微信交流群

点击购买

Flutter 路由管理中有两个非常重要的概念：

- **Route**：路由是应用程序**页面**的抽象，对应 Android 中 Activity 和 iOS 中的 ViewController，由 Navigator 管理。
- **Navigator**：Navigator 是一个组件，管理和维护一个基于**堆栈**的历史记录，通过 push 和 pop 进行页面的跳转。

push 和 pop

假设现在有2个页面 A 和 B，A中有一个按钮，点击跳转到 B 页面，A 页面代码：

```
ass APage extends StatelessWidget {
@override
Widget build(BuildContext context) {
  return Container(
    alignment: Alignment.center,
    child: RaisedButton(
      child: Text('A 页面'),
      onPressed: () {
        Navigator.of(context).push(MaterialPageRoute(builder: (context) {
          return BPage();
        }));
      },
```

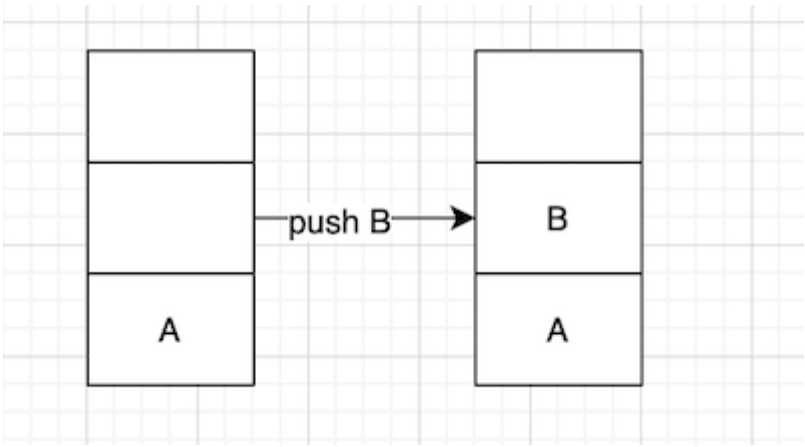
```
    ),
  );
}
```

B 页面代码：

```
class BPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        alignment: Alignment.center,
        child: RaisedButton(
          child: Text('B 页面'),
          onPressed: () {

          },
        ),
      ),
    );
  }
}
```

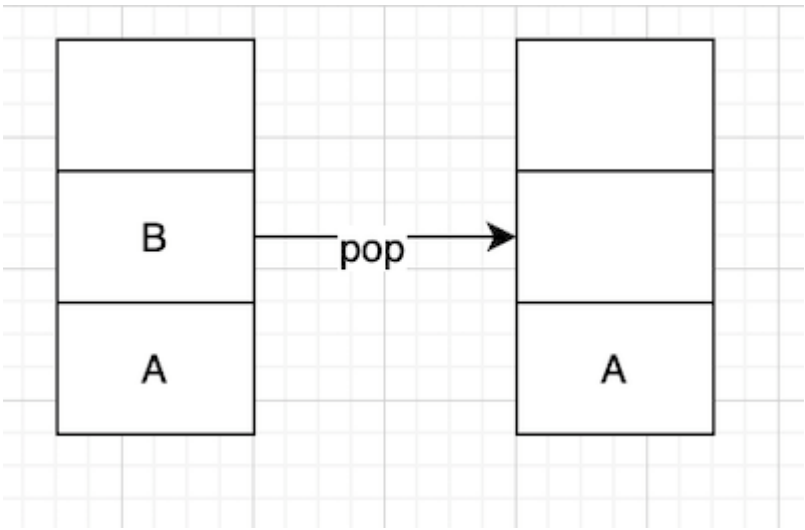
当应用程序位于A页面时，**路由堆栈**中只有A，点击按钮跳转到B页面，路由堆栈中有 B 和 A，且 B 处于栈顶。



点击 B 页面的按钮返回到 A 页面，修改 B 页面按钮点击事件：

```
RaisedButton(
  child: Text('B 页面'),
  onPressed: () {
    Navigator.of(context).pop();
  },
)
```

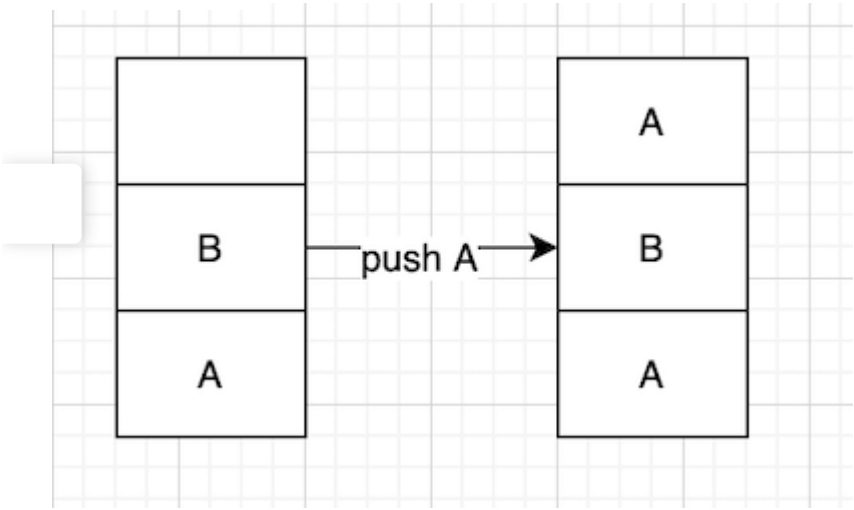
路由堆栈的变化：



上面案例的效果是从 B 页面跳转到 A 页面，那是否也可以使用 push 方法？修改 B 页面按钮点击事件：

```
RaisedButton(
  child: Text('B 页面'),
  onPressed: () {
    Navigator.of(context).push(MaterialPageRoute(builder: (context) {
      return APage();
    }));
  },
)
```

从效果上看也可以跳转到 A 页面，路由堆栈：



那是否可以使用 push 代替 pop 呢? 答案肯定是不可以的,

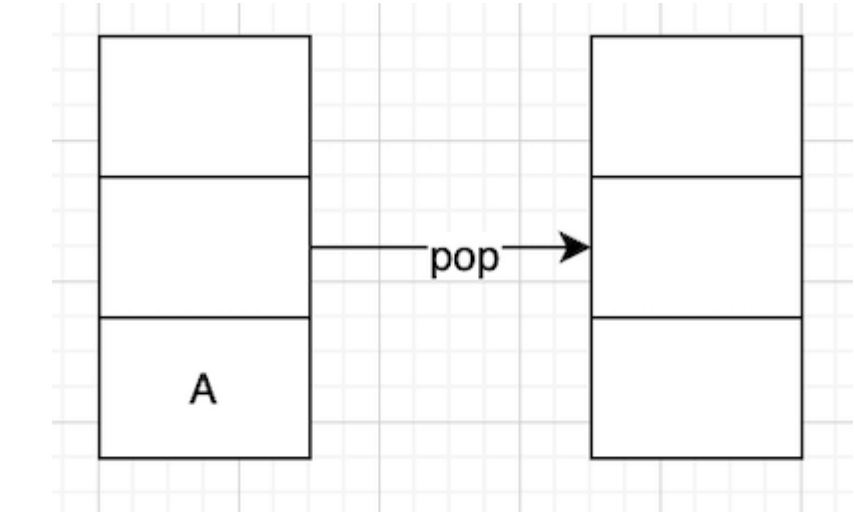
1. 试想如下场景，进入购物App，展示购物列表，点击其中一个进入商品详细页面，使用 push 再次进入购物列表，然后在进入商品详细页面...，如此反复，路由堆栈中将会存放大量的购物列表和商品详细页面的路由，点击返回按钮，会将反复显示购物列表和商品详细页面。
2. 页面切换时路由动画 push 和 pop 是不同。

maybePop 和 canPop

上面案例如果点击 A 页面按钮直接调用 pop 会如何?

```
RaisedButton(
  child: Text('A 页面'),
  onPressed: () {
    Navigator.of(context).pop();
  },
)
```

在 A 页面时路由堆栈中只有 A，调用 pop 后，路由堆栈变化:



此时路由堆栈为空，没有可显示的页面，应用程序将会退出或者黑屏，好的用户体验不应如此，此时可以使用 maybePop，maybePop 只在路由堆栈有可弹出路由时才会弹出路由。

面的案例在 A 页面执行maybePop：

```
RaisedButton(
  child: Text('A 页面'),
  onPressed: () {
    Navigator.of(context).maybePop();
  },
)
```

点击后不会出现弹出路由，因为当前路由堆栈中只有 A，在 B页面执行maybePop，将会返回到 A 页面。

也可以通过 canPop 判断当前是否可以 pop：

```
RaisedButton(
  child: Text('B 页面'),
  onPressed: () {
    if(Navigator.of(context).canPop()){
      Navigator.of(context).pop();
    }
  },
)
```

pushNamed

pushNamed 是命名路由的方式，需要在 MaterialApp 中配置路由名称：

```
MaterialApp(
      title: 'Flutter Demo',
      routes: <String, WidgetBuilder>{
        '/A': (context) => APage(),
        '/B': (context) => BPage(),
      },
      home: Scaffold(
        body: APage(),
      ),
    )
```

从 A 跳转到 B：

```
RaisedButton(
  child: Text('A 页面'),
  onPressed: () {
    Navigator.of(context).pushNamed('/B');
  },
)
```

pushReplacementNamed 和 popAndPushNamed

有A、B、C 三个页面，A页面通过 pushNamed 跳转到 B：

```
RaisedButton(
  child: Text('A 页面'),
  onPressed: () {
    Navigator.of(context).pushNamed('/B');
  },
)
```

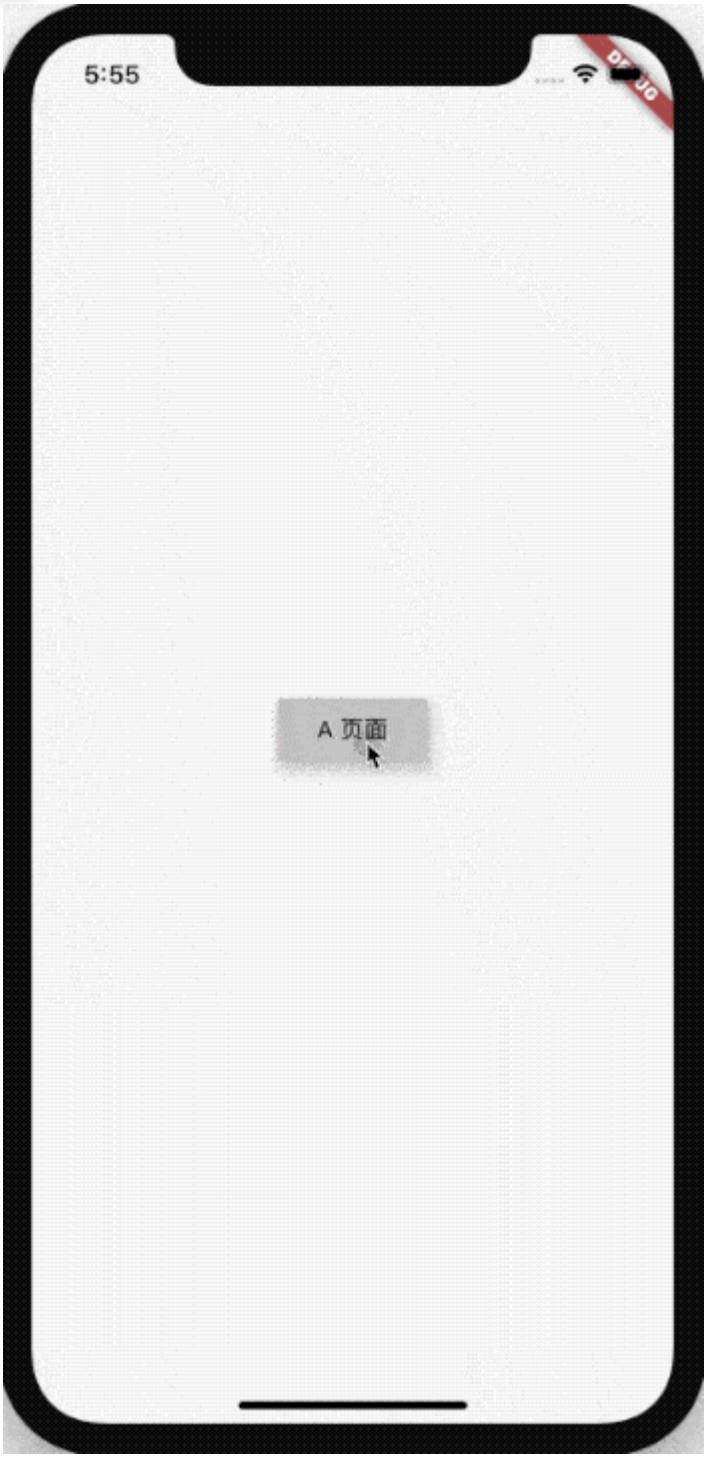
B 通过 pushReplacementNamed 跳转到 C：

```
RaisedButton(
  child: Text('B 页面'),
  onPressed: () {
      Navigator.of(context).pushReplacementNamed('/C');
  },
)
```

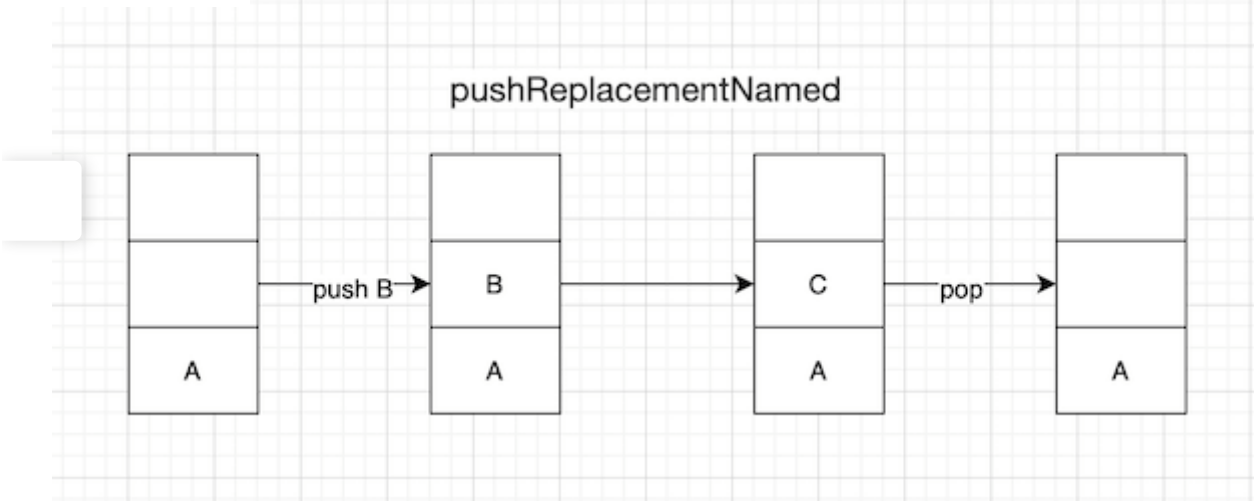
点击 C 页面按钮执行 pop：

```
RaisedButton(
  child: Text('C 页面'),
  onPressed: () {
    if(Navigator.of(context).canPop()){
      Navigator.of(context).pop();
```

```
      }
    },
  )
```



