14. Flutter – Accessing REST API

Flutter provides http package to consume HTTP resources. http is a Future-based library and uses await and async features. It provides many high level methods and simplifies the development of REST based mobile applications.

Basic Concepts

http package provides a high level class and http to do web requests.

- http class provides functionality to perform all types of HTTP requests.
- http methods accept a url, and additional information through Dart Map (post data, additional headers, etc.,). It requests the server and collects the response back in async/await pattern. For example, the below code reads the data from the specified url and print it in the console.

```
print(await http.read('https://flutter.dev/'));
```

Some of the core methods are as follows:

- read - Request the specified url through GET method and return back the response as Future<String>
- get - Request the specified url through GET method and return back the response as Future<Response>. Response is a class holding the response information.
- post - Request the specified url through POST method by posting the supplied data and return back the response as Future<Response>
- put - Request the specified url through PUT method and return back the response as Future<Response>
- head - Request the specified url through HEAD method and return back the response as Future<Response>
- delete - Request the specified url through DELETE method and return back the response as Future<Response>

http also provides a more standard HTTP client class, client. client supports persistent connection. It will be useful when a lot of request to be made to a particular server. It needs to be closed properly using close method. Otherwise, it is similar to http class. The sample code is as follows:

```
var client = new http.Client();
try {
  print(await client.get('https://flutter.dev/'));
} finally {
  client.close();
}
```

Accessing Product service API

Let us create a simple application to get product data from a web server and then show the products using *ListView*.

- Create a new *Flutter* application in Android studio, *product_rest_app*
- Replace the default startup code (main.dart) with our *product_nav_app* code.
- Copy the assets folder from *product_nav_app* to *product_rest_app* and add assets inside the pubspec.yaml file

```
flutter:

  assets:
    - assets/appimages/floppy.png
    - assets/appimages/iphone.png
    - assets/appimages/laptop.png
    - assets/appimages/pendrive.png
    - assets/appimages/pixel.png
    - assets/appimages/tablet.png
```

- Configure http package in the pubspec.yaml file as shown below:

```
dependencies:
   http: ^0.12.0+2
```

- Here, we will use the latest version of the http package. Android studio will send a package alert that the pubspec.yaml is updated.

- Click Get dependencies option. Android studio will get the package from Internet and properly configure it for the application.
- Import http package in the main.dart file:

```
import 'dart:async';
import 'dart:convert';
import 'package:http/http.dart' as http;
```

- Create a new JSON file, products.json with product information as shown below:

```
[
   {
       "name": "iPhone",
       "description": "iPhone is the stylist phone ever",
       "price": 1000,
       "image": "iphone.png"
   },
   {
       "name": "Pixel",
       "description": "Pixel is the most feature phone ever",
       "price": 800,
       "image": "pixel.png"
```

```
    },
    {
        "name": "Laptop",
        "description": "Laptop is most productive development tool",
        "price": 2000,
        "image": "laptop.png"
    },
    {
        "name": "Tablet",
        "description": "Tablet is the most useful device ever for meeting",
        "price": 1500,
        "image": "tablet.png"
    },
    {
        "name": "Pendrive",
        "description": "Pendrive is useful storage medium",
        "price": 100,
        "image": "pendrive.png"
    },
    {
        "name": "Floppy Drive",
        "description": "Floppy drive is useful rescue storage medium",
        "price": 20,
        "image": "floppy.png"
    }
]
```

- Create a new folder, JSONWebServer and place the JSON file, products.json.
- Run any web server with JSONWebServer as its root directory and get its web path. For example, http://192.168.184.1:8000/products.json. We can use any web server like apache, nginx etc.,
- The easiest way is to install node based http-server application. Follow the steps given below to install and run http- server application.
    - Install Nodejs application (https://nodejs.org/en/)
    - Go to JSONWebServer folder.

```
cd /path/to/JSONWebServer
```

- Install http-server package using npm

```
npm install -g http-server
```

- Now, run the server.

```
  http-server . -p 8000

Starting up http-server, serving .
Available on:
  http://192.168.99.1:8000
  http://192.168.1.2:8000
```

```
  http://127.0.0.1:8000
Hit CTRL-C to stop the server
```

- Create a new file, Product.dart in the lib folder and move the Product class into it.

