Utilisation des projets Android library

par Nicolas KLEIN

Date de publication : 11 octobre 2010

Dernière mise à jour :

Ce tutoriel suppose que vous utilisiez Eclipse en version 3.5.2 (la version 3.6 est déconseillé pour le plugin Android), le SDK en révision 7 et le plugin ADT pour Eclipse en version 0.9.8
Cet article a pour objectif d'expliquer ce que sont et comment utiliser les Android library dans vos projets Android.

I - Présentation

Cette fonctionnalité relativement récente (intégré au SDK en mai 2010 avec la révision 6) permet de factoriser du code commun à plusieurs applications et donc de profiter d'une réutilisabilité plus forte de vos développements.

Elle est compatible avec les versions d'Android suivantes : 1.5, 1.6, 2.1 et 2.2.

La grande différence par rapport à l'utilisation d'un JAR contenant son code source est que les projets library Android permettent également de factoriser les ressources (layout, drawable, ...) et donc de pouvoir réutiliser également des Activities ou des Providers !

Par exemple dans le cas d'une application disponible en version gratuite et payante, toute la partie commune peut être mise dans un projet library. Les 2 versions, payante et gratuite, vont inclure cette library et vont seulement ajouter le mécanisme spécifique de leur version : des pubs dans la version gratuite et le mécanisme de vérification d'achat réel de l'application dans la version payante par exemple.

Le code est du coup beaucoup plus simple à maintenir plutôt que de devoir continuellement copier/coller du code entre les 2 applications à chaque modification de la partie commune.

Au niveau architecture, un projet library Android ne se différencie pas, ou très peu, d'un projet standard Android. Il contient également un fichier manifest, des dossiers src/ et res/. Il est possible d'inclure des layout ou de stocker des images via le dossier standard drawable. Le code source peut utiliser ces ressources via la classe R standard.

Un projet library Android n'est par contre pas compilable en l'état en un fichier .apk. De la même façon, il n'est pas possible d'exporter ce projet sous la forme d'un JAR. A la place, il faut le référencer dans un projet standard Android qui va, lui, générer l'application complète. Il est par contre totalement possible d'intégrer une library JAR dans un projet library et même de référencer un projet library dans une projet library (mais on y reviendra plus tard dans l'article)

Cette génération va fusionner les sources provenant du projet principal et celle venant du projet library. Les ressources (contenu dans le dossier res/) seront également fusionnées. Si une ressource a la même ID dans le projet principal et le projet library, c'est celle du projet principal qui aura la priorité. Cela va permettre entre autres de surcharger des ressources.

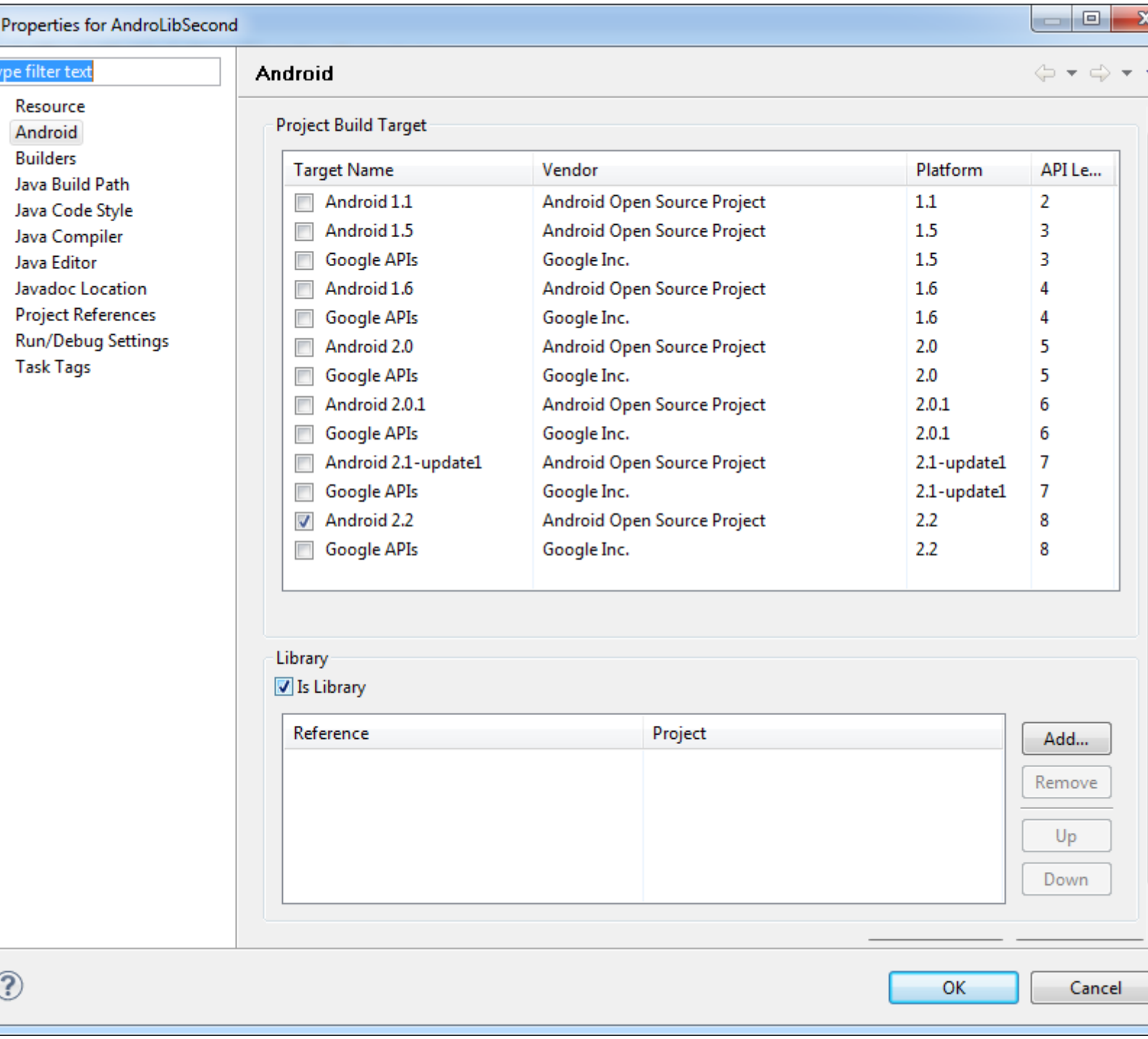
Un projet peut référencer sans problème plusieurs projets library. Cela va permettre de segmenter son projet encore mieux. Comme on le verra tout à l'heure, il va être possible de prioriser un projet library par rapport à une autre (cela est principalement utile pour les ressources).

Les ressources du projet final ayant la priorité, si vous ne voulez pas que quelqu'un qui inclut votre librairie écrase une de vos ressources par erreur en en ajoutant une avec le même nom dans son projet, penser à utiliser des noms spéciaux ou encore mieux à préfixer toutes vos ressources par un identifiant spécifique à votre projet.

II - Création d'un projet library

La création d'un projet library est en tout point identique à la création d'un projet standard Android. Une fois ce projet créé (nommé dans le cas de ce tutoriel AndroLibFirst), nous allons changer un paramètre afin de transformer ce projet standard en projet library:

1. Dans l'onglet **Package Explorer** , faire un clic droit sur le projet et choisir **Properties**.
2. Dans la fenêtre qui s'ouvre, choisir la section "Android". On remarque alors la présence d'une section **Library** en bas de la fenêtre
3. Cocher la case "Is Library" et cliquer sur **Apply**.
4. Valider les changements en cliquant sur le bouton **OK**.



Le projet est maintenant devenu un projet library ! Il est maintenant possible de le référencer dans un projet principal.

