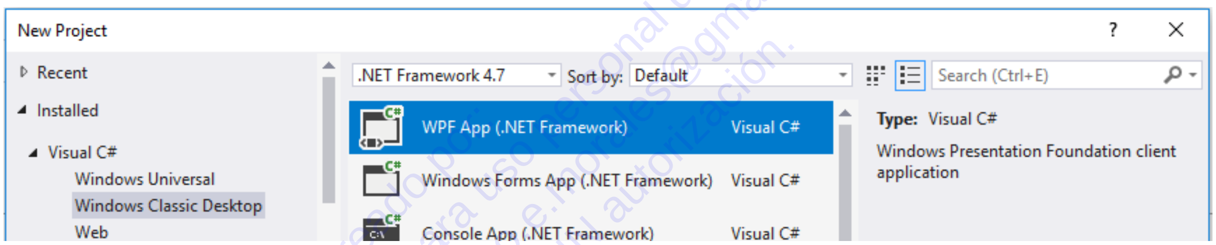
Utilizando el objeto Dispatcher

En el .NET Framework, cada hilo es asociado con un objeto ***Dispatcher***. El *Dispatcher* es responsable de mantener una cola de elementos de trabajo para el hilo. Cuando trabajamos con múltiples hilos, por ejemplo, mediante la ejecución de tareas asíncronas, podemos utilizar el objeto *Dispatcher* para invocar la lógica de un hilo específico. Normalmente necesitamos hacer esto cuando utilizamos operaciones asíncronas en aplicaciones gráficas.

Ejercicio
Utilizando el objeto Dispatcher

Realiza los siguientes pasos para conocer el uso del objeto ***Dispatcher***.

1. Crea una aplicación WPF utilizando la plantilla ***WPF App (.NET Framework)***.



2. Dentro del archivo ***MainWindow.xaml***, remplaza el elemento ***<Grid>*** por el siguiente código.

```
<StackPanel>
    <Label x:Name="lblResult"/>
    <Button x:Name="btnGetResult"
        Content="Obtener resultado" Click="btnGetResult_Click"/>
</StackPanel>
```

   El código anterior simula una aplicación sencilla que consiste de un botón y una etiqueta. Cuando el usuario haga clic en el botón, utilizaremos una tarea para obtener un resultado que será mostrado en la etiqueta.

   Si un usuario hace clic en el botón, el manejador del evento ***Click*** del botón será ejecutado en el hilo de la interfaz de usuario. Si el manejador del evento inicia una tarea asíncrona, esa tarea será ejecutada en un hilo de segundo plano.

3. Agrega el siguiente código dentro de la clase ***MainWindow*** del archivo ***MainWindow.xaml.cs*** para definir un método que muestre un mensaje en la etiqueta ***lblResult***.

```
private void ShowMessage(string message)
{
    lblResult.Content = message;
}
```

4. Agrega el siguiente código dentro de la clase ***MainWindow*** para definir el manejador del evento ***Click*** del botón ***btnGetResult***.

```
private void btnGetResult_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        string Result = "Resultado obtenido";
        ShowMessage(Result);
    });
}
```

En el código anterior, la tarea simplemente asigna un valor a una variable, pero en muchos escenarios, nuestras aplicaciones podrían recuperar datos de un servicio Web o de bases de datos en respuesta a la actividad de la interfaz de usuario.

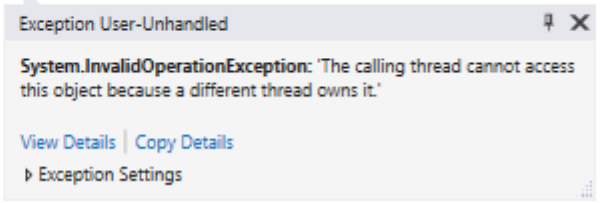
Después de obtener el resultado, el método *ShowMessage* es invocado para mostrar el resultado.

El hilo que ejecuta la lógica de la tarea no tiene acceso a los controles de la interfaz de usuario debido a que estos son propiedad del hilo principal (el hilo de la interfaz de usuario).

5. Ejecuta la aplicación.

6. Haz clic en el botón ***Obtener Resultado***. Puedes notar que se genera una excepción.

```
private void ShowMessage(string message)
{
    lblResult.Content = message;
}
```



Puedes notar que obtenemos una excepción ***InvalidOperationException***. El mensaje de error nos indica que la tarea no puede acceder al objeto ***Label*** ya que este es propiedad de un hilo distinto. Esto es debido a que el método ***ShowMessage*** se está ejecutando en un hilo de segundo plano, pero el objeto ***Label*** fue creado por el hilo de la interfaz de usuario.

Para actualizar el contenido del objeto *Label*, debemos ejecutar el método *ShowMessage* en el hilo de la interfaz de usuario. Para hacer esto, podemos obtener el objeto *Dispatcher* que está asociado con el objeto *Label*.

7. Modifica el código de la tarea para que sea similar al siguiente.

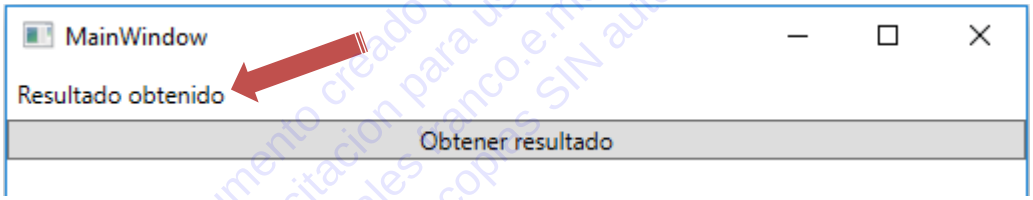
```
private void btnGetResult_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        string Result = "Resultado obtenido";
        lblResult.Dispatcher.BeginInvoke(
            new Action(()=>
        ShowMessage(Result)));
    });
}
```

El código anterior obtiene el objeto *Dispatcher* que está asociado con el objeto *lblResult*. Después de obtener el objeto *Dispatcher*, podemos utilizar el método ***BeginInvoke*** del objeto *Dispatcher* para invocar al método *ShowMessage* en el hilo de la interfaz de usuario.

El método *Dispatcher.BeginInvoke* coloca la lógica que actualiza al objeto *lblResult* en la cola de elementos de trabajo del hilo de la Interfaz de usuario.

8. Ejecuta la aplicación.

9. Haz clic en el botón ***Obtener Resultado***. Puedes notar que la etiqueta *lblResult* muestra el mensaje esperado.



10. Regresa a Visual Studio y detén la ejecución.

    El método *BeginInvoke* no acepta un método anónimo como delegado. Nuestro ejemplo utiliza un delegado *Action* para invocar al método *ShowMessage*, sin embargo, podemos utilizar cualquier delegado que cumpla con la firma del método que deseamos invocar.

11. Agrega el siguiente código dentro de la clase MainWindow para definir un tipo delegado.

```
delegate void ShowDelegate(string message);
```

12. Modifica el código de la tarea para utilizar el delegado definido.

```
private void btnGetResult_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        string Result = "Resultado obtenido";
        lblResult.Dispatcher.BeginInvoke(
            new ShowDelegate(ShowMessage), Result);
    });
}
```

13. Ejecuta la aplicación.

14. Haz clic en el botón ***Obtener Resultado***. Puedes notar que la etiqueta *lblResult* muestra el mensaje esperado.

En este ejercicio, mostramos como utilizar el objeto *Dispatcher* para invocar la lógica de un hilo específico.