

A photograph of a brown plush monkey toy with a smiling face, looking out from a circular stone window. The window offers a scenic view of a city skyline under a clear sky.

XAM150

Consuming REST-based Web Services



Xamarin University

Objectives

1. Obtain the device's network capabilities
2. Introduce REST
3. Consume REST services with Xamarin
4. Integrate with platform-specific network features

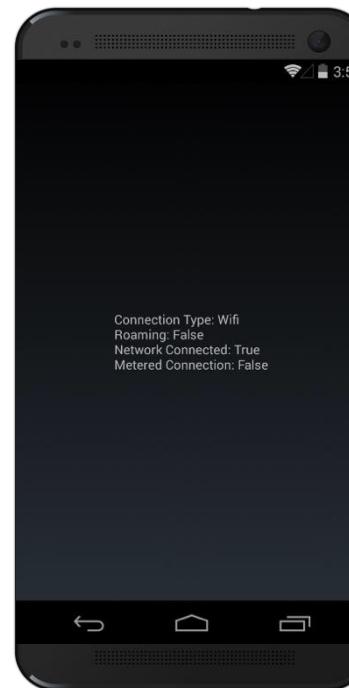




Obtain the device's network capabilities

Tasks

- ❖ Determine if the device has a connection
- ❖ Obtain the device's connection type
- ❖ Determine when network availability changes



Web Services

- ❖ More often than not, mobile apps need to access and use external data - most commonly as REST or SOAP based web services
- ❖ Xamarin.Forms apps have full support for both styles and the code you build to interact with your services can often be shared



Preparing for challenges

- ❖ Cellular network isn't always the most reliable transfer mediums and can cause your app to fail
- ❖ Slow transfer speeds can add latency and performance issues in your app
- ❖ Unexpected roaming and data usage charges can make users unhappy



Working with Mobile Networks

- ❖ Mobile applications that utilize network data are interested in several key pieces of information which are obtained using platform-specific APIs



Working with Mobile Networks

- ❖ Mobile applications that utilize network data are interested in several key pieces of information which are obtained using platform-specific APIs



Connection
Type

Connection
Status

Disconnected,
Available,
Connecting,
Connected, etc.

Working with Mobile Networks

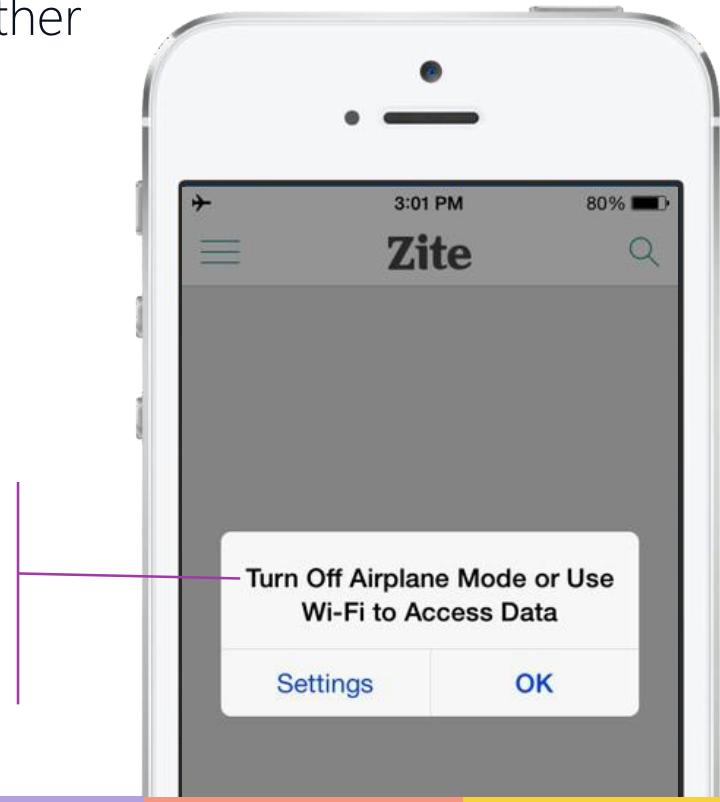
- ❖ Mobile applications that utilize network data are interested in several key pieces of information which are obtained using platform-specific APIs



First things first: is there a network?

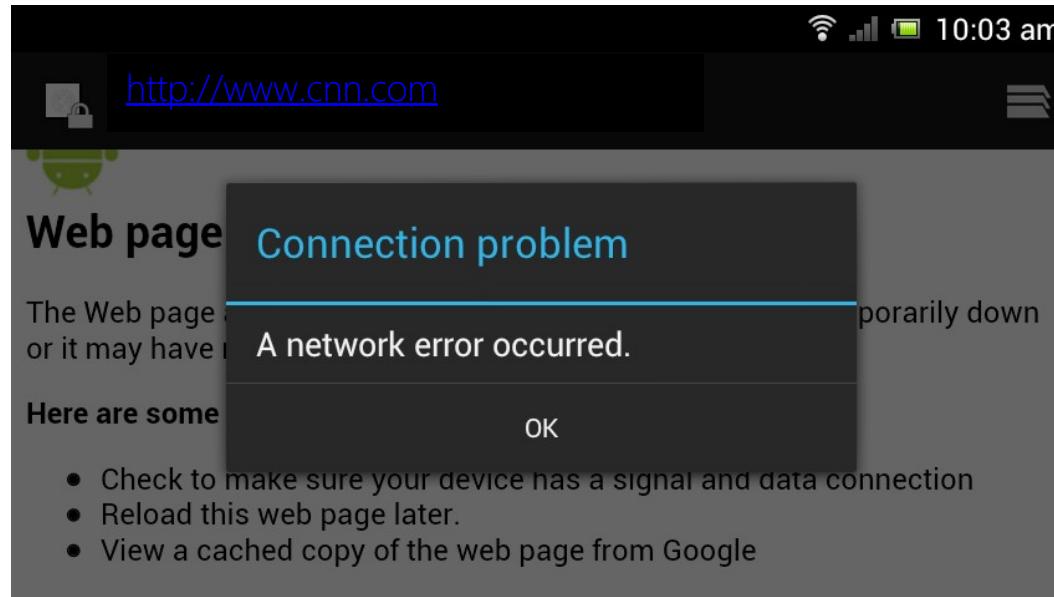
- ❖ Applications should always determine whether a network is available before starting a network operation

When no network is available, the application can provide a helpful prompt to request user intervention



Connection status

- ❖ Connection status can change at any time; the application is responsible for monitoring connection status and responding in a user-friendly fashion



Connection types

- ❖ Mobile devices can access networks using three different network styles, each has different pricing, performance and reliability



Cellular
(CDN)



Roaming
Cellular



Wi-Fi

Note: Devices can also be configured to not allow certain connection types which will generally be reported as no network available to the application



Connection type comparisons

- ❖ Depending on the connection type the device is using, the bandwidth and cost will vary greatly

Network Type	Typical download speed	4MB download
2G (EDGE)	125kbps	~2m 16s
3G	800kbps	~40s
4G (LTE)	1.5mbps	~21s
WiFi	5-40mbps	~1 - 7s



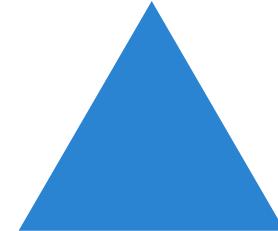
It's important to know what network type the device is on because the app can change the user experience in response, e.g. "This is taking longer than expected..."

High cost networks

- ❖ Android and Windows allow you to detect higher-cost networks, for example when roaming or the connection is metered
- ❖ Allows applications to prompt the user for permission before performing larger data transfers



Android



Windows Mobile

Users can tell when they are roaming through status bar icons, or through the displayed carrier name on iPhone

Platform-specific APIs

- ❖ Each platform has unique APIs to detect, monitor and work with the networking hardware

```
using Android.Net;  
...  
ConnectivityManager connectivityManager =  
    (ConnectivityManager) Application.Context  
        .GetSystemService(Context.ConnectivityService);  
  
bool isConnected = connectivityManager.ActiveNetworkInfo != null  
    && connectivityManager.ActiveNetworkInfo.IsConnected;
```

For Android, use the
ActiveNetworkInfo property



Cross Platform network detection

- ❖ Open source Connectivity Plugin includes PCL support with implementations for UWP, Mac, iOS and Android

Connectivity Plugin for Xamarin and... 2.1.1

Get network connectivity information such as network type, speeds, and if connection is available. Additional functionality includes the ability to ping a specific host and port number. Ensure you have proper permissions set by reading the README.

To install Connectivity Plugin for Xamarin and Windows, run the following command in the [Package Manager Console](#)

```
PM> Install-Package Xam.Plugin.Connectivity
```

Using the connectivity plug-in

- ❖ Connectivity plug-in exposes **CrossConnectivity.Current** instance to access connection, bandwidth and connection change notifications

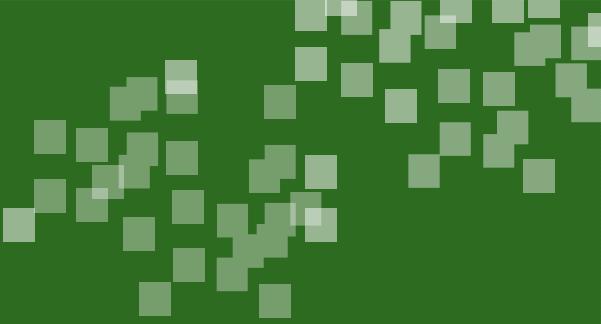
```
bool isConnected = CrossConnectivity.Current.IsConnected;  
...  
CrossConnectivity.Current.ConnectivityChanged += (sender,e) =>  
{  
    bool stillConnected = e.IsConnected;  
    ...  
};
```

Reporting network activity

- ❖ Common to use activity indicator, or indeterminate progress ring to report network activity; can use platform-specific approach, or Xamarin.Forms has page-level property

```
this.IsBusy = true;      // On Page instance method  
  
try {  
    // Network code goes here  
}  
finally {  
    this.IsBusy = false;  
}
```





Flash Quiz

Flash Quiz

- ① Monitoring network connections requires platform-specific APIs be used (True or False)?
 - a) True
 - b) False

Flash Quiz

- ① Monitoring network connections requires platform-specific APIs be used (True or False)?
 - a) True
 - b) False

Flash Quiz

- ② To determine if an iOS device is roaming, you need to:
 - a) Check the IsRoaming property on the ConnectivityManager
 - b) Subscribe to the ReachabilityChanged event
 - c) You cannot detect roaming conditions on iOS