Gestion du fichier manifest d'un projet library

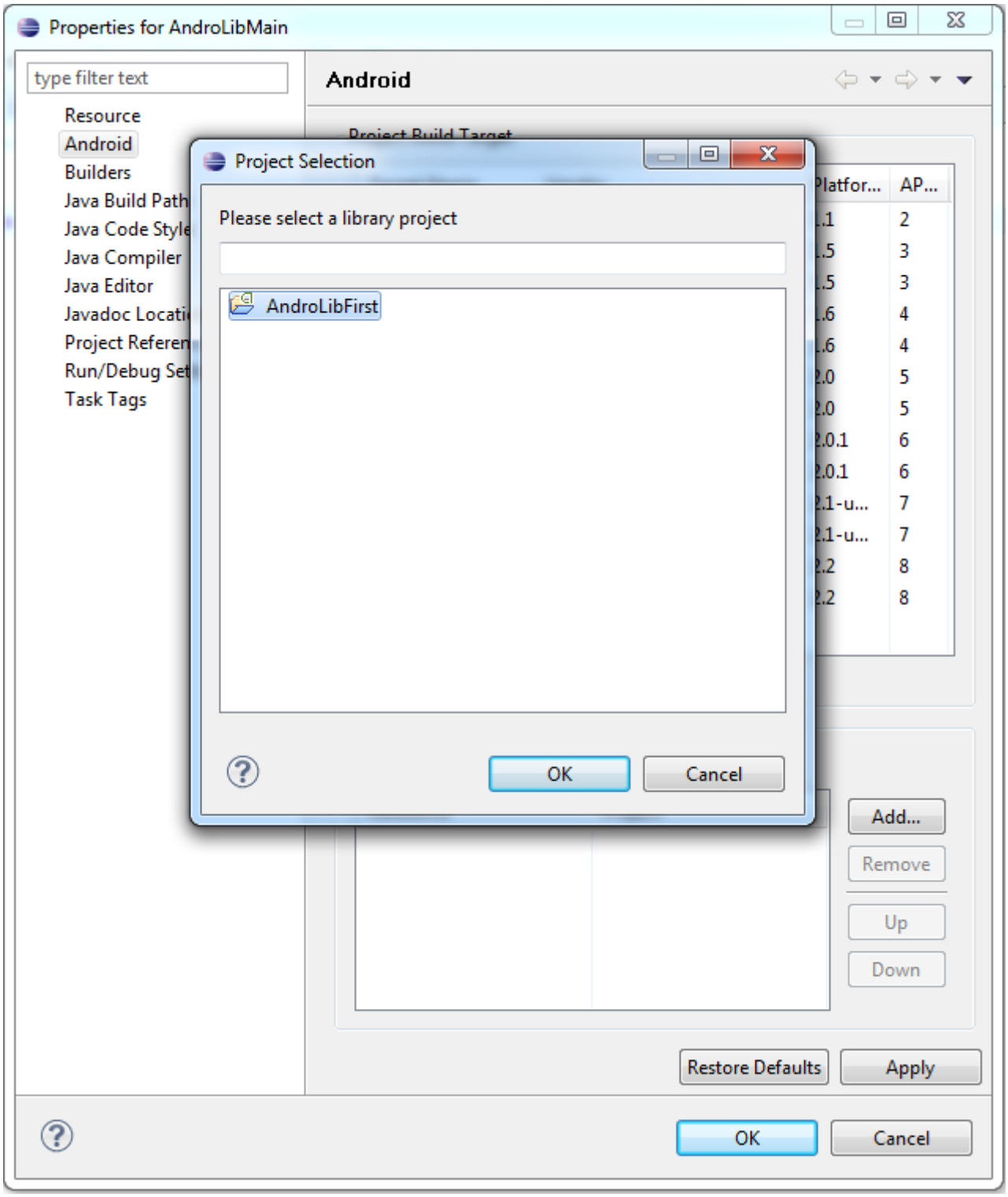
De la même façon que pour un projet standard, le fichier manifest doit définir les composants partagés de la librairie : les Activities, les Providers, ...

III - Référencer un projet library dans le projet principal

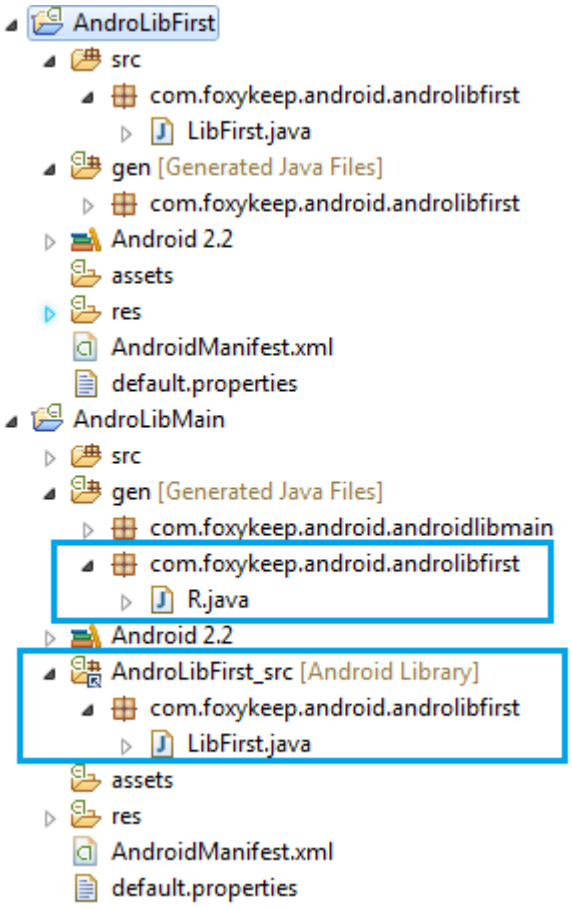
Dans l'interface qu'on vient d'utiliser pour transformer notre projet standard en projet library, on remarque qu'il y a 4 boutons : Add, Remove, Up et Down. Ils vont permettre d'ajouter ou de retirer une référence vers un projet library et d'ordonner les références aux projets library entre elles. Il faut savoir que la librairie la plus haute dans la liste est prioritaire sur celles qui la suivent et que les ressources sont donc choisies en fonction de ces priorités.

Pour ajouter un référence vers notre projet AndroLibFirst depuis notre projet principal AndroLibMain, les actions sont les suivantes :

1. Dans l'onglet **Package Explorer** , faire un clic droit sur le projet et choisir **Properties**.
2. Dans la fenêtre qui s'ouvre, cliquer sur le bouton **Add**. Une popup s'ouvre alors montrant la liste des projets library disponibles dans le workspace.
3. Choisir le projet *AndroLibFirst* que l'on veut référencer et cliquer sur le bouton **OK**.
4. Validez les changements en cliquant sur le bouton **OK**.



La référence est alors mise en place par Eclipse et le code et les ressources du projet library(**encadrés en bleu ci-dessous**) sont maintenant accessibles dans le projet principal comme on peut le voir dans ce screenshot :



Gestion spéciale des composants library dans le fichier manifest du projet standard

Les composants du projet library que l'on veut rendre accessible dans l'APK final doivent être redéfini dans le manifest afin qu'ils soient pris en compte lors de la création de l'APK.

Cela concerne les balises *<activity>, <service>, <receiver>, <provider>* ainsi que les *<permission>* et *<uses-library>* entre autres. Les déclarations doivent utiliser les noms complets avec les packages afin qu'ADT puisse savoir exactement où chercher les éléments.

IV - Fonctionnement spécial : l'imbrication des projets library

On va commencer ici à s'amuser un peu en s'amusant à enchaîner les projets library.

Exemple : un projet principal qui référence un projet library (jusque là c'est standard), qui lui-même référence un projet library qui peut lui aussi référencer un projet library, etc, etc, etc (Toute ressemblance avec Inception n'est que pur coïncidence. On pourrait parler d'Androidception !)

