.NET Framework 1.1/3.5 移行ホワイトペーパー

2008 年 8 月
日本ユニシス株式会社
マイクロソフト株式会社

目次

概要

2002 年の.NET Framework 1.0 のリリースから既に 6 年が経過し、.NET Framework 1.0 や.NET Framework 1.1 上に実装されたシステムの更改や機能追加を検討する際に、.NET Framework 2.0/3.0/3.5 の新機能の利用を検討するお客様が増えるものと考えています。そこで本書では主として、C#/VB.NET で実装されたビジネス・アプリケーションで利用されている.NET Framework 1.1 から.NET Framework 2.0/3.0/3.5 への移行をスムーズに行うために知っておくべき点、考慮すべき点について説明します。

本書の内容

- .NET Framework 移行検討のための前提知識
- 事前調査・移行検討段階で行うこと
- 事前検証段階で行うべきこと
- 移行計画策定
- 移行作業ポイント

.NET Framework 移行検討のための前提知識

この章では、あなたのシステムの.NET Framework の移行検討をするにあたり、事前に知っておくべきことを挙げます。

- .NET Framework のバージョンの関係
- 製品サポートライフサイクルについて
- 移行についての公式情報
- サイドバイサイド実行（SxS 実行）

これらについて、既に十分な知識をお持ちの読者は、次の章の「事前調査・移行検討段階で行うこと」からお読みいただいても構いません。

.NET Framework のバージョンの関係

.NET Framework のバージョンの関係は次の図のように示すことができます。

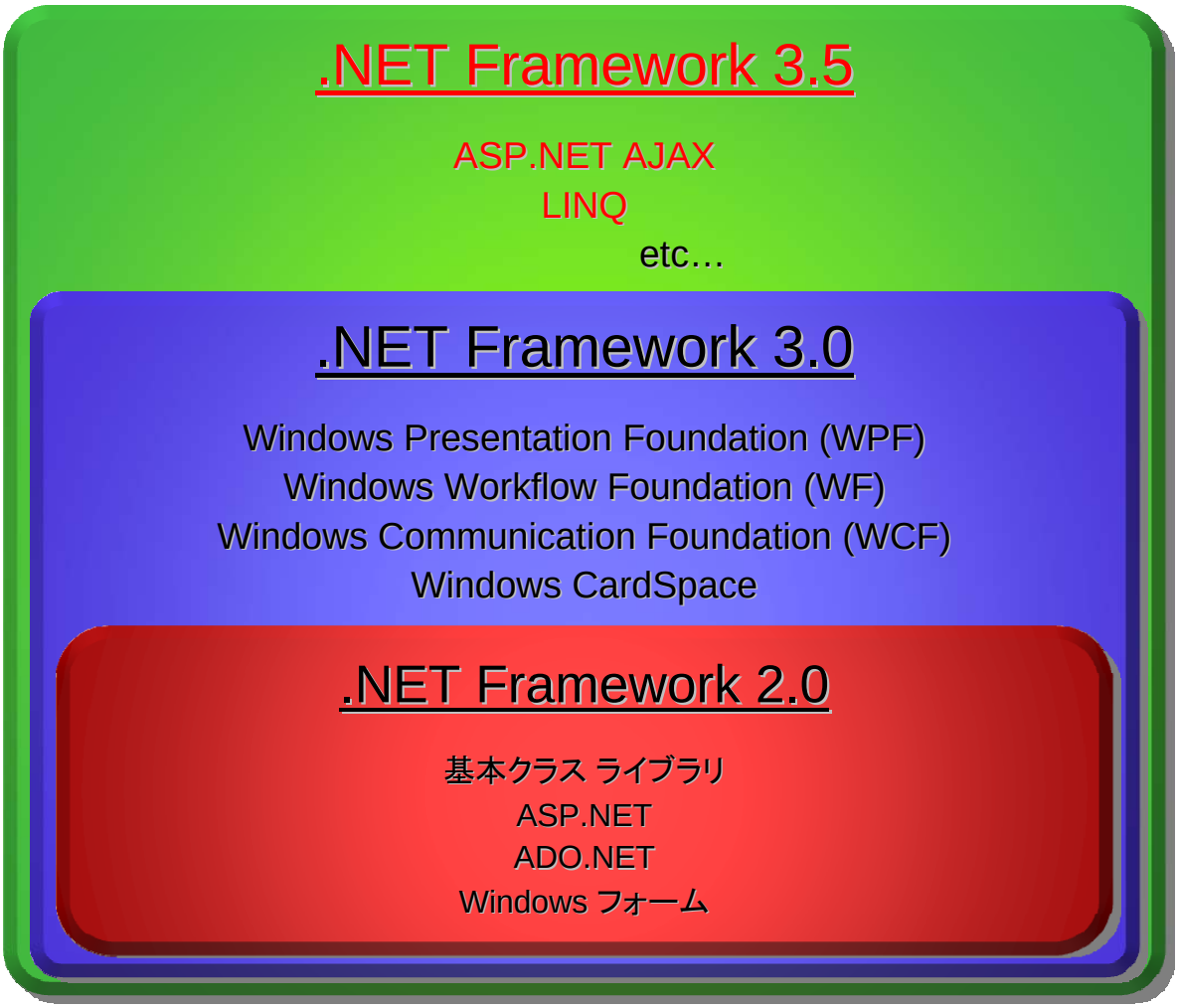

図 1　.NET Framework のバージョンの関係

.NET Framework 2.0 は.NET Framework 1.1 の後継バージョンであり、若干の仕様変更を除けば、.NET Framework 1.1 を置き換えるバージョンです。

一方、.NET Framework 3.0/3.5 は、.NET Framework 2.0 の拡張機能として提供され、.NET Framework 2.0 の共通言語ランタイム(CLR)、.NET FCL（フレームワーククラスライブラリ）上で動作します。

なお、.NET Framework 3.5 は.NET Framework 2.0 SP 1 が動作環境として要求されますので、今後の.NET Framework 2.0 移行においては、SP1 を利用する方がメリットは多いと考えられます。

すなわち、.NET Framework 1.x で構築されたシステムは、.NET Framework 2.0 への移行検討が必須であり、システムに新たな価値を付加するために.NET Framework 3.0/3.5 の新機能の採用を検討すればよいと言えます。

また、.NET Framework 2.0 への移行をしておけば、.NET Framework 3.5 の新機能を利用するためのシステム改修、機能追加を検討することは容易です。

Visual Studio のバージョンの関係

Visual Studio と.NET Framework のバージョンの関係

「表 1　Visual Studio が生成する.NET アセンブリのバージョン」に示すように Visual Studio 2008 以外の Visual Studio は生成する.NET アセンブリのバージョンが 1 つに決まっています。Visual Studio 2008 は、複数バージョンの.NET アセンブリをターゲットにすることができます。

表 1　Visual Studio が生成する.NET アセンブリのバージョン

Visual Studio	生成するアセンブリ
Visual Studio.NET (2002)	**.NET Framework 1.0**
Visual Studio.NET (2003)	**.NET Framework 1.1**
Visual Studio 2005	**.NET Framework 2.0**
Visual Studio 2008	**.NET Framework 2.0**
	.NET Framework 3.0
	.NET Framework 3.5

しかしながら、Visual Studio 2008 でターゲットを.NET Framework 2.0 にしても、Visual Studio 2005 でもビルド可能なソース・コードを生成しない場合があることに注意してください。

Visual Studio の代表的な開発支援機能のバージョン間の関係

Visual Studio の代表的な開発支援機能の関係は次の 2 つの図の様に示すことができます。

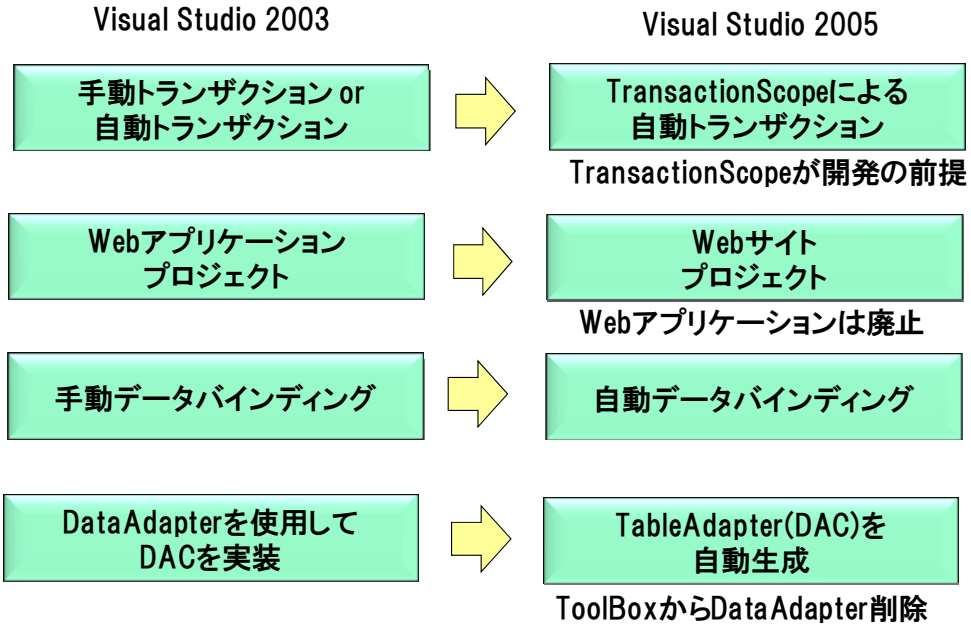

図 2　Visual Studio.NET 2003/2005 の代表的な開発機能の関係

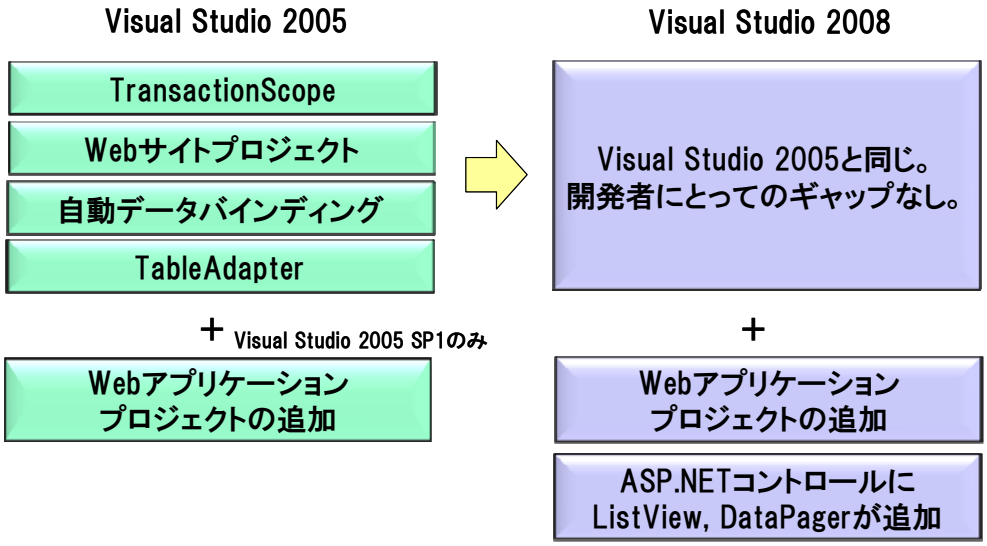

図 3　Visual Studio 2005/2008 の代表的な開発支援機能の関係

「.NET Framework のバージョンの関係」で示した.NET Framework 1.x から.NET Framework 2.0/3.0/3.5 の関係と同様に Visual Studio.NET 2003 と Visual Studio 2005 にはアーキテクチャ変更がありますが、Visual Studio 2005 から Visual Studio 2008 は.NET Framework 3.0/3.5 以外の部分でもわずかな変更にとどまっています。更に Visual Studio 2008 には「図 3　Visual Studio 2005/2008 の代表的な開発支援機能の関係」に示した以外にも、.NET Framework 3.0 新機能

(WCF, WF, WPF など)と.NET Framework 3.5 新機能(LINQ, ASP.NET AJAX など)の開発支援機能が追加されています。

これらの.NET Framework 3.0/3.5 の新機能やアーキテクチャ変更は.NET Framework 1.1 から.NET Framework 2.0 への移行において必ず採用しなければいけないものではありません。アーキテクチャ変更があっても、.NET Framework 1.1 から.NET Framework 2.0 の移行は高い互換性により、比較的容易です。

製品サポートライフサイクルについて

マイクロソフト製品のライフサイクルには以下の 2 つのサポート期間があります。個々のシステムに要求される可用性、信頼性、保守性に応じて、どちらのサポートレベルが必要であるのか、明確にしておく必要があります。

- メインストリームサポート
- 延長サポート

これらのサポートで得られるサポート内容はユーザーごとの契約によって異なります。詳細については、「マイクロソフト サポート ライフサイクル」をご覧ください。

http://support.microsoft.com/default.aspx?scid=fh;[ln];lifecycle

移行についての公式情報

.NET Framework 1.1 と.NET Framework 2.0 間の変更点は、下記の URL で公開されています。移行検討段階ではこの情報を参照して、移行を検討しているシステムに該当する項目があるのか調査してください。

.NET Framework 2.0 での重大な変更点

http://www.microsoft.com/japan/msdn/netframework/programming/breakingchanges/

Breaking Changes in .NET Framework 2.0（原文）

http://msdn.microsoft.com/en-us/netframework/aa570326.aspx

サイド バイ サイド実行（SxS 実行）

.NET Framework は 1 台のコンピュータに複数のバージョンの.NET Framework を導入することができ、お互いに別のランタイムに影響を及ぼさないように設計されています。しかしながら、.NET Framework 1.1 で開発したアセンブリと.NET Framework 2.0 で開発したアセンブリを 1 台のコンピュータに混在させてサイドバイサイド実行させるには、仕様、制限事項について理解しておく必要があります。

一括移行/段階移行の方法を考える上で必要な知識ですが、ここでは概要を紹介するにとどめます。詳しくは、「.NET Framework 1.1/2.0/3.0/3.5 の混在で注意すべきこと」を参照してください。

事前調査・移行検討段階で行うこと

.NET Framework の移行を検討する段階では、目的、制約事項の優先度を決める必要があります。様々な対象システムに関わる条件を調査し、移行計画立案の材料とします。

移行の目的、制約事項と優先度の明確化

システム移行を検討する際には、通常は 1 つ以上の目的があり、制約事項が存在します。それらの目的と制約事項の優先度、リスクを明確にしておいてください。

移行目的の例

- 既存システムの機能を追加、改修することで、ビジネスに新たな価値を付加したり、ビジネスが制約を受ける法改正などに対応したい
- （そのシステム改修の際に）最新プラットフォーム(.NET Framework 2.0/3.0/3.5 や Windows Server 2008 など)の新機能を利用したい
- .NET Framework 2.0/3.0/3.5 を利用しないと実現が困難な JIS 2004 漢字対応や WCF で提供される WS-*の機能を利用するなどのシステム要件が発生した
- .NET Framework 1.1 のメインストリームサポート終了前に、.NET Framework 2.0/3.0/3.5 に移行しておきたい
- 最新プラットフォームの採用により、セキュリティレベルの向上を容易にしたい

制約事項の例

- 既存システムの切り替え日程や時間
- 開発期間
- コスト制約
- 既存のハードウェアの性能
- OS、ミドルウェアのバージョン（社内標準、連携するシステムからの制約なども含む）
- 運用、監視体制

移行しないリスクの例

- 利用している製品ベンダーのサポートが切れ、十分な障害対応ができなくなる

目的と制約事項とそれぞれの優先度、リスクから、移行の範囲、方法を考え、移行計画のゴール設定を行い、移行をしない場合のリスクも考え合わせて、投資対効果、実現可能性を検討してください。

対象とするシステムのライフサイクルとシステム資産の調査、維持・保守計画の立案

移行を検討しているシステムを利用する期間（ライフサイクル）を明確化してください。そのシステムが利用している、.NET Framework 以外の製品（マイクロソフト製品に限りません。ハードウェアも含みます。）のサポート期間よりも長いことが明らかであれば、システムの維持のためには構成要素となる全ての製品のバージョンアップを含めた保守計画が必要です。

製品ライフサイクルの調査対象

- .NET Framework 1.x 上に構築されたアプリケーションプログラム、部品
- .NET Framework 以外で構築されたアプリケーションプログラム、部品
- マイクロソフト製品(OS, RDBMS, .NET Framework, 開発ツールなど)
- マイクロソフト以外の製品(RDBMS, 開発ツール、部品など)
- ハードウェア（サーバー、、クライアント、ネットワーク機器、入出力の周辺機器など）

アプリケーションプログラム以外の構成要素も含めて保守期間を検討し、同じ業務のサブシステムや機能に繰り返しテストしなくてもよいように、他の構成要素も保守期間切れ前に移行しておくと、テスト工数を節約できることもあります。

実行プラットフォームのバージョンアップを行うか

前述したシステムの維持・保守計画に基づいて、ハードウェア、OS、ミドルウェア（RDBMS など）、ライブラリのバージョンアップを今回の移行計画に含めるのか決定してください。OS やミドルウェア、ライブラリのバージョンアップを行う際には、利用する新バージョンの非互換機能の調査、対応方法の検討が必要になることもあります。

例えば、既存プラットフォームで利用している機能よりも向上している新機能（Windows Server 2008 に含まれる Internet Information Server 7.0 の統合モードなど）を利用するための

システム変更も検討してください。システム価値を向上させ、投資対効果を高めることが可能であるかもしれません。逆に Windows Vista や Windows Server 2008 に実装されているユーザアカウント制御(UAC: User Account Control)などにより動作しなくなる機能もあるかもしれません。

どのバージョンに移行するのか

先に「.NET Framework のバージョンの関係」でも述べたとおり、.NET Framework 1.x から.NET Framework 2.0/3.0/3.5 への移行は以下の 2 つの観点で考えることができます。

1. .NET Framework 1.x から.NET Framework 2.0 (SP1)への移行
2. .NET Framework 3.0/3.5 の新機能採否の検討

LINQ など、.NET Framework 3.0/3.5 新機能の採用においては、アーキテクチャ変更を伴うこともありえますが、.NET Framework 2.0 SP1 まで移行したアプリケーションプログラムに.NET Framework 3.0/3.5 の新機能を追加することは、Visual Studio 2008 による開発支援機能もあり比較的容易です。

3rd.パーティ製品

移行するシステムが利用している 3rd. パーティ製品の.NET Framework 2.0/3.0/3.5 の対応版が入手可能であるか調査を行ってください。対応製品が入手できない場合は該当機能の代替手段（類似する製品の採用や自社開発など）の検討が必要です。

アーキテクチャ変更の検討

構築時から要件が変わってしまった場合や既存システムの拡張性、保守性、運用性などに問題がある場合は移行にあたり、アーキテクチャの変更を検討する必要があるかもしれません。

また、先に「実行プラットフォームのバージョンアップを行うか」や「どのバージョンに移行するのか」で述べた通り、プラットフォームや.NET Framework の新機能の採用のためにアーキテクチャ変更が伴うかもしれません。

『.NET Framework 2.0 での重大な変更点』の影響度調査

「.NET Framework 移行検討のための前提知識」で示したとおり、移行を検討するシステムに変更点の影響があるかどうかは、「.NET Framework 2.0 での重大な変更点」を参照して調査を行う必要があります。

ただし、一般的な業務システムにおいては該当する項目はそれほど多くはないものと考えられます（該当する可能性が高い変更点は 30 数個程度です）。

一括移行/段階的移行（サイドバイサイド実行の検討）

.NET Framework 1.1 で実装されたシステムを.NET Framework 2.0 SP1 に一括で移行できるのであれば望ましいのですが、開発期間やコストなどの種々の制約により、全てを一度に移行できることは少ないと考えます。そもそも、単純な一括移行では、ビジネス価値の向上はほとんど望めず、投資対効果が非常に低いため、あまり多くないものと考えられます[1]。

一括移行を行わない場合、機能追加や改修を行うサブシステムだけを段階的に.NET Framework 2.0/3.0/3.5 に移行するなどの段階的移行を選択せざるをえないことがあります。段階的移行を行う場合、.NET Framework 1.x で動作するアセンブリと.NET Framework 2.0/3.0/3.5 で動作するアセンブリが 1 台のコンピュータで混在する「サイドバイサイド実行（SxS 実行）」を利用することになります。

SxS 実行には制限事項がありますので、移行計画立案の段階で「.NET Framework 1.1/2.0/3.0/3.5 の混在で注意すべきこと」の内容を考慮して、実現可能な計画であるのか確認してください。

達成すべき品質レベルとテストの網羅性の検討

.NET Framework 1.1 から.NET Framework 2.0 SP1 への互換性は高いとは言え、システムに求められる安定性、可用性のレベルが高ければ、十分なテストを実施する必要があります。テストの工数は求められる品質保証レベルに応じて、大きくなる可能性があります（最大は「システム構築～保守・改修の通算のテスト工数＋新規開発機能のテスト工数の合計」）。

システムの移行における再テストの工数は、ビジネス価値を向上するものではなく、リスク管理のためのものであり、ミッションクリティカルなシステムであっても投資対効果は低いと言えます。移行計画段階ではテストに費やすコストと達成すべき品質のバランスを考慮して、どこまでテストを実施するのか決定してください。

テスト工数削減方法の例

- 優先度の高い処理だけ入念なテストを行い、優先度の低い処理のテストは軽くする
- システム処理のパターン化がなされている場合、同一パターンの中で代表的なものについてのみテストを行う
- 可用性要件の低い処理、再実行可能な処理のテストは軽くする

[1] システムに要求されるサービスレベルを得るためには、製品サポートレベルが高くないと問題がある場合など。

しかしながら、B2C/B2B システムのトランザクション処理など、ビジネス上の機会逸失、信用失墜につながる可能性のある処理は、網羅的かつ完全なテストを行うことを推奨します。

品質保証の手段としては、新旧システムの並行本番により移行したプログラムの正当性を確認するのも有効です。

移行作業計画

システム移行作業の立案においては、以下のような計画が必要です。

- 既存システムの停止日時の決定
- 基盤構築作業計画
- アプリケーションプログラムの導入、入れ替えの計画
- 新システムのテスト、効率測定
- データ移行プログラム開発計画
- 移行データ検証プログラム開発計画
- 各工程の実施予定時間、クリティカルパス
- 次工程に進むためのクライテリアや移行中止判断のクライテリアの策定
- 各工程担当作業者、承認者、予備要員のアサイン
- 移行中止時の既存システムへのロールバック手順
- 並行本番による検証方法
- 移行完了クライテリアの策定
- 移行リハーサル（複数回）

システム移行後の障害のリスク管理

既存システムを新システムに切り替えた後に問題が発覚することも考えられます。実現可能であれば、旧システムに戻すためのシナリオやその判断基準を事前に決定しておくことや、新システムの障害対応のための要員体制を整えておくなどのコンティンジェンシープランを事前に策定しておくことも重要です。

移行工数見積

ここまで述べてきた移行のための事前調査、検討、計画だけでは、システム移行のコストを正しく見積もることは困難です。次章の「事前検証段階で行うべきこと」を参考にして、移行作業における問題点を事前に洗い出すことで見積精度を向上させることができますが、事前検証作業の実施が困難である場合は「達成すべき品質レベルとテストの網羅性の検討」を参考にして移行工数の最低ラインを考えることもできます。

事前検証段階で行うべきこと

事前検証段階では、事前調査・検討段階では予知不能なリスクを抽出することを目的にします。事前検証でエラーが発生する場合には移行に必要な改修作業項目となります。

事前検証段階で移行に伴う改修作業を完了させてしまい、ランタイムエラー以外の問題を全て解決してしまえば、後続の工程でリスクを減らすことが可能です。.NET Framework 1.1 から.NET Framework 2.0/3.0/3.5 への移行は比較的容易であるため、少々余裕のあるスケジュール、コストと要員体制を確保できるのであれば、それほど困難ではありません。

この改修作業を事前検証段階で完了させてしまうか、移行プログラムの開発工程に実施するのかは、制約条件次第で判断してください。

移行前システムのソリューションファイル一式の入手、ビルド試行環境、実行試行環境の構築

事前検証を行うためには、Visual Studio 2008 を導入したビルド環境に、Visual Studio.NET 2003 で作成された移行前システムのソリューションファイル一式、その他のビルドに必要なライブラリなどを導入します（以降、「ビルド試行環境」と呼びます）。

また、OS やミドルウェア類を導入し、必要な設定を施した実行試行環境も用意します（以降、「実行試行環境」と呼びます）。

事前コード修正

既存システムで「『.NET Framework 2.0 での重大な変更点』の影響度調査」に該当する修正項目があれば、事前に対処するか、移行プログラムの開発工程の作業項目とします。プログラムだけではなく、設定ファイルの設定を行わなければならないこともあります。また、3rd. パーティ製品を利用している箇所については、各ベンダーが提供する情報を基にコード修正などを行います。中には独自の移行ウィザードを提供している製品もあります。

ソリューションファイル変換テスト

この工程は段階移行を行う場合、移行するソリューションファイルのみを対象にします。

ビルド試行環境で、Visual Studio.NET 2003 で作成されたソリューションファイル一式を Visual Studio 2008 で開くと、自動的に変換ウィザード[2]が起動し、Visual Studio 2008 形式のソリューションに変換[3]されます。

---

[2] Visual Studio 2005 にも同様に移行ウィザードが含まれています。
[3] Visual Studio 2008 のプロジェクト移行ウィザードによる、Visual Studio.NET 2003 用のプロジェクトファイル一式の変換は不可逆な処理ですので、バックアップを取っておく必要があ

この変換において発生したエラーや警告は、ログ・ファイルを参照して適切に対処してください。

詳しくは、後述する「移行作業ポイント」を参照してください。

一括リビルドテスト

この工程は段階移行を行う場合、移行するソリューションファイルのみを対象にします。

ビルド試行環境でソリューションファイルの変換、エラーと警告に対する対処を行うことができたら、全プロジェクトファイルのリビルドを実施します。

一括リビルドでは、「事前コード修正」で漏れていたコード修正点が C#/VB.NET コンパイラからエラー、警告が発せられる可能性があります。「『.NET Framework 2.0 での重大な変更点』の影響度調査」も参考にしながら修正を行うか、実際の移行プログラム開発作業の見積りの参考としてください。

また、.NET Framework 1.1 プログラムを移行する際に推奨されないコーディング[4]については、コンパイラが示してくれます。

ランタイムエラーテスト

ソリューションファイル変換、リビルドが完了した場合、.NET アセンブリを実行試行環境にデプロイし、テストを実施し、ランタイムエラーが発生するか確認します。設定変更などにより障害への対処ができない場合には、実際の移行プログラム開発作業の工数見積りの参考としてください。

段階的移行を行う場合には、.NET Framework の下位互換機能の利用、サイドバイサイド実行の利用を伴う可能性がありますので、一括移行に比べて注意が必要です。

ります。

[4] 「【問題点・注意点 4】古いコーディング形式に対する警告」を参照してください。

移行計画策定

移行計画の策定段階では、前述の「事前調査・移行検討段階で行うこと」、「事前検証段階で行うべきこと」で収集してきた移行対象システムの情報により、システム化計画を行います。

システムの移行による新たなシステム価値と工数見積を基にして、ユーザー企業のシステム部門の方であれば経営層に対して稟議を上げ、SIer であればお客様に対して提案を行うことができます。

システム移行の実行許可が得られれば、通常のシステム開発と同様にシステム化計画を策定してください。本書を参考にして、漏れのない、綿密な作業計画とリスク管理を行い、移行作業の失敗を防いでください。

移行作業ポイント

ここでは、Web アプリケーションの移行において、特に注意の必要な Visual Studio 2005、Visual Studio 2005 SP1、Visual Studio 2008 の移行ウィザードの差異に触れながら Web アプリケーション移行について説明します。

移行の判断

Visual Studio 2005/2008 フォーマットへの変換は、非互換部分の改修や再テストが発生する可能性があります。一方、サイドバイサイド実行（SxS 実行）を利用する場合には、サイドバイサイド実行の制限事項に抵触しないことを事前に検討する必要があります。.NET CLR の下位互換性を利用する場合には再テストが必要であるかもしれません。

以上のことを検討し、移行すると判断した場合は、次に Web 開発モデルを決定します。Visual Studio.NET 2003 Web プロジェクトモデルに近い開発を行う場合は、Visual Studio 2005 SP1[5]もしくは Visual Studio 2008 を使用します（Web アプリケーションプロジェクト[6]）。Visual Studio 2005 から登場した Web 開発モデル（Web サイトモデル）を使用する場合は、一度 SP1 が当たっていない Visual Studio 2005 で変換を行う必要があります（Web サイトプロジェクト[7]）。各 Web 開発モデルのメリット、デメリットを「表 2　Web 開発モデルのメリット、デメリット」に、具体的な採用フローを「図 4　Web 開発モデルの採用フロー」に示します。

[5] http://msdn.microsoft.com/msdnmag/issues/07/04/ExtremeASPNET/default.aspx?loc=jp

[6] http://msdn.microsoft.com/ja-jp/library/9d9ats98(VS.80).aspx

[7] http://msdn.microsoft.com/ja-jp/library/9d9ats98(VS.80).aspx

表 2　Web 開発モデルのメリット、デメリット

選択肢		メリット	デメリット
変換を行う	Web サイトモデル <Visual Studio 2005>	・ASP.NET 2.0 の新機能 ・ASP.NET 2.0 のセキュリティ＆パフォーマンス強化 ・様々な配置パターン ・ASP.NET 開発サーバーでの開発も可能 ・ツール[8]を使用することによりコマンドラインから Web サイトを発行することが可能。	・変換による改修コスト
	Web Application Projects (WAP) <Visual Studio 2005 SP1, Visual Studio 2008>	・ASP.NET 2.0 の新機能 ・ASP.NET 2.0 のセキュリティ＆パフォーマンス強化 ・なじみのある Web プロジェクトモデルでの開発 ・出力アセンブリ名の制御 ・ビルド前後のイベントが指定可能	・変換による改修コスト ・ASP.NET 開発サーバーでの開発が不可能 ・コマンドラインから発行するツールが用意されていない[9]。
変換を行わない	ASP.NET 1.1	・変換による改修コストが発生しない	・ASP.NET 2.0 の新機能が使用できない ・ASP.NET 2.0 のセキュリティ＆パフォーマンス強化が使用できない
	ASP.NET 2.0	・ASP.NET 2.0 のセキュリティ＆パフォーマンス強化 ・変換による改修コストが発生しない	・ASP.NET 2.0 の新機能が使用できない ・下位互換性のテストが必要

