确保您的 Silverlight 2 或 Silverlight 3 Beta 应用程序能够用于 Silverlight 3 发行版

# 简介

本主题讨论在 Silverlight 2 和 Silverlight 3 之间，Silverlight 运行时和 Silverlight 工具所发生的变化。本主题重点论述可能会导致基于旧版 Silverlight 的应用程序现在运行失败或行为发生变化的更改，而不是该发行版的新功能/增强功能。本主题分为以下三个主要部分：

1. 自 Silverlight 3 Beta 以来的重大更改：此部分中描述的更改可能会破坏为 Silverlight 3 Beta 编写的应用程序。
2. 自 Silverlight 2 以来的重大更改：此部分中描述的更改可能会破坏为 Silverlight 2 编写的应用程序。
3. 升级重大更改。这些更改将不会破坏 Silverlight 2 应用程序，除非您使用 Silverlight 3 重新编译您的程序。

注意   [此处](http://go.microsoft.com/fwlink/?LinkId=119604)列出了本主题的更正/添加内容（如果存在）。

# 目录

[1 简介 1](#_Toc233611065)

[2 目录 1](#_Toc233611066)

[3 自 Silverlight 3 Beta 以来的重大更改 4](#_Toc233611067)

[3.1 DockPanel、WrapPanel、Expander、HeaderedContentControl、Viewbox、DataForm 迁移到了 Silverlight 工具包 4](#_Toc233611068)

[3.2 AutoCompleteBox：Search -> Filter 4](#_Toc233611069)

[3.3 AutoCompleteBox 的新模板部件：“Selector”, typeof(Selector) 4](#_Toc233611070)

[3.4 AutoCompleteBox.TryGetSelectionAdapter(object o) 现在是 GetSelectionAdapterPart() 5](#_Toc233611071)

[3.5 AutoCompleteBox：ToggleButton 类型的“DropDownToggle”模板部件已删除 5](#_Toc233611072)

[3.6 DataGrid RowGroupExpanding 和 RowGroupCollapsing 事件已删除 6](#_Toc233611073)

[3.7 DataGrid.SortDescriptions 和 GroupDescriptions 属性已删除 6](#_Toc233611074)

[3.8 用 ValidationSummary 而非 ListBox 显示 DataGrid 实体级别验证错误 6](#_Toc233611075)

[3.9 DataGrid 不再默认选择第一项 7](#_Toc233611076)

[3.10 DataForm 更改 7](#_Toc233611077)

[3.11 针对 System.ComponentModel.dll 和 DataPager 的重大更改 9](#_Toc233611078)

[3.12 PagedCollectionView 9](#_Toc233611079)

[3.13 DataPagerAutomationPeer 9](#_Toc233611080)

[3.14 BindableAttribute 和 BindingDirection 已删除 9](#_Toc233611081)

[3.15 对 ValidationAttribute 的更改 10](#_Toc233611082)

[3.16 [CustomValidation] 方法的签名已更改 11](#_Toc233611083)

[3.17 框架仅支持导航到页面 11](#_Toc233611084)

[3.18 与 UriMapper 相关的框架重大更改 12](#_Toc233611085)

[3.19 效果文件 (.ps) 不再作为 Content 加载 12](#_Toc233611086)

[3.20 WritableBitmap 构造函数参数已删除 13](#_Toc233611087)

[3.21 RenderOptions.TextRenderingOptions 现在是 TextOptions.TextHintingMode 13](#_Toc233611088)

[3.22 透明平台扩展不在 Microsoft.com 上寄放 13](#_Toc233611089)

[3.23 浏览器外 API 名称更改 14](#_Toc233611090)

[3.24 ExecutionStates.DetachedUpdatesAvailable 已删除 15](#_Toc233611091)

[3.25 Icon.Size 类型更改并且 <Icon Size=”…”> 格式已更改 15](#_Toc233611092)

[3.26 XmlConvert.IsNCNameSurrogatePair() 和 IsStartNCNameSurrogatePair() 已删除 15](#_Toc233611093)

[3.27 SaveFileDialog.File -> SaveFileDialog.SafeFileName 16](#_Toc233611094)

[4 自 Silverlight 2 以来的重大更改 16](#_Toc233611095)

[4.1 System.Web.Silverlight.dll 已从 SDK 中删除 16](#_Toc233611096)

[4.2 Silverlight 应用程序现在响应浏览器缩放 17](#_Toc233611097)

[4.3 弹出项 Tab 键更改 17](#_Toc233611098)

[4.4 ComboBox 弹出项剪切已修复 17](#_Toc233611099)

[4.5 在 PopupRenderTransform 更改后 Silverlight 始终重绘弹出项 18](#_Toc233611100)

[4.6 ContentPresenter.Content = "string" 不再更改 ContentTemplate 属性 18](#_Toc233611101)

[4.7 在使用空的 ContentTemplate 时，ContentControl 现在将忽略内容 18](#_Toc233611102)

[4.8 设置 ContentControl.Content 不会导致不需要的 OnApplyTemplate 调用 18](#_Toc233611103)

[4.9 在 ItemsSource 设置为具有 typeconverter 的枚举时，ComboBox 和 ContentControl 现在显示字符串，而非数字 19](#_Toc233611104)

[4.10 没有所选项的 ComboBox 采用额外的击键来选择第二个项 19](#_Toc233611105)

[4.11 HorizontalScrollBarVisibility 和 VerticalScrollBarVisibility 现在可以在 ComboBox 和 ListBox 的样式中设置 19](#_Toc233611106)

[4.12 ListBoxItem.HorizontalContentAlignment 现在受到重视 19](#_Toc233611107)

[4.13 ListBoxItem.Style 现在优先于 ListBox.ItemContainerStyle 19](#_Toc233611108)

[4.14 在 ListBox 内使用 Tab 键定位现在将焦点移到下一个控件，而非下一个 ListBox 项 19](#_Toc233611109)

[4.15 从 PasswordBox/TextBox 中删除了 TemplatePart 声明 19](#_Toc233611110)

[4.16 TextBox 中的 TextAlignment 现在相对于文档而非 TextBox 工作 20](#_Toc233611111)

[4.17 只读 TextBox 的默认颜色已更改 20](#_Toc233611112)

[4.18 ENTER 键现在支持 Button.ClickMode，后者默认为 Release (KeyUp) 20](#_Toc233611113)

[4.19 Slider 控件的命中测试区域现在与其可见部分匹配 20](#_Toc233611114)

[4.20 clientaccesspolicy.xml 内的格式错误的 URL 现在导致该 URL 被忽略，而非文件的其余部分被忽略 20](#_Toc233611115)

[4.21 错误修复：更改 Opacity 没有始终导致重绘 20](#_Toc233611116)

[4.22 DataBinding 要求属性 setter 是公共的 21](#_Toc233611117)

[4.23 不允许在 SetBinding() 后更改 Binding.Path 21](#_Toc233611118)

[5 升级重大更改 21](#_Toc233611119)

[5.1 OpenFileDialog.ShowDialog() 现在只能从用户启动的事件打开 22](#_Toc233611120)

[5.2 Tab 键定位处理和关联的回调从异步更改为同步 22](#_Toc233611121)

[5.3 弹出项内的鼠标事件现在提供相对于弹出项的位置，而非相对于整个应用程序的位置 22](#_Toc233611122)

[5.4 弹出项现在只能具有一个父级 23](#_Toc233611123)

[5.5 弹出项内的名称范围现在正常工作 23](#_Toc233611124)

[5.6 FindElementsInHostCoordinates 现在搜索弹出项 23](#_Toc233611125)

[5.7 删除了与在 ItemsControl 中替换项相关联的多余事件 23](#_Toc233611126)

[5.8 ScrollViewer 现在将滚动条位置固定到 ScrollableWidth/ScrollableHeight 23](#_Toc233611127)

[5.9 在设置了 Stretch 并且 Height 和/或 Width 未设置时，矩形和椭圆现在出现 24](#_Toc233611128)

[5.10 TextBlock、TextBox 和 PasswordBox 的宽度现在向上舍入 24](#_Toc233611129)

[5.11 textblock.FontFamily = null 引发 ArgumentNullException 24](#_Toc233611130)

[5.12 空格不再在模板内的 Run 元素之间呈现 24](#_Toc233611131)

[5.13 RadioButton.GroupName 现在在模板外搜索 RadioButtons 24](#_Toc233611132)

[5.14 ComboBox 内容现在令 IsHitTestVisible="false" 24](#_Toc233611133)

[5.15 TextBox 现在注重 Opacity 属性 24](#_Toc233611134)

[5.16 具有空参数的 FrameworkElementAutomationPeer ctor 引发 NullReferenceException 24](#_Toc233611135)

[5.17 HyperlinkButton 现在将导航整个窗口，而不只是 Silverlight 位于其中的 IFRAME 25](#_Toc233611136)

[5.18 更正了在 Silverlight v2+ 应用程序不具有 XAP 文件扩展名时使用的基 URI 25](#_Toc233611137)

[5.19 Application.Current.Host.Source 现在即使是在 .xap URL 具有查询字符串时也会返回绝对 URL 25](#_Toc233611138)

[5.20 ResourceDictionary 枚举错误已修复 25](#_Toc233611139)

[5.21 ReadOnlyObservableCollection 已从 System.Windows.Controls.Data.dll 迁移到 System.Windows.dll 25](#_Toc233611140)

[5.22 DataGridEndingEditEventArgs 已从 SDK 中删除 26](#_Toc233611141)

[5.23 PollingDuplexHttpBinding 更改 26](#_Toc233611142)

[5.24 与删除事件处理程序有关的错误修复 26](#_Toc233611143)

[5.25 错误修复：允许 <Cursors>，而非 <Cursor> 26](#_Toc233611144)

[5.26 XAML 中的自定义附加属性要求 xmlns 为前缀 26](#_Toc233611145)

[5.27 错误修复：子类控制可能会加载错误的资源 26](#_Toc233611146)

[5.28 错误修复：在路径上的叶对象为空时不调用 ValueConverter 26](#_Toc233611147)

[5.29 Thumb 控件现在相对于绝对位置计算拖放，而非相对于其父级计算拖放 26](#_Toc233611148)

[5.30 不再支持在 SetBinding() 后设置 Binding 属性 27](#_Toc233611149)

[5.31 ContentPresenter 的子级现在只能具有一个父级 27](#_Toc233611150)

[5.32 指向 .xap 外部资源的相对 URL 现在是相对于 .xap 文件而不是 HTML 页面 27](#_Toc233611151)

# 自 Silverlight 3 Beta 以来的重大更改

## DockPanel、WrapPanel、Expander、HeaderedContentControl、Viewbox、DataForm 迁移到了 Silverlight 工具包

以下控件和类型已从 Silverlight SDK 中删除：

         DockPanel

         WrapPanel

         Expander

         HeaderedContentControl

         Viewbox

         DataForm

         ExpandDirection

         ExpanderAutomationPeer

         LengthConverter

         StretchDirection

它们现在位于 Silverlight 工具包中，该工具包可从 <http://www.codeplex.com/Silverlight> 获得。

## AutoCompleteBox：Search -> Filter

* AutoCompleteBox.SearchMode 属性现在是 AutoCompleteBox.FilterMode
* AutoCompleteSearchMode 枚举现在是 AutoCompleteFilterMode
* AutoCompleteSearchPredicate 委托现在是 AutoCompleteFilterPredicate

**在这一更改前：**

<input:AutoCompleteBox SearchMode=”Contains” />

**在这一更改后：**

            <input:AutoCompleteBox FilterMode=”Contains” />

## AutoCompleteBox 的新模板部件：“Selector”, typeof(Selector)

如果您的 AutoCompleteBox 控件模板具有名为“SelectionAdapter”的部件，则应该将其重命名为“Selector”。

以前，AutoCompleteBox 将获取名为“SelectionAdapter”的任何模板部件，并且尝试或者将其用作适配器，或者将其包装在预定义的 SelectorSelectionAdapter 中。

经过这个更改后，为 AutoCompleteBox 准备选择适配器的基本逻辑是：

         找到名为“Selector”的模板部件。

o   如果该控件*实现* ISelectionAdapter，则使用它。

o   否则，如果找到了 Selector，则为其创建一个新的 SelectorSelectionAdapter。

         如果未准备 SelectionAdapter，则定位到名为 SelectionAdapter 的模板部件。

## AutoCompleteBox.TryGetSelectionAdapter(object o) 现在是 GetSelectionAdapterPart()

以前，AutoCompleteBox API 包含了一个名为“TryGetSelectionAdapter”的受保护的虚方法。该 API 实际上并未实现 TryGet 模式。

新的 API 仅仅是一个受保护的虚方法，没有名为 GetSelectionAdapterPart 的参数。

## AutoCompleteBox：ToggleButton 类型的“DropDownToggle”模板部件已删除

AutoCompleteBox 的默认控件模板从未利用过此模板部件。为了改进 Blend 中的模板化体验，此模板部件已删除。

**如何更新现有应用程序？**

在具有自定义控件模板定义的文件中，在 XAML 和代码隐藏文件中添加了 Click 事件处理程序。然后，在可视化树层次结构中定位到自动完成控件，并且切换其 IsDropDownOpen 依赖属性。

从 AutoCompleteBox 执行派生的任何用户应该仍能够定义自己的 ToggleButton 模板部件并切换 IsDropDownOpen 属性。

根据更新的示例应用程序，下面是自定义控件模板部分：

|  |  |
| --- | --- |
| 之前 (XAML) | 之后 (XAML) |
| <ToggleButton x:Name="DropDownToggle"  Style="{StaticResource ComboToggleButton}" /> | <ToggleButton ~~x:Name="DropDownToggle"~~  Style="{StaticResource ComboToggleButton}"  **Click=”DropDownToggle\_Click”**  /> … |

并且代码隐藏文件：

|  |  |
| --- | --- |
| 之前 (C#) | 之后 (C#) |
| （不适用） | private void DropDownToggle\_Click(object sender, RoutedEventArgs e)  {       FrameworkElement fe = sender as FrameworkElement;       AutoCompleteBox acb = null;       while (fe != null && acb == null)       {           fe = VisualTreeHelper.GetParent(fe) as FrameworkElement;           acb = fe as AutoCompleteBox;       }       if (acb != null)       {           acb.IsDropDownOpen = ! acb.IsDropDownOpen;       }  } |

## DataGrid RowGroupExpanding 和 RowGroupCollapsing 事件已删除

这些事件在 Beta 版中没有完全生效，因此已删除。

## DataGrid.SortDescriptions 和 GroupDescriptions 属性已删除

SortDescriptions 和 GroupDescriptions 属性已从 DataGrid 中删除。您可以使用代码隐藏文件设置组说明。

Silverlight 3 Beta

[Xaml]

<dataGrid.GroupDescriptions>

      <windata:PropertyGroupDescription PropertyName=”State” />

</dataGrid.GroupDescriptions>

Silverlight 3 RTM

[c#]

pagedCollectionView.GroupDescriptions.Add(new PropertyGroupDescription(“State”));

## 用 ValidationSummary 而非 ListBox 显示 DataGrid 实体级别验证错误

影响范围：将 DataGrid 重新模板化为包括 ListBox 以用于验证错误的自定义模板的 Silverlight 3 Beta 托管应用程序。

摘要：为了实体级别验证错误具有标准的外观和体验，DataGrid 现在使用与 DataForm 类似的 ValidationSummary。以前，DataGrid 使用 ListBox。