点击 C 页面按钮直接返回到了 A 页面，而不是 B 页面，因为 B 页面使用 pushReplacementNamed 跳转，路由堆栈变化：



B 页面跳转到 C 页面，使用 popAndPushNamed：

```
RaisedButton(
  child: Text('B 页面'),
  onPressed: () {
      Navigator.of(context).popAndPushNamed('/C');
  },
)
```

popAndPushNamed 路由堆栈和 pushReplacementNamed 是一样，唯一的区别就是 popAndPushNamed 有 B 页面退出动画。

popAndPushNamed 和 pushReplacementNamed 使当前页面不在路由堆栈中，所以通过 pop 无法返回此页面。

适用场景：

- **欢迎页面**：应用程序打开后首先进入欢迎界面，然后进入首页，进入首页后不应该再进入欢迎界面。
- **登录页面**：登录成功后进入相关页面，此时按返回按钮，不应再进入登录页面。

pushNamedAndRemoveUntil

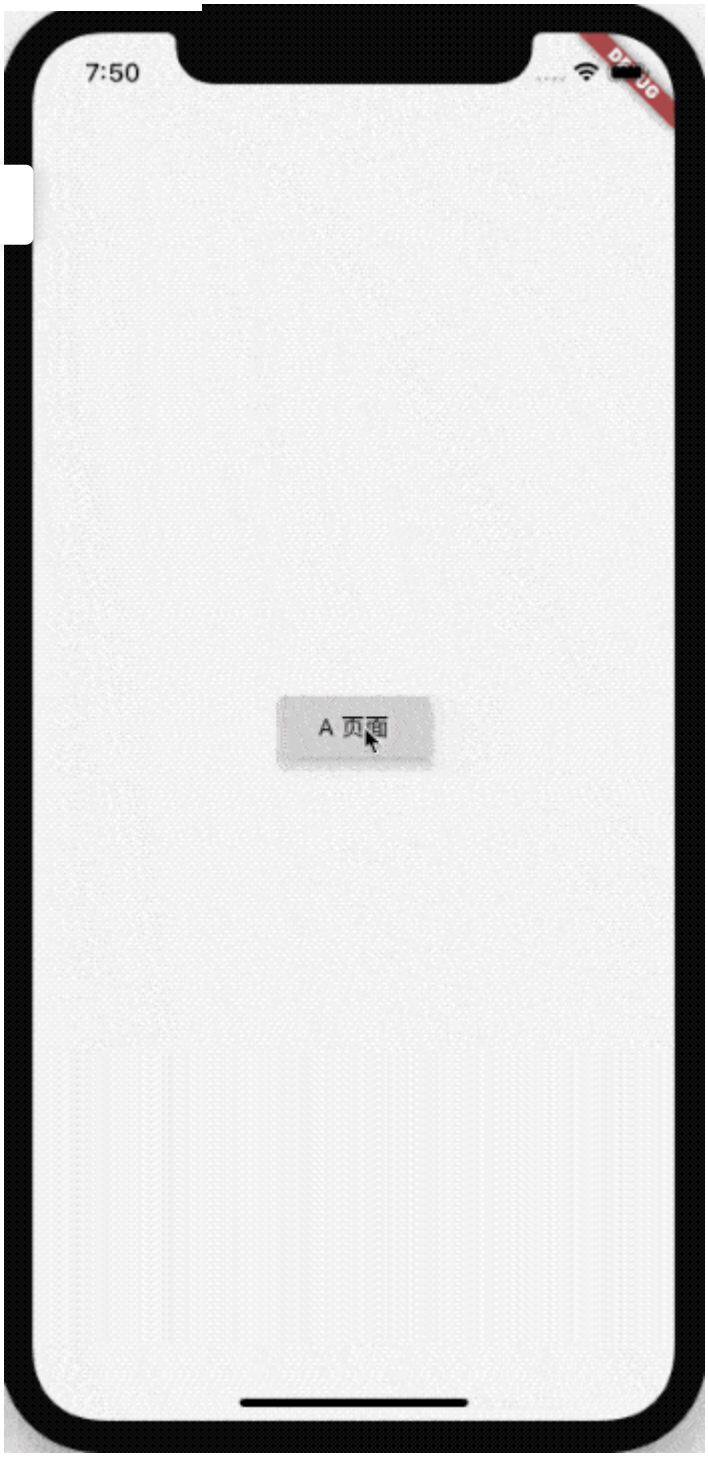
有如下场景，应用程序进入首页，点击登录进入登录页面，然后进入注册页面或者忘记密码页面...，登录成功后进入其他页面，此时不希望返回到登录相关页面，此场景可以使用 pushNamedAndRemoveUntil。

