要約版

ビジネス アプリケーション向け .NET テクノロジ ガイド

2013 年 6 月　Microsoft Corporation
2013 年 9 月　日本語版 Rev.1

Microsoft .NET

目次

概要

現在、テクノロジの使用環境は、クラウドで提供される各種サービスを活用したマルチデバイス エクスペリエンスへの移行途上にあります。また、タッチ操作、センサー、モビリティといったローカル ハードウェアの機能に加え、Web との接続性や、データ ストレージ、メディア ストリーミング、ソーシャル機能のようなバックエンド サービスの性能が、利用パターンに影響を及ぼすようになっています。

こうしたデバイスとサービスの結び付きは、ビジネス シナリオとコンシューマー シナリオの両方で広がりつつあります。コンシューマー環境では、モバイル コンピューティングが普及し、情報の閲覧を目的としたデバイスが次々と開発されたことが契機となり、ハードウェア機能およびテクノロジの発展へとつながりました。一方、企業環境では、IT のコンシューマライゼーションと、個人所有デバイスの業務利用 (BYOD) という 2 つの社会現象によって、コンシューマー エクスペリエンスが、ビジネス コンピューティングおよび基幹業務 (LOB) アプリケーションの進化を後押しするという構図が出来上がっています。

デバイスおよびサービスに依存した次世代のアプリケーションは、単体では成立しません。既存アプリケーションときわめて密接に統合され、新たなユーザーに向けて、または新しい意思疎通の方法として、既存アプリケーションの価値を広げる存在となる必要があります。そのため、すべてのアプリケーション開発者は、2 つの異なるパターンに対処しなくてはなりません。

- **従来型のアプリケーション パターン:** クライアント/サーバー方式などのテクノロジ パターンを使用して開発されたアプリケーションや、デスクトップ ブラウザーに最適化された Web アプリケーション。基盤アプリケーションとして機能するもので、既存の**ビジネス プロセス**に大きな比重を置いて設計されています。
- **次世代型のアプリケーション パターン:** マルチデバイスやクラウドといった新しいテクノロジで実現されるアプリケーション。アプリケーションの対象を**エンド ユーザー**にまで拡大することで、従来型のパターンを補完します。

今日では、エンド ユーザー向けに従来型のパターンが拡張される傾向にあり、これによって開発者が革新的な技術を取り入れ、競合との差別化を図るための絶好の機会が生み出されています。小売、通信、金融、物流、顧客サービスなど、現在のビジネス環境ではあらゆる企業がソフトウェア企業であると言っても過言ではありません。企業が顧客ニーズを満たし、競争を優位に進められるかどうかは、ソフトウェアの革新にかかっています。

ただし、既存のアプリケーションを拡張して前述の新しいニーズに対応するには、困難な変革のプロセスが伴います。現在使用されている開発テクノロジは従来型のパターンに深く根差しており、最新のソフトウェアで必要とされる新たなパターンとの統合は簡単ではありません。既存のクライアント/サーバー方式から新たなデバイス/クラウド方式への明確な変換方法は、既存のツールから提供されません。

開発者のこうした課題を解消するのが、マイクロソフト プラットフォームです。マイクロソフト プラットフォームは**既存アプリケーション**に基づいて構築されており、次世代型のアプリケーション パターンへの拡張を可能にします。また、**複数の開発テクノロジがサポート**されるため、開発者のスキルや、既存アプリケーションで使用されているテクノロジに応じ、最適な方法を選ぶことができます。サービスの開発に関しては、Windows Azure は Java、Node.js、PHP、Python、Ruby といった開発者の多くが使用できるテクノロジを豊富にサポートしており、特に .NET について抜群のサポート性を備えています。マイクロソフトのプラットフォームでは、クライアント開発に関しても、.NET、HTML5/JavaScript、C++ といった広範なテクノロジが標準でサポートされます。

このガイドでは、.NET 開発に焦点を当て、特にビジネス アプリケーションを中心に説明を進めます。具体的には、既存アプリケーションを形成している従来型パターンでの .NET 開発の使用方法に加え、将来の**モダン ビジネス アプリケーション**を実現する新たなパターンに対応する方法を解説します。

.NET Framework と開発の将来

Microsoft .NET Framework は、マイクロソフト プラットフォーム上で魅力的なアプリケーションを開発できるようにするために構築され、総じて市場では優れた成功を収めています。.NET は現在、あらゆる規模とセグメントの企業において、何百万という開発者によってアプリケーションの作成に利用されています。.NET は、コンシューマー アプリケーション、小規模ビジネス アプリケーション、およびミッション クリティカルな大規模アプリケーションの構築に必要となる、いずれも卓越した品質、パフォーマンス、生産性を誇る各種のコア サービスを提供します。

さらに、.NET は先ほど述べた、台頭しつつある新たなパターンも想定して構築されています。かつて Forum 2000 において、Bill Gates は「Web サイトが個々に独立している状態から、交換可能なコンポーネントから成るインターネットへと移行し、さまざまなデバイスとサービスを組み合わせることで、一貫性のあるユーザー主導のエクスペリエンスを実現する」ことが .NET の目標だと説明しました。こうした .NET の初期のビジョンは、多様なデバイスとサービスを通じたエクスペリエンスが、ソフトウェア開発に対する業界全体の意識を変化させつつある点など、今日開発者を取り巻いている環境と驚くほど一致しています。

各種サービスを活用したマルチデバイス エクスペリエンスの実現は、.NET の特性として、構想当初から重視されていました。その後、.NET は進化を続け、今ではアプリケーションの新しいニーズに対し、業界屈指の開発エクスペリエンスを提供しています。

- **サーバー側では、**.NET はオンプレミス環境向けサービスとクラウド環境向けサービスの両方に対し、共通の開発プラットフォームを提供します。Windows Server および Windows Azure との緊密な統合機能により、各プラットフォームの最も優れた点を生かしながら、アプリケーションを徐々にクラウドへと拡張すると共に、2 つの環境にまたがるハイブリッド アプリケーションを実現できます。また、.NET Framework ライブラリは短い間隔で提供されるため、最新技術を継続的に取り入れ、サービスの軽量化、リアルタイムの通信、モバイル Web アプリケーション、認証といった領域における、クラウド ベースのアプリケーションの新しいニーズにも応えられます。
- **クライアント側では、**.NET はマイクロソフト デバイスはもちろん、すべてのデスクトップ エクスペリエンス、Windows Phone アプリケーション、Windows ストア アプリにわたって、一貫性ある卓越した開発エクスペリエンスを提供します。.NET を使用すると、デスクトップ上で継続的に基盤アプリケーションを開発し、新しい画期的なエクスペリエンスを追加すると同時に、常に開発者の既存のスキルを生かしたり、デバイス間でコードを再利用したりすることが可能です。マイクロソフト デバイス以外も対象となるシナリオでは通常、HTML5 ブラウザー ベースのソリューションを使用します。.NET を Visual Studio と併用すると、多種多様なデバイス上で動作する標準ベースの Web アプリケーションを作成するためのモダン ソリューションとして使用できます。特定のデバイスにさらに特化した、ネイティブ エクスペリエンスの実現を求める開発者向けには、Visual Studio Industry Partner (VSIP) プログラムを通じて、C# のスキルやコードを Windows 以外のデバイスで再利用するためのソリューションが提供されています。

このガイドでは、今後アプリケーションを発展させるにあたり、現在のスキルとアプリケーション要件に応じた適切な決定を下せるよう、これまで紹介したすべての .NET 開発オプションを取り上げます。以降の内容は、次の 2 つのアプリケーション パターンに基づいて構成されています。

- 「次世代型のアプリケーション パターン」では、さまざまなデバイスとサービス間で実行される新たなアプリケーションを形作る、次世代型のパターンを使用したアプリケーションの構築方法について説明します。
- 「従来型のアプリケーション パターン」では、基盤ビジネス アプリケーションを作成する際に利用可能なテクノロジを紹介し、こうしたテクノロジの刷新方法を提案します。

次世代型のアプリケーション パターン

既に述べたように、現在、次世代型のアプリケーション パターンによって、将来のアプリケーション像が形作られつつあります。今日、企業のお客様や社員の皆様は、より個人のニーズに合ったエクスペリエンスを実現するアプリケーションを求めており、また同時に、必要なサービスに常に接続していたいとも考えています。

このセクションでは、こうした新種のアプリケーションの開発にあたって対処が必要となる、次の 2 つの重要なテーマについて考察していきます。

- 種類の異なる複数のデバイスにわたる、共通のエクスペリエンスを開発する
- クラウドを通じて拡張可能な、標準的かつ軽量なサービスを開発する

デバイス

次世代型のアプリケーション パターンの重要な特色として、各デバイスに特化した新しいエクスペリエンスを提供できる点が挙げられます。最適なテクノロジを選択して各デバイス専用のアプリケーションを作成する作業は難しく、次のような多くの要素が関係します。

- 担当者の既存スキルおよび得意とするテクノロジ
- ローカル ハードウェアの機能と連携可能な、デバイス専用のエクスペリエンスを作成する能力
- アプリケーションがターゲットとするデバイスの多様性
- 既存アプリケーションによって使用され、デバイスへの拡張または移行が必要となるテクノロジ

業界では 2 種類の対応方法が広く確立されていますが、互いにまったく異なるアプローチに基づいています。

- **ネイティブ アプリケーション:** それぞれのデバイスの特長を最大限に引き出せるが、プラットフォームごとに固有のスキルとコードが必要
- **Web アプリケーション:** 共通のスキル セットとコードで作成できるが、各デバイスに特化したエクスペリエンスは実現不可

マイクロソフト プラットフォームは、両方のアプローチを完全にサポートするだけでなく、それぞれのデメリットを軽減します。まず、Windows デバイスでは固有のネイティブ開発モデルが要求されず、開発者のスキルと既存アプリケーションに応じた、最も合理的なテクノロジを使用できます。.NET、HTML/JavaScript、および C++ とデバイスとの高度な連携が実現されるため、エクスペリエンスについて妥協することなく、ニーズに最適な意思決定が可能です。さらに、.NET と Visual Studio を併用すると、任意のデバイス上で動作する Web アプリケーションをたいへん容易に作成できるようになります。ASP.NET は、近年使用されている標準をすべてサポートしているため、Visual Studio が備える最新かつ独自の革新

的技術を併用することで、最新のブラウザーをフル活用し、さまざまなデバイスで動作する新しいタイプの Web アプリケーションを構築できます。

ネイティブ アプリケーション

ネイティブ アプリケーションとは、クライアント デバイス上で実行され、そのデバイス固有の機能をフル活用することで、たいへん優れたカスタマー エクスペリエンスを実現するアプリケーションです。既に述べたように、Windows プラットフォームでは、このコンセプトを C++ 以外のテクノロジにまで拡張でき、新しいフォーム ファクターでも既存のコードやスキルを再利用できる可能性が大幅に向上します。

次に、各種テクノロジの相違点と、それぞれの適材適所を示します。

- **.NET/XAML:** 既に .NET と XAML の使用経験が豊富な場合、または既存の .NET/XAML アプリケーションを拡張する場合に使用。移植可能なクラス ライブラリを使用して、Windows ストア アプリ、Windows Phone アプリケーション、Windows デスクトップ アプリケーション、およびその他のマイクロソフト プラットフォームの間で、.NET Framework のコードおよびライブラリを共有することも可能
- **JavaScript/HTML5:** 既に HTML テクノロジと JavaScript テクノロジの使用経験が豊富な場合、または既存の Web アプリケーションのために、Windows ストア仕様のエクスペリエンスを作成する場合に使用。既存のブラウザー ベース Web アプリケーションに使用されている、JavaScript または HTML/CSS のカスタム アセットを再利用するほか、新しい WinJS ライブラリを使用して、Windows ストア アプリ/API のネイティブ機能にアクセス可能
- **C++:** 既に C++ の使用経験が豊富な場合に使用。グラフィックスを多用したアプリケーションやゲームを最適化し、最善のパフォーマンスを実現。Windows、Windows Phone、およびその他のプラットフォームの間で C++ コードを共有するほか、Direct3D を使用して低レベルのグラフィックスにアクセスすることも可能

