

Android2EE vous présente :

JCertifApp Architecture

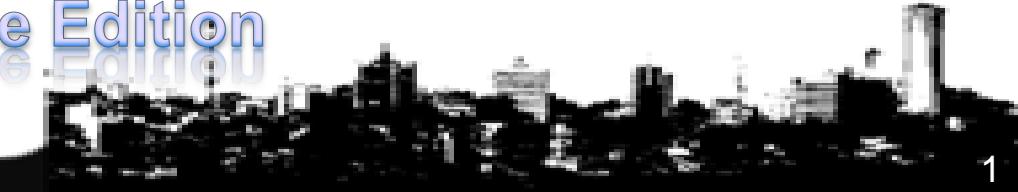
Dr. Mathias Séguy

Spécialiste Android Java J2EE

Expert Formateur Consultant Android

Auteur du livre

Android A Complete Course, From Basics to Enterprise Edition
disponible sur android2ee.com



Spéciale dédicace

Je remercie chaleureusement



Rossi

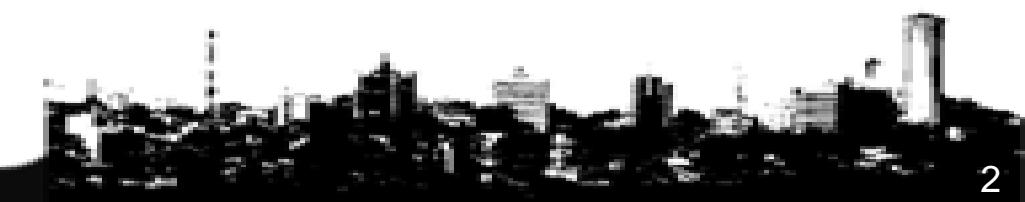


Max



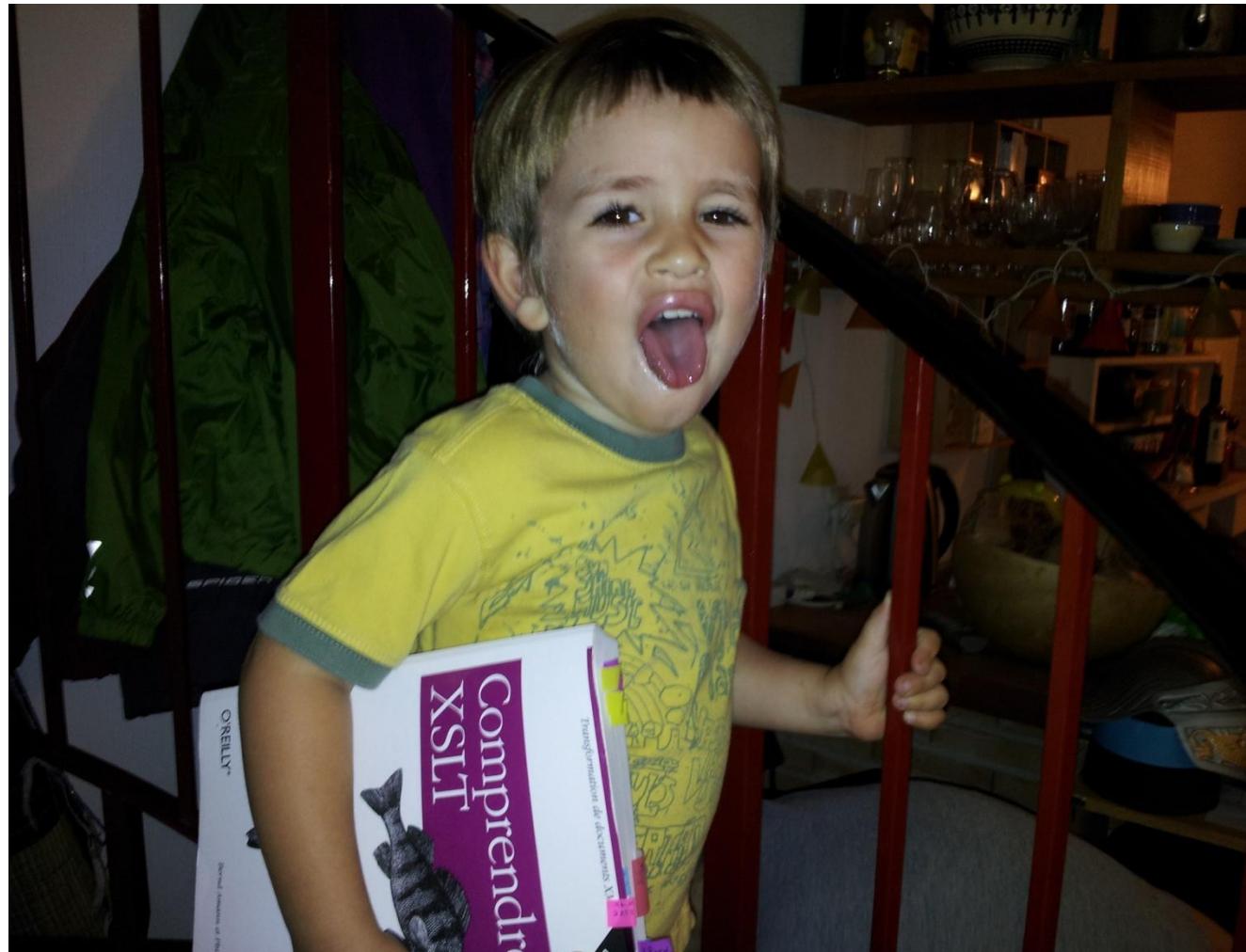
Chrisbel

Car sans eux je ne serai pas là :o)



Spéciale dédicace

Et j'embrasse chaleureusement

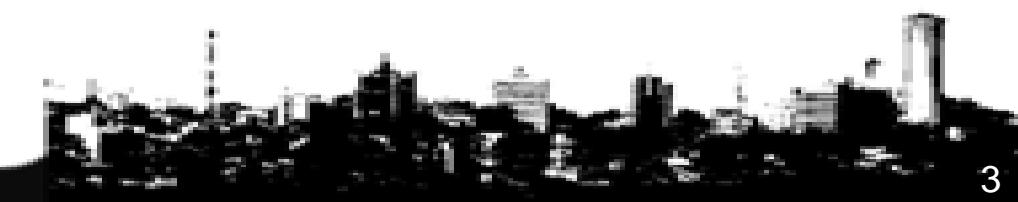


Mon fils Basile



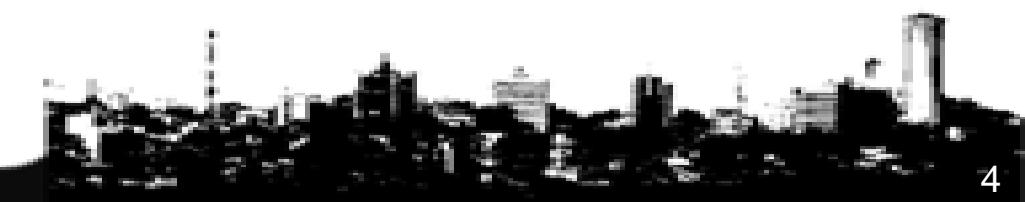
Et ma femme Céline

pour leur patience et leur amour.
Qui ne m'ont pas vu de l'été parce que "Papa travaille".



Au programme

- Vision globale de l'application
 - anim
- L'objet JCApplication
- La couche des Vues et transverse
 - Vision globale
 - Pre et post HC – LauncherActivity
 - FragmentSwitcher
 - ArrayAdapter
 - Ressources :
 - Drawable
 - Colors, Dimens, Style, Bools
 - Layout
 - String (string, url, errorMessage, constant)
 - Menu
- Couche transverse :
 - Model
 - HttpTools et UrlFactory
- Couche service
 - ServiceAndroid : UpdaterService
 - ThreadTraitement
- Gestion des StaredEvents
- Couche Dao
- Couche Com
- Couche Broadcast
- Projet de test





Dr. Mathias Seguy - Fondateur Android2EE



Formateur Consultant Expert Android

mathias.seguy.it@gmail.com (mail)

@android2ee (twitter)

Fondateur Android2EE – Formation Expertise Android

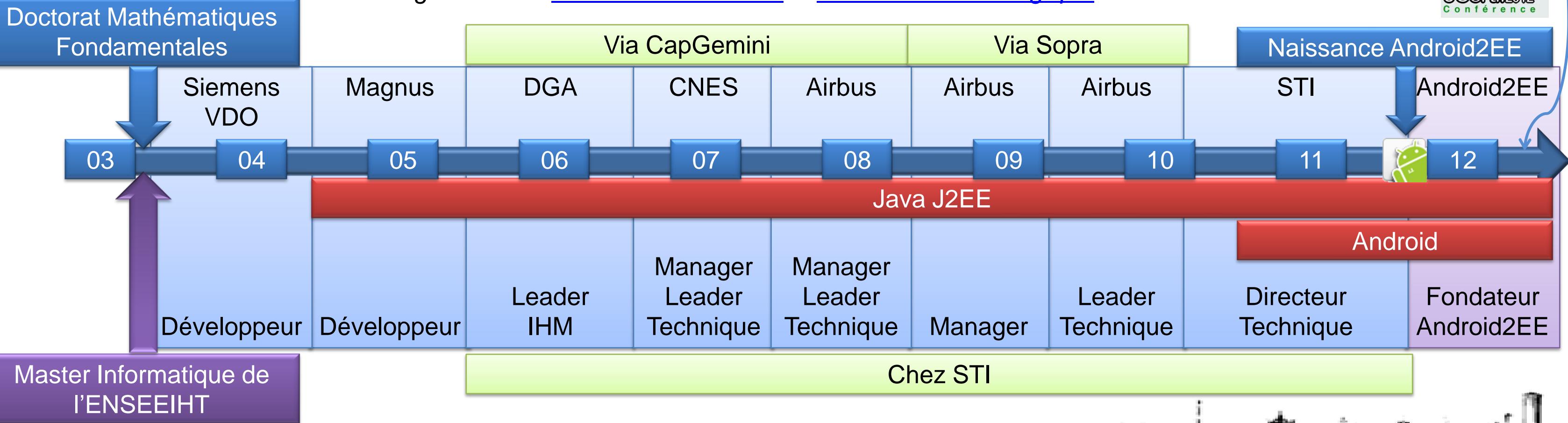
Auteur d'EBooks sur la programmation Android (Android2ee.com)

Docteur en Mathématiques Fondamentales

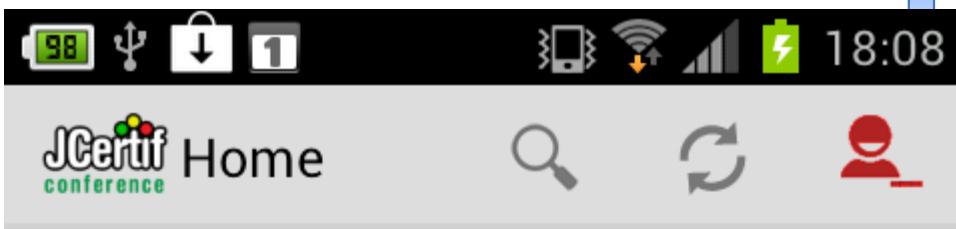
Expert Technique de l'Agence Nationale de la Recherche

