



Como viene siendo norma, en el recién finalizado evento MIX, celebrado en Las Vegas, Estados Unidos, se anunció la liberación de la beta de Silverlight 5, con un montón de novedades muy interesantes, que presentamos aquí.

**C**al y como Microsoft ya había anunciado a principios del pasado diciembre en evento online **Silverlight Firestarter**, en el marco del MIX'2011 se liberó la beta de **Silverlight 5**, añadiendo nuevas características que incrementan sustancialmente el rendimiento y la productividad de la plataforma, tanto a nivel del *plug-in* como para los que desarrollamos con ella. Gran parte de estas novedades aparecen como respuesta a peticiones de la comunidad mediante el *uservoice*<sup>1</sup> disponible en la URL http://dotnet.uservoice.com/forums/4325-silverlight-feature-suggestions, una demostración de democracia digital que parece estar funcionando muy bien y que desde aquí animo al lector a visitar.

Antes de ver las novedades aparecidas en la beta de Silverlight 5, debemos tener en cuenta que estamos hablando de una versión de prueba, por lo que no es recomendable desplegarla en una máquina de producción. Es mucho mejor poder disponer de una máquina física sobre la que instalar las betas, por rendimiento y capacidad; pero si no, siempre podemos montar una máquina virtual para ello, aunque no disfrutaremos de las capacidades 3D, ya que éstas se basan en un acceso directo a la GPU.

Para empezar, descargue e instale la beta de Silverlight 5 y las **Silverlight 5 Beta Tools for Visual Studio SP1** desde la página oficial [1], donde adicionalmente encontrará otras descargas muy interesantes como **Expression Blend Preview for Silverlight 5**, **Silverlight 5 Beta SDK** y **WCF RIA Services for Silverlight 5 Beta**.

en nuestro caso, tecnología.

Entrando en materia, a continuación vamos a ver en detalle y de forma ejemplificada algunas de las principales novedades que ofrece la beta de Silverlight 5.

## Herramientas

Aparte de las ya citadas herramientas para que Visual Studio SP1 soporte Silverlight 5 (por supuesto, mediante *multi-targeting*; es decir, podremos continuar desarrollando para otras versiones de Silverlight que ya tuviéramos instaladas anteriormente), tenemos como principal novedad en esta área la depuración de los enlaces a datos (*bindings*) en XAML.

Anteriormente, la forma de realizar una depuración de los enlaces a datos se basaba en implementar un conversor ad-hoc y poner puntos de interrupción (*breakpoints*) dentro del mismo.

Veamos cómo lograr esto de manera más efectiva en Silverlight 5 a través de un ejemplo.

Para ello, crearemos una nueva aplicación Silverlight 5, a la que añadiremos dos clases: **Cursos-Model**, que albergará el contenido del curso (en este caso, una simple propiedad de tipo **string**), y **Cursos\_ViewModel**, que contendrá una propiedad de tipo **CursosModel**. En el *code-behind* de **MainPage.xaml** declararemos una variable de tipo **Cursos\_ViewModel**, que asignaremos al **DataContext** de **MainPage** en su constructor. El código puede verse en el listado 1.

Seguidamente, en el XAML de MainPage, definimos un StackPanel que contiene un TextBlock,

José Luis Latorre *CEO de Brainsiders v* 

profesor de Aula Vulcan. Microsoft MVP & MCP



Uservoice es un servicio de opinión abierta en el que una comunidad (usuarios, clientes, etc.) se convierten en "prosumers" (consumidores y productores a la vez), y que ayuda a definir las necesidades y requisitos de un servicio o,

```
(CursosModel.cs)
public class CursosModel
{
    public String NombreDelCurso { get; set; }
}
(Cursos ViewModel.cs)
public class Cursos ViewModel
    public CursosModel CursoSeleccionado
        get; set;
    public Cursos_ViewModel()
        CursoSeleccionado = new CursosModel() {
            NombreDelCurso = "Introducción a Silverlight 5"
        };
    }
}
(MainPage.xaml.cs)
public partial class MainPage : UserControl
    private Cursos ViewModel Cursos = new Cursos ViewModel();
    public MainPage()
        InitializeComponent();
```

```
this.DataContext = Cursos;
}
```

Listado 1

```
<Grid x:Name="LayoutRoot"
Background="White">
<StackPanel>
<TextBlock Text=
"{Binding NombreDelCurso}">
</TextBlock>
</StackPanel>
</Grid>
```

Listado 2

al que enlazamos al campo **NombreDel-Curso** (ver listado 2).

Si ubicamos un punto de interrupción en la línea de XAML que contiene el enlace a datos, veremos que se marca un punto de depuración y se resalta el *binding* en sí. Y si ejecutamos la aplicación, veremos que ésta se detiene exactamente en este punto y nos indica, además, un error, como se puede apreciar en la figura 1. Puede observar cómo el punto de depuración es resaltado de forma similar a cuando depuramos código, y ofrece una información muy interesante sobre el estado del *binding* (**BindingState**), en la que podemos observar la acción que se está realizando, "Updating Target" (actualizando el destino), así como los detalles del error, donde podemos ver que Silverlight no ha hallado la propiedad; además de todo ello, también se nos muestra el **FinalSource** del enlace a datos: ahora es cuando nos damos cuenta de que nos hemos olvidado de mencionar el objeto que contiene a esa propiedad, con lo que el *binding* correcto debería ser:

Text="{Binding CursoSeleccionado.
 NombreDelCurso}"

## **Medios**

Silverlight 5 Beta ofrece toda una serie de mejoras en la parte de medios como son: mejoras en la API de sonido y la nueva clase SoundEffect para la ejecución de sonidos de baja latencia; la tecnología TrickPlay, para alcanzar velocidades variables en la reproducción, lo que, conjuntamente con un ajuste dinámico del tono de la voz, nos permite ver vídeos a 2x, entendiendo perfectamente las narraciones; y, además, la decodificación por hardware y la presentación de H.264 con rendimiento mejorado, basado en aceleración por GPU. Para la *release* final también se esperan mejoras en DRM (Digital Rights Management), así como el soporte para sistemas de control remoto.

	a 🖞 CurrosModel.cs 🖞 MainPagasami 4 🔀			IntellTrace + 1
			á	
				IntelliTrace is not collecting data for this debugging ression.
2 Design 14 / P II 2000. Conserved average as Classes "Des (Schasse") and an entry (J/consense, allow and on entry (J/consense, allow and on entry (J/consense, allow) and (Joseph and Joseph and Joseph and Joseph and (Joseph and Joseph a	<pre>training.mining." training.mining</pre>		1000 + - - -	The project type may well be suppreted as process pairs of the displaying only have been added to display the displaying only have been to display the displaying only have been to display the displaying only and the displaying only and the displaying only and the displaying only and the displaying only and the displaying only and all by the process being displaying only and all by the process being displaying only and all by the process being displaying and the displaying only and the displaying only and the displaying and the displaying and the d
		A R A Incordict Victor		
teres	New Spectra Display Systems Data Series Devide Spectra Display Devide Spectra Display Spectra Display Devide Spectra Display Devide Spectra Display Devide Spectra Display Devide	Type 1 per 1 per 1 per 1 System Visionan D System Visionan D		



```
private void ReproducirSonido() {
   SoundEffect Sonido;
   var SonidoStream = Application.GetResourceStream(
        new Uri("beep.wav", UriKind.RelativeOrAbsolute));
   Sonido = SoundEffect.FromStream(SonidoStream.Stream);
   SoundEffectInstance instance = Sonido.CreateInstance();
   instance.Play();
}
```

Listado 3

### La nueva clase SoundEffect

Una de las más interesantes es la reproducción de sonidos de baja latencia, lo que es vital para cuando queremos reproducir de forma inmediata un sonido, por ejemplo, en respuesta a una acción del usuario. Esto se ha conseguido mediante una clase que "sonará" bastante a todos aquellos que se hayan introducido en el desarrollo para Windows Phone: la clase **SoundEffect**, proveniente de XNA, que es de uso obligatorio para efectos de sonido de baja latencia para juegos o interfaces en este dispositivo móvil. Esta clase, así como otras de XNA, han sido integradas dentro del núcleo de Silverlight 5, con lo que no necesitamos agregar referencias a librerías externas.

