How to preserve widget states in flutter, when navigating using BottomNavigationBar?

Asked 3 years, 10 months ago Active 4 months ago Viewed 49k times

88

39

I'm currently working on building a Flutter app that will preserve states when navigating from one screen, to another, and back again when utilizing BottomNavigationBar. Just like it works in the Spotify mobile application; if you have navigated down to a certain level in the navigation hierarchy on one of the main screens, changing screen via the bottom navigation bar, and later changing back to the old screen, will preserve where the user were in that hierarchy, including preservation of the state.

I have run my head agair Please log in or register to bookmark this question. × :cess.

I want to know how I can prevent the pages in `pageChooser()`, when toggled once the user taps the BottomNavigationBar item, from rebuilding themselves, and instead preserve the state they already found themselves in (the pages are all stateful Widgets).

```dart
import 'package:flutter/material.dart';
import './page_plan.dart';
import './page_profile.dart';
import './page_startup_namer.dart';

void main() => runApp(new Recipher());

class Recipher extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new Pages();
  }
}

class Pages extends StatefulWidget {
  @override
  createState() => new PagesState();
}

class PagesState extends State<Pages> {
  int pageIndex = 0;


  pageChooser() {
    switch (this.pageIndex) {
      case 0:
        return new ProfilePage();
        break;

      case 1:
        return new PlanPage();
        break;

      case 2:
        return new StartUpNamerPage();
```

```
          break;

        default:
          return new Container(
            child: new Center(
              child: new Text(
                'No page found by page chooser.',
                style: new TextStyle(fontSize: 30.0)
                )
              ),
            );
      }
    }

    @override
    Widget build(BuildContext context) {
      return new MaterialApp(
        home: new Scaffold(
          body: pageChooser(),
          bottomNavigationBar: new BottomNavigationBar(
            currentIndex: pageIndex,
            onTap: (int tappedIndex) { //Toggle pageChooser and rebuild state with the
  index that was tapped in bottom navbar
              setState(
                (){ this.pageIndex = tappedIndex; }
                );
              },
            items: <BottomNavigationBarItem>[
              new BottomNavigationBarItem(
                title: new Text('Profile'),
                icon: new Icon(Icons.account_box)
                ),
                new BottomNavigationBarItem(
                  title: new Text('Plan'),
                  icon: new Icon(Icons.calendar_today)
                ),
                  new BottomNavigationBarItem(
                  title: new Text('Startup'),
                  icon: new Icon(Icons.alarm_on)
                )
              ],
            )
        )
      );
    }
  }
```

dart flutter

Share Follow

edited Mar 22 '18 at 22:02

asked Mar 22 '18 at 21:51

ArtBelch
889 1 6 4

keep each pages using Stack. This might be helpful: stackoverflow.com/questions/45235570/... – najeira
Mar 23 '18 at 2:47

Nice, thank you. This worked. I actually already tried to use a IndexedStack, but I now see that this does not work the same way as Offstage and Tickermode. – ArtBelch Mar 24 '18 at 13:19

8 Answers

Active Oldest Votes

93

For keeping state in `BottomNavigationBar`, you can use `IndexedStack`

```
@override
  Widget build(BuildContext context) {
    return Scaffold(
      bottomNavigationBar: BottomNavigationBar(
        onTap: (index) {
          setState(() {
            current_tab = index;
          });
        },
        currentIndex: current_tab,
        items: [
          BottomNavigationBarItem(
            ...
          ),
          BottomNavigationBarItem(
            ...
          ),
        ],
      ),
      body: IndexedStack(
        children: <Widget>[
          PageOne(),
          PageTwo(),
        ],
        index: current_tab,
      ),
    );
  }
```

Share Follow

answered Mar 20 '20 at 13:50

Newaj
2,775 2 20 39

24 But it has performance issue. IndexedStack loads all tab initially which is unnecessary. – Jaya Prakash Aug 11 '20 at 13:38

2 This should be the accepted answer. Works great. @ArtBelch: You'll do us all a favor by marking this answer as the correct one. :) – Teekin Sep 23 '20 at 10:07

2 @JayaPrakash Did you find a soltion for not building all stacks at once? – anoop4real Oct 5 '20 at 22:13

1 @anoop4real I am using `IndexedStack` to keep preserve the all widget state of tabs and i keep reference for each tab child reference. with on tab change callback i used to call any method inside the child widget with the help of tab child reference. If you need my answer then post a question, i will answer for that. – Jaya Prakash Oct 6 '20 at 15:08

1 @JayaPrakash Regarding performance, take a look at this solution: stackoverflow.com/questions/66404759/... – filipetrm Feb 28 '21 at 0:36

43

Late to the party, but I've got a simple solution. Use the `PageView` widget with the `AutomaticKeepAliveClinetMixin` .