有A、B、C、D 四个页面，A 通过push进入 B 页面，B 通过push进入 C 页面，C 通过 **pushNamedAndRemoveUntil** 进入 D 页面同时删除路由堆栈中直到 /B 的路由，C 页面代码：

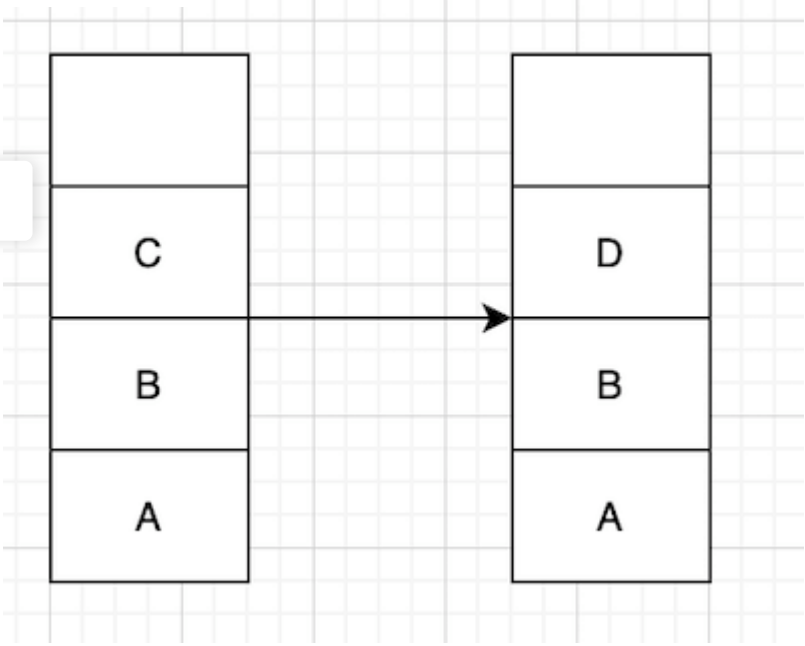
```
RaisedButton(
  child: Text('C 页面'),
  onPressed: () {
    Navigator.of(context).pushNamedAndRemoveUntil('/D', ModalRoute.with
  },
),
```

D 页面按钮执行 pop：

```
RaisedButton(
  child: Text('D 页面'),
  onPressed: () {
    Navigator.of(context).pop();
  },
)
```



从 C 页面跳转到 D 页面路由堆栈变化：

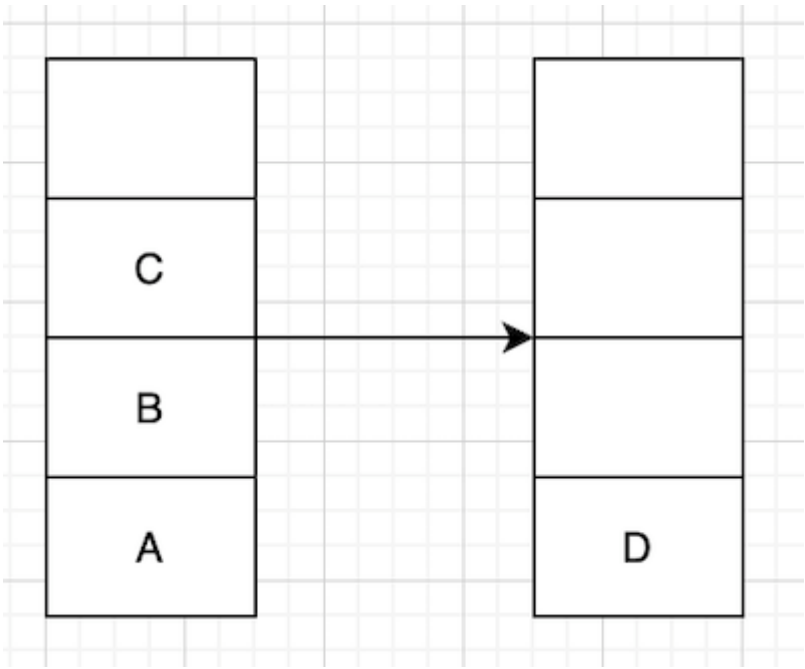



```
Navigator.of(context).pushNamedAndRemoveUntil('/D', ModalRoute.withName
```