Flash Quiz

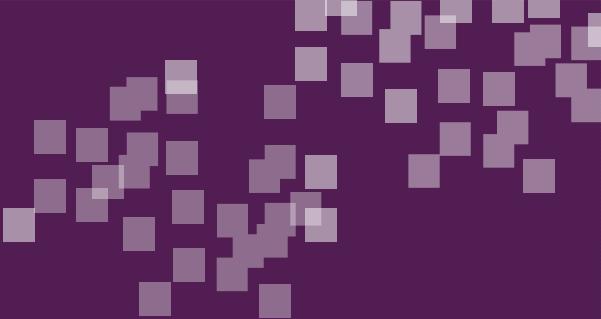
- ② To determine if an iOS device is roaming, you need to:
 - a) Check the IsRoaming property on the ConnectivityManager
 - b) Subscribe to the ReachabilityChanged event
 - c) You cannot detect roaming conditions on iOS

Flash Quiz

- ③ You can obtain network information about an Android device using the method call:
 - a) `Android.GetNetworkInformation`
 - b) `ConnectivityManager.ActiveNetworkInfo`
 - c) `Context.Connection`

Flash Quiz

- ③ You can obtain network information about an Android device using the method call:
- a) `Android.GetNetworkInformation`
 - b) `ConnectivityManager.ActiveNetworkInfo`
 - c) `Context.Connection`



Individual Exercise

Determine the network connectivity

Summary

- ❖ Determine if the device has a connection
- ❖ Obtain the device's connection type
- ❖ Determine when network availability changes





Introduce REST

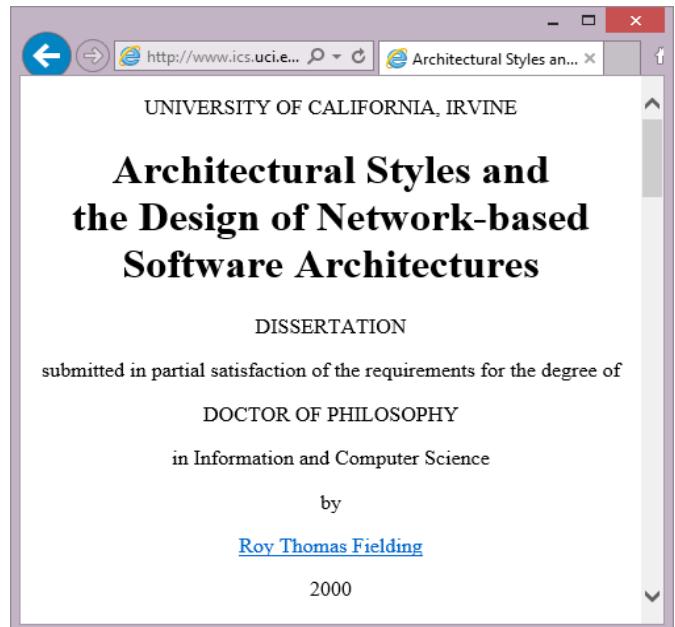
Tasks

- ❖ Identify REST services
- ❖ Utilize URLs in REST
- ❖ Describe guidelines for using REST



What are REST services?

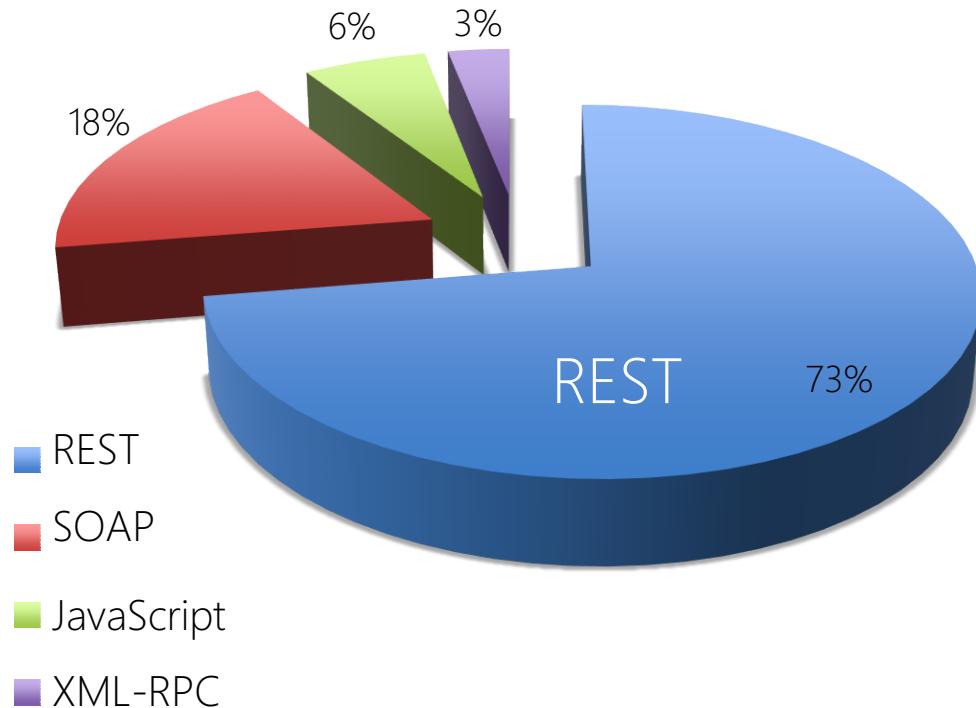
- ❖ REST (Representational State Transfer) is an architecture for creating distributed applications which is modeled around the HTTP specification



Why use REST?

- ❖ REST is designed to take advantage of the architecture of the WWW
 - Operations are implemented as HTTP verbs
 - URLs represent accessible resources

Why use REST?



REST has become the dominant architecture for web services, primarily due to it being highly accessible from JavaScript

REST operations

- ❖ CRUD operations are modeled after HTTP verbs



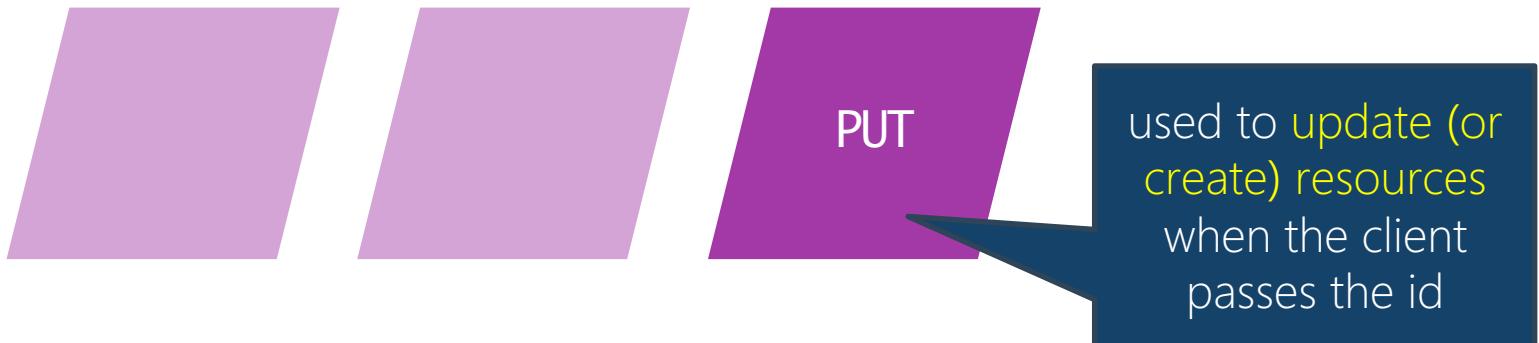
REST operations

- ❖ CRUD operations are modeled after HTTP verbs



REST operations

- ❖ CRUD operations are modeled after HTTP verbs



REST operations

- ❖ CRUD operations are modeled after HTTP verbs



used to delete an
identified
resource



There are other verbs defined by the HTTP specification which can be used as well, but

URLs + Operations

- ❖ URLs are used to identify and organize accessible resources

```
GET https://www.some_address.com/customers/12345
```

 or

```
GET https://www.some_address.com/customers?id=12345
```

REST is very flexible with regards to the URL structure, the main takeaway is that the *URL is predictable and unique* for the resource being accessed

URLs + Operations

- ❖ URLs are used to identify and organize accessible resources

```
GET https://www.some_address.com/customers/12345
```

or

```
GET https://www.some_address.com/customers?id=12345
```

HTTP/1.1 200 OK

Content-Type: text/xml · charset=utf-8

Content-Length: ####

...

HTTP status codes are useful in REST, for example 404 Not Found would be the response if the record does not exist

URLs + Operations

- ❖ URLs are used to identify and organize accessible resources

```
GET https://www.some_address.com/customers/12345
```

or

```
GET https://www.some_address.com/customers?id=12345
```

```
HTTP/1.1 200 OK
```

```
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: ####
```

```
...
```

Content-Type indicates the format of the response body, typically this is XML or JSON, but can also be an image, plain text, or any other valid HTTP format

URLs + Operations

- ❖ URLs are used to identify and organize accessible resources

```
GET https://www.some_address.com/customers/12345
```

or

```
GET https://www.some_address.com/customers?id=12345
```

```
HTTP/1.1 200 OK
```

```
...
```

```
<customer>
  <id>12345</id>
  <name>Joe</name>
  ...
</customer>
```

Response body contains the
requested data

Safe HTTP methods

- ❖ Safe HTTP methods do not modify the resource representation
- ❖ Middleware client proxy servers, networks stacks, and ISPs can *cache* the response for performance (particularly on cellular networks)
- ❖ This provides high scalability for safe operations

HTTP Method	Safe
OPTIONS	yes
GET	yes
HEAD	yes
PUT	no
POST	no
DELETE	no
PATCH	no

Idempotent HTTP methods

- ❖ Idempotent HTTP methods can be called multiple times with the **same data** and it will always produce the **same result** on the server (e.g. no side effects)
- ❖ This means the operation is guaranteed to happen *only* once even if we send multiple requests

HTTP Method	Idempotent
OPTIONS	yes
GET	yes
HEAD	yes
PUT	yes
POST	no
DELETE	yes
PATCH	no

RESTful guidelines

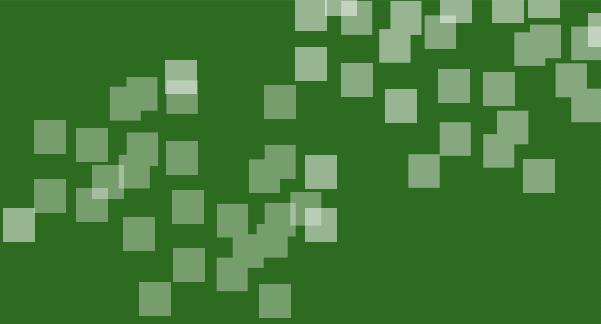
- ❖ Favor JSON if you have a choice (many services will return the data in a variety of formats)
- ❖ Pay attention to status codes and reissue requests to idempotent and safe operations when outcome is uncertain (timeout, etc.)
- ❖ JSON/XML + HTTP doesn't mean the service is really RESTful



Security in REST

- ❖ Security is ultimately decided by the service – the client can only conform to what the service allows
- ❖ Should always prefer https to protect the data peer-to-peer
- ❖ Most services use OAuth2 for authorization and authentication





Flash Quiz

Flash Quiz

- ① What HTTP verb should be used to update an existing record?
- a) GET
 - b) POST
 - c) PUT
 - d) UPDATE

