Office 評価ガイド

Office 用アプリによるソリューションの開発と展開

目次

[はじめに 6](#_Toc331149376)

[Office 用アプリの概要 7](#_Toc331149377)

[Office 用アプリとは 7](#_Toc331149378)

[活用シナリオ 9](#_Toc331149379)

[Bing Maps Office 用アプリの利用 9](#_Toc331149380)

[Office 用アプリの種類 16](#_Toc331149381)

[作業ウィンドウ アプリ 16](#_Toc331149382)

[コンテンツ アプリ 17](#_Toc331149383)

[メール アプリ 17](#_Toc331149384)

[各 Office 製品がサポートする Office 用アプリの種類 26](#_Toc331149385)

[Office 用アプリのアーキテクチャー 27](#_Toc331149386)

[構成概要 27](#_Toc331149387)

[Web 標準準拠 31](#_Toc331149388)

[セキュリティ 33](#_Toc331149389)

[Office 用アプリによるソリューション開発 40](#_Toc331149390)

[Office 用アプリ開発概要 40](#_Toc331149391)

[必要環境 40](#_Toc331149392)

[開発手順 42](#_Toc331149393)

[Hello World の開発 42](#_Toc331149394)

[Hello World の実装 42](#_Toc331149395)

[Hello World の展開 44](#_Toc331149396)

[Visual Studio 2012 による Office 用アプリの開発 58](#_Toc331149397)

[プロジェクト テンプレート 59](#_Toc331149398)

[ドキュメントからデータの取得 61](#_Toc331149399)

[外部サービスとの連携 69](#_Toc331149400)

[ドキュメントへのデータ設定 78](#_Toc331149401)

[データのバインド 86](#_Toc331149402)

[Office 用アプリの再利用性 96](#_Toc331149403)

[各 Office 製品に対応する機能範囲 98](#_Toc331149404)

[Office 用アプリの展開 100](#_Toc331149405)

[Office 用アプリ展開概要 100](#_Toc331149406)

[Office 用アプリの展開の種類 101](#_Toc331149407)

[共有フォルダー 101](#_Toc331149408)

[内部ディレクトリ 102](#_Toc331149409)

[Office ストア 103](#_Toc331149410)

[SharePoint Online による Office 用アプリの展開 104](#_Toc331149411)

[SharePoint Online への登録 104](#_Toc331149412)

[クライアントの設定 104](#_Toc331149413)

[まとめ 105](#_Toc331149414)

本資料で使用する製品の名称と略称を記述します。

* Microsoft Excel 2013 (Excel 2013)
* Microsoft Outlook 2013 (Outlook 2013)
* Microsoft Word 2013 (Word 2013)
* Microsoft Project 2013 (Project 2013)
* Microsoft SharePoint Server(SharePoint Server)
* Microsoft Exchange Server (Exchange Server)
* Microsoft Office 365 (Office 365)
* Microsoft SharePoint Online (SharePoint Online)
* Microsoft Exchange Online (Exchange Online)
* Microsoft Windows 8 (Windows 8)
* Microsoft Windows 7 (Windows 7)
* Microsoft Visual Studio 2012 (Visual Studio 2012)
* Microsoft Office Web Apps (Office Web Apps)
* Visual Basic for Application (VBA)
* Microsoft Silverlight (Silverlight)
* Microsoft Windows Azure Platform (Windows Azure)

著作権

このドキュメントに記載されている情報 （URL 等のインターネット Web サイトに関する情報を含む）

は、将来予告なしに変更することがあります。別途記載されていない場合、このソフトウェアおよび関連するドキュメントで使用している会社、組織、製品、ドメイン名、電子メールアドレス、ロゴ、人物、場所、出来事などの名称は架空のものです。実在する名称とは一切関係ありません。お客様ご自身の責任において、適用されるすべての著作権関連法規に従ったご使用を願います。マイクロソフトは、このドキュメントに記載されている内容に関し、特許、特許申請、商標、著作権、またはその他の無体財産権を有する場合があります。別途マイクロソフトのライセンス契約上に明示の規定のない限り、このドキュメントはこれらの特許、商標、著作権、またはその他の無体財産権に関する権利をお客様に許諾するものではありません。

©2012 Microsoft Corporation. All rights reserved.

Windows 8、Windows 7、Excel、Word、Outlook、Project、SharePoint、Exchange、Office 365、Office Web Apps、Visual Studio、Silverlight、Windows Azure、Bing Maps、Internet Explorer は、米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。記載されている会社名、製品名には、各社の商標のものもあります。

記載されている会社名、製品名には、各社の商標のものもあります。

# はじめに

Office 用アプリは、次期 Office にユーザーが必要とする機能を追加できる新しいテクノロジーです。開発において、Web 標準技術を採用することにより、開発者は、今まで習得してきた技術をもちいて、豊富な既存資産を流用でき、ユーザーに対して有用な機能を高い生産性で提供できます。モバイルデバイスでの動作もサポートされるため、マルチデバイス対応も容易です。ユーザーにとっては、使い慣れた Office ドキュメントに高機能な外部サービスや社内システムが統合されたより便利なシステムを利用でき、業務の生産性を向上できます。また、Office 用アプリが豊富に公開されている Office ストアから、業務にあったものをダウンロードしてすぐに業務に活用することもできます。

本ドキュメントは、Office マクロでの開発を経験したことのある IT 管理者や Web 開発を経験したことのある開発者の方向けに構成されています。Office 用アプリを理解し、Office 用アプリによるソリューションを開発し、組織に展開するまでの一連の手順を以下の流れで説明しています。

* Office 用アプリの概要

Office 用アプリの開発に必要となる基礎知識の習得を行います。

* Office 用アプリによるソリューション開発

実際に、サンプルプログラムのコーディングを行い、基本的な Office 用アプリ開発手法を習得します。

* Office 用アプリの展開

開発した Office 用アプリを組織へ展開する基本的な方法を習得します。

これらの Office 用アプリ開発における基礎技術の習得をとおして、ユーザーの業務生産性を向上させるセキュアなソリューションを短期に開発するための入門書として、本ドキュメントをご利用ください。

# Office 用アプリの概要

Office 用アプリによるソリューション開発のため、開発の前提知識として必要となる Office 用アプリの概要を説明します。これらの理解により、開発者は Office 用アプリによるソリューション開発を円滑に進められ、組織内のユーザーへ業務の生産性を上げるソリューションを提供できます。

* Office 用アプリとは

Office 用アプリの概要および開発者、IT 管理者、ユーザーに対する利点を説明します。

* 活用シナリオ

標準で利用可能な Office 用アプリを例にとって、利用シーンを説明します。

* Office 用アプリの種類

3つの種類の Office 用アプリおよび 各 Office 製品がサポートする Office 用アプリの種類を説明します。

* Office 用アプリのアーキテクチャー

Office 用アプリの構成概要、 Web 標準準拠、セキュリティについて説明します。

## Office 用アプリとは

Office 用アプリとは Web 標準技術を利用した開発が行えるように設計された次期 Office の新機能であり、Office 製品とシームレスに連携する Office ドキュメントベースのアプリケーション開発ができます。

開発者にとって Office 用アプリによるソリューション開発は HTML 5, XML, CSS 3 などの既存の Web 標準技術を利用して開発を行えるため、新たな技術の取得を必要としません。また、一般に公開された jQuery などの豊富なライブラリや開発者が今まで蓄積してきた既存資産を活用し、短期に開発を行うこともできます。さらには、REST や Web サービスによりインターフェイスが公開されている外部サービスや組織内の業務システムとの親和性も高いため、それらのサービスやシステムとシームレスに連携した統合性の高い Office ドキュメントベースのアプリケーションを開発できます。開発した Office 用アプリの一部は、モバイルデバイスでも動作するため、クロスデバイス対応も容易になります。セキュリティの観点では、サンドボックスと Permission モデルにより、ランタイムレベルでのセキュリティを確保しているため、一定以上の堅牢なソリューションを開発できます。

IT 管理者にとって Office 用アプリはマクロやアドインと比較し、SharePoint Online や SharePoint、ファイル共有を利用した集中管理が可能なため、Office 用アプリの配布やアップグレードを容易に行うことができます。そのため、ユーザーへの展開およびアップグレードの計画と実施にかかるコストを削減でき、さらに運用の簡素化が可能です。また、Office ストアで提供されているサードパーティ製の Office 用アプリも同様に組織へ容易に展開できます。ユーザーのOffice ストアの利用は、グループ ポリシーにより制限することもできるため、組織における統制を行うこともできます。

ユーザーにとって Office 用アプリは次期 Office と統合されたアプリケーションとして利用できます。使い慣れた Office 内に統合されたシームレスなアプリケーションとして利用できるため、多くのユーザーが利用している Office ドキュメントを使った既存業務をより効率的に実施できます。また、ユーザーは、Office ストアの豊富な選択枠のなかから、必要に応じて、自身の業務にあった Office 用アプリを入手し利用できるため、個々の業務生産性を向上させることができます。

開発者、IT 管理者、ユーザーにとって Office 用アプリによるソリューション開発は、それぞれの利点があります。組織におけるソリューションの開発で Office 用アプリを採用することにより、従来のマクロやアドインと比較し、開発生産性が高く、容易にセキュアなソリューションを開発することができ、管理面でも集中管理による管理の簡素化ができます。そしてなにより、ユーザーにとって Office 用アプリは次期 Office 内で動作するシームレスなアプリケーションとして利用でき、業務をより効率化できます。

## 活用シナリオ

ユーザーがどのように Office 用アプリを利用するかを理解するために、Bing Maps Office 用アプリの利用手順を紹介します。Outlook Web App とシームレスに連携する Office 用アプリを確認し、Office 用アプリによる業務の効率化を確認します。

### Bing Maps Office 用アプリの利用

今回は、Outlook 2013 およびOutlook Web Apps上で動作する Bing Maps Office 用アプリの環境を例にご紹介します。なお、Bing Maps Office 用アプリの利用にあたって、Exchange Online 側で特別な設定、カスタマイズは必要ありません。

以降の手順を実施するためには以下のアカウントが必要となります。あらかじめご用意ください。

* + 次期バージョンの Office 365 の**「全体管理者」**の権限を持つユーザー : 1 ユーザー

**参考**

次期バージョンの Office 365 のアカウントをお持ちでない場合は、試用版を利用可能です。以下のサイトから「Office 365 Enterprise プレビュー」をご利用ください。

* 「Office プレビューを入手する」

<http://www.microsoft.com/ja-jp/office/preview/try-office-preview.aspx>

以下のステップにより、実際に Bing Maps Office 用アプリを利用し、管理画面で Bing Maps Office 用アプリの設定項目を確認します。

|  |
| --- |
| ステップ１：Office 365 へサインイン |
| ステップ2：Outlook Web App で Bing Maps Office 用アプリの動作を確認 |
| ステップ3：Exchange Online のOffice 用アプリ管理画面の確認 |

「全体管理者」の権限を持つユーザーで Office 365 へサインインします。

|  |  |
| --- | --- |
| **ステップ１：Office 365 へサインイン** | |
| 1 | 1. サインイン ページ (<https://portal.microsoftonline.com>) にアクセスします。 |
| 2 | 1. [ユーザー ID] に「Office 365 のユーザー ID」を入力し、 [パスワード] に「Office 365 のパスワード」を入力します。 2. [サインイン] ボタンをクリックします。 |

次に、実際にメールを送信して Bing Maps Office 用アプリの動作を確認します。

|  |  |
| --- | --- |
| **ステップ2：Outlook Web App で Bing Maps Office 用アプリの動作を確認** | |
| 1 | 1. 画面上部の [Outlook] をクリックし、Outlook Web App に移動します。 2. [メール本文] に「Redmond, WA 98052-7329」 (英語形式の住所) を入力し、「サインイン中のユーザーのメールアドレス」を [宛先] に入力し、[送信] ボタンをクリックします。   **参考**  2012年7月時点での、Bing Maps Office 用アプリ(バージョン 1.0) は、日本の住所では動作しません。  Bing Maps Office 用アプリの動作を確認するため、サインイン中のユーザーへメールを送信します。 |
| 2 | 1. 受信したメールをクリックし、画面右のメール本文上部にある [Bing Maps] のタブをクリックします。   メール本文に住所が含まれているため、Bing Maps Office 用アプリが利用できます。 |
| 3 | 1. メール本文内に含まれる住所が Bing Maps Office 用アプリにより表示されていることを確認します。   Bing Maps Office 用アプリにより住所が認識され、ハイライトされています。  Bing Maps Office 用アプリは、外部サービスである Bing Maps を利用することにより、静的なマップではなく、拡大縮小が可能なリッチなマップを表示できます。 |

Outlook 2013 および Outlook Web App 画面内で Office 用アプリを利用できます。これにより、メールの本文や署名に含まれる住所をユーザーがコピーをして検索する手間を省き、シームレスに Bing Maps に表示され、メールをより効果的に利用でき、業務の効率化を行うことができます。

最後に、管理画面を確認し、Bing Maps Office 用アプリの各種設定項目を確認します。

|  |  |
| --- | --- |
| **ステップ3：Exchange Online の Office 用アプリ管理画面の確認** | |
| 1 | 1. 画面右上の [管理者] タブをクリックし、ドロップダウンメニューから [Exchange] をクリックします。 |
| 2 | 1. 左のメニューの [組織] をクリックします。 2. 画面上部の [アプリ] をクリックします。 3. アプリ一覧から [Bing Maps] をクリックし [ユーザーの規定値] が 「Enabled」 になっており、[有効な組織] が 「true」 になっていることを確認します。   これらの項目を参照し、Bing Maps Office 用アプリが標準で利用できるようになっていることが確認できます。 |

このように、Exchange Online には、標準で利用できるいくつかのメール アプリが用意されています。IT 管理者は管理画面からメール アプリの追加や削除、有効や無効にするといったことを集中管理でき、展開やアップグレードを容易に行うことができます。