Rédacteur sur Developpez.com

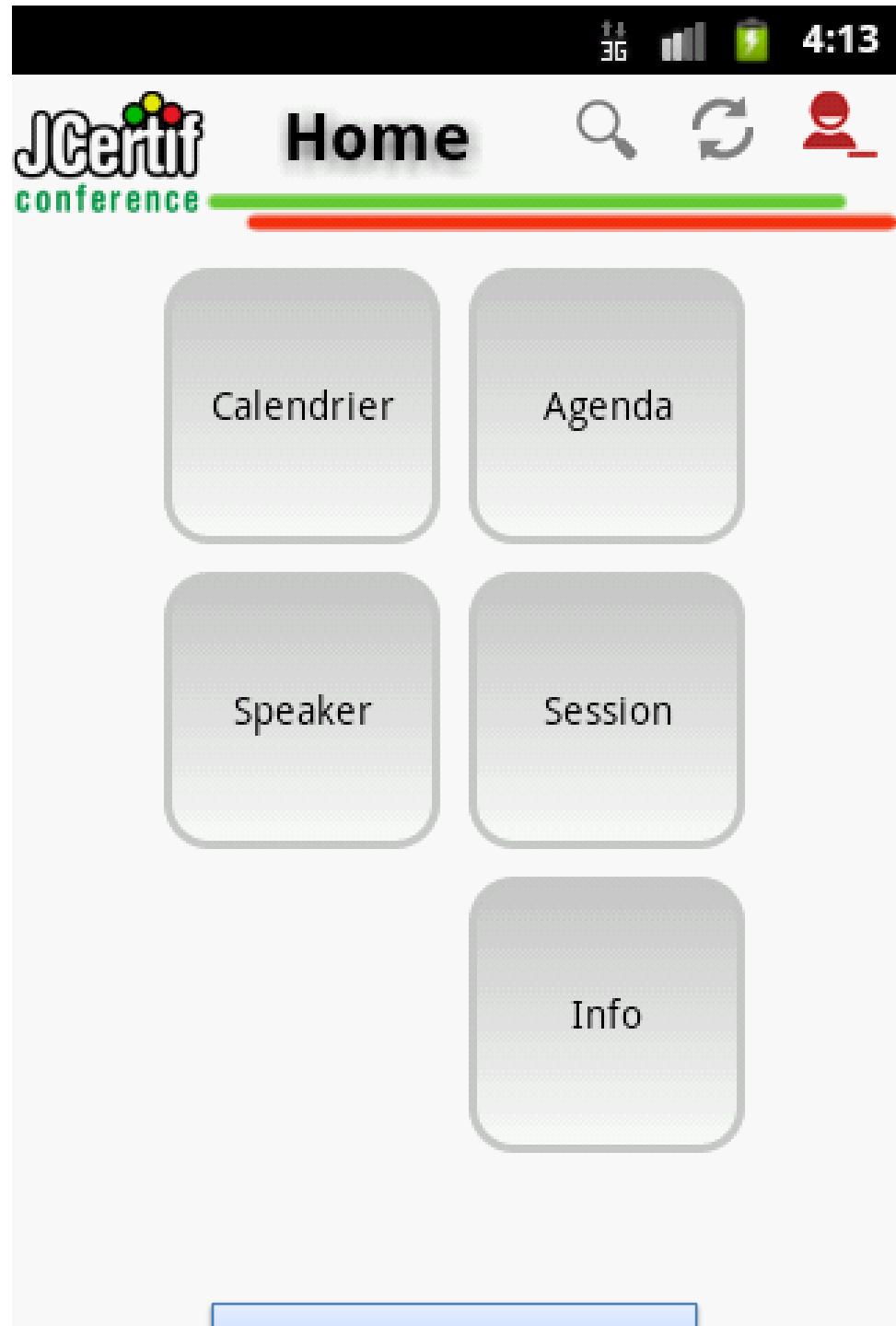
Blogs Android : Android2EE sur DVP et Android2ee sur BlogSpot



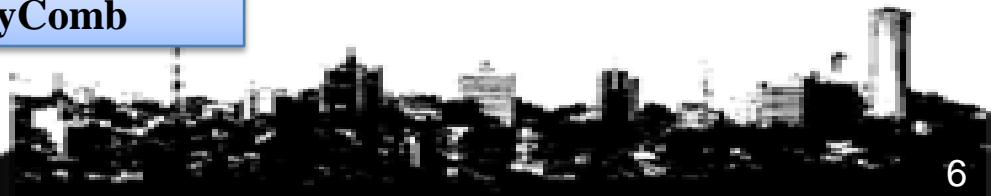
L'application



PostHoneyComb



PreHoneyComb

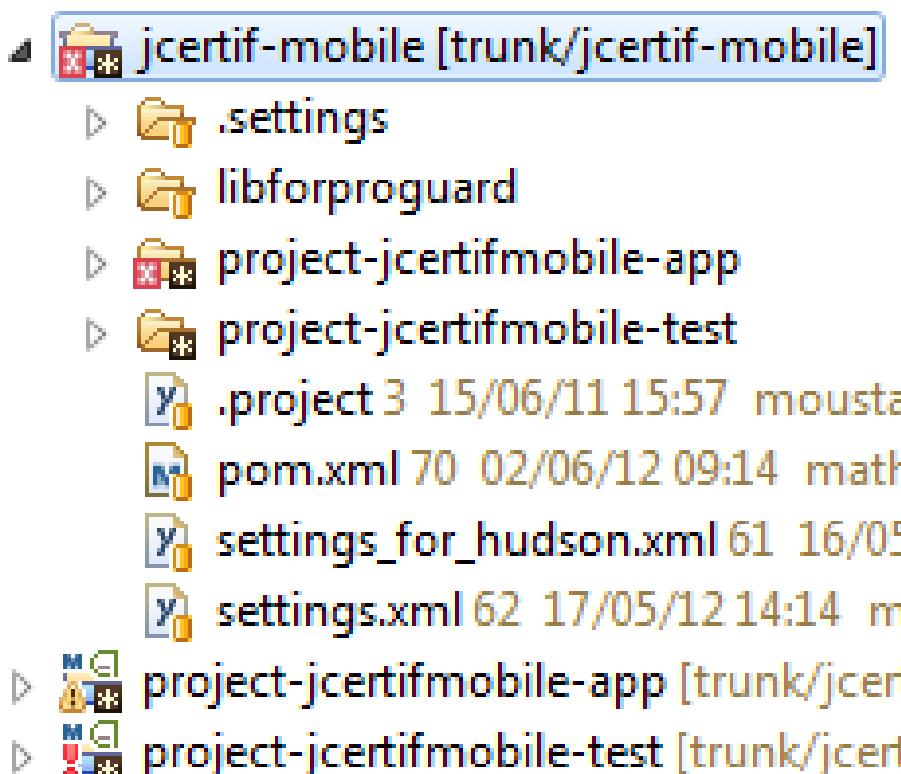


Vision globale projet

Le projet Jcertif est un projet multi-module Mavenisé

Maven permet de construire le projet et de le déployer au niveau de Jenkins. Il y a un problème avec Jackson pour la compilation de production.

Il faut vérifier que lors de l'import le *.classpath* et le *.project* du projet –app (de même pour test) ne sont pas modifiés par Eclipse.



```
classpath
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
<classpathentry kind="con" path="org.maven.ide.eclipse.MAVEN2_CLASSPATH_CONTAINER"/>
<classpathentry kind="con" path="com.android.ide.eclipse.adt.ANDROID_FRAMEWORK"/>
<classpathentry kind="con" path="com.android.ide.eclipse.adt.LIBRARIES"/>
<classpathentry kind="src" output="bin/classes" path="gen">
<attributes>
<attribute name="optional" value="true"/>
</attributes>
</classpathentry>
<classpathentry kind="src" output="bin/classes" path="src/main/java"/>
<classpathentry exported="true" kind="con"
path="org.eclipse.m2e.MAVEN2_CLASSPATH_CONTAINER"/>
<classpathentry kind="output" path="bin/classes"/>
</classpath>
```



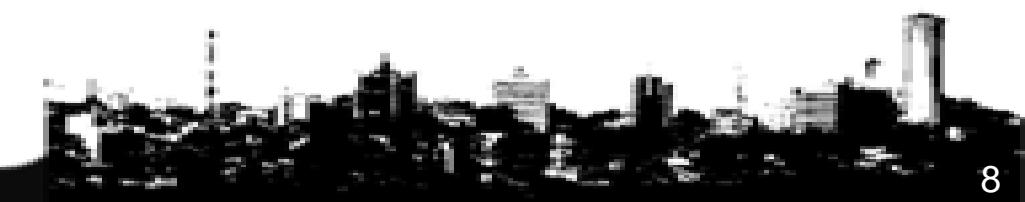
Vision globale projet

Le *.project*

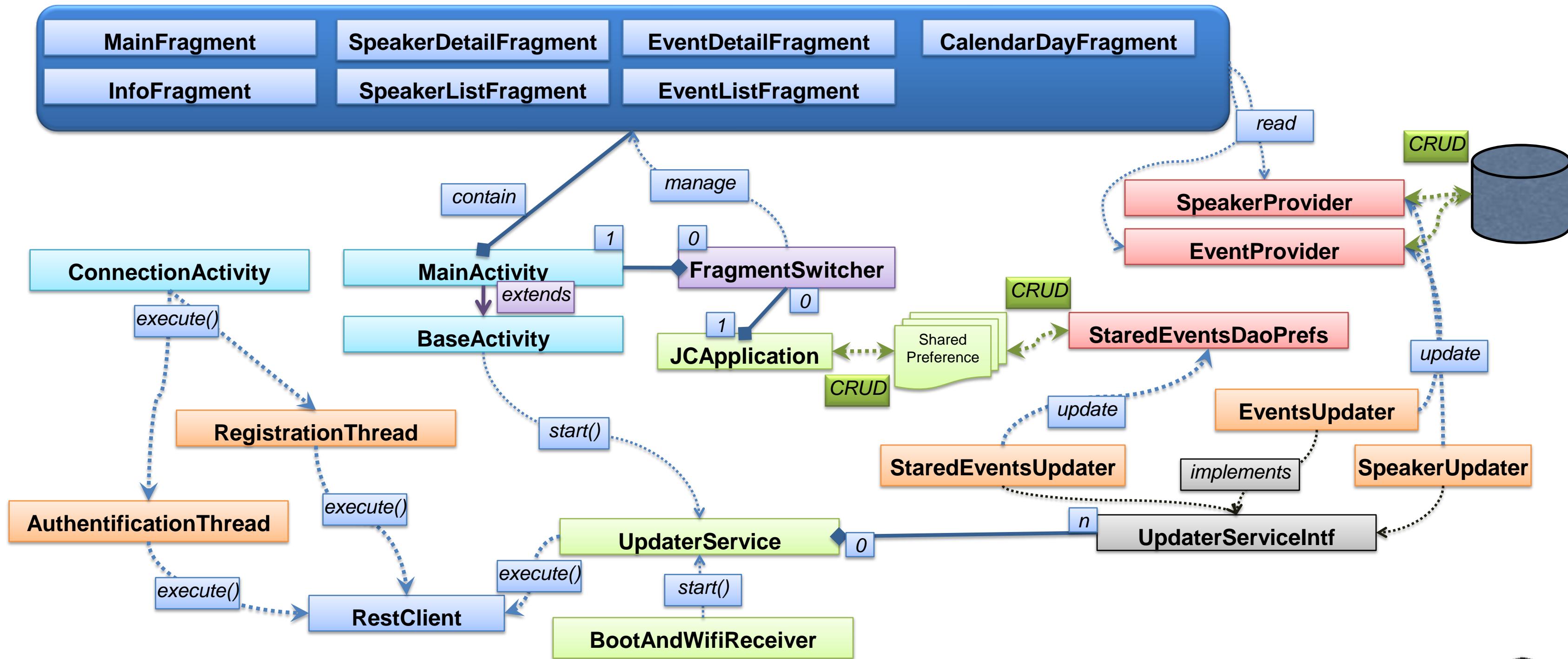
```
.project
<projectDescription>
<name>project-jcertifmobile-app</name>
<comment></comment>
<projects>
</projects>
<buildSpec>
<buildCommand>
<name>com.android.ide.eclipse.adt.ResourceManagerBuilder</name>
<arguments>
</arguments>
</buildCommand>
<buildCommand>
<name>com.android.ide.eclipse.adt.PreCompilerBuilder</name>
<arguments>
</arguments>
</buildCommand>
<buildCommand>
<name>org.eclipse.jdt.core.javabuilder</name>
<arguments>
</arguments>
</buildCommand>
<buildCommand>
<name>org.maven.ide.eclipse.maven2Builder</name>
<arguments>
</arguments>
```

.project (seconde partie)

```
</buildCommand>
<buildCommand>
<name>com.android.ide.eclipse.adt.ApkBuilder</name>
<arguments>
</arguments>
</buildCommand>
<buildCommand>
<name>org.eclipse.m2e.core.maven2Builder</name>
</buildCommand>
</buildSpec>
<natures>
<nature>com.android.ide.eclipse.adt.AndroidNature</nature>
<nature>org.eclipse.jdt.core.javanature</nature>
<nature>org.maven.ide.eclipse.maven2Nature</nature>
<nature>org.eclipse.m2e.core.maven2Nature</nature>
</natures>
</projectDescription>
```



Vision globale



L'objet JCApplication.

L'objet **JCApplication** hérite de l'objet application

L'objet Application est un élément central de toute application Android, il est le premier à naître, le dernier à mourir.

On peut y accéder de tout endroit de l'application.

Il s'utilise comme étant celui qui conserve l'état de l'application et les éléments qui doivent persister quels que soient l'activité et son cycle de vie.

Récupération JCApplication

```
// Retrieve the Jcapplication, within activity, service, broadcastReceiver  
((JCApplication) getApplication())
```

Récupération JCApplication

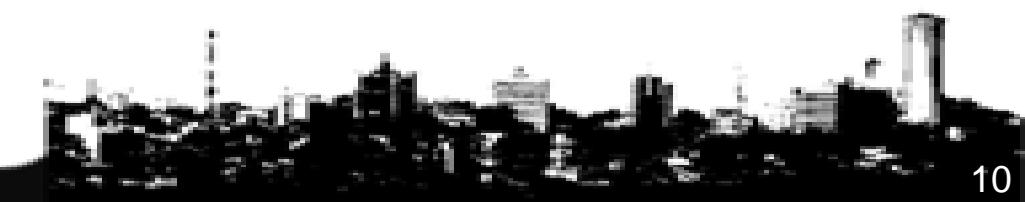
```
// Retrieve the Jcapplication, outside activity, service, broadcastReceiver  
JCApplication.getInstance()
```

Manifest.xml

```
<application  
    android:name="com.jcertif.android.JCertifApp"  
    android:debuggable="true"  
    android:icon="@drawable/ic_launcher"  
    android:label="@string/app_name" >
```

JCApplication Singleton

```
public class JCApplication extends Application {  
    /** Access Every Where **/  
    /** * instance of this */  
    private static JCApplication instance;  
    /** * @return The instance of the application */  
    public static JCApplication getInstance() {  
        return instance;  
    }  
    ...
```



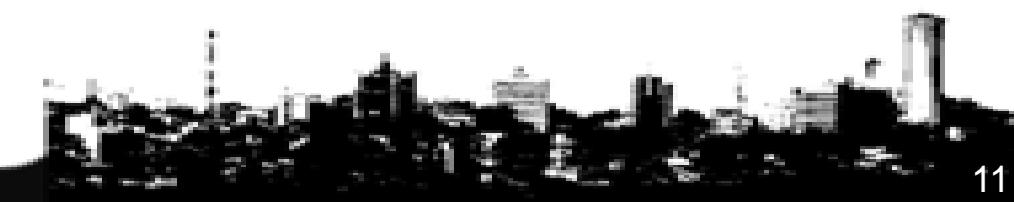
G JCApplication 90 16/08/12 13:56 mathias.seguy.it@gmail.com

- urlFactory : UrlFactory
- user : User
- △ fragmentSwitcher : FragmentsSwitcherLegacy
- △ fragmentSwitcherHC : FragmentsSwitcherHC
- △ baseActivity : BaseActivityIntf
- isServiceUpdaterUpdating : Boolean
- S instance : JCApplication
- S getInstance() : JCApplication
- △ onCreate() : void
- △ onLowMemory() : void
- F getUrlFactory() : UrlFactory
- F getUser() : User
- F setUser(User) : void
- clearDefaultUser() : void
- initializeUser() : void
- updateUser(int) : void
- userUpdate(SharedPreferences, String) : void
- saveUser() : void
- isValidUser() : boolean
- isDefaultValidUser() : boolean
- getUserKey() : String
- △ staredEventsUpdated : Boolean
- isStaredEventsUpdated() : Boolean
- setStaredEventsUpdatesTookIntoAccount() : void
- setStaredEventsUpdatedFromServer() : void
- F initialise(FragmentActivity, Boolean, Boolean) : FragmentsSw
- F initialise(Activity, Boolean, Boolean) : FragmentsSwitcherHC
- F setBaseActivity(BaseActivityIntf) : void
- F onDataUpdate() : void
- F onDataUpdateOver() : void

L'objet JCApplication.

L'objet JCApplication a à charge:

- Le renvoie *l'UrlFactory*
- Le stockage de la référence du *FragmentSwitcher* (pour qu'il ne soit pas déconstruit quand l'activité tourne)
- La gestion de l'utilisateur courant (stockage, vérification, récupération)
- Un mécanisme pour prévenir l'application que les *StaredEvents* ont été synchronisés avec le serveur
- Un mécanisme pour mettre à jour l'icône de mise-à-jour de l'application



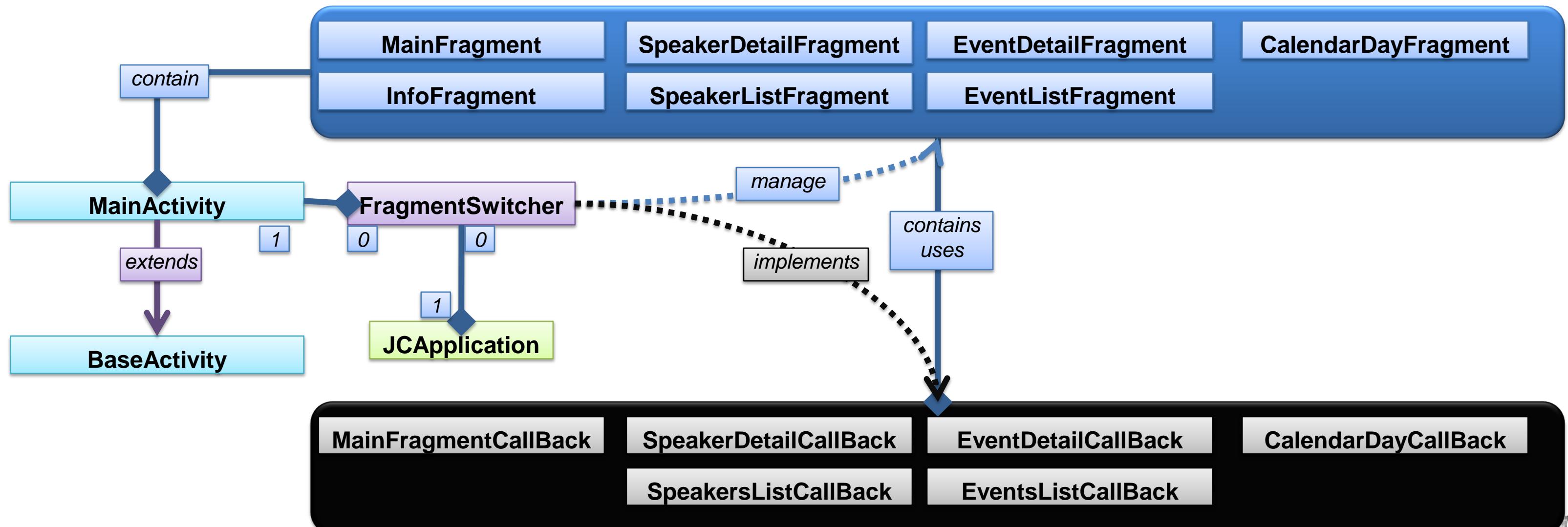
La couche Vue

FragmentSwitcher – MainActivity – Fragment et CallBack

L'objet *JCAplication* contient le *FragmentSwitcher* (il l'instancie et le stocke).

Le *mainActivity* récupère le *FragmentSwitcher* et permet aux fragments d'y accéder via lui.

Le *FragmentSwitcher* a à charge de gérer les transactions et l'affichage des différents fragments. Il est contenu par l'objet *JCAplication*, de cette manière il est indépendant du cycle de vie de l'activité et ne dépend que de celui de l'application.



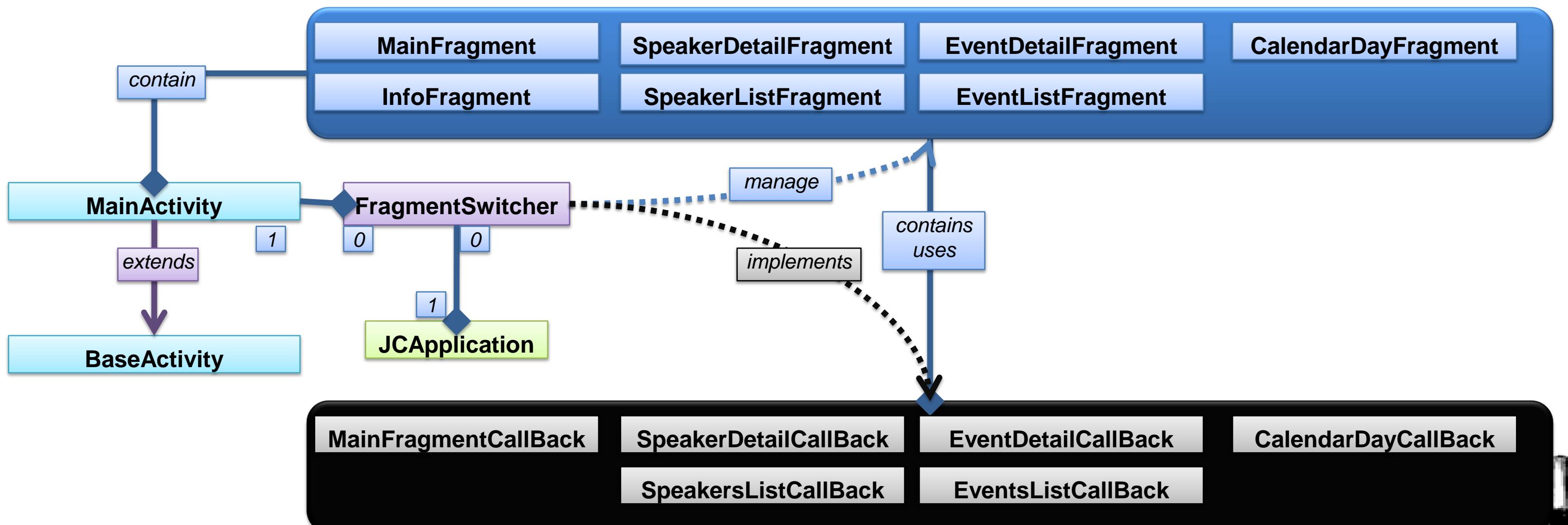
La couche Vue

FragmentSwitcher – MainActivity – Fragment et CallBack

Le « show » d'un fragment s'effectue via le *FragmentManager* et une *FragmentTransaction* qui ajoute le fragment dans le layout dédié.

Le *FragmentSwitcher* implémente les *CallBack* des Fragments, ainsi quand un Fragment souhaite lui parler (pour afficher un autre fragment) il utilise le système du *CallBack*.

