openGauss+Socket设计指导书

openGauss 环境安装与数据库服务启动

合肥工业大学 计算机与信息学院 张国富

孙丹

openGauss1.1.0 镜像文件：

链接：https://pan.baidu.com/s/1aKZZvnD7gZTrzxuEb1zZVg

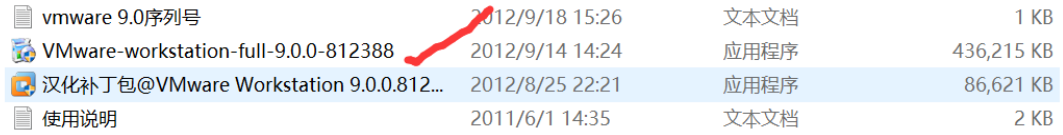
提取码：74nn

Vmware：

链接：https://pan.baidu.com/s/1-MooS2KiQQZlpA9cooAbyg

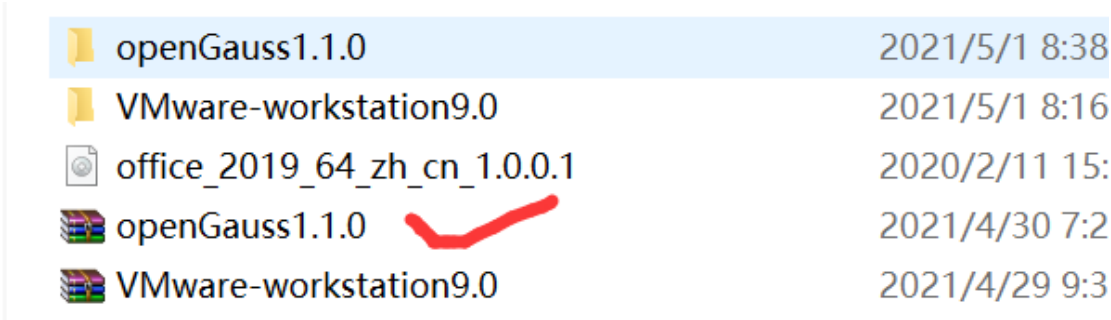
提取码：3q8r

一、VMware 安装

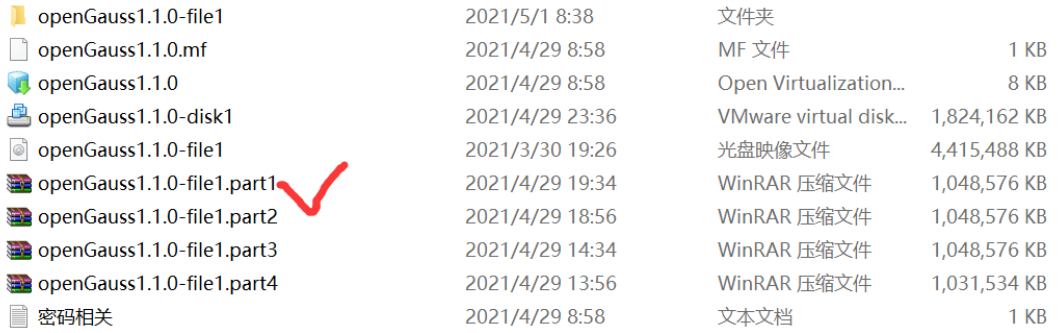


以管理员身份运行安装文件，一路安装下去，然后双击汉化补丁即可，需要序列号的查看 txt 文档。

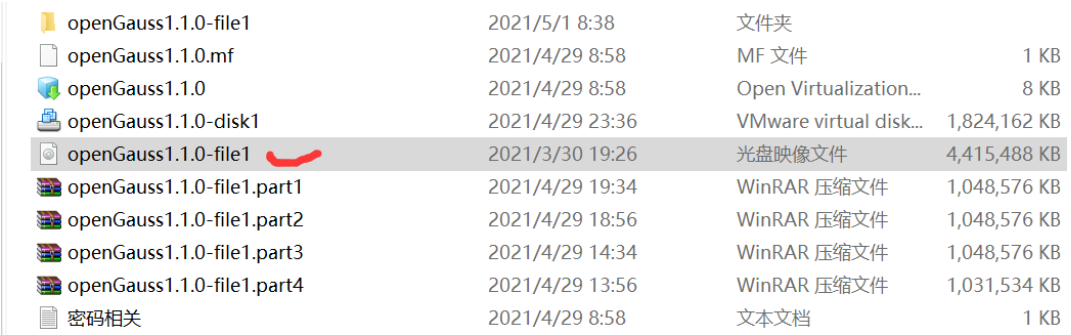
二、openGauss 安装



解压缩 opendGauss1.1.0 后，进入文件夹，

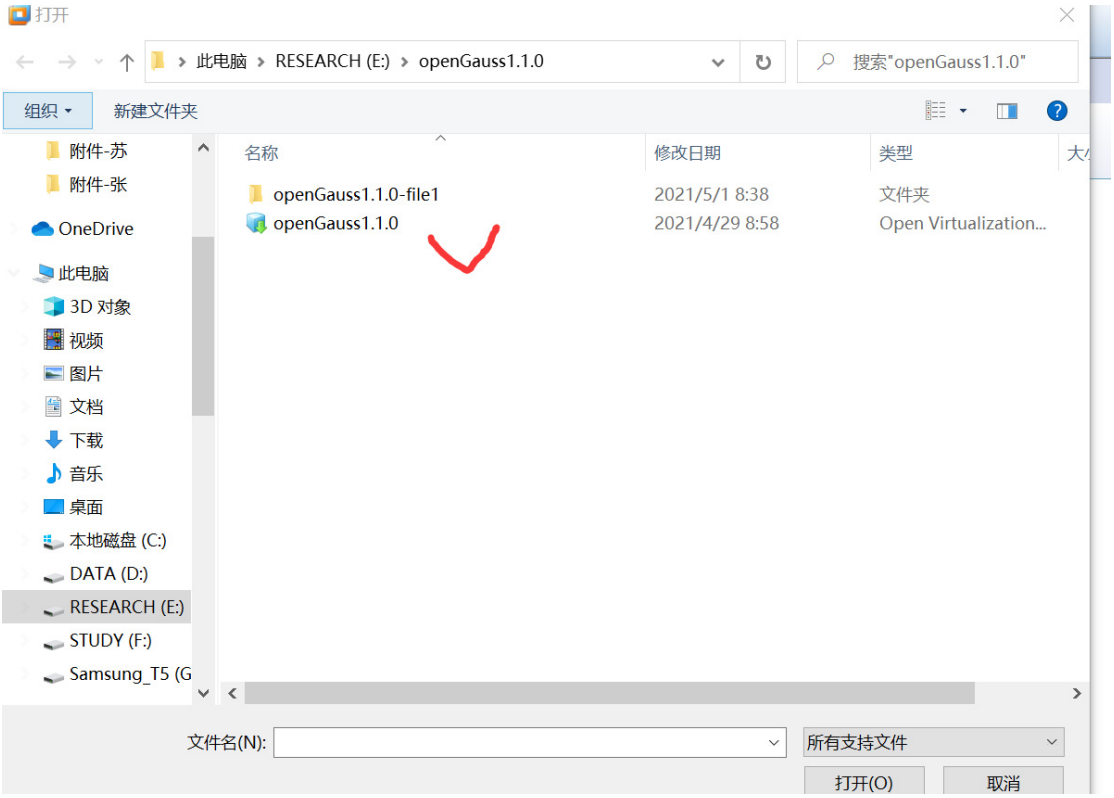


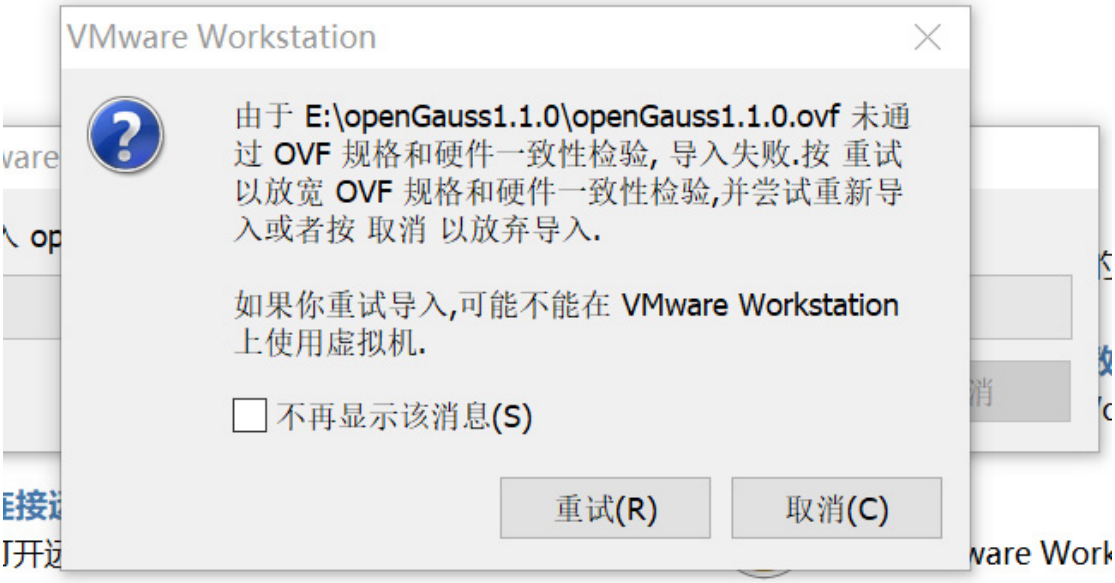
继续解压缩 openGauss1.1.0-file1.part1，将 openGauss1.1.0-file1 文件夹下的 openGauss1.1.0-file1.iso 光盘镜像文件拷贝到与虚拟机文件 openGauss1.1.0-disk1 同目录，如下：



打开 vmware，选择中间的“打开虚拟机”，

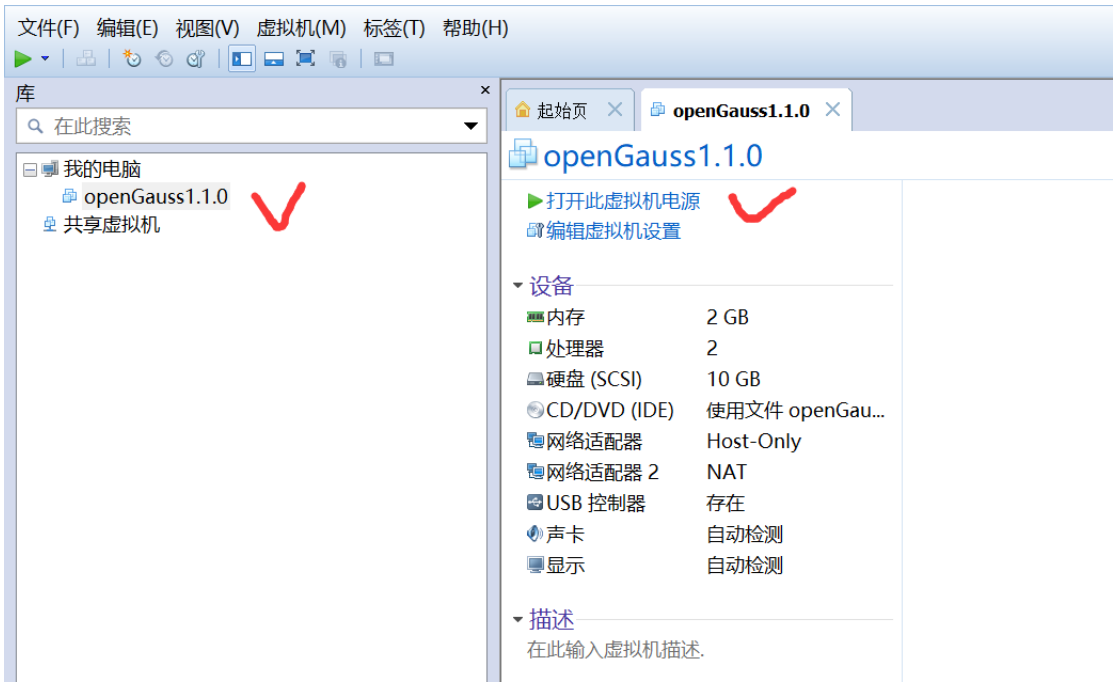
找到解压缩的 openGauss1.1.0 文件夹，选择虚拟机文件，点击右下角“打开”。



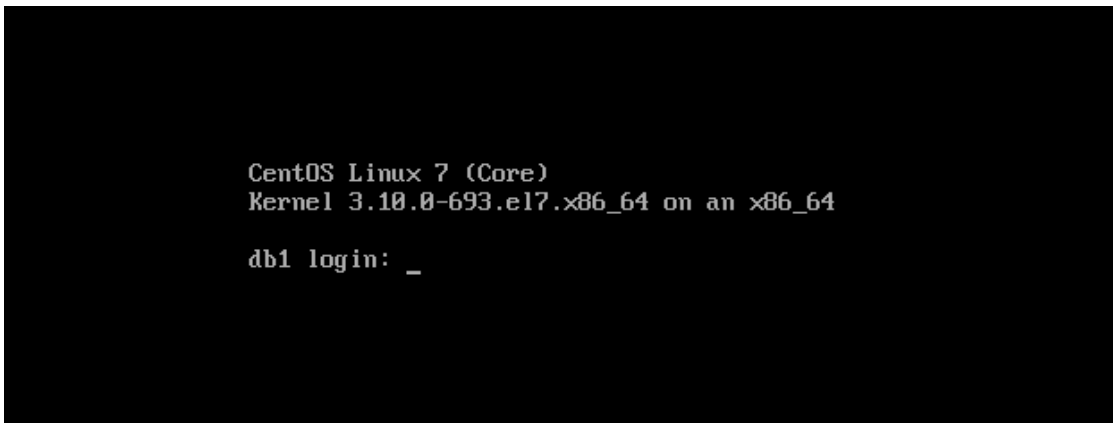


如遇到上述对话框，点击“重试”即可。

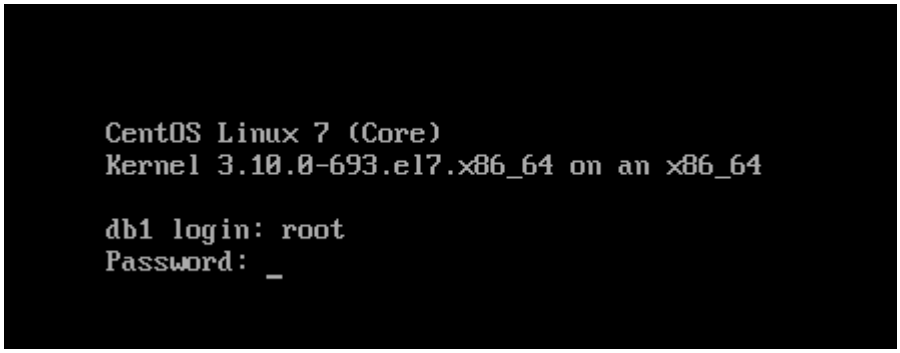
安装完成后，点击左上角的虚拟机“openGauss1.1.0”，点击右边的“打开此虚拟机电源”。



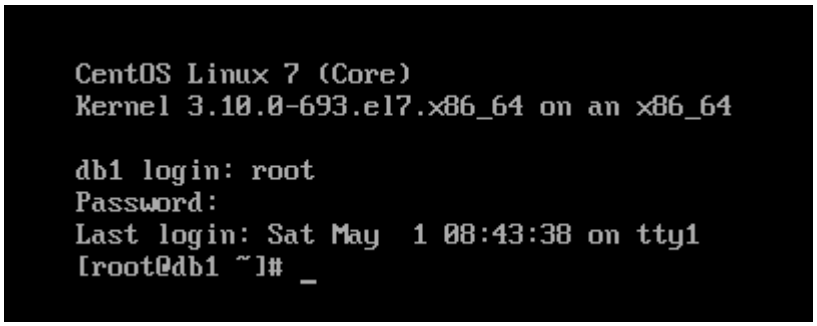
进入 linux 后，



输入：root



Password 输入：openGauss@123



按如下步骤，进行 IP 地址配置。

IP 地址修改步骤

注意，每次启动 vmware 后，IP 地址都会发生变化，导致数据库启动失败，因为需要重新查询和配置虚拟机 ip 地址。

步骤一：确认网络。

在 Linux 操作系统上，通过 ifconfig 来查看二张网卡是否都正常启动，具体如下：

```
[root@db1 ~]# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.28.129  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::ac2f:dc4f:edfe:1d57  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:0f:78:e3  txqueuelen 1000  (Ethernet)
        RX packets 519  bytes 48509 (47.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 178  bytes 52937 (51.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.220.129  netmask 255.255.255.0  broadcast 10.0.3.255
        inet6 fe80::bedc:2040:4b9:23ed  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:45:8d:f0  txqueuelen 1000  (Ethernet)
        RX packets 72  bytes 10702 (10.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 124  bytes 11664 (11.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
.....
virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:05:11:90  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

记录下第二个网卡的网址，例如本案例中的 **192.168.220.129** 。

步骤二：修改数据库的 pg_hba.conf 文件。

在 GS_HOME 中查找 pg_hba.conf 文件，本实验中数据库 GS_HOME 设置的为/gaussdb/data/db1，实际操作中 GS_HOME 地址可以查看安装时的配置文件：<PARAM name="dataNode1" value="**/gaussdb/data/db1**"/>。分别执行如下命令：

```
cd /gaussdb/data/db1/
vi pg_hba.conf
```

输入“:90”找到对应位置，然后输入“i”切换到 INSERT 模式，将以下“host all all 192.168.119.131/32 trust”中的 IP 地址换为记录下的**第二个网卡的地址（请根据实际情况修改）**，修改后按下“ECS”键，退出 INSERT 模式，输入“:wq”后回车保存。

修改前：

```
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all    all      192.168.119.131/32      trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
```

修改后：

```
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all    all      192.168.220.129/32      trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
```

使用 omm 用户登陆，使用 gs_ctl 将策略生效。

```
su - omm
gs_ctl reload -D /gaussdb/data/db1/
```

返回结果为：

```
[2020-07-23 15:39:55.398][71828][][gs_ctl]: gs_ctl reload ,datadir is -D "/gaussdb/data/db1"
server signaled
```

注：如果之前没有启动过数据库，返回结果如下，继续操作即可，不用管相关的提示：

```
[2021-03-18 15:37:57.305][3093][][gs_ctl]: gs_ctl reload ,datadir is /gaussdb/data/db1
[2021-03-18 15:37:57.306][3093][][gs_ctl]:  PID file “/gaussdb/data/db1/postmaster.pid” dose not exist
[2021-03-18 15:37:57.306][3093][][gs_ctl]: Is server running?
```

步骤三：修改数据库监听地址。

在 GS_HOME 中，本实验中数据库 GS_HOME 设置的为/gaussdb/data/db1。

```
cd /gaussdb/data/db1/
vi postgresql.conf
```

输入“:60”找到对应位置，然后输入“i”切换到 INSERT 模式，将 listen_addresses 的值修改成为**第二个网卡的地址（请根据实际情况修改）**，修改后按下“ECS”键，退出 INSERT 模式，输入“:wq”后回车保存。

修改前：

```
listen_addresses = '192.168.119.131'                    # what IP address(es) to listen on;
```

修改后：

```
listen_addresses = '192.168.220.129'                    # what IP address(es) to listen on;
```

修改完成后重启数据库生效（-D 后面的数据库默认路径，需要根据实际情况进行修改）。

```
gs_ctl restart -D /gaussdb/data/db1/
```

步骤四：启动数据库成功。

```
[omm@db1 db1]$ gs_om -t start
Starting cluster.
=========================================
[SUCCESS] db1:
[2021-04-01 15:09:02.959][4472][][gs_ctl]: gs_ctl started,datadir is /gaussdb/data/db1
[2021-04-01 15:09:02.966][4472][][gs_ctl]:  another server might be running; Please use the restart command
=========================================
Successfully started.
```

```
su - omm
```

使用如下命令连接数据库。

```
gsql -d postgres -p 26000 -r
```

输入 help 指令。

```
postgres=#help
```

显示如下帮助信息：

```
You are using gsql, the command-line interface to gaussdb.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with gsql commands
       \g or terminate with semicolon to execute query
       \q to quit
```

查看版权信息。

