Beta 1 と Beta 2 との間の断層的変更

このトピックでは、Silverlight 2 Beta 1 リリースと Beta 2 リリースとの間で Silverlight ランタイムと Silverlight Tools に行われた変更を議論します。本稿で議論する変更は、皆さんの以前の Silverlight ベースのアプリケーションが落ちたり、違ったふるまいが発生するかもしれない変更に焦点を当てており、本リリースにおける新しい機能／拡張には触れません。

注意: 本ドキュメントに修正／追加があった場合はここにリストします。

注意: Beta 1 でビルドした Beta 1 アプリケーションは、Beta 2 ランタイム ビルド上では動作しません。エンドユーザーにはそのアプリケーションが以前のバージョンの Silverlight でビルドされたというメッセージが表示されます。Beta 1 アプリケーションは再ビルドする必要がありますー詳細は「重要! 古いコードを新しいリリースに移行」を参照してください。

注意: Silverlight API の多くの（しかし全てではない）変更は、Windows Presentation Foundation との互換性のために行われました、

ヒント: 多数の API 変更があります、特にその他の API 変更のセクションに多くリストされていますが、CTRL+F を使ってドキュメントを検索すると良いでしょう。

内容

重要! 古いコードを新しいリリースに移行

Silverlight 2 Beta 1 リリースやそれ以前のリリースで作成した Silverlight アプリケーションを Silverlight 2 Beta 2 に移行しようとするなら、最新の Silverlight 2 Tools for Visual Studio 2008 を取得し、そのプロジェクトを再コンパイルする必要があるでしょう。

1. 最新バージョンの Silverlight 2 Tools for Visual Studio 2008 を取得してください。 Silverlight デベロッパーセンターからこれらのツールが取得できます。このインストールを行う前に古い Tools をアンインストールしなければならないかもしれません。
2. 古いプロジェクト（ex. .csproj ファイル）を開きます。そのプロジェクトが古いバージョンの Silverlight Tools で作成されていると通知され、プロジェクトをアップグレードしたいかどうかが尋ねられます。Yes をクリックしてください。
3. デバッグしてください。 遭遇するであろう断層的変更の多くは本ドキュメント内で見つかるはずです。

Silverlight.js の更新

影響を受けるのは: Silverlight のバージョンとインスタンス化を Silverlight.js に依拠している全ての人。Silverlight.js ファイルを Object タグの中に配置してプラグインをインスタンス化していれば、この変更の影響は受けません。

概要

Silverlight.js は変更されたので、アプリケーションのバージョンのアップデートが必要です。古いバージョンの Silverlight.js ではエンドユーザー用の Silverlight 2 Beta 2 のインストールを保証できません。

必要な修正

Silverlight.js のバージョンをアップデートしてください。このファイルの新バージョンは Silverlight 2 Beta 2 SDK にあります。

Socket の断層的変更

影響を受けるのは: Socket クラスを使っている Silverlight マネージコード アプリケーション

概要

Socket に関する「元のサイトのみに接続」ポリシーは、HTTP に関するポリシーシステムと同様の、完全なクロスドメイン ポリシーに置き換えられました。Socket 接続が許可される前に、ターゲット サーバーは要求されたポートへのアクセスを許可するポリシーを提供しなければなりません。元のアプリケーションのサイトは特別なアクセス権を受け取りません：「元のサイトのみに接続」ポリシーの基で以前は許可された接続は、その接続を許可するポリシーをそのサーバーが配置しない限り、失敗するようになるでしょう。

- アプリケーションのアクセス許可ポリシーを配置した任意のサーバーに対してのみ接続が可能になりました。
- DNS 記録はもはや参照しないので、PTR 記録やアプリケーションの参照にどの URL を使ったのかに関して気にする必要はありません。

必要な修正

Silverlight アプリケーションからの接続を許可したい全てのサーバーは、その接続が「元のサイトのみに接続」ポリシーの基で以前は許可されていたとしても、クロスドメイン ポリシー サーバーを配置する必要があります。

> **注意**　クライアントのコードは変更や再コンパイルの必要はありません、これは純粋にふるまいの変更です。

コントロールのビルトイン スタイルに関する変更

影響を受けるのは: コントロールのサブクラスの開発者

概要

コントロール用のビルトイン スタイルは、**DefaultStyleKey** プロパティの参照によって決定するようになりました。このプロパティのデフォルトは **null** です（どのビルトイン スタイルも使っていないことを意味する）。

コントロールクラスがその親と異なるビルトイン スタイルを提供したいとき、そのコントロール コンストラクタ内で **DefaultStyleKey** を設定すべきです。この変更によって可能となるシナリオは、コントロールのサブクラスを実装して機能を追加したいが、コントロールのビルトインの見え方は変更したくない場合です。

例： Button で DefaultStyleKey を設定

```
[C#]
public class Button : Control
{
    public Button() : base()
    {
        ...
        DefaultStyleKey = typeof(Button);
    }
}
```

例： Button と異なるビルトイン スタイルを提供する MyButton

```
[C#]
public class MyButton : Button()
{
    public MyButtion() : base()
    {
        ...
        DefaultStyleKey = typeof(MyButton);
    }

    // other custom PME
    public bool MyProperty {get; set;}
    ...
}
```

例：Button からビルトイン スタイルを継承した MyButton

```
[C#]
public class MyButton : Button()
{
    public MyButtion() : base()
    {
        ...
    }
```

```
    // other custom PME
    public bool MyProperty {get; set;}
    ...
}
```

MIME 型とインストール URL の変更

影響を受けるのは: Silverlight 2 Beta 2 をターゲットとし、かつ（Silverlight.js の配置に）**OBJECT** タグを使った任意のツール／ドキュメント／コンテンツは新しい MIME 型を使うべきです。

概要

MIME 型の変更について：Beta 2 の Object タグによるインスタンス化をサポートし、Beta 2 用のインスタンス化の試みから Silverlight 1.0 と 2.0 Beta 1 を守るために、Silverlight の第二 MIMI 型が"**application/x-silverlight-2-b1**" から "**application/x-silverlight-2-b2**"に変更になりました。

インストーラ URL 変更について：Beta 2 MIME 型を使うときは Beta 2 インストーラ URL（http://www.microsoft.com/silverlight/handlers/getsilverlight.ashx?v=2.0）と一致すべきです。Beta 2 が出荷されると Beta 1 インストーラ URL は、Beta 1 はもうサポートされないことを説明する文章にリダイレクトされるので、サイト制作者は Beta 2 への接続に更新する必要があります。

必要な修正

プラグインをインスタンス化する **OBJECT** タグを使ったページの MIME 型を修正しなければなりませんーこれには Silverlight プロジェクトが作成されたとき Web に追加された HTML テストページも含まれます。

インストーラ URL 変更について：Beta 2 MIME 型を使うときは Beta 2 インストーラ URL（http://www.microsoft.com/silverlight/handlers/getsilverlight.ashx?v=2.0）と一致すべきです。

SetTargetProperty と GetTargetProperty の変更

影響を受けるのは: マネージ Silverlight アプリケーションからこれらのメソッドを呼び出す場合。この変更は JavaScript アプリケーションには全く影響を与えません。

概要

Storyboard.TargetProperty 型用のアタッチプロパティは、Windows Presentation Foundation (WPF)との互換性のために変更されました。

Beta 1

[C#]

```
public static string GetTargetProperty(Timeline element);
public static void SetTargetProperty(Timeline element, string value);
```

Beta 2

[C#]

```
public static PropertyPath GetTargetProperty(Timeline element);
```

```
public static void SetTargetProperty(Timeline element, PropertyPath
value);
```

必要な修正

例えば:

Beta 1

```
[C#]
String strPath = Storyboard.GetTargetProperty(timeline);
Storyboard.SetTargetProperty(timeline, "(Canvas.Left)");
```

Beta 2

```
[C#]
String strPath = Storyboard.GetTargetProperty(timeline).Path;
Storyboard.SetTargetProperty(timeline, new
PropertyPath("(Canvas.Left)"));
```

System.Windows.Control.dll は System.Windows.dll に統合

影響を受けるのは: SDK と共に出荷された **System.Windows.Control.dll** を参照する Silverlight Beta 1 アプリケーション。

概要

SDK（ランタイムではなく）で出荷された System.Windows.Controls.dll と呼ばれる DLL は、完全に削除されました。その DLL 内にあったコントロールは、ランタイムと共に出荷される System.Windows.dll 内に存在するようになりました。アプリケーションを動作させるには、そのアプリケーションを再コンパイルしなければならないでしょう（「重要！古いコードを新しいリリースに移行」を参照）。

次のコントロールはランタイムと共に出荷される System.Windows.dll 内に存在するようになります:

- **ContentControl**
- **ContentPresenter**
- **RangeBase**
- **Slider**
- **ScrollBar**
- **Thumb**
- **ScrollViewer**
- **ButtonBase**
- **Button**
- **RepeatButton**
- **ToggleButton**
- **HyperlinkButton**
- **CheckBox**

- **RadioButton**
- **ListBox**
- **ListBoxItem**
- **OpenFileDialog**
- **Tooltip**
- **TooltipService**
- **ClickMode**
- **DialogResult**
- **DisplayMemberValueConverter**
- **FileDialogFileInfo**
- **ScrollBarVisibility**
- **ScrollContentPresenter**
- **SelectionChangedEventArgs**
- **SelectionChangedEventHandler**
- **SelectionMode**
- **DragCompletedEventArgs**
- **DragDeltaEventArgs**
- **DragStartedEventArgs**
- **ScrollEventArgs**
- **ScrollEventType**

必要な修正

プロジェクトを再コンパイルしてください（「重要！古いコードを新しいリリースに移行」を参照）。さらに、`xmlns:local='clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls'`のような参照を取り除き、local:*Control* 型構文(ex. `local:ListBox`)をクリーンアップする必要があります。

Tooltip の変更

影響を受けるのは: Tooltip を使った Silverlight Beta 1 アプリケーション

概要

Tooltip は見えなくなり、もはや直接使うことはできませんーTooltip は **TooltipService** を経由したコントロールに追加されました（詳細は次の例を参照）。さらに、**TooltipService** API が刈り込まれました。次のパブリック プロパティとメソッドが **TooltipService** から取り除かれました。

- **BetweenShowDelayProperty**
- **InitialShowDelayProperty**
- **ShowDurationProperty**
- **GetBetweenShowDelay**
- **GetInitialShowDelay**
- **GetShowDuration**

- **SetBetweenShowDelay**
- **SetInitialShowDelay**
- **GetTooltip**
- **SetShowDuration**

必要な修正

Tooltip は **TooltipService** 経由で設定しなければならなくなりました。XAML では次の 2 つの方法のいずれかを使います：

[xaml]

```
<Button TooltipService.Tooltip="This is Tooltip text"/>
<Button Content="Button">
  <TooltipService.Tooltip>
    <TextBlock Text="text"/>
  </TooltipService.Tooltip>
</Button>
```

あるいはコードでは:

[C#]

```
TooltipService.SetTooltip(myElementNeedingATooltip, new TextBox());
```

注意：Tooltip はテキストだけではなくより複雑な要素を持つかもしれません:

[xaml]

```
<Button Content="Button">
  <TooltipService.Tooltip>
    <StackPanel>
      <Button Content="A Button Tooltip"></Button>
    </StackPanel>
  </TooltipService.Tooltip>
</Button>
```

TooltipService 用のパブリック API はほとんど最小のものになりました:

[C#]

```
public static class TooltipService
{
    public static readonly DependencyProperty TooltipProperty;
    public static void SetTooltip(DependencyObject element, object
value);
}
```

System.Windows*dll にあるコントロールからいくつかのプロパティを取り除いた

影響を受けるのは: 独自のフォントプロパティを定義したコントロールの制作者はそのコードとテンプレートを変更しなければなりません。

概要

Control 基本クラスに、次の継承クラスを追加した：

- **FontFamily**
- **FontSize**
- **FontWeight**
- **FontStyle**
- **Foreground**
- **FontStretch**

結果的に、個々のプロパティとこれらのプロパティに対応したテンプレートバインディングが system.windows*.dll 内のいくつかのコントロールから取り除かれました。

XAML 内での System.Windows.Controls.Extended.dll の扱いの変更

影響を受けるのは: System.Windows.Controls.Extended.dll からコントロールを使用した XAML

概要と必要な修正

XAML ではもう、明示的に名前空間とプレフィックスを指定しないと System.Windows.Controls.Extended.dll からの型は使えません。System.Windows.Control.Extended.dll アセンブリから **XmlnsDefinition** 属性も取り除かれました。

Beta 1

[xaml]

```
<Canvas>
  <DatePicker ...>
</Canvas>
```

Beta 2

[xaml]

```
<Canvas xmlns:swce="clr-
namespace:System.Windows.Controls;assembly=system.windows.controls.exte
nded">
  <swce:DatePicker ...>
</Canvas>
```

HtmlElement.GetAttribute と HtmlElement.GetProperty に対する変更

概要と必要な修正

Beta 1 リリースでは、文字列として型付けられた DOM プロパティについて、**HtmlElement.GetAttribute** と **HtmlElement.GetProperty** はどちらでも使えました。Beta 2 ではこのふるまいが変更されました。

HtmlElement.GetAttribute は DOM 要素属性の取得にのみ使えます、一方 DOM 要素プロパティの取得には **HtmlElement.GetProperty** を使わなければなりません。**HtmlElement.GetAttribute** 経由での DOM 要素プロパティの取得はもはやサポートされていません。

Calendar/DatePicker の変更

影響を受けるのは: DatePicker あるいは **Calendar** の使用者

概要

Calendar と **DatePicker** には以下の変更が行われました：

Calendar

- [削除] **Style DayStyle**
- [削除] **Style MonthStyle**
- [削除] EventHandler<CalendarDateChangedEventArgs> DateSelected
- [追加] EventHandler<SelectionChangedEventArgs> SelectedDatesChanged

DatePicker

- [削除] **Tooltip Tooltip**
- [削除] EventHandler<DatePickerTextParseErrorEventArgs**> TextParseError**
- [追加] EventHandler<DatePickerDateValidationErrorEventArgs> DateValidationError
- [削除] EventHandler<DatePickDateChangedEventArgs> DateSelected
- [追加] EventHandler<SelectionChangedEventArgs> SelectedDateChanged

Calendar と **DatePicker**

- [削除] **bool IsEnabled** {**get**; **set**;}
- [削除] **DateTime? SelectableDateStart**
- [削除] **DateTime? SelectableDateEnd**
- [削除] **bool AreDatesInPastSelectable**

HtmlPage.UnregisterScriptableObject は取り除かれました。

影響を受けるのは: HtmlPage.UnregisterScriptableObject の使用者

必要な修正

HtmlPage.UnregisterScriptableObject を使う代わりに、開発者は **RegisterScriptableObject** 用の同じスクリプトキーを再利用できます。これによって、開発者はスクリプタブルのエントリーポイントと関連付けられた基のマネージオブジェクトを変更できます。

WebClient と HttpWebRequest の変更

影響を受けるのは: WebClient と **HttpWebRequest** を使うマネージコード アプリケーション

概要

次の変更がコードをブレークするかもしれません

- **WebClient** は System.Net.dll に移動しました。
- **WebClient.BaseAddress** プロパティの型は、デスクトップとの互換性のために、Uri ではなく文字列になりました。
- 例外処理ロジックが変更されました。
- **HttpWebRequest** デリゲートは、バックグラウンド スレッド上で戻るように変更されました。
- **WebClient** と **HttpWebRequest** に関連した新しい機能：
  - **WebClient** API にアップロード(i.e. "POST") セマンティクスとリクエストヘッダーが含まれます。
  - **WebClient** は（UI スレッドに加え）バックグラウンド スレッドから呼び出せます。
  - **HttpWebRequest** デリゲートはバックグラウンド スレッドで戻るように変更されました。
  - **HttpWebRequest** はバックグラウンド スレッド上で呼び出せます
  - **WebClient** と **HttpWebRequest** は、**AllowReadStreamBuffering** を **false** に設定することにより、ストリーミングをサポートします。
  - **WebClient** のようにイベントベースのコンポーネントの実装を容易にするために、**AsyncOperationManager** が利用可能になりました。
  - スレッド管理をより簡単にするために **SynchronizationContext** と関連クラスが追加されました。

更新された **WebClient** API は、より充実した機能の **WebClient** を提供します。これはどのネットワーク API を使うかについてガイダンスを変更します：

- **WebClient**: 容易に使えるイベント ベースの API を使いたいとき使用するクラス。
- **HttpWebRequest**: デリゲート ベースのモデルが必要なときや、応答ストリームのプログレッシブ読み込みが可能なときに使用するクラス。

必要な修正

1. **WebClient** の顧客は **System.Net.dll** を必ず参照し再コンパイルする必要があります（詳細は、「重要！古いコードを新しいリリースに移行」を参照）。
2. **WebClient.BaseAddress** の型は URI から文字列に変更になりました。必要に応じて次のように Uri に変換してください：`Uri myUri = new Uri(WebClient.BaseAddress);`
3. 前の **WebClient** は、非同期メソッドが呼び出されたとき、ポリシーチェック失敗中に **SecurityException** を投げる場合がありました。非同期呼び出しパターンを真に可能にするために、非同期例外の結果が有効となるようにこれを変更し、デスクトップ上にあるときに、コールバック **EventArgs** の **Result** プロパティにアクセスするようにしました。同様の変更を **HttpWebRequest** にも行いました。

Beta 1

[C#]

```
public void Download()
{
    WebClient client = new WebClient();
    client.DownloadStringCompleted += new
DownloadStringCompletedEventHandler(
        client_DownloadStringCompleted);

    try
    {
        client.DownloadStringAsync(new
Uri("http://Contoso.com/myfile.txt"));
    }
    catch (SecurityException ex)
    {
        // 非同期例外の処理
    }
}

void client_DownloadStringCompleted(
    object sender, DownloadStringCompletedEventArgs e)
{
    // コントロールのテキストコンテンツを結果に設定
    MyControl.Text = e.Result;
}
```

Beta 2

```
[C#]
public void Download()
{
    WebClient client = new WebClient();
    client.DownloadStringCompleted += new
DownloadStringCompletedEventHandler(
        client_DownloadStringCompleted);

    client.DownloadStringAsync(new
Uri("http://Contoso.com/myfile.txt"));
}
```

```
void client_DownloadStringCompleted(
    object sender, DownloadStringCompletedEventArgs e)
{
    try
    {
        // コントロールのテキストコンテンツを結果に設定
        MyControl.Text = e.Result;
    }
    catch (SecurityException ex)
    {
        //非同期例外の処理
    }
}
```

System.Xml における Null 引数チェックの改善

影響を受けるのは: System.Xml 内のメソッドを使用する Silverlight 2 アプリケーション。

概要

System.Xml 内の多くのパブリック メソッドでは、**null** プロパティに関する引数チェックを行わず、null 引数が使われたとき **NullReferenceException** を投げていました。この変更では、null 引数について明示的なチェックが追加され、代わりに **ArgumentNullException** を投げるようになりました。

以下のメソッドがこの改善された引数チェックを行います：

- **XmlReader.IsName**
- **XmlReader**[**null**]
- **XmlReader.MoveToAttribute**
- **XmlReader:IsNameToken**
- **XmlNamespaceManager** コンストラクタ
- **XmlReader.Create**
- **XmlReader.Nametable.Add**
- **XmlReader.Nametable.Get**
- **XmlCharCheckingWriter.WriteQualifiedName(null, null)**
- **XmlCharCheckingWriter.WriteDocType(null, null, null, null)**
- **XmlConvert.ToBoolean**
- **XmlConvert.ToDouble**
- **XmlConvert.ToSingle**
- **XmlUrlResolver.GetEntity**
- **XmlConvert.ToDateTime**

- **XmlConvert.ToBinHexString**
- **XmlConvert.FromBinHexString**

BackgroundWorker の移動

影響を受けるのは: BackgroundWorker を使用する Silverlight 2 マネージ アプリケーション。

概要

BackgroundWorker は System.dll に移動し、**AsyncOperationManager** が利用可能になりました。

必要な修正

すでに System.dll への参照を仮定しているので、**BackgroundWorker** を利用するアプリケーションは、場所変更を反映するために再コンパイルする必要があるだけです。

Deep Zoom Image とコレクションのフォーマット変更

影響を受けるのは: Deep Zoom を使用する Silverlight 2 マネージ アプリケーション。

概要

Deep Zoom ファイルフォーマットは、バイナリ フォーマット（.bin）から XML 使用に変更されました。この変更により、フォーマットは XAML スタイル・バージョンと一致し、内部的な一貫性が改善されました。

必要な修正

既存の全ての Deep Zoom コンテンツは、この更新フォーマットで動作するように再生成する必要があります。最新バージョンの Deep Zoom Composer を使えばこれができます。コレクションと画像に関する新しいファイル フォーマットは以下のとおりです：

myimage.dzi (あるいはコレクションの場合 myimage.dzc)
myimage_files\thumbs\0\0_0.jpg
myimage_files\thumbs\1\0_0.jpg
myimage_files\thumbs\2\0_0.jpg
myimage_files\thumbs\3\0_0.jpg
myimage_files\thumbs\3\0_1.jpg
myimage_files\thumbs\3\1_0.jpg
myimage_files\thumbs\3\1_1.jpg

