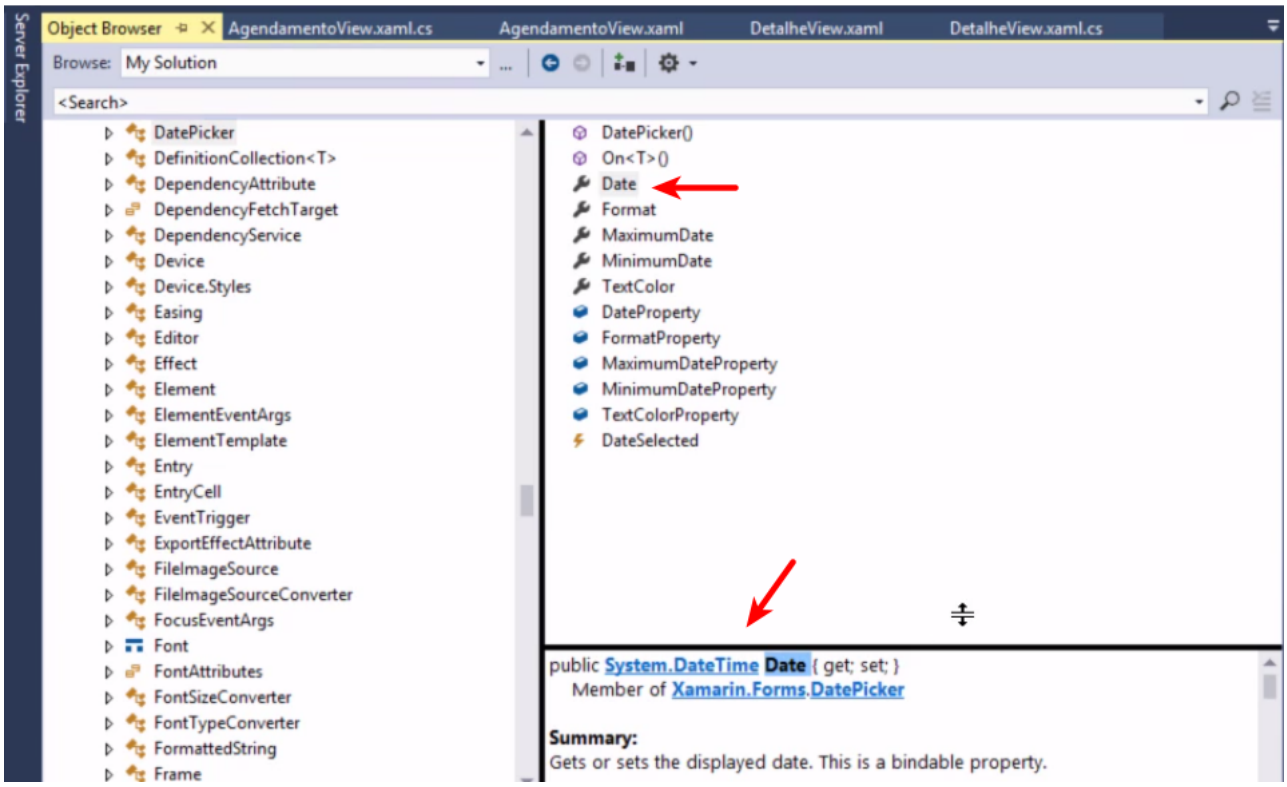
08

# Binding para DatePicker e TimePicker

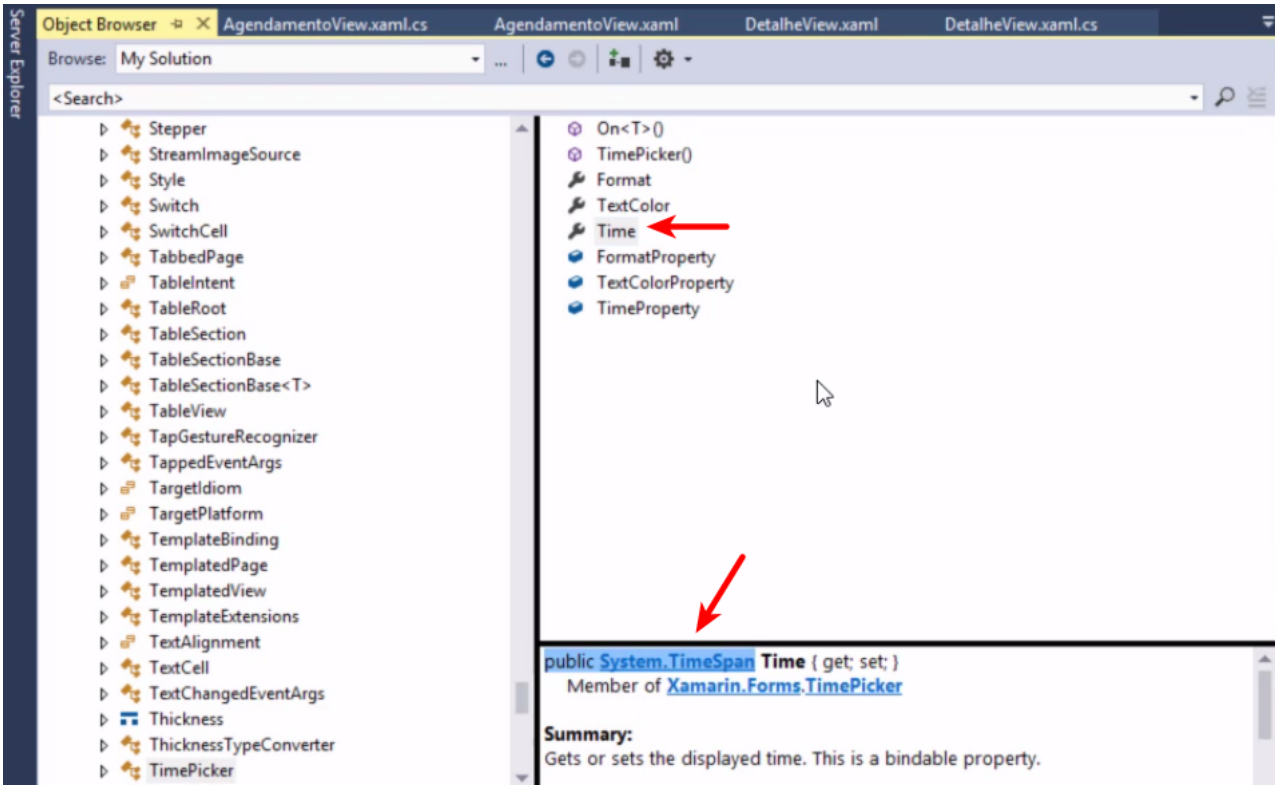
## Transcrição

Para os controles novos ( `DatePicker` e `TimePicker` ), faremos o `Binding` das propriedades que irão armazenar a data e a hora do agendamento de *Test Drive*, as quais serão criadas no *code behind*.

Quais são os tipos destas propriedades, destes controles? Com o cursor em cima de `DatePicker` podemos, pela tecla `F12` , navegar até a definição deste objeto no "*Object Browser*". Ali, temos a propriedade `Date` , cujo tipo é `DateTime` :




Em relação ao `TimePicker` , o tipo da propriedade que precisaremos usar para armazenar o tempo é `Time` que, na definição, se encontra como `TimeSpan` :




Utilizaremos estes dois tipos na criação das propriedades a serem utilizadas neste `Binding` no *code behind*. Abrindo-se `AgendamentoView.xaml.cs` criaremos as duas propriedades, uma para data e outra para a hora, após o e-mail:

```
public DateTime DataAgendamento { get; set; }
public TimeSpan HoraAgendamento { get; set; }
```

Usaremos o `DataAgendamento` como `Binding` no `DatePicker` presente em `AgendamentoView.xaml` :

```
<ViewCell>
    <StackLayout Orientation="Horizontal" Margin="12,0,0,0">
        <Label Text="Data:"></Label>
        <DatePicker Date="{Binding DataAgendamento}"></DatePicker>
