Aptoide 2017 Summer Internship

Weekly report 7: Aug 21 - Aug 25

Project: Translations on demand
Intern: Válter Santos
Coordinator: Ana Lara Simões
Co-Coordinator: Diogo Loureiro

Completed tasks:

- Fixed HTML bug (more about this in the next paragraph);
- Fixed download and transfer from Database to Memory bug (now transfers directly from the download to memory);
- Hid the first download behind wizard (I'll need Fábio Dias' help for the case when the user skips the wizard before the download is complete);
- Since the master branch of aptoide is very different from the dev branch, I implemented my solution in dev's branch;
- Improved the solution's design

Sadly, the HTML bug isn't a bug, as it appears that HTML elements can only exist inside the strings resources, meaning it's impossible to put them in hardcoded text inside a layout and this prevents the modify xml at runtime solution (because of 26 strings that have the html elements).

I went back to my previous solution, creating a factory and changing the view's translation there and managed to create a Menu Inflater which automatically translates the menu items without requiring to modify their text manually. For the very odd cases that neither use menu xmls nor layout xmls, I let them still call the function that modifies the xml at runtime, as this cases don't use html and problaby never will.

I tried to create a Layout Inflator class instead of a Layout Factory, as the factory only works if there is a switch case that creates a view based on the tag name it receives (Ex: if "TextView", create a TextView; if a "CheckBox", create a Checkbox). This has the problem of having to add new views to the switch case whenever new views that have text are used (although this should rarely happen and is a very small issue). However, creating a layout inflater class and translate the text there requires to parse the entire list of view's attributes inside the layout xml. This can easily get very complex and very hard to maintain in case something changes.

However, I had the idea of using both solutions in order to resolve the HTML problem. What this means is instead of hardcoding the translated text inside the binary XML, I hardcode the string ID inside the binary xml, and since this is hardcoded text, the views don't replace the ID by their default translation, which allows me to iterate over every view of a viewgroup inside a custom Layout Inflater and check if it's a TextView and in the case it is, then I read the text (which I'll know if it's a string ID) and in case it is, then I translate it there, which works with HTML elements.

So the solution is:

CustomLayoutInflater -> calls normal layout inflater to do its business;
Normal LayoutInflater -> calls getLayout() to receive binary XML of the views;
getLayout() -> return a modified binary XML with string IDs hardcoded as text, so they aren't modified afterwards by the android framework;
Normal LayoutInflater -> returns a view / viewgroup which contains views that in the case they have text, said text is the string Ids
CustomLayoutInflater -> Iterate over every view and check if it has text, if it does, call custom resources and replace the string ID inside the text with the translation (or default text if there isn't one).

The same thing happens for the menus with MenuInflater.

Next week:

- Update strings with dev strings (some text isn't being translated because I'm using master branch strings and their IDs changed in dev);
- Deal with wizard in case the user skips while the translation is still being downloaded;
- Fix a bug with arrays (this happened due to a design change, arrays are always awkward to deal with due to being made of string pointers)
- Test

About ProGuard, I'm not sure if I'll get on it and I'd rather make sure the translations solution is absolutely perfect with zero overhead for any programmer working on Aptoide , although I believe this solution is final as there aren't any bugs that can cause problems.