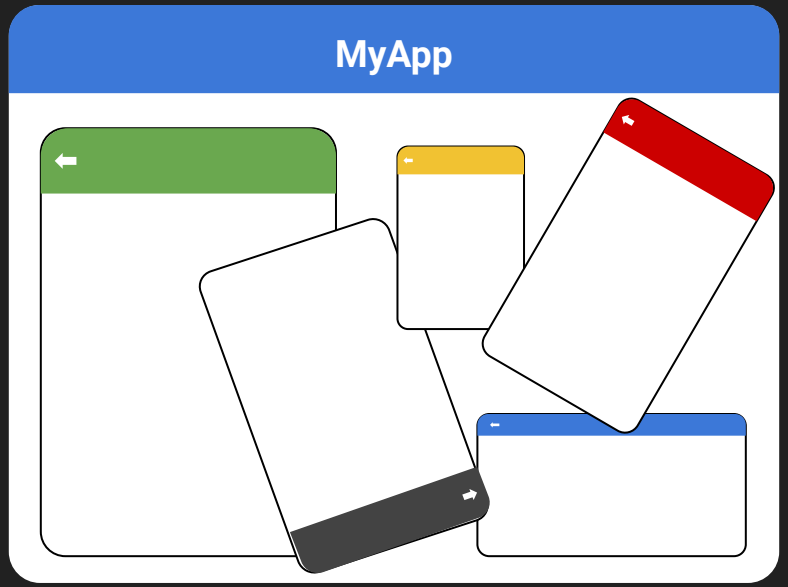
Nested Navigation in Flutter (+ Beamer)



@slovnicki

About me

- Sandro Lovnički, **@slovnicki**
- Flutterer at **Friendly Fire**
- Organizer at **GDG Zagreb** and **Flutter Croatia**
- Creator of **Beamer**
- Co-host of **#FlutterDartist**

- GitHub: https://github.com/slovnicki
- Twitter: https://twitter.com/slovnicki
- LinkedIn: https://www.linkedin.com/in/slovnicki
- Medium: https://medium.com/@slovnicki

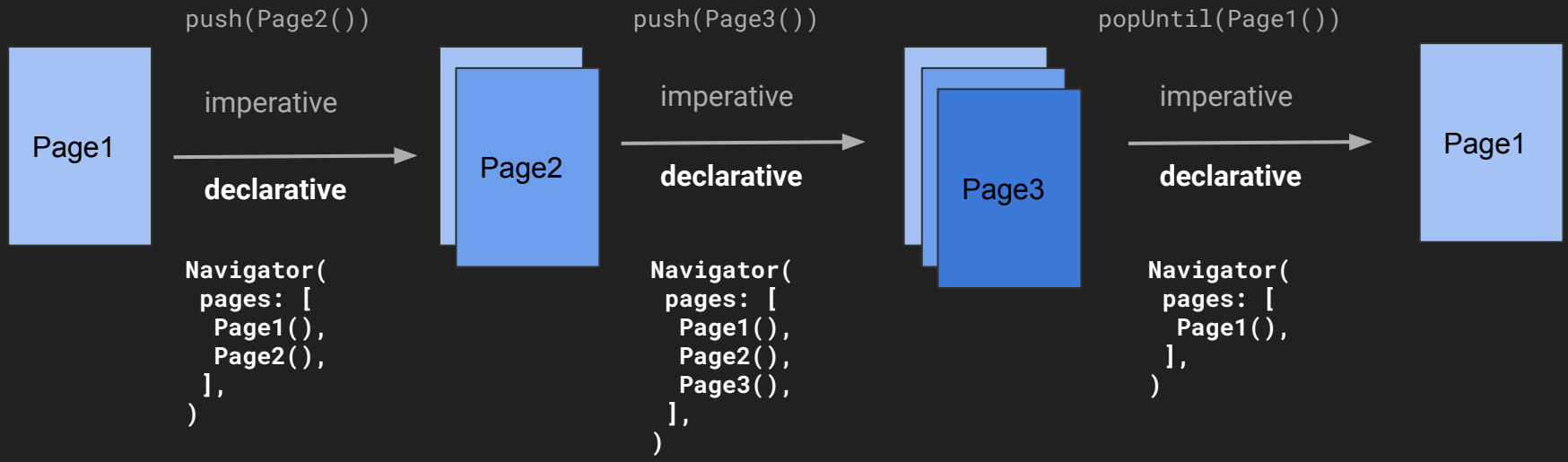
Table of Contents

Navigator

- A **Widget** that holds and manages a list of pages currently displayed (**stacked**) on the screen; https://api.flutter.dev/flutter/widgets/Navigator-class.html

- Supports 2 different APIs; imperative and declarative
  - **Imperative**: Navigator.push, Navigator.pop, etc.
  - **Declarative**: rebuilding Navigator with a new List of pages

