CENTRO PARA EL DESARROLLO DE LA INDUSTRIA DEL SOFTWARE | UANL

Xamarin

Desarrollo Móvil con Xamarin

4 – Data binding y ListView

Contenido

- Manejo de archivos
- Data binding
  - Tipos de bindings
- ListView

Manejo de archivos

- Tanto Android como iOS tienen su propio sistema de archivos. La lectura/escritura de archivos se puede realizar utilizando las APIs nativas de cada plataforma.

- Este enfoque tiene sentido cuando las plataformas requieren archivos distintos, por su contenido, formato o tipo.

- Los archivos se almacenan en los proyectos de las plataformas.

- De manera alternativa, los **embedded resources** (recursos incorporados) simplifican esta tarea.

- Estos recursos se almacenan solamente en el proyecto .NET Standard. No es necesario que también se incluyan en los proyectos de las plataformas.

- Este enfoque tiene sentido cuando ambas plataformas requieren exactamente el mismo archivo.

Data binding

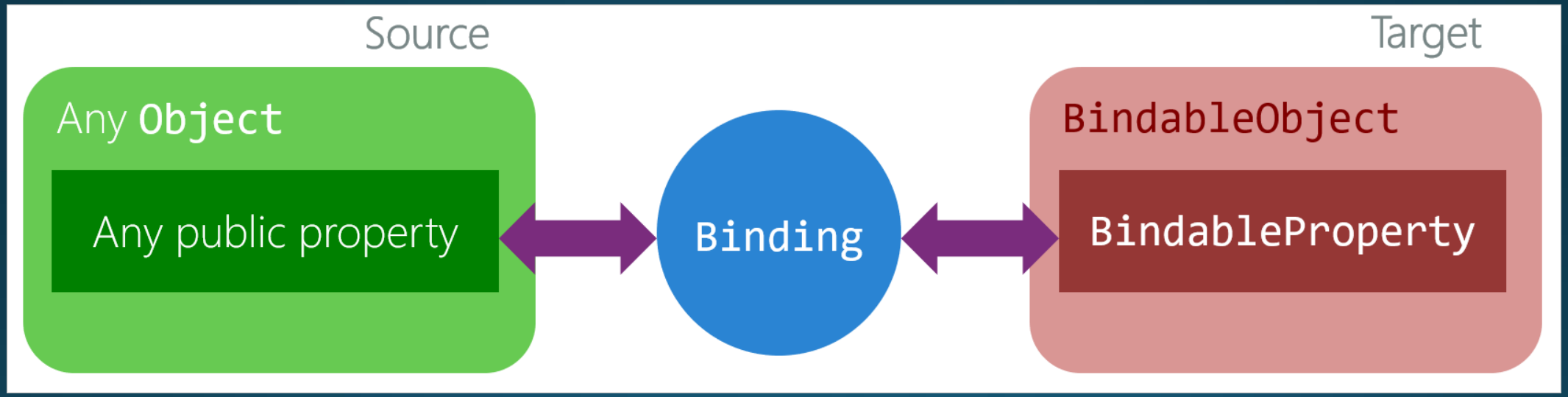
- **Data binding** (unión de datos) es una propiedad de Xamarin.Forms que permite establecer relaciones entre datos y controles de la UI.

- Los datos y controles en una relación de este tipo se "sincronizan", de manera que actualizan sus valores cuando alguno de los dos cambia.