Como ejemplo, agregaremos al proyecto creado anteriormente un archivo de tipo .WAV, en modo "contenido", como puede observarse en el código que acompaña a este artículo. En el código, cargaremos el archivo mediante el método GetResourceStream del objeto Application, para continuar pasándole el Stream obtenido al método FromStream de la clase SoundEffect, como puede apreciarse en el listado 3.

Aunque esto pueda parecer sencillo, SoundEffect requiere de archivos .WAV codificados en modo PCM, 8 ó 16 bits en mono o estéreo, y fallará si se le suministran otros formatos, siendo estas limitaciones las mismas que en XNA o en WP7.

Otro detalle interesante es que esta clase nos permite controlar varias características del sonido a reproducir: volumen, *pitch* (la frecuencia, que determina si un sonido es agudo o grave) y *pan* (el balance del sonido, que emula el dial de los equipos de audio y nos permite ajus-

Parámetro	Valores
Volume	o a 1, siendo 1 el más alto posible.
Pitch	-1 a 1, siendo -1 la frecuencia más baja y 1 la más alta.
Pan	-1 a 1, siendo -1 el desplazamiento de la potencia totalmente a la izquierda y 1 el desplazamiento absoluto a la derecha.

```
Tabla 1
```

```
SoundEffect SonidoDeFondo;
SoundEffect SonidoClick;
private void InicializarSonidos()
    var SonidoStream = Application.GetResourceStream(
        new Uri("beep.wav", UriKind.RelativeOrAbsolute));
    SonidoClick = SoundEffect.FromStream(SonidoStream.Stream);
    var SonidoStream2 = Application.GetResourceStream(
        new Uri("Loop Banjo.wav", UriKind.RelativeOrAbsolute));
    SonidoDeFondo = SoundEffect.FromStream(SonidoStream2.Stream);
}
void MainPage_Loaded(object sender, RoutedEventArgs e)
{
    InicializarSonidos();
}
private void button1_Click(object sender, RoutedEventArgs e)
    SoundEffectInstance instance = SonidoClick.CreateInstance();
    instance.Play();
}
private void button2_Click(object sender, RoutedEventArgs e)
ł
    SoundEffectInstance instance = SonidoDeFondo.CreateInstance();
    instance.IsLooped = true;
    instance.Pitch = -0.5f;
    instance.Volume = 0.2f;
    instance.Play();
}
```

Listado 4

tar la potencia de los ejes izquierdo y derecho del mismo). A las correspondientes propiedades pueden asignársele valores en los rangos que se indican en la tabla 1.

msdn

ya que se trata de un sonido "de fondo". A un segundo botón asociaremos la reproducción de otro sonido. Todo ello podemos verlo en el listado 4.

Un ejemplo práctico de aplicación de estas propiedades sería disponer de un sonido con un volumen bajo ejecutándose permanentemente, como suelen hacer los sonidos de fondo. Para ello, asociaremos a un botón una reproducción de sonido asociándole **true** a la propiedad **IsLooped**, un valor bajo a **Volume** y un pitch también bajo, Como observación, si el botón **button2** se pulsa varias veces, estaremos iniciando el sonido de fondo repetidamente, creando diferentes instancias del mismo efecto de sonido.