8 Aspnet_compiler.exe。詳細は MSDN を参照。

9 Visual Studio 2005 から発行、もしくは Web 配置プロジェクトで可能である。

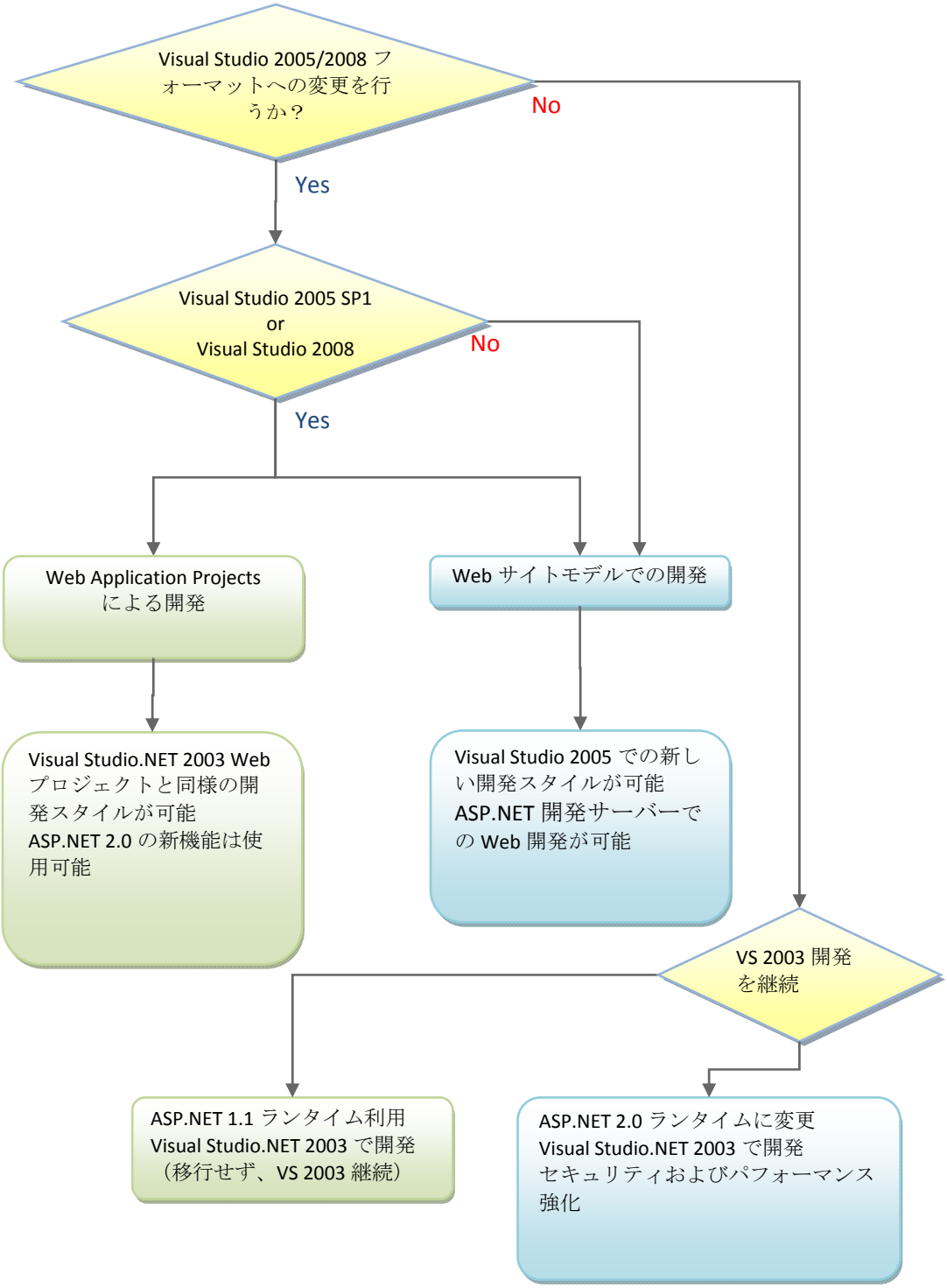

図 4　Web 開発モデルの採用フロー

移行手順

ここでは移行を行うことを決定した場合の手順（以下の 3 つ）を順に追って説明します。

1) 移行前の準備・確認事項

2) 変換ウィザード（自動変換）

3) 移行時の問題点・注意点

移行前の準備・確認事項

Web プロジェクトの URL と IIS を確認

Visual Studio.NET 2003 の Web プロジェクトは URL、IIS のメタデータによってプロジェクトファイルの物理パスが解決されます。そのため、Visual Studio 2005/2008 での変換を行う前には IIS のメタデータを同じにする、もしくは解決される物理パス上に Web プロジェクトを配置しなければなりません[10]。もちろん、この作業は同じマシンで行う場合には必要がありません。そのため、変換環境には Visual Studio.NET 2003 で正常にソリューションを読み込めるマシンに Visual Studio 2005/2008 をインストールすることを推奨します。

また、Visual Studio 2005/2008 からファイル、クラス名に使えない文字が増えました。そのため、変換対象であるソリューションに含まれるファイル名を確認する必要があります。詳細は「【問題点・注意点 7】クラス名に含まれる特殊文字」を参照してください。

Visual Studio 2005 フォーマットでの変更点

Visual Studio 2005 の Web サイトモデルに移行する場合に（Visual Studio.NET 2003 からの）注意しなければならない主要な変更点を説明します。ただし、Visual Studio 2005 SP1、Visual Studio 2008 は対象外です。

参考：Web プロジェクトの一般的な変換問題およびソリューション
http://www.microsoft.com/japan/msdn/net/aspnet/conversionissuesasp_net.aspx

Web プロジェクトファイルが存在しない

ASP.NET 2.0 では Web プロジェクト構成物を管理するプロジェクトファイルは存在せず、Web サイト内にあるファイルすべてがプロジェクトに含まれることになります。

プロジェクトの情報はソリューションファイルや Web.config で管理されます。例えばプロジェクト参照の参照先情報はソリューションファイルで管理され、Web サービス参照の参照先 URL は Web.config で管理されています。

[10] IIS がインストールされていない、メタデータによる物理パスが解決できない場合は、1) Web プロジェクトが認識できない、2) 異なる箇所に空の Web プロジェクトが作成される、などの変換ミスが生じます。

特殊フォルダの追加

ASP.NET 2.0 では bin フォルダのほかに App_Code や App_Data 等の特殊フォルダを追加することができ、用途別にソースファイルが格納されます。

例えばコードビハインド以外のクラスファイルや DataSet は App_Code フォルダに格納され、グローバルリソースファイルは App_GlobalResources に格納されます。

図 5　特殊フォルダの追加

コードビハインドモデル

ASP.NET 1.x では Default.aspx.cs のようなコードファイルにデザイナ生成コードと開発者が記述するコードの両方が含まれていました。ASP.NET 2.0 ではコードビハインドモデルが強化され[11]、デザイナ生成コードと開発者が記述するコードはそれぞれ別のファイルに書き込まれるようになりました。

そのため開発者が記述するコードファイルには、サーバーコントロールをインスタンス化するような記述はなく、さらにデザインとロジックを分離した開発ができるようになっています。

11 パーシャル クラス（http://msdn.microsoft.com/ja-jp/library/wa80x488.aspx）

変換ウィザード（自動変換）

変換ウィザードの使い方

1. 変換ウィザードの開始

Visual Studio 2005/Visual Studio 2008 で Visual Studio.NET 2003 フォーマットのソリューションもしくはプロジェクトを開くことにより、変換ウィザードが開始されます。

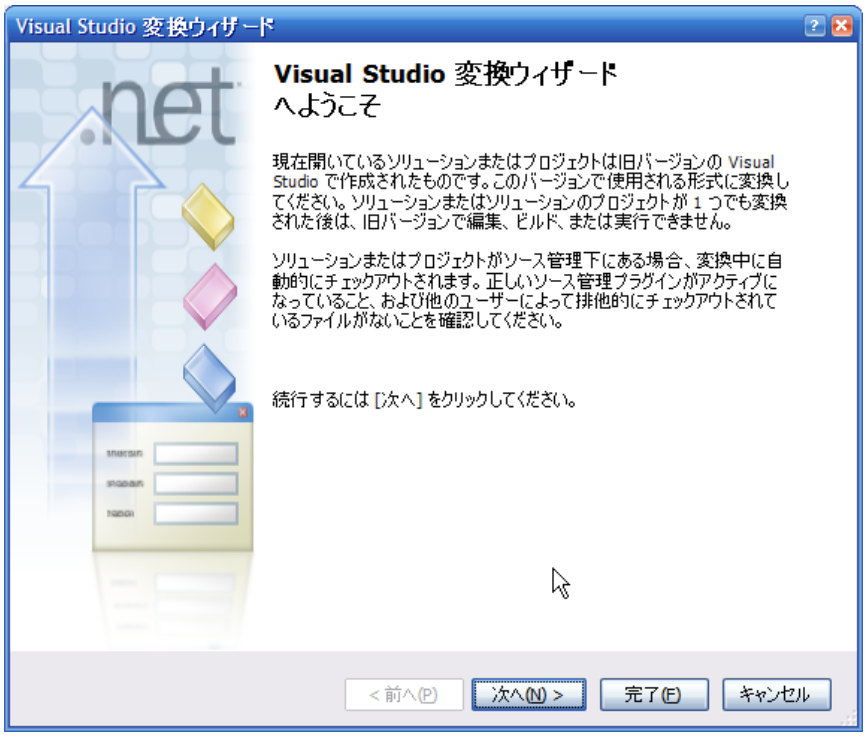

図 6　Visual Studio の変換ウィザード

2. バックアップ作成の選択

変換すると以前のバージョンとの互換性は失われるので、バックアップすることを推奨します。

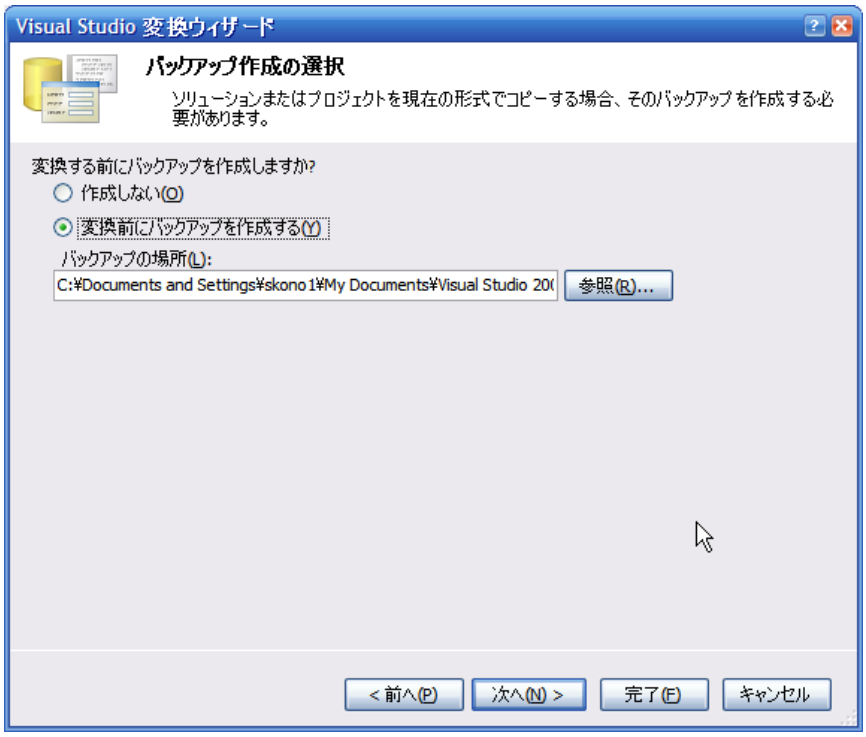

図 7　変換ウィザードーバックアップ作成の選択

ソリューション、プロジェクトの変換は不可逆処理であるので、事前にバックアップを取っていない場合は、このバックアップ処理を実行してください。

3. 変換準備完了

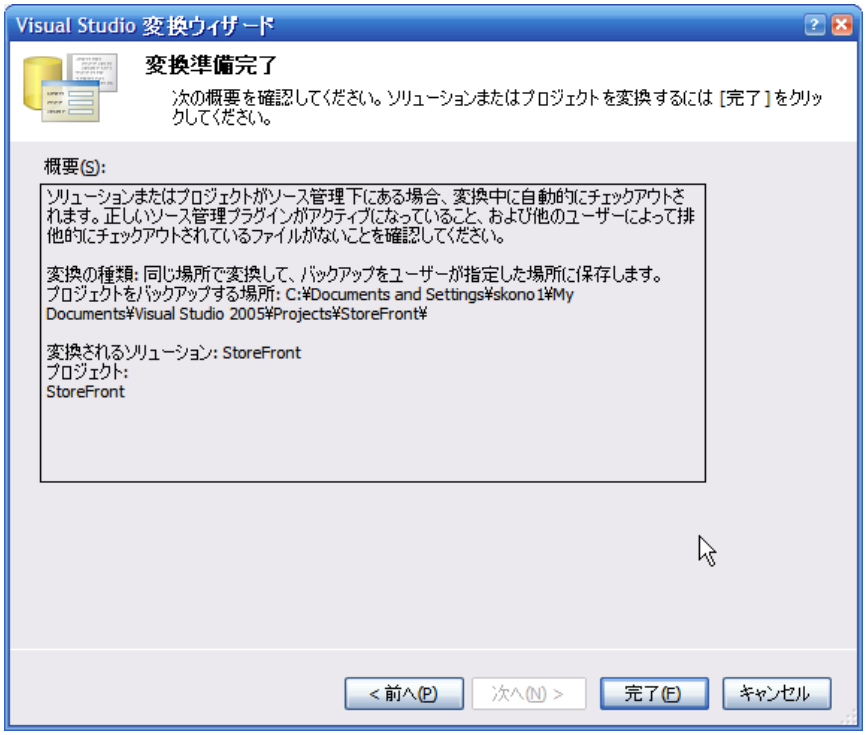

図 8　変換ウィザードー変換準備完了

4. Visual Studio 2008 の場合、次のダイアログが表示されます。

.NET Framework 3.5 を使用する場合は「はい」、2.0 を使用する場合は「いいえ」ボタンをクリックします。ただし、.NET Framework のバージョンを選択できるのは Web プロジェクトのみで、クラスライブラリなどは 2.0 へ変換されます。