表示跳转到 D 页面，同时删除D 到 B 直接所有的路由，如果删除所有路由，只保存 D：

```
Navigator.of(context).pushNamedAndRemoveUntil('/D', (Route route)=>fals
```

路由堆栈变化：

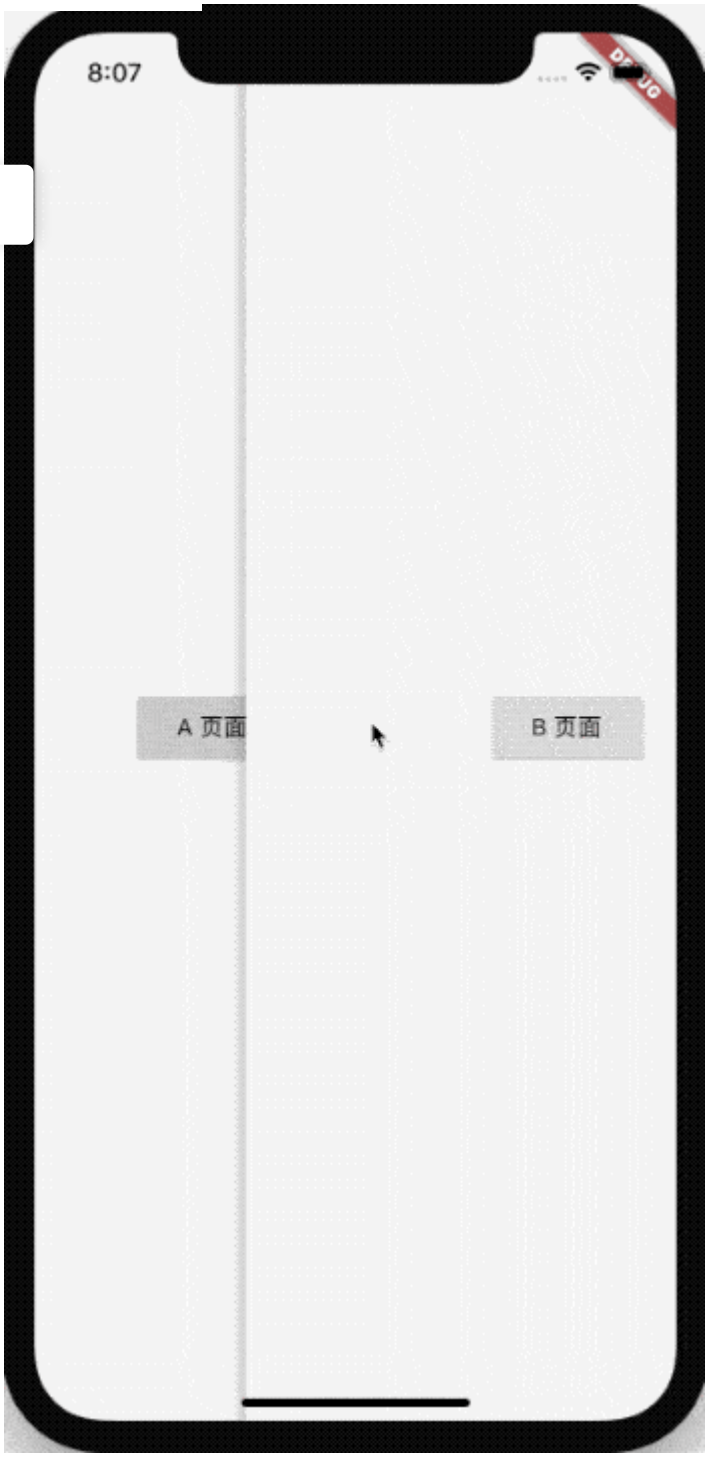


popUntil

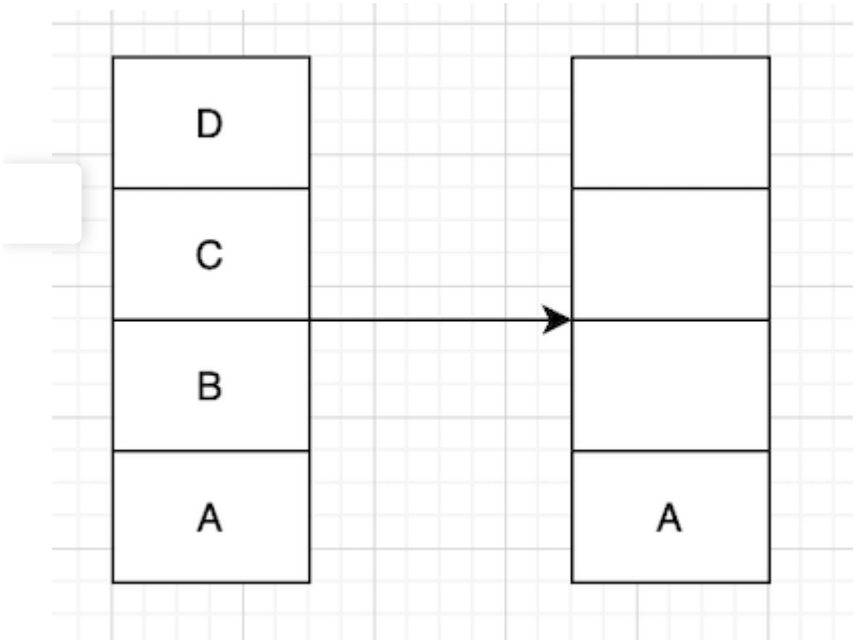
有如下场景，在入职新公司的时候，需要填写各种信息，这些信息分为不同部分，比如基本信息、工作信息、家庭信息等，这些不同模块在不同页面，填写信息时可以返回上一页，也可以取消，取消返回到首页，此场景可以使用 popUntil，一直 pop 到指定的页面。