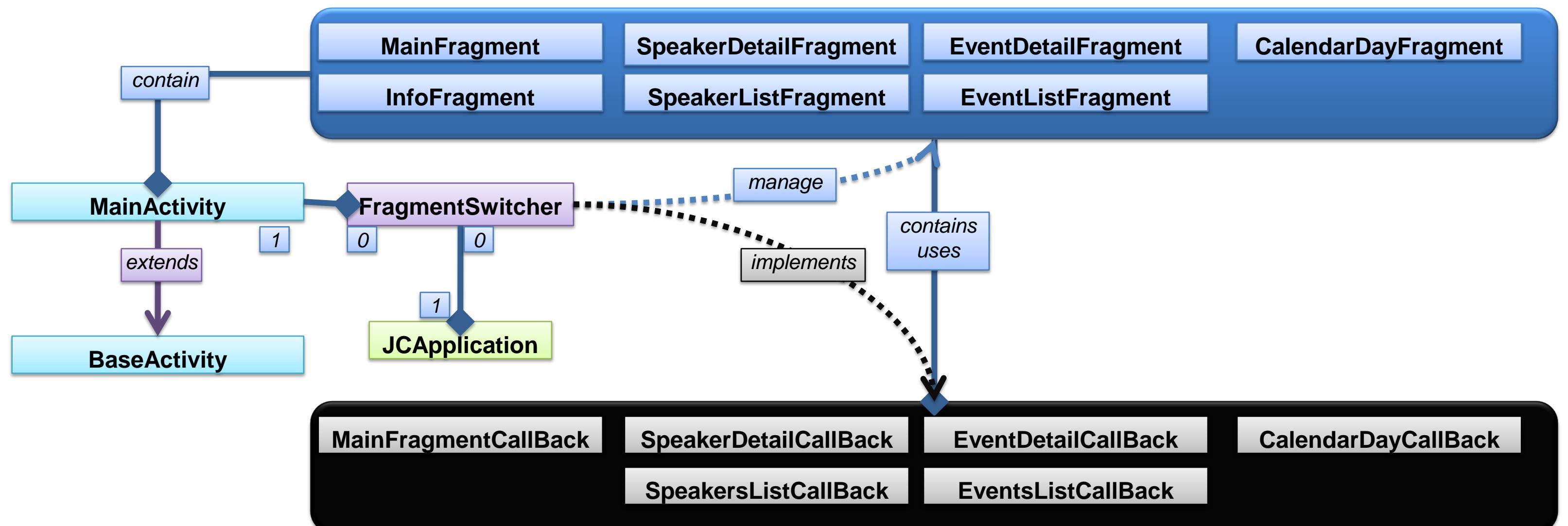
Par exemple, quand le *SpeakersListFragment* souhaite afficher un speaker, il passe par son callBack et appelle la méthode *showSelectedSpeaker*. A la création du *SpeakersListFragment*, le callback qui lui est donné est le *FragmentSwitcher*. Quand le *FragmentSwitcher* reçoit l'appel *showSelectedSpeaker*, il invoque juste un « show » sur le *SpeakerDetailFragment*.



La couche Vue

FragmentSwitcher – MainActivity – Fragment et CallBack

Un dernier détail, le `SpeakerDetailCallBack` possède la méthode `fillSpace`, en effet lors de l'appui sur la méthode back du téléphone, le fragment est détruit et passe par sa méthode `destroy`. Cette méthode appelle la méthode `fillSpace` de son `callBack` permettant si nécessaire au fragment `SpeakersListFragment` de remplir tout l'espace disponible.



Pre et PostHC

Duplication complète du code des fragments et activity

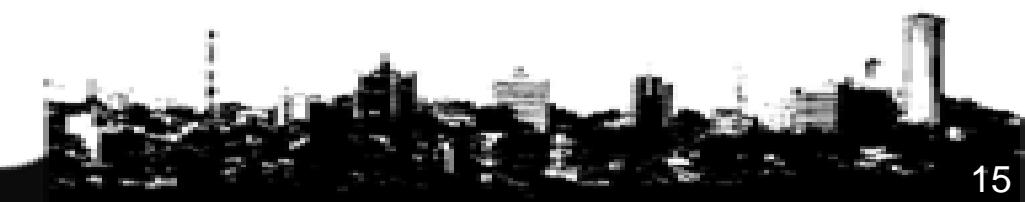
L'application est construite pour être compatible à HoneyComb avant et après avec l'utilisation des fragments et de l'ActionBar.

De ce fait elle est complètement multi-versionnée. Le problème étant que la *SupportLibrary* et les versions post HC n'ont pas d'interfaces communes, d'où la duplication du code.

Pour instancier l'une ou l'autre des activités on utilise le Parallel Activity Pattern.

```
generic
    BaseActivityHC.java 88 16/08/12 10:51
    BaseActivityIntf.java 76 28/06/12 13:51
    BaseActivityLegacy.java 88 16/08/12 11:51
info
    InfoFragmentHC.java 85 15/08/12 16:51
    InfoFragmentLegacy.java 83 14/08/12 16:51
main
    fragment
        MainFragment.java 77 15/07/12 11:51
        MainFragmentCallBack.java 66 30 14/08/12 17:01
        MainFragmentHC.java 83 14/08/12 17:01
        FragmentsSwitcherHC.java 83 14/08/12 17:01
        FragmentsSwitcherLegacy.java 83 14/08/12 17:01
        MainActivityHC.java 83 14/08/12 17:01
        MainActivityLegacy.java 83 14/08/12 17:01
    speaker
        detail
            SpeakerDetailCallBack.java 66 30 14/08/12 17:01
            SpeakerDetailFragment.java 85 15 14/08/12 17:01
            SpeakerDetailFragmentHC.java 85 14/08/12 17:01
        list
JApplication.java 90 16/08/12 13:56 mathias.senoussi
LauncherActivity.java 83 14/08/12 17:07 mathias.senoussi
```

ALLONS VOIR LE CODE DE
LAUNCHACTIVITY



ArrayAdapter

SpeakerListFragment et EventListFragment hérite de ListFragment

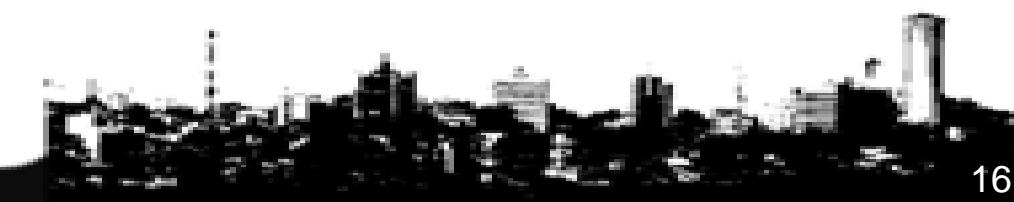
Les ArrayAdapter associés à ces deux composants utilisent le pattern du ViewHolder

ListAdapter Code

```
public View getView(int position, View convertView, ViewGroup parent) {  
    // ViewHolder will buffer the access to the individual fields of the row  
    // layout  
    final Event event = getItem(position);  
    // Recycle existing view if passed as parameter  
    // This will save memory and time on Android  
    // This only works if the base layout for all classes are the same  
    View rowView = convertView;  
    if (rowView == null) {  
        rowView = mInflater.inflate(R.layout.event_list_item, null);  
        ViewHolder viewHolder = new ViewHolder();  
        viewHolder.place = (TextView) rowView.findViewById(R.id.txtPlace);  
        viewHolder.time = (TextView) rowView.findViewById(R.id.txtTime);  
        ...  
        rowView.setTag(viewHolder);  
    }  
    final ViewHolder holder = (ViewHolder) rowView.getTag();  
    holder.name.setText(event.name);  
    ...  
    return rowView;  
}
```

ListAdapter Code

```
// static to save the reference to the outer class and to avoid access to  
// any members of the containing class  
static class ViewHolder {  
    public TextView place = null;  
    public TextView time = null;  
    public TextView name = null;  
    public TextView description = null;  
    public ImageView star=null;  
    public LinearLayout eventLay=null;  
}
```



Les Resources

drawable

action_search.png	76	28/06/12 13:51	mathias
button_focused.xml	64	20/05/12 00:48	r
button_normal.xml	64	20/05/12 00:48	m
button_pressed.xml	64	20/05/12 00:48	n
button.xml	64	20/05/12 00:48	mathias.s
calendar_evt_shape.xml	71	11/06/12 16:3	
calendar_hour_shape.xml	81	15/07/12 1:	
empty_star.png	76	28/06/12 13:51	math
header_bottom.png	76	28/06/12 13:51	r
ic_action_connect.png	83	14/08/12 17:0	
ic_action_disconnect.png	83	14/08/12 1:	
ic_action_refreshing_data.png	83	14/08/	
ic_action_share.png	83	14/08/12 17:07	r
ic_launcher.png	76	28/06/12 13:51	math
icon.png	76	28/06/12 13:51	mathias.seg
list_item_odd.xml	68	31/05/12 18:11	ma
list_item.xml	68	31/05/12 18:11	mathias
logo.png	76	28/06/12 13:51	mathias.seg
navigation_refresh.png	76	28/06/12 13:5	
panel_shape.xml	68	31/05/12 18:11	mat
speaker_detail_event_separator.png	76	2	
star_icon.png	76	28/06/12 13:51	mathia
star.png	83	14/08/12 17:07	mathias.segi

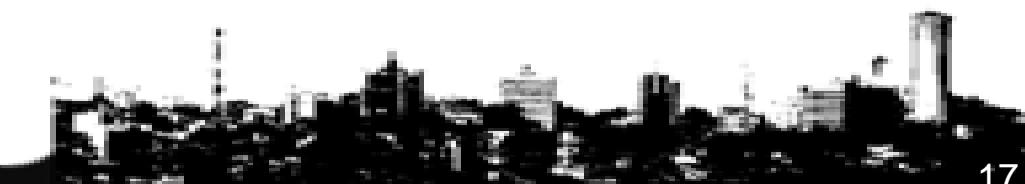
Resources are your best friends

Commençons par les Drawables:

D'une part, toutes les images sont dans les quatre formats ldpi, mdpi, hdpi, xhdpi et surtout tous les fichiers mdpi sont recopier dans le dossier *res\drawable*.

Dans ce dossier, se trouvent aussi certaines ressources Xml, des ShapeDrawables:

- Button (les bouton du *MainFrgament*)
- *Calendar_evt_shape* et *calendar_hour_shape* pour l'affichage des cellules du calendrier
- *List_item_odd* et *list_item* qui permettent l'affichage des listes sessions et speakers
- *Panel_shape* qui est utilise pour afficher le *linearLayout* principal des Speaker et des sessions



Colors

Resources are your best friends

Le fichier *Colors* est un fichier important de l'application, il centralise l'ensemble des couleurs utilisées par l'application.

D'une part il définit les *Root_Colors* qui est la charte couleurs de l'application, puis il définit les couleurs des composants, dites *Referent_Color*. Seules les *ReferentColor*s'utilisent dans l'application et aucune référence à une autre couleur n'est acceptée.

