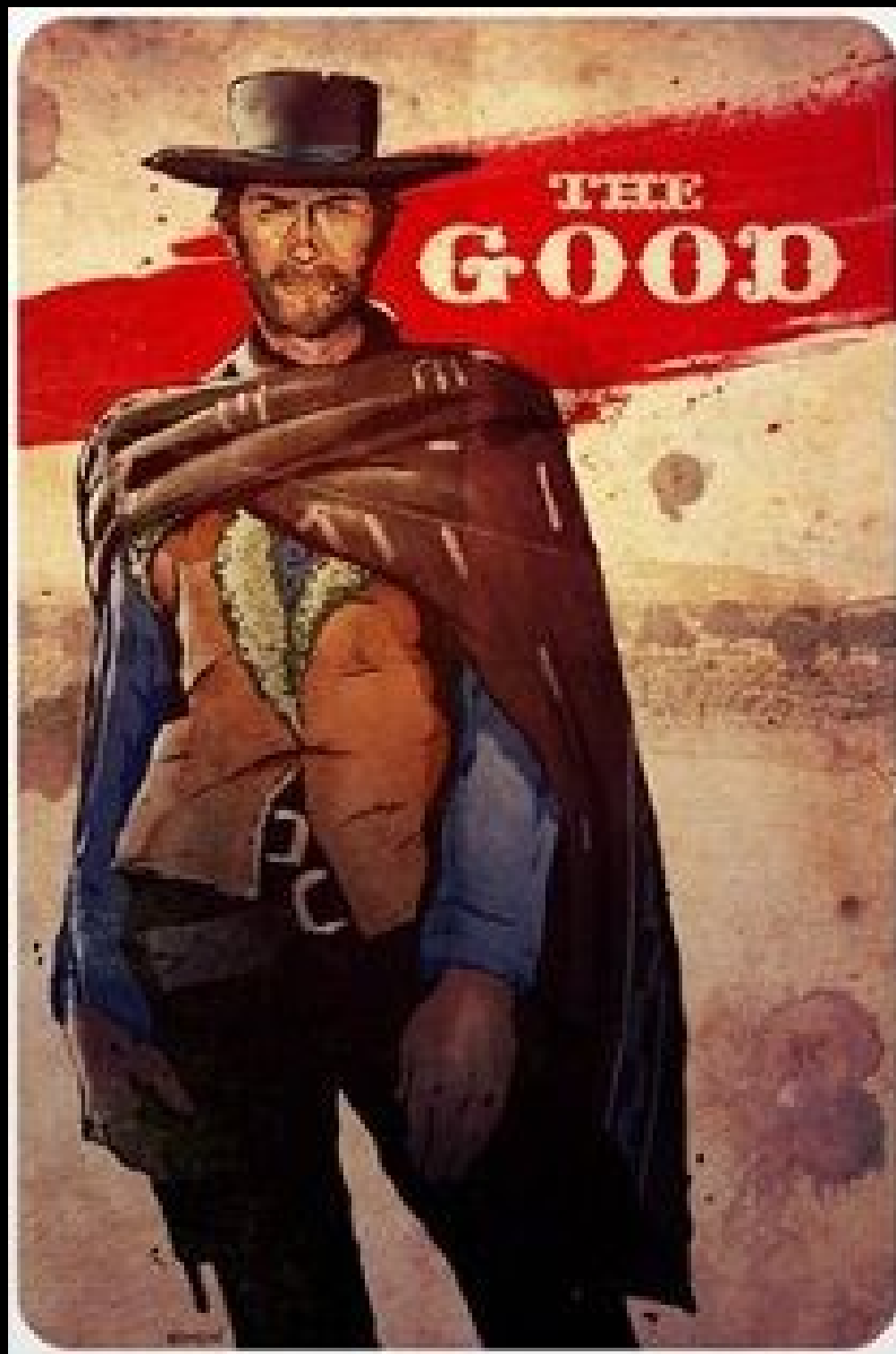


From Objective-C to Xamarin

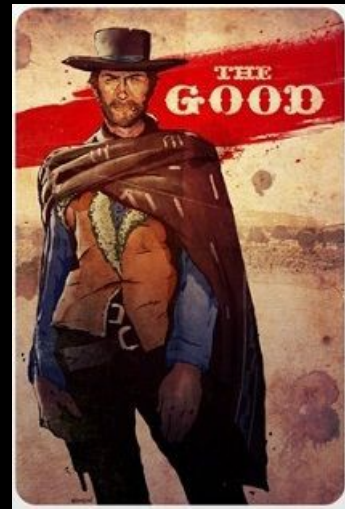
First impressions and common pitfalls

Alexander Dodatko
2014

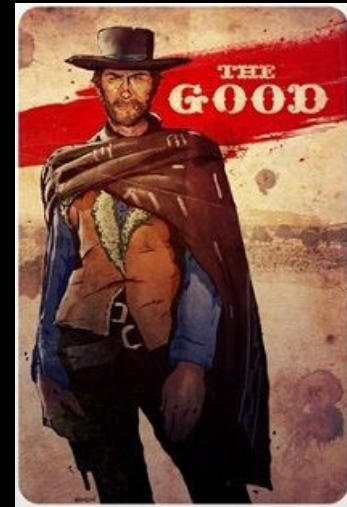
Xamarin == C# for Mobile



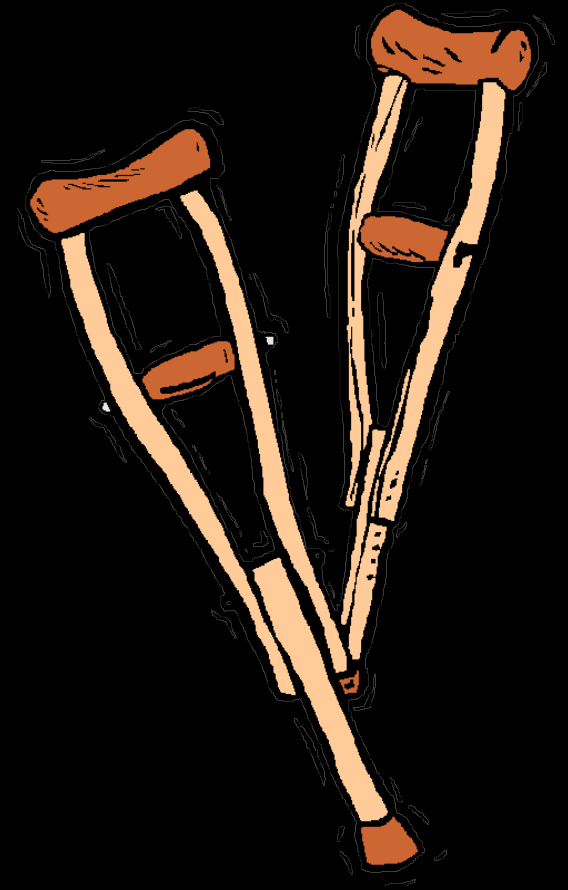
Xamarin's Promise

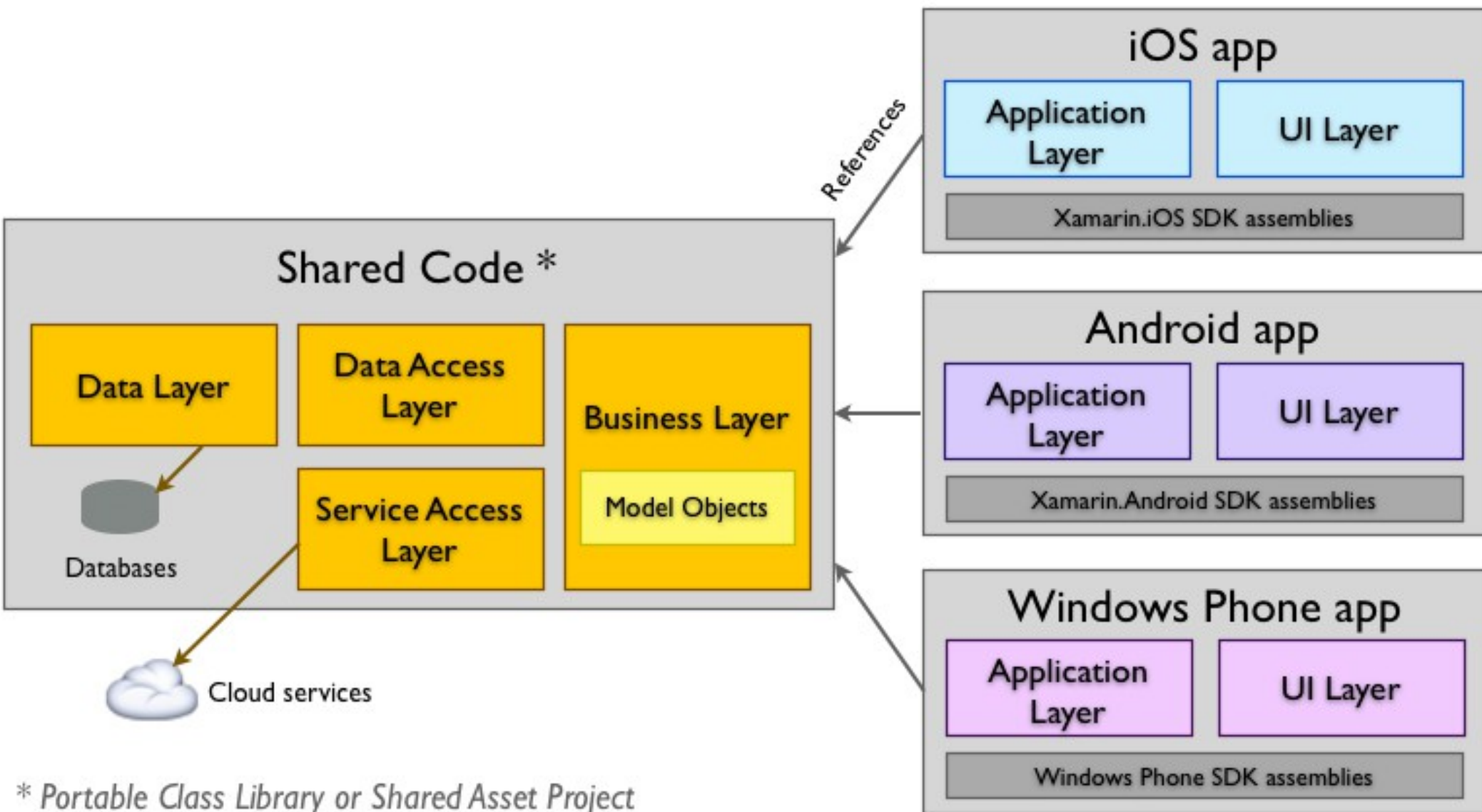


One Codebase — Many Platforms



Native UI and UX





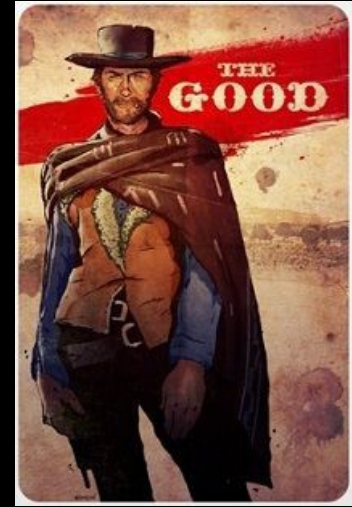
<http://bit.ly/1xV1ATx>

PCL - Portable Class Libraries

A standard by M\$ and Xamarin

Your Code Runs Everywhere

If a platform has .NET



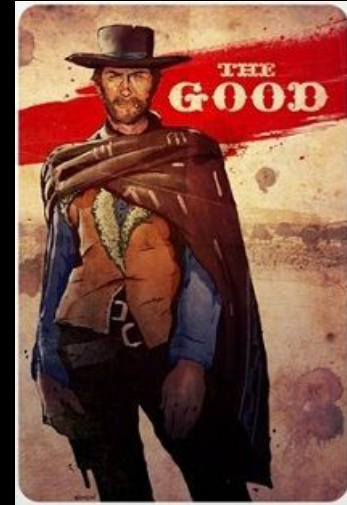
CI requires NO Simulator







NuGet

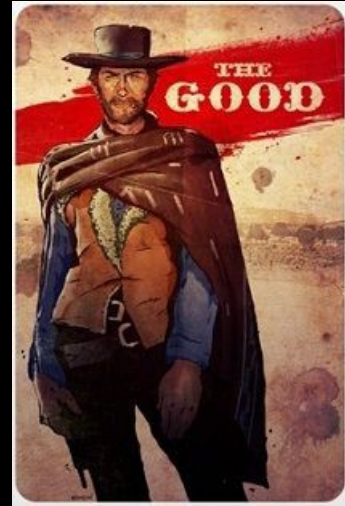


Binary Packets



As Good as Swift

- Closures
- Optionals
- Namespaces
- Generics
- Type Safety
- Functional



You can use exceptions

And you should do so