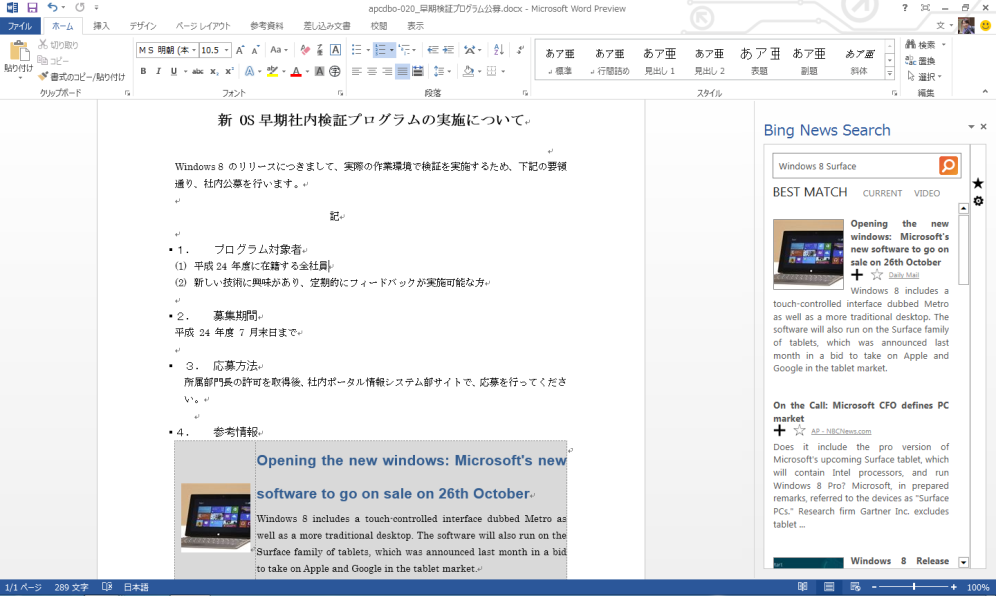
## Office 用アプリの種類

Office 用アプリは、特性や用途に合わせて 3 種類のタイプが用意されています。それぞれのタイプの特性と、利用するにあたっての利点を説明します。ここで紹介する Office 用アプリはOffice ストアで配布されているため、どなたでも簡単に利用できます。

### 作業ウィンドウ アプリ

作業ウィンドウ アプリは 専用の領域に表示される Office 用アプリです。Word 2013, Excel 2013, Excel Web App, Project 2013 へ共通の Office 用アプリの提供ができます。次の図の作業ウィンドウ アプリではドキュメント内の文字列を Bing News で検索し、その結果を表示しています。これにより、Internet Explorer と Office を、切り替えることなく作業を進めることができ、手間を削減でき、作業の効率を上げることができます。

**作業ウィンドウ アプリ**

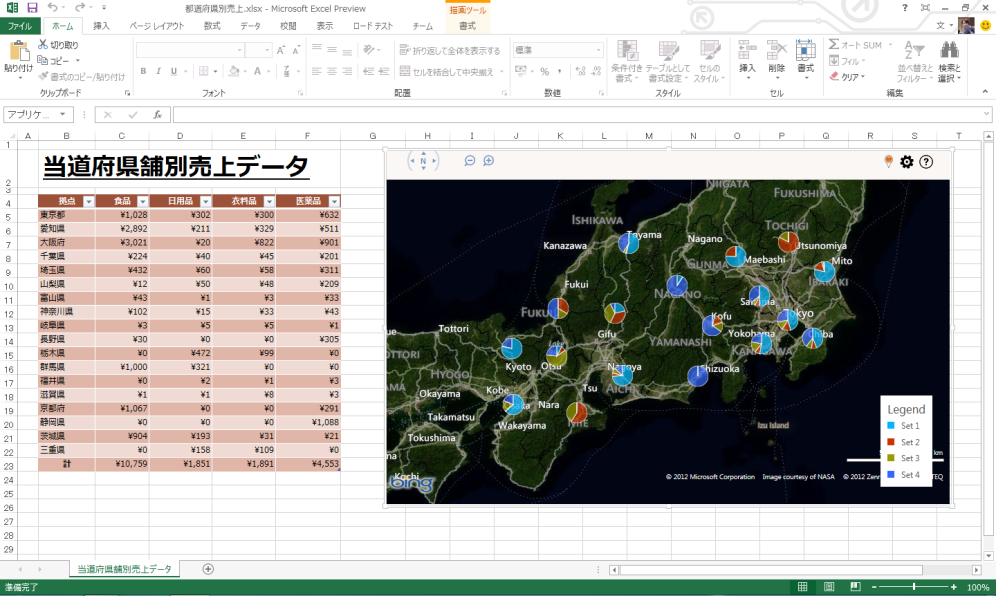


この Office 用アプリ「Bing News Search」は Bing News と連携するひとつの例です。その他の外部サービスと連携する Office 用アプリもOffice ストアに公開されています。

### コンテンツ アプリ

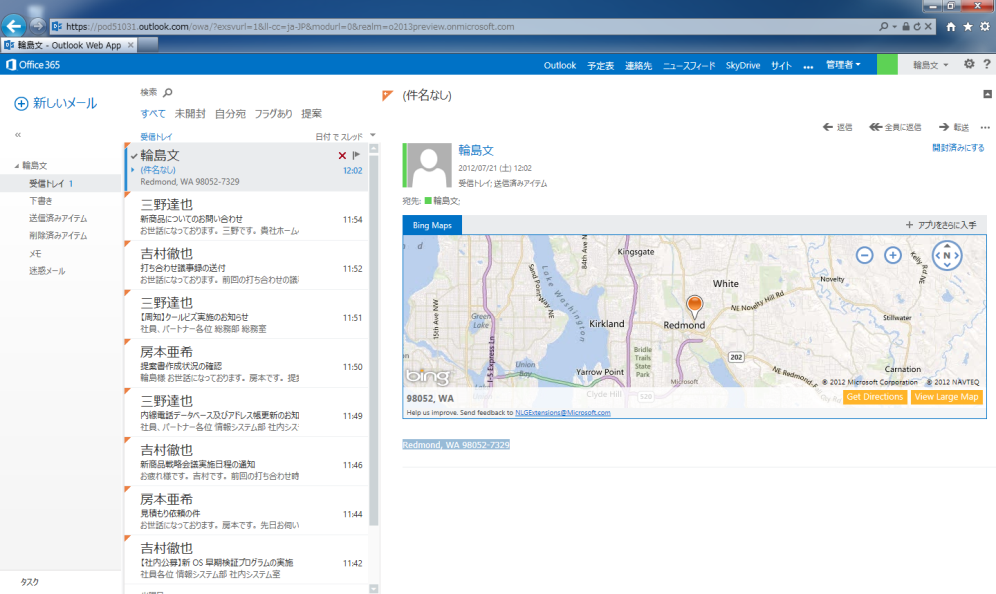
コンテンツ アプリはドキュメント内に表示が可能な Office 用アプリです。Excel 2013 および Excel Web App でサポートされています。ドキュメント内に Office 用アプリが配置できることにより、ドキュメントの一部として Office 用アプリを利用できます。次の図のコンテンツ アプリでは、ドキュメント内のデータをもとに、地図と表を描画しています。これにより、あらかじめ用意されたデザインを利用する事ができ、より早く、より理解しやすいドキュメントを作成できます。

**コンテンツ アプリ**



### メール アプリ

メール アプリは、メールの件名、本文または宛先のメールアドレスをもとに動作します。メール アプリはOutlook 2013, Outlook Web App および Mobile Outlook Web App によりサポートされており、多くのモバイルデバイスでの利用が可能です。これにより、クロスデバイス対応が容易にでき、開発コストを削減できます。次の図のように、Bing Maps のOffice 用アプリを利用し、メール上部に Bing Maps を表示させ、メール本文内に含まれる、場所や署名の会社情報などの住所が Bing Maps により表示でき、先に紹介した作業ウィンドウ アプリと同様に、Internet Explorer を起動して検索する必要がなく、メールを効果的に利用できます。



**メール アプリ**

ここで紹介した 3 種類の Office 用アプリの例は、Office ストアからインストールして利用できます。以降の手順では、Office ストアから Office 用アプリをインストールする方法をBing Maps Office 用アプリを例にとって紹介します。

**参考**

Office ストアの概要は後述の、「Office 用アプリの展開」をご参照ください。

|  |
| --- |
| ステップ１： Excel 2013 からOffice ストアを参照する |
| ステップ2： Office ストアから Bing Maps Office 用アプリをインストール |
| ステップ3： Bing Maps Office 用アプリを挿入 |

Excel 2013 のファイルを新規作成し、Office ストアを開きます。

|  |  |
| --- | --- |
| **ステップ１：Excel 2013 からOffice ストアを参照する** | |
| 1 | 1. [スタート] - [すべてのプログラム] - [Microsoft Office] - [Microsoft Excel 2013] を選択し Excel 2013 を起動します。 2. [空白のブック] を選択し、ファイルを新規作成します。      1. [挿入] タブの [Office 用アプリ] ボタンをクリックします。   **参考**  既存の Excel 2013 ファイルも同手順で Office 用アプリを追加できます。 |
| 2 | 1. [アプリケーションの挿入] ダイアログの [Office Store でその他のアプリケーションを検索] リンクをクリックします。   Internet Explorer で、Office ストアが起動します。 |

Office ストアから Bing Maps Office 用アプリ をインストールします。

|  |  |
| --- | --- |
| **ステップ2： Office ストアから Bing Maps Office 用アプリをインストール** | |
| 1 | 1. [U.S. サイト] リンクをクリックします。     **参考**  2012 年 7 月時点では、日本向けのOffice ストアは提供されていないため、ここでは、U.S. サイトから Office 用アプリをインストールする手順を紹介しています。 |
| 2 | 1. 検索ボックスに [Bing Maps] を入力し、検索します。 |
| 3 | 1. [Bing Maps] をクリックします。 |
|  | 1. [Add] ボタンをクリックします。   [Add] ボタンをクリック後、Office 用アプリがインストールされ、利用手順が表示されます。内容を確認し、Internet Explorer を終了します。 |

ファイルに Bing Maps Office 用アプリを挿入します。

|  |  |
| --- | --- |
| **ステップ3： Bing Maps Office 用アプリを挿入** | |
| 1 | 1. Excel 2013 の [アプリケーションの挿入] ダイアログ上の [最新の情報に更新] ボタンをクリックします。 2. [Bing Maps] のアイコンをクリックし、[挿入] ボタンをクリックします。   Bing Maps Office 用アプリが挿入されます。 |
| 2 | 1. Bing Maps Office 用アプリが挿入されることを確認します。 2. 任意のセルを選択し、[Insert Sample Data] をクリックします。 |
| 3 | 1. サンプルデータが設定され、Bing Maps Office 用アプリが動作します。   サンプルデータをもとに図が表示されます。 |

ユーザーは、Office ストアにある豊富な Office 用アプリを利用することで、データのさらなる活用方法や表現力豊かなプレゼンテーション手段を手に入れることができます。

### 各 Office 製品がサポートする Office 用アプリの種類

各 次期 Office 製品に対する Office 用アプリのタイプごとのサポートは以下のとおりです。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Office 用アプリの種類** | **Word 2013** | **Excel 2013 / Excel WAC** | **Outlook 2013 / OWA /**  **Mobile OWA** | **Project 2013** |
| **作業ウィンドウ アプリ** | ○ | ○ |  | ○ |
| **コンテンツ アプリ** |  | ○ |  |  |
| **メール アプリ** |  |  | ○ |  |

作業ウィンドウ アプリは4つの 次期 Office 製品に対応しており、 共通の Office 用アプリを開発するのに適しています。コンテンツ アプリはドキュメント内に埋め込むことができ、より早く、より理解しやすいドキュメントを作成できます。また、メール アプリは Mobile Outlook Web App にも対応しているため、モバイルのユーザーでも Office 用アプリを利用できます。

## Office 用アプリのアーキテクチャー

Office 用アプリによるソリューションを開発するにあたり、アーキテクチャーを理解することは重要です。Office 用アプリの各構成要素がどのように機能しているか、開発技術である Web 標準準拠、ランタイムレベルでサポートされているセキュリティモデルを説明します。

### 構成概要

* Office 用アプリの構成

次の図は、Office 用アプリの構成要素を示しています。Office 用アプリは、XML マニフェスト ファイルと Web ページで構成されます。ホストアプリケーションである次期 Office がカタログサーバーに配置されているXML マニフェスト ファイルを取得 (①) します。XML マニフェスト ファイルには Office 用アプリの実体である Web ページの URL が定義されています。次に、次期 Office は Web ページを読み込み (②)、Office 用アプリランタイム上で Office 用アプリを実行 (③) します。

2

1

3

Office 用アプリ

Office用 アプリ

ランタイム

XML マニフェスト

ファイル

Web

ページ

取得

読み込み

Web サーバー

カタログ

サーバー

次期 Office

実行

Office 用アプリは HTML や CSS による UI と、JavaScript などを利用したロジックを実装した、Web 標準に準拠した Web ページです。

XML マニフェスト ファイルは XML 形式のファイルであり、次のような項目を定義します。

1. 作業ウィンドウ アプリまたはメール アプリのタイプの定義
2. Office 用アプリの ID
3. Office 用アプリのバージョン
4. Office 用アプリの提供者名
5. 標準の地域
6. Office 用アプリの表示名
7. Office 用アプリの説明
8. Office 用アプリのアイコンのパス
9. サポートする Office アプリケーションの種類
10. UI と処理を実装した Office 用アプリ(Web ページ) の URL
11. Office 用アプリが必要とする Permission レベル

以下が XML マニフェスト ファイルのサンプルです。

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  **①**<OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="TaskPaneApp">  **②**<Id>d2ae0231-41df-4d28-b935-c9a75d859798</Id>  **③**<Version>1.0</Version>  **④**<ProviderName> Your Company Name</ProviderName>  **⑤**<DefaultLocale>JA-JP</DefaultLocale>  **⑥**<DisplayName DefaultValue="App" />  **⑦**<Description DefaultValue="App Description"/>  **⑧**<IconUrl DefaultValue="~remoteAppUrl/Images/App.png" />  <Capabilities>  **⑨**<Capability Name="Workbook" />  </Capabilities>  <DefaultSettings>  **⑩**<SourceLocation DefaultValue="~remoteAppUrl/Pages/App1.html" />  </DefaultSettings>  **⑪**<Permissions>ReadWriteDocument</Permissions>  </OfficeApp> |

次の図は、SharePoint Online による Office 用アプリの展開の構成および各ユーザーの役割を示しています。

IT 管理者

Web サーバー

SharePoint Online

SharePoint Online による Office 用アプリの展開の構成



次期 Office



利用者

Office 用

アプリ

開発者

Office 用アプリ



XML マニフェスト　 ファイル

ファイル

一覧からOffice

用アプリを取得

XML マニフェスト ファイルをアップロード

Web ページを配置



Web ページ

ページ

**参考**

XML マニフェストファイルの基本的な設定手順は後述の、「Office 用アプリによるソリューション開発」を参照してください。また、詳細は以下のサイト (英語) をご参照ください。

* 「Anatomy of an app for Office」

<http://msdn.microsoft.com/en-us/library/office/apps/jj220082(v=office.15)#StartBuildingApps_AnatomyofApp>

* + 開発者

Office 用アプリを開発し、XML マニフェスト ファイルと Web ページを作成します。

* + IT 管理者

SharePoint Online の内部ディレクトリに Office 用アプリの XML マニフェスト ファイルを登録し、リストベースの集中管理を行います。グループポリシーにより内部ディレクトリを利用者へ展開します。

任意の Web サーバーへ Web ページを配置します。

* + 利用者

内部ディレクトリで管理されている Office 用アプリを一覧から取得し、利用できます。

**参考**

Office 用アプリ展開の詳細および手順は後述の、「Office 用アプリの展開」をご参照ください。

### Web 標準準拠

Office 用アプリはHTML, JavaScript, CSS などの Web 標準技術を利用して開発します。そのため、インターネット上で配布されている Web 標準で作成されたライブラリの利用や、Web 標準のインターフェイスで公開されているサービスとの連携ができます。

例として、次のような実装が可能です。

* + HTML ベースの UI とJavaScript によるロジックの実装
  + jQuery などの JavaScript フレームワークの使用
  + REST API や Web Service のインターフェイスをもつ SharePoint, Office 365, SNS との連携
  + ASP.NET や PHP などのサーバー サイド テクノロジーによるロジックの実装

Office 用アプリによるソリューション開発では、UI層、ロジック層、サービス層で次の図のような実装ができます。静的な HTML コンテンツをはじめ、JavaScript を利用した UI およびロジックの実装、ASP.NET や PHP などのサーバーサイドロジックを実装でき、Web 標準のインターフェイスで公開されているオンプレミスやクラウドの各種サービスとの連携ができます。

**UI**

**ロジック**

**サービス**

ASP.NET, PHP etc...