Flash Quiz

- ① What HTTP verb should be used to update an existing record?
- a) GET
 - b) POST
 - c) PUT
 - d) UPDATE

Flash Quiz

- ② One advantage of REST is that many operations are cacheable
- a) True
 - b) False

Flash Quiz

- ② One advantage of REST is that many operations are cacheable
- a) True
 - b) False

Flash Quiz

- ③ Which of these choices would potentially be valid to retrieve a resource with an id of "1" and a type of "fruit"?
- a) GET www.store.com/api/food/1
 - b) GET www.store.com/api/food/fruit?id=1
 - c) GET www.store.com/api/food/fruit
 - d) POST www.store.com/api/food
 - e) All of the above are possible

Flash Quiz

- ③ Which of these choices would potentially be valid to retrieve a resource with an id of "1" and a type of "fruit"?
- a) GET www.store.com/api/food/1
 - b) GET www.store.com/api/food/fruit?id=1
 - c) GET www.store.com/api/food/fruit
 - d) POST www.store.com/api/food
 - e) All of the above are possible

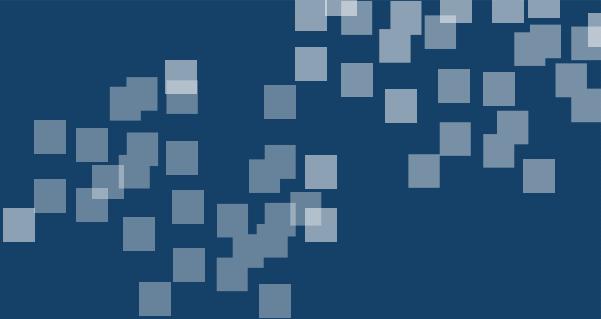


Keep in mind that while these are possible URLs to access the given resource, the actual allowed URL(s) are determined by the *service*

Summary

- ❖ Identify what REST services are
- ❖ Utilize URLs in REST
- ❖ Describe guidelines for using REST





Consuming REST services with Xamarin

Tasks

- ❖ Connect to a REST service
- ❖ Serialize data
- ❖ Send and receive data from a REST service



Working with REST services

- ❖ Xamarin applications have several API options when working with REST-based services



HttpClient

Most common
approach, built into
.NET

Working with REST services

- ❖ Xamarin applications have several API options when working with REST-based services



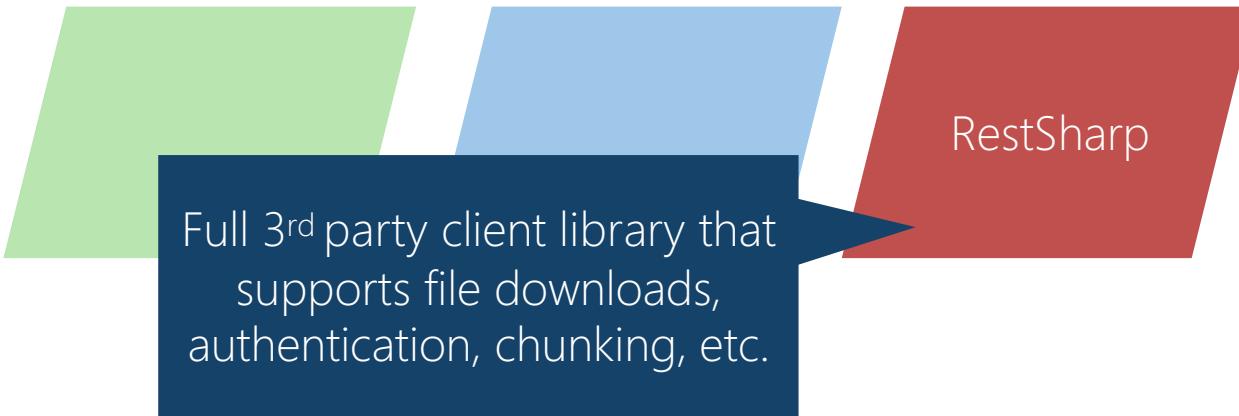
HttpClient

ServiceStack

Full fledged 3rd party web services framework, has client library for consuming REST services

Working with REST services

- ❖ Xamarin applications have several API options when working with REST-based services

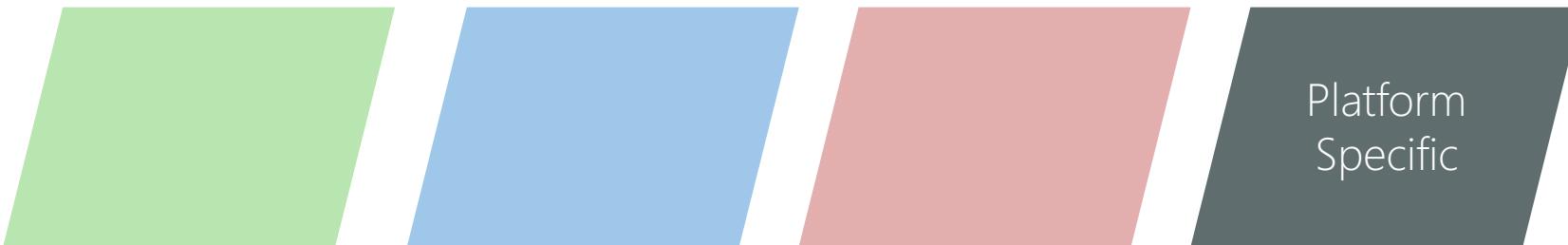


RestSharp

Full 3rd party client library that supports file downloads, authentication, chunking, etc.

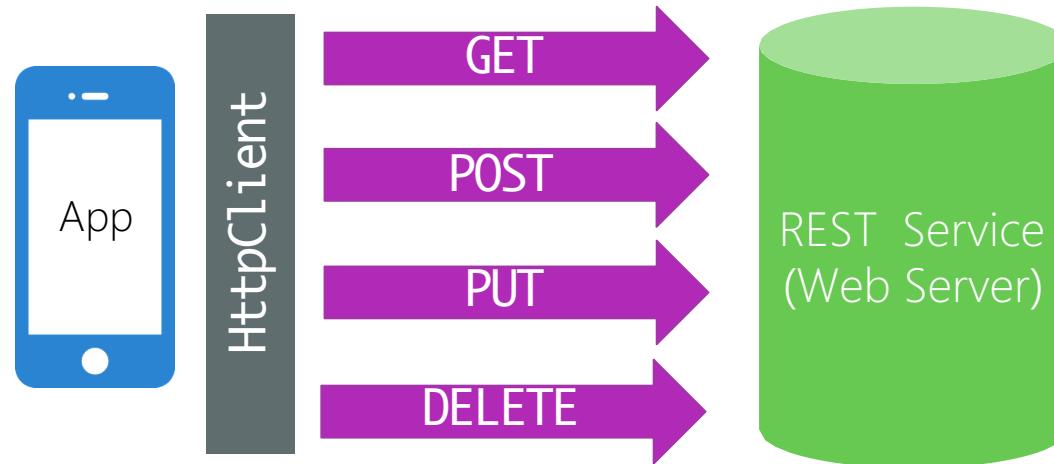
Working with REST services

- ❖ Xamarin applications have several API options when working with REST-based services



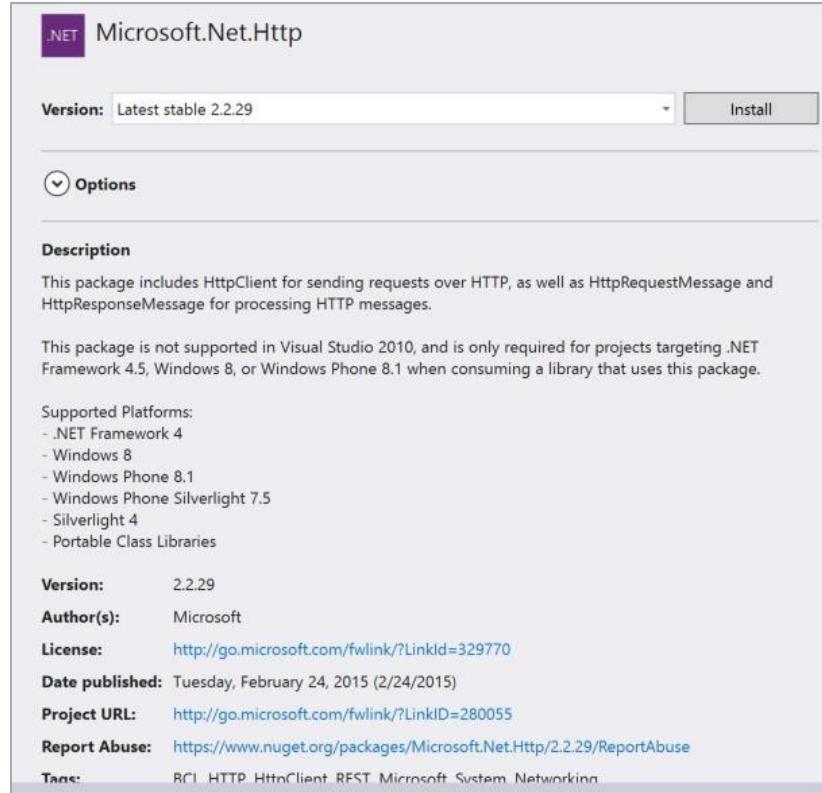
Introducing HttpClient

- ❖ Mobile apps can use `System.Net.Http.HttpClient` class to send basic requests and receive responses over HTTP



HttpClient in .NET Standard

- ❖ **HttpClient** is available in .NET, Android and iOS projects
- ❖ Not accessible in PCLs unless you add a NuGet package



The screenshot shows the NuGet package page for Microsoft.NuGet.Http. At the top, there's a purple ".NET" badge and the package name "Microsoft.NuGet.Http". Below that is a dropdown menu showing "Version: Latest stable 2.2.29" and a grey "Install" button. Underneath is a section titled "Options" with a dropdown arrow. The "Description" section contains text about the package's purpose and compatibility. It states: "This package includes HttpClient for sending requests over HTTP, as well as HttpRequestMessage and HttpResponseMessage for processing HTTP messages." and "This package is not supported in Visual Studio 2010, and is only required for projects targeting .NET Framework 4.5, Windows 8, or Windows Phone 8.1 when consuming a library that uses this package." The "Supported Platforms" section lists: ".NET Framework 4", "Windows 8", "Windows Phone 8.1", "Windows Phone Silverlight 7.5", "Silverlight 4", and "Portable Class Libraries". Below this are detailed package metadata fields: Version: 2.2.29, Author(s): Microsoft, License: <http://go.microsoft.com/fwlink/?LinkId=329770>, Date published: Tuesday, February 24, 2015 (2/24/2015), Project URL: <http://go.microsoft.com/fwlink/?LinkId=280055>, Report Abuse: <https://www.nuget.org/packages/Microsoft.NuGet.Http/2.2.29/ReportAbuse>, and Tags: PCL, HTTP, HttpClient, REST, Microsoft, System, Networking.

HttpClient async APIs

- ❖ **HttpClient** uses **Tasks** and asynchronous APIs to keep I/O operations from affecting the UI thread

```
public async Task<string> GetData()
{
    HttpClient client = new HttpClient();

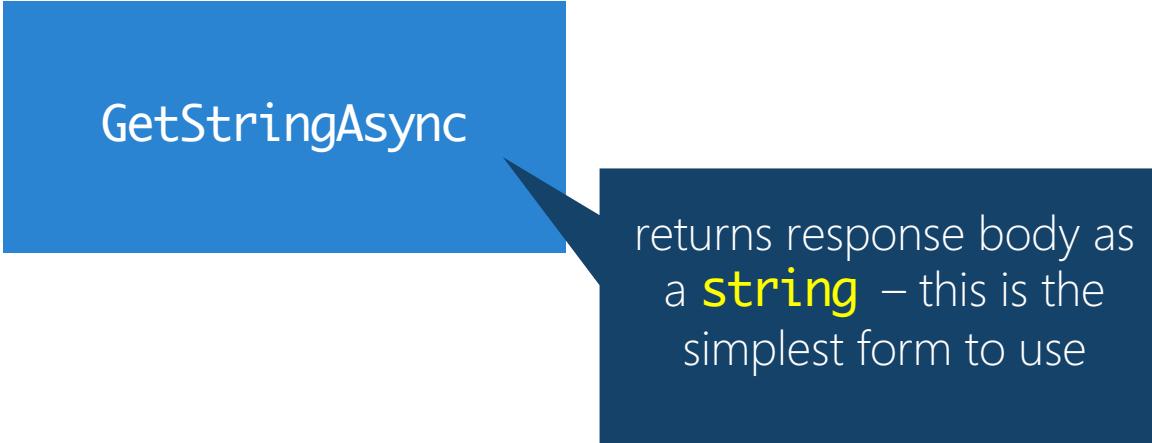
    return await client.GetStringAsync(
        "https://itunes.apple.com/search?term=comics");
}
```



Can use **async** / **await** keywords to easily work with APIs

How do I retrieve data with HttpClient?

- ❖ **HttpClient** supports several **Get** method styles to retrieve data

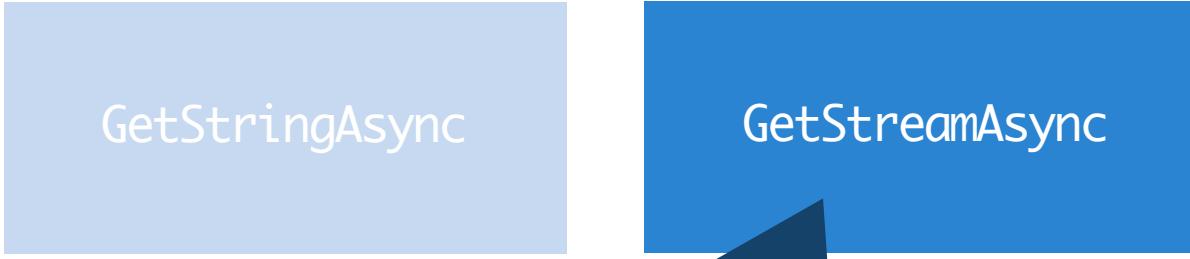


GetStringAsync

returns response body as
a **string** – this is the
simplest form to use

How do I retrieve data with HttpClient?

- ❖ **HttpClient** supports several **Get** method styles to retrieve data



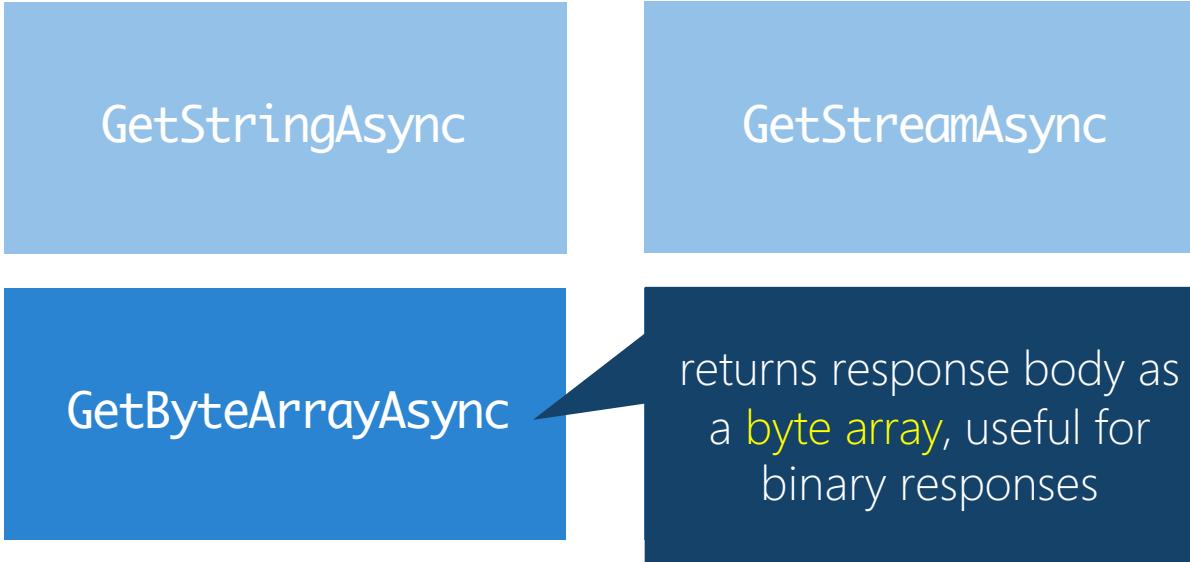
GetStringAsync

GetStreamAsync

returns response body as a
Stream, useful for large data
packets where you can perform
partial processing

How do I retrieve data with HttpClient?

- ❖ **HttpClient** supports several **Get** method styles to retrieve data



GetStringAsync

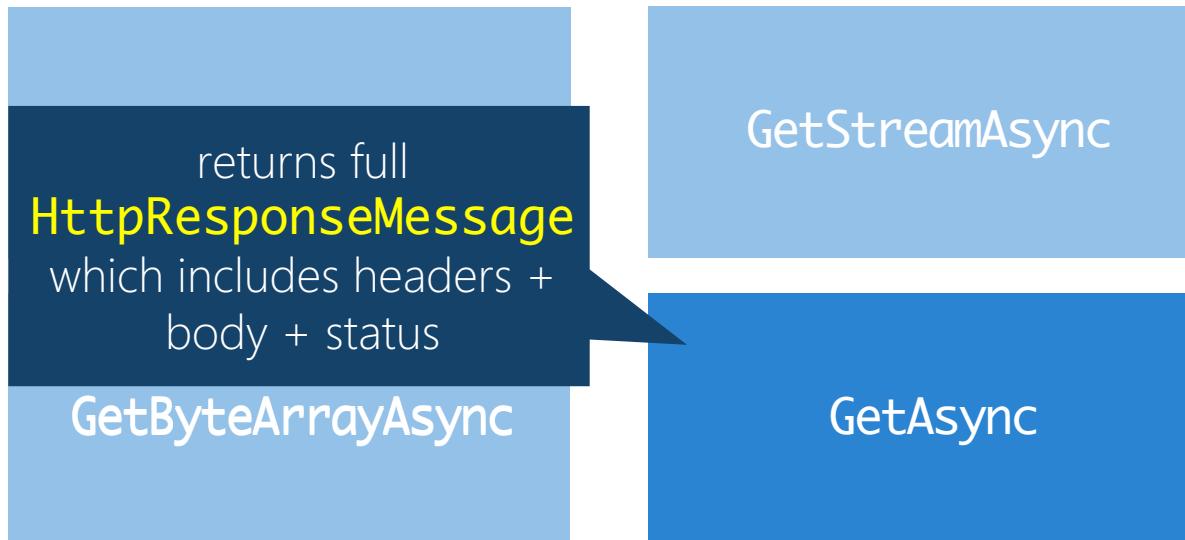
GetStreamAsync

GetByteArrayAsync

returns response body as
a **byte array**, useful for
binary responses

How do I retrieve data with HttpClient?

- ❖ **HttpClient** supports several **Get** method styles to retrieve data



returns full
HttpResponseMessage
which includes headers +
body + status

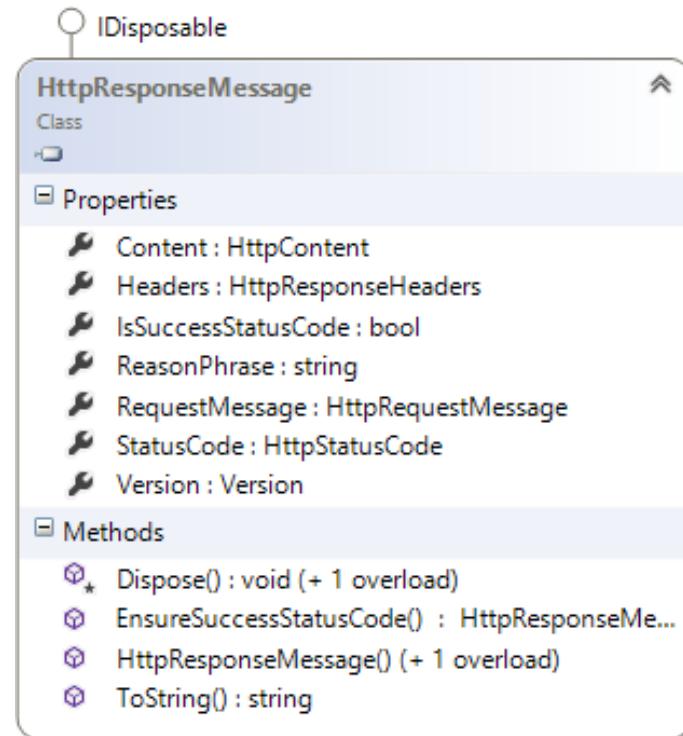
GetByteArrayAsync

GetStreamAsync

GetAsync

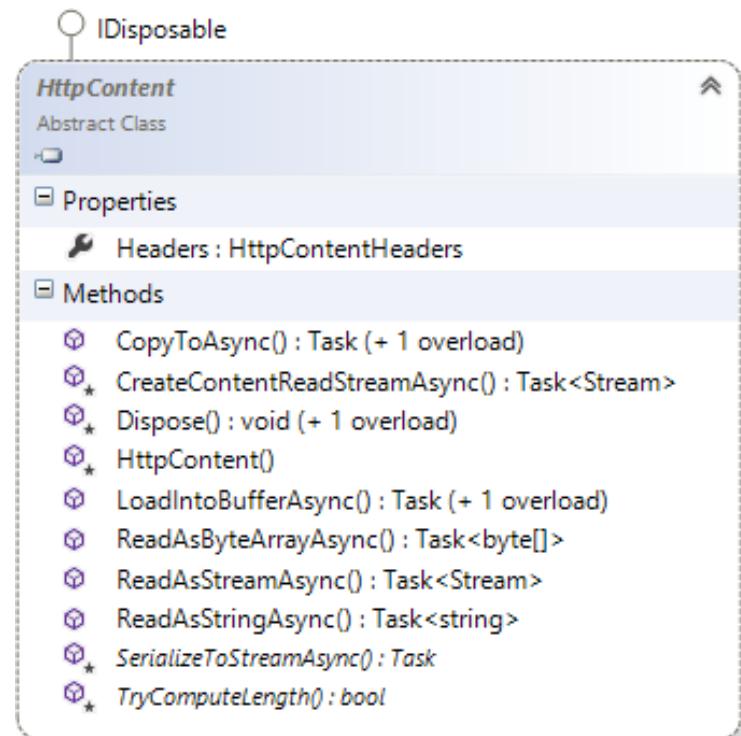
HttpResponseMessage

- ❖ `GetAsync` returns a full **response message** which contains information about the state of the request, the data result and error information
- ❖ Check **IsSuccessStatusCode** property to determine result and then either access **Content** or **StatusCodes**



