ASP.NET

Multi Web UI アプリケーション開発の実践—その1
第8回 UIアーキテクチャの差異とユーザビリティを考える

西谷 亮 *NISHIYA, Ryo*
マイクロソフト株式会社
.NETマーケティング部
フィールドテクニカル
エバンジェリスト

はじめに

本連載ではこれまで、Internet Explorerをはじめとするリッチな環境下でのWebアプリケーション開発、そして携帯電話に代表されるような端末によるシンプルなWebアプリケーション開発を紹介してきました。WebフォームやMicrosoft Mobile Internet Toolkit（以下MMIT）を利用することで、サーバー側での自動的なUI（ユーザーインターフェイス）レンダリングやイベントの処理などを実現することができました。

実際のビジネスの局面においては前述の開発のうちどちらかひとつというわけではなく、同時に双方をサポートしソリューションを提供することで、その付加価値を高めてゆくことが可能となるはずです。

しかし、双方のアプリケーションが単独に存在しているかのように開発を進めてゆく、もしくはシステムを展開してゆくことは決してベストではありません。なぜならば、開発生産性の低下を招くばかりではなく、管理コストの増加、Time-to-Marketでのシステム導入の障壁となることなどが考えらるからです。

今回から2回にわたって、Multi Web UIアプリケーションを実際にどのように設計／実装すれば、これらの潜在的な問題点を解消できるのかを検証してゆくことにしましょう。

本稿で前提となるもの

OS	Windows 2000（SP2）以降
開発環境	Visual Studio.NET Internet Informaiton Services 5.0以降 Internet Explorer 6.0 Mobile Internet Toolkit

初級　中級　上級

現状の問題を把握する

問題の解決にあたるためには、今ある問題を認識しなければなりません。現存するシステムを検証しながら、その問題点を洗い出してみましょう。

ここでいうMulti Web UIアプリケーションを「クライアント環境を選ばないWebアプリケーションである」と定義すると、対応しなければならないデバイス群／クライアント環境には次のようなものがあげられます。

- Windows上での各種ブラウザ
- 他のプラットフォーム上での各種ブラウザ
- 携帯電話（各キャリアをすべて包含）
- PDAなどのスモールデバイス

これらは、いずれもインターネット

に接続し、アプリケーション機能を利用できる環境です。システムとしての受け口を広げるには、これらへ柔軟に対応する必要があります。

ここで問題になるのは、クライアントによってレンダリング言語が異なるという点です。HTML3.2/4.0、DHTML、CHTML、WML、XHTMLなどさまざまなものが存在しています。デバイスに応じた言語でコンテンツを記述するということ自身は正しいといえますが、それによって開発に必要とされるスキルセットは増加してしまいます。

実際には、単なるレンダリング言語の違いだけではありません。表示可能な画像形式の違いやフォームに対するアーキテクチャの違いなど、実はさまざまな面でデバイス間、クライアント間での差異が存在しています。

これまでは、これらの差異を開発者が理解していなければ、マルチクライアント化されたWebアプリケーションは実装できませんでした。そこには、開発者の学習にかかる時間的なコスト、実装にかかわる作業の煩雑さなど潜在的な問題点が見え隠れしてきます。真のMulti Web UIを目指すには、言語の乱立やアーキテクチャの違いを吸収するための仕組みが必要となってくると考えられます。

▶ユーザビリティからみた問題点

では、言語という視点からユーザビリティに目を移してみましょう。各キャリアの携帯電話を包括的にサポートできるサイトを紹介するときには、次のようなURLを見かけることがよくあります。

iモード用：
http://www.DomainName.com/i
ezWeb用：
http://www.DomainName.com/ez

これは、デバイスに応じた仮想ディレクトリをそれぞれ用意することによって、適切なUIを提供してゆくという趣旨で行なわれている方法です。しかし、情報を提供したい、システムを利用させたい側の都合でこのようにアドレスが異なることは正しいのでしょうか。できることならば単一のURLだけを紹介し、どのようなクライアント環境からでも適切なUIを提供するほうが、ユーザーの側に立ったソリューションとなるでしょう。

また、いくつも仮想ディレクトリを用意し、その中で各々のコンテンツ管理を行なうということは管理や更新にかかわる作業を煩雑なものとしてしまい、システムにおける迅速性が損なわれる恐れもあるでしょう。これらの問題点は、最終的にシステムの実装へと影響をおよぼし始めます。よって、これらも潜在的な問題のひとつと言えるでしょう。

▶管理面から見た問題点

また、たとえば、仮想ディレクトリを分割していたり、デバイスに応じたUIを各々作成していたとなると、そこで行なわれるべき処理をつかさどるビジネスロジックは、どのように実装されるのでしょうか。多くの場合、インターフェイスから呼び出されるロジックは、フォームなどのアーキテクチャの違いから別々のものが用意され、それらが最終到達点のデータベースやストレージにアクセスするという形式にならざるを得ないのではないでしょうか。

しかし、このようなシステム設計にしてしまうと、完成したときは動いているため問題ないように思われますが、将来的にメインテナンスなどの管理の面を考慮すると大きな問題となってきます。ある一機能を更新しようと考えたときにビジネスロジックが分散してしまっているため、更新箇所が多くなってしまいます。つまり、すべてのデバイスに対しての一括的な更新ではなく、順次作業を行なってゆかなければならなくなります。問題が局所化されていないことによって、本来“1”で済む作業が“2”にも“3”にもなってしまい負担が増えるため、タイムリーなシステム展開を阻害してしまう可能性があるわけです。

ひとくちに「Multi Web UIの実現」や「ユビキタスネットワークの実現」といったところで、いくつかの考慮を行なわないと一筋縄ではゆかないということがおわかりになると思います。

解消策を検討する

しかし、裏を返せばこれらの問題点を解消してゆくことで、より柔軟性の高いMulti Web UIアプリケーションを構築することが可能です。本連載では、先に提示した潜在的な問題点をどのように解消してゆくかを2回にわたって考えてゆくことにします。

まず今回ご紹介するのは、1点目にあ