Microsoft Cognitive Services を活用した
顔分析アプリ(Xamarin / UWP)の構築

概要

今日、AI (artificial intelligence: 人工知能) という言葉を聞かない日はないほど注目を浴びており、人間、またはそれ以上の知能 (の一部) をコンピューター上で実現することが可能になりました。

Microsoft Cognitive Services は 画像、文章、言語、情報を処理する機能を API 経由で利用できる “人工知能パーツ” とも言えるサービスです。ビックデータを投入、統計解析手法を用いて、人工知能として機能する分析モデルを作成する、という作業をご自身で行うことなしに、データを投入するだけで分析結果を取得できるようになりました。

このラボでは、まずは Cognitive Services を利用したアプリケーションに触れ、Cognitive Services の利用方法について学習します。その後、Cognitive Services Face API を活用して、取り込んだ画像を分析して人間の顔に関する情報 (年齢、性別、表情) を出力するアプリを、マルチプラットフォーム向け開発環境である Xamarin を用いたユニバーサル Windows アプリ (UWP) として構築します。

目的

このハンズオン ラボでは、以下の方法について学習します。

- Cognitive Services を利用したアプリケーションを操作し、Cognitive Services が提供する機能について理解する。
- Cognitive Services Face API を利用して、画像に含まれる人物の顔情報を取得し、表示する機能を Xamarin / UWP アプリとして実装する

前提条件

このハンズオン ラボを実行するには、以下が必要です。

- マイクロソフトアカウント
- Visual Studio 2017 Community エディションまたはこれ以降

目次

このラボの推定所要時間: 45 分

演習 1: Cognitive Services の各 API の動作を確認する

この演習では Cognitive Services を利用したデモサイトに触れ、Cognitive Services の各 API の動作を確認します。

Cognitive Services の動作をデモサイトで確認する

Cognitive Services の各 API 詳細画面には、デモが用意されており、サブスクリプション申込なしで動作を確認できます。

1. ブラウザーで Cognitive Services の Web サイト (http://microsoft.com/cognitive) を開きます。こちらのサイトでは、Cognitive Services の 5 つの API グループ別 (視覚、音声、知識、言語、検索) に説明が書かれています。

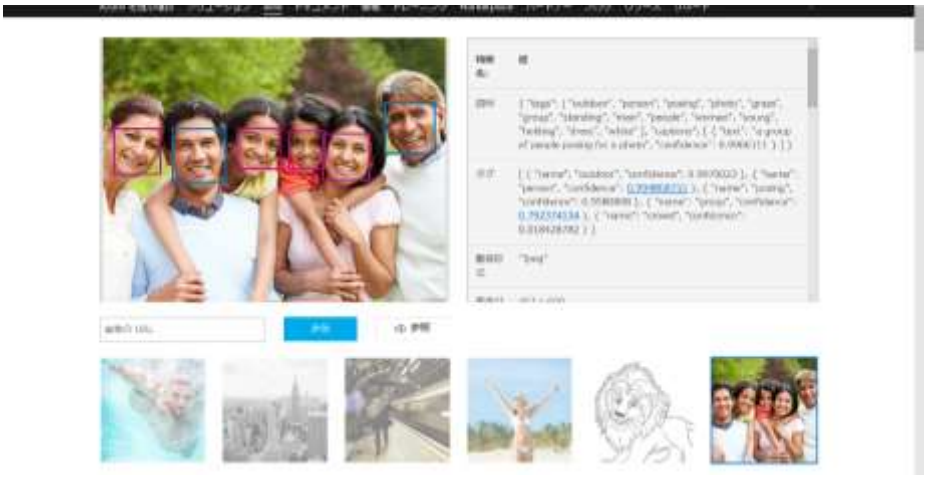
Computer Vision API の動作確認

2. **[視覚]** をクリックします。[視覚] タブ内には、Computer Vision API など 画像・動画の分析をおこなう API が表示されています。

3. **Computer Vision API** をクリックして開きます。Computer Vision API の下記の機能およびデモが用意されています。

   - 画像の分析によるタグ付け、キャプション生成、顔検出、カラー抽出
   - OCR (画像からの文字認識)
   - 著名人およびランドマークの検出
   - サムネイル作成

4. 画像の分析 デモサイトで、画像をクリック、またはお手持ちの画像をアップロードして、分析結果を確認してください。



5. ブラウザーのバック (←) ボタンでひとつ前のページに戻ります。

Face API の動作確認

6. 今度は **Face API** をクリックして開きます。Face API の下記の機能およびデモが用意されています。

   - 顔検出 : 画像内の顔の位置、顔属性 (年齢、性別、感情、眼鏡や髭、髪の色など)
   - 顔識別、似た顔の検索、グループ化

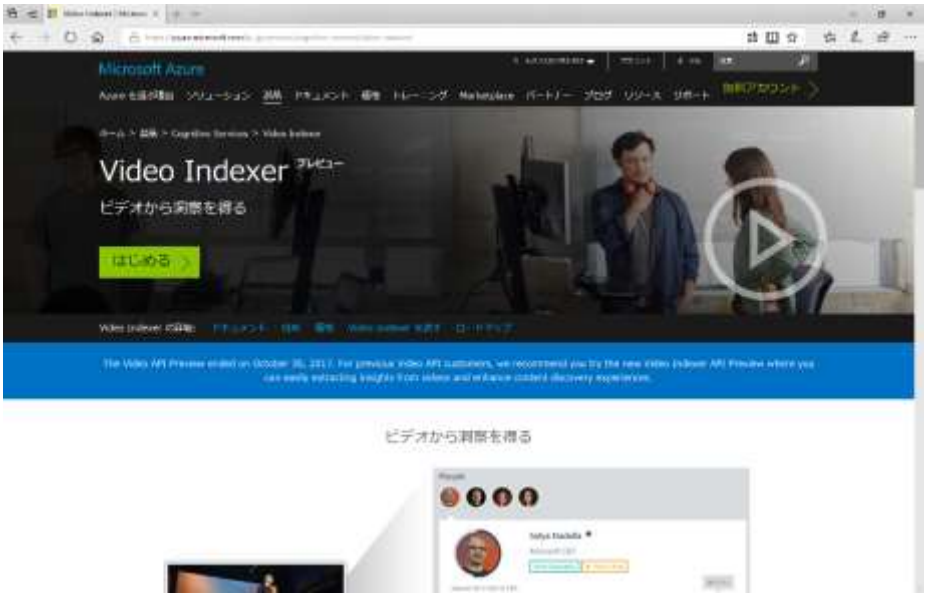
7. 画面をスクロールします。**顔検出** のデモサイトで、画像をクリック、またはお手持ちの画像をアップロードして、分析結果を確認してください。

画像の分析結果は、画像の右枠に表示されるように JSON 形式で返されます。他の API でも同様に、決まった形式の JSON で結果を取得できます。

8. ブラウザーのバック (←) ボタンでひとつ前のページに戻ります。

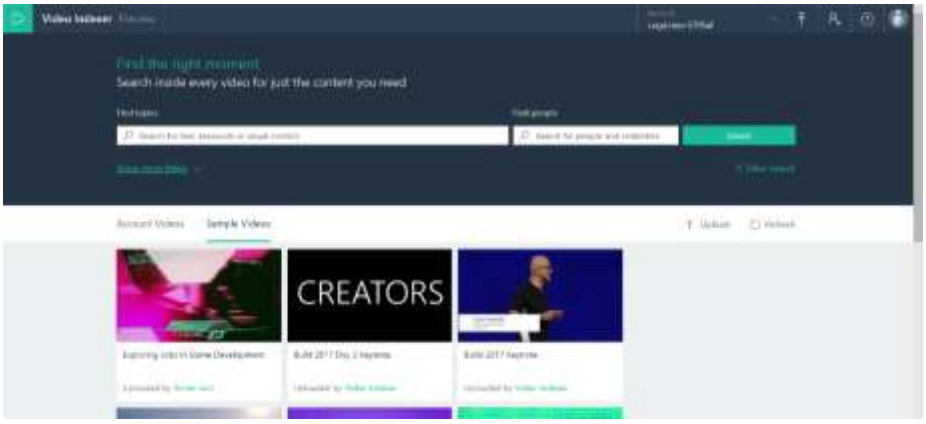
Video Indexer の動作確認

9. **Video Indexer** をクリックして開きます。Video Indexer はアップロードされたビデオを分析し、登場する人物、発話内容の分析を行い、翻訳して一般公開可能なプラットフォームです。**Video Indexer を試す** をクリックして、デモサイトを開きます。

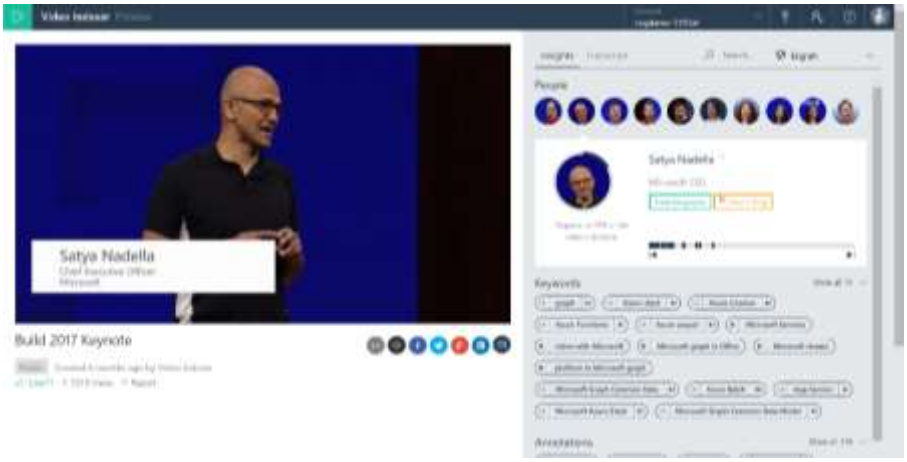


10. Video Indexer の Web サイトで **Sign in** をクリックしてサインインします。アプリアクセス許可の画面が表示されたら、[はい] をクリックして進みます。

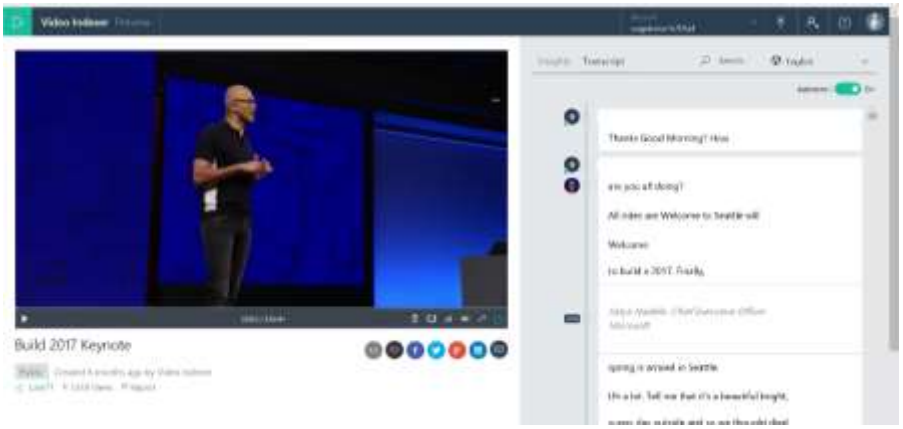
11. Video Indexer の画面で、**Sample Video** タブをクリックします。表示されているビデオのうち、いずれかをクリックします。



12. ビデオ再生画面の右側に、ビデオ分析結果が表示されます。Insight タブには、登場する人物とビデオの登場箇所、発言キーワード、タグ、発話の感情などが表示されます。



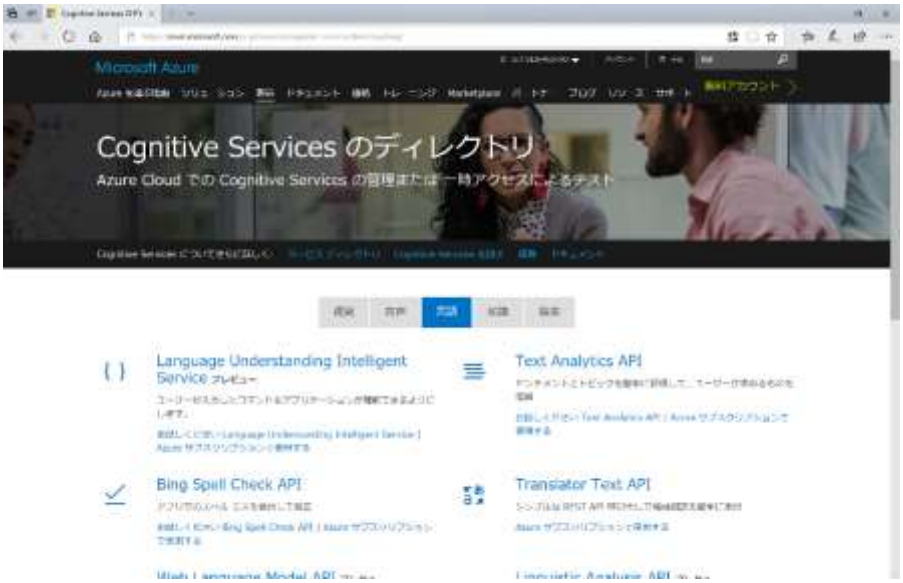
13. **Transcript** タブをクリックすると、発話内容がテキスト表示されます。言語のドロップダウンリストを操作すると、翻訳されて表示されることを確認してください。



