Week 16 : Data Storage and Management

ผศ. ดร. เก็จแก้ว ธเนศวร

kejkaew.tha@mail.kmutt.ac.th

ครั้งที่	วันที่	หัวข้อ
6	21/2/2566	Flutter – Basics of Dart Programming
7	28/2/2566	Article: Methodology + submit background
8	7/3/2566	Flutter – OOP and AppBar Widget
9	14/3/2566	Flutter – More Widgets and Layouts
10	21/3/2566	Article: Experiment and results + submit methodology
12	28/3/2566	Flutter – Layouts and Stateful Widget
11	4/4/2566	Article: Conclusions and future work + submit experiment and results
13	11/4/2566	ไม่มีเรียน (GDM443/DMT443)
14	18/4/2566	No class
15	25/4/2566	Article: Abstract
16	2/5/2566	Flutter - Data Storage and Management
17	9/5/2566	Flutter – Project example
18	16/5/2566	No class
19	30/5/2566	Flutter - ส่งโปรเจค

Last week

Refactoring widgets

Navigator

Stateful widget

Textfield

Layout: Gridview

Layout: Gridview builder

Layout: Stack

Today outline

Asynchronous Programming

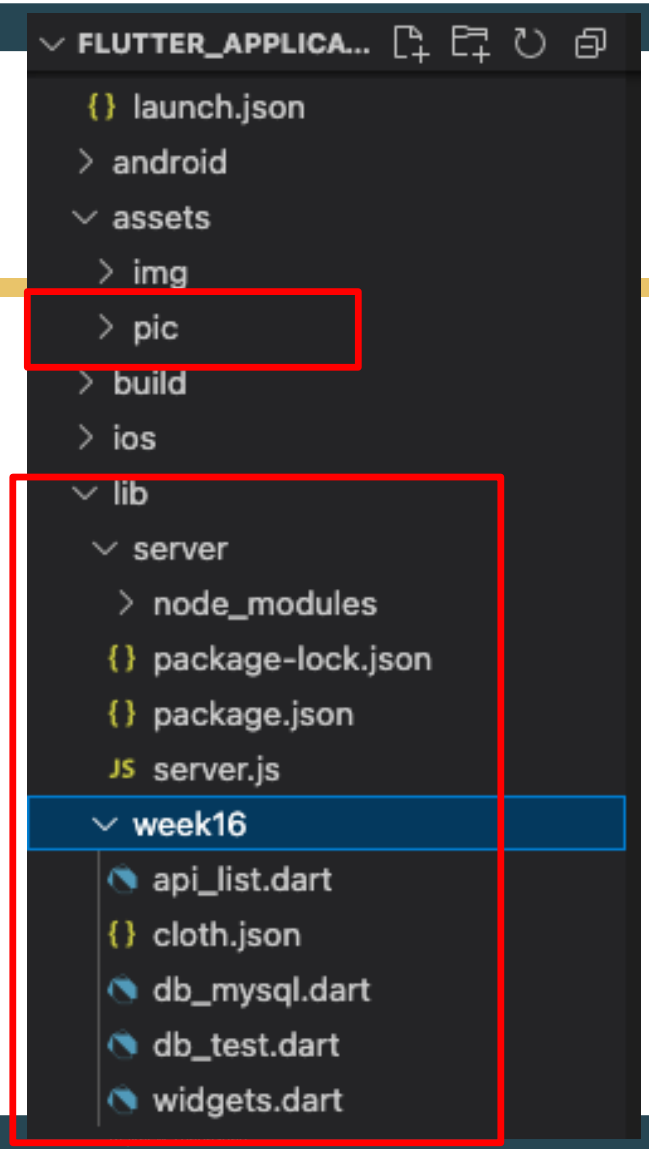
- Future and Then
- Async / Await

REST APIs

Flutter and NodeJS

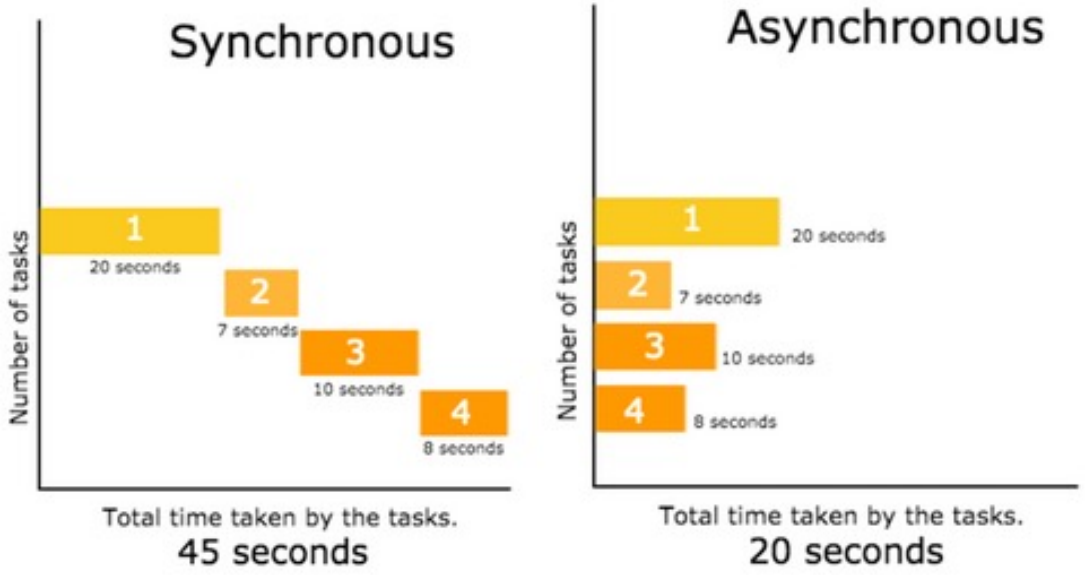
Today files

In flutter folder:



Asynchronous Programming

Asynchronous Programming is a way of writing code that allows a program to do multiple tasks at the same time.



Why We Need Asynchronous

To Fetch Data From Internet,

To Write Something to Database,

To execute a long-time consuming task,

To Read Data From File, and

To Download File etc.

Example Of Asynchronous Programming

ใน Dart: week16_async.dart

```dart
// main function
Run | Debug
void main() {
  print("First Operation");
  Future.delayed(Duration(seconds: 3), () => print('Second Big Operation'));
  print("Third Operation");
  print("Last Operation");
```

Output

```
First Operation
Third Operation
Last Operation
Second Big Operation
Exited
```

Future In Dart

In dart, the Future represents a value or error that is not yet available.

It is used to represent a potential value, or error, that will be available at some time in the future.

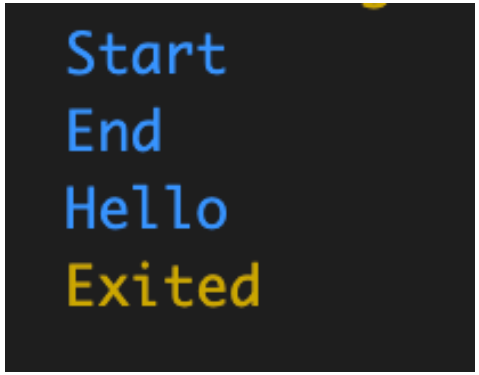
When you call an asynchronous function, it returns to an uncompleted future.

It means the future is waiting for the function asynchronous operation to finish or to throw an error.

Example:

```
31  // future in dart, using then
32  print("Start");
33  getUserName().then((value) => print(value));
34  print("End");
```

Output



Async and Await In Dart

Async/await is a feature in Dart that allows us to write asynchronous code that looks and behaves like synchronous code, making it easier to read.

To define an Asynchronous function, add async before the function body.

The await keyword work only in the async function.

Example

```
36  // using await
37  print("Start");
38  getData();
39  print("End");
```

Output

```
Start
End
Hello world
Exited
```

Example: Try-catch

```
15  void getData_2() async {
16    try {
17      String data = await middleFunction();
18      print(data);
19    } catch (err) {
20      print("Some error $err");
21    }
22  }
23
```

main

```
41    // try-catch
42    print("Start");
43    getData_2();
44    print("End");
45  }
```

Output

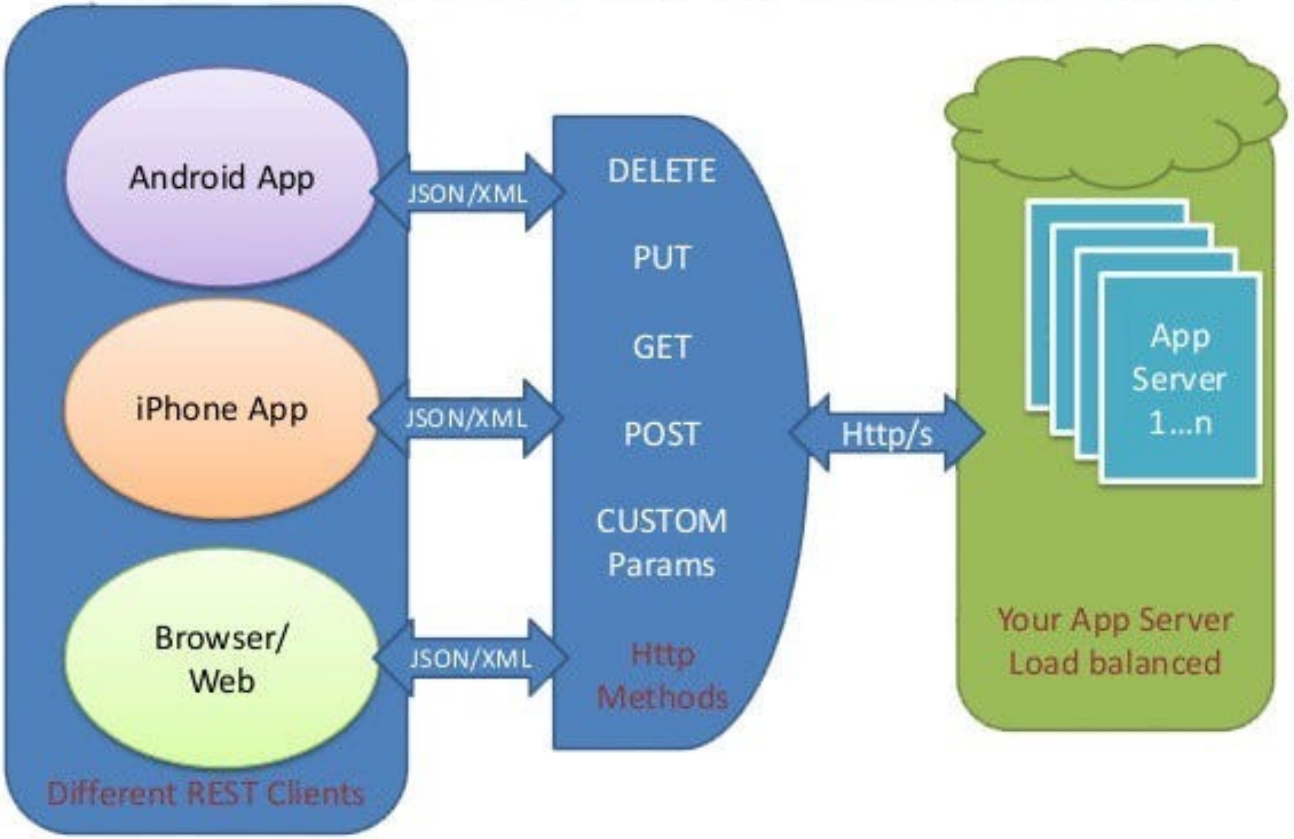
```
Start
End
Hello world
Exited
```

REST APIs

REST ย่อมาจาก Representational State Transfer เป็นรูปแบบการส่งข้อมูลระหว่าง Server-Client รูปแบบหนึ่ง อยู่บนพื้นฐานของ HTTP Protocol

API ย่อมาจาก “Application Program Interface” (ส่วนต่อประสานโปรแกรมประยุกต์) เป็นการกำหนดวิธีที่ทั้งสองสื่อสารกันโดยใช้คำขอและการตอบกลับ

REST API Architecture



HTTP package

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

Some of the core methods are as follows –

- get – Request the specified url through GET method and return back the response as Future<Response>.
- 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>
- delete – Request the specified url through DELETE method and return back the response as Future<Response>

Response is a class holding the response information.

FutureBuilder in Flutter

ใน flutter FutureBuilder คือ function app asynchronous เทียบเท่า Promise ของ javascript

ตัวอย่างใน api_list.dart

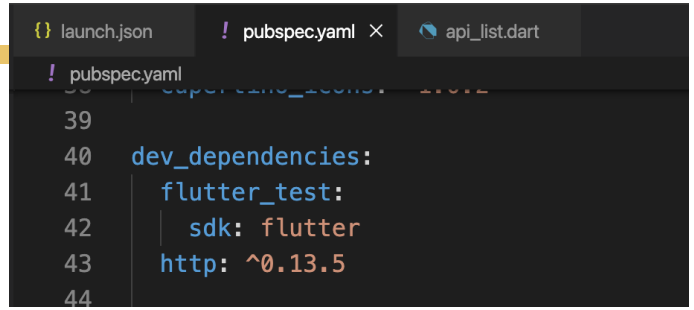
```dart
Future<List<Item>> getRequest() async {
  //replace your restFull API here.
  String url = "https://mocki.io/v1/a463a7a1-62ba-4be4-9ab9-baf7d9b61eec";
  final response = await http.get(Uri.parse(url));

  var responseData = json.decode(response.body);

  //Creating a list to store input data;
  List<Item> items = [];
  for (var singleItem in responseData) {
    Item item = Item(
        brandname: singleItem["brandname"],
        price: singleItem["price"],
        pic: singleItem["pic"]);

    //Adding user to the list.
    items.add(item);
  }
  return items;
}
```

```dart
      body: Container(
        padding: EdgeInsets.all(16.0),
        child: FutureBuilder(
          future: getRequest(),
          builder: (BuildContext ctx, AsyncSnapshot snapshot) {
            if (snapshot.data == null) {
              return Container(
                child: Center(
                  child: CircularProgressIndicator(),
                ), // Center
              ); // Container
            } else {
              return ListView.builder(
                itemCount: snapshot.data.length,
                itemBuilder: ((context, index) {
                  return Container(
                      padding: EdgeInsets.all(2),
                      height: 140,
                      child: Card(
                        child: Row(
```

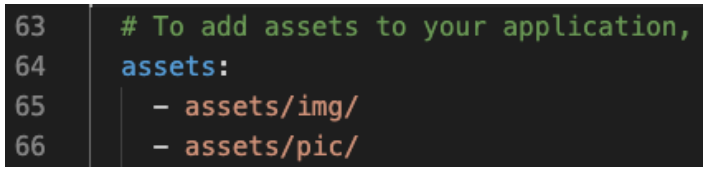
Example (api_list.dart)