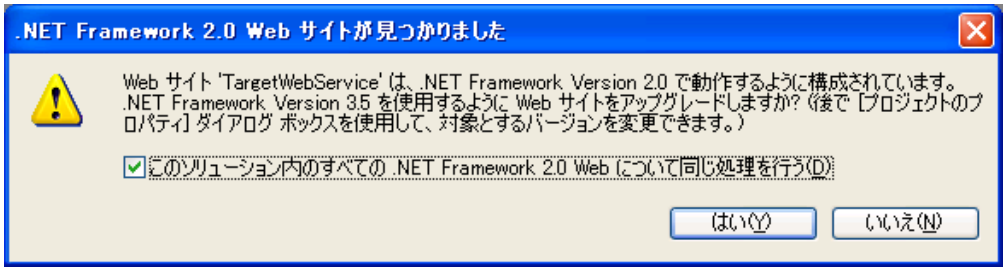

図 9　変換ウィザードー.NET Framework 2.0/3.5 の選択

5. 変換の完了

変換エラーが発生した場合は変換ログを確認します。

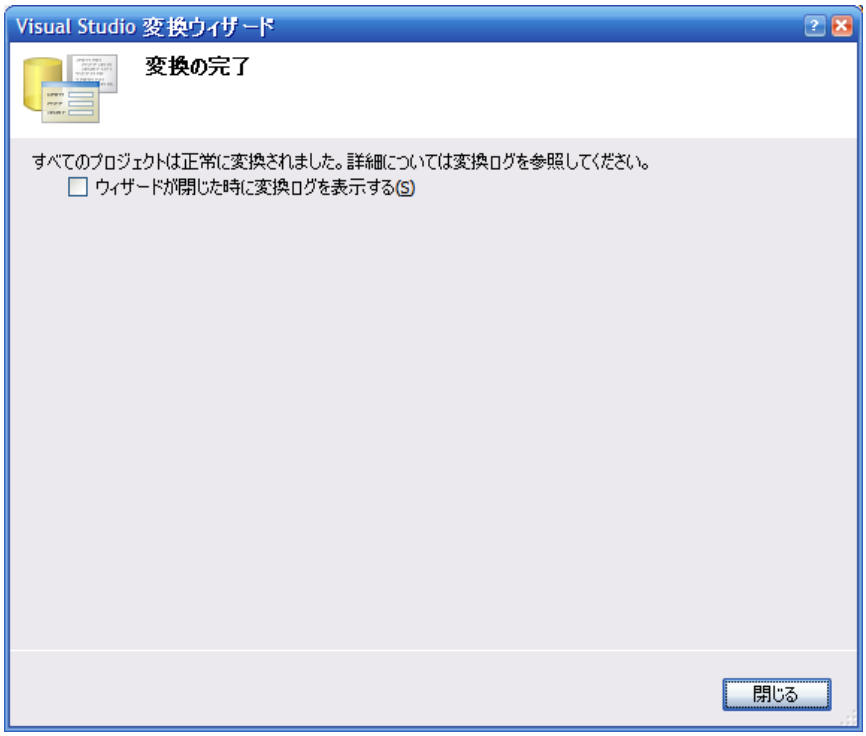

図 10　変換ウィザードー変換の完了

6. Visual Studio 2005 SP1、Visual Studio 2008 を適用した場合 Web プロジェクトは WAP の形式に変換されます。ただし、変換ウィザードでの変換は最初のステップであり、完了していません。そのため全ての変換作業を完了するためには、さらに「Web アプリケーションに変換」を実施してください。

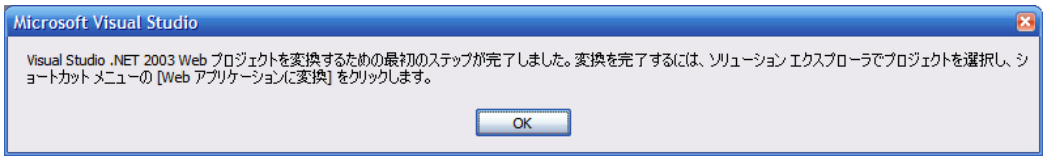

図 11　変換ウィザードー完了後メッセージ

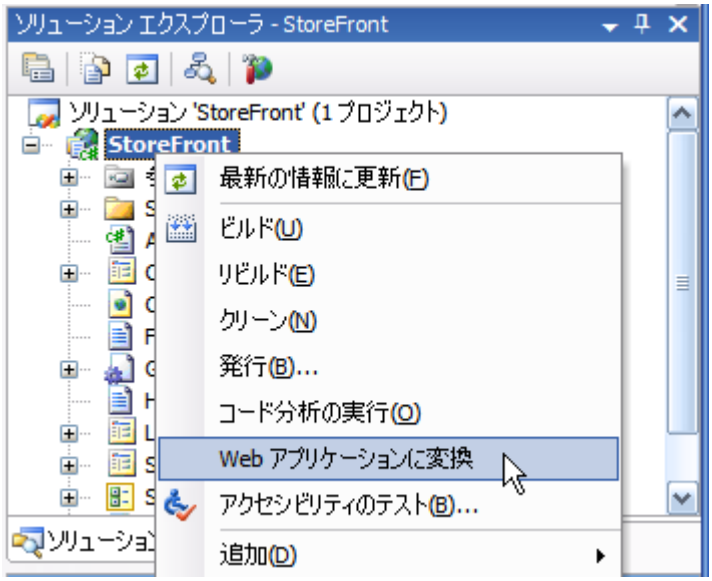

図 12　Web アプリケーションへの変換

なお、ソリューション、プロジェクトファイルの変換後に変換レポートが表示され、変換中にエラー、警告が発生した場合には内容を確認できます。

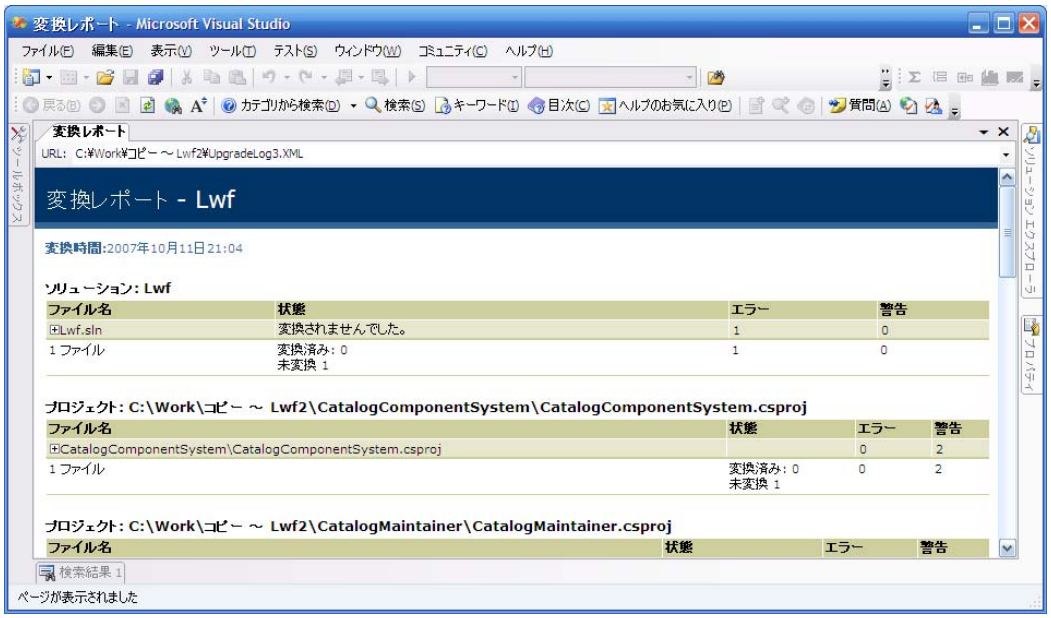

図 13　変換レポート

エラー、警告に対して、適切な修正を加えてください。

変換ウィザードで変換されるもの、されないもの

- desinger.cs の作成で自動生成コードと開発者が記述するコードが振り分けられる。
- 新しいプロパティと属性が作成され、旧式のプロパティの属性が削除される。
- Visual Studio.NET 2003 で作成された DataAdapter, Connection の自動生成コードは、DataTable へ書き換えられない。
- Visual Studio.NET 2003 で作成した Fill、Update メソッドがあれば変更されずに移行され、作成していない場合は自動的に作成されるない。
- Properties フォルダの新規作成は行われない。

また、AssemblyInfo.cs ファイルの移動も行われません。Visual Studio.NET 2003 で AssemblyInfo に[assembly:AssemblyKeyFile(“~~.snk”)]で厳密名を指定した場合、Visual Studio 2005 でリビルドを行うと、警告が出力されます。修正するためには、Properties フォルダより GUI で署名タブの設定を行わなう必要があります。そのため、各自 Properties フォルダを新規作成しなければなりません。

「移行時の問題点・注意点」の「【問題点・注意点 8】変換後に変更されるファイル」も参照してください。

http://msdn.microsoft.com/ja-JP/library/bw8z90c1.aspx

変換ウィザードで変換後に注意すべきこと

Visual Studio 2005 ではエラーチェックの強化が図られているため、Visual Studio.NET 2003 ではエラーや警告にならなかった下記の事象がコンパイル時に問題になることがあります。

- 暗黙の変換が行われないような方法でメソッドパラメータ指定を行った時
- 論理式で到達できない式コードが検出された時
- クラスに static と sealed の両方を指定していた時
- アセンブリの署名属性を使用した時

このような場合は修正を行ってください。

移行時の問題点・注意点

以降に.NET Framework 1.1 ベースのアプリケーションを.NET Framework 2.0 ベースのアプリケーションに変換する際の問題点・注意点を列挙します。

【問題点・注意点 1】Web サイトのプロジェクト参照

開発環境

Visual Studio 2005

現象

Web サイトからのプロジェクト参照が変換できない。

プロジェクト: StoreFront.csproj

ファイル名	状態	エラー	警告
⊟ StoreFront.csproj:(Error List)		4	0
変換の問題 - StoreFront.csproj:(Error List):			
エラー: プロジェクト参照 Diagnostics を変換できません。			
エラー: プロジェクト参照 Exceptions を変換できません。			
エラー: プロジェクト参照 CatalogComponentSystem を変換できません。			
エラー: プロジェクト参照 RetailComponentSystem を変換できません。			
⊞ StoreFront.csproj:(Warning List)		0	1
⊞StoreFront.csproj	変換済み	0	0
3 ファイル	変換済み: 1 未変換 2	4	1

図 14　プロジェクト変換時のエラーメッセージ

原因

APS.NET 2.0 ではサイト内に含まれるファイルはすべてプロジェクトに含まれてしまうため、bin 配下にアセンブリファイルが存在し、プロジェクト参照が設定されていないとアセンブリ参照と判断してしまいます。

対処方法

KB 898904 に対するアップデートまたは Visual Studio 2005 SP1 を適用することにより解決されます。
http://support.microsoft.com/kb/898904/ja

【問題点・注意点 2】不要な resx ファイルがサイトに含まれる

開発現象

Visual Studio 2005、Visual Studio 2005 SP1、Visual Studio 2008

現象

Default.aspx.resx のような不要なローカルリソースファイルが多数 Web サイトに含まれてしまう。

原因

Visual Studio .NET 2003 は必要であるなしにかかわらず、これらリソース ファイルをすべての aspx、ascx に対して自動で作成していたためです。

対処方法

多くの場合このリソースファイルの内容は空か意味をなさないものなので、国際化対応等で明示的に使用していない限り削除して構いません。

【問題点・注意点 3】aspx/ascx/asmx ファイル名が小文字に変更される

開発環境

Visual Studio 2005

現象

Web サイトのルート以降に配置してある aspx、ascx、asmx ファイルが変換後にすべて小文字に変換されてしまう。（コードファイルは誤変換されない）

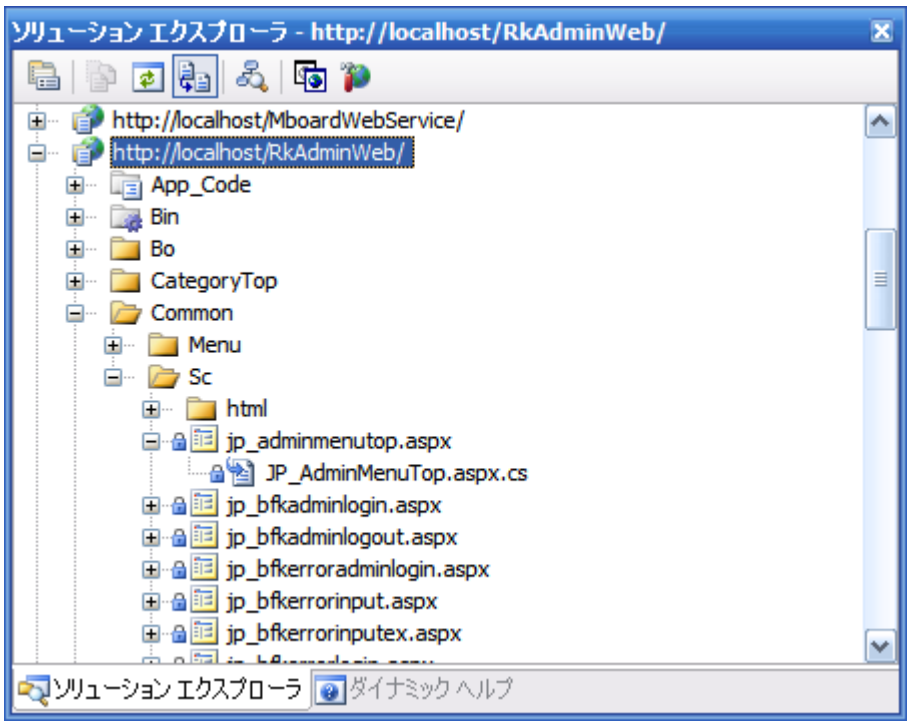

図 15　aspx/ascx/asmx の誤変換

対処方法

KB 898904 に対するアップデートまたは Visual Studio 2005 SP1 を適用することにより解決されます。

【問題点・注意点 4】古いコーディング形式に対する警告

開発環境

Visual Studio 2005、Visual Studio 2005 SP1、Visual Studio 2008

現象

.NET Framework 2.0 では使用を推奨しない古い形式のコーディングが警告される。

対処方法

古い形式も互換性のために機能的には残されているので、このまま使用しても構いませんが、警告を消すためには改修もしくは特定の警告非表示の設定が必要となります。
http://msdn.microsoft.com/library/ja/cscomp/html/vcerrnowarnsuppressspecifiedwarnings.asp

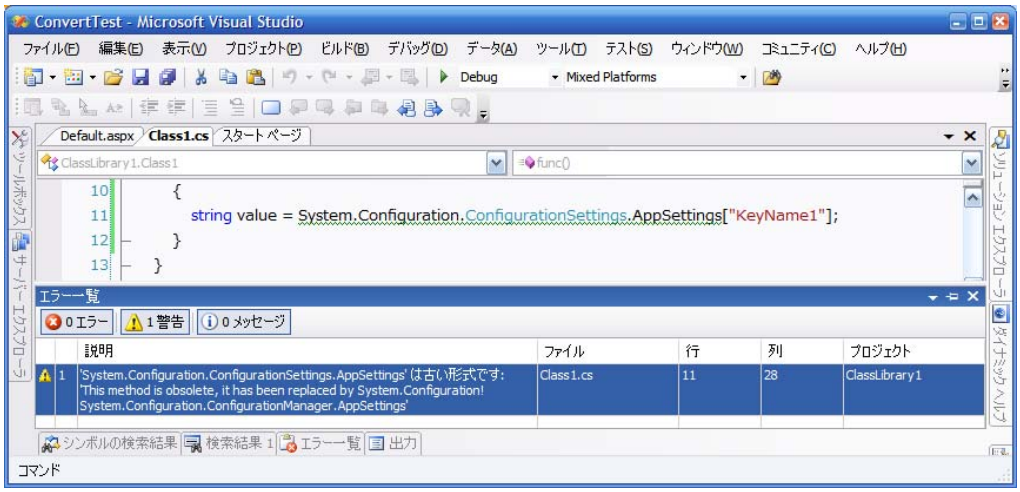

図 16　古いコーディング形式に対する警告

> 警告 CS0618: 'System.Configuration.ConfigurationSettings.AppSettings' は古い形式です: 'This method is obsolete, it has been replaced by System.Configuration!System.Configuration.ConfigurationManager.AppSettings'

コードの改修

古い形式：System.Configuration.ConfigurationSettings.AppSettings["KeyName1"];

新しい形式：System.Configuration.ConfigurationManager.AppSettings["KeyName1"];

特定の警告非表示の設定

Web サイトモデルの Web サイトに対しては設定できません。（WAP では可能）

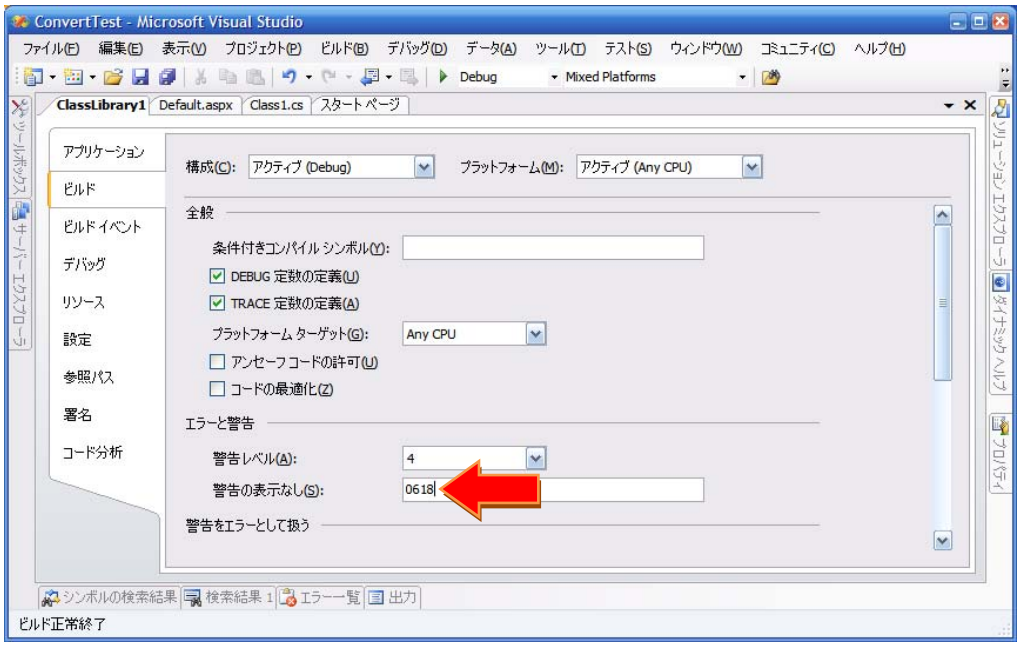

図 17　特定の警告非表示の設定

以上の設定を行った場合のビルドコマンド例

```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Csc.exe /noconfig
/nowarn:0618,1701,1702・・・
```

【問題点・注意点 5】HTML の検証エラー

開発環境

Visual Studio 2005、Visual Studio 2005 SP1、Visual Studio 2008

現象

aspx や ascx を Visual Studio 2005 のデザイナで表示しようとすると HTML 検証ターゲットを XHTML1.0 Transitional 等に設定している場合エラーが表示される。

原因

Visual Studio.NET 2003 で作成した aspx や ascx ファイルが XHTML 1.0 Transitional に準拠していないためです。コンパイル エラーではないため、その多くがアプリケーションの機能には影響せず、ページを表示することも可能です。

しかし、このエラーが表示されると、他のエラーと区別が付きづらく、エラーが表示されないように改修するか、検証レベルを下げる必要があります。

対処方法

検証ターゲットを Internet Explorer(IE) 6.0 に変更することによって Visual Studio.NET 2003 フォーマットで記述したソースに対するエラーのほとんどを非表示にすることができます。

ただしクライアント対象ブラウザが IE6 や IE7 以外にも存在する場合には表示上問題が発生する場合があるので、ブラウザで表示して確認する等のテストが必要です。

① 「HTML ソースの編集」タグでの変更

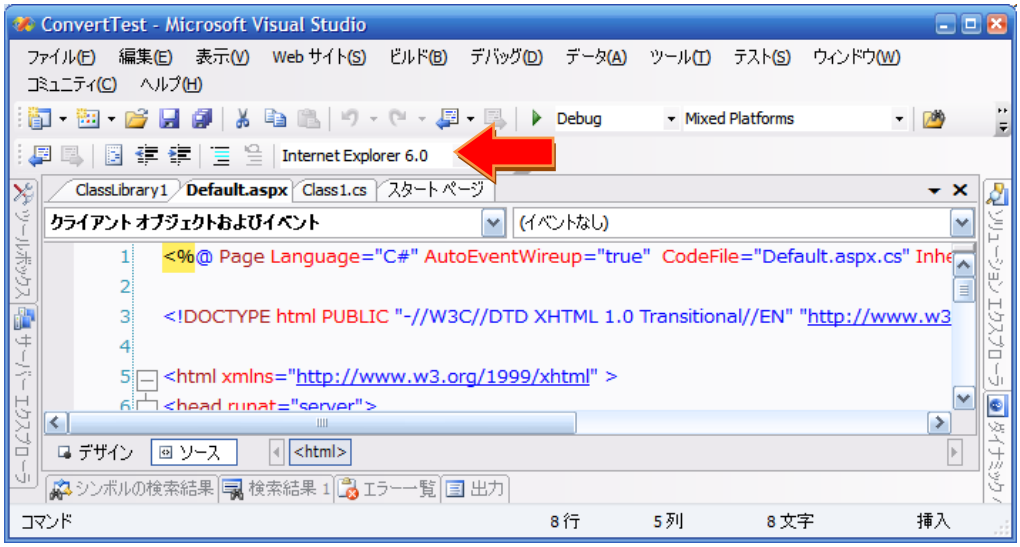

図 18 HTML 検証ターゲットの変更 1

② 「ツール」-「オプション」からの変更

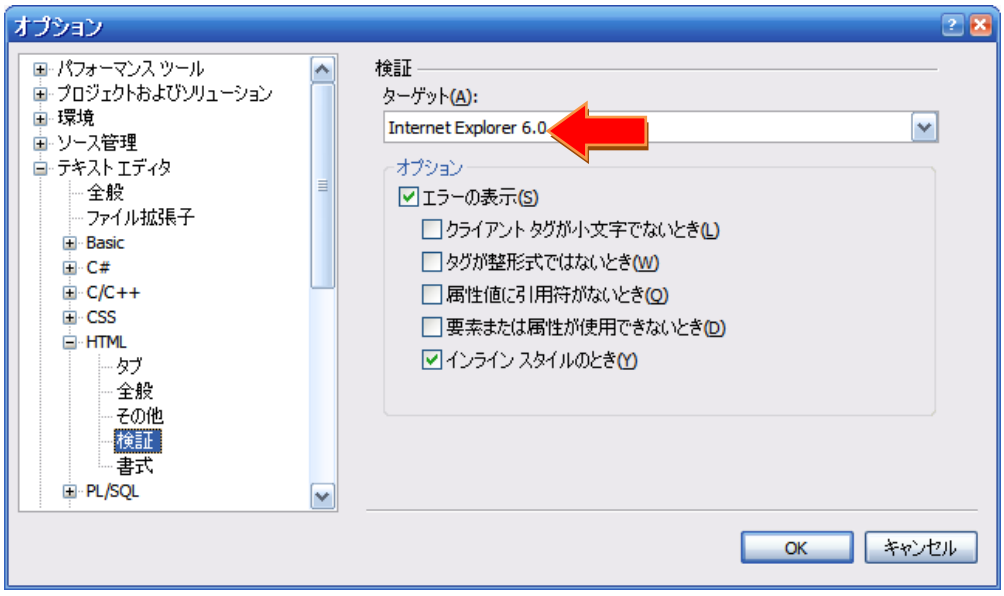

図 19 HTML 検証ターゲットの変更 2

【問題点・注意点 6】型付 DataSet の変換

開発環境

Visual Studio 2005、Visual Studio 2005 SP1、Visual Studio 2008

現象

Visual Studio 2005/2008 へ変換した場合、型付 DataSet はソースコードファイル名[12]、実装が変更され、インターフェース（型付 DataSet、型付 DataTable）が増える。

対処方法

変換を防止する方法はありません。削除されるメソッドやプロパティはなく、変換前に VS 2003 でコンパイル可能でであれば、変換後も VS 2005/2008 でコンパイルは可能であるはずです。変換後は Visual Studio 2005/2008 の型付 DataSet となり、VS 2003 の型付 DataSet のロジックが変換されます。非常に品質要求の厳しい開発プロジェクトでは再テストを検討してください。

> 注：厳密には.NET Framework 2.0 と 3.5 で型付 DataSet の実装が異なります[13]。ただし、Visual Studio.NET 2003 からの変換では強制的に.NET Framework 2.0 の型付 DataSet へ変換されます。3.5 への型付 DataSet に変換するためにはプロジェクトファイルで.NET Framework 3.5 を選択し、カスタムツールを実行しなければなりません。

【問題点・注意点 7】クラス名に含まれる特殊文字

開発環境

Visual Studio 2005、Visual Studio 2005 SP1、Visual Studio 2008

現象

- 特殊な文字を使用した場合、指定したクラス名、ファイル名がアンダースコアに変換される。
- 特殊な文字を使用しているファイル名を Visual Studio 2005 で変換した場合、変換後のコンパイルでエラーとなる。

対処方法

Visual Studio 2005 で変換する前に特殊文字をアンダースコアに変換します。

特殊文字一覧

.NET Framework 1.1 から 2.0 に移行するにあたり、.NET Framework がサポートする UNICODE のバージョンが変更されています。そのため、Visual Studio.NET

12 ソースコードファイル名が DataSet1.cs の場合、DataSet1.Designer.cs となる。

13 .NET Framework 2.0 では型付 DataTable の継承元が System.Data.DataTable と System.Collections.IEnumeable であり、NET Framework 3.5 では System.Data.TypedTableBase<T>である。

2003 ではクラス名に使えた文字が Visual Studio 2005 では使えなくなったものが存在します[14]。以下にその一覧を示します。日本語環境の場合は基本的に最後の 2 つの中黒が使用可能で、残りの文字はフォントが存在しないため使用されていないと考えられます[15]。

表 3　変換される特殊文字一覧（中黒を除き、表示されない）

UTF-16	文字	分類
0x1369	[፩]	エチオピア文字
0x136a	[፪]	同上
0x136b	[፫]	同上
0x136c	[፬]	同上
0x136d	[፭]	同上
0x136e	[፮]	同上
0x136f	[፯]	同上
0x1370	[፰]	同上
0x1371	[፱]	同上
0x17b4	[□]	クメール文字
0x17b5	[□]	クメール文字
0x30fb	[・]	カタカナ
0xff65	[･]	半角カタカナ

Visual Studio.NET 2003 からの変換の際、上記一覧最後の 2 文字の中黒（0x30fb, 0xff65）が含まれるファイル名が 2 つ以上ある場合コンパイルエラーになることがあります。これは.NET Framework 1.1 では名前の 1 文字として認識されていたが、2.0 では名前を区切る文字（他には、例えばドット:'.'）として認識されるためです。従って、Visual Studio 2005 へ変換する前にファイルにこれら文字が含まれているか確認する必要があります。

コンパイルエラーメッセージ

14 これらの文字を使用した場合は強制的にアンダースコア（UTF-16 で 0x005f）に変換される。エラー、警告ダイアログは表示されない。

15 各国語に対応した言語パックをインストールすれば、利用可能である。

エラー 1 項目 "obj\Debug\ClassLibrary1.「クラス名の一部」.resources" は "Resources" パラメータで 1 度以上指定されました。重複した項目は "Resources" パラメータではサポートされていません。 ClassLibrary1

http://support.microsoft.com/kb/948575/ja

【問題点・注意点 8】変換後に変更されるファイル

開発環境

Visual Studio 2005、Visual Studio 2005 SP1、Visual Studio 2008

現象

ウィザード画面での変換後、自動生成ファイル（型付 DataSet のソースファイルなど）はロジックが変更される。一覧は以下の通りです。

表 4　ウィザードにより変換されるファイル一覧

ファイル	開発環境	説明
データセットファイル	Visual Studio 2005, Visual Studio 2005 SP1, Visual Studio 2008	・ファイル名が以下のように変更。 XXXDataSet.cs →XXXDataSet.Designer.cs ・ロジック（コード）が変更される。
Web サービスプロキシクラスファイル[16]	Visual Studio 2005 SP1, Visual Studio 2008	・ロジック（コード）が変更される。 *注：Visual Studio 2005 と Visual Studio 2005 SP1 の間には Web サービスプロキシクラスの変更がない。*
Web.config	Visual Studio 2008 (.NET Framework 3.5 への変換のみ)	項目が追加される。 例：<system.codedom> や <system.webServer> など
Aspx ファイル	Visual Studio 2005, Visual Studio 2005 SP1, Visual Studio 2008	・@Page ディレクティブの AutoEventWrapper 属性値が false → true に変更 *注：Visual Studio 2005 の場合は AutoEventWrapper 属性が削除される。* ・サーバータグにイベント ハンドラ名が記述される 例：Button コントロールの Click イベント
Aspx.cs ファイル	Visual Studio 2005, Visual Studio 2005 SP1, Visual Studio 2008	・Page クラスが partial に変更 ・Page_Load メソッドのアクセス修飾子が private → protected に変更 ・InitializeComponent メソッドからイベント ハンドラの登録コードが削除（Aspx ファイルに記述されたため）

16 ファイル名：Reference.cs 。ソリューション エクスプローラの「すべてのファイルを表示」ボタンが押されていないと表示されない。

対処方法

特になし

.NET Framework 1.1/2.0/3.0/3.5 の混在で注意すべきこと

サイドバイサイド実行とは

.NET Framework は、単一のシステムに複数バージョンの.NET Framework CLR（ランタイム）を混在可能な設計になっています。その様なシステム環境において、.NET アセンブリはデフォルトでは、コンパイラによってビルドされた時のバージョンをメタデータに持っており、と同じバージョンのランタイムによって実行されます。次図の例では、アセンブリ A は.NET CLR 1.1 で実行され、アセンブリ B は.NET CLR 2.0 で実行されます。

このビルド時と同じバージョンのランタイムでアセンブリが動作する機構をサイドバイサイド実行と呼びます。

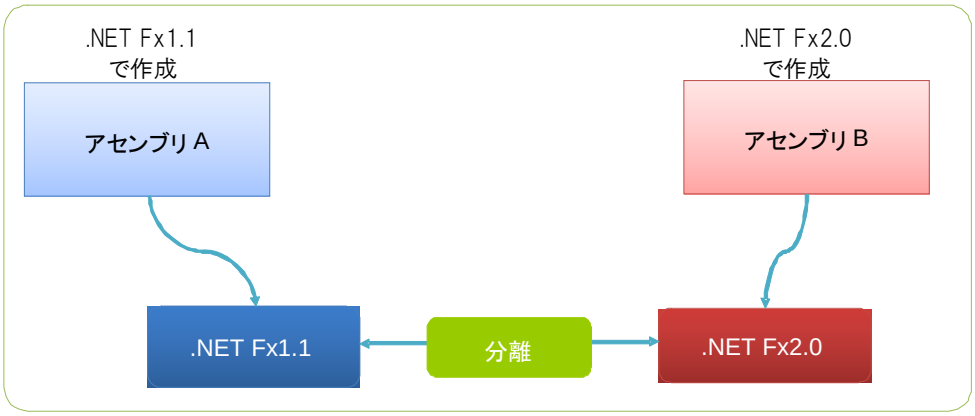

図 20　サイドバイサイド実行

また、複数バージョンの.NET アセンブリがそれぞれのバージョンのランタイムで実行される際に、お互いに影響を及ぼしません。

このサイドバイサイド実行の機構により、システム開発者は”DLL Hell”と呼ばれた実行環境の DLL が新しいバージョンに上書きされた際に起こるアプリケーションの非互換から解放されます。

バージョンリダイレクトとは

古いバージョンでビルドされたアセンブリを新しい.NET ランタイムで実行することも可能です。これは、メタデータに書き込まれたバージョンを、アプリケーション構成ファイルの指定などにより上書きし、新しいランタイムで実行する方法で、これをバージョンリダイレクトと呼びます。次図の例で.NET 1.1 アセンブリ C は、バージョンリダイレクトによって、.NET 2.0 CLR の下位互換機能により実行されています。

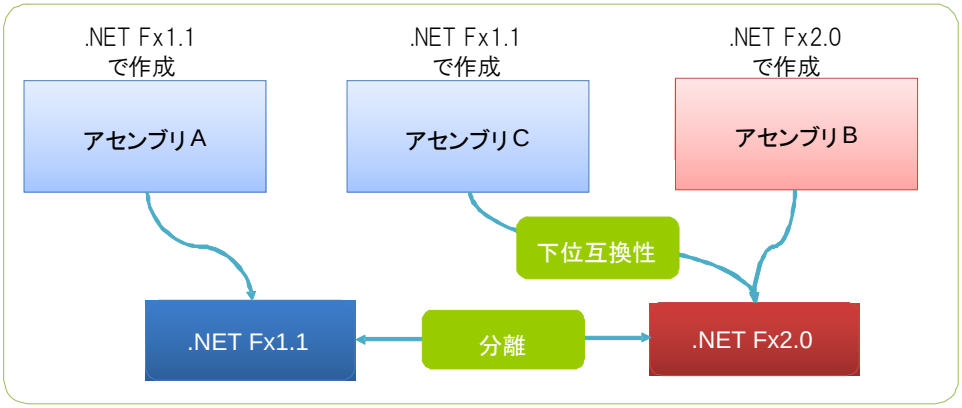

図 21　バージョンリダイレクト

PE ファイルの実行バージョンの決定

① アプリケーション構成ファイルの指定

次図のように、アプリケーション構成ファイル（アプリケーション名.exe.config）の <supportedRuntime> 要素[17]に実行する CLR のバージョンを指定します。

