Cookbook Développement Android 4

60 recettes de pros

Tout le catalogue sur
www.dunod.com

ÉDITEUR DE SAVOIRS

Dans la même collection :

Cookbook référencement Google, 80 recettes de pros
N. Ghouti-Terki, 216 pages, Dunod, 2013.

Cookbook développement iOS7, 60 recettes de pros
N. Zinovieff, M. Pybourdin, F. Impérial, D. Gosset, 216 pages, Dunod, 2013.

Cookbook Développement Android 4

60 recettes de pros

Damien Gosset
Fabrice Impérial
Marc Pybourdin
Nicolas Zinovieff

DUNOD

Maquette de couverture : Ici et ailleurs
Illustration de couverture : © Vlastimil Šesták-Fotolia.com
Maquette intérieure : Belle page

ISBN 978-2-10-070087-5

À PROPOS DE LA COLLECTION COOKBOOK

Informatique et cuisine

L'informatique, c'est parfois un peu comme la cuisine : il faut assembler un certain nombre d'ingrédients et d'actions selon un enchaînement très précis. C'est pourquoi quand un nouvel « ingrédient » apparaît, plutôt que de tâtonner seul dans son coin et risquer de rater son plat, il est beaucoup plus efficace de se référer à une recette rédigée par ceux qui ont déjà rencontré le problème et optimisé la solution. On peut ainsi travailler plus sereinement et adapter ensuite la recette à ses préférences.

Pour les développeurs aussi, rien ne vaut un bon conseil et une solution éprouvée pour gagner du temps et écrire du code propre et efficace. C'est le principe de la collection Cookbook : rassembler dans un seul ouvrage un certain nombre de « recettes » qui fournissent **des réponses concrètes à des problèmes précis.**

Comment ça marche, ces recettes ?

Chaque titre de la collection comporte plusieurs dizaines de recettes qui sont regroupées par thème (les chapitres) mais qui sont indépendantes les unes des autres pour pouvoir être consultées et utilisées de façon ponctuelle.

Les titres des recettes ont été rédigés de façon claire pour vous aider à trouver rapidement celle qui correspond à votre problème du moment. Si le titre n'est pas suffisamment explicite, un index détaillé en fin d'ouvrage vous aidera à aller directement au bon endroit grâce aux mots-clés.

Toutes les recettes sont structurées en trois parties :

Ce qu'il faut savoir : cette première partie rappelle le contexte et les connaissances de base relatives à cette question.

Ce qu'il faut faire **:** c'est le cœur de la recette qui fournit des explications précises et « prêtes à l'emploi ».

Ce qu'il ne faut pas faire **:** plus originale, cette dernière partie met en garde contre les erreurs les plus courantes et les impasses tentantes. Des retours d'expérience fort utiles pour éviter les pièges…

À qui s'adressent ces cookbooks ?

Aux développeurs bien sûr, aux testeurs, aux chefs de projet, aux étudiants et élèves ingénieurs… et d'une manière plus large à tous ceux qui développent des applis pour le travail ou pour le plaisir.

Les compléments en ligne

Des ressources complémentaires (code à télécharger…) ou de nouvelles recettes peuvent être téléchargées gratuitement sur le site Dunod, sur la page dédiée à l'ouvrage, en suivant le QR code imprimé en couverture ou en utilisant l'URL : www://dunod.com/contenus–complémentaire/9782100…

Partager le savoir

En règle générale les recettes sont faites pour être partagées, transmises et améliorées. Si vous avez des suggestions ou des recettes nouvelles à proposer n'hésitez pas à nous en faire part à l'adresse infos@dunod.com. Après validation, nous les intégrerons aux compléments en ligne, voire à la prochaine édition de la version papier.

Merci d'avance.

L'éditeur

SOMMAIRE

Ressources numériques

En complément de cet ouvrage, vous trouverez sur le site Dunod, à l'adresse suivante :

www.dunod.com/contenus-complementaires/9782100700875

1. Les sources des recettes traitées dans cet ouvrage ;

2. Une série de liens utiles vers des ressources Android.

Avertissement pour le lecteur

Nous avons incorporé à cet ouvrage de nombreux exemples de code directement réutilisables pour vos projets. Dans les recettes de l'ouvrage, ce code est commenté. La mise en page nous a imposé certains retours à la ligne. En cas de doute, n'hésitez pas à consulter les ressources numériques en ligne.

AVANT-PROPOS

Ces dernières années, les usages du numérique ont été révolutionnés avec l'arrivée de systèmes intégrant des capacités de communication avancées dans la plupart des périphériques. Aujourd'hui, qu'il s'agisse de téléviseurs, de téléphones et même de réfrigérateurs, on est quasiment sûr de trouver une version embarquant un système d'exploitation permettant d'utiliser des applications.

C'est autour de ce concept que le système Android est arrivé depuis près de 10 ans. Au départ au sein d'une start-up, puis rapidement racheté par Google, pour proposer un système ouvert, adaptable et personnalisable par n'importe quel constructeur désirant l'intégrer dans ses produits.

Android est aujourd'hui devenu un système mature et une réelle alternative à ce que peuvent proposer Apple, Microsoft et les constructeurs historiques de smartphones.

La philosophie autour d'Android rappelle beaucoup celle des environnements GNU/Linux et son modèle communautaire qui a permis de nombreuses avancées dans l'informatique. Cependant, Android s'est vu complété par de nombreuses surcouches intégrées par les constructeurs, soit pour ajouter des fonctionnalités dédiées à leurs appareils, soit pour faire évoluer l'expérience utilisateur.

Ainsi, Android, dans la lignée de son langage principal qu'est le Java, est un système qui mise sur l'interopérabilité et la portabilité du code. Cependant, il faut (comme c'est le cas avec Java) tenir compte de l'ensemble des spécificités liées aux différents périphériques où le programme pourra être exécuté.

Le développement sous Android suppose alors de prendre en compte un grand nombre de paramètres pour garantir le bon fonctionnement et le succès d'une application. En effet, le développeur ne connaît pas nécessairement le type d'appareil sur lequel sera exécuté son programme, ni les fonctionnalités ou les programmes embarqués par ce dernier. Dès lors, une attention toute particulière devra être portée à ces aspects pour garantir la meilleure expérience utilisateur possible.

L'objectif de cet ouvrage est de fournir à ses lecteurs un ensemble de recettes « prêtes à l'emploi » portant sur des problématiques récurrentes rencontrées par les développeurs. Nous avons souhaité proposer une approche puisée dans nos expériences professionnelles respectives et vous permettre ainsi de les adapter pour gagner du temps dans vos développements quotidiens.

L'ouvrage est ainsi structuré en grandes thématiques, chacune portant sur un aspect métier bien précis, qu'il s'agisse de l'utilisation des composants, des périphériques, de l'utilisation de service de Cloud, etc.

L'intégralité du code a été réalisé pour les environnements Android 4 et suivant. À l'heure actuelle, cette version est la plus utilisée sur l'ensemble du parc Android et surtout sur la majorité des périphériques des utilisateurs consommateurs d'applications.

Pour développer sur Android, de nombreux outils existent, et chaque développeur est libre de choisir les siens. Pour notre part, nous avons utilisé, tout au long de cet ouvrage, Éclipse ainsi que Android Studio.

Nous avons également souhaité agrémenter nos recettes de différents retours d'expérience dans les parties « Ce qu'il ne faut pas faire ». Ces parties comportent à la fois des exemples d'erreurs à ne pas reproduire mais également et surtout un rappel sur des oublis à éviter pour optimiser votre travail.

Pour compléter ces recettes, vous retrouverez sur www.dunod.com un espace en ligne dédié à l'ouvrage où nous vous proposons certaines mises à jour, quelques ajouts et différentes recettes supplémentaires.

Remerciements

Damien Gosset remercie chaleureusement sa famille ainsi que son équipe pour leur compréhension et leur enthousiasme motivant pendant la rédaction de cet ouvrage.

Fabrice Impérial souhaite remercier ses parents ainsi que ses grands-parents pour leur soutien.

Marc Pybourdin remercie affectueusement Aude et Nathan pour leur patience, leur compréhension et leur soutien.

PREMIÈRE PARTIE

FONDAMENTAUX DES APPLICATIONS

STOCKAGE ET RESTITUTION DES DONNÉES

CHAPITRE 1

RECETTE N° 01 Accéder à des données en local sur le périphérique

Ce qu'il faut savoir

La visualisation des données présuppose que l'on ait évidemment des données à afficher. Ces données peuvent être stockées à distance ou en local, l'objectif étant à chaque fois de garantir un accès rapide et fiable à ces dernières pour les présenter à l'utilisateur.

À la différence d'autres environnements, Android est un système dit « ouvert » : si l'on connaît le chemin d'un fichier et que l'on a les droits appropriés, il est possible d'y accéder, en lecture comme en écriture.

Android fournit par ailleurs un moyen d'exposer les données de son application via un « *content provider* ». Ce composant permet un accès en lecture/écriture aux fichiers de votre application.

Ce qu'il faut faire

Pour obtenir l'URL du répertoire où sont généralement stockées les images, nous utiliserons la méthode ci-dessous qui nous retourne l'URL du répertoire concerné.

```
File dir = getDir(Environment.DIRECTORY_PICTURES, Context.MODE_
PRIVATE);
```

Ce qu'il ne faut pas faire

Le stockage de données en local privilégie la disponibilité, compte tenu que les données de l'application sont directement accessibles sur le périphérique, ce qui garantit une rapidité d'accès.

Cependant, se pose la question de l'exactitude de ces dernières et ce, encore plus, lorsque les données sont utilisées par plusieurs utilisateurs et qu'un référentiel existe et est stocké à distance.

Le SDK nous fournit également d'autres options de stockage permettant d'éviter les problèmes de permissions ou d'optimisations :

- Le stockage dans les préférences (petit volume, si possible),
- Le stockage dans les données « privées » de l'application (accessibles en théorie à l'application elle-même seulement, et effacées lors de la désinstallation de l'application).

On appelle ces trois formes de stockage External, Preferences et Internal.

Les deux premiers ne nécessitent aucune permission particulière, le dernier, lui, a besoin des permissions `android.permission.READ_EXTERNAL_STORAGE` ou `android.permission.WRITE_EXTERNAL_STORAGE`.

Android propose également un stockage dans une base de données SQLite ainsi qu'au travers du réseau.

Le choix de la solution de stockage est à faire selon trois critères, que les différentes méthodes de stockage vont plus ou moins favoriser :

- **Exactitude** : la donnée doit être la plus récente et la plus pertinente possible,

- **Disponibilité** : la donnée doit être accessible lorsque l'on en a besoin et ce, quelque soit le contexte d'utilisation (absence de réseau par exemple),
- **Rapidité** : la donnée doit prendre un temps aussi réduit que possible pour être récupérée.

RECETTE N° 02 Stockage dans les Préférences de l'application

Ce qu'il faut savoir

Le stockage dans les Préférences fonctionne sur le modèle d'une table clefs/valeurs, et en mode transactionnel.

Par ailleurs, il ne sera pas nécessaire de se poser la question du stockage en lui-même, étant donné que le système le gérera pour nous.

Ce qu'il faut faire

Supposons une application qui a besoin d'accéder à deux variables :

- Le nom de l'utilisateur,
- S'il souhaite être en mode silencieux.

Le code permettant de réaliser cela sera :

```
// Récupérer l'objet préférences (PREFS_NAME détermine le nom
// de la table)
SharedPreferences settings = getSharedPreferences(PREFS_NAME,
0);

// Récupérer l'une des valeurs de la liste des préférences
boolean silent = settings.getBoolean("silentMode", false);

// Récupérer une autre valeur
String displayName = settings.getString("display", System.
    getProperty("user.name"));
```

Note : Dans notre exemple, la valeur système « user.name » sera probablement vide sur un appareil Android, mais est intéressante en tant qu'exemple d'accès aux propriétés globales de l'appareil.

Pour écrire dans les préférences, il faut fonctionner en mode transactionnel, à savoir que l'on fait les modifications dans un « contexte » (*editor* en anglais), et que les modifications ne seront prises en compte qu'une fois qu'elles auront été validées (*commit* en anglais).

```
// On crée un contexte d'édition
SharedPreferences.Editor editor = settings.edit();
// On renseigne les valeurs
editor.putBoolean("silentMode", true);
editor.putString("display", "Android dev");
// On valide les modifications
editor.commit();
```

Ce qu'il ne faut pas faire

Le deuxième paramètre des fonctions `get*` de Préférences sert à indiquer la valeur par défaut. Par ailleurs la méthode `System.getProperty` permet d'accéder à certaines préférences système.

Attention l'accès à ces dernières n'est pas généralisé et se fait bien souvent selon des méthodes spécifiques à chacune pour des raisons de sécurité.

RECETTE N° 03 Stockage Internal et External

Ce qu'il faut savoir

Le stockage dans les données et donc Internal ou External fonctionnent sur le système plus traditionnel de `java.io` et l'utilisation des classes `File` et `File*Stream`.

- `File` est une abstraction du système de fichier au niveau de la machine virtuelle Java : on peut parcourir l'arborescence, accéder et modifier les propriétés des fichiers et dossiers, les copier ou déplacer, etc.

➠ `File*Stream` sont des accesseurs sur le contenu des fichiers.

> **Note :** Le but de l'opération est d'ignorer la disposition physique des éléments dans le système ainsi que le type de stockage qui est utilisé.
> L'ensemble de l'arborescence est ainsi transposé dans un arbre d'objets Java que la machine virtuelle maintient pendant l'exécution des applications.

Dans le cas du stockage interne, les classes systèmes nous donnent directement accès au contenu des fichiers au travers de `*Stream` et donc la possibilité d'écrire ou de lire du texte, des objets java, etc.

Ce qu'il faut faire

Supposons une application qui a besoin d'accéder à un fichier « test » :

```
FileOutputStream fos = openFileOutput("test", Context.MODE_
PRIVATE);
FileInputStream fis = openFileInput("test");
```

Si le fichier n'existe pas, il sera alors créé.

Les différents modes de création sont :

- `MODE_PRIVATE` : on crée le fichier, et il n'est accessible que par l'application elle-même,
- `MODE_APPEND` : on ajoute au fichier déjà créé s'il existe, on le crée sinon,
- `MODE_WORLD_READABLE` : le fichier créé sera lisible par tout le monde,
- `MODE_WORLD_WRITABLE` : le fichier créé sera accessible en écriture à tout le monde.

Dans le cas du stockage External, il faut d'abord construire le `File` correspondant pour accéder au `*Stream`.

> **Conseil** : Il est également possible d'accéder directement au Stream en faisant :
>
> ```
> new FileInputStream(chemin);
> ```
>
> Attention, cela n'est pas recommandé car il n'y a aucune garantie sur le format du chemin de l'appareil cible.
>
> Le fait de construire le File avant permet de récupérer la racine et les séparateurs.