Root Color

```
<!-- Colors To use -->
<color name="red">#FFF6633</color>
<color name="red_pure">#FFFA300C</color>
<color name="green">#FF45D929</color>
<color name="violet">#FFCCCCFF</color>
<color name="violet_bright">#FFD5D5FC</color>
<color name="violet_dark">#FF9F9FF4</color>
<color name="violet_trans">#77CCCCFF</color>
<color name="violet_bright_trans">#77D5D5FC</color>
<color name="violet_dark_trans">#779F9FF4</color>
<color name="yellow">#88ECEC0B</color>
<color name="blue">#880000FF</color>
<color name="deep_blue">#FF333399</color>
<color name="black">#FF000000</color>
<color name="black_translucent">#33000000</color>
<color name="white">#FFFFFFFF</color>
<color name="gray_bright">#FFCBCBCB</color>
<color name="gray_dark">#FF707070</color>
<color name="white_trans">#88FFFFFF</color>
<color name="gray_bright_trans">#88CBCBCB</color>
<color name="gray_bright_lesstrans">#BBCBCBCB</color>
<color name="gray_dark_trans">#88707070</color>
<color name="translucent">#00889988</color>
```

Root Color

```
<!-- Referential colors → <!-- Background -->
<color name="background">@color/white</color>
<color name="background_dark">@color/gray_bright</color>
<color name="background_toast">@color/gray_bright_trans</color>
<color name="background_toast_lesstrans">@color/gray_bright_lesstrans</color>
<color name="btn_background">@color/gray_bright</color>
<color name="btn_background_selected">@color/gray_dark</color>
<color name="edt_background">@color/gray_bright</color>
<color name="list_background">@color/gray_bright</color>
<color name="list_odd_background">@color/gray_dark</color>
<!-- Foreground -->
<color name="text_frgd">@color/black</color>
<color name="edt_frgd">@color/black</color>
<color name="btn_frgd">@color/black</color>
<color name="list_frgd">@color/black</color>
<color name="list_odd_frgd">@color/black</color>
<!-- Shape -->
<color name="button_border">@color/gray_bright</color>
<color name="button_border_selected">@color/gray_bright</color>
<color name="shape_end_color">@color/gray_bright</color>
```

• • •

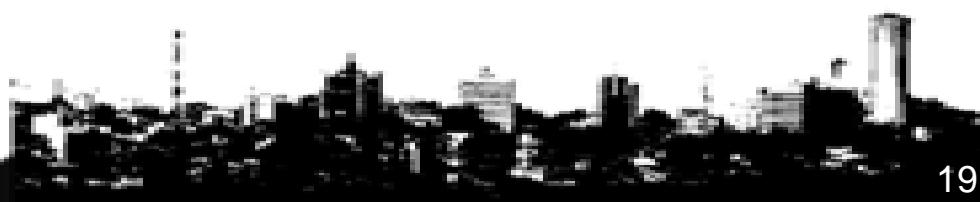
Dimens.xml

Centralisation des dimensions

Toutes les dimensions de l'application sont centralisées dans ce fichier. Elles sont de plus déclinées en trois dimensions pour s'adapter aux différentes tailles des appareils : *res\values\dims.xml*, *res\values-large\dims.xml*, *res\values-xlarge\dims.xml*

Dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Margins -->
    <dimen name="marginLeft">5dp</dimen>
    <dimen name="marginRight">5dp</dimen>
    <dimen name="marginTop">5dp</dimen>
    <dimen name="marginBottom">5dp</dimen>
    <dimen name="marginLeftLarge">15dp</dimen>
    <dimen name="marginRightLarge">15dp</dimen>
    <dimen name="marginTopLarge">15dp</dimen>
    <dimen name="marginBottomLarge">15dp</dimen>
    <!-- Padding -->
    <dimen name="paddingLeft">5dp</dimen>
    <dimen name="paddingRight">5dp</dimen>
    <dimen name="paddingTop">5dp</dimen>
    <dimen name="paddingBottom">5dp</dimen>
    <!-- Text Size -->
    <dimen name="textSizeXSmall">10sp</dimen>
    <dimen name="textSizeSmall">12sp</dimen>
    <dimen name="textSize">15sp</dimen>
    <dimen name="textSizeLarge">17sp</dimen>
    <dimen name="textSizeXLarge">24sp</dimen>
    <!-- Button Dimensions →...>
```



Style et thème

Tous les composants ont un style défini dans ces fichiers

Le style de tous les composants de l'application est défini dans ce fichier. De plus le style s'adapte en fonction de la version du système. En effet, deux balises sont définies permettant d'hériter de l'ensemble des styles soit de *android.Theme* soit de *android.Theme.Holo*

res\values\theme.xml

```
<resources>
    <!-- Style for Button -->
    <style name="root_style" parent="root_theme">
        <item name="android:paddingLeft">@dimen/paddingLeft</item>
        <item name="android:paddingRight">@dimen/paddingRight</item>
        <item name="android:paddingTop">@dimen/paddingTop</item>
        <item name="android:paddingBottom">@dimen/paddingBottom</item>
        <item name="android:layout_marginLeft">@dimen/marginLeft</item>
        <item name="android:layout_marginRight">@dimen/marginRight</item>
        <item name="android:layout_marginBottom">@dimen/marginBottom</item>
        <item name="android:layout_marginTop">@dimen/marginTop</item>
    </style>
    <!-- Style for TextView -->
    <style name="txv_style" parent="root_style">
        <item name="android:textSize">@dimen/textSize</item>
        <item name="android:textStyle">normal</item>
        <item name="android:background">@color/translucent</item>
        <item name="android:textColor">@color/text_frgd</item>
    </style>

    <!-- Style for TextView No Vertical Margin No Vertical Padding -->
    <!-- ***** -->
```

res\values\theme_root.xml

```
<resources>
    <!-- Root Theme -->
    <style name="root_theme"
          parent="@android:style/Theme"></style>
</resources>
```

res\values-v11\theme_root.xml

```
<resources>
    <!-- Root Theme -->
    <style name="root_theme"
          parent="@android:style/Theme.Holo"></style>
</resources>
```



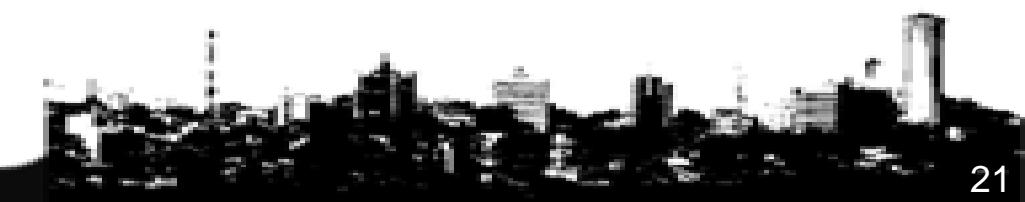
Bools

L'usage des Booleans sert à la mise en place du // Activity pattern

Deux fichiers sont ainsi présents dans l'application:

```
res\values-v11\bools.xml  
<resources>  
  
    <bool name="postHC">true</bool>  
    <bool name="preHC">false</bool>  
  
</resources>
```

```
res\values\bools.xml  
<resources>  
  
    <bool name="postHC">false</bool>  
    <bool name="preHC">true</bool>  
  
</resources>
```



Layouts

layout

- account_information_dialog.xml 81 15/07/12 15:41 m
- account_profile_dialog.xml 81 15/07/12 15:41 m
- calendar_event_cell.xml 83 14/08/12 17:07 mathias.seguy.it
- calendar_hour_cell.xml 83 14/08/12 17:07 mathias.seguy.it
- calendar.xml 81 15/07/12 15:41 mathias.seguy.it
- connection.xml 83 14/08/12 17:07 mathias.seguy.it
- event_detail.xml 81 15/07/12 15:41 mathias.seguy.it
- event_list_fragment.xml 81 15/07/12 15:41 mathias.seguy.it
- event_list_item.xml 87 15/08/12 23:20 mathias.seguy.it
- event_list.xml 81 15/07/12 15:41 mathias.seguy.it
- info_fragment.xml 83 14/08/12 17:07 mathias.seguy.it
- jcertif_header.xml 83 14/08/12 17:07 mathias.seguy.it
- main_activity.xml 83 14/08/12 17:07 mathias.seguy.it
- main_fragment.xml 81 15/07/12 15:41 mathias.seguy.it
- speaker_detail.xml 76 28/06/12 13:51 mathias.seguy.it
- speaker_event_detail.xml 68 31/05/12 18:11 mathias.seguy.it
- speaker_list_frgmt.xml 81 15/07/12 15:41 mathias.seguy.it
- speaker_list_item.xml 87 15/08/12 23:20 mathias.seguy.it
- toast_account.xml 83 14/08/12 17:07 mathias.seguy.it
- toast_syncro_starredevents_error.xml 83 14/08/12 17:07 mathias.seguy.it
- toast_update.xml 83 14/08/12 17:07 mathias.seguy.it

layout-land

- main_activity.xml 83 14/08/12 17:07 mathias.seguy.it
- main_fragment.xml 83 14/08/12 17:07 mathias.seguy.it

SpeakerListFragment et EventListFragment hérite de ListFragment

Les layouts définissent les différentes vues. Pas de tricks particulière. Ils sont tous là, seules *main* possède un *layout* spécifique pour le mode *landscape* permettant l'affichage de deux fragments côté à côté.

Par contre les composants utilisent tous les styles, les couleurs (via les styles) et les dimensions définis dans les fichiers xml. Pas d'exception.

Layout exemple

```
<LinearLayout  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:orientation="horizontal"  
    android:paddingBottom="@dimen/paddingBottom"  
    android:paddingLeft="@dimen/paddingLeft"  
    android:paddingRight="@dimen/paddingRight">  
  
<ImageView  
    android:id="@+id/speakerImg"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:paddingLeft="@dimen/paddingLeft" />  
  
<TextView  
    android:id="@+id/speakerBioPart1"  
    style="@style/txtv_style_nvm"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="0dp" />  
  
</LinearLayout>
```

String

Les fichiers de string sont en français (pas de traduction encore)

Il y a plusieurs fichiers de type String:

- constants.xml qui stocke les constantes de l'application (clef du sharedPreference, nom d'intent,...)
- errorMessage.xml pour les messages d'erreurs
- info_message.xml pour le fragment info
- string.xml pour les chaines usuelles de l'application
- url.xml pour le stockage des url de l'application



Menu

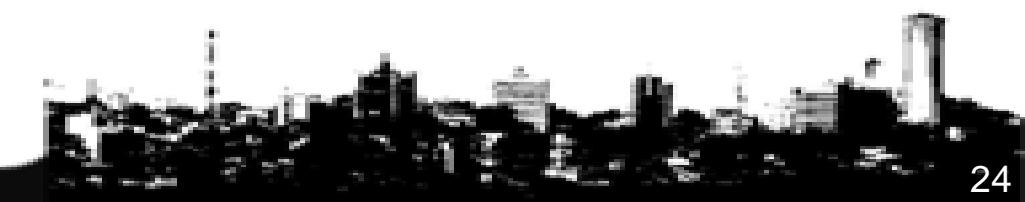
Le fichier de menu est identique pour toute l'application

Pour l'heure il n'y a pas d'intérêt à scinder le menu en post et pré HC, un menu global s'affiche tout aussi bien dans les 2 versions.

