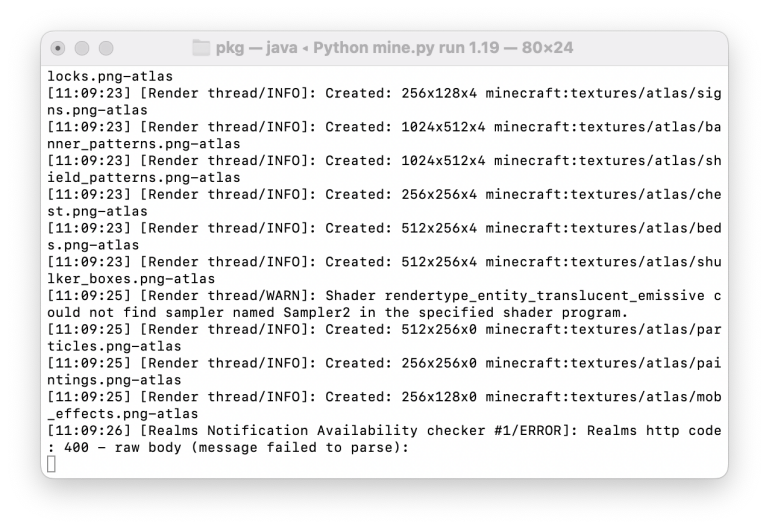


PythonMinecraftLauncher

开发教程

RATE TEAM



Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,

and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by

the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all

other entities that control, are controlled by, or are under common

control with that entity. For the purposes of this definition,

"control" means (i) the power, direct or indirect, to cause the

direction or management of such entity, whether by contract or

otherwise, or (ii) ownership of fifty percent (50%) or more of the

outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity

exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,

including but not limited to software source code, documentation

source, and configuration files.

"Object" form shall mean any form resulting from mechanical

transformation or translation of a Source form, including but

not limited to compiled object code, generated documentation,

and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or

Object form, made available under the License, as indicated by a

copyright notice that is included in or attached to the work

(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object

form, that is based on (or derived from) the Work and for which the

editorial revisions, annotations, elaborations, or other modifications

represent, as a whole, an original work of authorship. For the purposes

of this License, Derivative Works shall not include works that remain

separable from, or merely link (or bind by name) to the interfaces of,

the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including

the original version of the Work and any modifications or additions

to that Work or Derivative Works thereof, that is intentionally

submitted to Licensor for inclusion in the Work by the copyright owner

or by an individual or Legal Entity authorized to submit on behalf of

the copyright owner. For the purposes of this definition, "submitted"

means any form of electronic, verbal, or written communication sent

to the Licensor or its representatives, including but not limited to

communication on electronic mailing lists, source code control systems,

and issue tracking systems that are managed by, or on behalf of, the

Licensor for the purpose of discussing and improving the Work, but

excluding communication that is conspicuously marked or otherwise

designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity

on behalf of whom a Contribution has been received by Licensor and

subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of

this License, each Contributor hereby grants to You a perpetual,

worldwide, non-exclusive, no-charge, royalty-free, irrevocable

copyright license to reproduce, prepare Derivative Works of,

publicly display, publicly perform, sublicense, and distribute the

Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of

this License, each Contributor hereby grants to You a perpetual,

worldwide, non-exclusive, no-charge, royalty-free, irrevocable

(except as stated in this section) patent license to make, have made,

use, offer to sell, sell, import, and otherwise transfer the Work,

where such license applies only to those patent claims licensable

by such Contributor that are necessarily infringed by their

Contribution(s) alone or by combination of their Contribution(s)

with the Work to which such Contribution(s) was submitted. If You

institute patent litigation against any entity (including a

cross-claim or counterclaim in a lawsuit) alleging that the Work

or a Contribution incorporated within the Work constitutes direct

or contributory patent infringement, then any patent licenses

granted to You under this License for that Work shall terminate

as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the

Work or Derivative Works thereof in any medium, with or without

modifications, and in Source or Object form, provided that You

meet the following conditions:

(a) You must give any other recipients of the Work or

Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices

stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works

that You distribute, all copyright, patent, trademark, and

attribution notices from the Source form of the Work,

excluding those notices that do not pertain to any part of

the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its

distribution, then any Derivative Works that You distribute must

include a readable copy of the attribution notices contained

within such NOTICE file, excluding those notices that do not

pertain to any part of the Derivative Works, in at least one

of the following places: within a NOTICE text file distributed

as part of the Derivative Works; within the Source form or

documentation, if provided along with the Derivative Works; or,

within a display generated by the Derivative Works, if and

wherever such third-party notices normally appear. The contents

of the NOTICE file are for informational purposes only and

do not modify the License. You may add Your own attribution

notices within Derivative Works that You distribute, alongside

or as an addendum to the NOTICE text from the Work, provided

that such additional attribution notices cannot be construed

as modifying the License.

You may add Your own copyright statement to Your modifications and

may provide additional or different license terms and conditions

for use, reproduction, or distribution of Your modifications, or

for any such Derivative Works as a whole, provided Your use,

reproduction, and distribution of the Work otherwise complies with

the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,

any Contribution intentionally submitted for inclusion in the Work

by You to the Licensor shall be under the terms and conditions of

this License, without any additional terms or conditions.

Notwithstanding the above, nothing herein shall supersede or modify

the terms of any separate license agreement you may have executed

with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade

names, trademarks, service marks, or product names of the Licensor,

except as required for reasonable and customary use in describing the

origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or

agreed to in writing, Licensor provides the Work (and each

Contributor provides its Contributions) on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or

implied, including, without limitation, any warranties or conditions

of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A

PARTICULAR PURPOSE. You are solely responsible for determining the

appropriateness of using or redistributing the Work and assume any

risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,

whether in tort (including negligence), contract, or otherwise,

unless required by applicable law (such as deliberate and grossly

negligent acts) or agreed to in writing, shall any Contributor be

liable to You for damages, including any direct, indirect, special,

incidental, or consequential damages of any character arising as a

result of this License or out of the use or inability to use the

Work (including but not limited to damages for loss of goodwill,

work stoppage, computer failure or malfunction, or any and all

other commercial damages or losses), even if such Contributor

has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing

the Work or Derivative Works thereof, You may choose to offer,

and charge a fee for, acceptance of support, warranty, indemnity,

or other liability obligations and/or rights consistent with this

License. However, in accepting such obligations, You may act only

on Your own behalf and on Your sole responsibility, not on behalf

of any other Contributor, and only if You agree to indemnify,

defend, and hold each Contributor harmless for any liability

incurred by, or claims asserted against, such Contributor by reason

of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following

boilerplate notice, with the fields enclosed by brackets "[]"

replaced with your own identifying information. (Don't include

the brackets!) The text should be enclosed in the appropriate

comment syntax for the file format. We also recommend that a

file or class name and description of purpose be included on the

same "printed page" as the copyright notice for easier

identification within third-party archives.

Copyright [RATE\_TEAM] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

目录

0 序章

01 CMD开发

01.1 CMD所需代码

02 GUI开发

02.1 GUI所需代码

附录1

附录2

0 序章

感谢您使用PML并积极参与到开发之中，在开始之前，请您认真阅读README.md文件用于了解PML

| **名称** | **安装方法** |
| --- | --- |
| python | 见下 |
| logging |  |
| sys |  |
| os |  |
| PySide6 | pip3 install PySide6 |
| docopt | pip3 install docopt |
| minecraft\_launcher\_lib | pip3 install minecraft-launcher-lib |
| subprocess |  |
| webbrowser |  |
| typing | pip3 install typing |
| IPython | pip3 install ipython |
| time |  |
| requests | pip3 install requests |

### **依赖**

见左表

### **安装Python**

如果您使用ubuntu系统

sudu apt-get python3

如果您使用centos系统

sudo yum -y install python3

如果您使用的是linux桌面发行版

https://www.python.org/downloads/

如果您使用macos系统

brew安装

brew install python3

pkg安装

https://www.python.org/downloads/

如果您使用的是Windows系统

https://www.python.org/downloads/

### 截屏2023-01-27 20.20.39**命令**

您可以通过 python3 mine.py help查看更多

1 CMD的开发

* 1. CMD所需代码

if arguments.get("<xxx>"):

pass

python3 mine.py <xxx> 设置用户进入功能的入口请在最后一个except:前

'''

Usage:

mine run <想要启动的我的世界版本号>

mine install <想要下载的我的世界版本号>

...

mine <xxx>

'''

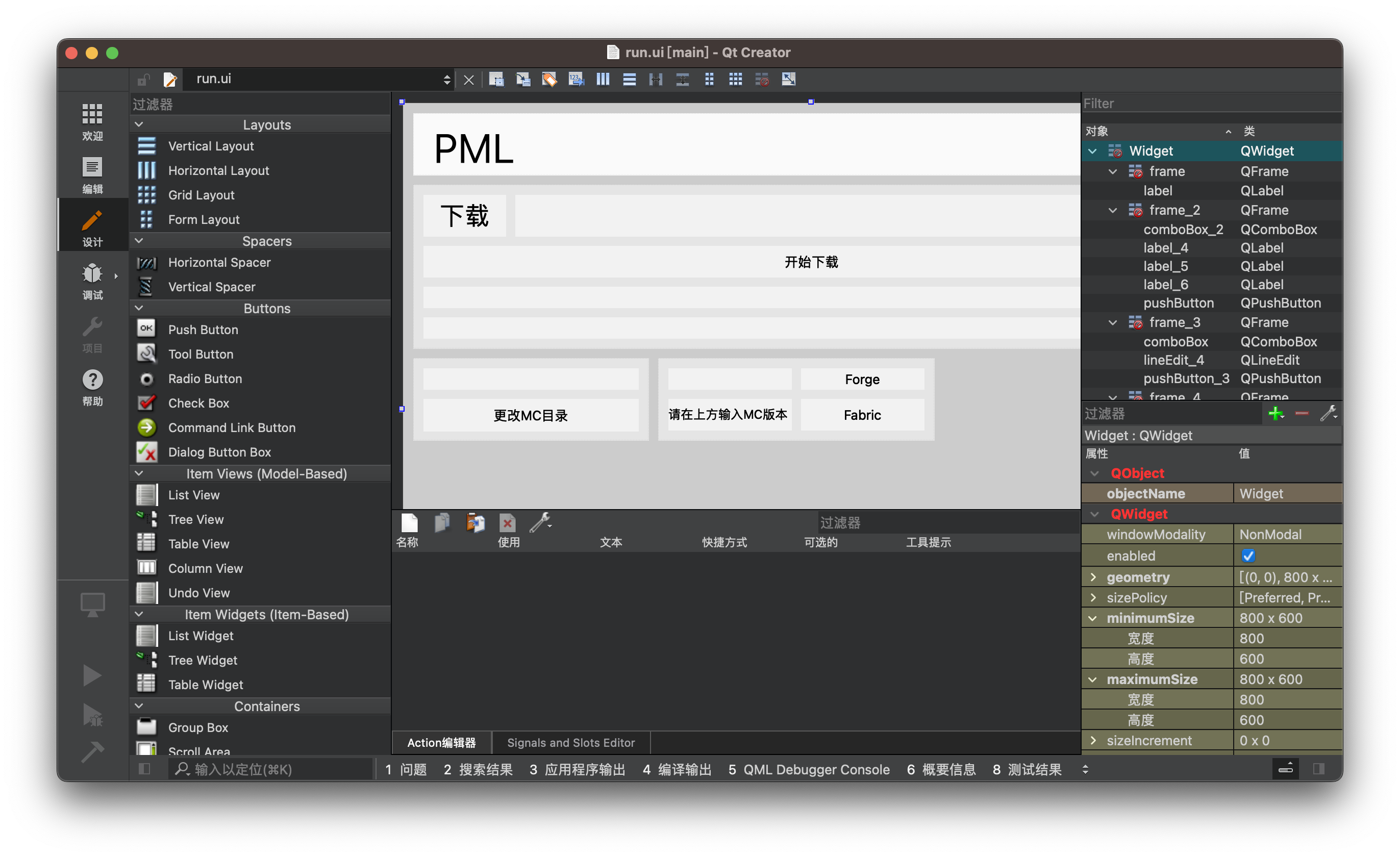
告诉程序您的入口（位于第1行，请在后方添加）

mincraft-launcher-lib程序

详见附录1

2.0 GUI的开发

2.1 GUI 所需代码



cd到QtGUI

使用Qt Designer(mac使用Qt Creator)打开run.ui进行编辑

并编辑mainwindow.py

self.ui.pushButton\_x.clicked.connect(self.xxx)

当pushButton\_x被点击时，运行xxx()

def xxx(self):

这是代码

设置xxx()为代码

其余详见附录二

附录1:Minecraft-Launcher-Lib文档

**minecraft-launcher-lib**

***Release 5.3***

**JakobDev**

**Jan 26, 2**

**CHAPTER**

**ONE**

**INSTALLATION**

Here are different methods to install minecraft-launcher-lib.

* **pip**

Like almost every other Python library, minecraft-launcher-lib can be installed directly from [PyPI](https://pypi.org/project/minecraft-launcher-lib/). That is the preferred way to install it for the most people.

pip install -U minecraft-launcher-lib

* **AUR**

If you use Arch Linux or a Arch based Distro like Manjaro, you may want to install it from the [Arch User](https://aur.archlinux.org/packages/python-minecraft-launcher-lib/) [Repository](https://aur.archlinux.org/packages/python-minecraft-launcher-lib/).

* **From Source**

You can also install minecraft-launcher-lib directly from source. That will give you the latest changes that are not in a release yet. Please do this only if you have a good reason.

pip install -U git+https://codeberg.org/JakobDev/minecraft-launcher-lib.git

**4** **Chapter 1. Installation**

**CHAPTER**

**TWO**

**TUTORIAL**

**2.1 Getting Started**

This first chapter of the documentation shows how to install and run Minecraft. The login with Microsoft is skipped here.

**2.1.1 Minecraft Directory**

To get started with minecraft-launcher-lib, you need a Minecraft Directory first. You can use

a new directory or the default Directory of Minecraft. You can get the default Directory with

minecraft\_launcher\_lib.utils.get\_minecraft\_directory().

# Get the Minecraft Directory of your System

minecraft\_directory = minecraft\_launcher\_lib.utils.get\_minecraft\_directory()

# Or use your own

minecraft\_directory = "path/to/your/minecraft/directory"

**2.1.2 Install Minecraft**

Before you can launch launch Minecraft, you need to install it. This can be done by using install\_minecraft\_version.

Let’s say we want to install Version 1.17 in our Minecraft Directory.

minecraft\_launcher\_lib.install.install\_minecraft\_version("1.17", minecraft\_directory)

To get the information how to install Minecraft, minecraft-launcher-lib looks first for a JSON file in your Minecraft Directory. In the case of 1.17 it’s minecraft\_directory/versions/1.17/1.17.json. This allows installing moded versions that are not official from Mojang. If the JSON file not does exists minecraft-launcher-lib tries to download it from the Mojang Servers. install\_minecraft\_version ensures that the Minecraft installation is correct, so you need to call it every time before you launch Minecraft, even if you had the version already installed.

**5**

**minecraft-launcher-lib, Release 5.3**

**2.1.3 Get Minecraft Versions**

If you don’t want to start a single version like 1.17 every time you need a list of all Minecraft version. To get that list use minecraft\_launcher\_lib.utils.get\_available\_versions(minecraft\_directory). It returns this list:

[

{

"id": "some\_id",

"type": "release"

},

{

"id": "some\_other\_id",

"type": "snapshot"

}

]

The id is the Minecraft version that can be used as argument for install\_minecraft\_version and other versions. The type says what type the version is. Possible values are: release, snapshot, beta, alpha.

To get the latest version, use get\_latest\_version().

latest\_release = minecraft\_launcher\_lib.utils.get\_latest\_version()["release"] latest\_snapshot = minecraft\_launcher\_lib.utils.get\_latest\_version()["snapshot"]

**2.1.4 Launch Minecraft**

Since you know how to install Minecraft, it’s now time to start it. First we need a dict with all options. The minimal options dict is this:

{

"username": "The Username",

"uuid": "The UUID",

"token": "The acces token"

}

The Username and UUID belongs to a Account. Since Name and UUID are public, the Token is used to log in. The token is generated every time when a User logs in with his Microsoft Account. Minecraft can be launched with a not existing user and a wrong token. This can be used for test cases. minecraft-launcher-lib allows creating a dict with a test user.

options = minecraft\_launcher\_lib.utils.generate\_test\_options()

We use the test options here to keep it simple. The login with Microsoft comes latter. Keep in mind that publish-ing a Launcher which allows User who haven’t bought Minecraft to play is illegal, so use this only for test cases in development. You can add more options to the dict like the resolution, but this is not needed to launch.

Now we have the options, we need to get the Minecraft command. In this case for Version 1.17.

minecraft\_command = minecraft\_launcher\_lib.command.get\_minecraft\_command("1.17",␣ ˓→minecraft\_directory, options)

The command that your get is a list of strings that can be used to run Minecraft e.g. with the subprocess module.

**6** **Chapter 2. Tutorial**

**minecraft-launcher-lib, Release 5.3**

**2.2 Microsoft Login**

Login with a Microsoft Account requires a Web browser and a Azure Application.

**2.2.1 Create Azure Application**

To login with Microsoft you need to create a Azure Application first. Follow [this tutorial](https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-register-app) to create one. You need the Clinet ID, the Secret and the redirect URL of your new Application.

**2.2.2 Let the User log in**

The login happens in a Web browser. This can be the normal Browser of the System or a Browser Widget embed in your Program. To get the url that is used for the login use minecraft\_launcher\_lib.microsoft\_account.get\_login\_url(client\_id: str, redirect\_uri: str). Open the URL and test if you can login. After you’ve logged in you will be redirected to <https:/>/<your redirect URL>?code=codegoeshere&state=<optional. codegoeshere is the code that you need. You can use minecraft\_launcher\_lib.microsoft\_account.get\_auth\_code\_from\_url(url: str) to get the code from the url. You can also use minecraft\_launcher\_lib.microsoft\_account.url\_contains\_auth\_code(url: str) to check if the given URL has a code.

**2.2.3 Secure option**

The minecraft\_launcher\_lib.microsoft\_account.get\_secure\_login\_data(client\_id: str, redirect\_uri: str, state: str = \_gen-erate\_state()) generates the login data for a secure login with pkce and state to prevent Cross-Site Request Forgery attacks and authorization code injection attacks. This is the recommended way to login. You can parse the auth code and verify the state with minecraft\_launcher\_lib.microsoft\_account.parse\_auth\_code\_url(url: str, state: str)

**2.2.4 Do the Login**

Use minecraft\_launcher\_lib.microsoft\_account.complete\_login(client\_id: str, redirect\_uri: str, auth\_code: str, code\_verifier: Optional[str]) to login to Minecraft. The auth code is the code from URL you’ve got in the previous step. The code verifier is the code verifier you’ve got if you used the secure login method. You get this result:

{

"id" : "The uuid",

"name" : "The username",

"access\_token": "The acces token",

"refresh\_token": "The refresh token",

"skins" : [{

"id" : "6a6e65e5-76dd-4c3c-a625-162924514568",

"state" : "ACTIVE",

"url" : "http://textures.minecraft.net/texture/

˓→1a4af718455d4aab528e7a61f86fa25e6a369d1768dcb13f7df319a713eb810b",

"variant" : "CLASSIC",

"alias" : "STEVE"

} ],

"capes" : []

}

As you can see it contains everything you need for the options dict of get\_minecraft\_command().

**2.2. Microsoft Login** **7**

**minecraft-launcher-lib, Release 5.3**

**2.2.5 Refresh**

To refresh just use minecraft\_launcher\_lib.microsoft\_account.complete\_refresh(client\_id: str, refresh\_token: str). The refresh token is from the function above. If the refresh fails, it will throw a InvalidRefreshToken exception. In this case you need the user to login again.

**2.3 Get Installation Progress**

Installing a new Minecraft version can, depending on the internet connection, take some time. It would be nice to show the user the progress e.g. in a Progressbar.

To tell your program the current progress, minecraft-launcher-lib uses callbacks. Callbacks are just normal functions that you write and that are called by minecraft-launcher-lib. Here is a example:

import minecraft\_launcher\_lib

current\_max = 0

def set\_status(status: str):

print(status)

def set\_progress(progress: int):

if current\_max != 0:

print(f"{progress}/{current\_max}")

def set\_max(new\_max: int):

global current\_max

current\_max = new\_max

minecraft\_directory = minecraft\_launcher\_lib.utils.get\_minecraft\_directory()

callback = {

"setStatus": set\_status,

"setProgress": set\_progress,

"setMax": set\_max

}

minecraft\_launcher\_lib.install.install\_minecraft\_version("1.17", minecraft\_directory,␣ ˓→callback=callback)

As you can see callback is a dict with functions. The functions are defined by you. You can write in these functions whatever you want. In the example above it prints the current status to the commandline.

**8** **Chapter 2. Tutorial**

**minecraft-launcher-lib, Release 5.3**

**2.4 More Launch Options**

minecraft-launcher-lib offers various options for launching Minecraft. This page shows the most important ones. For a full list check out the documentation of the command module.

**2.4.1 JVM Arguemnts**

JVM Arguemnts are a list of strings. Each argument is a entry in the list. Here is a example:

# Right

options["jvmArguments"] = ["-Xmx2G", "-Xms2G"]

# Wrong

options["jvmArguments"] = ["-Xmx2G -Xms2G"]

# Wrong

options["jvmArguments"] = "-Xmx2G -Xms2G"

Make sure every argument starts with a -, otherwise Minecraft will not start with a “Could not find or load main class” error.

**2.4.2 Java Executable**

The Java Executable is the path the Java which is used to run Minecraft. If the version.json contains a Java Runtime, it minecraft-launcher-lib will download and use these version. Otherwise it will just use the java command. minecraft-launcher-lib allows to overwrite this. This can be useful, if you want to start a older version which needs a older Java and does not contain a runtime in the version.json.

There are 2 options to overwrite the Java Executable: executablePath and defaultExecutablePath. The difference is, that executablePath is always used. defaultExecutablePath is only used, when the version.json has set no Java Runtime. If the version.json contains a Runtime, the Runtime will be prefered over the defaultExecutablePath.

options["executablePath"] = "path/to/java"

options["defaultExecutablePath"] = "path/to/java"

**2.4.3 Custom Resolution**

minecraft-launcher-lib allows starting Minecraft with a custom resolution. The first thing you have to do is enable the custom resolution. After that you can set it:

* Enable custom resolution options["customResolution"] = True
* Set custom resolution

options["resolutionWidth"] = "600"

options["resolutionHeight"] = "500"

Make sure you use strings and not int for the resolution.

**2.4. More Launch Options** **9**

**minecraft-launcher-lib, Release 5.3**

**2.4.4 Game Directory**

The Game Directory is the directory where Minecraft saves all his stuff like Worlds, Resourcepacks, Options etc. By default your Minecraft Directory is used as Game Directory.

options["gameDirectory"] = "path/to/your/game/directory"

If the directory does not exists, Minecraft will create it.

**2.4.5 Use Demo Mode**

Minecraft has a build-in Demo mode, which is used, if somebody who does not bought Minecraft launches the Game through the official launcher. minecraft-launcher-lib allows you to enable the Demo mode. You need at least Minecraft version 1.3.1 to use the Demo mode.

options["demo"] = True

**2.5 Install Forge**

This tutorial shows how to install forge using minecraft-launcher-lib. Please note that the Forge Devs do not want automatic installations unless you donated to them.

**2.5.1 Get Forge Version**

Before you install Forge, you need to know what Forge version you need. Use find\_forge\_version():

forge\_version = minecraft\_launcher\_lib.forge.find\_forge\_version("1.17.1")

if forge\_version is None:

print("This Minecraft Version is not supported by Forge")

return

In this case we get the latest Forge version for 1.17.1.

**2.5.2 Install Forge Version**

Now we have the Forge version, so we can install it. Use install\_forge\_version().

minecraft\_launcher\_lib.forge.install\_forge\_version(forge\_version, minecraft\_ ˓→directory)

install\_forge\_version() supports the same callbacks as install\_minecraft\_version(). install\_forge\_version() does not support versions prior to 1.12. Use supports\_automatic\_install(9 to check if your Forge version is supported by this function.

**10** **Chapter 2. Tutorial**

**minecraft-launcher-lib, Release 5.3**

**2.5.3 Launch Forge Version**

Unfortunately, the version we got with find\_forge\_version() can’t be used for get\_minecraft\_command(), because forge installs it under a little different name. Use get\_installed\_versions() to get the right version id for the launch.

**2.6 Install Fabric**

This tutorial shows how to install forge using minecraft-launcher-lib.

**2.6.1 Install Fabric**

Installing Fabric is very easy. Let’s say you want to install Fabric for 1.17:

minecraft\_launcher\_lib.fabric.install\_fabric("1.17", minecraft\_directoy)

install\_fabric() supports the same callbacks as install\_minecraft\_version().

**2.6.2 Launch Fabric**

Use get\_installed\_versions() to get a list of all installed versions. Your new installed fabric version should be in the list.

You can launch it like any other Minecraft version.

**2.7 Custom Types**

You will see types like e.g MinecraftOptions or MinecraftVersionInfo. They are defined in *types* and *microsoft\_types*.

They are all normal Dicts. Let’s take a look at MinecraftVersionInfo:

class MinecraftVersionInfo(TypedDict):

id: str

type: str

releaseTime: datetime.datetime

It means the following: This function returns a Dict with these keys:

* id: A string
* type: A string
* releaseTime: A datetime.datetime

This type definition is just there to help your IDE. The function itself returns just a normal Dict.

version\_list = minecraft\_launcher\_lib.utils.get\_version\_list()

print(version\_list[0]["id"])

print(type(version\_list[0]))

# <class 'dict'>

As said above, it is there to help your IDE. When the function definition just say, that it returns a Dict, your IDE will not know what the Dict contains. But when using a TypedDict, your IDE, will exactly know what the Dict contains and can offer your better autocompletion and a better type checking.

**2.6. Install Fabric** **11**

**minecraft-launcher-lib, Release 5.3**

You can even use it when you are calling a function:

import minecraft\_launcher\_lib

options: minecraft\_launcher\_lib.types.MinecraftOptions = {} options["username"] = "Test123"

When using a IDE, you will see that it will start autocompleting the keys of the Dict while writing.

For more information about TypedDict see [PEP 589](https://peps.python.org/pep-0589/).

**2.8 Getting Help**

Here are some sources if you need more information.

* This documentation contains under modules a list of modules with every function you can use. You should check it out. You can test the functions in a interactive Python Shell.
* Check out the [Examples on Codeberg](https://codeberg.org/JakobDev/minecraft-launcher-lib/src/branch/master/examples).
* You can check out the source of the programs in the *Showcase*.
* If none of them above help, feel free to [open a Issue](https://codeberg.org/JakobDev/minecraft-launcher-lib/issues).

**12** **Chapter 2. Tutorial**

**CHAPTER**

**THREE**

**MODULES**

**3.1 command**

command contains the function for creating the minecraft command.

get\_minecraft\_command(version: str, minecraft\_directory: Union[str, os.PathLike],␣

˓→options: MinecraftOptions) -> List[str]

Returns the command for running minecraft as list. The given command can be executed with subprocess. utils contains a function to get the default minecraft directory.

options is a dict:

options = {

#This is needed

"username": The Username,

"uuid": uuid of the user,

"token": the accessToken,

#This is optional

"executablePath": "java", # The path to the java executable

"defaultExecutablePath": "java", # The path to the java executable if the version. ˓→json has none

"jvmArguments": [], #The jvmArguments

"launcherName": "minecraft-launcher-lib", # The name of your launcher

"launcherVersion": "1.0", # The version of your launcher

"gameDirectory": "/home/user/.minecraft", # The gameDirectory (default is the path␣ ˓→given in arguments)

"demo": False, # Run Minecraft in demo mode

"customResolution": False, # Enable custom resolution

"resolutionWidth": "854", # The resolution width

"resolutionHeight": "480", # The resolution heigth

"server": "example.com", # The ip of a server where Minecraft connect to after start

"port": "123", # The port of a server where Minecraft connect to after start

"nativesDirectory": "minecraft\_directory/versions/version/natives", # The natives␣ ˓→directory

"enableLoggingConfig": False, # Enable use of the log4j configuration file

"disableMultiplayer": False, # Disables the multiplayer

"disableChat": False # Disables the chat

}

You can use the account module to get the needed information. For more information about the options take a look at the *More Launch Options* tutorial.

**13**

**minecraft-launcher-lib, Release 5.3**

**3.2 install**

install allows you to install minecraft.

install\_minecraft\_version(versionid: str, minecraft\_directory: Union[str, os.PathLike],␣

˓→callback: Optional[CallbackDict] = None) -> None

Installs a minecraft version into the given path. e.g. install\_version(“1.14”,”/tmp/minecraft”). utils contains a function to get the default minecraft directory.

callback is a dict with functions that are called with arguments to get the progress. You can use it to show the progress to the user.

callback = {

"setStatus": some\_function, #This function is called to set a text "setProgress" some\_function, #This function is called to set the progress. "setMax": some\_function, #This function is called to set to max progress.

}

Files that are already exists will not be replaced.

**3.3 natives**

natives contains a function for extracting natives libraries to a specific folder.

extract\_natives(versionid: str,path: Union[str, os.PathLike],extract\_path: str) -> None

Extract all native libraries from a version into the given directory. The directory will be created, if it does not exist.

Note: The natives are all extracted while installing. So you don’t need to use this function in most cases.

**3.4 account**

**Warning:** Minecraft has moved to Microsoft Accounts, so this module is now deprecated

account contains functions for interacting with your mojang account.

login\_user(username: str, password: str) -> Dict[str,Any]

Login to your mojang account. The response contains things like accessToken, clientToken, uuid and something else.

Note: You should never save username and password! Only the client token.

validate\_access\_token(access\_token: str) -> bool

Returns true, if the accessToken is valid. Otherwise it will return false. You should check that before running minecraft.

refresh\_access\_token(access\_token: str, client\_token: str) -> Dict[str,Any]

Get a new accessToken.

**14** **Chapter 3. Modules**

**minecraft-launcher-lib, Release 5.3**

logout\_user(username: str, password: str) -> bool

Log a user out.

invalidate\_access\_token(access\_token: str, client\_token: str) -> Any

Make an accessToken invalid.

upload\_skin(uuid: str, access\_token: str, path: str, slim: bool=False) -> Any

Upload a Skin.

reset\_skin(uuid: str, access\_token: str) -> Any

Reset a Skin.

**3.5 microsoft\_account**

microsoft\_account contains functions for login with a Microsoft Account. Before using this module you need to [create](https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-register-app) [a Azure application](https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-register-app). Many thanks to wiki.vg for it’s [documentation of the login process](https://wiki.vg/Microsoft_Authentication_Scheme). For a list of all types see

*microsoft\_types*.

get\_login\_url(client\_id: str, redirect\_uri: str) -> str

Returns the url to the website on which the user logs in.

For a more secure alternative, use get\_secure\_login\_data()

generate\_state() -> str

Generates a random state

get\_secure\_login\_data(client\_id: str, redirect\_uri: str, state: Optional[str] = None) ->␣ ˓→Tuple[str, str, str]:

Generates the login data for a secure login with pkce and state.

Prevents Cross-Site Request Forgery attacks and authorization code injection attacks.

Returns the url to the website on which the user logs in, the state and the code verifier.

url\_contains\_auth\_code(url: str) -> bool

Checks if the given url contains a authorization code.

get\_auth\_code\_from\_url(url: str) -> Optional[str]

Get the authorization code from the url.

**3.5. microsoft\_account** **15**

**minecraft-launcher-lib, Release 5.3**

If you want to check the state, use parse\_auth\_code\_url(), which throws errors instead of returning an optional value.

Returns the auth code or None if the the code is nonexistent.

parse\_auth\_code\_url(url: str, state: Optional[str]) -> str:

Parse the authorization code url and checks the state if supplied.

Returns the auth code

complete\_login(client\_id: str, client\_secret: Optional[str], redirect\_uri: str, auth\_

˓→code: str, code\_verifier: Optional[str] = None) -> CompleteLoginResponse:

Do the complete login process. It returns the following:

{

"id" : "The uuid",

"name" : "The username",

"access\_token": "The acces token",

"refresh\_token": "The refresh token",

"skins" : [{

"id" : "6a6e65e5-76dd-4c3c-a625-162924514568",

"state" : "ACTIVE",

"url" : "http://textures.minecraft.net/texture/

˓→1a4af718455d4aab528e7a61f86fa25e6a369d1768dcb13f7df319a713eb810b",

"variant" : "CLASSIC",

"alias" : "STEVE"

} ],

"capes" : []

}

complete\_refresh(client\_id: str, client\_secret: Optional[str], redirect\_uri:␣

˓→Optional[str], refresh\_token: str) -> CompleteLoginResponse:

Do the complete login process with a refresh token. It returns the same as complete\_login().

get\_authorization\_token(client\_id: str, client\_secret: Optional[str], redirect\_uri: str,␣

˓→auth\_code: str, code\_verifier: Optional[str]) -> AuthorizationTokenResponse:

Get the authorization token.

refresh\_authorization\_token(client\_id: str, client\_secret: Optional[str], redirect\_uri:␣

˓→Optional[str], refresh\_token: str) -> AuthorizationTokenResponse:

Refresh the authorization token.

authenticate\_with\_xbl(access\_token: str) -> XBLResponse

Authenticate with Xbox Live.

authenticate\_with\_xsts(xbl\_token: str) -> XSTSResponse

**16** **Chapter 3. Modules**

**minecraft-launcher-lib, Release 5.3**

Authenticate with XSTS.

authenticate\_with\_minecraft(userhash: str, xsts\_token: str) ->␣ ˓→MinecraftAuthenticateResponse

Authenticate with Minecraft.

get\_store\_information(access\_token: str) -> MinecraftProfileResponse

Get the store information.

get\_profile(access\_token: str) -> MinecraftProfileResponse

Get the profile.