所需的修复：如果用户的自定义 DataGrid 模板包括用于验证错误的 ListBox，则需要更新其模板。

Silverlight 3 Beta

[Xaml]

<ListBox Name="ErrorsListBox" Grid.Row="3" Grid.ColumnSpan="3" MaxHeight="91" Background="#FFEFEFEA" BorderThickness="0,1,0,0">

…

Silverlight 3 RTM

[Xaml]

<dataInput:ValidationSummary Name="ValidationSummary" Filter="ObjectErrors" Grid.Row="3" Grid.ColumnSpan="3" MaxHeight="90"/>

## DataGrid 不再默认选择第一项

**影响范围**: 依赖于默认选择第一项的 DataGrid 的 Silverlight 2 应用程序。

摘要

根据客户反馈，DataGrid 不再默认选择第一项，这样，它与 ListBox 之类的其他基于列表的控件的行为匹配。现在，DataGrid 的当前项默认为第一项，但不再选择该项。按 Tab 键进入 DataGrid 将导致选择当前项。但是，如果其 ItemsSource 设置为 ICollectionView 实现方式，则 DataGrid 默认将选择 ICollectionView 的当前项。这与 ListBox 一致并与 ICollectionView 的其他使用者一致，并且它还通过 CollectionView 支持主从方案。

所需的修复

依赖于 DataGrid 默认选择第一项的 Silverlight 2 用户现在需要在设置 ItemsSource 后显式设置 SelectedItem 或 SelectedIndex。

## DataForm 更改

与 **DataForm** 相关的最重大更改影响在 **DataForm** 中使用 **Fields** 集合的任何用户，如下所示：

**DataFormFields** 已全都删除并且 **DataForm.Fields** 集合已删除。它们已被新的 **DataField** 控件替换，该控件使用标签和 **DescriptionViewer** 包装内容。此外，还要注意的是，您现在必须对 **DataField** 中的绑定显式指定 **Mode=TwoWay**，而以前，您可以只指定路径，而 **DataFormField** 会将 TwoWay 插入到绑定中。例如，如果您在 Fields 集合中具有以下内容：

<dataControls:DataFormTextField Binding="{Binding FirstName}" />

...您要改为将以下内容放置于 EditTemplate 中：

<dataControls:DataField>

<TextBox Text="{Binding FirstName, Mode=TwoWay}" />

</dataControls:DataField>

请注意，因为它们现在是控件，所以 **DataFormFieldGroups** 只能被常规面板替换。

还要注意，因为它们现在是针对自身的控件，所以 **DataForm** 不再处理这些字段中标签的所有对齐方式。**DataForm** 会在顶层将这些字段组合在一起，但如果您希望更优化的分组（例如，两列字段），则可以对 **Panel** 使用附加的属性 **DataField.IsFieldGroup**，以便指定 **Panel** 内的 **DataFields** 应在宽度方向上将其标签组合在一起。下面显示一个示例：

<StackPanel Orientation="Horizontal">

<StackPanel dataControls:DataField.IsFieldGroup="True">

<dataControls:DataField>

<TextBox Text="{Binding FirstName, Mode=TwoWay}" />

</dataControls:DataField>

<dataControls:DataField>

<TextBox Text="{Binding LastName, Mode=TwoWay}" />

</dataControls:DataField>

</StackPanel>

<StackPanel dataControls:DataField.IsFieldGroup="True">

<dataControls:DataField>

<TextBox Text="{Binding StreetAddress, Mode=TwoWay}" />

</dataControls:DataField>

<dataControls:DataField>

<TextBox Text="{Binding State, Mode=TwoWay}" />

</dataControls:DataField>

</StackPanel>

</StackPanel>

这将导致 FirstName/LastName DataFields 使其标签具有相同的宽度，并且对于 StreetAddress/State DataFields 是相似的。

上述内容介绍了这一重大更改的主要部分。此外，针对 **DataForm** 还有如下其他重大更改：

* **FieldEditEnding** 和 **FieldEditEnded** 已删除。
* **GetFieldElement()** 已删除。
* **ItemEditEnding**、**ItemEditEnded** 和 **CommitItemEdit()** 已分别更改为 **EditEnding**、**EditEnded** 和 **CommitEdit()**。
* CanMoveToXXXXItem 已删除，因为它们可以从提供给 **DataForm** 的集合派生。
* **CanUserAdd**/**DeleteItems** 已删除，因为它们通过 **CommandButtonsVisibility** 暗含。
* **IsEditing** 和 **IsAddingItem** 已删除，因为它们通过 **Mode** 暗含。
* **FieldLabelPosition** 和 **FieldLabelStyle** 已变为 **LabelPosition** 和 **LabelStyle**。
* **WrapAfter** 和 **Orientation** 已删除。
* 类型名称 **DataFormFieldLabelPosition** 和 **DataFormDescriptionViewerPosition** 已更改为 **DataFieldLabelPosition** 和 **DataFieldDescriptionViewerPosition**。

## 针对 System.ComponentModel.dll 和 DataPager 的重大更改

下面概要介绍了这些重大更改：

1. **System.ComponentModel.dll** 已删除。
2. 以下类/接口已从 **System.ComponentModel** 迁移到新的 **System.Windows.Data.dll**（命名空间保持不变）
   1. PagedCollectionView
   2. CollectionViewGroup
   3. IEditableCollectionView
   4. IPagedCollectionView
   5. PageChangingEventArgs
   6. PropertyGroupDescription
3. 以下类/接口将不再是 SDK 的一部分
   1. EntityAction
   2. IEditableCollection
   3. IIndexableCollection
4. **IPagedCollection** 已删除。**PagedCollectionView** 不再具有特殊的 **IPagedCollection** 实施方。
5. **DataPager.Source** 不再为 **IPagedCollectionView** 类型，现在而是 **IEnumerable**（还可以是 **IPagedCollectionView**）。在将非 IPagedCollectionViews 提供给 DataPager 时，其行为就像具有单个大型页的集合绑定到该控件。要调用 **IPagedCollectoinView** API，请将 **Source** 属性转换为 **IPagedCollectionView**。

## PagedCollectionView

PagedCollectionView 不再支持任意远程页面。因此，PagedCollectionView.NewItemPlaceholder 已删除，并且 PagedCollectionView.ItemCount 和 TotalItemCount 属性现在是只读的。

## DataPagerAutomationPeer

DataPagerAutomationPeer 现在显式实现 IRangeValueProvider，而不是隐式实现。因此，以下成员已从 DataPagerAutomationPeer 类中删除：

        public bool IsReadOnly { get; set; }

        public double LargeChange { get; set; }

        public double Maximum { get; set; }

        public double Minimum { get; set; }

        public double SmallChange { get; }

        public double Value { get; }

        public void SetValue(double value);

## BindableAttribute 和 BindingDirection 已删除

在 Silverlight 3 Beta 中，Bindable 属性用于向业务对象的属性加批注，从而允许用户界面以两种方式响应：

         在用户界面中将该属性视为只读。

         如果控件应该为该属性生成用户界面，则指示控件执行自动生成，例如 DataGrid 和 DataForm。

在 Silverlight 3 发行版中，Editable 和 Display 属性应该用于这些目的。

在用户界面中将属性标记为 ReadOnly：

|  |  |
| --- | --- |
| 之前 | 之后 |
| **[Bindable (true, BindingDirection.OneWay)]**  public string foo {get;set;} | **[Editable (false)]**  public string foo {get;set;} |

将属性标记不让控件自动为其生成用户界面：

|  |  |
| --- | --- |
| 之前 | 之后 |
| **[Bindable (false)]**  public int bar {get;set;} | **[Display (AutoGenerateField=false)]**  public int bar {get;set;} |

