**Flutter Routing**

| Key Points | Notes |
|---|---|

**1. Navigator Widget**

a. Mobile applications are made of many screens and pages. In Flutter these elements are called routes and they are managed by a Navigator widget.

b. A Navigator widget manages a stack of route objects and provide methods or functions for managing the stack:

i. `Navigator.push()` to route to a new screen.

```
// Within the `FirstRoute` widget
onPressed: () {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) =>
SecondRoute()),
  );
}
```

ii. `Navigator.pop()` to return to the previous screen.

```
// Within the SecondRoute widget
onPressed: () {
  Navigator.pop(context);
}
```

c. The Navigator widget maintain the stack-based history of routes and the top of the stack is your current page or screen. You need at least two screens or pages to use route in Flutter.

d. For code example refer to https://flutter.dev/docs/cookbook/navigation/navigation-basics

**2. Named Route**

a. A named route is used when you want to avoid code duplication navigate to the same screen as part of your mobile application logic.

b. Instead of using `Navigator.push()` method, you can implement `Navigator.pushNamed()`.

Summary

| Key Points | Notes |
|---|---|

c. You need to define the routes before you can use named route in your Flutter.

```
MaterialApp(
  // Start the app with the "/" named route.
  // In this case, the app starts
  // on the FirstScreen widget.
  initialRoute: '/',
  routes: {
    // When navigating to the "/" route,
    // build the FirstScreen widget.
    '/': (context) => FirstScreen(),
    // When navigating to the "/second" route,
    // build the SecondScreen widget.
    '/second': (context) => SecondScreen(),
  },
);
```

d. When using `initialRoute`: do **NOT** define a home property.

e. For code example refer to https://flutter.dev/docs/cookbook/navigation/named-routes

**3. Passing Data or Arguments to a named route**

a. The best way to pass data or arguments to another screen is to use `onGenerateRoute()` method or function, which it will extract an argument and pass it to the intended widget.

b. The `MaterialPageRoute()` responsible to capture the arguments or data to be used within a widget.

Summary

| Key Points | Notes |
|---|---|

```
MaterialApp(
// Provide a function to handle named routes. Use this
// function to
// identify the named route being pushed, and create the
// correct screen.
  onGenerateRoute: (settings) {
// If you push the PassArguments route
    if (settings.name == PassArgumentsScreen.routeName) {
// Cast the arguments to the correct type: ScreenArguments.
      final ScreenArguments args = settings.arguments;

// Then, extract the required data from the arguments and
// pass the data to the correct screen.
      return MaterialPageRoute(
        builder: (context) {
          return PassArgumentsScreen(
            title: args.title,
            message: args.message,
          );
        },
      );
    }
  },
);
```

c. For code example refer to https://flutter.dev/docs/cookbook/navigation/navigate-with-arguments

**4. References**


a. Windmill, E. (2020). Flutter in Action (1st Ed.). USA: Manning Publications.

b. Flutter Official Documentation. Retrieved on 1 December 2020 from https://flutter.dev/docs

c. Singh, N. K., (2018). Flutter: Advance Routing and Navigator (Part 1). Retrieved on 1 December 2020 from https://blog.usejournal.com/flutter-advance-routing-and-navigator-df0f86f0974f


Summary