ทดลองดึงข้อมูล api และแสดงผลใน flutter

ใน pubspec.yaml จะต้องเพิ่ม http: ^0.13.5 ก่อน ดังรูป



ทำการเพิ่ม assets ใน pubspec.yaml

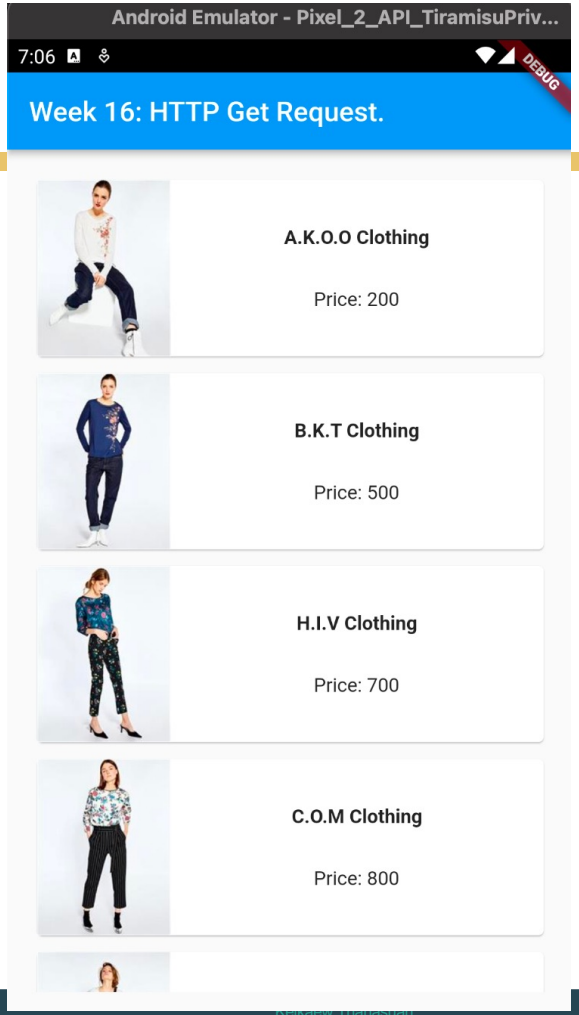


□ - assets/pic

สร้าง API endpoints เราจะใช้ อันนี้

https://mocki.io/v1/a463a7a1-62ba-4be4-9ab9-baf7d9b61eec

Example

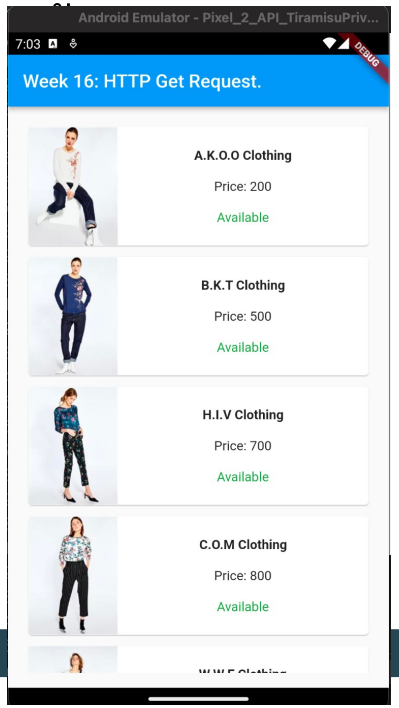

Android Emulator - Pixel_2_API_TiramisuPriv...
7:06
DEBUG
Week 16: HTTP Get Request.
A.K.O.O Clothing
Price: 200
B.K.T Clothing
Price: 500
H.I.V Clothing
Price: 700
C.O.M Clothing
Price: 800

Activity 1

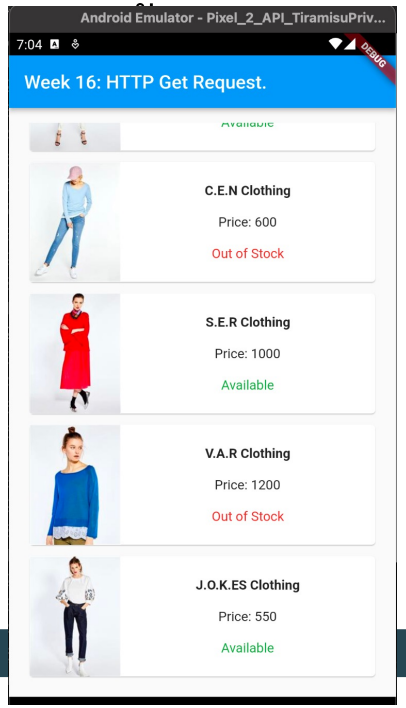
จาก API ข้อมูลเสื้อผ้า จะมีข้อมูลอีกตัวที่ยังไม่ได้ใช้คือ stock

ให้ นศ นำมาข้อมูล stock มาใช้เพื่อแสดงผล มียังมีสินค้าเหลืออยู่หรือไม่ ดังนี้

Available



Out of Stock



Flutter and NodeJS

NodeJS

Node.js is an asynchronous event-driven JavaScript runtime framework

Designed to build scalable network applications.

