Future Technology Days

Microsoft Visual Studio

“7”時代の次世代 Windows 開発

マイクロソフト株式会社
デベロッパー & プラットフォーム統括本部

Microsoft

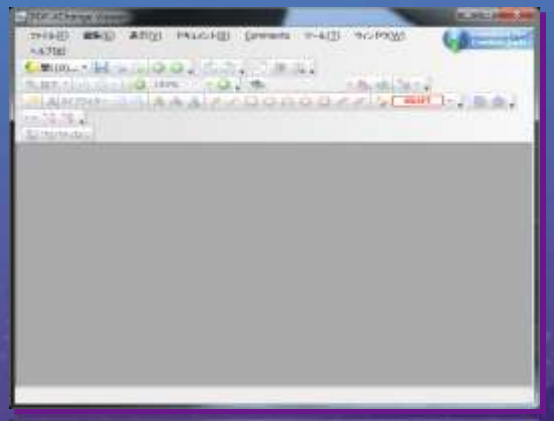
アジェンダ

- Widows アプリケーション開発
  - ユーザーインターフェース
    - Windows Forms
    - Windows Presentation Foundation
  - データアクセス技術
    - ADO.NET ファミリー
  - 配布技術
    - XCopy、ClickOnce
    - セキュリティシステム
  - .NET Framework 4 の新機能
- Windows 7 アプリケーション開発
  - タスクバー 統合
  - マルチタッチ
  - リボン インターフェース
  - センサー と ロケーション
- まとめ

ユーザー　インターフェース

Windows Forms

Windows デスクトップ
アプリケーションで
従来と互換性を持つ UX
を実現するテクノロジ



WPF
Windows Presentation Foundation

Windows デスクトップ
アプリケーションで
最高レベルの UX を
実現するテクノロジ

ASP.NET / Silverlight

Web アプリケーションで
UX を実現し、
クロスブラウザ環境で
UX を実現するテクノロジ

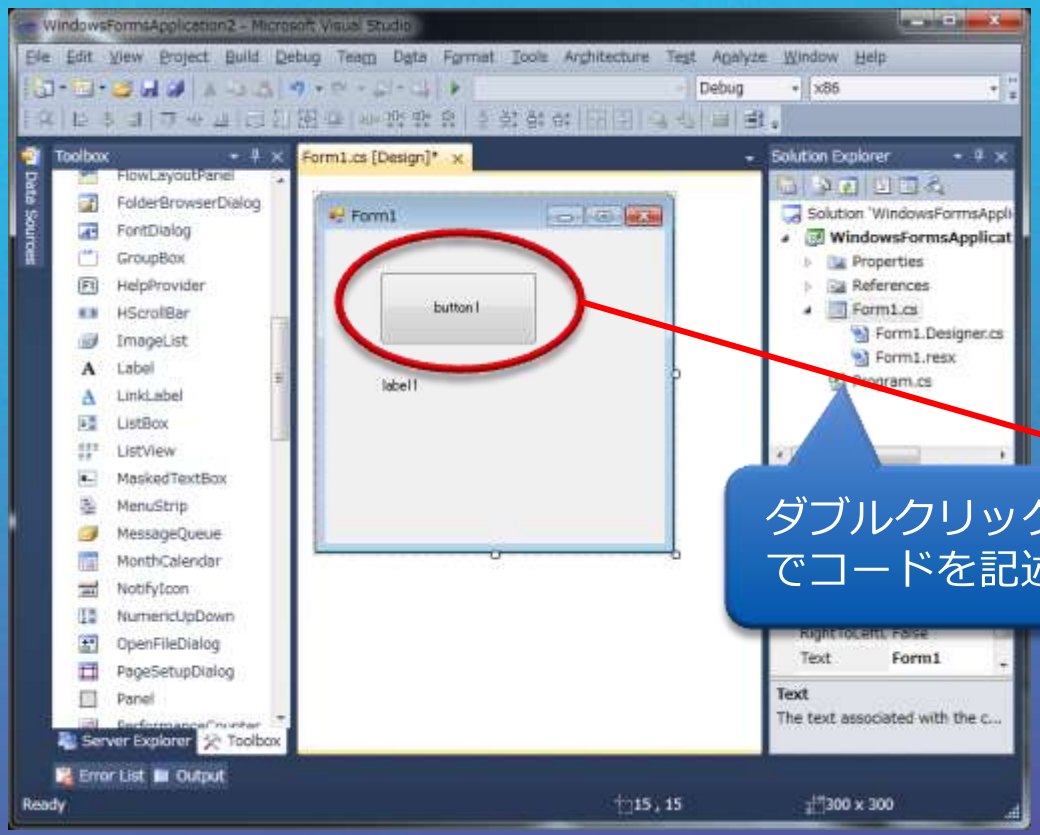
Microsoft Visual Studio

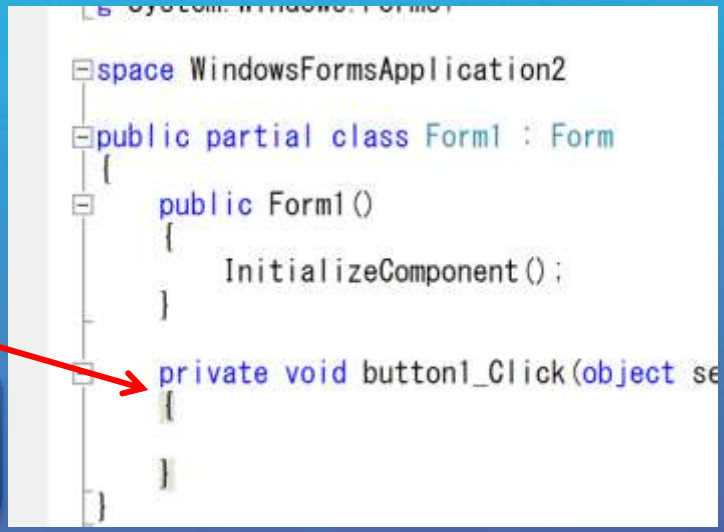
Windows Forms

Windows Forms とは

- 標準コントロールが豊富
- サードパーティ製のコントロールも豊富
- カスタム コントロールも作成できる
  - ベースは Win32 GDI 系の API
  - 矩形を実現するには、オーナードローというテクニックを必要とする
- 標準コントロール以外は、API などを使用する必要がある
  - メディア
  - グラフィクス
  - etc

開発スタイル





ダブルクリック
でコードを記述

- GUI デザイナでコントロールを配置
- コードは 2 種類に分離される
  - Form1.cs(vb) ファイル：記述したコード
  - Form1.Designer.cs(vb) ファイル：デザイナが生成したコード
- Visual Studio 2010 でも従来通りのサポート

Microsoft Visual Studio

チャート

New

- Windows Forms と ASP.NET で使用可能
  - .NET Framework 4 で標準サポート
    .NET Framework 3.5 の追加コンポーネント
  - 25 種類のチャート
  - 3D チャート
  - 3D のカスタマイズ

Microsoft Visual Studio

Demo

チャート

Microsoft® Visual Studio®

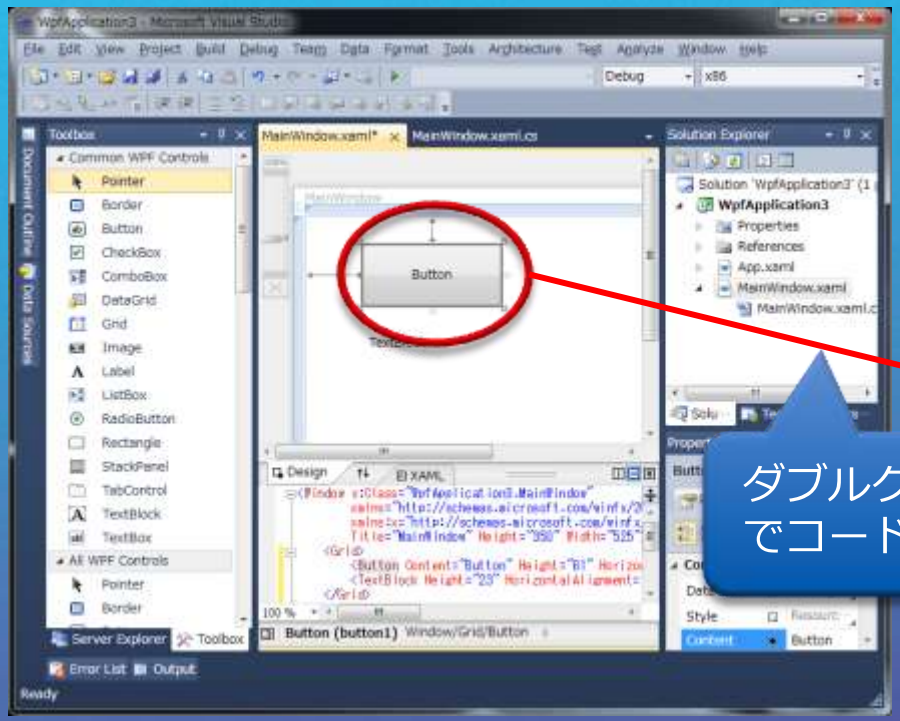
Windows Presentation Foundation

WPF とは

- プレゼンテーション テクノロジの統合
  - 高度な表現力
  - UIコンポーネント
  - メディア
  - 2D、3D、オーディオ、ビデオ、アニメーション
  - ドキュメント
  - クリアテキスト、フロードキュメント
  - Windows Forms と同等のコントロール
  - 統一的なプログラミング スタイル
  - Windows Forms との相互連携
- グラフィックス サポート
  - ベクタ ベースによる高品質な表示
  - GPU によるレンダリング
  - パフォーマンスの向上
- .NET Framework 3.0 以上標準に搭載
  - 3.0 -> 3.5 -> 4 で 三世代目

Microsoft Visual Studio

Windows Forms と同様の開発スタイル




```
namespace WpfApplication3
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender,
        {
            |
        }
    }
```

New

- GUI デザイナでコントロールを配置
- コードは 2 種類に分離される
  - Window.xaml ファイル：デザイナが生成した UI 定義の XML
  - Windows.xaml.cs(vb) ファイル：記述したコード
- Visual Studio 2010 では、ダブルクリックによるイベントハンドラの記述をサポート

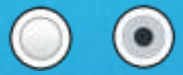
Microsoft Visual Studio

Demo

WPF デザイナ

Microsoft Visual Studio

コントロール



Push Me!

Search...