Text Analytics API の動作確認

14. Cognitive Services の Web サイト (http://microsoft.com/cognitive) に戻ります。**[言語]** をクリックします。

15. [言語] タブには、自然言語を分析する API 群が表示されています。**Text Analytics API** をクリックします。

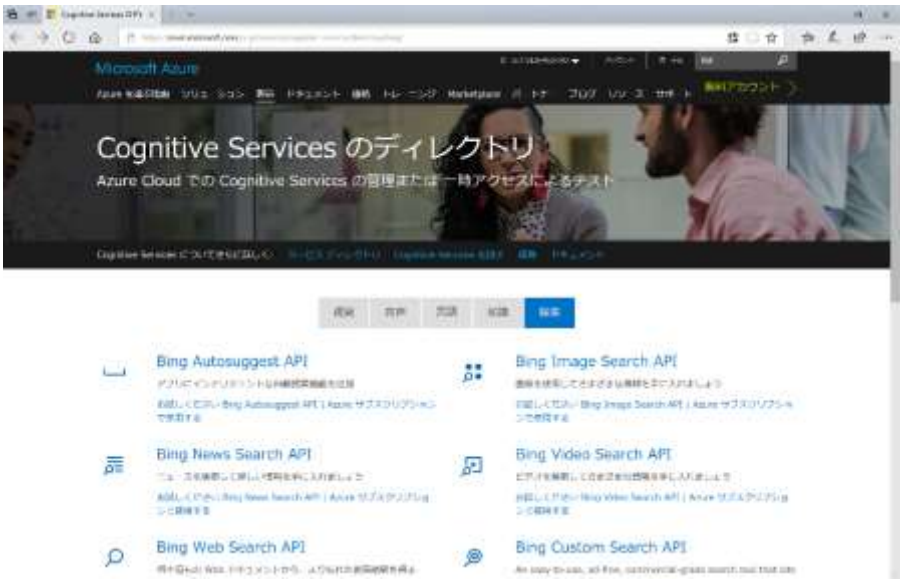


16. Text Analytics API は、文章の言語、キーワード、センチメント (感情) を分析します。デモ画面で、サンプル文章 または 自由に文章を入力して、分析結果を確認してください。

17. ブラウザーのバック (←) ボタンでひとつ前のページに戻ります。

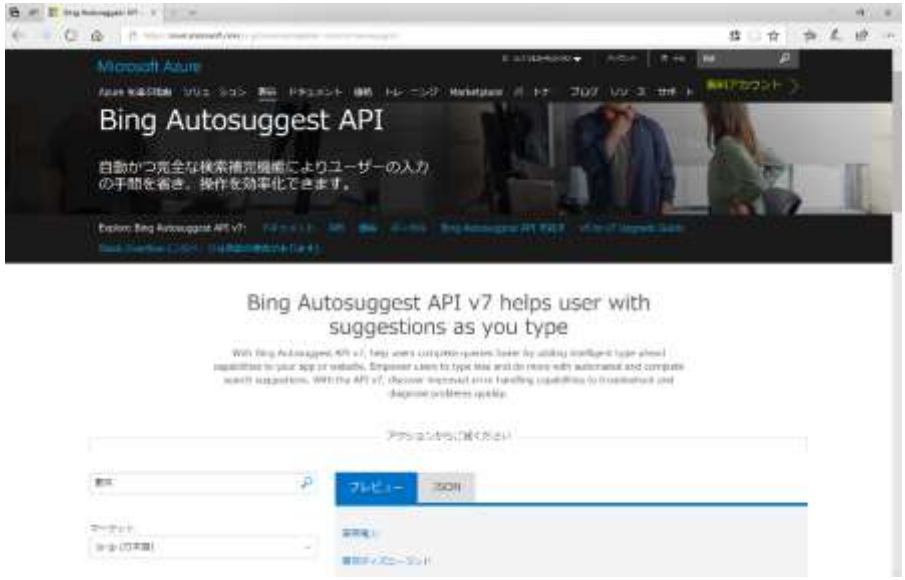
Bing Web Search API, Bing Autosuggest API の動作確認

18. **[検索]** をクリックします。[検索] タブには、Web サイトや画像、動画、ニュースなどを様々な条件でインテリジェント検索する API 群が表示されています。



19. **Bing Web Search API** をクリックして開きます。マーケットや鮮度(情報の新しさ)、フィルター を指定して、Web 検索結果をアプリで利用しやすい形で取得できることを確認してください。

20. ブラウザーのバック（←）ボタンでひとつ前のページに戻り、今度は Bing Autosuggest API をクリックします。こちらはキーワードの一部を入力すると、それに続く言葉を推測し、表示します。



他にも Cognitive Services にはさまざまな機能を提供する API が用意されています。Cognitive Services の API の機能を調べる場合は、まず Web サイトで確認し、デモサイトで動作を確認してください。

演習 2: Cognitive Services のサブスクリプションを申し込み、サンプルアプリケーションで利用シナリオを学ぶ

この演習では、まず Cognitive Services の利用に必要なサブスクリプション申し込みを行います。その後、申し込んだサブスクリプションを用いて Cognitive Services のサンプルアプリケーションを稼働させ、Cognitive Services を活用した代表的なシナリオについて学習します。

Cognitive Services のサブスクリプション申込

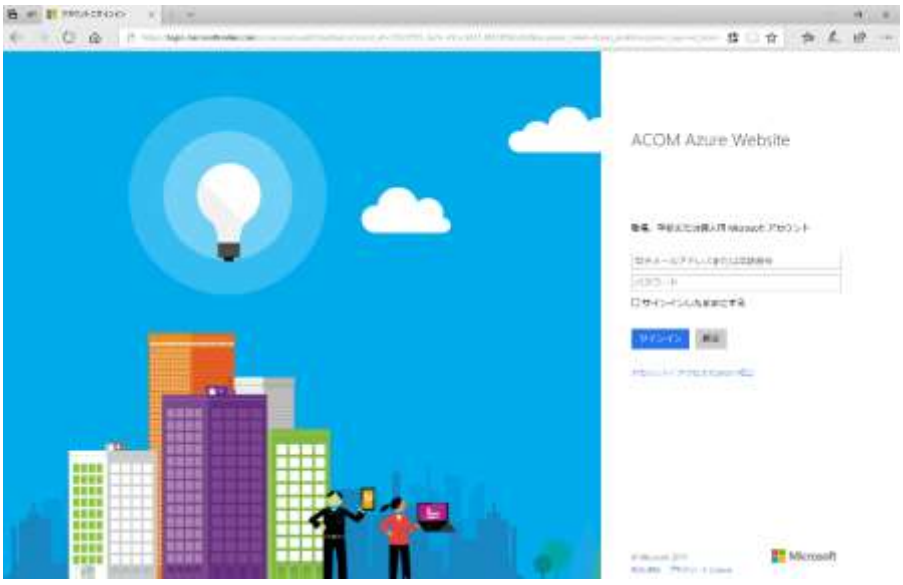
1. Cognitive Services の Web サイト (http://microsoft.com/cognitive) をブラウザーで再度開きます。 **[Cognitive Services を無料で試す]** をクリックして、無料試用サブスクリプション申込ページを開きます。

2. [視覚] タブ内の Computer Vision API の行にある **[API キーの取得]** をクリックします。

3. Microsoft Cognitive Services 使用条件 の画面で、サービス条件に合意してチェックボックスをクリックし、チェック (✓) をつけます。[国とリージョンを選択] で **日本** を選択します。**[次へ]** をクリックします。

4. Microsoft をクリックして、マイクロソフトアカウントでサインインを行います。

5. お持ちのマイクロソフトアカウントの情報を入力してサインインします。



6. 「Computer Vision API を正常にサブスクリプションに追加しました。」と表示され、Computer Vision API の試用版サブスクリプションの申し込みは完了です。

7. 画面を下にスクロールし、同じページ内にある　Face API の欄にある **[追加]** をクリックし、Face API のサブスクリプションを追加します。

8. 同様のプロセスで、Emotion API、Text Analytics API、Bing (Web) Search API、Bing Auto Suggest API のサブスクリプションを追加します。

すべての API を表示するには、**[すべて表示]** をクリックします。

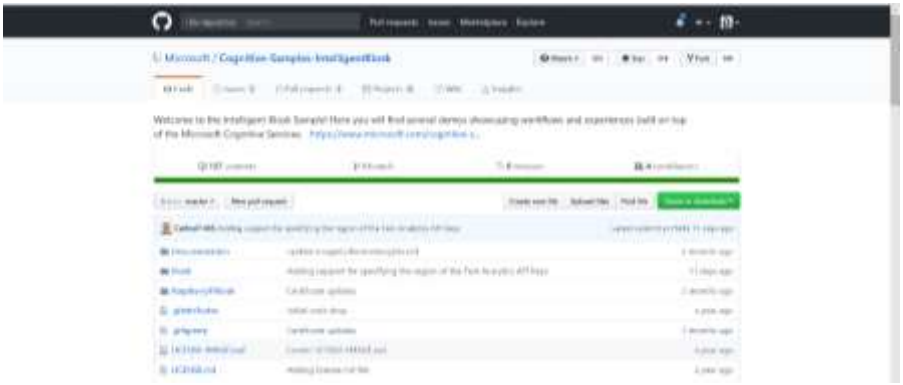
9. Computer Vision API および Face API は、それぞれ表示されている エンドポイントの URL および キー1 に記載されている文字列をコピーして保管しておきます。

10. その他の API は、それぞれ キー1 に記載されている文字列をコピーして保管しておきます。

Intelligent Kiosk のダウンロードと動作確認

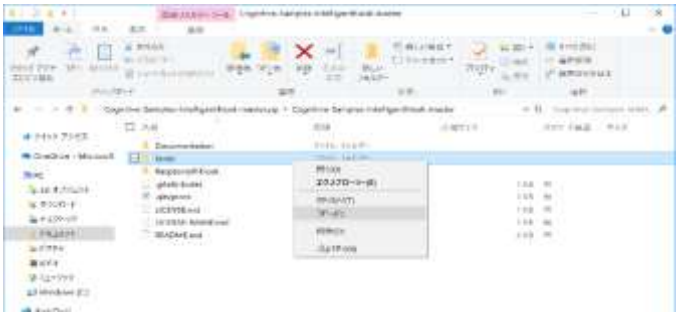
このタスクでは、上記で申し込んだ Cognitive Services のサブスクリプションを用いて、ユニバーサル Windows アプリ (UWP) のサンプルアプリケーション (Intelligent Kiosk) を稼働させます。

11. ブラウザーで Intelligent Kiosk Sample の GitHub ページ (https://github.com/Microsoft/Cognitive-Samples-IntelligentKiosk) を開きます。

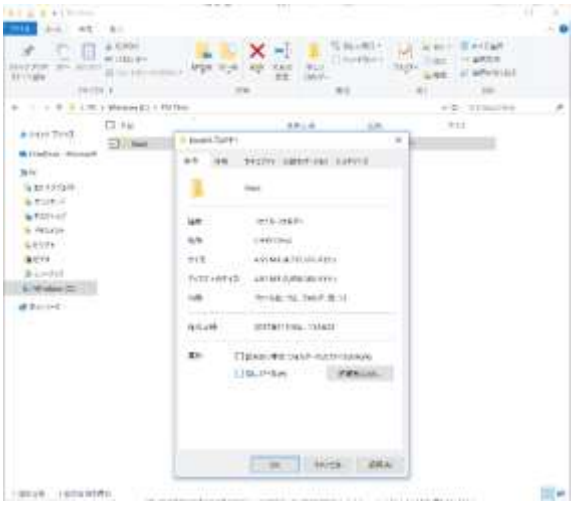


12. **[Clone or download]** をクリックし、**[Download Zip]** を選択して UWP の C# プロジェクトファイル、ドキュメントをまとめてダウンロードします。

13. ダウンロードした Zip ファイルをクリックして内容を確認します。**Cognitive-Samples-IntelligentKiosk-master** フォルダーをクリックします。**Kiosk** フォルダーを右クリックしてコピーし、デスクトップ (または作業フォルダー) にペーストします。

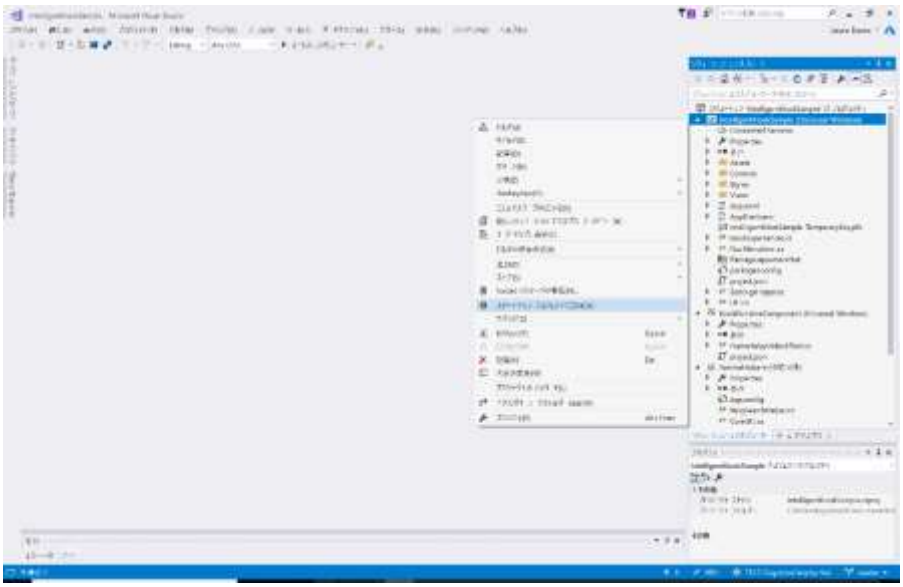


14. 展開したファイルを右クリックして、**[プロパティ]** をクリックします。[読み取り専用] のチェックボックスをクリックして、選択されていない状態にして **[OK]** をクリックします。[すべてのフォルダーとファイルに適用する] を選択します。

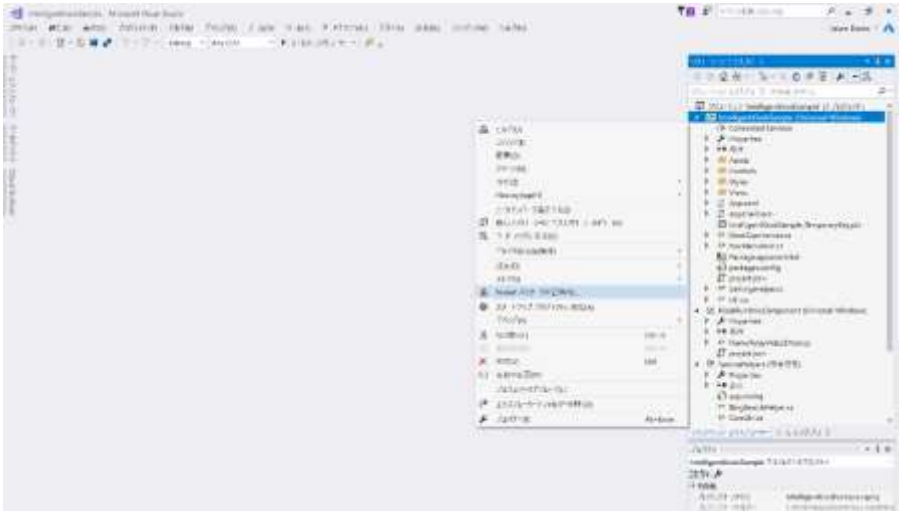


15. Kiosk フォルダー内にある **IntelligentKioskSample.sln** をクリックすると、Visual Studio が起動してソリューションが開きます。セキュリティ警告が表示されたら [OK] をクリックしてプロジェクトを開きます。

16. IntelligentKioskSample (Universal Windows) というプロジェクトを右クリックして、**[スタートアップ プロジェクトに設定]** を選択します。

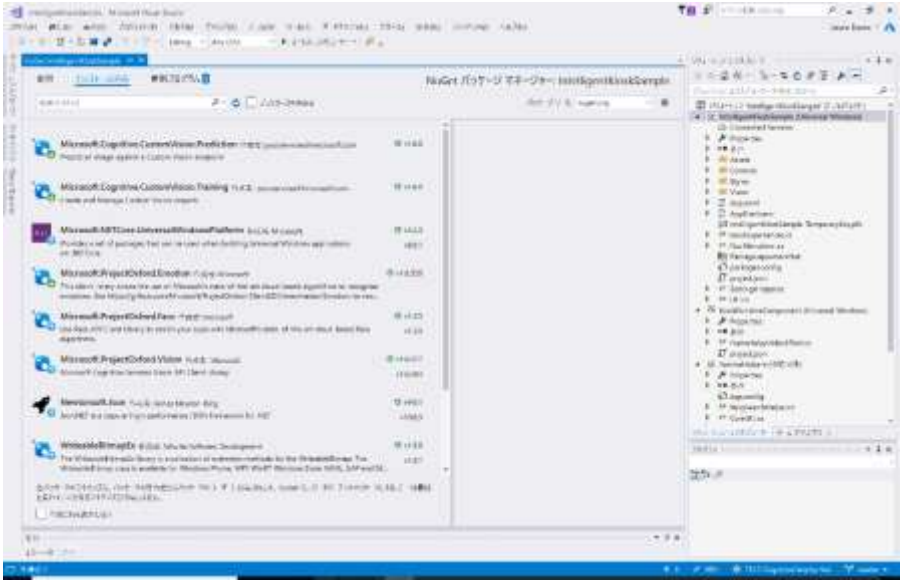


17. 再度 IntelligentKioskSample (Universal Windows) を右クリックして、**[Nuget パッケージの管理]** を選択します。



18. 「このソリューションに一部の NuGet パッケージが見つかりません。…」と表示された列の右にある **[復元]** をクリックして、足りないパッケージを復元します。