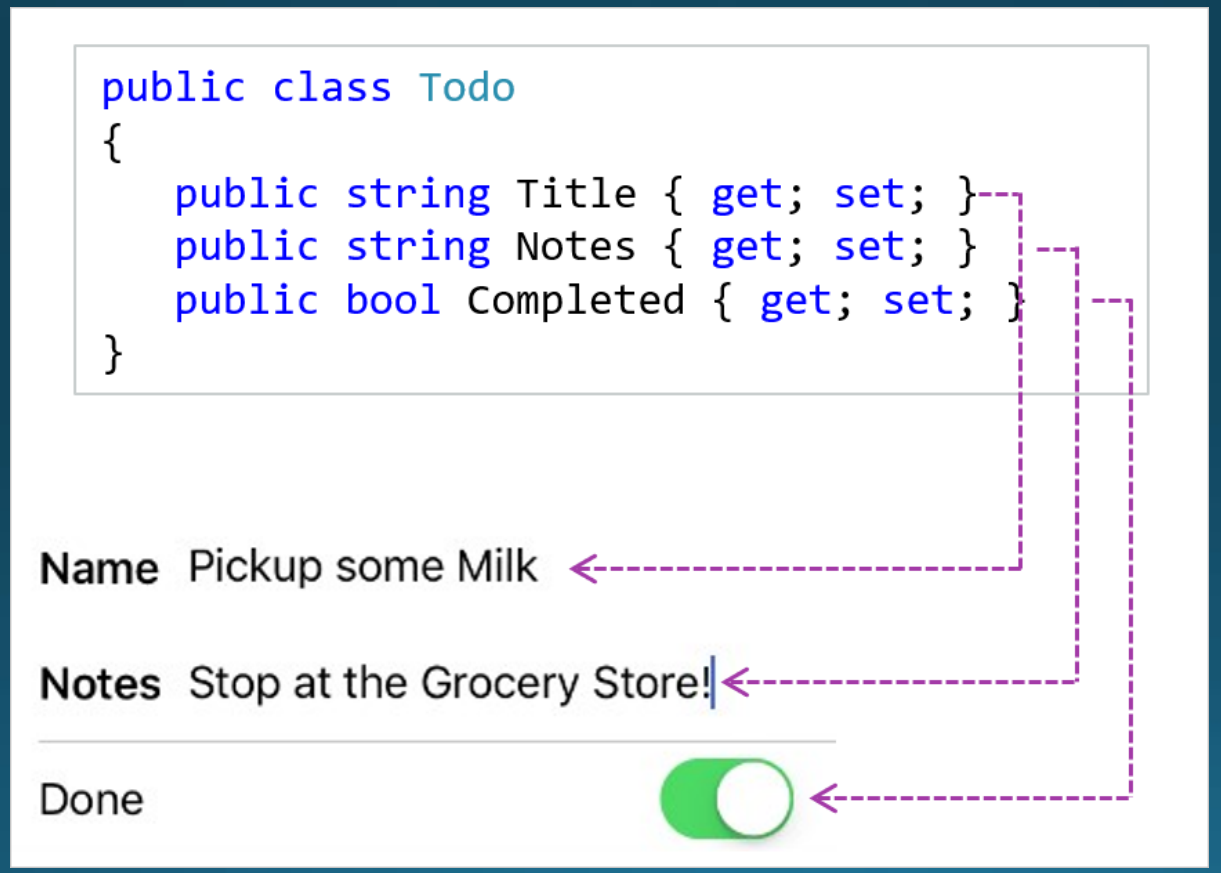
- Los objetos involucrados en esta relación se identifican como **source** y **target**.

- **Source**: Puede ser un objeto de cualquier tipo. En la práctica, normalmente será un objeto de datos de C#.

- **Target**: El objeto target debe descender de **BindableObject**. Todos los controles de Xamarin.Forms descienden de esta clase, por lo que, en la práctica, normalmente el objeto target será un control de Xamarin.Forms.


Source
Target
Any Object
Any public property
Binding
BindableObject
BindableProperty

- En la mayoría de los escenarios, tendremos múltiples controles que utilizan la información de un mismo objeto **source**.



- **BindingContext** es una propiedad que permite exponer el objeto **source** a los controles de una pantalla.

