Linnæus University

Introduction to App Development with Flutter

Tobias Andersson Gidlund

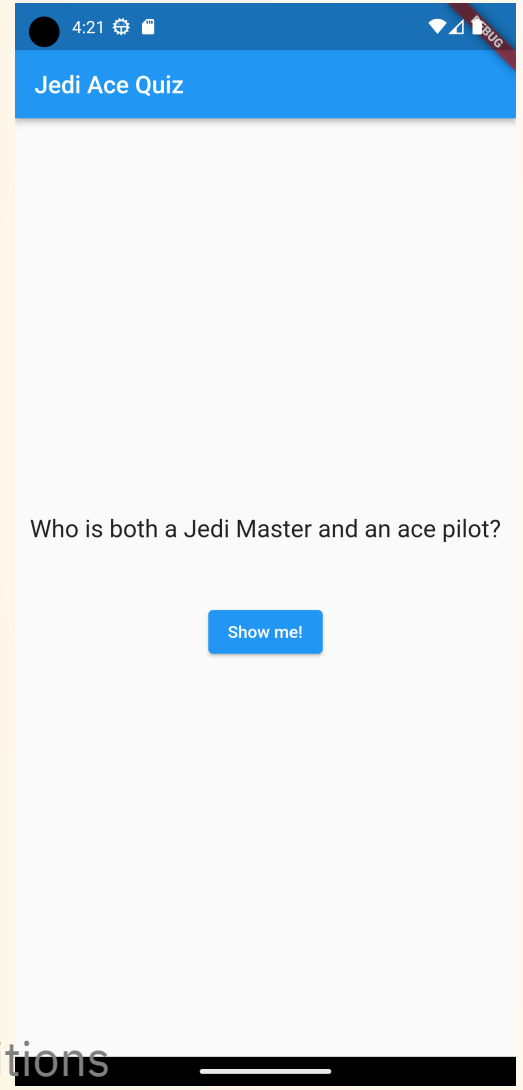
Transition between screens

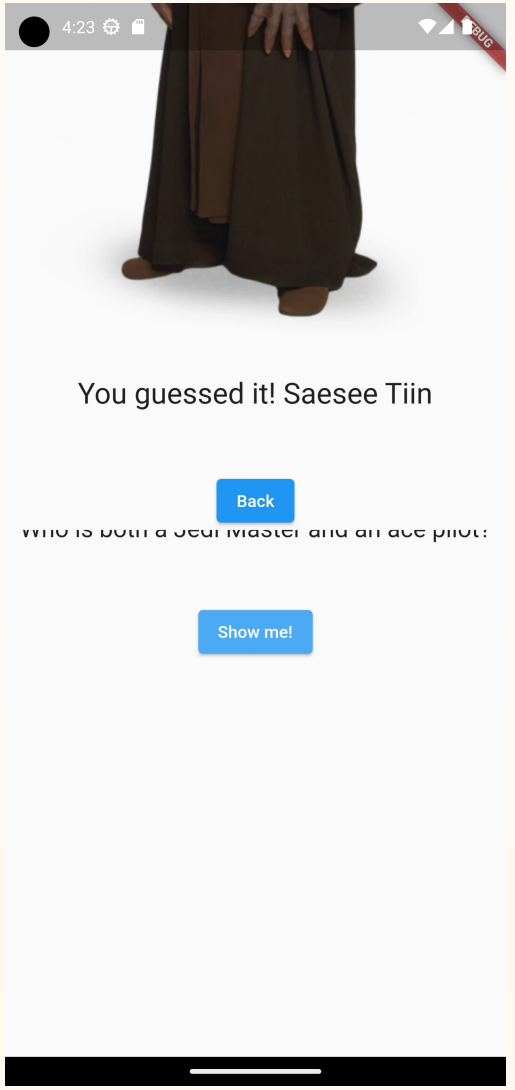
- The previous lecture dealt with navigation, but Flutter also wants to do things in style
- It is, therefore, possible to animate and style the transitions between two screens
- This is done using the `PageRouteBuilder` which creates an `Animation` object
  - Animation is a powerful part of Flutter
- This animation is then further described using a `SlideTransition`

Building the transition

- The first thing to do is to create a route using `PageRouteBuilder`
- It has two callbacks (methods that will be called):
  - `pageBuilder` for defining what screen/page to go to
  - `transitionsBuilder` for the animation itself during the transition
- In the `transitionsBuilder` you define the transition to be made *as well as* the `SlideTransition` widget
  - A subclass of `AnimatedWidget`

Running app


4:21
Jedi Ace Quiz
Who is both a Jedi Master and an ace pilot?
Show me!


