Module J2 : Fondamentaux des Widgets et Mise en Page (Partie 1)

Objectif : Comprendre l'architecture basée sur les Widgets et construire des interfaces utilisateur simples.

2.1. Introduction aux Widgets (Durée Estimée : 40 min)

A. Le Principe Fondamental : "Tout est Widget"

Dans Flutter, l'ensemble de l'interface utilisateur (UI) est construit à partir de **Widgets**. Un Widget est une description d'une partie de l'interface utilisateur. Il peut s'agir d'un élément visible (texte, image, bouton) ou d'un élément invisible qui gère la mise en page, le style, l'interaction ou l'état (comme `Row`, `Column`, ou `GestureDetector`).

Le Widget Tree (Arbre de Widgets) : Votre application est une arborescence de Widgets imbriqués. Chaque Widget est un composant qui délègue ses responsabilités à d'autres Widgets, permettant une composition puissante et flexible.

B. Les Deux Types de Widgets Fondamentaux

Il existe deux catégories principales de Widgets, basées sur leur capacité à gérer des données changeantes (l'état) :

1. StatelessWidget (Widget sans État)

Un `StatelessWidget` est un Widget qui ne change pas après sa construction initiale. Il est **immuable**.

- **Utilisation :** Pour afficher des informations statiques qui ne dépendront pas des interactions de l'utilisateur ou des changements de données (exemples : un logo, un titre, une icône).
- **Méthode Clé :** La méthode `build(BuildContext context)` est la seule méthode requise. Elle décrit l'interface utilisateur que ce Widget doit créer.

Exemple de `StatelessWidget` :

```
import 'package:flutter/material.dart';

class MonTitre extends StatelessWidget {
  final String titre; // Propriété immuable

  const MonTitre({Key? key, required this.titre}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // Le Widget retourne un simple Text
    return Text(
      titre,
      style: const TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
    );
  }
}
```

2. StatefulWidget (Widget avec État)

Un `StatefulWidget` est un Widget qui peut changer dynamiquement en réponse à des événements (clics, données reçues, minuteurs, etc.). Il est **mutable**.

- **Structure :** Un `StatefulWidget` est composé de deux classes :
  1. La classe `StatefulWidget` elle-même (qui est immuable).
  2. La classe `State` associée, qui contient les données changeantes et la méthode `build()`.
- **Méthode Clé :** La méthode `setState(() { ... })` est utilisée dans la classe `State`. Elle notifie Flutter que l'état interne a changé et que le Widget doit être reconstruit (re-appel de la méthode `build()`) pour refléter les nouvelles données.

Exemple de `StatefulWidget` (Compteur) :

```dart
import 'package:flutter/material.dart';

class MonCompteur extends StatefulWidget {
  const MonCompteur({Key? key}) : super(key: key);

  @override
  State<MonCompteur> createState() => _MonCompteurState();
}

class _MonCompteurState extends State<MonCompteur> {
  int _compteur = 0; // L'état (la donnée changeante)

  void _incrementerCompteur() {
    // Appel de setState pour mettre à jour l'état et reconstruire le Widget
    setState(() {
      _compteur++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Text('Valeur actuelle : $_compteur'),
        ElevatedButton(
          onPressed: _incrementerCompteur,
          child: const Text('Incrémenter'),
        ),
      ],
    );
  }
}
```

C. Le Cycle de Vie du Développement : Hot Reload et Hot Restart

Flutter offre deux fonctionnalités puissantes pour accélérer votre développement :

Fonctionnalité	Description	État de l'Application	Quand l'utiliser ?
Hot Reload	Injecte le nouveau code source dans l'application en cours d'exécution.	**Conservé** (l'état de la variable `_compteur` reste le même).	Pour les changements visuels ou les petites modifications de logique.
Hot Restart	Redémarre l'application depuis le début (`main()`).	**Perdu** (l'état de la variable `_compteur` reviendrait à 0).	Pour les changements de structure importants (ex: modification du `main()` ou ajout de dépendances).