有A、B、C、D 四个页面，D 页面通过 popUntil 一直返回到 A 页面，D 页面代码：

```
RaisedButton(
  child: Text('D 页面'),
  onPressed: () {
    Navigator.of(context).popUntil(ModalRoute.withName('/A'));
  },
)
```



路由堆栈变化：



传递数据

有如下场景，商品列表页面，点击跳转到商品详情页面，商品详情页面需要商品的唯一id或者商品详情数据，有两种方式传递数据：

第一种：通过构造函数方式：

```
class ProductDetail extends StatelessWidget {
  final ProductInfo productInfo;

  const ProductDetail({Key key, this.productInfo}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

跳转代码：

```
Navigator.of(context).push(MaterialPageRoute(builder: (context){
  return ProductDetail(productInfo: productInfo,);
}));
```

此种方式无法用于命名路由的跳转方式。

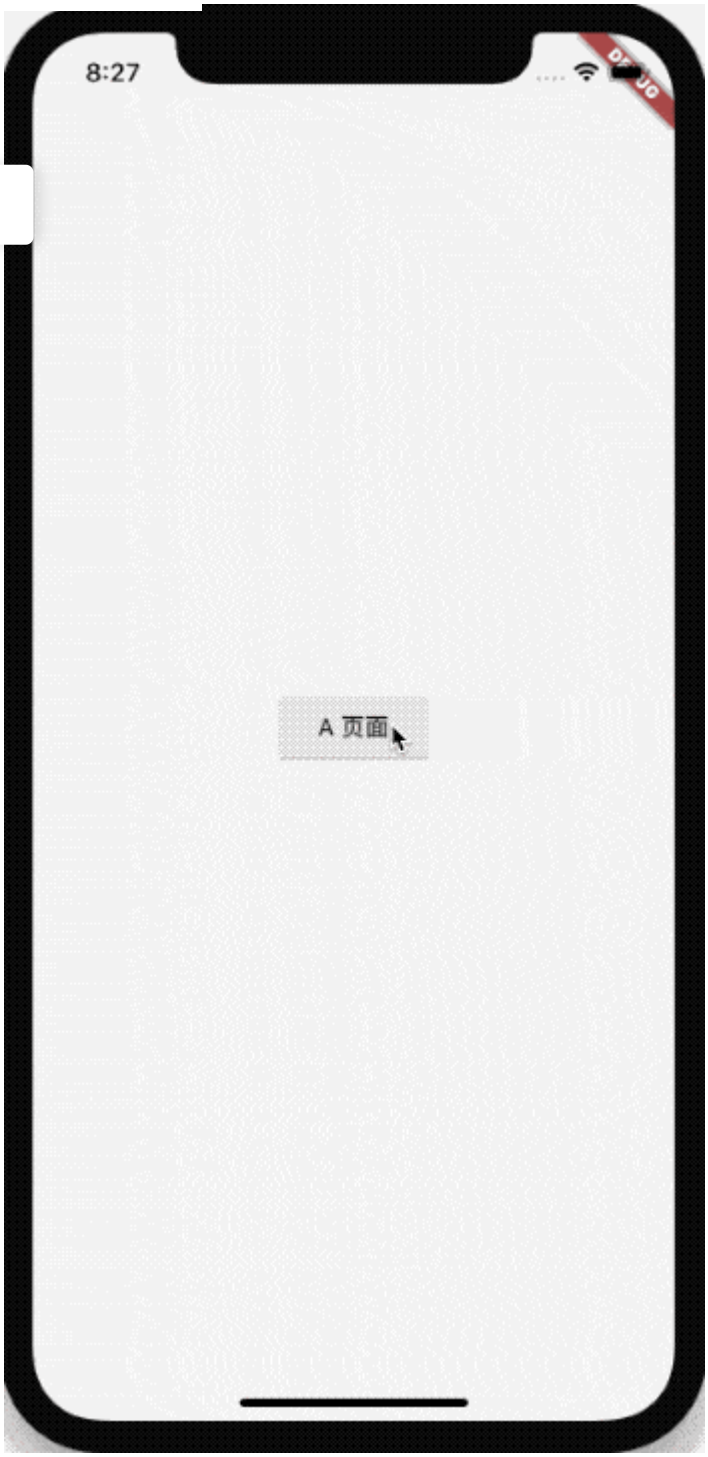
第二种：通过命名路由设置参数的方式：

A 页面传递数据，