New

New

- 豊富なコントロールを提供
  - 自由度のあるコントロールモデル
  - カスタマイズが容易
    コンポジションモデルを採用

Microsoft Visual Studio

Demo

WPF コントロール

Microsoft Visual Studio

WPF データグリッド

機能強化

- カラムの自動生成
  - 規定カラムタイプ
    - テキストボックス
    - チェックボックス
    - コンボボックス
    - ハイパーリンク
- 各種の機能
  - 行ヘッダー
  - 列ヘッダー
  - 列幅・行高の変更
  - 列の並び替え
  - ソートとフィルタ
  - etc

Microsoft Visual Studio

サードパーティ製コントロール

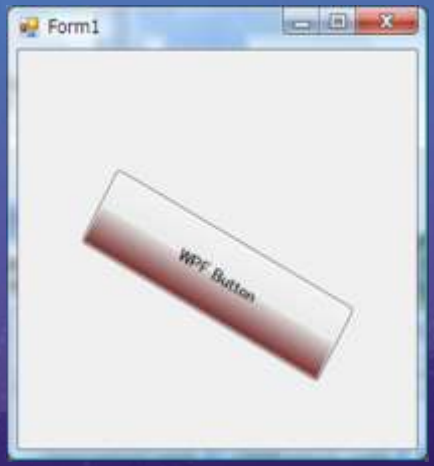
インフラジスティック様

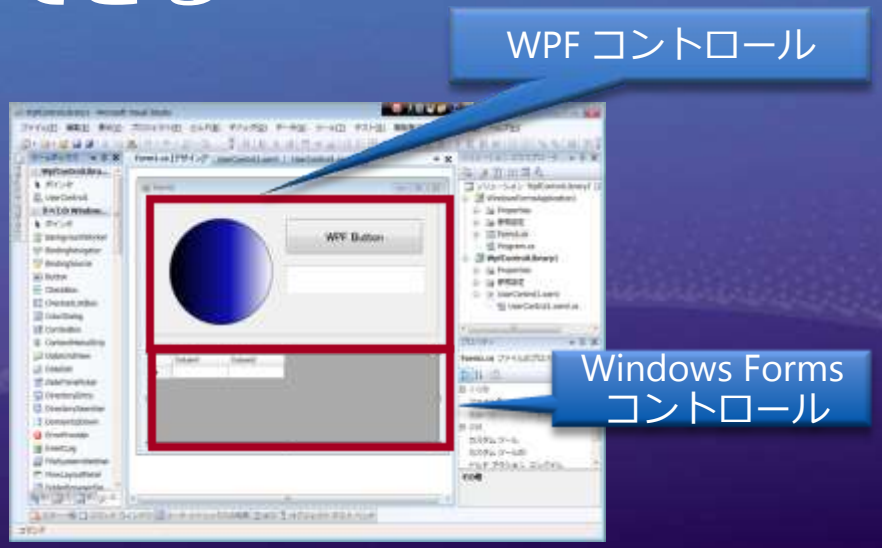
グレープシティ様

Microsoft Visual Studio

Windows Forms との相互連携

- 既存コードで WPF を使用可能
  - Windows フォームの中で WPF
  - WPF の中で Windows フォーム
- コンポーネント レベルの相互運用性
  - 既存の資産を有効活用できる





Microsoft Visual Studio

WPF の基礎知識

- UI 定義とロジックの分離
  - eXtensible Application Markup Language(XAML) で UI を定義
    - 宣言型プログラミング
  - コードファイルでロジックを定義
- 開発ツール
  - .NET Framework 3.5 までは Visual Studio と Expression Blend の組み合わせ
  - Visual Studio 2010 からは、アニメーション以外は Expression Blend が不要

New

Microsoft Visual Studio

ユーザーインターフェースを選択するために

Microsoft Visual Studio

ユーザー インターフェース 技術の位置づけ

Microsoft Visual Studio

UI 以外の技術
データアクセス
配布技術など

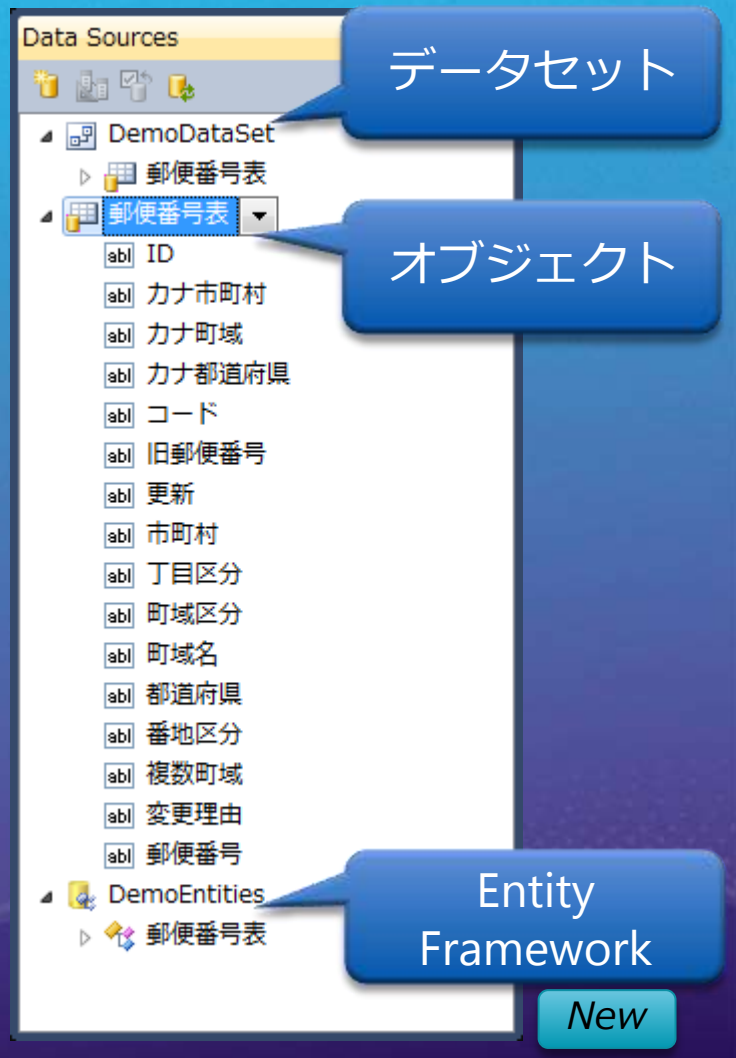
Microsoft Visual Studio

ADO.NET ファミリー

- DataSet は引き続き提供
- ADO.NET Entity Framework を機能強化
  - .NET Framework 3.5 SP1 から提供
  - .NET Framework 4 で標準提供
    POCO(Plain Old CLR Object) サポート
    変更トラッキングのサポート
- RDBMS 向けのマネージ プロバイダ
  - SQL Server 向けは、SQL Server 2008 R2 向けの機能強化
  - Oracle 向けは、残念ながら廃止
    Oracle 様を含めてサードパーティ製が充実しているため

データソースの機能強化

- Entity Framework をデータソースへ追加
- データバインドの機能強化
  - WPF では、Entity Framework のドラッグ & ドロップ をサポート
  - Windows Forms は従来通り
    - Object データソースを使用する



Microsoft Visual Studio

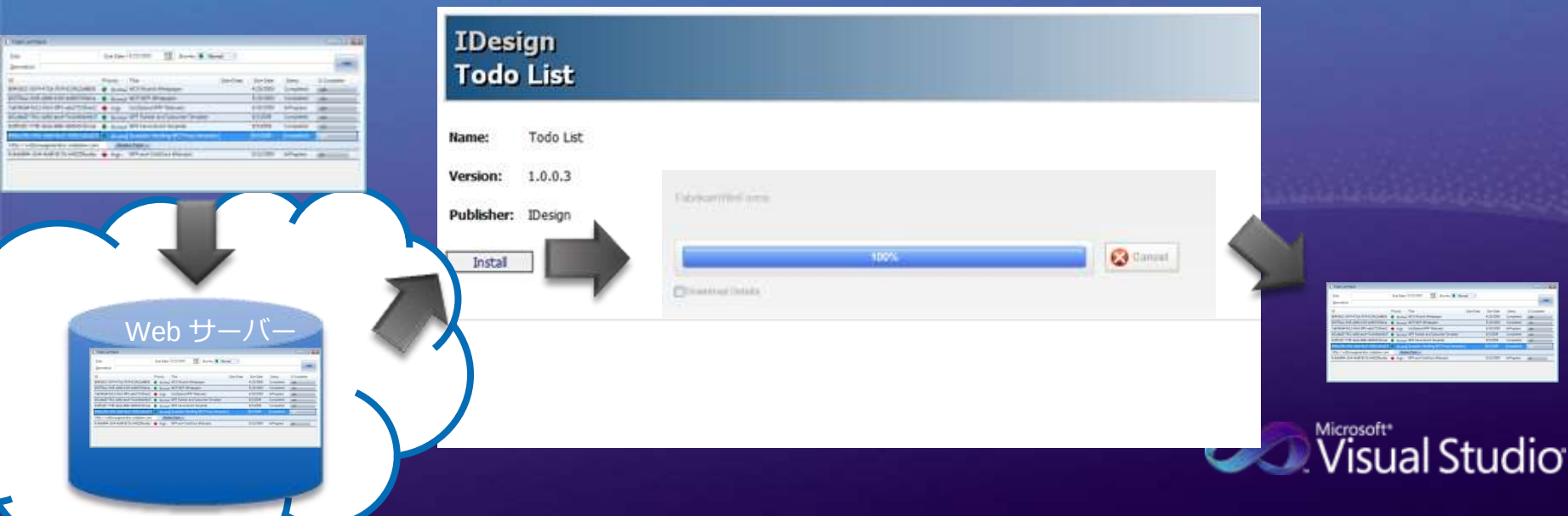
配布方法

- XCopy 配布
- セットアップ プロジェクト支援
  - Visual Stduio インストーラー プロジェクト New
  - CAB プロジェクト
- ClickOnce
- ノータッチ デプロイメントは、残念ながら廃止
  - .NET Framework 2.0 以降の利用率が少ないため
- ブラウザ アプリケーション
  - XBAP (WPFのみ)
    - 完全信頼モードを追加 New
      Silverlight 4 の ブラウザ外アプリケーション(OOB)と同じ

Microsoft Visual Studio

ClickOnce とは

- .NET Framework 2.0 から提供
- .NET Framework 3.5 で機能強化
  - 配布場所の制限を緩和
  - マニフェスト生成ツールと編集ツールの提供
- .NET Framework 3.5SP1 での機能強化
  - 署名なしでの配布サポート
  - バックグラウンド アップデートのサポート
  - コマンドライン 引数サポート
  - ファイルの拡張子関連付けのサポート



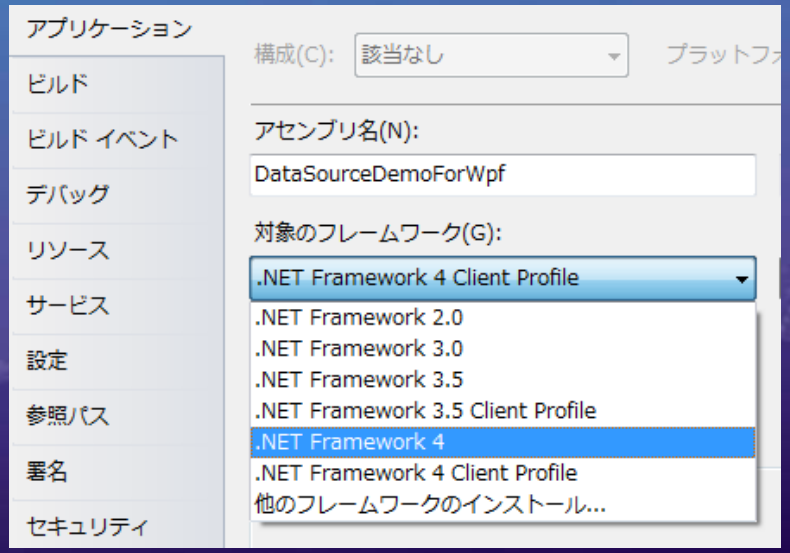
Microsoft Visual Studio

クライアント プロファイル

- クライアント アプリケーション配布時のランタイム サイズの削減
  - Windows Forms、 WPF アプリケーションの実行環境
  - .NET Framework 3.5 SP1 より提供
- .NET Framework 4 で機能強化
  - Windows Forms、WPF、VSTO へ対応
  - 28 MB にサイズを削減(78%)
    フルサイズでは 48 MB

VB

C#



Microsoft Visual Studio

セキュリティ モデルの変更

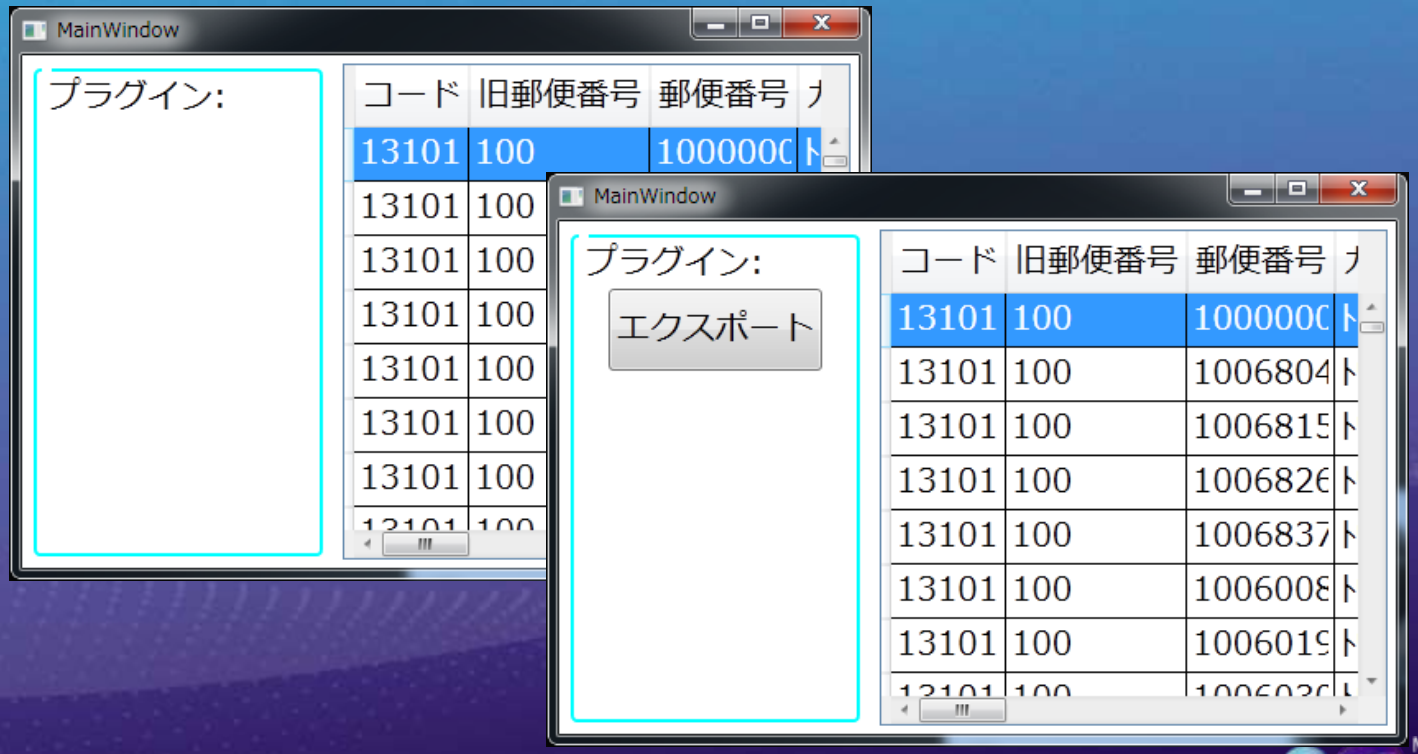
- レベル 2 のセキュリティ透過性モデル (Security Transparency Level 2) New
  - CAS に代わる新たなセキュリティ モデル
    - CAS⇒ポリシー設定が複雑、ネイティブコードが常に完全信頼
    - CAS は引き続きサポートされるが、デフォルトでは無効
      - <NetFx40_LegacySecurityPolicy enabled="ture" />
      - ポリシーとエビデンスは無効だか、アクセス許可は有効
    - 信頼性の低い環境で実行しても安全なコードと、そうでないコードを分離
  - 信頼レベル
    - 完全信頼 (Full Trust)：すべてのコードの実行権限を持つ
    - 部分信頼 (Partial Trust)：コードの実行権限は制限される
  - 信頼レベルの決定
    - 自己ホスト⇒完全信頼
    - サンドボックス環境下⇒ホストから与えられる信頼レベル
- Silverlight で採用したセキュリティ モデル

.NET Framework 4 新機能

Microsoft Visual Studio

プラグイン アプリケーション

- 機能を後から追加したり、削除可能なアプリケーション
- コンポーネント部品の動的な追加や削除



Microsoft Visual Studio

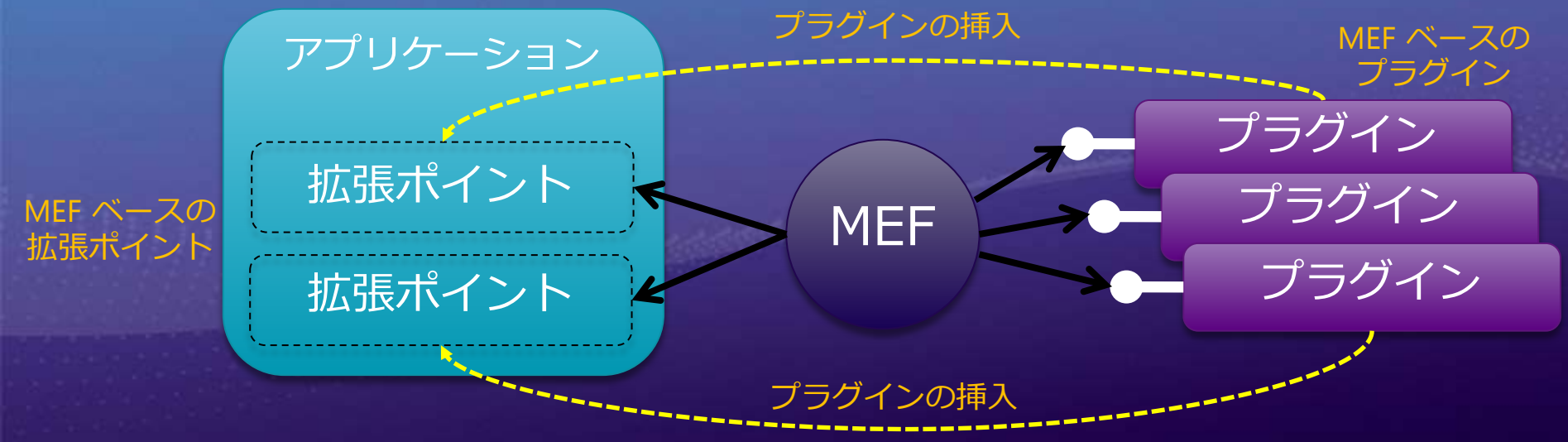
Demo

プラグイン
遅延バインディング

Microsoft Visual Studio

Managed Extensibility Framework

- アプリケーションに拡張性を追加する仕組
- IoC/DI コンテナであり、プラグイン作成基盤
  - 実行時に動的に機能(拡張可能なプラグイン)を追加
    - アプリケーションとプラグインの結合を疎にする
    - Visual Stduio 2010 のプラグイン基盤として採用
  - System.ComponentModel.Composition



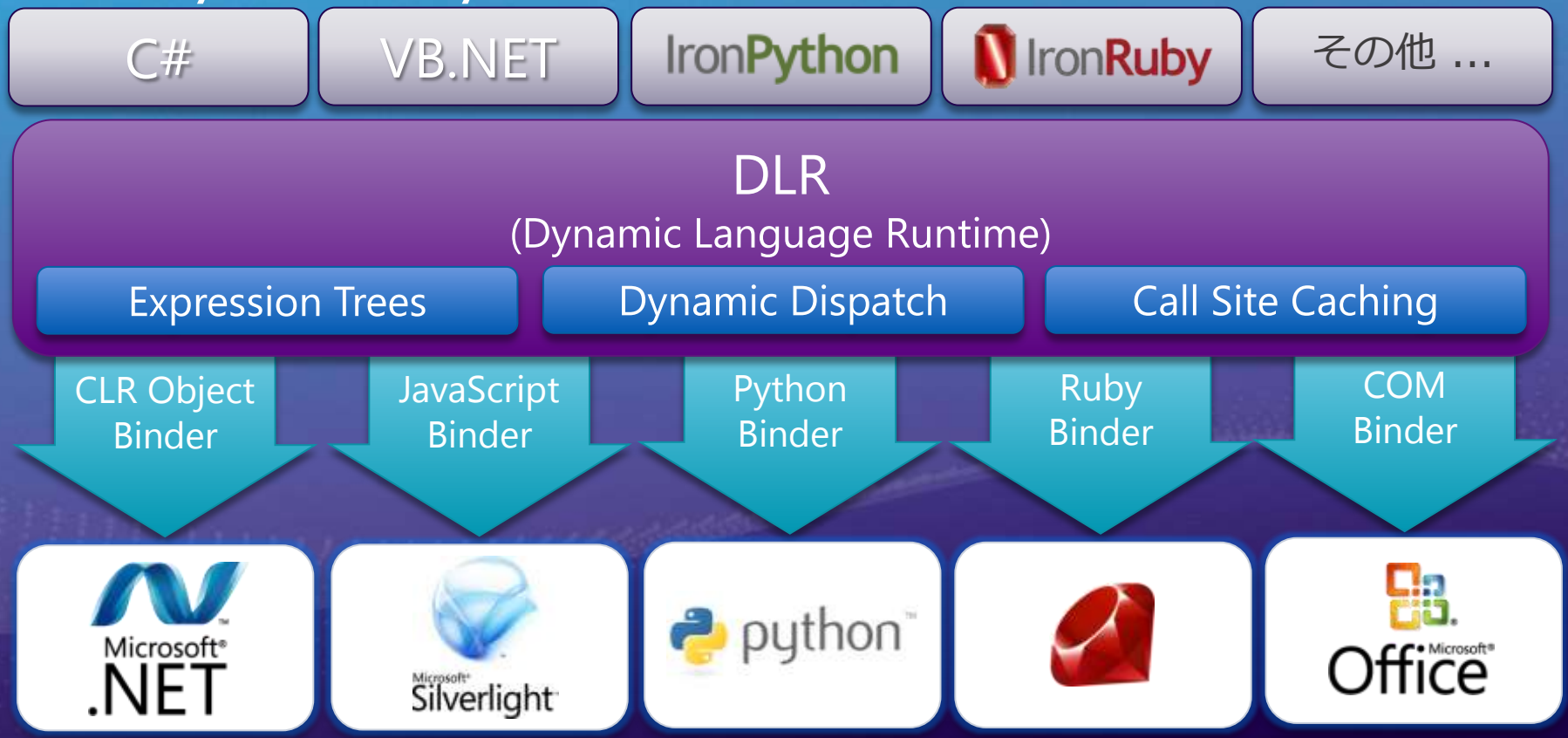
遅延バインディング

- 新しい COM インタロップ
- C# でも遅延バインディングをサポート
  - VB ではお馴染みの機能
- VB でも遅延バインディングの強化

```
dynamic excel = CreateObject("Excel.Application");
excel.Visible = true;
var books = excel.Workbooks;
var book = books.Add();
var sheet = book.Sheets("Sheet1");
// データを設定


private dynamic CreateObject(string progID)
{    var type = Type.GetTypeFromProgID(progID);
     return Activator.CreateInstance(type);}
```

動的言語ランタイム

- DLR (Dynamic Language Runtime)
- 動的言語と静的言語 (C# や VB) が相互に連携可能
  - System.Dynamic