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
    <supportedRuntime version="v2.0.50727" />←このバージョンの CLR が呼び出される
    <supportedRuntime version="v1.1.4322"/>←上記が利用できない場合はこのバージョン
    </startup>
</configuration>
```

<supportedRuntime> 要素で指定された CLR のバージョンが利用できない場合、またはアプリケーション構成ファイルで使用する CLR のバージョンを指定しなかった場合、②のように実行バージョン決定が行われます。

② PE ファイルから CLR のバージョン情報の取得

EXE/DLL ファイルなどの PE（Portable Executable）ファイルと呼ばれる Windows の実行ファイルのヘッダー情報から CLR ベースのプログラムであることを示す証拠を探し、それが見つかると mscoree.dll をロードします。mscoree.dll は.NET Framework のインストール時に %Systemroot%\System32 に配置される DLL で、%Systemroot%Microsoft.NET\Framework\<バージョン>フォルダに存在する複数のバージョンのどの CLR を起動するかを指示します（バージョンリダイレクト）。

[17] http://msdn.microsoft.com/ja-jp/library/w4atty68.aspx

③アプリケーション構成ファイル及び PE ファイルで指定されたバージョン CLR が存在しない場合、エラーを出力して終了します。

.NET Framework のバージョンが異なるアセンブリ間の呼び出し

.NET Framework 2.0 で作成された EXE より呼び出された.NET Framework 1.x で作成されたアセンブリは、アプリケーション構成ファイル（アプリケーション名.exe.config）で指定しなかった場合、CLR2.0 上で動作します。アプリケーション構成ファイルはアプリケーションで 1 つしか持てないことと、CLR は 1 つのプロセスに 1 つしかロードできないため、最初に起動するプロセスがロードする CLR のバージョンを決定します。

なお、プロジェクトファイルの参照設定に DLL を追加する静的はアセンブリ呼び出し、及び Assembly.Load()メソッドからの動的なアセンブリの呼び出しの可否をまとめると次の表のようになります。

表 5　アセンブリを呼び出し可能なバージョン間の関係

	1.x→2.0	**1.x リビルド→2.0**	**2.0→1.x(CLR2.0)**
参照設定	×	○	○
動的呼び出し	×	○	○

静的呼び出し、動的呼び出しともに、.NET Framework 1.x アセンブリから.NET Framework 2.0 アセンブリの呼び出しはできません。

部分移行を検討する場合の制約

上に述べたとおり、.NET Framework 1.x アセンブリから.NET Framework 2.0 アセンブリの呼び出しはできないため、.NET Framework 1.x で開発されている画面と DLL がある場合、DLL だけを.NET Framework 2.0 アセンブリにすることはできません。DLL を改修して.NET Framework 2.0 アセンブリにするのであれば、呼び出し側の画面も.NET Framework 2.0 アセンブリにする必要があります。

ASP.NET アプリケーションの実行バージョンの決定

ここでは、ASP.NET のアプリケーションの実行バージョンの決定方法と、複数バージョンの ASP.NET のアプリケーションが混在する場合の動作についてまとめるとともに、注意点について述べます。

次図のように、ある仮想ディレクトリで動作する ASP.NET のバージョンは、仮想ディレクトリのプロパティ画面で設定した 1 つに固定されます。

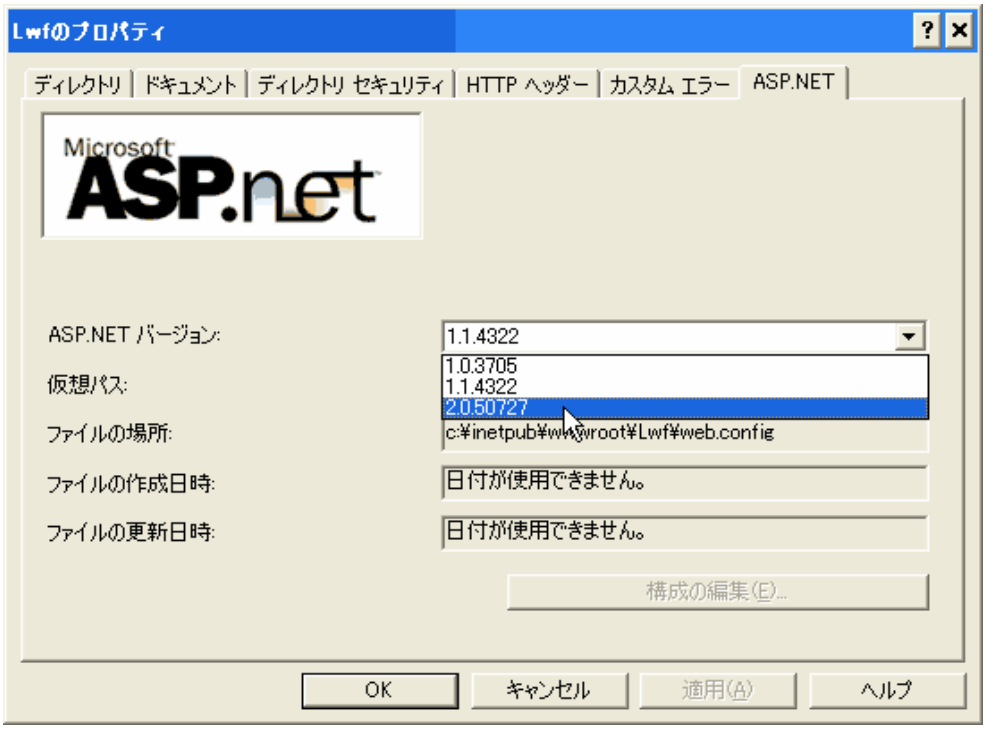

図 22　仮想ディレクトリの.NET バージョン指定

ASP.NET 1.1 で作成した aspx と ASP.NET 2.0 で作成した aspx を同一の仮想ディレクトリに配置した場合の動作

ASP.NET 1.1 Web アプリケーションと ASP.NET 2.0 Web アプリケーションを混在させた仮想ディレクトリの動作をまとめると次表のようになります。

表 6　aspx の呼び出しと複数バージョンの仮想ディレクトリ間の関係

内容	**ASP.NET 1.1 の仮想ディレクトリ**	**ASP.NET 2.0 の仮想ディレクトリ**
ASP.NET 1.1 の aspx から ASP.NET 2.0 の aspx 呼び出し	×	○
	ASP.NET 1.1 の aspx は表示されるが、ASP.NET 2.0 の aspx へは画面遷移不可。	呼び出し可能だが、ASP.NET 1.1 の aspx に遷移するとセッション ID が変わる。
ASP.NET 2.0 の aspx から ASP.NET 1.1 の aspx 呼び出し	×	○
	ASP.NET 2.0 の aspx は表示不可。	呼び出し可能だが、ASP.NET 1.1 の aspx に遷移するとセッション ID が変わる。

ASP.NET 1.1 の仮想ディレクトリ内では、ASP.NET 2.0 で作成した aspx のコードビハインドは動作できません。よって、仮想ディレクトリに ASP.NET 1.1 アプリケーションと ASP.NET 2.0 アプリケーションを混在させる場合、実行バージョンは 2.0 以降を指定する必要があります。

ASP.NET のバージョンが異なる仮想ディレクトリ間の遷移

ASP.NET のバージョンが異なる仮想ディレクトリ間で ASPX 画面を遷移した時の動作をまとめると次表のようになります。

表 7　aspx を呼び出し可能なバージョン間の関係

内容	結果
ASP.NET 1.1 の仮想ディレクトリにある aspx から ASP.NET 2.0 の仮想ディレクトリにある aspx 呼び出し	○ 呼び出し可能。
ASP.NET 2.0 の仮想ディレクトリにある aspx から ASP.NET 1.1 の仮想ディレクトリにある aspx 呼び出し	○ 呼び出し可能だが、ASP.NET 1.1 の aspx に遷移するとセッション ID が変わる。

ASP.NET のバージョンが異なる仮想ディレクトリ間で画面遷移は可能ですが、セッション ID が変わってしまうため、セッション変数の引き継ぎが困難になります。セッション変数の引継ぎのある画面遷移を生ずる仮想ディレクトリについては、実行バージョンを全て ASP.NET 2.0 以降に合わせることにより、移行作業の工数を減らすことができます。

バージョンが異なる.NET Framework 間のトランザクション連携

.NET Framework 1.x 及び.NET Framework 2.0 で作成されたアセンブリが異なった.NET Framework のバージョン間でアセンブリの呼び出しを行う場合の動作をまとめると次表のようになります。

表 8　トランザクション連携可能なバージョン間の関係

	1.x→2.0	1.x リビルド→2.0	2.0→1.x(CLR2.0)
Transaction	―	○	○

.NET Framework 2.0 で作成されたアセンブリから、.NET Framework 1.1 で作成されたアセンブリの連携を行っても、同一スコープのトランザクションとして処理されます。

バージョンが異なる.NET Framework 間のリモート呼び出し

表 9　リモート呼び出し可能なバージョン間の関係

クライアントのバージョン	XML Web サービス		.NET Remoting	
	1.1	2.0	1.1	2.0
.NET 1.1	○	○	○	○
.NET 2.0	○	○	○	○

クライアントでリモーティングオブジェクトの型を知る必要があるため、.NET 1.1 上で同じ I/F を持ったダミーを定義しなければならない。（.NET 1.1 では、.NET 2.0 のアセンブリを参照設定できないため。）

ただし、シリアライズに BinaryFormatter を利用している場合には、.NET Framework 2.0 アセンブリ側で OptionalField 属性の付与が必要です。

バージョン トレラントなシリアル化

http://msdn.microsoft.com/ja-jp/library/ms229752(VS.80).aspx

まとめ

.NET Framework のプログラミング言語レベルでは非互換も少なく、Visual Studio の支援機能もあり、移行作業は困難なものではありません。しかしながら、OS やミドルウェアなどの実行プラットフォームを含めたアーキテクチャまで考慮をした上で、移行計画を練ることが重要です。

本書を参考にして、安全かつ確実な移行計画を策定し、最新テクノロジーのメリットを活用して、システムに新たなビジネス価値を付加してください。

参考情報

- Visual Studio 2005 Service Pack 1 ダウンロード

  http://www.microsoft.com/japan/msdn/vstudio/downloads/sp/vs2005/sp1/

- Visual Studio 2005 Service Pack 1 に修正される問題の一覧

  http://support.microsoft.com/kb/918526/ja

- Visual Studio .NET 2002/2003 から Visual Studio 2005 への Web プロジェクト変換ステップバイステップ ガイド

  http://www.microsoft.com/japan/msdn/net/aspnet/webprojectsvs05.aspx

- Visual Studio Developer Center

  http://www.microsoft.com/japan/msdn/vstudio/

- .NET Framework Developer Center

  http://www.microsoft.com/japan/msdn/netframework/