    </StackLayout>
</ViewCell>
<ViewCell>
    <StackLayout Orientation="Horizontal" Margin="12,0,0,0">
        <Label Text="Hora:"></Label>
        <TimePicker Time="{Binding HoraAgendamento}"></TimePicker>
    </StackLayout>
</ViewCell>
```

Verificaremos se nada foi quebrado rodando a aplicação. Veremos que houve uma pequena mudança: a data padrão anteriormente era a atual, agora, a data será `1/1/1900` , porque de acordo com o padrão do sistema, a propriedade contida no *code behind* ( `DataAgendamento` ) tem esta data. Modificaremos a definição da propriedade `DataAgendamento` para trazer como data a atual.

Voltando ao `AgendamentoView.xaml.cs` , modificaremos a inicialização de `DataAgendamento` expandindo-se a propriedade, tendo-se uma pública e uma privada.

Trata-se da forma tradicional de se criar uma propriedade, que não utilizará *Auto-Property* como feito em relação aos campos em "Seus Dados", por exemplo. Isto serve para melhorarmos a inicialização de `dataAgendamento` a partir de `DateTime.Today`, que permitirá a exibição da data de hoje:

```
DateTime dataAgendamento = DateTime.Today;
public DateTime DataAgendamento
{
    get
    {
        return dataAgendamento;
    }
    set
    {
        dataAgendamento = value;
    }
}
```

Agora rodaremos a app para ver se isto se reflete no emulador, o que de fato acontece.