

**Pick Pic**

**Documents for developers**

**2017 - 1 Human ICT Software Engineering | 2017.06.02**

|  |  |
| --- | --- |
| 20155652 | KangSoYeon |
| 20154019 | LeeYeWon |
| 20151722 | MyungSeKyo |
| 20152164 | LeeWonJun |
| 20151535 | ParkGunHoo |

Table of Contents

[1. How to obtain the source code 3](#_Toc484036389)

[2. The layout of our directory structure 3](#_Toc484036390)

[3. Self explains of source codes 4](#_Toc484036391)

[PickPic Client 4](#_Toc484036392)

[PickPic Server 7](#_Toc484036393)

[4. How to build the software 8](#_Toc484036394)

[4.1 Server 8](#_Toc484036395)

[4.1.1 set up python 8](#_Toc484036396)

[4.1.2. set up node js 8](#_Toc484036397)

[4.1.3. set up tesseract-ocr 8](#_Toc484036398)

[4.1.4. set up modules to node js 8](#_Toc484036399)

[4.1.5 run server 8](#_Toc484036400)

[4.2 Client 9](#_Toc484036401)

[5. How to test the software 9](#_Toc484036402)

[6. How to set up an automated daily build and test 14](#_Toc484036403)

[7. How to release a new version of our software 14](#_Toc484036404)

[8. How to access the list of outstanding bugs and the list of resolved bugs, and how to resolve a bug 15](#_Toc484036405)

# How to obtain the source code

We use github to share code, so the source code could be obtained through github. Because our program is open source program, anyone who want source code can obtain a code through github. You should access the link below.

* client

<https://github.com/PickPic/PickPic_Client>

(if you have git or git bash, then enter “git clone <https://github.com/PickPic/PickPic_Client.git>”)

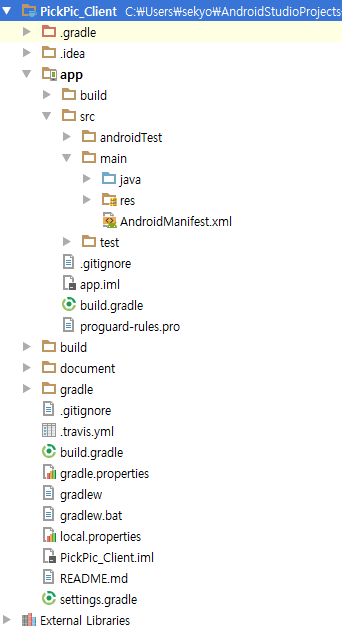
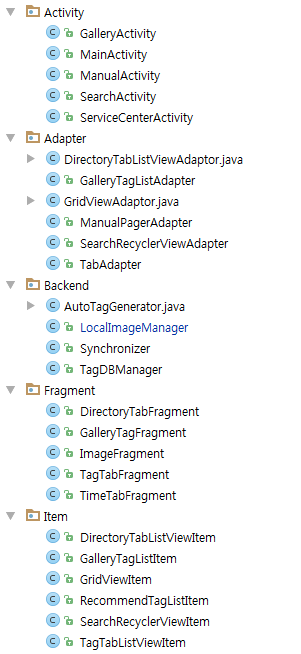
* server

<https://github.com/PickPic/PickPic_Server>

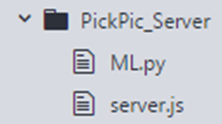
(if you have git or git bash, then enter “git clone <https://github.com/PickPic/PickPic_Server.git>”)

# The layout of our directory structure

1. PickPic’s android client part basically follows the directory structure of the Android project. And each class is organized in the following directory structure.