```
res\menu\menu.xml

<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/menu_search"
        android:icon="@drawable/action_search"
        android:showAsAction="always"
        android:title="@string/menu_search"/>
    <item
        android:id="@+id/menu_forceupdate"
        android:icon="@drawable/navigation_refresh"
        android:showAsAction="always/withText"
        android:title="@string/menu_forceupdate"/>
    <item
        android:id="@+id/menu_disconnect_user"
        android:icon="@drawable/ic_action_disconnect"
        android:showAsAction="always"
        android:title="@string/menu_disconnect_user"/>
    <item
        android:id="@+id/menu_connect_user"
        android:icon="@drawable/ic_action_connect"
        android:showAsAction="always"
        android:title="@string/menu_connect_user"/>
</menu>
```



Animation

Animation post et pre HC

Les animations ne sont pas compatibles en pre et post HC. Elles sont utilisées lors des changements de fragments.

res\anim\anim_push_left.xml

```
<set  
    xmlns:android="http://schemas.android.com/ap  
k/res/android" >  
  
    <translate  
        android:duration="500"  
        android:fromXDelta="100%p"  
        android:toXDelta="0" />  
  
    <alpha  
        android:duration="500"  
        android:fromAlpha="0.0"  
        android:toAlpha="1.0" />  
  
    <scale  
        android:duration="500"  
        android:fromXScale="0.0"  
        android:fromYScale="0.0"  
        android:pivotX="50%"  
        android:pivotY="50%"  
        android:toXScale="1.0"  
        android:toYScale="1.0" />  
  
</set>
```

res\anim-v11\anim_push_left.xml

```
<objectAnimator  
    xmlns:android="http://schemas.android.com/ap  
k/res/android"  
    android:duration="1000"  
    android:propertyName="translationX"  
    android:valueFrom="-250"  
    android:valueTo="0"  
    android:valueType="floatType" />
```

anim

anim_push_left_in.xml 66 30/05/12 21:

anim_push_left_out.xml 66 30/05/12 2

anim_push_right_in.xml 83 14/08/12 1

anim_push_right_out.xml 83 14/08/12

anim-v11

anim_push_left_in.xml 83 14/08/12 17:

anim_push_left_out.xml 83 14/08/12 1

anim_push_right_in.xml 83 14/08/12 1

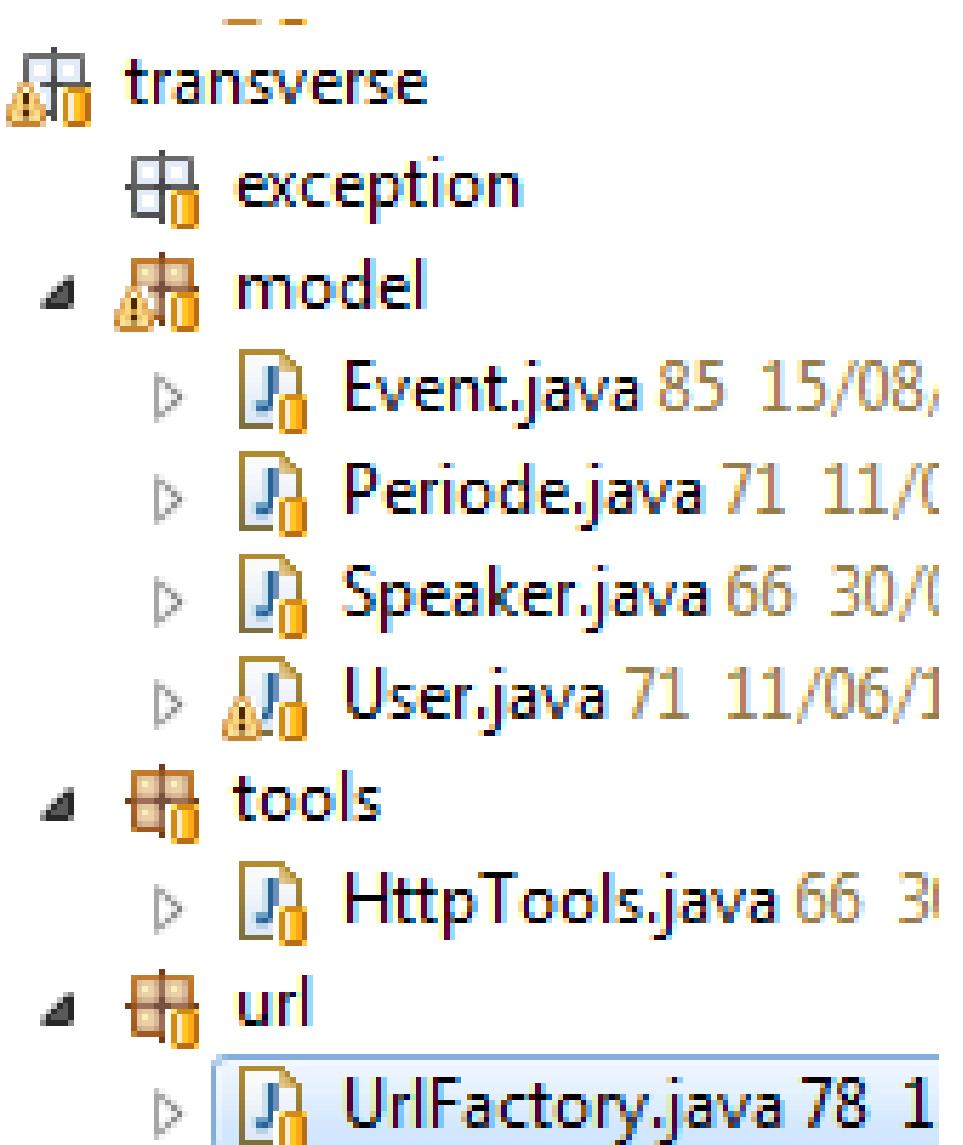
anim_push_right_out.xml 83 14/08/12

Couche transverse

Modèle métier et classes utilitaires

La couche transverse possède la définition des objets métiers *Events*, *Period*, *Speaker*, *User* et deux classes utilitaires, l'une *HttpUtil* permet de savoir si un code retour est un code d'erreur et l'autre *UrlFactory* renvoie l'Url demandée.

Il manque dans la classe transverse toute la gestion des Exceptions.



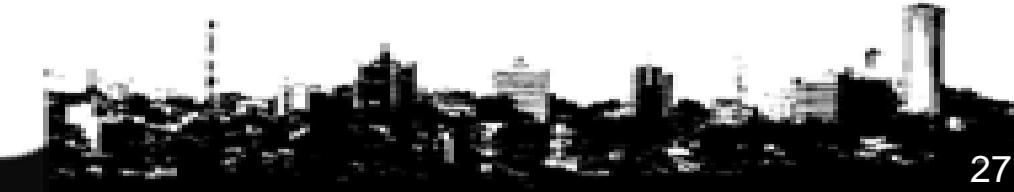
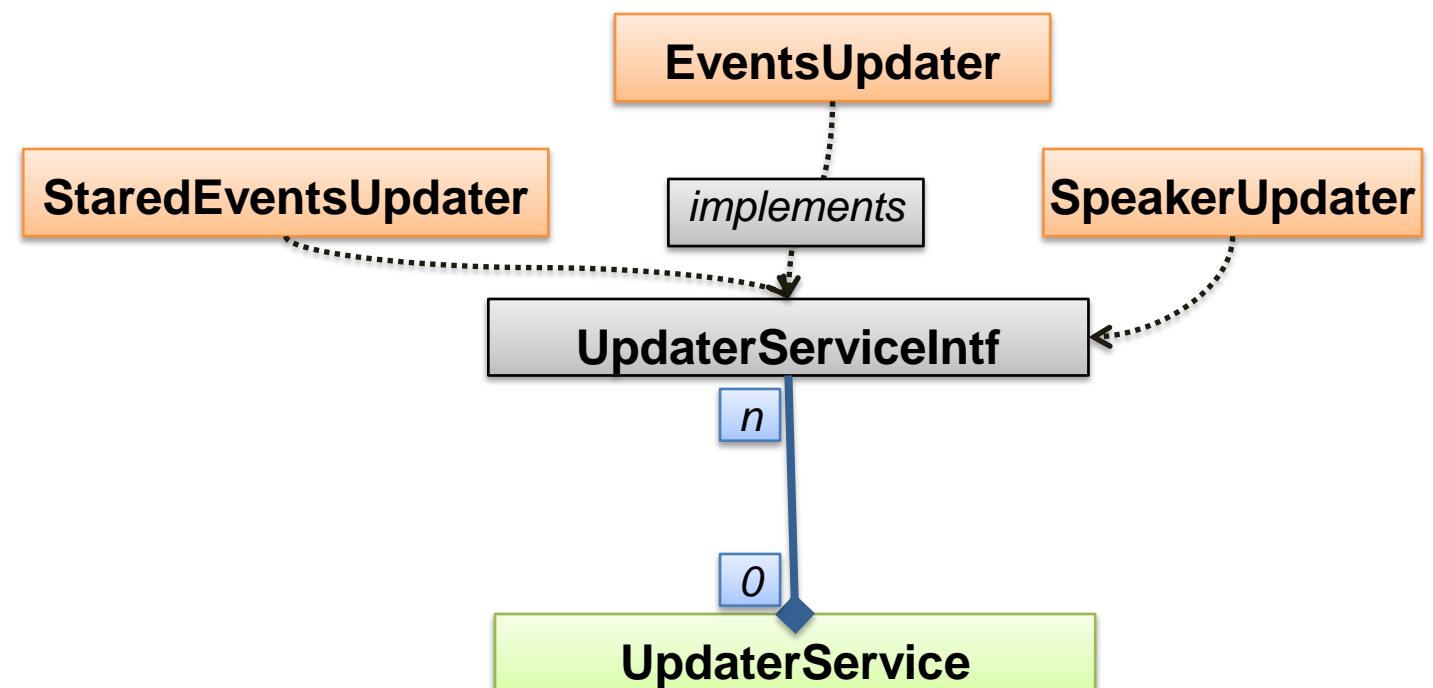
Couche Service

Le ServiceUpdater

Trois sous-packages principaux de ce package: Celui des *Threads* de l'application, celui du service des *StaredEvents* et enfin le *ServiceUpdater*. Ce dernier n'a pas la notification (evol). Le pattern mis en place permet de lui ajouter facilement d'autres *updaters*.

JCAplication Singleton

```
public void onCreate() {
    super.onCreate();
    // instantiate all the update services
    updaters = new ArrayList<UpdaterServiceElementIntf>();
    updaters.add(new SpeakersUpdater());
    updaters.add(new EventsUpdater());
    //Rajouter d'autres Updater ici
    // create the Threads
    backgroundThreads = new ArrayList<Thread>();
    Thread backgroundThread;
    for (final UpdaterServiceElementIntf updater : updaters) {
        // Define the Threads
        backgroundThread = new Thread(new Runnable() {
            public void run() {
                try {updater.onUpdate();} catch (Throwable t) {
                    // just end the background thread }
            });
        backgroundThread.setName("UpdateThread" + updater.getName());
        backgroundThreads.add(backgroundThread);
    }
}
```



Couche Service

Les Threads

Les Threads héritent de *BasicBackgroundThread* qui encapsule un ensemble de chose et permet de n'avoir qu'une méthode à mettre en place (*workingMethod*) le reste étant géré par la classe mère.

Un exemple d'héritage

```
public class RegistrationThread extends BasicBackgroundThread {
    /** @param mCallBack */
    public RegistrationThread(RegistrationThreadCallBack mCallBack) {
        super(mCallBack);
    }
    @Override
    public void workingMethod(Handler handler) {
        String responseString = null;
        int responseStatus=STATUS_NOTSET;
        String url = JCApplication.getInstance().getUrlFactory().getRegisterUrl();
        RestClient client = new RestClient(url);
        try {
            client.Execute(RequestMethod.POST);
            responseString = client.getResponse();
            responseStatus=client.getResponseCode();
        } catch (Exception e) {
            Log.d(this.getClass().getSimpleName(), "LocalServiceBinder : json " + responseString);
        }
        Message mess = handler.obtainMessage();
        Bundle data = new Bundle();
        data.putString(RESPONSE_KEY, responseString);
        data.putInt STATUS_KEY, responseStatus);
        mess.setData(data);
        handler.sendMessage(mess);
    }
}
```

Lancement d'une Thread

```
public class ConnectionActivityLegacy extends Activity
implements AuthentThreadCallBack

...
// Launch the thread
// authThread.execute(email, password);
Bundle params = new Bundle();
params.putString(AuthenticationThread.EMAIL_KEY, email);
params.putString(AuthenticationThread.PASSWORD_KEY,
password);
authThread.execute(this, params);
...

public void handleMessage(Message message) {
    String httpGetResponse =
message.getData().getString(AuthenticationThread.RESPONSE_K
EY);
    int httpGetRespStatus =
message.getData().getInt(AuthenticationThread.STATUS_KEY);
    ...
}
```

L'interface BasicBackgroundCallBack