Callback Hell



<http://bit.ly/11suEXP>

```

- (void)suspendURLIfDownloading:(NSURL *)url automaticallyResumeLater:(BOOL)autoResume
{
    [_session getTasksWithCompletionHandler:^(NSArray *dataTasks, NSArray *uploadTasks, NSArray *downloadTasks) {
        [downloadTasks enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL *stop) {
            NSURLSessionDownloadTask *task = (NSURLSessionDownloadTask *)obj;
            if ([task.originalRequest.URL.absoluteString isEqualToString:url.absoluteString]) {
                *stop = YES;
                [task cancelByProducingResumeData:^(NSData *resumeData) {
                    [_queue addOperationWithBlock:^(
                        [_db executeUpdate:
                            @"UPDATE FCDownloadQueue SET state = ?, resumeData = ? WHERE url = ?",
                            @( autoResume ? FCDownloadStatePausedAutoResume : FCDownloadStatePausedDoNotAutoResume ),
                            resumeData ? resumeData : [NSNull null],
                            url.absoluteString
                        ];
                    [self.delegate downloadQueue:self didPauseDownloadingURL:url userInfo:[self userInfoForURL:url]];
                    [self ensureEnoughDownloadsAreRunning];
                }];
            }
        }];
    }];
}

```

<http://bit.ly/1ryzV4Z>

Async/Await

Async but Looks Linear

```
public class RestApiCallFlow
{
    public static async Task<TResult> LoadRequestFromNetworkFlow
    <TRequest, THttpResult, TResult>(
        TRequest request,
        IRestApiCallTasks<TRequest,
        THttpResult, TResult> stages)
    {
        string requestUrl =
            await stages.BuildRequestUrlForRequestAsync(request);

        // TODO : check cache

        THttpResult serverResponse =
            await stages.SendRequestForUrlAsync(requestUrl);

        TResult parsedData =
            await stages.ParseResponseDataAsync(serverResponse);

        return parsedData;
    }
}
```

```
public class RestApiCallFlow
{
    public static async Task<TResult> LoadRequestFromNetworkFlow
    <TRequest, THttpRequest, TResult>(
        TRequest request,
        IRestApiCallTasks<TRequest,
        THttpRequest, TResult> stages)
    {
        string requestUrl =
            await stages.BuildRequestUrlForRequestAsync(request);

        try
        {
            TResult parsedDataFromCache =
                await stages.LoadDataFromCacheForRequestAsync(request);

            return parsedDataFromCache;
        }
        catch
        {
            // Load from network code
        }
    }
}
```