#### Tecnología TrickPlay

Continuando con las mejoras en la reproducción de medios, toca el turno de presentar **TrickPlay**, que mucha gente considera como uno de los mejores atributos de Silverlight 5. Fundamentalmente, ahora disponemos en el control **MediaElement** de una nueva propiedad, **PlaybackRate**, a la que podremos asignar un valor en el rango de 0.25 a 2, siendo 1 la velocidad normal de reproducción. TrickPlay también añade una corrección de la frecuencia del sonido para escucharlo debidamente incluso a velocidades altas – ya que normalmente se escucha como si el ponente hubiese aspirado helio con antelación. Como detalle, en la beta la corrección de frecuencia aún no está habilitada, así que solo notaremos los cambios en la velocidad de reproducción.

Como ejemplo, añadiremos al proyecto un **MediaE**lement asociado a la reproducción automática de un vídeo, seguido de un control tipo *slider*, al que asignaremos como valor mínimo 0.25 y máximo 2, y que utilizaremos como origen de un enlace a datos desde el control **MediaElement**; concretamente, a la propiedad **PlaybackRate** del mismo. Ello hará que podamos cambiar la velocidad del video sobre la marcha directamente desde la interfaz de usuario. El código necesario puede verse en la figura 2.



## Texto

La renderización y el proceso de disposición (*layout*) del texto ha sido uno de los puntos menos fuertes de Silverlight, al menos hasta la fecha. Silverlight 5 nos llega

con unas sustanciales mejoras en el rendimiento de la renderización y disposición del texto, así como en la calidad y claridad del mismo, algo que podemos apreciar en la comparativa de la figura 3.

for Wintows Phone 7. It's fully hardware-accelerated; it ena he default look-and-feel of the Windows Phone app [light text f. What this means is, you can build killer applications for this programming model is the same Silverlight you already know 's the same set of tools. This isn't 'Silverlight Lite,' this isn't 'S

ns for Windows Phone 7. It's fully hardware-accelerated; it e ther the default look-and-feel of the Windows Phone app [lig purself. What this means is, you can build killer applications Phone programming model is the same Silverlight you alr same code, it's the same set of tools. This isn't 'Silverlight

Figura 3

Las mejoras en la disposición del texto nos permiten ahora construir *layouts* similares a los de una revista, ya sea permitiendo ubicarlo en múltiples columnas en las que el texto fluya, definir en detalle el espaciado del mismo, y a la vez, mediante *Pixel Snapping*, presentarlo de una forma más clara. También disponemos de soporte ampliado para fuentes Open Type, y una optimización significativa en cuanto a su rendimiento.

Pero veamos cómo funciona todo esto en la práctica. Por un lado, disponemos de una nueva propiedad, **Run.CharacterSpacing**, en los controles **TextBlock** y **RichTextBlock**, que nos ofrece control sobre el espaciado de los caracteres contenidos en dicho elemento. Un ejemplo se muestra en el listado 6, cuyo resultado se puede apreciar en la figura 4.



Figura 4

Con respecto a las capacidades renovadas en lo que a la disposición del texto se refiere, tenemos el nuevo control **RichTextBoxOverflow**, que nos permite enlazar diferentes contenedores de texto de manera que éste fluya hacia otro elemento cuando no quepa más en el mismo. Con ello podemos conseguir disponer el texto en múltiples columnas que se ajustan dinámicamente al espacio disponible. Este control puede asociarse únicamente a un control **Rich-TextBox**, que encadenará su contenido mediante una



```
<StackPanel>
  <TextBlock FontSize="18" CharacterSpacing="250"
             Text="Three Rings for the Elven-kings under the sky," />
  <RichTextBox FontSize="18" CharacterSpacing="200" BorderBrush="{x:Null}"</pre>
               BorderThickness="0">
     <Paragraph>Seven for the Dwarf-lords in their halls of stone,</Paragraph>
  </RichTextBox>
  <TextBlock FontSize="18">
    <Run CharacterSpacing="150"
      Text="Nine for Mortal Men doomed to die," />
  </TextBlock>
  <TextBlock FontSize="18">
    <Run CharacterSpacing="100"
      Text="One for the Dark Lord on his dark throne" />
  </TextBlock>
  <TextBlock FontSize="18">
    <Run CharacterSpacing="50"
      Text="In the Land of Mordor where the Shadows lie." />
  </TextBlock>
  <TextBlock FontSize="18">
   <Run CharacterSpacing="0"
      Text="One Ring to rule them all, One Ring to find them," />
  </TextBlock>
  <TextBlock FontSize="18">
    <Run CharacterSpacing="-50"
      Text="One Ring to bring them all and in the silver light bind them" />
  </TextBlock>
  <TextBlock FontSize="18">
    <Run CharacterSpacing="-100"
      Text="In the Land of Developers where the Shadows lie." />
  </TextBlock>
</StackPanel>
```