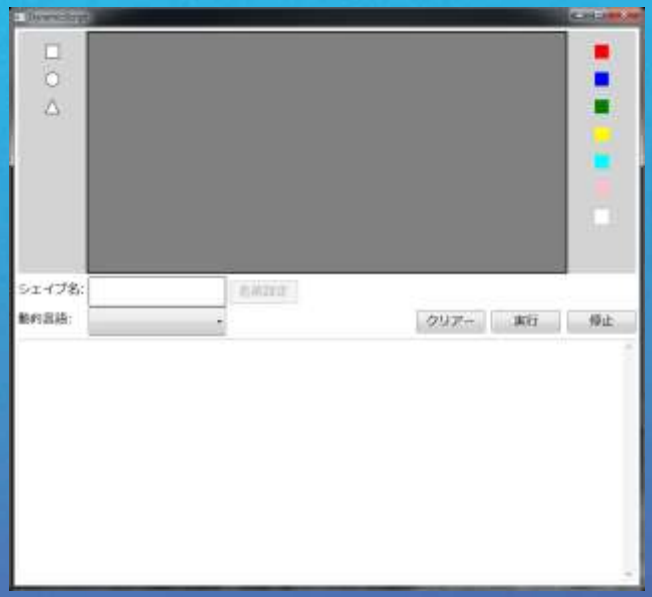


Demo

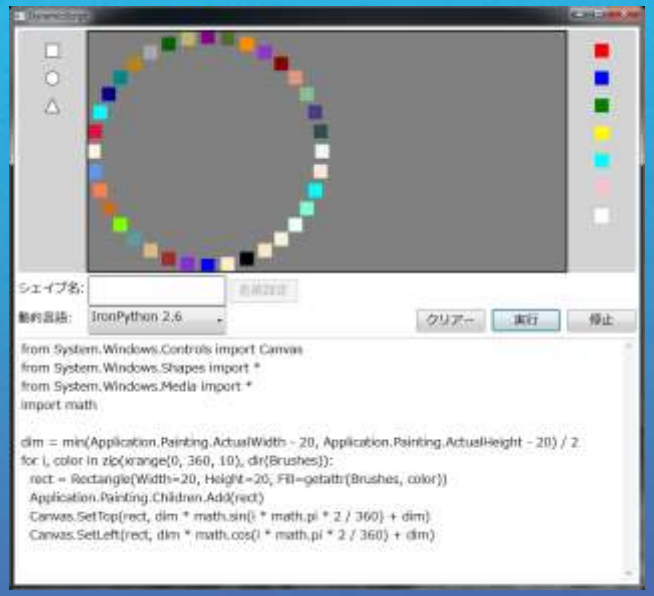
新しいタイプの
アプリケーション

Microsoft Visual Studio

スクリプトとの連携

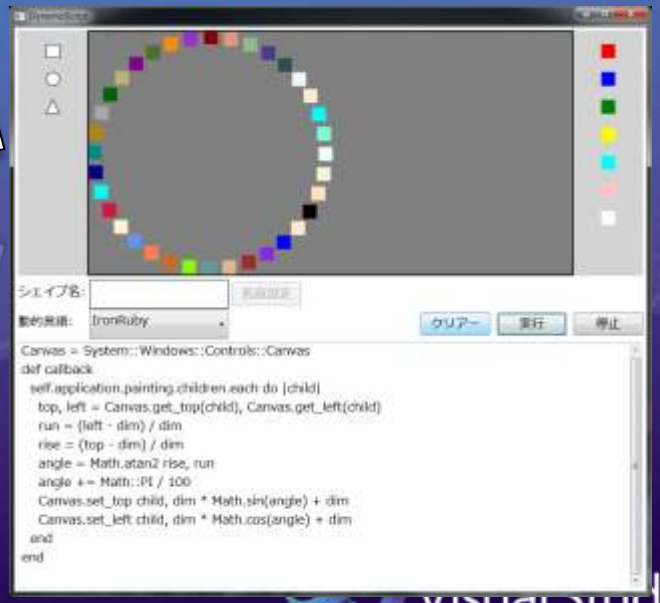


②描画は Python



③callbackは Ruby

④callback を呼び出すのは C#



Windows 7 アプリケーション開発

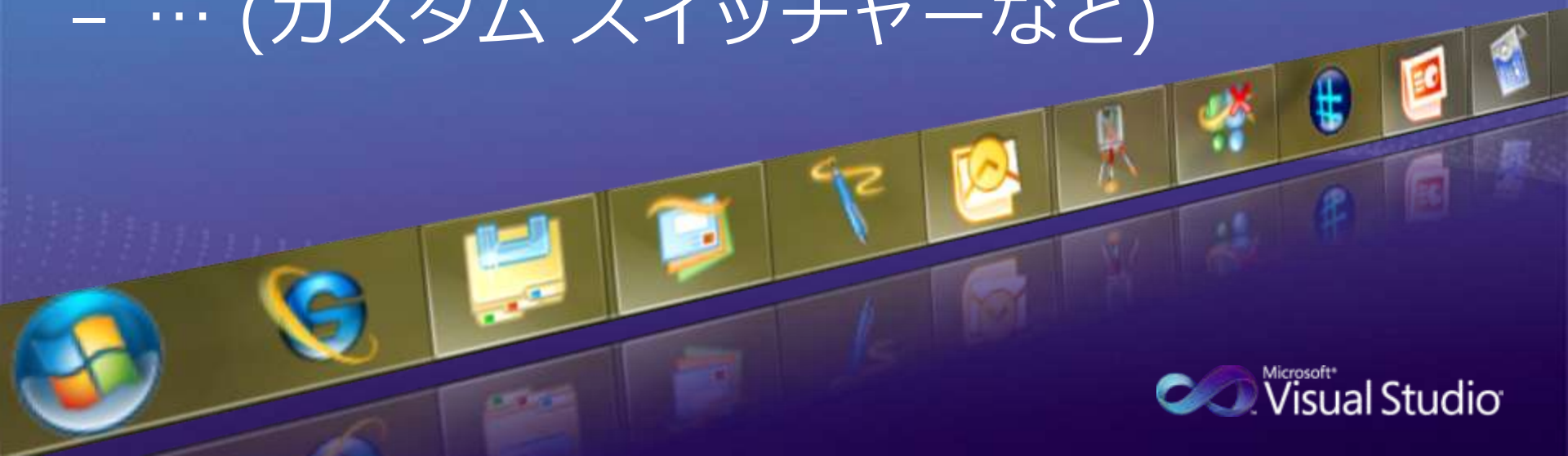
Microsoft Visual Studio

Windows 7 機能とは

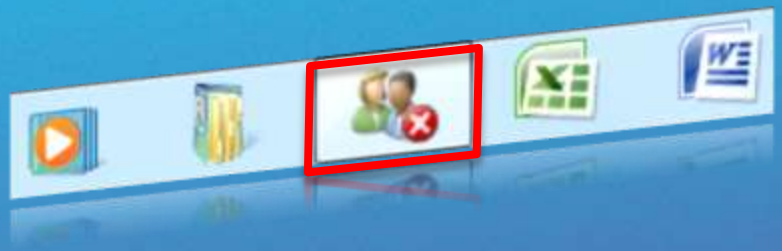
- 新しいタスクバーを提供
- リボン インタフェースを提供
- ライブラリを新規提供
- センサーとロケーション API を提供
- フォントダイアログの改善(表示、非表示が可能)
- ユーザー アカウント制御の改善
- 特権の分離 (UIPC)
- Hight DPI の提供(リブート無しに変更可能)
- トリガー スタート サービス(イベントによる Windows サービスの起動)
- …

タスクバー

- オーバーレイ アイコン
- 進行状況バー
- サムネイル クリップ
- サムネイル ツールバー
- ジャンプリスト
- … (カスタム スイッチャーなど)

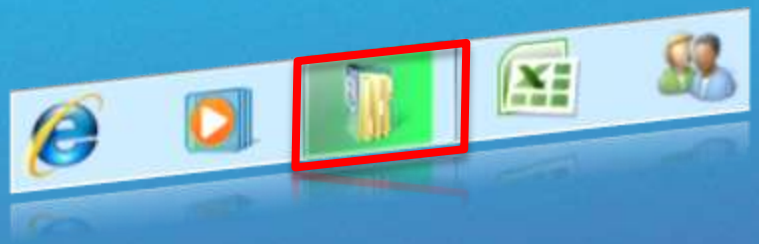


オーバーレイ アイコン



- アプリケーションに関する情報をグラフィカルに表示
- プログラム アイコンに重ねて表示される

進行状況バー



- 進行状況を表示
- プログラム アイコンの背景に表示される

サムネイル クリップ

- サムネイルに表示するプレビュー
- プレビューを任意の範囲に設定する

WPF による実装

```
<Window.TaskbarItemInfo>
   <!- オーバーレイアイコン、進行状況バー、サムネイル クリップ
       を TaskbarItemInfo で設定できる/-->
   <TaskbarItemInfo x:Name="_taskbarItem"
      Overlay="{Binding ElementName=_overlaySelection,
                                Path=SelectedItem.Source}"
      ProgressState=
            "{Binding ElementName=_progressState, Path=SelectedValue}"
      ProgressValue="{Binding ElementName=_progressSlider, Path=Value}"
      ThumbnailClipMargin="{Binding
                        RelativeSource={RelativeSource FindAncestor,
                        AncestorType={x:Type TabControl}},
                                      Path=BorderThickness}">
   </TaskbarItemInfo>
</Window.TaskbarItemInfo>
```

サムネイル ツールバー

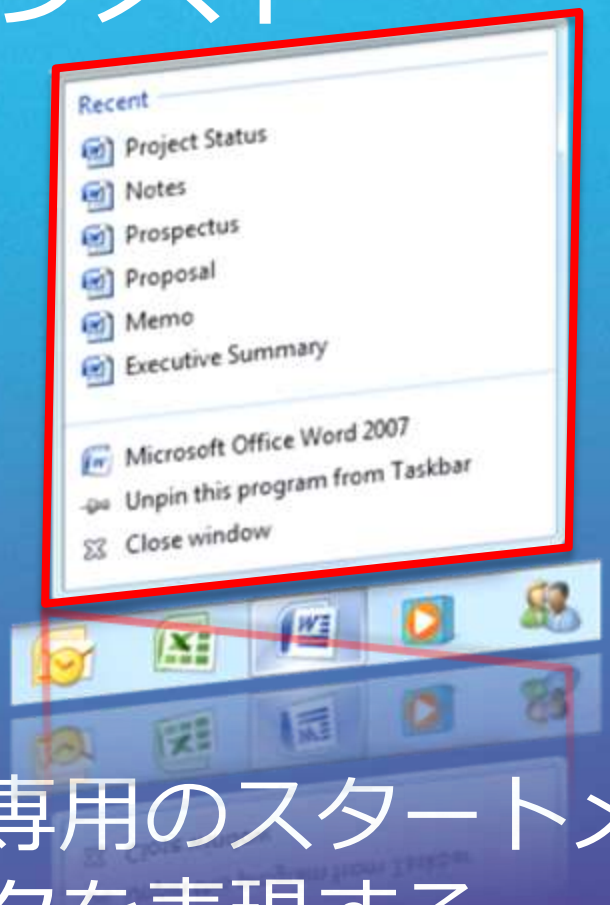
- サムネイルから操作可能なボタン集 (ツールバー)
- アプリケーションの操作コマンドとして機能する
- 最大 7 つのボタンが利用可能

WPF による実装

```
<Window.TaskbarItemInfo>
   <!- サムネイル ツールバーを ThumbButtonInfo で設定できる /-->
   <TaskbarItemInfo x:Name="_taskbarItem">
       <ThumbButtonInfo
                    CommandParameter="buttonFirst"
                    ImageSource="Images/ToolbarButtons/first.ico"
                    Visibility="Hidden"
                    IsInteractive="True"
                    DismissWhenClicked="False"
                    Description="先頭へ移動します"
                    />
       <ThumbButtonInfo  …  />
       .........
   </TaskbarItemInfo>
</Window.TaskbarItemInfo>
```