HttpContent

- ❖ The actual data from the web service request is returned in the **Content** property in the form of an **HttpContent** class, this can also be used when *sending* data
- ❖ Can use **ReadAs** methods to pull data out in the form of a **string**, byte array or **Stream**

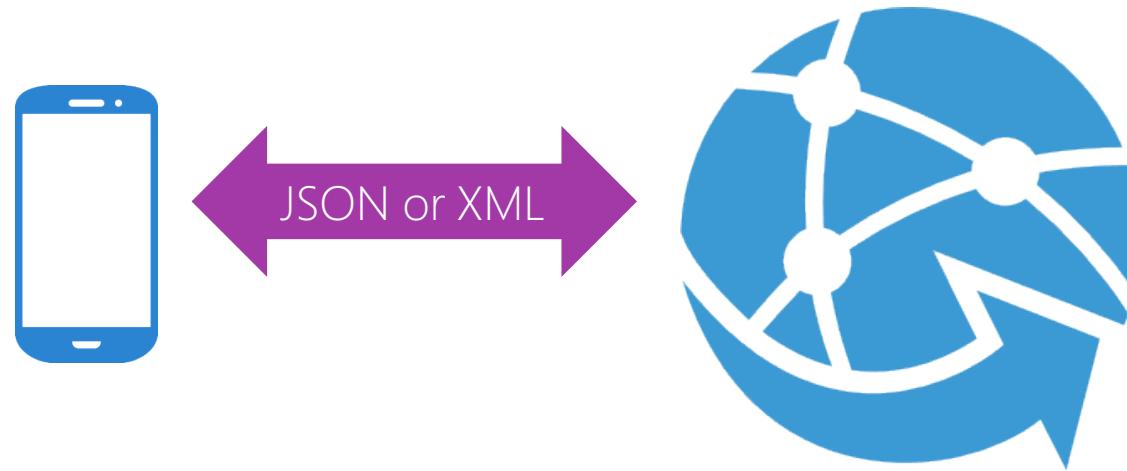


Data Serialization

- ❖ .NET objects must be turned into bytes in order to be sent or received from a network peer, this process is called *serialization*
- ❖ Serialization happens anytime we are communicating over a network, regardless of the technology being used to transfer information

Serialization Options

- ❖ REST services typically transfer data in either JSON or XML



 JSON has become the de-facto standard for RESTful services: most services either default to, or will respect the **Accept** header type and return JSON when requested

JSON

- ❖ JavaScript Object Notation is a very popular serialization format using name/value text pairs
 - ✓ Compact + easy to parse = fast
 - ✓ Flexible data representation
 - ✓ Widely supported, popular with client-side scripting

```
{ "contacts": [  
    {  
        "name": "Alice",  
        "email": "alice@contoso.com"  
    },  
    {  
        "name": "Bob",  
        "email": "bob@contoso.com"  
    },  
    {  
        "name": "Nigel",  
        "email": "nigel@contoso.com"  
    }  
]
```

Requesting JSON with HttpClient

- ❖ Most services either look at the **Accept** header, or take a URL parameter which indicates that JSON should be returned

```
HttpClient client = new HttpClient();
client.DefaultRequestHeaders.Accept.Add(
    new MediaTypeWithQualityHeaderValue(
        "application/json"));
```

...



Can request that service respond with JSON data

Parse and format data with JSON

- ❖ Applications typically choose a JSON library to work with, there are two very popular implementations commonly used

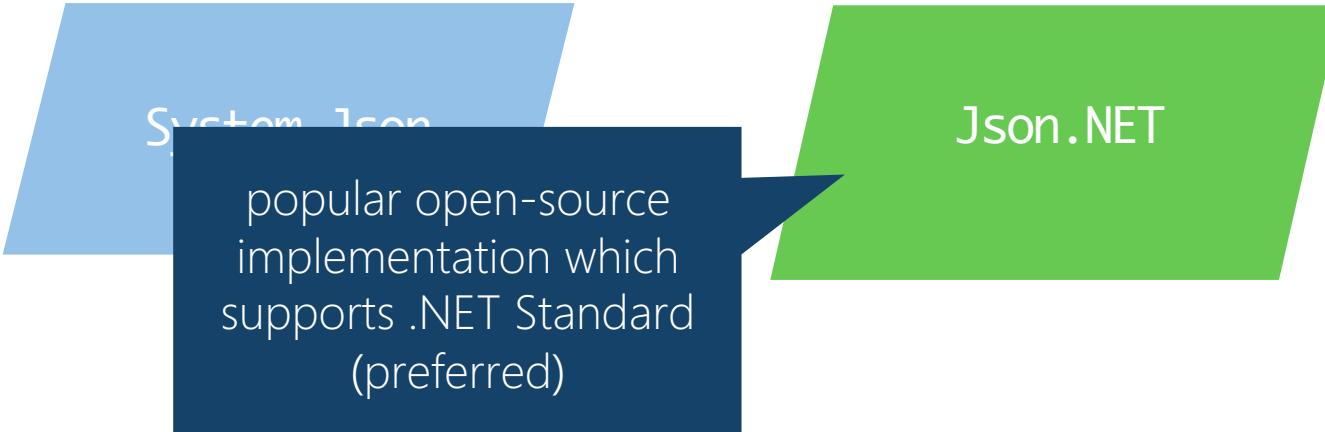


System.Json

part of .NET 4.5, supports
iOS + Android, but not
.NET Standard

Parse and format data with JSON

- ❖ Applications typically choose a JSON library to work with, there are two very popular implementations commonly used



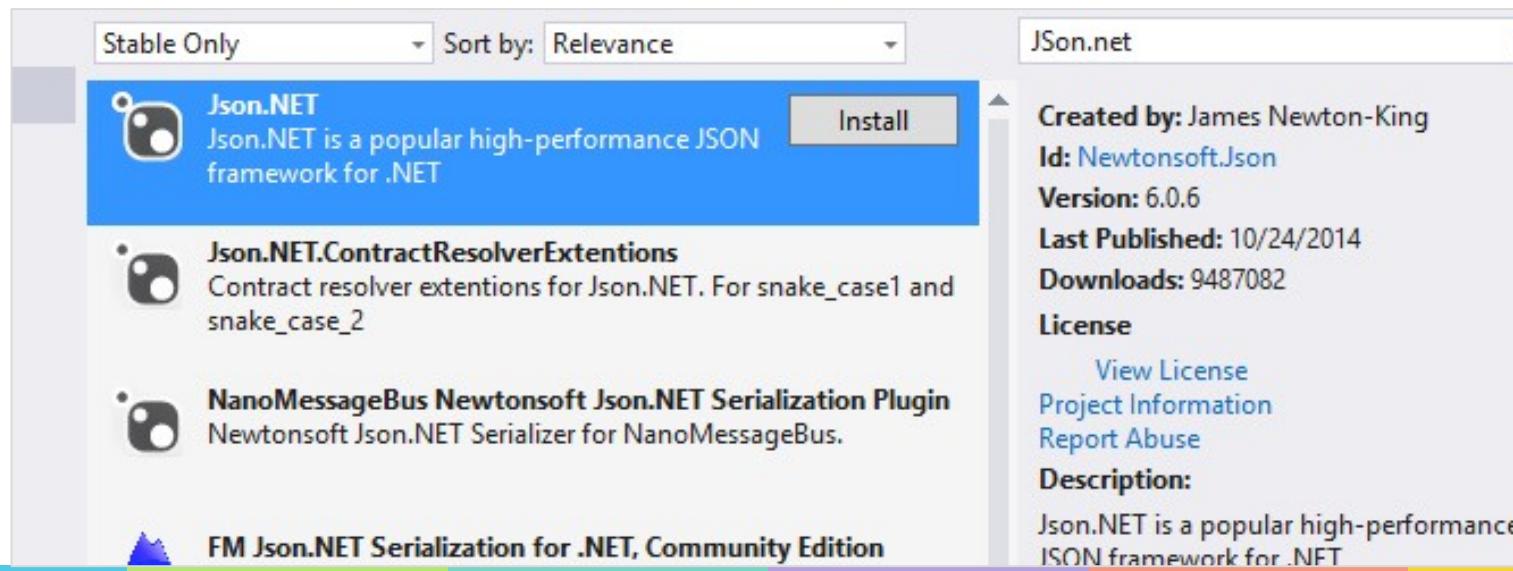
System.Text.Json

popular open-source
implementation which
supports .NET Standard
(preferred)

Json.NET

Adding support for Json.NET

- ❖ Json.NET is a 3rd party library available through Nuget, should add it to your platform-specific projects *and* your shared project(s)



The screenshot shows the NuGet package manager interface. At the top, there are filters for "Stable Only" and "Sort by: Relevance". A search bar contains the text "Json.net". Below the search bar, a list of packages is displayed:

- Json.NET**: Description: "Json.NET is a popular high-performance JSON framework for .NET". There is an "Install" button.
- Json.NET.ContractResolverExtentions**: Description: "Contract resolver extention for Json.NET. For snake_case1 and snake_case_2".
- NanoMessageBus Newtonsoft.Json.NET Serialization Plugin**: Description: "Newtonsoft.Json.NET Serializer for NanoMessageBus."
- FM Json.NET Serialization for .NET, Community Edition**: Description: "FM Json.NET is a popular high-performance JSON framework for .NET".