Figura 6

## **Enlace a datos**

En Silverlight 5 dispondremos de plantillas de datos (*Data Templates*) implícitas, que nos permitirán asociar un plantilla de datos a un tipo de datos concreto; tendremos por fin entre nosotros a **Ancestor RelativeSource**, que a partir de una plantilla nos permitirá establecer un enlace a

#### Listado 6

nueva propiedad, **OverflowContentTar**get; si el texto no cabe en el control "normal", el texto sobrante se presentará en el control enlazado con **Over**flowContentTarget. En el listado 7, se dispone de un control **RichTextBox** llamado **RTB1**, cuya propiedad **Overflow-ContentTarget** se enlaza mediante un *binding* a un control **RichTextBoxOver**flow de nombre **RTBOverflow2**; éste, a su vez, canaliza su *overflow* hacia un segundo **RichTextBoxOverflow** de nombre **RTBOverflow3**.

El resultado podemos verlo en las figuras 5 y 6. Recomendamos al lector que descargue el ejemplo, disponible tanto en la web de **dotNetManía** como en mi blog [2], pruebe a ajustar el tamaño de la ventana del navegador, y vea como fluye el texto, ajustándose de forma dinámica y muy rápidamente a los cambios de su tamaño.

```
<RichTextBox x:Name="RTB1"
             OverflowContentTarget="{Binding ElementName=RTBOverflow2}"
             HorizontalAlignment="Stretch"
             VerticalScrollBarVisibility="Disabled"
             HorizontalScrollBarVisibility="Disabled"
             BorderThickness="0" TextAlignment="Justify">
    <Paragraph> Lorem ipsum ... </Paragraph>
    <Paragraph>...</Paragraph>
</RichTextBox>
<RichTextBoxOverflow x:Name="RTBOverflow2"
                     OverflowContentTarget="{Binding ElementName=RTBOverflow3}"
                     VerticalScrollBarVisibility="Disabled"
                     HorizontalScrollBarVisibility="Disabled"
                     BorderThickness="0"
                     Grid.Column="1"
                     1>
<RichTextBoxOverflow x:Name="RTBOverflow3"
                     VerticalScrollBarVisibility="Disabled"
                     HorizontalScrollBarVisibility="Disabled"
                     BorderThickness="0"
                     Grid.Column="2"
                     1>
```

Listado 7



datos a una propiedad del elemento que lo contiene, algo muy necesario y vital para todo tipo de aplicaciones; dispondremos de extensiones de marcado (*Markup Extensions*) personalizables, lo que nos permitirá crear mediante código extensiones personalizadas para, por ejemplo, añadir localización; y, finalmente, podremos realizar enlace a datos en los *setters* de estilos.

#### Plantillas de datos implícitas

Silverlight 5 ofrece plantillas de datos implícitas, que nos permiten asociar una plantilla de datos a un tipo de datos; es decir, elegir cómo se presenta un tipo de datos concreto en base a éste. Como ejemplo, crearemos una colección de entidades de tipo **IPersona**, que implementaremos en dos clases diferentes, **Geek** y **MVP**. A continuación, enlazamos la colección para ser visualizada en el siguiente **ListBox**:

```
<ListBox HorizontalAlignment="Stretch"
Margin="10,10,10,10"
Name="listBox1"
VerticalAlignment="Stretch"
ItemsSource="{Binding ocPersonas}" />
```

Si ejecutamos ahora la aplicación sin asignar plantillas de datos a los tipos, lo que visualizaremos será la representación predeterminada de la clase, resultado de aplicar el método **ToString()**, como podemos ver en la figura 7.