HTML 5, CSS 3

JavaScript

クラウド

Office 365

Azure Service

オンプレミス

SharePoint

Exchange

基幹システム

SNS

**参考**

**「jQuery」** とは JavaScript のライブラリです。詳細は以下のサイトをご参照ください。

* 「Visual Studio 2010 自習書 ～ Do-It-Yourself シリーズ ～ ASP.NET による Web アプリケーション開発の基礎」

<http://msdn.microsoft.com/ja-jp/asp.net/gg638968.aspx>

**「REST」** とはアーキテクチャスタイルの１つです。詳細は以下のサイトをご参照ください。

* 「REST の詳細」

<http://msdn.microsoft.com/ja-jp/magazine/dd942839.aspx>

**「Ajax」** とは Web ブラウザ内で非同期通信を行う技術です。詳細は以下のサイトをご参照ください。

* 「Microsoft Ajax の概要」

<http://msdn.microsoft.com/ja-jp/library/bb398874.aspx>

**「PHP」** とは動的 Web ページを作成するための言語です。詳細は以下のサイトをご参照ください。

* 「PHP の概要」

<http://msdn.microsoft.com/ja-jp/library/cc294979.aspx>

**「Azure」** とは Microsoft が提供する PaaS クラウドです。詳細は以下のサイトをご参照ください。

* 「Windows Azure Platform とは」

<http://www.microsoft.com/japan/windowsazure/products/>

### セキュリティ

Office 用アプリでは、ランタイムレベルでサポートされているセキュリティモデルとして、サンドボックスおよび Permission モデルが提供されています。クライアント Office アプリケーションおよび、Web Apps 上でOffice 用アプリが実行される際、サンドボックスを利用し、実行プロセスを分離し、セキュリティを担保しています。さらに、Permission モデルの採用により、Office 用アプリがアクセスするリソースのレベルを明確化できます。

* サンドボックス

クライアント Office アプリケーション上で動作する Office 用アプリはサンドボックス内で動作するため、高いセキュリティを担保しています。また、 Web Apps ではサンドボックスモードの IFrame により動作し、同様にセキュリティを担保しています。

クライアント Office アプリケーションのサンドボックスおよび Web Apps のサンドボックスモードを使用して、Office 用アプリを他のプロセスに影響しない領域で動作させることにより、サンドボックス内のプロセスは他のシステムに対して介入ができません。そのため、他システムへの影響を防ぐことができます。サンドボックスによるプロセスレベルでのセキュリティの担保により、開発者は Office 用アプリによるソリューション実装に注力する事ができ、短期間でより高い品質の Office 用アプリが開発できます。

* Permission モデル

1. 作業ウィンドウ アプリの Permission モデル

作業ウィンドウ アプリでは 5 段階の Permission モデルがOffice 用アプリのセキュリティの基本として提供されます。開発者は必要に応じて 5 段階の Permission レベルから Office 用アプリが必要とするレベルを選択し、XML マニフェスト ファイルに定義します。最もセキュアな Permission レベルは「Restricted」で、 Office 用アプリからドキュメントのデータへアクセスできません。Office 用アプリからドキュメントのデータにアクセスできないことが明確にされるため、利用者はドキュメントのデータが保護されていることを確認できます。また、Office 用アプリからドキュメントへのデータの読み書きが必要な場合は、Permission レベルの「ReadWrite Document」を指定し、ドキュメントのデータに対して読み書きすることを明確化します。これにより、ユーザーは、Office 用アプリがドキュメントのデータにアクセスすることを明示的に知ることができます。

Permission モデル

Restricted

Write  
Document

Read  
Document

ReadAll  
Document

ReadWriteDocument

各 Permission レベルにおける可能な操作は、以下の通りです。

|  |  |  |
| --- | --- | --- |
| **Permission レベル** | **説明** | 高  低 |
| **Restricted** | アプリの設定読み込み |
| **ReadDocument** | アプリの設定読み込み  ドキュメントからの読み込み  バインディングの作成  チェンジイベントの取得  ※ Restricted レベルを含む |
| **ReadAllDocument** | アプリの設定読み込み  ドキュメントからの読み込み  ドキュメントの OOXML バイナリ取得（Document.getSelectedDataAsync メソッド）  バインディングの作成  チェンジイベントの取得  ※ Restricted および ReadDocument レベルを含む |
| **WriteDocument** | アプリの設定の書き込み  ドキュメントへの書き込み （Document.setSelectedDataAsyncメソッド）  ※ Restricted レベルを含む |
| **ReadWriteDocument** | アプリの設定読み込みおよび書き込み  ドキュメントへの書き込み（全ての API）  バインディングの作成  チェンジイベントの取得  ※ 全てのレベルを含む |

1. コンテンツ アプリの Permission モデル

コンテンツアプリでは4段階の Permission モデルがOffice 用アプリのセキュリティの基本として提供されます。アクセス許可レベルが最も高い「ReadWriteDocument」から、最も低い「Restricted」と定義されています。開発者は必要に応じて 4 段階の Permission レベルから Office 用アプリが必要とするレベルを選択し、XML マニフェスト ファイルに定義します。最もセキュアな Permission レベルは「Restricted」で、 Office 用アプリからドキュメントのデータへアクセスできません。Office 用アプリからドキュメントのデータにアクセスできないことが明確されるため、利用者はドキュメントのデータが保護されていることを確認できます。また、Office 用アプリからドキュメントへのデータの読み書きが必要な場合は、Permission レベルの「ReadWrite Document」を指定し、ドキュメントのデータに対して読み書きすることを明確化します。これにより、ユーザーは、Office 用アプリがドキュメントのデータにアクセスすることを明示的に知ることができます。

Permission モデル

Restricted

Write  
Document

Read  
Document

ReadWriteDocument

各 Permission レベルにおける可能な操作は、以下の通りです。

低

|  |  |  |
| --- | --- | --- |
| **Permission レベル** | **説明** | 高 |
| **Restricted** | アプリの設定読み込み |
| **ReadDocument** | アプリの設定読み込み  ドキュメントからの読み込み  バインディングの作成  チェンジイベントの取得  ※ Restricted レベルも含む |
| **WriteDocument** | アプリの設定の書き込み  ドキュメントへの書き込み （Document.setSelectedDataAsyncメソッド）  ※ Restricted レベルも含む |
| **ReadWriteDocument** | アプリの設定読み込みおよび書き込み  ドキュメントへの書き込み（全ての API）  バインディングの作成  チェンジイベントの取得  ※ 全てのレベルを含む |

3. メール アプリのPermission モデル

メール アプリでは 3 段階の Permission モデルが提供されます。アクセス許可モデルの最も高い「ReadWwite Mailbox」から、最も低い「Restricted」と定義されています。各レベルは下位レベルの権限を継承しています。最もセキュアな「Restricted」では、JSOM の利用のみが許可されています。「Restricted」レベルが定義された Office 用アプリでは、メールの送信を実行する機能が利用できず、ユーザーは自分の利用する Office 用アプリからメール送信が行われないことを明示的に確認できます。開発者が、Office 用アプリから、メールの送信を実行したい場合は、「ReadWrite Mailbox」を指定し、Office 用アプリによるメールの送信を許可し、ユーザーにOffice 用アプリの振る舞いを提示できます。

ReadWrite Mailbox

Read item

Restricted

Permission モデル

**参考**

**「JSOM」** とはJavaScript object model の略語で、Office 用アプリ開発の際に利用し、Office 用アプリとドキュメントのデータのやり取りなどに利用します。実際の利用方法は後述の「Office 用アプリによるソリューション開発」をご参照ください。

各 Permission レベルにおける可能な操作は、以下の通りです。

|  |  |  |
| --- | --- | --- |
| **Permission レベル** | **説明** | 低  高 |
| **Restricted** | JSOM の利用 |
| **Read item** | カスタム正規表現による Active ルールの利用  JSOM によるフルアクセス (makeEWSRequestAsync を除く)  メールのプロパティの読み取り、書き込み |
| **ReadWrite mailbox** | Exchange Web Service の利用  フォルダーおよびメールの作成、読み取り、書き込み  メールの送信 |

Office 用アプリでは Permission モデルによりアクセス可能な範囲および、振る舞いを明確化し、制限できます。これにより、利用者またはIT 管理者は Office 用アプリのインストール前に、要求する Permission レベルを確認でき、Office 用アプリによる影響範囲を認識できます。開発者は Office 用アプリによるソリューション開発を行う際に、必要とする最小限の Permission レベルを選択することにより、利用者が安心して利用できるようにします。

開発者はサンドボックス モデルにより、実行するOffice 用アプリのプロセスが他のプロセスへ与える影響を意識しないでセキュアに動作させることができます。さらに、Permission モデルを利用するにあたり、高いアクセス許可レベルの利用による、できることが増えることによる開発の自由度と、Office 用アプリからアクセスできるリソースの増加によるセキュリティの低下のトレードオフを開発する際に意識でき、Office 用アプリのセキュリティに対する意識を持つことができます。

**参考**

**「Active ルール」**とはメール アプリが表示されるかを制御するルールです。詳細は以下のサイト (英語) をご参照ください。

* 「Defining rules to show a mail app in Outlook」

<http://msdn.microsoft.com/en-us/library/office/apps/fp142188(v=office.15)>

**「makeEWSRequestAsync」**とは Office 用アプリから Exchange Web Service へアクセスするメソッドです。詳細は以下のサイト (英語) をご参照ください。

* 「makeEwsRequestAsync method (Mailbox object)」

<http://msdn.microsoft.com/en-us/library/office/apps/fp161019(v=office.15)>

**「Exchange Web Service」**とは Exchange の各種API を公開している Web サービスです。詳細は以下のサイト (英語) をご参照ください。

* 「Exchange Web Services Reference」

<http://msdn.microsoft.com/en-us/library/bb204119(v=exchg.140).aspx>

# Office 用アプリによるソリューション開発

Office 用アプリの開発は、メモ帳などのテキスト エディターにより開発が可能です。HTML や JavaScriptの編集に特化したエディターは数多くありますが、これらの中でも、「Visual Studio 2012」では、Office 用アプリ開発に特化したプロジェクト テンプレートが用意されています。これらのテンプレートのいずれかを使用すると、基本となるプロジェクト コンテナが作成されるので、開発に必要なファイルを最初から作成するという手間を省くことができます。また、コーディング補助機能ツールであるインテリセンスや、デバッグ機能などを利用することで、開発生産性の向上にもつながります。

## Office 用アプリ開発概要

### 必要環境

Office 用アプリ開発の推奨開発環境は、以下のとおりです。一般的な Office 用アプリ開発では、Visual Studio 2012 を開発環境として利用します。

|  |  |
| --- | --- |
|  | **推奨環境** |
| **OS** | Windows 7 または Windows 8 |
| **Office** | Office 2013 Preview / Office 365 ProPlus Preview  (Windows 8 を利用している場合は 32bit 版を推奨) |
| **開発ツール** | HTML ,JavaScript エディター  (Notepad, Visual Studio, サードパーティ製の Web 開発ツール など) |
| **Webサーバー** | IIS Express 7.5（Visual Studio 2012 に付属） |
| **ブラウザ** | Internet Explorer 9.0 以上 |

次に示すのは、本ドキュメントで実装する「Hello World の開発」と「Visual Studio 2012 による Office 用アプリの開発」に必要な環境です。

|  |  |  |
| --- | --- | --- |
|  | **Hello World の開発** | **Visual Studio 2012 による Office 用アプリの開発** |
| **OS** | Windows 7 | Windows 7 |
| **Office** | Office 2013 Preview / Office 365 ProPlus Preview | Office 2013 Preview / Office 365 ProPlus Preview |
| **開発ツール** | メモ帳 | Visual Studio 2012 RC |
| Microsoft Office Developer Tools for Visual Studio 2012 |
| **Webサーバー** | IIS 7.5（OSに標準機能。既定では無効） | IIS Express 7.5（Visual Studio 2012 に付属） |
| **ブラウザ** | Internet Explorer 9.0 | Internet Explorer 9.0 |

これらの環境は、以降の手順で必要となりますので、あらかじめご用意ください。

**参考**

以降の手順において、**IIS 7.5 （Windows 7 に付属）**を利用するためには、「Windows の機能の有効化」により、以下の機能のインストールが必要です。

* + [インターネット インフォメーション サービス] - [Web 管理ツール] - [IIS 管理コンソール]
  + [インターネット インフォメーション サービス] - [World Wide Web サービス] - [HTTP 共通機能] - [静的コンテンツ]

