TIERS LIMITED SUMMER INTERNSHIP 2024

MOBILE APP DEVELOPMENT

Stateless Class and Stateless Widgets

CLASS # 4

TIERS
Limited

Overview

In this class, we will delve into the details of stateless classes and stateless widgets in Flutter. Understanding these concepts is fundamental to building static user interfaces that do not require mutable state.

Objectives

- Understand the concept of stateless classes and stateless widgets.
- Learn how to create and use various stateless widgets.
- Explore the widget tree and how to build UI hierarchies.
- Understand how to set up and use assets in a Flutter project.

Stateless Class:

Stateless Class:

- **Definition**: A stateless class is one that does not maintain any state of its own. Once it is created, it cannot change.
- **Purpose**: Used for widgets that do not change over time or in response to user interactions.

Stateless Widgets:

- **Definition**: Stateless widgets are immutable, and their properties cannot change all values are final.
- **Use Case**: Ideal for static content or when the widget's configuration depends only on the parent widget and doesn't change over time.

Key Points:

- Stateless widgets are created by extending the **`StatelessWidget`** class.
- They implement a **`build`** method that describes the part of the user interface represented by the widget.

Creating a Stateless Widget

1. **Define the Widget Class**:
    - Extend **`StatelessWidget`**.
    - Override the **`build`** method to describe the UI.

```
class MyStatelessWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text('Hello, Flutter!');
  }
}
```

2. **Using the Widget**:
    - You can use this widget just like any other widget in the widget tree.

```
void main() {
  runApp(MaterialApp(
    home: Scaffold(
      body: Center(
        child: MyStatelessWidget(),
      ),
    ),
  ));
}
```

Common Stateless Widgets

1. **Text**:
    - Used to display a string of text with a single style.

```
Text(
  'Hello, Flutter!',
  style: TextStyle(
    fontSize: 24,
    fontWeight: FontWeight.bold,
    color: Colors.blue,
  ),
);
```

2. **Image**:
    - Used to display images. `Image.network('url of image');`
    - **Image.asset**: Used to display images from the local assets.

```
Image.asset('assets/images/my_image.png');
```

Setting Up Assets in Flutter Project:

- Add your image files to an `assets/images` directory in your Flutter project.
- Update the `pubspec.yaml` file to include the assets:

```
flutter:
  assets:
    - assets/images/my_image.png
```

3. **Icon**:
    - Displays a graphical icon.

```
Icon(Icons.star, color: Colors.red);
```

4. **ElevatedButton**:
    - A material design button that triggers an action when pressed.

```
ElevatedButton(
  onPressed: () {
    print('Button Pressed');
  },
  child: Text('Press Me'),
);
```

5. **Checkbox**:
    - A material design checkbox.

```
Checkbox(
  value: true,
  onChanged: (bool? newValue) {
    // Handle
  },
);
```

6. **Switch**:
    - A material design switch.

```
Switch(
  value: true,
  onChanged: (bool newValue) {
    // Handle
  },
);
```

7. **ListView**:
    - A scrollable list of widgets arranged linearly.

```
ListView(
  children: <Widget>[
    ListTile(
      leading: Icon(Icons.map),
      title: Text('Map'),
    ),
    ListTile(
      leading: Icon(Icons.photo_album),
      title: Text('Album'),
    ),
    ListTile(
      leading: Icon(Icons.phone),
      title: Text('Phone'),
    ),
  ],
);
```

8. **SingleChildScrollView**:
    - A box in which a single widget can be scrolled.

```
SingleChildScrollView(
  child: Column(
    children: <Widget>[
      Text('First'),
      Text('Second'),
      Text('Third'),
    ],
  ),
);
```

9. **Padding**:
    - Adds padding around a child widget.

```
Padding(
  padding: EdgeInsets.all(10.0),
  child: Text('Hello, Flutter!'),
);
```

10. **Center**:
    - Centers its child widget within itself.

```
Center(
  child: Text('Hello, Flutter!'),
);
```

Types of Buttons:

1. **IconButton**:
    - A button that shows an icon instead of text.

```
IconButton(
  icon: Icon(Icons.volume_up),
  tooltip: 'Increase volume',
  onPressed: () {
    print('Volume up');
  },
);
```

2. **OutlinedButton**:
    - A material design outlined button.

```
OutlinedButton(
  onPressed: () {
    print('Outlined Button Pressed');
  },
  child: Text('Outlined Button'),
);
```

3. **TextButton**:
    - A material design flat button.

```
TextButton(
  onPressed: () {
    print('Text Button Pressed');
  },
  child: Text('Text Button'),
);
```

Exercises:

Exercise 1:

Create a vertically scrollable list of containers with different background colors.

Exercise 2:

Create the given UI: