Fiche de Référence : Flutter Avancé

Appeler des APIs dans Flutter

Type	Description	Exemple de Code

Faire Votre Premier Appel API	Apprenez à effectuer une requête HTTP GET basique	`Future<void> fetchData() async {`
		`  var response = await http.get(Uri.parse('https://api.example.com/data'));`
		`  print(response.body);`
		`}`

Gestion des erreurs d'API	Implémentez la gestion des erreurs pour les requêtes API	`try {`
		`  var response = await http.get(Uri.parse('https://api.example.com/data'));`
		`  print(response.body);`
		`} catch (e) {`
		`  print('Caught error: $e');`
		`}`

Analyse des données JSON	Décoder les données JSON reçues d'une API en objets Dart	`var data = jsonDecode(response.body);` `print('User name: ${data['username']}');`

Programmation Asynchrone	Utilisez async et await pour les appels API asynchrones	`Future<void> getUserData() async {`
		`  var response = await http.get(Uri.parse('https://api.example.com/users/1'));`
		`  var data = jsonDecode(response.body);`
		`  print('User name: ${data['name']}');`
		`}`

Envoyer des données à l'API	Envoyer des données à une API en utilisant la méthode POST	`Future<void> postData() async {`
		`  var response = await http.post(Uri.parse('https://api.example.com/add'), body: {'name': 'John Doe'});`
		`  print('Response status: ${response.statusCode}');`
		`}`

Récupérer des données avec des en-têtes	Effectuer des requêtes HTTP avec des en-têtes supplémentaires	`Future<void> fetchWithHeaders() async {`
		`  var response = await http.get(`
		`  Uri.parse('https://api.example.com/data'),`
		`  headers: {'Authorization': 'Bearer your_token_here'}`
		`  );`
		`  print('Data with headers: ${response.body}');`
		`}`

Introduction aux fonctionnalités natives mobiles dans Flutter

Type	Description	Exemple de code

Accéder à la caméra	Utiliser la fonctionnalité native de la caméra via le plugin Flutter	`Future<void> takePicture() async {`
		`  final cameras = await availableCameras();`
		`  final firstCamera = cameras.first;`
		`  final CameraController controller = CameraController(firstCamera, ResolutionPreset.high);`
		`  await controller.initialize();`
		`  final image = await controller.takePicture();`
		`  controller.dispose();`
		`}`

Utilisation du GPS	Récupérer la position actuelle en utilisant le plugin de géolocalisation	`Position position = await`
		`Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.high);`
		`print('Current location: ${position.latitude}, ${position.longitude}');`

Stockage local	Stockez des données localement en utilisant le plugin shared_preferences	`SharedPreferences prefs = await`
		`SharedPreferences.getInstance();`
		`await prefs.setString('username', 'exampleUser');`

Accéder aux capteurs de l'appareil	Interagir avec les capteurs natifs de l'appareil comme l'accéléromètre	`StreamSubscription<dynamic> subscription = AccelerometerEvents.listen((AccelerometerEvent event) {`
		`  print(event);`
		`});`

Gestion des notifications	Planifiez et gérez les notifications localement sur l'appareil	`FlutterLocalNotificationsPlugin notificationsPlugin = FlutterLocalNotificationsPlugin();`
		`var androidDetails = AndroidNotificationDetails('channelId', 'channelName', 'channelDescription');`
		`var generalNotificationDetails = NotificationDetails(android: androidDetails);`
		`await notificationsPlugin.show(0, 'Test Title', 'Test Body', generalNotificationDetails);`

Gestion de l’état dans Flutter

Type	Description	Exemple de code

Gérer les changements d'état local des widgets dans un StatefulWidget	Fournit un moyen de gérer et de mettre à jour l'état local d'un widget	`setState(() {`
		`  _counter++;`
		`});`

InheritedWidget	Fournit un moyen de transmettre des données dans l'arbre des widgets et permet aux descendants de se reconstruire lorsque les données changent	`MyInheritedWidget(`
		`  data: counter,`
		`  child: ChildWidget()`
		`);`

GlobalKey et FormState	Utilisez GlobalKey pour accéder à l'état de n'importe où dans l'application, couramment utilisé avec des formulaires	`final _formKey = GlobalKey<FormState>();`
		`Form(`
		`  key: _formKey,`
		`  child: Column(`
		`  children: [`
		`  TextFormField(validator: (value) => value.isEmpty ? 'Cannot be empty' : null),`
		`  ElevatedButton(`
		`  onPressed: () {`
		`  if (_formKey.currentState.validate()) {`
		`  // Process data`
		`  }`
		`  },`
		`  child: Text('Submit')`
		`  )`
		`  ],`
		`  )`
		`);`

Fournisseur pour la gestion d'état	Un wrapper autour d'InheritedWidget pour faciliter et rendre plus efficace la gestion d'état	`ChangeNotifierProvider(`
		`  create: (context) => DataModel(),`
		`  child: Consumer<DataModel>(`
		`  builder: (context, dataModel, child) {`
		`  return Text('${dataModel.value}');`
		`  },`
		`  ),`
		`);`

Persistance Flutter avec Stockage Local

Type	Description	Exemple de Code

SharedPreferences	Idéal pour des données simples telles que les préférences et les paramètres de l'utilisateur	`final prefs = await SharedPreferences.getInstance();`
		`prefs.setInt('counter', 10);`
		`int counter = prefs.getInt('counter') ?? 0;`

SQLite	Idéal pour les données structurées et les requêtes complexes. Il fonctionne comme une base de données traditionnelle	
Fichiers	Lire/écrire des fichiers pour des documents ou des médias	
Packages Tiers	**Hive** : Une base de données NoSQL rapide pour Flutter. **LocalStorage** : Semblable au stockage local web, idéal pour stocker des données JSON.	
Initialiser le Stockage Local	Configurez la méthode de stockage choisie au début de votre application.	`final prefs = await SharedPreferences.getInstance();`

Opérations CRUD	Implémentez les fonctions Créer, Lire, Mettre à jour et Supprimer.	`// Saving data`
		`prefs.setInt('counter', 10);`
		`// Retrieving data`
		`int counter = prefs.getInt('counter') ?? 0;`
		`// Deleting data`
		`prefs.remove('counter');`

Sérialisation/Désérialisation	Encoder et décoder des données en utilisant JSON pour des structures complexes	`import 'dart:convert';`
		`Map userMap = jsonDecode(jsonString);`
		`var user = User.fromJson(userMap);`

Intégration avec la gestion d'état	Utilisez des outils de gestion d'état (par exemple, Provider, Riverpod) pour gérer les mises à jour du stockage local de manière dynamique dans l'interface utilisateur	`class UserProvider with ChangeNotifier {`
		`  User _user;`
		`  User get user => _user;`
		`  void loadUser() {`
		`  String userJson = storage.getItem('user');`
		`  _user = User.fromJson(jsonDecode(userJson));`
		`  notifyListeners();`
		`  }`
		`}`

Plugins dans Flutter

Type	Description	Exemple de code

Utilisation du plugin url_launcher	Ouvrir une URL dans le navigateur mobile depuis une application Flutter	`if (await canLaunch(url)) {`
		`  await launch(url);`
		`} else {`
		`  throw 'Could not launch $url';`
		`}`

Gestion de la compatibilité des plugins	Assurez-vous que les plugins sont compatibles avec la version de Flutter que vous utilisez	`// Typically handled in your pubspec.yaml by specifying compatible versions`
		`dependencies:`
		`  flutter:`
		`  sdk: flutter`
		`  url_launcher: ^6.0.3`

Plugin de caméra	Accédez à la caméra de l'appareil pour capturer des photos et des vidéos	`final cameras = await availableCameras();`
		`final firstCamera = cameras.first;`
		`final cameraController = CameraController(firstCamera, ResolutionPreset.medium);`
		`await cameraController.initialize();`

Plugin de Lecteurs Audio	Lire des fichiers audio et des flux dans Flutter avec le plugin Lecteurs Audio	`final player = AudioPlayer();` `await player.play(UrlSource('https://example.com/audio.mp3'));`

Plugin de Cartes Flutter	Intégrez des cartes dans vos applications Flutter en utilisant le plugin de cartes	`GoogleMap(`
		`  onMapCreated: _onMapCreated,`
		`  initialCameraPosition: CameraPosition(`
		`  target: LatLng(0.0, 0.0),`
		`  zoom: 10.0,`
		`  ),`
		`);`

Skills Network