How to use develops manual:

The introduction explains the existing package structure of the application and explains some of the problems and situations that might have an impact on the application further ahead.

Table of contents:

1. Introduction

1.1. Structure of the package design

The base package of the client side is an abbreviation of the application name with small letters '.ad'. On the client side of the Appsolut Distribute application we are enforcing the MVC –model, the letters 'm', 'v', and 'c' stands for Model, View, and Controller. The ideal usage of this model is to separate the classes and packages into these categories, the purposes of doing so is too keep the code clean and orderly which will increase developers' production rate.
Following description is from www.wikipedia.org:

- A **controller** can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document). It can send commands to the model to update the model's state (e.g., editing a document).
- A **model** notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A *passive* implementation of MVC omits these notifications, because the application does not require them or the software platform does not support them.
- A **view** requests from the model the information that it needs to generate an output representation.

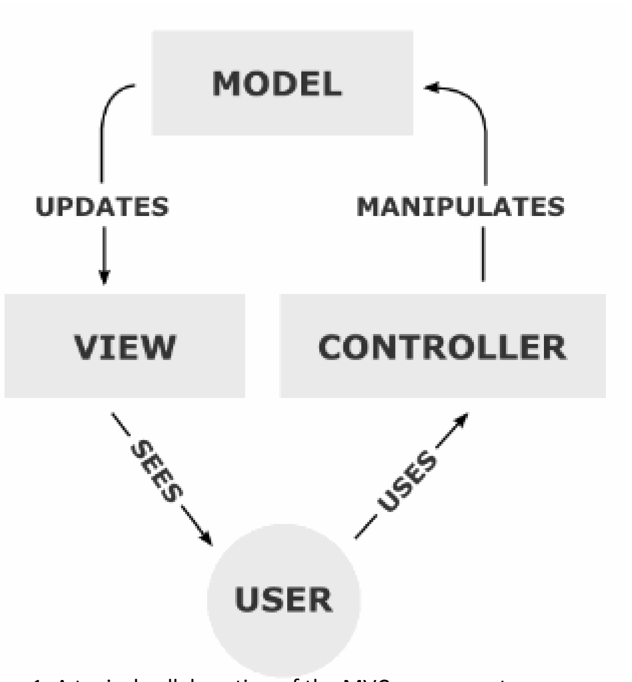
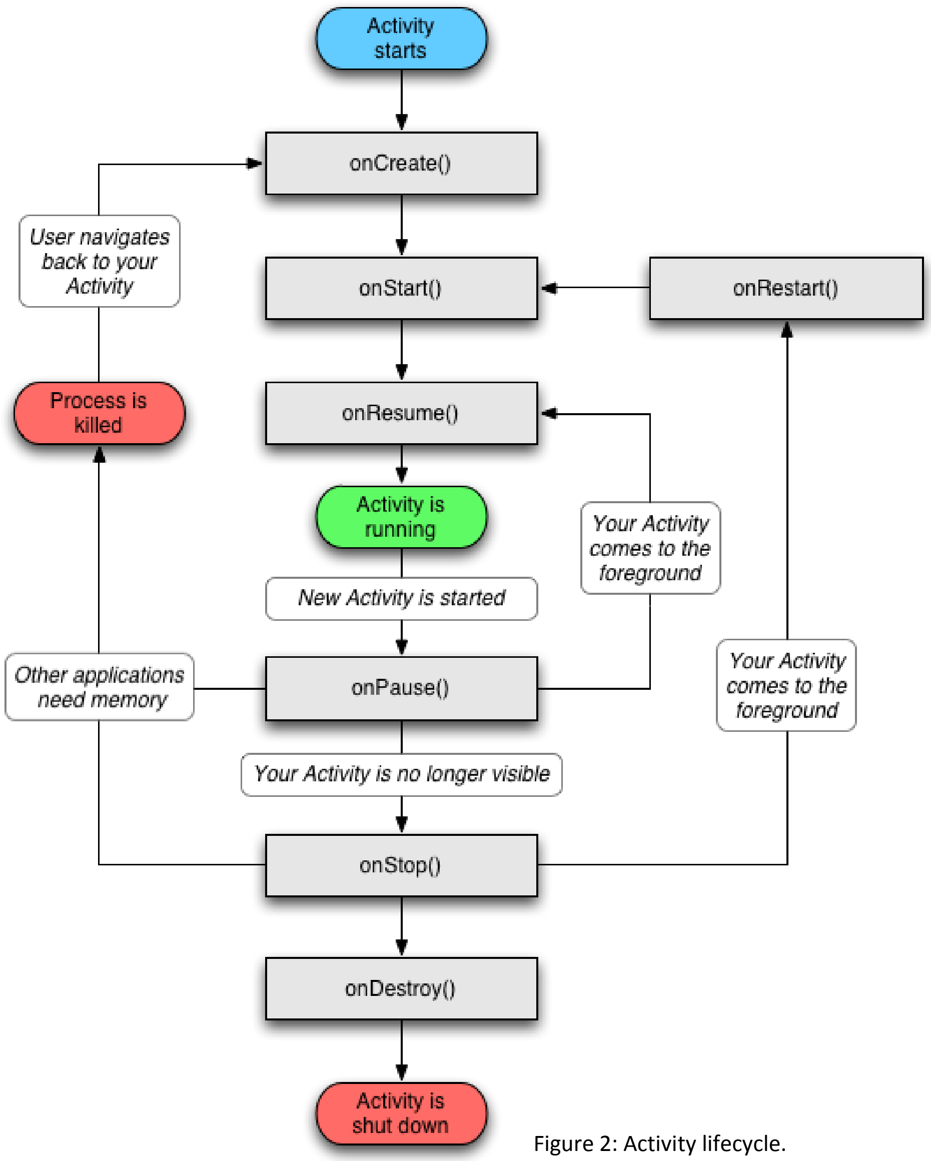


Figure 1: A typical collaboration of the MVC components.

The developed package structure is divided into these three cases. Making the packages in the base package as following:
- '.ad.controller'
- '.ad.model'
- '.ad.view'

1.2. Activity and its lifecycle

An activity creates the window of the application in which the developer can place an UI and is an important part of the applications overall lifecycle. It can also be a smaller floating window (see API) but this solution will not be applied in this application, for now custom made dialogs will be applied. Appsolut Distribute will consist of several activities, because the plan is to make several features and it is likely that each feature will be an activity of its own. The activities will be placed in their own package in view (since they are the window and contain the UI): '.ad.view.activity'
Figure 2 describes android activities lifecycle, the image is from http://www.androidjavadoc.com/:



Activity starts
onCreate()
User navigates back to your Activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
Your Activity comes to the foreground
New Activity is started
Your Activity comes to the foreground
Other applications need memory
onPause()
Your Activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Figure 2: Activity lifecycle.

How will this lifecycle effect the application? If the clients activity is doing an operation, being connected to the server or running a feature it may be halted one way or another. It is important to be able to handle such event and following up on what happens in these cases. To save states, use the Bundle class.

1.3. Structure of views on android

On the android OS there is an underlying structure of how to handle views and their layouts. The views are managed by group views. Each group view and view is managed by several xml files which determine their layouts. Following two images is from API guides on http://developer.android.com.

The figure 3 first is illustrating the relationship between the view group classes and view classes.

