Dart Async

1. Async Cache, AsyncMemoizer

2025.04.15 suojae

Async Cache

lib > main.dart

```dart
import 'dart:async';
import 'package:async/async.dart';

final cache = AsyncCache<String>(Duration(seconds: 5));

Future<String> getData() {
  return cache.fetch(() async {
    print('⏳ 실제 데이터 요청...');
    await Future.delayed(Duration(seconds: 1));
    return '데이터 받아옴';
  });
}

Run Application
void main() async {
  print('1회 요청');
  print(await getData()); // ⏳ 출력됨

  print('2회 요청 (3초 후)');
  await Future.delayed(Duration(seconds: 3));
  print(await getData()); // 캐시 사용, ⏳ 출력 안됨

  print('3회 요청 (6초 후)');
  await Future.delayed(Duration(seconds: 3));
  print(await getData()); // ⏳ 다시 출력됨 (캐시 만료됨)
}
```

LOGS PROBLEMS 2 OUTPUT

```
[13:30:42] Starting build...
[13:30:43] ✅ Build successful
[13:30:46] 1회 요청
[13:30:46] ⏳ 실제 데이터 요청...
[13:30:47] 데이터 받아옴
[13:30:47] 2회 요청 (3초 후)
[13:30:50] 데이터 받아옴
[13:30:50] 3회 요청 (6초 후)
[13:30:53] ⏳ 실제 데이터 요청...
[13:30:54] 데이터 받아옴
```

AsyncCache 3가지 주요함수

(1) fetch()
(2)~~fetchStream()~~ (스트림 처리)
(3) invalidate()

```dart
import 'dart:async';
import 'package:async/async.dart';

final _cache = AsyncCache<String>(Duration(seconds: 5));

Future<String> getData() {
  return _cache.fetch(() async {
    print('📡 실제 API 요청');
    return '결과 데이터';
  });
}
```

```dart
import 'package:async/async.dart';

final cache = AsyncCache<Stream<List<Message>>>(Duration(minutes: 5));

Future<Stream<List<Message>>> getChatMessages(String chatId) {
  return cache.fetch(() async {
    return FirebaseFirestore.instance
      .collection('chats/$chatId/messages')
      .snapshots()
      .map((snap) => snap.docs.map((d) => Message.fromJson(d.data())).toList());
  });
}
```

```dart
import 'package:async/async.dart';

final cache = AsyncCache<String>(Duration(seconds: 30));

Future<String> getData() {
  return cache.fetch(() async {
    print('📡 실제 데이터 요청');
    return '서버에서 받아온 데이터';
  });
}

void clearCache() => cache.invalidate(); // 👈 캐시 무효화 (다음 fetch는 반드시 새로 실행됨)
```

Async Memoizer

```dart
class SomeResource {
  final _closeMemo = AsyncMemoizer<void>();

  Future<void> close() {
    return _closeMemo.runOnce(() async {
      print('🔒 리소스 정리 중...');
      await Future.delayed(Duration(seconds: 2));
      print('✅ 리소스 정리 완료');
    });
  }
}

Run Application
void main() async {
  final resource = SomeResource();

  print('🟢 1회 호출');
  await resource.close(); // 실제 실행됨

  print('🟢 2회 호출');
  await resource.close(); // 캐시된 Future 반환

  print('🟢 3회 호출');
  await resource.close(); // 여전히 캐시된 Future
}
```

LOGS PROBLEMS 2 OUTPUT

```
[13:51:53] Starting build...
[13:51:57] ✅ Build successful
[13:52:01] 🟢 1회 호출
[13:52:01] 🔒 리소스 정리 중...
[13:52:03] ✅ 리소스 정리 완료
[13:52:03] 🟢 2회 호출
[13:52:03] 🟢 3회 호출
```

AsyncMemoizer 주요함수

(1) runOnce()

```dart
class SomeResource {
  final _closeMemo = AsyncMemoizer<void>();

  Future<void> close() {
    return _closeMemo.runOnce(() async {
      print('🔒 리소스 정리 중...');
      await Future.delayed(Duration(seconds: 2));
      print('✅ 리소스 정리 완료');
    });
  }
}

Run Application
void main() async {
  final resource = SomeResource();

  print('🟢 1회 호출');
  await resource.close(); // 실제 실행됨

  print('🟢 2회 호출');
  await resource.close(); // 캐시된 Future 반환

  print('🟢 3회 호출');
  await resource.close(); // 여전히 캐시된 Future
}
```

LOGS PROBLEMS 2 OUTPUT

```
[13:51:53] Starting build...
[13:51:57] ✅ Build successful
[13:52:01] 🟢 1회 호출
[13:52:01] 🔒 리소스 정리 중...
[13:52:03] ✅ 리소스 정리 완료
[13:52:03] 🟢 2회 호출
[13:52:03] 🟢 3회 호출
```

UseCase: Throttle

```dart
import 'dart:async';

Future<String>? _inFlight;
DateTime? _lastExecuted;

Future<String> getDataManualThrottle() async {
  final now = DateTime.now();

  if (_inFlight != null && (_lastExecuted != null && now.difference(_lastExecuted!) < Duration(seconds: 2))) {
    print('🕒 [Throttle] 요청 무시하고 기존 Future 반환');
    return _inFlight!;
  }

  print('📡 [Throttle] 실제 API 호출!');
  _lastExecuted = now;
  _inFlight = Future.delayed(Duration(seconds: 2), () => '서버 응답');

  final result = await _inFlight!;
  _inFlight = null;
  return result;
}

Run Application
void main() async {
  for (int i = 0; i < 5; i++) {
    Future.delayed(Duration(milliseconds: i * 500), () async {
      print('🔔 호출 $i');
      final result = await getDataManualThrottle();
      print('✅ 결과 $i: $result');
    }); // Future.delayed
  }
}
```

직접 구현

```dart
import 'package:async/async.dart';

final AsyncCache<String> _cache = AsyncCache(Duration(seconds: 2));

Future<String> getData() {
  return _cache.fetch(() async {
    print('📡 [AsyncCache] 실제 API 호출!');
    await Future.delayed(Duration(seconds: 2));
    return '서버 응답';
  });
}

Run Application
void main() async {
  for (int i = 0; i < 5; i++) {
    Future.delayed(Duration(milliseconds: i * 500), () async {
      print('🔔 호출 $i');
      final result = await getData();
      print('✅ 결과 $i: $result');
    }); // Future.delayed
  }
}
```

AsyncCache 사용

Thanks!