**3.6 utils**

utils just contains a few functions for helping you.

get\_minecraft\_directory() -> str

Returns the path to the standard minecraft directory.

get\_latest\_version() -> LatestMinecraftVersions

Returns the latest versions of snapshot and release.

get\_version\_list() -> List[MinecraftVersionInfo]

Returns a list of all versions with the type and release time.

get\_installed\_versions(minecraft\_directory: Union[str, os.PathLike]) ->␣ ˓→List[MinecraftVersionInfo]

Returns a list with all installed versions in the given path.

get\_available\_versions(minecraft\_directory: Union[str, os.PathLike]) ->␣ ˓→List[MinecraftVersionInfo]

Returns a list with all installable and only local installed (e.g. Forge) versions.

get\_java\_executable() -> str

Return the path to the java executable. This may not work correctly on all systems.

get\_library\_version() -> str

Return the version of the library.

generate\_test\_options() -> MinecraftOptions

Generates test options for get\_minecraft\_command(). Use this function to test launching without logging in. This should not be used in production.

**3.6. utils** **17**

**minecraft-launcher-lib, Release 5.3**

is\_version\_valid(version: str,minecraft\_directory: Union[str, os.PathLike]) -> bool

Checks if the given version exists

get\_minecraft\_news(page\_size: int = 20) -> Articles

Get the news from minecraft.net

is\_vanilla\_version(version: str) -> bool

Checks if the given version is a vanilla version

is\_platform\_supported() -> bool

Checks if the current platform is supported

**3.7 forge**

**Note:** Before using this module, please read this comment from the forge developers:

Please do not automate the download and installation of Forge.

Our efforts are supported by ads from the download page.

If you MUST automate this, please consider supporting the project through https://www. ˓→patreon.com/LexManos/

It’s your choice, if you want to respect that and support forge.

forge allows you to install forge.

install\_forge\_version(versionid: str, path: str, callback: Optional[CallbackDict] = None,

˓→ java: str = None) -> None

Installs the given forge version in the given path. versionid must be one of the ids you get with list\_forge\_versions().

callback is the same dict as in the install module. java is a Java runtime to execute with.

This function does not work for minecraft versions older than 1.13.

run\_forge\_installer(version: str, java: str = None) -> None

Download and execute the forge installer of the given forge version. java is a Java runtime to execute with.

list\_forge\_versions() -> List[str]

Returns a list with all forge versions.

find\_forge\_version(vanilla\_version: str) -> Optional[str]

Returns the newest forge version for the given vanilla version. Returns None, if the given vanilla version has no forge version.

is\_forge\_version\_valid(forge\_version: str) -> bool

Checks if a forge version is valid.

**18** **Chapter 3. Modules**

**minecraft-launcher-lib, Release 5.3**

supports\_automatic\_install(forge\_version: str) -> bool

Checks if install\_forge\_version() supports the given forge version.

forge\_to\_installed\_version(forge\_version: str) -> str

Returns the Version under which Forge will be installed from the given Forge version. Raises a ValueError if the Version is invalid.

**3.8 fabric**

fabric contains functions for dealing with the fabric modloader

install\_fabric(minecraft\_version: str, minecraft\_directory: Union[str, os.PathLike],␣

˓→loader\_version: str = None, callback: Optional[CallbackDict] = None, java: str = None)␣ ˓→-> None

Installs the fabric modloader. path is the path to your Minecraft directory. minecraft\_version is a vanilla version that is supported by fabric. loader\_version is the loader version. If not given it will use the latest. callback is the same dict as in the install module. java is a path to a java runtime to execute with.

get\_all\_minecraft\_versions() -> List[FabricMinecraftVersion]

Returns all available Minecraft Versions for fabric

get\_stable\_minecraft\_versions() -> List[str]

Returns a list which only contains the stable Minecraft versions that supports fabric

get\_latest\_minecraft\_version() -> str

Returns the latest unstable Minecraft versions that supports fabric. This could be a snapshot.

get\_latest\_stable\_minecraft\_version() -> str

Returns the latest stable Minecraft version that supports fabric

is\_minecraft\_version\_supported(version: str) -> bool

Checks if a Minecraft version supported by fabric

get\_all\_loader\_versions() -> List[FabricLoader]

Returns all loader versions

get\_latest\_loader\_version() -> str

Get the latest loader version

get\_latest\_installer\_version() -> str

Returns the latest installer version

**3.8. fabric** **19**

**minecraft-launcher-lib, Release 5.3**

**3.9 runtime**

runtime allows to install the java runtime. This module is used by install\_minecraft\_version(), so you don’t need to use it in your code most of the time.

install\_jvm\_runtime(jvm\_version: str, minecraft\_directory: Union[str, os.PathLike],␣

˓→callback: Optional[CallbackDict] = None) -> None

Installs the given jvm runtime. callback is the same dict as in the install module.

get\_jvm\_runtimes() -> List[str]

Returns a list of all available runtimes.

get\_installed\_jvm\_runtimes(minecraft\_directory: Union[str, os.PathLike]) -> List[str]

Returns a list of all installed jvm runtimes.

get\_executable\_path(jvm\_version: str, minecraft\_directory: Union[str, os.PathLike]) ->␣ ˓→Optional[str]

Returns the path to the executable. Returns None if none is found.

**3.10 exceptions**

exceptions contains all custom exceptions that can be raised

**VersionNotFound**

Raised when a given version does not exists

**UnsupportedVersion**

Raised when install\_fabric() is called with a unsupported version.

**ExternalProgramError**

Raised when a external program fails to execute

**InvalidRefreshToken**

Raised when complete\_refresh() is called with a invalid refresh token

**3.11 types**

This module contains all Types for minecraft-launcher-lib. It may help your IDE. You don’t need to use this module directly in your code. If you are not interested in static typing just ignore it. For more information about TypeDict see [PEP 589](https://peps.python.org/pep-0589/).

class MinecraftOptions(TypedDict, total=False):

username: str

uuid: str

token: str

executablePath: str

defaultExecutablePath: str

jvmArguments: List[str]

(continues on next page)

**20** **Chapter 3. Modules**

**minecraft-launcher-lib, Release 5.3**

(continued from previous page)

launcherName: str

launcherVersion: str

gameDirectory: str

demo: bool

customResolution: bool

resolutionWidth: str

resolutionHeight: str

server: str

port: str

nativesDirectory: str

enableLoggingConfig: bool

disableMultiplayer: bool

disableChat: bool

class CallbackDict(TypedDict, total=False):

setStatus: Callable[[str], None]

setProgress: Callable[[int], None]

setMax: Callable[[int], None]

class LatestMinecraftVersions(TypedDict):

release: str

snapshot: str

class MinecraftVersionInfo(TypedDict):

id: str

type: str

releaseTime: datetime.datetime

complianceLevel: int

class FabricMinecraftVersion(TypedDict):

version: str

stable: bool

class FabricLoader(TypedDict):

separator: str

build: int

maven: str

version: str

stable: bool

* ----
* News
* ----

class \_ImageBase(TypedDict):

(continues on next page)

**3.11. types** **21**

**minecraft-launcher-lib, Release 5.3**

(continued from previous page)

content\_type: Literal["image", "outgoing-link", "video"]

imageURL: str

class Image(\_ImageBase, total=False):

alt: str

videoURL: str

videoType: str

videoProvider: str

videoId: str

linkurl: str

background\_color: Literal["bg-blue", "bg-green", "bg-red"]

class Tile(TypedDict):

sub\_header: str

image: Image

tile\_size: Literal["1x1", "1x2", "2x1", "2x2", "4x2"]

title: str

class \_ArticleBase(TypedDict):

default\_tile: Tile

articleLang: Literal["en-us"]

primary\_category: str

categories: List[str]

article\_url: str

publish\_date: str

tags: List[str]

class Article(\_ArticleBase, total=False):

preferred\_tile: Tile

class Articles(TypedDict):

article\_grid: List[Article]

article\_count: int

**3.12 microsoft\_types**

This module contains all Types for the *microsoft\_account* module. It has it’s own module because of the many types needed that are not used somewhere else.

class AuthorizationTokenResponse(TypedDict):

access\_token: str

token\_type: Literal["Bearer"]

expires\_in: int

scope: str

refresh\_token: str

(continues on next page)

**minecraft-launcher-lib, Release 5.3**

(continued from previous page)

class \_Xui(TypedDict):

uhs: str

class \_DisplayClaims(TypedDict):

xui: List[\_Xui]

class XBLResponse(TypedDict):

IssueInstant: str

NotAfter: str

Token: str

DisplayClaims: \_DisplayClaims

class XSTSResponse(TypedDict):

IssueInstant: str

NotAfter: str

Token: str

DisplayClaimns: \_DisplayClaims

class \_MinecraftStoreItem(TypedDict):

name: str

signature: str

class MinecraftStoreResponse(TypedDict):

items: List[\_MinecraftStoreItem]

signature: str

keyId: str

class MinecraftAuthenticateResponse(TypedDict):

username: str

roles: List[Any]

access\_token: str

token\_type: str

expires\_in: int

class \_MinecraftProfileInfo(TypedDict):

id: str

state: Literal["ACTIVE", "INACTIVE"]

url: str

class \_MinecraftProfileSkin(\_MinecraftProfileInfo):

variant: str

(continues on next page)

**minecraft-launcher-lib, Release 5.3**

(continued from previous page)

class \_MinecraftProfileCape(\_MinecraftProfileInfo):

alias: str

class MinecraftProfileResponse(TypedDict):

id: str

name: str

skins: List[\_MinecraftProfileSkin]

capes: List[\_MinecraftProfileCape]

class CompleteLoginResponse(MinecraftProfileResponse):

access\_token: str

refresh\_token: str

**24** **Chapter 3. Modules**

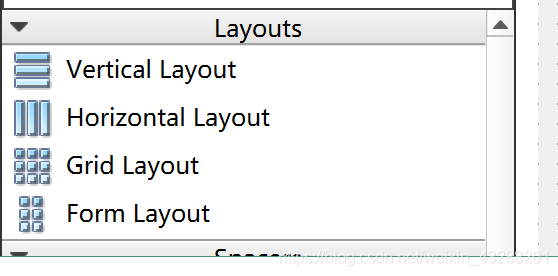
附录二 Qt常用组建和使用方法

# **1、容器的布局**

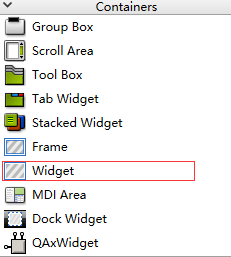
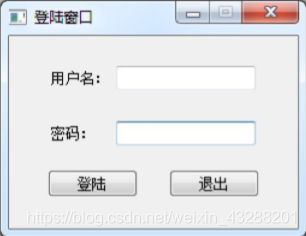
Qt 提供的布局中以下三种是我们最常用的：

* QHBoxLayout：按照水平方向从左到右布局；
* QVBoxLayout：按照竖直方向从上到下布局；
* QGridLayout：在一个网格中进行布局，类似于 HTML 的 table；