To the right of the list, detailed information about the top result is shown:

- Created by:** James Newton-King
- Id:** [Newtonsoft.Json](#)
- Version:** 6.0.6
- Last Published:** 10/24/2014
- Downloads:** 9487082
- License**
 - [View License](#)
 - [Project Information](#)
 - [Report Abuse](#)
- Description:** "Json.NET is a popular high-performance JSON framework for .NET"

Building objects with JSON

- ❖ JSON takes the network data and turns it into an object graph, but you must know the shape of the data and define the object to map it to

```
{ "contacts": [ ←  
  { ←  
    "name": "Alice",  
    "email": "alice@contoso.com"  
  },  
  { ←  
    "name": "Bob",  
    "email": "bob@contoso.com"  
  },  
  { ←  
    "name": "Nigel",  
    "email": "nigel@contoso.com"  
  },  
]  
}
```

The JSON data shown here is an array of contact elements, each with a name and email

To serialize or de-serialize this, we must define a set of objects which can be mapped to this data

Building objects with JSON

- ❖ JSON takes the network data and turns it into an object graph, but you must know the shape of the data and define the object to map it to

```
{ "contacts": [  
    {  
        "name": "Alice",  
        "email": "alice@contoso.com"  
    },  
    {  
        "name": "Bob",  
        "email": "bob@contoso.com"  
    },  
    {  
        "name": "Nigel",  
        "email": "nigel@contoso.com"  
    }  
]
```

```
public class Contact  
{  
    public string Name { get; set; }  
    public string Email { get; set; }  
}
```

Json.NET will map public properties by name + type, best to keep it simple and consider these as *data transfer objects* (DTOs)

Building objects with JSON

- ❖ JSON takes the network data and turns it into an object graph, but you must know the shape of the data and define the object to map it to

```
{ "contacts": [  
    {  
        "name": "Alice",  
        "email": "alice@contoso.com"  
    },  
    {  
        "name": "Bob",  
        "email": "bob@contoso.com"  
    },  
    {  
        "name": "Nigel",  
        "email": "nigel@contoso.com"  
    },  
]
```

```
public class Contact  
{  
    public string Name { get; set; }  
    public string Email { get; set; }  
}
```

```
public class ContactManager  
{  
    public List<Contact> Contacts {  
        get; set;  
    }  
}
```

 Can do this conversion manually, or use online tools such as <http://jsonutils.com> and <http://json2csharp.com>, there's even an IDE Add-in available: <http://bit.ly/json-addin>

Retrieve data from a REST service

- ❖ Use HTTP **GET** verb to retrieve data and use Json.NET to parse it out

```
HttpClient client = new HttpClient();
string text = await client.GetStringAsync("https://...");

ContactManager blackBook =
    JsonConvert.DeserializeObject<ContactManager>(text);
```

...



JsonConvert is a Json.NET class that can serialize and deserialize data from a JSON string or stream based on a specified **Type**

Modifying data with HttpClient

- ❖ Use **PostAsync**, **PutAsync** and **DeleteAsync** to modify resources

```
public async Task<Contact> Add(Contact c)
{
    HttpClient client = new HttpClient();
    StringContent content = new StringContent(
        JsonConvert.SerializeObject(c),
        Encoding.UTF8, "application/json");

    var response = await client.PutAsync("https://...", content);
    if (response.IsSuccessStatusCode) {
        return JsonConvert.DeserializeObject<Contact>(
            await response.Content.ReadAsStringAsync());
    }

    throw new Exception(response.ReasonPhrase);
}
```

Must serialize body
and include encoding
and content type

Modifying data with HttpClient

- ❖ Use **PostAsync**, **PutAsync** and **DeleteAsync** to modify resources

```
public async Task<Contact> Add(Contact c)
{
    HttpClient client = new HttpClient();
    StringContent content = new StringContent(
        JsonConvert.SerializeObject(c),
        Encoding.UTF8, "application/json");

    var response = await client.PutAsync("https://...", content);
    if (response.IsSuccessStatusCode) {
        return JsonConvert.DeserializeObject<Contact>(
            await response.Content.ReadAsStringAsync());
    }
    throw new Exception(response.ReasonPhrase);
}
```

Always use async
versions of APIs for
performance

Modifying data with HttpClient

- ❖ Use **PostAsync**, **PutAsync** and **DeleteAsync** to modify resources

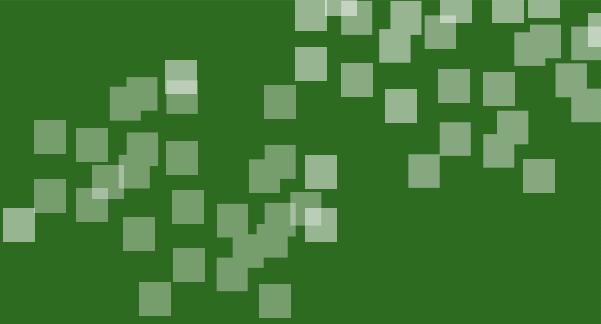
```
public async Task<Contact> Add(Contact c)
{
    HttpClient client = new HttpClient();

    StringContent content = new StringContent(
        JsonConvert.SerializeObject(c),
        Encoding.UTF8, "application/json");

    var response = await client.PutAsync("https://...", content);
    if (response.IsSuccessStatusCode) {
        return JsonConvert.DeserializeObject<Contact>(
            await response.Content.ReadAsStringAsync());
    }

    throw new Exception(response.ReasonPhrase);
}
```

Retrieve body from response on success and convert back into object, the response depends on the operation being performed – i.e. **DELETE** will just be a status code



Flash Quiz

Flash Quiz

- ① Which serialization format is generally more compact?
 - a) XML
 - b) JSON

Flash Quiz

- ① Which serialization format is generally more compact?
 - a) XML
 - b) JSON

Flash Quiz

- ② How do you inform a service that you prefer JSON-formatted data to be returned?
 - a) Add an **Accept** header to your request
 - b) Use a URL parameter
 - c) Either of the above, it depends on the service

Flash Quiz

- ② How do you inform a service that you prefer JSON-formatted data to be returned?
 - a) Add an **Accept** header to your request
 - b) Use a URL parameter
 - c) Either of the above, it depends on the service

Flash Quiz

- ③ When using **HttpClient** to interact with an HTTP service, which type gives you the Status Code of the result?
 - a) **HttpRequestMessage**
 - b) **HttpResponseMessage**
 - c) **HttpClient**

Flash Quiz

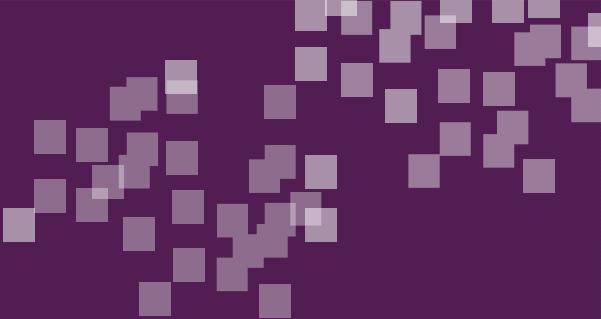
- ③ When using **HttpClient** to interact with an HTTP service, which type gives you the Status Code of the result?
 - a) **HttpRequestMessage**
 - b) **HttpResponseMessage**
 - c) **HttpClient**

Flash Quiz

- ④ **HttpClient** has convenience methods that make it easy to get which types of data from a service?
 - a) `int`, `float`, and `double`
 - b) `String` and `Object`
 - c) `String`, `Stream`, and `byte[]`

Flash Quiz

- ④ **HttpClient** has convenience methods that make it easy to get which types of data from a service?
 - a) `int`, `float`, and `double`
 - b) `String` and `Object`
 - c) `String`, `Stream`, and `byte[]`



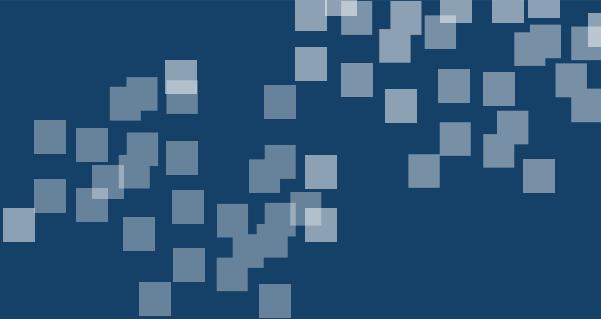
Individual Exercise

Communicating with a Book Service

Summary

- ❖ Connect to a REST service
- ❖ Serialize data
- ❖ Send and receive data from a REST service





Integrate with platform-specific network features

Tasks

- ❖ Customize the **HttpClient** handler
- ❖ Leverage platform network stacks
- ❖ Use App Transport Security on iOS



HttpClient customizations

- ❖ **HttpClient** can be passed a message handler with options to control how authentication, redirect, cookies, and other HTTP options are managed

```
var handler = new HttpClientHandler () {  
    AllowAutoRedirect = false,  
    UseProxy = true,  
    AutomaticDecompression = DecompressionMethods.GZip,  
    Credentials = new NetworkCredential("user", "passwd")  
};  
  
var client = new HttpClient (handler);
```

Using custom message handlers

- ❖ Can build delegating message handlers to pre/post process requests

```
public class MyTraceHandler : DelegatingHandler
{
    public MyTraceHandler() : this(new HttpClientHandler()) { }
    public MyTraceHandler(HttpMessageHandler inner) : base(inner) { }

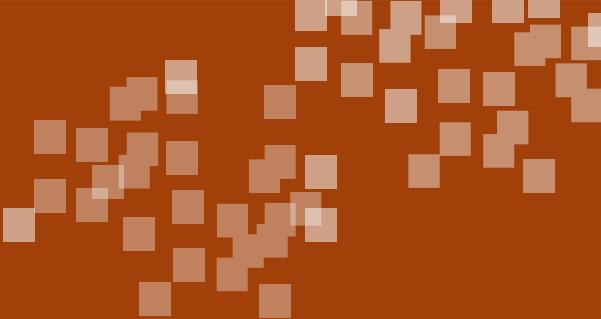
    protected override async Task<HttpResponseMessage> SendAsync(
        HttpRequestMessage request, CancellationToken cancellationToken)
    {
        Debug.WriteLine(">> {0}", request);
        var response = await base.SendAsync(request, cancellationToken);
        Debug.WriteLine("<< {0}", response);
        return response;
    }
}
```

Using custom message handlers

- ❖ Can build delegating message handlers to pre/post process requests

```
HttpClient client = new HttpClient (new MyTraceHandler());
string data = await client.GetStringAsync(
    "https://api.duckduckgo.com/?q=donald%20duck&format=json");
...
}
```

```
>> Method: GET, RequestUri: 'https://api.duckduckgo.com/?q=donald duck&format=json', Version: 1.1, Content: <null>, Headers: { }
<< StatusCode: 200, ReasonPhrase: 'OK', Version: 1.1, Content: System.Net.Http.StreamContent, Headers:
{
Server: nginx
Date: Wed, 04 May 2016 18:15:43 GMT
Connection: keep-alive
Cache-Control: max-age=1
Strict-Transport-Security: max-age=0
X-DuckDuckGo-Locale: en_US
Content-Type: application/x-javascript
Content-Length: 6286
Expires: Wed, 04 May 2016 18:15:44 GMT
}{"DefinitionSource": "", "Heading": "Donald Duck", "ImageWidth": 0, "RelatedTopics": [{"Result": "..."}]}
```