さらに、コードでは画像やコレクションの正しいソースを指す必要があります。

Beta 1

```
[C#]
MutliScaleImage msi = new MultiScaleImage();
msi.source="http://bla/image.bin";
```

Beta 2

```
[C#]
MutliScaleImage msi = new MultiScaleImage();
```

```
msi.source="http://bla/image.dzi";
```

MultiScaleImage の変更

影響を受けるのは: 手続き型コード（ex. C#, VB .NET など）で **MultiScaleImage** を使用する Silverlight 2 Beta 1 マネージ アプリケーション。XAML 内の文字列で **Source** を URI に設定している場合と JavaScript は変更ありません。

概要

MultiScaleImage Source プロパティは **Uri** オブジェクトから **MultiScaleTileSource** に変更されました。

必要な修正

Beta 1

```
[C#]
MultiScaleImage msi = msi.Source = new System.Uri("images/info.bin");
```

Beta 2

```
[C#]
msi.Source = new DeepZoomImageTileSource(new
System.Uri("images/info.dzi"));
// あるいは
msi.Source = new YourMultiScaleTileSourceOverride();
```

注意： **MultiScaleImage** ソースファイルについてファイル フォーマットも変更になっています。詳細は「Deep Zoom 画像とコレクションのフォーマット変更」を参照してください。

AllowInboundCallsFromXDomain の変更

影響を受けるのは: クロスドメインから提供された HTML ページ上にホストされている Silverlight 2 Beta 1 アプリケーション。

概要

Silverlight 2 アプリケーション マニフェストのルート要素上 **AllowInboundCallsFromXDomain** 属性は、クロスドメイン シナリオにおける、ホストページと Silverlgit アプリケーションの間のインタラクションを制御します。このプロパティは以下のようにリファクタリングされました：

- **AllowInboundCallsFromXDomain** は **ExternalCallersFromCrossDomain** に変更。
- 値の型は真偽値から列挙値（マネージコード型 **System.Windows.CrossDomainAccess**）に変更、値として **NoAccess**, **FullAccess**, **ScriptableOnly** を持ちます。その意図は、アプリケーション制作者にどのようなアクセス権を提供しているかを明示的に提示することです。
  - **NoAccess** は、現在の”false”値に対応し、クロスドメイン シナリオにおけるデフォルトです。
  - **FullAccess** は現在の”true”値に対応します。

- **ScriptableOnly** は新しい値で、クロスドメイン ホスト ページから Scriptable オブジェクトへのアクセスをトグルします。

スクロール関連の API と Drag*EventArgs に対する変更

影響を受けるのは: 以下のリストにある API を使用する Silverlight 2 Beta 1 アプリケーション。

概要

API シグネチャを WPF とそろえるために、**ScrollBar**/**Thumb**/**scrolling-related** クラスに以下の変更が行われました：

- **ScrollBar.Scroll** イベントは **ScrollEventHandler** 型になりました。
- **ScrollEventArgs** と **ScrollEventType** は **System.Windows.Controls.Primitives** 名前空間に移動しました。
- **ScrollContentPresenter.HorizontalOffset** と **ScrollContentPresenter.VerticalOffset** は読み取りのみになりました。
- **DragEventArgs** と **DragEventHandler** は取り除かれました。
- **DragCompletedEventArgs**, **DragDeltaEventArgs**, **DragStartedEventArgs** は、**DragEventArgs** ではなく、**RoutedEventArgs** の派生クラスになり、**DragEventArgs** からの継承を置き換えるために、適切なドラッグ プロパティを持ちます。

Glyphs 要素には UniCodeString 属性か Indices 属性が必要

影響を受けるのは: XAML 内で **Glyphs** 要素を使用する Silverlight 2 Beta 1 アプリケーション。

概要

WPF との互換性のために、**UnicodeString** と **Indices** のどちらの属性も設定されていないか、その属性が無効の **Glyphs** 要素を XAML 内に持つと、例外を投げるようになりました。

必要な修正

Indices 属性を”0”の値に設定すればこの例外に対処できます。

ServiceReferences.ClientConfig ファイルの変更

影響を受けるのは: ServiceReferences.ClientConfig ファイルを削除・編集する全ての人。

概要

Beta1 では:

- “ServiceReferences.ClientConfig”と呼ばれるファイルを生成するサービス参照を追加すると、XAP に追加されました。このファイルは使われないので、編集したり削除しても安全で何の影響もありませんでした。
- Silverlight WCF プロキシ用のデフォルト コンストラクタ（サービス参照ダイアログの追加を使うと生成される）は必ずデフォルト バインディング設定と、サービスメタデータ(WSDL)内に存在する元のアドレスを持つプロキシを作成しました。

Beta2 では:

Silverlight WCF プロキシ用のデフォルト コンストラクタ（サービス参照ダイアログの追加を使うと生成される）は、ServiceReferences.ClientConfig ファイルからバインディング設定とサービスアドレスを持つプロキシを作成します。このファイルを編集すると影響を与えるようになり、このファイルを削除するともはやデフォルト プロキシ コンストラクタの使用が許されなくなります。

通常の使用では（ServiceReferences.ClientConfig を編集せず削除もしないとき）、Beta 1 と Beta 2 のふるまいは同じです。

ItemsControl.Items の型は IList ではなく ItemCollection になった

必要な修正

ソースコードの変更は不要です（**ItemsControl.Items** はまだ **IList** をサポートするので）が、新しいシグネチャに対してアプリケーション バイナリを再コンパイルしなければなりません、そうしないと次のエラーが発生します。

Error
`System.MissingMethodException: Method not found: 'System.Collections.IList System.Windows.Controls.ItemsControl.get_Items()'.`

RoutedEventArgs.Handled=true イベントはもはやバブルしない

影響を受けるのは: これは主に、コントロールを使い、そのコントロールの外にバブルしたイベントを処理する人々に主に影響を与えます。ルート要素で Key や Mouse イベントを取得していたなら、そのいくつかはページ内のコントロールによって見えなくなるでしょう。Handled についてすでにチェックしていれば変更はありません。また、そのコントロールが見えなくしたイベントについてイベントハンドラを登録することはできなくなります一特に **TextBox** の Key イベント。

概要

JavaScript と DHTML では、イベントに `cancelBubble=true` を設定するとバブリングを停止します。`WPF` では同等のことをするために **RoutedEventArgs.Handled** を **true** にします。Silverlight では、Handled のイベントもバブルを続け、Handled のイベントを取得したくなければ、イベントハンドラ内でユーザーによる"if (e.Handled == true)"コードの追加が必要でした。これはユーザーを混乱させ、WPF と JavaScript との互換性の問題だったので、"Handled=true"イベントを変更しバブルしないようにしました。

TargetType と互換性のないコントロールに Style は適用できない

影響を受けるのは: Style を使用する全ての人々

概要と必要な修正

以下のコードは Beta 1 では動作しました。

Beta 1

[xaml]

```
<Style x:Key="myStyle" TargetType="Button"/>
<TextBox Style={StaticResource myStyle}"/>
```

Style.TargetType は設定を受けるオブジェクトの型と互換である必要になりました。

Beta 2

[xaml]

```
<Style x:Key="myStyle" TargetType="TextBox"/>
<TextBox Style={StaticResource myStyle}"/>
```

SetValue は正しい型にのみ許可される（変換はない）

影響を受けるのは: SetValue を使用する Silverilght 2 Beta 1 マネージ アプリケーション。

概要と必要な修正

SetValue は、与えられた **DependencyProperty** について正しい型にのみ許可されます。例えば：依存プロパティ **Canvas.LeftProperty** が **System.Double** 型なら、**System.Double** 型のオブジェクトを提供しなければなりません。変換はもはや行われません。別の例を以下に示します：

Beta 1

[C#]

```
ellipse.SetValue(Ellipse.StrokeProperty, "Black");
```

Beta 2

[C#]

```
ellipse.SetValue(Ellipse.StrokeProperty, new
SolidColorBrush(Colors.Black));
```

Beta 1 では文字列”Black”は **Black** 色の **SolidColorBrush** に変換されていました。Beta 2 では **SolidColorBrush** 型のオブジェクトを渡さなければなりません。

Control.InitializeFromXaml は削除された

影響を受けるのは: InitializeFromXaml を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

必要な修正

InitializeFromXaml ではなく **LoadComponent** を使わなければならない。

Beta 1

[C#]

```
public class MyControl : Control
{
    public MyControl() : base()
    {
        System.IO.Stream s =
this.GetType().Assembly.GetManifestResourceStream(
            "MyNamespace.UserControl1.xaml");
        FrameworkElement content = this.InitializeFromXaml(
            new System.IO.StreamReader(s).ReadToEnd());
        this.namedElement = (Rectangle)
content.FindName("namedElement");
    }
    private Rectangle namedElement; // xaml コンテンツ
x:Name="namedElement"
```

```
}
```

[xaml]

```
<Grid
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <Rectangle x:Name="namedElement"/>
  <Ellipse />
</Grid>
```

Beta 2

[C#]

```
public class MyControl : UserControl
{
    public MyControl() : base()
    {
        // VS 外でビルドするとき
        Application.LoadComponent(this,
            new Uri("MyApp;component/MyControl.xaml",
UriKind.Relative));
        FrameworkElement content = this.Content;
        this.namedElement = (Rectangle)
content.FindName("namedElement");
    }
    private Rectangle namedElement; // xaml contains
x:Name="namedElement"
}
```

[xaml]

```
<UserControl x:Class="Namespace.MyControl"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <Grid>
    <Rectangle x:Name="namedElement"/>
    <Ellipse />
  </Grid>
</UserControl>
```

同じ要素に Name と x:Name を指定することはできない

概要と必要な修正

Beta 1

[xaml]

```
<Canvas xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Name="rootCanvas" x:Name="testCanvas"
  Width="20" Height="20" Background="Violet">
</Canvas>
```

Name と **x:Name** の両方があるにもかかわらず、上の XAML は Beta 1 で動作しました。上の XAML はエラーを出すようになります。**x:Name** や **Name** の指定は有効ですが、両方を指定することはできません。

クロスドメイン ポリシーに対する変更

影響を受けるのは: crossdomain.xml と clientaccesspolicy.xml の使用者。

概要

クロスドメイン ポリシーはもはや（デフォルトでの）ヘッダーの送付をサポートしません。これは clientaccesspolicy.xml と crossdomain.xml ポリシー ファイルの両方に適用されます。リマインダーとしてクロスドメイン POST 要求だけはヘッダーを送付する場合があります。