1. PickPic’s server part is organized simply in the following directory structure



# Self explains of source codes

## PickPic Client

* Activity
* GalleryActivity.java: This is an image detail view activity class.
  + - changeClickListener : This listener call ChangeFragment method when change button is pressed.
    - ChangeFragment(View v) : This method switch two fragments, ImageFragment and GalleryTagFragment.
    - backActionListener : This listener returns to the previous screen.
    - onKeyDown(int keycode, KeyEvent event) : When a back button of the device is pressed, it do the same action as when the back button above is pressed.
    - menuClick(View v) : It organize and pop up the menu screen
    - menuItemClickListener : This method performs the operation of each item of the menu.
    - ShareImage() : This method invokes sharing applications.
* MainActivity.java: This is the application's main activity class.
* ManualActivity.java: This activity is application’s manual
* SearchActivity.java: This is an activity class for tag search.
* ServiceCenterActivity.java: This activity shows service center page.
* Adapter
* DirectoryTabListViewAdaptor.java: It is an adapter for list view of directory tab.
* GalleryTagListAdapter.java: It is an adapter for the gallery activity's tag list view.
  + ArrayList<GalleryTagListItem> galleryTagListItemArrayList : This is an ArrayList variable that stores GalleryTagListItem.
  + getView() : This method construct taglist and include the codes that is related to delete tag.
* GridViewAdapter.java: It is an adapter for a grid view in which images are directly displayed.
* ManualPagerAdapter.java: It is an adapter for image sliding of manual activity.
* SearchRecyclerViewAdapter.java: It is an adapter of recycler view for multi-search.
* TabAdapter.java: It is the adapter for the tab in the main activity.
* BackendModule
* AutoTagGenerator.java: a class that communicates with the server and generates tags for the photos.
  + class WaitSlot: Temporary case to limit the number of pictures to send at one time so that there is no data overload in the picture transfer
  + void autoTagGenerate(final Context context, String path): If the image path is passed as an argument, the image is sent to the server, and the tag is received and stored in the DB.
* LocalImageManager.java: the class that manages accessing the internal repository
  + ArrayList<GridViewItem> getGridViewItemList(Context context, ArrayList<String> paths) : Passing a list of image paths will create a list of grid view items.
  + String getBucketName(Context context, String bucketId) : Passing the bucket ID returns the bucket display name.
  + ArrayList<GridViewItem> getTimeTabGridViewItemList(Context context) : When the function is called, it returns all image list as grid view item.
  + String getDateByPath(Context context, String path) : Passing the image's path as an argument returns the date of the image.
  + GridViewItem getGridViewItem(Context context, String path) : If you pass the path of the image as an argument, it will be made as a grid view item and returned.
  + ArrayList<String> getAllImagePath(Context context, String order): Passing the sort order as an argument returns a list of paths to all images.
  + ArrayList<GridViewItem> getImagesInDirectory(Context context, String bucketId) : Passing the bucket ID as an argument will return the path to all photos in the specified directory.
  + ArrayList<DirectoryTabListViewItem> getDirectoryTabListViewItem(Context context) : Returns the list view item in the directory tab.
* Syncronizer.java: a class that handles synchronization tasks.
  + Void onPreExecute(): Creates a loading dialog in the UI Thread.
  + Void doInBackground(Void... voids): First check deleted pictures and delete them from DB and the newly added photos, add them to the DB, and generate tags.
  + void onPostExecute(Void voids): Delete the loading dialog.
* TagDBManager.java: the class that manages the database that stores the tags
  + ArrayList<String> getAllImages(): Retrieves the path to all images stored in the DB.
  + void initTable(): Initialize the DB.
  + ArrayList<String> getToBeErasedPaths(ArrayList<String> paths): Retrieves a list of image paths to be erased.
  + void insertImagesIfNotExist(ArrayList<String> paths): If you pass image paths as arguments, only non-existent images will store information in the DB.
  + ArrayList<String> getPathsNotAutoGenerated(): Returns a list of images that have not been tagged.
  + ArrayList<TagTabListViewItem> getTagTabListViewItem(): Returns the list view item list of the tag tab.
  + ArrayList<String> getTagsByPath(String path): Passing the image path as an argument passes the tags of the image.
  + ArrayList<String> getRecommendTagList(ArrayList<String> inputedTag, String inputingString):
  + ArrayList<String> getPathsByTags(ArrayList<String> tags): If the tag list is passed as an argument, multiple search is performed and the retrieved image list is returned.
  + ArrayList<String> getPathsByTags(ArrayList<String> tags, String dirTag): Passing an input string as an argument returns a list of tags similar to that.
  + void insertImage(String path): Store image information in DB.
  + insertTag(String path, String tag, int tagType): Save the tag in the DB.
  + void removeImage(String path): Erase image information from DB.
* removeTag(String path, String tag): Delete the tag from the DB.
* Fragment
* DirectoryTabFragment.java: the fragment class of the directory tab of the main activity.
* GalleryTagFragment.java: a fragment class that populates the image information in the image detail view.
  + ArrayList<GalleryTagListItem> galleryTagListItems : This is an ArrayList variable that stores GalleryTagListItem.
  + OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle saveInstanceState) :
  + taginputDialog() : This method pops up a notification window for the user to input the tag.
* ImageFragment: It is a fragment for image detail view.
  + PhotoViewAttacher photoViewAttacher: It is an open source variable that zoom in / out of the image.
  + rotateImage() : This method is called when you click the Rotate menu in the GalleryActivity.
  + rotateImage(Bitmap src, float degree) : This method contains actual code that rotates the image.
* TagTabFragment: This is a fragment showing the tag tab in the main activity.
* TimeTabFragment.java: the fragment class of the time tab of the main activity.
* Item
  + DirectoryTabListViewItem.java: It has directory information as an item class in the list view of the directory tab.
  + GalleryTagListItem: This is an item in the tag list view of the gallery activity.
    - setTag(Sring tag) : This method is used when the user enter a new tag.
  + GridViewItem: An item in the grid view that shows the image thumbnails.
  + RecommendTagListItem: It is a list view item that shows recommend tag when searching.
  + SearchRecyclerViewItem: This is an item in the recycler view for multiple searches.
  + TagTabListViewItem: It is an item for list view of tag tab.