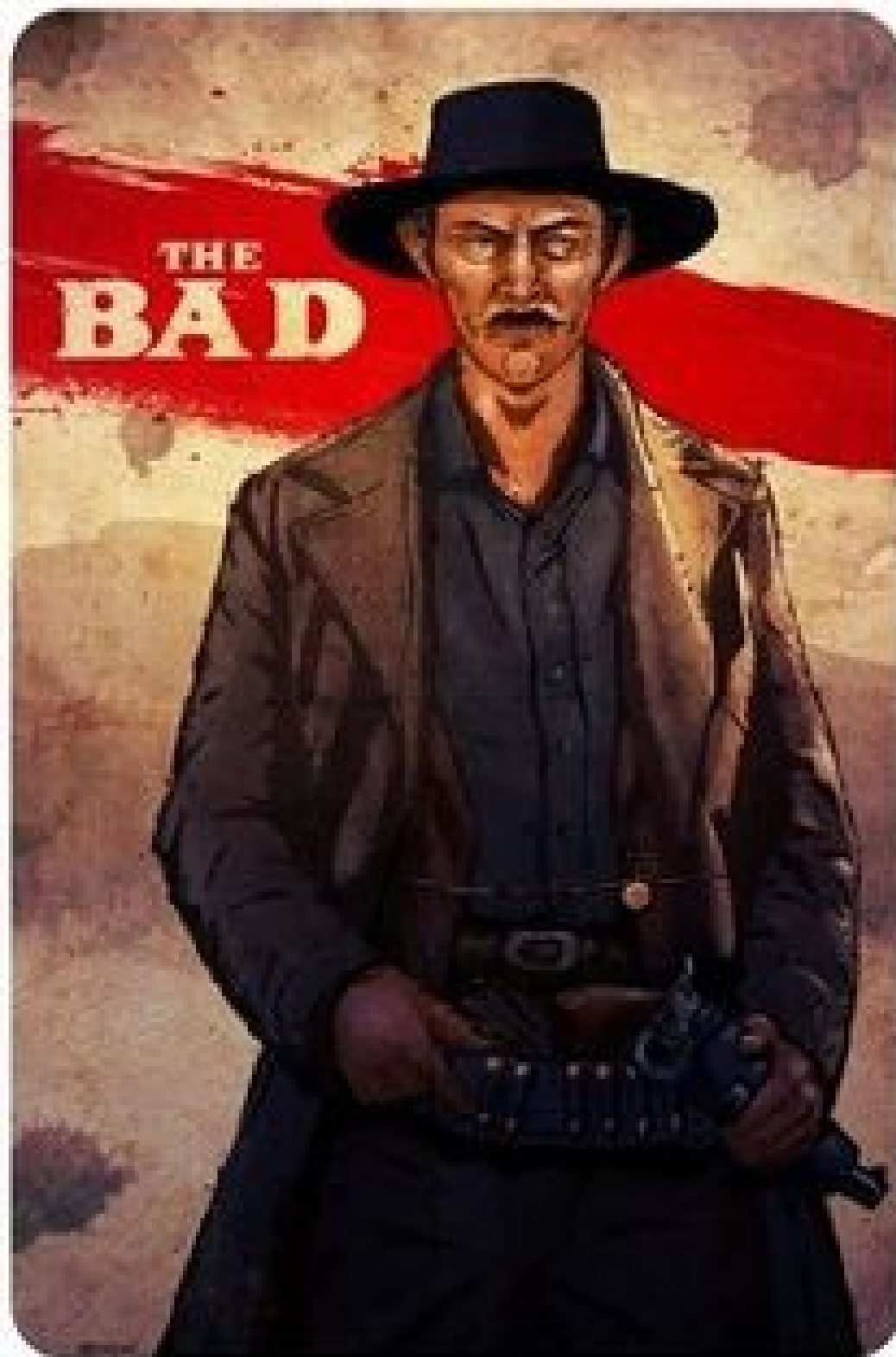
```
public interface IRestApiCallTasks<TRequest, THttpResult, TResult>
{
    Task<string> BuildRequestUrlForRequestAsync(TRequest request);

    Task<THttpResult> SendRequestForUrlAsync(string requestUrl);

    Task<TResult> ParseResponseDataAsync(THttpResult httpData);

    Task<TResult> LoadDataFromCacheForRequestAsync(TRequest request);
}
```

SOLD



Is this lib portable?

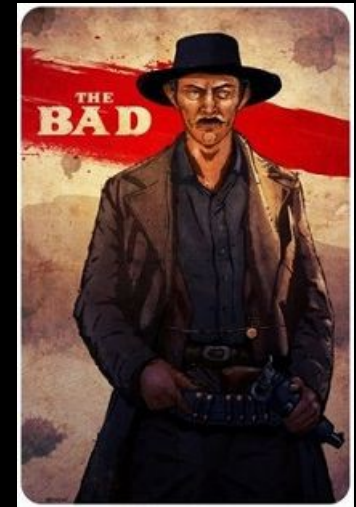
Same API may Work Differently



Xamarin is Slow and has Lots of Memory Leaks

From some holy war

Image Loading : A Typical Implementation



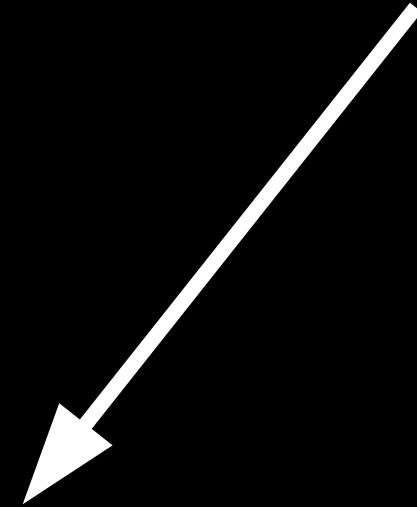
```
using (var webclient = new WebClient())  
{  
    var imageBytes = webclient.DownloadData(url);  
  
    return UIImage.LoadFromData(NSData.FromArray(imageBytes));  
}
```

[http://bit.ly/
1rMJL3B](http://bit.ly/1rMJL3B)

Words cannot express how much I
don't care.