19. **[▶ローカルコンピューター]** または **F5** キーをクリックして、アプリのビルドおよびローカル環境でのデバッグ実行を行います。



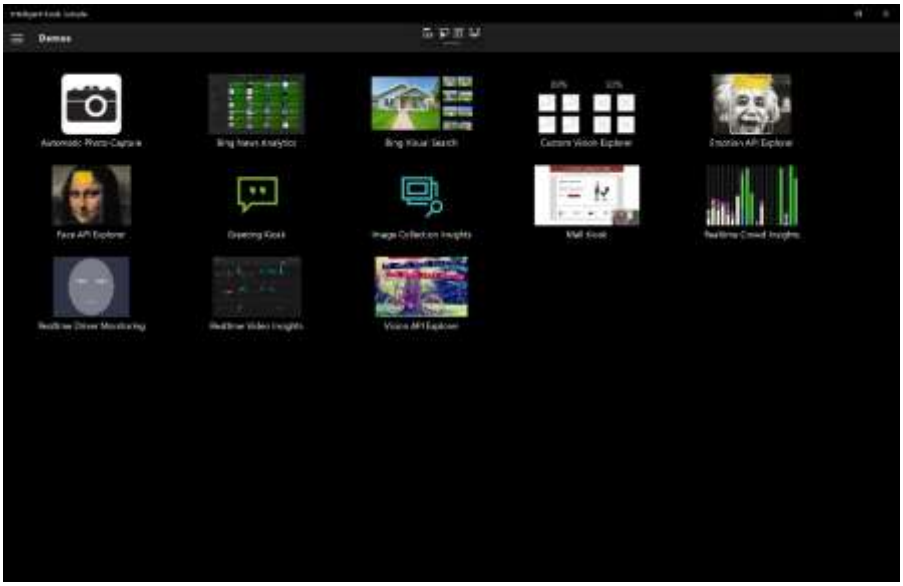
20. Intelligent Kiosk のアプリが起動します。最初に Cognitive Services のサブスクリプションキーを登録します。アプリ画面で左上の ≡（メニュー）をクリックして、**[Settings]** を選択します。

21. Keys の画面で、Face API Key、Emotion API Key、Computer Vision API Key, Bing Search API Key, Bing Auto Suggestion API Key, Text Analytics Key の欄に、演習 1 で取得した サブスクリプションキー をコピーします。

Face API Key、Computer Vision API Key、Text Analytics Key は、サブスクリプションキーと一緒に表示された URL に含まれるリージョン（Region）を選択してください。

22. サブスクリプションキーの登録後、実際にアプリに用意されているデモを確認します。
≡（メニュー）をクリックして、**[Demos]** を選択します。

23. Cognitive Services の各 API を利用して作成されたデモ機能が表示されます。その中で主なデモの動作を確認します。



Face API Explorer、Emotion API Explorer、Vision API Explorer のアプリでの動作確認

視覚グループの API (Face API、Emotion API、Computer Vision API) を利用した以下のデモを稼働させ、動作を確認します。

- Face API Explorer：Face API
- Emotion API Explorer：Emotion API
- Vision API Explorer：Computer Vision API

これらのデモでは、画像の Web 検索に Bing Image Search API が使われています。Bing Search API 群は Bing Web Search API で申し込んだ同じサブスクリプションキーを利用します。

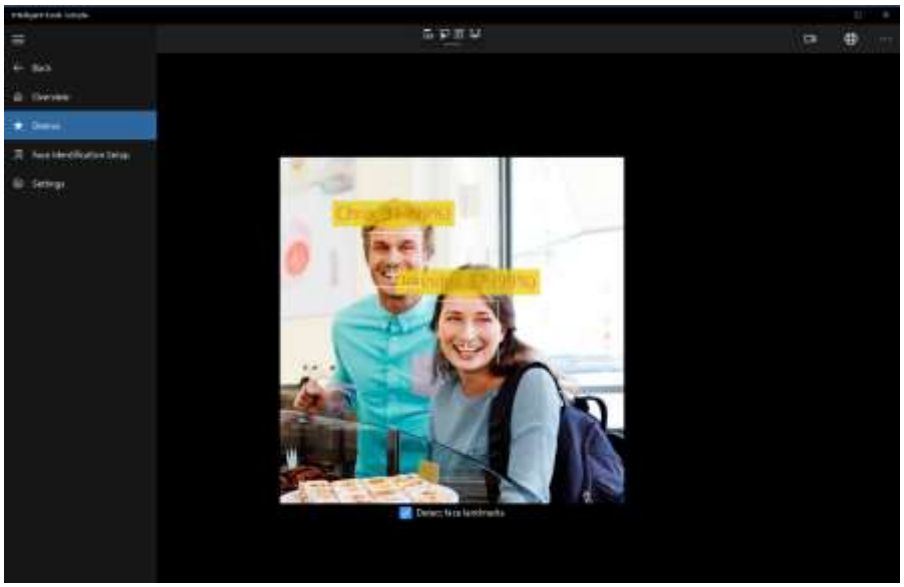
24. Demo 画面から **Face API Explorer** をクリックして開きます。

25. マシンのカメラ機能を利用するため、**[はい]** をクリックして、アプリからカメラへのアクセスを許可します。

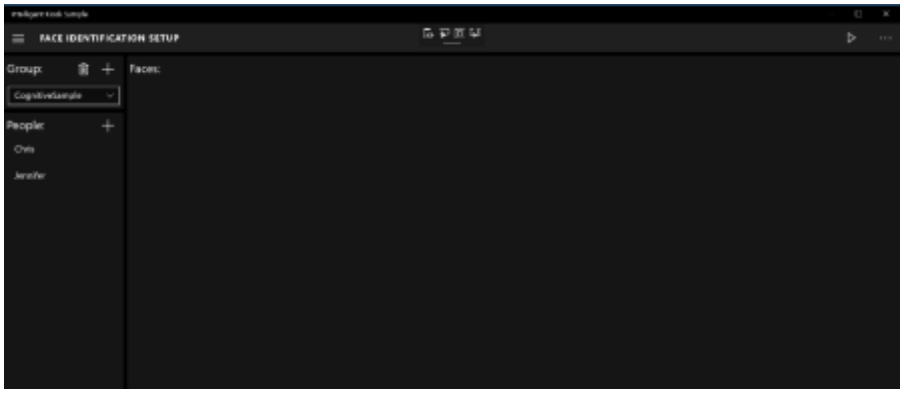
26. 中央に表示されているカメラ画像のフレームの右下にある カメラ アイコンをクリックすると画像が取り込まれ、顔の判定が行われます。または画面右上の Picture アイコンをクリックして、Web から検索した画像、またはローカルにある画像を利用できます。

Detect Face Landmark のチェックボックスをオンにすると、Face API で判定される顔のパーツの認識点 (27 か所) が表示されます。

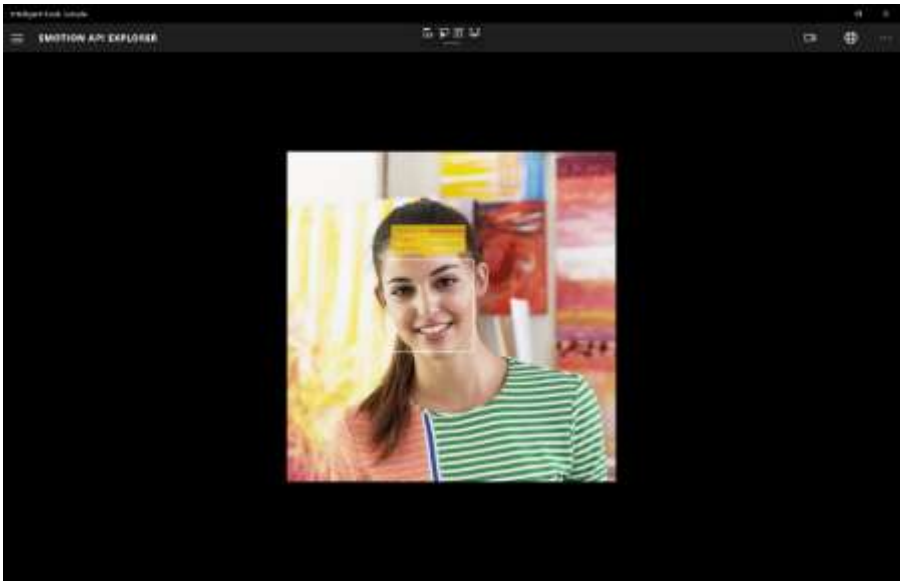
予め顔 (写真) を登録しておくと、顔検証機能により同一の可能性の高い人物の名前が表示されます。顔を登録するには、アプリ画面で左上の ≡ (メニュー) をクリックして、**[Face Detection Setup]** を選択します。



FACE IDENTIFICATION SETUP のページで、Group: の欄にある **+** をクリックして、まずグループを登録します。その後、People: の欄にある + をクリックして人物名を登録後、登録した人物をクリックして、マシンから撮影した画像 または Web から検索した画像やローカルファイルを登録します。画面右上の ▶ (Train Group) をクリックして、画像の学習を行います。