```
postgres=#\copyright
```

显示如下版权信息：

```
openGauss Database Management System
Copyright (c) Huawei Technologies Co., Ltd. 2020. All rights reserved.
```

查看 openGauss 支持的所有 SQL 语句。

```
postgres=#\h
```

显示如下信息：

```
Available help:
  ABORT
  ALTER AGGREGATE
  ALTER APP WORKLOAD GROUP
... ...
```

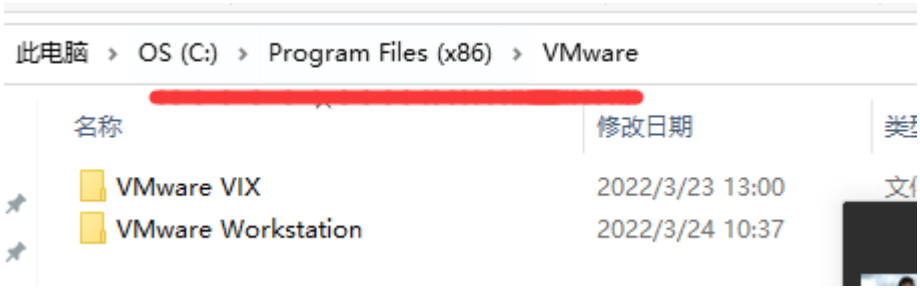
查看 CREATE DATABASE 命令的参数可使用下面的命令。

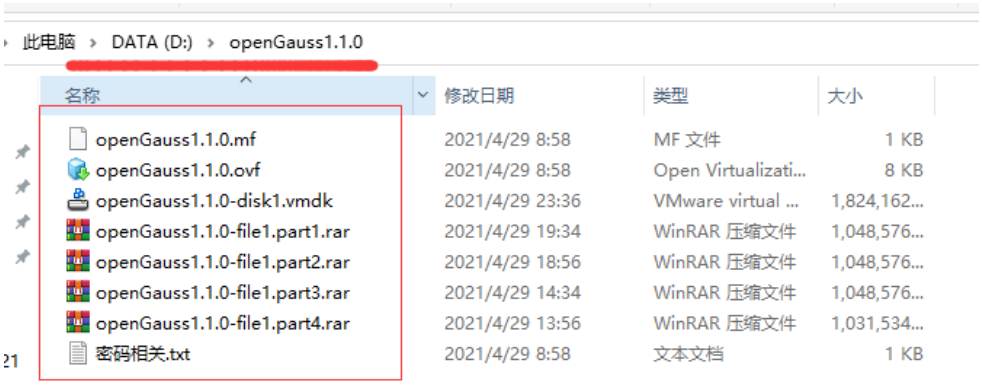
输入指令：create database school;
输入指令：/c school，切换到 school 数据库
根据前面学习的内容创建三张表格并插入数据，测试大作业的题目。

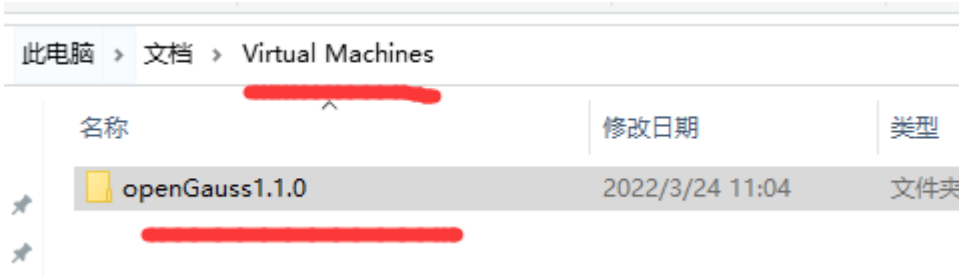
退出虚拟机时，选择关闭虚拟机电源。

1.只针对 VMware+OpenGauss 虚拟镜像模式，其他模式请自行搜索资料解决，解决不了建议回到上述模式！

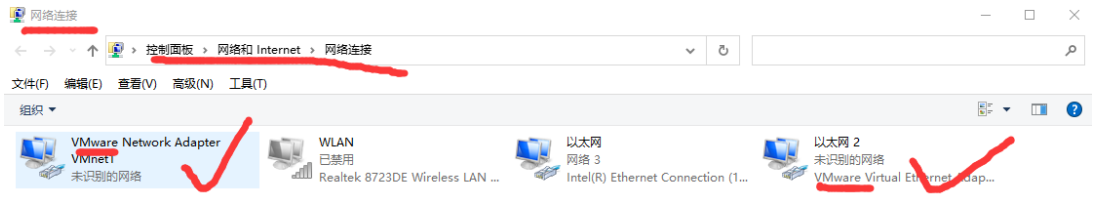
2. VMware 安装目录，OpenGuass 虚拟镜像保存目标，虚拟机目录，一律采用英文路径，不要出现中文文件夹。





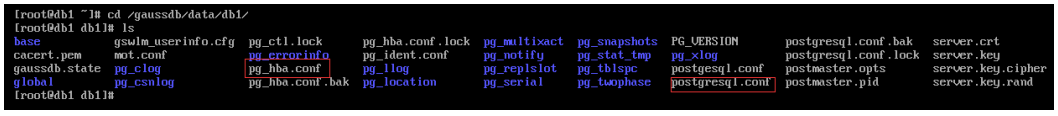


VMware 安装后“网络连接”中会有两个虚拟网卡，没有的重新安装。



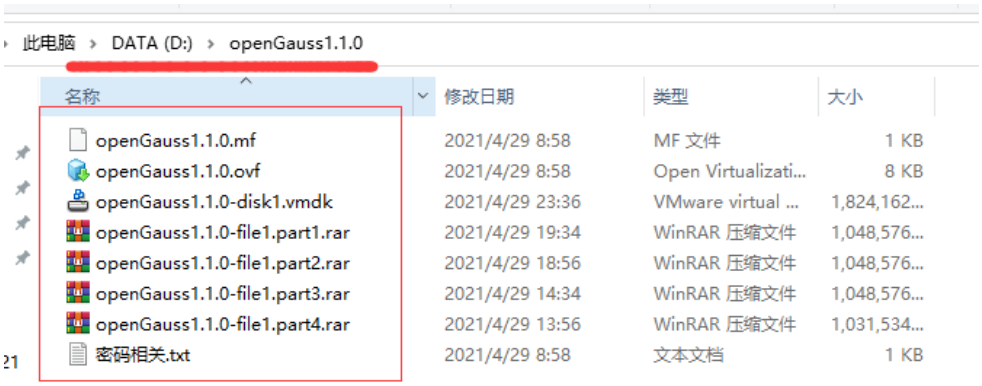
在 VMware 中，左上角：编辑->虚拟网络编辑器中会看到 IP 地址的自动分配。

3. 用 root 用户进入后，进入/gaussdb/data/db1 目录，注意输入的空格。用 ls 指令查看，里面有常用的 2 个配置文件。



如果显示没有这个目录，则你解压的 OpenGuass 镜像不完整，缺失文件。这个时候，重新解压，删掉 VMware 中的原始 openGuass 镜像，重新加载你解压的镜像文件。如何还是显示没有这个目录，检查有没有中文路径出现。

4.如果还是解决不了，将别的同学顺利安装配置成功的 opengauss 解压镜像文件拷到你的电脑里面，用 VMware 加载他的 opengauss 镜像，按照教程修改配置自己的 IP 地址！



5. 使用 Data Studio 之前按照如下教程，

https://blog.opengauss.org/zh/post/lihongda/opengauss%E5%B8%B8%E7%94%A8%E7%9A%84%E5%AE%A2%E6%88%B7%E7%AB%AF%E8%BF%9E%E6%8E%A5%E5%B7%A5%E5%85%B7/

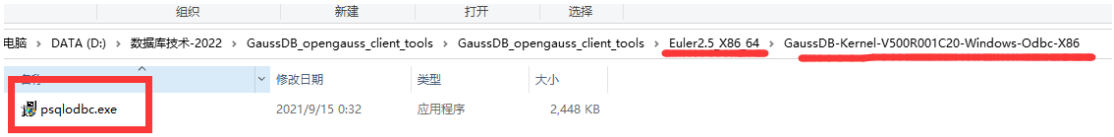
openGauss 常用的客户端连接工具

下载群里面的 Java 运行环境安装包，解压缩并安装。

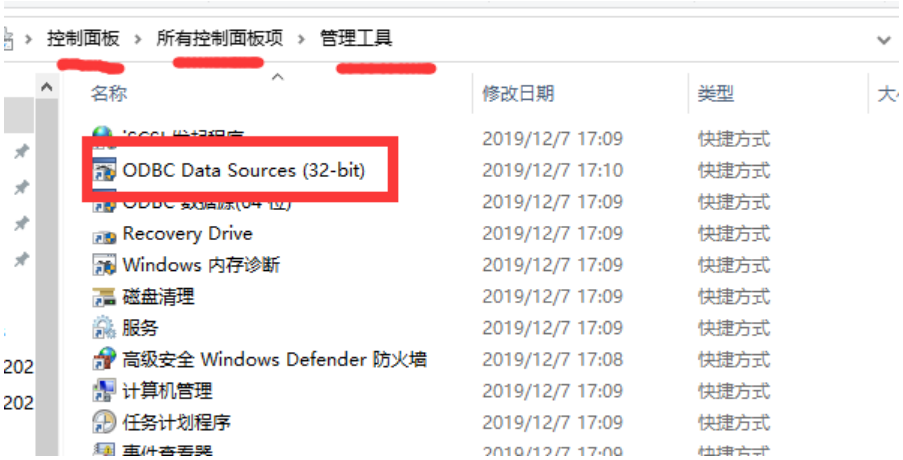
OpenGauss 的 ODBC 驱动安装和配置

合肥工业大学 计算机与信息学院 张国富

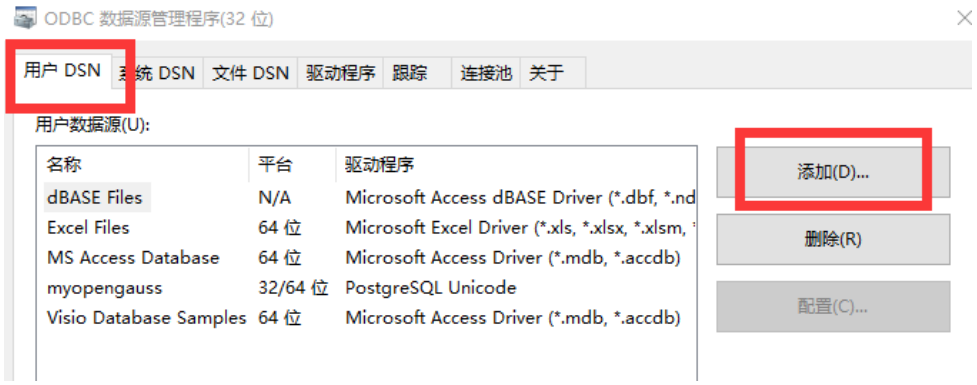
1. 下载群里的 GaussDB_opengauss_client_tools.zip，解压缩。进入 Euler2.5_X86_64 文件夹，选择 GaussDB-Kernel-V500R001C20-Windows-Odbc-X86.tar.gz 进行解压缩，以管理员身份运行安装程序 psqlodbc.exe，按照默认设置安装即可。



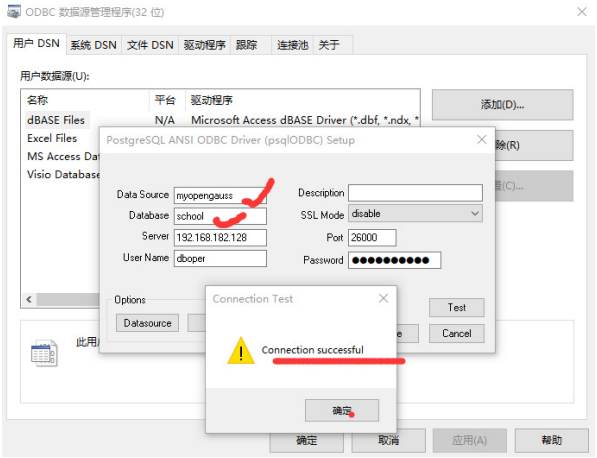
2. 打开“控制面板”，选择“管理工具”，选择“ODBC Data Source (32-bit)”并双击运行。



选择“用户 DSN”分页，点击右边的添加，



从下拉菜单中选择“PostgreSQL Unicode”，点击“完成”，开始配置 ODBC 数据源驱动。



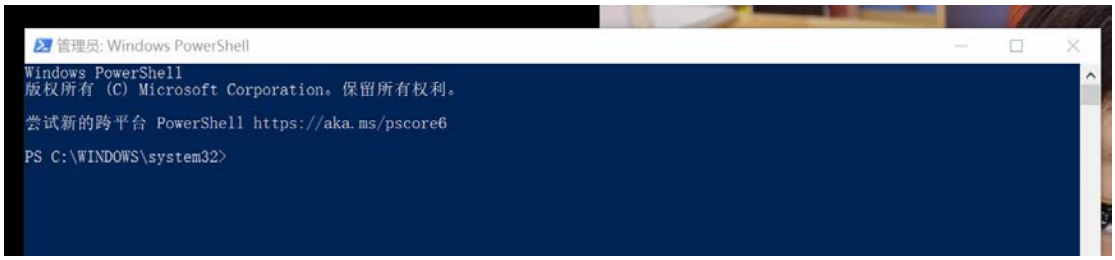
Data Source：名称随便取，但要记住这个名称，后面 c++连接需要。
Database：需要打开哪一款数据库，比如我们的 school 案例库。
Server：数据库服务器，与 Data Studio 的配置一致。
Port：26000
User Name：dboper
Password：dboper@123
注意：需要在测试 Data Studio 工具的时候创建好上述 dboper 用户账号，可查看群里面的“Data Studio 连接虚拟机方法.doc”。

点击右下角的“Test”，如果出现“successful”提示框，表示连接成功，点击“Save”保存配置即可。

windows 环境下利用 Data Studio 连接虚拟机 openGauss 的方法

合肥工业大学 计算机与信息学院 张国富

1.在 windows 下，鼠标移至左下角开始，鼠标右键，要选择“windows PowerShell(管理员)”运行



2.输入 ipconfig - all 指令，查看 VMware 虚拟网卡 IP 地址，这里注意选择与 wmware 连接方式相同的 VMnet8 的地址。例如，本机为：192.168.44.1

```
S C:\Users\DELL> ipconfig -all

indows IP 配置

   主机名  . . . . . . . . . . . . . : DESKTOP-EBQ48OR
   主 DNS 后缀 . . . . . . . . . . . :
   节点类型  . . . . . . . . . . . . : 混合
   IP 路由已启用 . . . . . . . . . . : 否
   WINS 代理已启用 . . . . . . . . . : 否

以太网适配器 以太网:

   连接特定的 DNS 后缀 . . . . . . . :
   描述. . . . . . . . . . . . . . . : Intel(R) Ethernet Connection (7) I219-LM
   物理地址. . . . . . . . . . . . . : E4-54-E8-AA-E4-61
   DHCP 已启用 . . . . . . . . . . . : 是
   自动配置已启用. . . . . . . . . . : 是
   IPv6 地址 . . . . . . . . . . . . : 240e:362:7bc:1500:143c:c3d6:2729:3(首选)
   获得租约的时间  . . . . . . . . . : 2022年3月23日 16:37:50
   租约过期的时间  . . . . . . . . . : 2022年3月23日 18:37:50
   IPv6 地址 . . . . . . . . . . . . : 240e:362:7bc:1537:5800:b9cd:f026:9e63(首选)
   临时 IPv6 地址. . . . . . . . . . : 240e:362:7bc:1537:7dc0:441c:9c64:2bec(首选)
   本地链接 IPv6 地址. . . . . . . . : fe80::5800:b9cd:f026:9e63%14(首选)
   IPv4 地址 . . . . . . . . . . . . : 192.168.3.29(首选)
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   获得租约的时间  . . . . . . . . . : 2022年3月23日 16:38:06
   租约过期的时间  . . . . . . . . . : 2022年3月24日 16:37:52
   默认网关. . . . . . . . . . . . . : fe80::1%14
                                       192.168.3.1
   DHCP 服务器 . . . . . . . . . . . : 192.168.3.1
   DHCPv6 IAID . . . . . . . . . . . : 165958888
   DHCPv6 客户端 DUID  . . . . . . . : 00-01-00-01-27-32-71-B4-E4-54-E8-AA-E4-61
   DNS 服务器  . . . . . . . . . . . : fe80::1%14
                                       192.168.3.1
                                       fe80::1%14
   TCPIP 上的 NetBIOS  . . . . . . . : 已启用

以太网适配器 VMware Network Adapter VMnet1:

   连接特定的 DNS 后缀 . . . . . . . :
   描述. . . . . . . . . . . . . . . : VMware Virtual Ethernet Adapter for VMnet1
   物理地址. . . . . . . . . . . . . : 00-50-56-C0-00-01
   DHCP 已启用 . . . . . . . . . . . : 是
   自动配置已启用. . . . . . . . . . : 是
   本地链接 IPv6 地址. . . . . . . . : fe80::b54d:1433:a651:c85d%7(首选)
   IPv4 地址 . . . . . . . . . . . . : 192.168.229.1(首选)
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   获得租约的时间  . . . . . . . . . : 2022年3月23日 16:38:10
   租约过期的时间  . . . . . . . . . : 2022年3月23日 17:07:55
   默认网关. . . . . . . . . . . . . :
   DHCP 服务器 . . . . . . . . . . . : 192.168.229.254
   DHCPv6 IAID . . . . . . . . . . . : 134238294
   DHCPv6 客户端 DUID  . . . . . . . : 00-01-00-01-27-32-71-B4-E4-54-E8-AA-E4-61
   DNS 服务器  . . . . . . . . . . . : fec0:0:0:ffff::1%1
                                       fec0:0:0:ffff::2%1
                                       fec0:0:0:ffff::3%1
   TCPIP 上的 NetBIOS  . . . . . . . : 已启用

以太网适配器 以太网 2:

   连接特定的 DNS 后缀 . . . . . . . :
   描述. . . . . . . . . . . . . . . : VMware Virtual Ethernet Adapter for VMnet8
   物理地址. . . . . . . . . . . . . : 00-50-56-C0-00-08
   DHCP 已启用 . . . . . . . . . . . : 是
   自动配置已启用. . . . . . . . . . : 是
   本地链接 IPv6 地址. . . . . . . . : fe80::2cf7:b3f5:ec1:3f5a%15(首选)
   IPv4 地址 . . . . . . . . . . . . : 192.168.44.1(首选)
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   获得租约的时间  . . . . . . . . . : 2022年3月23日 16:38:10
   租约过期的时间  . . . . . . . . . : 2022年3月23日 17:07:54
   默认网关. . . . . . . . . . . . . :
   DHCP 服务器 . . . . . . . . . . . : 192.168.44.254
   DHCPv6 IAID . . . . . . . . . . . : 721440854
   DHCPv6 客户端 DUID  . . . . . . . : 00-01-00-01-27-32-71-B4-E4-54-E8-AA-E4-61
   DNS 服务器  . . . . . . . . . . . : fec0:0:0:ffff::1%1
                                       fec0:0:0:ffff::2%1
                                       fec0:0:0:ffff::3%1
   主 WINS 服务器  . . . . . . . . . : 192.168.44.2
   TCPIP 上的 NetBIOS  . . . . . . . : 已启用
```

3. 打开虚拟机、进入 openGauss，以 root 用户进入。修改数据库的 pg_hba.conf 文件。

```
[root@db1 ~]# cd /gaussdb/data/db1
[root@ecs-b5cb db1]# vi pg_hba.conf
```

找到 IPv4 部分，在两行 host all all 下面添加如下三行代码：

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
                                                                trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all    all    192.168.44.128/32    trust
host all all 0.0.0.0/0 sha256
host all all 192.168.44.1/32 md5
host all all 0.0.0.0/0 md5
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     omm                                     trust
#host    replication     omm         127.0.0.1/32                trust
#host    replication     omm         ::1/128                     trust
```

将以下内容添加进 pg_hba.conf 文件。

```
host all all 0.0.0.0/0 sha256
host all all 192.168.44.1/32 md5
host all all 0.0.0.0/0 md5
```