Microsoft Visual Studio

ジャンプリスト



- プログラム専用のスタートメニュー
- 目的やタスクを表現する
- カスタマイズが可能
- ドラッグや右クリックでアクセス可能

Microsoft Visual Studio

ジャンプリスト

対象
(Destinations)

タスク
(操作)

FrequentCategory
(良く使うもの)

RecentCategory
(最近使ったもの)

カスタム カテゴリ
(タスク)

アプリケーション
タスク

タスク バー共通

- 分類するのはカテゴリ
  - 用意されたカテゴリ(Frequent、Recent、タスク)
  - カスタム カテゴリは、ユーザーで変更可能になる

Microsoft Visual Studio

WPF による実装

```
<Application x:Class="Windows7FeatureIntegration.App" … />
   <!- JumpList で「良く使うもの」「最近使ったもの」の表示を設定 /-->
   <JumpList ShowFrequentCategory="True"
             ShowRecentCategory= “False” >
        <!- JumpTask でアプリケーション専用タスクを設定 /-->
        <JumpTask Title="関連付けを登録します"
                  Description="レジストリへアプリの関連付けを行います"
                  IconResourcePath= “フルパス” IconResourceIndex="0"
                  Arguments="/register" />
        <!- カスタムカテゴリでユーザーが変更できるタスクを設定 /-->
        <JumpTask Title="引数１ “
                  Description="/引数1を使ってプロセスを起動します “
                  CustomCategory= “タスクカテゴリ”
                  Arguments="/引数1" />
        <!- JumpPath で関連付けられたファイルを設定 /-->
        <JumpPath Path="a.txt"
                  CustomCategory="パスカテゴリ" />
        ……
</Application>
```

マルチタッチ

Microsoft Visual Studio

タッチ操作

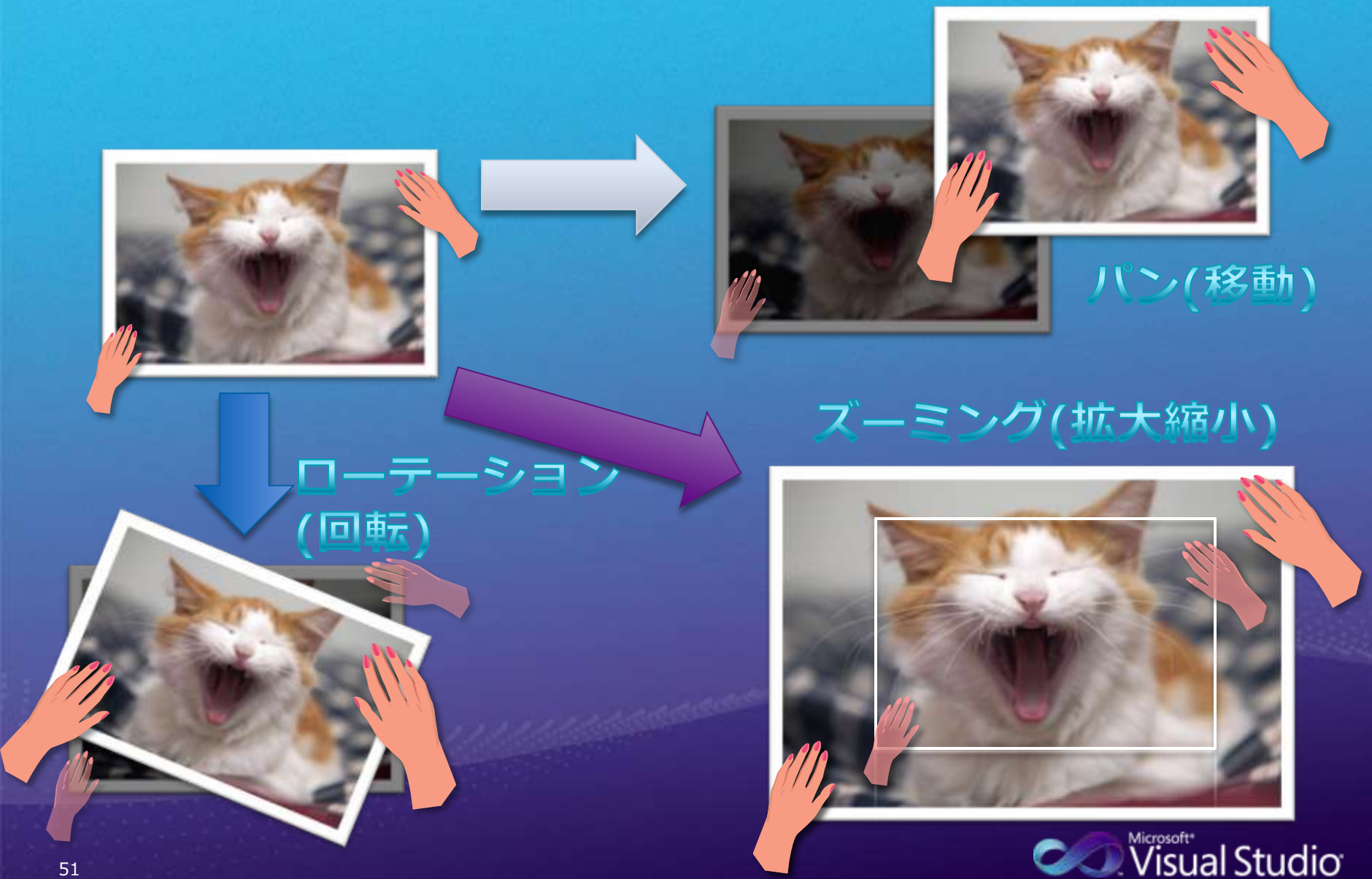
- **タッチ**であること
  - 指によるタッチ操作
  - マウスと同等の動作
    - ドラッグ：スクロール、移動

- **マルチ**であること
  - 複数ポイントによるタッチ操作
  - 2 点を使った動作：ジェスチャー
    - 移動、ズーム、回転

WPF による実装：タッチ

```
<Canvas x:Name="canvas" Background="LightYellow"

            TouchDown="Canvas_TouchDown"
            TouchMove="Canvas_TouchMove"
            TouchUp="Canvas_TouchUp">

</Canvas>

<!-
イベントハンドラを XAML に記述してタッチの動作を記述します
-->
```

マルチタッチ操作：ジェスチャー



Microsoft Visual Studio

WPF による実装：ジェスチャー

```
<Canvas x:Name="canvas"
        ManipulationStarting="canvas_ManipulationStarting"
        ManipulationDelta="canvas_ManipulationDelta"  >

    <Image IsManipulationEnabled="True" x:Name="image1 “
                                Width="200" Source="Guitar.jpg">
            <Image.RenderTransform>
                <MatrixTransform />
            </Image.RenderTransform>
    </Image>

</Canvas>
<!-
  イベントハンドラを XAML に記述してジェスチャー動作を記述します
  ジェスチャー操作の対象に「IsManipulationEnabled」を設定します
-->
```

Microsoft Visual Studio

リボン インターフェース

- Office System 2007 から提供したインターフェース
- Windows 7 から OS として提供
- WPF では、 Ribbon コントロールライブラリを追加することで利用可能
  - http://msdn.microsoft.com/officeui/

Microsoft Visual Studio

WPF による実装

```
<r:Ribbon Title="自分で定義したリボン" x:Name="myRibbon" >
  <!-- リボン アプリケーション メニュー -->
    <r:Ribbon.ApplicationMenu >
      <r:RibbonApplicationMenu>
        <r:RibbonApplicationMenu.Command>
          <r:RibbonCommand
             Executed="CloseCommand_Executed"
             LabelTitle="定義したコマンド"
             LabelDescription="定義したコマンドを実行します"
             SmallImageSource="Images/RibbonIcons/Coins.png"
             LargeImageSource="Images/RibbonIcons/Coins.png"
          </r:RibbonCommand>
        </r:RibbonApplicationMenu.Command>
        <r:RibbonApplicationMenuItem>
          <r:RibbonApplicationMenuItem.Command>
            <r:RibbonCommand
              LabelTitle="閉じる"
              LabelDescription="閉じます"
              Executed="CloseCommand_Executed"/>
```

センサー & ロケーション

- PC を取り巻く環境と、位置情報をアプリケーションで活用
- センサードライバで活用できる
- 小さなアイディアで**無限の可能性**へ

Microsoft Visual Studio

.NET 4 によるロケーション

```
// 標準でロケーションを取得するクラスを提供
// (ネットワーク機器から位置情報を取得するプロバイダ)
GeoCoordinateWatcher watcher = new GeoCoordinateWatcher();


// 位置情報を取得開始
atcher.TryStart(false,   TimeSpan.FromMilliseconds(1000));


// 位置情報の取得(イベントハンドラでも取得できる)
GeoCoordinate coord = watcher.Position.Location;
if (coord.IsUnknown != true)
{
    Latitude.Text = coord.Latitude.ToString();
    Longitude.Text = coord.Longitude.ToString();
}
```

まとめ

- Windows アプリケーション
  - Windows ユーザーインターフェース
    - Windows Forms は成熟している
    - Windows Presentation Foundation は業務アプリケーションに適用可能
  - データアクセス技術
    - ADO.NET に Entity Framework を追加
    - データバインドで活用できる
  - 配布
    - ClickOnce の機能強化
    - クライアント プロファイルの強化
    - セキュリティモデルの変更
  - .NET Framework 4 の新機能
    - 新しいアプリケーション構造へ対応できる
- Windows 7 アプリケーション
  - WPF なら簡単に開発ができる

Appendix

ツールキット シリーズ

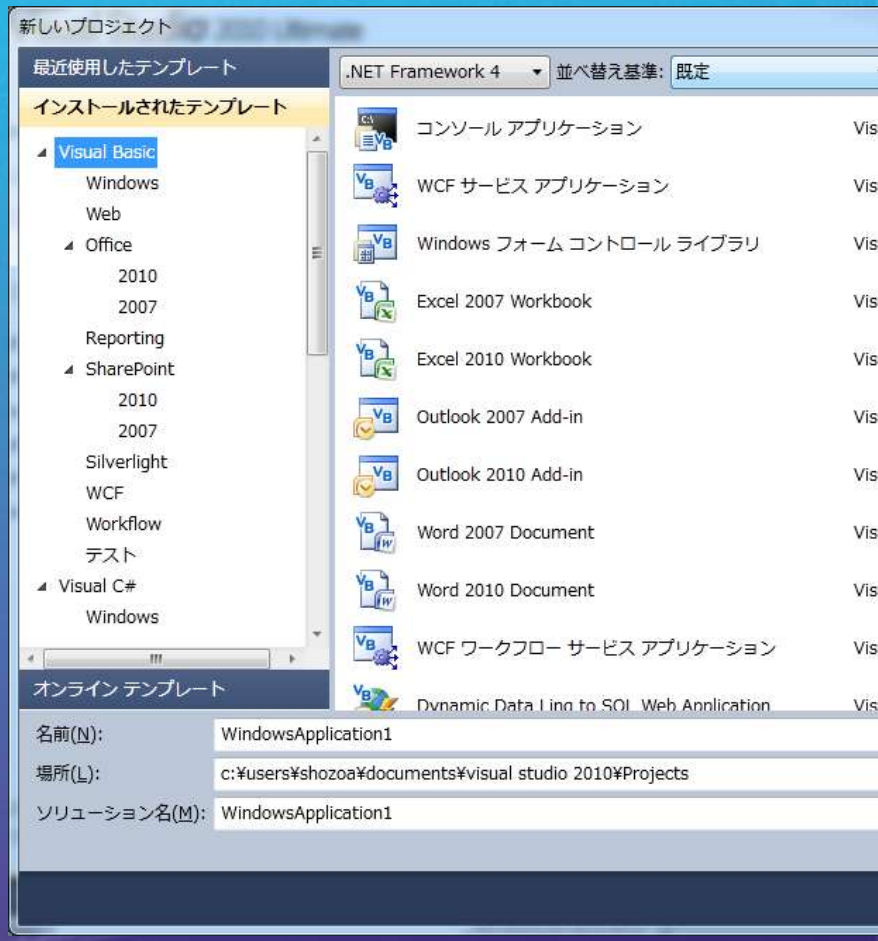
- 標準コントロールの機能不足を補完
  - ソースコード付きで公開
- WPF Toolkit (チャートなど)
  - http://wpf.codeplex.com/
- リボン コントロール ライブラリ
  - http://msdn.microsoft.com/officeui/
- リボン コントロール スイート
  - http://fluent.codeplex.com/
- Silverlight Toolkit (チャートなど)
  - http://silverlight.codeplex.com/
- Windows API Code Pack (Windows Vista / 7 機能)
  - http://code.msdn.microsoft.com/WindowsAPICodePack

