Visual Studio 2015 による Apache Cordova アプリの作成

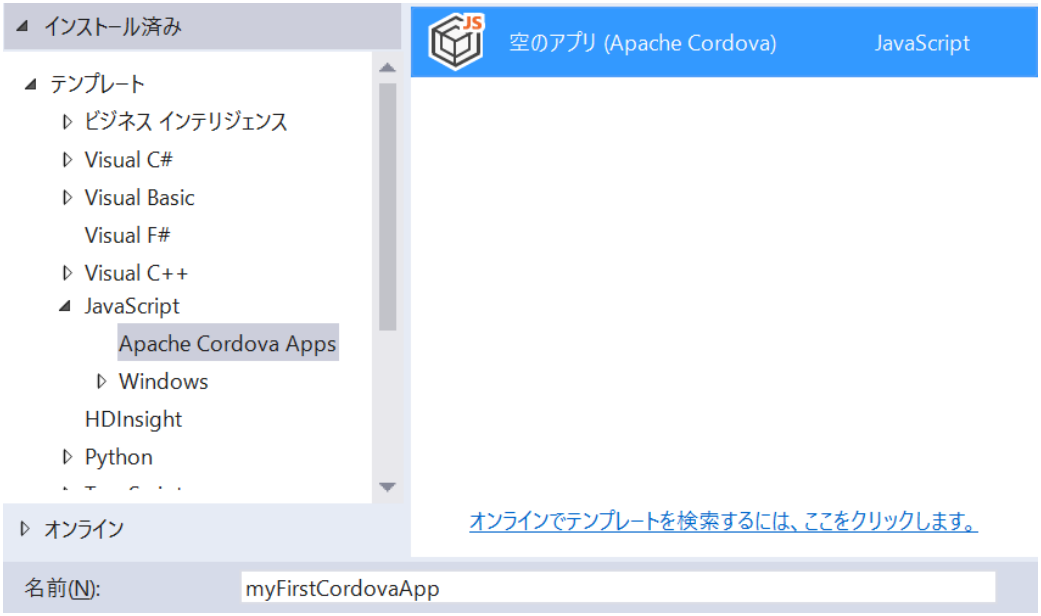
更新日: 2016 年 10 月 13 日

作業を始める前に、ツールがインストールされていることを確認します。詳細については、「Visual Studio Tools for Apache Cordova のインストール 」を参照してください。準備が整ったら作業を始めます。

プロジェクトの作成

1. Visual Studio のメニュー [ファイル] から [新規作成] – [プロジェクト] を選択して[新しいプロジェクト] ダイアログボックスを表示します。

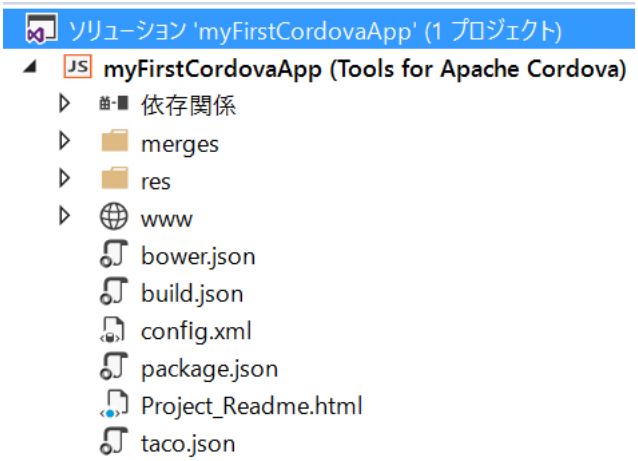
2. [新しいプロジェクト] ダイアログボックスの左側のツリービューを [テンプレート] – [JavaScript] - [Apache Cordova Apps] と展開し、ダイアログボックス右のテンプレートの一覧から 「空のアプリ (Apache Cordova)」を選択します。
   ダイアログボックス下部にある [名前] テキストボックスに myFirstCorvobaApp と入力し、[OK] ボタンをクリックします。



メモ: TypeScript を使用する場合は、[TypeScript]、[Apache Cordova Apps] にテンプレートがあります。

以上、プロジェクトの作成は完了です。

プロジェクト作成後のソリューションは以下のようになります。

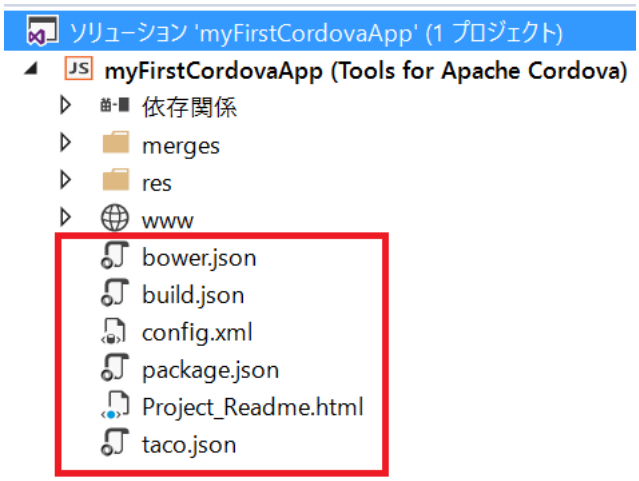


プロジェクトの概要

プロジェクトには多くのファイルがあるため、すべてをすぐに理解する必要はありません。このセクションでは、各ファイルの役割を説明していきます。じっくりと読み進めれば、ワークフローに多くのファイルを自然に組み込めるようになります。

プロジェクトの構成ファイル

プロジェクトの構成ファイルは、プロジェクトのルートにあります。



以下の表に、各ファイルの用途についての基本的な考え方を示します。

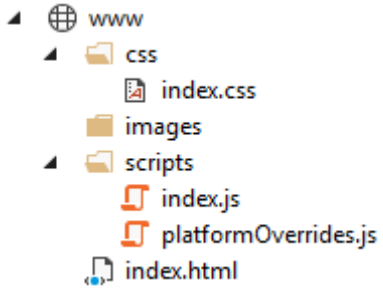
ファイル	プロジェクトでの役割
bower.json	アプリと Bower パッケージとの依存関係を管理します。 Bower はパッケージ マネージャーです。本ガイドでは Bower を使用しません。詳細についてはこちら (英語) を参照してください。

build.json	Visual Studio が署名付き Android パッケージをビルドするために使用するパラメーターを含みます。 本ガイドでは署名付き Android パッケージを作成しません。詳細についてはこちら (英語) を参照してください。
config.xml	アプリの設定を含みます。 アプリの設定は、本ガイド後半で変更します。
taco.json	Visual Studio がプロジェクトをビルドするために使用する Cordova CLI のバージョンを定義します。

プロジェクトのフォルダー

www

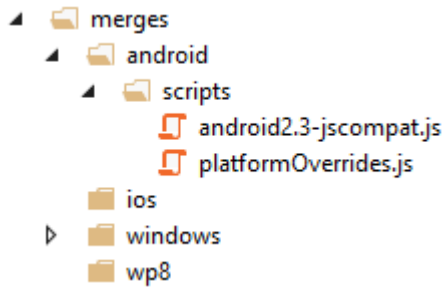
このフォルダーには、アプリで使用する HTML、JavaScript、スタイル シート、および画像を含みます。



このフォルダーには既にいくつかファイルが含まれ、このアプリがすぐに実行できるようになっています。アプリをビルドするときに、既存のファイルの変更や、ファイルの追加を行います。

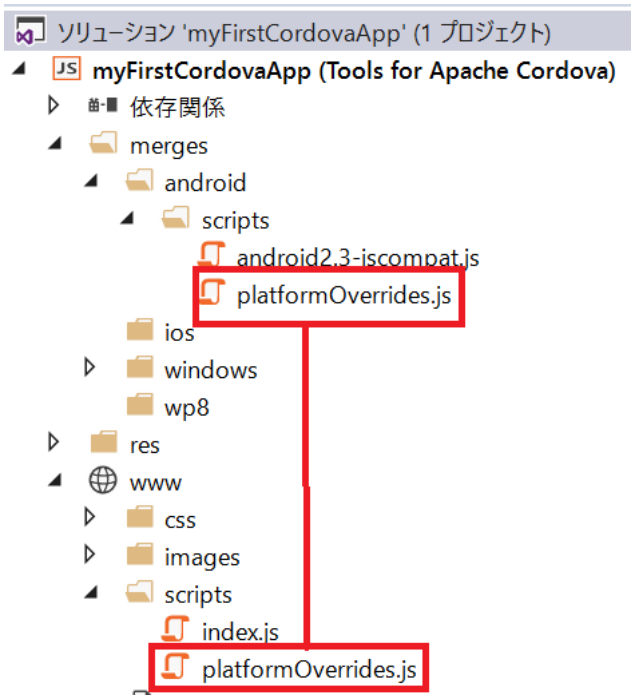
merges

このフォルダーには、特定のプラットフォームに適用する HTML、JavaScript、およびスタイル シートを含みます。



アプリをビルドするときに、[merges/プラットフォーム] にあるすべてのファイルとフォルダーが最終フォルダー構造にコピーされ、ルート プロジェクトにある同じ名前のファイルがすべて上書きされます。

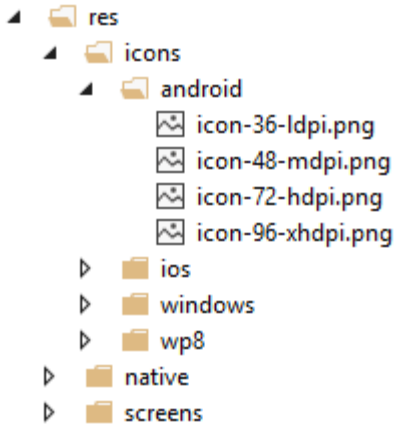
たとえば、`scripts/platformOverrides.js` ファイルは、Android では `merges/android/scripts/platformoverrides.js` ファイルに置き換えられます。このしくみを利用して、プラットフォーム固有のコードを組み込むことができます。



