XAM150

Consuming RESTful Web Services

- Lecture will begin shortly
- Download class materials from university.xamarin.com

Xamarin University

Information in this document is subject to change without notice. The example companies, organizations, products, people, and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user.

Xamarin may have patents, patent applications, trademarked, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any license agreement from Xamarin, the furnishing of this document does not give you any license to these patents, trademarks, or other intellectual property.

© 2016 Xamarin. All rights reserved.

Xamarin, MonoTouch, MonoDroid, Xamarin.iOS, Xamarin.Android, and Xamarin Studio are either registered trademarks or trademarks of Xamarin in the U.S.A. and/or other countries.

Other product and company names herein may be the trademarks of their respective owners.

Xamarin University

Objectives

1. Obtain the device's network capabilities
2. Introduction to REST
3. Consuming REST services with Xamarin
4. Integrating with the platform

Obtain the device's network capabilities

Xamarin University

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

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

Xamarin University

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 Phone 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 Phone

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;
```

Cross Platform network detection

- Open source **Connectivity Plugin** includes PCL support with implementations for Windows (Phone), 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
```

github.com/jamesmontemagno/Xamarin.Plugins

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

Xamarin University

Flash Quiz

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

a) True

b) False

Xamarin University

Flash Quiz

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

a) True

b) False

Xamarin University

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

Xamarin University

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 Reachability event

c) <u>You cannot detect roaming conditions on iOS</u>

Xamarin University

Flash Quiz

③ You can obtain network information about an Android device using the method call:

a) Android.GetNetworkInformation

b) ConnectivityManager.ActiveNetworkInfo

c) ConnectivityManager.GetNetworkType

Xamarin University

Flash Quiz

③ You can obtain network information about an Android device using the method call:

a) Android.GetNetworkInformation

b) ConnectivityManager.ActiveNetworkInfo

c) ConnectivityManager.GetNetworkType

Individual Exercise

Determine the network connectivity

Xamarin University

Summary

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

Introduction to REST

Xamarin University

Xamarin University

Tasks

- Identify what REST services are
- 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

Internet APIs, Data: Programmable Web, Feb 10, 2012

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

There are other verbs defined by the HTTP specification which can be used as well, but they are far less common – check the specification for more information

URLs + Operations

- URLs are used to identify and organize accessible resources

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

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

or

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

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

or

```
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

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

or

```
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

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

or

```
HTTP/1.1 200 OK
...
<customer>
   <id>12345</id>
   <name>Barry Allen</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 – check out **ENT171** and **ENT172** for more information

Flash Quiz

Xamarin University

Xamarin University

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*

Xamarin University

Summary

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

Consuming REST services with Xamarin

Xamarin University

Tasks

- Connecting to a REST service
- Serializing data
- Performing basic operations

Working with REST services

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

Working with REST services

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

Working with REST services

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

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 PCLs

- **`HttpClient`** is supported in .NET, Android and iOS applications natively

- Not accessible in PCLs or Windows Phone unless you add a **Nuget package**

HttpClient async APIs

- **`HttpClient`** uses **`Task`**s 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=batman");
}
```

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

How do I retrieve data with HttpClient?

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

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 `StatusCode`

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": "Clark Kent",
    "email": "mmguy@planet.com"
  },
  {
    "name": "Bruce Wayne",
    "email": "ceo@wayne.net"
  },
  {
    "name": "Tony Stark",
    "email": "playboy@stark.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

Parse and format data with JSON

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

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)

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": "Clark Kent",
    "email": "mmguy@planet.com"
  },
  {
    "name": "Bruce Wayne",
    "email": "ceo@wayne.net"
  },
  {
    "name": "Tony Stark",
    "email": "playboy@stark.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": "Clark Kent",
    "email": "mmguy@planet.com"
  },
  {
    "name": "Bruce Wayne",
    "email": "ceo@wayne.net"
  },
  {
    "name": "Tony Stark",
    "email": "playboy@stark.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": "Clark Kent",
    "email": "mmguy@planet.com"
  },
  {
    "name": "Bruce Wayne",
    "email": "ceo@wayne.net"
  },
  {
    "name": "Tony Stark",
    "email": "playboy@stark.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.Deserialize<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

Xamarin University

Xamarin University

Flash Quiz

① REST is an architectural style based on the architecture of _____

a) HTTP and XML
b) TCP/IP networking
c) The World Wide Web (WWW)
d) None of the above

Xamarin University

Flash Quiz

① REST is an architectural style based on the architecture of _____

a) HTTP and XML

b) TCP/IP networking

c) The World Wide Web (WWW)

d) None of the above

Xamarin University

Flash Quiz

② One way to send HTTP requests to a REST based web service is _____

a) HttpContent
b) HttpWebClient
c) HttpClient
d) HttpRequest

Xamarin University

Flash Quiz

② One way to send HTTP requests to a REST based web service is _____

a) HttpContent

b) HttpWebClient

c) HttpClient

d) HttpRequest

Individual Exercise

Communicating with a Book Service

Xamarin University

Xamarin University

Summary

- Connecting to a REST service
- Serializing data
- Performing basic operations

Integrating with the platform

Xamarin University

Tasks

- Customize the `HttpClient` handler
- Integrating ModernHttpClient
- Changes in iOS9

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);
```

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 harder to use

Customize HttpClient for iOS

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

Customize HttpClient for iOS

- Xamarin.iOS includes a message handler 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
- X requires iOS6+
- X platform-specific

Customize HttpClient for iOS

- Xamarin.iOS also includes a message handler which integrates **`HttpClient`** with the **`NSUrl`** stack

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

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

Customize HttpClient for Android

- Xamarin.iOS also includes a message handler which integrates **`HttpClient`** with the **`UrlConnection`** stack

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

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

Xamarin University

Optimizing HttpClient cross-platform

ModernHttpClient is a 3rd party, open-source library of message handlers for `HttpClient` which supports the native iOS (`NSUrlSession`) and Android (`OkHttp`) network stacks

https://github.com/paulcbetts/ModernHttpClient

Using ModernHttpClient is easy

- Add **ModernHttpClient** package using Nuget, and use pass **`NativeMessageHandler`** to **`HttpClient`** constructor and your code gets an immediate performance improvement

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

This class comes from **ModernHttpClient** and has a different implementation for each platform; when your code executes, it will get a platform-specific version

Demonstration

Adding ModernHttpClient to our app

Xamarin University

Security changes in iOS9

- Apple has made some security and privacy changes to iOS9 which will affect **any application linked to the latest SDK**

- Even if your application does not *use* iOS9 APIs, if you link to the latest SDK, these changes will apply to your app!

App Transport Security

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

App Transport Security

- New security policy enforces tighter requirements on network connections
  - Requires TLS 1.2 or better
    - Must use a modern key exchange algorithm that provides forward security
    - Server certificate must be signed with SHA256, 2048-bit RSA key, or better

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

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.

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

Minimum version of TLS to allow

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>
```

Do not require Forward Secrecy

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>
```

Allow non-https data transfer

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>
```

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**

CanOpenUrl whitelisting

- `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.

Group Exercise

Add ModernHttpClient and turn off ATS

Xamarin University

Summary

- Customize the `HttpClient` handler
- Use the `ModernHttpClient`

Thank You!

Please complete the class survey in your profile:
university.xamarin.com/profile