必要な修正

クロスドメイン ポリシー上でヘッダーの送付を可能にするには、clientaccesspolicy.xml 内の allow-from タグに headers 属性を追加する必要があります：

clientaccesspolicy.xml

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
  <cross-domain-access>
    <policy>
      <allow-from http-request-headers="*" />
        <domain uri="*"/>
      </allow-from>
      <grant-to>
        <resource path="/" include-subpaths="true"/>
      </grant-to>
    </policy>
  </cross-domain-access>
</access-policy>
```

crossdomain.xml では allow-http-request-headers-from タグを使います：

crossdomain.xml

```
<?xml version="1.0"?>
```

```
<!DOCTYPE cross-domain-policy SYSTEM
  "http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
  <allow-http-request-headers-from domain="* " headers="*" />
</cross-domain-policy>
```

どちらのポリシーファイルでも、headers 属性として有効は値は以下のとおりです：

- "*" ワイルドカード
  全ての非ブラックリスト ヘッダーを許可
- カンマで区切った許可ヘッダーのリスト
  注意: これらのヘッダーはワイルドカード"*"接尾辞を持つかもしれません(ex. "X-CUSTOM-*")

現在のヘッダー ブラックリストは以下のとおりです：

- Accept-Charset
- Accept-Encoding
- Accept-Language
- Age
- Allow
- Allowed
- Authorization
- CONNECT
- Connection
- Content-Length
- Content-Location
- Content-Range
- Content-Transfer-Encoding
- Cookie
- Date
- Delete
- ETag
- Expect
- GET
- HEAD
- Host
- Keep-Alive
- Last-Modified
- Location
- Max-Forwards
- Options
- POST
- Proxy-Authenticate

- Proxy-Authorization
- Proxy-Connection
- Public
- PUT
- Range
- Referer
- Retry-After
- Request-Range
- Server
- TE
- User-Agent
- Trace
- Trailer
- Transfer-Encoding
- Upgrade
- URI
- Vary
- Via
- Warning
- WWW-Authenticate
- X-flash-version

ストーリーボードはライブツリーの外にあってもアクティブな場合がある

これは API の変更ではなく、どのコードもブレークしないはずです。これは 1.0 からふるまいが変更されただけです。これまで、Silverlight のストーリーボードとアニメーションはライブツリー内にあるときのみ **Active** 状態にありました。これが、**Storyboard** を使ってビジュアルの状態を維持する堅固なコントロールを書くことを困難にしていました。この変更で、開発者は **UIElement** の **Resources** や **Triggers** セクションに追加することなしに、**Storyboard** の Begin を呼び出すことができます。

Storyboard.Duration 上の GetValue は Storyboard.Duration だけを返す

Beta 1

GetValue(Storyboard.DurationProperty)は double 時間幅がある場合に double を返し、その値が **Forever** のとき文字列を返し、その値が **Automatic** のとき null を返しました。

Beta 2

GetValue(Storyboard.DurationProperty)は **Storyboard.Duration** だけを返します。

Image と ImageBrush クラスの変更

Beta 1

[C#]

```
// Image クラス:
    public event ExceptionRoutedEventHandler ImageFailed;
    public double DownloadProgress { get; set;}

// ImageBrush クラス:
    public event ExceptionRoutedEventHandler ImageFailed;
    public double DownloadProgress { get; set;}
```

Beta 2

[C#]

```
// Image クラス:
    public event EventHandler<ExceptionRoutedEventArgs> ImageFailed;
    Removed DownloadProgress property

// ImageBrush クラス:
    public event EventHandler<ExceptionRoutedEventArgs> ImageFailed;
    Removed DownloadProgress property
```

TextBox テンプレートの変更

TextBox テンプレートが更新されました。

- **ELEMENT_Content** は **ContentElement** になりました。
- マネージ アプリケーション内の **TextBox** デフォルト **ContentElement** は **Border** ではなく **ScrollViewer** になりました。

  注意　API 変更ではありませんが、この結果 TextBox のデフォルトの見え方が変化しました。これによって、ビットマップ比較によるテストの変更が必要になり、新しい見え方が他の UI と統合に影響を与えるかもしれません。

古いテンプレート

[xaml]

```
<ControlTemplate
  xmlns='http://schemas.microsoft.com/client/2007'
  xmlns:x='http://schemas.microsoft.com/winfx/2006/xaml'>
  <Border x:Name='ELEMENT_Content'"
    BorderThickness='{TemplateBinding BorderThickness}'
    BorderBrush='{TemplateBinding BorderBrush}'
    Background='{TemplateBinding Background}'
    Padding='{TemplateBinding Padding}'/>
</ControlTemplate>
```

新しいテンプレート

[xaml]

```
<ControlTemplate
  xmlns='http://schemas.microsoft.com/client/2007'
  xmlns:x='http://schemas.microsoft.com/winfx/2006/xaml'>
  <Grid x:Name='RootElement'>
    <Grid.Resources>
      <SolidColorBrush x:Key='Background' Color='#FF003255'/>

      <SolidColorBrush x:Key='BorderBrush' Color='#FF000000'/>
      <SolidColorBrush x:Key='AccentBrush' Color='#FFFFFFFF'/>

      <LinearGradientBrush x:Key='FocusedStrokeBrush'
        StartPoint='0.5,0' EndPoint='0.5,1'>
        <GradientStop Color='#B2FFFFFF' Offset='0'/>
        <GradientStop Color='#51FFFFFF' Offset='1'/>
        <GradientStop Color='#66FFFFFF' Offset='0.325'/>
        <GradientStop Color='#1EFFFFFF' Offset='0.325'/>
      </LinearGradientBrush>

      <Storyboard x:Key='Normal State'>
        <DoubleAnimation Storyboard.TargetName='FocusVisualElement'
          Storyboard.TargetProperty='Opacity' To='0'
Duration='0:0:0.0'/>
      </Storyboard>

      <Storyboard x:Key='Focused State'>
        <DoubleAnimation Storyboard.TargetName='FocusVisualElement'
          Storyboard.TargetProperty='Opacity' To='1'
Duration='0:0:0.0'/>
      </Storyboard>
    </Grid.Resources>

    <Rectangle StrokeThickness='.5' RadiusX='2'
      RadiusY='2' Fill='{TemplateBinding Background}'/>
```

```
    <Rectangle StrokeThickness='.5' RadiusX='2' RadiusY='2'
      Stroke='#FF003255'/>
    <Border x:Name='ContentElement' Padding='{TemplateBinding
Padding}'/>