## PickPic Server

* PickPicServer
  + ML.py : google’s Inception v3 model. This is trained for the ImageNet Large Visual Recognition Challenge using the data from 2012, and it can differentiate between 1,000 different classes, like Dalmatian or dishwasher - <https://goo.gl/U8QG8m>
    - create\_graph() : load trained graph
    - run\_inference\_on\_image(image) : printf a tag about the image
    - maybe\_download\_and\_extract() : If you don’t have data about the trained data, then it download the data.
  + Server.js : node js server with express. It takes an image.
    - creates a tag for the object with ML.py
    - creates a domain color tag using domain-color module
    - extracts the text from the image using textract

# How to build the software

## 4.1 Server

### 4.1.1 set up python

PickPic\_Server use Tensorflow for Python, so you need to set up python and Tensorflow for python.

To set up python: https://www.python.org/

To set up Tensorflow: https://www.tensorflow.org/install/

### 4.1.2. set up node js

PickPic\_Server is node js server, so you need to set up node js.

use https://nodejs.org/

### 4.1.3. set up tesseract-ocr

To extract text from image, so you need to set up tesseract-ocr.

sudo apt-get install tesseract-ocr

sudo apt-get install tesseract-ocr-eng

\*if you want detect more language from image,then see "<https://ubuntu.flowconsult.at/linux/ocr-tesseract-text-recognition-ubuntu-14-04/>" and "<https://github.com/tesseract-ocr/langdata>"

\*if you are not linux, then see "<https://github.com/tesseract-ocr/tesseract/wiki>")

### 4.1.4. set up modules to node js

PickPic\_Server use variable modules for ndoe js, so you need to set up modules for nodejs.

npm install

### run server

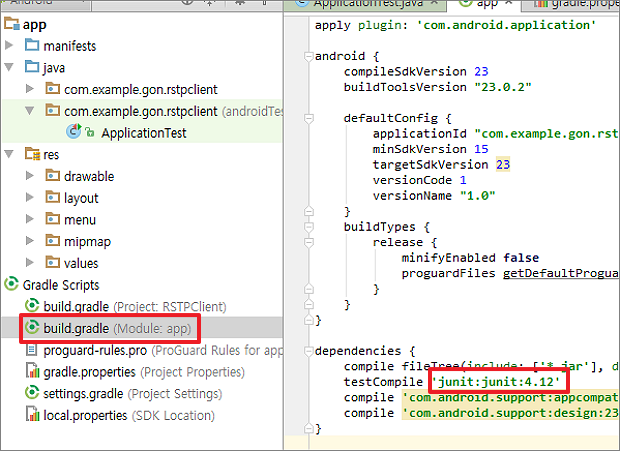
Enter “sudo node server.js” to execute server