Navigator

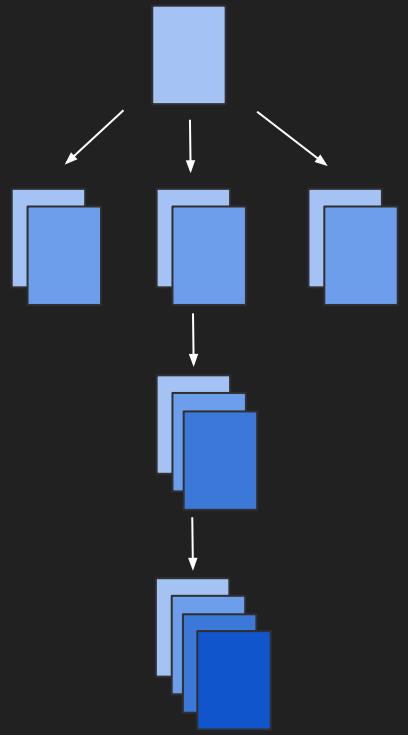


“Every navigation event is deep-linking”

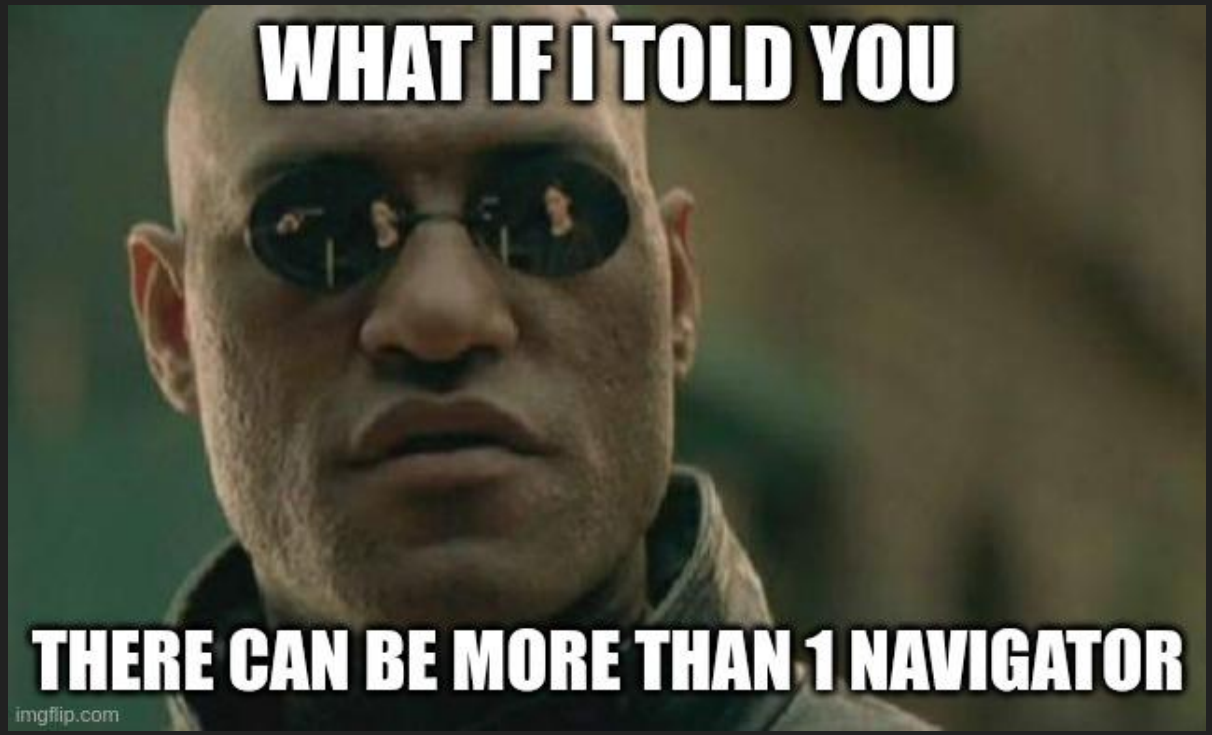
Navigator

Navigator controls **1 stack** of pages.

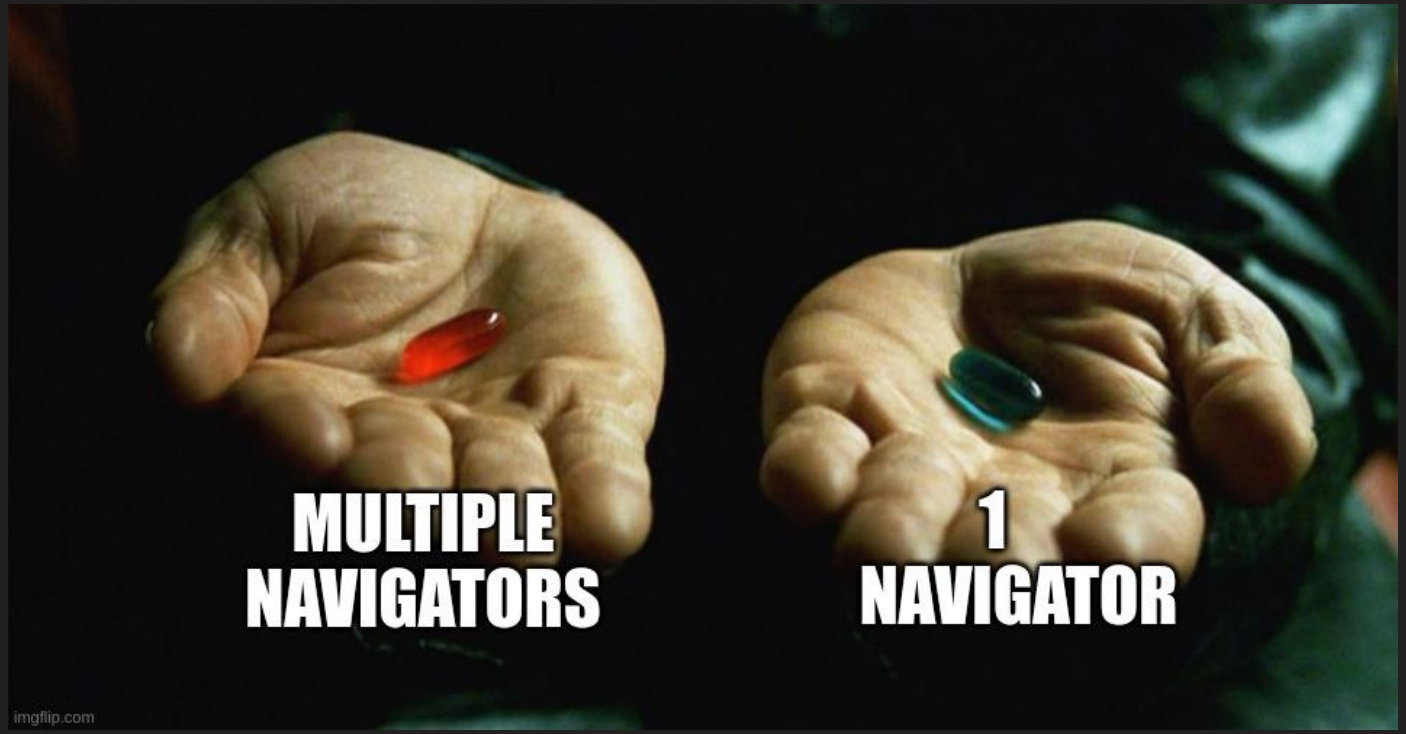
Is this enough for **all** application?



Navigator

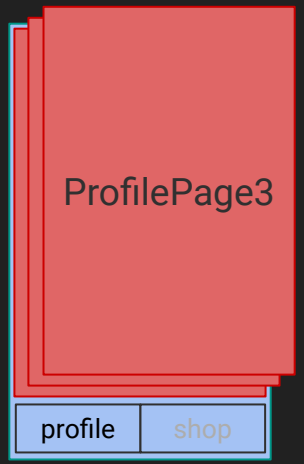

WHAT IF I TOLD YOU
THERE CAN BE MORE THAN 1 NAVIGATOR
imgflip.com

Navigator



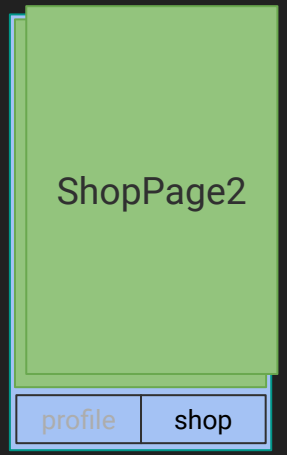
Multiple Navigators

- Application with BottomNavigationBar where each tab has an independent stack of pages, e.g. Twitter.



Multiple Navigators

- Application with BottomNavigationBar where each tab has an independent stack of pages, e.g. Twitter.



Multiple Navigators

How?

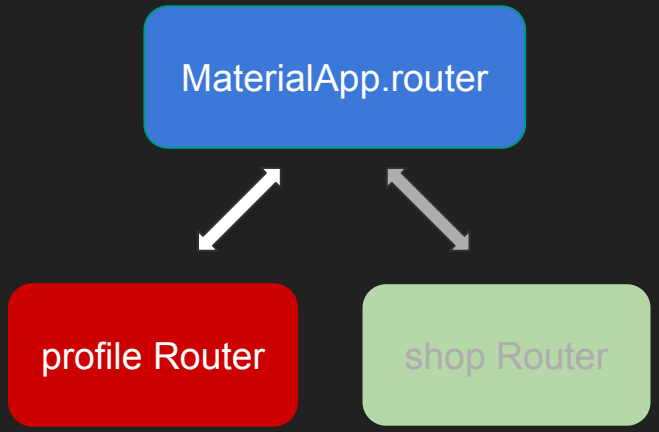
Let’s first see how can we build a Navigator…

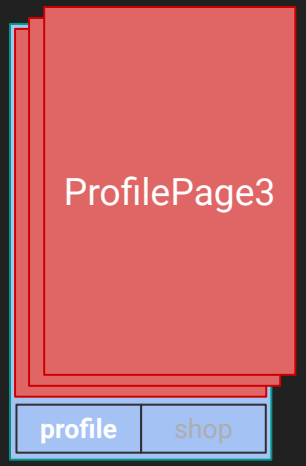
One way is to (re)build a Navigator widget directly, with setState in a custom StatefulWidget, but this will lack web functionalities. If we want to have

- Uniform representation on all platforms
- Support for web URLs

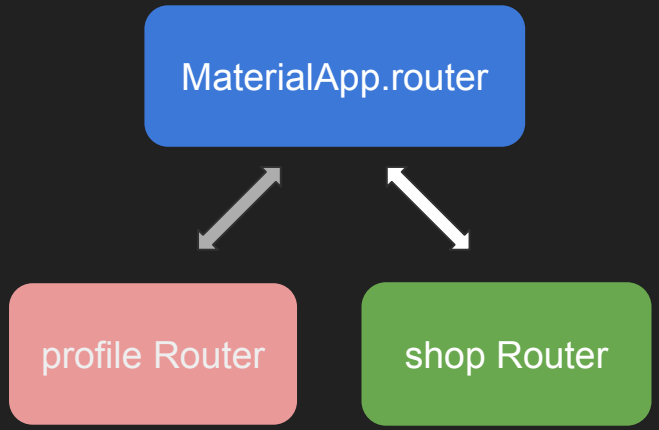
We need to manage the **Navigator** through **Router**.

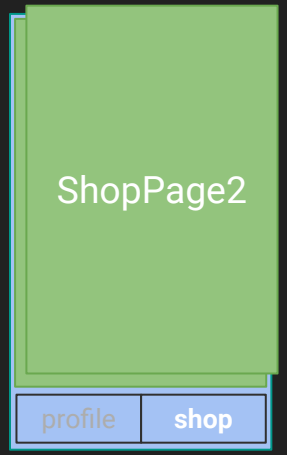
Multiple Navigators





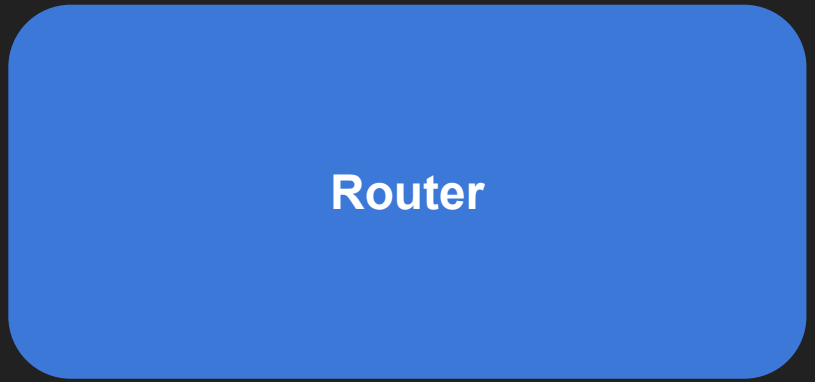
Multiple Navigators





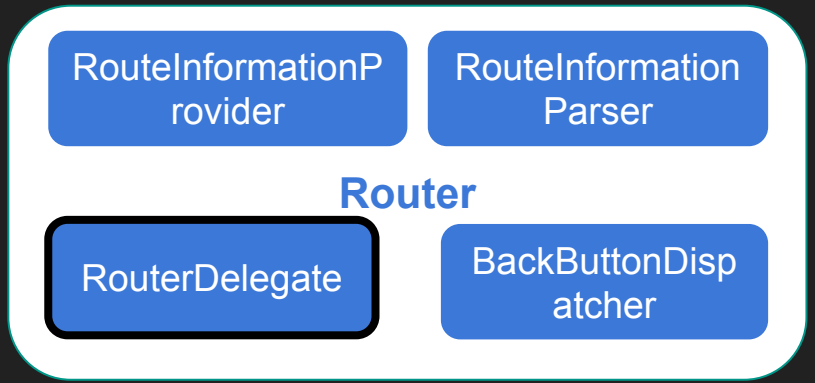
Router

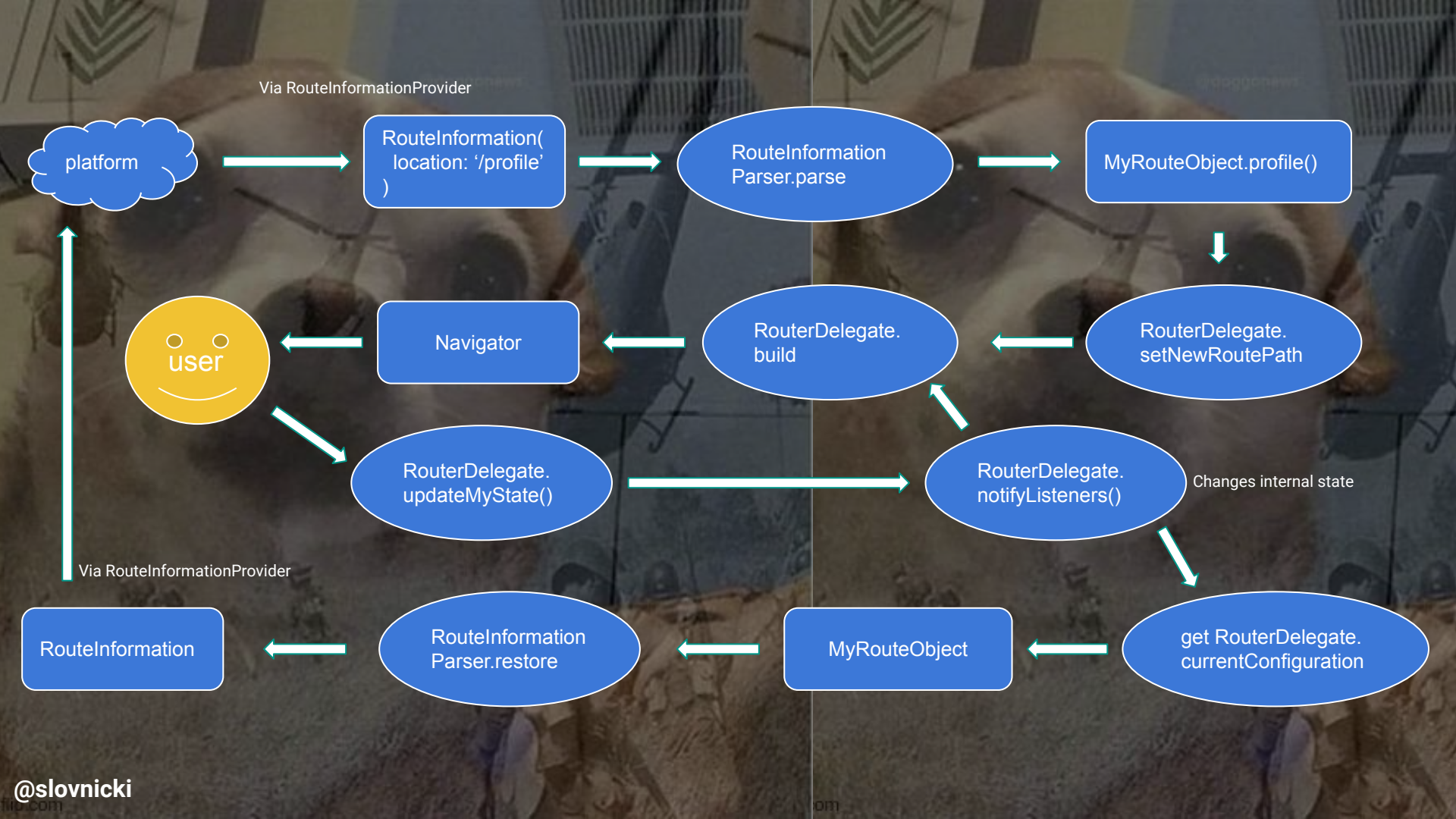
- https://api.flutter.dev/flutter/widgets/Router-class.html
- A **Widget** that uses its components to
  - Get a piece of routing information from and to platform (**RouteInformationProvider**, usually *PlatformRouteInformationProvider*)
  - Parse the routing information into an object that it wants to work with internally (**RouteInformationParser**)
  - Build a Navigator widget and notifies of any changes in routing (**RouterDelegate**)
  - Handle the (Android) back button (**BackButtonDispatcher**)



Router

- https://api.flutter.dev/flutter/widgets/Router-class.html
- A **Widget** that uses its components to
  - Get a piece of routing information from and to platform (**RouteInformationProvider**, usually *PlatformRouteInformationProvider*)
  - Parse the routing information into an object that it wants to work with internally (**RouteInformationParser**)
  - Build a Navigator widget and notifies of any changes in routing (**RouterDelegate**)
  - Handle the (Android) back button (**BackButtonDispatcher**)




Via RouteInformationProvider
platform
RouteInformation(
location: '/profile'
)
RouteInformation
Parser.parse
MyRouteObject.profile()
user
Navigator
RouterDelegate.
build
RouterDelegate.
setNewRoutePath
RouterDelegate.
updateMyState()
RouterDelegate.
notifyListeners()
Changes internal state
Via RouteInformationProvider
RouteInformation
RouteInformation
Parser.restore
MyRouteObject
get RouterDelegate.
currentConfiguration

@slovnicki

Router

How to use it? (**in short**)

Router

How to use it? (**in short**)

- MaterialApp.router

Router

How to use it? (**in short**)

- MaterialApp.router