```
RaisedButton(
  child: Text('A 页面'),
  onPressed: () {
    Navigator.of(context).pushNamed('/B',arguments: '来自A');
  },
)
```

B 页面通过 `ModalRoute.of(context).settings.arguments` 接收数据：

```
RaisedButton(
  child: Text('${ModalRoute.of(context).settings.arguments}'),
  onPressed: () {
    Navigator.of(context).pushNamed('/C');
  },
)
```



返回数据

B 页面返回代码：

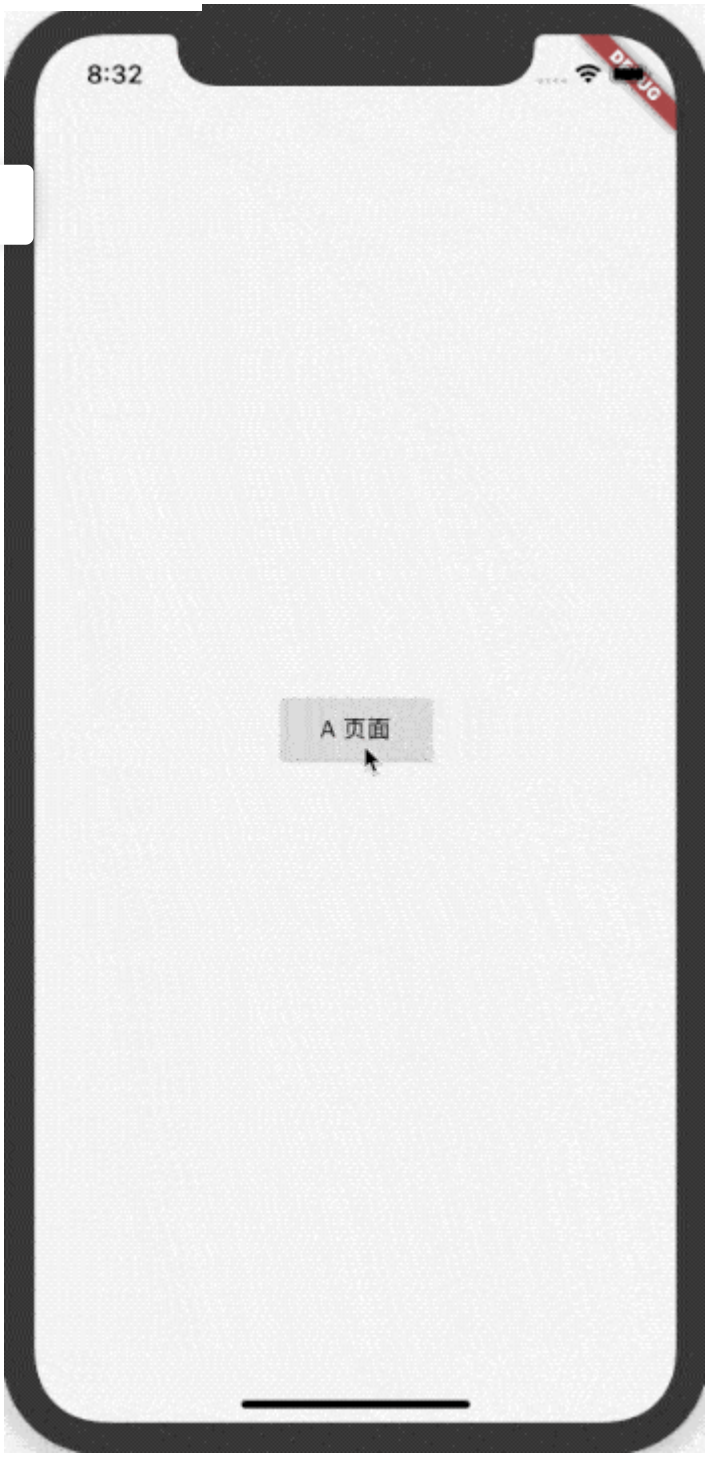
```
RaisedButton(
  child: Text('${ModalRoute.of(context).settings.arguments}'),
  onPressed: () {
    Navigator.of(context).pop('从B返回');
  },
)
```

A 页面接收返回的数据：

```
class APage extends StatefulWidget {
  @override
  _APageState createState() => _APageState();
}

class _APageState extends State<APage> {
  String _string = 'A 页面';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        alignment: Alignment.center,
        child: RaisedButton(
          child: Text(_string),
          onPressed: () async {
            var result =
                await Navigator.of(context).pushNamed('/B', arguments:
            setState(() {
              _string = result;
            });
          },
        ),
      ),
    );
  }
}
```



push 相关方法返回 Future 类型，使用 await 等待返回结果。

喜欢作者

< 12.17 案例-雷达扫描效果

13.2 监听路由堆栈变化 >