同じアプローチを使用して、CSS、画像などのファイルもマージされます。

res

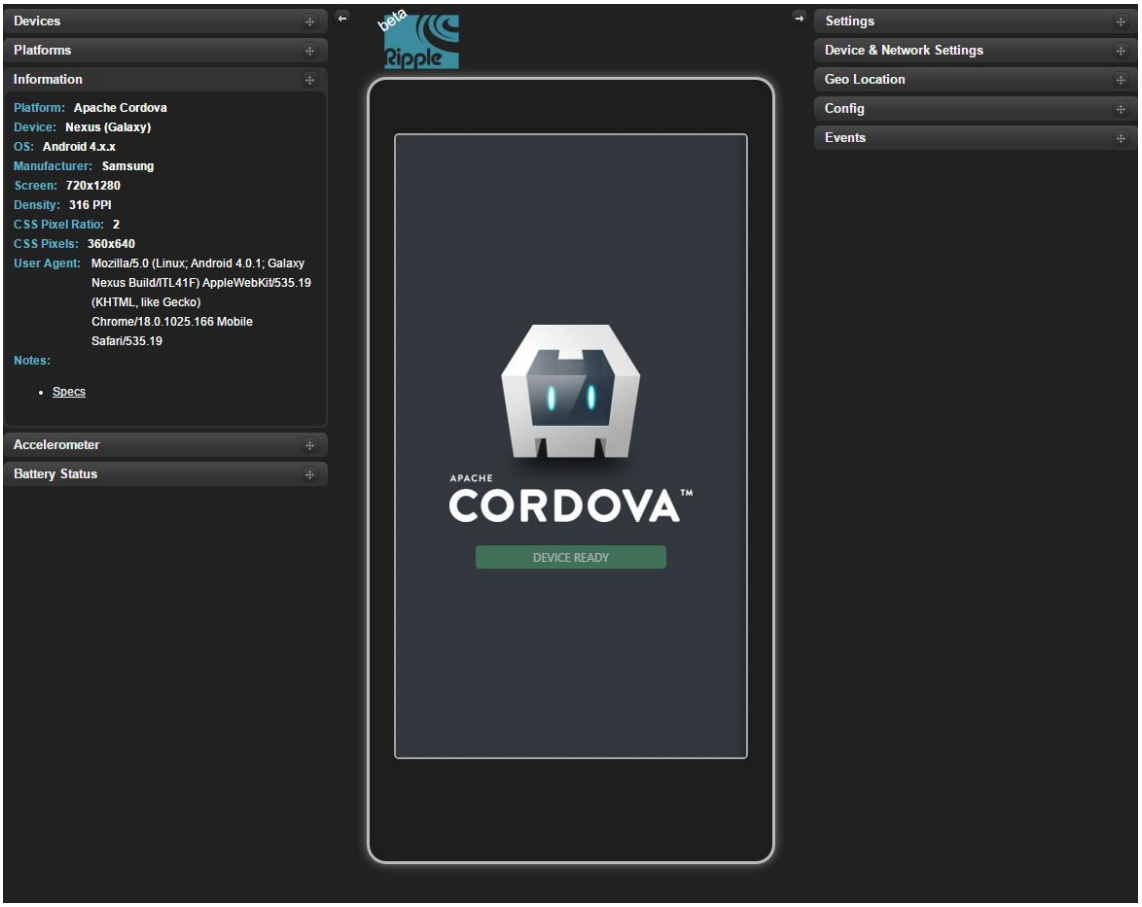
このフォルダーには、アイコン、スプラッシュ スクリーン、署名証明書など、プラットフォーム固有のファイルを含みます。



アプリの実行

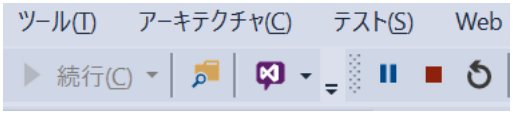
1. 標準ツール バーの [Ripple – Nexus (Galaxy)] ボタンをクリックします。

まだコードを追加していないため、アプリが以下のように Apache Ripple で開かれます。



Apache Ripple (英語) は、無料のモバイル シミュレーターです。これまで使用したことがなければ、この状態でいくつか操作を試してみます。たとえば、デバイスの向きやプラットフォームを変えて、アプリの表示を確認します。おそらく、開発中に変更の影響を確認する最も簡単な方法が Apache Ripple です。

2. アプリを停止します。アプリを停止するボタンは、標準ツール バーにあります。



アプリは、Android、iOS、および Windows の各種デバイス エミュレーターで実行することもできます。エミュレーターでは、もう少し現実的なデバイス エクスペリエンスを確認できます。

各デバイスのエクスペリエンスを確認するには、標準ツール バーのプラットフォームの一覧から任意のプラットフォームを選択します。

次に、エミュレーターを選択します。

どのエミュレーターでもアプリを実行できますが、iOS エミュレーターを使用する場合は Mac が必要です。Visual Studio と Mac との接続のセットアップはやや高度なので、本ガイドでは取り上げません。

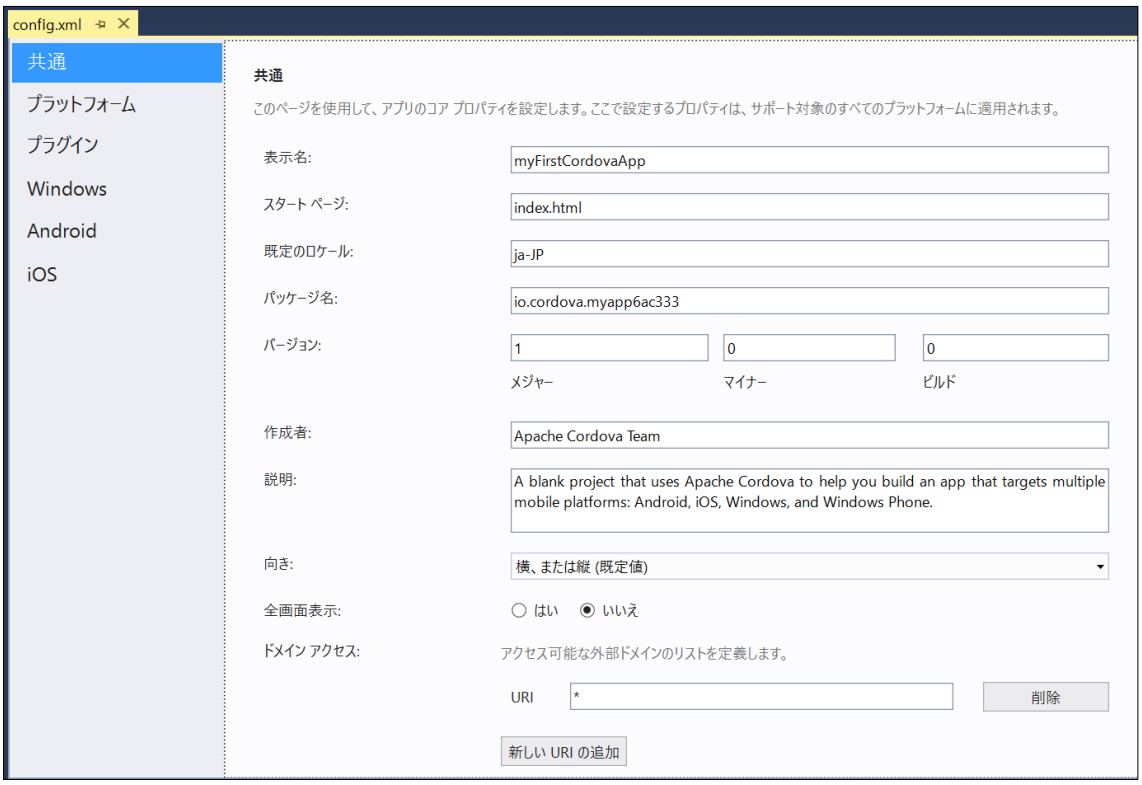
iOS でアプリを実行する場合は、Ripple シミュレーターを使用します。本ガイドを最後まで読み進めた後、iOS エミュレーターまたはデバイスでアプリを実行する方法について、こちらの資料 (英語) を参照してください。

アプリ設定の確認

アプリの名前、バージョン番号のインクリメントなど、アプリ動作のさまざまな側面を指定するには、グローバル構成ファイル **config.xml** を変更します。

このファイルは、デザイナーを使用して変更します。特に必要でない限り、XML を直接編集しないようにします。

デザイナーを開くには、プロジェクトの **config.xml** をダブル クリックします。



Cordova アプリの開発

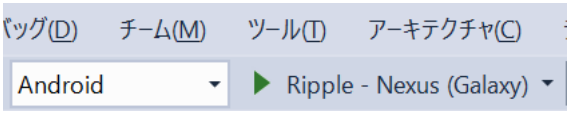
Cordova アプリの開発は www フォルダ以下に *.html や *.css、*.js といったアプリに必要なファイルを追加し、Web コンテンツと同じように開発していきます。