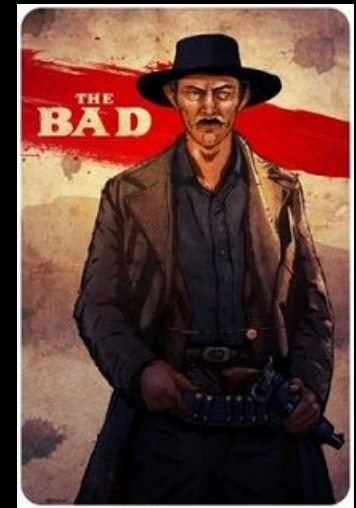


Synchronous



```
using (var webclient = new WebClient())  
{  
    var imageBytes = webclient.DownloadData(url);  
  
    return UIImage.LoadFromData(NSData.FromArray(imageBytes));  
}
```

[http://bit.ly/
1rMJL3B](http://bit.ly/1rMJL3B)



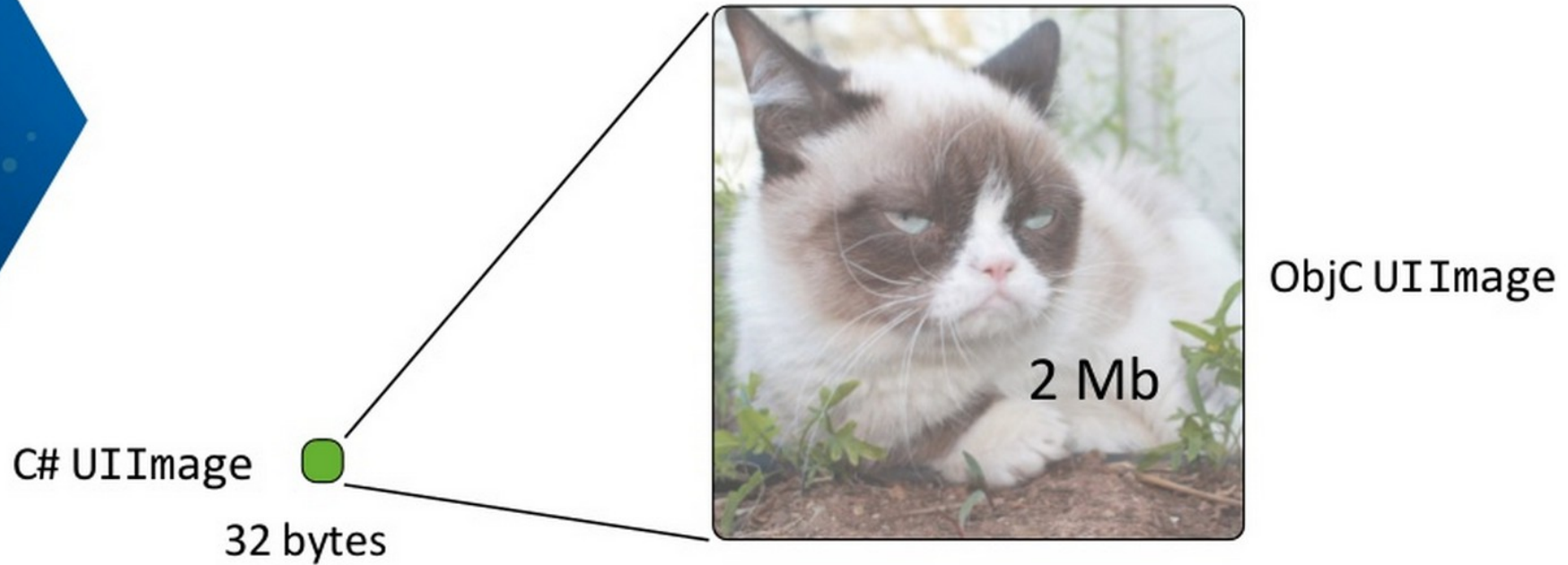
```
using (var webclient = new WebClient())  
{  
    var imageBytes = webclient.DownloadData(url);  
  
    return UIImage.LoadFromData(NSData.FromArray(imageBytes));  
}
```

Leaks

<http://bit.ly/1rMJL3B>

The Illusion

The garbage collector cannot see what's
behind an innocent object



<http://bit.ly/1cay6bO>

Dispose Your Resources

```
01 public override void ViewDidLoad ()
02 {
03     var imageView = new UIImageView (IMG_VIEW_POSITION);
04     var image = UIImage.FromBundle ("grumpy-cat.jpg");
05     imageView.Image = image;
06     View.Add (imageView);
07
08     var button = UIButton.FromType (UIButtonType.RoundedRect);
09     button.Frame = BUTTON_POSITION;
10     View.Add (button);
11
12     button.TouchUpInside += (sender, e) => {
13         imageView.RemoveFromSuperview ();
14         imageView.Dispose ();
15         image.Dispose ();
16     };
17 }
```

<http://bit.ly/1cay6bO>

What If I Told You...