Demonstration

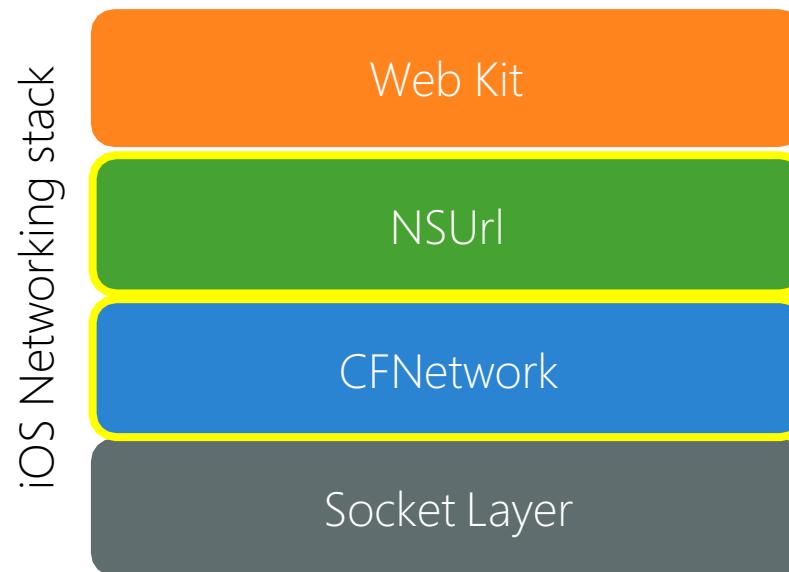
Using a custom Http message handler

Issues with HttpClient

- ❖ **HttpClient** uses **HttpWebRequest** under the covers which is a managed networking stack sitting on a socket layer
- ❖ Android and iOS both have native networking stacks which are more efficient, but have unique APIs and are harder to use from C#

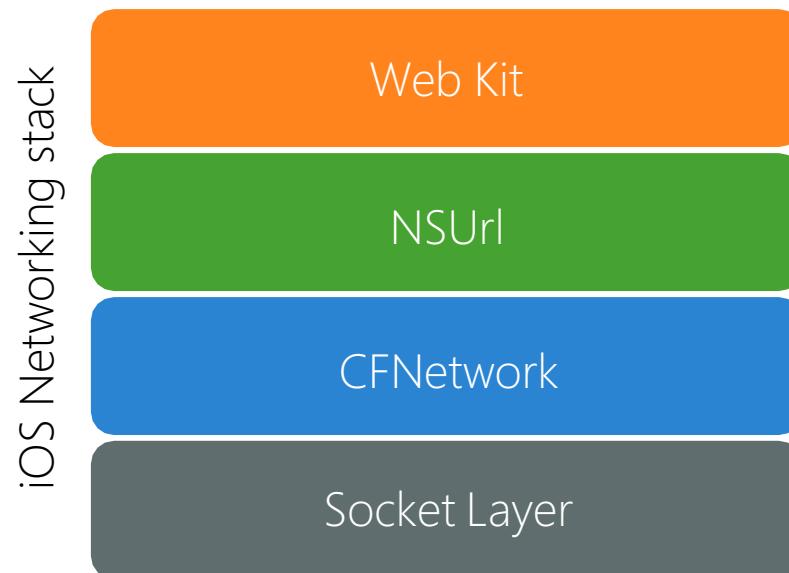
Customize HttpClient for iOS

- ❖ Xamarin.iOS includes two specialized message handlers to allow you to integrate more deeply with the iOS networking stack



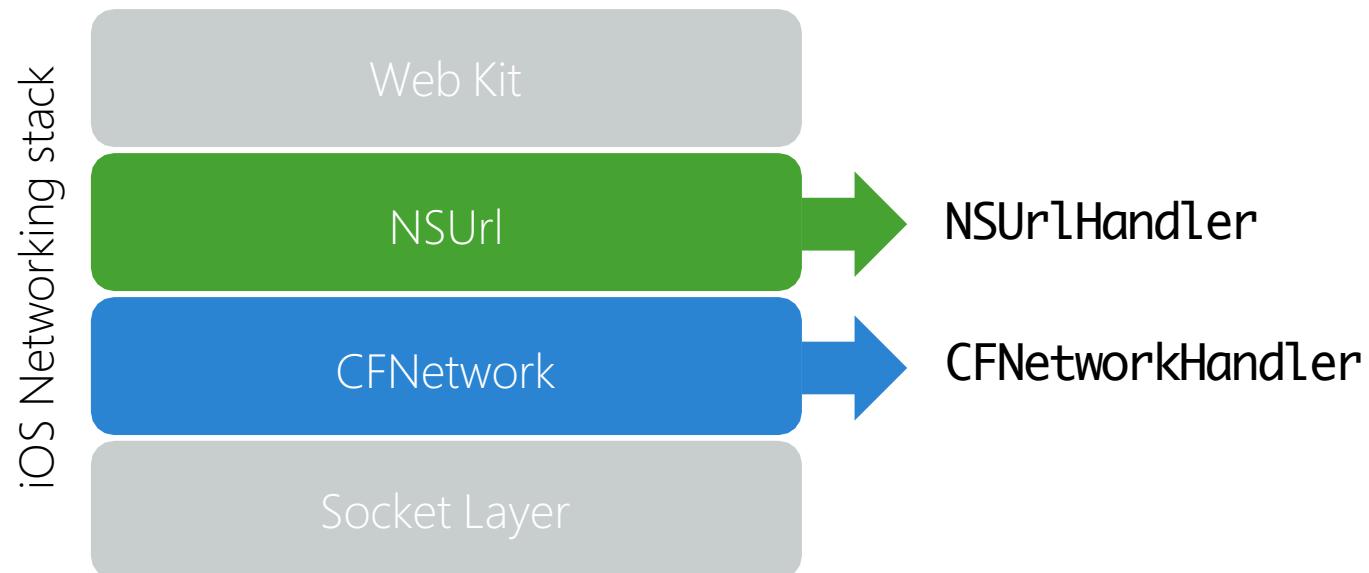
iOS native networking stack

- ❖ iOS supplies a native networking stack that makes it convenient to do networking on iOS (e.g. automatically turns on the networking radio)



Customize HttpClient for iOS

- ❖ Xamarin.iOS includes two specialized message handlers to allow you to integrate more deeply with the iOS networking stack



Using CFNetworkHandler

- ❖ Xamarin.iOS includes **CFNetworkHandler** which integrates **HttpClient** with the **CFNetwork** stack

```
var client = new HttpClient (new CFNetworkHandler());
```

- ✓ automatically turns the radio on before starting the request
- ✓ utilizes iOS connection pooling
- ✓ automatically applies iOS proxy and network settings
- ✓ uses dispatch queues instead of managed threads
- ✗ requires iOS6+
- ✗ platform-specific

Using NSUrlSessionHandler

- ❖ Xamarin.iOS includes **NSUrlSessionHandler** which integrates **HttpClient** with the **NSURL** stack

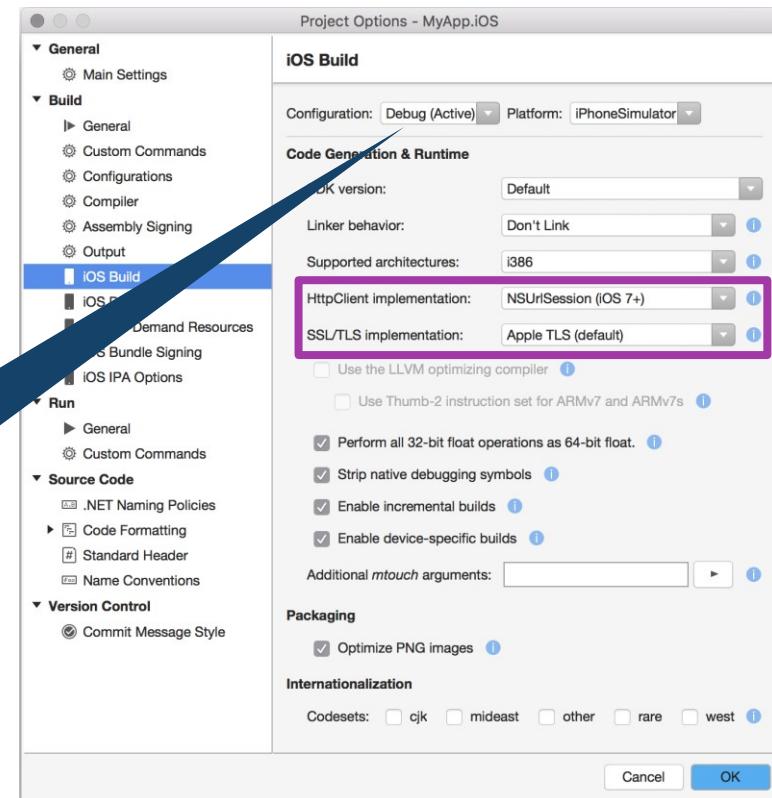
```
var client = new HttpClient (new NSUrlSessionHandler());
```

- ✓ does everything **CFCNetworkHandler** does
- ✓ big performance boost for TLS + app size is reduced!
- ✗ requires iOS7+
- ✗ platform-specific
- ✗ not all **HttpClient** features are supported

iOS Native in project settings

- ❖ Visual Studio allows you to select a networking stack and TLS implementation in the iOS project properties – this allows you to use the default **HttpClient** constructor in .NET Standard