たとえば、既定の index.html を編集して「Hello World.」と表示させるには、以下の手順を行います。

> **メモ:** 日本語版 Apache Cordova のプロジェクトテンプレートの index.html では Shift-JIS が使用されているので、そのままでは日本語が文字化けしてしまいます。これを回避するには index.html を以下の手順で UTF-8 で保存しなおします。
>
> 1. Visual Studio 2015 内で index.html をオープンします。
> 2. 最初の meta タグ内の charset の指定を shift_jis から uft-8 に変更します。
> 3. メニュー[ファイル] – [保存オプションの詳細設定] をクリックします。
> 4. [保存オプションの詳細設定] ダイアログボックスが表示されるので [エンコード] ドロップダウンリストボックスから “Unicode (UTF-8 シグネチャ付き) -コードページ 65001” を選択して [OK] ボタンをクリックして保存します。
>
> 以上の手順で日本語の文字化けが発生しなくなります。

index.html での Hollo world の表示

1. Visual Studio 2015 内で index.html をオープンします。
2. ページに Cordova の画像を表示させないようにするために以下のタグをコメントアウトします。
   ```
   <link rel="stylesheet" type="text/css" href="css/index.css">
   ```
3. h1 タグに書かれている Apache Cordova を Hello World に書き換えます。
4. 標準ツール バーの [Ripple – Nexus (Galaxy)] ボタンをクリックしてプロジェクトを実行します。



5. Chrome ブラウザーが起動して Ripple の画面に入力した Hello World が表示されます。

index.html が参照しているファイル

index.html では Cordova が提供する独自の機能にアクセス可能とするためのファイルが参照されています。
ここではそれらのファイルの役割について紹介します。

cordova.js

cordova.js はフレームワークが動的に生成するファイルで Cordova フレームワークの JavaScript 側のインターフェースを提供します。アプリ開発者の記述する JavaScript コードは cordova.js が提供する関数 (厳密にはインターフェース) を介してネイティブ側の機能にアクセスします。

platformOverrides.js

platformOverrides.js は、前述の「プラットフォームのフォルダー」で説明したとおり、インストール対象となるプラットフォームごとに異なる固有の JavaScript コードを含ませるためのものです。
アプリをビルドするときに、[merges/プラットフォーム] にあるすべてのファイルとフォルダーが最終フォルダー構造にコピーされ、ルート プロジェクトにある同じ名前のファイルがすべて上書きされます。
たとえば、scripts/platformOverrides.js ファイルは、Android では merges/android/scripts/platformoverrides.js ファイルに置き換えられます。このしくみを利用して、プラットフォーム固有のコードを組み込むことができます。

index.js

index.js は index.html が参照しているファイルで、Cordova アプリがハンドリングしなければならない以下のイベントが記述されています。

- deviceready
  Cordova フレームワークの初期化が完了したことを通知します。
  deviceready イベントが発生するまでプラグインを介したハードウェアリソースへのアクセスなど Cordova フレームワークが提供する機能は使用することができません。
  Cordova アプリ開始時の Web ビュー内におけるイベントの発生順は DOMContentLoaded、load、deviceready です。
- pause
  モバイル OS 上で、アプリがバックグラウンドに送られるなどして中断する際に発生します。
  モバイル OS では、リソースの消費状況によってバックグラウンドに送られて停止しているアプリを終了させますが、そういった状況でも処理に支障がでないように必要なデータを保存するなどに使用します。

- **resume**
  アプリケーションが再アクティブ化されたときに発生します。
  pause イベントで保存したデータを読みだしたりするのに使用します。

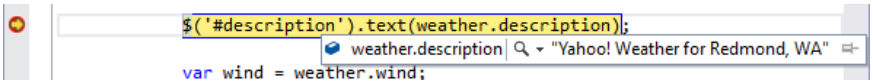
Cordova アプリケーションを開発する際、ある程度の機能を Web ブラウザーで実装してから Cordova で開発を行うというケースも多いと思いますが、index.html が参照している *.js ファイル、index.js 内でハンドリングしているイベントを扱うことによって Cordova フレームワークが提供する機能を有効に利用することができます。

問題の検出と修正

Visual Studio の最大の利点は、強力なデバッガーを備えていることです。デバッガーに精通している方は、このセクションを読み飛ばしてください。ここでは、デバッガーを使って実行できることを簡単に説明します。

ブレークポイントと条件付きブレークポイントの設定

ブレークポイントは、実行可能コード (JavaScript コードなど) の任意の行に設定できます。Apache Cordova アプリの状態表示、変数値の監視、呼び出し履歴の確認が可能です。



JavaScript コンソールの使用

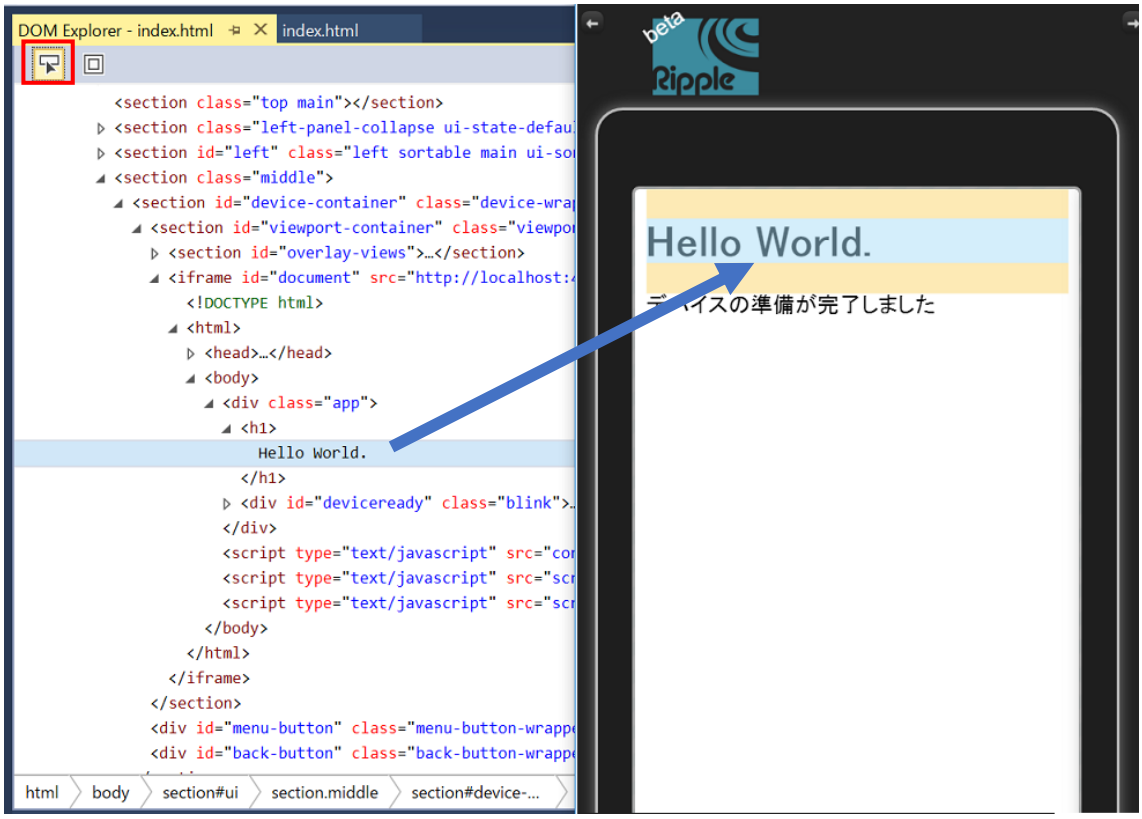
JavaScript コンソールは、Cordova アプリの起動時に表示されます。表示されない場合は、簡単に開くことができます。[デバッグ] をクリックし、[ウィンドウ] をポイントして、[JavaScript コンソール] をクリックするだけです。

`console.log` メソッドを使用してメッセージをログに記録しておき、このメッセージをコンソールに表示することができます。このコンソールを使用して、アプリケーションの実行中に JavaScript 関数を実行することも可能です。JavaScript コンソールの詳細についてはこちら (英語) で確認してください。

DOM (ドキュメント オブジェクト モデル) Explorer の使用

ページのレイアウト上の問題点の検出と修正には、DOM Explorer を使用します。ページのレンダリング時にページの構造をテストして、アプリの実行中に調整を加えます。

DOM Explorer を開くには、アプリの実行中に [デバッグ] をクリックし、[ウィンドウ] をポイントして、[DOM Explorer] をクリックします。DOM Explorer の詳細についてはこちらを確認してください。



プラットフォームに合わせたアプリの調整

Cordova アプリはクロスプラットフォームのアプリをシングルコードで行いますが、プラットフォームごとに異なる動作を実装することもできます。

具体的には **merges** フォルダーを利用します。このフォルダーについては、本ガイドの前半で取り上げました。ここでは実際にアプリにプラットフォーム名を表示してその機能を確認してみましょう。

アプリ動作の調整

1. ソリューション エクスプローラーで、**www** フォルダー、**scripts** フォルダーの順に展開します。
2. **platformOverrides.js** ファイルをダブルクリックして開き、以下のコードを追加します。

```
function showTitle() {
    document.getElementsByTagName('h1')[0].textContent = 'iOS';
}
```

3. ソリューション エクスプローラーで、**merges** フォルダーにある **android** サブフォルダーを展開します。

4. **android** フォルダーの **platformOverrides.js** ファイルをダブルクリックして開き、既に記述してあるコードの閉じ括弧の外側に以下のコードを追加します。

```
function showTitle() {
    document.getElementsByTagName('h1')[0].textContent = 'Android';
}
```

5. **index.js** ファイル で、`onDeviceReady` メソッドの最後に以下のコードを追加します。

```
showTitle();
```

