**AnimatedWidget**

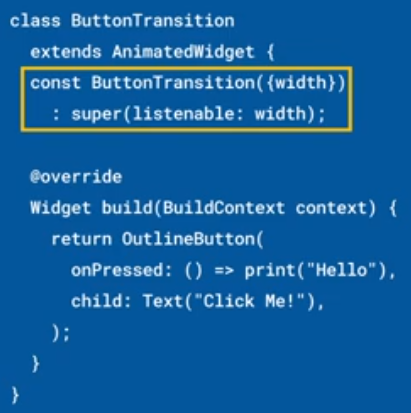
Hay varios AnimatedWidgets ya prearmados:

* SizeTransition
* FadeTransition
* AlignTransition
* ScaleTransition
* DecoratedBoxTransition
* DefaultTextStyleTransition
* PositionedTransition
* SlideTransition
* RotationTransition
* FooTransition
* ButtonTransition

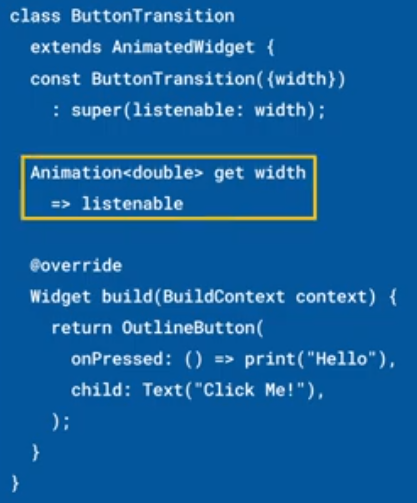
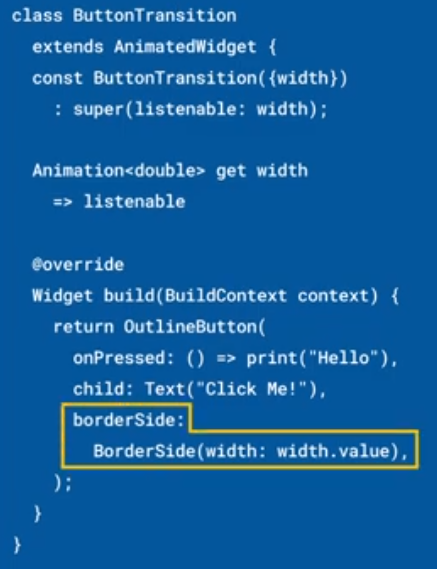
Comenzamos extendiendo AnimatedWidget .

Se requieren dos cosas:

* Un método build que devuelva el widget que se desa animar
* Algo que notifique al widget los valores actualizados como en una animación



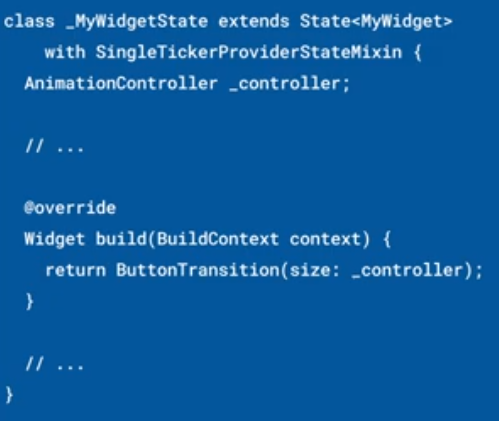
Al constructor hay que pasarle la superclase AnimatedWidget como argumento escuchable. Y finalmente en el builder usamos el valor:

Se necesita algún tipo de animación para usar este widget. La forma más fácil es usando un AnimationController en un widget con estado,

El controlador se inicializa en el initState y luego hay que desecharlo con el dispose.

Con el controlador se puede iniciar, detener o repetir la animación.



|  |  |
| --- | --- |
| **Código** | **Comentarios** |
| import 'package:flutter/material.dart';  class AnimatedWidgetScreen extends StatefulWidget {    const AnimatedWidgetScreen({Key? key}) : super(key: key);    @override    State<AnimatedWidgetScreen> createState() => \_AnimatedWidgetScreenState();  }  class \_AnimatedWidgetScreenState extends State<AnimatedWidgetScreen>      with SingleTickerProviderStateMixin {    late AnimationController \_controller;    late Animation<Color?> \_animation;    bool \_isAnimating = false;  //---------------------------------------------------------------    @override    void initState() {      super.initState();      \_controller = AnimationController(        duration: const Duration(seconds: 1),        vsync: this,      );      \_animation =          ColorTween(begin: Colors.green, end: Colors.blue).animate(\_controller);    }  //---------------------------------------------------------------    void \_handleClick() {      if (\_isAnimating) {        \_controller.reverse();      } else {        \_controller.forward();      }      \_isAnimating = !\_isAnimating;    }  //---------------------------------------------------------------    @override    void dispose() {      \_controller.dispose();      super.dispose();    }  //---------------------------------------------------------------    @override    Widget build(BuildContext context) {      return Scaffold(        appBar: AppBar(          title: const Text('AnimatedWidget'),          centerTitle: true,        ),        body: Center(          child: Column(            mainAxisAlignment: MainAxisAlignment.center,            children: [              ColorTransition(animation: \_animation),              const SizedBox(height: 20),              ElevatedButton(                onPressed: \_handleClick,                child: const Text('Click Me'),              ),            ],          ),        ),      );    }  }  //-----------------------------------------------------------------------------  class ColorTransition extends AnimatedWidget {    const ColorTransition({Key? key, required Animation<Color?> animation})        : super(key: key, listenable: animation);    @override    Widget build(BuildContext context) {      final animation = listenable as Animation<Color?>;      return Container(        height: 100,        width: 100,        color: animation.value,      );    }  } |  |

Imagen que contiene Gráfico

Descripción generada automáticamente Gráfico, Gráfico en cascada

Descripción generada automáticamente

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -