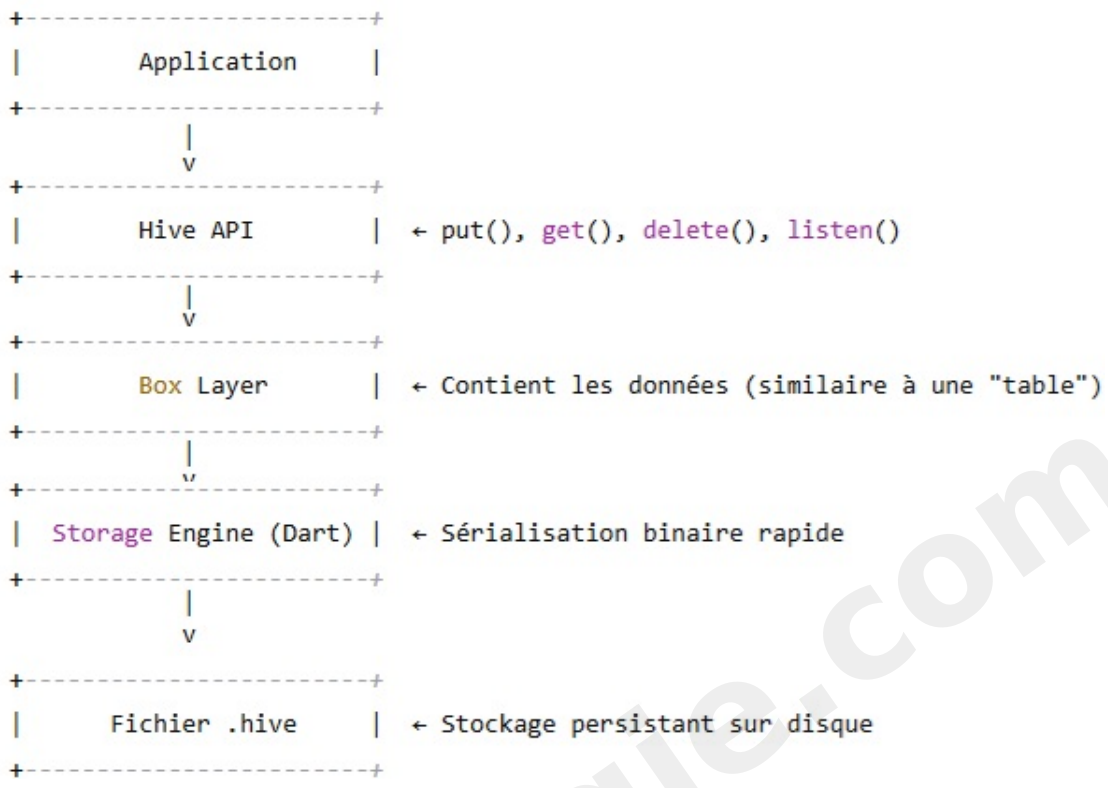
Introduction à Hive et comparaison avec SQLite, Drift et ObjectBox

Introduction à Hive et comparaison avec SQLite, Drift et ObjectBox

1. **Objectif**
    - Comprendre ce qu'est **Hive**.
    - Savoir pourquoi l'utiliser dans une application Flutter.
    - Apprendre à installer et configurer Hive.
    - Maîtriser les opérations CRUD (Create, Read, Update, Delete).
    - Utiliser Hive avec des **objets personnalisés** grâce aux adapters.
    - Découvrir les bonnes pratiques et comparer Hive avec d'autres solutions de persistance.
2. **? Qu'est-ce que Hive ?**
    - **Hive** est une base de données **NoSQL** écrite en pur Dart.
    - Elle est rapide, légère et **offline-first** (fonctionne sans internet).
    - Hive stocke les données sous forme de paires clé-valeur et est optimisée pour des opérations de lecture-écriture rapides.
    - Elle permet de stocker des données localement sur l'appareil.
    - Idéale pour :
        - Stocker des paramètres (ex : thème sombre/clair).
        - Sauvegarder des favoris.
        - Créer un cache local pour éviter les appels API répétés.
        - Stocker des objets simples (utilisateurs, produits, etc.).
3. **Architecture interne de Hive**
    - Hive repose sur une architecture clé-valeur optimisée.

apcpedagogie.com



- **Composants internes**
  - **Boxes :**
    - Contiennent les données (clé-valeur).
    - Peuvent être persistantes ou temporaires (in-memory).
  - **HiveObject / TypeAdapters :**
    - Permettent de stocker des objets Dart complexes.
    - Ex. UserAdapter pour convertir User en binaire et inversement.
  - **Moteur de stockage :**
    - Basé sur des fichiers binaires .hive.
    - Optimisé pour I/O séquentiel.
  - **LazyBox :**
    - Chargement paresseux (utile pour grandes quantités de données).