6. 標準ツールバーの ソリューションプラットフォームの内容を [Android]、[iOS]と切り替えて Ripple シミュレーターでアプリを実行します。

アプリのタイトルがプラットフォームごとに変わるのを確認してください。

merges フォルダーをうまく利用していつでもコードをリファクタリングできるようになります。今回はプロジェクトに既定で含まれている **platformOverrides.js** を使用しましたが、開発者が作成した任意の js ファイルも CSS ファイルも同様に使用できます。

また、以降でプラグインの使用方法を紹介しますが、各プラグインの readme ファイルで、どのような種類のコードをデバイス固有に記述する必要があるかを確認できます。

プラグインによるハードウェアリソースへのアクセス

Cordova アプリは Web ブラウザー内で動作するアプリケーションとは異なり、「プラグイン」と呼ばれ

るネイティブコードで作られたインターフェースを介してカメラのフラッシュやバイブレータといったハードウェアリソースにアクセスすることができます。
Visual Studio 2015 を使用した Cordova アプリの開発ではこうしたプラグインの入手やインストール作業を GUI を使用して直感的に行えます。
たとえば、index.html に表示したボタンをクリックしてスマートフォンのバイブレーション機能を動作させるには以下の手順を行います。

プラグインを介したバイブレーション機能の利用

1. index.html の h1 タグの下の行にボタンを表示するための以下のタグを追加します。

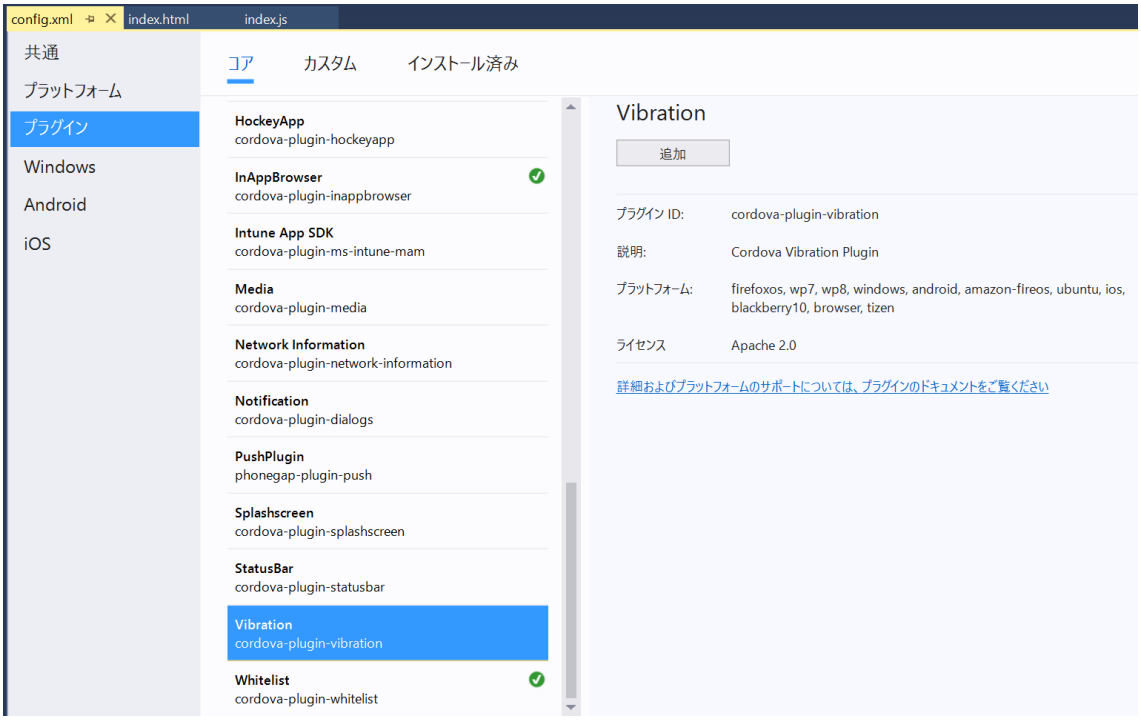
   ```
   <button id="vibrationButton">バイブレーションの実行</button>
   ```

2. index.js の onResume 関数定義の下に vibrationButton のイベントハンドラを定義するための setHandlers 関数を以下のように記述します。

   ```
   function setHandlers() {
       document.getElementById('vibrationButton').addEventListener('click', function () {
           //ここにプラグイン参照後にコードを記述します。
       });
   }
   ```

3. プラグインの参照を行います。
   ソリューションエクスプローラーで config.xml をダブルクリックして設定用画面を表示します。

4. [プラグイン] タブ内のプラグインの一覧から Vibration を選択し、画面右のパネルにある [追加] ボタンをクリックしてプラグインを追加します。



5. setHandlers 関数の addEventListener 内でプラグインの提供する vibrate メソッドを実行するよう、以下のようにコードを書き変えます。

```
function setHandlers() {
    document.getElementById('vibrationButton').addEventListener('click', function () {
        //2 秒間バイブレーションさせる
        navigator.vibrate(2000);
    });
}
```

6. onDeviceReady 関数の閉じ括弧 (}) の前の行に setHandlers 関数を呼び出す以下のコードを追加

```
setHandlers();
```

します。

7. 標準ツール バーの [Ripple – Nexus (Galaxy)] ボタンをクリックしてプロジェクトを実行します。

8. Ripple シミュレーターの画面な [バイブレーションの実行] ボタンが表示されるのでクリックし、画

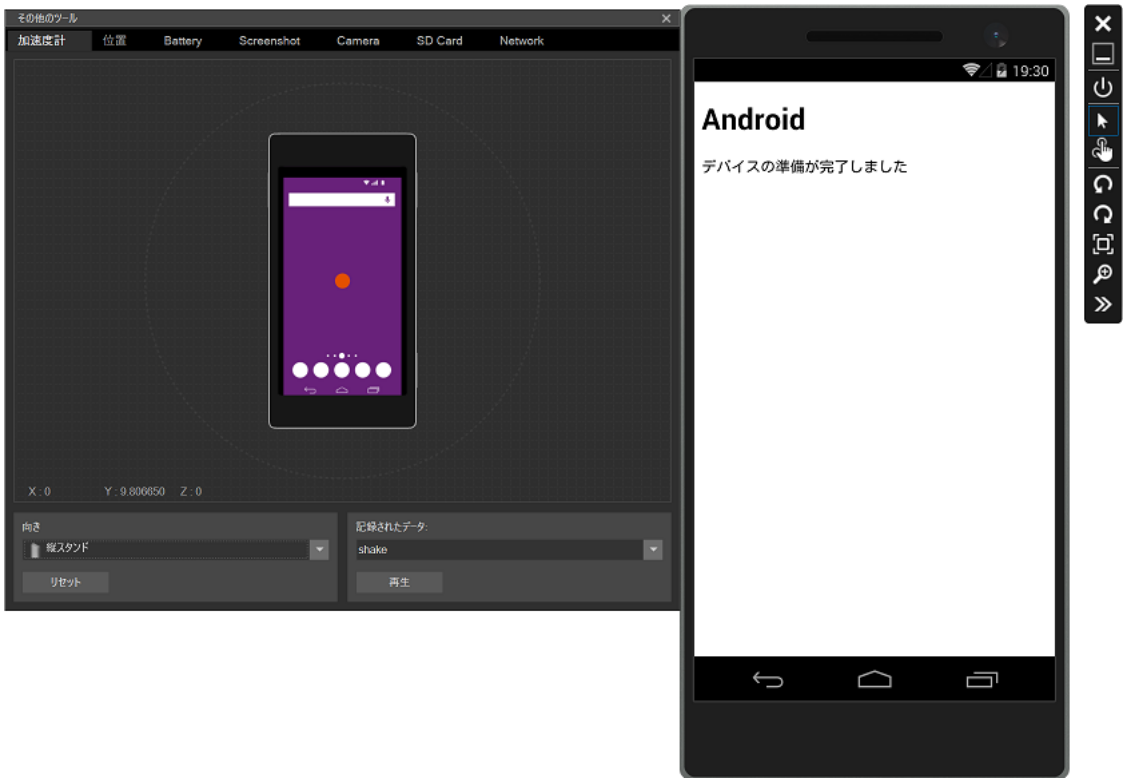
面が震えるのを確認します。

ここではプラグインを介したハードウェアリソースへのアクセス方法について紹介しました。
同様の方法でプラグインを使用し、Web ブラウザー上のアプリケーションではアクセスできない、プラットフォームのさまざまな情報にアクセスすることができます。

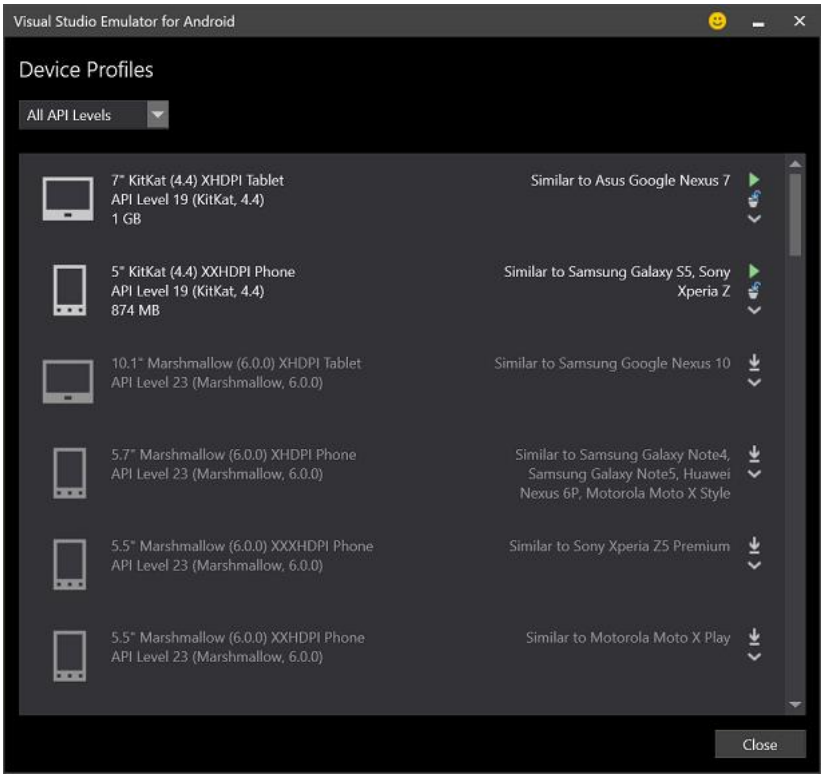
エミュレーターでのアプリの実行

ここまで Google Chrome の拡張である Apache Ripple でアプリを実行してきましたが、Visual Studio 2015 は Hyper-V ベースで動作するより高度なエミュレーターも提供しています。
これらのエミュレーターを起動するには、標準ツールバーのエミュレーターの選択ドロップボックスより「VS」から始まるものを選択します。

