Flutter API 호출 (Dio)

Dio와 Axios 비교

특징	Dio (Dart/Flutter)	Axios (JavaScript/Node.js)
주요 플랫폼	Flutter/Dart	JavaScript/Node.js
기능 지원	- 요청 취소- 인터셉터- 파일 업로드/다운로드- 글로벌 설정	- 요청 취소- 인터셉터- 브라우저와 Node.js에서 모두 사용 가능
기본 요청	`http` 패키지 기반에서 확장	브라우저의 `XMLHttpRequest` 및 Node.js의 `http` 모듈 기반
API 사용 방식	REST API 호출과 JSON 직렬화에 최적화	REST API 호출과 JSON 직렬화에 최적화
커스텀 설정	BaseOptions를 통한 글로벌 설정 지원	AxiosInstance를 통한 글로벌 설정 지원
성능	Flutter에 최적화된 네이티브 퍼포먼스	JavaScript 환경에서 빠른 HTTP 요청

Dio의 사용법

Dio는 Flutter/Dart에서 HTTP 요청을 간단하고 유연하게 처리할 수 있도록 설계된 라이브러리입니다.

1. Dio 설치

`pubspec.yaml` 파일에 Dio 패키지를 추가합니다.

```
yaml
복사편집
dependencies:
  dio: ^5.0.0
```

명령어 실행:

```
bash
복사편집
```

```
flutter pub get
```

2. Dio 기본 사용법

1. **Dio 인스턴스 생성**:

```
dart
복사편집
import 'package:dio/dio.dart';

void main() async {
  final dio = Dio(); // 기본 인스턴스 생성
  final response = await dio.get('https://jsonplaceholde
r.typicode.com/posts');
  print(response.data); // 응답 데이터 출력
}
```

2. **BaseOptions로 기본 설정 추가**:

```
dart
복사편집
final dio = Dio(BaseOptions(
  baseUrl: 'https://api.example.com',
  connectTimeout: Duration(seconds: 10), // 연결 타임아웃
  receiveTimeout: Duration(seconds: 10), // 응답 타임아웃
  headers: {
    'Authorization': 'Bearer YOUR_TOKEN',
    'Content-Type': 'application/json',
  },
));
```

3. HTTP 요청 예제

1. **GET 요청**:

```
dart
복사편집
final response = await dio.get('/users');
print(response.data);
```

2. **POST 요청**:

```
dart
복사편집
final response = await dio.post('/users', data: {
  'name': 'John Doe',
  'email': 'john.doe@example.com',
});
print(response.data);
```

3. **PUT 요청**:

```
dart
복사편집
final response = await dio.put('/users/1', data: {
  'name': 'Updated Name',
});
print(response.data);
```

4. **DELETE 요청**:

```
dart
복사편집
final response = await dio.delete('/users/1');
print(response.statusCode);
```

---

4. Interceptors (요청/응답 인터셉터)

1. **요청 인터셉터**:

```
dart
복사편집
dio.interceptors.add(InterceptorsWrapper(
  onRequest: (options, handler) {
    print('Request: ${options.method} ${options.path}');
    return handler.next(options);
  },
));
```

2. **응답 인터셉터**:

```
dart
복사편집
dio.interceptors.add(InterceptorsWrapper(
  onResponse: (response, handler) {
    print('Response: ${response.statusCode}');
    return handler.next(response);
  },
));
```

3. **오류 인터셉터**:

```
dart
복사편집
dio.interceptors.add(InterceptorsWrapper(
  onError: (DioError e, handler) {
    print('Error: ${e.message}');
    return handler.next(e);
  },
));
```

---

5. 파일 업로드

Dio는 파일 업로드를 간단히 처리할 수 있습니다.

```
dart
복사편집
final file = await MultipartFile.fromFile('path/to/file');
final formData = FormData.fromMap({
  'name': 'My File',
  'file': file,
});

final response = await dio.post('/upload', data: formData);
print(response.data);
```

---

6. 요청 취소

```
dart
복사편집
final cancelToken = CancelToken();

dio.get('/users', cancelToken: cancelToken).catchError((e)
{
  if (CancelToken.isCancel(e)) {
    print('Request canceled!');
  }
});

// 요청 취소
cancelToken.cancel();
```

Dio를 사용할 때 유의할 점

1. **타임아웃 설정**: 네트워크가 불안정할 경우 적절한 타임아웃 설정 필수.
2. **에러 핸들링**: 요청 실패 시 예외 처리(`try-catch`)를 통해 오류를 관리.
3. **인터셉터 활용**: 요청 로깅, 인증 토큰 추가 등으로 인터셉터를 활용.

결론

- Dio 는 axios 처럼 기본 옵션 설정(전역 설정) 을 한 뒤에 get , post , formdata 등을 통해 api 요청을 하므로 사실상 axios 와 매우 유사하다.
- 따라서 axios 를 사용해 본 경험이 있다면 쉽게 사용할 수 있을 거라 생각한다.