## 对 ValidationAttribute 的更改

从测试版的 ValidationAttribute 派生的任何类都可能需要更新。代替从 IsValid 返回 bool，实现方式应只覆盖基于 ValidationContext 的 IsValid 方法并返回 ValidationResult。ValidationContext 在不需要时可忽略。

ValidationAttributeAPI 已更改为以下内容：

public abstract class ValidationAttribute: Attribute {

~~public void Validate(object value, string name);~~

public void Validate(object value, ValidationContext validationContext);

~~public bool TryValidate(object value, ValidationContext validationContext, ref ValidationResult validationResult);~~

public ValidationResult GetValidationResult(object value, ValidationContext validationContext~~, out ValidationResult validationResult~~);

~~public abstract bool IsValid(object value);~~

protected virtual ~~bool~~ ValidationResult IsValid(object value, ValidationContext validationContext~~, out ValidationResult validationResult~~);

}

代表成功和失败：ValidationAttribute 实现方式需要返回 ValidationResult 以代表成功和失败。为了代表成功，public static readonly ValidationResult Success 已添加到 ValidationResult。对于失败情况，ValidationResult 具有两个构造函数，用于提供错误消息和可选的成员名称列表。这些构造函数没有更改，但添加了 Success 字段。

下面是 ValidationAttribute 的非常简单的实现方式，用于阐释成功和失败。

    public class IsTrueAttribute : ValidationAttribute {

        protected override ValidationResult IsValid(object value, ValidationContext validationContext) {

            if (value is bool && (bool)value) {

                return ValidationResult.Success;

            }

            else {

                return new ValidationResult("The value provided was not true");

            }

        }

    }

Validator 类已更新为调用 GetValidationResult，而非 TryValidate。调用 TryValidate 的任何其他内容也将需要更新，尽管没有受此更改影响的其他框架代码。

## [CustomValidation] 方法的签名已更改

对于实施了自定义验证方法的任何地方，都必须更新签名。冲突在运行时将不是可发现的。如果您已经实施了任何自定义验证方法，则必须找到这些实现方式并更新方法签名，以便匹配新支持的签名之一。

CustomValidationAttribute 用于提供为验证而要调用的类型和方法。这是用于扩展验证框架以调用自定义验证例程的简单机制，它使用针对方法签名的约定，以便可以调用引用的方法。

在 Silverlight 3 Beta 中，[CustomValidation] 属性要求以下方法签名之一：

public static bool MethodName([type] value) {…}

public static bool MethodName([type] value, ValidationContext context, out ValidationResult result) {…}

在 Silverlight 3 的最终版本中，支持的方法签名已更改为以下内容：

public static ValidationResult MethodName([type] value) {…}

public static ValidationResult MethodName([type] value, ValidationContext context) {…}

返回类型已更改，并且“out ValidationResult”参数已删除。

## 框架仅支持导航到页面

对于从页面派生的控件，框架仅支持导航到 XAML。尝试导航到非页面的其他项将导致异常。

在 Silverlight 3 Beta 中，可以导航到 UserControl 的 XAML，这增加了复杂性并要求用户知道如何采用导航服务以获取该 UserControl 内与导航相关的信息。如果仍需要这样做，将 UserControl 包装在 Page 中或者使用 Page 作为父类。

## 与 UriMapper 相关的框架重大更改

以前，向 Frame 控件提供 UriMapper 的方法是使用“uriMapper”的键将其作为资源添加：

**在这一更改前：**

<navigation:Frame.Resources>

    <nav:UriMapper x:Key=”uriMapper”>

    …

    </nav:UriMapper>

</navigation:Frame.Resource>

**在这一更改后：**

Frame 控件具有名为 UriMapper 的属性，该属性取 UriMapperBase：

    public static readonly DependencyProperty **UriMapperProperty**;  
    public UriMapperBase **UriMapper** { get; set; }

现在，您可以显式声明和指定 UriMapper，例如通过绑定到某一资源：

<navigation:Frame UriMapper=”{StaticResource uriMapper}” …>

</navigation:Frame>

或者，您可以直接指定它：

<navigation:Frame …>

    <navigation:Frame.UriMapper>

     <nav:UriMapper>

       …

       </nav:UriMapper>

    </navigation:Frame.UriMapper>

</navigation:Frame>

UriMapper 本身的 API 未更改。

## 效果文件 (.ps) 不再作为 Content 加载

在 Silverlight 3 Beta 中，效果文件（像素着色器）可以包装为“Content”或“Resource”（在 Visual Studio 属性窗口中，Build Action = Content 或 Resource）。Silverlight 现在只支持包装为 Resource 的效果文件。（对于内置的 DropShadow 和 Blur 效果没有更改。）

要使用某一 PS 文件，请在 Visual Studio 中将其 build action 设置为“Resource”，并且通过“<assemblyName>;component/<filename>.ps”形式的程序集 URI 访问它。下面显示加载 GrayScaleEffect.fx.ps 的 SilverlightCustomEffect 应用程序的示例：

pshader = new PixelShader() { UriSource = new Uri("SilverlightCustomEffect;component/GrayScaleEffect.fx.ps", UriKind.Relative) };

## WritableBitmap 构造函数参数已删除

WritableBitmap 构造函数的 PixelFormat 参数已删除：

public [WriteableBitmap](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Windows:2.0.5.0:7cec85d7bea7798e/System.Windows.Media.Imaging.WriteableBitmap/.ctor(Int32,Int32,System.Windows.Media.PixelFormat))([int](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.5.0:7cec85d7bea7798e/System.Int32) pixelWidth, [int](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.5.0:7cec85d7bea7798e/System.Int32) pixelHeight**~~,~~** [**~~PixelFormat~~**](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Windows:2.0.5.0:7cec85d7bea7798e/System.Windows.Media.PixelFormat) **~~format~~**);

唯一支持的 PixelFormat 现在是 Pbgra32。

同样，PixelFormat 和 PixelFormats 类型已删除。

## RenderOptions.TextRenderingOptions 现在是 TextOptions.TextHintingMode

**在这一更改前：**

public static class RenderOptions {

public static DependencyProperty TextRenderingMode TextRenderingModeProperty;

}

public enum TextRenderingMode {

Default = 0,

RenderForAnimation = 1

}

**在这一更改后：**

public static class TextOptions {

public static DependencyProperty TextHintingMode TextHintingModeProperty;

}

public enum TextHintingMode {

Fixed = 0,

Animated = 1

}

## 透明平台扩展不在 Microsoft.com 上寄放

透明平台扩展将不在 Microsoft.com 上寄放。它们而是默认为在应用程序的源站点上寄放。应用程序开发人员可以通过编辑与程序集一起放置的程序集到扩展映射文件 (<assemblyShortName>.extmap.xml)，覆盖此设置。

## 浏览器外 API 名称更改

在 System.Windows.Application 类型中对 Detach 方法进行的更改。

|  |  |  |
| --- | --- | --- |
|  | 从 | 到 |
| 签名 | public bool Detach();  public bool RunningOffline;  public event EventHandler ExecutionStateChanged; | public bool Install();  public bool RunningOutOfBrowser;  public event EventHandler InstallStateChanged; |

对 ExecutionStates 枚举进行的更改。

|  |  |  |
| --- | --- | --- |
|  | 从 | 到 |
| 签名 | public enum ExecutionStates  {      RunningOnline,      Detaching,      Detached,      DetachedUpdatesAvailable,      DetachFailed  } | public enum InstallState  {      NotInstalled,      Installing,      Installed,      (已删除，见下文)      InstallFailed  } |

对应用程序清单 (AppManifest.xaml) 的 Deployment 元素中的 ApplicationIdentity 属性进行的更改。

|  |  |  |
| --- | --- | --- |
|  | 从 | 到 |
| XAML | <Deployment.ApplicationIdentity … | <Deployment.OutOfBrowserSettings … |
| 签名 | public ApplicationIdentity ApplicationIdentity; | public OutOfBrowserSettings OutOfBrowserSettings ; |

将 ApplicationIdentity.Title 迁移到了 OutOfBrowserSettings.WindowSettings.Title。

|  |  |  |
| --- | --- | --- |
|  | 从 | 到 |
| XAML | <ApplicationIdentity                         ShortName=”…”                         Title=”…”  > | <OutOfBrowserSettings ShortName=”…”>      <OutOfBrowserSettings.WindowSettings>            <WindowSettings Title=”…”     />   <OutOfBrowserSettings.WindowSettings /> |

## ExecutionStates.DetachedUpdatesAvailable 已删除

ExecutionStates.DetachedUpdatesAvailable 值已从 ExecutionStates 枚举（现在是 InstallState 枚举）中删除。考虑改为使用以下新的 API 之一：

* Application.CheckAndDownloadUpdateAsync() 方法
* Application.CheckAndDownloadUpdateCompleted 事件
* CheckAndDownloadUpdateCompletedEventArgs 类
* CheckAndDownloadUpdateCompletedEventArgs.UpdateAvailable 属性

## Icon.Size 类型更改并且 <Icon Size=”…”> 格式已更改

此类型通常用于 appmanifest.xaml 内，以便指定浏览器外应用程序的图标信息。更改将影响 XAML 和编程用法：

* Icon.Size 属性已从字符串类型更改为大小类型。
* 对于 Icon.Size，将不再接受值“small”、“medium”和“large”
* XAML 格式现在与大小类型的其他属性相同。例如，“16,16”代替了“16x16”。
* 现在将验证图标大小。有效图标大小是“16,16”、“32,32”、“48,48”、“128,128”、“256,256”、“512,512”。

**示例：**

**之前：**

<Icon Size="16x16">16.png</Icon>

<Icon Size="32x32">32.png</Icon>

<Icon Size="48x48">48.png</Icon>

<Icon Size="128x128">128.png</Icon>

**之后：**

<Icon Size="16,16">16.png</Icon>

<Icon Size="32,32">32.png</Icon>

<Icon Size="48,48">48.png</Icon>

<Icon Size="128,128">128.png</Icon>

## XmlConvert.IsNCNameSurrogatePair() 和 IsStartNCNameSurrogatePair() 已删除

以下方法已删除：

class XmlConvert {

  public static bool IsNCNameSurrogatePair(char lowChar, char highChar);

  public static bool IsStartNCNameSurrogatePair(char lowChar, char highChar);

}

## SaveFileDialog.File -> SaveFileDialog.SafeFileName

SaveFileDialog.File 属性已删除。它已被 SaveFileDialog.SafeFileName 属性替代：

public String SafeFileName { get; }

只包含所选文件的文件名的字符串。默认值为 Empty，它也是未选择任何文件或选择了某一目录时的值。

# 自 Silverlight 2 以来的重大更改

在此部分中介绍的更改可能会损坏 Silverlight 2 应用程序。其中许多修改实际是错误修复；不过，如果您的 Silverlight 2 应用程序已经针对这些错误采用了解决方法，您可能必须取消这些解决方法，您的应用程序才会正常工作。

## System.Web.Silverlight.dll 已从 SDK 中删除

ASP.NET Silverlight 和 MediaPlayer 控件已从 Silverlight 3  SDK 中删除。它们包含在名为 System.Web.Silverlight.dll 的服务器程序集中，该程序集通常复制到测试网站的 BIN 文件夹。

*<asp:silverlight ../>*

*<asp:mediaplayer ../>*

现有的 Silverlight 2 网站将继续工作，因为它们在其 Web 应用程序的 BIN 文件夹中具有 System.Web.Silverlight 程序集（即使 Silverlight 2 SDK 已卸载也是如此）。

在将现有项目升级到 Silverlight 3 时，在大多数情况下，您应该用等效的 <object> 标记替换这些控件。对于 <asp:silverlight>，标记是：

      <div id="silverlightControlHost">

            <object data="data:application/x-silverlight-2," type="application/x-silverlight-2" width="100%" height="100%">

                <param name="source" value="{XAP\_FILE}"/>

                <param name="onerror" value="onSilverlightError" />

                <param name="background" value="white" />

                <param name="minRuntimeVersion" value="{BUILD}" />

                <param name="autoUpgrade" value="true" />

                <a href="http://go.microsoft.com/fwlink/?LinkID=149156&v={BUILD}" style="text-decoration: none;">

                     <img src="http://go.microsoft.com/fwlink/?LinkId=108181" alt="Get Microsoft Silverlight" style="border-style: none"/>

                </a>

            </object><iframe id="\_sl\_historyFrame" style='visibility:hidden;height:0;width:0;border:0px'></iframe></div>

**现有的** Silverlight 应用程序将继续在其 Web 应用程序的 BIN 文件夹中具有 System.Web.Silverlight 程序集，即使 Silverlight 2 SDK 已卸载也是如此。因此，升级到 Silverlight 3 SDK 将不会删除此本地程序集，并且应用程序应该继续工作。理论上，开发人员将为 Silverlight 3 设置相关的版本标志。

* 在客户端，呈现的较旧的 fwlink 现在应指向 Silverlight 3。
* 已将 System.Web.Silverlight 程序集添加到 GAC 的开发人员必须为每个安装从 GAC 手动删除它。

创建**新应用程序**的开发人员将要使用包含 OBJECT 标记和其他标记的最新模板。

* 开发人员“可以”手动添加 System.Web.Silverlight 程序集并且使用服务器控件。但是，这些控件将不提供最新的安装逻辑，或者能够为 Silverlight 历史记录支持呈现 iframe（举例）。

## Silverlight 应用程序现在响应浏览器缩放

Silverlight 应用程序（旧应用程序和新应用程序）将有机会根据其当前行为响应浏览器缩放。如果应用程序未处理 Resized 事件，Silverlight 应用程序现在默认情况下将缩放。对于应用程序，缩放等效于 RenderTransform = new ScaleTransform() 应用于根元素的虚父级。

应用程序可通过使用以下新的 API 来自定义此行为：

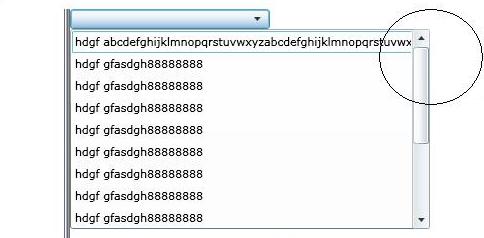
* System.Windows.Interop.Content 的 Zoomed 事件和 ZoomFactor 属性。（Content 类通常用作 App.Current.Host.Content）
* 针对 Silverlight 插件的 EnableAutoZoom 参数

## 弹出项 Tab 键更改

在 Silverlight 2 中，如果您在某一弹出项内具有焦点并且按下 TAB 键，则焦点将完全离开 Silverlight 并转到浏览器，并且忽略在该弹出项内的其他制表位。Silverlight 3 中纠正后的新行为是 Tab 键移动首先在该弹出项内进行。如果您希望保留旧行为，则可以对弹出项的根设置 **TabNavigation=”Cycle”**。

## ComboBox 弹出项剪切已修复

在 Silverlight 2 中，如果在 **ComboBox** 弹出项中存在垂直滚动条，并且弹出项宽于 **ComboBox** 本身，则按滚动条的宽度，弹出项太窄。



如果您通过人工增加 **ComboBox** 项的大小而解决了这一 Silverilght 2 错误，在 Silverlight 3 中则会存在多余的空间，因此您可能要取消您的解决方法。

## 在 PopupRenderTransform 更改后 Silverlight 始终重绘弹出项

在 Silverlight 2 中，在 **Popup.RenderTransform** 更改后 Silverlight 并不始终重绘弹出项。此问题已在 Silverlight 3 中修复。