Hyper-V ベースのエミュレーターは、より実際のスマートフォンに近い実行環境を提供します。



Visual Studio 2015 に同梱されている Hyper-V ベースのエミュレーターのイメージは 2 つですが、Visual Studio Emulator for Android を使用してイメージをダウンロードして追加することができます。



Visual Studio Emulator for Android を起動するには Windows 10 のアプリケーションの一覧からアイコンをクリックするか、Windows 10 の検索ボックスで検索してください。

アプリデータの保存

Cordova アプリの扱うデータの保存は、HTML5 の標準機能である localStorage や IndexedDB 、File API といった複数の方法を使用することができます。
しかし、これらの機能が使用するのはデバイス内の記憶領域であるためデバイスにまたがる情報の共有は行えません。
たとえば、スマートフォンとタブレットで同じユーザーが同じアプリを使用してもアプリのデータはデバイスごとに保存されるので同じデータを参照することはできません。
デバイスやユーザーにまたがるデータの共有を行うには、クラウド上の記憶領域にデータを保存します。

クラウドへのデータの保存

Azure Mobile Apps プラグインを使用すると Cordova アプリの使用するデータをクラウド上に保存し、複数の異なるデバイス上のアプリから参照することができます。
ここからは電話番号のリストをクラウド上に保存する方法について、簡単なサンプルを作成しながら紹介します。

Microsoft Azure アカウントの作成

Azure Mobile Apps を使用するには Microsoft Azure のアカウントが必要です。

以下のページの右上にある [無料アカウント>] ボタンをクリックしてアカウントの作成を開始します。

- Microsoft Azure: クラウド コンピューティング プラットフォームとサービス

遷移先のページで [無料で始める] ボタンをクリックすると、Microsoft アカウントの認証が行われ、Azure アカウントの登録フォームが表示されますので内容に従い無料アカウントを作成してください。

Azure アカウントの作成か完了したら Azure ポータルにログインし、ダッシュボード画面が表示されることを確認しましょう。

Azure ポータルには以下の URL からアクセスすることができます。

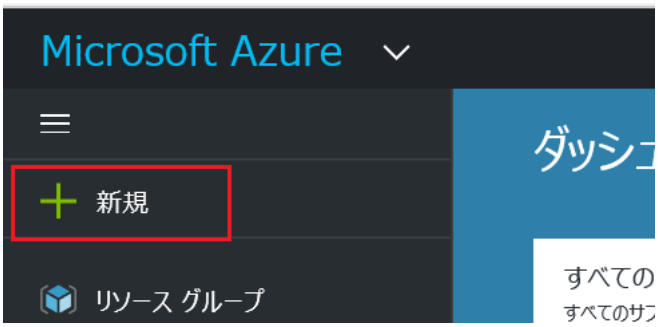
https://portal.azure.com/

Azure Mobile Apps の開始

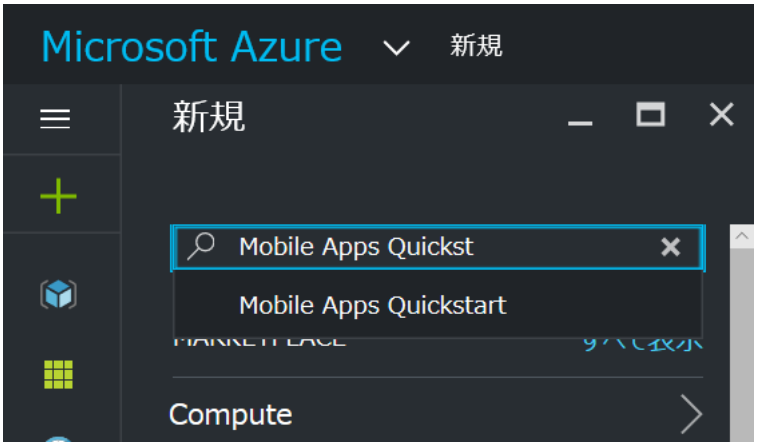
Cordova アプリから Azure の記憶領域に接続できるようにするために、Azure Mobile Apps を構成します。

具体的な手順は以下のとおりです。

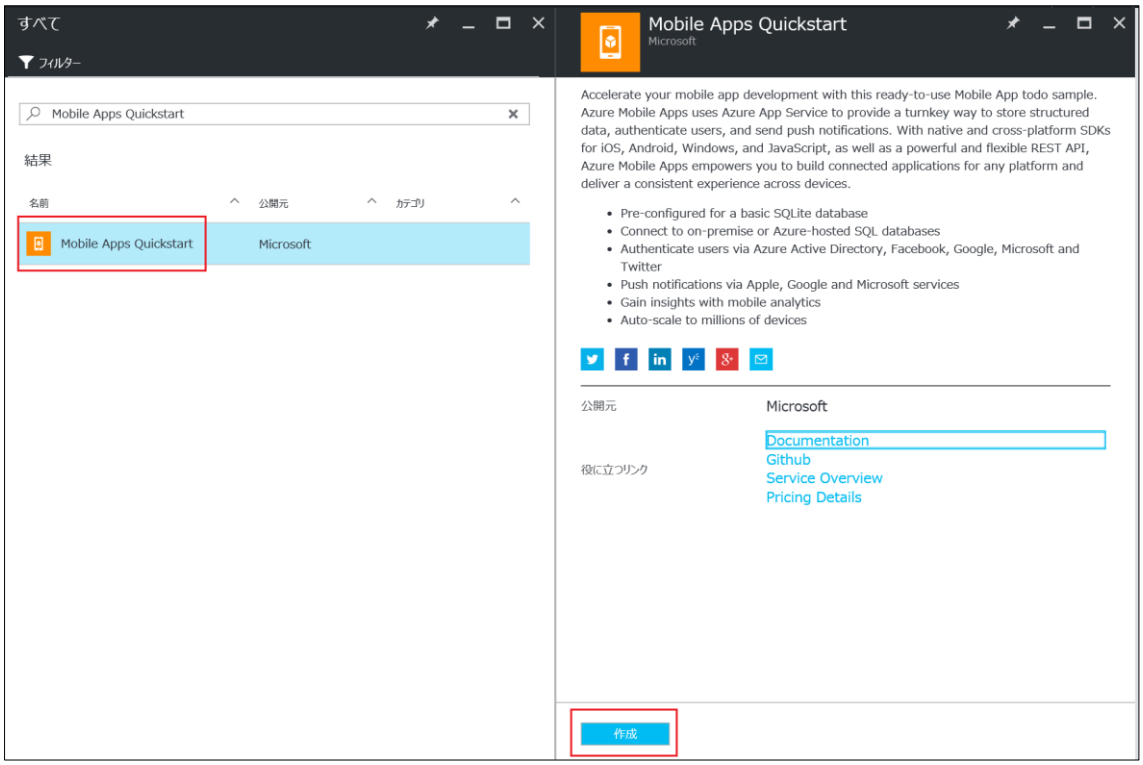
1. Azure ポータルにアクセスし、ダッシュボードの左上にある [+ 新規] ボタンをクリックします。



2. 右側に選択したメニューのパネルが表示されるので、[Marketplace を検索] と表示されたボックスに「Mobile Apps Quickstart」と入力して検索します。



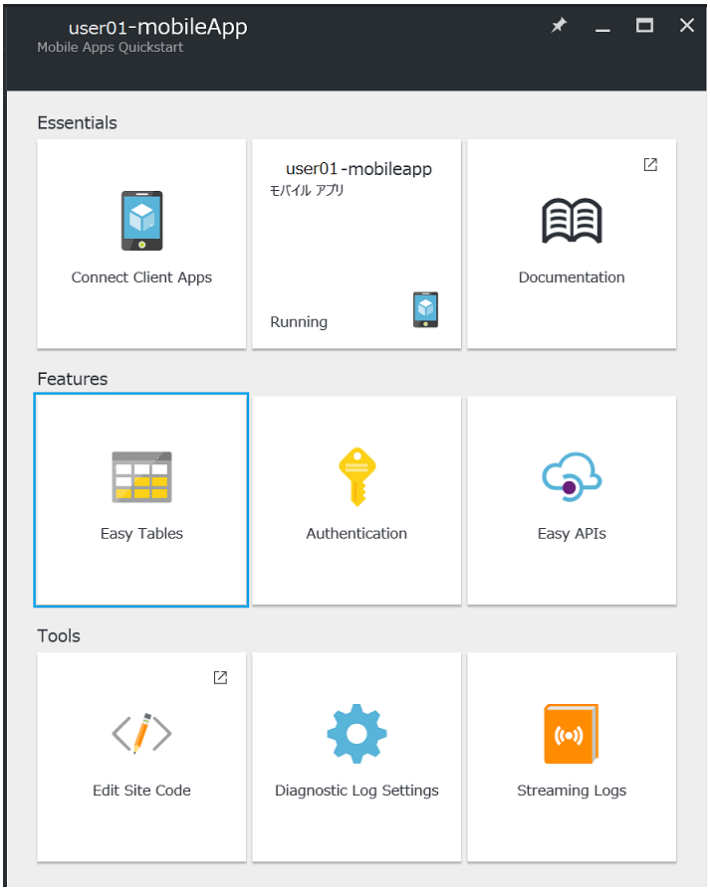
3. 検索結果に表示された Mobile Apps Quickstart をクリックすると、右側に Mobile Apps Quickstart のパネルが表示されるので、同パネル内の [作成] ボタンをクリックします。



