

Información general sobre documentos dinámicos

Artículo

18/04/2022

Tiempo de lectura: 28 minutos

1 colaborador

Los documentos dinámicos están diseñados para optimizar su visualización y legibilidad. En lugar de establecer un diseño predefinido, los documentos dinámicos ajustan y redistribuyen dinámicamente su contenido basándose en variables en tiempo de ejecución, como el tamaño de la ventana, la resolución del dispositivo y las preferencias opcionales del usuario. Además, los documentos dinámicos ofrecen características de documento avanzadas, como paginación y columnas. En este tema se proporciona información general sobre los documentos dinámicos y cómo crearlos.

Qué es un documento dinámico


Un documento dinámico está diseñado para "redistribuir el contenido" según el tamaño de la ventana, la resolución del dispositivo y otras variables del entorno. Además, los documentos dinámicos tienen varias características integradas, como la búsqueda, los modos de visualización que optimizan la legibilidad, y la capacidad de cambiar el tamaño y el aspecto de las fuentes. Los documentos dinámicos se utilizan mejor cuando la facilidad de lectura es el escenario principal de consumo del documento. En cambio, los documentos fijos están diseñados para tener una presentación estática. Los documentos fijos son útiles cuando la fidelidad del contenido de origen es esencial.

En la ilustración siguiente se muestra un ejemplo de documento dinámico visualizado en varias ventanas de tamaños diferentes. A medida que cambia el área de visualización, el contenido se redistribuye para aprovechar mejor el espacio disponible.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula tortor, dapibus et, facilisis a, aliquet et, diam. Cras convallis. Fusce at urna. Quisque interdum, turpis sed dictum fringilla, magna arcu gravida libero, elementum tincidunt dolor mauris sed erat. Curabitur egestas aliquet nibh. Etiam quis sem in lorem auctor consequat. Suspendisse vitae lorem. Proin eleifend elementum ligula. Donec mattis consectetur erat. Donec fermentum consectetur neque. Suspendisse tempus faucibus magna. Pellentesque sodales velit eget nibh. Phasellus velit magna, malesuada vitae, rhoncus eget, dignissim ut, metus. Praesent pulvinar suscipit diam.

1 of 4


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula tortor, dapibus et, facilisis a, aliquet et, diam. Cras convallis. Fusce at urna. Quisque interdum, turpis sed dictum fringilla, magna arcu gravida libero, elementum tincidunt dolor mauris sed erat. Curabitur egestas aliquet nibh. Etiam quis sem in lorem auctor consequat. Suspendisse vitae lorem. Proin eleifend elementum ligula. Donec mattis consectetur erat. Donec fermentum consectetur neque. Suspendisse tempus faucibus magna. Pellentesque sodales velit eget nibh. Phasellus velit magna, malesuada vitae, rhoncus eget, dignissim ut, metus. Praesent pulvinar suscipit diam.



Morbi ut tellus at tellus semper consectetur. Nullam fringilla nonummy justo. Proin rutrum, purus id adipiscing malesuada, enim velit accumsan nisi, pellentesque rhoncus nibh nisi ut magna. Nunc massa nulla, volutpat sit amet, pulvinar ac, scelerisque ac, magna. Nulla facilisi. Integer pharetra felis vitae risus. Nunc aliquet lacus pretium urna varius hendrerit. Duis odio nisl, venenatis at, sollicitudin eu, viverra sed, nibh. Nullam consequat. Duis felis felis, rutrum viverra, auctor eget, iaculis ut, mi. Integer sagittis pharetra ipsum. Donec lacinia scelerisque nisl. Nullam aliquet. In et sapien. Fusce blandit odio et mi.

1 of 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula tortor, dapibus et, facilisis a, aliquet et, diam. Cras convallis. Fusce at urna. Quisque interdum, turpis sed dictum fringilla, magna arcu gravida libero, elementum tincidunt dolor mauris sed erat. Curabitur egestas aliquet nibh. Etiam quis sem in lorem auctor consequat. Suspendisse vitae lorem. Proin eleifend elementum ligula. Donec mattis consectetur erat. Donec fermentum consectetur neque. Suspendisse tempus faucibus magna. Pellentesque sodales velit eget nibh. Phasellus velit magna, malesuada vitae, rhoncus eget, dignissim ut, metus. Praesent pulvinar suscipit diam.



Morbi ut tellus at tellus semper consectetur. Nullam fringilla nonummy justo. Proin rutrum, purus id adipiscing malesuada, enim velit accumsan nisi, pellentesque rhoncus nibh nisi ut magna. Nunc massa nulla, volutpat sit amet, pulvinar ac, scelerisque ac, magna. Nulla facilisi. Integer pharetra felis vitae risus. Nunc aliquet lacus pretium urna varius hendrerit. Duis odio nisl, venenatis at, sollicitudin eu, viverra sed, nibh. Nullam consequat. Duis felis felis, rutrum viverra, auctor eget, iaculis ut, mi. Integer sagittis pharetra ipsum. Donec lacinia scelerisque nisl. Nullam aliquet. In et sapien. Fusce blandit odio et mi.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Suspendisse laoreet condimentum purus. Etiam vel enim. Suspendisse at dolor. Pellentesque gravida. Aenean convallis. Vivamus diam. Aenean lorem libero, suscipit vel, tempor eu, fermentum aliquam, metus. Quisque volutpat urna at augue. Nam placerat. In viverra congue tellus. Proin auctor. Fusce nisl lorem, dapibus vitae, porta sed, malesuada a, lacus. Phasellus mollis elementum enim. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam vitae urna vel velit volutpat tempor. Quisque ac dolor. Suspendisse potenti. Nunc nec tortor. Curabitur pharetra pellentesque diam.