4.修改数据库监听地址。

修改数据库的 postgresql.conf 文件。

```
[root@db1 ~]# vi postgresql.conf
```

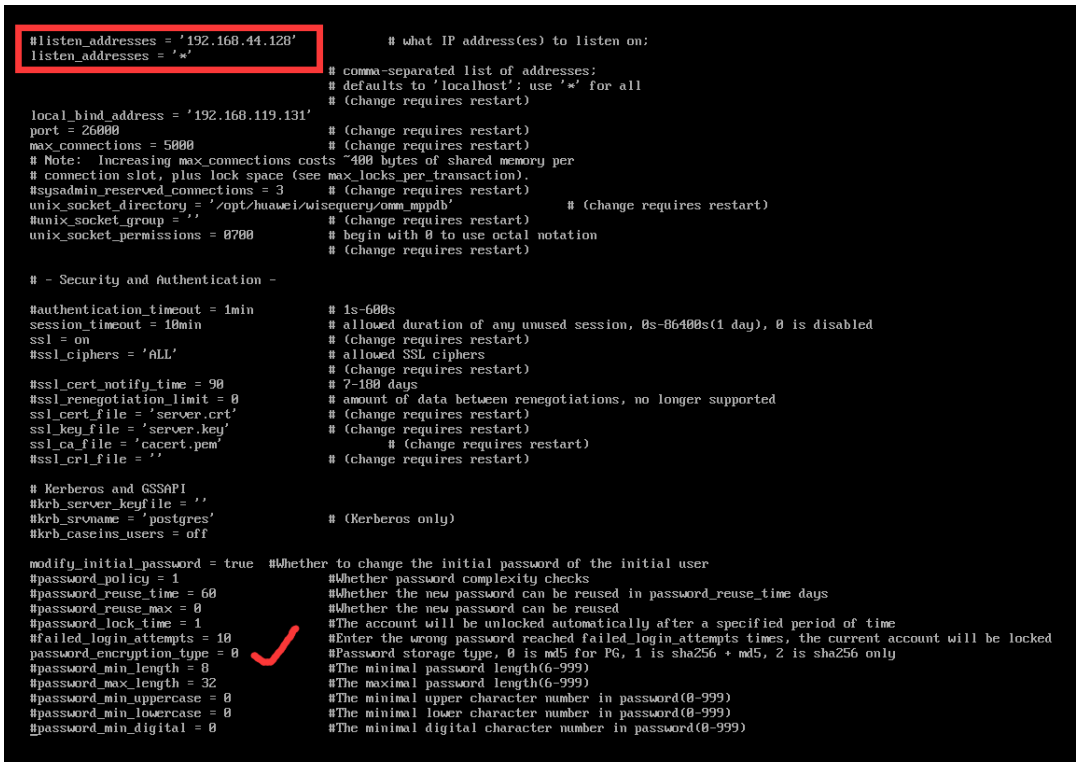
将 listen_addresses 的值修改成为 * 。

```
listen_addresses = '*'
```

并删掉 password_encryption_type 前面的#，且赋值 0。

```
password_encryption_type = 0
```

如下：



修改完成后切换至 omm 用户环境，使用 gs_ctl 将策略生效。

```
[root@db1 db1]#su - omm
[omm@db1 ~]$gs_ctl reload -D /gaussdb/data/db1/
```

返回结果为：

```
[2020-07-23 15:39:55.398][71828][][gs_ctl]: gs_ctl reload ,datadir is -D "/gaussdb/data/db1"
server signaled
```

使用 gs_om -t restart 命令重启数据库。

5. 登陆数据库并创建“dboper”用户，密码为“**dboper@123**”（密码可自定义），同时进行授权，并退出数据库。

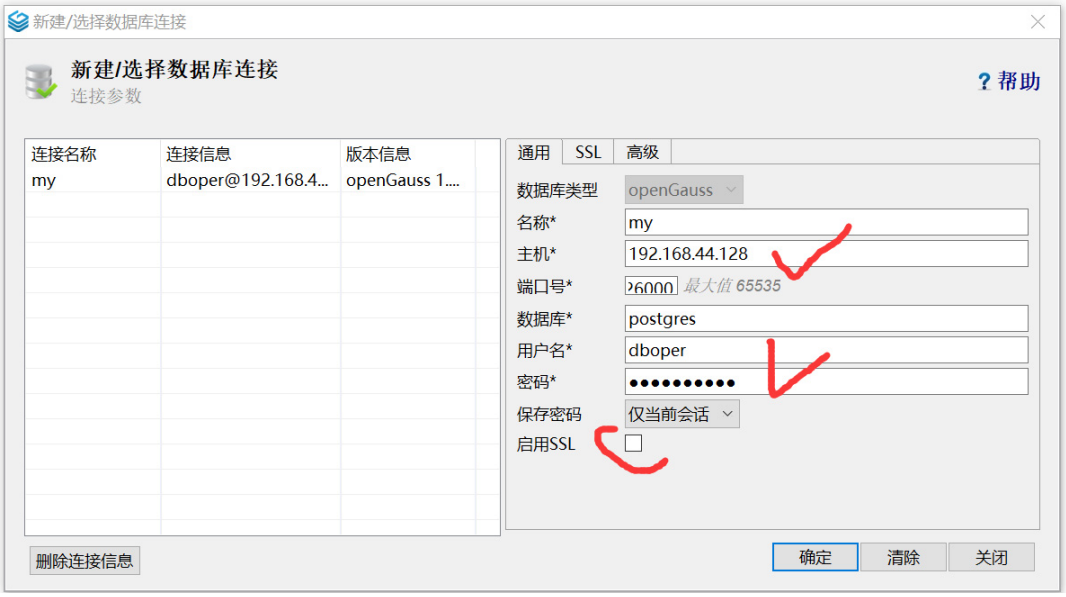
```
[omm@db1 ~]$gsql -d postgres -p 26000 -r
postgres=#CREATE USER dboper IDENTIFIED BY 'dboper@123';
CREATE ROLE
postgres=#alter user dboper sysadmin;
ALTER ROLE
postgres=# \q
```

退出 OMM 用户环境

```
[omm@ecs-b5cb ~]$ exit
logout
[root@ecs-b5cb ecs-b5cb]#
```

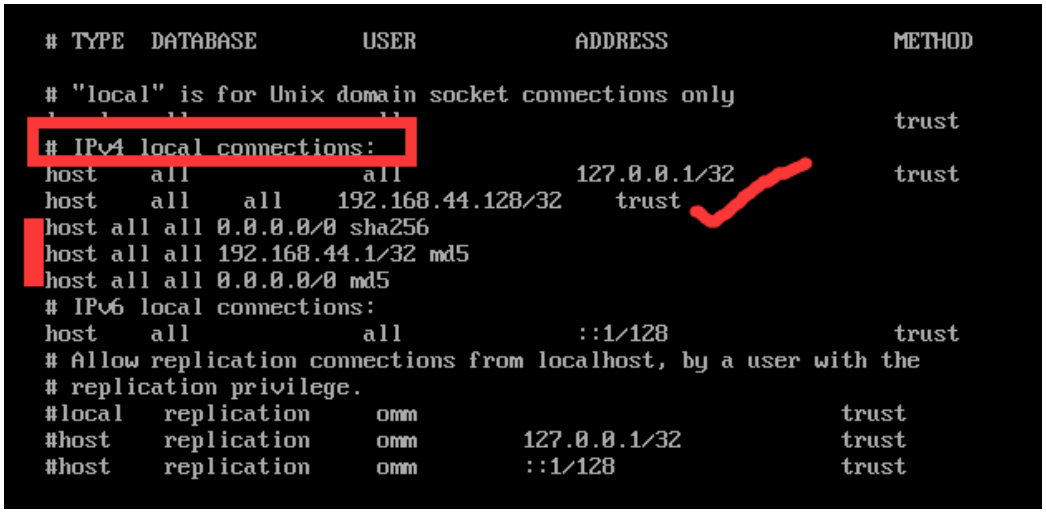
6. 进行 Data Sudio 连接

打开 Data Studio，点击菜单栏文件-新建连接，



名称：随便起

主机：虚拟机的 ip，这里要注意，是 IPv4 下面第二行 trust 对应的那个 IP 地址，不是其他 IP，这个 IP 一定和你前面查询的 VMnet8 网卡的 IP 地址在一个地址段内！例如下图中的 192.168.44.128 与 192.168.44.1 在同一地址段内，选择 192.168.44.128 作为数据库服务器 IP 进行连接。



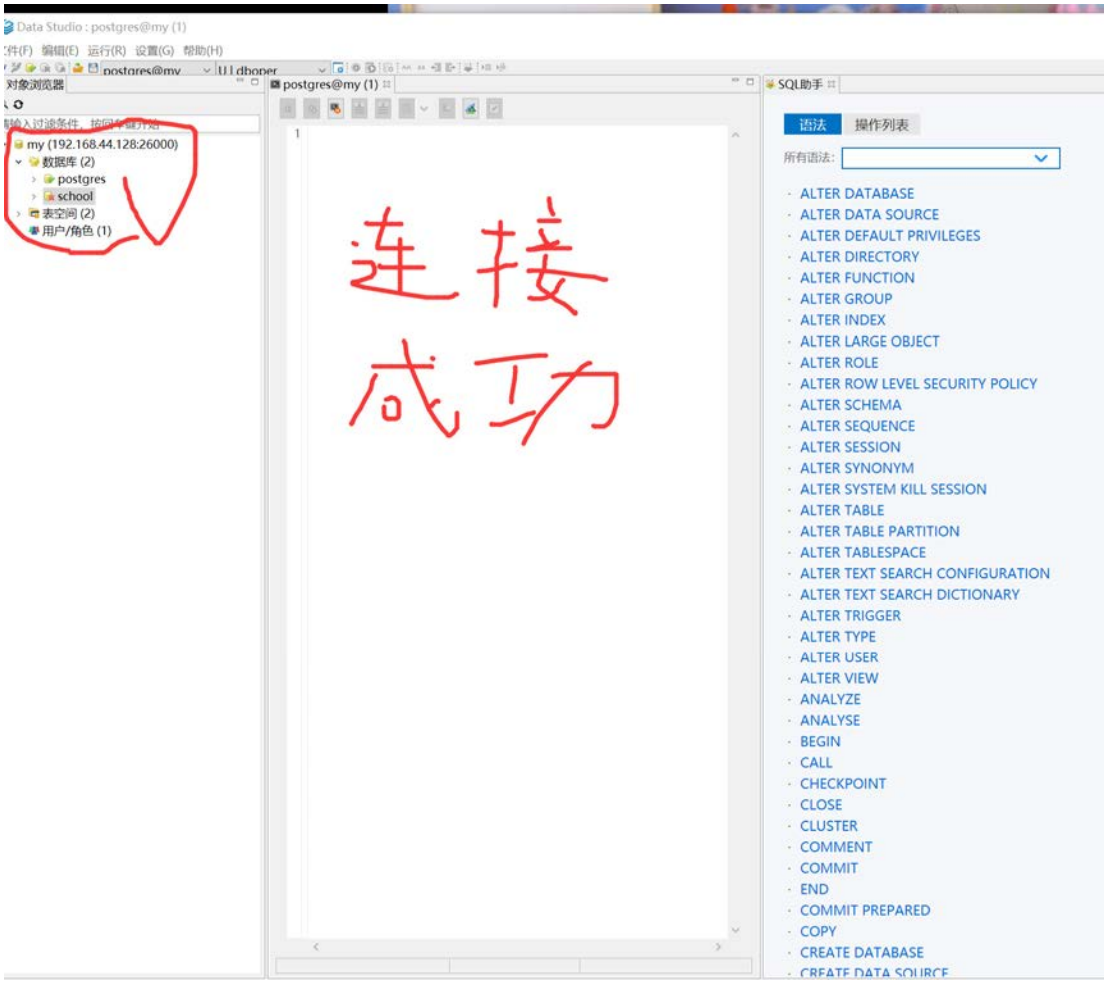
端口号：26000

数据库：填 postgres

用户名、密码为刚刚创建的 dboper 用户

不启用 SSL

然后点击确定，首次连接会出现测试连接，大约 30s 后就可以了。

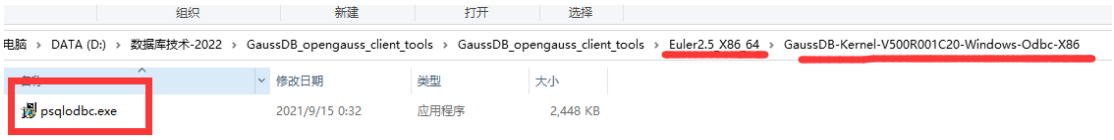


VS2013+OpenGuass 开发教程（一）

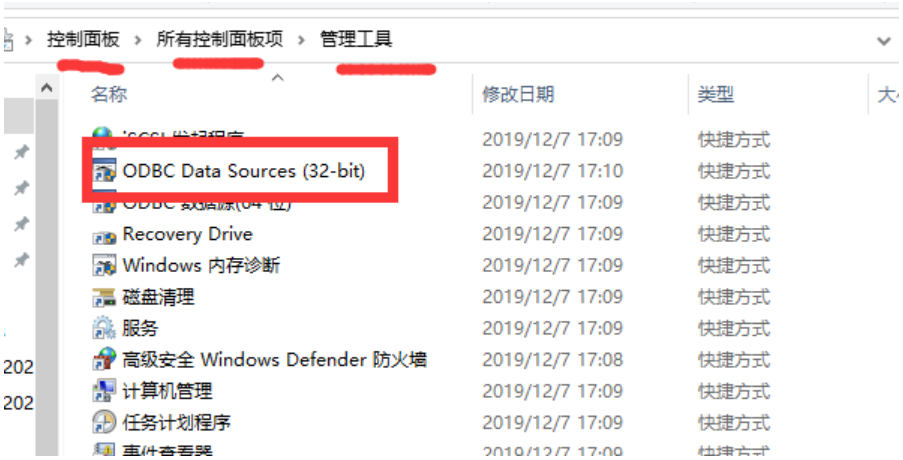
合肥工业大学 计算机与信息学院 张国富

zgf@hfut.edu.cn

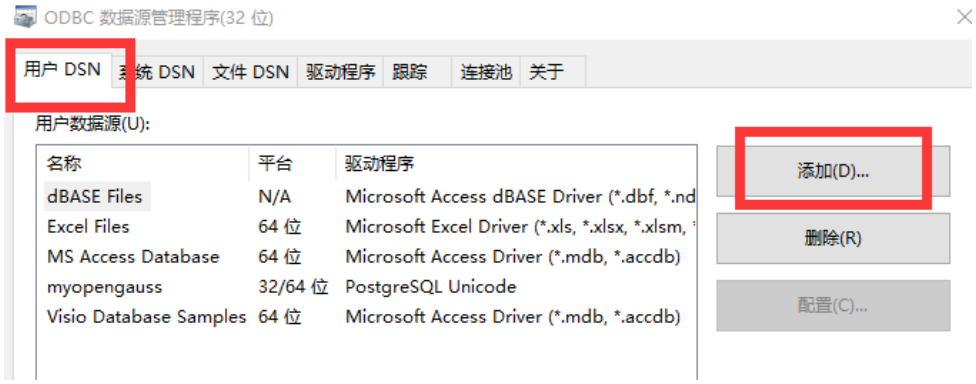
1.下载群里的 GaussDB_opengauss_client_tools.zip，解压缩。进入 Euler2.5_X86_64 文件夹，选择 GaussDB-Kernel-V500R001C20-Windows-Odbc-X86.tar.gz 进行解压缩，以管理员身份运行安装程序 psqlodbc.exe，按照默认设置安装即可。



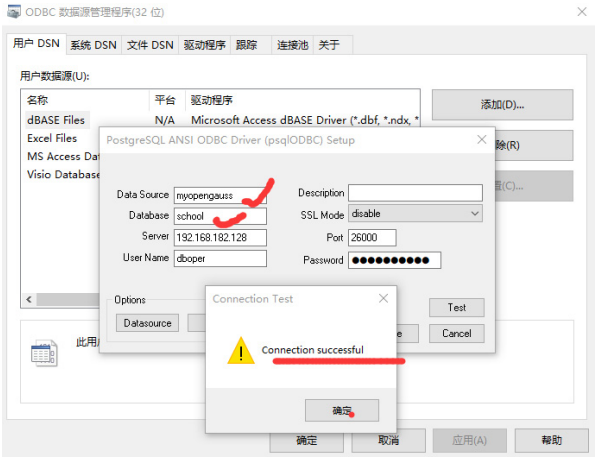
2.打开“控制面板”，选择“管理工具”，选择“ODBC Data Source (32-bit)”并双击运行。



选择“用户 DSN”分页，点击右边的添加，



从下拉菜单中选择“PostgreSQL Unicode”，点击“完成”，开始配置 ODBC 数据源驱动。



Data Source：名称随便取，但要记住这个名称，后面 c++连接需要。
Database：需要打开哪一款数据库，比如我们的 school 案例库。
Server：数据库服务器，与 Data Studio 的配置一致。
Port：26000
User Name：dboper
Password：dboper@123
注意：需要在测试 Data Studio 工具的时候创建好上述 dboper 用户账号，可查看群里面的“Data Studio 连接虚拟机方法.doc”。

点击右下角的“Test”，如果出现“successful”提示框，表示连接成功，点击“Save”保存配置即可。

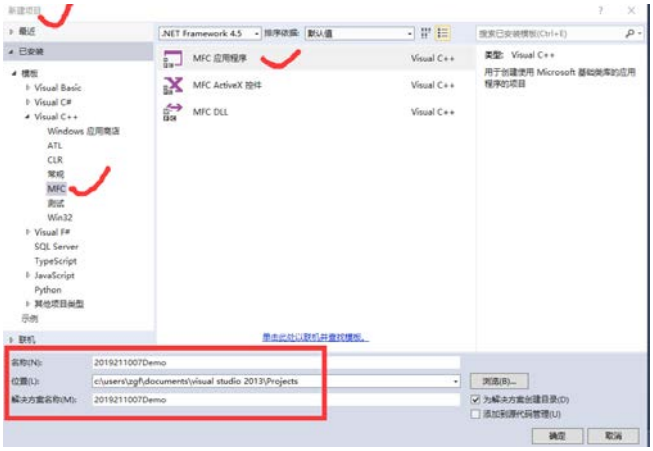
VS2013+OpenGuass 开发教程（二）

合肥工业大学 计算机与信息学院 张国富

zgf@hfut.edu.cn

一． 创建一个初始工程

1. 打开 VS2013，选择左上角“文件”→“新建”→“项目”；
2. 在“新建项目”中，左手边选择“Virtual C++”，右手边选择“MFC 应用程序”，在下面的“名称”中输入你的工程名，工程名要包含学号，例如：2019211007Demo。注意查看工程保存的“位置”，以便后期查找这个工程。



点击右下角“确定”。

3. 点击左手边“应用程序类型”，选择“基于对话框”； 再点击左手边的“高级功能”，勾选“Windows 套接字”。最后点击“完成”，一个新的工程就创建好了。

二． 连接 Opengauss 数据库

1. 选择右手边中间的“解决方案资源管理器”，展开“头文件”，点击“stdafx.h”。在

   #include <afxsock.h>　　// MFC 套接字扩展

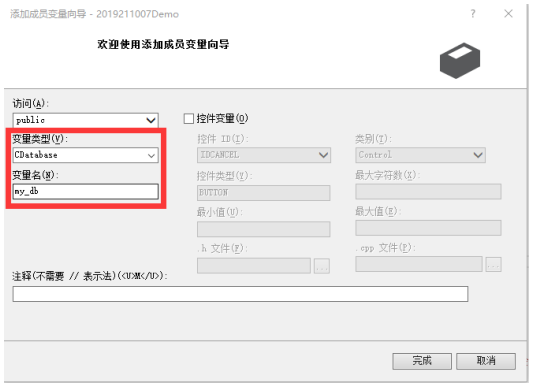
   这个语句下面添加代码。

   ****

   #include <afxdb.h>　　// CDatabase 类头文件

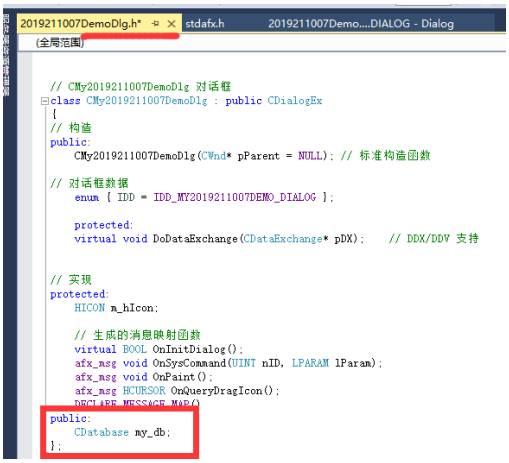
   ****

2. 创建 CDatabase 对象。选择右手边中间的“类视图”，展开总类“***Demo”，选择“***DemoDlg”类，点击鼠标右键，选择“添加”→添加变量。