Node.js was developed in 2009

Node.js is free

Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)

Node.js uses JavaScript on the server

Run JS Code in Visual Studio Code

- install NodeJS บน Windows/Macbook จาก link นี้ https://nodejs.org/en/download/
- หลังจาก install และ setting ต่างๆ แล้ว จะทำการรัน server.js โดยใช้คำสั่ง node server.js

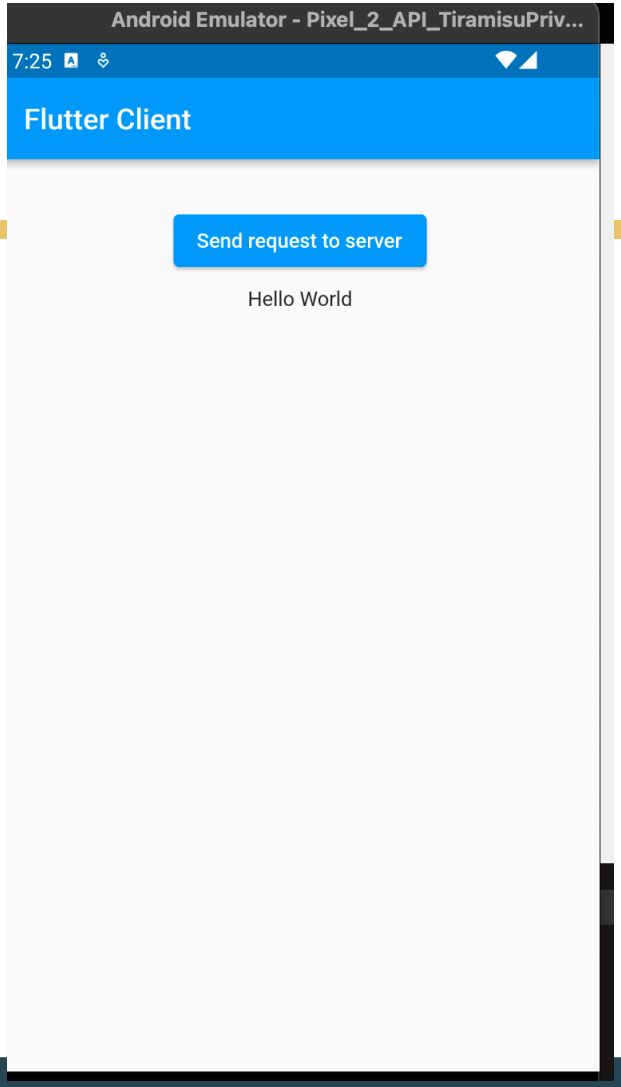
The Built-in HTTP Module

- Node.js มี built-in module ที่ชื่อว่า HTTP ซึ่งใช้สำหรับส่งข้อมูลผ่าน HTTP protocol
- สำหรับการใช้ module นี้ ต้องทำการ import ก่อนโดย ใช้ require() method:
- HTTP module จะทำการสร้าง HTTP server ที่รอการร้องขอจาก client และส่งข้อมูลกลับไปหา client ตามที่ขอ
- ใช้ createServer() method สำหรับ สร้าง HTTP server

Example: nodeJS and Flutter

```
1  const http = require('http');
2  const hostname = '127.0.0.1';
3  const port = 3000;
4  const server = http.createServer((req, res) => {
5    res.statusCode = 200;
6    res.setHeader('Content-Type', 'text/plain');
7    res.end('Hello World\n');
8  });
9  server.listen(port, hostname, () => {
10   console.log(`Server running at http://${hostname}:${port}/`);
11 });
```

server.js



NPM

- Node Package Manager
- NPM คือ package manager สำหรับ Node.js packages หรือ modules.
- วิธีการ install

npm install <Module Name>

- ตัวอย่างเช่น install express framework

npm install express

kejkaew.tha@mail.kmutt.ac.th

Express Framework

- Express คือ Node.js web application framework ที่ใช้สำหรับ พัฒนา web และ mobile applications.
- ทำให้พัฒนา applications ได้ง่ายและไวขึ้น
- ตัวอย่างที่ Express สามารถทำได้
  - สร้าง middlewares สำหรับจัดการกับ HTTP Requests.
  - กำหนด routing เพื่อทำงานบางอย่างได้ โดยใช้ร่วมกับ HTTP Method และ URL.
  - สามารถ render dynamic HTML Pages ได้ โดยการกำหนด arguments