- Decide how to represent the routing state

Router

How to use it? (**in short**)

- MaterialApp.router
- Decide how to represent the routing state
- Implement RouteInformationParser

Router

How to use it? (**in short**)

- MaterialApp.router
- Decide how to represent the routing state
- Implement RouteInformationParser
    - Override parseRouteInformation

Router

How to use it? (**in short**)

- MaterialApp.router
- Decide how to represent the routing state
- Implement RouteInformationParser
  - Override parseRouteInformation
  - Override restoreRouteInformation

Router

How to use it? (**in short**)

- MaterialApp.router
- Decide how to represent the routing state
- Implement RouteInformationParser
  - Override parseRouteInformation
  - Override restoreRouteInformation
- Implement RouterDelegate

Router

How to use it? (**in short**)

- MaterialApp.router
- Decide how to represent the routing state
- Implement RouteInformationParser
  - Override parseRouteInformation
  - Override restoreRouteInformation
- Implement RouterDelegate
  - Override setNewRoutePath

Router

How to use it? (**in short**)

- MaterialApp.router
- Decide how to represent the routing state
- Implement RouteInformationParser
  - Override parseRouteInformation
  - Override restoreRouteInformation
- Implement RouterDelegate
  - Override setNewRoutePath
  - Override build

Router

How to use it? (**in short**)

- MaterialApp.router
- Decide how to represent the routing state
- Implement RouteInformationParser
  - Override parseRouteInformation
  - Override restoreRouteInformation
- Implement RouterDelegate
  - Override setNewRoutePath
  - Override build; return a Navigator populated with a list of pages reflecting the routing state **and** make sure to update routing state in Navigator.onPopPage

Router

How to use it? (**in short**)

