# Melodienet\_android\_client

**ANALYSE ORGANIQUE**

Table des matières

[Melodienet\_android\_client 1](#_Toc482976500)

[1.1 Schéma de fonctionnement général 2](#_Toc482976501)

[1.2 Parametres 3](#_Toc482976502)

[1.2.1 SettingFragment 3](#_Toc482976503)

[1.2.2 VueParametres 3](#_Toc482976504)

[1.3 Objets utilises 4](#_Toc482976505)

[1.3.1 Structure de données des informations fournies par le service Web 4](#_Toc482976506)

[1.4.2 CryptageMdp(package securite) : 5](#_Toc482976507)

[1.4.3 ConstructeurUrl(package connexionserviceweb) : 6](#_Toc482976508)

[1.4 Asynctask 8](#_Toc482976509)

[1.5 Description par activite 10](#_Toc482976510)

[1.6.1 ActiviteLancement(package activites) 10](#_Toc482976511)

[1.6.2 ActiviteAuthent(package activites) 11](#_Toc482976512)

[1.6.3 ActiviteMenu(package activities) 13](#_Toc482976513)

[1.6.4 ActiviteProductionParam(package activities) 13](#_Toc482976514)

[1.6.5 ActiviteMachinesParam(package activities) 14](#_Toc482976515)

[1.6.6 ActiviteModesMarche(package activities) 15](#_Toc482976516)

[1.6.6 ActiviteProduction(package activities) 16](#_Toc482976517)

[1.6 Adapteurs 19](#_Toc482976518)

[1.7.1 AdapteurProduits(package production) : 19](#_Toc482976519)

[1.7.2 AdapteurModesMarche(package modemarche) : 20](#_Toc482976520)

[1.7 Gestion des langues 21](#_Toc482976521)

## Schéma de fonctionnement général

En bleu : ce qui est interne à l’application

En rouge : ce qui est externe à l’application

Get Running Mode()

Get Line()+ Get Station()

Line + station

Get Line()

Get (« Date »)Production()

Line + Production Date

WebService

Parameters

Send the phone langage

Check password

DataBase

## Communication service web - pRINCIPE DE FONCTIONNEMENT

Il existe à ma connaissance deux moyens de communiquer avec un service Web sous Android :

**Android Asynchronous Http Client**

**HttpURLConnection**

Android n’autorise pas plus de 5 secondes de traitement et donc de blocage de l’interface utilisateur. C’est pour cela que quelque soit la technologie de communication utilisée pour communiquer avec le service Web, il faut encapsuler le traitement dans une AsyncTask. Il s’agit d’un thread indépendant qui va ici lire et écrire sur le Service Web en ne bloquant donc pas l’interface utilisateur.

## Parametres

### SettingFragment

Variables :

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| urlPref | EditTextPreference | EditText de saisie de l’URL |
| switchPref | SwitchPreference | Switch permettant de choisir si nous souhaitons utiliser l’URL de base dans le code ou celui entré par l’utilisateur dans « urlPref ». |

Algorithme :

A chaque changement d’état du switch :

-Si le switch est sur Off, on met une valeur par défaut pour « urlPref » avec setDefaultValue(URL\_CONSTANTE) et on met cette valeur dans « urlPref » avec setText(URL\_CONSTANTE).

-Si le switch est sur On, on met dans urlPref la valeur courante qu’il contient actuellement avec : setText(urlPref.getText()).

### VueParametres

Variables :

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| fm | FragmentManager | Gestionnaire de fragments |
| ft | FragmentTransaction | Permet d’effectuer des actions à partir de fragments à l’aide d’un FragmentManager |

## Objets utilises

### Structure de données des informations fournies par le service Web

#### **ObjetAuthent**(package securite) :

Constructeur ? OUI

**Attributs :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom d’attribut** | **Type** | **Muni d’un setter ?** | **Muni d’un getter ?** |
| name | String | OUI | OUI |
| agreement | Booléen | OUI | OUI |

#### **Ligne**(package général) :

Constructeur ? OUI

**Attributs :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom d’attribut** | **Type** | **Muni d’un setter ?** | **Muni d’un getter ?** |
| lineNumber | int | OUI | OUI |
| lineLabel | String | OUI | OUI |

#### **Production**(package général) :

Constructeur ? OUI

**Attributs :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom d’attribut** | **Type** | **Muni d’un setter ?** | **Muni d’un getter ?** |
| LineNumber | int | OUI | OUI |
| reference | String | OUI | OUI |
| goodPartsQty | int | OUI | OUI |
| badPartsQty | int | OUI | OUI |

#### **Cellule**(package général) :

Constructeur ? OUI

**Attributs :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom d’attribut** | **Type** | **Muni d’un setter ?** | **Muni d’un getter ?** |
| cellLabel | String | OUI | OUI |
| cellNumber | int | OUI | OUI |

#### **ModeMarche**(package général) :

Constructeur ? OUI

**Attributs :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom d’attribut** | **Type** | **Muni d’un setter ?** | **Muni d’un getter ?** |
| cellNumber | int | OUI | OUI |
| modeColor | int | OUI | OUI |
| ModeLabel | String | OUI | OUI |
| RunningMode | Int | OUI | OUI |

### 1.4.2 CryptageMdp(package securite) :

Constructeur ? NON, méthode globale.

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| *Static* hashMdp | String : mdp | String | Applique un hashage de type « md5 » au mot de passe passé en paramètre (voir algorithme) |

*Algorithme :*

*-On crée un string mdpMd5 initialisé à nul*

*-Si le mot de passe passer en paramètre est nul on retourne un hashage nul*

*-On crée un objet de type « MessageDigest » en précisant que l’algorithme utilisé sera de type MD5.*

*-on charge le mot de passe dans cet objet*

*-On crée un String à partir d’un BigInteger contenant le hashage md5 du mot de passe via l’appel à la méthode « digest() »*

update(mdp)

crée

MessageDigest

CryptageMdp

Digest()

ToString()

### 1.4.3 ConstructeurUrl(package connexionserviceweb) :

Constructeur ? OUI

**Attributs :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom d’attribut** | **Type** | **Muni d’un setter ?** | **Muni d’un getter ?** |
| baseUrl | String | NON | NON |

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| obtenirLogin | String : utilisateur  String : motDePasse | String | Retourne L’URL permettant d’obtenir le résultat de l’authentification à partir de l’URL de base et des deux chaines passées en paramètre. |
| obtenirListeCelulles | String : numLigne | String | Retourne l’URL permettant d’obtenir la liste des cellules en fonction de l’URL de base et du numéro de ligne passé en paramètre. |
| obtenirListeLignes | Vide | String | Retourne l’URL permettant d’obtenir la liste des lignes en fonction de l’URL de base |
| obtenirModesMarche | String : numLigne  String : numCellules | String | Retourne l’URL permettant d’obtenir les modes de marche en fonction de l’URL de base, du numéro de ligne et du numéro de cellule. |
| obtenirProdHeure | String : numLigne | String | Retourne l’URL permettant d’obtenir la production horaire en fonction de l’URL de base et du numéro de ligne. |
| obtenirProdJour | String : numLigne | String | Retourne l’URL permettant d’obtenir la production journalière en fonction de l’URL de base et du numéro de ligne. |
| obtenirProdSemaine | String : numLigne | String | Retourne l’URL permettant d’obtenir la production hebdomadaire en fonction de l’URL de base et du numéro de ligne. |
| envoyerLang | String lang | String | Retourne l’URL permettant d’envoyer la langue au service web en fonction de l’URL de base et du paramètre. |

Construction avec l’URL de base + choix de la méthode

Retourne l’URL spécifique

## Asynctask

#### **LectureDonnéesWeb**(package connexionserviceweb) :

**doInBackground :**

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom d’attribut** | **Type** | **Valeur initiale** |
| connexionService | HttpURLConnection | Nulle |
| reponse | String | Chaine vide |
| lecteur | BufferedReader | Nulle |
| fluxEntree | InputStream | Non définie |
| tampon | StringBuffer | Non définie |

**Algorithme :**

-On crée un URL avec le string passé en paramètre.

-On ouvre la connection

-On la définie comme une connection de type « GET » et on configure le « Timeout »

-On se connecte

-On crée un InputStream sur la connexion

-On lit sur cet inputStream et on met les informations lues sur un StringBuffer.

-On ferme la connexion et le BufferedReader

-On retourne la chaine envoyée par le service web

Crée un URL à partir du String + openConnection

URL en format String

Connect() + getInputStream

Retourne la chaine résultat provenant du service web

*InputStream*

Crée

Rempli un StringBuffer

Crée

#### **EcritureLangTel** (package connexionserviceweb) :

**doInBackground :**

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom d’attribut** | **Type** | **Valeur initiale** |
| connexionService | HttpURLConnection | Nulle |
| fluxSortie | OutputStreamWriter | Non définie |
| resultat | String | Chaine vide |

**Algorithme :**

-On crée un URL avec le string passé en paramètre.

-On récupère le code de la langue dans le second paramètre

-On ouvre la connection

-On la définie comme une connection de type « POST » et on configure le « Timeout »

-On se connecte

-On crée un OutputStreamWriter sur la connexion

-On écrit la langue issue du second paramètre sur cet OutputStreamWriter

-On ferme l’OutputStreamWriter et la connexion

-On retourne le résultat de la requête « POST »

Crée un URL à partir du String + openConnection

URL en format String

Connect() + getOutputStream

OutputStream

write(langue)

Retourne la chaine confirmant le bon déroulement de la commande POST

## Description par activite

### 1.6.1 ActiviteLancement(package activites)

**Fichier xml associé :**  activite\_lancement.xml

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| ivFsa | ImageView | Contient l’image View de nom « logoView » |
| ivMelodie | ImageView | Contient l’image View de nom « melodieView » |

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| genereAnim | Animation, ImageView | vide | Applique une animation à une image |
| quitteAnim | Animation, ImageView | vide | Quitte l’activité « ActiviteLancement » dès la fin de l’animation sur l’image |

### 1.6.2 ActiviteAuthent(package activites)

**Fichier xml associé :**  activite\_authent.xml

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| utilisateurActuelEditText | EditText | Contient la vue ou l’utilisateur entre son nom d’utilisateur |
| motPasseEditText | EditText | Contient la vue ou l’utilisateur entre son mot de passe |
| authentBoutton | Button | Contient le boutton qui permet à l’utilisateur de se connecter |
| utilisateurActuel | String | Contient le nom d’utilisateur entré par l’utilisateur |
| motPasseActuel | String | Contient le mot de passe entré par l’utilisateur |

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| chargeParam | vide | String | Retourne le String de l’URL venant des paramètres de l’application |
| envoiLangue | ConstructeurUrl | Vide | Envoie la langue courante du téléphone au service web |
| authentification | vide | vide | Vérifie les informations entrées pour procéder à l’authentification |
| quitter | View | vide | Quitte l’activité courante et donc, ici, l’application |
| allerAuxParam | View | vide | Va aux paramètres de l’application |

**Authentification**, *algorithme* :

-Si l’un des deux champs est vide, informer l’utilisateur qu’il faut les remplir.

-Charge l’URL des paramètres (méthode chargeParam).

-Applique le hashage MD5 au mot de passe entré.

-Construit l’URL permettant d’obtenir le résultat de l’authentification.

-Stocke le résultat de la requête au service Web dans une chaine.

-Converti cette chaine constituée d’un seul objet dans une instance de la classe « ObjetIdentification » via la librairie Gson.

-Si l’authentification est bonne, on va à l’activité « ActiviteMenu », Sinon on demande à l’utilisateur de recommencer.

Paramètres

URL de base

Résultat de l’authentification

URL d’authentification

Service Web

URL d’authentification

URL de base

----------------

### 1.6.3 ActiviteMenu(package activities)

**Fichier xml associé :**  activite\_menu.xml

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| allerSuiviMachine | View | Vide | Va à l’activité « ParamModeMarcheActivity » |
| allerSuiviProd | View | Vide | Va à l’activité « ParamProductionActivity » |
| quitter | View | Vide | Quitte l’activité courante et donc, ici, cela représente une déconnexion. |
| allerAuxParam | View | Vide | Va aux paramètres de l’application |

### 1.6.4 ActiviteProductionParam(package activities)

**Fichier xml associé :**  activite\_production\_param.xml

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| choixProdTab | TypeProduit[] | Contient les différents types de production (horaires, journalière, hebdommadaire) |
| listeLignes | ArrayList<Ligne> | Contient la liste des numéros de lignes récupérés sur le service Web |
| spinnerProd | Spinner | Spinner du type de production |
| spinnerLigne | Spinner | Spinner des numéros de ligne |
| productionChoisie | String | Contient le type de production choisi |
| ligneChoisie | String | Contient la ligne choisie |
| urlBase | String | Contient l’URL de base |
| requeteSpinner | ConstructeurUrl | Constructeur de requêtes du spinner des lignes |

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| chargeParam | vide | String | Retourne le String de l’URL venant des paramètres de l’application |
| rempliSpinnerLigne | Vide | Vide | Rempli le spinner avec les valeurs de ligne retournées par le service web |
| quitter | View | Vide | Quitte l’activité courante et donc, ici, cela permet de revenir au menu. |
| allerAProduction | View | Vide | Va à l’activité Production qui affichera les informations correspondants aux choix faits par l’utilisateur dans les deux Spinner. |