Routing

- Express framework สามารถจัดการ route (เส้นทางการทำงาน, event) ต่างๆ ของ web application ได้ โดยใช้

app.method(path, callback)

- Method สามารถเป็น get, post, use เป็นต้น
- path คือ route ที่ต้องการให้ทำงานบางอย่าง

kejkaew.tha@mail.kmutt.ac.th

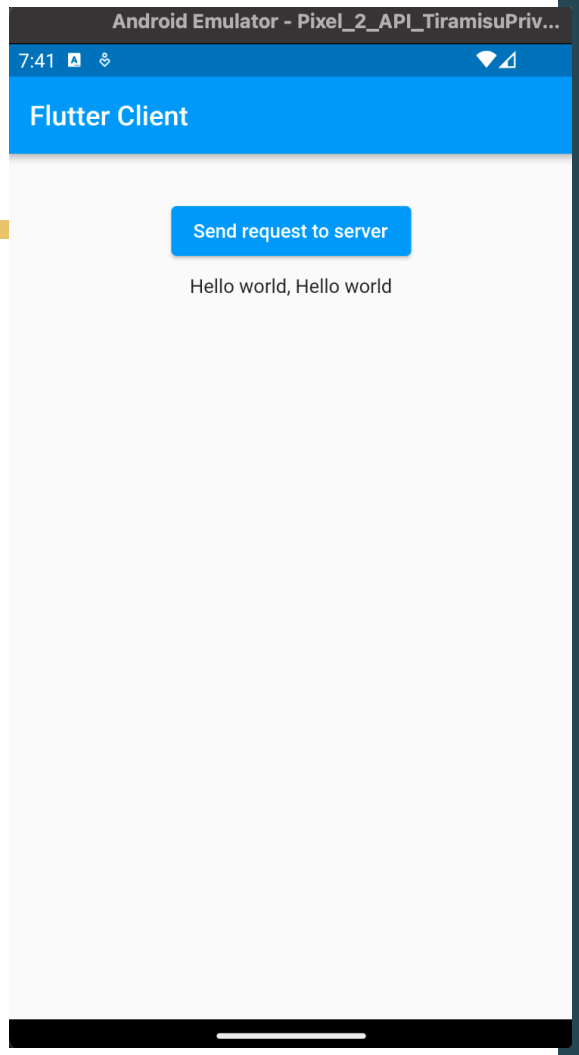
Example

```
13  // routing
14  const express = require('express');
15  const app = express();
16  const hostname = '127.0.0.1';
17  const port = 3000;
18
19  app.get('/hello', (req, res) => {
20      res.send("Hello world, Hello world");
21  });
22
23  app.listen(port, hostname, () => {
24          console.log(`Server running at   http://${hostname}:${port}/`);
25  });
```

server.js

```
62    _makeGetRequest() async {
63      // final url = Uri.parse(_localhost());
64      final url = Uri.parse("${_localhost()}/hello");
65      Response response = await get(url);
66      setState(() {
67        serverResponse = response.body;
68      });
69    }
```