Avant de voir comment faire cela, une chose à savoir ! Cela ne fonctionne que depuis la toute récente (hier soir) mise à jour du SDK et du plugin ADT. Il faut donc avoir le SDK révision 7 et le plugin ADT en version 0.9.8. Mettez à jour votre environnement avant de tester cette nouvelle fonctionnalité donc

Prenons un exemple bidon : un projet principal *AndroLibMain* qui inclut un projet library *AndroLibFirst* qui lui-même inclut un projet library *AndroLibSecond*. Le projet *AndroLibMain* contient une classe *LibMain* qui étend la classe *LibFirst* du projet *AndroLibFirst* qui elle-même étend la classe *LibSecond* du projet *AndroLibSecond*. Totalement inutile comme fonctionnement mais cela ira très bien comme exemple. Voyons maintenant comment réaliser cela :

1. Pour commencer, on va créer nos 3 projets sans se soucier de savoir si ce sont des projets library ou non
2. Une fois cela fait, on commence par transformer le projet AndroLibSecond en projet library. Pour ceux qui ont oublié comment faire, remonter un peu dans l'article pour trouver la marche à suivre
3. On attaque ensuite la nouveauté ! A savoir transformer le projet AndroLibFirst en projet library et référencer dans celui-ci le projet AndroLibSecond.
4. 1. Pour la partie, transformer le projet en projet library, on fonctionne comme précédemment.
   2. Pour ajouter le projet library AndroLibSecond, on remarque que la zone d'ajout d'un projet library n'est pas grisé (ce qui était le cas avant la mise à jour du SDK r7 et du plugin ADT en version 0.9.8). Il nous suffit d'ajouter le projet library comme expliqué précédemment.
   3. On valide le tout. On a donc désormais un projet library AndroLibFirst référençant le projet AndroLibSecond
5. Pour finir, il nous faut référencer ce projet AndroLibFirst dans le projet principal. On ajoute donc cette référence comme précédemment.
6. Et la surprise ! Le projet principal AndroLibMain contient désormais une référence vers le projet AndroLibFirst ce qui est logique mais aussi vers le projet AndroLibSecond. Le plugin ADT s'est chargé de reporter le référence entre le projet AndroLibFirst et AndroLibSecond afin que lors de la création de l'apk du projet principal toutes les librairies nécessaires soient présentes !

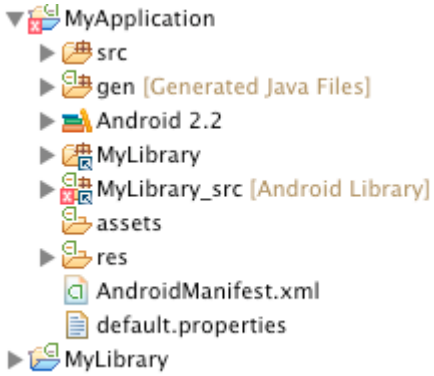
Le résultat est donc le suivant :

On voit donc dans le projet AndroLibFirst les éléments entourés en bleu du référence du projet AndroLibSecond

Dans le projet principal, on a en vert les éléments du référencement direct du projet AndroLibFirst et en rouge les éléments indirectement référencés du projet AndroLibSecond

Migration des projets library de ADT 0.9.7 à 0.9.8

Afin de pouvoir gérer l'imbrication des projets library, le fonctionnement des projets library a été revu entre la version 0.9.7 et 0.9.8 du plugin ADT. La conséquence de cette modification est que si vous ouvrez un workspace ayant un projet library après avoir mis à jour le plugin ADT, vous allez tomber sur ce problème :



La référence MyLibrary est celle d'ADT 0.9.7 tandis que MyLibrary_src est celle d'ADT 0.9.8. Il faut donc déréférencer l'ancienne. Pour cela, les étapes à effectuer sont les suivantes :

1. aire un clic-droite sur MyLibrary et choisir Build Path puis Remove from Build Path
2. Ensuite valider la pop-up qui s'ouvre en vérifiant que le choix □Also unlink the folder from the project□ est bien coché

Le temps que Eclipse rebuild le projet et ca devrait être nickel !

V - Conclusion

Voila ! c'est la fin de ce tutoriel sur les projets library Android. Si vous avez encore des interrogations, n'hésitez pas à poser vos questions dans les commentaires, je ferais de mon mieux pour y répondre et sinon je vous conseille l'article plus complet disponible sur le site officiel Android (en anglais par contre) : **A lire ici**