The beauty of it that it doesn't load any tab until you click on it.

The page that includes the `BottomNavigationBar` :

```
var _selectedPageIndex;
List<Widget> _pages;
PageController _pageController;

@override
void initState() {
  super.initState();

  _selectedPageIndex = 0;
  _pages = [
    //The individual tabs.
  ];

  _pageController = PageController(initialPage: _selectedPageIndex);
}

@override
void dispose() {
  _pageController.dispose();

  super.dispose();
}

@override
Widget build(BuildContext context) {
  ...
    body: PageView(
      controller: _pageController,
      physics: NeverScrollableScrollPhysics(),
      children: _pages,
    ),
   bottomNavigationBar: BottomNavigationBar(
      ...
      currentIndex: _selectedPageIndex,
      onTap: (selectedPageIndex) {
        setState(() {
          _selectedPageIndex = selectedPageIndex;
          _pageController.jumpToPage(selectedPageIndex);
        });
      },
  ...
}
```

The individual tab:

```
class _HomeState extends State<Home> with AutomaticKeepAliveClientMixin<Home> {
  @override
  bool get wantKeepAlive => true;

  @override
  Widget build(BuildContext context) {
    //Notice the super-call here.
    super.build(context);
    ...
  }
}
```

I've made a video about it here.

Share Follow edited Nov 14 '20 at 12:12 answered Sep 25 '20 at 3:03

Tayan
951 9 17

2 Works really great thanks ! All _pages are not loaded at first like IndexedStack, but just saved. – Dimitri Leurs Nov 30 '20 at 12:24

1 Works perfectly... Each page loads separately and maintains state. – Lefty Dec 14 '20 at 8:56

Neat 🤠 I was looking for this! Hope that the bottom navigation bar is not redrawn every time. – Anoop Thiruonam Apr 3 '21 at 20:20

Best solution so far, thanks! IndexedStack loads everything at launch. It just loads as soon as you navigate to the pages and it just keeps states of the pages that have `wantKeepAlive=true` . Very useful if you just want to keep state of some of them but not all of them. – JFValdes Aug 6 '21 at 7:33

41

Use AutomaticKeepAliveClientMixin to force your tab content to not be disposed.

```
class PersistantTab extends StatefulWidget {
  @override
  _PersistantTabState createState() => _PersistantTabState();
}

class _PersistantTabState extends State<PersistantTab> with
AutomaticKeepAliveClientMixin {
  @override
  Widget build(BuildContext context) {
    return Container();
  }

  // Setting to true will force the tab to never be disposed. This could be dangerous.
  @override
  bool get wantKeepAlive => true;
}
```

To make sure your tab does get disposed when it doesn't require to be persisted, make `wantKeepAlive` return a class variable. You must call `updateKeepAlive()` to update the keep alive status.

Example with dynamic keep alive:

```
// class PersistantTab extends StatefulWidget ...

class _PersistantTabState extends State<PersistantTab>
    with AutomaticKeepAliveClientMixin {
  bool keepAlive = false;

  @override
  void initState() {
    doAsyncStuff();
  }

  Future doAsyncStuff() async {
    keepAlive = true;
    updateKeepAlive();
    // Keeping alive...

    await Future.delayed(Duration(seconds: 10));

    keepAlive = false;
    updateKeepAlive();
    // Can be disposed whenever now.
  }

  @override
  bool get wantKeepAlive => keepAlive;

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

Share Follow

edited Jan 6 '20 at 0:04

answered Aug 8 '18 at 3:35
Charles Crete
866 7 14

to whom we should conform with `AutomaticKeepAliveClientMixin` the parent widget or the widget that will be shown when button is pressed? – Anirudha Mahale Apr 7 '19 at 14:53

7 From the docs: `their build methods must call super.build`, which is missing in your answer. – Michel Feinstein Jan 5 '20 at 15:36

15

Instead of returning new instance every time you run `pageChooser`, have one instance created and return the same.

Example:

```
class Pages extends StatefulWidget {
  @override
  createState() => new PagesState();
}

class PagesState extends State<Pages> {
  int pageIndex = 0;

  // Create all the pages once and return same instance when required
  final ProfilePage _profilePage = new ProfilePage();
  final PlanPage _planPage = new PlanPage();
  final StartUpNamerPage _startUpNamerPage = new StartUpNamerPage();