1 of 1

Tal como se muestra en la imagen anterior, el contenido dinámico puede incluir muchos componentes, como párrafos, listas, imágenes, etc. Estos componentes corresponden a elementos de marcado y objetos en el código de procedimiento. Estas clases se repasarán con detalle más adelante en Flow sección Clases relacionadas de esta información general. Por ahora, este es un ejemplo de código simple


que crea un documento de flujo que consta de un párrafo con texto en negrita y una lista.

```
<!-- This simple flow document includes a paragraph with some
      bold text in it and a list. -->
<FlowDocumentReader xmlns="http://schemas.microsoft.com/winfx/2006/xaml/preser
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <FlowDocument>
    <Paragraph>
      <Bold>Some bold text in the paragraph.</Bold>
      Some text that is not bold.
    </Paragraph>

    <List>
      <ListItem>
        <Paragraph>ListItem 1</Paragraph>
      </ListItem>
      <ListItem>
        <Paragraph>ListItem 2</Paragraph>
      </ListItem>
      <ListItem>
        <Paragraph>ListItem 3</Paragraph>
      </ListItem>
    </List>

  </FlowDocument>
</FlowDocumentReader>
```

C#

 Copiar

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;

namespace SDKSample
{
    public partial class SimpleFlowExample : Page
    {
        public SimpleFlowExample()
        {

            Paragraph myParagraph = new Paragraph();

            // Add some Bold text to the paragraph
            myParagraph.Inlines.Add(new Bold(new Run("Some bold text in the paragraph.")));

            // Add some plain text to the paragraph
            myParagraph.Inlines.Add(new Run(" Some text that is not bold."));

            // Create a List and populate with three list items.
            List myList = new List();

            // First create paragraphs to go into the list item.
            Paragraph paragraphListItem1 = new Paragraph(new Run("ListItem 1"));
            Paragraph paragraphListItem2 = new Paragraph(new Run("ListItem 2"));
            Paragraph paragraphListItem3 = new Paragraph(new Run("ListItem 3"));
```

```

// Add ListItems with paragraphs in them.
myList.ListItems.Add(new ListItem(paragraphListItem1));
myList.ListItems.Add(new ListItem(paragraphListItem2));
myList.ListItems.Add(new ListItem(paragraphListItem3));

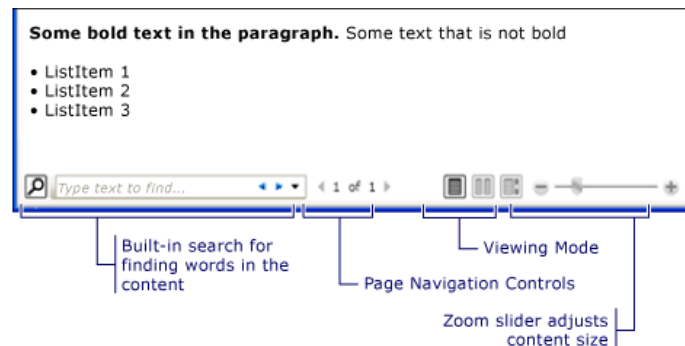
// Create a FlowDocument with the paragraph and list.
FlowDocument myFlowDocument = new FlowDocument();
myFlowDocument.Blocks.Add(myParagraph);
myFlowDocument.Blocks.Add(myList);

// Add the FlowDocument to a FlowDocumentReader Control
FlowDocumentReader myFlowDocumentReader = new FlowDocumentReader()
myFlowDocumentReader.Document = myFlowDocument;

this.Content = myFlowDocumentReader;
}
}
}

```

En la ilustración siguiente se muestra el aspecto de este fragmento de código.



En este ejemplo, el FlowDocumentReader control se usa para hospedar el contenido del flujo. Consulte Flow tipos de documento para obtener más información sobre los controles de hospedaje de contenido dinámico. Paragraph Los Lstelementos , ListItem, y Bold se usan para controlar el formato de contenido, en función de su orden en el marcado. Por ejemplo, el Bold elemento abarca solo una parte del texto del párrafo; como resultado, solo esa parte del texto está en negrita. Si ha utilizado HTML, esto le resultará familiar.

Como se resalta en la ilustración anterior, hay varias características integradas en Flow Documents:

- **Buscar:** permite al usuario realizar una búsqueda de texto completo en todo un documento.
- **Modo de visualización:** el usuario puede seleccionar su modo de visualización preferido, incluido el modo de visualización de una sola página (una página a la vez), el modo de visualización de dos páginas a la vez (formato de lectura de libro) y un modo de desplazamiento continuo (sin límite).
- **Controles de navegación de página:** si el modo de visualización del documento utiliza páginas, los controles de navegación de páginas incluyen un botón para ir a la página siguiente (flecha abajo) o la página anterior (flecha arriba), así como indicadores para el número de página actual y el número total de páginas. Las páginas también pueden voltearse mediante las teclas de flecha del teclado.
- **Zoom:** los controles de zoom permiten al usuario aumentar o disminuir el nivel de zoom haciendo clic en los botones de más o menos, respectivamente. Los controles de zoom incluyen también un control deslizante para ajustar el nivel de zoom. Para obtener más información.

Estas características pueden modificarse según el control utilizado para hospedar el contenido dinámico. En la siguiente sección se describen los distintos controles.

Tipos de documentos dinámicos

La visualización del contenido de los documentos dinámicos y cómo aparece depende de qué objeto se utilice para hospedar el contenido dinámico. Hay cuatro controles que admiten la visualización de contenido dinámico: `FlowDocumentReader`, `FlowDocumentPageViewer`, `RichTextBox` y `FlowDocumentScrollView`. Estos controles se describen brevemente a continuación.

ⓘ Nota

FlowDocument es necesario para hospedar directamente el contenido dinámico, por lo que todos estos controles de visualización consumen un **FlowDocument** habilitar el hospedaje de contenido dinámico.

FlowDocumentPageViewer y FlowDocumentScrollViewer

FlowDocumentPageViewer muestra el contenido en modo de visualización de página en página, mientras que **FlowDocumentScrollViewer** muestra el contenido en modo de desplazamiento continuo. Tanto **FlowDocumentPageViewer** como **FlowDocumentScrollViewer** se fijan en un modo de visualización determinado. Compare con **FlowDocumentReader**, que incluye características que permiten al usuario elegir dinámicamente entre los distintos modos de visualización (**FlowDocumentReaderViewingMode** tal y como se proporciona en la enumeración), a costa de que se consumen más recursos que **FlowDocumentPageViewer** o **FlowDocumentScrollViewer**.

De manera predeterminada, siempre se muestra una barra de desplazamiento vertical y una barra de desplazamiento horizontal se vuelve visible cuando es necesario. La interfaz de usuario predeterminada **FlowDocumentScrollViewer** de no incluye una barra de herramientas; sin embargo, **IsToolBarVisible** la propiedad se puede usar para habilitar una barra de herramientas integrada.

RichTextBox

Se usa cuando **RichTextBox** se quiere permitir que el usuario edite el contenido dinámico. Por ejemplo, si quisiera crear un editor que permitiese a un usuario manipular elementos como tablas, formato en cursiva y negrita, etc., usaría **RichTextBox**. Vea Información general de **RichTextBox** para obtener más información.

Nota

Flow contenido dentro de un **RichTextBox** no se comporta exactamente igual que el contenido dinámico contenido en otros controles. Por ejemplo, no hay columnas en **RichTextBox** , por lo

tanto, no hay ningún comportamiento de cambio de tamaño automático. Además, las características integradas normalmente de contenido dinámico como la búsqueda, el modo de visualización, la navegación por páginas y el zoom no están disponibles dentro de `.RichTextBox`

Crear contenido dinámico

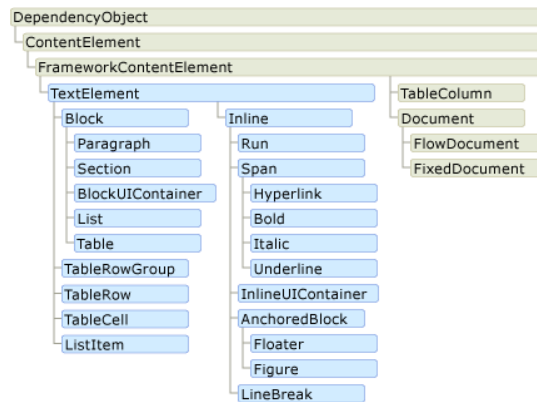
Flow contenido puede ser complejo, que consta de varios elementos, como texto, imágenes, `UIElement` tablas e incluso clases derivadas como controles. Para entender cómo crear contenido dinámico complejo, los puntos siguientes son fundamentales:

- Clases relacionadas con el flujo: cada clase que se utiliza en el contenido dinámico tiene un propósito específico. Además, la relación jerárquica entre las clases dinámicas le ayuda a comprender cómo se utilizan. Por ejemplo, las clases derivadas de la `Block` clase se usan para contener otros objetos, mientras que las clases derivadas de `Inline` contienen objetos que se muestran.
- Esquema de contenido: un documento dinámico puede requerir un número importante de elementos anidados. El esquema de contenido especifica las relaciones posibles entre elementos primarios y secundarios.

En las secciones siguientes se repasará cada una de estas áreas con más detalle.

Clases relacionadas con el flujo

En el diagrama siguiente se muestran los objetos más utilizados con el contenido dinámico:



A los efectos del contenido dinámico, hay dos categorías importantes

1. Clases derivadas de Block: también denominadas "elementos de contenido de Block" o simplemente "elementos Block". Los elementos que heredan de Block se pueden usar para agrupar elementos en un elemento primario común o para aplicar atributos comunes a un grupo.
2. Clases derivadas de Inline: también denominadas "elementos de contenido de Inline" o simplemente "elementos de Inline". Los elementos que heredan Inline de se encuentran dentro de un elemento Block u otro elemento inline. Los elementos Inline se utilizan a menudo como contenedor directo del contenido que se representa en la pantalla. Por ejemplo, un Paragraph (elemento Block) Run puede contener un (elemento Inline), Run pero contiene realmente el texto que se representa en la pantalla.

Cada clase de estas dos categorías se describe brevemente a continuación.

Clases derivadas de Block

Paragraph

Paragraph normalmente se usa para agrupar contenido en un párrafo. El uso más sencillo y común de Paragraph es crear un párrafo de texto.