- Write a factory constructor in the Product class, Product.fromMap to convert mapped data Map into the Product object. Normally, JSON file will be converted into Dart Map object and then, converted into relevant object (Product)

```
factory Product.fromJson(Map<String, dynamic> data) {
    return Product(
      data['name'],
      data['description'],
      data['price'],
      data['image'],
    );
}
```

- The complete code of the Product.dart is as follows:

```
class Product {
  final String name;
  final String description;
  final int price;
  final String image;

  Product(this.name, this.description, this.price, this.image);

  factory Product.fromMap(Map<String, dynamic> json) {
    return Product(
      json['name'],
      json['description'],
      json['price'],
      json['image'],
    );
  }
}
```

- Write two methods - parseProducts and fetchProducts - in the main class to fetch and load the product information from web server into the List<Product> object.

```
List<Product> parseProducts(String responseBody) {
  final parsed = json.decode(responseBody).cast<Map<String, dynamic>>();
  return parsed.map<Product>((json) => Product.fromJson(json)).toList();
}

Future<List<Product>> fetchProducts() async {
  final response = await
http.get('http://192.168.1.2:8000/products.json');

  if (response.statusCode == 200) {
    return parseProducts(response.body);
  } else {
    throw Exception('Unable to fetch products from the REST API');
```

```
  }
}
```

- Note the following points here:
  - Future is used to lazy load the product information. Lazy loading is a concept to defer the execution of the code until it is necessary.
  - http.get is used to fetch the data from the Internet.
  - json.decode is used to decode the JSON data into the Dart Map object. Once JSON data is decoded, it will be converted into List<Product> using fromMap of the Product class.
  - In MyApp class, add new member variable, products of type Future<Product> and include it in constructor.

```
class MyApp extends StatelessWidget {
  final Future<List<Product>> products;

  MyApp({Key key, this.products}) : super(key: key);

  ...
```

  - In MyHomePage class, add new member variable products of type Future<Product> and include it in constructor. Also, remove items variable and its relevant method, getProducts method call. Placing the products variable in constructor. It will allow to fetch the products from Internet only once when the application is first started.

```
class MyHomePage extends StatelessWidget {
  final String title;
  final Future<List<Product>> products;

  MyHomePage({Key key, this.title, this.products}) : super(key: key);
  ...
```

- Change the home option (MyHomePage) in the build method of MyApp widget to accommodate above changes:

```
 home: MyHomePage(
          title: 'Product Navigation demo home page', products:
products),
```

- Change the main function to include Future<Product> arguments:

```
void main() => runApp(MyApp(fetchProduct()));
```

- Create a new widget, ProductBoxList to build the product list in the home page.

```
class ProductBoxList extends StatelessWidget {
  final List<Product> items;
```

```
  ProductBoxList({Key key, this.items});

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: items.length,
      itemBuilder: (context, index) {
        return GestureDetector(
          child: ProductBox(item: items[index]),
          onTap: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => ProductPage(item: items[index]),
              ),
            );
          },
        );
      },
    );
  }
}
```

Note that we used the same concept used in Navigation application to list the product except it is designed as a separate widget by passing products (object) of type List<Product>.

- Finally, modify the *MyHomePage* widget's build method to get the product information using Future option instead of normal method call.

```
Widget build(BuildContext context) {
   return Scaffold(
       appBar: AppBar(title: Text("Product Navigation")),
       body: Center(
         child: FutureBuilder<List<Product>>(
           future: products,
           builder: (context, snapshot) {
             if (snapshot.hasError) print(snapshot.error);

             return snapshot.hasData
                 ? ProductBoxList(
                     items: snapshot.data) // return the ListView widget
                 : Center(child: CircularProgressIndicator());
           },
         ),
       ));
 }
```

- Here note that we used FutureBuilder widget to render the widget. FutureBuilder will try to fetch the data from it's future property (of type Future<List<Product>>). If future property returns data, it will render the widget using ProductBoxList, otherwise throws an error.