Web アプリケーション

マイクロソフトは、任意のデバイス上で動作するブラウザー ベースの HTML5 アプリケーションを作成するための、業界屈指のツールおよびテクノロジを提供しています。HTML5 は Windows でもネイティブ テクノロジとしてサポートされるため、複数のデバイスをターゲットとする HTML5 Web アプリケーションを開発すれば、そのコードをネイティブ Windows ストア アプリ用に最適化して再利用できます。次に、利用可能なマイクロソフトの Web テクノロジ一覧を示します。

- **HTML5 をサポートする ASP.NET MVC:** 柔軟性に優れたモダン モバイル Web アプリケーション、ブラウザー間での互換性、あらゆる最新デバイス上で動作するモバイル Web クライアントを実現し、ASP.NET MVC の各種モバイル機能 (現在のブラウザーのユーザー エージェントに基づく異なるページの使用やレンダリングなど) の活用が可能
- **ASP.NET Single Page Application (SPA):** 対話式操作の多い Web アプリケーションやスムーズな UI の更新 (サーバー ページの再読み込みおよび可視性の切り替えに伴うラウンド トリップを最小限にする場合) のほか、クライアント側で HTML5、CSS3、および JavaScript による重要な対話式操作を行う場合にも使用。SPA はフレームワークではなく、ASP.NET MVC や JavaScript ライブラリ (Knockout など) を使用して実装可能な設計パターン
- **LightSwitch プロジェクトの HTML5 クライアント:** 主にデータ駆動型 (CRUD) Web アプリケーションまたはモジュールに使用。すべての最新デバイス上で動作し、自動 HTML レンダリングの活

用や、さまざまなフォーム ファクターへの適応が可能な、データを中心としたクロスブラウザーのモバイル Web アプリケーションを作成する最も容易な方法

- **ASP.NET Web ページ:** ASP.NET Web ページおよび Razor 構文は、サーバー コードと HTML を組み合わせて動的 Web コンテンツを作成するための、アプローチが容易で高速かつ軽量な手段を提供

サービス

複数のデバイスをターゲットとする開発プロセスは、バックエンドの構築から始まります。アプリケーションでは、あらゆるデバイス上で使用でき、インターネットへと拡張可能なサービスを公開する必要があります。次に、サービス構築時に選択可能な、さまざまなアプローチに対応するテクノロジを紹介します。

- **ASP.NET Web API: REST** アプローチ、**OData**、**JSON** などの要件を満たし、サービス開発を柔軟に行えるため、最も推奨されるテクノロジ。使用するテクノロジを評価する際は、まず Web API を試し、Web API がニーズに合わない場合は、他のいずれかのテクノロジを使用
- **Windows Communication Foundation (WCF):** SOAP との相互運用性が必要な場合、または HTTP 以外で転送を行う場合に使用。任意のプロトコル (HTTP、TCP、名前付きパイプなど)、データ形式 (SOAP、バイナリ、JSON など)、およびホスティング プロセスを使用可能。RPC スタイル (タスク/コマンド指向) のサービスと、企業内アプリケーション間通信に最適なテクノロジ
- **WCF Data Services:** 小規模/中規模のデータ駆動型アプリケーションなど、データ/リソース指向性の高いサービスに使用。**OData** のみをサポート。使用方法は容易だが、柔軟性と制御性は ASP.NET Web API に劣る
- **ワークフロー サービス:** 内部的なサービス ロジックが Windows Workflow Foundation (WF) の場合に使用。外観は WCF サービスそのもの
- **ASP.NET SignalR:** クライアント側でのリアルタイム機能を提供。サーバー側コードから、接続されたクライアントへとリアルタイムでコンテンツをプッシュし、数百万にのぼる大量のユーザーに配信可能

モダン ビジネス アプリケーションは、数多くのインターネット ユーザー (エンド カスタマーやパートナーなど) に対応することが当たり前のこととなっており、各種のバックエンド サービスを企業の社内データセンター内で維持することは、必ずしも合理的とは言えません。その点で、クラウド内でのサービスの展開は理にかなっています。また、弾力性や、実稼働環境を迅速にコスト効率よく準備できる点など、クラウドの核となる特長をサービスに活用できるというメリットもあります。

ビジネス アプリケーションがさまざまな環境 (クラウドとオンプレミスの両方) に展開される多様なエコシステムでは、多くの新しいニーズを満たすことが求められます。さらに、現在オンプレミス環境で運用している基盤アプリケーションを拡張するため、自社専用のデータセンターとクラウドをリンクさせる必要がありますが、双方の環境をセキュアに連携させなくてはなりません。

マイクロソフトのクラウド プラットフォームを使用すると、クラウドとオンプレミスの両方のインフラストラクチャをサポートする対称型のアーキテクチャ、テクノロジ、および製品だけでなく、ハイブリッド環境内で 2 種類のインフラストラクチャにまたがるアプリケーションを管理する共通のサービスが提供されます。さらに、アプリケーションの移行や構築を、簡単かつ段階的に実施することも可能です。Windows Azure は、ニーズに応じて選択可能な複数の実行モデルをサポートしています。

- **インフラストラクチャ サービス (サービスとしてのインフラストラクチャ/IaaS):** 従来型の柔軟性の高いアプローチが必要な場合に使用。内部仮想マシン インフラストラクチャの運用、ソフトウェアの保守、および (オペレーティング システム、サービスなどの) 管理が必要で、SQL Server のフル インストールなど、ソフトウェアの従来型のインストールをサポート
- **Web サイト (Web ホスティング):** 管理された IIS Web サイト環境で Web アプリケーションを簡単に展開し、インフラストラクチャの管理業務を必要としないため、短期間での利用開始が可能。低コストで初期段階の拡張性に優れた、用途の広いプラットフォーム
- **クラウド サービス**または**サービスとしてのプラットフォーム (PaaS):** Web サイトと同様の考え方に基づくが、より高い拡張性と柔軟性を備えたプラットフォーム。高度なサービス品質要件が求められる場合や、制御性を高めるために使用。PaaS としての信頼性および管理に必要となるほとんどの作業にも対応
- **Windows Azure モバイル サービス:** Windows ストア、Windows Phone、Apple iOS、Android、HTML/JavaScript 対応の各アプリケーションに、構造化ストレージ、ユーザー認証機能、プッシュ通知機能、およびバックエンドのジョブとサービスを提供する拡張性の高いクラウド バックエンドを実現

アプリケーションをクラウドに拡張する際は、Windows Azure が提供する各種の追加サービスを利用することもできます。

- **データ管理**
  - Windows Azure SQL データベース: オンプレミスの SQL Server に匹敵する、機能豊富なリレーショナル データベースを提供し、オンプレミスの SQL Server データベースからクラウドへの移行、または両環境の同期を簡単に実行可能
  - Windows Azure テーブル: NoSQL アプローチによるシンプルな非構造化データをベースとし、データ ソースにきわめて高い拡張性が求められる場合に最適
  - Windows Azure BLOB ストレージ: ビデオやファイル、バックアップ データまたはその他のバイナリ情報のような、非構造化バイナリ データを格納するためのストレージ
- **ネットワーク**
  - Windows Azure 仮想ネットワーク: オンプレミスのローカル ネットワークを、一連の定義済みの Windows Azure 仮想マシン (VM) に接続する、VPN に類似したサーバーおよびサブネットワーク指向のアプローチ
  - Windows Azure トラフィック マネージャー: Windows Azure アプリケーションを複数のデータセンターで実行している場合に、ユーザーからの要求を、複数のアプリケーション インスタンスにインテリジェントにルーティング
- **ビジネス分析**
  - SQL レポート: レポート生成のプラットフォームを必要とする場合に、SQL Reporting Services と同様の機能を提供
  - HDInsight (Hadoop on Windows Azure): Windows Azure 内で、ビッグ データを PaaS としてホスト可能
- **メッセージング**
  - キュー: 同一の永続的キューにアクセスすることで、異なるアプリケーションまたはプロセス間の非同期通信を実現

  - サービス バス:永続的なメッセージング機能を提供するが、イベント駆動型アプリケーションと統合、および Publish/Subscribe パターンのほか、サービス バスのリレー機能を通じてクラウドからオンプレミスのデータセンターに簡単にアクセスする場合 (ファイアウォールを間に挟んだピアツーピア アプリケーションでも利用可能) など、より高度なシナリオに適する
- **キャッシュ**
  - Windows Azure キャッシュは、インメモリの分散キャッシュ。外部サービスとして使用するほか、自社の仮想マシン間で展開したキャッシュを共有することが可能
  - Windows Azure CDN は、HTTP 経由でアクセスするデータ (ビデオ、ファイルなど) をキャッシュできる "インターネット キャッシュ"
- **ID**
  - Windows Azure Active Directory (AD): オンプレミスの ID 管理サービスとの統合により、クラウド アプリケーション間のシングル サインオンを実現
  - Windows Azure AD Access Control: Web アプリケーションおよびサービスにアクセスする必要のあるユーザーを、コードに複雑な認証ロジックを組み込むことなく、簡単に認証できる手段を提供
- **ハイパフォーマンス コンピューティング (HPC):** 多数の VM を同時に実行し、複数のタスクを並列処理する機能。Windows Azure では、これらのインスタンスに処理を分散するための HPC スケジューラを提供。これは、業界標準の Message Passing Interface (MPI) を使用するために構築された各種 HPC アプリケーションと連携させることができ、クラウド内で実行する HPC アプリケーションを簡単に構築することを目的とするコンポーネント
- **メディア:** Windows Azure メディア サービスは、メディアの取り込み、エンコード、コンテンツ保護、広告の挿入、ストリーミングなどに対応した一連のクラウド コンポーネントを提供することで、ビデオなどのメディアを使用するアプリケーションの、作成および実行のプロセスを大幅に簡略化

従来型のアプリケーション パターン

モダン ビジネス アプリケーションを構築する際、重要となるのは新しいモバイル アプリケーションの作成だけではありません。各企業のコア アプリケーションが既に備えている価値を最大限に引き出すために、ユーザーから要求される新しいエクスペリエンスをビジネス プロセスと絶妙に統合させる必要があります。

このセクションでは、従来型のパターンで一般的に使用されているテクノロジについて説明すると共に、こうしたアプリケーションを拡張し、モダン ビジネス アプリケーションのコンセプトに対応するための推奨アプローチを紹介します。ここでは、企業内で見られるアプリケーションを大きく 2 つに分け、小規模/中規模アプリケーションと、ミッション クリティカルな大規模アプリケーションとして扱います。

小規模/中規模ビジネス アプリケーション

社員や社内部門から成る管理された環境に対応する場合、ビジネス アプリケーションでは従来型のアプローチに従うのが通例です。このセクションでは、頻繁に変更され、ライフサイクルが比較的短い、小規模/中規模アプリケーションについて説明します。こうしたアプリケーションは主に、データ中心またはデ

ータ駆動型のいわゆる CRUD (Create: 生成、Read: 読み取り、Update: 更新、Delete: 削除) シナリオに利用されるという特徴もあります。また、このタイプのアプリケーションは、コラボレーションとドキュメント管理に対する要件を持つものが多く、Microsoft SharePoint のようなコラボレーション製品、Microsoft Dynamics CRM や Microsoft Dynamics ERP といったビジネス マーケティング製品、SAP をベースとしたカスタム フロントエンド アプリケーションと統合させるというニーズもあります。しかし、このカテゴリは、きわめて限定されたカスタム シナリオを対象とする部門別のスタンドアロン アプリケーションを指すことがほとんどです。

データ中心のビジネス Web アプリケーション

データ中心のアプリケーションの大半は、マスターおよび詳細のシナリオを含む、CRUD 操作に対応しています。このタイプのアプリケーションの小規模なサブセットでは、ビジネス ロジックを実装する必要がありますが、対象範囲は広くありません。ビジネス Web アプリケーションは、複雑なドメインと大量のビジネス ルールを扱う大規模アプリケーションとはならないのが一般的で、通常はデータ駆動性のきわめて高い、単純でわかりやすい構造をしています。こうしたアプリケーションで最も重視される優先事項は、生産性、コスト、およびビジネスへの価値です。開発ライフサイクルを短縮し、できる限りコストを抑えながら、いかにビジネスに対して機動的に価値をもたらせるかが重要になります。

このタイプのアプリケーションで考慮すべきテクノロジは次のとおりです。

- **LightSwitch プロジェクトの HTML5 クライアント:** すべての最新デバイス上で動作し、自動 HTML レンダリングの活用や、さまざまなフォーム ファクターへの適応が可能な、データを中心としたクロスブラウザーのモバイル Web クライアントを作成する最も容易な方法
- **HTML5 をサポートする ASP.NET Web フォーム:** Web フォームを使い慣れた開発者が簡単に開発に着手でき、コードへの制御性も完全に維持されるアプローチ
- **ASP.NET Web ページ:** ASP.NET Web ページおよび Razor 構文は、サーバー コードと HTML を組み合わせて動的 Web コンテンツを作成するための、アプローチが容易で高速かつ軽量な手段を提供
- **HTML5 をサポートする ASP.NET MVC:** HTML のレンダリングを完全に制御でき、優れた単体テスト機能とフェイク機能を備えた、最も柔軟かつ強力なオプション

データ中心のビジネス デスクトップ アプリケーション

アプリケーションで大量のデータ入力を行う場合や、ローカル ストレージ、COM との相互運用性、および自動処理が関係する複雑なオフライン シナリオに対応する場合などは、Web アプリケーションではなくデスクトップ アプリケーションを構築する場合が多いでしょう。また、エンド ユーザーが業務上のニーズとスキルセットに基づいて、デスクトップ アプリケーションを好むケースもあるかもしれません。

次に、データ中心のデスクトップ アプリケーションに関する主要テクノロジを示します。

- **.NET Windows Presentation Foundation (WPF):** 複雑な UI や、スタイルのカスタマイズを必要とし、グラフィックスを多用するようなデスクトップ シナリオで推奨される、Windows ベース デスクトップ アプリケーション向けのテクノロジ。WPF では XAML のビューも活用。WPF の開発スキルは Windows ストア アプリの開発スキルと似通っているため、Windows ストア アプリへの移行は、Windows フォームからよりも WPF から行う方が容易
- **.NET Windows フォーム:** デスクトップ アプリケーションの構築のため、.NET Framework で初めて提供された UI テクノロジであり、現在も多くのビジネス デスクトップ アプリケーションに活

用可能。使いやすく、WPF よりも軽量なため、UI スタイルのカスタマイズを必要としないシンプルなシナリオに適する

- **LightSwitch プロジェクトのデスクトップ クライアント:** LightSwitch 中間層および HTML5 によるソリューションを既に採用している場合、LightSwitch が標準でサポートするデスクトップ エクスペリエンスを使用可能

Microsoft Office および Microsoft SharePoint 用のアプリケーション

Office および SharePoint を軽量なアプリケーションにより拡張できるよう、Office 2013 にはクラウド アプリケーション モデルが導入されています。これにより、開発したビジネス アプリケーションの機能を、利用者が既に使用している Office 生産性アプリケーションを通じて提供できるようになりました。このクラウド アプリケーション モデルは、HTML、CSS、JavaScript、REST、OData、OAuth といったクライアント上で使用される標準の Web テクノロジと、ASP.NET などの、サーバー上で使用される任意のサーバー テクノロジに基づいて構築します。そのため、Web 開発者は、既存のスキルを使用してアプリケーションを構築し、使い慣れたツールや言語、ホスティング サービスを活用することができます。アプリケーションを開発した後は、クラウド内ですばやく展開、更新、および保守を行うことができ、Office ストアで公開して販売したり、内部のアプリケーション カタログを使用して IT 部門の承認済みアプリケーションを自社内に配布したりすることも可能です。

こうした統合アプリケーション モデルは、次のタイプのアプリケーションに適用されます。

- **Office 用アプリ**(Office 2013、Office 365、Project Professional 2013、Word 2013、Excel 2013、PowerPoint 2013、Outlook 2013、Outlook Web App、Excel Web App、および Exchange Server 2013 に対応)
- **SharePoint Server 2013** および Office 365 の **SharePoint Online 用アプリ**

Office 用アプリおよび SharePoint 用アプリの開発に使用できる、主要なツールとテクノロジは次のとおりです。

- **“Napa” Office 365 開発ツール** (ブラウザーでコードを作成できる、軽量な IDE): Visual Studio を使用せずに、Office 用/SharePoint 用アプリケーションの開発を最も簡単に開始できる方法
- **Office Developer Tools for Visual Studio** (フル機能版): 完全な機能と柔軟性が求められる Office 用および SharePoint 用のアプリケーションを開発する場合に使用。Visual Studio Professional、Premium、および Ultimate の購入者に提供されるツール
- **Visual Studio LightSwitch:** オンプレミス環境およびクラウド環境で実行される SharePoint 用アプリをサポートするツール。SharePoint のリソース (リストおよびサービス) または SharePoint 以外のサービスとデータ ソースを利用する、データ駆動型 (CRUD) アプリケーションを作成する場合に使用。SharePoint と統合される、柔軟なフォームおよびデータ中心アプリケーションを作成する最も簡単な手段

