C# 5.0 Async

Pratik Khasnabis

DDD Melbourne 2012

About Me

Quick background
Lead Developer at Bupa Australia
Developer for 15 years
C# & .Net 8 years
C++ before that

Disclaimer
This presentation is my own and I do not represent my company.

@softveda

The need for Async

Responsive UI

UI thread is running as a message pump in a loop

Waiting on IO or long computation will stop message processing => Frozen UI

Touch ready Metro Apps

In WinRT if an API is likely to take more than 50ms to run the API is asynchronous

Scalable Server Applications

A CLR thread is used to service requests

A blocked thread => less scalable

Service thousands of IO bound requests on a small pool of threads

Async in .Net 4.5

First introduced in PDC 2010 and the refresh in MIX 2011 (VS 2010 SP1)

Two new keywords async and await

An unsupported out-of-band release. Updated C# and VB compilers.

History

Built to take advantage of Task and Task<T> Introduced in .Net 4.0

AsyncCtpLibrary.dll introduced async extension methods for common classes.

Async methods Pervasive in .Net 4.5

.NET 4 and Silverlight 5

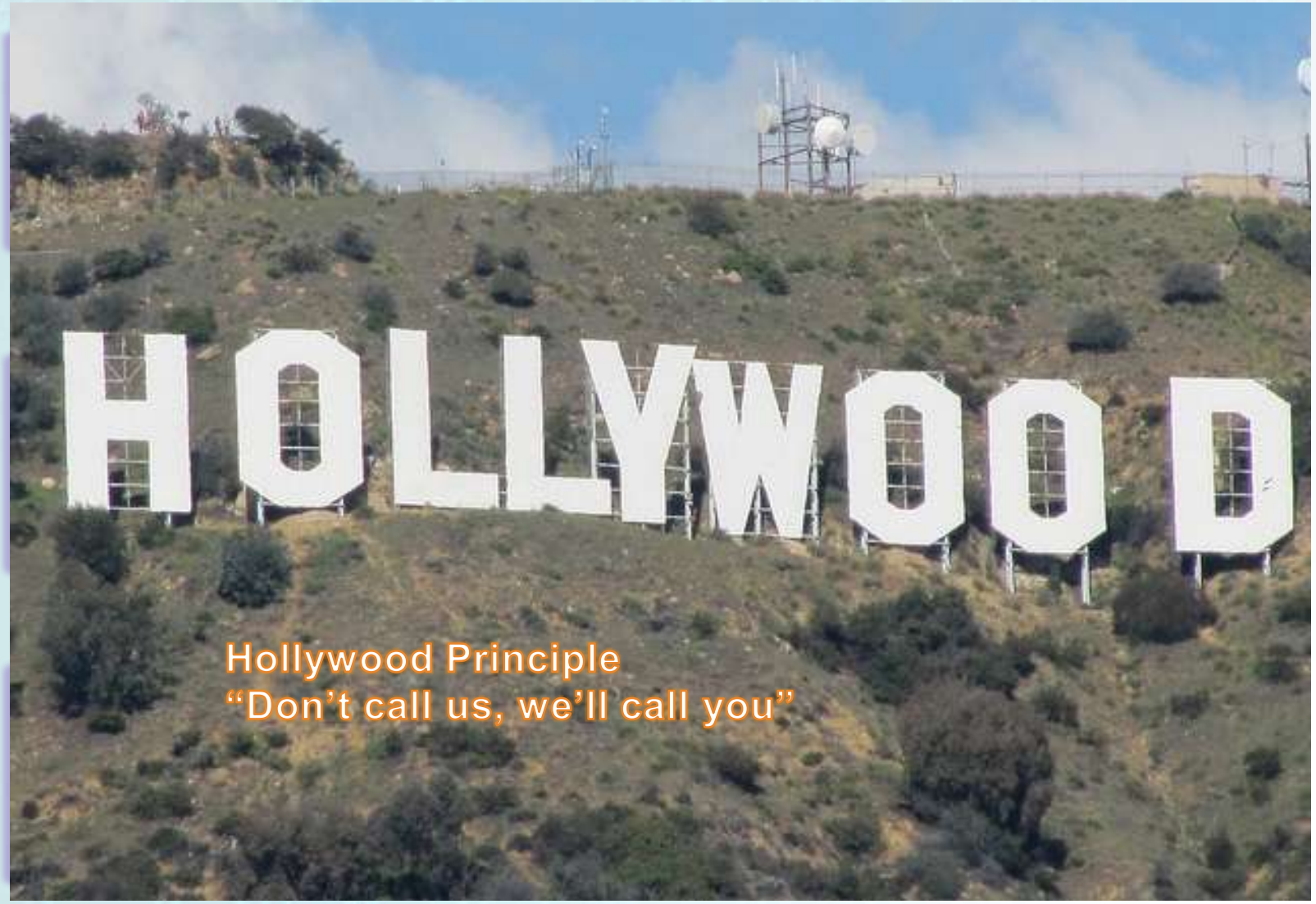
Async Targeting Pack
Download using NuGet
Microsoft.CompilerServices.AsyncTargetingPack

Requires VS 2012

Differences in Behaviour
Read the release notes
Static helper methods are in TaskEx class instead of Task
e.g. Task.Run(…) vs TaskEx.Run(…)

Windows Phone and Azure
No support yet

Task



Async and Await

Keyword: async

Only methods or lamdbas with async modifier can use the await keyword.

Return type must be void, Task or Task<T>

The method actually returns void or T, the compile creates the Task object

Doesn't make the method asynchronous

Keyword: await

Makes the rest of method a continuation

When the method completes execution resumes where it left off on the right synchronisation context

Compiler generates a state machine

```
var result = await expression;
<code block>
```

```
var awaiter = expression.GetAwaiter();
if (awaiter.IsCompleted)
  Console.WriteLine (awaiter.GetResult());
else
  awaiter.OnCompleted (() =>
  {
    var result = awaiter.GetResult();
    <code block>
  );
```

Task-Based Async Pattern

TAP methods has an async modifier , returns a running Task or Task<TResult> and ends with an “Async” suffix by convention

TAP methods should return quickly to caller with a small synchronous phase.

TAP methods should have the same parameters as the synchronous one in the same order

Avoids out and ref parameters

Can have CancellationToken parameter

Can have IProgress<T> parameter

Async in .Net 4.5 BCL

Methods doing IO

System.Xml.XmlReader
public virtual Task<bool> ReadAsync()

System.IO.Stream
public Task CopyToAsync(Stream destination)

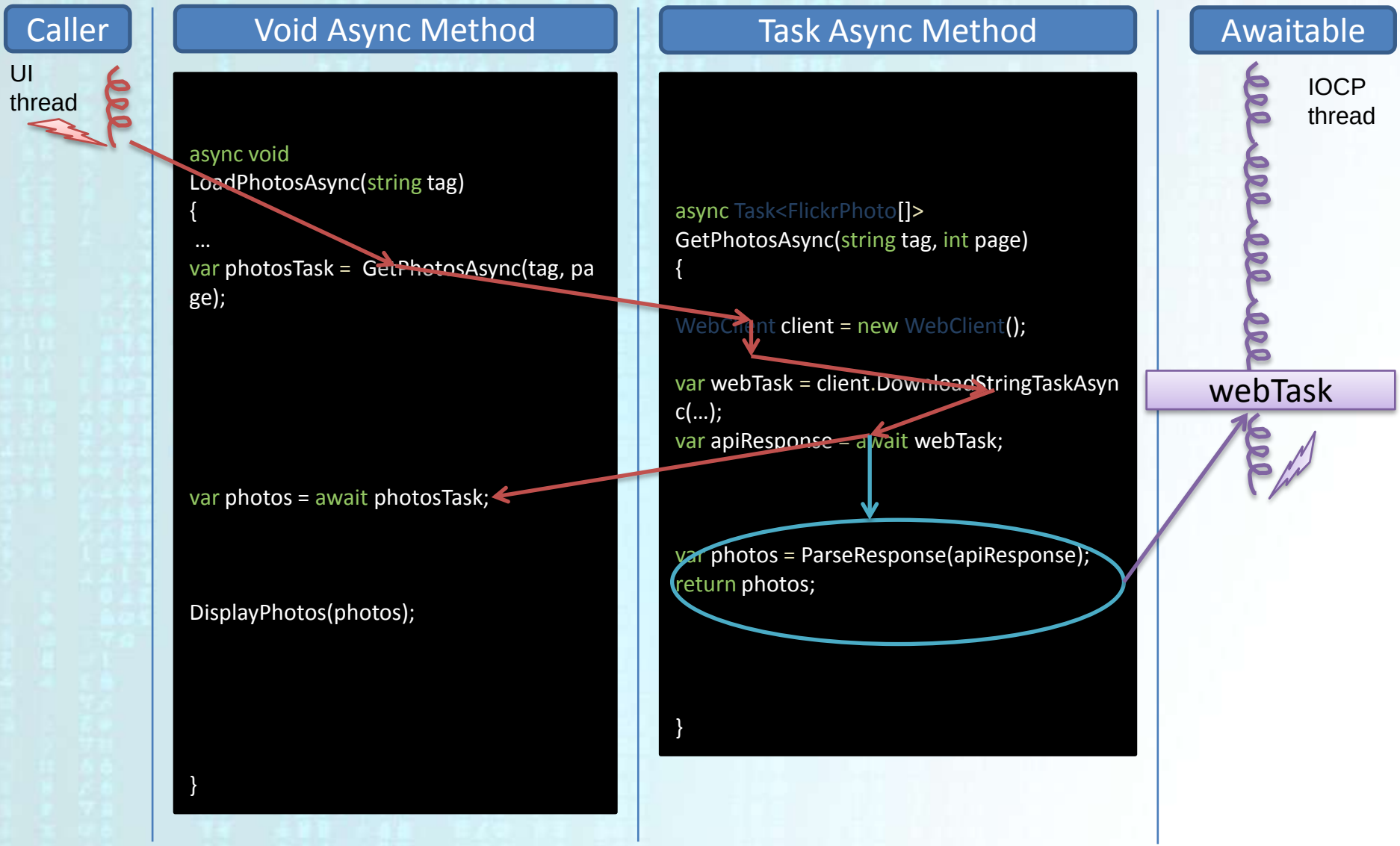
System.IO.TextReader
public virtual Task<int> ReadAsync(char[] buffer, int index, int count)

System.Net.Mail