```
XAML Copiar
<FlowDocument xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <Paragraph>
    Some paragraph text.
  </Paragraph>
</FlowDocument>
```

```
C# Copiar
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;

namespace SDKSample
{
    public partial class ParagraphExample : Page
    {
        public ParagraphExample()
        {
            // Create paragraph with some text.
            Paragraph myParagraph = new Paragraph();
            myParagraph.Inlines.Add(new Run("Some paragraph text.));

            // Create a FlowDocument and add the paragraph to it.
            FlowDocument myFlowDocument = new FlowDocument();
            myFlowDocument.Blocks.Add(myParagraph);

            this.Content = myFlowDocument;
        }
    }
}
```

Sin embargo, también puede contener otros elementos derivados en línea, como verá a continuación.

Sección

Section solo se usa para contener otros Block elementos derivados de . No aplica ningún formato predeterminado a los elementos que contiene. Sin embargo, los valores de propiedad establecidos en se aplican a Section sus elementos secundarios. Una sección permite iterar mediante programación en su colección secundaria. Section se usa de forma similar a la etiqueta <DIV> en HTML.

En el ejemplo siguiente, se definen tres párrafos en un .Section La sección tiene un valor Background de propiedad de Rojo, por lo que el color de fondo de los párrafos también es rojo.

XAML Copiar

```

<FlowDocument xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <!-- By default, Section applies no formatting to elements contained
        within it. However, in this example, the section has a Background
        property value of "Red", therefore, the three paragraphs (the block)
        inside the section also have a red background. -->
  <Section Background="Red">
    <Paragraph>
      Paragraph 1
    </Paragraph>
    <Paragraph>
      Paragraph 2
    </Paragraph>
    <Paragraph>
      Paragraph 3
    </Paragraph>
  </Section>
</FlowDocument>

```

C# Copiar

```

using System;
using System.Windows;
using System.Windows.Media;
using System.Windows.Controls;
using System.Windows.Documents;

namespace SDKSample
{
    public partial class SectionExample : Page
    {
        public SectionExample()
        {
            // Create three paragraphs
            Paragraph myParagraph1 = new Paragraph(new Run("Paragraph 1"));
            Paragraph myParagraph2 = new Paragraph(new Run("Paragraph 2"));
            Paragraph myParagraph3 = new Paragraph(new Run("Paragraph 3"));

            // Create a Section and add the three paragraphs to it.
            Section mySection = new Section();
            mySection.Background = Brushes.Red;

            mySection.Blocks.Add(myParagraph1);
            mySection.Blocks.Add(myParagraph2);
            mySection.Blocks.Add(myParagraph3);

            // Create a FlowDocument and add the section to it.
            FlowDocument myFlowDocument = new FlowDocument();
            myFlowDocument.Blocks.Add(mySection);

            this.Content = myFlowDocument;
        }
    }
}

```

BlockUIContainer

BlockUIContainer permite UIElement que los elementos (es decir,) Button se inserten en contenido de flujo derivado de bloques.

InlineUIContainer (consulte a continuación) se usa para insertar elementos UIElement en contenido de flujo derivado en línea.

BlockUIContainer y InlineUIContainer son importantes porque no hay otra manera de usar un en el contenido dinámico a UIElement menos que esté contenido dentro de uno de estos dos elementos.

En el ejemplo siguiente se muestra cómo usar el elemento BlockUIContainer para hospedar objetos UIElement dentro del contenido dinámico.