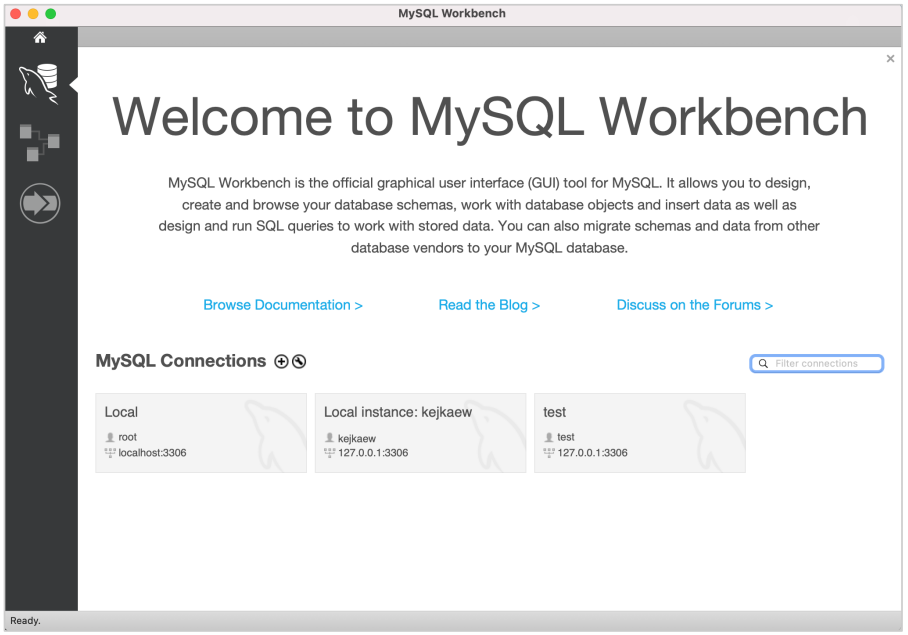
db_test.dart



MySQL

MySQL Workbench

- Download MySQL Workbench ได้จาก https://dev.mysql.com/downloads/workbench/
- ทำการติดตั้ง MySQL Workbench



Install MySQL module

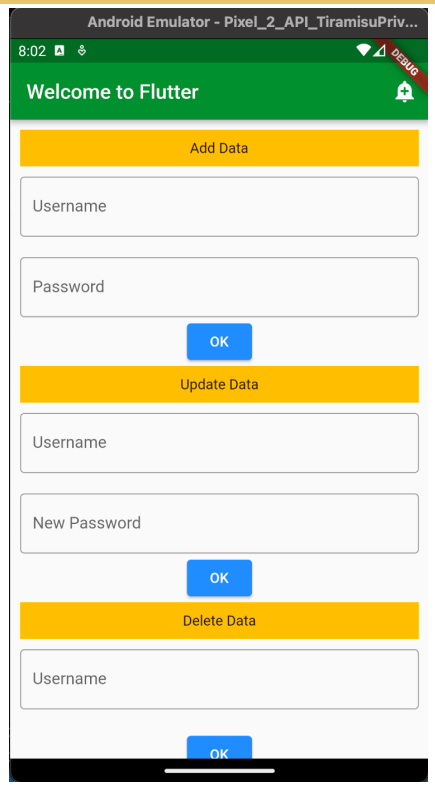
- การใช้ MySQL ต้อง install MySQL module ก่อน โดยใช้
- To install it, use the following code

```
npm install mysql
```

- การใช้ MySQL ต้องทำการ import ก่อน

```
const mysql = require('mysql');
```

Example: Flutter+ NodeJS + MySQL




```
JS server.js > extended
27  // MySQL
28  const express = require('express');
29  const app = express();
30  const hostname = '127.0.0.1';
31  const port = 3000;
32  const bodyParser = require('body-parser');
33  const mysql = require('mysql');
34
35  app.use(bodyParser.json());
36  app.use(bodyParser.urlencoded({extended: false}));
37
38  // ใส่ค่าตามที่เราตั้งไว้ใน mysql
39  const con = mysql.createConnection({
40      host: "127.0.0.1",
41      user: "root",
42      password: "mdt312root",
43      database: "mydb"
44  })
45
```

server.js
(uncomment ตั้งแต่บรรทัดที่ 28 ลงไป)

Activity 2

จาก db_mysql.dart ลองสร้าง function ใน flutter สำหรับลบข้อมูลใน Database และเรียกใช้งาน function นั้นด้วย เพื่อลบข้อมูลออกจาก database

Flutter Project (ส่งวันที่ 30 พ.ค. 2566)

งานกลุ่ม กลุ่มละไม่เกิน 6 คน

ทำ Online shopping Application โดยใช้ Flutter ประกอบด้วย

- Products
- Login page
- Shopping page
- สามารถกดเข้าไปดูสินค้าได้ กดซื้อได้ หรือกดให้คะแนนได้
- มี 1 database หรือ ใช้ json api ก็ได้

เน้นการออกแบบ UI application

Week 16: Classroom game

ตอบคำถามตาม link นี้เลย...

https://forms.gle/W92pwJkxpcNJafeC7

Reference

https://dart-tutorial.com/introduction-and-basics/

https://www.tutorialspoint.com/flutter/flutter_introduction_to_widgets.htm

https://docs.flutter.dev/development/ui/widgets-intro