public Task SendMailAsync(MailMessage message)

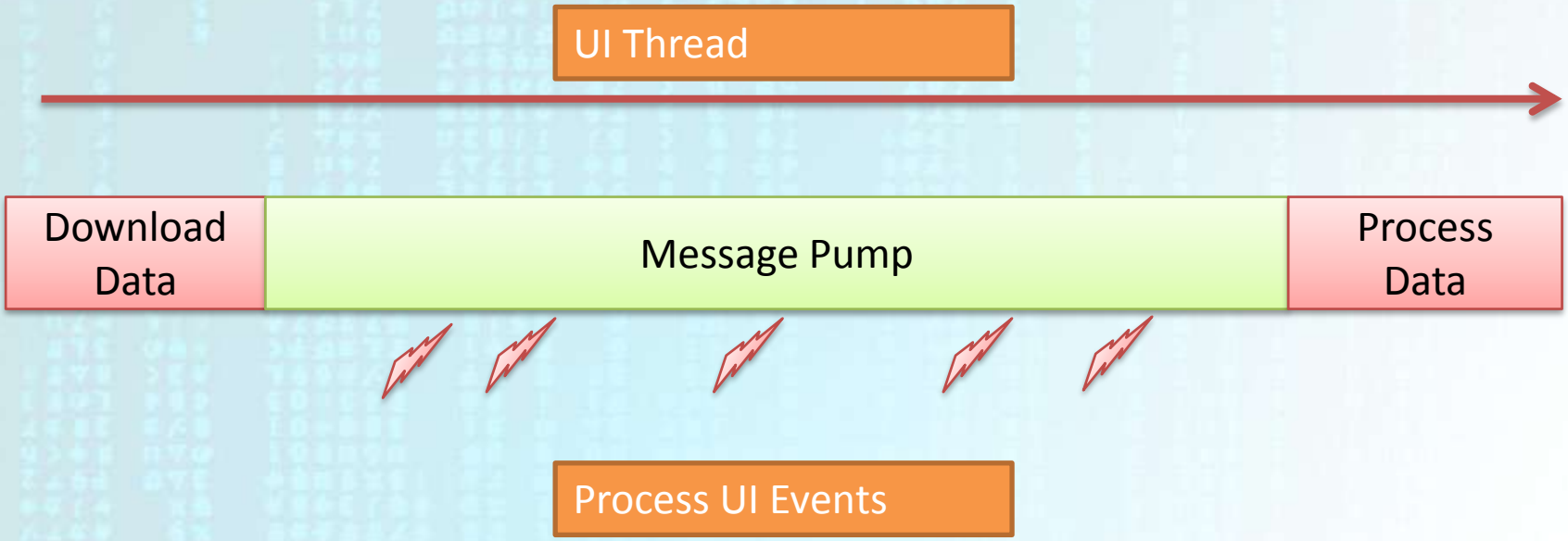
System.Net.Http.HttpClient
public Task<string> GetStringAsync(string requestUri)