### 1.6.5 ActiviteMachinesParam(package activities)

**Fichier xml associé :**  activite\_machines\_param.xml

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| listeCellules | ArrayList<Cellule> | Contient la liste des numéros de celulles récupérés sur le service Web |
| listeLignes | ArrayList<Ligne> | Contient la liste des numéros de lignes récupérés sur le service Web |
| spinnerCelulles | Spinner | Spinner contenant les numéros de celulles en fonction de la ligne choisie |
| spinnerLigne | Spinner | Spinner contenant les numéros de ligne |
| celulleChoisie | String | Contient la celulle choisie |
| ligneChoisie | String | Contient la ligne choisie |
| urlBase | String | Contient l’URL de base |
| requeteSpinner | ConstructeurUrl | Constructeur de requêtes des Spinner |
| gson | Gson | Objet permettant de convertir la chaine reçue en une arrayList |

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| chargeParam | vide | String | Retourne le String de l’URL venant des paramètres de l’application |
| rempliSpinnerLignes | ConstructeurUrl : requeteSpinnerLigne  Gson : objetGson | Vide | Rempli le spinner avec les valeurs de ligne retournées par le service web |
| quitter | View | Vide | Quitte l’activité courante et donc, ici, cela permet de revenir au menu. |
| allerAModesMarche | View | Vide | Va à l’activité produit qui affichera les informations correspondants aux choix faits par l’utilisateur dans les deux Spinner. |

### 1.6.6 ActiviteModesMarche(package activities)

**Fichier xml associé :**  activite\_modes\_marche.xml

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| vueRafraichissement | SwipeRefreshLayout | Stocke la vue de rafraichissement de la liste |
| vueListe | ListView | Contient la liste visible des modes de marches |
| listeModes | ArrayList<RunningMode> | Liste contenant les modes de marches obtenus à partir du service Web |
| adapteurModesMarche | AdapteurModesMarche | Adapteur servant à remplir la liste de modes de marche |
| gson | Gson | Outil permettant la conversion du string envoyé par le service Web en une liste d’objets exploitable |
| constructRequetes | ConstructeurUrl | Objet permettant de construire les requêtes utilisée pour chercher des informations sur le service Web. |
| informationEntrante | String | Chaine retournée par le service Web |

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| rempliVue | Vide | Vide | Rempli la liste de modes de marche |
| genereModesMarches | Vide | Vide | Rempli la liste « listeModes » de modes de marche récupéré sur le service Web |
| rafraichissement | Vide | Vide | Thread permettant de rafraichir la liste |
| quitter | View | Vide | Quitte l’activité courante et donc, ici, cela nous ramène sur « ActiviteMachineParam ». |
| chargeParam | vide | String | Retourne le String de l’URL venant des paramètres de l’application |

### 1.6.6 ActiviteProduction(package activities)

**Fichier xml associé :**  activite\_modes\_marche.xml

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| vueRafraichissement | SwipeRefreshLayout | Stocke la vue de rafraichissement de la liste |
| vueListe | ListView | Contient la liste visible des instances de « Product» |
| listeProduits | ArrayList< Product > | Liste contenant les instances de « Product» obtenus à partir du service Web |
| adapteurProduction | AdapteurProduction | Adapteur servant à remplir la liste d’objets de type « Product» |
| gson | Gson | Outil permettant la conversion du string envoyé par le service Web en une liste d’objets exploitable |
| constructRequetes | ConstructeurUrl | Objet permettant de construire les requêtes utilisée pour chercher des informations sur le service Web. |
| informationEntrante | String | Chaine retournée par le service Web |

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| rempliVue | Vide | Vide | Rempli la liste de produits |
| genereProduction | Vide | Vide | Rempli la liste « listeProduits» de modes de marche récupéré sur le service Web |
| rafraichissement | Vide | Vide | Thread permettant de rafraichir la liste |
| quitter | View | Vide | Quitte l’activité courante et donc, ici, cela nous ramène sur « ActiviteProductionParam». |
| chargeParam | vide | String | Retourne le String de l’URL venant des paramètres de l’application |

**Schéma de fonctionnement de ActiviteProduction**

Get the baseURL

Base URL+parameter

ConstructeurUrl

ActiviteProduction *Build the ArrayList thanks to the Response String*

Web Service

Interne à l’AsyncTask

Return the Web Service response in String

Give the Input Stream

Buffered Reader

+

String Buffer

Return InputStream

Connect with the URL

Populate listView with the ArrayList

List View

Response String

Usable URL

AsyncTask LectureDonnéesWeb

*Build the response String with getting data in the Web Service*

Usable URL

## Adapteurs

### 1.7.1 AdapteurProduits(package production) :

Constructeur ? OUI. Deux paramètres, l’activité courante et la liste d’objets

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| mapProduits | HashMap<View, Product> | Stockage de chaque produit en lui associant une vue |