   在“变量类型”下输入：CDatabase；在变量名下输入：my_db（可任意取）。输入后点击右下角的“完成”。

   选择右手边中间的“解决方案资源管理器”，展开“头文件”，点击“***DemoDlg.h”。在类声明的最下面会发现刚刚添加的 CDatabase 对象 my_db。



```
// CMy2019211007DemoDlg 对话框
class CMy2019211007DemoDlg : public CDialogEx
{
// 构造
public:
    CMy2019211007DemoDlg(CWnd* pParent = NULL); // 标准构造函数

// 对话框数据
    enum { IDD = IDD_MY2019211007DEMO_DIALOG };

    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV 支持


// 实现
protected:
    HICON m_hIcon;

    // 生成的消息映射函数
    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    DECLARE_MESSAGE_MAP()
public:
    CDatabase my_db;
};
```

3. 回到“解决方案资源管理器”，展开“源文件”，点击“***DemoDlg.cpp”。找到 OnInitDialog()函数。在

   // TODO: 在此添加额外的初始化代码

   这个语句下面添加代码。

```
BOOL CMy2019211007DemoDlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();

    // 将“关于...”菜单项添加到系统菜单中。

    // IDM_ABOUTBOX 必须在系统命令范围内。
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        BOOL bNameValid;
        CString strAboutMenu;
        bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
        ASSERT(bNameValid);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // 设置此对话框的图标。  当应用程序主窗口不是对话框时，框架将自动
    //  执行此操作
    SetIcon(m_hIcon, TRUE);         // 设置大图标
    SetIcon(m_hIcon, FALSE);        // 设置小图标

    // TODO:  在此添加额外的初始化代码

    return TRUE;  // 除非将焦点设置到控件，否则返回 TRUE
}
```

```
**********
//数据库初始化
if (my_db.Open(_T("myopengauss")))//打开ODBC数据源驱动配置的用户DNS名称，返回值TRUE则表示正确连接上
{
        AfxMessageBox(_T("OpenGauss数据库连接成功！"));
}
else
{
        AfxMessageBox(_T("数据库连接失败，请检查虚拟机Opengauss是否运行或者ODBC驱动是否配置正确！"));
}

**********
```

（1）_T("")就是把引号内的字符串转换为宽字节的 Unicode 编码

（2）*关于 Open 函数：*

```
virtual BOOL Open(
    LPCTSTR lpszDSN, //打开 ODBC 管理器里面注册的 DSN 值
    BOOL bExclusive = FALSE,
    BOOL bReadOnly = FALSE,
    LPCTSTR lpszConnect = _T("ODBC;"), //指示连接是访问 ODBC 数据源
    BOOL bUseCursorLib = TRUE
        );
```

（3）关于 OnInitDialog()函数：

构造函数主要针对的是 C++的类对象的**成员变量的初始化**，是内在的；OnInitDialog()主要针对的是与类对象相关联的 windows **窗体上控件的初始化**问题，是外在的。

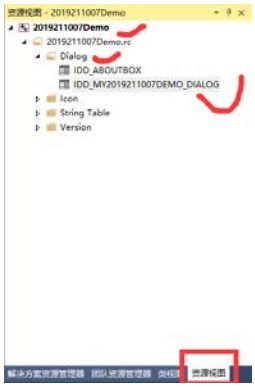
4. 选择右手边中间的“资源视图”，展开“Dialog”，选择“***DEMO_DIALOG”调出界面，把左手边界面上的“确定”按钮删除，仅保留按钮“取消”，然后选中“取消”，鼠标右键，“属性”，把右下角属性中的 Caption 值改为“关闭”，保存后，选中“关闭”按钮，鼠标右键，选择“添加事件处理程序”，选择第一个消息类型“BN_CLICKED”，表示点击鼠标一次响应对应的函数，下面的函数处理程序名称会自动添

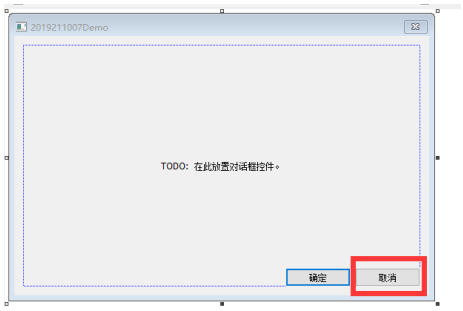
加，选择右下角的“添加编辑”，在“***DemoDlg.cpp”中会自动添加函数体 OnBnClickedCancel() ，在 “ ***DemoDlg.h ” 中会自动添加函数声明。

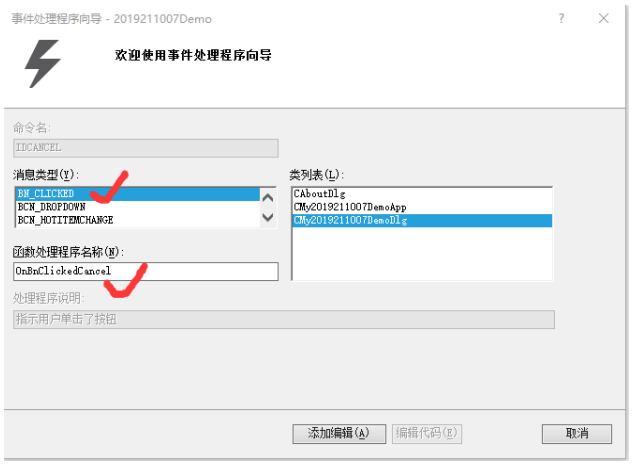
```
afx_msg void OnBnClickedCancel();
```

在“***DemoDlg.cpp”的上面“消息映射”的地方会有相应的宏定义。

```
BEGIN_MESSAGE_MAP(CMy2019211007DemoDlg, CDialogEx)
ON_BN_CLICKED(IDCANCEL, &CMy2019211007DemoDlg::OnBnClickedCancel)
END_MESSAGE_MAP()
```







回到 OnBnClickedCancel()函数体中，在

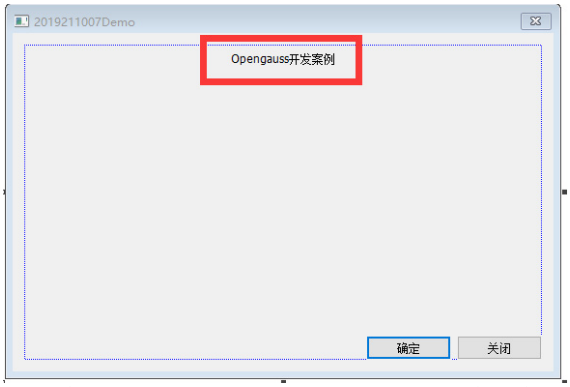
```
        // TODO: 在此添加控件通知处理程序代码
```

这一行的下面加入代码。

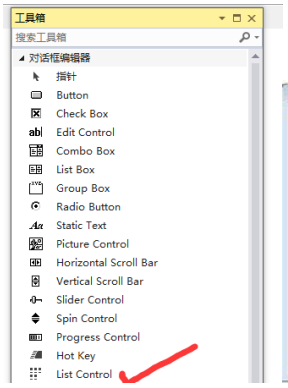
```
**********************************************************************
if (my_db.IsOpen())//判断是否处于数据库连接状态
{
        my_db.Close();//如果当前处于连接状态，从数据源断开连接，释放系统资源
}
**********************************************************************
```

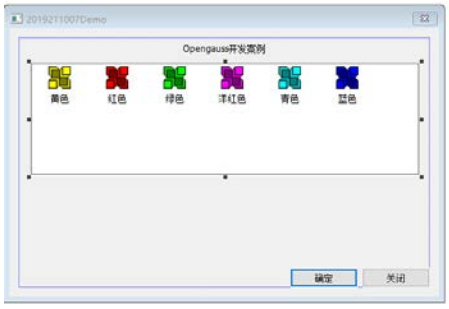
三． ListCtr 控件的初始化

1. 选择右手边中间的“资源视图”，展开“Dialog”，选择“***DEMO_DIALOG”调出界面。选中界面中间的 Static Text 控件，查看属性，将右下角属性中的 Caption 值改为“Opengauss 开发案例”，并将控件托到软件的最上面。



2. 打开“工具箱”，从中选择“List Control”，回到界面区域，在适当位置绘制这个控件。

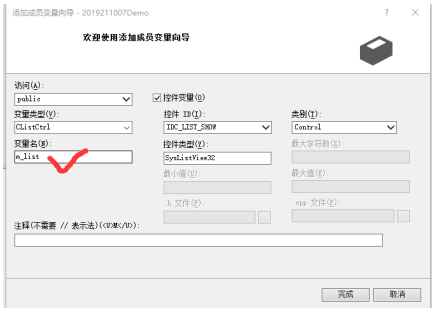




选中所画的控件，在右下角属性中，首先修改“ID”为“IDC_LIST_SHOW”，再将“View”改为“Report”。

注意：每个控件都有一个唯一的 ID 号，相当于数据库中的“主码”，可以唯一区分各个控件，这个 ID 号赋值时要按照控件的功能去给，以便于程序员浏览代码。

3. 选中界面上的 list 控件，鼠标右键，选择“添加变量”，在“变量名”下面输入 m_list，点击“完成”。



编译器在“***DemoDlg.h”中会自动添加 *CListCtrl* 对象，

CListCtrl m_list;

在“***DemoDlg.cpp”的上面“控件绑定”的地方会有相应的宏定义。

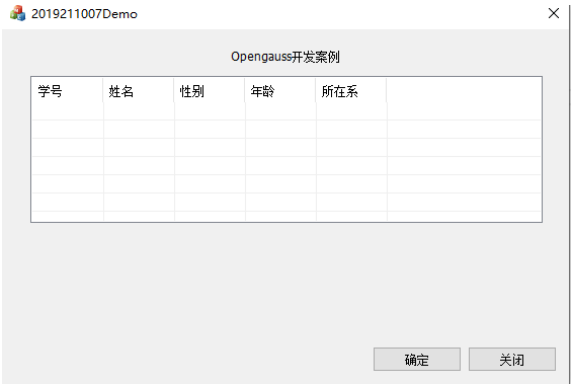
```
void CMy2019211007DemoDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialogEx::DoDataExchange(pDX);
	DDX_Control(pDX, IDC_LIST_SHOW, m_list);
}
```

这样，要想控制和操作界面上的 list 控件，我们在底层代码中只需要对 m_list 这个对象进行操作即可。

4. 回到“解决方案资源管理器”，展开“源文件”，点击“***DemoDlg.cpp”。找到 OnInitDialog()函数。在数据库初始化语句的下面添加代码对 list 控件进行初始化。

```
**************************************************************************
    //list 控件初始化
    m_list.InsertColumn(0, _T("学号"), LVCFMT_LEFT, 70);
    m_list.InsertColumn(1, _T("姓名"), LVCFMT_LEFT, 70);
    m_list.InsertColumn(2, _T("性别"), LVCFMT_LEFT, 70);
    m_list.InsertColumn(3, _T("年龄"), LVCFMT_LEFT, 70);
    m_list.InsertColumn(4, _T("所在系"), LVCFMT_LEFT, 70);
    m_list.SetExtendedStyle(LVS_EX_FULLROWSELECT | LVS_EX_GRIDLINES);//
选择节点的时候选中整行
    **************************************************************************
```

运行效果如下：



四． 数据库中表格数据的加载

1. 回到“类视图”，展开总类“***Demo”，选择“***DemoDlg”类，点击鼠标右键，选择“添加”→添加函数。在“返回类型”中选择“void”，表示不返回任何值；在“函数名”下面输入：ShowData。点击“完成”。

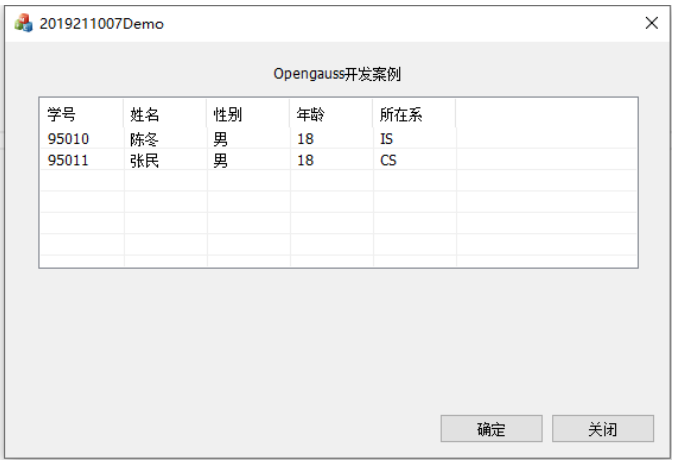
这时在“***DemoDlg.cpp”中会自动添加函数体 ShowData()，在“***DemoDlg.h”中会自动添加函数声明。

在 ShowData ()函数体中加入代码，加载表格数据。

```
**********************************************************************
        m_list.DeleteAllItems();//清空原有数据

        CRecordset my_set(&my_db);//申明CRecordset类的对象，用于调取表格数据

        if (my_set.Open(AFX_DB_USE_DEFAULT_TYPE, _T("select * from student")))//获取数据
        {
                //AfxMessageBox(_T("读取数据集成功"));
        }
        else
        {
                AfxMessageBox(_T("读取数据集失败！请检查连接是否异常！"));
                return;
        }

        if (my_set.IsBOF())//判断数据集是否为空
        {
                AfxMessageBox(_T("该表格数据集为空"));
                return;
        }

        int i = 0;//表示记录的行号，注意从第0行开始往list控件插入数据

        //5个字符串变量，用于保存一条记录的各个属性
        CString str_sno = _T("");
        CString str_sname = _T("");
```

```
CString str_ssex = _T("");
CString str_sage = _T("");
CString str_sdept = _T("");

while (!my_set.IsEOF()){//判断当前是否有数据可供读取

        my_set.GetFieldValue((short) 0, str_sno);//取第1个属性
        m_list.InsertItem(i, str_sno);

        my_set.GetFieldValue((short)1, str_sname);//取第2个属性
        m_list.SetItemText(i, 1, str_sname);

        my_set.GetFieldValue((short)2, str_ssex);//取第3个属性
        m_list.SetItemText(i, 2, str_ssex);

        my_set.GetFieldValue((short)3, str_sage);//取第4个属性
        m_list.SetItemText(i, 3, str_sage);

        my_set.GetFieldValue((short)4, str_sdept);//取第5个属性
        m_list.SetItemText(i, 4, str_sdept);

        my_set.MoveNext();//数据指针下移一行
        i++;

}

//释放当前资源
my_set.Close();
```

****



2. 回到 OnInitDialog()函数。在 list 控件初始化代码的下面调用 ShowData 函数。

```
**********
 ShowData();//显示表格数据
**********
```

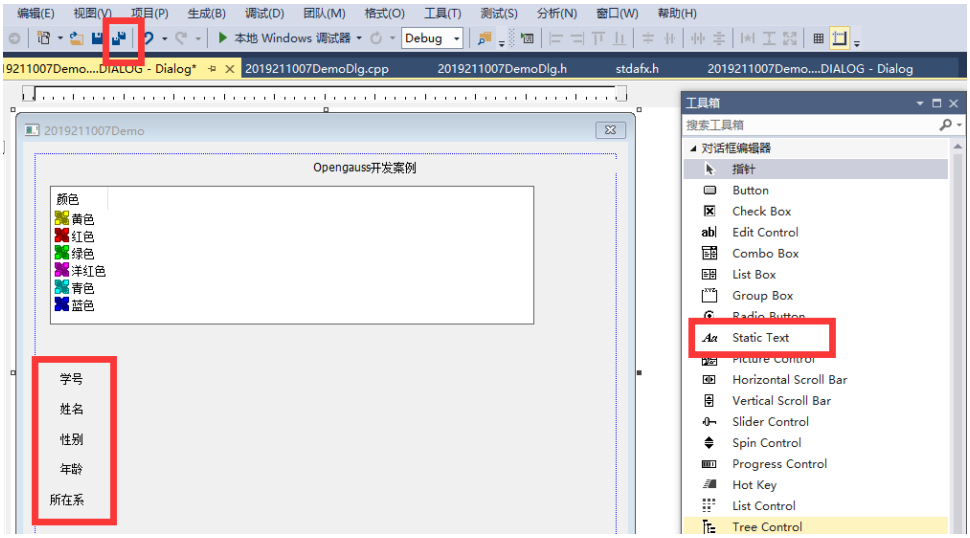
VS2013+OpenGauss 开发教程（三）

合肥工业大学 计算机与信息学院 张国富

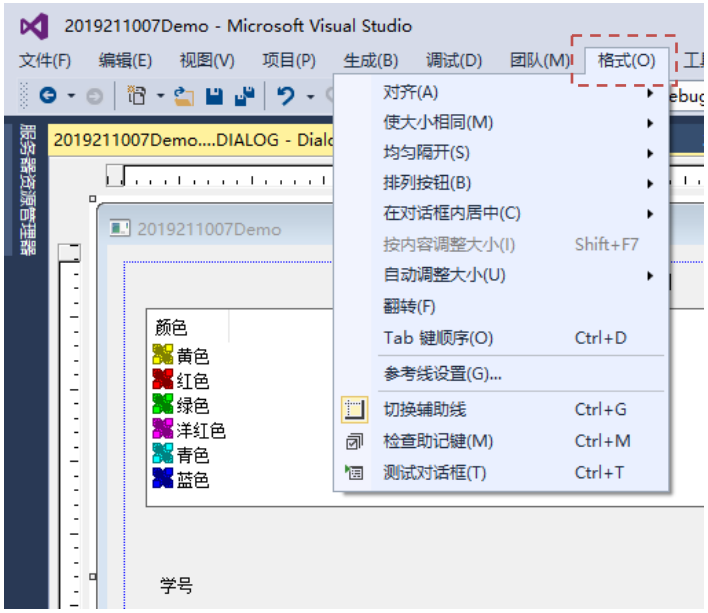
zgf@hfut.edu.cn

一．界面控件的绘制

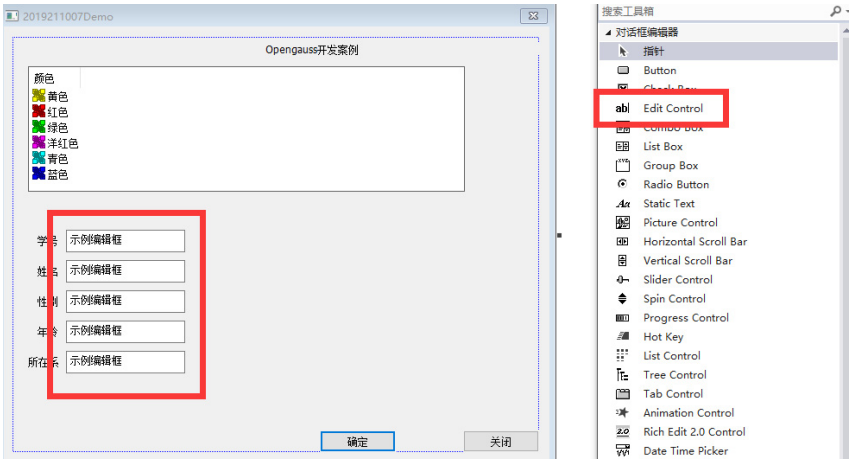
1. 选择右手边中间的“资源视图”，展开“Dialog”，选择“***DEMO_DIALOG”调出界面。打开“工具箱”，从中选择“Static Text”，回到界面区域，在适当位置绘制这个控件，将其 Caption 值修改为“学号”，Align Text 改为“right”（右对齐）；再从工具箱选择这个控件，依次绘制“姓名”、“年龄”、“性别”、“所在系”。完成后点击上面的“全部保存”。



全部选中这 5 个 Static Text 控件，通过“格式”，“按内容调整大小”，“对齐->右对齐“，“均匀隔开->纵向”，使其美观。



2. 从工具箱选择“Edit Control”控件，
在“学号”后面绘制该控件，并将其 ID 号修改为“IDC_EDIT_SNO”，选中这个控件，鼠标右键，“添加变量”，变量名取 m_edtsno；
在“姓名”后面绘制该控件，并将其 ID 号修改为“IDC_EDIT_SNAME”，选中这个控件，鼠标右键，“添加变量”，变量名取 m_edtsname；
在“性别”后面绘制该控件，并将其 ID 号修改为“IDC_EDIT_SSEX”，选中这个控件，鼠标右键，“添加变量”，变量名取 m_edtssex；
在“年龄”后面绘制该控件，并将其 ID 号修改为“IDC_EDIT_SAGE”，选中这个控件，鼠标右键，“添加变量”，变量名取 m_edtsage；
在“所在系”后面绘制该控件，并将其 ID 号修改为“IDC_EDIT_SDEPT”，选中这个控件，鼠标右键，“添加变量”，变量名取 m_edtsdept；
完成后点击上面的“全部保存”。



通过“格式”，“使大小相同->两者”，“对齐->左对齐”，“均匀隔开->纵向”，调整至合理位置。

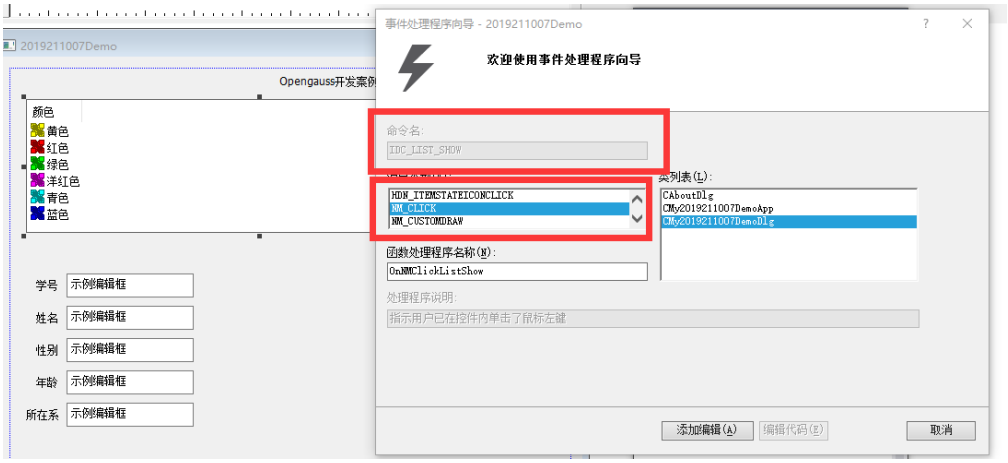
3. 切换到“解决方案资源管理器”，在头文件“*DemoDlg.h”中，会看到声明的 Edit 控件变量，在源文件“*DemoDlg.cpp” **DoDataExchange** 函数中，会看到这些变量和界面上控件的映射关系。

```
// 实现
protected:
    HICON m_hIcon;

    // 生成的消息映射函数
    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    DECLARE_MESSAGE_MAP()
public:
    CDatabase my_db;
    afx_msg void OnBnClickedCancel();
    CListCtrl m_list;
    CEdit m_edtsno;
    CEdit m_edtsname;
    CEdit m_edtssex;
    CEdit m_edtsage;
    CEdit m_edtsdept;
    afx_msg void OnNMClickListShow(NMHDR *pNMHDR, LRESULT *pResult);
};
```

```
void CMy2019211007DemoDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_LIST_SHOW, m_list);
    DDX_Control(pDX, IDC_EDIT_SNO, m_edtsno);
    DDX_Control(pDX, IDC_EDIT_SNAME, m_edtsname);
    DDX_Control(pDX, IDC_EDIT_SSEX, m_edtssex);
    DDX_Control(pDX, IDC_EDIT_SAGE, m_edtsage);
    DDX_Control(pDX, IDC_EDIT_SDEPT, m_edtsdept);
}
```

二．实现表格中选中行的数据显示

1. 回到“资源视图”，展开“Dialog”，选择“***DEMO_DIALOG”调出界面。选中 list 控件，鼠标右键，“添加事件处理程序”，在“消息类型”中选择“NM_CLICK”，点击“添加编辑”。



2. 切换到“解决方案资源管理器”，在头文件“*DemoDlg.h”中，会看到声明的函数，在源文件“*DemoDlg.cpp” BEGIN_MESSAGE_MAP 函数中，会看到这个函数与界面上控件的映射关系和映射类型。

```
public:
    CDatabase my_db;
    afx_msg void OnBnClickedCancel();
    CListCtrl m_list;
    void ShowData();
    CEdit m_edtsno;
    CEdit m_edtsname;
    CEdit m_edtssex;
    CEdit m_edtsage;
    CEdit m_edtsdept;
    afx_msg void OnNMClickListShow(NMHDR *pNMHDR, LRESULT *pResult);
};
```

```
BEGIN_MESSAGE_MAP(CMy2019211007DemoDlg, CDialogEx)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDCANCEL, &CMy2019211007DemoDlg::OnBnClickedCancel)
    ON_NOTIFY(NM_CLICK, IDC_LIST_SHOW, &CMy2019211007DemoDlg::OnNMClickListShow)
END_MESSAGE_MAP()
```

3. 在“*DemoDlg.cpp”中最下面的 OnNMClickListShow()函数体中，

   // TODO: 在此添加控件通知处理程序代码

   在上述语句的下面添加代码。

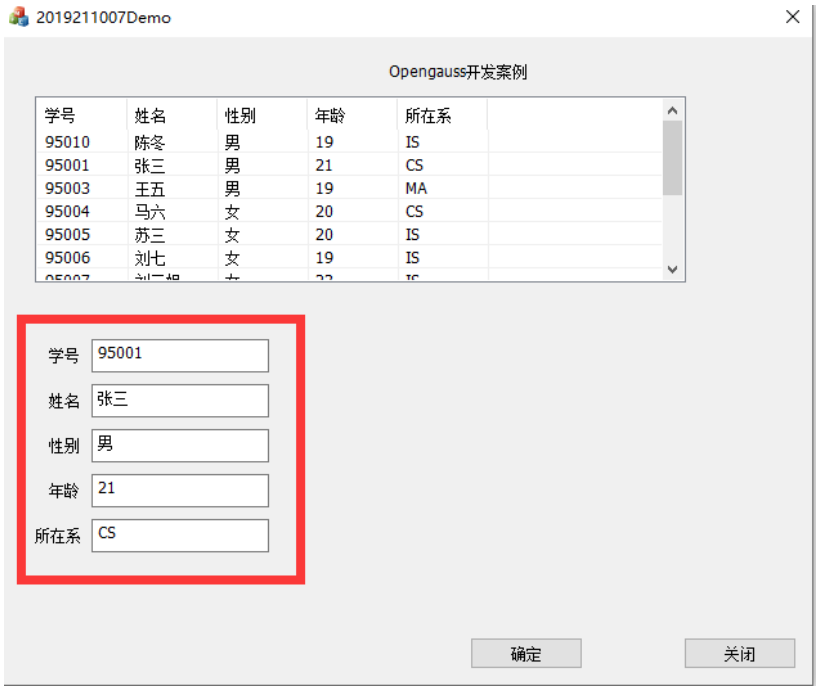
   ****

```
CString strsno, strsname, strssex, strsage, strsdept;//用于保存从list控件上获取的值
if (pNMItemActivate->iItem != -1)//pNMItemActivate->iItem为鼠标焦点处对应的行号
{
    strsno = m_list.GetItemText(pNMItemActivate->iItem, 0);
    m_edtsno.SetWindowText(strsno);

    strsname = m_list.GetItemText(pNMItemActivate->iItem, 1);
    m_edtsname.SetWindowText(strsname);
```

```
    strssex = m_list.GetItemText(pNMItemActivate->iItem, 2);
    m_edtssex.SetWindowText(strssex);

    strsage = m_list.GetItemText(pNMItemActivate->iItem, 3);
    m_edtsage.SetWindowText(strsage);

    strsdept = m_list.GetItemText(pNMItemActivate->iItem, 4);
    m_edtsdept.SetWindowText(strsdept);

    UpdateData(FALSE);//刷新界面将底层代码的数据传递给上层的界面

}
```

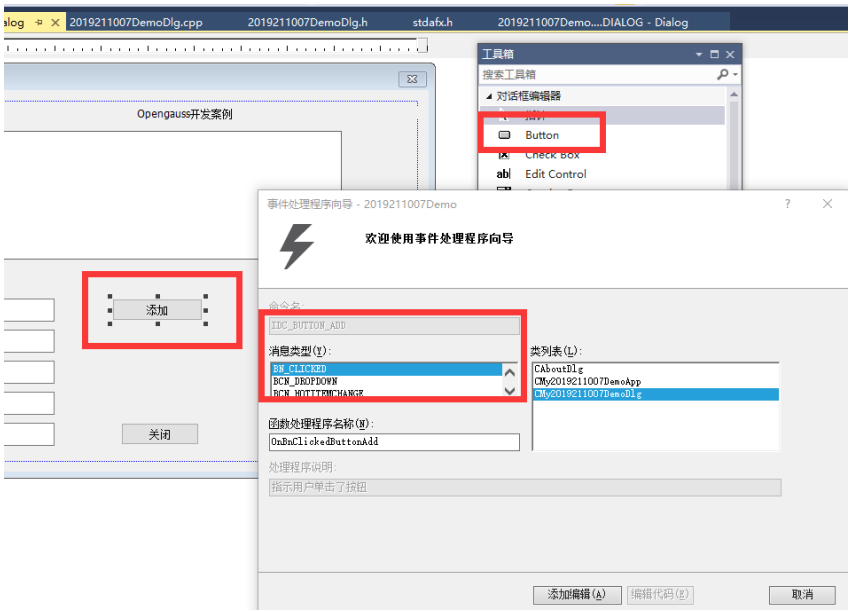
****

执行效果如下：



三．实现数据的添加

1. 回到“资源视图”，展开“Dialog”，选择“***DEMO_DIALOG”调出界面。打开“工具箱”，从中选择“Button”，回到界面区域，在适当位置绘制这个控件，将其 Caption 值修改为“添加”，ID 改为“IDC_BUTTON_ADD”。选中这个控件，鼠标右键，选择“添加事件处理程序”，按照默认的参数即可，直接点“添加编辑”。



2. 切换到“解决方案资源管理器”，在头文件“*DemoDlg.h”中，会看到声明的函数，在源文件“*DemoDlg.cpp” BEGIN_MESSAGE_MAP 函数中，会看到这个函数与界面上控件的映射关系和映射类型。

```
public:
    CDatabase my_db;
    afx_msg void OnBnClickedCancel();
    CListCtrl m_list;
    void ShowData();
    CEdit m_edtsno;
    CEdit m_edtsname;
    CEdit m_edtssex;
    CEdit m_edtsage;
    CEdit m_edtsdept;
    afx_msg void OnNMClickList1Show(NMHDR *pNMHDR, LRESULT *pResult);
    afx_msg void OnBnClickedButtonAdd();
};
```

```
BEGIN_MESSAGE_MAP(CMy2019211007DemoDlg, CDialogEx)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDCANCEL, &CMy2019211007DemoDlg::OnBnClickedCancel)
    ON_NOTIFY(NM_CLICK, IDC_LIST1_SHOW, &CMy2019211007DemoDlg::OnNMClickList1Show)
    ON_BN_CLICKED(IDC_BUTTON_ADD, &CMy2019211007DemoDlg::OnBnClickedButtonAdd)
END_MESSAGE_MAP()
```

3. 在“*DemoDlg.cpp”中最下面的 OnBnClickedButtonAdd()函数中，

// TODO: 在此添加控件通知处理程序代码

在上述语句下面添加代码。

```
//**********************************************************************

UpdateData(TRUE);//刷新界面，把数据传递给底层代码

CString strsno, strsname, strssex, strsage, strsdept;//用于保存从edit控件上获取的值

//从界面上的edit控件获取用户输入的值
m_edtsno.GetWindowText(strsno);
m_edtsname.GetWindowText(strsname);
m_edtssex.GetWindowText(strssex);
m_edtsage.GetWindowText(strsage);
m_edtsdept.GetWindowText(strsdept);

if (strsno == _T(""))
{
	AfxMessageBox(_T("学号为主码，取值不能为空！"));
	return;//等待用户重新输入
}
else
{
	if (strsname == _T(""))
		strsname = _T("NULL");
	else
		strsname = _T("'") + strsname + _T("'");

	if (!strssex.CompareNoCase(_T("男")) || !strssex.CompareNoCase(_T("女")))
	{
		strssex = _T("'") + strssex + _T("'");
	}
	else{
		AfxMessageBox(_T("性别只能输入“男”或“女”！"));
		return; return;//等待用户重新输入
	}

	if (strsage == _T(""))
		strsage = _T("NULL");

	if (strsdept == _T(""))
		strsdept = _T("NULL");
	else
		strsdept = _T("'") + strsdept + _T("'");
```

```
        CString mstrsql;
        mstrsql.Format(_T("insert into student values('%s',%s,%s,%s,%s)"), strsno, strsname, strssex,
strsage, strsdept);
        //AfxMessageBox(mstrsql);//测试生成的sql语言，如果添加数据失败，可打开这个语句查看
SQL代码是否符合语法

        try
        {
            my_db.ExecuteSQL(mstrsql);
        }
        catch (CDBException* pe)
        {
            //如果有异常发生，弹出错误消息框，帮助纠正bug
            pe->ReportError();
            pe->Delete();
        }

        ShowData();//加载数据
    }
```

****

点击“添加”按钮，顺利添加数据到OpenGauss中。

注意：CDatabase类的成员函数ExecuteSQL()只负责执行输入的SQL代码，不返回任何数据集，只适用于不需要返回任何数据的SQL语句的执行，简单方便。如果需要检查SQL代码的执行结果是否有数据，或者进行数据匹配，不能使用此函数，应该使用CRecordset的对象，可参考ShowData()函数。

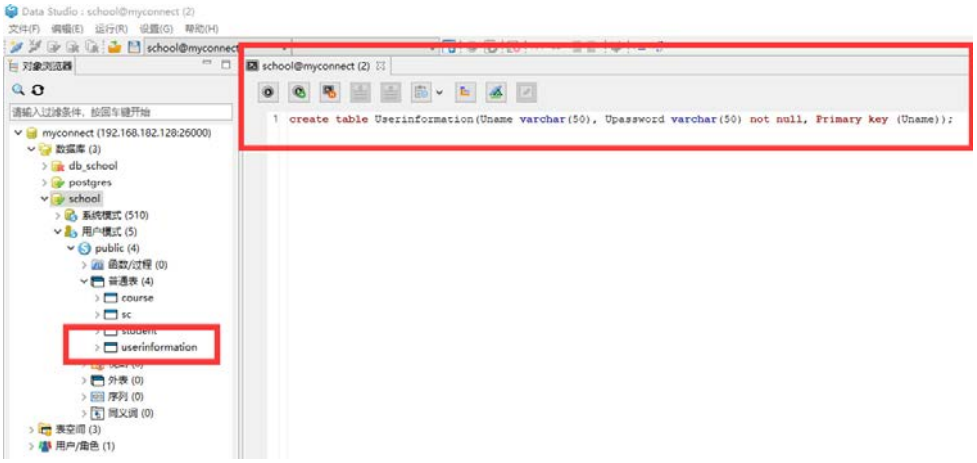
VS2013+OpenGauss 开发教程（四）

合肥工业大学 计算机与信息学院 张国富
zgf@hfut.edu.cn

一．用户表格的创建

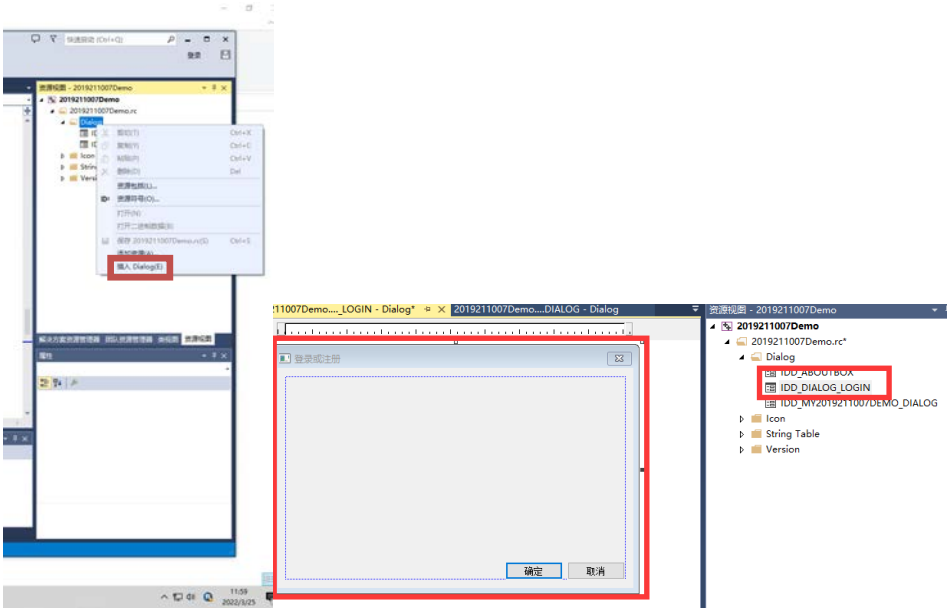
打开 Opengauss 虚拟机，启动 opengauss 数据库，以管理员身份运行 Data Studio，连接上 Opengauss 虚拟机，连接上 school 数据库，选中 school 数据库，鼠标右键，打开新的终端，输入如下 SQL 语言创建：用户表。

```
create table Userinformation(Uname varchar(50), Upassword varchar(50) not null, Primary key (Uname));
```

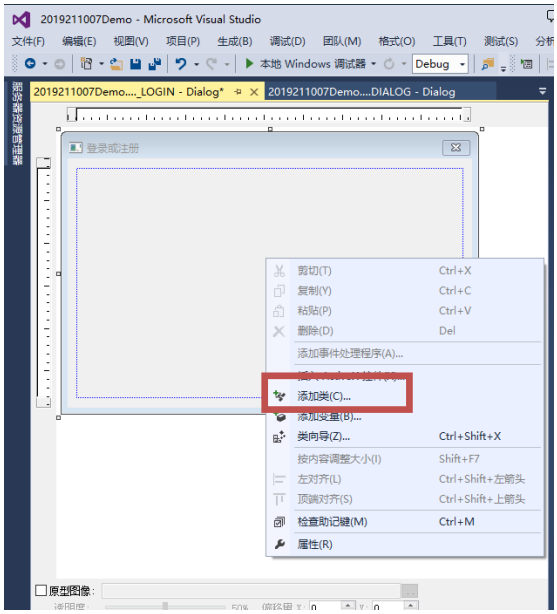


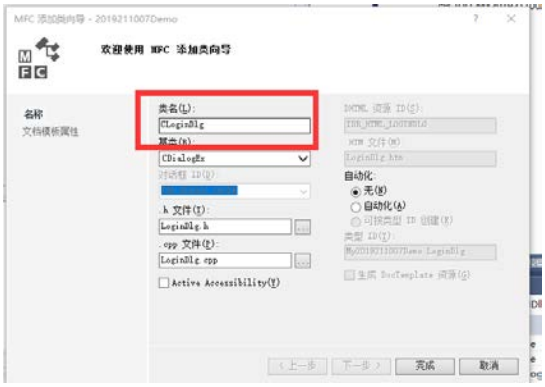
二．登录对话框的创建

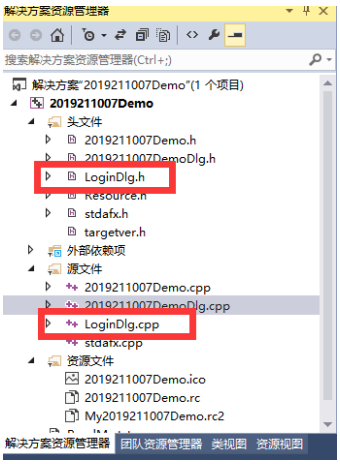
1. 打开教程前面创建的“*Demo”工程（**记得把这个工程做个备份**），点击中间的“资源视图”，展开“*Demo.rc”，选中“Dialog”，点击鼠标右键，选择“插入 Dialog(E)”，对新插入的对话框，修改属性值，将 ID 改为“IDD_DIALOG_LOGIN”，将 Caption 改为“登陆或注册”。