ビジネス向けデスクトップ アプリケーションの刷新

デスクトップ上でのエクスペリエンスを構築する際は、利用者が寄せている新たな期待にも配慮するべきです。こうした期待に応えられるよう、.NET ではデスクトップ アプリケーションに関する複数の最新テクノロジを取り入れると共に、アーキテクチャを変更することなく新しいプラットフォームへとアプリケーションを拡張し、コードの再利用もできる機能を提供しています。次の推奨事項に留意してデスクトップ アプリケーションを構築すれば、アプリケーションの寿命が延び、新しいデバイスへの拡張も容易になるだけでなく、将来的にアプリケーション全体を移行することも可能になります。

- **Model-View-ViewModel (MVVM) デザイン パターンを使用する:** マイクロソフトのクライアント プラットフォーム (WPF を含む) では、MVVM パターンによるアプリケーションの構築を簡単に行えます。このパターンを使用すると、状態および動作と画面表示とが明確に分離されるため、クリーンで保守容易性に優れたコードを作成し、複数のデバイス間で簡単に共有できるようになります。
- **クライアント ロジックに、ポータブル クラス ライブラリを使用する:** .NET ポータブル クラス ライブラリは、デスクトップ、Windows ストア アプリ、Windows Phone アプリケーションなど、複数のプラットフォーム間でバイナリを共有する機能を提供します。クライアント ロジックの実装に .NET の移植可能なライブラリを使用すると、複数のプラットフォーム上で複数のエクスペリエンスを作成する場合でも、作業が格段に容易になります。
- **ユーザー エクスペリエンスを刷新する:** 今日のエンド ユーザーが求める各種のコンセプトは、.NET に取り入れられたデスクトップの最新テクノロジによって実装することができます。"軽快かつ柔軟"、"真のデジタル化を目指す"、および "より少ない要素でより大きな効果を上げる" といった設計原則を既存のデスクトップ アプリケーションに適用するには、XAML 設計にモダン UI を採用し、アニメーションを慎重に使用すると共に、.NET の非同期プログラミングを広範にわたって実装します。
- **ビジネス ロジックをサーバーに移行する:** 2 階層アプリケーション (クライアント/サーバー方式) は、新しいデバイス向けに拡張しようとするときわめて手間がかかります。この対応策として、ビジネス ロジックをサービスごとに明確に切り離し、後から他のデバイスおよびフォーム ファクターで再利用できるようにしておくことをお勧めします。
- **クラウドへの拡張を図る:** クライアントから分離したビジネス ロジックは、Windows Azure が提供する複数のソリューションを利用して、クラウドへと移行することができます。ロジックをクラウド サービスに変換することにより、既存ソリューションの順応性と拡張性が著しく向上し、マルチ デバイス アプローチへの対応が可能になります。

Visual Studio パートナーからも、既存の .NET アプリケーションを刷新するのに役立つ、各種の広範なツールやテクノロジが提供されています。

- **Citrix** が提供する Mobile SDK for Windows Apps を使用すると、リッチな開発ツール キットを活用して Windows 向け LOB アプリケーションをモバイル化できるほか、セントラル サーバー (Citrix XenApp/XenDesktop) 上で実行され、Citrix Receiver によって任意のモバイル デバイスからアクセスできる、タッチ操作対応の新しいアプリケーションを作成可能
- **Xamarin** は、Windows または Windows Phone をターゲットとしたアプリケーションと iOS/Android デバイスとの間で、C# コードを共有する手段を提供。基盤となる特定の API にアクセスしてデバイス専用のビューを作成するほか、デバイス間でクライアント ロジックを再利用することが可能
- **ITR Mobility** は、C# でエンタープライズ モバイル アプリケーションを構築して主要モバイル プラットフォーム上で提供するためのソリューションを提供。抽象 UI やエンタープライズ データ同期といったサービスを提供し、広範なデバイスに対応するビジネス アプリケーションを実現
- **Mobilize.Net** は、既存のソース コードを新しいコードに変換することで、出力先アプリケーションのランタイムを使用することなく、Web、モバイル、クラウドといったモダン プラットフォームにレガシ アプリケーションを移行するためのソリューションとサービスを提供

Visual Studio パートナーが提供するソリューションの詳細については、VSIP partner directory (英語) を参照してください。

RIA コンテナー ベースのアプリケーションの刷新

わずか数年前、リッチ インターネット アプリケーション (RIA) のコンテナーおよびプラグインが流行していたころ、IT を取り巻く状況は今とはまったく違っていました。RIA によってほとんどの開発ニーズがまかなわれていた 5 年前、そのターゲットは Windows ベースの PC と Mac コンピューターのみでした。2010 年の "デバイス革命" 以降、さまざまなオペレーティング システム (Windows 8、Windows Phone 8、iOS、Android、Chrome OS など) を使用するさまざまなデバイス (タブレット、スマートフォン、およびコンピューター) が登場しましたが、その多くは RIA プラグインをサポートしていません。

同時に、以前ならプラグインを必要としていたような、よりリッチなシナリオが、HTML5 によってサポートされるようになりました。HTML5 は現在、あらゆるデバイスに広くサポートされており、従来型のプラグインに代わって、クロスプラットフォームのクライアントを開発する優れた手段となっています。

各デバイス固有の機能をフル活用して最も魅力的なカスタマー エクスペリエンスを実現できるため、市場ではネイティブ アプリケーションの提供も増加傾向にあります。

マイクロソフト プラットフォームは、ユーザー インターフェイスの 3 種類のアプローチ (Web、ネイティブ、および RIA) をすべてサポートしていますが、複数のデバイスに対応するモダン エクスペリエンスの開発には、主に Web テクノロジとネイティブ テクノロジが使用されることも考慮されています。どの段階でこうした移行に踏み切るかは、最終的にはそれぞれの要件とニーズに基づいて決定しますが、マイクロソフトは、お客様の選択された移行アプローチをプロセス全体にわたってご支援します。