## 4.2 Client

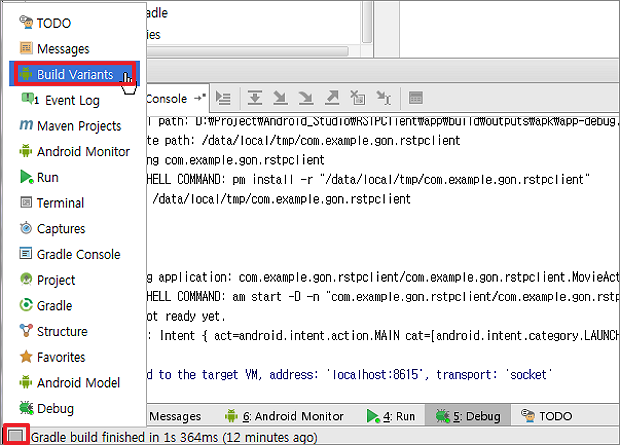
Install android studio and run PickPic code(<https://github.com/PickPic/PickPic_Client>) with Android Studio.

# How to test the software

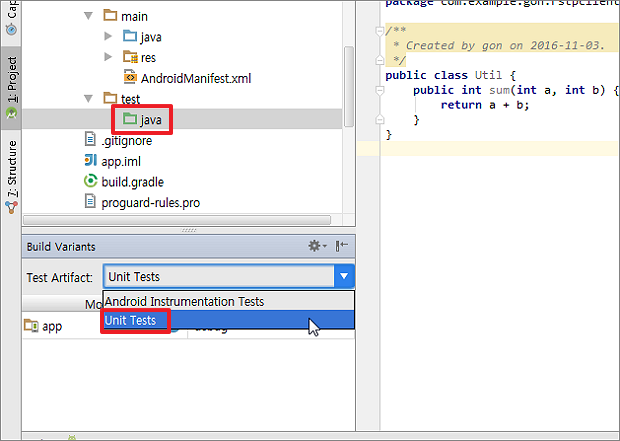
▼ First you need a library for Local Unit Testing. By default, when you create a project, junit is added to the build.gradle library. If it does not exist, you will have to type it directly in dependencies. Then click Sync now to download the library.



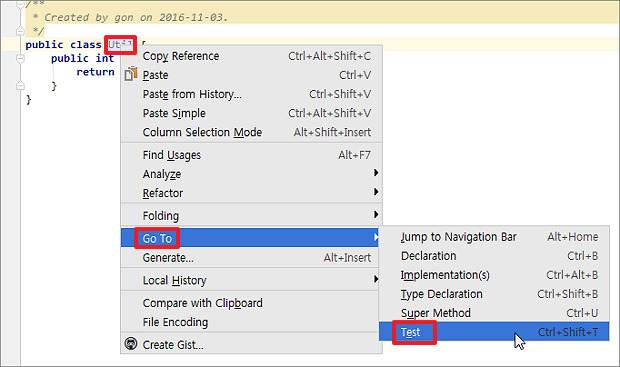
▼ Next, you need to create a folder to contain the unit test files. This is called Test Artifact. When you create a unit test for a particular class, it refers to the folder where the file is to be added. When you hover your mouse on the bottom left corner of the screen, a menu popup window appears. Select Build Variants from the list.



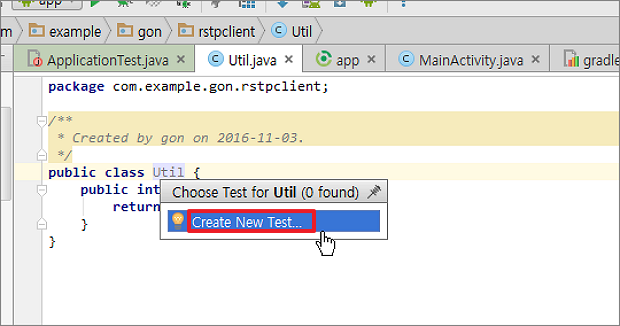
▼ When the Build Variants screen appears, select the newly created folder java and assign the Test Artifact as Unit Tests. Then the color of the folder turns green. Now, when you create a unit test class, the file will be created here. However, if you create a project in the Android studio, you do not have to do this. By default, the folder for Test Artifact, androidTest> java, is created.