    <Grid x:Name='FocusVisualElement' Opacity='0'
IsHitTestVisible='False'>
      <Rectangle RadiusX='1' RadiusY='1' Margin='2'
        Stroke='{StaticResource AccentBrush}' StrokeThickness='1'/>
      <Rectangle RadiusX='1' RadiusY='1'
        Stroke='{StaticResource Background}' StrokeThickness='2'/>
      <Rectangle RadiusX='1' RadiusY='1'
        Stroke='{StaticResource FocusedStrokeBrush}'
StrokeThickness='2'/>
    </Grid>
  </Grid>
</ControlTemplate>
```

カスタム BorderBrush/BorderThickness の変更

影響を受けるのは: BorderBrush/BorderThickness に独自のプロパティを持つカスタムコントロールは、この変更の影響を受けます。そのコードは Silverlight 2 Beta 2 でコンパイルできないでしょう。

概要

Control クラスに **BorderBrush** と **BorderThickness** プロパティが追加されました。

必要な修正

基本クラスのプロパティを使うだけです。それらのプロパティ用のテンプレートやテンプレートバインディングを変更する必要はないでしょう。

ButtonBase の変更

影響を受けるのは: ButtonBase クラスの使用者

概要

ButtonBase は部分抽象ではなく抽象クラスになりました。さらに、**ButtonBase** から以下のメンバが削除されました。:

- **OnIsEnabledChanged**
- **OnGotFocus**
- **OnIsPressedChanged**
- **IsFocused**
- **IsMouseOver**
- **IsMouseOverProperty**
- **IsFocusedProperty**

- **OnKeyDown**
- **OnKeyUp**
- **OnLostFocus**
- **OnMouseEnter**
- **OnMouseLeave**
- **OnMouseLeftButtonDown**
- **OnMouseLeftButtonUp**
- **OnMouseMove**

ListBox と ListBoxItem の変更

影響を受けるのは: ListBox や **ListBoxItem** クラスの使用者

概要

ListBox から以下のメンバが削除されました：

- **SelectedItemsProperty**
- **SelectionModeProperty**
- **SelectedItems**
- **SelectionMode**
- **OnGotFocus(RoutedEventArgs)**
- **OnItemContainerStyleChanged(Style, Style)**
- **OnKeyDown(KeyEventArgs)**
- **OnLostFocus(RoutedEventArgs)**
- **OnSelectedItemsChanged(IList, IList)**
- **OnIsSelectionActiveChanged(Boolean, Boolean)**
- **OnSelectedIndexChanged(Int32, Int32)**
- **OnSelectedItemChanged(Object, Object)**
- **OnSelectionChanged(SelectionChangedEventArgs)**
- **OnSelectionModeChanged(SelectionMode, SelectionMode)**

ListBoxItem から以下のメンバが削除されました：

- **OnGotFocus(RoutedEventArgs)**
- **OnLostFocus(RoutedEventArgs)**
- **OnMouseEnter(MouseEventArgs)**
- **OnMouseLeave(MouseEventArgs)**
- **OnMouseLeftButtonDown(MouseButtonEventArgs)**
- **IsMouseOver**

さらに、**SelectionMode** 列挙子も削除されました。

System.ServiceModel.ClientBase の変更

影響を受けるのは: System.ServiceModel.ClientBase.Open()と **Close()**、あるいは **System.ServiceModel.ClientBase.Dispose()**の使用者。

概要

System.ServiceModel.ClientBase.Open()と **Close()**と **System.ServiceModel.ClientBase.Dispose()**は削除されました。

必要な修正

Open()を使用したコードの対応策：

- [推奨] 呼び出さない：最初のウェブサービス呼び出しで、我々は自動的に Open を非同期で行うので、明示的にそれを呼び出す必要はありません。
- [推奨] OpenAsync と OpenCompleted イベントを使用したコードに変更する。
- [準推奨] プロキシを ICommunicationObject に型変換し、BeginOpen/EndOpen を使用。
- [非推奨] プロキシを ICommunicationObject に型変換し、Open()を使用。

Close()を使用したコードの対応策：

- [推奨] CloseAsync を使用したコードに変更する。Colse が完了したときの通知は通常不要なので、CloseCompleted イベントは通常必要でない。
- [準推奨] Close をまったく呼び出さない。プロキシに使われたリソースはやがて再利用されます。
- [準推奨] プロキシを ICommunicationObject に型変換し、BeginClose/EndClose を使用。
- [非推奨] プロキシを ICommunicationObject に型変換し、Close()を使用。

Dispose()や‘using’ C# パターンを使用したコードの対応策：

- [推奨] CloseAsync 呼び出しにコードを変更。
- [準推奨] 上述の Close()に関する代替を参照。

ContentControl と ContentPresenter の変更

影響を受けるのは: ContentControl や **ContentPresenter** クラスの使用者

概要

ContentControl から以下のメンバが削除されました：

- **HasContent**
- **ContentTemplateSelector**
- **LogicalChildren**
- **OnContentTemplateSeleectorChanged(Style, Style)**
- **OnContentTemplateChanged(Style, Style)**
- **FontFamily**
- **FontSize**
- **FontStretch**
- **FontStyle**
- **FontWeight**
- **Foreground**
- **IsEnabled**
- **OnIsEnabledChanged (Boolean)**
- **Background**
- **Tooltip**
- **TextAlignment**
- **TextDecorations**

- **TextWrapping**

ContentPresenter から以下のメンバーが削除されました：

- **ContentSource** - **String** return type.
- **ContentSource** - **Template** return type
- **ContentTemplateSelector**
- **ContentSourceProperty**
- **ContentTemplateSelectorProperty**
- **ChooseTemplate()**
- **OnContentTemplateChanged (DataTemplate, DataTemplate)**
- **OnTemplateChanged (DataTemplate, DataTemplate)**
- **OnContentTemplateSeleectorChanged(Style, Style)**
- **Background**
- **FontFamily**
- **FontSize**
- **FontStretch**
- **FontStyle**
- **FontWeight**
- **Foreground**
- **IsEnabled**
- **OnIsEnabledChanged (Boolean)**

GetValue の変更

- **GetValue(ArcSegment.SizeProperty)**は基になっている **Point** 型のオブジェクトを返すようになりました。
- **GetValue(DesignerProperties.IsInDesignModeProperty)**は基になっている **bool** 型のオブジェクトを返すようになりました。
- **GetValue(FrameworkElement.CursorProperty)**は基になっている **CursorType** 型のオブジェクトを返すようになりました。
- **GetValue(FrameworkElement.LanguageProperty)**は基になっている **XmlLanguage** 型のオブジェクトを返すようになりました。
- **GetValue(FontFamilyProperty)**は基になっている **FontFamily** 型のオブジェクトを返すようになりました。
- **GetValue(FontWeightProperty)**は基になっている **FontWeightType** 型のオブジェクトを返すようになりました。
- **GetValue(FontStyleProperty)**は基になっている **FontStyleType** 型のオブジェクトを返すようになりました。
- **GetValue(FontStretchProperty)**は基になっている **FontStretchType** 型のオブジェクトを返すようになりました。
- **GetValue(TextDecorationsProperty)**は基になっている **TextDecorationCollection** 型のオブジェクトを返すようになりました。
- **GetValue(Inline.LanguageProperty)**は基になっている **XmlLanguage** 型のオブジェクトを返すようになりました。

- **GetValue(MatrixProperty)**は基になっている **Matrix** 型のオブジェクトを返すようになりました。
- **GetValue(TextBox.TextProperty)**は基になっている **String** 型のオブジェクトを返すようになりました。
- **GetValue(TextBox.SelectedTextProperty)**は基になっている **String** 型のオブジェクトを返すようになりました。

コントロール スタイルの更新

VisualStateManager 用の新しいスタイルモデルのために変更が行われ、ツールとの互換性、サイズ、性能が改善されました。

ScrollBar から削除されたもの

[TemplatePart(Name = ScrollBar.StateNormalName, Type = typeof(Storyboard))]

[TemplatePart(Name = ScrollBar.StateMouseOverName, Type = typeof(Storyboard))]

[TemplatePart(Name = ScrollBar.StateDisabledName, Type = typeof(Storyboard))]

ScrollBar に追加されたもの

[TemplateVisualState(Name = ScrollBar.StateNormal, GroupName = ScrollBar.GroupCommon)]

[TemplateVisualState(Name = ScrollBar.StateMouseOver, GroupName = ScrollBar.GroupCommon)]

[TemplateVisualState(Name = ScrollBar.StateDisabled, GroupName = ScrollBar.GroupCommon)]

Button から削除されたもの

[TemplatePart(Name = Button.ElementRootName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = Button.ElementFocusVisualName, Type = typeof(UIElement))]

[TemplatePart(Name = Button.StateNormalName, Type = typeof(Storyboard))]

[TemplatePart(Name = Button.StateMouseOverName, Type = typeof(Storyboard))]

[TemplatePart(Name = Button.StatePressedName, Type = typeof(Storyboard))]

[TemplatePart(Name = Button.StateDisabledName, Type = typeof(Storyboard))]

Button に追加されたもの

[TemplatePart(Name = Button.ElementFocusVisual, Type = typeof(UIElement))]

[VisualState(Name = Button.StateNormal, GroupName = Button.GroupCommon)]

[VisualState(Name = Button.StateMouseOver, GroupName = Button.GroupCommon)]

[VisualState(Name = Button.StatePressed, GroupName = Button.GroupCommon)]

[VisualState(Name = Button.StateDisabled, GroupName = Button.GroupCommon)]

CheckBox から削除されたもの

[TemplatePart(Name = CheckBox.ElementRootName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = CheckBox.ElementFocusVisualName, Type = typeof(UIElement))]

[TemplatePart(Name = CheckBox.ElementContentFocusVisualName, Type = typeof(UIElement))]

[TemplatePart(Name = CheckBox.StateCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StateIndeterminateName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StateNormalName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StateMouseOverCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StateMouseOverIndeterminateName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StateMouseOverUncheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StatePressedCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StatePressedIndeterminateName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StatePressedUncheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StateDisabledCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StateDisabledIndeterminateName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.StateDisabledUncheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = CheckBox.ElementFocusVisual, Type = typeof(UIElement))]

[TemplatePart(Name = CheckBox.ElementContentFocusVisual, Type = typeof(UIElement))]

CheckBox に追加されたもの

[VisualState(Name = CheckBox.StateNormal, GroupName = CheckBox.GroupCommon)]

[VisualState(Name = CheckBox.StateMouseOver, GroupName = CheckBox.GroupCommon)]

[VisualState(Name = CheckBox.StatePressed, GroupName = CheckBox.GroupCommon)]

[VisualState(Name = CheckBox.StateDisabled, GroupName = CheckBox.GroupCommon)]

[VisualState(Name = CheckBox.StateChecked, GroupName = CheckBox.GroupCheck)]

[VisualState(Name = CheckBox.StateUnchecked, GroupName = CheckBox.GroupCheck)]

[VisualState(Name = CheckBox.StateIndeterminate, GroupName = CheckBox.GroupCheck)]

HyperlinkButton から削除されたもの

[TemplatePart(Name = HyperlinkButton.ElementRootName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = HyperlinkButton.ElementFocusVisualName, Type = typeof(UIElement))]

[TemplatePart(Name = HyperlinkButton.StateNormalName, Type = typeof(Storyboard))]

[TemplatePart(Name = HyperlinkButton.StateMouseOverName, Type = typeof(Storyboard))]

[TemplatePart(Name = HyperlinkButton.StatePressedName, Type = typeof(Storyboard))]

[TemplatePart(Name = HyperlinkButton.StateDisabledName, Type = typeof(Storyboard))]

HyperlinkButton に追加されたもの

[TemplatePart(Name = HyperlinkButton.ElementFocusVisual, Type = typeof(UIElement))]

[VisualState(Name = HyperlinkButton.StateNormal, GroupName = HyperlinkButton.GroupCommon)]

[VisualState(Name = HyperlinkButton.StateMouseOver, GroupName = HyperlinkButton.GroupCommon)]

[VisualState(Name = HyperlinkButton.StatePressed, GroupName = HyperlinkButton.GroupCommon)]

[VisualState(Name = HyperlinkButton.StateDisabled, GroupName = HyperlinkButton.GroupCommon)]

ListBoxItem から削除されたもの

[TemplatePart(Name = ListBoxItem.ElementRootName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = ListBoxItem.ElementFocusVisualName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = ListBoxItem.StateNormalName, Type = typeof(Storyboard))]

[TemplatePart(Name = ListBoxItem.StateSelectedName, Type = typeof(Storyboard))]

[TemplatePart(Name = ListBoxItem.StateSelectedFocusedName, Type = typeof(Storyboard))]

[TemplatePart(Name = ListBoxItem.StateMouseOverName, Type = typeof(Storyboard))]

[TemplatePart(Name = ListBoxItem.StateMouseOverSelectedName, Type = typeof(Storyboard))]

[TemplatePart(Name = ListBoxItem.StateMouseOverSelectedFocusedName, Type = typeof(Storyboard))]

ListBoxItem に追加されたもの

[VisualState(Name = ListBoxItem.StateNormal, GroupName = ListBoxItem.GroupCommon)]

[VisualState(Name = ListBoxItem.StateMouseOver, GroupName = ListBoxItem.GroupCommon)]

[VisualState(Name = ListBoxItem.StateUnselected, GroupName = ListBoxItem.GroupSelection)]

[VisualState(Name = ListBoxItem.StateSelected, GroupName = ListBoxItem.GroupSelection)]

[VisualState(Name = ListBoxItem.StateSelectedUnfocused, GroupName = ListBoxItem.GroupSelection)]

RadioButton から削除されたもの

[TemplatePart(Name = RadioButton.ElementFocusVisual, Type = typeof(UIElement))]

[TemplatePart(Name = RadioButton.ElementContentFocusVisual, Type = typeof(UIElement))]

[TemplatePart(Name = RadioButton.ElementRootName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = RadioButton.ElementFocusVisualName, Type = typeof(UIElement))]

[TemplatePart(Name = RadioButton.ElementContentFocusVisualName, Type = typeof(UIElement))]

[TemplatePart(Name = RadioButton.StateCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = RadioButton.StateNormalName, Type = typeof(Storyboard))]

[TemplatePart(Name = RadioButton.StateMouseOverCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = RadioButton.StateMouseOverUncheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = RadioButton.StatePressedCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = RadioButton.StatePressedUncheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = RadioButton.StateDisabledCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = RadioButton.StateDisabledUncheckedName, Type = typeof(Storyboard))]

RadioButton に追加されたもの

[VisualState(Name = RadioButton.StateNormal, GroupName = RadioButton.GroupCommon)]

[VisualState(Name = RadioButton.StateMouseOver, GroupName = RadioButton.GroupCommon)]

[VisualState(Name = RadioButton.StatePressed, GroupName = RadioButton.GroupCommon)]

[VisualState(Name = RadioButton.StateDisabled, GroupName = RadioButton.GroupCommon)]

[VisualState(Name = RadioButton.StateChecked, GroupName = RadioButton.GroupCheck)]

[VisualState(Name = RadioButton.StateUnchecked, GroupName = RadioButton.GroupCheck)]

RepeatButton から削除されたもの

[TemplatePart(Name = RepeatButton.ElementRootName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = RepeatButton.ElementFocusVisualName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = RepeatButton.StateNormalName, Type = typeof(Storyboard))]

[TemplatePart(Name = RepeatButton.StateMouseOverName, Type = typeof(Storyboard))]

[TemplatePart(Name = RepeatButton.StatePressedName, Type = typeof(Storyboard))]

[TemplatePart(Name = RepeatButton.StateDisabledName, Type = typeof(Storyboard))]

RepeatButton に追加されたもの

[TemplatePart(Name = RepeatButton.ElementRoot, Type = typeof(FrameworkElement))]

[TemplatePart(Name = RepeatButton.ElementFocusVisual, Type = typeof(FrameworkElement))]

[VisualState(Name = RepeatButton.StateNormal, GroupName = RepeatButton.GroupCommon)]

[VisualState(Name = RepeatButton.StateMouseOver, GroupName = RepeatButton.GroupCommon)]

[VisualState(Name = RepeatButton.StatePressed, GroupName = RepeatButton.GroupCommon)]

[VisualState(Name = RepeatButton.StateDisabled, GroupName = RepeatButton.GroupCommon)]

ScrollBar から削除されたもの

[TemplatePart(Name = ScrollBar.ElementRootName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = ScrollBar.StateNormalName, Type = typeof(Storyboard))]

[TemplatePart(Name = ScrollBar.StateMouseOverName, Type = typeof(Storyboard))]

[TemplatePart(Name = ScrollBar.StateDisabledName, Type = typeof(Storyboard))]

ScrollBar に追加されたもの

[VisualState(Name = ScrollBar.StateNormal, GroupName = ScrollBar.GroupCommon)]

[VisualState(Name = ScrollBar.StateMouseOver, GroupName = ScrollBar.GroupCommon)]

[VisualState(Name = ScrollBar.StateDisabled, GroupName = ScrollBar.GroupCommon)]

Thumb から削除されたもの

[TemplatePart(Name = Thumb.ElementRootName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = Thumb.StateNormalName, Type = typeof(Storyboard))]

[TemplatePart(Name = Thumb.StateMouseOverName, Type = typeof(Storyboard))]

[TemplatePart(Name = Thumb.StatePressedName, Type = typeof(Storyboard))]

[TemplatePart(Name = Thumb.StateDisabledName, Type = typeof(Storyboard))]

Thumb に追加されたもの

[VisualState(Name = Thumb.StateNormal, GroupName = Thumb.GroupCommon)]

[VisualState(Name = Thumb.StateMouseOver, GroupName = Thumb.GroupCommon)]

[VisualState(Name = Thumb.StatePressed, GroupName = Thumb.GroupCommon)]

[VisualState(Name = Thumb.StateDisabled, GroupName = Thumb.GroupCommon)]

ToggleButton から削除されたもの

[TemplatePart(Name = ToggleButton.ElementRootName, Type = typeof(FrameworkElement))]

[TemplatePart(Name = ToggleButton.ElementFocusVisualName, Type = typeof(UIElement))]

[TemplatePart(Name = ToggleButton.ElementContentFocusVisualName, Type = typeof(UIElement))]

[TemplatePart(Name = ToggleButton.StateNormalName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StateCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StateIndeterminateName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StateMouseOverCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StateMouseOverIndeterminateName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StateMouseOverUncheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StatePressedCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StatePressedIndeterminateName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StatePressedUncheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StateDisabledCheckedName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StateDisabledIndeterminateName, Type = typeof(Storyboard))]

[TemplatePart(Name = ToggleButton.StateDisabledUncheckedName, Type = typeof(Storyboard))]

ToggleButton に追加されたもの

[TemplatePart(Name = ToggleButton.ElementFocusVisual, Type = typeof(UIElement))]

[TemplatePart(Name = ToggleButton.ElementContentFocusVisual, Type = typeof(UIElement))]

[VisualState(Name = ToggleButton.StateNormal, GroupName = ToggleButton.GroupCommon)]

[VisualState(Name = ToggleButton.StateMouseOver, GroupName = ToggleButton.GroupCommon)]

[VisualState(Name = ToggleButton.StatePressed, GroupName = ToggleButton.GroupCommon)]

[VisualState(Name = ToggleButton.StateDisabled, GroupName = ToggleButton.GroupCommon)]

[VisualState(Name = ToggleButton.StateChecked, GroupName = ToggleButton.GroupCheck)]

[VisualState(Name = ToggleButton.StateUnchecked, GroupName = ToggleButton.GroupCheck)]

DataGrid の断層的変更

以下は DataGrid に関する多数の断層的変更です。

AutoGenerateColumns のふるまいの修正

影響を受けるのは: DataGrid を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

自働列生成のふるまいは劇的に修正されました。以下の違いを参照してください。

必要な修正

DataGrid を使用するユーザーは、コードを修正して dataGrid.AutoGenerateColumns をオンにしたりオフにしたりする必要があるかもしれません。さらに、自動生成によってインスタンス化された列に依存したユーザーは、コードを変更して、Loaded イベントを待ったり、AutogeneratingColumn イベントを使う必要があるかもしれません。

Beta 1

- dataGrid.AutoGenerateColumns はデフォルトで false。
- DataGrid は列を ItemSource ごとに即座に一度生成。
- AutoGenerateColumns を true から false に設定しても、既存の列に影響を与えない。
- AutoGenerateColumns を false から true に設定すると、その ItemsSource から列がまだ生成されていないとき、列を生成する。
- DataGrid が列を自動生成するとき、全ての既存の列は、それらが自動生成されたかどうかにかかわらず新しい列を生成する前に、最初にクリアされる。
- 自動生成された列はユーザーの列と違いはない。

Beta 2

- dataGrid.AutoGenerateColumns はデフォルトで true。

- 列は、自動生成された列のとき ture を返す IsAutoGenerated 読み取り専用のパブリック フラグを持つ。
- DataGrid がロードされる前に以下のシーケンスでの列生成を避けるため、DataGrid がロードされるまで、列の生成は遅れる：
  - dataGrid.ItemsSource = myList;
  - dataGrid.AutoGenerateColumns = false;
- 実行時に AutoGenerateColumns を true から false に設定すると、生成された列は列コレクションから削除される。
- 実行時に AutoGenerateColumns を false から true に設定すると、DataGrid は必ず列を再生成する。
- DataGrid が列を自動生成するとき、既存の自動生成済みの列は最初に列コレクションから削除される。新しく生成された列はユーザーが定義した列の後の列の末尾に追加される。

DependencyProperty DataGrid.SelectedItemsProperty は削除された

影響を受けるのは: DataGrid.SelectedItemsProperty 依存プロパティを使用する Silverlight 2 マネージ アプリケーション。

概要

SelectedItems はコレクションのメソッドで修正できますが、プロパティ自身は読み取り専用プロパティです。Beta 1 では依存プロパティによって保持されており、そのプロパティが設定されると InvalidOperationException を投げました。それを行う目的は、WPF ListBox から来たものですが、WPF に話す中で、ListBox が行っていることが間違っていることを確認しました。それは依存プロパティではなく、読み取り専用 CLR プロパティになりました。

必要な修正

SelectedItems はもはや依存プロパティとして使うことができません。

PreparingRow/CleaningRow は LoadingRow/UnloadingRow に名前が変更

影響を受けるのは: dataGrid.PreparingRow や dataGrid.CleaningRow を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

PreparingRow イベントと CleaningRow イベントは LoadingRow イベントと UnloadingRow イベントに名前が変更されました。さらに CleaningRow イベントはもはやキャンセルできません。

必要な修正

PreparingRow と CleaningRow の代わりに LoadingRow と UnloadingRow を使う。

Beta 1

[C#]

```
dataGrid.PreparingRow += new
EventHandler<DataGridRowEventArgs>(dataGrid_PreparingRow);
```

```
dataGrid.CleaningRow += new
EventHandler<DataGridRowCancelEventArgs>(dataGrid_CleaningRow);
```

Beta 2

[C#]

```
dataGrid.LoadingRow += new
EventHandler<DataGridRowEventArgs>(dataGrid_LoadingRow);
dataGrid.UnloadingRow += new
EventHandler<DataGridRowEventArgs>(dataGrid_UnloadingRow);
```

DataGridColumnBase は DataGridColumn に名前が変更

影響を受けるのは: DataGridColumnBase 型を明示的に使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGridColumnBase は DataGridColumn に名前が変更されました。

必要な修正

DataGridColumnBase を明示的に使用している場合は名前を DataGridColumn に変更。

Beta 1

[C#]

```
DataGridColumnBase column = dataGrid.Columns[0];
```

Beta 2

[C#]

```
DataGridColumn column = dataGrid.Columns[0];
```

DataGridBoundColumnBase は DataGridBoundColumn に名前が変更

影響を受けるのは: DataGridBoundColumnBase 型を明示的に使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGridBoundColumnBase は DataGridBoundColumn に名前が変更になりました。

必要な修正

DataGridBoundColumnBase を明示的に使用している場合は名前を DataGridBoundColumn に変更。

Beta 1

[C#]

```
DataGridBoundColumnBase boundColumn = dataGrid.Columns[0] as
DataGridBoundColumnBase;
```

Beta 2

[C#]

```
DataGridBoundColumn boundColumn = dataGrid.Columns[0] as
DataGridBoundColumn;
```

DataGridTextBoxColumn は DataGridTextColumn に名前が変更

影響を受けるのは: DataGridTextBoxColumn 型を明示的に使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGridTextBoxColumn は DataGridTextColumn に名前が変更になりました。

必要な修正

DataGridTextBoxColumn を明示的に使用している場合は名前を DataGridTextColumn に変更。

Beta 1

[C#]

```
((DataGridTextBoxColumn)dataGrid.Columns[0]).FontSize = 11;
```

Beta 2

[C#]

```
((DataGridTextColumn)dataGrid.Columns[0]).FontSize = 11;
```

HeaderTemplate プロパティは削除されました

影響を受けるのは: dataGrid.HeaderTemplate プロパティを使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

RowHeaders や ColumnHeaders に適用可能な HeaderTemplate プロパティは削除されました。

必要な修正

RowHeaders や ColumnHeaders に対し HeaderTemplate を以前使用していたユーザーは、dataGrid.RowHeaderStyle を使って RowHeaders 用のテンプレートを設定するか、dataGrid.ColumnHeaderStyle を使って ColumnHeaders 用のテンプレートを設定します。それ以外にも、ユーザーは個別の RowHeaders や ColumnHeaders 上のスタイルやテンプレートを明示的に設定しても構いません。

RowDetailsVisibility は RowDetailsVisibilityMode に名前が変更

影響を受けるのは: dataGrid.RowDetailsVisibility 列挙子を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

RowDetailsVisibility は RowDetailsVisibilityMode に名前が変更になり、それが何かをよりよく反映し、Visibility 型でないことを示します。

必要な修正

RowDetailsVisibility を使用している場合は RowDetailsVisibilityMode に名前を変更する必要があります。

Beta 1

[C#]

```
dataGrid.RowDetailsVisibility =
RowDetailsVisibility.VisibleWhenSelected;
```

Beta 2

[C#]

```
dataGrid.RowDetailsVisibilityMode =
RowDetailsVisibilityMode.VisibleWhenSelected;
```

CheckBoxContentBinding は Content 型に変更

影響を受けるのは: dataGridCheckBoxColumn.CheckBoxContentBinding を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

この場合、バインディングは不要です、我々はこれを、Content と名付けたオブジェクトに変更し、列内の CheckBoxes の Content プロパティにマップしました。

必要な修正

Users using CheckBoxContentBinding を使用するユーザーは Content 自身をバインディングする必要があります。

Beta 1

[C#]

```
dataGridCheckBoxColumn.CheckBoxContentBinding = new
Binding(“checkBoxContent”);
```

Beta 2

[C#]

```
dataGridCheckBoxColumn.Content = checkBoxContent;
```

GetElement は GetCellContent に名前が変更

影響を受けるのは: DataGridColumnBase.GetCellContent を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

与えられた DataGridRow や DataItem のセルからコンテンツを取得する DataGridColumn 上の GetElement メソッドは GetCellContent に名前が変更になりました。

必要な修正

User calling GetElement を呼び出すユーザーは GetCellContent の呼び出しに変更する必要があります。

Beta 1

[C#]

```
dataGrid.Columns[0].GetElement(myItems[0]);
```

Beta 2

[C#]

```
dataGrid.Columns[0].GetCellContent(myItems[0]);
```

UpdateElement は RefreshCellContent に名前が変更

影響を受けるのは: カスタム列あるいはテンプレート列を持つ DataGrid を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGridColumnBase 上の UpdateElement メソッドは RefreshCellContent に名前が変更になりました。

必要な修正

その列内で UpdateElement をオーバーライドしていたユーザーは、代わりに RefreshCellContent をオーバーライドする必要があります。

Beta 1

[C#]

```
public MyColumn : DataGridBoundColumnBase
{
      public override void UpdateElement(Frameworkelement element,
string propertyName) {…}
}
```

Beta 2

[C#]

```
public MyColumn : DataGridBoundColumn
{
```

```
      public override void RefreshCellContent(Frameworkelement element,
string propertyName) {…}
}
```

UpdateElements は NotifyPropertyChanged に名前が変更

影響を受けるのは: 独自のカスタム列を持つ DataGrid を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGridBoundColumnBase 上の UpdateElement メソッドは NotifyPropertyChanged に名前が変更になりました。

必要な修正

カスタム列内で UpdateElement メソッドを使用していたユーザーは NotifyPropertyChanged を使う必要があります。

Beta 1

```
[C#]
public MyColumn : DataGridBoundColumnBase
{
        public double FontSize
        {
            get {…}
            set
            {
                    this._fontSize = value;
                    UpdateElements(“FontSize”);
            }
        }
}
```

Beta 2

```
[C#]
public MyColumn : DataGridBoundColumn
{
        public double FontSize
        {
            get {…}
            set
            {
```

```
                    this._fontSize = value;
                    NotifyPropertyChanged("FontSize");
                }
            }
}
```

GenerateElement と GenerateEditingElement に dataItem パラメータが追加

影響を受けるのは: カスタム バインド列を持つ DataGrid を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGridBoundColumnBase 上の GenerateElement メソッドと GenerateEditingElement メソッドはその要素が入る列とバインドされた dataItem をとるようになりました。

必要な修正

カスタム バインド列内の GenerateElement や GenerateEditingElement をオーバーライドするユーザーは、dataItem を追加する必要があります。

Beta 1

```
[C#]
public MyColumn : DataGridBoundColumnBase
{
      public override void GenerateElement() {…}
      public override void GenerateEditingElement() {…}
}
```

Beta 2

```
[C#]
public MyColumn : DataGridBoundColumn
{
      public override void GenerateElement(object dataItem) {…}
      public override void GenerateEditingElement(object dataItem) {…}
}
```

DataGridBoundColumnBase 上の Index はインターナルになった

影響を受けるのは: dataGridColumnBase.Index プロパティを使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGridColumns 用に Index と DisplayIndex の両方を公開する必要はありません。Index は細部の実装なのでインターナルになりました。

必要な修正

DataGridColumns のために Index プロパティはもはや使えません。列の並べ替えをしたり、DataItem 内のどのフィールドがその列とバインドされているかを知りたいとき DisplayMemberBinding にアクセスするには、DisplayIndex を使うべきです。

dataGrid.GetRowDetailsVisibility は DataGridRow.GetRowContainingElement に変更

影響を受けるのは: DataGrid 上の GetRowDetailsVisibility メソッドを使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGrid 上の行内部の要素を与えたとき行の行細部の Visibility を返すメソッドの代わりに、行内部の要素を与えたとき行を返すより柔軟なメソッドを作成しました。この個別シナリオのためには、その行の DetailsVisibility をチェックすればよいでしょう。

必要な修正

Users using dataGrid.GetRowDetailsVisibility を使用するユーザーは以下のように呼び出しを更新する必要があります。

Beta 1

```
[C#]
Visibility detailsVisibility =
dataGrid.GetRowDetailsVisibility(clickedButton);
```

Beta 2

```
[C#]
DataGridRow clickedRow =
DataGridRow.GetRowContainingElement(clickedButton);
Visibility detailsVisibility = clickedRow.DetailsVisibility;
```

列の幅は double から DataGridLength に変更

影響を受けるのは: XAML 外で dataGridColumnBase.Width を設定する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

列幅用にさまざまなサイズ操作をサポートするために、dataGridColumnBase.Width を double から DataGridLength に変更しました。

必要な修正

XAML 外で dataGridColumnBase.Width 設定を使用するユーザーは、double に設定する代わりに、DataGridLength をインスタンス化するか、静的 DataGridLengths(DataGridLength.Auto, DataGridLength.SizeToHeader, DataGridLength.SizeToCells)の 1 つを使用します。

Beta 1

[C#]

```
dataGrid.Columns[0].Width = 20;
```

[Xaml]

```
<DataGridTextBoxColumn Width=”20” DisplayMemeberBinding=”{Binding foo}”
/>
```

Beta 2

[C#]

```
dataGrid.Columns[0].Width = new DataGridLength(20);
dataGrid.Columns[0].Width = DataGridLength.Auto;
dataGrid.Columns[0].Width = DataGridLength.SizeToHeader;
dataGrid.Columns[0].Width = DataGridLength.SizeToCells;
```

[Xaml]

```
<DataGridTextColumn Width=”20” DisplayMemeberBinding=”{Binding foo}” />
<DataGridTextColumn Width=”Auto” DisplayMemeberBinding=”{Binding bar}”
/>
<DataGridTextColumn Width=”SizeToHeader”
DisplayMemeberBinding=”{Binding hello}” />
<DataGridTextColumn Width=”SizeToCells” DisplayMemeberBinding=”{Binding
hello}” />
```

OverrideRowDetailsScrolling は AreRowDetailsFrozen に名前が変更

影響を受けるのは: dataGrid.OverrideRowDetailsScrolling を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGrid 上の OverrideRowDetailsScrolling プロパティは AreRowDetailsFrozen に名前が変更されました。

必要な修正

OverrideRowDetailsScrolling を使用するユーザーはその名前を AreRowDetailsFrozen に変更する必要があります。

Beta 1

[C#]

```
dataGrid.OverrideRowDetailsScrolling = true;
```

Beta 2

[C#]

```
dataGrid.AreRowDetailsFrozen = true;
```

ColumnHeadersHeight は ColumnHeaderHeight に名前が変更

影響を受けるのは: dataGrid.ColumnHeadersHeight を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGrid 上の ColumnHeadersHeight プロパティは ColumnHeaderHeight に名前が変更されました。

必要な修正

ColumnHeadersHeight を使用するユーザーはその名前を ColumnHeaderHeight に変更する必要があります。

Beta 1

[C#]

```
dataGrid.ColumnHeadersHeight = 25;
```

Beta 2

[C#]

```
dataGrid.ColumnHeaderHeight = 25;
```

RowHeadersWidth は RowHeaderWidth に名前が変更

影響を受けるのは: dataGrid.RowHeadersWidth を使用する Silverlight 2 Beta 1 マネージ アプリケーション。

概要

DataGrid 上の RowHeadersWidth プロパティは RowHeaderWidth に名前が変更されました。

必要な修正

RowHeadersWidth を使用するユーザーはその名前を RowHeaderWidth に変更する必要があります。

Beta 1

[C#]

```
dataGrid.RowHeadersWidth = 30;
```

Beta 2

[C#]

```
dataGrid.RowHeaderWidth = 30;
```

Editing API の変更

影響を受けるのは: DataGrid を使用し、Editing インターフェイスをフックする Silverlight 2 Beta 1 マネージ アプリケーション。

概要

Editing インターフェイスは劇的に変更されました、以下を参照してください。

必要な修正

DataGrid を使用し、Editing インターフェイスをフックするユーザーは新しい Editing API を基にしたコードに更新する必要があります。

Beta 1

以下の DataGrid Beta 1 Editing API はそのあとの Beta 2 API に置き換わりました。

```
public class DataGrid : Control
{
     // イベント
     public event EventHandler<DataGridCellEditingCancelEventArgs>
BeginningCellEdit;
     public event EventHandler<DataGridCellEventArgs> CommitCellEdit;
     public event EventHandler<DataGridCellCancelEventArgs>
CommittingCellEdit;
     public event EventHandler<DataGridRowCancelEventArgs>
CommittingRowEdit;

     protected virtual void
OnBeginningCellEdit(DataGridCellEditingCancelEventArgs e);
     protected virtual void OnCommitCellEdit(DataGridCellEventArgs e);
     protected virtual void
OnCommittingCellEdit(DataGridCellCancelEventArgs e);
     protected virtual void
OnCommittingRowEdit(DataGridRowCancelEventArgs e);
}
```

Beta 2

新しい Beta 2 API は以下のとおりです：