4:23
You guessed it! Saesee Tiin
Back
Show me!


4:22
Answer
You guessed it! Saesee Tiin
Back

First page

```
class FirstPage extends StatefulWidget {
  const FirstPage({super.key});

  @override
  State<FirstPage> createState() => _FirstPageState();
}

class _FirstPageState extends State<FirstPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Jedi Ace Quiz")),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text(
              "Who is both a Jedi Master and an ace pilot?",
              style: TextStyle(fontSize: 20),
            ),
            const SizedBox(height: 50),
            ElevatedButton(
              onPressed: () => _navigateToSecondPage(context),
              child: const Text('Show me!'),
            ),
          ],
        ),
      ),
    );
  }
```

Navigation transition

- Again, the _ are used to denote that no variables are sent as parameters
- Also present is the `transitionDuration` property that simply states the duration of the transition

```
  void _navigateToSecondPage(BuildContext context) {
    Navigator.of(context).push(
      PageRouteBuilder(
        pageBuilder: (_, animation1, __) => const SecondPage(),
        transitionDuration: const Duration(milliseconds: 500),
        transitionsBuilder: (_, animation, __, child) {
          return SlideTransition(
            position: Tween(begin: const Offset(0, -1), end: const Offset(0, 0))
                .animate(animation),
            child: child,
          );
        },
      ),
    );
  }
}
```

Second page

```
class SecondPage extends StatelessWidget {
  const SecondPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Answer")),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Expanded(child: Image.asset("images/saeseetiin.png")),
            const SizedBox(height: 30),
            const Text(
              "You guessed it! Saesee Tiin",
              style: TextStyle(fontSize: 24),
            ),
            const SizedBox(height: 50),
            ElevatedButton(
              onPressed: () => Navigator.pop(context),
              child: const Text("Back"),
            ),
          ],
        ),
      ),
    );
  }
}
```

More on animation

- There are several ways to do animations in Flutter
- In the example, the class `Tween` was used
  - Gives a linear change of a value from one state to another
  - Therefore, it has two properties – `begin` and `end`
  - The value in the example is going from *offset* -1 (outside the top of the screen) to 0
  - After that, the `animate()` method is called to do the animation
- All of this is set to the `position` property of the animated widget

Another animation class

- Instead of (or in addition to) `Tween`, it is also possible to use `Curve`
  - Via the class `CurveTween`, combining the two
- The animation is then done on a curve, instead of linear
  - For example `easeIn` and `easeInOut`
  - Almost 40 predefined curves are available, and it is possible to create custom curves
- The curve is sent to the `CurveTween` as a property

Bouncing example

- The next example will create a kind of “bounce” sideways when the new screen appears.
- It will show another way of dividing the code and creating a method that returns a `PageRouteBuilder` widget
- Instead of using `animate()` it will use `animation.drive()` to achieve the same
- As the animation is quite visual, it is not possible to show a static image of it...

First screen

```
lass FirstScreen extends StatelessWidget {
  const FirstScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Screen 1")),
      body: Center(
        child: ElevatedButton(
          onPressed: () => _navigateToSecondScreen(context),
          child: const Text("To the next"),
        ),
      ),
    );
  }

  void _navigateToSecondScreen(BuildContext context) {
    Navigator.of(context).push(
      _createPageRoute(const SecondScreen()),
    );
  }
}
```

Second screen

```
class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Screen 2")),
      body: Center(
        child: ElevatedButton(
          onPressed: () => _navigateToFirstScreen(context),
          child: const Text("And back again"),
        ),
      ),
    );
  }

  void _navigateToFirstScreen(BuildContext context) {
    Navigator.of(context).pushReplacement(
      _createPageRoute(const FirstScreen()),
    );
  }
}
```

The `Curve` animation returned as a `PageRouteBuilder`

- Both a `Curve` and a `Tween` is needed to make the transition

```
PageRouteBuilder _createPageRoute(Widget destination) {
  return PageRouteBuilder(
    pageBuilder: (_, __, ___) => destination,
    transitionDuration: const Duration(milliseconds: 1500),
    transitionsBuilder: (_, animation, __, child) {
      var begin = const Offset(1.0, 0.0);
      var end = Offset.zero;
      var curve = Curves.elasticInOut;

      var tween = Tween(begin: begin, end: end).chain(CurveTween(curve: curve));

      return SlideTransition(
        position: animation.drive(tween),
        child: child,
      );
    },
  );
}
```

More on animation

- Animation in Flutter is not only for transitions
- Almost anything can be animated
  - Text, images, any widget...
- This lecture, however, was about transitions
- There is plenty of sources to find out how to make animated transitions
  - The official docs are a good starting point