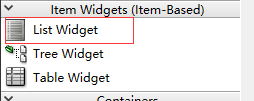
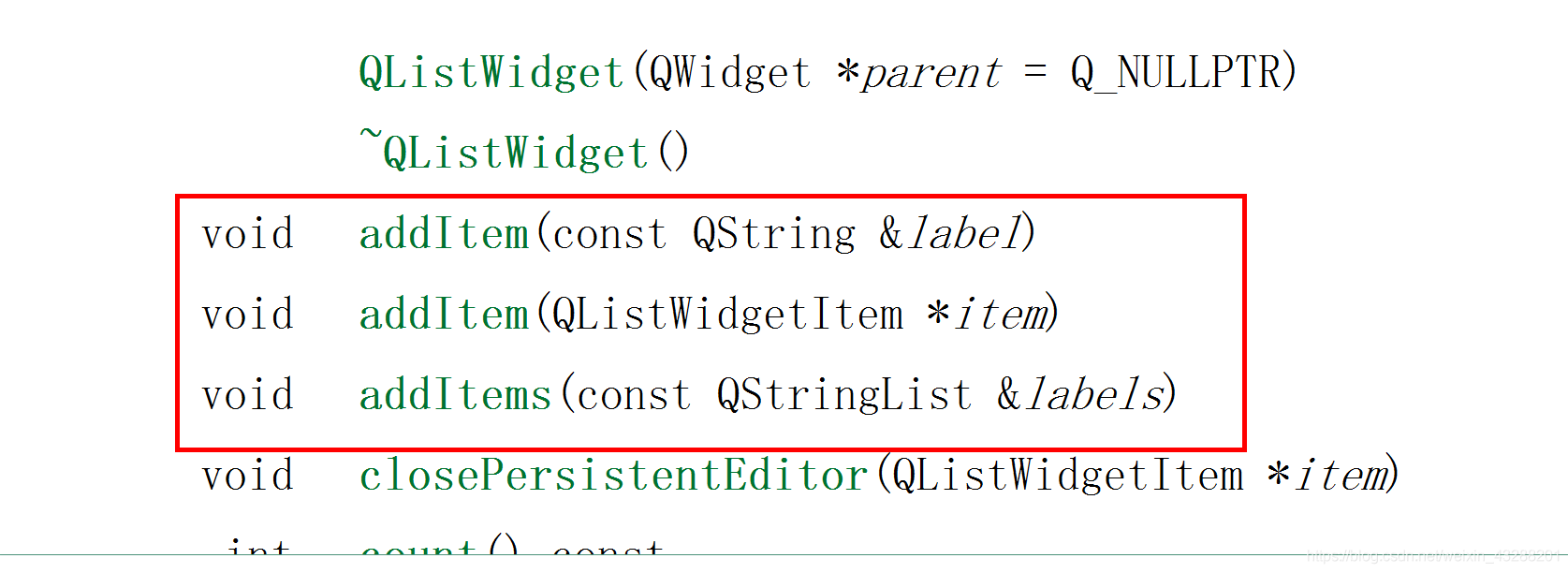
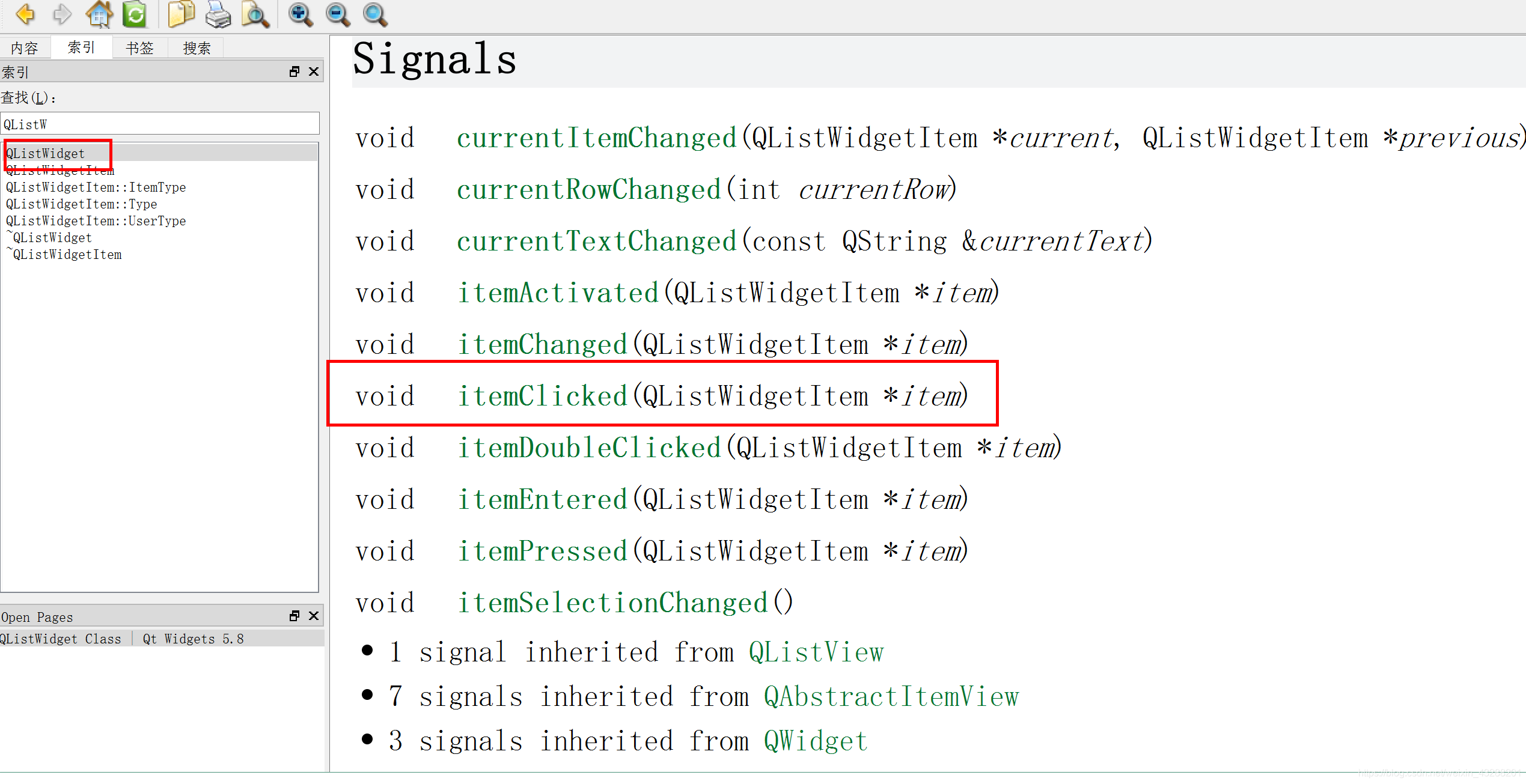
## **系统提供的布局控件**

  
这 4 个为系统给我们提供的布局的控件，但是使用起来不是非常的灵活，这里就不详细介绍 了。

## **利用 widget 做布局**

第二种布局方式是利用控件里的 widget 来做布局，在 Containers 中  
  
在 widget 中的控件可以进行水平、垂直、栅格布局等操作，比较灵活。 再布局的同时我们需要灵活运用弹簧的特性让我们的布局更加的美观，下面是一个登陆窗 口，利用 widget 可以搭建出如下登陆界面：  


# **2、QListWidget（列表控件）**

  
查找相关api  
  
  
构造函数中

*//列表控件*

*//往列表控件里面添加 item*

QListWidgetItem \*item = new QListWidgetItem("处处吻");*//单个添加*

ui->listWidget->addItem(item);

*//王列表控件中连续添加*

QStringList list;

list<<"下山"<<"把孤独当晚餐"<<"世间美好与你环环相扣";

ui->listWidget->addItems(list);