- The complete code of the main.dart is as follows:

```
import 'package:flutter/material.dart';

import 'dart:async';
import 'dart:convert';
import 'package:http/http.dart' as http;

import 'Product.dart';

void main() => runApp(MyApp(products: fetchProducts()));

List<Product> parseProducts(String responseBody) {
  final parsed = json.decode(responseBody).cast<Map<String, dynamic>>();
  return parsed.map<Product>((json) => Product.fromMap(json)).toList();
}

Future<List<Product>> fetchProducts() async {
  final response = await
http.get('http://192.168.1.2:8000/products.json');

  if (response.statusCode == 200) {
    return parseProducts(response.body);
  } else {
    throw Exception('Unable to fetch products from the REST API');
  }
}

class MyApp extends StatelessWidget {
  final Future<List<Product>> products;

  MyApp({Key key, this.products}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(
          title: 'Product Navigation demo home page', products:
products),
    );
  }
}

class MyHomePage extends StatelessWidget {
  final String title;
  final Future<List<Product>> products;

  MyHomePage({Key key, this.title, this.products}) : super(key: key);

  // final items = Product.getProducts();
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(title: Text("Product Navigation")),
        body: Center(
          child: FutureBuilder<List<Product>>(
            future: products,
            builder: (context, snapshot) {
              if (snapshot.hasError) print(snapshot.error);

              return snapshot.hasData
                  ? ProductBoxList(
                      items: snapshot.data) // return the ListView widget
                  : Center(child: CircularProgressIndicator());
            },
          ),
        ));
  }
}

class ProductBoxList extends StatelessWidget {
  final List<Product> items;

  ProductBoxList({Key key, this.items});

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: items.length,
      itemBuilder: (context, index) {
        return GestureDetector(
          child: ProductBox(item: items[index]),
          onTap: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => ProductPage(item: items[index]),
              ),
            );
          },
        );
      },
    );
  }
}

class ProductPage extends StatelessWidget {
  ProductPage({Key key, this.item}) : super(key: key);

  final Product item;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      appBar: AppBar(
        title: Text(this.item.name),
      ),
      body: Center(
        child: Container(
          padding: EdgeInsets.all(0),
          child: Column(
              mainAxisAlignment: MainAxisAlignment.start,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: <Widget>[
                Image.asset("assets/appimages/" + this.item.image),
                Expanded(
                    child: Container(
                        padding: EdgeInsets.all(5),
                        child: Column(
                          mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
                          children: <Widget>[
                            Text(this.item.name,
                                style: TextStyle(fontWeight:
FontWeight.bold)),
                            Text(this.item.description),
                            Text("Price: " + this.item.price.toString()),
                            RatingBox(),
                          ],
                        )))
              ]),
        ),
      ),
    );
  }
}

class RatingBox extends StatefulWidget {
  @override
  _RatingBoxState createState() => _RatingBoxState();
}

class _RatingBoxState extends State<RatingBox> {
  int _rating = 0;

  void _setRatingAsOne() {
    setState(() {
      _rating = 1;
    });
  }

  void _setRatingAsTwo() {
    setState(() {
      _rating = 2;
    });
  }

  void _setRatingAsThree() {
    setState(() {
```

```
      _rating = 3;
    });
  }

  Widget build(BuildContext context) {
    double _size = 20;
    print(_rating);

    return Row(
      mainAxisAlignment: MainAxisAlignment.end,
      crossAxisAlignment: CrossAxisAlignment.end,
      mainAxisSize: MainAxisSize.max,
      children: <Widget>[
        Container(
          padding: EdgeInsets.all(0),
          child: IconButton(
            icon: (_rating >= 1
                ? Icon(
                    Icons.star,
                    size: _size,
                  )
                : Icon(
                    Icons.star_border,
                    size: _size,
                  )),
            color: Colors.red[500],
            onPressed: _setRatingAsOne,
            iconSize: _size,
          ),
        ),
        Container(
          padding: EdgeInsets.all(0),
          child: IconButton(
            icon: (_rating >= 2
                ? Icon(
                    Icons.star,
                    size: _size,
                  )
                : Icon(
                    Icons.star_border,
                    size: _size,
                  )),
            color: Colors.red[500],
            onPressed: _setRatingAsTwo,
            iconSize: _size,
          ),
        ),
        Container(
          padding: EdgeInsets.all(0),
          child: IconButton(
            icon: (_rating >= 3
                ? Icon(
                    Icons.star,
                    size: _size,
                  )
```

```
                : Icon(
                    Icons.star_border,
                    size: _size,
                  )),
            color: Colors.red[500],
            onPressed: _setRatingAsThree,
            iconSize: _size,
          ),
        ),
      ],
    );
  }
}

class ProductBox extends StatelessWidget {
  ProductBox({Key key, this.item}) : super(key: key);

  final Product item;

  Widget build(BuildContext context) {
    return Container(
        padding: EdgeInsets.all(2),
        height: 140,
        child: Card(
          child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: <Widget>[
                Image.asset("assets/appimages/" + this.item.image),
                Expanded(
                    child: Container(
                        padding: EdgeInsets.all(5),
                        child: Column(
                          mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
                          children: <Widget>[
                            Text(this.item.name,
                                style: TextStyle(fontWeight:
FontWeight.bold)),
                            Text(this.item.description),
                            Text("Price: " + this.item.price.toString()),
                            RatingBox(),
                          ],
                        )))
              ]),
        ));
  }
}
```

Finally run the application to see the result. It will be same as our *Navigation* example except the data is from Internet instead of local, static data entered while coding the application.