- **ネイティブ アプリケーションに移行**する場合は、任意の Windows デバイス上で XAML/.NET をネイティブのターゲットとすることにより、既存のスキルはもちろん、コードも再利用できます。ポータブル クラス ライブラリを使用して、Silverlight を含めた複数のプラットフォーム間でバイナリを共有することも可能です。
- **ブラウザー ベースの HTML5 アプリケーション**については、最新の標準に基づいてあらゆるデバイスに対応したアプリケーションを作成できるよう、優れたツールとフレームワークを提供しています。Silverlight と HTML の相互運用性を活用すれば、ハイブリッド アプリケーションによる段階的な移行も可能です。
- 依然として **Silverlight によってのみ**サポートされるシナリオ (ビデオ コンテンツの保護など) をターゲットとする場合や、今のところアプリケーションに次世代型のパターンを適用する必要のない場合は、Silverlight を引き続き使用できます。Silverlight は、安定性に優れる成熟したテクノロジです。既存の投資を最大限に活用し、HTML5 またはネイティブ ソリューションに段階的に移行できるよう、最新バージョン (Silverlight 5) には 10 年間の延長サポートが用意されています。

『ビジネス アプリケーション向け .NET テクノロジ ガイド』の完全版では、既存のアプリケーションで一般によく見られるシナリオと、それぞれに対応する今後の推奨パスについてさらに詳しく説明しています。

ミッション クリティカルな大規模ビジネス アプリケーション

ミッション クリティカルなビジネス アプリケーションは、主要事業を適切に運営するうえで、欠くことのできない任意のビジネス アプリケーションだと定義できます。こうした性質を持つアプリケーションに、たとえ短時間でも障害が発生すれば、事業の失敗につながりかねません。

ミッション クリティカルなコア ビジネス アプリケーションには、新たな目標と固有の優先事項があります。絶えず進化する大規模アプリケーションには、長期的な持続可能性と保守容易性が不可欠です。

ミッション クリティカルな大規模アプリケーションの開発では、特に 2 つの点に注意する必要があります。

- **コア ドメインの複雑性を解消する**
  - ドメインに関する複雑な専門知識およびビジネス ルールのほか、これらをソフトウェアに効果的に反映するための方法が必要です
  - この点に配慮しながら、コンテキスト (ドメイン駆動設計アーキテクチャ アプローチ、依存関係挿入の手法およびコンテナーに基づく疎結合アーキテクチャなど) に応じたアーキテクチャ、設計、実装方法 (モノリシック アプローチにするか、構造化/複数層アプローチにするかなど) を決定し、最適なパターンとベスト プラクティスを判断する必要があります。また、アプリケーション設計では、アプリケーションの将来的な発展および保守しやすさを考慮する必要があります。
- **十分な QoS (サービス品質) を確保する**
  - 可用性、拡張性、セキュリティなどの問題と関連があります。
  - この点に配慮しながら、分野横断的な対策 (セキュリティ、運用、インストルメンテーションなど) の設計方法および実装方法を検討する必要があります。
  - また、インフラストラクチャ アーキテクチャも QoS と緊密にリンクさせます。たとえば、必要となる拡張性と順応性、ユーザー タイプ、実稼働までの機動性のニーズに応じて、オンプレミス環境またはクラウド環境のインフラストラクチャ アーキテクチャを検討する必要があります。

ミッション クリティカルな大規模アプリケーション向けマイクロソフト開発テクノロジ

ミッション クリティカルな大規模アプリケーション向けにマイクロソフトが提供している開発テクノロジは、中間層テクノロジ、インフラストラクチャ テクノロジ、および UI テクノロジの 3 つに分けられます。

- **中間層テクノロジ**
  - **サービス:** 分散型エンタープライズ アプリケーションには継続的なサービスが不可欠であり、パフォーマンスや、軽量で相互運用可能な HTTP サービス、各種の標準 (REST、OData、SOAP、WS-*)、およびエンタープライズ サービスの各要件のサポートが優先事項となる。マイクロソフトでは、ASP.NET Web API、WCF、および ASP.NET SignalR に基づく広範なテクノロジを提供
  - **ドメイン モデル (エンティティ、集計、ドメイン モデル ロジック):** ドメイン モデルはアプリケーションの中核であり、大規模なドメインの複雑性解消、長期的な保守に対応した適切な設計の作成のほか、ドメイン コード (POCO エンティティ) とインフラストラクチャ テクノロジの分離が優先事項となる。マイクロソフトでは、ドメイン駆動設計パターンを適用する際に使用可能な、成熟したドメイン モデルおよびデータ アクセスのテクノロジ (Entity Framework、LINQ、ADO.NET など) を提供
  - **複合、統合、ビジネス プロセス、ワークフロー:** すべてのエンタープライズ アプリケーションには、たとえば、依存関係の挿入および IoC コンテナーに基づいた疎結合アーキテクチャ設計、時間のかかるプロセスおよびワークフロー、イベント駆動型サブシステムの統合、最新のクレームベース セキュリティの分離、トランザクションの実装などといった、大規模アプリケーションの典型的なシナリオに対処するバックボーンが必要。マイクロソ

フトはこうした領域に対応する、制御の反転 (IoC) コンテナー (Unity Application Block および MEF)、Windows Workflow Foundation (WF)、Windows Identity Foundation (WIF)、NET Framework System.Transactions API といった各種の実績あるテクノロジを提供