Exercices Pratiques (2.1)

1. **Convertir un `StatelessWidget` en `StatefulWidget` :** Prenez le `MonTitre` ci-dessus et convertissez-le en `StatefulWidget` pour qu'il puisse changer de couleur après 5 secondes (sans utiliser `setState` pour l'instant, juste pour comprendre la structure).
2. **Modifier le texte avec Hot Reload :** Modifiez le texte "Incrémenter" du bouton dans l'exemple `MonCompteur` en "Ajouter 1" et utilisez le Hot Reload pour voir le changement instantanément, sans perdre la valeur actuelle du compteur.

2.2. Widgets de Mise en Page (Layout) (Durée Estimée : 80 min)

A. Widgets d'Affichage de Base

Ces Widgets sont les briques minimales pour afficher du contenu :

- `Text('Bonjour')` : Affiche une chaîne de caractères.
- `Image.asset('chemin/image.png')` : Affiche une image locale (stockée dans le dossier `assets`).
- `Icon(Icons.star)` : Affiche une icône du jeu d'icônes Material Design.

B. Les Widgets de Conteneur et d'Espacement

1. Container

Le `Container` est le Widget le plus polyvalent. Il permet d'ajouter du style, de la décoration, de la couleur, des marges et du remplissage à son enfant.

Propriété	Description
`child`	Le Widget contenu dans le `Container`.
`color`	La couleur de fond du `Container`.
`padding`	L'espace vide **à l'intérieur** du `Container` (entre la bordure et l'enfant).
`margin`	L'espace vide **à l'extérieur** du `Container` (entre le `Container` et les autres Widgets).
`decoration`	Pour des styles avancés (bordures, ombres, dégradés).

2. Padding et SizedBox

- `Padding` : Widget dédié à l'ajout d'espace autour de son enfant. Il est souvent préféré au `Container` si vous n'avez besoin que d'un remplissage.
- `SizedBox` : Utilisé principalement pour forcer un Widget à avoir une taille spécifique ou pour créer un espace vide précis entre deux Widgets (`SizedBox(height: 10)`).

C. Organisation Verticale et Horizontale : Row et Column

Les Widgets `Row` et `Column` sont la base de la mise en page. Ils prennent une liste de Widgets enfants (`children`) et les alignent.

Widget	Direction	Alignement Principal (`mainAxisAlignment`)	Alignement Croisé (`crossAxisAlignment`)
`Row`	**Horizontal**	Gère l'espace horizontal (ex: `start`, `end`, `center`, `spaceBetween`).	Gère l'espace vertical (ex: `start`, `end`, `center`, `stretch`).
`Column`	**Vertical**	Gère l'espace vertical (ex: `start`, `end`, `center`, `spaceBetween`).	Gère l'espace horizontal (ex: `start`, `end`, `center`, `stretch`).

Règle d'or : Dans une `Row`, le `mainAxisAlignment` est horizontal. Dans une `Column`, le `mainAxisAlignment` est vertical.

D. Alignement : Center et Align

- `Center` : Centre son enfant dans l'espace disponible.
- `Align` : Permet un contrôle plus fin en spécifiant un point d'alignement (ex: `Alignment.bottomRight` pour placer l'enfant en bas à droite).

Exercice Pratique (2.2) : Création d'une Carte de Profil Utilisateur

Objectif : Créer une carte de profil en utilisant `Column`, `Row`, `Container` et `Padding`.

Consignes :

1. Utiliser un `Column` comme Widget principal pour empiler les éléments.
2. Placer une `Image` ou un `Icon` en haut.
3. Ajouter un `Text` pour le nom et un autre pour le titre.
4. Utiliser un `Row` pour aligner une icône de téléphone et le numéro de contact.
5. Utiliser `Padding` pour espacer les éléments du bord de l'écran.
6. Utiliser un `Container` pour encadrer le numéro de contact avec une couleur de fond.