Windows 7 タスクバー

- アプリケーション ユーザー モデル ID でグルーピング
  - SetCurrentProcessExplicitAppUserModelID API
  - シェル アイテムで保持
- ジャンプリストのカスタマイズには、設定が必須
  - 拡張子の関連付けと AppUserModelID のレジストリ登録
  - JumpPath を XAML に記述する場合は、Application インスタンスを作成する前に設定する
- WPF 4 で未サポートの Windows 7 機能
  - Windows API Code Pack を提供
    - Windows Forms、.NET Framework 3.5 もサポート

開発できるプロジェクト

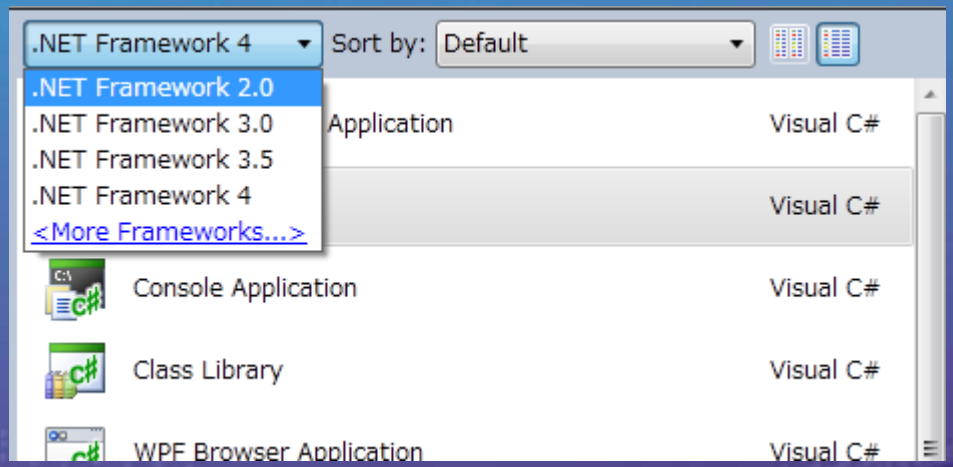
- Windows アプリケーション
  - Windows Presentation Foundation
  - Windows Forms
  - コンソール アプリケーション
  - ATL/MFC/Win32
- Office アプリケーション
  - Office 2007 System
  - Office 2010 System *New*
- Web アプリケーション
  - ASP.NET Web アプリケーション
  - ASP.NET Web サービス
  - ASP.NET MVC2 アプリケーション *New*
- Silverlight アプリケーション
  - Silverlight 3 *New*
  - Silverlight 4
- サービス アプリケーション
  - WCF サービス
  - WF アプリケーション
  - Sharpoint 2007/2010 *New*
  - クラウド アプリケーション
- その他
  - UML モデリング *New*
  - SQL Server データベース
  - …

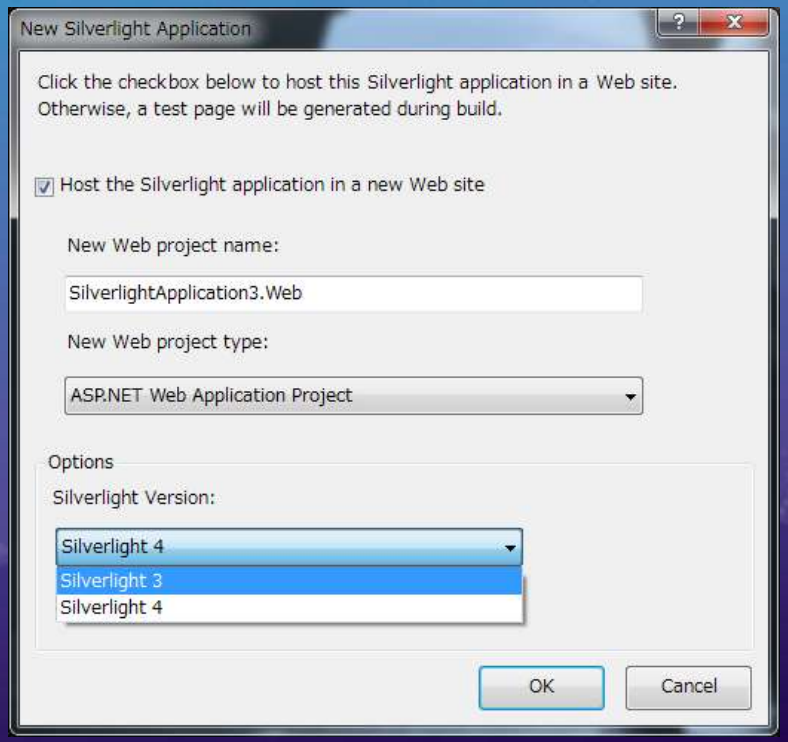


Microsoft Visual Studio

マルチターゲットの進化

- マルチターゲット
  - .NET Framework 2.0/3.0/3.5/4.0 作成後も変更できる *New*
  - インテリセンスの機能強化
  - Silverlight 3.0/4.0 *New*

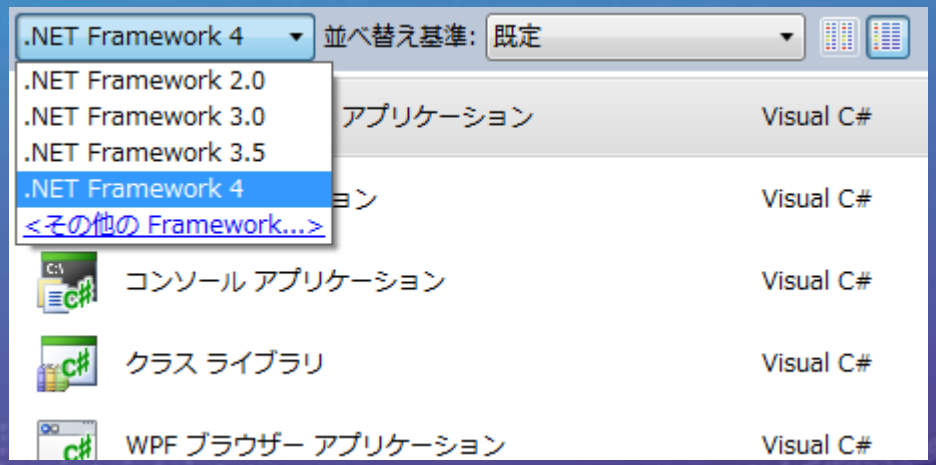


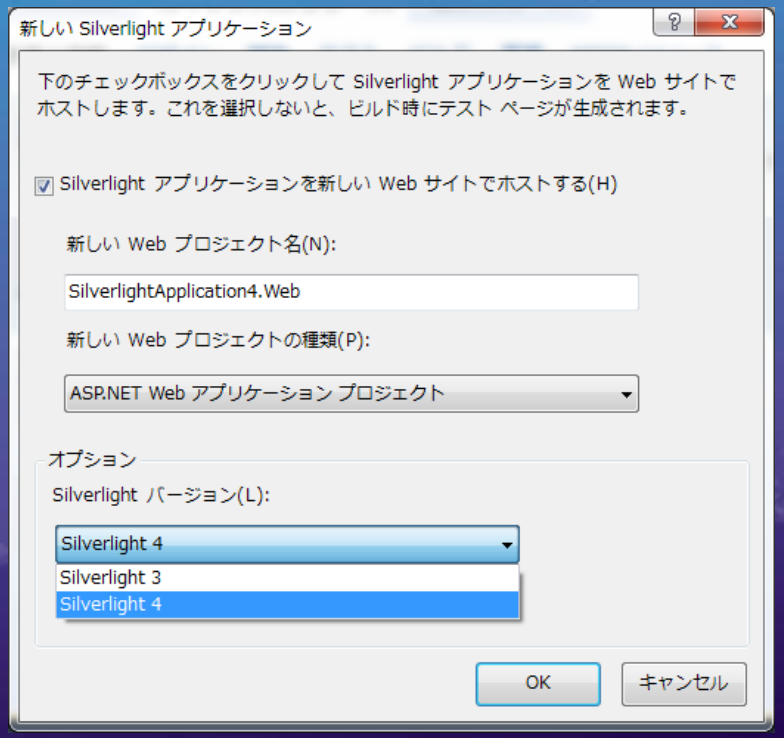


Microsoft Visual Studio

マルチターゲットの進化

- マルチターゲット
  - .NET Framework 2.0/3.0/3.5/4.0 作成後も変更できる *New*
  - インテリセンスの機能強化
  - Silverlight 3.0/4.0 *New*





Microsoft Visual Studio

IDE の機能強化

- マルチモニター サポート
- インテリセンスの強化
- コード ハイライト
- スタブ 生成機能 (メソッド、クラス)
  - TDD スタイルのサポート
- コード検索機能の強化
- デバッガの強化
  - インテリ トレース (x86)
  - マルチスレッド
  - ブレークポイント、データチップ
- …

.NET Framework 4

セキュリティモデルの変更

- 部分信頼レベル下での 3 つのセキュリティ区分
  - .NET クラス ライブラリに割り当てられている区分
    - 部分信頼レベル下では、危険なコードを呼び出すことは不可
    - 属性未指定＝安全なコード

Microsoft Visual Studio

WPF 4 の機能強化

- データバインディングの機能強化
- Visual State Manager
  - Silverlight の機能をサポート
- キャッシュモード プロパティ
  - 再描画の高速化
- レイアウト ルーティング
  - 描画をピクセル単位に揃えるかどうか
- テキスト レンダリング
  - TextFormattingMode、TextRenderingMode プロパティ
- 選択 とキャレット ブラシ
- カスタム ディクショナリ サポート
- ピクセル シェーダー 3.0 サポート
- コモンダイアログ (独自仕様から汎用化)
- ClickOnce の完全信頼モード (XBAP)
- XBAP-HTML 相互運用 (BrowserInteropHost)

便利なショートカット –抜粋-

分類	説明	C#	VB	備考
コメントの操作	コメントアウト	CTRL+K, C	←	
	コメントの解除	CTRL+K, U	←	
アウトライン	全ての展開と折り畳み	CTRL+M, L	←	
	展開と折り畳み	CTRL+M, M	←	
	停止	CTRL+M, P	←	開始はメニューコマンド
選択範囲	隠す	CTTL+M, CTRL+H	←	
	表示する	CTRL+M, O	←	
スニペット	スニペットの挿入	CTRL+K, X	←	
置換	クイック置換	CTRL+H	←	
	ファイルを指定して置換	CTRL+SHIFT+H	←	新機能
インテリセンス	メンバーリストの表示	CTRL+J	←	
	メンバー情報の表示	CTRL+K, I	CTRL+I	
	パラメータ表情の表示	CTRL+K, P	CTRL+SHIFT+I	
	透過モードで表示	CTRL	←	
ナビゲーション	移動 (Navigate To)	CTRL+,(コンマ)	←	新機能
	シンボルの検索	SHIFT+F12	ALT+F2	
	定義へ移動	F12	←	
	元の位置へ移動	CTRL+-(ハイフン)	←	
	次の位置へ移動	CTRL+SHIFT+-(ハイフン)	←	