- MaterialApp.router
- Decide how to represent the routing state
- Implement RouteInformationParser
  - Override parseRouteInformation
  - Override restoreRouteInformation
- Implement RouterDelegate
  - Override setNewRoutePath
  - Override build; return a Navigator populated with a list of pages reflecting the routing state **and** make sure to update routing state in Navigator.onPopPage
  - Override currentConfiguration

Router

How to use it? (**in short**)

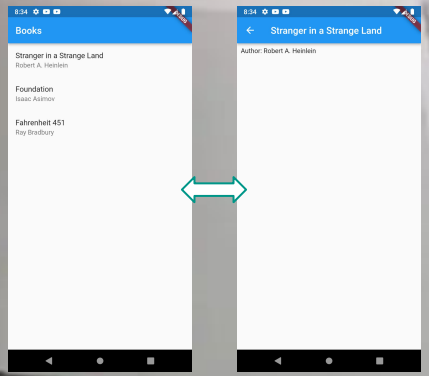
- MaterialApp.router
- Decide how to represent the routing state
- Implement RouteInformationParser
  - Override parseRouteInformation
  - Override restoreRouteInformation
- Implement RouterDelegate
  - Override setNewRoutePath
  - Override build; return a Navigator populated with a list of pages reflecting the routing state **and** make sure to update routing state in Navigator.onPopPage
  - Override currentConfiguration
- Optionally override BackButtonDispatcher

Router

How to use it? (**in short**)

- MaterialApp.router
- Decide how to represent the routing state
- Implement RouteInformationParser
  - Override parseRouteInformation
  - Override restoreRouteInformation
- Implement RouterDelegate
  - Override setNewRoutePath
  - Override build; return a Navigator h a l
    of pages reflecting the routing s ure t
    update routing state in Navigat
  - Override currentConfiguration
- Optionally override BackButtonD

Router


8:34
Books
Stranger in a Strange Land
Robert A. Heinlein
Foundation
Isaac Asimov
Fahrenheit 451
Ray Bradbury
8:34
Stranger in a Strange Land
Author: Robert A. Heinlein

Beamer

beamer 1.5.3

Published 5 months ago • beamer.dev Dart 3 ready

SDK FLUTTER PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS 978

Readme Changelog Example Installing Versions Scores Admin Activity log

beamer.dev

pub v1.5.3 codecov 97% Awesome Flutter

commits 0/month stars 502 forks 123

closed issues 412 closed pull requests 102

contributors 34 chat 36 online

Buy me a coffee

Beamer uses the power of Router and implements all the underlying logic for you, letting you explore arbitrarily complex navigation scenarios with ease.

Flutter Favorite

978 LIKES 140 PUB POINTS 97% POPULARITY

Publisher

beamer.dev

Metadata

A routing package built on top of Router and Navigator's pages API, supporting arbitrary nested navigation, guards and more.

Homepage

Repository (GitHub)

Documentation

API reference

License

MIT (LICENSE)

Dependencies

@slovnicki

Beamer

- A routing package built on top of **Router** and **Navigator's pages API**, supporting arbitrary nested navigation, guards and more.

- Beamer uses the power of Router and implements all the underlying logic for you, letting you explore **arbitrarily complex navigation scenarios** with ease.

- **Beamer** Widget builds a **Router** Widget, which in turn is responsible for building a **Navigator** via its RouterDelegate

- **1 Beamer = 1 Router = 1 Navigator**

Beamer

RoutesLocationBuilder

```
locationBuilder: RoutesLocationBuilder(
  routes: {
    '/': (_, __, ___) => const HomeScreen(),
    '/books': (_, __, ___) => const BooksScreen(),
    '/books/:bookId': (_, state, __) => BookDetailsScreen(
          book: findBook(state.pathParameters['bookId']),
        ), // BookDetailsScreen
  },
), // RoutesLocationBuilder
```

Custom BeamLocations

```
class BooksLocation extends BeamLocation<BeamState> {
  @override
  List<Pattern> get pathPatterns => ['/books/:bookId'];

  @override
  List<BeamPage> buildPages(BuildContext context, BeamState state) {
    final pages = [
      const BeamPage(
        key: ValueKey('home'),
        title: 'Home',
        child: HomeScreen(),
      ), // BeamPage
      if (state.uri.pathSegments.contains('books'))
        const BeamPage(
          key: ValueKey('books'),
          title: 'Books',
          child: BooksScreen(),
        ), // BeamPage
    ];
    final String? bookIdParameter = state.pathParameters['bookId'];
    if (bookIdParameter != null) {
      final bookId = int.tryParse(bookIdParameter);
      final book = books.firstWhereOrNull((book) => book.id == bookId);
      pages.add(
        BeamPage(
          key: ValueKey('book-$bookIdParameter'),
          title: 'Book #$bookIdParameter',
          child: BookDetailsScreen(book: book),
        ), // BeamPage
      );
    }
    return pages;
  }
}
```

Beamer

RoutesLocationBuilder

```
final beamerDelegate = BeamerDelegate(
  locationBuilder: RoutesLocationBuilder(
    routes: {
      '/': (_, __, ___) => const HomeScreen(),
      '/books': (_, __, ___) => const BooksScreen(),
      '/books/:bookId': (_, state, __) => BookDetailsScreen(
            book: findBook(state.pathParameters['bookId']),
          ), // BookDetailsScreen
    },
  ), // RoutesLocationBuilder
); // BeamerDelegate
```