connect(ui->listWidget,&QListWidget::itemClicked,[=](QListWidgetItem \*item){

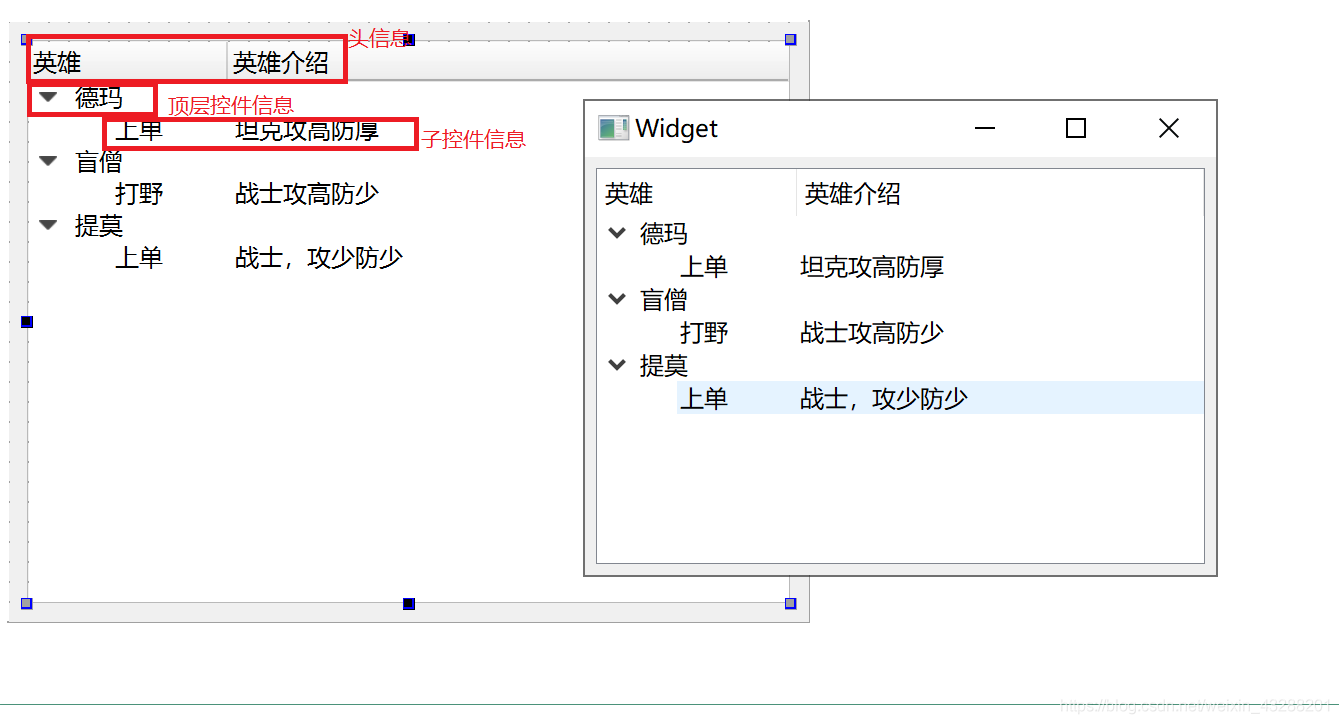
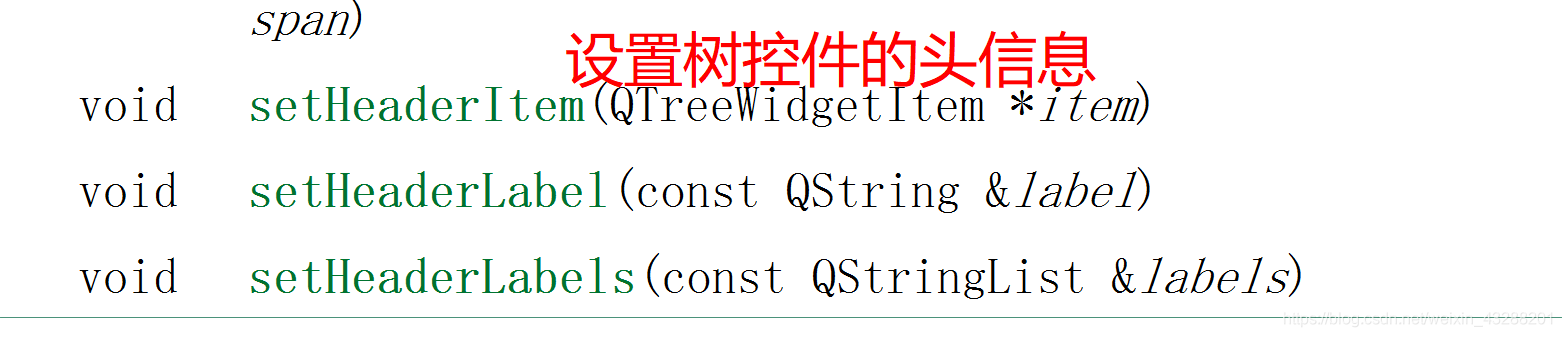
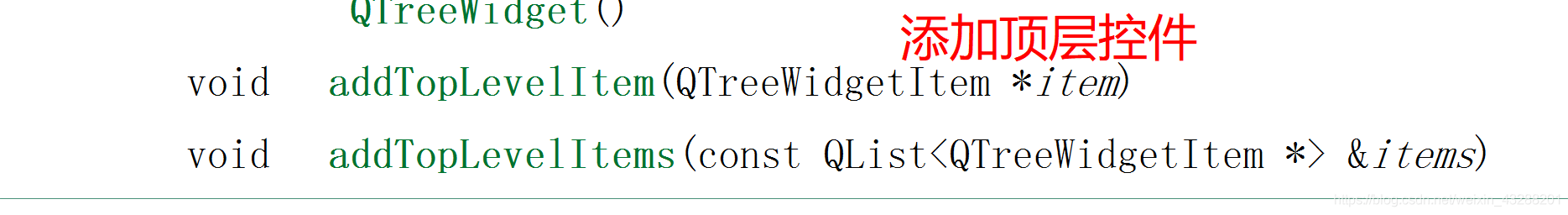
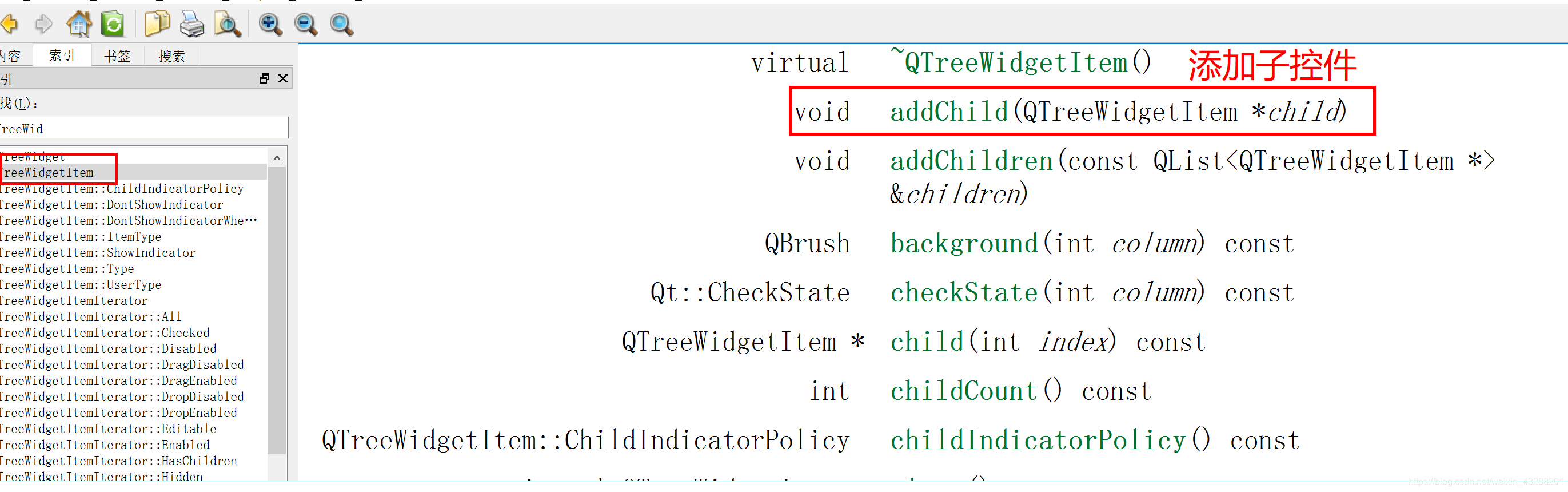
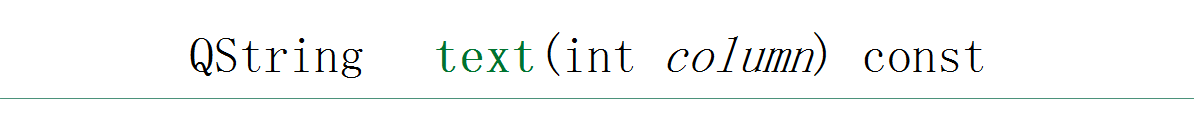
qDebug()<<item->text().toUtf8().data(); *//toUtf8().data() 的作用是去除双引号*

});

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14

运行结果：  


# **3、QTreeWidget（树控件）**

  
手动输入  
  
  
查找相关api  
  
  
  
  


*//设置树控件的头信息*

QStringList List;

List<<"英雄"<<"英雄介绍";

ui->treeWidget->setHeaderLabels(List);

*//添加顶层控件*

QTreeWidgetItem \*item1 = new QTreeWidgetItem(QStringList()<<"德玛西亚");

ui->treeWidget->addTopLevelItem(item1);

*//QTreeWidgetItem \*child = new QTreeWidgetItem(QStringList()<<"上单"<<"坦克，攻高防厚");*

*//item1->addChild(child);*

item1->addChild(new QTreeWidgetItem(QStringList()<<"上单"<<"坦克，攻高防厚"));

QTreeWidgetItem \*item2 = new QTreeWidgetItem(QStringList()<<"盲僧");

ui->treeWidget->addTopLevelItem(item2);

item2->addChild(new QTreeWidgetItem(QStringList()<<"打野"<<"战士，攻高防少"));

QTreeWidgetItem \*item3 = new QTreeWidgetItem(QStringList()<<"提莫");

ui->treeWidget->addTopLevelItem(item3);

item3->addChild(new QTreeWidgetItem(QStringList()<<"上单"<<"法师，攻高防少"));

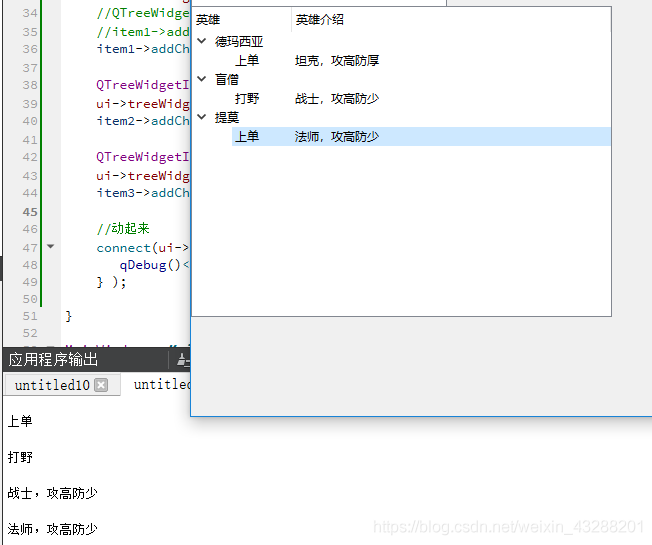
*//动起来*

connect(ui->treeWidget, &QTreeWidget::itemClicked,[](QTreeWidgetItem \*item,int column){