「Windows の機能の有効化」の手順の詳細は以下のサイトをご参照ください。

* 「Windows 7 Professional、Enterprise、または Ultimate への IIS 7.5 のインストール」

[http://technet.microsoft.com/ja-jp/library/cc725762(v=ws.10).aspx](http://technet.microsoft.com/ja-jp/library/cc725762(v=ws.10).aspxttp:/msdn.microsoft.com/ja-jp/asp.net/gg638968.aspx)

**参考**

**「Office 用アプリ開発のシステム要件」** の詳細については以下のサイト (英語) をご参照ください。

* 「Overview of apps for Office」

http://msdn.microsoft.com/en-us/library/office/apps/jj220082(v=office.15)#StartBuildingApps\_Requirements

### 開発手順

Office 用アプリの開発を進めるうえで、基本となる流れを紹介します。

* XML マニフェスト ファイルの作成

Office 用アプリの開発作業は、まず、Office 用アプリを定義するXML マニフェスト ファイルの作成から始めます。XML マニフェスト ファイルには、名前、説明、バージョン番号などを定義します。

* ロジック の実装（HTML/JavaScript）

ユーザー インターフェイスをHTML で、イベントごとに実行する処理や機能を JavaScript や ASP.NET などのサーバーサイドスクリプトで実装します。

* Office 用アプリの展開

作成した Office 用アプリの Web サーバーへの配置と、XML マニフェスト ファイルの配布方法を選択し、ユーザーへ公開します。

## Hello World の開発

ここでは、テキスト エディター（メモ帳）を使用して Office 用アプリの作成を行います。

### Hello World の実装

作業ウィンドウに「Hello World!」を表示する Office 用アプリを実装してみましょう。

|  |
| --- |
| ステップ1：XML マニフェスト ファイルの作成 |
| ステップ2：HTML の実装 |

前述の「開発手順」に従って、開発を進めていきます。（今回は、機能がないため、JavaScript の実装は行いません）

|  |  |
| --- | --- |
| **ステップ１：XML マニフェスト ファイルの作成** | |
| 1 | 1. [スタート] - [すべてのプログラム] - [アクセサリ] - [メモ帳] を選択しメモ帳を起動します。 2. メモ帳に以下を記述します。  |  | | --- | | <?xml version="1.0" encoding="UTF-8"?>  <OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="TaskPaneApp">  <Id>37ee5724-0aae-4ff8-b144-488525b7cfb1</Id>  <Version>1.0</Version>  <ProviderName>Your Company Name</ProviderName>  <DefaultLocale>ja-JP</DefaultLocale>  <DisplayName DefaultValue="Hello Apps For Office" />  <Description DefaultValue="My First App For Office"/>  <IconUrl DefaultValue="http://o2013-pc/AppsForOffice-te/HelloWorld.png" />  <Capabilities>  <Capability Name="Workbook" />  </Capabilities>  <DefaultSettings>  <SourceLocation DefaultValue="http://o2013-pc/AppsForOffice-te/HelloWorld.html" />  </DefaultSettings>  <Permissions>ReadWriteDocument</Permissions>  </OfficeApp> |  |  |  | | --- | --- | | **要素** | **説明** | | **Id** | Office 用アプリを識別するためのユニークな ID | | **Version** | Office 用アプリのバージョン番号 | | **ProviderName** | Office 用アプリを提供する個人または組織の名前 | | **DefaultLocale** | 対象のユーザーのロケーション | | **DisplayName** | Office 用アプリの表示名 | | **Description** | Office 用アプリの説明 | | **IconUrl** | Office 用アプリのアイコン URL | | **Capabilities/Capability** | 対象となる Office アプリケーション | | **DefaultSettings/** **SourceLocation** | 参照先のWebサイト URL | | **Permissions** | 実行するための要求許可 |  1. [ファイル] - [名前を付けて保存] を選択し、ファイル名を「HelloWorld.xml」として「c:\AppsForOffice\_manifest\HelloWorld.xml」に保存します。 2. ファイル名「HelloWorld.png」のアイコンファイル (PNG 形式、縦横幅 32 ピクセル 推奨) を用意し、「c:\AppsForOffice\text\HelloWorld.png」に格納します。 |

|  |  |
| --- | --- |
| **ステップ2：HTML の実装** | |
| 1 | 1. [スタート] - [すべてのプログラム] - [アクセサリ] - [メモ帳] を選択します。 2. メモ帳に以下を記述します。  |  | | --- | | <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8" />  <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>  </head>  <body>  <p>Hello World!</p>  </body>  </html> |  1. [ファイル] - [名前を付けて保存] を選択し、ファイル名を「HelloWorld.html」として「c:\AppsForOffice\text\HelloWorld.html」に保存します。 |

以上で、実装は終わりです。

### Hello World の展開

「Hello World の実装」で作成したOffice 用アプリを「ファイル共有」で展開して Office で利用してみましょう。

|  |
| --- |
| ステップ1：Web サーバーの構成 |
| ステップ2：XML マニフェスト ファイルの配置 |
| ステップ3：動作確認 |

作成したHTMLファイルは、Web サーバーに配置して公開する必要があります。ローカル PC上の IIS に仮想ディレクトリを作成して配置を行います。

**参考**

今回は、ローカル PC 上の IIS に配置していますが、本番稼働では、公開用の Web サーバーに配置してください。

|  |  |
| --- | --- |
| **ステップ1：Web サーバーの構成** | |
| 1 | 1. [スタート] - [コントロール パネル] - [管理ツール] を選択し、[インターネット インフォメーション サービス (IIS) マネージャー] を起動します。 2. IIS マネージャーの左のペインから、[Default Web Site] を右クリックして、 [仮想ディレクトリの追加] を選択します。        1. [仮想ディレクトリの追加] ダイアログが表示されたら、[エイリアス] に「AppsForOffice-te」、[物理パス] に「c:\AppsForOffice\text」を入力し、[OK] ボタンをクリックします。            1. [Default Web Site] に、作成した仮想ディレクトリを確認できます。       **参考**  **「仮想ディレクトリ」** とは 物理ディレクトリを Webサイトのホームディレクトリのサブディレクトリとして存在するように割り当てたディレクトリのことです。詳細は以下のサイトをご参照ください。   * 「方法：IIS 7.0 内で仮想ディレクトリを作成および構成する」   <http://msdn.microsoft.com/ja-jp/asp.net/gg638968.aspx>   1. IIS マネージャーの左のペインから、[AppsForOffice-te] フォルダーをクリックして、右のペインの [認証] をダブル クリックします。 2. [匿名認証] を右クリックして、[編集] をクリックします。              1. [アプリケーション プール ID] を選択して、[OK] ボタンをクリックします。 |
| 2 | 1. 次に、[AppsForOffice-te] フォルダーを右クリックして、[アクセス許可の編集] を選択します。        1. [text のプロパティ] ダイアログが表示されたら、[セキュリティ] タブを選択し、[編集] ボタンをクリックします。          1. [text のアクセス許可] ダイアログ が表示されたら、[追加] ボタンをクリックします。 2. [ユーザー または グループ の選択] ダイアログが表示されたら、[場所の指定] がローカル PC になっていることを確認し、[選択するオブジェクト名を入力してください] に   「IIS AppPool\DefaultAppPool」（このアカウントを追加して、[名前の確認] ボタンをクリックすると下図のように [DefaultAppPool] が下線付きで表示されます）を入力して、[OK] ボタンをクリックしてください。           1. [text のアクセス許可] ダイアログに戻ったら、[DefaultAppPool] に [読み取りと実行] 権限が付与されていることを確認できます。         **参考**  IIS 上の Web アプリケーションは、アプリケーションプールアカウントで動作します。そのため、物理パスで指定した「c:\AppsForOffice\text」に、「Default Web Site」のアプリケーションプールアカウントである「IIS AppPool\DefaultAppPool」のアクセス権限が必要となるために、「アクセス許可の編集」を行います。 |

|  |  |
| --- | --- |
| **ステップ2：XML マニフェスト ファイルの配置** | |
| 1 | 1. [エクスプローラー] で「c:\」を開きます。 2. 新規フォルダーを作成し、フォルダー名を「AppForOffice\_catalog」にします。 3. [AppForOffice\_catalog] フォルダーに「HelloWorld.xml」をコピーします。 4. [AppForOffice\_catalog] フォルダーを共有化します。   **参考**  **「フォルダーの共有化」**の詳細については以下のサイトをご参照ください。   * 「他のユーザーとファイルを共有する」 * <http://windows.microsoft.com/ja-JP/windows7/Share-files-with-someone>   **注意**  「ファイル共有」で展開する場合、通常は、クライアントが参照可能なファイルサーバーに配置します。ここでは、ローカル PCを使用するため、XML マニフェスト ファイルを格納したフォルダーの共有化が必要になります。 |
| 2 | 1. Excel 2013を起動し、[空白のブック] を選択します。 2. [ファイル] メニューから [オプション] を選択し、[Excelのオプション] ダイアログを表示します。 3. [セキュリティ センター] を選択し、[セキュリティ センターの設定] ボタンをクリックし、[セキュリティ センター] ダイアログを表示します。 4. [信頼されているアプリケーション カタログ] を選択し、[カタログの URL] に「\\localhost\AppForOffice\_catalog」と入力し、[カタログの追加] ボタンをクリックします。 5. 追加したカタログの [メニューに表示する] のチェックを [ON] にし、[OK] ボタンをクリックします。      1. [カタログと Office 用アプリ の管理] ダイアログの [OK] ボタンをクリックします。 2. Excel 2013 を終了します |

|  |  |
| --- | --- |
| **ステップ3：動作確認** | |
| 1 | 1. Excel 2013 を起動し、[空のブック] を選択します。 2. [挿入] タブから [Office 用アプリ] – [全ての表示] を選択します。 3. [Office 用アプリの挿入] ダイアログから、[共有フォルダー] を選択します。 4. [共有フォルダー] の中から、[Hello Apps For Office] を選択し、[挿入] ボタンをクリックします。     **参考**  XML マニフェスト ファイルで設定した、「DisplayName」、「Description」「アイコン」は、ここで表示を確認できます。   1. 挿入後、ワークシートの右側の領域に、作成した作業ウィンドウ アプリを確認することができます。 |

以上が、「Hello World」の開発となります。メモ帳のみで Office 用アプリの開発を行えることが確認できたと思います。

## Visual Studio 2012 による Office 用アプリの開発

ここでは、開発ツールに Visual Studio 2012 を利用した Office 用アプリの開発について学習していきます。

実装する Office 用アプリは、以下の機能を有する作業ウィンドウ アプリです。

* + Excel のセルに入力された住所から地図検索し、ピンを設定する

Office ドキュメントからのデータの取得方法について学習します。

* + 検索結果をブラウザで表示するための URL をExcel のセル へ設定する

Office ドキュメントへのデータの設定方法について学習します。

* + 住所の変更に伴い、地図に設定したピンを自動で再設定する

Office ドキュメントデータのバインドについて学習します。

各機能の実装をすすめながら、Office 用アプリ開発について理解を深めていきましょう。

**注意**

地図検索は、Webページ用の地図コントロールおよび API として JavaScript で公開されている「Bing Maps AJAX Control」と、Bing が地図機能として公開しているWeb APIの「Bing Maps REST Services」を連携させて行います。Bing Maps API を利用するには、「Bing Maps Key」 が必要です。Key の取得は、「Bing Maps Account Center」で登録を行い Key を取得します。また、「Bing Maps Account Center」へのサインインには、Windows Live ID が必要となります。

「Bing Maps Account Center」

* <https://www.bingmapsportal.com>

取得した 「Bing Maps Key」は、後で利用しますので、メモ帳などに保存しておいてください。

### プロジェクト テンプレート

Visual Studio 2012 を利用すると、[Office/SharePoint] - [アプリ] のグループに格納する「App for Office 2013」テンプレートを選択して、Office 用アプリの開発を始めることができます。「App for Office 2013」プロジェクトで開発できるOffice 用アプリの種類は以下の通りです。

* + 作業ウィンドウ アプリ（Excel, Word, Project）

複数の次期 Office 製品に対応した作業ウィンドウ アプリの開発に適したアプリ テンプレートです。

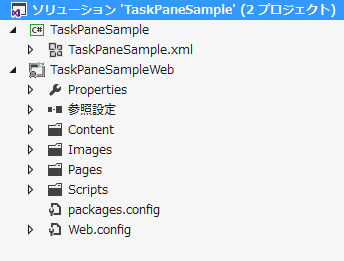
* + コンテンツ アプリ（Excel）

Excel 2013 向けのコンテンツ アプリ テンプレートです。

* + メール アプリ（Outlook）

Outlook 2013 向けのメール アプリ テンプレートです。

この章の冒頭で記述したように、プロジェクト テンプレートを利用してプロジェクトを作成すると、開発やデバッグに必要なファイルが、既定で用意されます。プロジェクトのファイル構成を以下に示します。



* + <ProjectName>.xml

Office 用アプリの XML マニフェスト ファイル。作成する Office 用アプリについて定義します。

* + Properties

Office 用アプリのアセンブリ情報の定義が格納されています。

* + Content

共通で利用するスタイルを定義するファイルを格納します。既定で「Style.css」および「Office.css」が格納されているので、必要に応じて編集を行います。

* + Images

Office 用アプリのアイコンファイルを格納します。既定で png 形式のアイコンファイルが格納されています。

* + Pages

Office 用アプリの UI を実装するファイルを格納します。既定で、<ProjectName>.html が格納されているので、必要に応じて編集を行います。

* + Scripts

Office 用アプリが使用するJavaScript ライブラリおよび、<ProjectName>.js が格納されています。Office 用アプリのイベントごとに実施する処理や機能を必要に応じて編集を行います。

* + packages.config

Office 用アプリが利用するパッケージが定義されます。通常は、直接、編集する必要はありません。

* + web.config

Office 用アプリの各種設定を定義します。

**参考**

プロジェクト テンプレートから作成したXML マニフェストファイルおよび HTML ファイル、JavaScript ファイルはプロジェクト名と同じ名前になるため <ProjectName> と表記しています。

### ドキュメントからデータの取得

ここからは、具体的な実装コード例を取り上げます。

|  |
| --- |
| ステップ1：プロジェクトの作成 |
| ステップ2：XML マニフェスト ファイルの編集 |
| ステップ3：HTML の実装 |
| ステップ4：JavaScript の実装 |
| ステップ5：動作確認 |

Visual Studio 2012 のプロジェクト テンプレートからプロジェクトを新規作成します。まず、XML マニフェスト ファイルを編集します。次に、ドキュメントからデータを取得するためのボタンをUI に配置します。そして、ロジックとして、[getSelectedDataAsync] 関数を利用して、ドキュメントからのデータの取得の実装を行います。最後に、動作確認を行います。

|  |  |
| --- | --- |
| **ステップ1：プロジェクトの作成** | |
| 1 | 1. [スタート] - [すべてのプログラム] - [Visual Studio 2012] を選択し Visual Studio 2012 を起動します。 2. [ファイル] - [新規作成] - [プロジェクト] を選択します。 3. [テンプレート] - [Office/SharePoint] - [アプリ] –[ App for Office 2013] を選択し、[名前] に「TaskPaneSample」と入力して [OK] ボタンをクリックします。          1. [ Office用アプリの作成] ダイアログで、[作業ウィンドウのアプリ] を選択し、 [Excel]のチェックボックスを選択して、[完了] ボタンをクリックします。      1. Office 用アプリ開発を行うためのプロジェクト（およびソリューション）が作成されます。 |

|  |  |
| --- | --- |
| **ステップ2：XML マニフェスト ファイルの編集** | |
| 1 | 1. 「TaskPaneSample.xml」 を開いて、以下のハイライトされた箇所を編集します。    * [ProviderName] 要素の値に「Office 用アプリを提供する個人または組織の名前」を記述します    * [DefaultLocale] 要素の値に対象のユーザーのロケーションを記述します。    * [DisplayName] 要素の [DefaultValue] 属性に「Office 用アプリの表示名」を記述します    * [Description] 要素の [DefaultValue] 属性に「Office 用アプリの説明」を記述します  |  | | --- | | <?xml version="1.0" encoding="UTF-8"?>  <OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="TaskPaneApp">    <Id>15e92f9d-6884-423b-abf0-d4b83496c82f</Id>    <Version>1.0</Version>    <ProviderName>Your Company Name</ProviderName>    <DefaultLocale>ja-JP</DefaultLocale>    <DisplayName DefaultValue="TaskPaneSample" />    <Description DefaultValue="TaskPaneSample Description"/>    <IconUrl DefaultValue="~remoteAppUrl/Images/TaskPaneSample.png" />    <Capabilities>      <Capability Name="Workbook" />    </Capabilities>    <DefaultSettings>      <SourceLocation DefaultValue="~remoteAppUrl/Pages/TaskPaneSample.html" />    </DefaultSettings>    <Permissions>ReadWriteDocument</Permissions>  </OfficeApp> |   **参考**  今回作成するOffice 用アプリは、Office ドキュメントへの読み書きを行うので、「RequestedCapabilities」の要素の値を「ReadWriteDocument」に設定します。 |

|  |  |  |
| --- | --- | --- |
| **ステップ3：HTML の実装** | | |
| 1 | 1. 「Pages」 - 「TaskPaneSample.html」を開いて、プロジェクト テンプレート作成時に追加される以下のハイライトされた個所を削除します。    * [body] 要素内の [div] 要素および [div] 要素内の [input] 要素３箇所を削除します  |  | | --- | | <body>  <div id="Content">  <input type='button' value='Set data' id="setDatabtn" style="margin-right: 10px; padding: 0px; width: 100px;" />  <input type='button' value='Get data' id="getDatabtn" style="padding: 0px; width: 100px;" />  <input type="text" id="selectedData" readonly="readonly" style="margin-top: 10px;width:210px" />  </div>  </body> |  1. 「TaskPaneSample.html」に以下のハイライトされた箇所について追加を行います。    * [body] 要素内に 「div」 要素を記述します    * [div] 要素内に 「input」 要素を記述して 「type="button"」属性、「value="検索"」属性、「onclick="getQuery();"」属性を記述します  |  | | --- | | <body>      <div>        <input type="button" value="検索" onclick="getQuery();" />      </div>      <div id="results"></div>    </body> | |

**参考**

「Visual Studio 2012 による Office 用アプリの開発」では基本的な Office 用アプリ開発手法の習得のために、プロジェクト テンプレートにより作成されたコントロールを削除し、コントロールを作成する手順から紹介します。

|  |  |
| --- | --- |
| **ステップ4：JavaScript の実装** | |
| 1 | 1. 「Scripts」 - 「TaskPaneSample.js」を開いて、[getData] 関数の下に以下の関数を追加します。  |  | | --- | | function getQuery() {      Office.context.document.getSelectedDataAsync("text", function (asyncResult) {          if (asyncResult.status == "failed") {              writeToPage("Error:" + asyncResult.error.message);          }          else {              writeToPage(asyncResult.value);          }      });  }  function writeToPage(text) {      document.getElementById('results').innerText = text;  } |   ドキュメントの値は、[getSelectedDataAsync] 関数を使用して取得することができます。[getSelectedDataAsync] 関数は、ドキュメントで選択中の値を取得する関数です。上記は、[検索] ボタンのクリックでドキュメントの選択中の値をポップアップ表示します。  **参考**  「getSelectedDataAsync」 の詳細は以下のサイト (英語) をご参照ください。   * 「Document.getSelectedDataAsync method (apps for Office)」   <http://msdn.microsoft.com/en-us/library/office/apps/fp142294(v=office.15)> |

|  |  |
| --- | --- |
| **ステップ5：動作確認** | |
| 1 | 1. [デバッグ] メニューから、[デバッグの開始] を選択し、デバッグを開始します。 2. ワークシートの任意のセルに 任意の文字列（ここでは、「東京都港区港南２」）と入力します。        1. 任意の文字列を入力したセルを選択し、作業ウィンドウ アプリの [検索] ボタンをクリックすると、「東京都港区港南２」が作業ウィンドウ アプリに表示されます。 |

以上で、このステップは終了です。

完成コード

動作確認で手順通り動作しない場合は、以下の完成コードを参照し、コードを正しく修正してください。

* + TaskPaneSample.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="TaskPaneApp">    <Id>15e92f9d-6884-423b-abf0-d4b83496c82f</Id>    <Version>1.0</Version>    <ProviderName>Your Company Name</ProviderName>    <DefaultLocale>ja-JP</DefaultLocale>    <DisplayName DefaultValue="TaskPaneSample" />    <Description DefaultValue="TaskPaneSample Description"/>    <IconUrl DefaultValue="~remoteAppUrl/Images/TaskPaneSample.png" />    <Capabilities>      <Capability Name="Workbook" />    </Capabilities>    <DefaultSettings>      <SourceLocation DefaultValue="~remoteAppUrl/Pages/TaskPaneSample.html" />    </DefaultSettings>    <Permissions>ReadWriteDocument</Permissions>  </OfficeApp> |

* + TaskPaneSample.html

|  |
| --- |
| <!DOCTYPE html>    <html>    <head>      <meta charset="UTF-8" />      <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>      <title>TaskPaneSample</title>        <link rel="stylesheet" type="text/css" href="../Content/Office.css" />      <link rel="stylesheet" type="text/css" href="../Content/App.css" />        <script type="text/JavaScript" src="../Scripts/jquery-1.6.2.js"></script>      <script type="text/JavaScript" src="../Scripts/Office/MicrosoftAjax.js"></script>      <script type="text/JavaScript" src="../Scripts/Office/Office.js"></script>      <script type="text/JavaScript" src="../Scripts/TaskPaneSample.js"></script>    </head>    <body>      <div>          <input type="button" value="検索" onclick="getQuery();" />      </div>      <div id="results"></div>    </body>  </html> |

* + TaskPaneSample.js

|  |
| --- |
| // Add any initialization logic to this function.  Office.initialize = function (reason) {  // Checks for the DOM to load.  $(document).ready(function () {  $("#getDatabtn").click(function () { getData("selectedData"); });  // Checks if setSelectedDataAsync is supported and adds appropriate click handler  if (Office.context.document.setSelectedDataAsync) {  $("#setDatabtn").click(function () { setData("Sample data"); });  }  else {  $("#setDatabtn").remove();  }  });  }  // Writes data to current selection.  function setData(dataToInsert) {  Office.context.document.setSelectedDataAsync(dataToInsert);  }  // Reads data from current selection.  function getData(elementIdToUpdate) {  Office.context.document.getSelectedDataAsync(Office.CoercionType.Text,  function (result) {  if (result.status == "succeeded") {  document.getElementById(elementIdToUpdate).value = result.value;  }  });  }  function getQuery() {      Office.context.document.getSelectedDataAsync("text", function (asyncResult) {          if (asyncResult.status == "failed") {              writeToPage("Error:" + asyncResult.error.message);          }          else {              writeToPage(asyncResult.value);          }      });  }  function writeToPage(text) {      document.getElementById('results').innerText = text;  } |

### 外部サービスとの連携

ドキュメントから取得したデータをもとに、「Bing Maps API」を利用して、外部サービスと連携して、地図検索を行う Office 用アプリの実装を行います。

|  |
| --- |
| ステップ1：HTML の実装 |
| ステップ2：JavaScript の実装 |
| ステップ3：動作確認 |

「ドキュメントからのデータの取得」で作成したプロジェクトを利用します。「Bing Maps API」を利用するため、JavaScriptの実装では、「Bing Maps Key」が必要になります。

|  |  |
| --- | --- |
| **ステップ1：HTML の実装** | |
| 1 | 1. 「Bing Maps AJAX コントロール」と「jQuery ライブラリ」を利用するために、「Pages」 - 「TaskPaneSample.html」を開いて 、[head] 要素内 に、ハイライトされた箇所を記述します。    * [head] 要素内に [script] 要素を記述し、「type="text/JavaScript"」属性、「src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0&amp;mkt=ja-jp"」属性を記述します    * [head] 要素内に [script] 要素を記述し、「type="text/JavaScript"」属性、「src=" http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.6.2.js "」属性を記述します  |  | | --- | | <head>      <meta charset="UTF-8" />      <meta http-equiv="X-UA-Compatible" content="IE=Edge" />      <title>TaskPaneSample</title>        <link rel="stylesheet" type="text/css" href="../Content/Office.css" />      <link rel="stylesheet" type="text/css" href="../Content/App.css" />        <script type="text/JavaScript" src="../Scripts/jquery-1.6.2.js"></script>      <script type="text/JavaScript" src="../Scripts/Office/MicrosoftAjax.js"></script>      <script type="text/JavaScript" src="../Scripts/Office/Office.js"></script>      <script type="text/JavaScript" src="../Scripts/TaskPaneSample.js"></script>        <script type="text/JavaScript" src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0&amp;mkt=ja-jp"></script>      <script type="text/JavaScript" src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.6.2.js"></script>    </head> | |
| 2 | 1. 「Pages」 - 「TaskPaneSample.html」を開いて、 [body] 要素 に、ハイライトされた箇所を記述します。    * [body] 要素内に 「div」 要素を記述し、「id="map"」 属性、「 style="position: relative; width: 512px; height: 512px"」 属性を記述します  |  | | --- | | <body>      <div>          <input type="button" value="検索" onclick="getQuery();" />      </div>      <div id="results"></div>      <div id="map" style="position: relative; width: 512px; height: 512px"></div>  </body> | |

|  |  |
| --- | --- |
| **ステップ2：JavaScript の実装** | |
| 1 | 1. 「Scripts」 - 「TaskPaneSample.js」を開いて、Office.initializeのデリゲートメソッドに、以下のハイライトされている箇所を追加します。  |  | | --- | | // Add any initialization logic to this function.  Office.initialize = function (reason) {  // Checks for the DOM to load.  $(document).ready(function () {  $("#getDatabtn").click(function () { getData("selectedData"); });  // Checks if setSelectedDataAsync is supported and adds appropriate click handler  if (Office.context.document.setSelectedDataAsync) {  $("#setDatabtn").click(function () { setData("Sample data"); });  }  else {  $("#setDatabtn").remove();  }  getMap();  });  } |  1. [getQuery] 関数を以下のように変更します。ハイライトされた箇所が変更の必要な箇所です。    * [getQuery] 関数内の if 文の else 句を以下のハイライトされた箇所のとおり、変更します  |  | | --- | | function getQuery() {      Office.context.document.getSelectedDataAsync("text", function (asyncResult) {          if (asyncResult.status == "failed") {              writeToPage("Error: " + asyncResult.error.message);          }          else {              // remove object on map              getMap();              map.entities.clear();              map.getCredentials(function (credentials) { createGeocodeRequest(credentials, [asyncResult.value]); });          }      });  } |  1. 「TaskPaneSample.js」の [writeToPage] 関数の下に以下のコードを追加します。「Your Bing Maps Key Here」には、事前に取得した Keyを記述してください。  |  | | --- | | var map = null;  var searched = null; // searched location  function getMap() {      var options = {          credentials: "Your Bing Maps Key"      };      var mElement = document.getElementById("map");      map = new Microsoft.Maps.Map(mElement, options);  } |   [getMap] 関数はページロード時に実行される処理になります。ここでは、「Bing Maps Ajax Control」のオブジェクトの生成を行います。   1. 「TaskPaneSample.js」の [getMap] 関数の下に以下の関数を追加します。  |  | | --- | | function createGeocodeRequest(credentials, query) {      if (credentials === null) {          writeToPage("Credentials is null.");          return;      }        $.each(query, function () {          var word = String(this);          if (word != "") {              $.ajax({                  type: "GET",                  url: "http://dev.virtualearth.net/REST/v1/Locations",                  dataType: "jsonp",                  data: {                      key: credentials,                      query: word,                      c: "ja-JP",                      o: "json"                  },                  jsonp: "jsonp",                  success: function (data, dataType) {                      geocodeCallback(data, query.length == 1);                  }              });          }      });  } |   [createGeocodeRequest] 関数は、[検索] ボタンのクリックにより、取得したドキュメントの値（住所）をパラメーターに、Bing Map API Restservice（外部サービス）を呼び出す関数です。結果は、コールバック関数の [geocodeCallBack] 関数で受け取ります。   1. 「TaskPaneSample.js」の [createGeocodeRequest] 関数の下に以下の関数を追加します。  |  | | --- | | function geocodeCallback(response, isone) {      if (response && response.errorDetails) {          var text = "";          $.each(response.errorDetails, function () {              text += this + "\n";          });          writeToPage(text) // error!          return false;      }        if (response &&          response.resourceSets &&          response.resourceSets.length > 0 &&          response.resourceSets[0].resources &&          response.resourceSets[0].resources.length > 0) {            // setting range of map          if (isone) {              var bbox = response.resourceSets[0].resources[0].bbox;              var bounds = Microsoft.Maps.LocationRect.fromLocations(          new Microsoft.Maps.Location(bbox[0], bbox[1]), new Microsoft.Maps.Location(bbox[2], bbox[3]));              map.setView({ bounds: bounds });          }            // add pushpin          var code = "A".charCodeAt(0);          $.each(response.resourceSets[0].resources, function () {              var point = this.point;              var location = new Microsoft.Maps.Location(point.coordinates[0], point.coordinates[1]);              searched = location;              var pin = new Microsoft.Maps.Pushpin(location, { text: String.fromCharCode(code++) });              map.entities.push(pin);          });      }  } |   [geocodeCallback] 関数は、「Bing Maps Api RestServices」のコールバック関数になります。検索のレスポンス結果をもとに、地図の縮尺とピンを設定します。 |
| **ステップ3：動作確認** | |
| 1 | 1. [デバッグ] メニューから、[ デバッグの開始] を選択し、デバッグを開始します。 2. ワークシートの任意のセルに 任意の文字列（ここでは、「東京都港区港南２」）と入力します。 3. 入力したセルにカーソルを合わせた状態で、[検索] ボタンをクリックすると、作業ウィンドウ アプリ上に住所の検索結果としてプッシュされたピンを確認できます。 |

以上で、このステップは終了です。

* 完成コード

動作確認で手順通り動作しない場合は、以下の完成コードを参照し、コードを正しく修正してください。

* + TaskPaneSample.xml

前回から変更ありません。

* + TaskPaneSample.html

前回から変更ありません。

* + TaskPaneSample.js

|  |
| --- |
| // Add any initialization logic to this function.  Office.initialize = function (reason) {  // Checks for the DOM to load.  $(document).ready(function () {  $("#getDatabtn").click(function () { getData("selectedData"); });  // Checks if setSelectedDataAsync is supported and adds appropriate click handler  if (Office.context.document.setSelectedDataAsync) {  $("#setDatabtn").click(function () { setData("Sample data"); });  }  else {  $("#setDatabtn").remove();  }  getMap();  });  }  // Writes data to current selection.  function setData(dataToInsert) {  Office.context.document.setSelectedDataAsync(dataToInsert);  }  // Reads data from current selection.  function getData(elementIdToUpdate) {  Office.context.document.getSelectedDataAsync(Office.CoercionType.Text,  function (result) {  if (result.status == "succeeded") {  document.getElementById(elementIdToUpdate).value = result.value;  }  });  }  function getQuery() {  Office.context.document.getSelectedDataAsync("text", function (asyncResult) {  if (asyncResult.status == "failed") {  writeToPage("Error:" + asyncResult.error.message);  }  else {  getMap();  map.entities.clear();  map.getCredentials(function (credentials) { createGeocodeRequest(credentials, [asyncResult.value]); });  }  });  }  function writeToPage(text) {  document.getElementById('results').innerText = text;  }  var map = null;  var searched = null; // searched location  function getMap() {  var options = {  credentials: "Your Bing Maps Key"  };  var mElement = document.getElementById("map");  map = new Microsoft.Maps.Map(mElement, options);  }  function createGeocodeRequest(credentials, query) {  if (credentials === null) {  writeToPage("Credentials is null.");  return;  }  $.each(query, function () {  var word = String(this);  if (word != "") {  $.ajax({  type: "GET",  url: "http://dev.virtualearth.net/REST/v1/Locations",  dataType: "jsonp",  data: {  key: credentials,  query: word,  c: "ja-JP",  o: "json"  },  jsonp: "jsonp",  success: function (data, dataType) {  geocodeCallback(data, query.length == 1);  }  });  }  });  }  function geocodeCallback(response, isone) {  if (response && response.errorDetails) {  var text = "";  $.each(response.errorDetails, function () {  text += this + "\n";  });  writeToPage(text) // error!  return false;  }  if (response &&  response.resourceSets &&  response.resourceSets.length > 0 &&  response.resourceSets[0].resources &&  response.resourceSets[0].resources.length > 0) {  // setting range of map  if (isone) {  var bbox = response.resourceSets[0].resources[0].bbox;  var bounds = Microsoft.Maps.LocationRect.fromLocations(  new Microsoft.Maps.Location(bbox[0], bbox[1]), new Microsoft.Maps.Location(bbox[2], bbox[3]));  map.setView({ bounds: bounds });  }  // add pushpin  var code = "A".charCodeAt(0);  $.each(response.resourceSets[0].resources, function () {  var point = this.point;  var location = new Microsoft.Maps.Location(point.coordinates[0], point.coordinates[1]);  searched = location;  var pin = new Microsoft.Maps.Pushpin(location, { text: String.fromCharCode(code++) });  map.entities.push(pin);  });  }  }  function geocodeCallback(response, isone) {  if (response && response.errorDetails) {  var text = "";  $.each(response.errorDetails, function () {  text += this + "\n";  });  writeToPage(text) // error!  return false;  }  if (response &&  response.resourceSets &&  response.resourceSets.length > 0 &&  response.resourceSets[0].resources &&  response.resourceSets[0].resources.length > 0) {  // setting range of map  if (isone) {  var bbox = response.resourceSets[0].resources[0].bbox;  var bounds = Microsoft.Maps.LocationRect.fromLocations(  new Microsoft.Maps.Location(bbox[0], bbox[1]), new Microsoft.Maps.Location(bbox[2], bbox[3]));  map.setView({ bounds: bounds });  }  // add pushpin  var code = "A".charCodeAt(0);  $.each(response.resourceSets[0].resources, function () {  var point = this.point;  var location = new Microsoft.Maps.Location(point.coordinates[0], point.coordinates[1]);  searched = location;  var pin = new Microsoft.Maps.Pushpin(location, { text: String.fromCharCode(code++) });  map.entities.push(pin);  });  }  } |

### ドキュメントへのデータ設定

外部サービスから取得した結果を加工し、ブラウザ表示用の URL を ドキュメントへ設定する機能の実装を行います。

|  |
| --- |
| ステップ1：HTML の実装 |
| ステップ2：JavaScript の実装 |
| ステップ3：動作確認 |

「外部サービスとの連携」で作成したプロジェクトを利用します。UI に値を設定するボタンを配置し、[setSelectedDataAsync] 関数を利用してドキュメントへのデータの設定を行います。

|  |  |
| --- | --- |
| **ステップ1：HTML の実装** | |
| 1 | 1. 「Pages」 - 「TaskPaneSample.html」を開いて、[body] 要素内にハイライトされた箇所を記述します。    * [div] 要素内に [input] 要素を記述し、「type="button"」属性、「value="検索結果設定"」属性、「onclick="setUrl();"」属性を記述します  |  | | --- | | <body>      <div>          <input type="button" value="検索" onclick="getQuery();" />          <input type="button" value="検索結果設定" onclick="setUrl();" />      </div>      <div id="results"></div>      <div id="map" style="position: relative; width: 512px; height: 512px"></div>  </body> | |

|  |  |
| --- | --- |
| **ステップ2：JavaScript の実装** | |
| 1 | 1. 「Scripts」 - 「TaskPaneSample.js」を開いて、[geocodeCallback] 関数の下に以下の関数を追加します。  |  | | --- | | function setUrl() {      var url = "http://www.bing.com/maps/default.aspx?q=" + encodeURI(searched.latitude + "," + searched.longitude) + "&x=0&y=0&form=MSNH69&qs=n&sk=&mkt=ja-JP";      Office.context.document.setSelectedDataAsync(url, { coercionType: Office.CoercionType.Text }, function (asyncResult) {          if (asyncResult.status == "failed") {              writeToPage(asyncResult.error.message);          }      });  } |   ドキュメントへの設定は、[setSelectedDataAsync] 関数を使用します。 |

**参考**

「setSelectedDataAsync」の詳細は以下のサイト (英語) をご参照ください。

* 「Document.setSelectedDataAsync method (apps for Office)」

<http://msdn.microsoft.com/en-us/library/office/apps/fp142145(v=office.15)>

|  |  |
| --- | --- |
| **ステップ3：動作確認** | |
| 1 | 1. [デバッグ] メニューから、[デバッグの開始] を選択し、デバッグを開始します。 2. ワークシートの任意のセルに 任意の文字列（ここでは、「東京都港区港南２」）と入力します。 3. 入力したセルにカーソルを合わせた状態で、[検索] ボタンをクリックします。 4. 任意のブランクのセルにカーソルを移動して、[検索結果設定] ボタンをクリックします。 5. 選択中のセルに 検索結果のURL が設定されることを確認できます。 |

以上で、このステップは終了です。

* 完成コード

動作確認で手順通り動作しない場合は、以下の完成コードを参照し、コードを正しく修正してください。

* + TaskPaneSample.xml

前回から変更ありません。

* + TaskPaneSample.html

|  |
| --- |
| <!DOCTYPE html>    <html>  <head>      <meta charset="UTF-8" />      <meta http-equiv="X-UA-Compatible" content="IE=Edge" />      <title>TaskPaneSample</title>        <link rel="stylesheet" type="text/css" href="../Content/Office.css" />      <link rel="stylesheet" type="text/css" href="../Content/App.css" />        <script type="text/JavaScript" src="../Scripts/jquery-1.6.2.js"></script>      <script type="text/JavaScript" src="../Scripts/Office/MicrosoftAjax.js"></script>      <script type="text/JavaScript" src="../Scripts/Office/Office.js"></script>      <script type="text/JavaScript" src="../Scripts/TaskPaneSample.js"></script>        <script type="text/JavaScript" src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0&amp;mkt=ja-jp"></script>      <script type="text/JavaScript" src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.6.2.js"></script>    </head>  <body>      <div>          <input type="button" value="検索" onclick="getQuery();" />          <input type="button" value="検索結果設定" onclick="setUrl();" />      </div>      <div id="results"></div>      <div id="map" style="position: relative; width: 512px; height: 512px"></div>  </body>  </html> |

* + TaskPaneSample.js

|  |
| --- |
| // Add any initialization logic to this function.  Office.initialize = function (reason) {  // Checks for the DOM to load.  $(document).ready(function () {  $("#getDatabtn").click(function () { getData("selectedData"); });  // Checks if setSelectedDataAsync is supported and adds appropriate click handler  if (Office.context.document.setSelectedDataAsync) {  $("#setDatabtn").click(function () { setData("Sample data"); });  }  else {  $("#setDatabtn").remove();  }  getMap();  });  }  // Writes data to current selection.  function setData(dataToInsert) {  Office.context.document.setSelectedDataAsync(dataToInsert);  }  // Reads data from current selection.  function getData(elementIdToUpdate) {  Office.context.document.getSelectedDataAsync(Office.CoercionType.Text,  function (result) {  if (result.status == "succeeded") {  document.getElementById(elementIdToUpdate).value = result.value;  }  });  }  function getQuery() {  Office.context.document.getSelectedDataAsync("text", function (asyncResult) {  if (asyncResult.status == "failed") {  writeToPage("Error:" + asyncResult.error.message);  }  else {  getMap();  map.entities.clear();  map.getCredentials(function (credentials) { createGeocodeRequest(credentials, [asyncResult.value]); });  }  });  }  function writeToPage(text) {  document.getElementById('results').innerText = text;  }  var map = null;  var searched = null; // searched location  function getMap() {  var options = {  credentials: "Your Bing Maps Key"  };  var mElement = document.getElementById("map");  map = new Microsoft.Maps.Map(mElement, options);  }  function createGeocodeRequest(credentials, query) {  if (credentials === null) {  writeToPage("Credentials is null.");  return;  }  $.each(query, function () {  var word = String(this);  if (word != "") {  $.ajax({  type: "GET",  url: "http://dev.virtualearth.net/REST/v1/Locations",  dataType: "jsonp",  data: {  key: credentials,  query: word,  c: "ja-JP",  o: "json"  },  jsonp: "jsonp",  success: function (data, dataType) {  geocodeCallback(data, query.length == 1);  }  });  }  });  }  function geocodeCallback(response, isone) {  if (response && response.errorDetails) {  var text = "";  $.each(response.errorDetails, function () {  text += this + "\n";  });  writeToPage(text) // error!  return false;  }  if (response &&  response.resourceSets &&  response.resourceSets.length > 0 &&  response.resourceSets[0].resources &&  response.resourceSets[0].resources.length > 0) {  // setting range of map  if (isone) {  var bbox = response.resourceSets[0].resources[0].bbox;  var bounds = Microsoft.Maps.LocationRect.fromLocations(  new Microsoft.Maps.Location(bbox[0], bbox[1]), new Microsoft.Maps.Location(bbox[2], bbox[3]));  map.setView({ bounds: bounds });  }  // add pushpin  var code = "A".charCodeAt(0);  $.each(response.resourceSets[0].resources, function () {  var point = this.point;  var location = new Microsoft.Maps.Location(point.coordinates[0], point.coordinates[1]);  searched = location;  var pin = new Microsoft.Maps.Pushpin(location, { text: String.fromCharCode(code++) });  map.entities.push(pin);  });  }  }  function geocodeCallback(response, isone) {  if (response && response.errorDetails) {  var text = "";  $.each(response.errorDetails, function () {  text += this + "\n";  });  writeToPage(text) // error!  return false;  }  if (response &&  response.resourceSets &&  response.resourceSets.length > 0 &&  response.resourceSets[0].resources &&  response.resourceSets[0].resources.length > 0) {  // setting range of map  if (isone) {  var bbox = response.resourceSets[0].resources[0].bbox;  var bounds = Microsoft.Maps.LocationRect.fromLocations(  new Microsoft.Maps.Location(bbox[0], bbox[1]), new Microsoft.Maps.Location(bbox[2], bbox[3]));  map.setView({ bounds: bounds });  }  // add pushpin  var code = "A".charCodeAt(0);  $.each(response.resourceSets[0].resources, function () {  var point = this.point;  var location = new Microsoft.Maps.Location(point.coordinates[0], point.coordinates[1]);  searched = location;  var pin = new Microsoft.Maps.Pushpin(location, { text: String.fromCharCode(code++) });  map.entities.push(pin);  });  }  }  function setUrl() {  var url = "http://www.bing.com/maps/default.aspx?q=" + encodeURI(searched.latitude + "," + searched.longitude) + "&x=0&y=0&form=MSNH69&qs=n&sk=&mkt=ja-JP";  Office.context.document.setSelectedDataAsync(url, { coercionType: Office.CoercionType.Text }, function (asyncResult) {  if (asyncResult.status == "failed") {  writeToPage(asyncResult.error.message);  }  });  } |

### データのバインド

データバインドを利用すると、Office 用アプリとOffice ドキュメントデータの関連付けを行うことができます。ドキュメント内のデータの値が変更された場合、自動的にUI（Office 用アプリ）に反映することもできます。