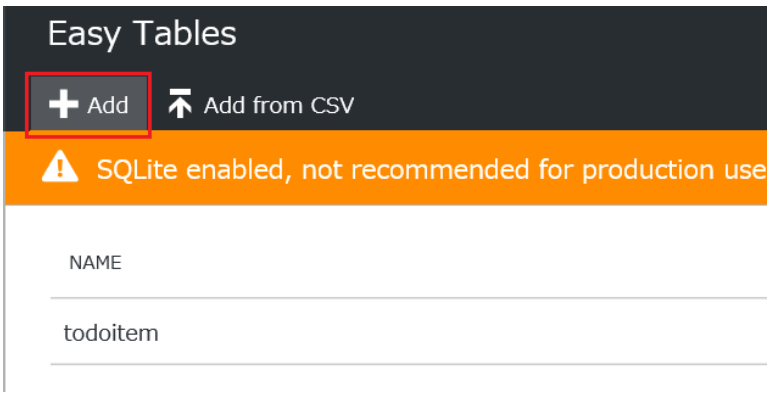
4. Mobile Apps Quickstart の設定画面が表示されるので、[アプリ名]、[サブスクリプション]、[リソースグループ]、[App Service プラン/場所] を適宜設定し [作成] ボタンをクリックしデプロイを開始

します。

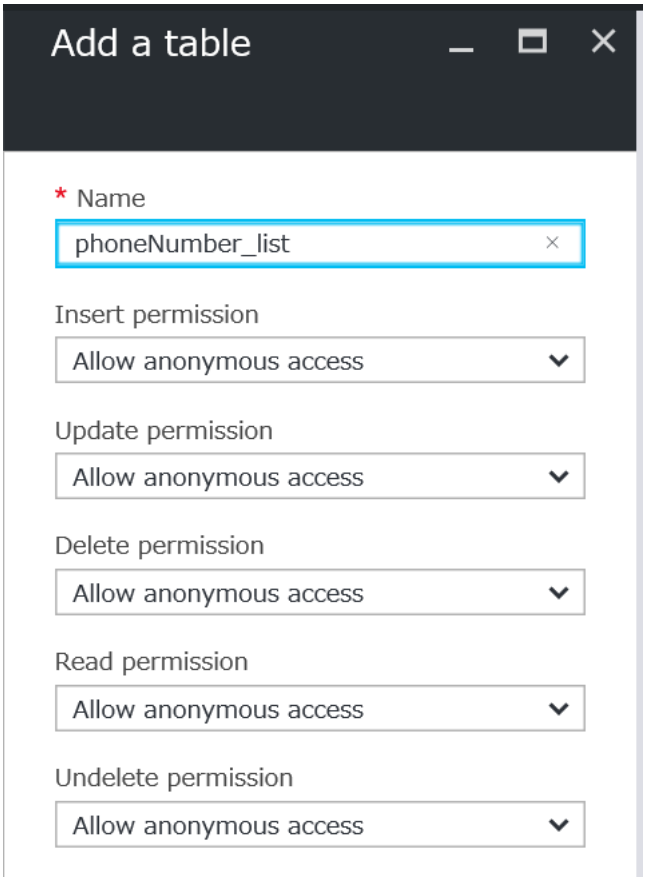
5. デプロイが完了すると Mobile Apps の設定画面が表示されるので、[Easy Tables] のタイルをクリックします。



6. Easy Tables のテーブルの一覧が表示されるので、メニューバーの [+Add]をクリックします。



7. テーブルの作成画が面表示されるので［* Name］に phoneNumber_List と入力しその他は既定のまま［OK］ボタンをクリックします。

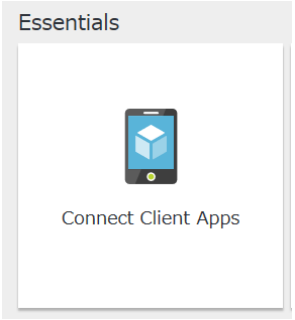


以上で Azure 上にデータを保存するための準備ができました。

備考 ：Azure Mobile Apps への接続コード入りの Cordova アプリの入手

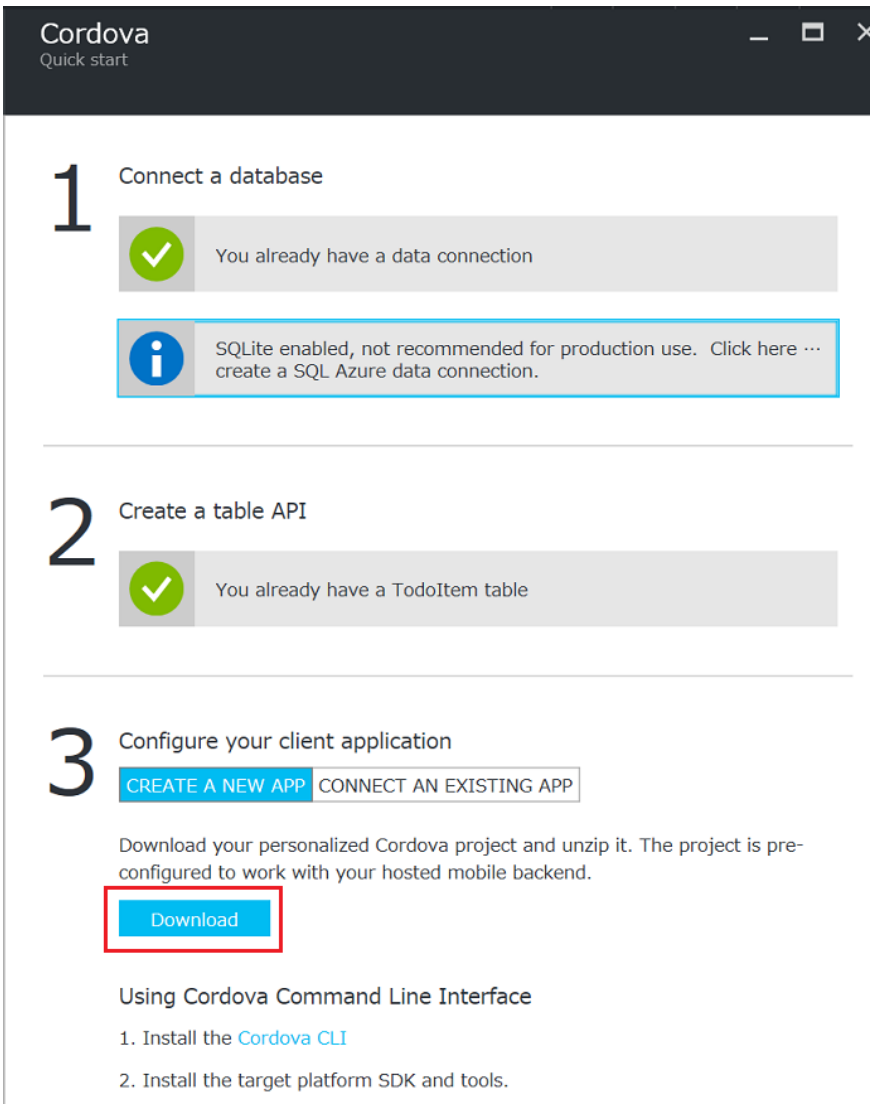
今回の手順では使用しませんが、Azure Mobile Apps 上のテーブルに対して接続と CRUD (Creat:追加、Read:読み取り、Update:更新、Delete:削除) 処理があらかじめ記述された Cordova アプリケーションのソリューションをダウンロード入手することができます。
ダウンロードの手順は以下の通りです。

1. ダッシュボード画面から作成した Mobile Apps のタイルをクリックします。
2. 機能の一覧がタイル表示されるので [Connect Client Apps] をクリックします。