```
public interface BasicBackgroundCallBack {
    public void handleMessage(Message message);
}
```



Couche DAO

EventProvider, SpeakerProvider, StaredEventsDao

La couche DAO possède deux façons distinctes d'accéder aux objets; *EventProvider* et *SpeakerProvider* stockent les données dans une base Sqlite en utilisant le framework *OrmLite*, *StaredEventsDao* stocke les données dans le *DefaultSharedPreference*.

Ce sont deux façons distinctes de persister les données.

EventProvider sample

```
public class EventProvider {  
  
    private Context mContext;  
    private DatabaseHelper mDbHelper;  
    private Dao<Event, Integer> mEventDao;  
  
    public EventProvider() throws SQLException {  
        mContext = JCApplication.getInstance().getApplicationContext();  
        mDbHelper = new DatabaseHelper(mContext);  
        mEventDao = mDbHelper.getEventDao();  
    }  
  
    public List<Event> getAllEvents() throws SQLException {  
        return mEventDao.queryForAll();  
    }  
  
    public Event getEventById(Integer eventId) throws SQLException {  
        return mEventDao.queryForId(eventId);  
    }  
  
    ...
```

JCAapplication Singleton

```
public class StaredEventsDaoPrefs implements StaredEventsDaoIntf {  
  
    @Override  
    public List<Integer> getStaredEvents() {  
        JCApplication app = JCApplication.getInstance();  
        String userKey = app.getUserKey();  
        SharedPreferences preferences =  
            PreferenceManager.getDefaultSharedPreferences(app.getApplicationContext());  
        String evtList = preferences.getString(userKey +  
            app.getString(R.string.shLocalStaredEvents), "-1");  
        if (evtList.length() == 0 || "-1".equals(evtList)) {  
            return new ArrayList<Integer>();  
        } else {  
            // unmarshall the value  
            return fromStringToList(evtList);  
        }  
    }
```

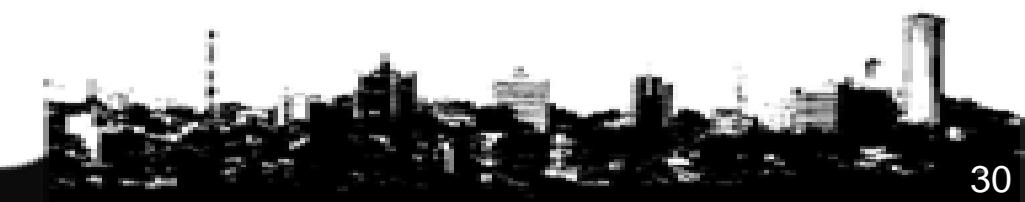
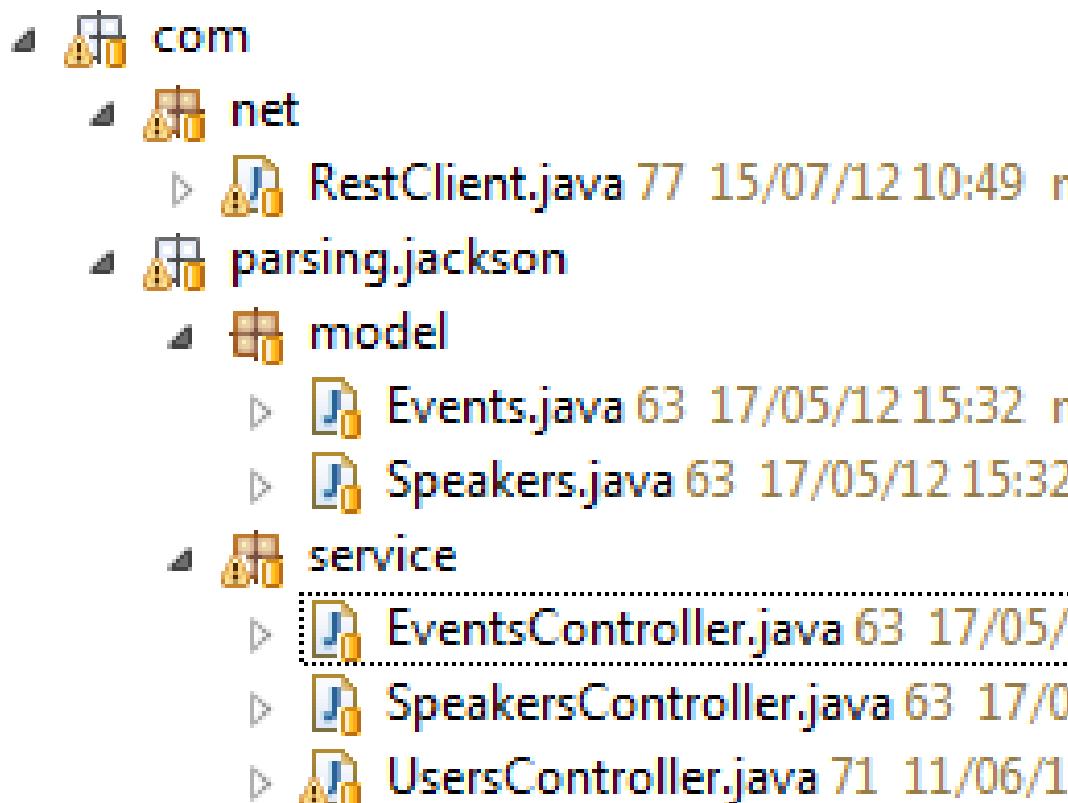


Couche Com

SpeakerListFragment et EventListFragment héritent de ListFragment

Cette couche contient deux packages:

- net qui permet avec la classe RestClient de communiquer avec un service Rest en utilisant HttpClient sur un client Android (Lucene code)
- parsing.jackson qui permet de marshall/unmarshall le flux http rentrant (Json) en un graphe d'objets. Les controllers sont les parsers, les models sont en fait des HashMap du Type `HashMap<String, ArrayList<Event>>`. Pourquoi sont-ils là ?o?



Couche BroadCast

La couche BroadCast contient les BroadCastReceiver

Il n'y en a qu'un qui a à charge de lancer l'update automatique de l'application.

Manifest.xml

```
<!-- BroadCastReceiver -->
<!-- Listening for Boot and connectivity changed -->
<receiver
    android:name="com.jcertif.android.broadcastreceiver.BootAndWifiReceiver" >
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
        <action android:name="android.net.wifi.STATE_CHANGE" />
    </intent-filter>
</receiver>
```

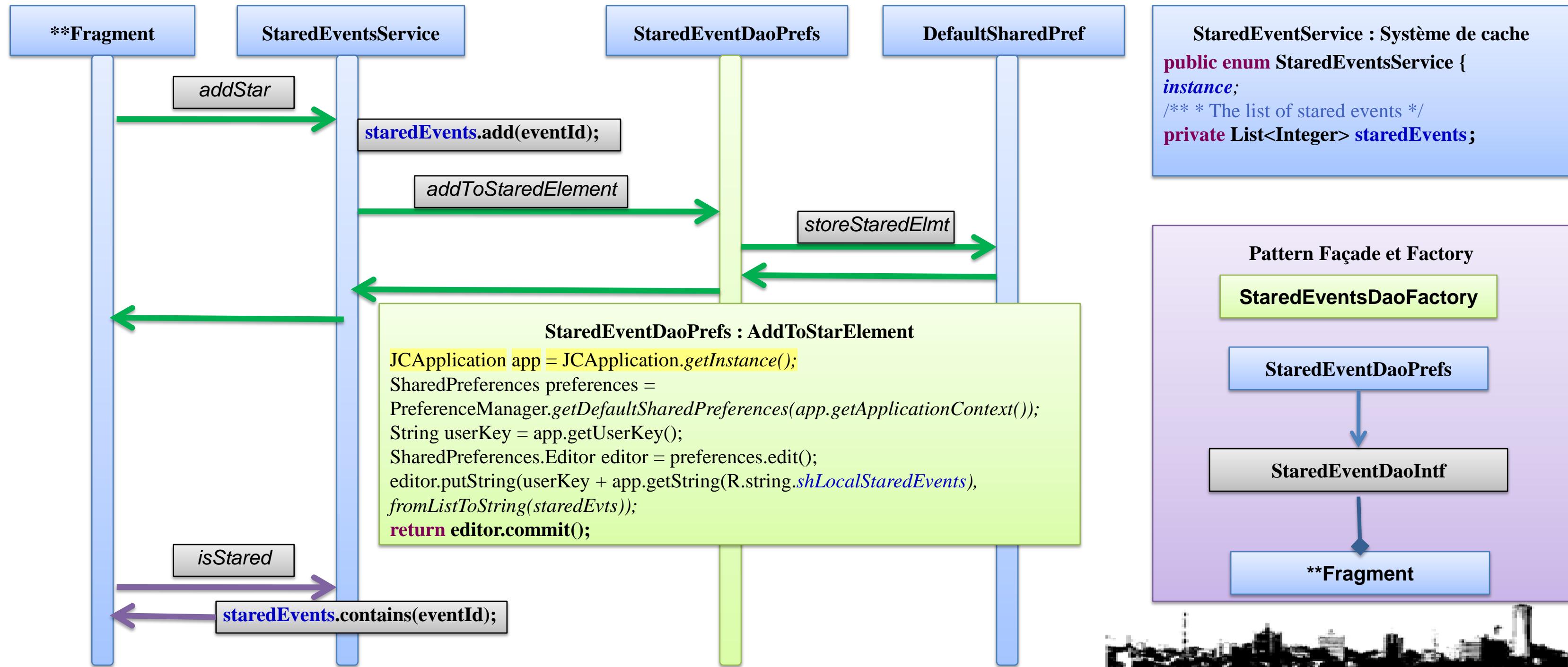
JavaCode