4. ? Les Boxes dans Hive

- Toutes les données dans **Hive Flutter** sont organisées et stockées dans des `boxes`.
- On peut les comparer à des **tables en SQL**, mais elles n’ont pas de structure fixe et peuvent stocker n’importe quel type de données.
- Une seule box peut suffire pour des petites applications.
- Les boxes peuvent aussi être **chiffrées** pour plus de sécurité.

apcpedagogie.com

1. **? Ouvrir une Box**

   - Une box doit être ouverte avant de pouvoir être utilisée.
   - Lorsqu'elle est ouverte, toutes les données du stockage local sont chargées en mémoire pour un accès immédiat.

```
var box = await Hive.openBox<E>('testBox');
```

2. **? Description des paramètres**

Paramètre	Description
`name`	Spécifie l'emplacement de stockage et vérifie si une box du même nom existe déjà (sensible à la casse).
`encryptionKey`	Tableau de bytes de taille 32 utilisé pour le chiffrement et le déchiffrement des valeurs.
`keyComparator`	Permet de personnaliser l'ordre de tri des clés (par défaut lexicographique).
`compactionStrategy`	Définit les règles de compaction automatique.
`crashRecovery`	En cas d'arrêt brutal pendant une écriture, la dernière entrée est supprimée. Peut être personnalisé.
`path`	Répertoire de stockage défini par `Hive.init()`. Peut être modifié.
`bytes`	Permet d'utiliser une box sous forme binaire comme backend.

3. **? Récupérer une Box ouverte**

   - Hive Flutter conserve une référence de toutes les boxes ouvertes.
   - Pour accéder à une box déjà ouverte :

```
var box = Hive.box('myBox');
```

4. **Fermer une Box**

   - Si une box n'est plus nécessaire, il faut la fermer.
   - Lors de la fermeture, tous les fichiers de cache et valeurs sont supprimés.
   - Si l'application est encore en cours, laissez la box ouverte.

apcpedagogie.com

```
var box = await Hive.openBox('myBox');
await box.put('hello', 'world');
await box.close();
```

⚠ **Avant de quitter l'application, vous devez fermer les boxes utilisées.**

5. **? Paramètre de type : `Box<E>`**

   - Vous pouvez spécifier le type des valeurs d'une box.

```
var box = await Hive.openBox<User>('users');

box.add(User());

box.add(5); // Erreur de compilation
```

   - La box peut aussi contenir des **sous-types** de `User`.
   - ⚠ Attention : la même box ne peut pas être ouverte plusieurs fois avec des paramètres de type différents.

6. **? Lazy Box**

   - Lorsqu'une box est ouverte, son contenu est chargé en mémoire.
   - Cela fonctionne pour les petites applications, mais pas pour les grandes.
   - Avec une **Lazy Box**, seules les clés sont gardées en mémoire. Les valeurs sont lues depuis le disque au moment de l'accès.

```
var lazyBox = await Hive.openLazyBox('myLazyBox');

var value = await lazyBox.get('lazyVal');
```

Obtenir une Lazy Box déjà ouverte :

```
var lazyBox = Hive.lazyBox('myLazyBox');
```

7. **? Encrypted Box**

   - Hive Flutter supporte le chiffrement **AES-256** pour stocker les données.
   - ? Une clé de 256 bits est nécessaire.

apcpedagogie.com

- Hive peut générer une clé sécurisée que vous stockez avant d'ouvrir la box.

```
import 'dart:convert';
import 'package:hive/hive.dart';
import 'package:flutter_secure_storage/flutter_secure_storage.dart';

void main() async {
  const secureStorage = FlutterSecureStorage();

  // Lire la clé stockée (ou null si absente)
  final encryptionKeyString = await secureStorage.read(key: 'key');

  if (encryptionKeyString == null) {
    final key = Hive.generateSecureKey();
    await secureStorage.write(
      key: 'key',
      value: base64UrlEncode(key),
    );
  }

  final key = await secureStorage.read(key: 'key');
  final encryptionKeyUint8List = base64Url.decode(key!);

  print('Clé de chiffrement Uint8List: $encryptionKeyUint8List');

  final encryptedBox = await Hive.openBox(
    'vaultBox',
    encryptionCipher: HiveAesCipher(encryptionKeyUint8List),
  );

  encryptedBox.put('secret', 'Hive est génial');
  print(encryptedBox.get('secret'));
}
```

- ⚠ Seules les **valeurs** sont chiffrées, les **clés** restent en clair.
- ⚠ Si une mauvaise clé est utilisée, des comportements inattendus peuvent survenir.

5. ⚡ Comparaison avec SQLite, Drift et ObjectBox

Critère	Hive ?	SQLite ?	Drift ⚙	ObjectBox ?
Type	NoSQL clé-valeur	SQL relationnelle	ORM SQL en Dart	NoSQL orienté objet

Performance	? Très rapide	Bonne	Bonne	? Très rapide
Complexité	Simple	Élevée (SQL)	Intermédiaire	Simple
Relations	□ Non natif	□ Oui	□ Oui	□ Oui
Cas d’usage	Cache, préférences	Données relationnelles	Apps complexes avec relations	Objets interconnectés

6. Architecture interne de Hive

Cours Flutter

Source : https://apcpedagogie.com/introduction-a-hive-et-comparaison-avec-sqlite-drift-et-objectbox · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com