2. 在界面编辑区选中刚插入的对话框，点击鼠标右键，选择“添加类”，在“类名”中输入“CLoginDlg”，编译器将自动在工程中添加“LoginDlg.h”和“LoginDlg.cpp”，可在“解决方案资源管理器”中查看。后面要想控制这个对话框，只需要声明这个类的一个具体对象即可。



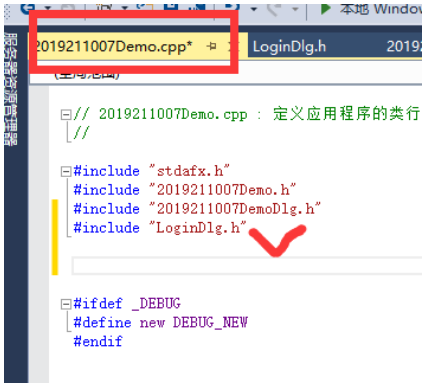




3. 转到“解决方案资源管理器”，在“源文件”中选择“*Demo.cpp”文件。在

#include "2019211007DemoDlg.h"

这个语句下面添加“#include "LoginDlg.h""将 CLoginDlg 类引入到工程中。



在“*Demo.cpp”中找到 InitInstance()函数，在该函数中找到初始化“*DemoDlg”实例的代码，即“CMy2019211007DemoDlg dlg;”。因为，我们要在通过认证之后才能出现这个 CMy2019211007DemoDlg 对话框，所以先进行 CLoginDlg 的初始化，调出 LoginDlg 对话框，当用户在这个对话框中点击“IDOK”这个控件后（点击后会撤销 LoginDlg 对话框），再进行 CMy2019211007DemoDlg 的初始化，调出 CMy2019211007DemoDlg 对话框。

因此，在

SetRegistryKey(_T("应用程序向导生成的本地应用程序"));

语句的下面添加代码将 CMy2019211007DemoDlg 的初始化约束起来。

****

CLoginDlg LoginDlg;//声明CLoginDlg这个类的一个对象

if (*IDOK* == LoginDlg.*DoModal*())//显示一个模态对话框，此时DoModal()函数并不返回，即并没有执行大括号内的内容，直到用户点击了确定按钮，那么DoModal()函数就返回IDOK，此时进入if语句

{

CMy2019211007DemoDlg dlg;

……..

else if (nResponse == -1)

{

TRACE(*traceAppMsg*, 0, "警告：对话框创建失败，应用程序将意外终止。\n");

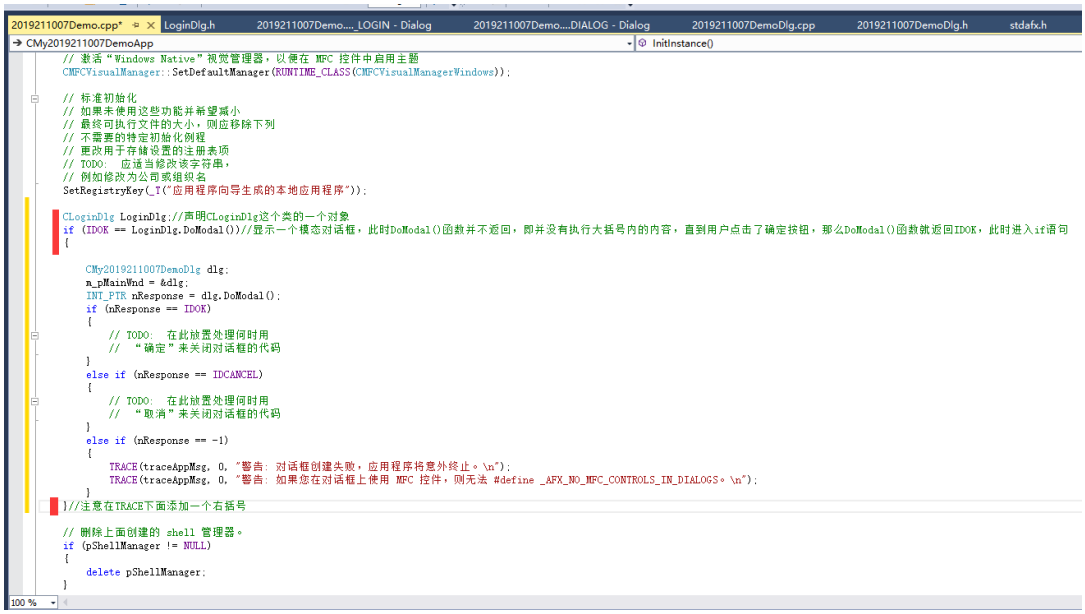
TRACE(*traceAppMsg*, 0, "警告：如果您在对话框上使用 MFC 控件，则无法 #define _AFX_NO_MFC_CONTROLS_IN_DIALOGS。\n");

}

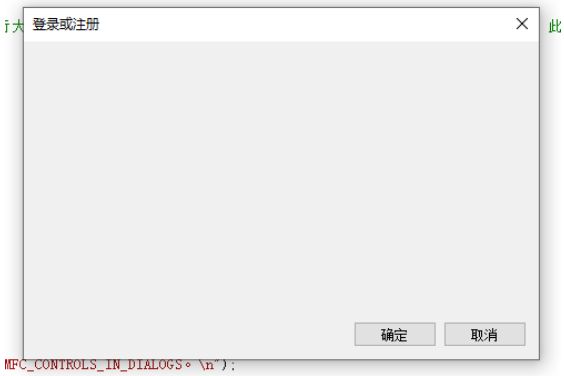
} //注意在 TRACE 下面添加一个右括号

****

注意：OnInitDialog()只是在 Dialog 初始化时调用，Dialog 相关子窗口就可以放在这里初始化。InitInstance()是初始化实例，在生成的一个新的实例的时候，完成一些初始化的工作。其会在 OnInitDialog()之前运行。

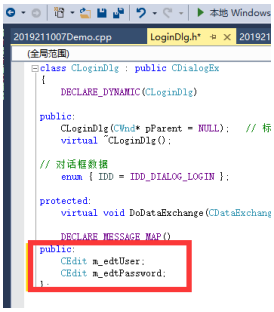


编译运行工程，首先弹出登录对话框，而不是原先的数据库操作界面。



三．登录界面的绘制

1. 回到“资源视图”，展开“*Demo.rc”，选中“Dialog”，点击“IDD_DIALOG_LOGIN”调出 CLoginDlg 对话框界面，在“工具箱”中选中 Static Text 控件，在界面上画两个静态文本控件，分别把 Caption 值修改为“用户名：”和“密码：”，Align Text 设为 Right。再选择“工具箱”中的 Edit Control，在“用户名：”后面绘制 edit 控件，将其 ID 号改为 IDC_EDIT_USER，选中这个控件，鼠标右键，添加变量，输入变量名“m_edtUser”。继续在“密码：”后面绘制 edit 控件，将其 ID 号改为 IDC_EDIT_PASSWORD，同时将其属性“行为”中的“Password”值改为“True”，选中这个控件，鼠标右键，添加变量，输入变量名“m_edtPassword”。这时，在"LoginDlg.h"和"LoginDlg.cpp"中会出现相应的变量声明和控件绑定。



2. 回到“LoginDlg”对话框界面，点击“确定”按钮。将其属性 Caption 值改为“登录”，选中按钮，鼠标右键，添加事件处理程序，选择默认的“BN_CLICKED”，点击“添加编辑”，生成 OnBnClickedOk()函数。
   再点击“取消”按钮。将其属性 Caption 值改为“退出”，选中按钮，鼠标右键，添加事件处理程序，选择默认的“BN_CLICKED”，点击“添加编辑”，生成 OnBnClickedCancel()函数。

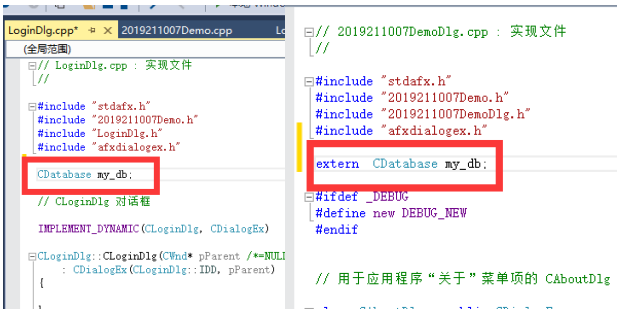
3. 回到“LoginDlg”对话框界面，打开工具栏，在界面上添加一个按钮控件，将其属性 Caption 值改为“注册”， 将其 ID 号改为 IDC_EDIT_PASSWORD，选中按钮，鼠标右键，添加事件处理程序，选择默认的“BN_CLICKED”，点击“添加编辑”，生成 OnBnClickedButtonRegister()函数。

   最后界面如下：

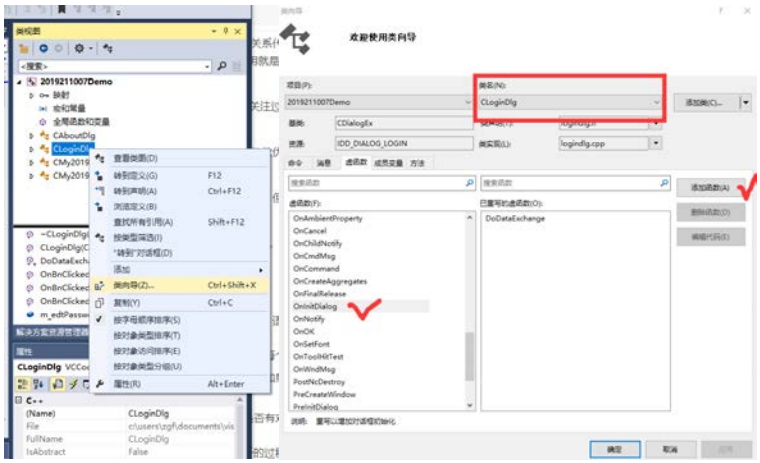
四．CDatabase 类的迁移

1. 回到“解决方案资源管理器”，点击“*DemoDlg.h”文件。将“CDatabase my_db;”剪切至“LoginDlg.cpp”的上面，作为全局变量。同时，在“*DemoDlg.cpp”文件上面输入“extern CDatabase my_db;”。

注意：extern 可以置于变量或者函数前，以标示变量或者函数的定义在别的文件中，提示编译器遇到此变量和函数时在其他模块中寻找其定义。



2. 回到“类视图”，选中“CLoginDlg”，点击鼠标右键，选择“类向导”，在“虚函数”中选中“OnInitDialog”，点击右手边的“添加函数”，确定。



将“*DemoDlg.cpp”文件中 OnInitDialog()函数内的关于 Opengauss 数据库的初始化代码剪切到“LoginDlg.cpp”的 OnInitDialog()函数中

// TODO: 在此添加额外的初始化

这一行的下面。让登录对话框来完成数据库的连接，因为它首先需要进行身份认证。

五．退出按钮的实现

在“LoginDlg.cpp”的 OnBnClickedCancel ()函数中，

// TODO: 在此添加控件通知处理程序代码

在这个语句的下面添加如下代码：

****

```
    if (my_db.IsOpen())//判断是否处于数据库连接状态
    {
        my_db.Close();//如果当前处于连接状态，从数据源断开连接，释放系统资源
    }
```

****

六．关于注册登录按钮的提示

注册按钮：只需要将SQL代码送给openGauss去执行，将用户名和密码输入到表格中，不需要返回数据。

登录按钮：不仅需要将SQL代码送给openGauss去执行，还需要openGauss返回执行的结果，以检查是否真的存在这个账号。

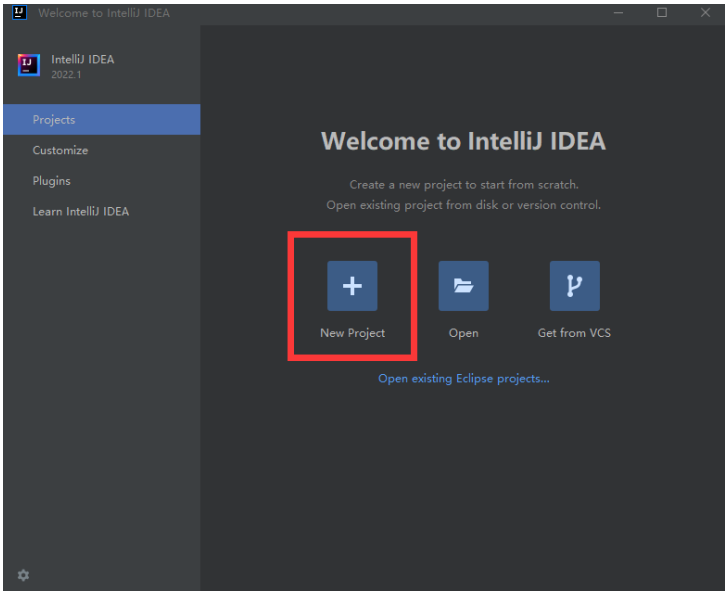
Java+OpenGauss 开发教程（一）

合肥工业大学 计算机与信息学院 张国富

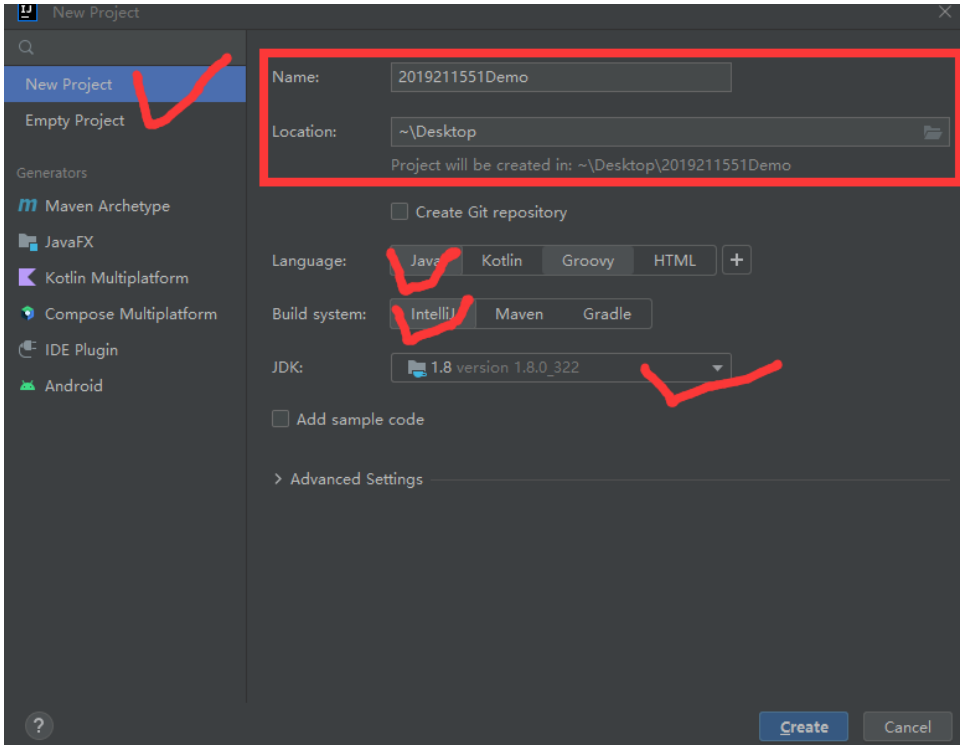
zgf@hfut.edu.cn

一、创建一个初始工程

1. 打开 Intellij IDEA Community 选择 New Project，进入工程创建向导。

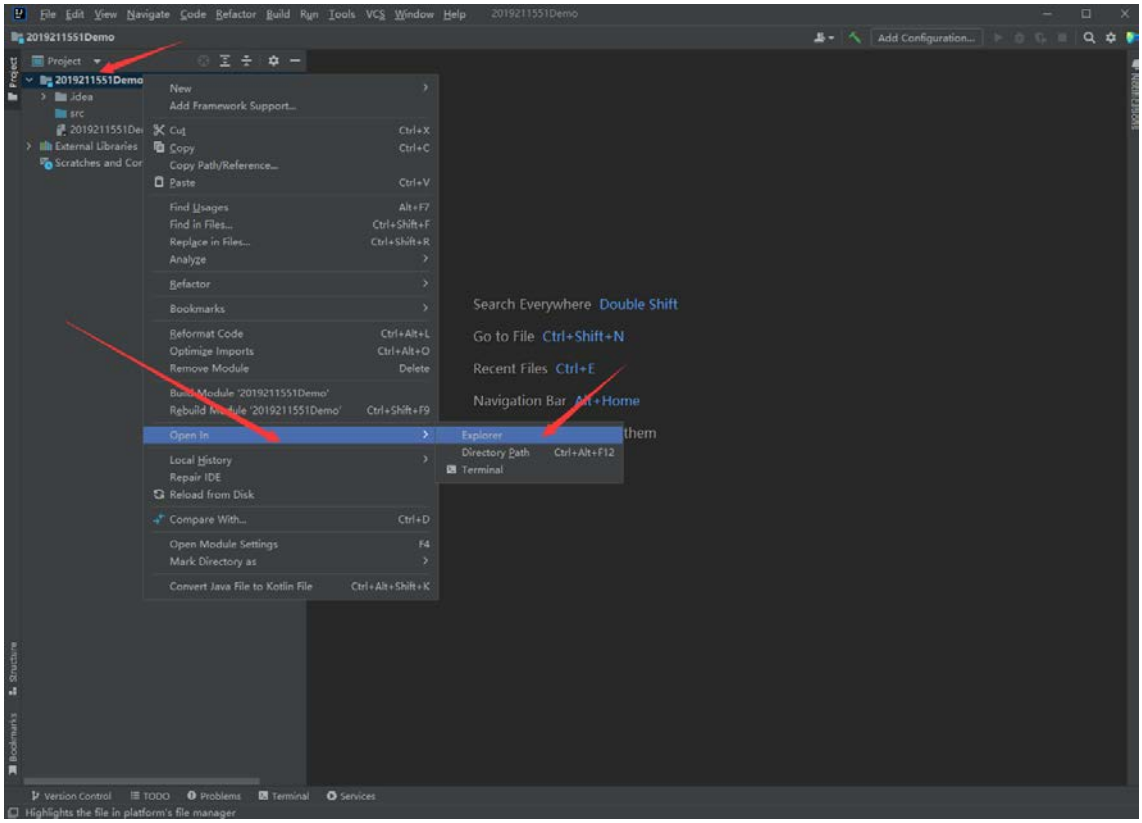


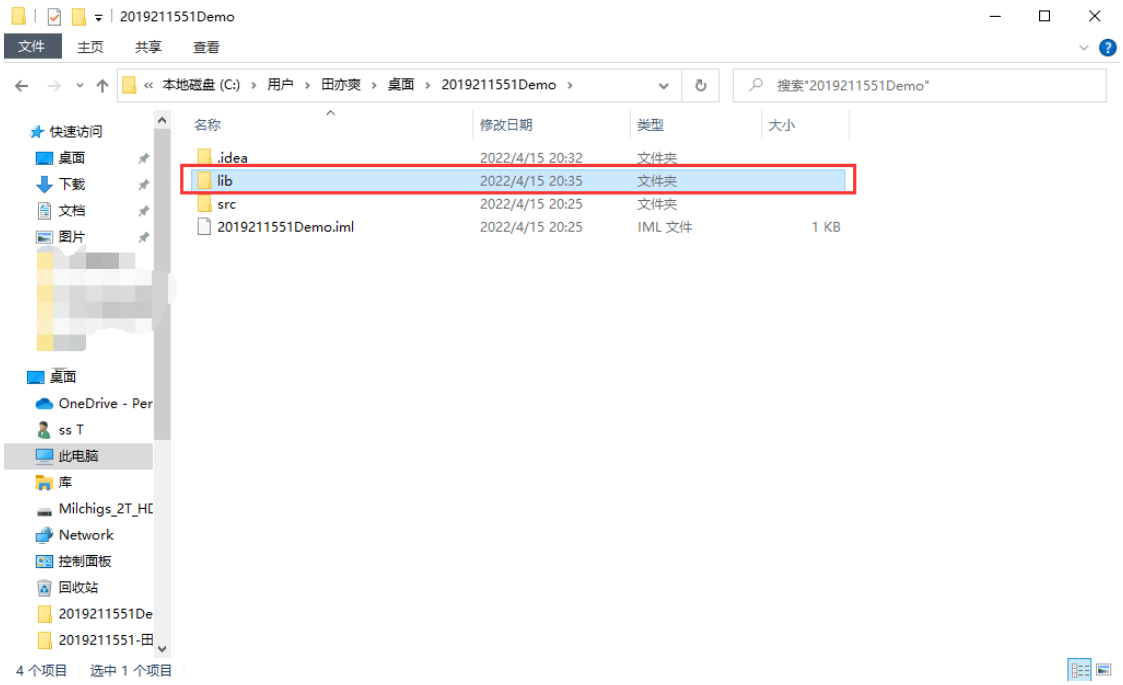
2. 在 New Project 向导中，左手边选择 New Project，右手边选择 Java，IntelliJ，JDK 选择安装好的 JDK1.8。“Name”中输入你的工程名，工程名要包含学号，例如 2019211007Demo。注意查看工程保存的“Location”，以便后期查找这个工程。