## ContentPresenter.Content = "string" 不再更改 ContentTemplate 属性

以前，如果您设置 **ContentPresenter.Content = string**，然后查询 **ContentPresenter.ContentTemplate**，您将会看到一个非空值，并且使用模板来显示该字符串。现在，设置 ContentPresenter.Content = string 不再更改 **ContentTemplate** 属性。这一更改已完成，这样，在 **ContentPresenter.Content** 设置为 string 后，将无法再对它进行设置。

请注意，这是 **ContentPresenter**，而非 **ContentControl**。**ContentPresenter** 通常仅用于如下控件模板内：

<ContentPresenter

Content="{TemplateBinding Content}"

ContentTemplate="{TemplateBinding ContentTemplate}"

HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"

VerticalAlignment="{TemplateBinding VerticalContentAlignment}"

Margin="{TemplateBinding Padding}"/>

## 在使用空的 ContentTemplate 时，ContentControl 现在将忽略内容

在 Silverlight 2 中，如果您向 ContentControl（例如 Button）提供一个空的 **ContentTemplate**，该控件将忽略模板并且使用 **ContentPresenter** 显示内容。例如，以下代码在 Silverlight 2 中显示“ABC”：

<ContentControl Content="ABC">

<ContentControl.ContentTemplate>

<DataTemplate></DataTemplate>

</ContentControl.ContentTemplate>

</ContentControl>

此问题在 Silverlight 3 中已修复，因此，以前的代码现在将不显示任何内容。

## 设置 ContentControl.Content 不会导致不需要的 OnApplyTemplate 调用

在 Silverlight 2 中，设置 ContentControl.Content 将会每次都调用 OnApplyTemplate，即使已设置 ContentControl.ControlTemplate 也是如此。在 Silverlight 3 中，如果已设置 ContentControl.ControlTemplate，OnApplyTemplate 只在首次时被调用。

## 在 ItemsSource 设置为具有 typeconverter 的枚举时，ComboBox 和 ContentControl 现在显示字符串，而非数字

在 Silverlight 2 中，如果您将枚举放置于 **ComboBox**（其 **ItemsSource** 属性）中，则 **ComboBox** 将显示枚举的数值，而非枚举的名称。例如：

enum MyEnum { First, Second, Third }

…

MyComboBox.ItemsSource = new MyEnum[] { MyEnum.First };

显示“1”而非“First”。此问题在 Silverlight 3 中已修复，因此，现在将显示枚举的字符串值。

## 没有所选项的 ComboBox 采用额外的击键来选择第二个项

在 Silverlight 2 中，当您打开没有所选项的 ComboBox 时，列表中的第一个项将视作具有焦点（即使未选择它），因此，第一个向下箭头键会将焦点和选择均移到第二个项。在 Silverlight 3 中，在打开该 ComboBox 时，第一个项不被视为具有焦点。第一个 keydown 会将选择和焦点移到第一个项，而第二个 keydown 会将选择和焦点移到第二个项。

## HorizontalScrollBarVisibility 和 VerticalScrollBarVisibility 现在可以在 ComboBox 和 ListBox 的样式中设置

在 Silverlight 2 中，这些属性在构造函数中在本地设置，并且覆盖来自样式的任何值。此问题已修复。

## ListBoxItem.HorizontalContentAlignment 现在受到重视

在 Silverlight 2 中，在使用 ListBoxItem 的默认模板时，HorizontalContentAlignment 被忽略。此问题已修复。

## ListBoxItem.Style 现在优先于 ListBox.ItemContainerStyle

在 Silverlight 2 中，如果您设置 ListBoxItem.Style，以后在代码中设置 ListBox.ItemContainerStyle，则会错误地覆盖您以前设置的 ListBoxItem.Style。此问题已修复。

## 在 ListBox 内使用 Tab 键定位现在将焦点移到下一个控件，而非下一个 ListBox 项

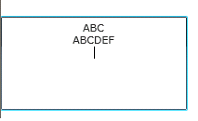
在 Silverlight 2 中，在焦点位于 **ListBox** 内时按下 Tab 键会将焦点移到下一个 **ListBox** 项，而非将焦点移到下一个控件。现在，Tab 键定位行为与其他平台一致。

## 从 PasswordBox/TextBox 中删除了 TemplatePart 声明

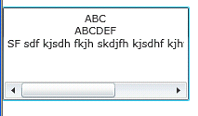
对于 Silverilght 2，**PasswordBox** 和 **TextBox** 具有多种 [TemplatePart] 属性，这些属性描述控件实际不具有的部件。这些属性已从 Silverlight 3 中删除。

## TextBox 中的 TextAlignment 现在相对于文档而非 TextBox 工作

在 Silverlight 2 中，**TextBox** 中的文本并不基于“document”而居中，而是基于 **TextBox** 的宽度属性。这导致了不正确的居中呈现。例如，假定您从如下的居中的文本开始：



在 Silverlight 2 中，如果您键入一个长字符串，则 **TextBox** 如下：



在 Silverlight 3 中，此行为得到了纠正。“ABC”的正确行为是对整个文档居中，而非对文档的可见部分居中。这一更改不会影响 **TextBox** 的整体大小；并且，如果用户需要，他们可以滚动视区以看到居中的文本。不过，如果您依赖以前的行为，则可能要调整您的应用程序。

## 只读 TextBox 的默认颜色已更改

在 Silverlight 2 中，只读 TextBox 看起来与可编辑文本框非常像。

## ENTER 键现在支持 Button.ClickMode，后者默认为 Release (KeyUp)

如果应用程序将 ClickMode 更改为 Press，则单击处理程序将无法执行导致重新进入的代码。（可能导致重新进入的代码示例包括 OpenFileDialog 和 SaveFileDialog。）

## Slider 控件的命中测试区域现在与其可见部分匹配

在 Silverlight 2 中，如果您令滑块真的很高，则您单击的区域可能看起来不像是该滑块的一部分，尽管它确实是其一部分。

## clientaccesspolicy.xml 内的格式错误的 URL 现在导致该 URL 被忽略，而非文件的其余部分被忽略

在 Silverlight 2 中，无效的 URL 导致了 clientaccesspolicy.xml 的其余部分被忽略。此问题已修复。

## 错误修复：更改 Opacity 没有始终导致重绘

在某些情况下，Silverlight 2 在 Opacity 更改后将不会重绘内容。此问题已修复。

## DataBinding 要求属性 setter 是公共的

Silverlight 数据绑定通常要求属性和类型是公共的。在 Silverlight 2 中，在具有公共 getter 和专用 setter 的属性可能被数据绑定到的地方存在错误。此问题已修复。

## 不允许在 SetBinding() 后更改 Binding.Path

在 Silverlight 2 中，可以在 SetBinding 调用中使用该绑定之后、但在设置 DataContext 之前，以编程方式设置 Binding.Path，例如：

Binding binding = new Binding();

myElement.SetBinding(HeightProperty, binding);

binding.Path = new PropertyPath("propertyname");

myElement.DataContext = …;

在 Silverlight 3 中，一旦调用了 SetBinding，Path 属性就不能更改，因此以上代码需要更改为：

Binding binding = new Binding();

binding.Path = new PropertyPath("propertyname");

myElement.SetBinding(HeightProperty, binding);

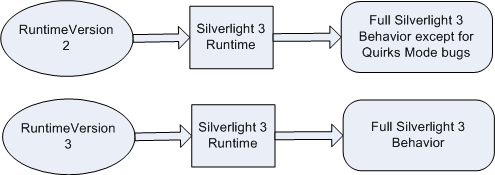
myElement.DataContext = …;