All Obj-C/Java bindings are
IDisposable

UIImage, NSData, sockets,
threads

```
byte[] data = null;
using (Stream response = await
session.DownloadResourceAsync(request))
using (MemoryStream responseInMemory = new MemoryStream())
{
    await response.CopyToAsync(responseInMemory);
    data = responseInMemory.ToArray();
}

BeginInvokeOnMainThread(delegate
{
    using ( NSData imageData = NSData.FromArray(data) )
    {
        using ( UIImage image = new UIImage(imageData) )
        {
            this.ImageView.Image = image;
        }
    }
});
```

<http://bit.ly/1ukw1ii>

```
byte[] data = null;
using (Stream response = await
session.DownloadResourceAsync(request))
using (MemoryStream responseInMemory = new MemoryStream())
{
    await response.CopyToAsync(responseInMemory);
    data = responseInMemory.ToArray();
}
```

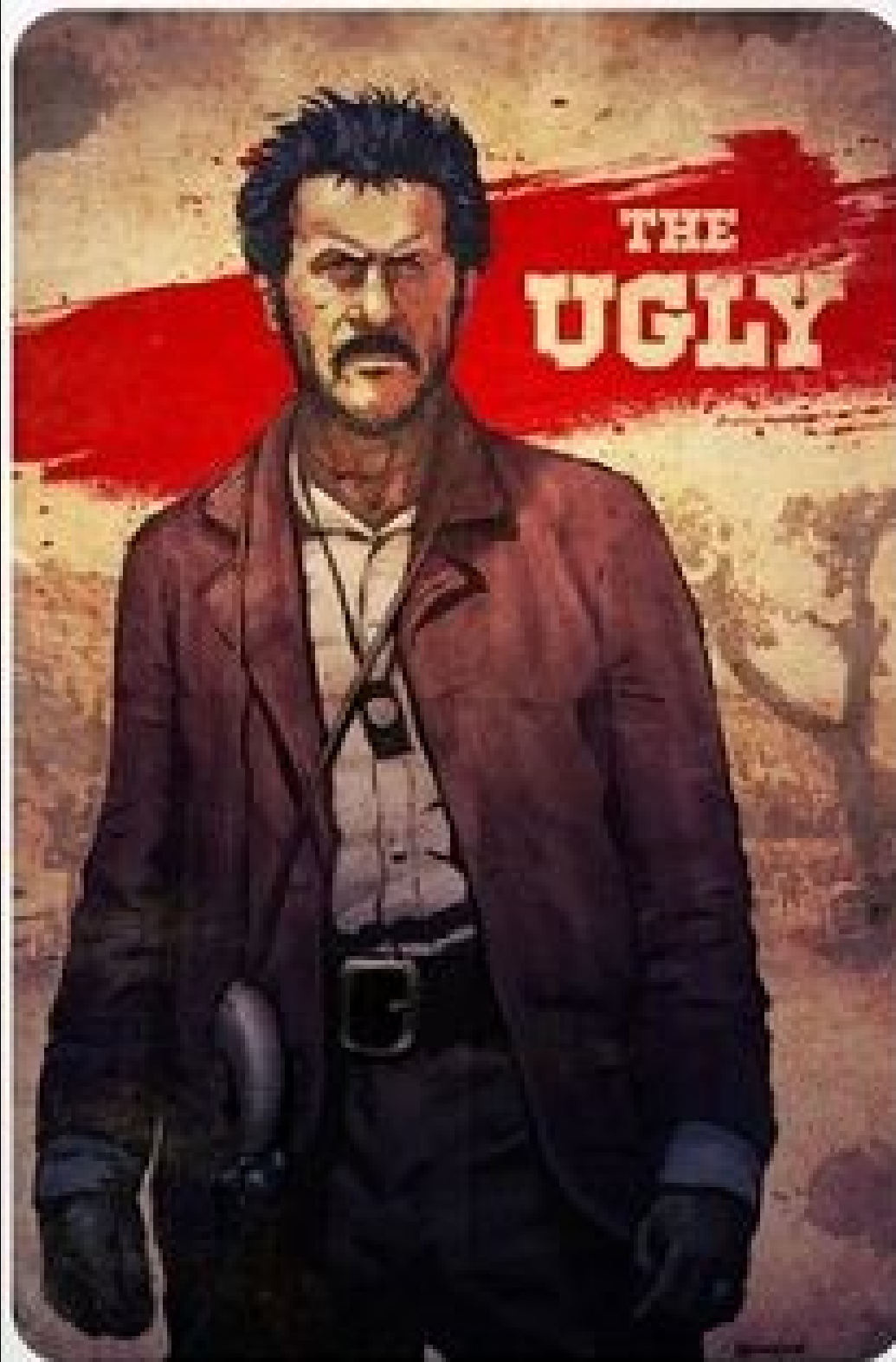
```
BeginInvokeOnMainThread(delegate
{
    using ( NSData imageData = NSData.FromArray(data) )
    {
        using ( UIImage image = new UIImage(imageData) )
        {
            this.ImageView.Image = image;
        }
    }
});
```