3. [Quick Start] パネルが表示されるので [Cordova] をクリックします。

4. ソリューションが生成されるので、[Download] ボタンをクリックして zip ファイルをダウンロードします。



なお、このソリューションに含まれるソースは jQuery の使用を前提で書かれているのでコードを読み解くには jQuery の知識が必須となります。

Cordova アプリからの Azure Mobile Apps への接続

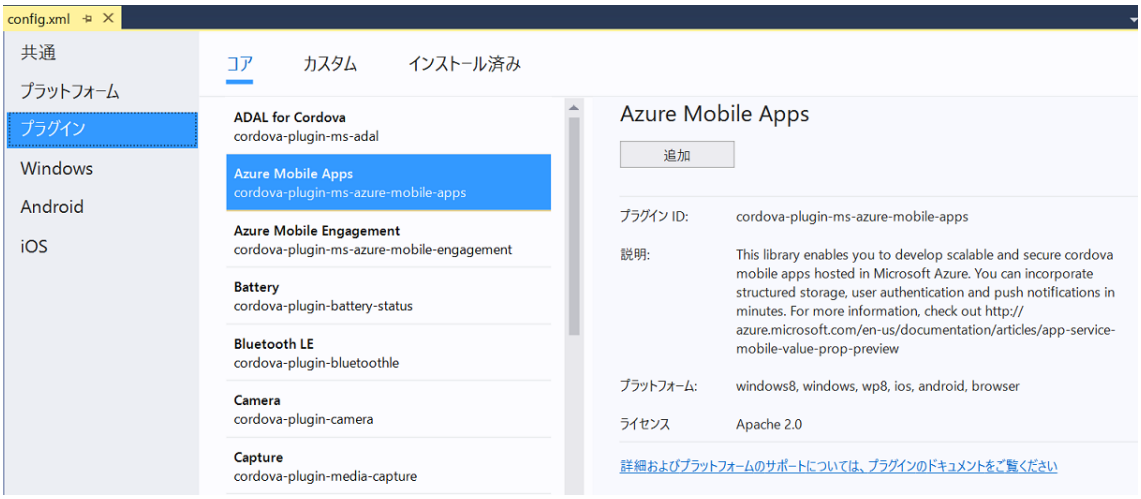
Visual Studio 2015 に同梱される Tools for Apache Cordova には Azure Mobile Apps 用のプラグインが用意されており、これを使用して Cordova アプリから Azure Mobile Apps に接続することができます。

Visual Studio 2015 で Azure Mobile Apps 用プラグインを参照する手順は以下のとおりです。

1. プラグインの参照を行います。

ソリューションエクスプローラーで config.xml をダブルクリックして設定用画面を表示します。

2. [プラグイン] タブ内のプラグインの一覧から Vibration を選択し、画面右のパネルにある [追加] ボタンをクリックしてプラグインを追加します。



以上でプラグインの参照は完了です。

ここからは実際に Azure Mobile Apps に接続してデータを表示するためのコードを記述していきます。

Cordova アプリからの Mobile Apps テーブルの CRUD 処理

Azure Mobile Apps 用プラグインを参照することにより Azure Mobile Apps に接続するための WindowsAzure 名前空間が使用できるようになります。

この名前空間を使用して Azure Mobile Apps 内に作成した phoneNumber_List テーブルのインスタンスを生成するには以下のように記述します。

```
//mobileApps Client のインスタンスを生成する
var client = new WindowsAzure.MobileServiceClient('https://mobileWebApps の名前.azurewebsites.net');
//テーブルのインスタンスを生成する
var ItemTable = client.getTable(' phoneNumber_List');
```

CRUD 処理は以下のように記述できます。

・データの追加

```
//テーブルにアイテムを追加する
let item = { フィールド名1 : "値1", フィールド名2 : "値2" };
ItemTable .insert(item).then(後処理, handleError);
```

・読み取り

```
//テーブルのアイテムの一覧を取得する
ItemTable
    .where({ complete : false }) //フィールド名 : 値
    .read() // 結果の読み取り
    .then(createItemList, handleError); //Promise の処理

//引数として渡されたアイテムの配列を処理する
function createItemList(items){
    let length = items.length;
    for (let i = 0; i < length; i++) {
        console.log(items[i].プロパティ名)
    }
}

//エラー処理
function handleError(error) {
    var text = error + (error.request ? ' - ' + error.request.status : '');
    console.log(text);
}
```

・**更新処理**

```
//更新処理
ItemTable
     .update({ id: item の id, text: newText })  // id: システムがつけた id, フィールド名: 値
     .then(後処理, handleError);
```

・**削除処理**

```
//削除処理
ItemTable.del({ id: item の Id })   // id: システムがつけた id
   .then(後処理,, handleError);
```

Azure Mobile Apps を使用したアプリの開発

Azure Mobile Apps の phoneNumber_List テーブルに名前と電話番号を記録する以下のような Cordova アプリを作成します。

この作業は、Visual Studio 2015 の Tools for Cordova が生成する既定のプロジェクトを使用します。

UI の準備

1. index.html の h1 タグ内のテキストを「電話帳」に書き換えます。

2. h1 タグの下の行に以下のタグを記述するか貼り付けます。

```
<div style="float:left;">
    <div>氏名</div>
    <input id="personName" type="text" value="" />
</div>
<div style="float:left;">
    <div>電話番号</div>
    <input id="phoneNumber" type="text" value="" />
</div>
<div style="float:left;">
     <br />
    <button id="addBtn">+</button>
</div>
<div style="clear:both;"></div>
<hr />
<div>
<table id="phoneNumberList">
</table>
<div id="resultMessage"></div>
```

JavaScript の記述

1. index.js の “use strict” の下の行に以下の変数を記述します。

```
"use strict";


//Azure Mobile Apps テーブルのインスタンスを格納する変数
let ItemTable;
//UI コントロールのインスタンスを格納する変数
let ctrl_personName,
      ctrl_phoneNumber,
      ctrl_itemsTable,
      ctrl_resultMessage;
```

2. index.js の onDeviceReady 関数の閉じ中括弧(}) の前に以下のコードを記述します。
   なお、Mobile Apps の URL は実際に作成したものに書き換えてください。

```
//UI コントロールのインスタンスを取得
setControls();
//イベントハンドラを設定
setHandlers();
//mobileApps Client のインスタンスを生成する
let client = new WindowsAzure.MobileServiceClient('Mobile Apps の URL');
//テーブルのインスタンスを生成する
ItemTable = client.getTable('phoneNumber_list');
//レコードを列挙する
enumListItem();
```

3. onDeviceReady 関数で呼び出している関数を定義します。
   onResume 関数を定義している下の行に以下のコードを追加します。

```
//コントロールのインスタンスを変数にセット
function setControls() {
   ctrl_personName = document.getElementById('personName');
   ctrl_phoneNumber = document.getElementById('phoneNumber');
   ctrl_itemsTable = document.getElementById('phoneNumberList');
   ctrl_resultMessage = document.getElementById('resultMessage');
}

//イベントハンドラをセット
function setHandlers() {
   document.getElementById('addBtn').addEventListener('click', ()=> {
       addListItem(ctrl_personName.value, ctrl_phoneNumber.value);
   });
}

//アイテムの追加
function addListItem(parsonName, number) {
    let item = { 'name': parsonName, 'phoneNumber': number };
    //Azure Mobile Apps テーブルにレコードを追加
    ItemTable.insert(item).then((newItem)=> {
        ctrateListItem(ctrl_itemsTable, newItem);
        showMessage(parsonName + ' の情報を追加しました');
        clearInput();
    }, handleError);
}

//テーブルのアイテムの一覧を取得
function enumListItem() {
    ItemTable.read()   // Read the results
       .then(createItemList, handleError);
}
```

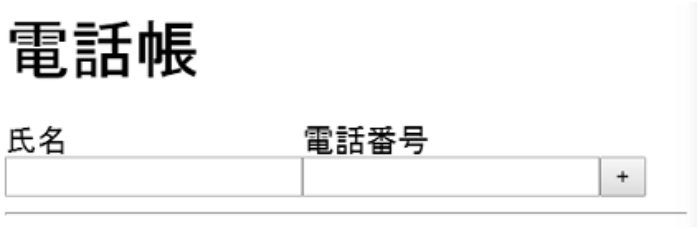
```
//引数として渡されたアイテムの配列を処理
function createItemList(items) {
    let length = items.length;
    for (let i = 0; i < length; i++) {
        ctrateListItem(ctrl_itemsTable,items[i])
    }
}

//表のコントロールを生成
function ctrateListItem(parentElement, itemData) {
    let itemRow = document.createElement('tr');
    let nameColumn = document.createElement('td');
    let phoneColumn = document.createElement('td');
    let buttonColumn = document.createElement('td')
    let nameTextBox = document.createElement('input');
    let numberTextBox = document.createElement('input');
    nameTextBox.setAttribute('type', 'text');
    nameTextBox.value = itemData.name;
    numberTextBox.value = itemData.phoneNumber;
    let id = itemData.id;
    //名前のボックスが変更された際の処理
    nameTextBox.addEventListener('change', ( (editId)=> {
        return function() {
            //Azure Mobile Apps テーブルの name フィールドを更新
            ItemTable.update({ id: editId, name: this.value });
           showMessage('名前情報を更新しました。');
        }
    })(id));
    //電話番号ボックスが変更された際の処理
    numberTextBox.addEventListener('change', ( (editId)=> {
       return function () {
           //Azure Mobile Apps テーブルの phoneNumber フィールドを更新
           ItemTable.update({ id: editId, phoneNumber: this.value });
           showMessage('電話番号情報を更新しました。');
       }
   })(id));
```

```
    let button = document.createElement('button');
    button.textContent = 'X';
    //削除ボタンをクリックされた際の処理
    button.addEventListener('click', ( (deleteId)=> {
        return function () {
            let tableElement = this.parentElement.parentElement.parentElement;
            let trElement = this.parentElement.parentElement;
            //Azure Mobile Apps テーブルからレコードを削除
            ItemTable.del({ id: deleteId }).then(()=> {
                tableElement.removeChild(trElement);
                showMessage('レコードを 1 件削除しました。');
            }, handleError);
        }
    })(id));

    //UI に生成したコントロールを追加
    nameColumn.appendChild(nameTextBox);
    phoneColumn.appendChild(numberTextBox);
    buttonColumn.appendChild(button);
    itemRow.appendChild(nameColumn);
    itemRow.appendChild(phoneColumn);
    itemRow.appendChild(buttonColumn);
    parentElement.appendChild(itemRow);
}

//入力 UI をクリア
function clearInput() {
    ctrl_personName.value = '';
    ctrl_phoneNumber.value = '';
}

 //メッセージを表示
 function showMessage(messageText) {
     ctrl_resultMessage.textContent = messageText;
 }
```

```
//エラー処理
function handleError(error) {
    var text = error + (error.request ? ' - ' + error.request.status : '');
    showMessage(text);
}
```