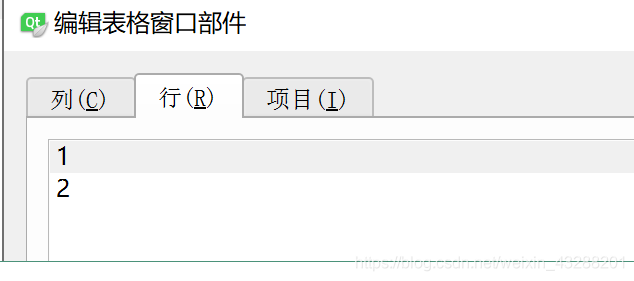
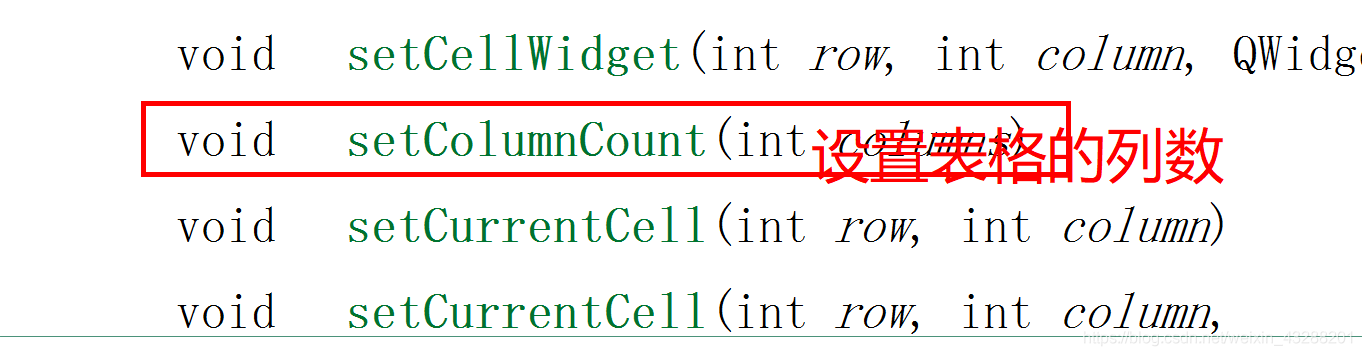
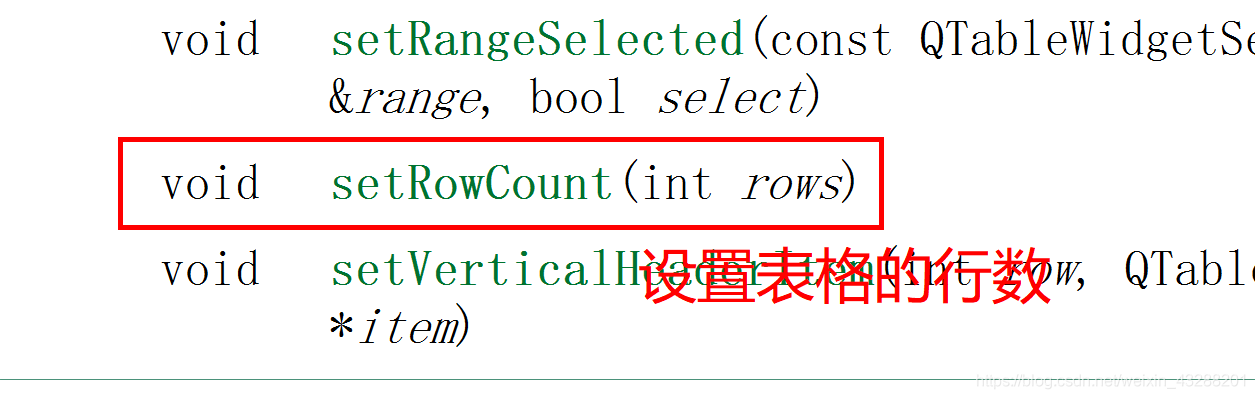
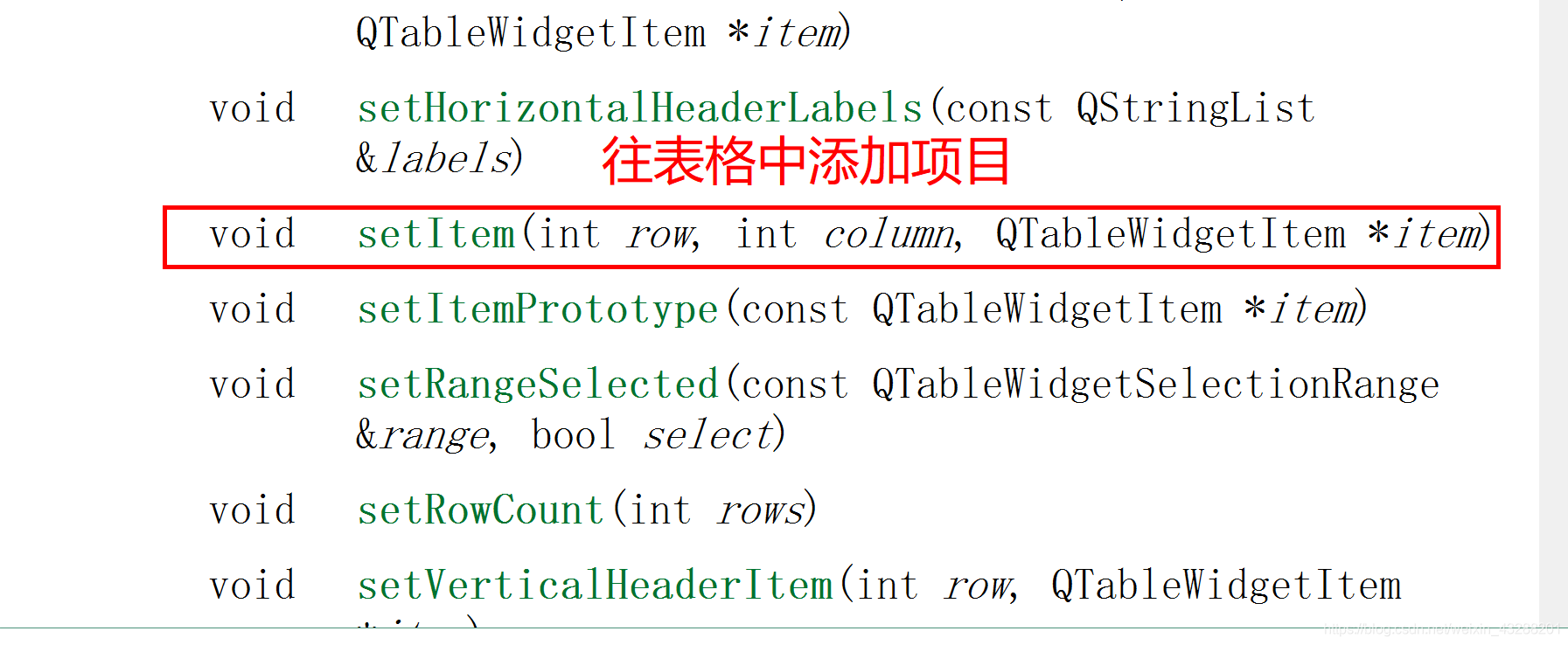
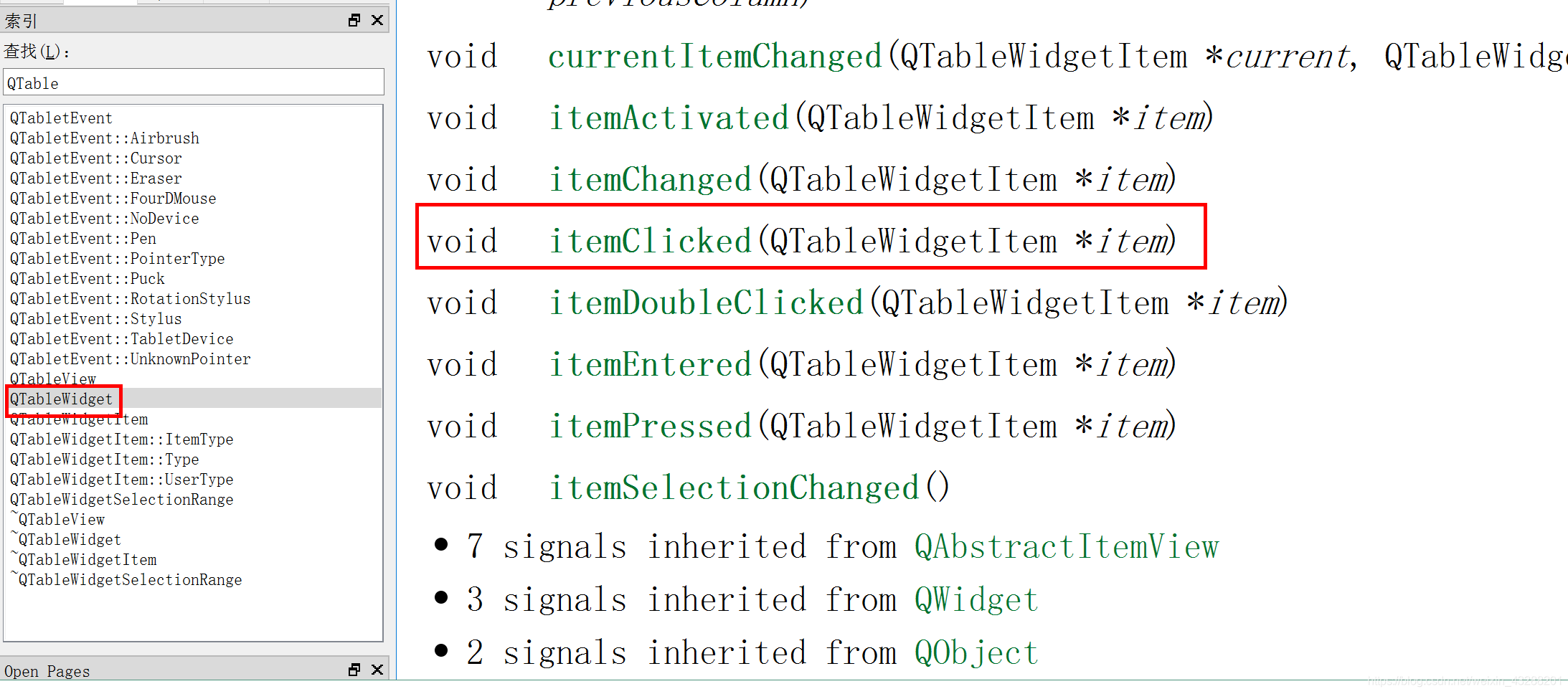
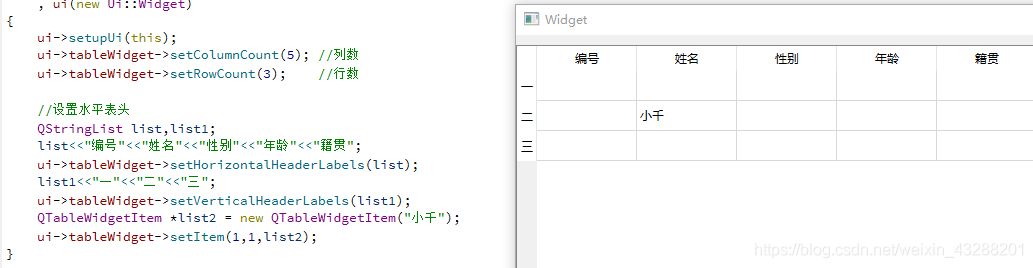
qDebug()<< item->text(column).toUtf8().data() <<endl;

} );