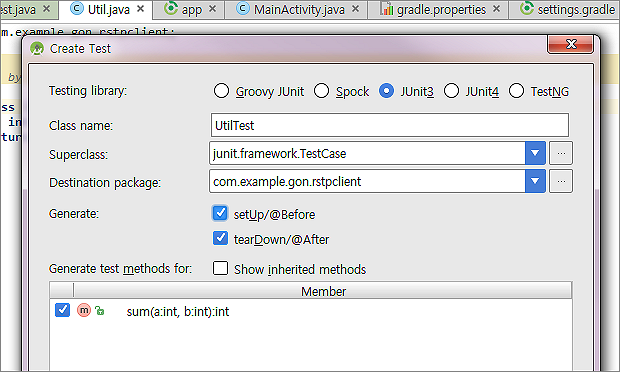
▼ Go to the class that requires unit testing and right-click on the class name and select Go To> Test menu. This is the procedure for creating the unit test code for that class.



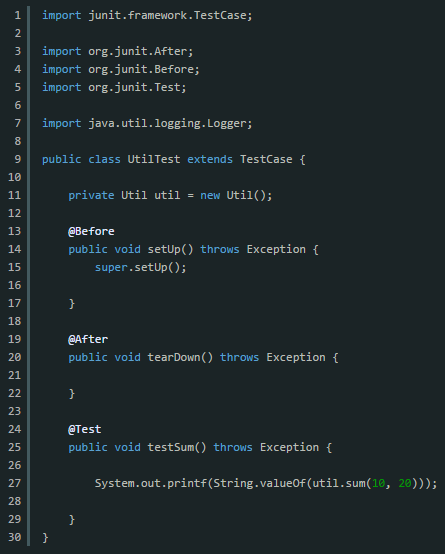
▼ When the pop-up window appears, select the Create New Test menu.



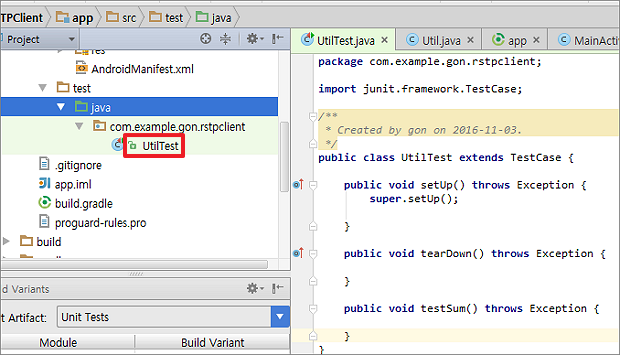
▼ To create a unit test pop-up window, select items Test library and Generate. Leave the rest as default. The test library is set to JUnits3 by default, but JUnits4 is also available. Two options of Generate are setup / testDown asking whether to create a function that inserts code to be processed before and before execution. When the upper option check is finished, go down to check the function to be tested and press the OK button to close the window.



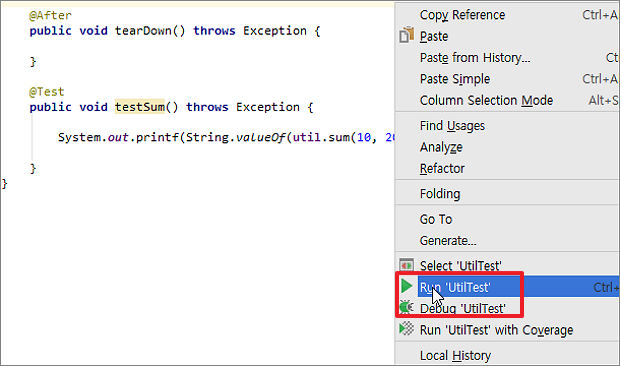
▼ When you finish setting in the Create Test popup window and close it, unit test class is created immediately. Now enter the function of the Util object to test in the testSum function.



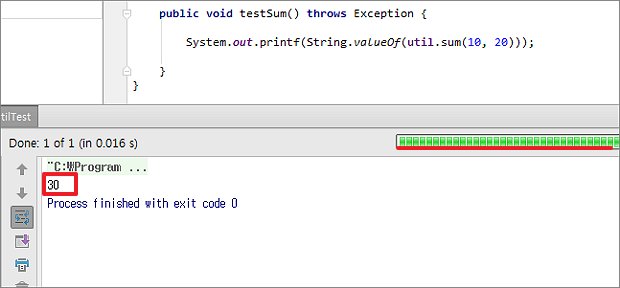
Go to the test folder to run the unit tests you have created so far. Then you will see the UtilTest class in the same package address as the Util class.



▼ Right-click and select Run / Debug from the pop-up menu. If you want to debug by hanging a breakpoint, you should choose Debug 'UtilTest' menu.



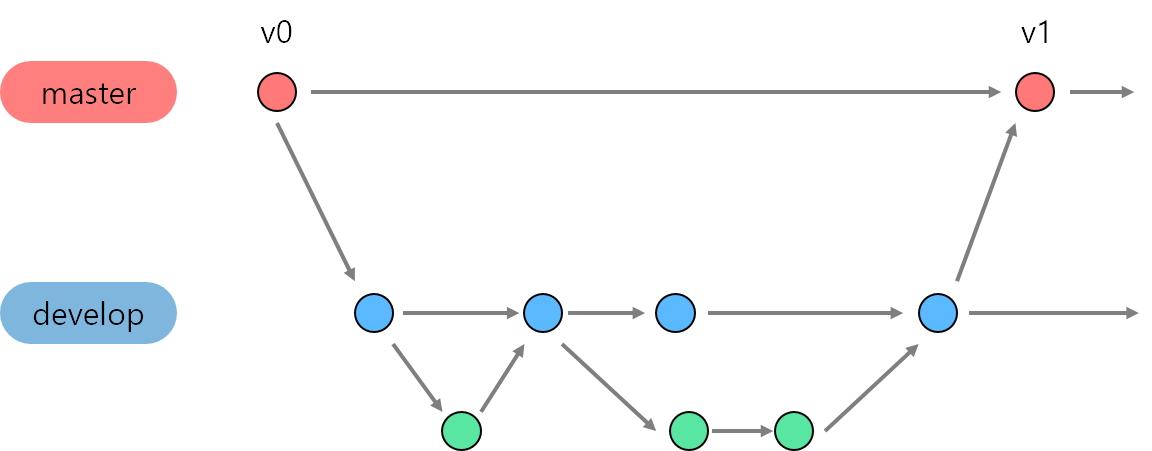
▼ The result of the execution is as follows. I got the result of calling the sum function of the Util class on the console. So far, we've seen how to make unit tests for specific classes easy and fast.



# How to set up an automated daily build and test

We use Travis CI with github. So no other settings are needed

# How to release a new version of our software



Our git strategy is ‘git-flow’. Please refer to above image.

Master branch manage the version of application. That is, a new version of application is released through Master branch. The process of development will occur in Develop branch. After sufficient development has been done, updated source code will merge to Master branch. A new version is released to user. Merge is only possible if other developers leave a review and approve the changes.

# How to access the list of outstanding bugs and the list of resolved bugs, and how to resolve a bug

If you find a bug, please let me know by using <https://goo.gl/forms/pw44cIG09RCFkyQU2>

OR you can create a new issue

for the client through <https://github.com/PickPic/PickPic_Client/issues>

for the server through <https://github.com/PickPic/PickPic_Server/issues>

If you want our bug history, then see issues

for client in <https://github.com/PickPic/PickPic_Client/issues>

for server in <https://github.com/PickPic/PickPic_Server/issues>

OR you can download the bug report document in "<https://github.com/PickPic/PickPic_Client/blob/develop/document/Bug_report.docx?raw=true>"