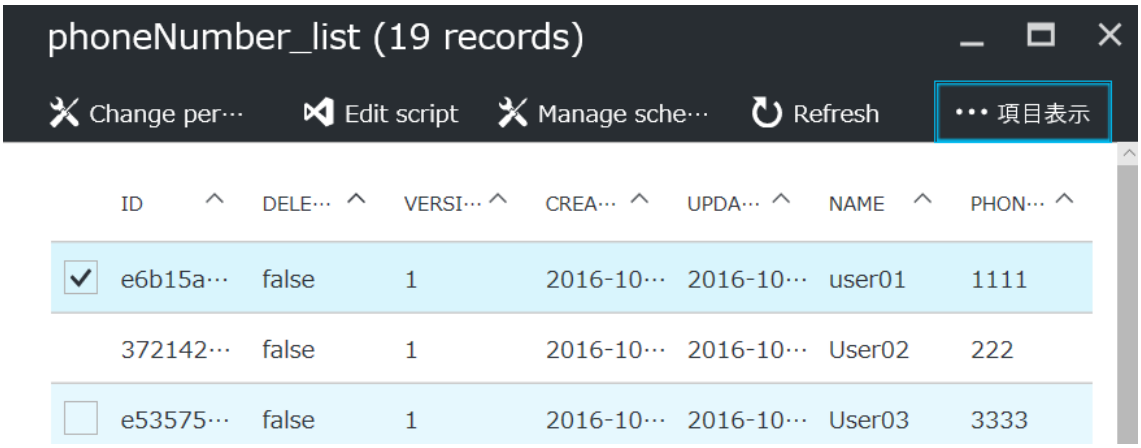
4. 標準ツール バーの [Ripple – Nexus (Galaxy)] ボタンをクリックしてプロジェクトを実行します。

5. Ripple の画面内にアプリの UI が表示されるので、[氏名]と[電話番号]に任意の情報を入力し、[+] ボタンをクリックします。



6. リストに入力した情報が追加されるのを確認し、同様の動作を複数回繰り返します。

7. Azure のダッシュボード画面から Mobile Apps の phoneNumber_list の内容を確認し、入力したデータがレコードとして追加されていることを確認します。



その他、リスト横の [X] ボタンをクリックするとレコードの削除されること(※データ上は DELETED フィールドの値が true に変化します)、リスト内のデータを書き変えると、フォーカスが変化したタイミングで Azure 上のデータも更新されることを確認してください。

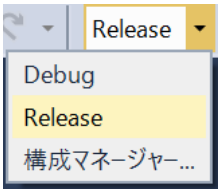
アプリ パッケージの作成

作成した Cordova アプリを、各プラットフォーム向けのアプリストアに提出するためのパッケージを作成します。

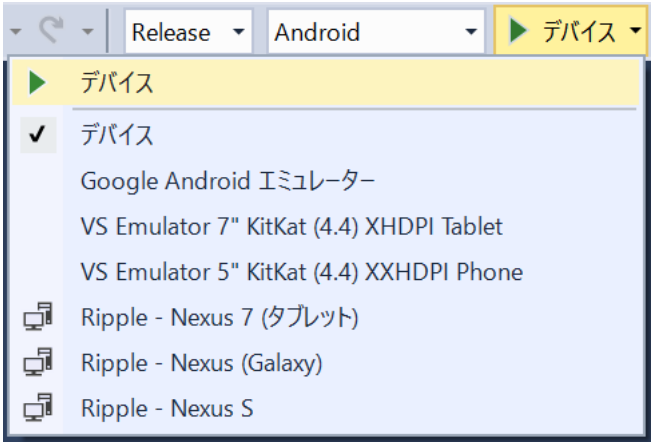
Android 用アプリパッケージの作成

Visual Studio 2015 で Android 用アプリのパッケージである apk ファイルを出力するには以下の手順を実行します。

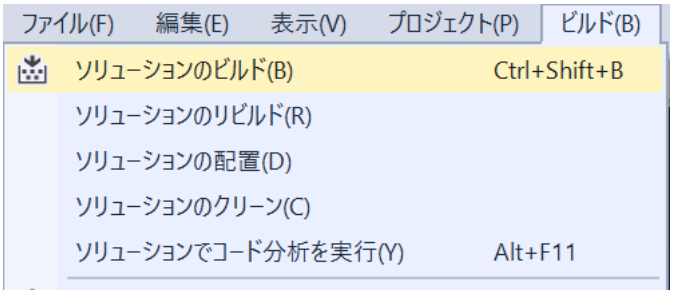
1. 標準ツール バーにあるソリューション構成ドロップダウンリストから [Release] を選択します。



2. 標準ツール バーにあるエミュレーター選択ドロップダウンリストから[デバイス]を選択します。



3. メニュー[ビルド] から [ソリューションのビルド] を選択するとビルドが開始されます。



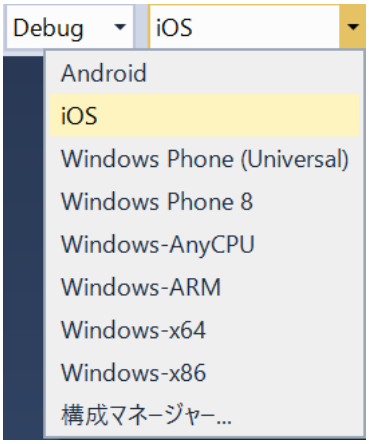
4. ビルドが完了したら [ソリューションエクスプローラー]でプロジェクトを右クリックし、表示されたコンテキストメニューから [エクスプローラーでフォルダーを開く] を選択します。
5. 作業しているプロジェクトのルード フォルダーが表示されます。
   apk ファイルは、プロジェクト ルートフォルダ下の bin¥Android¥Release フォルダーに出力されています。

iOS アプリ用 Xcode プロジェクトの生成

Windows 環境では iOS アプリのコンパイルやパッケージを行うことができません。
しかしながら、Mac 上にエージェントプログラムをインストールし、Visual Studio 2015 からリモートでコンパイルを行うことができます。この方法についての詳しい手順は、こちらの資料 (英語) を参照してください。

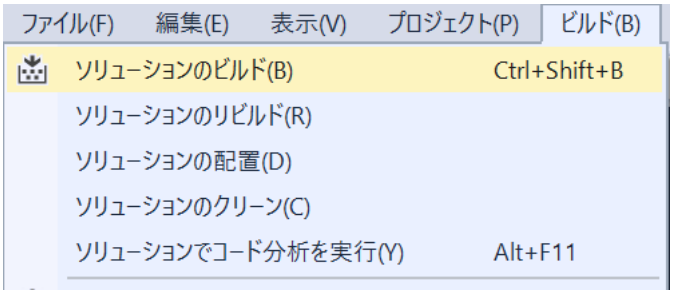
また、Visual Studio 2015 が出力した iOS アプリのプロジェクトを Mac 上の Xcode でコンパイルすることにより iOS 用アプリのパッケージも作成することができます。
Visual Studio 2015 で iOS アプリの Xcode プロジェクトを出力するには以下の手順を実行します。

1. 標準ツール バーにある[ソリューションプラットフォーム]ドロップダウンリストから [iOS] を選択します。



2. 標準ツール バーにあるエミュレーター選択ドロップダウンリストから[Ripple -]のどれかを選択します。

3. メニュー[ビルド] から [ソリューションのビルド] を選択するか、デバッグ実行するとビルドが開始されます。



ビルド中に iOS 用のプラットフォーム設定が追加されます。

```
1>  ------    language: ja-JP
1>  ------ プラットフォームを追加しています: ios
1>MDAVSCLI : warning : Applications for platform ios can not be built on this OS - win32.
1>  Executing "before_platform_add"  hook for all plugins.
1>  No version supplied. Retrieving version from config.xml...
```

4. ビルドが完了したら [ソリューションエクスプローラー]でプロジェクトを右クリックし、表示されたコンテキストメニューから [エクスプローラーでフォルダーを開く] を選択します。
5. 作業しているプロジェクトのルード フォルダーが表示されます。
   Xcode 用プロジェクトのファイル一式は、プロジェクト ルートフォルダ下の platforms¥ios フォルダーに出力されているので、これを Mac 側にコピーして Xcode でコンパイルすることができます。

次の手順

これで、最初のクロスプラットフォーム モバイル アプリをビルドできました。これで着手はできましたが、学ぶべきことはまだたくさんあります。

HTML、JavaScript、Visual Studio を使用してモバイル アプリのビルドを続けるための次の手順については、以下を参照してください。

プラグイン レジストリの調査

Cordova プラグインの検索 (英語)。

ヒント: 構成デザイナーの [プラグイン] タブに表示されないプラグインでも使用できます。詳細 (英語)。

Bower を使用したプロジェクトへのパッケージの追加

Bower を使用したパッケージの追加 (英語)。

Visual Studio デバッガーをうまく利用する方法

デバッガーの基本事項。

既存の JavaScript フレームワーク

インターネットで検索してください。さまざまなところで見つかります。たとえば、AngularJS や Ionic があります。

Mac と iOS エミュレーターによるアプリの実行、または iOS デバイスでのアプリの実行

セットアップ ガイド: Visual Studio Tools for Apache Cordova プロジェクトで iOS モバイル デバイスをターゲットにする (英語)。

Azure バックエンドに取り組む – 無料トライアル

クラウド サービスへの接続 (英語)。

TypeScript – プロジェクトでの完全サポート

TypeScript の使用 (英語)。