Custom BeamLocations

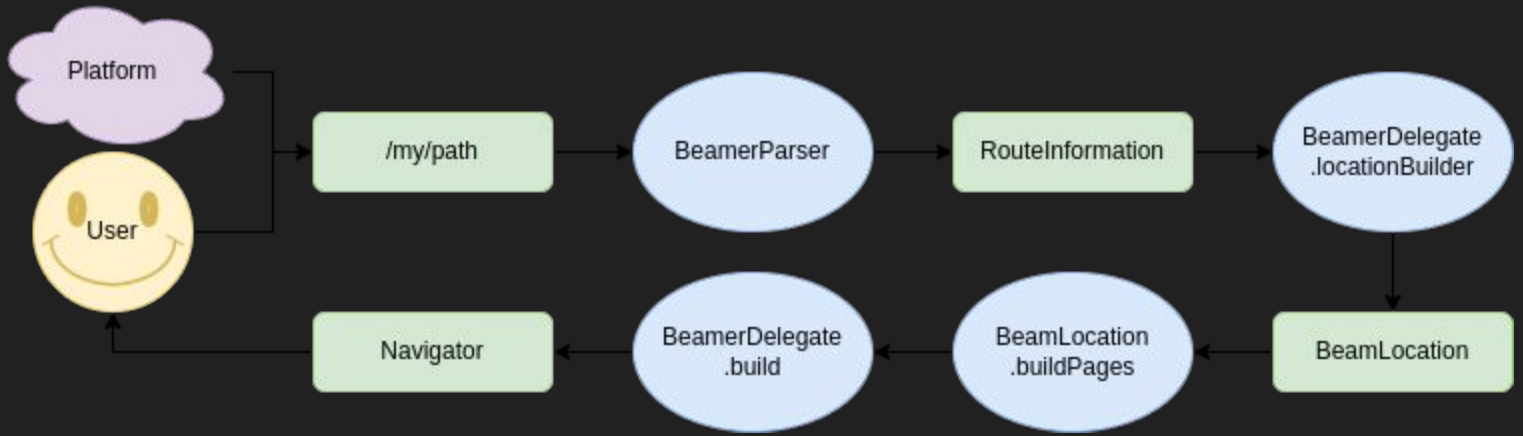
```
final beamerDelegate = BeamerDelegate(
  locationBuilder: (routeInformation, _) {
    if (routeInformation.location!.contains('books')) {
      return BooksLocation(routeInformation);
    }
    if (routeInformation.location!.contains('articles')) {
      return ArticlesLocation(routeInformation);
    }
    return NotFound(path: routeInformation.location!);
  },
); // BeamerDelegate
```

Beamer

```dart
@override
Widget build(BuildContext context) {
  return MaterialApp.router(
    routerDelegate: beamerDelegate,
    routeInformationParser: BeamerParser(),
    backButtonDispatcher: BeamerBackButtonDispatcher(
      delegate: beamerDelegate,
    ), // BeamerBackButtonDispatcher
  ); // MaterialApp.router
}
```