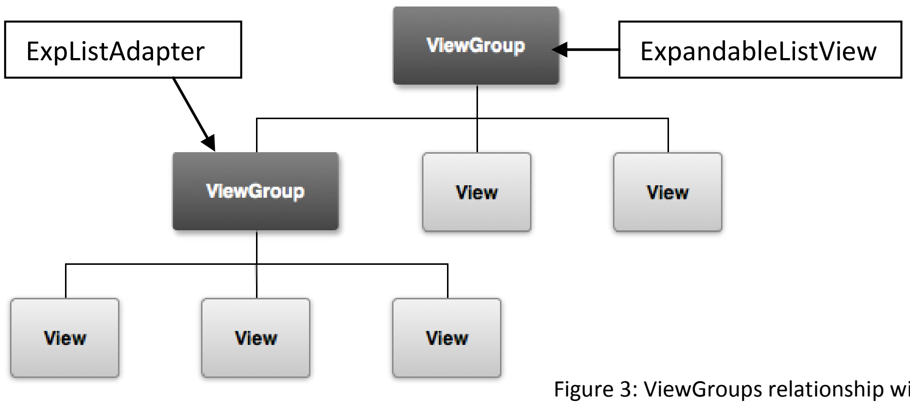
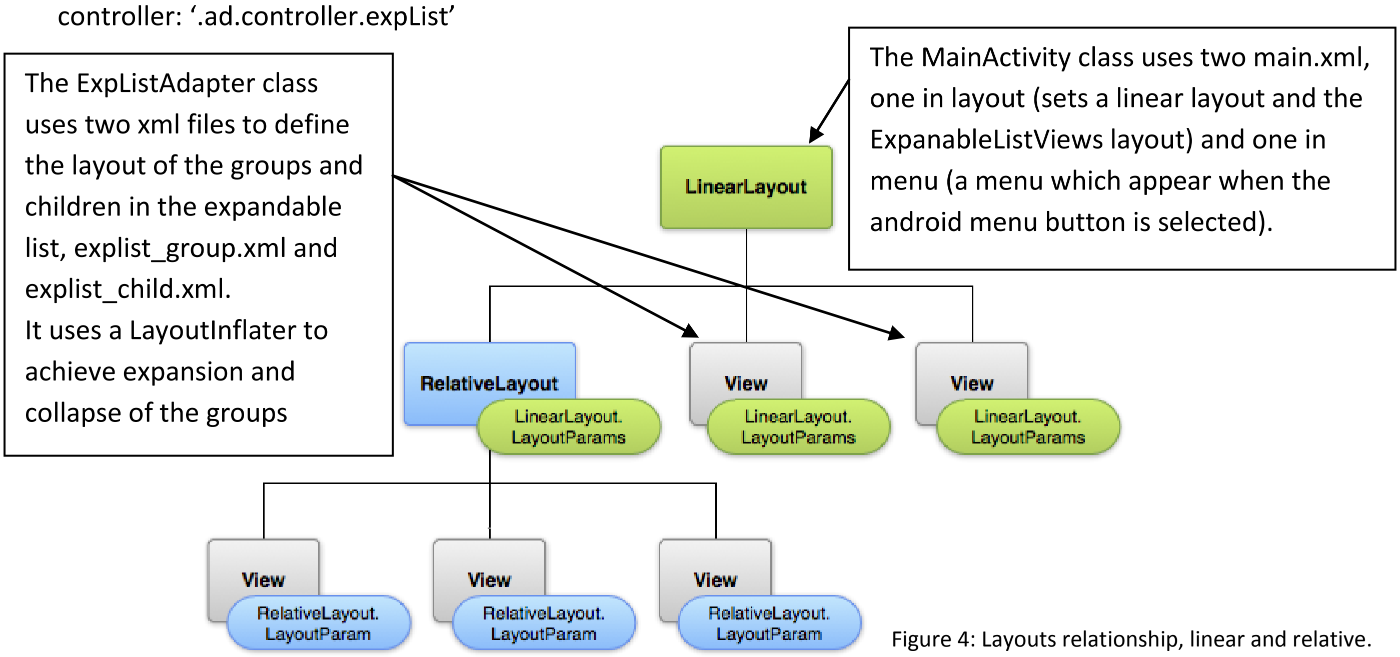


Figure 3: ViewGroups relationship with Views.