```
public class BootAndWifiReceiver extends BroadcastReceiver {
    /** * The min Delay between two updates * 1000*60*60*6 */
    private final Long minDelay = 21600000L;
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.w("BootAndWifiReceiver", "Intent received :" + intent.getAction());
        // Here we are because we receive either the boot completed event, either the connection changed event
        // either the wifi state changed event
        ConnectivityManager cm = (ConnectivityManager)
            context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = cm.getActiveNetworkInfo();
        if (null != networkInfo) {
            boolean isConnected = networkInfo.isConnected();
            boolean isWifi = networkInfo.getType() == ConnectivityManager.TYPE_WIFI;
            if (isConnected && isWifi) {
                // test if the update have been done 6h ago or not
                SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(context);
                Long lastUpdateTime = prefs.getLong(context.getString(R.string.shLastUpdateTime), 0L);
                Long now = System.currentTimeMillis();
                makeALog(lastUpdateTime, now, "Before update (before the if)");
                if (minDelay < (now - lastUpdateTime)) {
                    // Launch the database update
                    Intent updateServiceIntent = new Intent(context, UpdaterService.class);
                    context.startService(updateServiceIntent);
                    // update the preferences
                    SharedPreferences.Editor editor = prefs.edit();
                    editor.putLong(context.getString(R.string.shLastUpdateTime), now);
                    editor.commit();
                    lastUpdateTime = prefs.getLong(context.getString(R.string.shLastUpdateTime), 0L);
                }
            }
        }
    }
}
```

Gestion des StaredEvents

Utilisation du service StaredEventsService

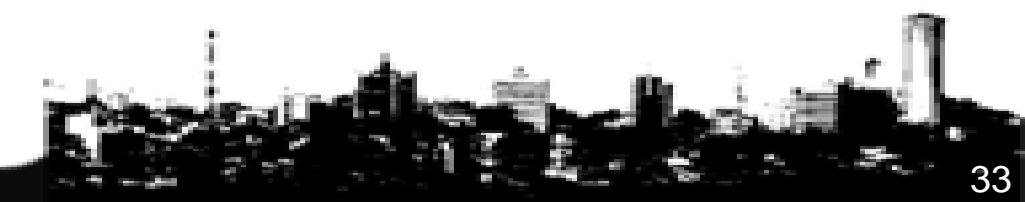
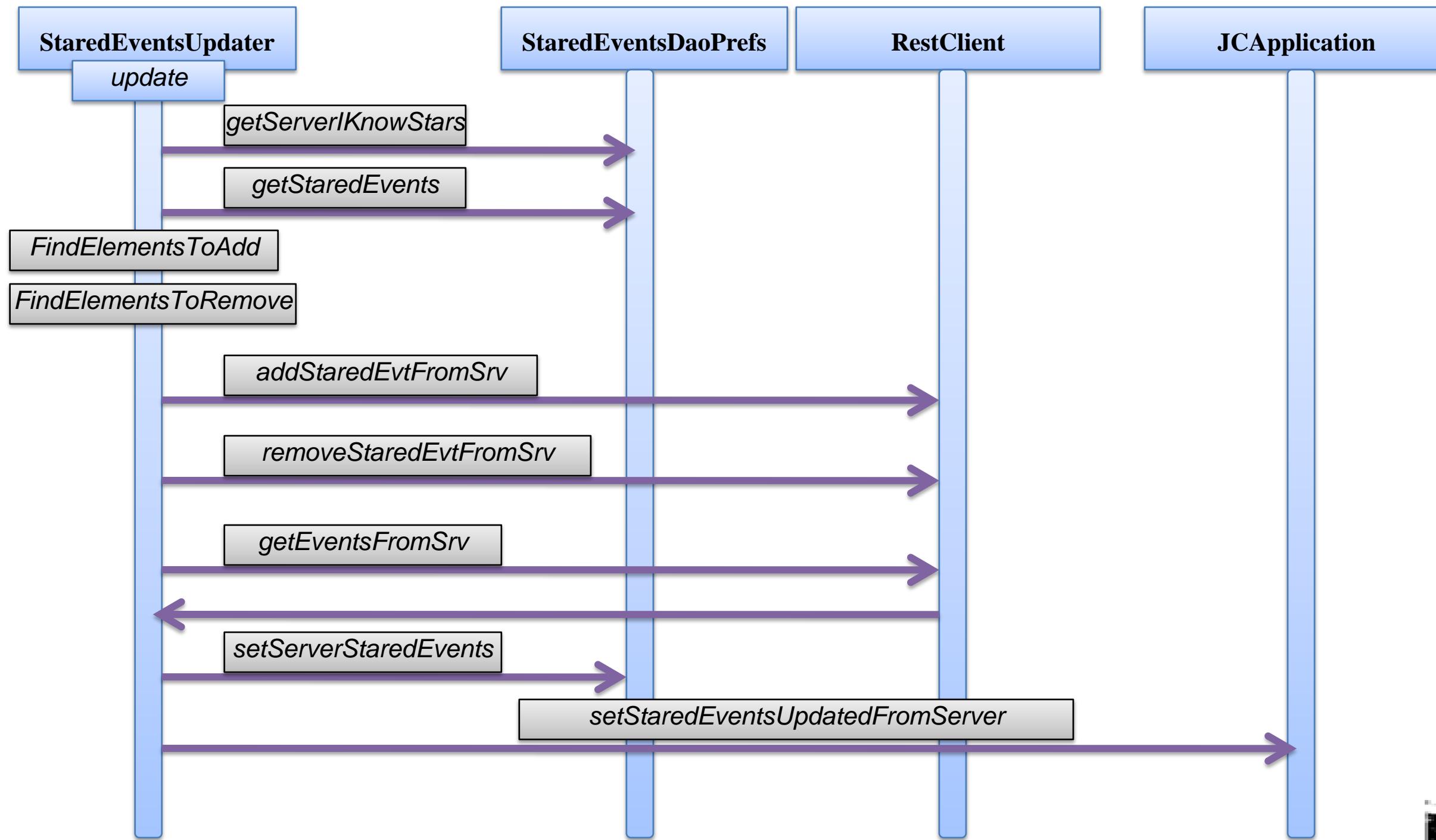
Le StaredEventService est un singleton (enum avec instance) qui possède un cache simple (une liste).



Gestion des StaredEvents

Diagramme de séquence du StaredEventsUpdater

Le StaredEventsUpdater a à charge la synchronisation des éléments favoris avec le serveur web via les services Rest.



Gestion des **Provider

OrmLite prend tout en charge

La simplicité est au rendez-vous, il suffit d'appeler la méthode souhaitée du ***Provider

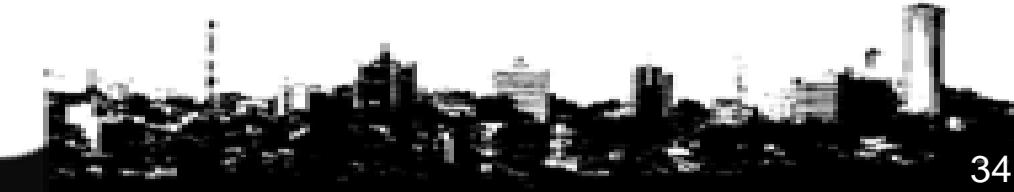
****Fragment**

```
if (eventProvider == null) {  
    eventProvider = new EventProvider();  
}  
  
List<Event> allEvent = eventProvider.getAllEvents();
```



EventProvider

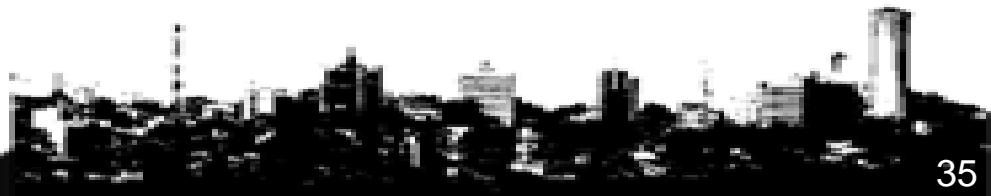
```
public class EventProvider {  
  
    private Context mContext;  
    private DatabaseHelper mDbHelper;  
    private Dao<Event, Integer> mEventDao;  
  
    public EventProvider() throws SQLException {  
        mContext =  
            JCApplication.getInstance().getApplicationContext();  
        mDbHelper = new DatabaseHelper(mContext);  
        mEventDao = mDbHelper.getEventDao();  
    }  
  
    public List<Event> getAllEvents() throws  
        SQLException {  
        return mEventDao.queryForAll();  
    }  
}
```



Projet de Test

A l'abandon

Le projet de Test existe et est à l'abandon.



Références et Tutoriels

Cette conférence utilise les références suivantes:

Les EBooks et tutoriels d'Android2ee : <http://www.android2ee.com>

Les sites de référence Android:



- <http://developer.android.com/index.html>
- <http://android-developers.blogspot.fr/>
- <http://www.google.com/events/io/2011/sessions.html>
- <https://developers.google.com/events/io/sessions>

Sur Android2ee, vous trouverez les tutoriels libres pour approfondir les notions présentées:

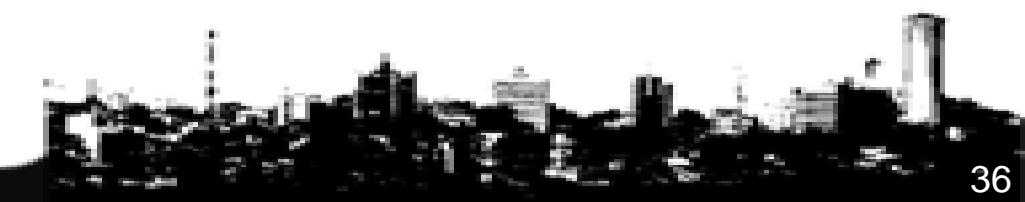


- Les capteurs
- Les thread de traitement (Handler et fuite mémoire)
- Les AppWidgets
- Construction dynamique d'IHM
- Service REST
- Lecteur de flux RSS
- Le fichier POM pour mavéniser vos projets
- Comment préparer ses livraisons sur GooglePlay

Sur developpez.com vous trouverez les articles du conférencier :



- [Déployer son application Android et obtenir sa clef MapView.](#)
- [Construire dynamiquement ses IHM Android](#)
- [Les capteurs Android](#)
- [Thread, Handler, AsyncTask et fuites mémoires](#)
- [Investir l'écran d'accueil Android avec les AppWidgets](#)
- [Android, Livrer son projet sur GooglePlay](#)



Questions, Remerciements

Merci pour votre attention.

Merci à **JCertif Africa 2012**

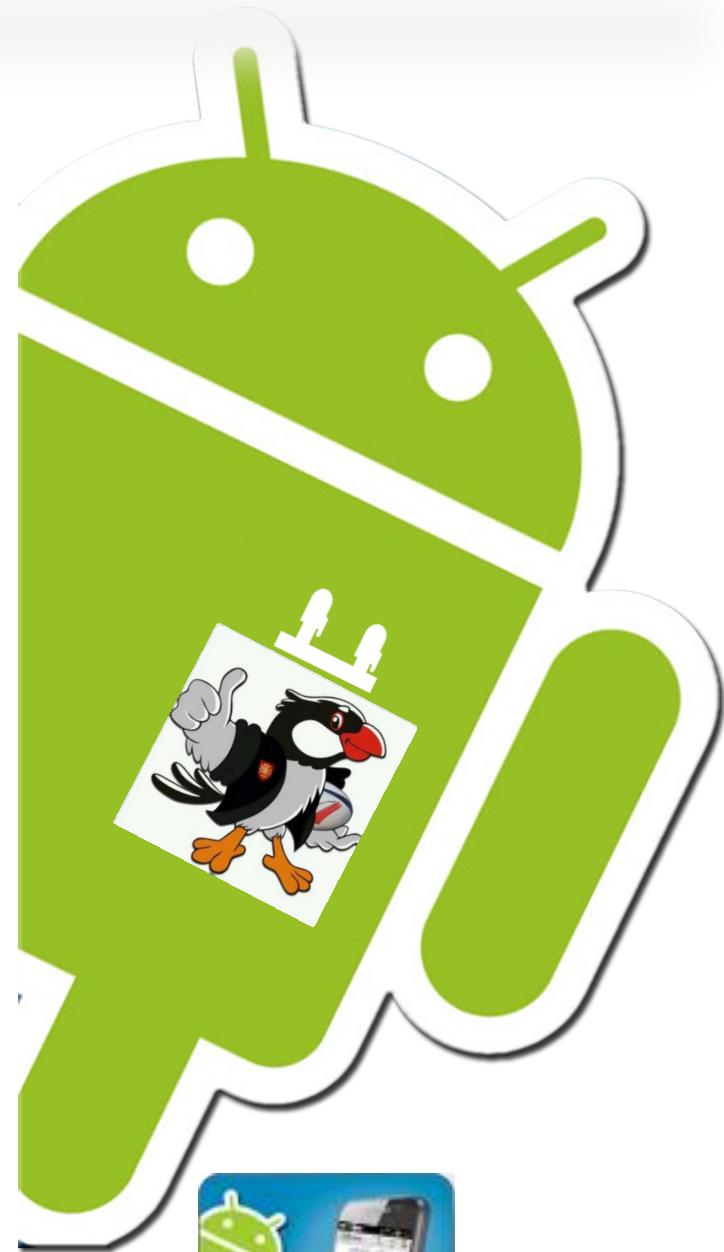


et Bravo !



MySensors, MyLight et MyTorch
disponible sur GooglePlay

android2ee.com.
#android2ee
mathias.seguy@androi2ee.com



Jcertif Mobile
sur GooglePlay