(→) ② @ http://localhost: ♀ = ≧ ♂ × @ DNM_SL5_ImplicitDat × 1
File Edit View Favorites Tools Help 🍰 🖮 Microsoft Research Kin 🔻 🎧 Sitios sugeridos 🔻 🗿 <b>Más complementos </b>
DNM_SL5_ImplicitDataTemplates.Geek DNM_SL5_ImplicitDataTemplates.Geek DNM_SL5_ImplicitDataTemplates.MVP DNM_SL5_ImplicitDataTemplates.Geek
DNM_SL5_ImplicitDataTemplates.Geek
Figura 7

Para generar una visualización adecuada, podemos crear dos plantillas de datos, una para el tipo **Geek** y otra para el tipo **MVP**. Básicamente, las plantillas de datos ahora tienen una nueva propiedad **DataType**, a la que se puede asociar el tipo de datos al que se debe aplicar. Agregando las dos plantillas que se muestran en el listado 8 y sin cambiar un ápice el código XAML del **ListBox**,

obtendremos una presentación completamente diferente, como se puede observar en la figura 8.

```
<DataTemplate DataType="idc:Geek">
    <StackPanel Orientation="Horizontal">
        <TextBlock>[GEEK!] - Nombre:</TextBlock>
        <TextBlock Text="{Binding Nombre}"></TextBlock>
        <TextBlock Text=" | Especialidad: "></TextBlock>
        <TextBlock Text="{Binding Especialidad}"></TextBlock>
    </StackPanel>
</DataTemplate>
<DataTemplate DataType="idc:MVP">
    <StackPanel Orientation="Horizontal" Background="Yellow">
        <TextBlock>[MVP!!] - Nombre: </TextBlock>
        <TextBlock Text="{Binding Nombre}"></TextBlock>
        <TextBlock Text=" | Especialidad: "></TextBlock>
        <TextBlock Text="{Binding Especialidad}"></TextBlock>
    </StackPanel>
</DataTemplate>
```

Listado 8



Figura 8

En Silverlight 5 dispondremos de plantillas de datos implícitas, que nos permitirán asociar un plantilla de datos a un tipo de datos concreto

#### Ancestor RelativeSource

Hasta ahora, la propiedad **RelativeSource** de los bindings solo permitía **Self** y **TemplatedParent**, pero al ampliarse con una nueva opción, **Ancestor**, nos acercamos bastante al modelo de enlace a datos de WPF, algo muy positivo. Un ejemplo de ello se puede ver en el listado 9.



<DataTemplate>

<TextBox Width="{Binding Width RelativeSource={RelativeSource AncestorType=Border, AncestorLevel=5}}" />

</DataTemplate>

Listado 9

#### Extensiones de marcado

En cuanto a las extensiones de marcado, éstas fundamentalmente nos permiten ejecutar código durante la interpretación del código XAML, tanto para propiedades como para manejadores de eventos, lo que las hace ideales para mejorar arquitecturas MVVM. Con Silverlight 4 teníamos cinco tipos diferentes:

- Binding Text="{Binding Path=FirstName, Mode=TwoWay}"
- StaticResource
   Foreground="{StaticResource
   BlueSolidColor}"
- TemplateBinding
   VerticalAlignment="{TemplateBinding
   VerticalContentAlignment}"
- NullExtension
   Background="{x:Null}"
- RelativeSource DataContext="{Binding RelativeSource={RelativeSource TemplatedParent}}"

La implementación de nuevas extensiones de marcado es realmente directa y requiere una simple derivación de la clase **System.Windows.Markup.MarkupExtension**: sobreescribir el método **ProvideValue**. A observar, que el nombre de toda clase que herede de **MarkupExtension** deberá finalizar con "**Extension**", de forma similar a lo que ocurre con la denominación de las propiedades de dependencia.

