# 9.3 自定义路由切换动画

我们在第二章“路由管理”一节中讲过：Material组件库中提供了一个MaterialPageRoute组件，它可以使用和平台风格一致的路由切换动画，如在iOS上会左右滑动切换，而在Android上会上下滑动切换。现在，我们如果在Android上也想使用左右切换风格，该怎么做？一个简单的作法是可以直接使用CupertinoPageRoute，如：

Navigator.push(context, CupertinoPageRoute(   
 builder: (context)=>PageB(),  
 ));

CupertinoPageRoute是Cupertino组件库提供的iOS风格的路由切换组件，它实现的就是左右滑动切换。那么我们如何来自定义路由切换动画呢？答案就是PageRouteBuilder。下面我们来看看如何使用PageRouteBuilder来自定义路由切换动画。例如我们想以渐隐渐入动画来实现路由过渡，实现代码如下：

Navigator.push(  
 context,  
 PageRouteBuilder(  
 transitionDuration: Duration(milliseconds: 500), //动画时间为500毫秒  
 pageBuilder: (BuildContext context, Animation animation,  
 Animation secondaryAnimation) {  
 return new FadeTransition(  
 //使用渐隐渐入过渡,  
 opacity: animation,  
 child: PageB(), //路由B  
 );  
 },  
 ),  
);

我们可以看到pageBuilder 有一个animation参数，这是Flutter路由管理器提供的，在路由切换时pageBuilder在每个动画帧都会被回调，因此我们可以通过animation对象来自定义过渡动画。

无论是MaterialPageRoute、CupertinoPageRoute，还是PageRouteBuilder，它们都继承自PageRoute类，而PageRouteBuilder其实只是PageRoute的一个包装，我们可以直接继承PageRoute类来实现自定义路由，上面的例子可以通过如下方式实现：

1. 定义一个路由类FadeRoute

* class FadeRoute extends PageRoute {  
   FadeRoute({  
   @required this.builder,  
   this.transitionDuration = const Duration(milliseconds: 300),  
   this.opaque = true,  
   this.barrierDismissible = false,  
   this.barrierColor,  
   this.barrierLabel,  
   this.maintainState = true,  
   });  
    
   final WidgetBuilder builder;  
    
   @override  
   final Duration transitionDuration;  
    
   @override  
   final bool opaque;  
    
   @override  
   final bool barrierDismissible;  
    
   @override  
   final Color barrierColor;  
    
   @override  
   final String barrierLabel;  
    
   @override  
   final bool maintainState;  
    
   @override  
   Widget buildPage(BuildContext context, Animation<double> animation,  
   Animation<double> secondaryAnimation) => builder(context);  
    
   @override  
   Widget buildTransitions(BuildContext context, Animation<double> animation,  
   Animation<double> secondaryAnimation, Widget child) {  
   return FadeTransition(   
   opacity: animation,  
   child: builder(context),  
   );  
   }  
  }

1. 使用FadeRoute

* Navigator.push(context, FadeRoute(builder: (context) {  
   return PageB();  
  }));

虽然上面的两种方法都可以实现自定义切换动画，但实际使用时应优先考虑使用PageRouteBuilder，这样无需定义一个新的路由类，使用起来会比较方便。但是有些时候PageRouteBuilder是不能满足需求的，例如在应用过渡动画时我们需要读取当前路由的一些属性，这时就只能通过继承PageRoute的方式了，举个例子，假如我们只想在打开新路由时应用动画，而在返回时不使用动画，那么我们在构建过渡动画时就必须判断当前路由isActive属性是否为true，代码如下：

@override  
Widget buildTransitions(BuildContext context, Animation<double> animation,  
 Animation<double> secondaryAnimation, Widget child) {  
 //当前路由被激活，是打开新路由  
 if(isActive) {  
 return FadeTransition(  
 opacity: animation,  
 child: builder(context),  
 );  
 }else{  
 //是返回，则不应用过渡动画  
 return Padding(padding: EdgeInsets.zero);  
 }  
}

关于路由参数的详细信息读者可以自行查阅API文档，比较简单，不再赘述。