System.Net.WebSockets
public abstract Task<WebSocketReceiveResult> ReceiveAsync(ArraySegment<byte> buffer, CancellationToken cancellationToken)

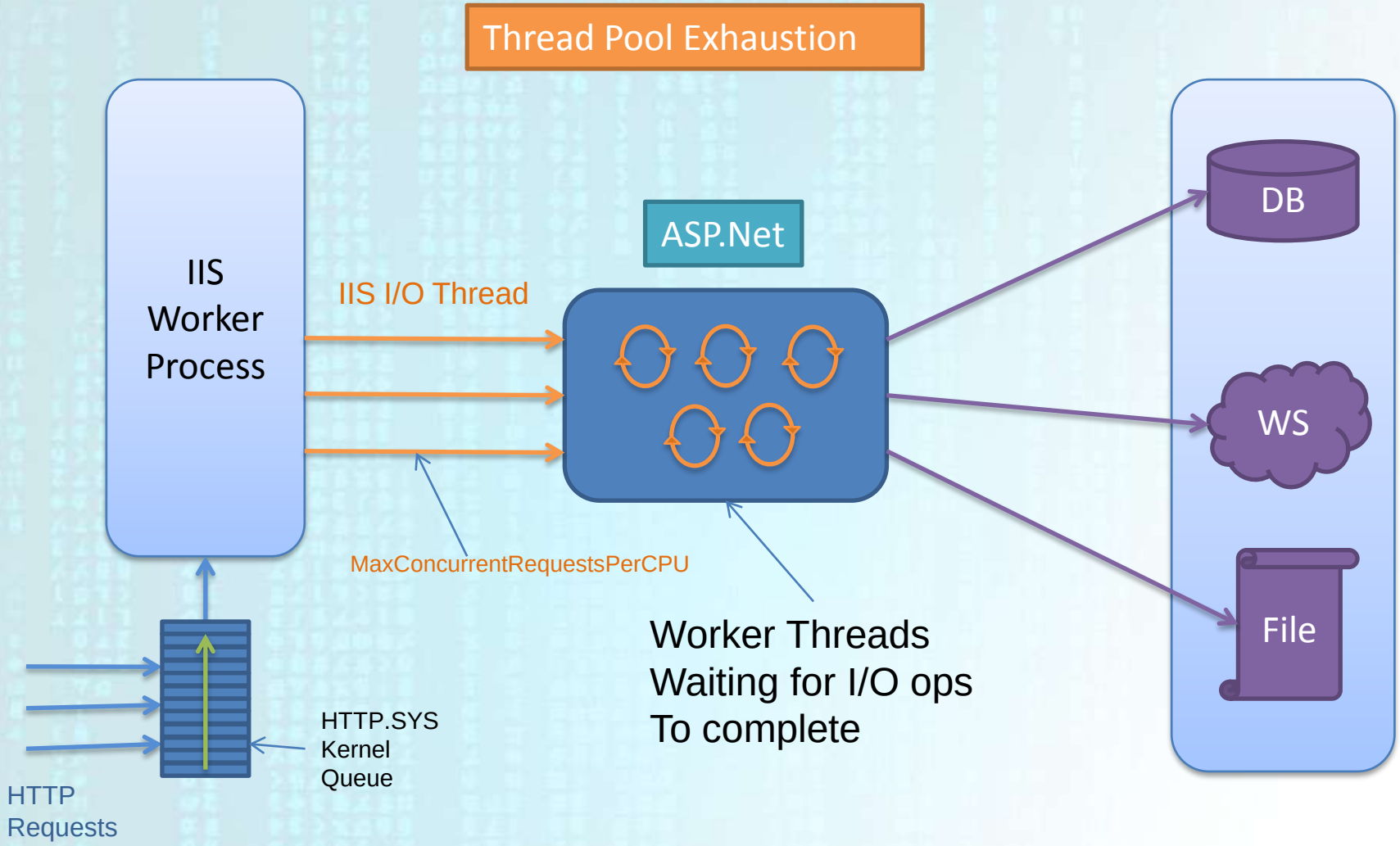
Async in WPF/WinForms apps



Concurrency without threads



Async in Server Apps



ASP.Net Core Services

Asynchronous HTTP modules

```
public class MyHttpModule : IHttpModule
{
private async Task
ScrapeHtmlPage(object caller, EventArgs e)
{
 await ...
}
  public void Init(HttpApplication
context)
 {
   EventHandlerTaskAsyncHelper helper =
   new EventHandlerTaskAsyncHelper(ScrapeHtmlPage);
   context.AddOnPostAuthorizeRequestAsync(
     helper.BeginEventHandler, helper.EndEventHandler);
 }
}
```

Asynchronous HTTP handlers

```
public class MyAsyncHandler :
HttpTaskAsyncHandler
{
    public override async Task
ProcessRequestAsync(HttpContext
context)
    {
        await …
    }
}
```

ASP.Net MVC 4

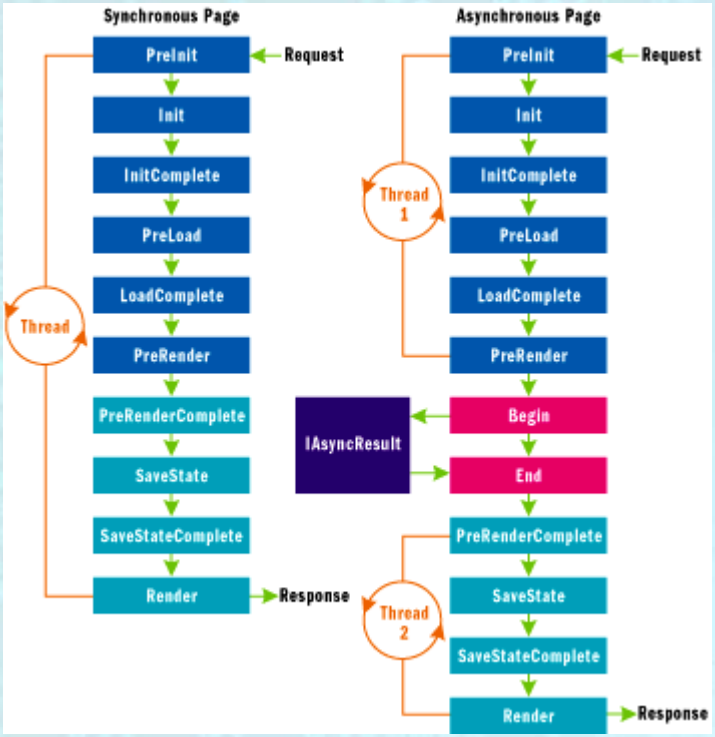
Controller
Derive from AsyncController ??

Actions
async methods returning Task or Task<ActionResult>

Timeout

```
[AsyncTimeout(2000)]
[HandleError(ExceptionType =
typeof(TaskCanceledException), View
= "TimedOut")]
```

ASP.Net WebForms



Page Markup

<%@ Page Language="C#" Async="true" AsyncTimeOut="2" CodeBehind="AsyncPage.aspx.cs" %>

Page Load method

Register the async method

The async mehtod will be asynchronoiusly executed after PreRender stage in page lifecycle

WCF

Service: Async operations

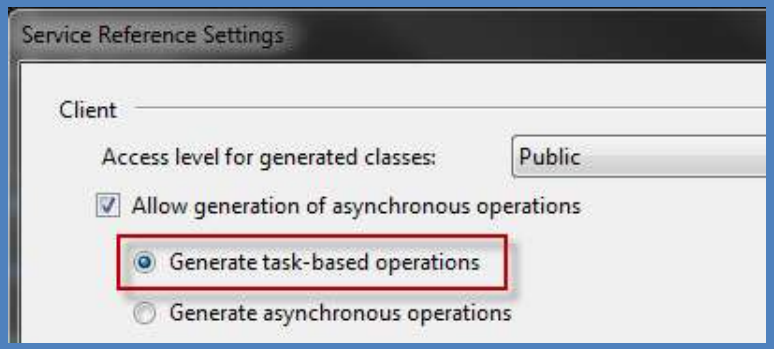
```
[ServiceContract]
public interface IDemoService
{
  [OperationContract]
  Task<string> GetStockQuoteAsync(string ticker);
}

public class DemoService : IDemoService
{
  public async Task<string> GetStockQuoteAsync
(string    ticker)
  {
    await ...
  }
}
```

Client: Async proxies

Use svcutil or Add Service Reference

```
var proxy = new
StockQuoteService.StockQuoteSoapClie
nt();
var quote = await
proxy.GetQuoteAsync("MSFT");
```



VS 2012 Unit Tests

Async Test Methods – return Task

```
[TestMethod]
public async Task UnitTestMethod()
{
    await Task.Delay(1000);
    Assert.AreEqual(1,1);
}
```

Execution Time

1 sec

Test Frameworks

MS Test – Supports async, Built in test provider

xUnit – Supports async in v1.9, downloadable test provider

NUnit – Doesn't support async, downloadable test provider

Metro and WinRT

http://blogs.msdn.com/b/windowsappdev/archive/2012/06/14/exposing-net-tasks-as-winrt-asynchronous-operations.aspx
http://blogs.msdn.com/b/windowsappdev/archive/2012/03/20/keeping-apps-fast-and-fluid-with-asynchrony-in-the-windows-runtime.aspx
http://blogs.msdn.com/b/windowsappdev/archive/2012/04/24/diving-deep-with-winrt-and-await.aspx

Watch the other sessions on Metro App Development

IAsyncInfo

All async methods in WinRT implements this interface and 4 other

IAsyncAction
IAsyncOperation<TResult>
IAsyncActionWithProgress<TProgress>
IAsyncOperationWithProgress<TResult, TProgress>

Compiler Magic

IAsyncInfo has the awaitable pattern
The C# and VB compiler will convert WinRT async methods to return Tasks