27. 同様に、Demo ページから Emotion API Explorer をクリックし、カメラ画像または画像をアップロードして、3 つの感情がグラフ表示されるのを確認してください。



28. 同様に、Demo ページから Vision API Explorer をクリックし、カメラ画像または 画像をアップロードして、タグ(Tags)、説明(Description)、著名人、写真から抽出されたカラーなどが表示されるのを確認してください。

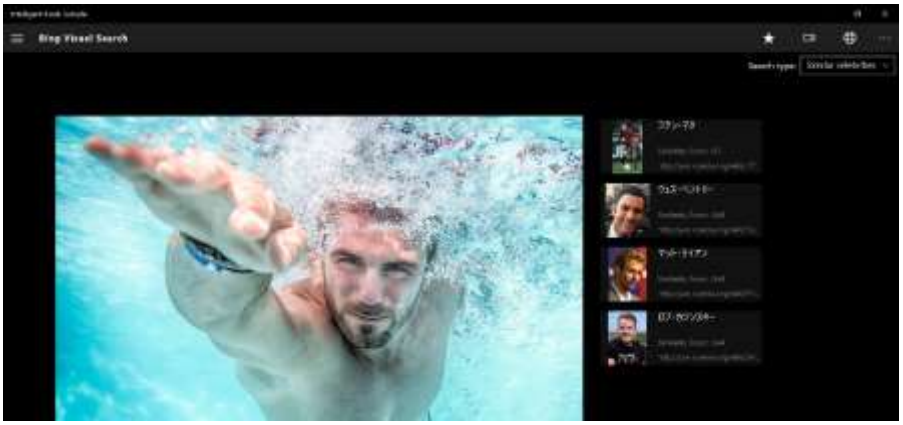
Text Analytics、Bing Web Search API、Bing Autosuggest API のアプリでの動作確認

言語グループの API (Text Analytics API) および 検索グループの API (Bing Image Search API, Bing News Search API) を利用した以下のデモを稼働させ、動作を確認します。

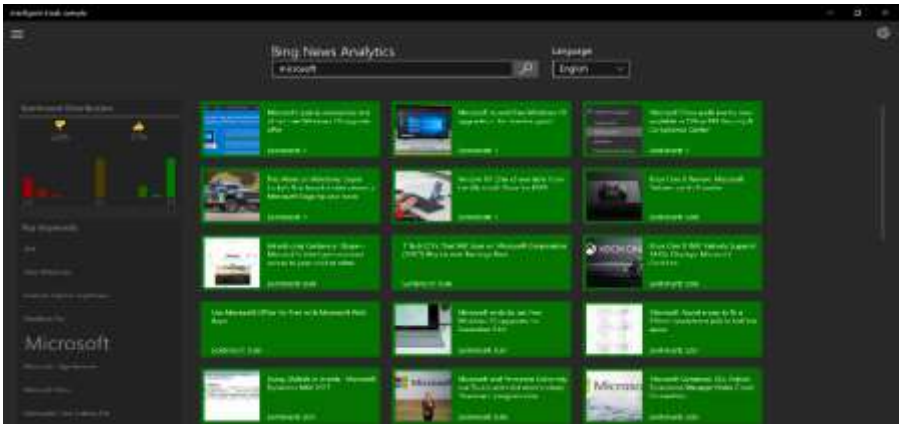
- Bing Visual Search：Bing Image Search API
- Bing News Analytics：Bing News Search API、Text Analytics API

29. Demo 画面から **Bing Visual Search** をクリックして開きます。画面の右上の ★ (Suggestions) をクリックして、予め用意されている画像を選択する、またはカメラで写真を撮る、ローカルの画像をアップロードする、いずれかの方法で画像を選択します。右上の Search type で [Similar images] が選択されている場合、選択した画像に似た画像が Web 検索され、表示されます。

30. 右上の Search type で [Similar celebrities] が選択されている場合は、選択した画像に含まれる人物に似た著名人が Web 検索され、表示されます。



31. Demo 画面から **Bing News Analytics** をクリックして開きます。画面上中央の検索欄に検索したい言葉を入力します。画面中央には関連するニュース、右側にはニュースのネガポジ分析、および ニュースから抽出されたキーワード が表示されます。



Language は English を選択し、英語のキーワードを入力してください。ドロップダウンリストに用意されている他の言語でも構いません。

他にも以下のような Cognitive Services を活用したデモシナリオが用意されています。

- Realtime Crowd Insights：Face API、Emotion API

  カメラ画像から検出される人物の年齢、性別、表情をタイムライン表示

- Realtime Driver Monitoring：Face API

  Face API で判定した顔のパーツの位置情報を利用して、ドライバーのよそ見や居眠り運転を検出

- Mail Kiosk：Face API

  カメラ画像から検出される人物の性別、年齢から、表示する Web サイト(広告)を変更

演習 1,2 を通じて、Cognitive Services の代表的な API とその利用方法について学習したところで、実際に Cognitive Services を利用したアプリの構築を行います。

演習 3: Face API を利用した 人物情報表示 UWP アプリを作成する

この演習では、Cognitive Services Face API を実際に利用して、取り込んだ画像を分析して人間の顔に関する情報 (年齢、性別、表情) を出力する ユニバーサル Windows (UWP) アプリを構築します。開発環境として、マルチプラットフォーム (Windows, iOS, Android) 向けアプリケーションを C#で開発できる Xamarin を利用します。

新しい Xamarin プロジェクトの作成

1. Visual Studio 2017 を起動して、**[ファイル]** > **[新規作成]** > **[プロジェクト]** の順にクリックします。

2. **新しいプロジェクト** の画面で、**Cross Platform App (Xamarin)** を選択、**CognitiveFaceApp** という名前を入力し、[OK] をクリックして新規プロジェクトを作成します。

3. テンプレートの選択 (Select a template) の画面で、**空のアプリ(Blank App)** を選択し、Code Shareing Strategy は **ポータブルクラスライブラリ(PCL) (Portable Class Library (PCL))** を選択、画面右下の **[OK]** をクリックします。

4. ユニバーサル Windows アプリケーションがサポートする、既定のターゲット バージョンと最小バージョンを確認し、[OK] をクリックします。

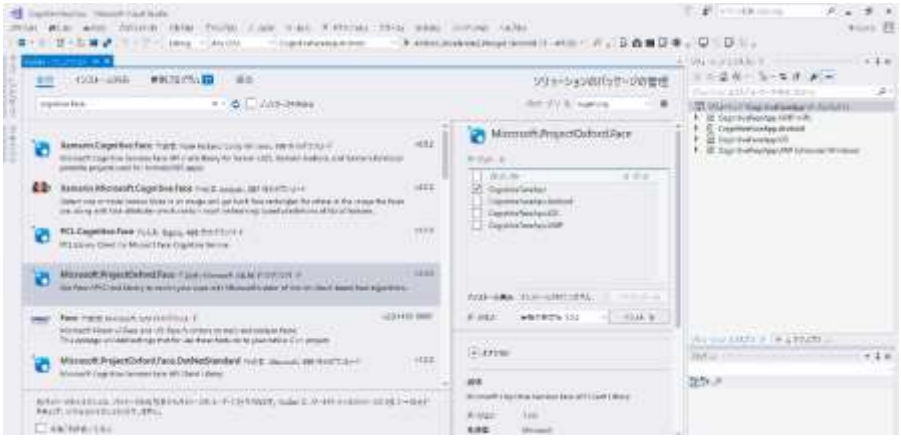
NuGet によるパッケージの管理と Cognitive Services Face API Client Library および必要なパッケージのインストール

NuGet は、Microsoft 管理プラットフォーム向けの無料のオープンソース パッケージです。NuGet では、さまざまなタスクを実行するコードが収録された数千ものライブラリーのパッケージを利用できます。NuGet は Visual Studio 2017 に統合されており、簡単に NuGet 経由でパッケージをプロジェクトに追加、管理することができます。