ここでは、バインドした住所の値が変更されると、地図にプッシュされたピンを再設定する機能の実装を行います。

|  |
| --- |
| ステップ1：HTML の実装 |
| ステップ2：JavaScript の実装 |
| ステップ3：動作確認 |

「ドキュメントへのデータ設定」で作成したプロジェクトを利用します。UI にデータバインドを行うボタンを配置します。データバインドは、[addFromSelectionAsync] 関数を利用します。バインドデータの変更時の処理は、イベントハンドラを追加して実現します。

|  |  |
| --- | --- |
| **ステップ1：HTML の実装** | |
| 1 | 1. 「Pages」 - 「TaskPaneSample.html」を開いて、[body] 要素内に、ハイライトされた箇所を記述します。    * [div] 要素内に [input] 要素を記述し、「type="button"」属性、「value="データバインド"」属性、「onclick="bindData();"」属性を記述します  |  | | --- | | <body>      <div>          <input type="button" value="検索" onclick="getQuery();" />          <input type="button" value="検索結果設定" onclick="setUrl();" />          <input type="button" value="データバインド" onclick="bindData();" />      </div>      <div id="results"></div>      <div id="map" style="position: relative; width: 512px; height: 512px"></div>  </body> | |

|  |  |
| --- | --- |
| **ステップ2：JavaScript の実装** | |
| 1 | 1. 「Scripts」 - 「TaskPaneSample.js」を開いて、[geocodeCallback] 関数の下に以下の関数を追加します。  |  | | --- | | function bindData() {      Office.context.document.bindings.addFromSelectionAsync(Office.BindingType.Matrix, { id: 'myBinding' }, function (asyncResult) {          if (asyncResult.status == "failed") {              writeToPage("Error: " + asyncResult.error.message);              return;          }      });        // add handler      Office.select("bindings#myBinding").addHandlerAsync("bindingDataChanged", bindDataChanged);      // show display      bindDataChanged();  } |   データバインドは、[addFromSelectionAsync] 関数を使用します。また、[addHandlerAsync] 関数を利用することで、バインドしたオブジェクトにイベントハンドラの追加を行うことができます。ここでは、バインドデータが変更された際のイベントハンドラを追加します。バインドしたオブジェクトの参照は、[Office.select] 関数を利用します。   1. 「TaskPaneSample.js」を開いて、[bindData] 関数の下に以下のコードを追加します。  |  | | --- | | function bindDataChanged() {      results = new Array();      var binding = Office.select("bindings#myBinding");      if (binding) {          binding.getDataAsync(function (dataResult) {              // remove object on map              map.entities.clear();              map.getCredentials(function (credentials) { createGeocodeRequest(credentials, dataResult.value); });          });      }  } |   データバインドオブジェクトからデータを参照する場合は、[getDataAsync] 関数を利用します。 |

**参考**

「addFromSelectionAsync」、「getDataAsync」の詳細は以下のサイト (英語) をご参照ください。

* 「Bindings.addFromSelectionAsync method (apps for Office)」

<http://msdn.microsoft.com/en-us/library/office/apps/fp142282(v=office.15)>

* 「Binding.getDataAsync method (apps for Office)」

<http://msdn.microsoft.com/en-us/library/office/apps/fp161073(v=office.15)>

|  |  |
| --- | --- |
| **ステップ3：動作確認** | |
| 1 | 1. [デバッグ] メニューから、[ デバッグの開始] を選択し、デバッグを開始します。 2. ワークシートの任意のセルに 存在する任意の住所（ここでは、「東京都渋谷区代々木２」、「東京都千代田区大手町１」、「東京都調布市調布ヶ丘１」）を入力します。 3. 入力したセルにカーソルを合わせた状態で、[検索] ボタンをクリックします。 4. 上記で入力したセルを選択し、作業ウィンドウ アプリの [バインド] ボタンをクリックします。      1. セルの住所を変更することで、（ここでは、東京都渋谷区代々木２→東京都港区港南２と変更する）ピンが新しい住所で、再マップすることを確認できます。選択中のセルに URL が設定されることを確認できます。 |

以上で、このステップは終了です。このようにデータバインドを利用すると、ドキュメントのデータの変更に対するOffice 用アプリの描画などの処理を自動化できます。ドキュメントと Office 用アプリを動的に連携させたいシナリオで活用できます。

* 完成コード

動作確認で手順通り動作しない場合は、以下の完成コードを参照し、コードを正しく修正してください。

* + TaskPaneSample.xml

前回から変更ありません。

* + TaskPaneSample.html

|  |
| --- |
| <!DOCTYPE html>    <html>  <head>      <meta charset="UTF-8" />      <meta http-equiv="X-UA-Compatible" content="IE=Edge" />      <title>TaskPaneSample</title>        <link rel="stylesheet" type="text/css" href="../Content/Office.css" />      <link rel="stylesheet" type="text/css" href="../Content/App.css" />        <script type="text/JavaScript" src="../Scripts/jquery-1.6.2.js"></script>      <script type="text/JavaScript" src="../Scripts/Office/MicrosoftAjax.js"></script>      <script type="text/JavaScript" src="../Scripts/Office/Office.js"></script>      <script type="text/JavaScript" src="../Scripts/TaskPaneSample.js"></script>        <script type="text/JavaScript" src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0&amp;mkt=ja-jp"></script>      <script type="text/JavaScript" src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.6.2.js"></script>    </head>  <body>      <div>          <input type="button" value="検索" onclick="getQuery();" />          <input type="button" value="検索結果設定" onclick="setUrl();" />          <input type="button" value="データバインド" onclick="bindData();" />      </div>      <div id="results"></div>      <div id="map" style="position: relative; width: 512px; height: 512px"></div>  </body>  </html> |

* + TaskPaneSample.js

|  |
| --- |
| // Add any initialization logic to this function.  Office.initialize = function (reason) {  // Checks for the DOM to load.  $(document).ready(function () {  $("#getDatabtn").click(function () { getData("selectedData"); });  // Checks if setSelectedDataAsync is supported and adds appropriate click handler  if (Office.context.document.setSelectedDataAsync) {  $("#setDatabtn").click(function () { setData("Sample data"); });  }  else {  $("#setDatabtn").remove();  }  getMap();  });  }  // Writes data to current selection.  function setData(dataToInsert) {  Office.context.document.setSelectedDataAsync(dataToInsert);  }  // Reads data from current selection.  function getData(elementIdToUpdate) {  Office.context.document.getSelectedDataAsync(Office.CoercionType.Text,  function (result) {  if (result.status == "succeeded") {  document.getElementById(elementIdToUpdate).value = result.value;  }  });  }  function getQuery() {  Office.context.document.getSelectedDataAsync("text", function (asyncResult) {  if (asyncResult.status == "failed") {  writeToPage("Error:" + asyncResult.error.message);  }  else {  getMap();  map.entities.clear();  map.getCredentials(function (credentials) { createGeocodeRequest(credentials, [asyncResult.value]); });  }  });  }  function writeToPage(text) {  document.getElementById('results').innerText = text;  }  var map = null;  var searched = null; // searched location  function getMap() {  var options = {  credentials: "Your Bing Maps Key"  };  var mElement = document.getElementById("map");  map = new Microsoft.Maps.Map(mElement, options);  }  function createGeocodeRequest(credentials, query) {  if (credentials === null) {  writeToPage("Credentials is null.");  return;  }  $.each(query, function () {  var word = String(this);  if (word != "") {  $.ajax({  type: "GET",  url: "http://dev.virtualearth.net/REST/v1/Locations",  dataType: "jsonp",  data: {  key: credentials,  query: word,  c: "ja-JP",  o: "json"  },  jsonp: "jsonp",  success: function (data, dataType) {  geocodeCallback(data, query.length == 1);  }  });  }  });  }  function geocodeCallback(response, isone) {  if (response && response.errorDetails) {  var text = "";  $.each(response.errorDetails, function () {  text += this + "\n";  });  writeToPage(text) // error!  return false;  }  if (response &&  response.resourceSets &&  response.resourceSets.length > 0 &&  response.resourceSets[0].resources &&  response.resourceSets[0].resources.length > 0) {  // setting range of map  if (isone) {  var bbox = response.resourceSets[0].resources[0].bbox;  var bounds = Microsoft.Maps.LocationRect.fromLocations(  new Microsoft.Maps.Location(bbox[0], bbox[1]), new Microsoft.Maps.Location(bbox[2], bbox[3]));  map.setView({ bounds: bounds });  }  // add pushpin  var code = "A".charCodeAt(0);  $.each(response.resourceSets[0].resources, function () {  var point = this.point;  var location = new Microsoft.Maps.Location(point.coordinates[0], point.coordinates[1]);  searched = location;  var pin = new Microsoft.Maps.Pushpin(location, { text: String.fromCharCode(code++) });  map.entities.push(pin);  });  }  }  function geocodeCallback(response, isone) {  if (response && response.errorDetails) {  var text = "";  $.each(response.errorDetails, function () {  text += this + "\n";  });  writeToPage(text) // error!  return false;  }  if (response &&  response.resourceSets &&  response.resourceSets.length > 0 &&  response.resourceSets[0].resources &&  response.resourceSets[0].resources.length > 0) {  // setting range of map  if (isone) {  var bbox = response.resourceSets[0].resources[0].bbox;  var bounds = Microsoft.Maps.LocationRect.fromLocations(  new Microsoft.Maps.Location(bbox[0], bbox[1]), new Microsoft.Maps.Location(bbox[2], bbox[3]));  map.setView({ bounds: bounds });  }  // add pushpin  var code = "A".charCodeAt(0);  $.each(response.resourceSets[0].resources, function () {  var point = this.point;  var location = new Microsoft.Maps.Location(point.coordinates[0], point.coordinates[1]);  searched = location;  var pin = new Microsoft.Maps.Pushpin(location, { text: String.fromCharCode(code++) });  map.entities.push(pin);  });  }  }  function setUrl() {  var url = "http://www.bing.com/maps/default.aspx?q=" + encodeURI(searched.latitude + "," + searched.longitude) + "&x=0&y=0&form=MSNH69&qs=n&sk=&mkt=ja-JP";  Office.context.document.setSelectedDataAsync(url, { coercionType: Office.CoercionType.Text }, function (asyncResult) {  if (asyncResult.status == "failed") {  writeToPage(asyncResult.error.message);  }  });  }  function bindData() {  Office.context.document.bindings.addFromSelectionAsync(Office.BindingType.Matrix, { id: 'myBinding' }, function (asyncResult) {  if (asyncResult.status == "failed") {  writeToPage("Error: " + asyncResult.error.message);  return;  }  });  // add handler  Office.select("bindings#myBinding").addHandlerAsync("bindingDataChanged", bindDataChanged);  // show display  bindDataChanged();  }  function bindDataChanged() {  results = new Array();  var binding = Office.select("bindings#myBinding");  if (binding) {  binding.getDataAsync(function (dataResult) {  // remove object on map  map.entities.clear();  map.getCredentials(function (credentials) { createGeocodeRequest(credentials, dataResult.value); });  });  }  } |

以上が、Visual Studio 2012 を利用した開発になります。Visual Studio 2012 のプロジェクト テンプレートの利用により、コーディングをはじめるまでの準備作業を大幅に削減でき、また、インテリセンスやデバッグ機能を利用することによる生産性の高さを実感できたと思います。

### Office 用アプリの再利用性

「Office 用アプリの概要 - Office 用アプリの種類 - 各 Office 製品がサポートする Office 用アプリの種類」で説明したように、サポートの範囲内であれば、同一の Office 用アプリを複数の Office アプリケーションで利用できます。

ここでは、前節で、Visual Studio 2012 により 作成した作業ウィンドウ アプリを Wordでも利用できることを確認してみましょう。

|  |
| --- |
| ステップ1：XML マニフェスト ファイルの記述 |
| ステップ2：アプリケーションの展開 |
| ステップ3：動作確認 |

対象とする Office アプリケーションは、XML マニフェスト ファイルで定義します。この XML マニフェスト ファイルに要素を追加・削除するだけで、対象アプリケーションを変更することができます。変更した XML マニフェスト ファイルをファイル共有で展開して、Word での動作確認を行います。

|  |  |
| --- | --- |
| **ステップ1：XML マニフェスト ファイルの記述** | |
| 1 | 1. 「TaskPaneSample.xml」を開いて以下のハイライトされた箇所について編集を行います。    * [Capability] 要素を記述し、「Name="Workbook"」属性を記述します    * [IconUrl] 要素の [DefaultValue] 属性に「ローカルの IIS に配置したTaskPaneSample.png のURL（「ステップ２」で設定する）」を記述します    * [SourceLocation] 要素の [DefaultValue] 属性に「ローカルの IIS に配置した TaskPaneSample.html のURL（「ステップ２」で設定する）」を記述します  |  | | --- | | <?xml version="1.0" encoding="UTF-8"?>  <OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="TaskPaneApp">    <Id>15e92f9d-6884-423b-abf0-d4b83496c82f</Id>    <Version>1.0</Version>    <ProviderName>Your Company Name</ProviderName>    <DefaultLocale>ja-JP</DefaultLocale>    <DisplayName DefaultValue="TaskPaneSample" />    <Description DefaultValue="TaskPaneSample Description"/>    <IconUrl DefaultValue="~remoteAppUrl/Images/TaskPaneSample.png" />    <Capabilities>      <Capability Name="Workbook" />      <Capability Name="Document" />    </Capabilities>    <DefaultSettings>      <SourceLocation DefaultValue="~remoteAppUrl/Pages/TaskPaneSample.html" />    </DefaultSettings>    <Permissions>ReadWriteDocument</Permissions>  </OfficeApp> |   HostApp 要素を追加することで、対象のOffice アプリケーションを追加することができます。また、Web アプリケーションは、ローカル PC の IIS に配置するため、SourceLocationをローカルの URL（「ステップ２」で設定する） 変更しておきます。 |

|  |  |
| --- | --- |
| **ステップ2：アプリケーションの展開** | |
| 1 | 1. 作成したWeb アプリケーションをローカル PC の IIS に配置します。前述の「Hello World の展開 - ステップ1：Web サーバーの構成」を参考に、[Default Web Site] に、仮想ディレクトリを作成します。作成に必要なパラメーターを以下に示します。  |  |  |  | | --- | --- | --- | | **項目名** | **設定値** | **備考** | | **エイリアス** | AppsForOffice-vs12 |  | | **物理パス** | { プロジェクトフォルダ― } | TaskPaneSampleWeb プロジェクトフォルダーパス |  1. 「TaskPaneSample.xml」を「c:\AppsForOffice\_catalog」フォルダーに格納します。 2. カタログの設定がまだの場合は、前述の「Hello World の展開 - ステップ2：XML マニフェスト ファイルの配置」を参照してカタログの設定を行ってください。 |

|  |  |
| --- | --- |
| **ステップ3：動作確認** | |
| 1 | 1. Word 2013 を起動します。 2. テンプレートから [白紙の文章] を選択します。 3. [挿入] タブから [Office用アプリ] - [全ての表示] を選択します。 4. [アプリケーションの挿入] ダイアログで、[共有フォルダー] から [Task Pane Sample] を選択します。 5. [挿入] - [表] を選択し、「3行 × 1列」の表を挿入します。 6. 追加した表に、存在する任意の住所を入力します。（ここでは、「東京都渋谷区代々木２」、「東京都千代田区大手町１」、「東京都調布市調布ヶ丘１」） 7. [東京都渋谷区代々木２] を選択して、[検索] ボタンをクリックすることで、地図検索できることを確認できます。 8. ドキュメント内の任意の空白行を選択し、[検索結果設定] ボタンをクリックすることで、ブラウザ表示用の URL が設定されることを確認できます。 9. [東京都渋谷区代々木２] 、[東京都千代田区大手町１] 、[東京都調布市調布ヶ丘１] を選択して、[バインド] ボタンクリック後、[東京都渋谷区代々木２] を [東京都港区港南２] に変更することで、プッシュピンが再設定されることが確認できます。 |