```
XAML Copiar
<FlowDocument ColumnWidth="400">
  <Section Background="GhostWhite">
    <Paragraph>
      A UIElement element may be embedded directly in flow content
      by enclosing it in a BlockUIContainer element.
    </Paragraph>
    <BlockUIContainer>
      <Button>Click me!</Button>
    </BlockUIContainer>
    <Paragraph>
      The BlockUIContainer element may host no more than one top-level
      UIElement. However, other UIElements may be nested within the
      UIElement contained by an BlockUIContainer element. For example,
      a StackPanel can be used to host multiple UIElement elements within
      a BlockUIContainer element.
    </Paragraph>
    <BlockUIContainer>
      <StackPanel>
        <Label Foreground="Blue">Choose a value:</Label>
        <ComboBox>
          <ComboBoxItem IsSelected="True">a</ComboBoxItem>
          <ComboBoxItem>b</ComboBoxItem>
          <ComboBoxItem>c</ComboBoxItem>
        </ComboBox>
        <Label Foreground="Red">Choose a value:</Label>
        <StackPanel>
          <RadioButton>x</RadioButton>
          <RadioButton>y</RadioButton>
          <RadioButton>z</RadioButton>
        </StackPanel>
      </StackPanel>
    </BlockUIContainer>
  </Section>
</FlowDocument>
```

```

</RadioButton>/></RadioButton>/>
</StackPanel>
<Label>Enter a value:</Label>
<TextBox>
    A text editor embedded in flow content.
</TextBox>
</StackPanel>
</BlockUIContainer>
</Section>
</FlowDocument>

```

En la ilustración siguiente se muestra cómo se representa este ejemplo:

A UIElement element may be embedded directly in flow content by enclosing it in a BlockUIContainer element.

Click me!

The BlockUIContainer element may host no more than one top-level UIElement. However, other UIElements may be nested within the UIElement contained by an BlockUIContainer element. For example, a StackPanel can be used to host multiple UIElement elements within a BlockUIContainer element.

Choose a value:

a

Choose a value:

☐ x
☐ y
☐ z

Enter a value:

A text editor embedded in flow content.

Lista

List se usa para crear una lista numérica o con viñetas. Establezca la MarkerStyle propiedad en un valor TextMarkerStyle de enumeración para determinar el estilo de la lista. En el ejemplo siguiente se muestra cómo crear una lista sencilla.

```
XAML Copiar

<FlowDocument xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <List>
    <ListItem>
      <Paragraph>
        List Item 1
      </Paragraph>
    </ListItem>
    <ListItem>
      <Paragraph>
        List Item 2
      </Paragraph>
    </ListItem>
    <ListItem>
      <Paragraph>
        List Item 3
      </Paragraph>
    </ListItem>
  </List>
</FlowDocument>
```

```
C# Copiar

using System;
using System.Windows;
using System.Windows.Media;
using System.Windows.Controls;
using System.Windows.Documents;

namespace SDKSample
{
    public partial class ListExample : Page
    {
        public ListExample()
        {
            // Create three paragraphs
            Paragraph myParagraph1 = new Paragraph(new Run("List Item 1"));
            Paragraph myParagraph2 = new Paragraph(new Run("List Item 2"));
            Paragraph myParagraph3 = new Paragraph(new Run("List Item 3"));

            // Create the ListItem elements for the List and add the
            // paragraphs to them.
            ListItem myListItem1 = new ListItem();
            myListItem1.Blocks.Add(myParagraph1);
            ListItem myListItem2 = new ListItem();
            myListItem2.Blocks.Add(myParagraph2);
            ListItem myListItem3 = new ListItem();
            myListItem3.Blocks.Add(myParagraph3);

            // Create a List and add the three ListItems to it.
            List myList = new List();
```

```

        myList.ListItems.Add(myListItem1);
        myList.ListItems.Add(myListItem2);
        myList.ListItems.Add(myListItem3);

        // Create a FlowDocument and add the section to it.
        FlowDocument myFlowDocument = new FlowDocument();
        myFlowDocument.Blocks.Add(myList);

        this.Content = myFlowDocument;
    }
}

```

Nota

List es el único elemento de flujo que usa para ListItemCollection administrar elementos secundarios.

Tabla

Table se usa para crear una tabla. Table es similar al elemento Grid , pero tiene más funcionalidades y, por lo tanto, requiere una mayor sobrecarga de recursos. Dado Grid que es UIElementun , no se puede usar en el contenido dinámico a menos que esté contenido en o BlockUIContainerInlineUIContainer. Para obtener más información sobre Table, vea Información general sobre tablas.

Clases derivadas de Inline

Run

Run se usa para contener texto sin formato. Es posible que los Run objetos se utilicen ampliamente en el contenido dinámico. Sin embargo, en el marcado, Run no es necesario que los elementos se utilicen explícitamente. Run debe usarse al crear o manipular documentos de flujo mediante código. Por ejemplo, en el marcado siguiente, el primero Paragraph especifica el Run elemento explícitamente mientras que el segundo no. Ambos párrafos generan el mismo resultado.

```

XAML Copiar

<Paragraph>
  <Run>Paragraph that explicitly uses the Run element.</Run>
</Paragraph>

<Paragraph>
  This Paragraph omits the Run element in markup. It renders
  the same as a Paragraph with Run used explicitly.
</Paragraph>

```

Nota

A partir de .NET Framework 4, la Text propiedad del objeto Run es una propiedad de dependencia. Puede enlazar la propiedad Text a un origen de datos, como .TextBlock La Text propiedad es totalmente compatible con el enlace un solo sentido. La Text propiedad también admite el enlace de dos vías, excepto para RichTextBox. Para obtener un ejemplo, consulte Run.Text.

Intervalo

Span agrupa otros elementos de contenido en línea. No se aplica ninguna representación inherente al contenido dentro de un Span elemento. Sin embargo, los elementos que heredan Span de incluir Hyperlink, Boldy ItalicUnderline aplican formato al texto.

A continuación se muestra un ejemplo Span de que se usa para contener contenido en línea, incluido texto, Bold un elemento y un Button.