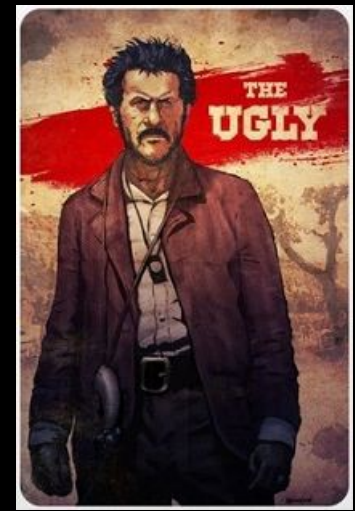
<http://bit.ly/1ukw1ii>


```
byte[] data = null;
using (Stream response = await
session.DownloadResourceAsync(request))
using (MemoryStream responseInMemory = new MemoryStream())
{
    await response.CopyToAsync(responseInMemory);
    data = responseInMemory.ToArray();
}

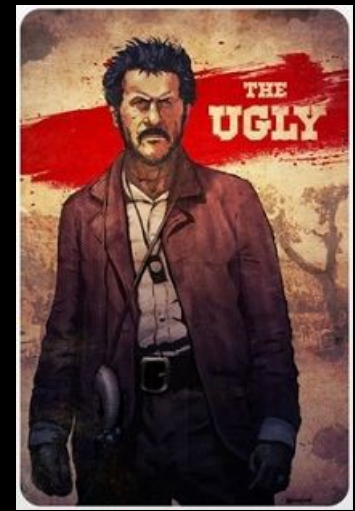
BeginInitInvokeOnMainThread(delegate
{
    using ( NSData imageData = NSData.FromArray(data) )
    {
        using ( UIImage image = new UIImage(imageData) )
        {
            this.ImageView.Image = image;
        }
    }
});
```

<http://bit.ly/1ukw1ii>

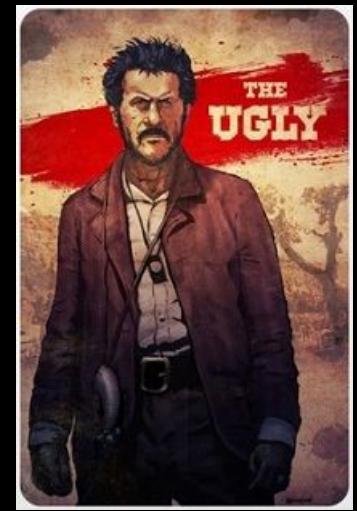




Xamarin.Forms



Aiming for a Single Codebase



Break their own Principles

Unlike other cross-platform mobile frameworks that only offer lowest common denominator experiences through UI abstraction libraries, we make 100% of the iOS and Android APIs available through our native bindings.



Per Seat Per Platform

	STARTER FREE	INDIE \$25 / month paid monthly or annually	BUSINESS \$83 / month paid annually (\$999 / year) MOST POPULAR	ENTERPRISE \$158 / month paid annually (\$1899 / year)
Permitted Use	Individual	Individual	Organization	Organization
Subscription Type	N/A	Monthly	Annual	Annual
Deploy to Device	✓	✓	✓	✓
Deploy to App Stores	✓	✓	✓	✓
Xamarin Studio	✓	✓	✓	✓
Unlimited App Size		✓	✓	✓
Xamarin.Forms		✓	✓	✓
Visual Studio Support			✓	✓
Business Features			✓	✓
Prime Components				✓
Email Support			✓	✓
One Business Day SLA				✓
Hotfixes				✓

Try it!

Test on All Platforms

Dispose Native Objects
Aggressively

Build Dedicated GUI for
Each Platform

Try it!