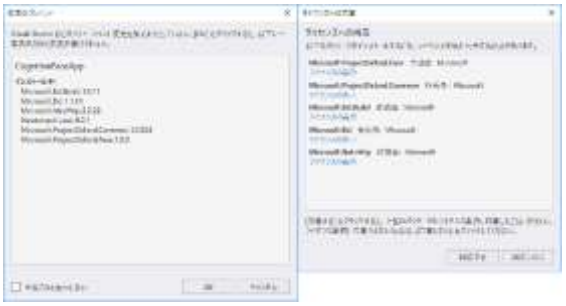
5. [ソリューションエクスプローラー] ウィンドウで、**CognitiveFaceApp** ソリューションを右クリックし、**[NuGet パッケージの復元]** を選択、プロジェクトに含まれていないパッケージを復元します。

6. [ソリューションエクスプローラー] ウィンドウで、再度 **CognitiveFaceApp** ソリューション を右クリックし、**[NuGet パッケージの管理..]** を選択します。

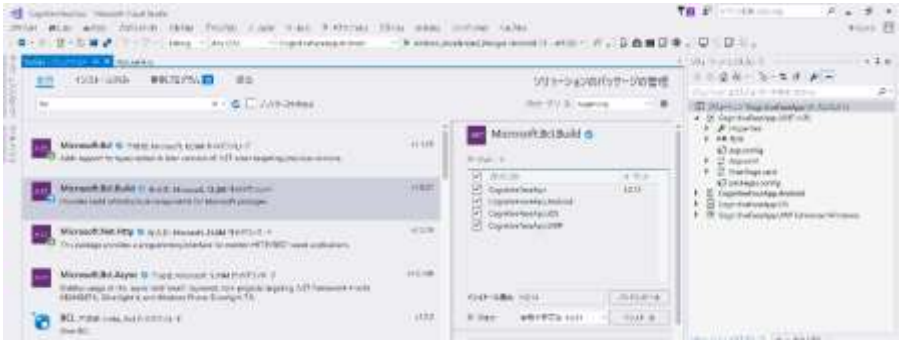
7. **[参照]** タブをクリックします。検索ボックスに、**cognitive face** と入力します。**Microsoft.ProjectOxford.Face** をクリックして、NuGet のこの クライアント ライブラリー を選択します。**[インストール]** をクリックして、**最新の安定版** のパッケージをインストールします。このパッケージには、アプリで Web サービスとの通信に使用する ヘルパー API が収録されています。



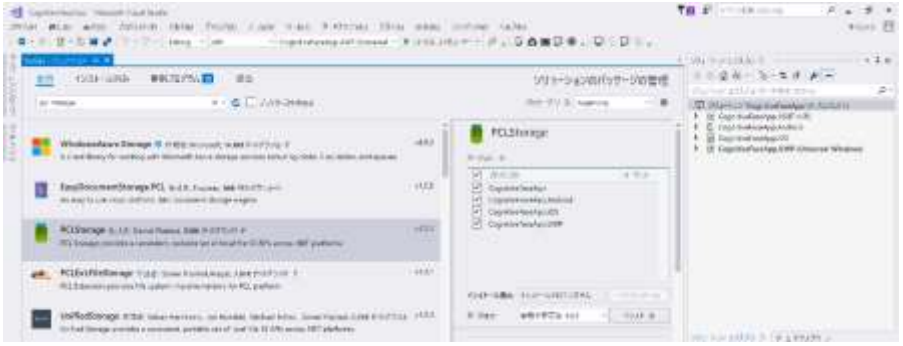
8. 変更の確認を求められたら [OK] をクリックし、ダウンロードしたパッケージのライセンスへの同意を求められたら [同意する] をクリックします。



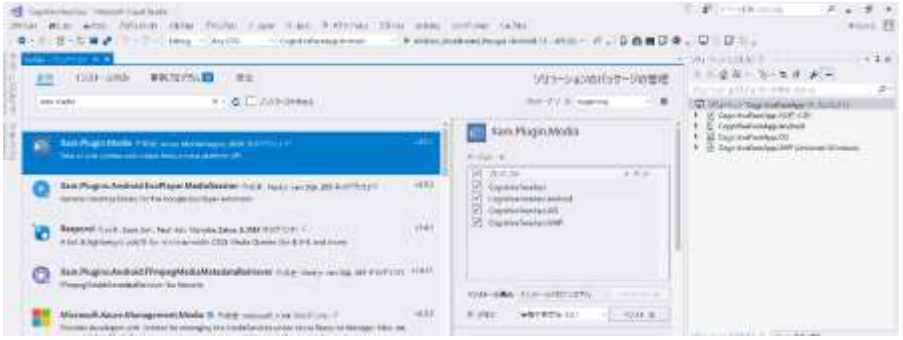
9. 次に、[参照] タブで、検索ボックスに **bcl build** と入力して **Microsoft.Bcl.Build** を探します。すべてのプロジェクトに **チェック** ✓ をつけ、**[インストール]** をクリックして、**最新の安定版** のパッケージをインストールします。



10. 同様に [参照] タブで、検索ボックスに **pcl storage** と入力して **PCLStorage** を探し、**[インストール]** をクリックして、**最新の安定版** のパッケージをインストールします。このパッケージにはモバイル環境でローカルストレージを利用するための SDK が収録されています。



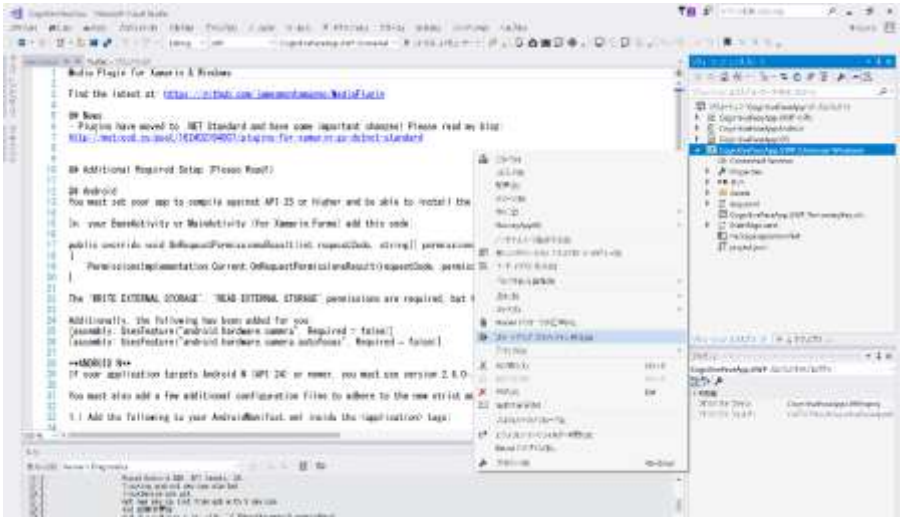
11. 最後に、[参照] タブで、検索ボックスに **xam media** と入力して **Xam.Plugin.Media** を探し、**[インストール]** をクリックして、**最新の安定版** のパッケージをインストールします。このパッケージにはモバイル環境でカメラやマイクを利用するための SDK が収録されています。



12. ソリューションエクスプローラーで package.appxmanifest をクリックして開きます。機能 タブで、Web カメラ のチェックボックスをクリックしてオンにします。

> 今回は UWP のみアプリからカメラを利用する設定を行いました。iOS または Android 向けアプリは、Xam.Plugin.Media のインストール直後に表示される readme.text の指示に従って設定を行ってください。

13. ソリューションエクスプローラーで **CognitiveFaceApp.UWP (Uiversal Windows)** というプロジェクトを右クリックし、**スタートアッププロジェクトに設定** をクリックして設定します。



14. 上部バーの保存ボタンをクリックして、プロジェクト全体の変更を保存しておきます。

UI (MainPage.xaml) のコーディング

15. **[ソリューションエクスプローラー]** ウィンドウで、**MainPage.xaml** を開きます。XAML ウィンドウ内のコードから、以下の空の ContentPage.Content 要素を見つけます。

```
<ContentPage.Content>

</ContentPage.Content>
```

16. 上記の空の ContentPage.Content 要素を次のコードに置き換えます。このコードでは、写真を表示するグリッド、2 種類の操作ボタン (写真の選択、リセット)、および 判定した年齢、性別、感情を表示するラベルを作成します。

```
<ContentPage.Content>
        <StackLayout>
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="*" />
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="Auto" />
                </Grid.RowDefinitions>
                <Image x:Name="ImagePreview"
                    Grid.Row="0" HorizontalOptions="Center"/>
                <Button Grid.Row="1"
                    Clicked="RunButton_OnClicked" Text="写真の選択" />
                <Button Grid.Row="2"
                    Clicked="ResetButton_OnClicked" Text="リセット" />

                <StackLayout Grid.Row="3" HorizontalOptions="Center">
                    <Label x:Name="Age">年齢</Label>
                    <Label x:Name="Gender">性別</Label>
                    <Label x:Name="Emotion">表情</Label>
                </StackLayout>
            </Grid>
        </StackLayout>
    </ContentPage.Content>
```

挿入したマークアップは、Extensible Application Markup Language (XAML) です。XAML は、ユーザー インターフェイスの構築用にマイクロソフトが作成した言語です。XAML を Xamarin Forms と組み合わせてと、ユニバーサル Windows アプリ、iOS、Android 向けのユーザー インターフェイスの構築にも使用できます。XAML は、Visual Studio やその他のよく使用されているツールでデザイナー サポートを提供する、表現がきわめて豊富な言語です。

実行部分 (MainPage.xaml.cs) のコーディング

17. **[ソリューションエクスプローラー]** ウィンドウで、**MainPage.xaml.cs** を開き、ページの上部に既にある using ステートメントに、以下の using ステートメントを追加します。

```
using Microsoft.ProjectOxford.Face;
using PCLStorage;
using Plugin.Media;
using Plugin.Media.Abstractions;
```

18. 続けて MainPage.xaml.cs の **MainPange : Content Page** の中に Cognitive Services Face API を呼び出して判定するコードを追加していきます。まず、Face API の戻り値を格納する FaceDetectResult クラスを追加します。

```
public class FaceDetectResult
{
        public double age { get; set; }
        public string gender { get; set; }
        public string emotionKey { get; set; }
        public float emotionValue { get; set; }
}
```

19. サブスクリプションキーを用いて Face API を呼び出して判定するタスクを以下のように作成します。コード内にある Your_FaceAPISubKey は、演習 2 で取得した Face API キー に置き換えます。

```
Public static async Task<FaceDetectResult> DetectFaceAsync(string photo)
{
    // Face API 呼び出し準備
    var subKey = "Your_FaceAPISubKey";
    var client = new FaceServiceClient(subKey);

    // Face API で判定
    var file = await FileSystem.Current.GetFileFromPathAsync(photo);
    var imageStream = await file.OpenAsync(FileAccess.Read);
    var result = await client.DetectAsync(imageStream, false, false,
                                Enum.GetValues(typeof(FaceAttributeType)).OfType
                                <FaceAttributeType>().ToArray());

    // 判定結果を代入
    var detectResult = new FaceDetectResult();

    detectResult.age = result[0].FaceAttributes.Age;
    detectResult.gender = result[0].FaceAttributes.Gender;
    detectResult.emotionKey
        = result[0].FaceAttributes.Emotion.ToRankedList().First<KeyValuePair
        <string, float>>().Key;
    detectResult.emotionValue
        = result[0].FaceAttributes.Emotion.ToRankedList().First<KeyValuePair
        <string, float>>().Value;

    return detectResult;
}
```

20. このアプリでは、モバイル内臓のカメラからの画像、またはローカルフォルダーに保存されている画像を利用します。以下のように、カメラの初期化と操作(TakePhotoAsync) および ローカルフォルダーの操作（PickPhotoAsync) を行うコードを追加します。

```
public static async Task<string> TakePhotoAsync()
{
    // カメラを初期化
    await CrossMedia.Current.Initialize();

    // カメラを使えるかどうか判定
    if (!CrossMedia.Current.IsCameraAvailable
```

```
            || !CrossMedia.Current.IsTakePhotoSupported)
        {
            throw new NotSupportedException
                ("カメラをアプリから利用できるように設定してください");
        }

        // 撮影し、保存したファイルを取得
        var photo = await CrossMedia.Current.TakePhotoAsync
            (new StoreCameraMediaOptions());

        // 保存したファイルのパスを取得
        return photo.Path;
}
```

```
public static async Task<string> PickPhotoAsync()
{
    // ローカルフォルダーから写真を選択させる
    var photo = await CrossMedia.Current.PickPhotoAsync();

    // 保存したファイルのパスを取得
    return photo.Path;

}
```

21. 最後に、アプリのボタンコントロール、結果を表示する箇所を実装します。[写真の選択] ボタン を押したときの動作（RunButton_OnClicked）で カメラ または ローカルフォルダーを選択して、画像を判定して結果を表示するコード、そして、[リセット] ボタンを押したときの動作（ResetButton_OnClicked）を下記のように追加します。

```
private async void RunButton_OnClicked(object sender, EventArgs e)
{
    // 画像の選択
    var photo = "";
    var imageChoiceResult = await DisplayAlert("どちらの画像を使いますか", "",
        "カメラ", "ローカルフォルダー");

    try
    {
        if (imageChoiceResult)
        {
```

```
                photo = await TakePhotoAsync();
        }
        else
        {
                photo = await PickPhotoAsync();
        }
}
catch (Exception exception)
{
        await DisplayAlert("エラーが発生しました", exception.Message, "OK");
}

// 画像の判定
ImagePreview.Source = photo;
var faceResult = new FaceDetectResult();

try
{
        faceResult = await DetectFaceAsync(photo);
}
catch (Exception exception)
{
        await DisplayAlert("Face API の呼び出しに失敗しました",
                exception.Message, "OK");
        return;
}

// 判定結果を配置
Age.Text = "年齢 : " + faceResult.age;
Gender.Text = "性別 : " + faceResult.gender;

Emotion.Text = "表情 : ";
switch (faceResult.emotionKey)
{
        case "Anger":
                Emotion.Text = Emotion.Text + "怒り";
                break;
        case "Contempt":
                Emotion.Text = Emotion.Text + "軽蔑";
                break;
        case "Disgust":
```

```
                Emotion.Text = Emotion.Text + "むかつき";
                break;
            case "Fear":
                Emotion.Text = Emotion.Text + "恐れ";
                break;
            case "Happiness":
                Emotion.Text = Emotion.Text + "喜び";
                break;
            case "Neutral":
                Emotion.Text = Emotion.Text + "無表情";
                break;
            case "Sadness":
                Emotion.Text = Emotion.Text + "悲しみ";
                break;
            case "Surprise":
                Emotion.Text = Emotion.Text + "驚き";
                break;
            default:
                break;
        }
        Emotion.Text = Emotion.Text + "(" +
                                faceResult.emotionValue.ToString("0.00%") + ")";

}
```

```
private async void ResetButton_OnClicked(object sender, EventArgs e)
{
    ImagePreview.Source = "";

    Age.Text = "年齢";
    Gender.Text = "性別";
    Emotion.Text = "表情";
}
```

上記のコードでは、指定した画像内で最初に判定される人物 1 名について、年齢、性別、感情(の一番スコアが大きいもの)のみ抽出しています。FaceServiceClient.DetectAsync で取得できる情報についての詳細は、Face API Reference (https://westus.dev.cognitive.microsoft.com/docs/services/563879b619845

50e40cbbe8d/operations/563879b61984550f30395236) を確認してください。

22. 以上でコーディング作業は終了です。[Visual Studio] ウィンドウの上部にある **[ビルド]** > **[ソリューションのビルド]** をクリックして、ソリューションをビルドします。ビルド エラーがある場合はそれを修正してから、Ctrl + F5 キーを押してアプリを起動し、アプリが動作することを確認してください。

アプリの動作確認

23. アプリが起動した状態で [写真を選択] をクリックします。カメラまたはローカルファイルを選択して、判定したい写真を選択します。

24. 選択した画像が表示され、判定された年齢、性別、感情が表示されることを確認してください。

おつかれさまでした。以上で、Cognitive Services を利用する Xamarin　ベースの ユニバーサル Windows アプリの完成です。

まとめ

このハンズオン ラボでは、以下の方法について学習しました。

- Cognitive Services を利用したアプリケーションを操作し、Cognitive Services が提供する機能について理解する。
- Cognitive Services Face API を利用して、画像に含まれる顔情報を取得し、表示する機能を Xamarin / UWP アプリとして実装する