Una implementación de una extensión muy sencilla la podemos ver en el listado 10. Un posible uso en de ella en XAML sería:



Controles

Listado 10

Listado 11

La implementación de nuevas extensiones de marcado es realmente directa y solo requiere una simple derivación de la clase System.Windows.Markup. MarkupExtension

<TextBlock FontSize="18" Margin="10" Text="{lcl:ss Text='hola dnm'}"> </TextBlock>

#### Enlace a datos en los setters de estilos

Por último, queda resaltar el novedoso uso de *bindings* en los *setters* de estilos, ya que, fundamentalmente, nos permitirán



detectar un doble clic de manera muy fácil. El listado 12 nos muestra cuán sencillo es utilizar esta nueva propiedad. Por otro lado, Silverlight 5 ofrece soporte para múltiples ventanas, que nuestra aplicación podrá mostrar y permitir interactuar con ellas de forma simultánea, en el caso de las aplicaciones OOB (*Out-Of- Browser*), que se ejecutan fuera del navegador.

referenciar otras propiedades, como pode-

mos ver en el código XAML del listado 11.

En lo relativo a los controles, se ofrecen varias mejoras generales, como la propiedad **ClickCount**, que nos ayuda a

# dotNetManía

```
private void OnMouseDownClickCount(object sender, MouseButtonEventArgs e)
{
    if (e.ClickCount == 1)
    {
        // Single click occurred
     }
    if (e.ClickCount == 1)
     {
        // Double click occurred
     }
    if (e.ClickCount == 1)
     {
        // Triple click occurred
     }
}
```



## **Gráficos 3D**

Una de las demostraciones más espectaculares del Silverlight Firestarter del pasado diciembre fue la relativa a las capacidades 3D. En este caso no se trata de una mejora, característica o funcionalidad, sino de una API totalmente nueva que añade, literalmente, una nueva dimensión a Silverlight.

En cualquier caso, tenemos ante nosotros una innovadora API 3D, acelerada por GPU, que nos proporciona las capacidades necesarias para construir avanzadas visualizaciones y experiencias de usuario.

# Aplicaciones de confianza

Algunas características de las aplicaciones de confianza se permiten ahora en el navegador sin que éstas deban ser instaladas, obviamente requiriendo que el fichero **XAP** disponga de permisos elevados en su seguridad. Las aplicaciones que se ejecuten fuera del navegador (OOB) podrán acceder al sistema de ficheros del usuario (hasta ahora, solo podían acceder a la carpeta **Mis Documentos**). También se ha añadido soporte de interoperabilidad COM y llamadas P/Invoke para aplicaciones de confianza.

# Y aún hay más...

- Una mejorada latencia de red mediante un interesante cambio en la arquitectura, utilizando un hilo en *background* para las funciones de red.
- Un *parser* de XAML mejorado que acelera el arranque, así como la ejecución de Silverlight en general.
- Soporte para sistemas operativos de 64 bits.
- Impresión vectorial.
- Aceleración por hardware en el modo "sin ventana" (*windowless*) en Internet Explorer 9.
- Conciencia de la energía, para poder decidir acciones como no poner el equipo en hibernación o mitigar la pantalla si se está viendo una película.
- Mejoras en el control **PivotViewer**.

## **Conclusiones**

Nos hemos dejado en el tintero presentar en mayor detalle algunas de las novedades en Silverlight 5, pero es que son realmente muchas; hasta podríamos decir que algunas de ellas, como las capacidades 3D de Silverlight 5, se merecen uno o más artículos completos. En cualquier caso, animo desde aquí al lector a investigar y probar esas nuevas características, que en poco tiempo estarán totalmente a nuestra disposición.

## Referencias

[1] Descarga de la beta de Silverlight 5: http://www.silverlight.net/getstarted/silverlight-5-beta/.

[2] Blog de José Luis Latorre: http://geeks.ms/blogs/jllatorre.