```

XAML Copiar

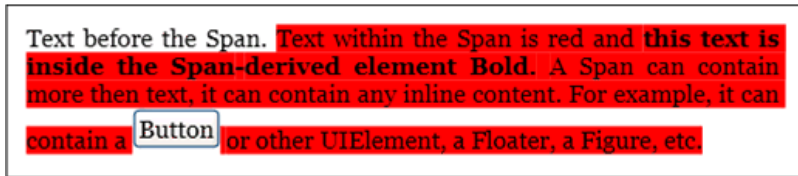
<FlowDocument xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

  <Paragraph>
    Text before the Span. <Span Background="Red">Text within the Span is
    red and <Bold>this text is inside the Span-derived element Bold.</Bold>
    A Span can contain more then text, it can contain any inline content. For
    example, it can contain a
    <InlineUIContainer>
      <Button>Button</Button>
    </InlineUIContainer>
    or other UIElement, a Floater, a Figure, etc.</Span>
  </Paragraph>

</FlowDocument>

```


En la captura de pantalla siguiente se muestra cómo se representa este ejemplo.



InlineUIContainer

InlineUIContainer permite UIElement que los elementos (es decir, un control como Button) se inserte en un elemento Inline de contenido. Este elemento es el equivalente en línea al descrito BlockUIContainer anteriormente. A continuación se muestra un ejemplo que InlineUIContainer usa para insertar Button un elemento en línea en .Paragraph

```
XAML Copiar
<FlowDocument xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

  <Paragraph>
    Text to precede the button...

    <!-- Set the BaselineAlignment property to "Bottom"
         so that the Button aligns properly with the text. -->
    <InlineUIContainer BaselineAlignment="Bottom">
      <Button>Button</Button>
    </InlineUIContainer>
    Text to follow the button...
  </Paragraph>

</FlowDocument>
```

```

C# Copiar

using System;
using System.Windows;
using System.Windows.Media;
using System.Windows.Controls;
using System.Windows.Documents;

namespace SDKSample
{
    public partial class InlineUIContainerExample : Page
    {
        public InlineUIContainerExample()
        {
            Run run1 = new Run(" Text to precede the button... ");
            Run run2 = new Run(" Text to follow the button... ");

            // Create a new button to be hosted in the paragraph.
            Button myButton = new Button();
            myButton.Content = "Click me!";

            // Create a new InlineUIContainer to contain the Button.
            InlineUIContainer myInlineUIContainer = new InlineUIContainer();

            // Set the BaselineAlignment property to "Bottom" so that the
            // Button aligns properly with the text.
            myInlineUIContainer.BaselineAlignment = BaselineAlignment.Bottom;

            // Assign the button as the UI container's child.
            myInlineUIContainer.Child = myButton;

            // Create the paragraph and add content to it.
            Paragraph myParagraph = new Paragraph();
            myParagraph.Inlines.Add(run1);
            myParagraph.Inlines.Add(myInlineUIContainer);
            myParagraph.Inlines.Add(run2);

            // Create a FlowDocument and add the paragraph to it.
            FlowDocument myFlowDocument = new FlowDocument();
            myFlowDocument.Blocks.Add(myParagraph);

            this.Content = myFlowDocument;
        }
    }
}

```

Nota

InlineUIContainer no necesita usarse explícitamente en el marcado. Si lo omite, se creará InlineUIContainer un de todos modos cuando se compile el código.

Figure y Floater Figurey Floater se usan para insertar contenido en documentos Flow con propiedades de selección de ubicación que se

pueden personalizar independientemente del flujo de contenido principal. Figure Los Floater elementos o se usan a menudo para resaltar o resaltar partes de contenido, hospedar imágenes de soporte u otro contenido dentro del flujo de contenido principal, o para insertar contenido relacionado de forma flexible, como anuncios.


En el ejemplo siguiente se muestra cómo insertar un en Figure un párrafo de texto.

```
XAML Copiar

<FlowDocument xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

  <Paragraph>
    <Figure
      Width="300" Height="100"
      Background="GhostWhite" HorizontalAnchor="PageLeft" >
      <Paragraph FontStyle="Italic" Background="Beige" Foreground="DarkGreen" >
        A Figure embeds content into flow content with placement properties
        that can be customized independently from the primary content flow
      </Paragraph>
    </Figure>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy
    nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wis
    enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobort
    nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure.
  </Paragraph>
</FlowDocument>
```

C#

 Copiar

```
using System;
using System.Windows;
using System.Windows.Media;
using System.Windows.Controls;
using System.Windows.Documents;

namespace SDKSample
{
    public partial class FigureExample : Page
    {
        public FigureExample()
        {
            // Create strings to use as content.
            string strFigure = "A Figure embeds content into flow content with  

                                " placement properties that can be customized"  

                                " independently from the primary content flow";
            string strOther = "Lorem ipsum dolor sit amet, consectetur adipis  

                                " elit, sed diam nonummy nibh euismod tincidunt  

                                " dolore magna aliquam erat volutpat. Ut wisi er  

                                " minim veniam, quis nostrud exerci tation ullam  

                                " suscipit lobortis nisl ut aliquip ex ea commoc  

                                " Duis autem vel eum iriure.";

            // Create a Figure and assign content and layout properties to it.
            Figure myFigure = new Figure();
            myFigure.Width = new FigureLength(300);
            myFigure.Height = new FigureLength(100);
```

```

myFigure.Background = Brushes.GhostWhite;
myFigure.HorizontalAnchor = FigureHorizontalAnchor.PageLeft;
Paragraph myFigureParagraph = new Paragraph(new Run(strFigure));
myFigureParagraph.FontStyle = FontStyles.Italic;
myFigureParagraph.Background = Brushes.Beige;
myFigureParagraph.Foreground = Brushes.DarkGreen;
myFigure.Blocks.Add(myFigureParagraph);

// Create the paragraph and add content to it.
Paragraph myParagraph = new Paragraph();
myParagraph.Inlines.Add(myFigure);
myParagraph.Inlines.Add(new Run(strOther));