- **インフラストラクチャ テクノロジ**
  - **メッセージング、セキュリティ、およびキャッシュ:** API による複合および統合のアプローチ (前述) では、拡張に備えた強固な基盤インフラストラクチャが求められ、非同期メッセージング インフラストラクチャ、メッセージ キュー インフラストラクチャのほか、分散キャッシュおよびセキュリティ/ID インフラストラクチャに関連したアセットが必要。マイクロソフトは、Active Directory、AD FS、Windows Azure Active Directory、サービス バス (Windows Azure および Windows Server 向け)、Windows Azure キュー、メッセージ キュー (MSMQ)、Windows Azure キャッシュ、Biztalk Server/Services といった、成熟した強固なインフラストラクチャ テクノロジを提供
  - **データ ソース:** ほぼすべての種類のビジネス アプリケーションは、ビジネス データを保持するためのデータ ソースを必要とするが、アプリケーションのコンテキストおよび優先事項に応じ、データ ソースを使い分けるのが適切。優先事項は、リレーショナル データの豊富さとトランザクション機能、データの可用性と拡張性、および膨大な非構造化データに対するアプローチが中心となる。マイクロソフトはこうした優先事項に対応するため、SQL Server、Windows Azure SQL データベース、Windows Azure NoSQL テーブルおよび BLOB のほか、HDInsight (ビッグ データ、Hadoop) といった各種のテクノロジを提供
  - **サービス展開インフラストラクチャ:** 最後に、すべてのエンタープライズ アプリケーションは、将来的にコンテキストがどう変化するかにかかわらず、アプリケーションの可用性が確保されるような信頼性の高いインフラストラクチャを必要とする。優先事項として、オンプレミスまたはクラウドの選択に対する一貫性ある代替手段、信頼性、パフォーマンス、可用性、拡張性、順応性、ハイブリッド IT、運用、および監視などが挙げられる。マイクロソフトはこうした要件に対し、Windows Server、Windows Azure、および Microsoft System Center が提供するインフラストラクチャを使用した、業界屈指のソリューションを提供

- **UI テクノロジ**
  - **Web テクノロジ (きめ細かい機能と完全な制御性):** ミッション クリティカルなビジネス アプリケーションに関する Web 開発を行う際は、疎結合アーキテクチャをサポートし、最適な Web ユーザー エクスペリエンスを作成できるよう、完全な制御性ときめ細かい機能を備えたテクノロジを提供しなければならない。そのためマイクロソフトでは、ASP.NET MVC (サーバーでの完全な制御性を実現) および ASP.NET SPA (Single Page Application) テンプレートを提供し、Knockout のような JavaScript ライブラリを多用して、JavaScript 内で MVVM パターンをサポートできるようにしている。また、高度なデータ管理に対応した Breeze ライブラリも使用可能
  - **Windows ストア ビジネス アプリケーション:** タッチ指向のモダン アプリケーション (Windows 8 または Windows Phone 向けの Windows ストア アプリ) には革新的かつ魅力的なユーザー エクスペリエンスが求められるが、企業ではさらに、そうしたユーザー エクスペリエンスを担当者の得意分野とスキルに沿って実現する必要がある。マイクロソフトはネイティブの Windows ストア アプリを作成するための広範なテクノロジおよび言語を提供しており、.NET/XAML、WinJS/HTML5、C++/XAML といった、開発チームの各種スキルをサポート

- **デスクトップ アプリケーション テクノロジ:** デスクトップ アプリケーションは、大規模ビジネス アプリケーションの一部として、特に既存システム内の大量のデータ入力を行うシステムに多く使用される。こうしたアプリケーションには、Windows ストア ビジネス アプリケーションへの移行パスを提供する WPF (Windows Presentation Foundation) のほか、UI スタイルのカスタマイズを必要としないシンプルなシナリオの場合には、WPF に比べ使いやすく軽量なソリューションを提供する Windows フォームの使用を推奨

ここまでご紹介してきたように、エンタープライズ アプリケーション開発には、きめ細かい低水準のマイクロソフト テクノロジが数多く活用されています。長期的に使用する大規模ビジネス アプリケーションを構築する際は、適切なアプローチ、設計、アーキテクチャ、およびテクノロジを選択することが、長期にわたる成功を収めるうえで欠かせません。

ミッション クリティカルなエンタープライズ アプリケーションの刷新

ミッション クリティカルな大規模アプリケーションを新たに作成する場合は、拡張性と順応性に優れた最新アプローチの適用が推奨されます。ただし、既存のエンタープライズ アプリケーションのほとんどは非同期エンジンおよびイベント駆動型アプローチに基づいて構築されておらず、スケールアウト アーキテクチャや順応性のあるアーキテクチャをサポートしていません。多くの場合、こうしたアプリケーションをレガシ システムととらえ、最新の (Web ベースまたはサービスベースの) ファサードに統合する方がよいでしょう。この統合は、さまざまなチャネルを集約するメッセージ指向の非同期統合システムを新たに構築する際に、同時に実施することをお勧めします。従来型のアプリケーションを変更および更新する際は、イノベーションの推進 (モダン アプリケーションの場合も同様)、また拡張性およびパフォーマンス関連の領域に重点的に取り組みます。ただし、レガシ システムを扱う場合は、ミッション クリティカルなアプリケーション全体をゼロから再構築するのではなく、クラウド内にフロントエンド サービスとキャッシュ (上記では "ファサード" として言及) を構築することで、より簡単に拡張性を向上できます。そのうえで、予算と時間が許すのであれば、並行してファサードの内部のシステムを再設計することも可能です。

詳細情報

ビジネス アプリケーションには多種多様なタイプがあり、必要となるアプローチやテクノロジは、顧客ニーズ、既存のスキル、および具体的なアプリケーションの要件によって異なります。このガイドでは、.NET 開発者の皆様がターゲットとするアプリケーションを各カテゴリに分類し、それぞれのタイプに適した各種オプションについて、大まかにご説明しました。さらに詳しい情報および詳細なガイダンスについては、『ビジネス アプリケーション向け .NET テクノロジ ガイド』の完全版をダウンロードしてご利用ください。