The setting is per configuration,
Debug is shown here



Android Native in code

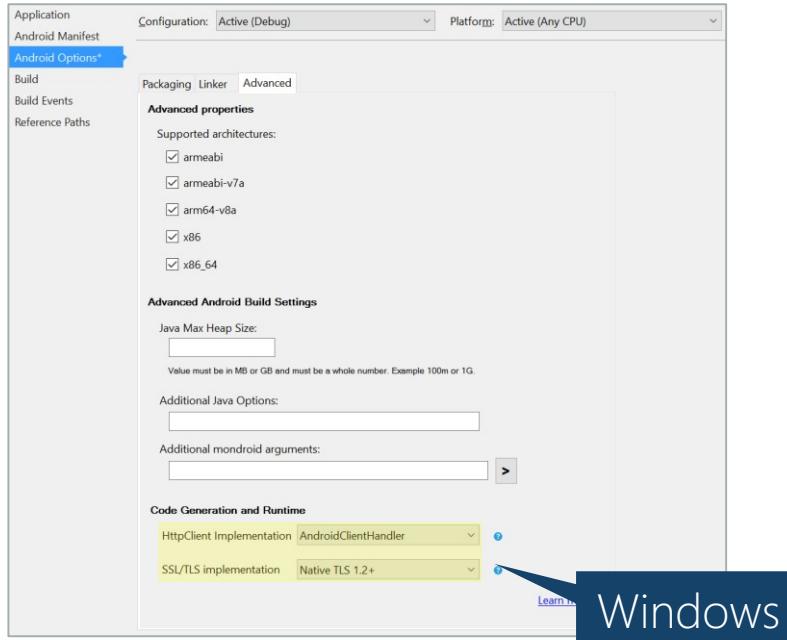
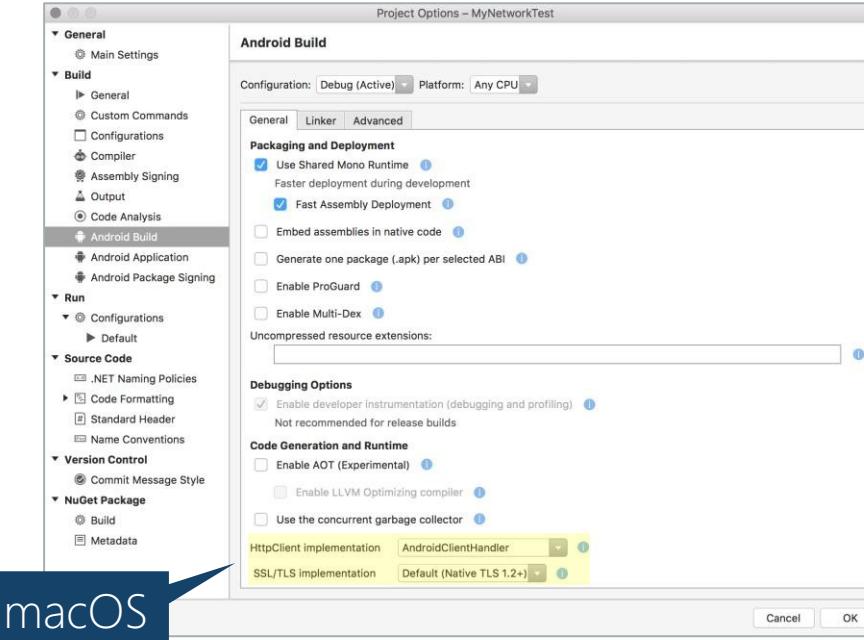
- ❖ Xamarin.Android includes **AndroidClientHandler** which integrates **HttpClient** with the **URLConnection** stack

```
var client = new HttpClient (new AndroidClientHandler());
```

- ✓ supports TLS 1.2 (in Android 5.0+ and where the device does)
- ✓ more work is delegated to hardware
- ✓ app can work with any protocols that Android understands
- ✗ platform-specific
- ✗ not all **HttpClient** features are supported

Android Native in project settings

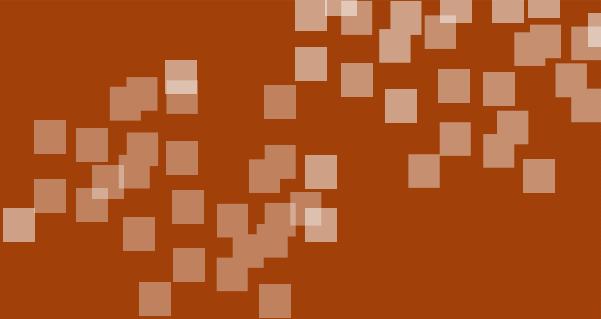
- ❖ Project options allow you set the Android HTTP client handler, this lets you use the default **HttpClient** constructor .NET Standard



The image shows two side-by-side screenshots of the 'Project Options' dialog from the Xamarin IDE, one for macOS and one for Windows.

macOS Screenshot: The 'Android Build' tab is selected. Under 'HttpClient implementation', the dropdown menu is open, showing 'AndroidClientHandler' as the selected option. Other options like 'Default (Native TLS 1.2+)' are also visible.

Windows Screenshot: The 'Android Options*' tab is selected. In the 'Code Generation and Runtime' section, the 'HttpClient Implementation' dropdown is set to 'AndroidClientHandler'. The 'SSL/TLS implementation' dropdown is set to 'Native TLS 1.2+'.



Demonstration

Leveraging the native platform network stack

App Transport Security

- ❖ iOS security policy enforces requirements on network connections
 - ✓ Requires TLS 1.2 or better (https)
 - ✓ Must use a modern key exchange algorithm that provides forward secrecy
 - ✓ Certificates must be signed with SHA256, 2048-bit RSA key, or better



App Transport Security

- ❖ iOS security policy enforces requirements on network connections

If your application is currently using https and good certificates, then this change will likely not affect you

2048-bit RSA key, or better



What APIs does this affect?

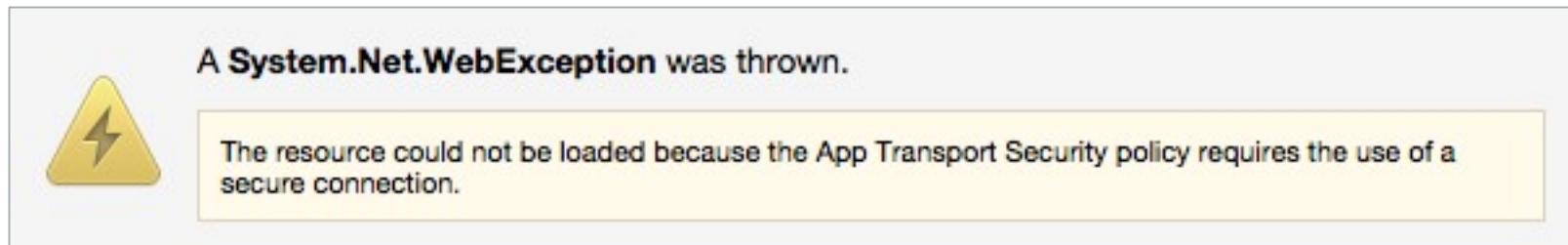
- ❖ ATS secures the native iOS stack:
 - **NSURLSession/Connection**
 - Embedded web views
 - Background transfers
 - ModernHttpClient (Nuget)

- ❖ Test edge areas of your app that perform network access such as ad-revenue, in-app OAuth logins, social media integration, etc.

```
backgroundTransfer {  
    class SimpleBackgroundTransferViewC  
        Identifier = "com.SimpleBackgroundTransferViewC  
        DownloadUrlString = "https://atmi  
        NSURLConnection downloadTask;  
        NSURLConnection session;  
        SimpleBackgroundTransferViewController  
  
    public override void ViewDidLoad ()  
    {  
        base.ViewDidLoad ();  
        if (session == null)  
            session = InitBackgroundSession ();  
        // Perform any additional setup after lo  
        progressView.Progress = 0;  
        imageView.Hidden = false;  
        progressView.Hidden
```

Detecting ATS problems

- ❖ ATS policy violations result in an exception, most common cause is connection to a non-TLS endpoint



Adding exceptions for ATS

- ❖ Must add *exceptions* into info.plist if your app cannot comply to restrictions – use new **NSAppTransportSecurity** key

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSExceptionDomains</key>
  <dict>
    <key>xam150.azurewebsites.net</key>
    <dict>
      <!-- specific options here -->
    </dict>
  </dict>
</dict>
```

Try to identify the specific endpoints your app uses and configure just those endpoints

Exclusion options

```
<key>xam150.azurewebsites.net</key>
<dict>
    <key>NSExceptionMinimumTLSVersion</key>
    <string>TLSv1.0</string>
    <key>NSExceptionRequiresForwardSecrecy</key>
    <false/>
    <key>NSExceptionAllowsInsecureHTTPLoads</key>
    <true/>
    <key>NSIncludesSubdomains</key>
    <true/>
</dict>
```

Options
expressed as
key/value pairs



Full description of **NSAppTransportSecurity** options are in Apple technical note referred to in [StartHere.html](#), check it out for details

Exclusion options

```
<key>xam150.azurewebsites.net</key>
<dict>
    <key>NSEExceptionMinimumTLSVersion</key>
    <string>TLSv1.0</string>
    <key>NSEExceptionRequiresForwardSecrecy</key>
    <false/>
    <key>NSEceptionAllowsInsecureHTTPLoads</key>
    <true/>
    <key>NSIncludesSubdomains</key>
    <true/>
</dict>
```

Minimum version
of TLS to allow

Exclusion options

```
<key>xam150.azurewebsites.net</key>
<dict>
    <key>NSEExceptionMinimumTLSVersion</key>
    <string>TLSv1.0</string>
    <key>NSEExceptionRequiresForwardSecrecy</key>
    <false/>
    <key>NSEExceptionAllowsInsecureHTTPRedirects</key>
    <true/>
    <key>NSIncludesSubdomains</key>
    <true/>
</dict>
```



Do not require
Forward Secrecy

Exclusion options

```
<key>xam150.azurewebsites.net</key>
<dict>
    <key>NSEExceptionMinimumTLSVersion</key>
    <string>TLSv1.0</string>
    <key>NSEExceptionRequiresForwardSecrecy</key>
    <false/>
    <key>NSEExceptionAllowsInsecureHTTPLoads</key>
    <true/>
    <key>NSIncludesSubdomains</key>
    <true/>
</dict>
```



Allow non-https
data transfer

Exclusion options

```
<key>xam150.azurewebsites.net</key>
<dict>
    <key>NSEExceptionMinimumTLSVersion</key>
    <string>TLSv1.0</string>
    <key>NSEExceptionRequiresForwardSecrecy</key>
    <false/>
    <key>NSEExceptionAllowsInsecureHTTPLoads</key>
    <true/>
    <key>NSIncludesSubdomains</key>
    <true/>
</dict>
```

Include subdomains of the listed top-level domain

Turn off ATS by default

- ❖ Can also disable App Transport Security for all unspecified URLs, allows arbitrary data access when the endpoint is unknown

```
<!-- Turn off ATS in iOS9 -->
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```



Should then turn ATS back on for known endpoints by including specific URL endpoint definitions with this key set to false

Whitelisting URLs

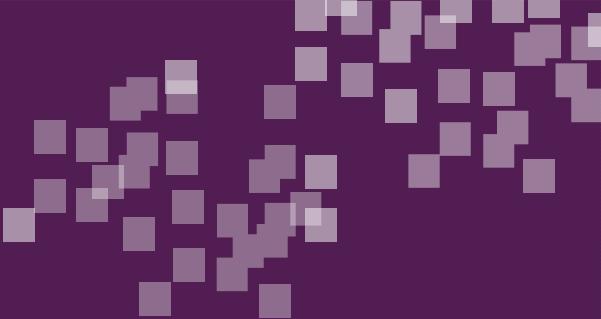
- ❖ `UIApplication.SharedApplication.CanOpenUrl` can now only check for specific URL schemes listed in `info.plist`, all unlisted schemes always return `false` even if the associated app is installed

```
<key>LSApplicationQueriesSchemes</key>
<array>
    <string>fbapi</string>
    <string>fb-messenger-api</string>
    <string>fbauth2</string>
    <string>fbshareextension</string>
</array>
```

Support Facebook URLs for login, share, etc.



This change does not impact system-provided URLs such as http:, https:, tel:, etc.



Homework

Add an exclusion for ATS on iOS

Summary

- ❖ Customize the **HttpClient** handler
- ❖ Leverage platform network stacks
- ❖ Use App Transport Security on iOS



Thank You!