Gestion des évènements

La gestion d'événements en MAUI sans MVVM utilise le **code-behind** pour traiter les interactions utilisateur. C'est l'approche la plus directe où chaque contrôle XAML déclenche des méthodes C# correspondantes.

1. Boutons - Événement Clicked

XAML

```
<Button Text="Cliquez-moi"
        Clicked="OnButtonClicked"
        x:Name="btnClickMe" />
```

Code-Behind

```
private void OnButtonClicked(object sender, EventArgs e)
{
    // Accès au bouton via sender
    Button button = (Button)sender;
    button.Text = "Cliqué !";

    // Ou via x:Name
    btnClickMe.BackgroundColor = Colors.Green;
}
```

> *On déduit de cet exemple, que la propriété ‘x:Name‘ définit le nom de l’attribut dans le code behind. C’est un peu comme si on ajoutait un attribut dans la classe avec ce nom là... Ceci est valable pour tout composant XAML.*

2. Champs de Saisie - Événement TextChanged

XAML

```
<Entry Placeholder="Tapez ici..."
       TextChanged="OnTextChanged"
       Completed="OnTextCompleted"
       x:Name="txtName" />
```

Code-Behind

```
private void OnTextChanged(object sender, TextChangedEventArgs e)
{
    // Accès aux valeurs anciennes et nouvelles
    string oldText = e.OldTextValue;
    string newText = e.NewTextValue;
```

```
    // Validation en temps réel
    if (newText.Length >= 3)
    {
        txtName.BackgroundColor = Colors.LightGreen;
    }
    else
    {
        txtName.BackgroundColor = Colors.LightPink;
    }
}

private void OnTextCompleted(object sender, EventArgs e)
{
    Entry entry = (Entry)sender;
    DisplayAlert("Info", $"Texte saisi : {entry.Text}", "OK");
}
```

Points clés :

- `TextChanged` se déclenche à chaque caractère tapé
- `TextChangedEventArgs` fournit les propriétés `NewTextValue` et `OldTextValue`
- `Completed` se déclenche quand l'utilisateur appuie sur Entrée

3. Cases à Cocher - Événement CheckedChanged

XAML

```
<CheckBox x:Name="AcceptTerms"
          CheckedChanged="OnCheckboxChanged"
          Color="Blue" />
<Label Text="J'accepte les conditions" />
```

Code-Behind

```
private void OnCheckboxChanged(object sender, CheckedChangedEventArgs e)
{
    CheckBox checkbox = (CheckBox)sender;
    bool isChecked = e.Value;

    if (isChecked)
    {
        DisplayAlert("Confirmé", "Conditions acceptées", "OK");
        // Activer d'autres contrôles
        btnSubmit.IsEnabled = true;
    }
    else
    {
        btnSubmit.IsEnabled = false;
```

```
    }
}
```

Point clé : `CheckedChanged` est déclenché quand `IsChecked` change, et `CheckedChangedEventArgs` contient la propriété `Value` de type `bool`.

Exemple Pratique Complet

XAML - Page d'inscription

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage x:Class="MonApp.RegisterPage"
             xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             Title="Inscription">

    <StackLayout Padding="20" Spacing="15">

        <!-- Nom utilisateur -->
        <Label Text="Nom d'utilisateur :" />
        <Entry x:Name="txtUsername"
               TextChanged="OnUsernameChanged"
               Placeholder="Min. 3 caractères" />

        <!-- Email -->
        <Label Text="Email :" />
        <Entry x:Name="txtEmail"
               TextChanged="OnEmailChanged"
               Keyboard="Email"
               Placeholder="votre@email.com" />

        <!-- Conditions -->
        <StackLayout Orientation="Horizontal">
            <CheckBox x:Name="chkConditionsAccepted"
                      CheckedChanged="OnConditionsChanged" />
            <Label Text="J'accepte les conditions d'utilisation" />
        </StackLayout>

        <!-- Newsletter -->
        <StackLayout Orientation="Horizontal">
            <CheckBox x:Name="chkNewsletter"
                      CheckedChanged="OnNewsletterChanged" />
            <Label Text="Recevoir la newsletter" />
        </StackLayout>

        <!-- Bouton inscription -->
        <Button x:Name="btnRegister"
                Text="S'inscrire"
                Clicked="OnRegisterClicked"
                IsEnabled="False"
                BackgroundColor="Gray" />
```

```
        <!-- Statut -->
        <Label x:Name="lblStatus"
               Text="Remplissez le formulaire"
               HorizontalOptions="Center" />

    </StackLayout>
</ContentPage>
```

Code-Behind Complet

```
public partial class RegisterPage : ContentPage
{
    private bool _isNameValid = false;
    private bool _isEmailValid = false;
    private bool _conditionsAccepted = false;

    public RegisterPage()
    {
        InitializeComponent();
    }

    private void OnUsernameChanged(object sender, TextChangedEventArgs e)
    {
        _isNameValid = e.NewTextValue.Length >= 3;

        if (_isNameValid)
        {
            txtUsername.BackgroundColor = Colors.LightGreen;
            lblStatus.Text = "Nom valide";
        }
        else
        {
            txtUsername.BackgroundColor = Colors.LightPink;
            lblStatus.Text = "Nom trop court (min. 3 caractères)";
        }

        CheckForm();
    }

    private void OnEmailChanged(object sender, TextChangedEventArgs e)
    {
        _isEmailValid = e.NewTextValue.Contains("@") &&
e.NewTextValue.Contains(".");

        if (_isEmailValid)
        {
            txtEmail.BackgroundColor = Colors.LightGreen;
        }
        else
        {
```

```
            txtEmail.BackgroundColor = Colors.LightPink;
        }

        CheckForm();
    }

    private void OnConditionsChanged(object sender, CheckedChangedEventArgs e)
    {
        _conditionsAccepted = e.Value;
        CheckForm();
    }

    private void OnNewsletterChanged(object sender, CheckedChangedEventArgs e)
    {
        if (e.Value)
        {
            DisplayAlert("Newsletter", "Merci pour votre abonnement !", "OK");
        }
    }

    private void CheckForm()
    {
        bool formulaireValide = _isNameValid && _isEmailValid &&
_conditionsAccepted;

        btnRegister.IsEnabled = formulaireValide;
        btnRegister.BackgroundColor = formulaireValide ? Colors.Green : Colors.Gray;

        if (formulaireValide)
        {
            lblStatus.Text = "Formulaire complet";
        }
    }

    private async void OnRegisterClicked(object sender, EventArgs e)
    {
        bool confirmation = await DisplayAlert(
            "Confirmation",
            $"Inscrire {txtUsername.Text} avec {txtEmail.Text} ?",
            "Oui", "Non"
        );

        if (confirmation)
        {
            // Simulation inscription
            btnRegister.Text = "Inscription...";
            btnRegister.IsEnabled = false;

            await Task.Delay(2000); // Simulation délai serveur

            await DisplayAlert("Succès", "Inscription réussie !", "OK");
```

```
            // Réinitialiser le formulaire
            txtUsername.Text = "";
            txtEmail.Text = "";
            chkConditionsAccepted.IsChecked = false;
            chkNewsletter.IsChecked = false;
        }
    }
}
```

Points Essentiels à Retenir

1. **Événements principaux** : `Clicked`, `TextChanged`, `CheckedChanged`
2. **Accès aux contrôles** : Via `sender` ou `x:Name`
3. **Validation temps réel** : Possible avec `TextChanged`
4. **Gestion d'état** : Variables privées pour suivre l'état du formulaire