// Create a FlowDocument and add the paragraph to it.
FlowDocument myFlowDocument = new FlowDocument();
myFlowDocument.Blocks.Add(myParagraph);

this.Content = myFlowDocument;
}
}
}

```

En la ilustración siguiente se muestra cómo se representa este ejemplo.

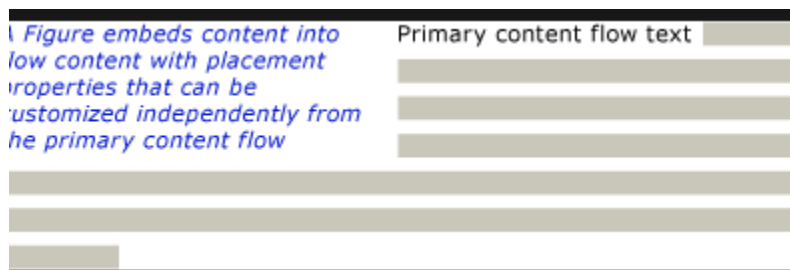


Figure y Floater difieren de varias maneras y se usan para distintos escenarios.

Figura:

- Se pueden determinar su posición: puede establecer sus delimitadores horizontal y vertical para acoplarlo con respecto a la página, el contenido, la columna o el párrafo. También puede usar sus propiedades HorizontalOffset y VerticalOffset para especificar desplazamientos arbitrarios.

- Se puede ajustar a más de una columna: Figure puede establecer el alto y el ancho en múltiplo de alto o ancho de página, contenido o columna. Tenga en cuenta que, en el caso de la página y el contenido, no se permiten múltiplos mayores que 1. Por ejemplo, puede Figure establecer el ancho de un para que sea "página 0,5", "contenido de 0,25" o "2 columna". También puede establecer el alto y ancho en valores absolutos de píxeles.
- No pagina: si el FigureFigurecontenido dentro de un no cabe dentro de , representará el contenido que quepa y se perderá el contenido restante.

Floater:

- No se puede establecer su posición y se representará en cualquier espacio que pueda estar a su disposición. No se puede establecer el desplazamiento ni delimitar un Floater.
- No se puede cambiar el tamaño a más de una columna: de forma predeterminada, Floater tamaños en una columna. Tiene una Width propiedad que se puede establecer en un valor de píxel absoluto, pero si este valor es mayor que un ancho de columna, se omite y el valor floater se establece en una columna. Puede ajustar su tamaño a menos de una columna estableciendo el ancho de píxel correcto, pero el tamaño no es relativo a Floater la columna, por lo que "0,5Column" no es una expresión válida para width. Floater no tiene ninguna propiedad height y su altura no se puede establecer, su altura depende del contenido.
- Floater paginaciones: si su contenido con el ancho especificado se extiende a más de 1 altura de columna, floater se interrumpe y pagina a la columna siguiente, la página siguiente, etc.

Figure es un buen lugar para colocar contenido independiente donde desea controlar el tamaño y la posición, y está seguro de que el contenido se ajustará al tamaño especificado. Floater es un buen lugar para colocar más contenido de flujo libre que fluya de forma similar al contenido de la página principal, pero que esté separado de él.

LineBreak

LineBreak hace que se produzca un salto de línea en el contenido dinámico. El siguiente ejemplo muestra el uso de LineBreak.

```
XAML Copiar
<FlowDocument xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <Paragraph>
    Before the LineBreak in Paragraph.
    <LineBreak />
    After the LineBreak in Paragraph.
    <LineBreak/><LineBreak/>
    After two LineBreaks in Paragraph.
  </Paragraph>

  <Paragraph>
    <LineBreak/>
  </Paragraph>

  <Paragraph>
    After a Paragraph with only a LineBreak in it.
  </Paragraph>
</FlowDocument>
```

En la captura de pantalla siguiente se muestra cómo se representa este ejemplo.

Before the LineBreak in Paragraph.
After the LineBreak in Paragraph.

After two LineBreaks in Paragraph.

After a Paragraph with only a LineBreak in it.

Elementos de colección dinámica

En muchos de los ejemplos anteriores, y `BlockCollection` `InlineCollection` se usan para construir contenido dinámico mediante programación. Por ejemplo, para agregar elementos a `Paragraph`, puede usar la sintaxis :

C#

 Copiar

```
myParagraph.Inlines.Add(new Run("Some text"));
```

Esto agrega un al Run de InlineCollectionParagraph. Esto es lo mismo que el implícito que se Run encuentra dentro de en Paragraph el marcado:

XML

 Copiar

```
<Paragraph>  
Some Text  
</Paragraph>
```

Como ejemplo de uso de , en BlockCollection el ejemplo siguiente se crea un nuevo y, a Section continuación, se usa el método Add para agregar un nuevo Paragraph al Section contenido.

C#

 Copiar

```
Section secx = new Section();  
secx.Blocks.Add(new Paragraph(new Run("A bit of text content...")));
```

Además de agregar elementos a una colección dinámica, también puede quitar elementos. En el ejemplo siguiente se elimina el último Inline elemento de Span.

C#

 Copiar

```
spanx.Inlines.Remove(spanx.Inlines.LastInline);
```

En el ejemplo siguiente se borra todo el contenido (Inline elementos) de Span.

C#

 Copiar