  Widget pageChooser() {
    switch (this.pageIndex) {
      case 0:
        return _profilePage;
        break;

      case 1:
        return _planPage;
        break;

      case 2:
        return _startUpNamerPage;
        break;

      default:
        return new Container(
          child: new Center(
              child: new Text(
                  'No page found by page chooser.',
                  style: new TextStyle(fontSize: 30.0)
              )
          ),
        );
    }
  }

  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
        home: new Scaffold(
            body: pageChooser(),
            bottomNavigationBar: new BottomNavigationBar(
              currentIndex: pageIndex,
              onTap: (int tappedIndex) { //Toggle pageChooser and rebuild state with
the index that was tapped in bottom navbar
                setState(
                        (){ this.pageIndex = tappedIndex; }
                );
              },
              items: <BottomNavigationBarItem>[
                new BottomNavigationBarItem(
                    title: new Text('Profile'),
                    icon: new Icon(Icons.account_box)
                ),
                new BottomNavigationBarItem(
                    title: new Text('Plan'),
                    icon: new Icon(Icons.calendar_today)
                  ),
                  new BottomNavigationBarItem(
                      title: new Text('Startup'),
                      icon: new Icon(Icons.alarm_on)
                  )
                ],
              )
          )
      );
    }
  }
```

Or you can make use of widgets like `PageView` or `Stack` to achieve the same.

Hope that helps!

Share Follow

answered Mar 23 '18 at 5:37
Hemanth Raj
27.5k 7 83 75

2 Thank you for your answer. I solved the problem using a Stack combined with Offstage and Tickermode. :-)) – ArtBelch Mar 24 '18 at 13:20

@ArtBelch, if the answer satisfy your question, please mark it as correct :) – Banana droid Jul 28 '18 at 9:27

4 @ArtBelch could you share your solution please? – NullPointer Oct 1 '18 at 20:42

7

Use “**IndexedStack Widget**” with “**Bottom Navigation Bar Widget**” to keep state of Screens/pages/Widget

Provide list of Widget to **IndexedStack** and index of widget you want to show because **IndexedStack** show single widget from list at one time.

```
final List<Widget> _children = [
    FirstClass(),
    SecondClass()
  ];

Scaffold(
  body: IndexedStack(
    index: _selectedPage,
    children: _children,
  ),
  bottomNavigationBar: BottomNavigationBar(
    ........
    ........
  ),
);
```

Share Follow

answered Jul 24 '20 at 16:49
Techalgoware

316 4 10

5

The most convenient way I have found to do so is using PageStorage widget along with PageStorageBucket, which acts as a key value persistent layer.

Go through this article for a beautiful explanation -> https://steemit.com/utopian-io/@tensor/persisting-user-interface-state-and-building-bottom-navigation-bars-in-dart-s-flutter-framework

Share Follow

answered Jun 8 '19 at 7:19
Anirudh Agarwal
1,748 2 13 8

In this case, my pages are still loading data – Wikki Aug 31 '19 at 12:51

1

proper way of preserving tabs state in bottom nav bar is by wrapping the whole tree with `PageStorage()` widget which takes a `PageStorageBucket bucket` as a required named parameter and for those tabs to which you want to preserve its state pas those respected widgets with `PageStorageKey(<str_key>)` then you are done !! you can see more details in this ans which i've answered few weeks back on one question : https://stackoverflow.com/a/68620032/11974847

there's other alternatives like `IndexedWidget()` but you should beware while using it , i've explained y we should be catious while using `IndexedWidget()` in the given link answer

good luck mate ..

Share Follow

answered Sep 23 '21 at 15:42
Himmat rai
49 1 4

0

Do not use IndexStack Widget, because it will instantiate all the tabs together, and suppose if all the tabs are making a network request then the callbacks will be messed up the last API calling tab will probably have the control of the callback.

Use AutomaticKeepAliveClientMixin for your stateful widget it is the simplest way to achieve it without instantiating all the tabs together.

My code had interfaces that were providing the respective responses to the calling tab I implemented it the following way.

Create your stateful widget

```
class FollowUpsScreen extends StatefulWidget {
  FollowUpsScreen();

  @override
  State<StatefulWidget> createState() {
    return FollowUpsScreenState();
  }
}

class FollowUpsScreenState extends State<FollowUpsScreen>
     with AutomaticKeepAliveClientMixin<FollowUpsScreen>
            implements OperationalControls {

  @override
  Widget build(BuildContext context) {
  //do not miss this line
    super.build(context);
    return .....;
  }

  @override
  bool get wantKeepAlive => true;
}
```

Share Follow

edited Jul 28 '21 at 10:02
MahMoos
755 5 13

answered Jul 19 '21 at 19:28
Swapnil Kadam
3,689 4 28 33