```
public class DataGrid : Control
{
      // イベント
      public event EventHandler<DataGridBeginningEditEventArgs>
BeginningEdit;
      public event EventHandler<DataGridEndingEditEventArgs>
CancelingEdit;
```

```
        public event EventHandler<DataGridEndingEditEventArgs>
CommittingEdit;


        protected virtual void OnBeginningEdit(DataGridEditingEventArgs
e);
        protected virtual void
OnCommittingEdit(DataGridEndingEditEventArgs  e);
        protected virtual void
OnCancelingEdit(DataGridEndingEditEventArgs e);
}
public enum DataGridEditingUnit
{
        Cell,
        Row
}
```

DataGrid クラス用のテンプレートの変更

影響を受けるのは: DataGrid や関連コンポーネントを再テンプレートする Silverlight 2 Beta 1 マネージ アプリケーション。

概要

Beta 1 から Beta 2 で DataGrid と関連クラス用のデフォルト テンプレートが変更されました。

必要な修正

DataGrid を再テンプレートするユーザーは、新しい Beta 2 テンプレートを使ってそのテンプレートを基にする必要があります。

その他の API の変更

- **System.IO.IsolatedStorage.IsolatedStorageFile.TryIncreaseQuotaTo(Int64)**は **System.IO.IsolatedStorage.IsolatedStorageFile.IncreaseQuotaTo(Int64)** に名前が変更になりました。この変更の理由は、この API は Try パターンを使わないので、「Try」接頭辞を使うべきではないからです。
- **System.ApplicationException** は削除されました。これは「廃止された」型であり、Silverlight に引き継がれるべきではありません。
- **System.Char.ToLowerInvariant(Char)** は削除されました。 この変更の理由は、String API との一貫性のためです、代替は **System.Char.ToLower(Char, CultureInfo.InvariantCulture)**です。
- **ButtonBase** クラスは、WPF との互換のために抽象クラスになりました。
- **VisualTreeHelper** は **System.Windows** に **System.Windows.Media** 移動しました。
- **AutomationPeer** は抽象メンバを持つ抽象クラスになりました。

- **AutomationPeer IsColumnHeader****と **IsRowHeader****メソッドは削除されました。
- **UIElement.OnCreateAutomationPeer** はプロテクト仮想クラスになりました。
- **AutomationPeer GetPattern** はプロテクト抽象クラスからパブリック抽象クラスになりました。
- **Color.Add** は削除されました。
- **Color.Multiply** は削除されました。
- **ErrorType** 列挙子は削除されました。
- **GetValue** は **string** ではなく **BitmapImage** を返すようになりました。
- **Register** メソッドの最後の引数の名前は"**propertyMetadata**"から"**typeMetadata**"に変更になりました。
- **FrameworkElement.Parent** の戻り型は **System.Object** から **System.Windows.DependencyObject** に変更になりました。
- **SetBinding** は void ではなく **BindingExpressionBase** を返すようになりました。
- **SupportedCulture** と **SupportedCultureCollection** クラスはもはや使われないため、削除されました。**Deployment.SupportedCultures** が最近削除されたので使われません。
- **ItemsControl.ItemsHost** はプライベートになりました。
- **Application.RootVisual** は型として **UIElement** を取るようになりました。これまでは **DependancyObject** を取っていました。
- **Control.OnApplyTemplate()**はパブリックになりました。これまではプロテクトでした。
- **PathGeometry.Figures** と **PathFigure.Segments** は、初期化時に null ではなく空のコレクションです。
- **SourceDownloadEventArgs** は削除されました。
- **EventTrigger.Actions** はパブリックではなく、プライベート依存プロパティになりました。
- **EventTrigger.RoutedEvent** はパブリックではなく、プライベート依存プロパティになりました。
- **FrameworkElement.Triggers** はパブリックではなく、プライベート依存プロパティになりました。
- **ObjectAnimationUsingKeyFrames.KeyFrames** はパブリックではなく、プライベート依存プロパティになりました。
- **Border.Child** はパブリックではなく、プライベート依存プロパティになりました。
- **Storyboard.GetClockState** は **GetCurrentState** に変更になりました。
- **BitmapImage.DownloadProgress** は 0 から 100 の間の実際のプログレス情報を提供します（これまでの 0-1 ではなく）。
- **EventTrigger.RoutedEvent** の型は文字列ではなく **RoutedEvent** です。
- **System.Windows.StyleSimulations** はフラグとして使えることを示すために、フラグ列挙子になりました。
- **PathGeometry** と **PathFigure** コレクションはインスタンス化したとき null ではなく空のコレクションになりました。
- **Controls.LayoutInformation** のコンストラクタは削除されました。

- **FrameworkElement._isDataContextBound** と **PresentationFrameworkCollection<T>._version** はプライベートになりました。
- **XamlReader.Load(string, bool)**は削除されました。
- **XamlReader.Load(string)**は変更され、必ず暗黙名前空間を作成します。
- **ControlTemplate** プロパティは削除されました。
- **System.Windows.FrameworkElement.SizeChangedEventHandler** と **System.Windows.FrameworkElement.LayoutUpdatedEventHandler** は削除されました。
- **Shape.GeometryTransformProperty** は削除されました。
- **Shape.GeometryTransfrom** プロパティの型は **System.Windows.Media.Matix** から **System.Windows.Media.Transform** に変更されました。
- **DependencyProperty.Register** と **DependencyProperty.RegisterAttached** 関数は引数として **PropertyChangedCallback** ではなく **PropertyMetadata** をとります。
- **ErrorEventArgs** と **ErrorEventHandler** は Silverlight Beta 2 マネージ API から削除されました。JavaScript を使用している場合には、この変更の影響は受けません。
- **OnUnchecked** と **OnIndeterminate** は **ToggleButton** から削除されました。**OnChecked** はインターナルになりました。
- **TimelineMarker** は **System.Windows.Media.Animation** 名前空間から **System.Windows.Media** に移動しました。
- **public static TimeSpan GetCurrentTime(Timeline timeline)**;は **public TimeSpan GetCurrentTime();**に変更されました。
- **public static ClockState GetCurrentState(Timeline timeline);**は **ClockState GetCurrentState();**に変更されました。
- **System.Windows.XDomainAccess** の名前は **System.Windows.CrossDomainAccess** になりました。
- **DependencyObject.NativePeerShutdown()**はインターナルになりました。
- **Application.ResourcesProperty** はプライベートになりました。
- **TabletDeviceType** はマネージ コードから削除されました。
- **DownloadProgressEventHandler** は削除されました。
- **IsolatedStorage.ApplicationSettings** の名前は **IsolatedStorage.IsolatedStorageSettings** に変更され、そのクラスはシールされました。
- **System.Exception.Source** プロパティはインターナルになりました。
- **OpenFileDialog.ShowDialog** は **DialogResult** ではなく **bool** を返します。
- **EnableMultipleSelection** は **Multiselect** に変更されました。
- **System.Array.TrueForAll<T>(T[], Predicate<T>)**は削除されました。
- **System.Collections.Generic.List<T>.ConvertAll<TOutput>(Converter<T, TOutput>)**は削除されました。
- **WatermarkedTextBox** は削除されました。しかし、これに関する可能な対応策についてはウェブ検索すべきです。
- **ListenUriMode** 列挙子は削除されました。

- **BindingContext.ctor(CustomBinding, BindingParameterCollection, Uri, string, ListenUriMode)** は削除されました。
- **XmlAnyElementAttributes** クラスはもはや **Collection<T>**, **ICollection<T>**, **IEnumerable<T>**, **IList<T>**を実装しません。
- **XmlArrayItemAttributes** クラスはもはや **Collection<T>**, **ICollection<T>**, **IEnumerable<T>**, **IList<T>**を実装しません。
- **XmlElementAttributes** クラスはもはや **Collection<T>**, **ICollection<T>**, **IEnumerable<T>**, **IList<T>**を実装しません。
- **System.Runtime.Serialization.ExtensionDataObject** クラスは削除されました。
- **XmlAnyAttributeAttribute** クラスは削除されました。
- **XmlAttributes.XmlAnyAttribute** メンバは削除されました。
- **CodeIdentifier** クラスは削除されました。
- **CodeIdentifiers** クラスは削除されました。
- 削除: デリゲート型 **XmlSerializationCollectionFixupCallback**, **XmlSerializationFixupCallback**, **XmlSerializationReadCallback**
- **XmlSerializationReader** の以下のプロテクト メンバは削除されました：
  - [削除] protected void **AddFixup**(Fixup fixup);
  - [削除] protected void **AddFixup**(CollectionFixup fixup);
  - [削除] protected void **AddReadCallback**(String name, String ns, Type type, XmlSerializationReadCallback read);
  - [削除] protected void **AddTarget**(String id, Object o);
  - [削除] protected void **FixupArrayRefs**(Object fixup);
  - [削除] protected Int32 **GetArrayLength**(String name, String ns);
  - [削除] protected Object **GetTarget**(String id);
  - [削除] protected Boolean **ReadReference**(out String fixupReference);
  - [削除] protected Object **ReadReferencedElement**(String name, String ns);
  - [削除] protected Object **ReadReferencedElement**();
  - [削除] protected void **ReadReferencedElements**();
  - [削除] protected Object **ReadReferencingElement**(String name, String ns, Boolean elementCanBeType, out String fixupReference);
  - [削除] protected Object **ReadReferencingElement**(String name, String ns, out String fixupReference);
  - [削除] protected Object **ReadReferencingElement**(out String fixupReference);
  - [削除] protected void **Referenced**(Object o);
  - [削除] protected void **UnreferencedObject**(String id, Object o);
  - [削除] protected class **CollectionFixup**; // removed
  - [削除] protected internal class **Fixup**; // removed

- **XmlSerializationWriter** の以下のプロテクト メンバは削除されました：
  - [削除] protected void **AddWriteCallback**(Type type, String typeName, String typeNs, XmlSerializationWriteCallback callback);
  - [削除] protected void **WriteId**(Object o);
  - [削除] protected void **WritePotentiallyReferencingElement**(String n, String ns, Object o, Type ambientType, Boolean suppressReference);
  - [削除] protected void **WritePotentiallyReferencingElement**(String n, String ns, Object o, Type ambientType);
  - [削除] protected void **WritePotentiallyReferencingElement**(String n, String ns, Object o, Type ambientType, Boolean suppressReference, Boolean isNullable);
  - [削除] protected void **WritePotentiallyReferencingElement**(String n, String ns, Object o);
  - [削除] protected void **WriteReferencedElements**();
  - [削除] protected void **WriteReferencingElement**(String n, String ns, Object o);
  - [削除] protected void **WriteReferencingElement**(String n, String ns, Object o, Boolean isNullable);
  - [削除] protected void **WriteRpcResult**(String name, String ns);