Zeeguu Android - Codebook

This book is meant as a handbook for programmers who want to extend the Zeeguu application. It contains the most important extension possibilities and how to apply them.

CODEBOOK WAY PREALPHA!

# Add new application languages

You have to create a directory into the res folder, which is called "values-<language code>". As language code, you need to take the short cut that android takes for the language. After that, add a "strings.xml" file inside this folder and put in it all strings in that language with the correct identifier. You probably want to copy paste from another "strings.xml" file that you already have and then just change the values of the words. See here for further information: [LINK](http://developer.android.com/training/basics/supporting-devices/languages.html)

At this stage, you have a functional language kit and your application would show up in that language provided your phone is set to that language in the main settings. In the case that your phone runs on a different language, you are not going to recognize any change in the application. For this case, we add the functionality that you can change the language during runtime.

To add some new languages for the application itself, you need first to extend the strings, which define the languages that are shown in the settings. So go to "strings.xml" (you have to do the change in every language) and add the language as an item in the "choosable\_languages" string-array. Also add the "choosable\_languages\_keys", for which you can take the android language shortcut.

Now we would have to add the new language to the code that changes the output language of the app, but unfortunately, this is not written yet. So it will be implemented later. //TODO!

# Add new native and learning language

It's pretty similar to new application languages, except that those two languages are on the server. So first add them to the preferences and name the keys the same as the server calls these language keys (you can check it with a request "/learned\_language" or "/native\_language").

After this, the language would already work, because the translation itself just adds the language key and so Zeeguu knows with what for languages it is dealing. But you don't see any flag yet, so you probably should think about adding them.

# Add new flags

To add new flags, you have to add them into the resources folder. There you find different "drawable-<resolution>" folders, where you have to insert 4 different pictures with different sizes:

* mdpi 32 x 32 pixels
* hdpi 48 x 48
* xhdpi 64 x 64
* xxhdpi 96 x 96

After you have done so, you can refer to this flag from everywhere just by calling @drawable/<name\_of\_flag>. Now we need to implement this flags into the dynamic play of the translator game.

In the Fragment\_Text class, we need to implement the flags into the setFlag method. In this method, the flags will be added dynamically to the EditText boxes. So just add a new case in the switch statement for every flag.

If you have done everything correctly, you should be able to use your new language now.

# Add new design

If you want to add new icons, put those 4 icons, like explained in the section flags, into the "drawable-<resolution>" directory. From now on, you can reference that icon and it should appear.

If you want to change or add a whole design, you first need to re/define the new style in the "styles.xml" sheet. In case you want to add a new icon in your style, add it also in the "attr.xml" file and look out that you call both of them the same, so that they can make a link to each other.

If you have a new style, go to the "strings.xml" and add in the "choosable\_themes" and " choosable\_themes\_keys" the name and shortcut of your new language. Finally go to ZeeguuActivity and add the new language in the "setTheme" method. And voila, you have a new theme.