This architecture is used by the ExpandableListView class in the main activity class, used by the application. The ExpandableListView needs an Adapter (ExpListAdapter) which acts as a group view for the groups and children (group items) in the expandable list. The adapter updates the model and changes the views presentation, which makes it a controller class located in sub package to controller: '.ad.controller.expList'

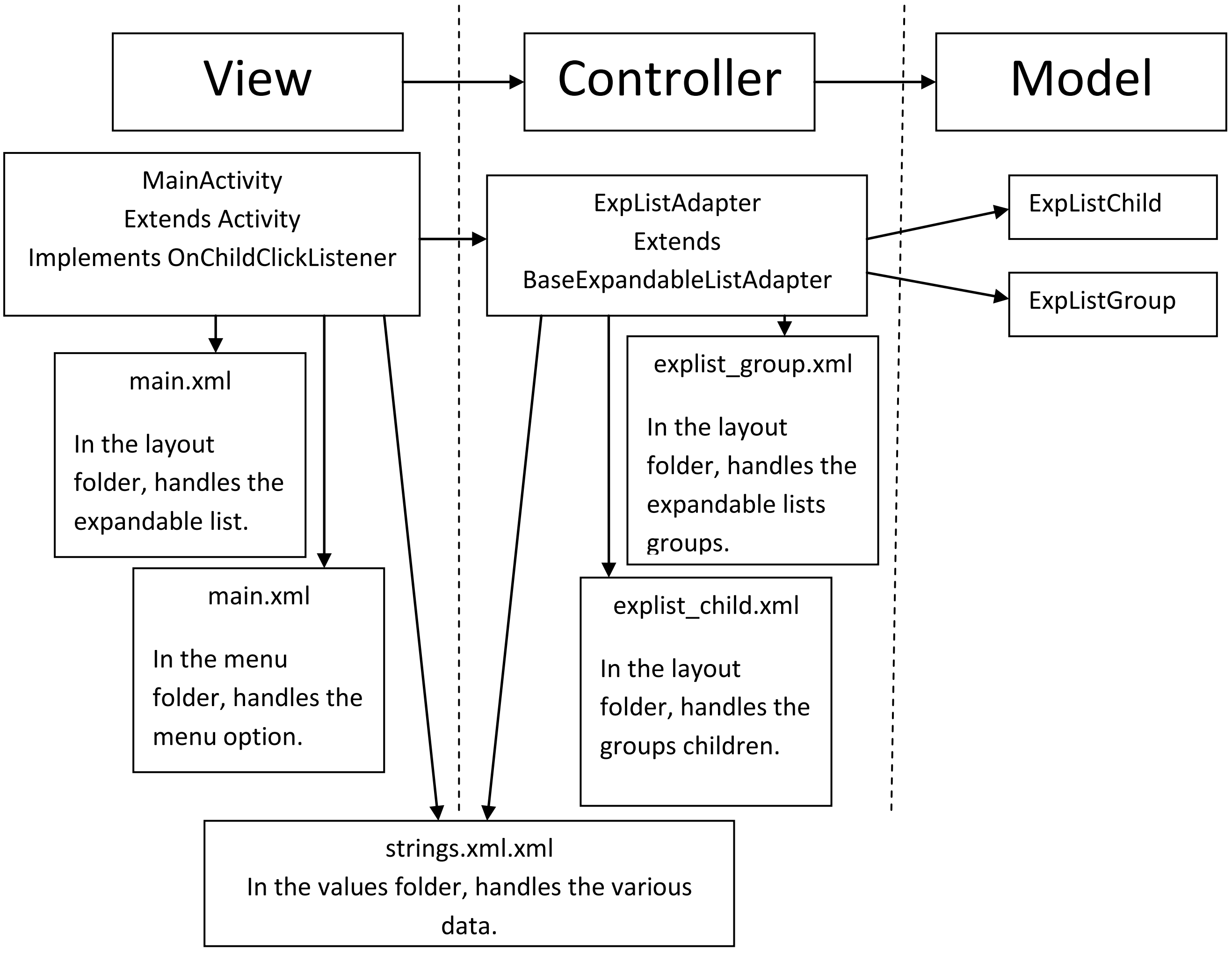

The ExpListAdapter class uses two xml files to define the layout of the groups and children in the expandable list, explist_group.xml and explist_child.xml.
It uses a LayoutInflater to achieve expansion and collapse of the groups

The MainActivity class uses two main.xml, one in layout (sets a linear layout and the ExpanableListViews layout) and one in menu (a menu which appear when the android menu button is selected).

Figure 4: Layouts relationship, linear and relative.

Next is the layout of the view groups and views, they are as said managed by xml files. Note that the relationships have a similar structure in figure 3 and 4. They are closely related and it is important to have that in mind while developing this application.

1.4. Structure of Appsolut Distribute GUI:

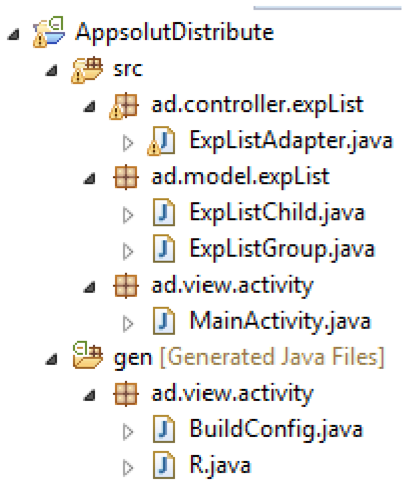

View
Controller
Model
MainActivity
Extends Activity
Implements OnChildClickListener
ExpListAdapter
Extends
BaseExpandableListAdapter
ExpListChild
ExpListGroup
main.xml
In the layout folder, handles the expandable list.
main.xml
In the menu folder, handles the menu option.
explist_group.xml
In the layout folder, handles the expandable lists groups.
explist_child.xml
In the layout folder, handles the groups children.
strings.xml.xml
In the values folder, handles the various data.

There is also a style.xml used by the android manifest xml file.
To have access to the xml files, located in the res folders subfolders, it is necessary to import the R class from a package specified in the AndroidManifest.xml, in our case the import looks like this:
import ad.view.activity.R;
There is also the icon image of the application in the res folder in the drawable subfolders.
An unusual solution was used in the ExpListAdapter due to the lack of support for two dimensional strings for building expandable list menus. The solution was to use a divider in a string array:

```
 <!-- Start menu -->
<string-array name = "start_groups" >
     <item>Connect To Server</item>  <!-- group 1 -->
     <item>Options</item>            <!-- group 2 -->
 </string-array>

<string-array name = "start_children" >
  <item>Login</item>                <!-- Here are the children belonging to group 1 -->
  <item>Create Account</item>
  <item>Select Server</item>
  <item></item>                     <!-- divider: group 1 ends here and group 2 begins -->
  <item>About</item>                <!-- Here are the children belonging to group 2 -->
  <item>Help</item>
  <item>Exit</item>
</string-array>
```

The package structure looks like this:



```
AppsolutDistribute
  src
    ad.controller.expList
      ExpListAdapter.java
    ad.model.expList
      ExpListChild.java
      ExpListGroup.java
    ad.view.activity
      MainActivity.java
  gen [Generated Java Files]
    ad.view.activity
      BuildConfig.java
      R.java
```

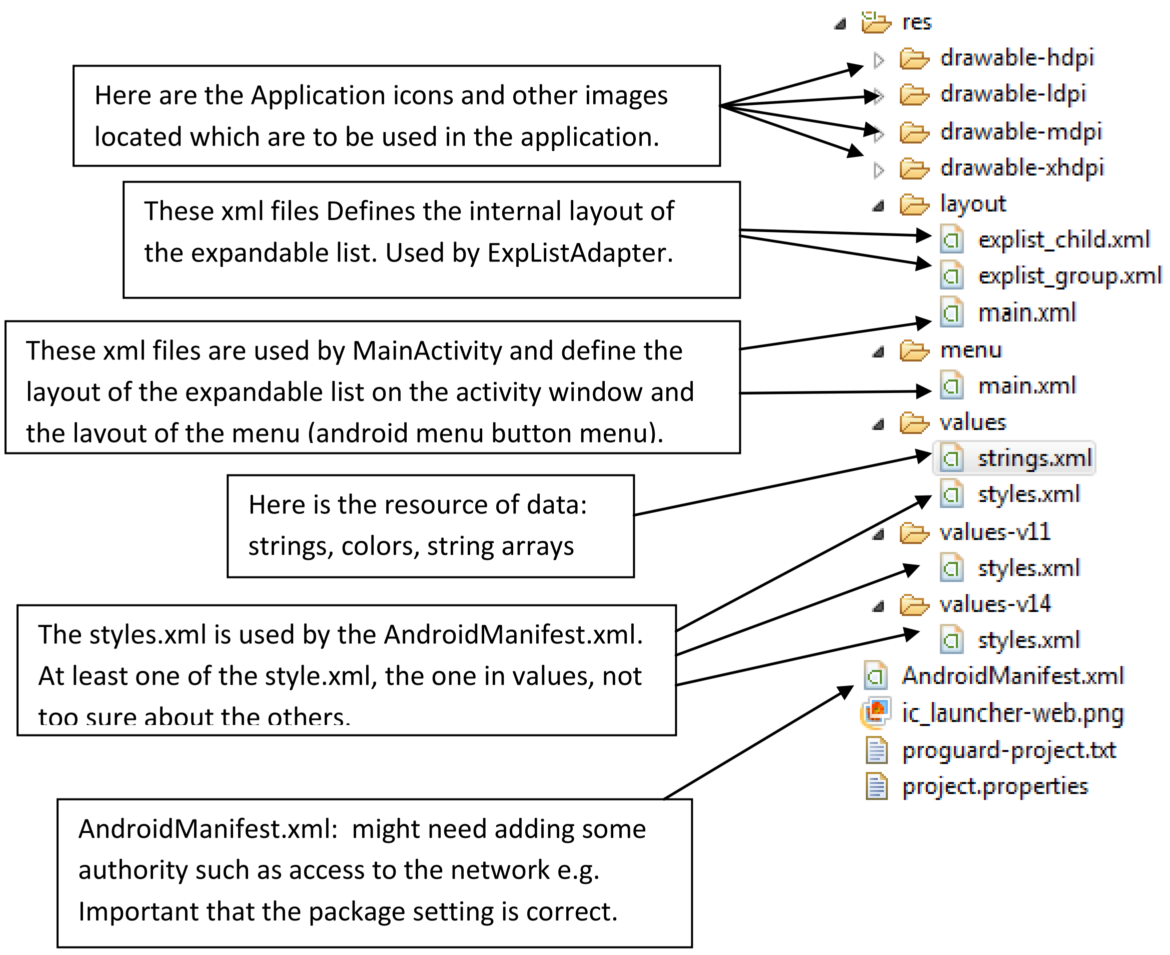
There will be some new packages and classes later, some known classes to be added are:

- Client, in the model package handles communication with server.
- Protocol, in controller package handles the

A class that might be added is:

- OptionDialogs, in View is to handle the communication between user and the Protocol class; it would be used by MainActivity to accomplish this.

The structure of the res folder:



```
res
  drawable-hdpi
  drawable-ldpi
  drawable-mdpi
  drawable-xhdpi
  layout
    explist_child.xml
    explist_group.xml
    main.xml
  menu
    main.xml
  values
    strings.xml
    styles.xml
  values-v11
    styles.xml
  values-v14
    styles.xml
  AndroidManifest.xml
  ic_launcher-web.png
  proguard-project.txt
  project.properties
```

Here are the Application icons and other images located which are to be used in the application.

These xml files Defines the internal layout of the expandable list. Used by ExpListAdapter.

These xml files are used by MainActivity and define the layout of the expandable list on the activity window and the layout of the menu (android menu button menu).

Here is the resource of data: strings, colors, string arrays

The styles.xml is used by the AndroidManifest.xml. At least one of the style.xml, the one in values, not too sure about the others.

AndroidManifest.xml: might need adding some authority such as access to the network e.g. Important that the package setting is correct.

2. Current code structure of the application

2.1. View

In view there is only the MainActivity class currently and the code in the class is divided into sections to make it easier to develop the class.

2.2. Controller

ExpListAdapt

2.3. Model

ExpListChild

ExpListGroup

3. How to add menus.

Add menus

Add menu item

4. Future development plans

Remove menu item

Dialogs

Protocol

Client