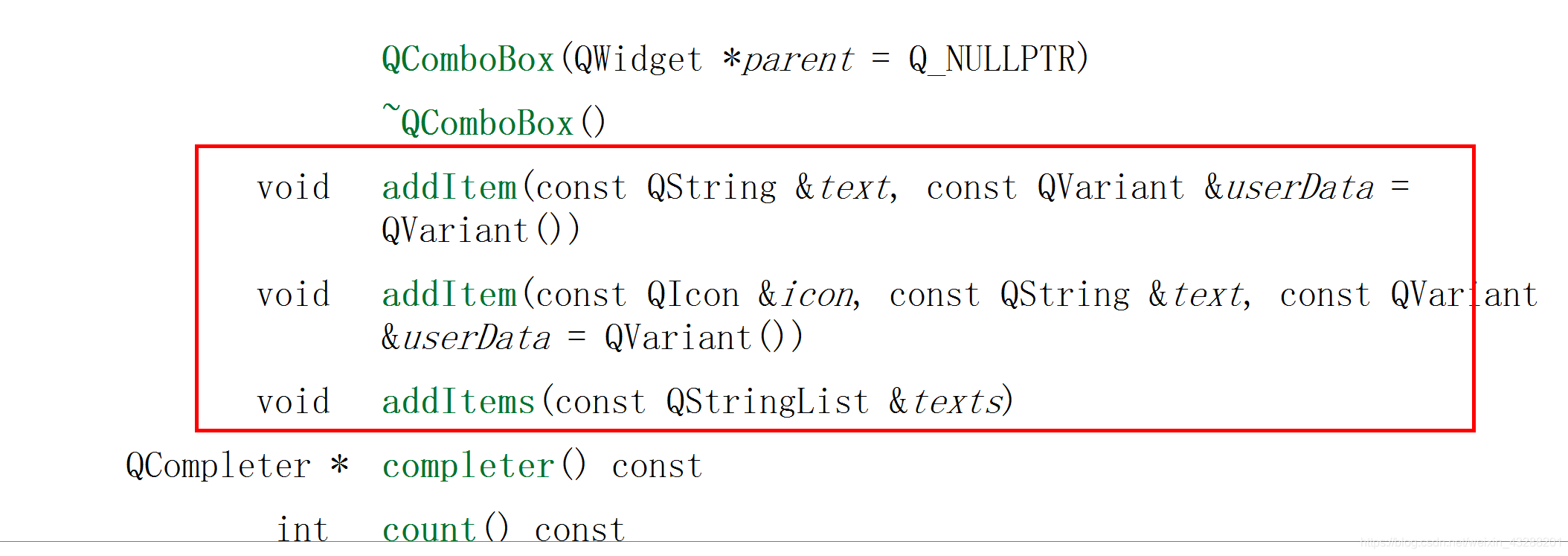
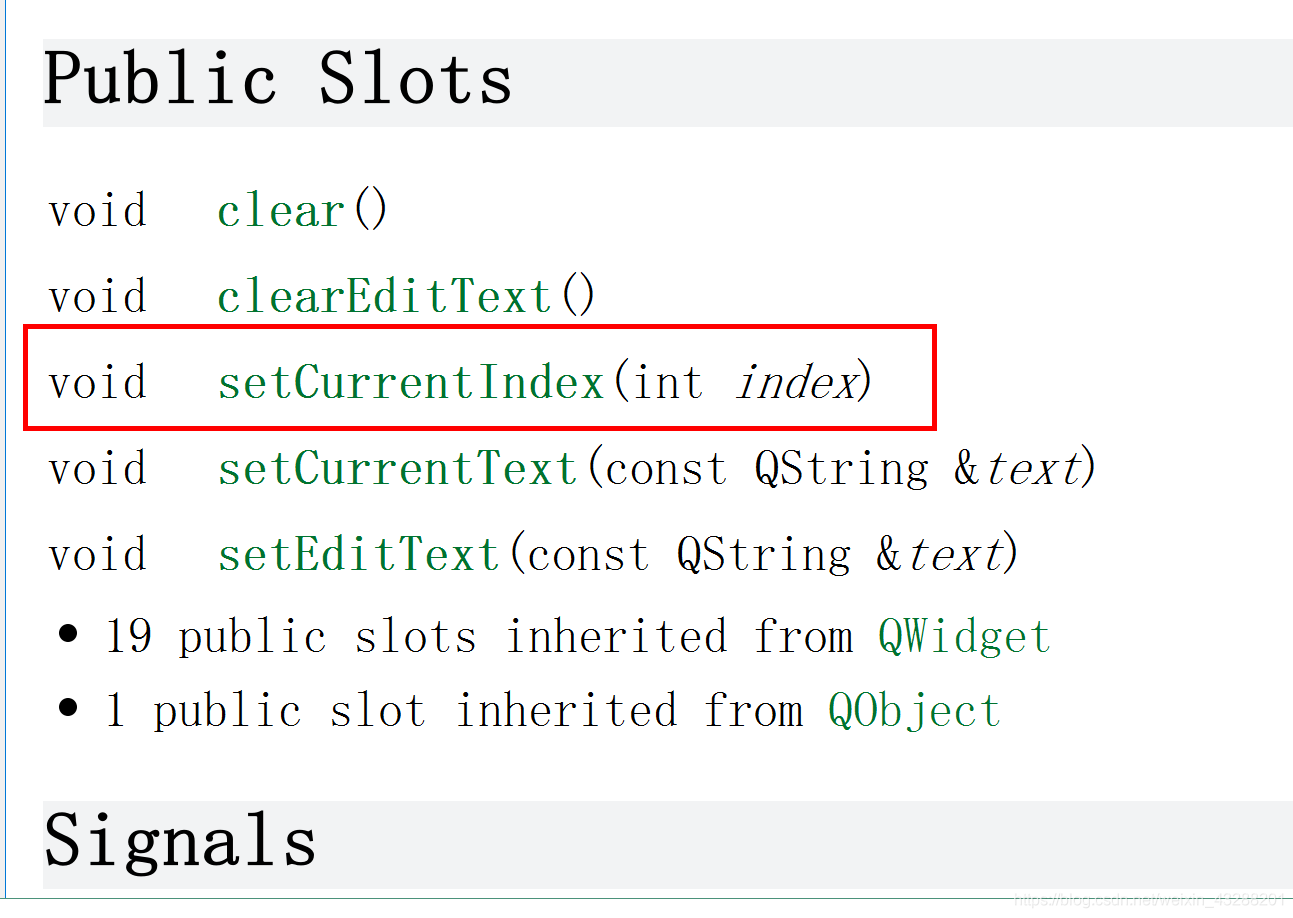
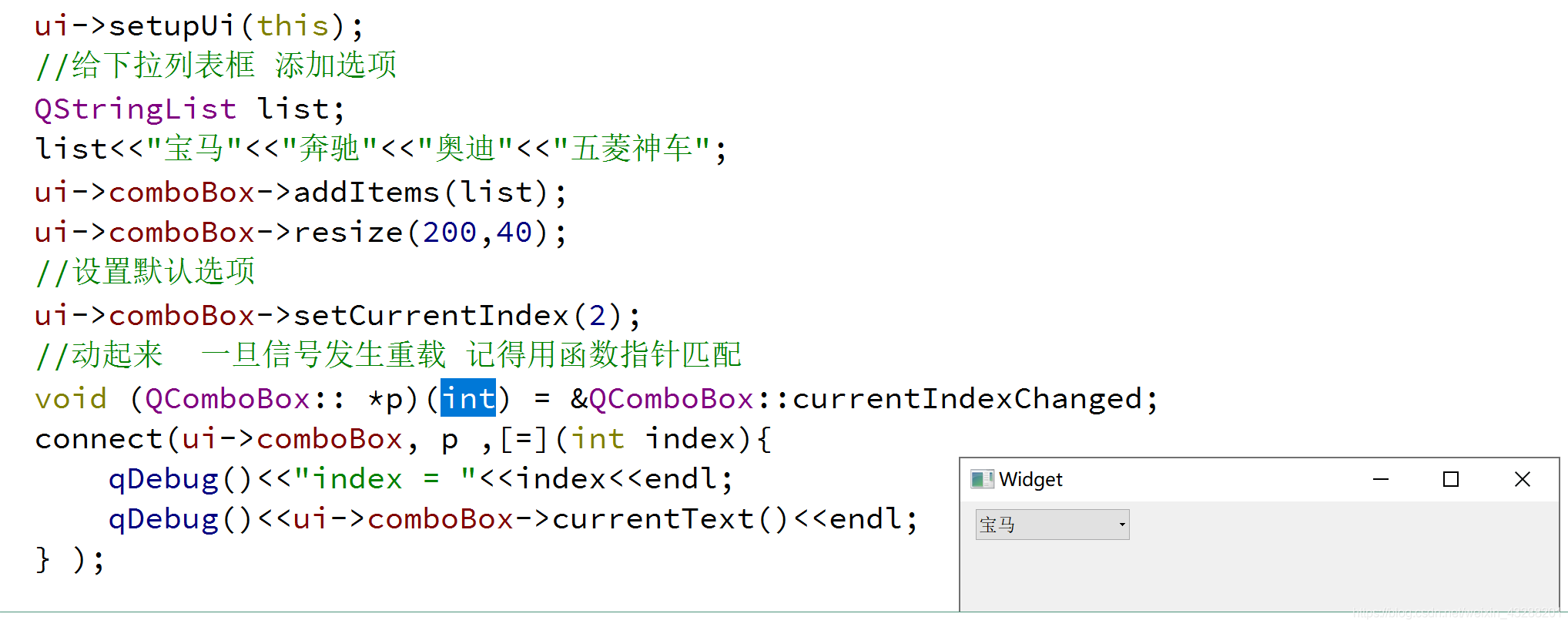
IMG_271

运行结果：  


# **4、QTableWidget（表格控件）**

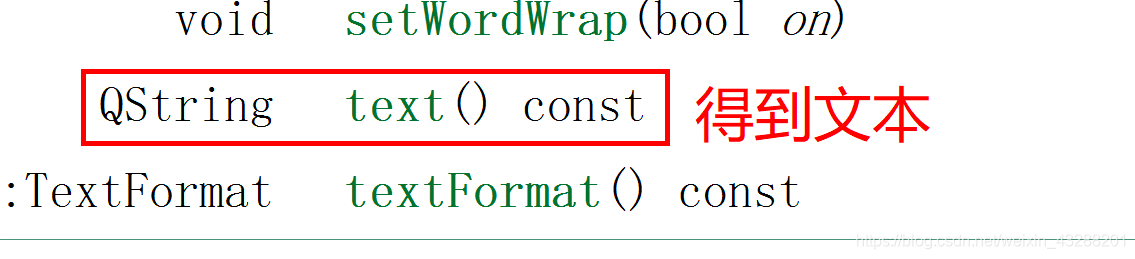
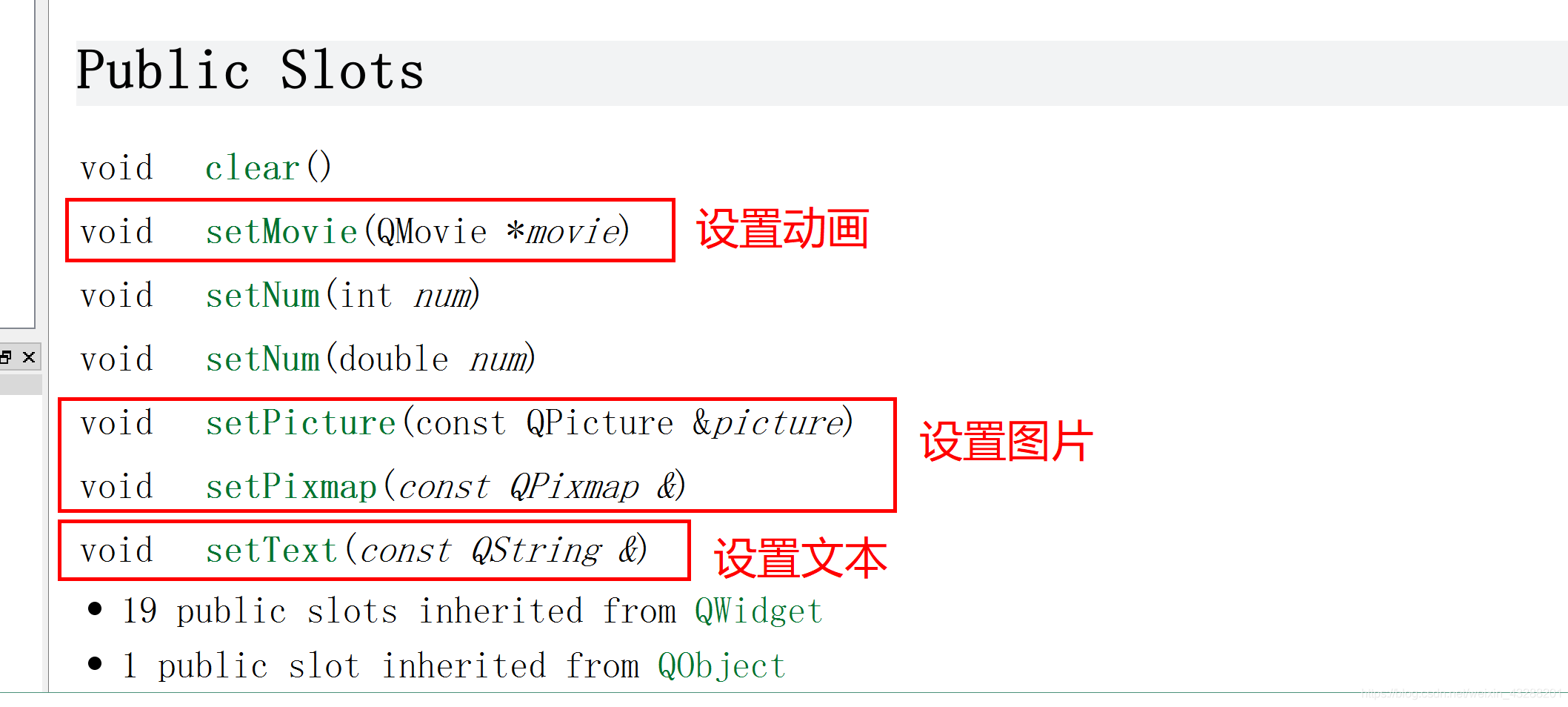
  
  
  
  
  
  
  
  


# **5、下拉列表框(Combo BOX)**

  
设置默认选项  
  
信号  
  


# **6、QLabel控件的使用**

## **6、1、Qlabel设置文本**

## **6、2、设置图片**

### **6.2.1、添加图片资源**

### **6.2.2、QPixmap 对象加载图片**