如果在调用 SetBinding() 后更改 Path，则 Silverlight 3 行为将依赖于编译该应用程序所使用的 Silverlight 的版本。如果使用 Silverlight 3 进行编译，则在 SetBinding() 后设置 Path 将引发异常。但是，为了降低损坏 Silverlight 2 应用程序的可能性，如果 .xap 是使用 Silverlight 2 生成的，则在 SetBinding() 后设置 Path 将被无提示忽略并且将使用旧的 Path（请参见第 5 部分中对“Quirks 模式”的论述）。

# 升级重大更改

Silverlight 团队想要在 Silverlight 3 中修复许多 Silverlight 2 错误。但是，通过修复其中一些错误，有可能某些现有的 Silverlight 2 应用程序会被破坏。为了避免此问题，Silverlight 开发人员使这些可能产生问题的更改成为“Quirks 模式更改”。Quirks 模式更改是这样一种更改：如果运行时检测到正运行的应用程序是为 Silverlight 2 设计的，将不强制执行 Silverlight 3 运行时。这样，在 Silverlight 2 应用程序正在运行时，Silverlight 3 将成为与 Silverlight 2“错误兼容的”。但是，如果您为 Silverlight 3 重新编译您的应用程序，则这些更改将会对您产生影响。

如下图所示，如果您的应用程序的 **RuntimeVersion** 指定 Silverlight 2，则运行时将在“Quirks 模式”下运行。在处于 Quirks 模式下时，运行时返回到与 Quirks 模式更改相关联的 Silverlight 2 行为。



从上图中，您可以看到：Silverlight 3 运行时通过使用 **RuntimeVersion** 检测应用程序是否是为 Silverlight 2 而设计的。**RuntimeVersion** 设置为 .xap 的 AppManifest.xaml 内的属性：

<Deployment xmlns="http://schemas.microsoft.com/client/2007/deployment"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" EntryPointAssembly="ElevatorSilverlight"

EntryPointType="ElevatorSilverlight.App"

RuntimeVersion="2.0.31005.0">

<Deployment.Parts>

<AssemblyPart x:Name="ElevatorSilverlight" Source="ElevatorSilverlight.dll" />

</Deployment.Parts>

</Deployment>

**RuntimeVersion** 反映在编译应用程序时位于开发人员计算机上的 Silverlight 的内部版本号。

## OpenFileDialog.ShowDialog() 现在只能从用户启动的事件打开

出于安全原因，OpenFileDialog.ShowDialog() 只能在用户启动的事件（MouseLeftButtonDown/Up 和 KeyDown/Up）或者派生的用户事件（例如单击一个按钮或者选中某一复选框）期间调用。这使得 Silverlight 在行为上与其他主要浏览器和插件相同。

## Tab 键定位处理和关联的回调从异步更改为同步

在 Silverlight 2 中，存在与 Tab 键定位到页面的最后控件上相关联的错误。作为修复此错误的一个副作用，Tab 键定位处理和关联的回调（例如 **GotFocus** 事件）从异步更改为同步。（“异步”对 PostMessage 确实很重要。它在已处理当前消息后在同一个线程上发生。）这是一个重大更改，因为在 Silverlight 中的异步事件期间，您不能调入 HTML DOM 中。这是 WPF 的意外重新进入保护的 Silverlight 版本。在 Quirks 模式中，应用程序继续获取异步 Tab 处理，但对于页面上最后的控件不是很有效。

## 弹出项内的鼠标事件现在提供相对于弹出项的位置，而非相对于整个应用程序的位置

假定您在侦听鼠标事件的弹出项内具有一个元素：

[xaml]

<Popup>

<Border Name="border" MouseLeftButtonDown="OnMouseLeftButtonDown"/>

</Popup>

[c#]

void OnMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

… e.GetPosition(border) ...

}

在 Silverlight 2 中，MouseButtonEventArgs.GetPosition 的坐标将相对于整个 Silverlight 插件，即使应用程序要求相对于边框控件的坐标（针对 GetPosition 的参数应指示参考框架）也是如此。此问题已在 Silverlight 3 中修复。

## 弹出项现在只能具有一个父级

在 Silverlight 2 中，在某些情况下 **Popup** 实际上可具有两个父级。Silverlight 3 对于 **Popup** 只允许单个父级。

## 弹出项内的名称范围现在正常工作

Silverlight 2 具有若干错误：在 Popup.Child 位于自己的名称范围时（例如，使用 XamlReader.Load 创建了子级），弹出项内的命名项被置于错误的名称范围中。这导致了不应查找的 FindName() 和 Storyboard.TargetName 查找元素（因为 Silverlight 在错误的名称范围中查找），而非应查找的查找元素。在某些情况下，FindName() 将会查找已从树中删除的元素。此问题已在 Silverlight 3 中修复。正确的名称范围行为是在弹出项添加到活动树并且 Popup.Child 不是名称范围的所有者时，弹出项子级将成为弹出项父级的名称范围的一部分。否则，弹出项的子级将处于附加到 Popup.Child 的名称范围中。

## FindElementsInHostCoordinates 现在搜索弹出项

在 Silverlight 2 中，FindElementsInHostCoordinates 将不在弹出项内搜索。此问题已修复。

## 删除了与在 ItemsControl 中替换项相关联的多余事件

在 Silverlight 2 中，在 **ItemsControl** 中替换项将生成三个事件：**Remove**、**Add** 和 **Replace**。此行为与 WPF 不兼容并且很难进行编码。在您接收到 remove 和 add 事件时，不知道它是否是要遵循的真正的替换。在 Silverlight 3 中，只有 **Replace** 事件发生。

## ScrollViewer 现在将滚动条位置固定到 ScrollableWidth/ScrollableHeight

因此，ScrollViewer 内的内容的 Width 和 Height 将是零，直到完成了布局处理过程。

## 在设置了 Stretch 并且 Height 和/或 Width 未设置时，矩形和椭圆现在出现

在 Silverlight 2 中，如果 **Stretch != UniformToFill** 并且对 **ContentControl** 内的 **Rectangle** 或 **Ellipse** 未设置 **Height** 或 **Width**，则不会呈现该对象。在 Silverlight 3 中，如果 **Rectangle** 或 **Ellipse** 的 **Stretch** 属性并非 None，则会呈现该对象。

## TextBlock、TextBox 和 PasswordBox 的宽度现在向上舍入

在 Silverlight 2 中，如果您让 **PasswordBox**、**TextBlock** 和 **TextBox** 自动调整自身大小，则在文本的自然宽度不是整数并且其小数部分小于 0.5 时，它们将截断最后的字符。例如，测量结果为 96.34 单位宽度的文本将向下舍入到 96.00，因此 **TextBlock** 没有足够的空间来显示该文本。现在，该宽度将向上舍入，这可能会影响您的应用程序的布局。

## textblock.FontFamily = null 引发 ArgumentNullException

在 Silverlight 2 中，**textblock.FontFamily = null** 引发 **NullRefException**，现在它将引发 **ArgumentNullException**。

## 空格不再在模板内的 Run 元素之间呈现

在模板内，Silverlight 2 隐式在元素之间插入一个空格，因此，<TextBlock><Run Text="Hel"/><Run Text="lo"/></TextBlock> 将在“Hel”和“lo”之间有一个空格。在 Silverlight 3 中，此空格不再呈现。

## RadioButton.GroupName 现在在模板外搜索 RadioButtons

在 Silverlight 2 中，如果您在模板内使用 **RadioButton.GroupName**，Silverlight 只会在该模板内查找具有相同组名称的 RadioButtons。这有别于 WPF，并且导致它很难执行许多有用的方案。因此，在 Silverlight 3 中，**RadioButton** 还在模板外搜索。

## ComboBox 内容现在令 IsHitTestVisible="false"

在 Silverlight 2 中，ComboBox 内的所选项将接收鼠标事件。这会妨碍 ComboBox 为鼠标悬停状态显示正确的视觉效果，因此，对于其控件模板内的 ContentPresenter，ComboBox 现在将设置 IsHitTestVisible="false"。ComboBox 本身继续获取鼠标事件。