Visual Studio

便利なショートカット –抜粋-

分類	説明	C#	VB	備考
ファイル	新しいプロジェクト	CTRL+SHIFT+N	←	
	プロジェクトを開く	CTRL+SHIFT+O	←	
	新しいクラスを追加	CTRL+ALT+C	×	
	既存の項目を追加	CTRL+ALT+A	CTRL-D	
	新しい項目を追加	CTRL+SHIFT+A	←	
ウィンドウ	クラスビュー	CTTL+W, C	CTRL+SHIFT+C	
	エラーリスト	CTRL+W, E	←	
	オブジェクトブラウザ	CTRL+W, J	F2	
	出力ウィンドウ	CTRL+W, O	CTRL+ALT+O	
	プロパティウィンドウ	CTRL+W, P	F4	
	タスクリスト	CTRL+W, T	CTRL+ALT+K	
	ツールボックス	CTRL+W, X	CTRL+ALT+X	
	データソース	SHIFT+ALT+D	←	
	ズームイン	CTRL+SHIFT+.(ドット)	←	新機能 CTRL+ホイール
	ズームアウト	CTRL+SHIFT+,(コンマ)	←	
デバッグ	イミディエイト	CLRL+D, I	CTRL+G	
	デバッグ無しで実行	CTRL+F5	←	
	デバッグの中止	SHIFT+F5	CTRL+ALT+Break	
	ブレークポイントの表示	CTRL+D, B	CTRL+ALT+B	
	コード呼出し階層	CTRL+W, K	×	C#のみの新機能

XAML 101

XAML とは

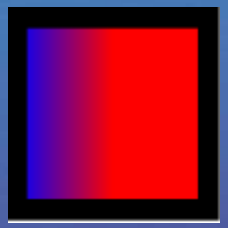
- eXtensible Application Markup Language
- WPF や Silverlight で UI を定義するために使用する XML
- Workflow Foundation でもワークフローを定義するために使用できる
- グラフィカルにデザインするためには、ツールを使用するのがベター
  - Visual Studio
  - Expression Blend
- XML のためテキスト エディタでも OK
  - 描画を確認するためには XAML Pad (Windows SDK で提供)が用意されている

図形の例 – Rectangle -

```
<Rectangle
    Canvas.Top= "10"
    Canvas.Left= "10"
    Height = "100"
    Width = “100"
    Fill = "LightBlue"
    Stroke = "Black"
    StrokeThickness= "10"
/>
```

グラデーションの例

```
<Rectangle
    Canvas.Top= "10"
    Canvas.Left= "10"
    Height = "100"
    Width = “100"
    Stroke = "Black"
    StrokeThickness= “10” >
  <Rectangle.Fill>
    <LinearGradientBrush
       StartPoint="0,0" EndPoint=“0,1">
      <LinearGradientBrush.GradientStops>
        <GradientStop
            Color="Blue" Offset="0.0" />
        <GradientStop
            Color="Red" Offset=“1" />
      </LinearGradientBrush.GradientStops>
    </LinearGradientBrush>
  </Rectangle.Fill>
</Rectangle>
```

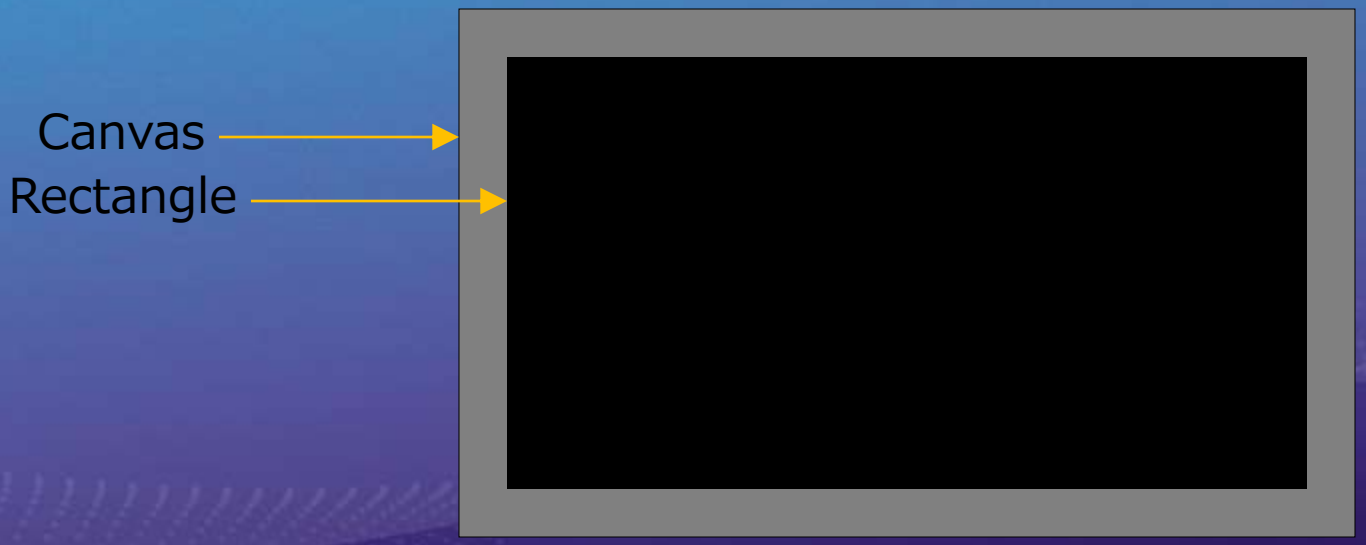


複数の図形を組合わせた例

```
<Canvas xmlns="..."
        xmlns:x="...">
  <Rectangle Width="200" Height="150" Fill="Black" />
  <Ellipse Width="200" Height="150" Stroke="Orange" />
</Canvas>
```

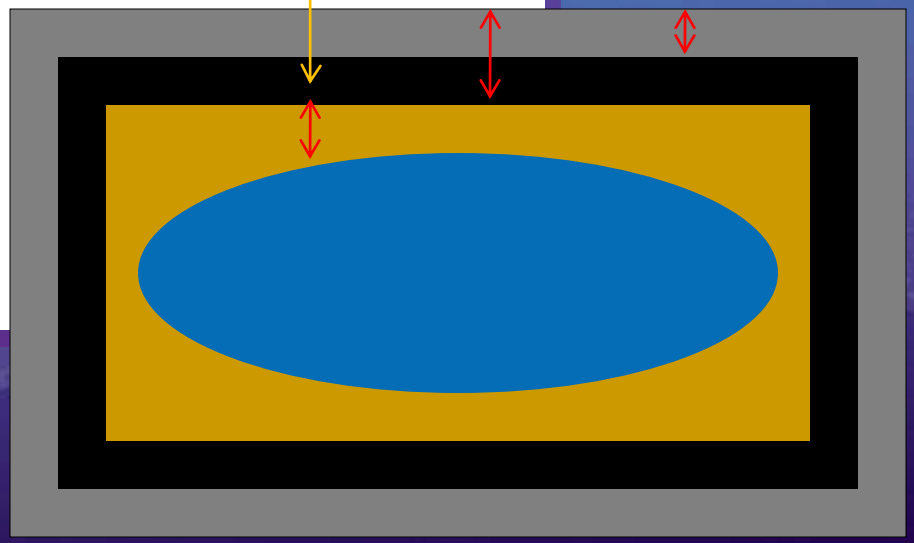
グルーピングした例

```
<Canvas Width="250" Height="200">
  <Rectangle Canvas.Top="25" Canvas.Left="25"
             Width="200" Height="150" Fill="Black" />
</Canvas>
```



相対座標の例

```
<Canvas xmlns="..."
        xmlns:x="..." Background="LightGray">
  <Rectangle Canvas.Top="25" Canvas.Left="25"
             Width="200" Height="150" Fill="Black" />
  <Canvas Canvas.Top="50" Canvas.Left="50"
          Width="150" Height="100"
          Background="Orange">
    <Ellipse Canvas.Top="25"
             Canvas.Left="25"
             Width= "100"
             Height= "50"
             Fill="Blue" />
  </Canvas>
</Canvas>
```



Microsoft Visual Studio

視覚効果

```
<Image Source="pic01.jpg"
       Height="80"
       Width="80">
<Image.RenderTransform>
  <RotateTransform Angle="40" />
</Image.RenderTransform>
</Image>
```

視覚効果

設定値	効　　果
RotaionTransform	回転
ScaleTransform	拡大／縮小
SkewTransform	傾斜
TranslateTransform	指定方向にオブジェクトを移動する
MatrixTransform	上記すべてを組み合わせたもの

RotateTransform

ScaleTransform

SkewTransform

TranslateTransform

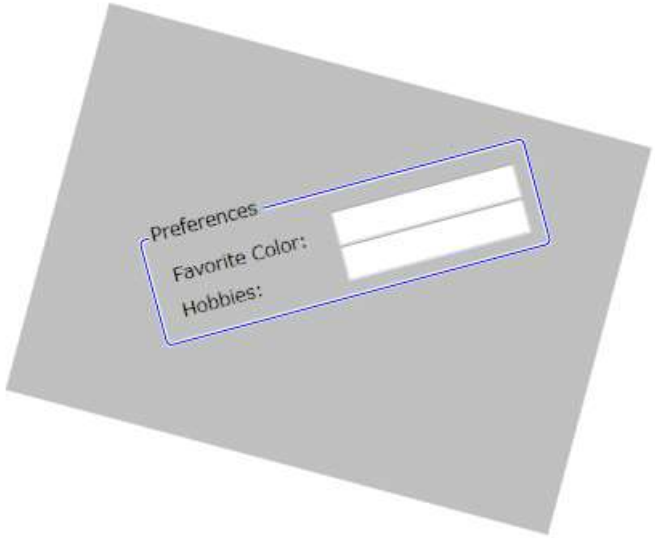
- 図形には、Opacity（透明）、Clip（切り取り）、Stretch（伸縮）などのさまざまな効果が用意されている

座標変換(視覚効果)

```
<Canvas VerticalAlignment="Stretch" Background="#FFC0C0C0"
        Margin="90,20,60,100" Height="250" Width="350">
  <Canvas.RenderTransform>
    <RotateTransform Angle="15"/>
  </Canvas.RenderTransform>

  <GroupBox Margin="50,130,0,0" Header="Preferences"
            BorderBrush="#FF0000FF" Height="80" Width="250">
    <GroupBox.RenderTransform>
      <RotateTransform Angle="-30"/>
    </GroupBox.RenderTransform>

    <Grid Margin="5,5,5,5">
      <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition />
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="110" />
        <ColumnDefinition Width="*"/>
      </Grid.ColumnDefinitions>
```

座標変換(視覚効果) -続き-

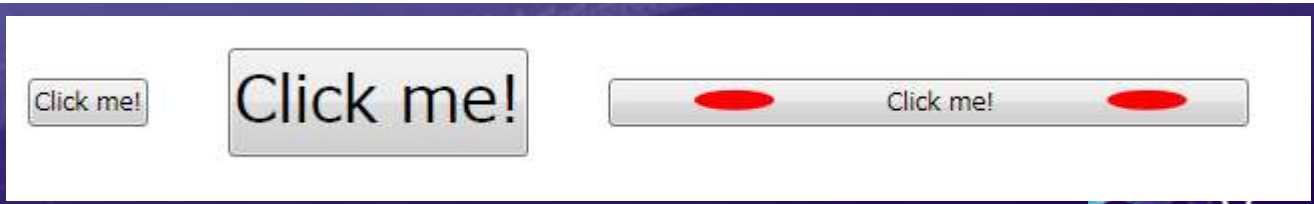
```
    <Label Grid.Row="0" Grid.Column="0"
           VerticalAlignment="Center"
           Content="Favorite Color:"/>
    <Label Grid.Row="1" Grid.Column="0"
           VerticalAlignment="Center" Content="Hobbies:"/>
    <TextBox Grid.Column="1" Grid.Row="0"
           HorizontalAlignment="Stretch"
           VerticalContentAlignment="Center"/>
    <TextBox Grid.Column="1" Grid.Row="1"
           HorizontalAlignment="Stretch"
           VerticalContentAlignment="Center"/>
  </Grid>
 </GroupBox>
</Canvas>
```



ボタンのカスタマイズ例

- 見た目を自由に変更できる

```
<Button Margin="20" VerticalAlignment="Center">
    Click me!
</Button>

<Button Margin="20" VerticalAlignment="Center" FontSize="24pt">
    Click me!
</Button>

<Button Margin="20" VerticalAlignment="Center">
  <TextBlock>
    <Ellipse Fill="Red" Width="40" Height="10" />
    Click me!
    <Ellipse Fill="Red" Width="40" Height="10" />
  </TextBlock>
</Button>
```



スタイル

- スタイルとは
  - 複数のプロパティをまとめて設定する定義
  - 一つのスタイルを予め定義し、複数のコントロールに適用可能
  - スタイルは、適用先のコントロールが持つ本質的な機能とは切り離して考えることができる

```
<!-- スタイル -->
<Grid.Resources>
   <Style x:Key="BigButtonText" TargetType="{x:Type Button}">
     <Setter Property="FontStyle" Value="Italic" />
     <Setter Property="FontSize" Value="36pt" />
   </Style>
</Grid.Resources>

<!-- スタイルの適用 -->
<Button Margin="20" VerticalAlignment="Center"
        Style="{DynamicResource BigButtonText}">
  Click me!
</Button>
```

Style 属性に特定のスタイルを適用する。

コントロールテンプレート

- コントロール テンプレートとは
  - コントロール内のコンテンツなどの構造を予め定義したテンプレート
  - スタイルとは異なり、マークアップ ベースでコントロールの構造を端的に表せる (特定のコントロールを想定した構造の定義に便利)
  - スタイルの中にコントロール テンプレートを含むこともできる

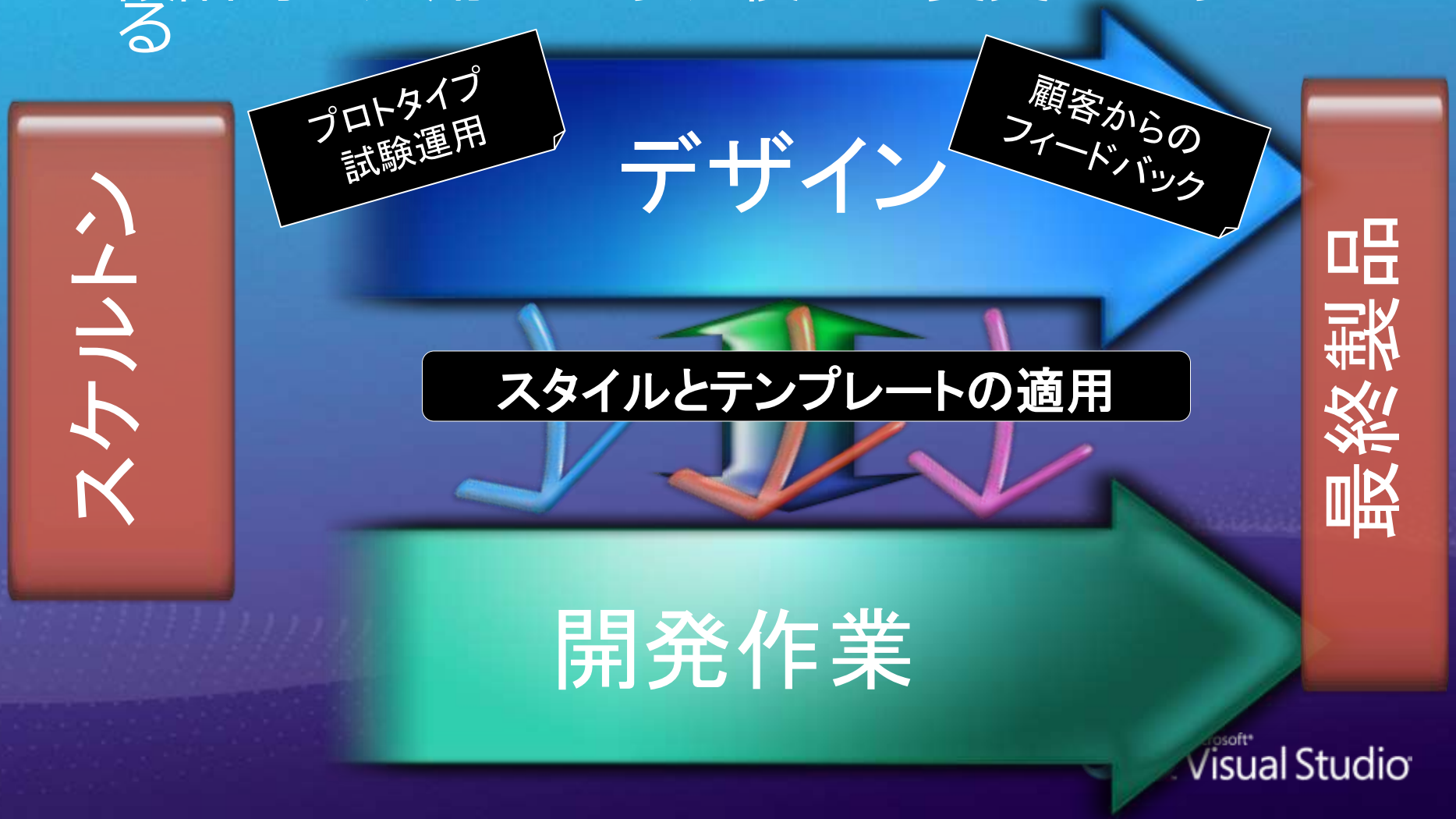
```
<!-- テンプレート -->
<Grid.Resources>
  <ControlTemplate x:Key="OurTemplate" TargetType="{x:Type Button}">
    <Grid>

    </Grid>
  </Controltemplate>
</Grid.Resources>

<!-- テンプレートの適用 -->
<Button Margin="20" VerticalAlignment="Center"
        Template="{DynamicResource BigButtonText}"
</Button>
```

適用先のコントロールの構造を直接マークアップで表現する

Template 属性に特定のテンプレートを適用する。

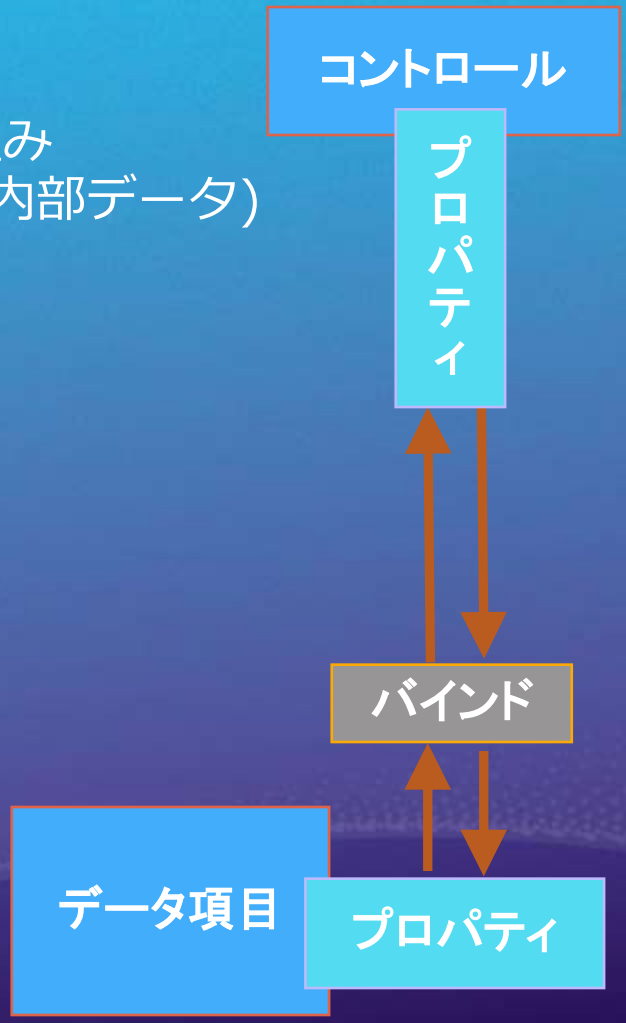
スタイルとテンプレートの活用

- 予め用意したスケルトンや基本部分に、スタイルやテンプレートとして
  段階的に適用したり、後から変更したりできる



データバインディング

- データ バインディングとは
  - UI と内部データが自動的に連動する仕組み
  - どう見せるか (UI) と、どう処理するか(内部データ)を分離して実装できる (連動は自動)
- WPF におけるデータ バインディング
  - ターゲット (UI)
    - WPF 要素の任意のプロパティ
  - データソース
    - WPF や .NET オブジェクトのプロパティ、ADO.NET データセット、XML データなど



Microsoft Visual Studio

基本的な使用方法

```
<Canvas>

    <Image Source="SampleImg.bmp"
     Canvas.Left="{Binding Path=Value, ElementName=horzPos}"/>

    <Slider Canvas.Top="310" Canvas.Left="100"
     Minimum="0" Maximum="100" Width="100"
     Orientation="Horizontal" Name="horzPos" Value="50" />

</Canvas>
```

ターゲットとなる <Image>

データソースとなる <Slider>

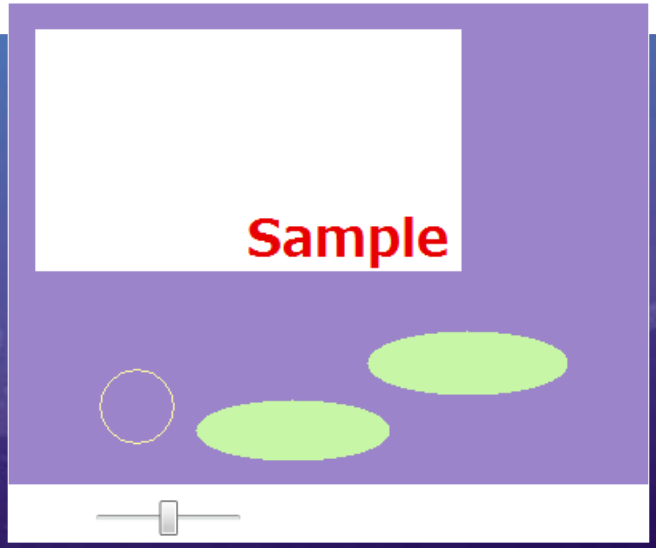


Image の Canvas.Leftプロパティ

Slider の Value プロパティ

Microsoft Visual Studio

リソース情報

- Future Technology Days
  - http://www.microsoft.com/japan/powerpro/ftd/default.mspx

- Tech Days 2010 テクニカルセッション ストリーミング
  - http://www.microsoft.com/japan/events/techdays/2010/session/download.aspx

Microsoft®

Future
Technology Days

Microsoft®
Visual Studio®