Solution (Code à commenter et expliquer) :

```dart
import 'package:flutter/material.dart';

class CarteProfil extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Carte de Profil')),
      body: Center(
        // Le Column empile les éléments verticalement
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center, // Centrage vertical
          children: <Widget>[
            // 1. Image de profil
            const Icon(Icons.person, size: 100, color: Colors.blue),

            // 2. Espace vertical
            const SizedBox(height: 20),

            // 3. Nom de l'utilisateur
            const Text(
              'Ibrahim Ahmad',
              style: TextStyle(fontSize: 28, fontWeight: FontWeight.bold),
            ),

            // 4. Titre
            const Text(
              'Développeur & Formateur Flutter',
              style: TextStyle(fontSize: 18, color: Colors.grey),
            ),

            // 5. Espace vertical
            const SizedBox(height: 30),

            // 6. Conteneur pour le numéro de contact
            // Utilisation de Padding pour l'espace extérieur
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 40.0),
              child: Container(
                padding: const EdgeInsets.all(15.0), // Padding intérieur
                decoration: BoxDecoration(
                  color: Colors.blue.shade50,
                  borderRadius: BorderRadius.circular(10),
                  border: Border.all(color: Colors.blue),
                ),
                // Le Row aligne l'icône et le texte horizontalement
                child: const Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: <Widget>[
                    Icon(Icons.phone, color: Colors.blue),
                    SizedBox(width: 10),
                    Text(
                      '+227 99 46 35 94',
                      style: TextStyle(fontSize: 20, fontWeight: FontWeight.w600),
                    ),
                  ],
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Exercices Pratiques (2.2) : Création d'une Carte de Profil Utilisateur

Objectif : Créer une carte de profil en utilisant `Column`, `Row`, `Container` et `Padding`.

Consignes :

1. Utiliser un `Column` comme Widget principal pour empiler les éléments.
2. Placer une `Image` ou un `Icon` en haut.
3. Ajouter un `Text` pour le nom et un autre pour le titre.
4. Utiliser un `Row` pour aligner une icône de téléphone et le numéro de contact.
5. Utiliser `Padding` pour espacer les éléments du bord de l'écran.
6. Utiliser un `Container` pour encadrer le numéro de contact avec une couleur de fond.

Solution (Code à commenter et expliquer) :

```dart
import 'package:flutter/material.dart';

class CarteProfil extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Carte de Profil')),
      body: Center(
        // Le Column empile les éléments verticalement
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center, // Centrage vertical
          children: <Widget>[
            // 1. Image de profil
            const Icon(Icons.person, size: 100, color: Colors.blue),

            // 2. Espace vertical
            const SizedBox(height: 20),

            // 3. Nom de l'utilisateur
            const Text(
              'Ibrahim Ahmad',
              style: TextStyle(fontSize: 28, fontWeight: FontWeight.bold),
            ),

            // 4. Titre
            const Text(
              'Développeur & Formateur Flutter',
              style: TextStyle(fontSize: 18, color: Colors.grey),
            ),

            // 5. Espace vertical
            const SizedBox(height: 30),

            // 6. Conteneur pour le numéro de contact
            // Utilisation de Padding pour l'espace extérieur
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 40.0),
              child: Container(
                padding: const EdgeInsets.all(15.0), // Padding intérieur
                decoration: BoxDecoration(
                  color: Colors.blue.shade50,
                  borderRadius: BorderRadius.circular(10),
                  border: Border.all(color: Colors.blue),
                ),
                // Le Row aligne l'icône et le texte horizontalement
                child: const Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: <Widget>[
                    Icon(Icons.phone, color: Colors.blue),
                    SizedBox(width: 10),
                    Text(
                      '+227 99 46 35 94',
                      style: TextStyle(fontSize: 20, fontWeight: FontWeight.w600),
                    ),
                  ],
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```