**参考**

Word 2013 ドキュメント内の表データの取得はMatrixデータ型または、Tableデータ型を利用することにより可能です。

## 各 Office 製品に対応する機能範囲

これまでの手順で確認してきた Office 用アプリの主な機能に対する Office製品のサポート範囲は次のとおりです。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **機能** | **Word 2013** | **Excel 2013 / Excel WAC** | **Outlook 2013 / OWA / Mobile OWA** | **Project 2013** |
| **Set / Get (text, table, matrix)** | ○ | ○ |  | ○ |
| **Set / Get (Html, ooxml)** | ○ |  |  |  |
| **Bindings** | ○ | ○ |  |  |
| **Custom Xml Parts** | ○ |  |  |  |
| **Settings** | ○ |  |  |  |
| **App Specific API** |  |  | ○（EWS） | ○（PSI） |

これまでの実装例で説明した [setSelectedDataAsync] 関数と [getSelectedDataAsync] 関数は、「Set / Get (text, table, matrix)」に該当します。Word 2013、Excel 2013、Project 2013 で利用できることがわかります。「Office 用アプリの再利用」において、Excel 2013 用に実装したこれらの関数が、Word 2013 でも動作することを確認できました。「Bindings」（データバインド）についても同様に動作することが確認できます。

**参考**

これまでの学習で出てきてない項目に関しましては、以下のサイト (英語) をご参照ください。

* 「Build apps for Office and SharePoint」

<http://msdn.microsoft.com/en-US/office/apps>

また、Office 用アプリはクロスデバイス対応しているため、デスクトップ用に作成した Office 用アプリも、モバイルで利用できます。変更なしで利用することもできますが、デバイス毎に専用ユーザー インターフェイスの作成も可能です。デバイスに適したデザイン、サイズ、情報量を選択することで、より高度なユーザー エクスペリエンスを実現することができます。

この章では、サンプルコードを利用して、Office 用アプリの開発について学習してきました。Office 用アプリ開発についてすべての機能を解説したわけではありませんが、その特徴や実装する上での注意点、着眼点、ヒントなどを確認できたと思います。ここでの学習内容が、今後の Office 用アプリによるソリューション開発に役立っていただければ幸いです。

# Office 用アプリの展開

## Office 用アプリ展開概要

IT 管理者は Office 用アプリの特性や目的に合わせて展開方法を選択できます。組織の規模や展開対象に応じて展開方法を決定します。決定した展開方法をユーザーに意識させることなく、ユーザーの Office 用アプリへ設定を反映できます。Office 用アプリを展開するため、以下の3つの展開方法が提供されます。

* 共有フォルダー

ファイルサーバーを利用する、最も簡易的な展開方法です。

* 内部ディレクトリ

SharePoint Online または SharePoint を利用する、組織向けの展開方法です。

* Office ストア

Microsoft が提供するグローバルな アプリケーションの販売および管理を行うプラットフォームです。

どの展開方法を適用する場合でも、ユーザーは Office 用アプリやカタログの格納場所を意識せず、Officeアプリケーション内でシームレスにドキュメントへ Office 用アプリを追加できます。

## Office 用アプリの展開の種類

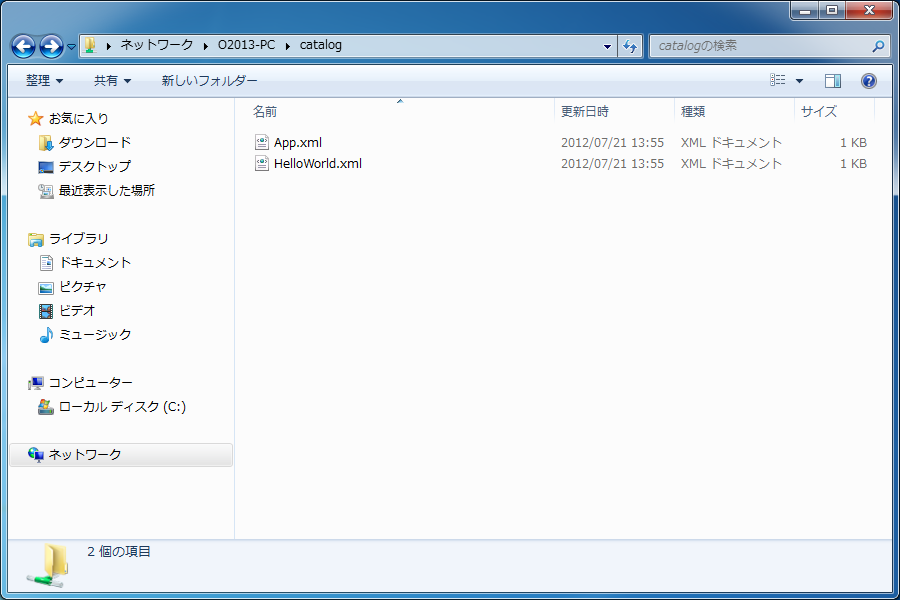
ここでは、Office 用アプリの展開の種類の特徴について説明します。

### 共有フォルダー

共有フォルダーには次の特徴があります。

* + ファイルサーバーによる展開および集中管理

チーム内や、同一拠点内などで利用できる最も簡易的な展開方法です。カタログサーバーとしてファイルサーバーを利用します。



共有フォルダーに XML マニフェスト ファイルを配置し、容易に Office 用アプリを展開できます。しかし、管理対象の Office 用アプリが多くなると、ファイルベースの管理のため、管理が煩雑になってしまうことがあります。共有フォルダーによる展開は他の展開の種類と比較し、容易に展開ができるメリットがあり、小規模な展開に適しています。

### 内部ディレクトリ

内部ディレクトリには次の特徴があります。

* + SharePoint または SharePoint Online による展開および集中管理

組織内の SharePoint または SharePoint Online をカタログサーバーとして利用した展開方法です。

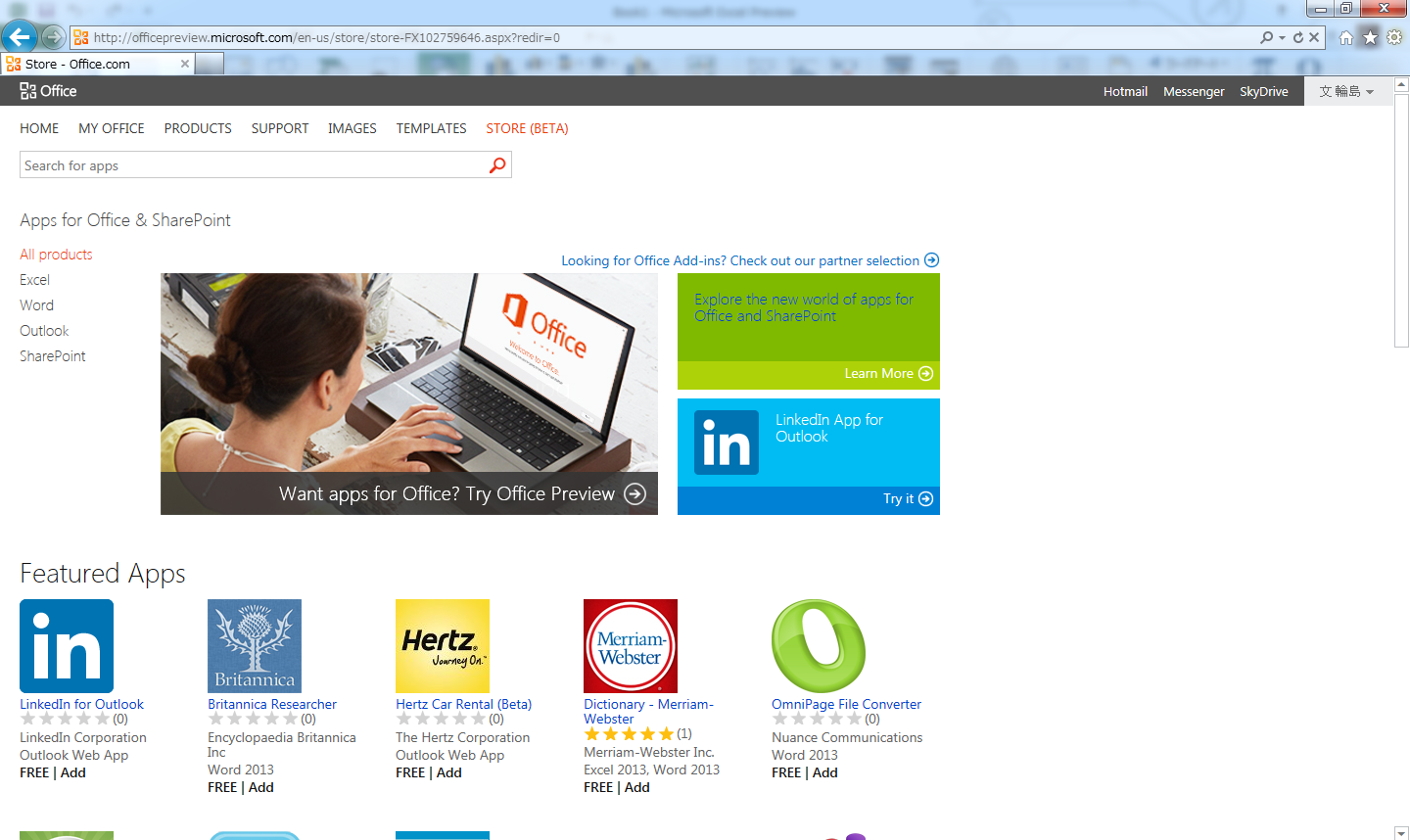


SharePoint または、SharePoint Online を利用している場合、内部ディレクトリに XML マニフェスト ファイルを配置し、組織内へ Office 用アプリを展開できます。 IT 管理者は SharePoint または、SharePoint Online の内部ディレクトリで XML マニフェスト ファイルをリストベースで管理できるため、Office 用アプリを集中的に効率よく管理できます。

### Office ストア

Office ストアには次の特徴があります。

* + グローバルのサードパーティにより豊富な Office 用アプリが公開されている
  + 利用者のレビューおよび評価を参照できる



ユーザーは、Office ストアで提供されている多くの Office 用アプリの中から業務の効率化につながる Office 用アプリを見つけ、利用できます。Office ストアで提供される Office 用アプリにはグローバルなユーザーのレビューや、評価があり、Office 用アプリを選定する際に確認できます。

IT 管理者は、組織内のユーザーに対してOffice ストアで公開されている Office 用アプリの利用を制限できます。グループ ポリシーにより制限ができるため、ユーザーの判断で、自由に Office 用アプリがインストールされることを防ぐことができます。

開発者は、Office ストアを通じて、自身で作成した Office 用アプリを全世界へ向けて公開できます。課金や管理のプラットフォームも用意されているため、少ない投資で、Office ストアの全世界のユーザーへ Office 用アプリを配布、販売できます。

## SharePoint Online による Office 用アプリの展開

組織内への Office 用アプリの展開は、SharePoint または SharePoint Online の内部ディレクトリを利用した展開方法が一般的です。ここでは、SharePoint Online による Office 用アプリの展開について説明します。

### SharePoint Online への登録

組織で SharePoint Online を利用している場合、SharePoint Online をカタログサーバーとして Office 用アプリを配布できます。SharePoint Online の内部ディレクトリに Office 用アプリの XML マニフェスト ファイルを登録します。Office 用アプリの実体である Web ページは、他の展開方法と同様にユーザーが参照可能な Web サーバーへ配置します。

### クライアントの設定

ユーザーは Office アプリケーション内で内部ディレクトリを参照でき、組織向けに公開された Office 用アプリを一覧から取得できます。ユーザーは組織の業務に特化して開発された Office 用アプリを利用できるため、業務を効率化できます。

# まとめ

次期 Office の新機能の Office 用アプリは開発者にとって、既存の Web 標準技術がベースになっているため、既存の知識を流用して開発ができます。加えて、サンドボックスや Permission モデルの実装により、セキュアなソリューションの開発ができるようになっています。Office 用アプリはアドインやマクロに加わる新しい Office アプリケーションのソリューションとして、ソリューションを開発する際に新たな選択肢の１つとして提供されています。さらに、IT 管理者にとって展開の容易さや、集中管理ができるなどの利点があります。組織において既存の基幹システムとの連携は業務効率化の要となっています。Web 標準に準拠したインターフェイスを持つ基幹システムと連携する、組織の業務に特化し、利用者に最適化された Office 用アプリにより、利用者の業務の効率化が可能です。 また、サードパーティ製の Office 用アプリをOffice ストアから組織へ導入することにより、既存のソリューションを利用者に提供でき、生産性の向上や管理費の削減ができます。組織にとって Office 用アプリとは単にアドインやマクロに加わった新しい機能ではなく、開発コストや管理コストを削減できるソリューションになっています。

本ドキュメントでは、Office 用アプリの概要、開発について、また各種展開方法について説明してきました。組織内で Office 用アプリによるソリューション開発と展開を行う際に参考にしてください。

Office 用アプリの利用による作業の効率化、管理の容易さを、ビジネスにおいて十分に活用していただくため、Office アプリケーションによる Office 用アプリの利用をお勧めします。