Méthodes :

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| getView | Int : position  View : convertView  ViewGroup : parent | View | Rempli physiquement la classe VueProduits de produits |

**Classes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Attributs** | **Type** | **Description** |
| VueProduits | TextView : ref  TextView : goodNum  TextView : badNum | class | Objet constituant la listView de la production |

### 1.7.2 AdapteurModesMarche(package modemarche) :

Constructeur ? OUI. Deux paramètres, l’activité courante et la liste d’objets

**Variables :**

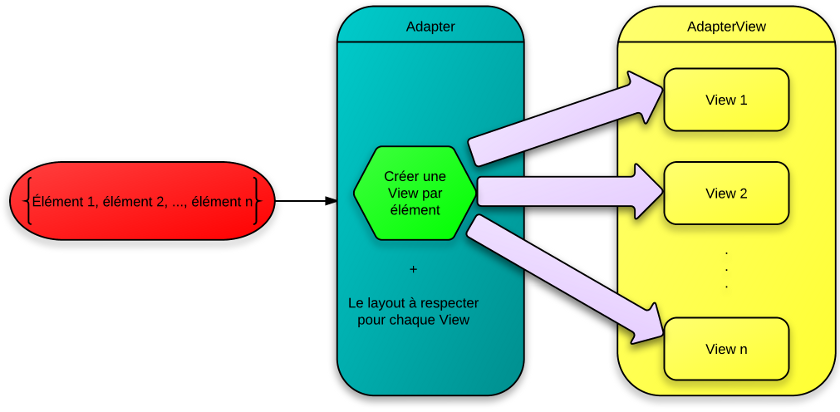
|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| mapModes | HashMap<View, ModeMarche > | Stockage de chaque mode de marche en lui associant une vue |

Méthodes :

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| getView | Int : position  View : convertView  ViewGroup : parent | View | Rempli physiquement la classe VueModes de modes de marche |

**Classes :**

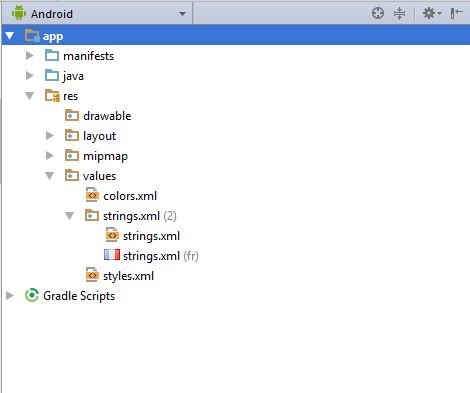
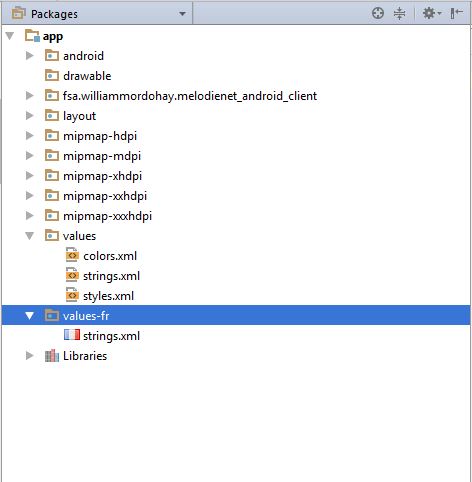
|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Attributs** | **Type** | **Description** |
| VueModes | TextView : num  TextView : details  TextView : couleur  ImageView :imageMode | class | Objet constituant la listView des modes de marche |



## Gestion des langues

Rajouter une langue :

* Création d’un « Android ressources directory » dans values avec la valeur « values-fr » pour le français. Dans le cas d’une autre langue, il suffit de remplacer le code fr par le code de la langue voulue.
* Dedans, on crée un fichier « strings.xml » qui contiendra toutes les chaines de caractères utilisées par l’application pour la langue en question



L’application est entièrement fonctionnelle pour l’anglais et le français. L’anglais est la langue définie par défaut, ce qui signifie que tout téléphone qui est dans une autre langue que le français démarrera automatiquement la version anglaise de l’application.

Chaque fois qu’une nouvelle chaine de caractère visible par l’utilisateur est créée, il faut penser à la définir dans les string.xml de chaque langue et utiliser le même identifiant quand on y fait référence de manière à permettre à l’application d’afficher la bonne traduction.