C#

```
BindingContext = new Todo()
{
    Title = "Pickup some
            milk",
    Notes = "Stop at the
             Grocery Store!",
    Completed = true
}
```

XAML

```
<StackLayout Padding="20">
    <Entry Text="{Binding Title}"/>
    <Entry Text="{Binding Notes}"/>
    <Switch IsToggled="{Binding Completed}"/>
</StackLayout>
```

- Ejercicio #1: Mostrar información de archivo json

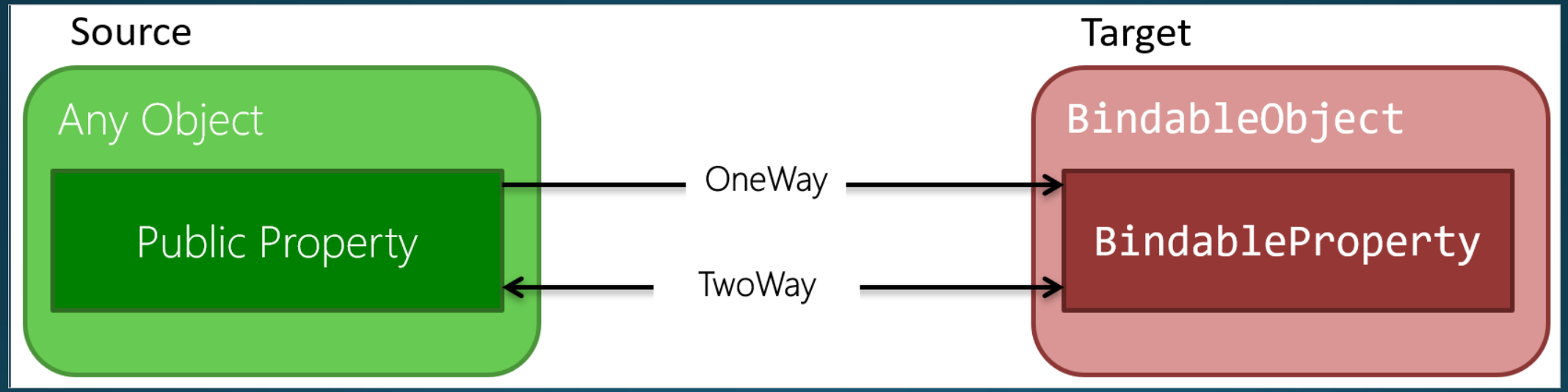
- Ejercicio #2: Reemplazar C# con bindings

Tipos de bindings

- Existen dos tipos de bindings que se pueden establecer:

- **One-way binding**: Transfiere información sólo del objeto **source** al objeto **target**. Las propiedades que no son editables utilizan este tipo (como Label.Text).

- **Two-way binding**: Envía información en ambas direcciones, entre el objeto **source** y el objeto **target**. Las propiedades que son editables utilizan este tipo (como Entry.Text).


Source
Target
Any Object
Public Property
OneWay
TwoWay
BindableObject
BindableProperty

- Cuando el valor de un objeto **target** cambia, automáticamente se genera un evento que actualiza el valor del objeto **source** (**BindableProperty** genera este evento).

- Pero si alguna propiedad de un objeto **source** cambia en tiempo de ejecución, es necesario generar un evento para notificar al objeto **target** de este cambio.

- Ejercicio #3: Generar evento para actualizar objeto target

- En ocasiones, la información que proviene del código (source) no va a coincidir con el tipo de dato del control de la UI (target).
  - Por ejemplo, cuando el objeto target es un control **Image**, y el objeto source es una ubicación de la imagen (de tipo string).

- En estos casos, lo recomendable es generar código personalizado para realizar la conversión.

- Xamarin.Forms provee la interfaz **IValueConverter**, que simplifica este procesos de conversión.

- Ejercicio #4: Crear un binding entre tipos de datos distintos.

ListView

- Una funcionalidad típica en las aplicaciones es mostrar una colección de datos en una lista desplazable (scrollable).

- **ListView** es el control de Xamarin.Forms que permite esta funcionalidad.
  - En Android, se crea el control nativo **ListView** y en iOS el control nativo **UITableview**.

- La fuente de datos de un control **ListView** debe ser una colección del tipo **IEnumerable**.

UITableView

ListView

- Ejercicio #5: Mostrar una colección de datos con ListView

- Ejercicio #6: Mostrar una pantalla al seleccionar elemento de ListView

- Ejercicio #7: Personalizar apariencia de ListView