06

http

네트워킹과 비동기

JSON 파싱하기

JSON 데이터 디코딩과 인코딩하기

- JSON 데이터를 디코딩하거나 인코딩하려면 dart:convert 패키지에서 제공하는 jsonDecode()와 jsonEncode() 함수를 이용
- JSON 문자열을 Map 형식으로 변경할 때는 jsonDecode() 함수를 이용
- Map 데이터를 문자열로 변경할 때는 jsonEncode() 함수를 이용

• JSON 문자열

```
String jsonStr = '{"id": 1, "title": "HELLO", "completed": false}';
```

• Map 형식으로 변환하기

```
Map<String, dynamic> map = jsonDecode(jsonStr);

setState(() {
  result = "decode : id: ${map['id']}, title: ${map['title']},
           completed: ${map['completed']}";
});
```

JSON 파싱하기

JSON 데이터 디코딩과 인코딩하기

- [{}, {}] 형식의 문자열이라면 결과를 List 타입으로 받을 수 있습니다.

• List 타입 데이터 디코딩

```
String jsonStr = '[{"id": 1, "title": "HELLO", "completed": false},{"id": 2, "title":
"WORLD", "completed": false}]';

... (생략) ...

onPressDecode() {
  List list = jsonDecode(jsonStr);
  var list1 = list[0];
  if (list1 is Map) {
    setState(() {
      result = "decode : id: ${list[0]['id']}, title: ${list[0]['title']},
               completed: ${list[0]['completed']}";
    });
  }
}
```

JSON 파싱하기

JSON 데이터 디코딩과 인코딩하기

- Map 데이터를 JSON 문자열로 변환할 때는 jsonEncode() 함수를 이용

• Map 데이터를 JSON 문자열로 변환하기

```
result = "encode : ${jsonEncode(map)}";
```

• List 타입 데이터 인코딩

```
result = "encode : ${jsonEncode(list)}";
```

JSON 파싱하기

모델 클래스로 JSON 데이터 이용하기

- JSON 데이터를 모델 클래스로 이용하려면 키값을 모델 클래스의 각 속성에 대입

• 모델 클래스 구현하기(JSON 데이터 매핑)

```
class Todo {
  int id;
  String title;
  bool completed;

  Todo(this.id, this.title, this.completed);

  Todo.fromJson(Map<String, dynamic> json)
    : id = json['id'], title = json['title'], completed = json['completed'];

  Map<String, dynamic> toJson() => {
    'id': id,
    'title': title,
    'completed': completed
  };
}
```

• 모델 클래스 이용하기

```
Map<String, dynamic> map = jsonDecode(jsonStr);
Todo todo = Todo.fromJson(map);
.... (생략) ....
String str = jsonEncode(todo);
```

http 패키지 이용하기

- 플러터 앱에서 서버와 HTTP 통신을 하려면 http 패키지를 이용

• http 패키지 추가하기

```
dependencies:
  http: ^0.13.4
```

• http 패키지 임포트

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

- 서버에 요청하는 get() 함수를 이용

• 서버에 요청하기

```
http.Response response =
    await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts/1'));
```

http 패키지 이용하기

- 응답 결과는 http.Response 타입으로 전달
- Response 객체의 statusCode 속성으로 응답 상태 코드를 확인할 수 있으며 서버에서 전달한 데이터는 body 속성으로 얻습니다.

• 서버에서 전달한 데이터 얻기

```
if (response.statusCode == 200) {
  String result =  response.body;
}
```

- 서버에 요청할 때 헤더를 지정하고 싶다면 Map 객체에 담은 후 get() 함수를 호출할 때 headers 매개변수에 지정

• 헤더 이용하기

```
Map<String, String> headers = {
  "content-type": "application/json",
  "accept": "application/json",
};
http.Response response =
    await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts/1'),
                   headers: headers);
```

http 패키지 이용하기

- get() 함수는 GET 방식으로 요청할 때 사용합니다. GET 이외에 post(), put(), delete() 함수를 이용해 각 방식으로 서버에 요청할 수 있습니다.

• POST 방식으로 요청하기

```
http.Response response =
await http.post(Uri.parse('https://jsonplaceholder.typicode.com/posts'),
                body: {'title':'hello', 'body':'world', 'userId': '1'});
```

http 패키지 이용하기

- 같은 URL로 반복해서 요청할 때는 매번 서버와 접속했다 끊었다 반복하는 것이 비효율적
- 한 번 연결된 접속을 유지하는 것이 효율적이며 이때 Client 객체를 사용

• 반복해서 요청하기

```
var client = http.Client();
try {
  http.Response response =
      await client.post(Uri.parse('https://jsonplaceholder.typicode.com/posts'),
                        body: {'title':'hello', 'body':'world', 'userId': '1'});

  if (response.statusCode == 200 || response.statusCode == 201) {
    response = await client.get(
      Uri.parse('https://jsonplaceholder.typicode.com/posts/1')
    );
    print('response: ${response.body}');
  } else {
    print('error.....');
  }
} finally {
  client.close();
}
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare