Localiser votre application

Android est utilisé sur un large éventail d'appareils dans de nombreuses régions. Pour toucher la plus vaste audience possible, votre application doit gérer le texte, les fichiers audio, les chiffres, la devise et les images de votre application conformément aux paramètres régionaux de vos utilisateurs.

Ce document décrit les bonnes pratiques à suivre pour localiser des applications Android.

Vous devez avoir une connaissance pratique de la programmation en Kotlin ou Java langage et vous familiariser avec Chargement des ressources Android (https://developer.android.com/guide/topics/resources/providing-resources?hl=fr) déclarer des éléments d'interface utilisateur en XML (https://developer.android.com/guide/topics/ui/declaring-layout?hl=fr), de développement, telles que cycle de vie de l'activité (https://developer.android.com/guide/components/activities/activity-lifecycle?hl=fr), et les principes généraux d'internationalisation et de localisation.

Il est recommandé d'utiliser le framework de ressources Android pour distinguer autant que possible les aspects localisés de votre application de ses principales fonctionnalités.

- Placez la plupart ou la totalité du *contenu* de l'interface utilisateur de votre application dans des fichiers de ressources, comme décrit sur cette page et dans la présentation des ressources de l'application (https://developer.android.com/guide/topics/resources/providing-resources?hl=fr).
- Le *comportement* de l'interface utilisateur, quant à lui, est déterminé par votre code basé sur Kotlin ou Java. Par exemple, si les utilisateurs saisissent des données qui doivent être formatées ou triées différemment en fonction des paramètres régionaux, vous pouvez utiliser Kotlin ou le langage de programmation Java pour gérer les données du point de vue de la programmation. Cette page n'aborde pas la localisation du code basé sur Kotlin ou Java.

Pour découvrir comment localiser des chaînes dans votre application, consultez Assurer la compatibilité avec différentes langues et cultures (https://developer.android.com/training/basics/supporting-devices/languages?hl=fr).

Présentation : Changer de ressources sous Android

Les ressources sont des chaînes de texte, des mises en page, des sons, des graphiques et toute autre donnée statique dont votre application Android a besoin. Une application peut inclure plusieurs ensembles de ressources, chacun personnalisé pour une configuration d'appareil différente. Lorsqu'un utilisateur exécute l'application, Android sélectionne et charge automatiquement les ressources qui correspondent le mieux à l'appareil.

Cette page porte sur la localisation et les paramètres régionaux. Pour obtenir une description complète du changement de ressources et de tous les types de configurations que vous pouvez spécifier, tels que l'orientation de l'écran ou le type d'écran tactile, consultez la section Fournir d'autres ressources (https://developer.android.com/guide/topics/resources/providing-resources?hl=fr#AlternativeResources).

Lorsque vous écrivez votre application, vous créez des ressources par défaut et d'autres ressources à utiliser dans votre application. Lorsque les utilisateurs exécutent votre application, le système Android sélectionne les ressources à charger en fonction des paramètres régionaux de l'appareil. Pour créer des ressources, vous devez placer les fichiers dans des sous-répertoires spécialement nommés du répertoire `res/` du projet.

Pourquoi les ressources par défaut sont-elles importantes ?

Lorsque l'application s'exécute dans une langue pour laquelle vous n'avez pas fourni de texte spécifique, Android charge les chaînes par défaut à partir de `res/values/strings.xml`. Si ce fichier par défaut est absent ou qu'il manque une chaîne nécessaire à votre application, l'application ne s'exécute pas et affiche une erreur. L'exemple ci-dessous illustre ce qui peut se produire lorsque le fichier texte par défaut est incomplet.

Exemple :

Le code basé sur Kotlin ou Java d'une application fait référence à deux chaînes seulement, `text_a` et `text_b`. L'application inclut un fichier de ressources localisé (`res/values-en/strings.xml`) qui définit `text_a` et `text_b` en anglais. Elle inclut également un fichier de ressources par défaut (`res/values/strings.xml`) qui comprend une définition pour `text_a`, mais pas pour `text_b` :

- Lorsque cette application est lancée sur un appareil dont les paramètres régionaux sont définis sur "Anglais", elle peut s'exécuter sans problème, car `res/values-en/strings.xml` contient les deux chaînes de texte requises.
- Toutefois, lorsque cette application est lancée sur un appareil configuré dans une langue autre que l'anglais, un message d'erreur et un bouton de fermeture forcée s'affichent. L'application ne charge pas.

Pour éviter cette situation, assurez-vous qu'un fichier `res/values/strings.xml` existe et qu'il définit toutes les chaînes nécessaires. Cette situation s'applique à tous les types de ressources, pas seulement aux chaînes : vous devez créer un ensemble de fichiers de ressources par défaut contenant toutes les ressources appelées par votre application, telles que des mises en page, des drawables ou des animations. Pour en savoir plus, consultez la section Tester les ressources par défaut (#test-for-default).

Utiliser des ressources pour la localisation

Cette section explique comment créer des ressources par défaut ainsi que d'autres ressources. Elle décrit également comment la priorité des ressources est définie et comment vous les référencez dans le code.

Créer des ressources par défaut

Saisissez le texte par défaut de l'application dans `res/values/strings.xml`. Pour ces chaînes, utilisez la langue par défaut, à savoir la langue parlée par la plupart des utilisateurs de votre application.

L'ensemble de ressources par défaut comprend également tous les drawables et les mises en page par défaut, et peut inclure d'autres types de ressources tels que les animations. Ces ressources se trouvent dans les répertoires suivants :

- `res/drawable/` : répertoire obligatoire contenant au moins un fichier image pour l'icône de l'application sur Google Play.
- `res/layout/` : répertoire obligatoire contenant un fichier XML qui définit la mise en page par défaut.
- `res/anim/` : obligatoire si vous avez des dossiers `res/anim-<qualifiers>`.
- `res/xml/` : obligatoire si vous avez des dossiers `res/xml-<qualifiers>`.
- `res/raw/` : obligatoire si vous avez des dossiers `res/raw-<qualifiers>`.

Conseil : Dans votre code, examinez chaque référence à une ressource Android. Assurez-vous qu'une ressource par défaut est définie pour chacune d'elles. Assurez-vous également que le fichier de chaînes par défaut est complet : un fichier de chaîne *localisé* peut contenir un sous-ensemble de chaînes, mais le fichier de chaîne *par défaut* doit toutes les contenir.

Créer d'autres ressources

La localisation d'une application sert avant tout à en proposer une version traduite en différentes langues. Dans certains cas, vous pouvez également fournir d'autres images, sons, mises en page et d'autres ressources spécifiques à vos paramètres régionaux.

Une application peut indiquer de nombreux répertoires `res/<qualifiers>/`, chacun avec des qualificatifs différents. Afin de créer une autre ressource pour des paramètres régionaux différents, vous devez utiliser un qualificatif qui précise une langue ou une combinaison langue-région. Le nom d'un répertoire de ressources doit être conforme au schéma de dénomination décrit dans la section Fournir d'autres ressources (https://developer.android.com/guide/topics/resources/providing-resources?hl=fr#AlternativeResources) pour que votre application puisse se compiler.

Exemple :

Supposons que la langue par défaut de votre application soit l'anglais et que vous souhaitiez localiser tout le texte de votre application en français, ainsi que tout le texte à l'exception du titre de l'application en japonais. Dans ce cas, vous allez créer trois fichiers `strings.xml`, chacun stocké dans un répertoire de ressources spécifique à chaque langue :

1. `res/values/strings.xml`
   contient le texte en anglais de toutes les chaînes utilisées par l'application, y compris le texte d'une chaîne nommée `title`.
2. `res/values-fr/strings.xml`
   contient le texte en français pour toutes les chaînes, y compris `title`.
3. `res/values-ja/strings.xml`
   contient le texte en japonais pour toutes les chaînes, *sauf* `title`.

Si votre code basé sur Kotlin ou Java fait référence à `R.string.title`, voici ce qui se passe au moment de l'exécution :

- Si l'appareil est configuré dans une langue autre que le français, Android charge `title` à partir du fichier `res/values/strings.xml`.
- Si l'appareil est configuré en français, Android charge `title` à partir du fichier `res/values-fr/strings.xml`.

Si l'appareil est configuré en japonais, Android recherche `title` dans le fichier `res/values-ja/strings.xml`. Toutefois, comme aucune chaîne de ce type n'est incluse dans ce fichier, Android utilise la valeur par défaut et charge `title` en anglais à partir du fichier `res/values/strings.xml`.

Quelles ressources sont prioritaires ?

Si plusieurs fichiers de ressources correspondent à la configuration d'un appareil, Android suit un ensemble de règles pour décider quel fichier utiliser. Parmi les qualificatifs pouvant être indiqués dans un nom de répertoire de ressources, les paramètres régionaux sont presque toujours prioritaires.

Exemple :

Supposons qu'une application inclue un ensemble de graphiques par défaut et deux autres ensembles de graphiques, chacun étant optimisé pour une configuration d'appareil différente :

- `res/drawable/`
  contient des graphiques par défaut.
- `res/drawable-small-land-stylus/`
  contient des graphiques optimisés pour un appareil adapté à un stylet comme instrument d'entrée et dispose d'un écran basse densité QVGA en mode paysage.
- `res/drawable-ja/`
  contient des graphiques optimisés pour le japonais.

Si l'application s'exécute sur un appareil configuré pour le japonais, Android charge les éléments graphiques à partir de `res/drawable-ja/`, même si l'appareil semble adapté à un stylet comme instrument d'entrée et disposer d'un écran basse densité QVGA en mode paysage.

Exception : Les seuls qualificatifs qui prévalent sur les paramètres régionaux lors du processus de sélection sont le mobile country code (MCC) et le mobile network code (MNC).

Exemple :

Supposons que vous rencontriez le problème suivant :

- Le code de l'application appelle `R.string.text_a`

  .
- Deux fichiers de ressources pertinents sont disponibles :
  - `res/values-mcc404/strings.xml`, qui inclut `text_a` dans la langue par défaut de l'application, dans ce cas l'anglais.
  - `res/values-hi/strings.xml`, qui inclut `text_a` en hindi.
- L'application s'exécute sur un appareil doté de la configuration suivante :
  - La carte SIM est connectée à un réseau mobile en Inde (MCC 404).

- La langue est définie sur hindi (`hi`).

Android charge `text_a` à partir de `res/values-mcc404/strings.xml` (en anglais), même si l'appareil est configuré pour l'hindi. En effet, lors du processus de sélection des ressources, Android privilégie une correspondance de MCC plutôt qu'une correspondance de langue.

Le processus de sélection n'est pas toujours aussi simple que ces exemples le suggèrent. Pour obtenir une description plus détaillée du processus, consultez la section Comment Android détermine-t-il la ressource la plus pertinente (https://developer.android.com/guide/topics/resources/providing-resources?hl=fr#BestMatch). Tous les qualificatifs sont décrits et indiqués par ordre de priorité dans la présentation des ressources de l'application (https://developer.android.com/guide/topics/resources/providing-resources?hl=fr#table2).

Faire référence à des ressources dans le code

Dans le code basé sur Kotlin ou Java, votre application fait référence à des ressources qui utilisent la syntaxe `R.resource_type.resource_name` ou `android.R.resource_type.resource_name`. Pour en savoir plus, consultez la section Accéder aux ressources de votre application (https://developer.android.com/guide/topics/resources/accessing-resources?hl=fr).

Gérer les chaînes en vue de leur localisation

Cette section décrit les bonnes pratiques à suivre pour gérer la localisation des chaînes.

Déplacer toutes les chaînes dans un fichier string.xml

Lorsque vous créez vos applications, ne codez en dur aucune chaîne. Déclarez plutôt toutes vos chaînes en tant que ressources dans un fichier `strings.xml` par défaut, ce qui facilitera la mise à jour et la localisation. Les chaînes du fichier `strings.xml` pourront ensuite être facilement extraites, traduites et réintégrées dans votre application (avec les qualificatifs appropriés) sans aucune modification du code compilé.

Si vous générez des images avec du texte, placez également ces chaînes dans `strings.xml`, puis générez à nouveau les images après la traduction.

Suivre les consignes Android pour les chaînes d'interface utilisateur

Lorsque vous concevez et développez une interface utilisateur, soyez particulièrement attentif à la façon dont vous vous adressez à vos utilisateurs. En règle générale, utilisez un style succinct, amical, mais bref, et veillez à sa cohérence dans toute l'interface utilisateur.

Assurez-vous de lire et de suivre les recommandations de Material Design pour le style d'écriture et le choix des mots (https://m1.material.io/style/writing.html#writing-language). Vos applications paraîtront ainsi plus soignées, et les utilisateurs pourront comprendre votre interface utilisateur plus rapidement.

Dans la mesure du possible, respectez toujours la terminologie Android standard, par exemple pour les éléments d'UI tels que la barre d'application, le menu d'options, la barre système et les notifications. Utiliser les termes Android correctement et uniformément facilite la traduction et aboutit à un meilleur produit fini pour les utilisateurs.

Fournir suffisamment de contexte pour les chaînes déclarées

Lorsque vous déclarez les chaînes dans votre fichier `strings.xml`, assurez-vous de décrire le contexte dans lequel la chaîne est utilisée. Ces informations sont précieuses pour le traducteur et permettent d'obtenir des traductions de meilleure qualité. Cela vous permet également de gérer vos chaînes plus efficacement.

Par exemple :

```
<!-- The action for submitting a form. This text is on a button that can fit 30 ch
<string name="login_submit_button">Sign in</string>
```

Envisagez de fournir des informations contextuelles telles que les suivantes :

- À quoi cette chaîne sert-elle ? Où et quand est-elle présentée à l'utilisateur ?
- Où figure-t-elle dans la mise en page ? Par exemple, la traduction des boutons est moins flexible que celle des zones de texte.

Marquer les parties du message qui ne doivent pas être traduites

Les chaînes contiennent souvent du texte qui n'est pas destiné à être traduit dans d'autres langues. Il peut s'agir d'un fragment de code, d'un espace réservé à une valeur, d'un symbole spécial ou d'un nom. Lorsque vous préparez vos chaînes en vue de leur traduction, repérez, puis signalez le texte qui doit rester tel quel sans être traduit, afin que le traducteur n'y touche pas.

Pour marquer le texte qui ne doit pas être traduit, utilisez une balise d'espace réservé `<xliff:g>`. Voici un exemple de balise qui indique que le texte `"%1$s"` ne doit pas être modifié pendant la traduction (sans quoi le message perdrait son sens) :

```
<string name="countdown">
  <xliff:g id="time" example="5 days">%1$s</xliff:g> until holiday
</string>
```

Lorsque vous placez une balise d'espace réservé, ajoutez un attribut "id" qui explique à quoi sert l'espace réservé. Si l'espace réservé est ensuite remplacé par une valeur dans l'application, assurez-vous de fournir un attribut d'exemple pour clarifier l'utilisation prévue.

Autres exemples de balises d'espace réservé :

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
<!-- Example placeholder for a special Unicode symbol -->
<string name="star_rating">Check out our 5
    <xliff:g id="star">\u2605</xliff:g>
</string>
<!-- Example placeholder for a URL -->
<string name="app_homeurl">
    Visit us at <xliff:g
    id="application_homepage">http://my/app/home.html</xliff:g>
</string>
<!-- Example placeholder for a name -->
<string name="prod_name">
    Learn more at <xliff:g id="prod_gamegroup">Game Group</xliff:g>
</string>
<!-- Example placeholder for a literal -->
<string name="promo_message">
    Please use the "<xliff:g id="promotion_code">ABCDEFG</xliff:g>" to get a disco
</string>
...
</resources>
```

Checklist pour la localisation

Pour une présentation complète du processus de localisation et de distribution d'une application Android, consultez la section Traduire et localiser votre application (https://support.google.com/googleplay/android-developer/answer/9844778?_ga=2.188605422.756657181.1679783723-1569914833.1659556491&hl=fr)
.

Conseils de localisation

Suivez ces conseils pour localiser votre application.

Concevoir votre application pour qu'elle s'adapte à tous les paramètres régionaux

Vous ne pouvez pas prévoir sur quel type d'appareil un utilisateur exécutera votre application. L'appareil peut disposer de matériel que vous n'aviez pas envisagé, ou il peut être configuré sur un paramètre régional que vous n'aviez pas prévu ou que vous ne pouvez pas tester. Concevez votre application pour qu'elle fonctionne normalement ou échoue en douceur, quel que soit l'appareil sur lequel elle s'exécute.

Important : Assurez-vous que votre application inclut un ensemble complet de ressources par défaut. Ajoutez les dossiers **`res/drawable/`** et **`res/values/`** contenant toutes les images et le texte nécessaires à votre application, sans modificateur supplémentaire dans le nom des dossiers.

Si une application manque ne serait-ce que d'une ressource par défaut, elle ne sera pas exécutée sur un appareil dont les paramètres régionaux sont incompatibles. Par exemple, si le fichier `res/values/strings.xml` par défaut ne comporte pas une chaîne spécifique dont l'application a besoin, l'utilisateur verra une erreur et un bouton de fermeture forcée lorsque l'application est exécutée dans une langue non prise en charge et qu'elle tente de charger `res/values/strings.xml`.

Pour en savoir plus, consultez la section Tester les ressources par défaut (#test-for-default).

Concevoir une mise en page flexible

Si vous devez réorganiser votre mise en page pour qu'elle s'adapte à une langue donnée, vous pouvez créer une autre mise en page pour cette langue, par exemple `res/layout-de/main.xml` pour une mise en page en allemand. Toutefois, cela peut rendre la gestion de votre application plus difficile. Il est préférable de créer une seule mise en page plus flexible.

Un autre problème typique est une langue qui requiert une mise en page différente. Par exemple, vous pouvez avoir un formulaire de contact qui comprend deux champs de nom lorsque l'application s'exécute en japonais, mais trois champs de nom lorsque l'application s'exécute dans une autre langue. Vous pouvez gérer cela de deux manières :

- Créez une mise en page avec un champ que vous pouvez activer ou désactiver par programmation, selon la langue.
- Assurez-vous que la mise en page principale propose une autre mise en page incluant le champ modifiable. La deuxième mise en page peut avoir différentes configurations selon

les langues.

Ne pas créer plus de fichiers de ressources et de chaînes de texte que nécessaire

Vous n'avez probablement pas besoin de créer une alternative spécifique aux paramètres régionaux pour chaque ressource de votre application. Par exemple, la mise en page définie dans le fichier `res/layout/main.xml` peut fonctionner avec n'importe quel paramètre régional. Dans ce cas, il n'est pas nécessaire de créer d'autres fichiers de mise en page.

De plus, vous n'aurez peut-être pas besoin de créer un texte alternatif pour chaque chaîne. Prenons l'exemple suivant :

- La langue par défaut de votre application est l'anglais américain. Chaque chaîne utilisée par l'application est définie dans `res/values/strings.xml` en suivant l'orthographe de l'anglais américain.
- Pour quelques expressions importantes, vous souhaitez proposer l'orthographe en anglais britannique. Vous voulez que ces autres chaînes apparaissent lorsque votre application s'exécute sur un appareil au Royaume-Uni.

Pour ce faire, créez un petit fichier nommé `res/values-en-rGB/strings.xml` qui n'inclut que les chaînes qui doivent être différentes lorsque l'application s'exécute au Royaume-Uni. Pour les autres, l'application utilisera les valeurs par défaut et ce qui est défini dans `res/values/strings.xml`.

Utiliser l'objet Android Context pour rechercher manuellement des paramètres régionaux

Vous pouvez rechercher les paramètres régionaux à l'aide de l'objet `Context` (https://developer.android.com/reference/android/content/Context?hl=fr) mis à disposition par Android, comme illustré dans l'exemple suivant :

Kotlin Java (#java)
(#kotlin)

```
val primaryLocale: Locale = context.resources.configuration.locales[0]
val locale: String = primaryLocale.displayName
```

Utiliser le service de traduction d'applications

Le service de traduction d'applications (https://support.google.com/l10n/answer/6359997?hl=fr) est intégré à la Play Console (https://support.google.com/l10n/answer/6341304?hl=fr). Il vous permet d'obtenir un devis instantanément et de passer commande auprès d'une agence de traduction.

Vous pouvez commander des traductions dans une ou plusieurs langues pour les chaînes de l'interface utilisateur de l'application, le texte de la fiche Play Store, les noms des achats via l'application et le texte de la campagne publicitaire.

Tester les applications localisées

Testez votre application localisée sur un appareil ou à l'aide d'Android Emulator. Testez en particulier votre application pour vous assurer que toutes les ressources par défaut nécessaires sont incluses.

Tester l'application sur un appareil

Gardez à l'esprit que l'appareil sur lequel vous effectuez le test peut être très différent de celui que les clients utilisent ailleurs. Les paramètres régionaux disponibles sur votre appareil peuvent différer de ceux disponibles sur d'autres appareils. En outre, la résolution et la densité de l'écran de l'appareil peuvent différer, ce qui peut affecter l'affichage des chaînes et des drawables dans votre interface utilisateur.

Pour modifier les paramètres régionaux ou la langue d'un appareil, utilisez l'application Paramètres.

Tester l'application sur un émulateur

Pour en savoir plus sur l'utilisation de l'émulateur, consultez la section Exécuter des applications sur Android Emulator (https://developer.android.com/studio/run/emulator?hl=fr).

Créer et utiliser des paramètres régionaux personnalisés

Les paramètres régionaux "personnalisés" correspondent à une combinaison langue/région que l'image système Android n'accepte pas de manière explicite. Vous pouvez tester l'exécution de votre application dans un paramètre régional personnalisé en créant un paramètre régional personnalisé dans l'émulateur. Pour ce faire, vous disposez de deux méthodes :

- Utilisez l'application des paramètres régionaux personnalisés, accessible depuis l'onglet "Application". Une fois les paramètres régionaux créés, vous pouvez les appliquer en appuyant de manière prolongée sur leur nom.
- Appliquez un paramètre régional personnalisé à partir de l'interface système `adb`, comme décrit dans la section ci-dessous.

Lorsque vous définissez l'émulateur sur des paramètres régionaux qui ne sont pas disponibles dans l'image système Android, le système s'affiche dans sa langue par défaut. Toutefois, votre application se localise correctement.

Modifier les paramètres régionaux de l'émulateur à partir de l'interface système adb

Pour modifier les paramètres régionaux dans l'émulateur à l'aide de l'interface système `adb`, procédez comme suit :

1. Choisissez les paramètres régionaux que vous souhaitez tester et déterminez leur balise de langue BCP-47, telle que `fr-CA` pour le français canadien.
2. Lancez un émulateur.
3. Depuis une interface système de ligne de commande sur l'ordinateur hôte, exécutez la commande
   `adb shell`
   ou, si vous disposez d'un appareil connecté, indiquez que vous souhaitez utiliser l'émulateur en ajoutant l'option `-e` :
   `adb -e shell`.
4. Lorsque l'invite d'interface système `adb` (`#`) s'affiche, exécutez la commande suivante :
   `setprop persist.sys.locale [`*`BCP-47 language tag`*`];stop;sleep 5;start`
   Remplacez les parties entre crochets par les codes appropriés définis à l'étape 1.

   Par exemple, pour effectuer un test en français canadien :
   `setprop persist.sys.locale fr-CA;stop;sleep 5;start`

Cela entraîne le redémarrage de l'émulateur. Lorsque l'écran d'accueil s'affiche à nouveau, relancez votre application. L'application se lancera avec les nouveaux paramètres régionaux.

Tester les ressources par défaut

Pour tester si une application inclut toutes les ressources de chaîne dont elle a besoin, procédez comme suit :

1. Définissez l'émulateur ou l'appareil sur une langue non compatible avec votre application. Par exemple, si l'application comporte des chaînes en français dans `res/values-fr/`, mais qu'aucune chaîne en espagnol n'existe dans `res/values-es/`, définissez les paramètres régionaux de l'émulateur sur l'espagnol. Vous pouvez utiliser l'application des paramètres régionaux personnalisés pour définir l'émulateur sur des paramètres régionaux non compatibles.
2. Exécutez l'application.

3. Si l'application affiche un message d'erreur et un bouton de fermeture forcée, elle peut rechercher une chaîne indisponible. Assurez-vous que votre fichier `res/values/strings.xml` inclut une définition pour chaque chaîne utilisée par l'application.

Si le test réussit, répétez-le pour d'autres types de configurations. Par exemple, si l'application comporte un fichier de mise en page nommé `res/layout-land/main.xml`, mais ne contient pas de fichier nommé `res/layout-port/main.xml`, définissez l'émulateur ou l'appareil en mode portrait et regardez si l'application s'exécute.

Suivant
Test your app with pseudolocales →
(https://developer.android.com/guide/topics/resources/pseudolocales?hl=fr)

Dernière mise à jour le 2025/07/27 (UTC).