```
spanx.Inlines.Clear();
```

Cuando trabaje con contenido dinámico mediante programación, probablemente utilizará mucho estas colecciones.

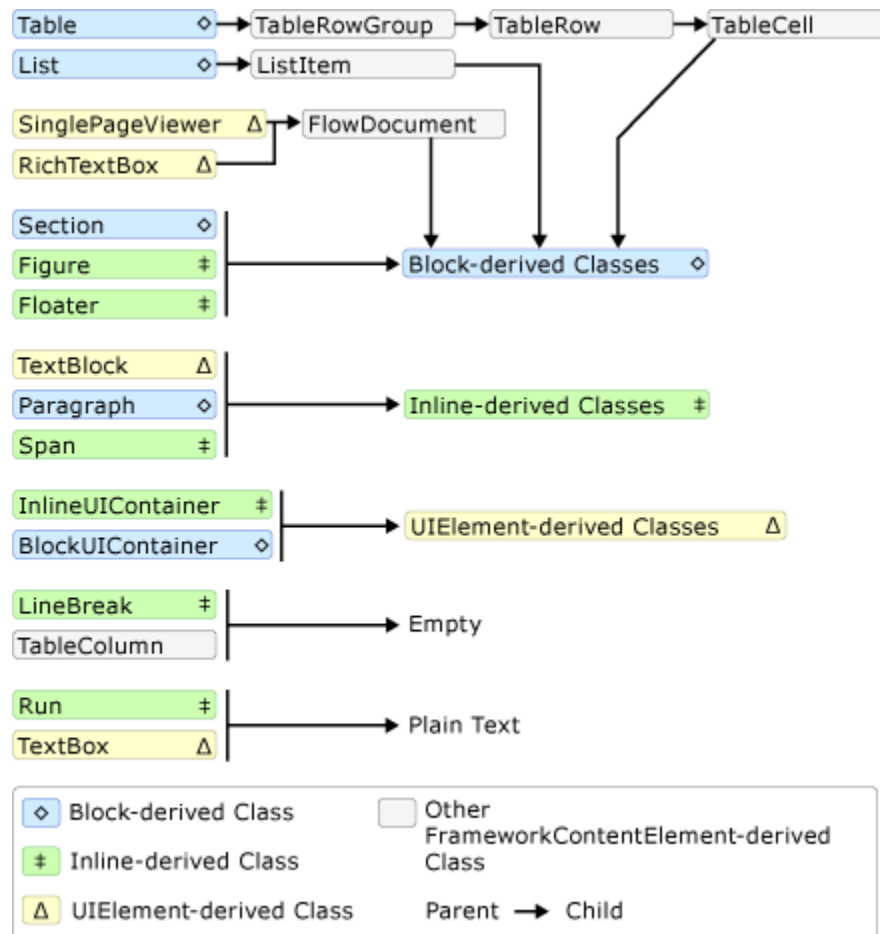
Si un elemento de flujo InlineCollection usa (Inserts) BlockCollection o (Blocks) para contener sus elementos secundarios depende del tipo de elementos secundarios (BlockInlineo) que puede contener el elemento primario. Las reglas de contención para los elementos de contenido dinámico se resumen en el esquema de contenido en la sección siguiente.

Nota

Hay un tercer tipo de colección que se usa con contenido dinámico, ListItemCollection, pero esta colección solo se usa con .List Además, hay varias colecciones que se usan con Table. Consulte Información general sobre tablas para obtener más información.

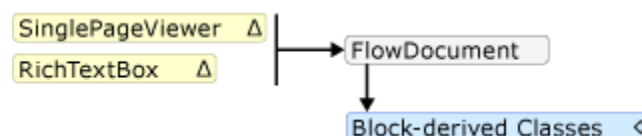
Esquema de contenido

Dado el número de los distintos elementos de contenido dinámico, puede resultar abrumador mantener un seguimiento de qué tipo de elementos secundarios puede contener un elemento. En el diagrama siguiente se resumen las reglas de contención para los elementos dinámicos. Las flechas representan las relaciones posibles entre elementos primarios y secundarios.



Como se puede ver en el diagrama anterior, los elementos secundarios permitidos para un elemento no están necesariamente determinados por si es Block un elemento o un Inline elemento. Por ejemplo, un Span (un elemento Inline) solo Inline puede tener elementos secundarios, Figure mientras que un (Inline también un elemento) solo puede tener Block elementos secundarios. Por tanto, un diagrama es útil para determinar rápidamente qué elemento puede incluirse en otro. Por ejemplo, vamos a usar el diagrama para determinar cómo construir el contenido dinámico de .RichTextBox

1. Un debe RichTextBox contener un que, a FlowDocument su vez, debe contener un Blockobjeto derivado de . A continuación, se muestra el segmento correspondiente del diagrama anterior.



Llegados a este punto, esta es la apariencia que podría tener el marcado.

```
XAML Copiar

<RichTextBox>
  <FlowDocument>
    <!-- One or more Block-derived object... -->
  </FlowDocument>
</RichTextBox>
```

2. Según el diagrama, BlockParagraph hay varios elementos entre los que elegir, incluidos , Section, TableList, y BlockUIContainer (vea Clases derivadas de bloques anteriores). Supongamos que queremos un .Table Según el diagrama anterior, contiene TableRowGroup un elemento que TableRowTableCell contiene elementos que contienen un Blockobjeto derivado de . A continuación se muestra el segmento correspondiente Table para tomado del diagrama anterior.

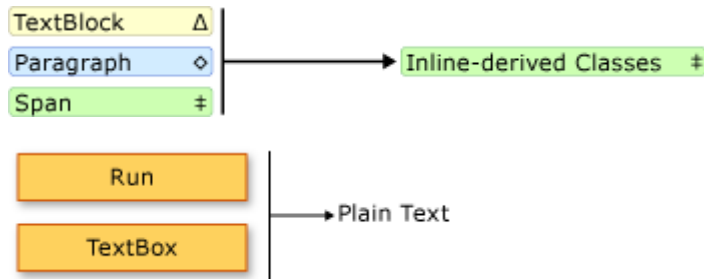


A continuación, se muestra el marcado correspondiente.

```
XAML Copiar

<RichTextBox>
  <FlowDocument>
    <Table>
      <TableRowGroup>
        <TableRow>
          <TableCell>
            <!-- One or more Block-derived object... -->
          </TableCell>
        </TableRow>
      </TableRowGroup>
    </Table>
  </FlowDocument>
</RichTextBox>
```

3. De nuevo, se requieren uno o varios Block elementos debajo de .TableCell Para facilitar el proceso, vamos a colocar texto dentro de la celda. Esto se puede hacer mediante con Paragraph un Run elemento . A continuación se muestran los segmentos ParagraphInlineRun correspondientes del diagrama que muestran que un puede tomar un elemento y que (Inline un elemento) solo puede tomar texto sin formato.



A continuación se muestra el ejemplo completo en el marcado.

```
XAML Copiar

<Page xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <RichTextBox>
    <FlowDocument>

      <!-- Normally a table would have multiple rows and multiple
            cells but this code is for demonstration purposes.-->
      <Table>
        <TableRowGroup>
          <TableRow>
            <TableCell>
              <Paragraph>

                <!-- The schema does not actually require
                     explicit use of the Run tag in markup. It
                     is only included here for clarity. -->
                <Run>Paragraph in a Table Cell.</Run>
              </Paragraph>
            </TableCell>
          </TableRow>
        </TableRowGroup>
      </Table>

    </FlowDocument>
  </RichTextBox>
</Page>
```

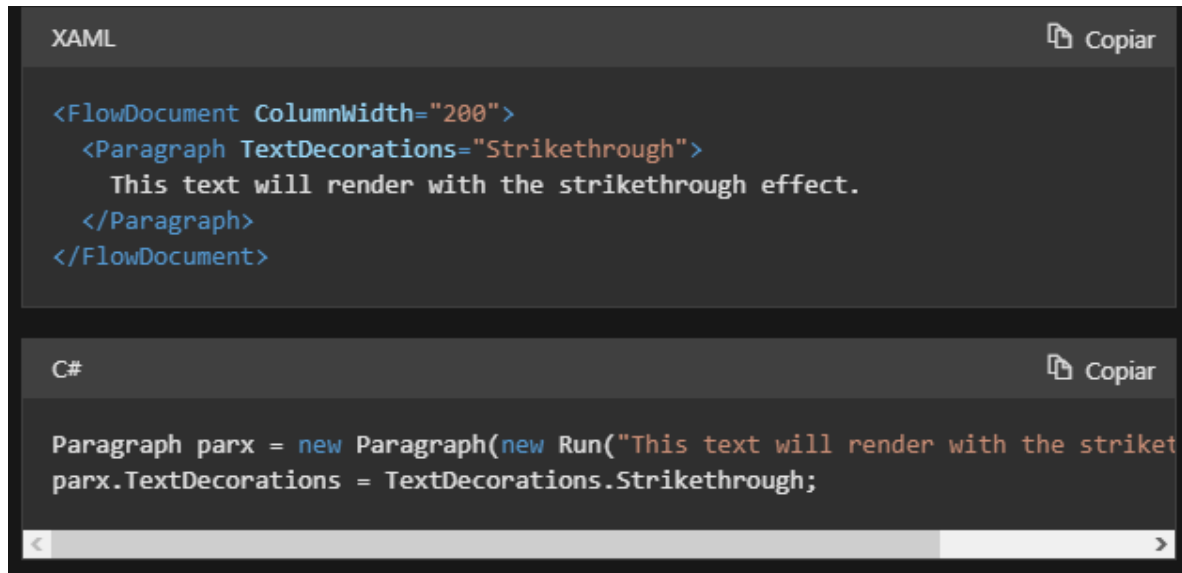
Personalizar texto

En general, el texto es el tipo de contenido más frecuente de un documento dinámico. Aunque los objetos descritos anteriormente pueden utilizarse para controlar la mayoría de los aspectos de cómo se representa el texto, hay otros métodos para personalizar el texto, según se describen en esta sección.

Decoraciones de texto

Las decoraciones de texto le permiten aplicar efectos de subrayado, línea alta, línea base y tachado al texto (consulte las imágenes siguientes). Estas decoraciones se agregan mediante la propiedad `TextDecorations` expuesta por varios objetos, incluidos `ParagraphTextBlock`, y `TextBoxInline`.

En el ejemplo siguiente se muestra cómo se establece la propiedad TextDecorations de Paragraph.



```
XAML
<FlowDocument ColumnWidth="200">
  <Paragraph TextDecorations="Strikethrough">
    This text will render with the strikethrough effect.
  </Paragraph>
</FlowDocument>

C#
Paragraph parx = new Paragraph(new Run("This text will render with the striket
parx.TextDecorations = TextDecorations.Strikethrough;
```

En la ilustración siguiente se muestra cómo se representa este ejemplo.

~~This text will render with the default strikethrough effect.~~

En las siguientes cifras se muestra cómo se representan las decoración de línea overline, baseline y underline , respectivamente.

This text will render with the default overline effect.

This text will render with the default baseline effect.

This text will render with the default underline effect.

Tipografía

La Typography mayoría del contenido relacionado con el flujo TextElement expone la propiedad , FlowDocument, TextBlocky TextBox. Esta propiedad se utiliza para controlar características/variaciones tipográficas del texto (es decir, versalitas o mayúsculas, superíndices y subíndices, etc.).

En el ejemplo siguiente se muestra cómo establecer el atributo Typography utilizando Paragraph como elemento de ejemplo.

```
XAML Copiar  
  
<Paragraph  
    TextAlignment="Left"  
    FontSize="18"  
    FontFamily="Palatino Linotype"  
    Typography.NumeralStyle="OldStyle"  
    Typography.Fraction="Stacked"  
    Typography.Variants="Inferior"  
>  
    <Run>  
        This text has some altered typography characteristics. Note  
        that use of an open type font is necessary for most typographic  
        properties to be effective.  
    </Run>  
    <LineBreak/><LineBreak/>  
    <Run>  
        0123456789 10 11 12 13  
    </Run>  
    <LineBreak/><LineBreak/>  
    <Run>  
        1/2 2/3 3/4  
    </Run>  
</Paragraph>
```

En la ilustración siguiente se muestra cómo se representa este ejemplo.

This text has some altered typography characteristics. Note that use
of an open type font is necessary for most typographic properties to
be effective.

0123456789 10 11 12 13

$\frac{1}{2}$ $\frac{2}{3}$ $\frac{3}{4}$

En cambio, en la siguiente ilustración se muestra cómo se representa un ejemplo similar con propiedades tipográficas predeterminadas.

This text has some altered typography characteristics. Note that use of an open type font is necessary for most typographic properties to be effective.

0123456789 10 11 12 13

1/2 2/3 3/4

En el ejemplo siguiente se muestra cómo establecer la propiedad `Typography` mediante programación.

```
C# Copiar

Paragraph par = new Paragraph();

Run runText = new Run(
    "This text has some altered typography characteristics. Note" +
    "that use of an open type font is necessary for most typographic" +
    "properties to be effective.");
Run runNumerals = new Run("0123456789 10 11 12 13");
Run runFractions = new Run("1/2 2/3 3/4");

par.Inlines.Add(runText);
par.Inlines.Add(new LineBreak());
par.Inlines.Add(new LineBreak());
par.Inlines.Add(runNumerals);
par.Inlines.Add(new LineBreak());
par.Inlines.Add(new LineBreak());
par.Inlines.Add(runFractions);

par.TextAlignment = TextAlignment.Left;
par.FontSize = 18;
par.FontFamily = new FontFamily("Palatino Linotype");

par.Typography.NumeralStyle = FontNumeralStyle.OldStyle;
par.Typography.Fraction = FontFraction.Stacked;
par.Typography.Variants = FontVariants.Inferior;
```