TIERS LIMITED SUMMER INTERNSHIP 2024

MOBILE APP DEVELOPMENT

Dialogs in Flutter

CLASS # 10

TIERS
Limited

Overview

In this class, we will learn how to create and manage dialogs and alerts in a Flutter application. Dialogs are used to show important messages to users, get user inputs, or present options. Understanding how to use dialogs effectively is essential for creating intuitive and user-friendly interfaces.

Objectives

- Understand the purpose of dialogs and alerts.
- Learn to create and display different types of dialogs.
- Manage user interactions with dialogs.
- Create stateful dialogs to manage user inputs dynamically.

Dialogs and Alerts in Flutter

AlertDialog:

`AlertDialog` is a material design alert dialog.

Example:

```
showDialog(
  context: context,
  builder: (BuildContext context) {
    return AlertDialog(
      title: Text('Alert Dialog'),
      content: Text('This is a basic alert dialog.'),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.of(context).pop();
          },
          child: Text('OK'),
        ),
      ],
    );
  },
);
```

SimpleDialog:

`SimpleDialog` is a material design simple dialog.

Example: SimpleDialog

```
showDialog(
  context: context,
```

```
  builder: (BuildContext context) {
    return SimpleDialog(
      title: Text('Simple Dialog'),
      children: [
        SimpleDialogOption(
          onPressed: () {
            Navigator.of(context).pop();
          },
          child: Text('Option 1'),
        ),
        SimpleDialogOption(
          onPressed: () {
            Navigator.of(context).pop();
          },
          child: Text('Option 2'),
        ),
      ],
    );
  },
);
```

Dialog

`Dialog` is a material design dialog.

Example: Custom Dialog

```
showDialog(
  context: context,
  builder: (BuildContext context) {
    return Dialog(
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            Text('Custom Dialog', style: TextStyle(fontSize: 24)),
            SizedBox(height: 16),
            Text('This is a custom dialog.'),
            SizedBox(height: 16),
            ElevatedButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: Text('Close'),
            ),
          ],
        ),
      ),
    );
  },
);
```

Input Dialog:

Dialogs can also be used to get user inputs.

Example:

```
Future<void> _showInputDialog(BuildContext context) async {
  String inputText = '';
  return showDialog<void>(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Input Dialog'),
        content: TextField(
          onChanged: (value) {
            inputText = value;
          },
          decoration: InputDecoration(hintText: "Enter your input here"),
        ),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: Text('Cancel'),
          ),
          TextButton(
            onPressed: () {
              print('Input: $inputText');
              Navigator.of(context).pop();
            },
            child: Text('Submit'),
          ),
        ],
      );
    },
  );
}
```

Stateful Dialog

Sometimes, you need to create a dialog that maintains state across user interactions. This is where a stateful dialog comes in handy.

Example: Stateful Dialog

```
void _showStatefulDialog(BuildContext context) {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return StatefulBuilder(
        builder: (BuildContext context, StateSetter setState) {
          int counter = 0;
```

```
            return AlertDialog(
              title: Text('Stateful Dialog'),
              content: Column(
                mainAxisSize: MainAxisSize.min,
                children: [
                  Text('Counter: $counter'),
                  ElevatedButton(
                    onPressed: () {
                      setState(() {
                        counter++;
                      });
                    },
                    child: Text('Increment'),
                  ),
                ],
              ),
              actions: [
                TextButton(
                  onPressed: () {
                    Navigator.of(context).pop();
                  },
                  child: Text('Close'),
                ),
              ],
            );
          },
        );
      },
    );
}
```

Exercises:

Exercise 1: Create an AlertDialog for Deleting an item.

Exercise 2: Create a SimpleDialog with multiple options that take you to another screen.

Exercise 3: Create a Dialog with Custom content and a close button.

Exercise 4: Create an input dialog that allows the user to enter text and submit it.

Exercise 5: Create a stateful dialog that maintains a counter and updates its content when a button is pressed.