Choosing Sync vs Async

Synchronous

Operations are simple or short-running

CPU bound operations

Simplicity is more important

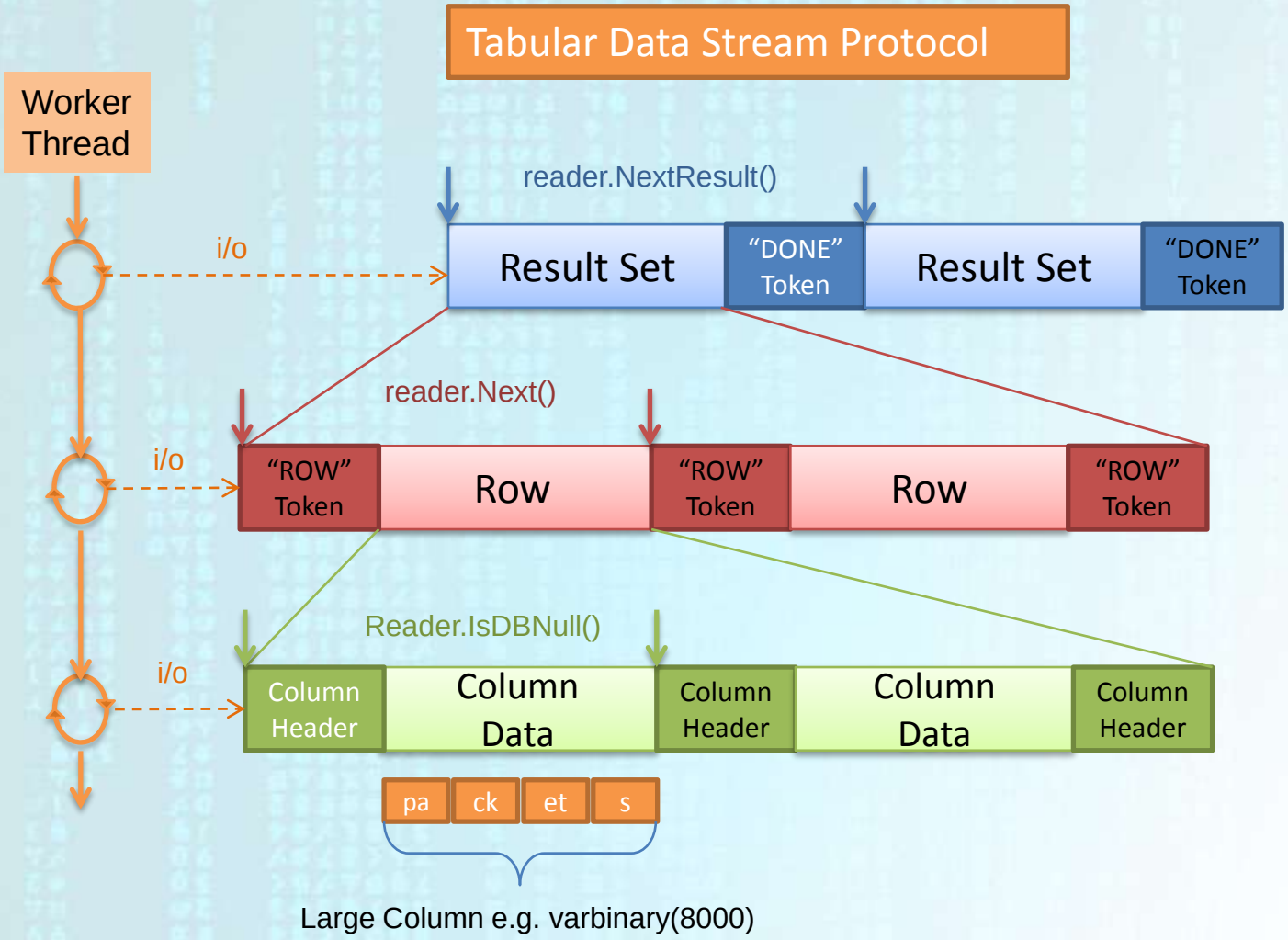
Asynchronous

I/O bound operations (DB, network, disk)

Provide cancellation support

Parallelism is more important

ADO.Net


Tabular Data Stream Protocol
Worker Thread
reader.NextResult()
i/o
Result Set
"DONE" Token
Result Set
"DONE" Token
reader.Next()
i/o
"ROW" Token
Row
"ROW" Token
Row
"ROW" Token
Reader.IsDBNull()
i/o
Column Header
Column Data
Column Header
Column Data
Column Header
pa
ck
et
s
Large Column e.g. varbinary(8000)

ADO.Net

Recommendations

Use
CommandBehavior.SequentialAccess
Unless reading large columns

Use
NextResultAsync() and ReadAsync()

Use
Synchronous column access
IsDBNull, GetFieldValue<T>
Unless reading large columns
IsDBNullAsync,
GetFieldValueAsync<T>

Use
Streaming for massive columns
GetStream()
GetTextReader()
GetXmlReader()

Resources

- http://msdn.microsoft.com/en-US/async
- VB.Net - http://blogs.msdn.com/b/lucian/
- C# (implementation details) - https://msmvps.com/blogs/jon_skeet/archive/tags/Eduasync/default.aspx