## TextBox 现在注重 Opacity 属性

在 Silverlight 2 中，Opacity 属性被忽略（始终视为 1）。此问题已修复。

## 具有空参数的 FrameworkElementAutomationPeer ctor 引发 NullReferenceException

在 Silverlight 2 中，具有空参数的 **FrameworkElementAutomationPeer** ctor 引发 **NullReferenceException**。现在，它将引发 **ArgumentNullException**。

## HyperlinkButton 现在将导航整个窗口，而不只是 Silverlight 位于其中的 IFRAME

在 Silverlight 2 中，HyperlinkButton 的行为在各浏览器之间不一致。在 Internet Explorer 中，HyperlinkButton 导航整个浏览器，而不只是当前的 IFrame。

在其他浏览器中，HyperlinkButton 只导航当前的 IFrame，这与 HTML 行为一致。在 Silverlight 3 Beta 中，Internet Explorer 内的这一行为已更改，以便与其他浏览器中的行为相符。

## 更正了在 Silverlight v2+ 应用程序不具有 XAP 文件扩展名时使用的基 URI

使用 Silverlight 2 插件，如果您重命名了某一 .xap 文件以具有不同的扩展名，Silverlight 会错误地认为该应用程序是为 Silverlight 1 设计的，并且 Silverlight 会在版本 1 兼容模式中运行它。这样做的后果之一是基 URI 被视作 HTML 页，而非重命名的 .xap。其他后果是默认字体不同于 .xap。此问题已修复。

## Application.Current.Host.Source 现在即使是在 .xap URL 具有查询字符串时也会返回绝对 URL

在 Silverlight 2 中，如果从其中加载 .xap 的 URL 包含查询字符串，则 Application.Current.Host.Source 可能返回相对 URL。在 Silverlight 3 中，此问题已修复，Application.Current.Host.Source 总是会返回绝对 URL。.xap 的 URL 通常会被指定为 <object> 标记内的 <param>，下面是带有查询字符串的 .xap URL 的示例：

<param name="source" value="Bin/Debug/bug24857.xap?a=b"/>

## ResourceDictionary 枚举错误已修复

在 Silverlight 2 中，如果您将某一枚举放置于 **ResourceDictionary** 中并再次将其取出，您取回的是 Integer32 类型的对象，而非预期的枚举类型。此问题已在 Silverlight 3 中修复。

## ReadOnlyObservableCollection 已从 System.Windows.Controls.Data.dll 迁移到 System.Windows.dll

**ReadOnlyObservableCollection** 类已移入 **System.Windows.dll**，以便 System.Windows.dll 中的其他 API 可以使用它。为了将对下载大小的影响降至最低，某些不常用的方法已删除。

请注意，这将不会影响为 Silverlight 2 编译的应用程序。如果您在 Silverlight 2 中使用 ReadOnlyObservableCollection，则您的 .xap 文件将包含 System.Windows.Controls.Data.dll 的副本。在 Silverlight 3 Beta 运行您的 .xap 时，它将使用该 .xap 的 System.Windows.Controls.Data.dll 副本以及 ReadOnlyObservableCollection 的该版本，而非在 Silverlight 3 Beta 的 System.Windows.dll 中随附的版本。

## DataGridEndingEditEventArgs 已从 SDK 中删除

没有返回此 **EventArgs** 的事件，因此它已被删除。

## PollingDuplexHttpBinding 更改

**PollingDuplexHttpBinding** 现在从 **HttpBinding** 派生，而非从 **BasicHttpBinding** 派生。**PollingDuplexHttpBinding** 现在使用 SOAP 1.2，而非 1.1。因此，Silverlight 3 **PollingDuplexHttpBinding** 将不会与运行 **System.ServiceModel.PollingDuplex.dll** 的 Silverlight 2 版本的 ASP.NET 服务器共用。

## 与删除事件处理程序有关的错误修复

通常，在删除一个事件侦听器（C# 中的 -=）时，会删除特定事件的特定委托。Silverlight 2 忽视了事件部分 –“foo.Bar -= Baz”将删除捆绑到 Baz 方法的所有事件侦听器，包括 Bar 外的所有事件。在 Silverlight 3 中，此问题已修复。

## 错误修复：允许 <Cursors>，而非 <Cursor>

在 Silverlight 2 中，允许以下语句：

<DiscreteObjectKeyFrame.Value>

<Cursors>Hand</Cursors> <-------note plural

</DiscreteObjectKeyFrame.Value>

这是不正确的，因为类型的名称是 Cursor，而非 Cursors。Silverlight 3 只允许 <Cursor>。

## XAML 中的自定义附加属性要求 xmlns 为前缀

在 Silverlight 2 中，以下非法 XAML 为 Silverlight 所接受：

<local:MyControl MyControl.DesignerID="12" />

此问题在 Silverlight 3 中已修复。正确的 XAML 对于附加的属性使用 XML 前缀：

<local:MyControl **local:**MyControl.DesignerID="12" />

## 错误修复：子类控制可能会加载错误的资源

在 Silverlight 2 中，在某些情况下，如果类 A 是在一个用户程序集中定义的，而标记继承自不同用户代码程序集中的类 B，则 Application.LoadComponent(A) 将尝试从类 B 的程序集而非类 A 的程序集加载 XAML。此问题已修复。

## 错误修复：在路径上的叶对象为空时不调用 ValueConverter

Silverlight 2 数据绑定在其处理空值的方式上不一致。如果您指定 Path=a.b.c.d 并且 a、b 或 c 为空，则 Silverlight 2 调用 ValueConverter。但是，如果您指定 Source=null，则它不调用值转换器，即使它在逻辑意义上相同。Silverlight 3 将始终调用值转换器。

## Thumb 控件现在相对于绝对位置计算拖放，而非相对于其父级计算拖放

在父元素由于拖动而移动时，这种改动最明显。

## 不再支持在 SetBinding() 后设置 Binding 属性

在 Silverlight 2 中，可以通过编程方式对在调用 SetBinding() 后、但在设置 DataContext 之前的 Binding 设置不同的属性：

Binding binding = new Binding();

myElement.SetBinding(HeightProperty, binding);

binding.Foo = value;

myElement.DataContext = …;

不同的属性在 SL2 中具有不同的行为：

* Converter、ConverterParameter、ConverterCulture、NotifyOnValidationError 和 ValidatesOnExceptions 属性会追溯更改使用 SetBinding() 建立的绑定。
* Mode 和 Source 属性将被忽略（在调用 SetBinding() 时复制其值）

如果在调用 SetBinding() 后更改了这些属性，Silverlight 3 将引发异常。（如果在设置 DataContext 后更改了这些属性，Silverlight 2 将引发类似的异常）

## ContentPresenter 的子级现在只能具有一个父级

在 Silverlight 2 中，在某些情况下，ContentPresenter 中的内容（子元素）可能实际具有两个父级，这是因为该内容已添加到第二个 ContentPresenter 并且没有从第一个 ContentPresenter 中删除它。如果某一元素添加到第二个父级，且没有从第一个父级中删除它，则 Silverlight 3 将引发一个异常。

## 指向 .xap 外部资源的相对 URL 现在是相对于 .xap 文件而不是 HTML 页面

假定 HTML 页面和 xap 文件在不同目录中（例如，http://example.com/ 和 http://example.com/ClientBin/），而您有一个未包含在 xap 文件内的指向某资源的相对 URL（例如 images/foo.jpg）。SL2 会相对于 HTML 页面的位置解释该 URL (http://example.com/images/foo.jpg)。为了比较容易地将 Silverlight 应用程序移动到不同位置，Silverlight 3 则会相对于 .xap 的位置来解释 URL (http://example.com/ClientBin/images/foo.jpg)。