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: IndexedStack(
      index: currentIndex,
      children: [
        Beamer(
          routerDelegate: beamerDelegates[0],
        ), // Beamer
        Container(
          padding: const EdgeInsets.all(16.0),
          child: Beamer(
            routerDelegate: beamerDelegates[1],
          ), // Beamer
        ), // Container
      ],
    ), // IndexedStack
```

Beamer


Platform
User
/my/path
BeamerParser
RouteInformation
BeamerDelegate
.locationBuilder
BeamLocation
BeamLocation
.buildPages
BeamerDelegate
.build
Navigator

Example


8:53
Books
Stranger in a Strange Land
Robert A. Heinlein
Foundation
Isaac Asimov
Fahrenheit 451
Ray Bradbury
Stranger in a Strange Land
Robert A. Heinlein
1
Foundation
Isaac Asimov
Fahrenheit 451
Ray Bradbury
Stranger in a Strange Land
Robert A. Heinlein
Foundation
Isaac Asimov
Fahrenheit 451
Books
Articles

Example

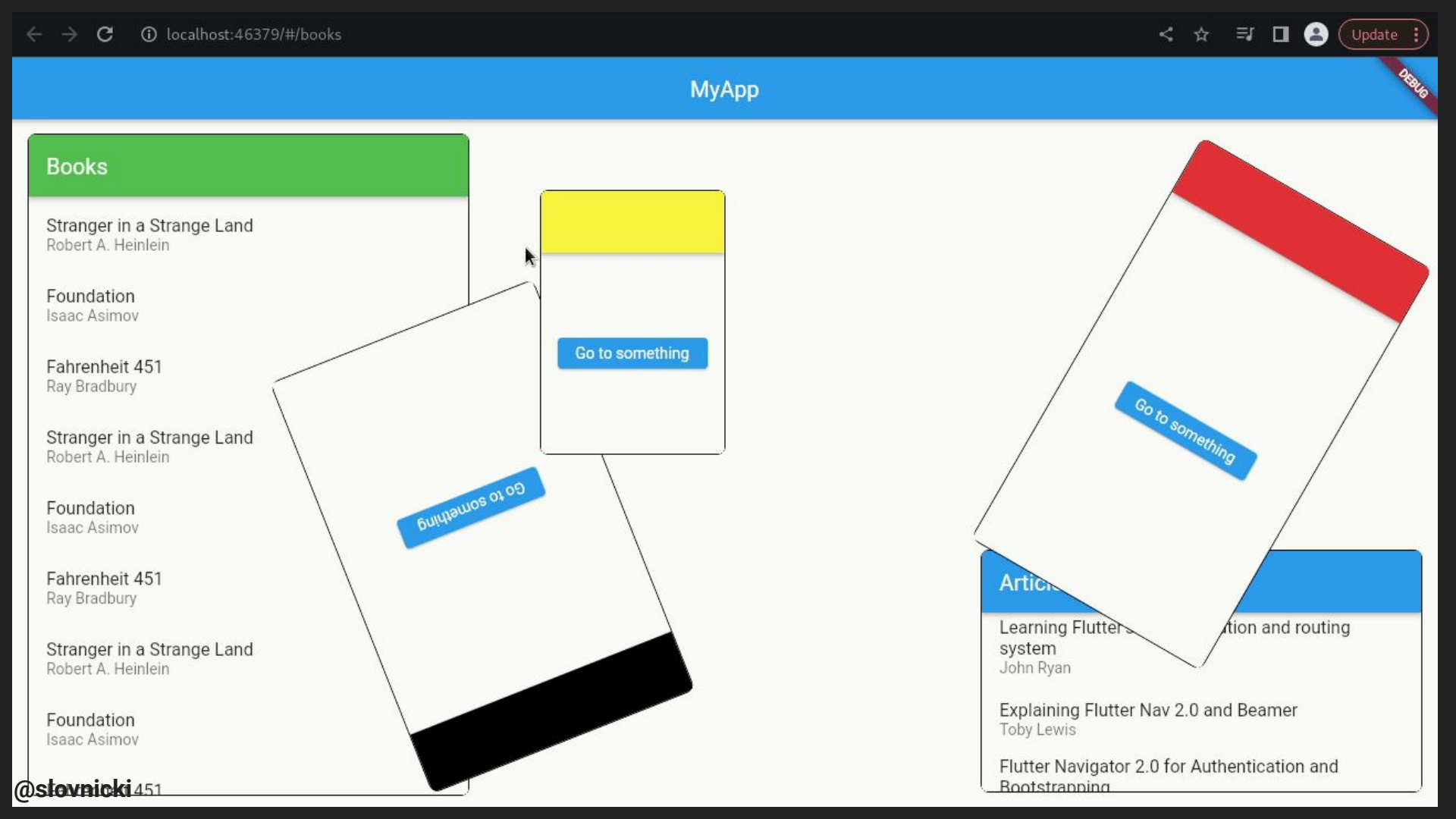
```dart
final beamerDelegates = [
  BeamerDelegate(
    initialPath: '/books',
    locationBuilder: RoutesLocationBuilder(
      routes: {
        '/books': (_, __, ___) => const BooksScreen(),
        '/books/:bookId': (_, state, __) => BookDetailsScreen(
              book: findBook(state.pathParameters['bookId']),
            ), // BookDetailsScreen
      },
    ), // RoutesLocationBuilder
  ), // BeamerDelegate
  BeamerDelegate(      You, 8 months ago • [init]
      initialPath: '/articles',
      locationBuilder: RoutesLocationBuilder(
        routes: {
          '/articles': (_, __, ___) => const ArticlesScreen(),
          '/articles/:articleId': (_, state, __) => ArticleDetailsScreen(
                article: findArticle(state.pathParameters['articleId']),
              ), // ArticleDetailsScreen
        },
      )), // RoutesLocationBuilder // BeamerDelegate
];
```

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: IndexedStack(
      index: currentIndex,
      children: [
        Beamer(routerDelegate: beamerDelegates[0]),
        Beamer(routerDelegate: beamerDelegates[1]),
      ],
    ), // IndexedStack
    bottomNavigationBar: BottomNavigationBar(
      currentIndex: currentIndex,
      items: const [
```

Bonus Example

Bonus Example

```
Positioned(
  right: 96,
  top: 48,
  child: Transform.rotate(
    angle: 1 / 6 * math.pi,
    child: ContainerWrapper(
      width: 240,
      height: 420,
      child: Beamer(
        routerDelegate: beamerDelegates[4],
      ), // Beamer
    ), // ContainerWrapper
  ), // Transform.rotate
), // Positioned
```


localhost:46379/#/books
Update
MyApp
DEBUG
Books
Stranger in a Strange Land
Robert A. Heinlein
Foundation
Isaac Asimov
Fahrenheit 451
Ray Bradbury
Stranger in a Strange Land
Robert A. Heinlein
Foundation
Isaac Asimov
Fahrenheit 451
Ray Bradbury
Stranger in a Strange Land
Robert A. Heinlein
Foundation
Isaac Asimov
Go to something
Go to something
Go to something
Learning Flutter
tion and routing system
John Ryan
Explaining Flutter Nav 2.0 and Beamer
Toby Lewis
Flutter Navigator 2.0 for Authentication and Bootstrapping
@slovnicki

Thanks!

- https://api.flutter.dev/flutter/widgets/Navigator-class.html
- https://api.flutter.dev/flutter/widgets/Router-class.html
- https://pub.dev/packages/beamer
- https://github.com/slovnicki/nested_navigation_in_flutter



@slovnicki
Sandro Lovnički