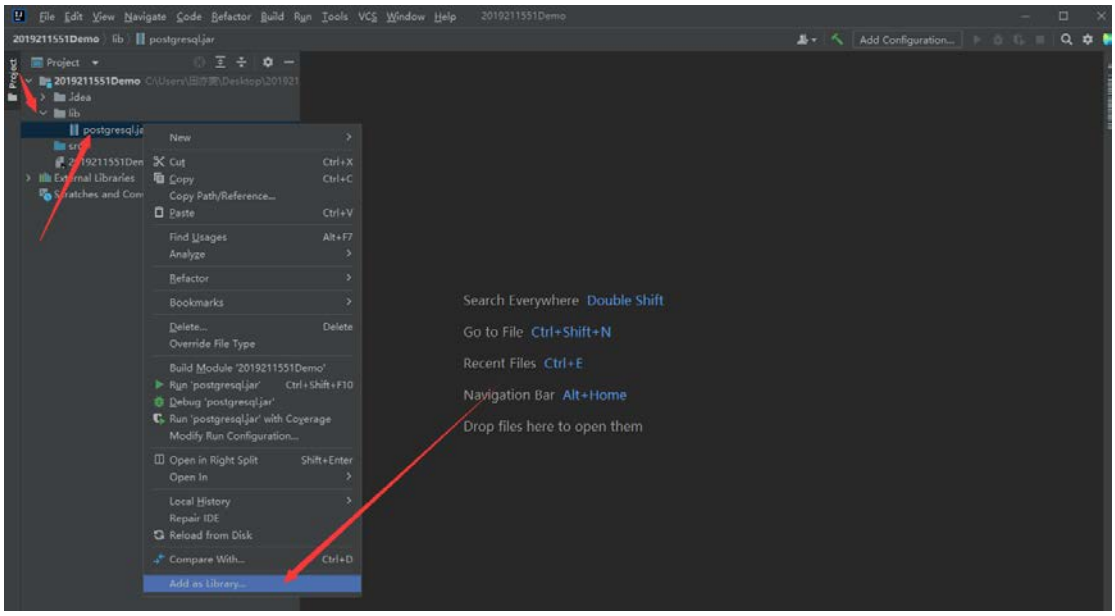
右下角点击 Create，完成工程创建。

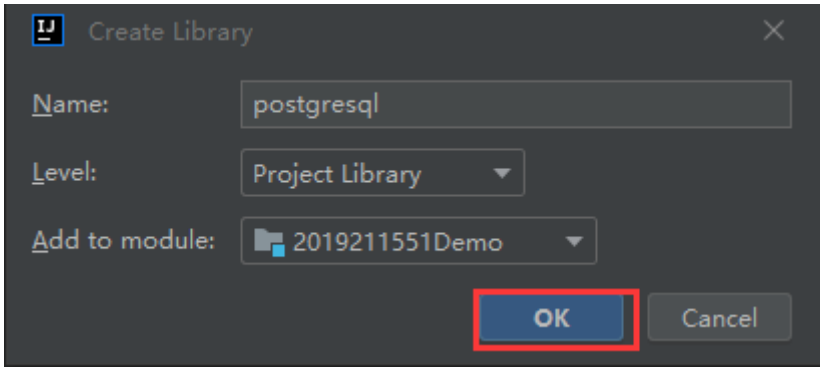
3. 左上角工程点击右键，找到 Open In 目录下的 Explorer，打开工程所在位置，进入工程目录，将所提供的 lib 外部库文件夹复制到工程目录中去。





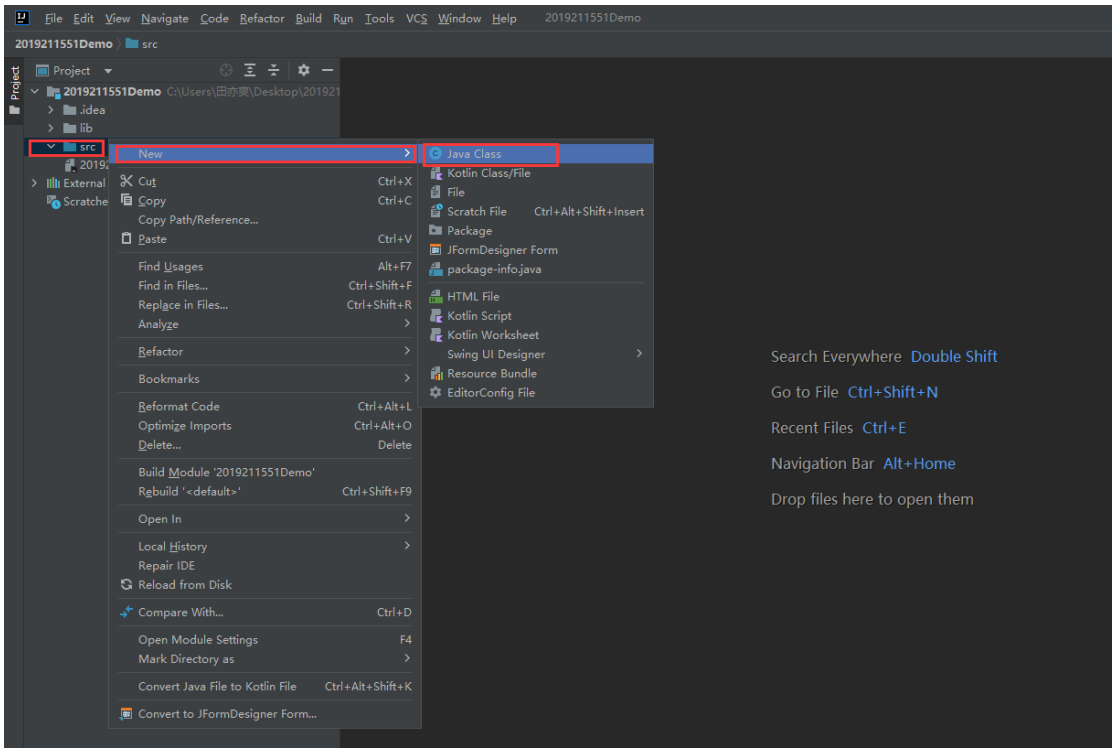
4. 左边找到 lib 文件加，展开文件夹找到 JDBC 驱动，右键点击 Add as Library，点击 OK 将外部库进行导入。这样就完成了一个 JDBC 工程的基本创建，我们就可以使用该外部库进行数据库的操作了。

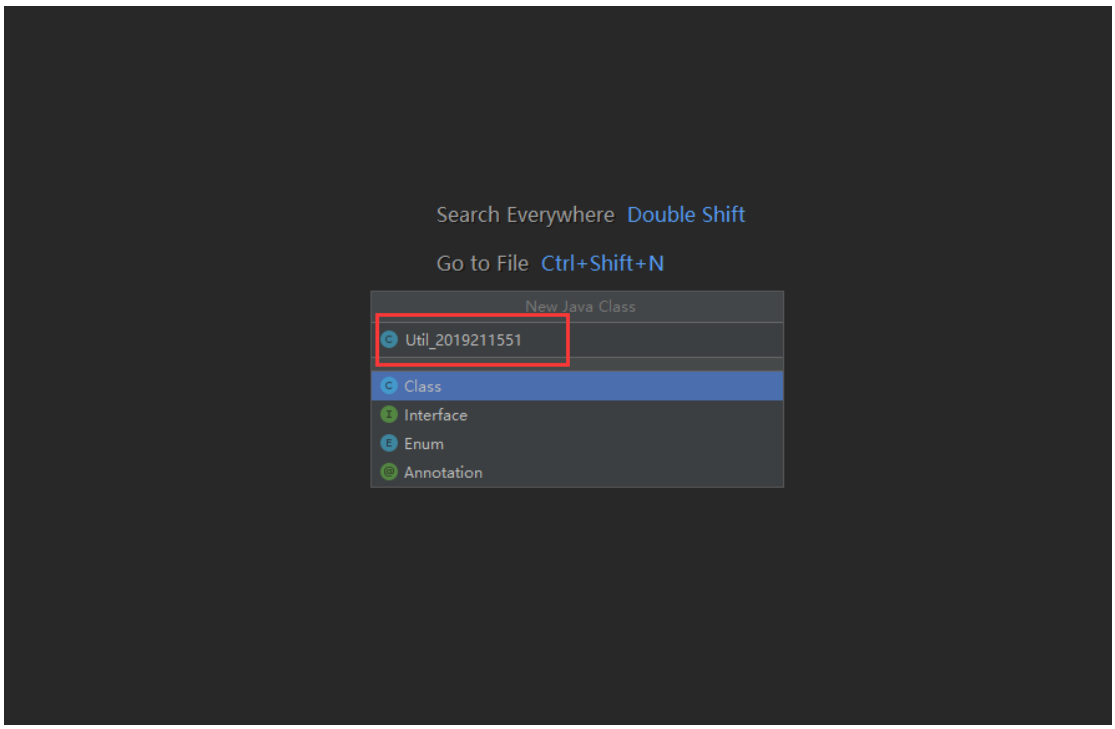




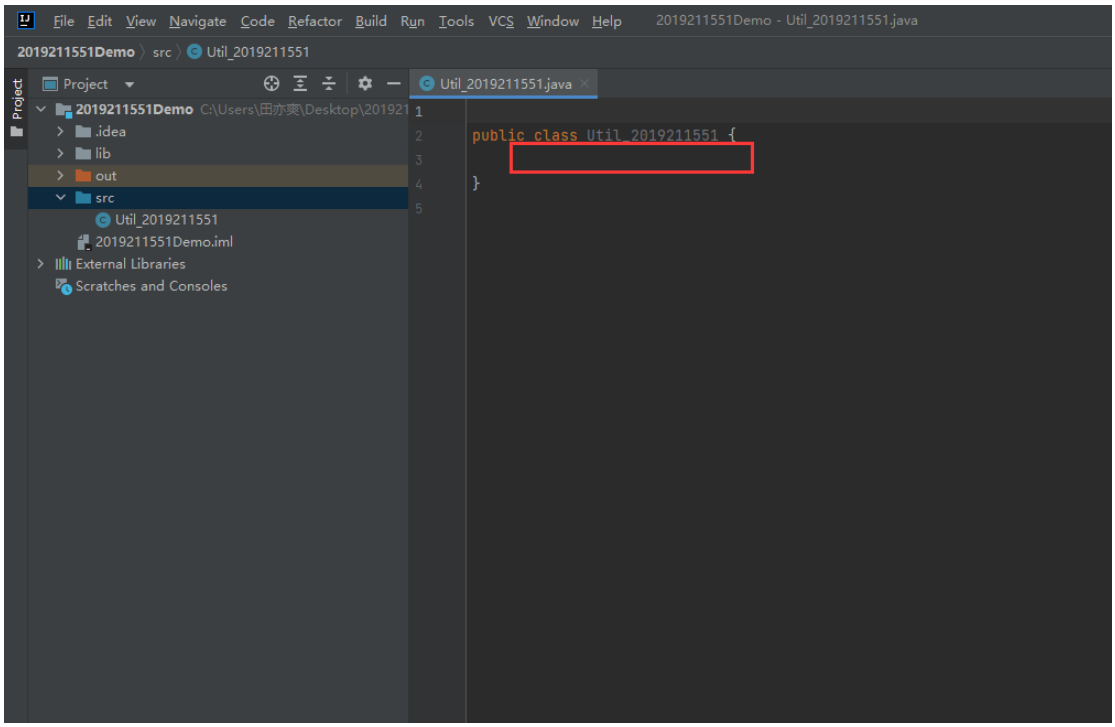
二、编写数据库工具类-链接数据库

1. 右键 src 点击“New”目录下的“Java Class”创建数据库连接工具类 Util，类名使用自己的学号。





2. 在 Util 工具类中添加如下代码。

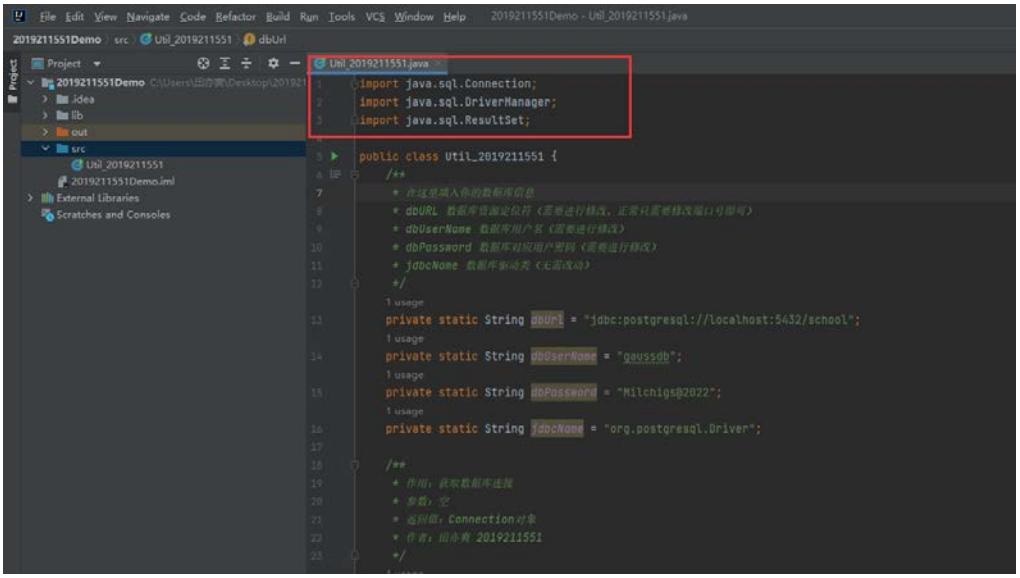



```
/**
 * 在这里填入你的数据库信息
 * dbURL 数据库资源定位符（需要进行修改，正常只需要修改端口号即可）
 * dbUserName 数据库用户名（需要进行修改）
 * dbPassword 数据库对应用户密码（需要进行修改）
 * jdbcName 数据库驱动类（无需改动）
 */
private static String dbUrl =
"jdbc:postgresql://localhost:5432/school";
private static String dbUserName = "gaussdb";
private static String dbPassword = "Milchigs@2022";
private static String jdbcName = "org.postgresql.Driver";

/**
 * 作用：获取数据库连接
 * 参数：空
 * 返回值：Connection 对象
 * 作者：田亦爽 2019211551
 */
public static Connection getConnection(){
    Connection connection = null;
    try {
        Class.forName(jdbcName);
        connection = DriverManager.getConnection(dbUrl, dbUserName,
dbPassword);
```

```
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println("数据库连接失败");
    }
    System.out.println("数据库连接成功");
    return connection;
}

/**
 * 作用：关闭数据库连接
 * 参数：Connection
 * 返回值：空
 * 作者：田亦爽 2019211551
 */
public static void closeConnection(Connection connection){
    if(connection != null) {
        try {
            connection.close();
        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("数据库连接资源释放失败");
        }
    }
}

/**
 * 作用：关闭数据库连接
 * 参数：Connection, ResultSet
 * 返回值：空
 * 作者：田亦爽 2019211551
 */
public static void closeConnection(Connection connection, ResultSet
resultSet){
    if(connection != null) {
        try {
            connection.close();
            resultSet.close();
        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("数据库连接资源释放失败");
        }
    }
}
```

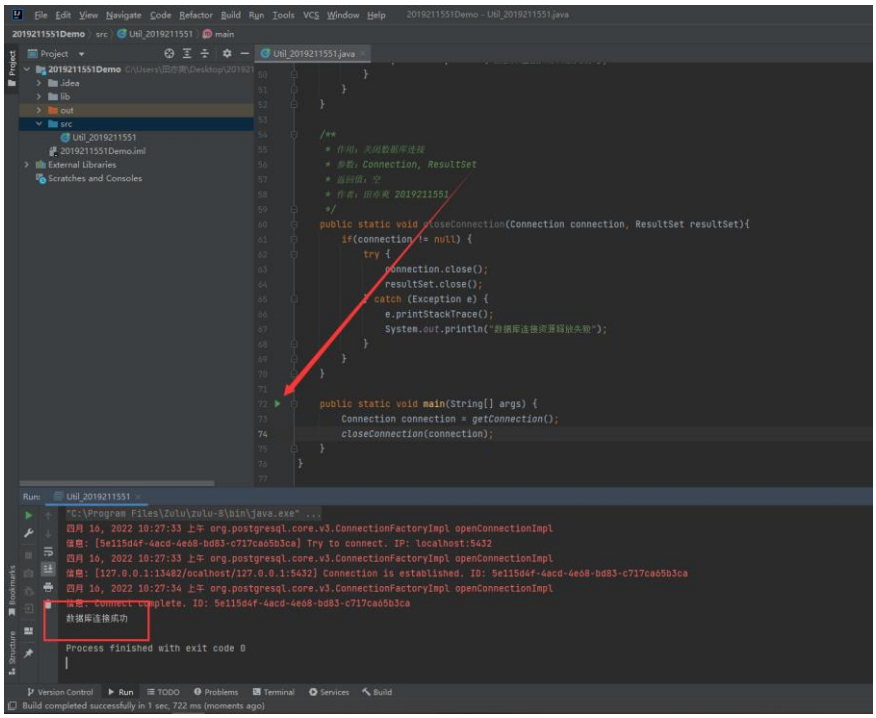
```
public static void main(String[] args) {
    Connection connection = getConnection();
    closeConnection(connection);
}
```

3. 引入对应的 Java 库。



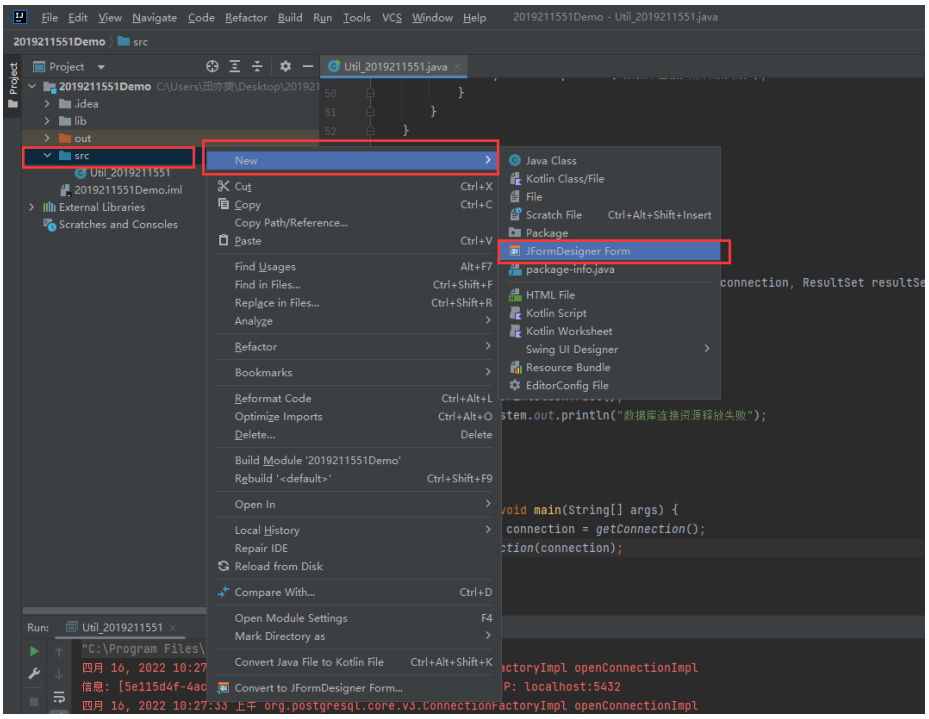
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
```

4. 进行数据库连接测试。找到下方的测试 main 方法，点击左边的绿色三角，进行数据库连接测试，终端出现“数据库连接成功”表明数据库成功连接，完成 Java 程序与数据库的通信。

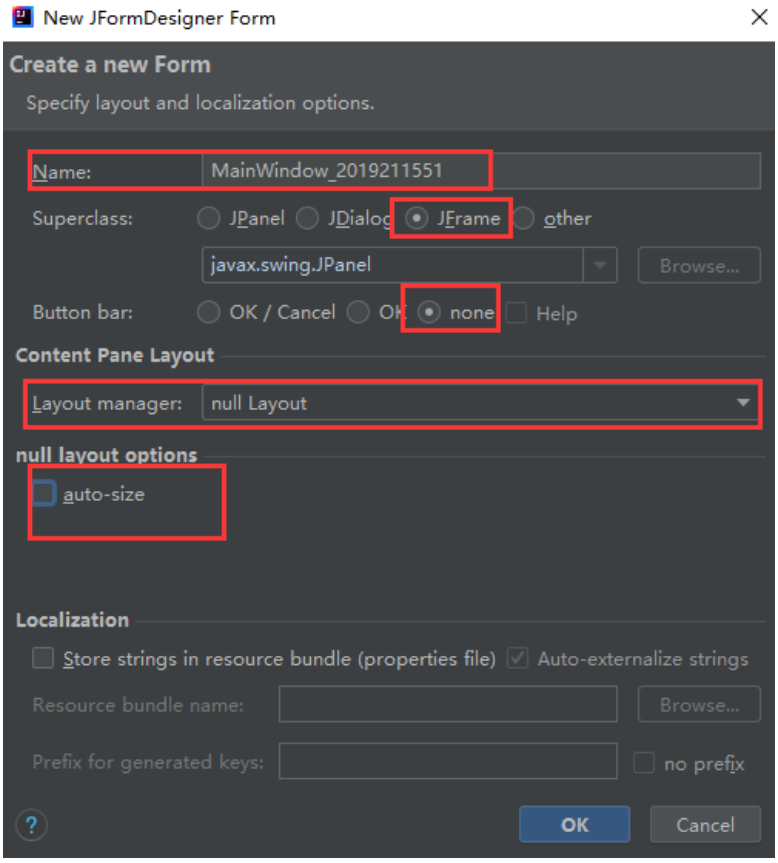


三、窗体 list 控件初始化

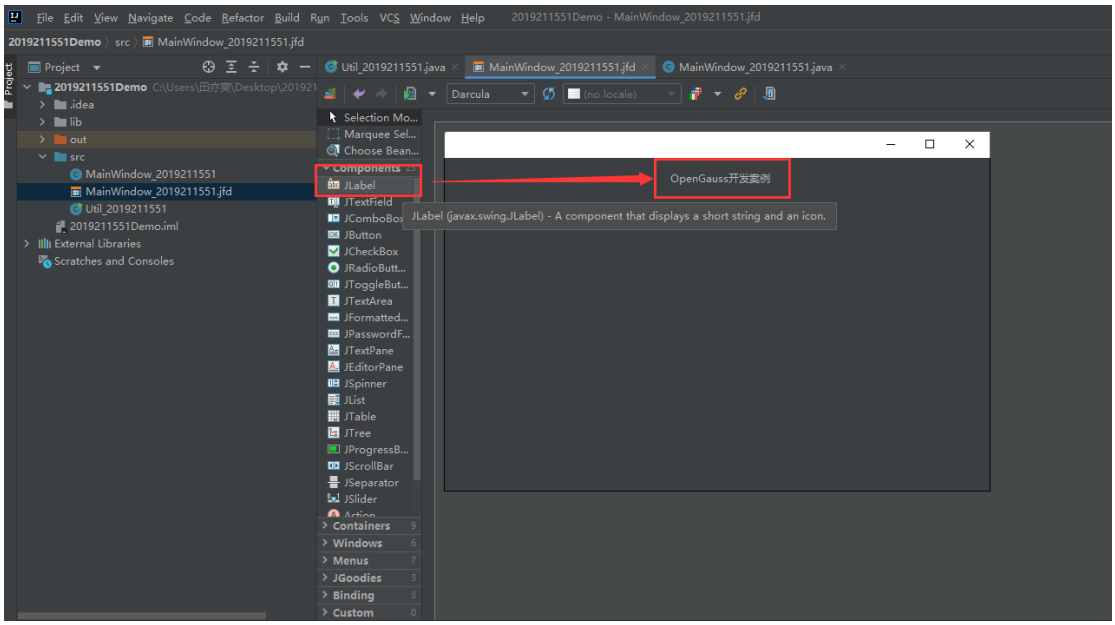
1. 找到左手边的 src 文件夹，点击右键，寻找到“New”目录下的“JFormDesigner Form”，创建一个窗体类。



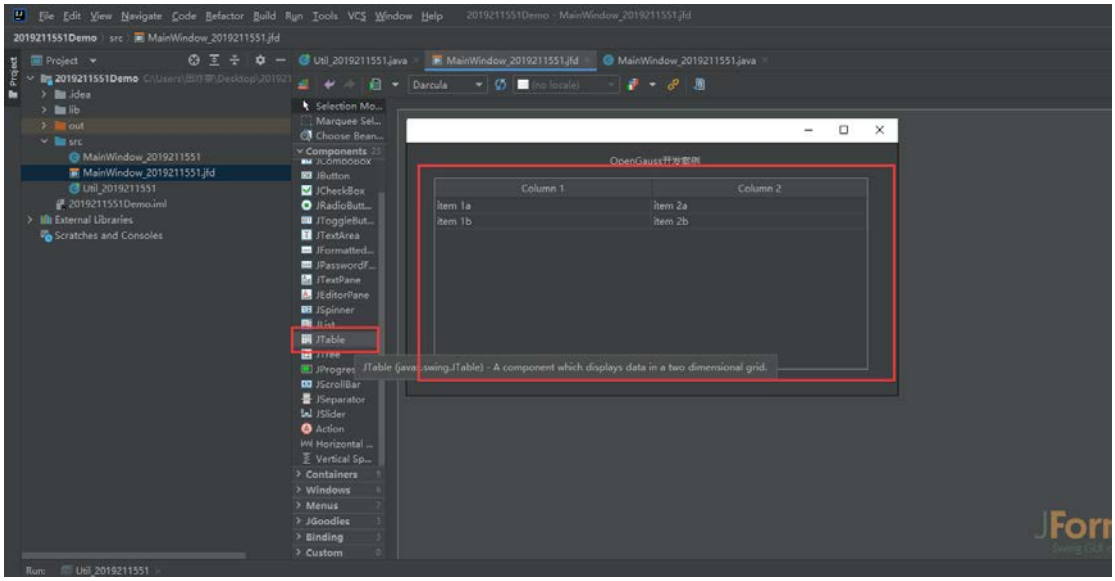
2. 将窗体类名字改成 MainWindow+你的学号，选择 JFrame，不需要 Button bar， 选择不需要 Layout，取消勾选 auto-size，最终点击“OK”完成窗体类创建。



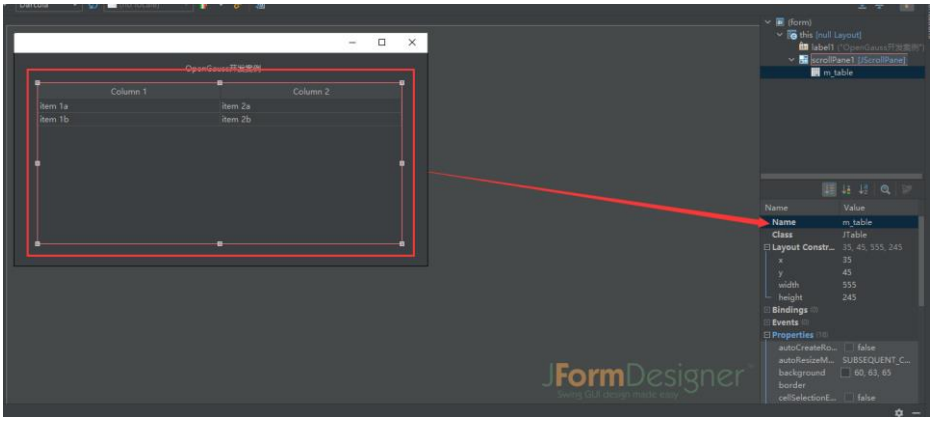
3. 在设计页面左边的“Components”中找到“JLabel”，点击“JLabel”将其放置在软件的最上面，并且双击控件，将里面的文字改为“OpenGauss 开发案例”。



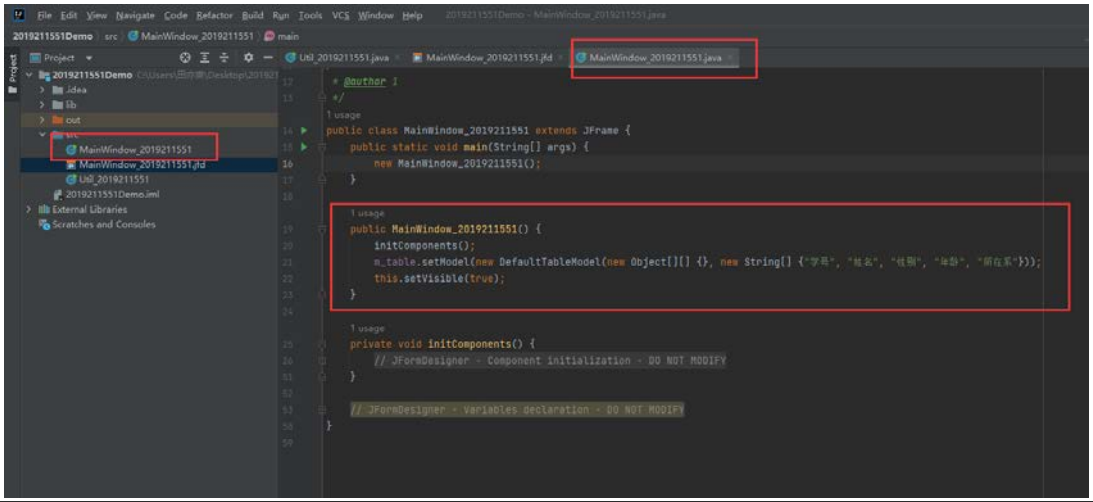
4. 继续找到设计页面中的“Components”下面的“JTable”，点击“JTable”将其放置在软件中的合适位置，并且将其大小调整为合适的大小。



5. 点击 Table 控件，将 Name 属性改为“m_table”

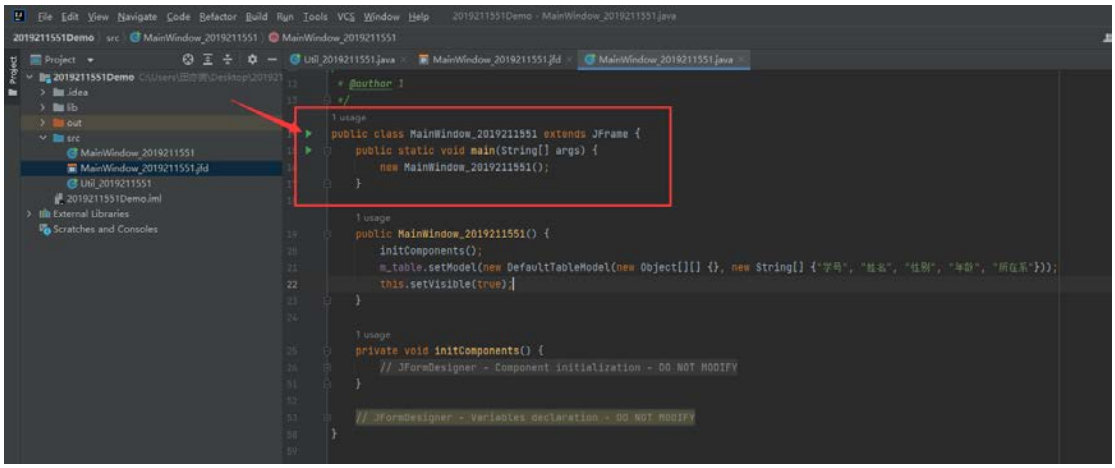


6. 在 Main_Windoow 的.java 文件中找到窗体类的构造方法，添加表格 Table 的初始化代码。



```
m_table.setModel(new DefaultTableModel(new Object[][] {}, new String[] {"学号", "姓名", "性别", "年龄", "所在系"}));
this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
this.setVisible(true);
```

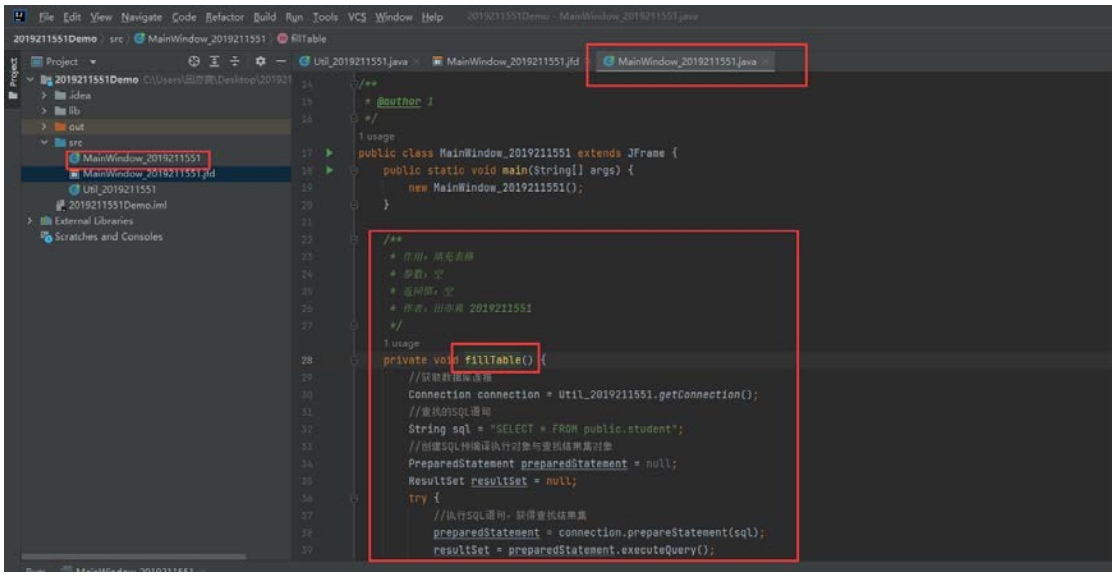
7. 最后编写一个 main 方法进行窗体的测试。测试结果如下图所示。



```
public static void main(String[] args) {
    new MainWindow_2019211551();
}
```

四、数据库中表格数据的加载

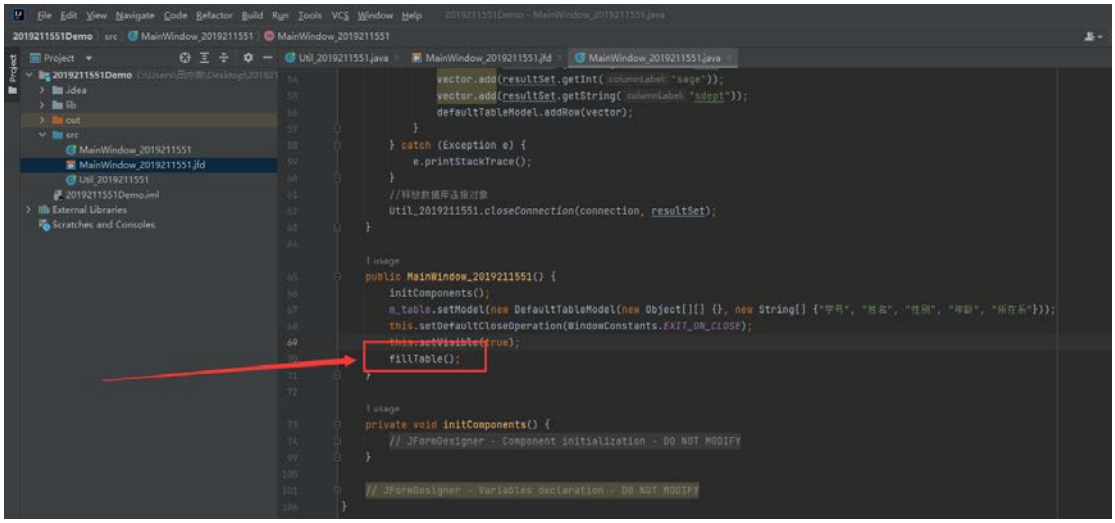
1. 在 MainWindow_2019211551.java 文件中添加一个 fillTable 方法，代码如下所示。

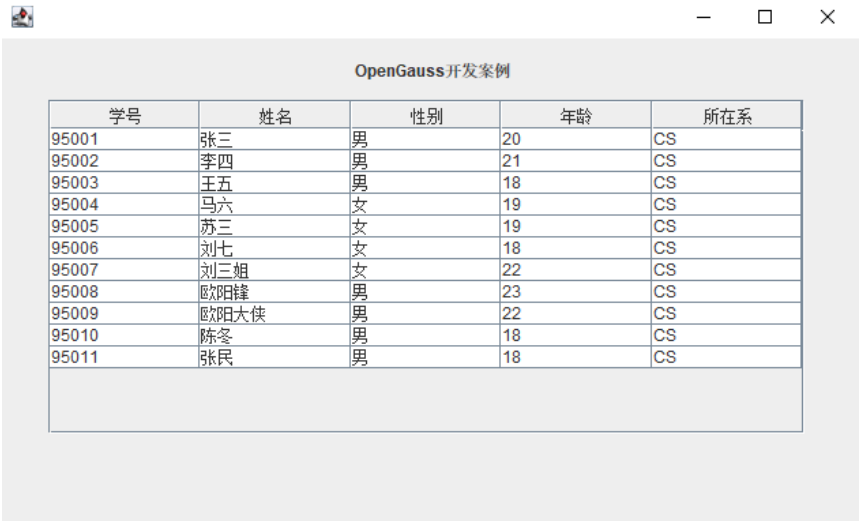


```
/**
 * 作用：填充表格
 * 参数：空
 * 返回值：空
 * 作者：田亦爽 2019211551
 */
private void fillTable() {
    //获取数据库连接
    Connection connection = Util_2019211551.getConnection();
    //查找的 SQL 语句
    String sql = "SELECT * FROM public.student";
    //创建 SQL 预编译执行对象与查找结果集对象
    PreparedStatement preparedStatement = null;
    ResultSet resultSet = null;
    try {
        //执行 SQL 语句，获得查找结果集
        preparedStatement = connection.prepareStatement(sql);
        resultSet = preparedStatement.executeQuery();
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "学生查找失败！");
        e.printStackTrace();
    }
    //刷新表格数据
    DefaultTableModel defaultTableModel = (DefaultTableModel)
m_table.getModel();
    defaultTableModel.setRowCount(0);
    try {
        //遍历查找集中的每一个子对象
```

```
        while (resultSet.next()) {
            Vector vector = new Vector();
            vector.add(resultSet.getString("sno"));
            vector.add(resultSet.getString("sname"));
            vector.add(resultSet.getString("ssex"));
            vector.add(resultSet.getInt("sage"));
            vector.add(resultSet.getString("sdept"));
            defaultTableModel.addRow(vector);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    //释放数据库连接对象
    Util_2019211551.closeConnection(connection, resultSet);
}
```

2. **在窗体构造函数 MainWindow()中引用该方法，完成数据表的刷新。运行类中的 main 方法，进行测试。测试结果如下所示。**





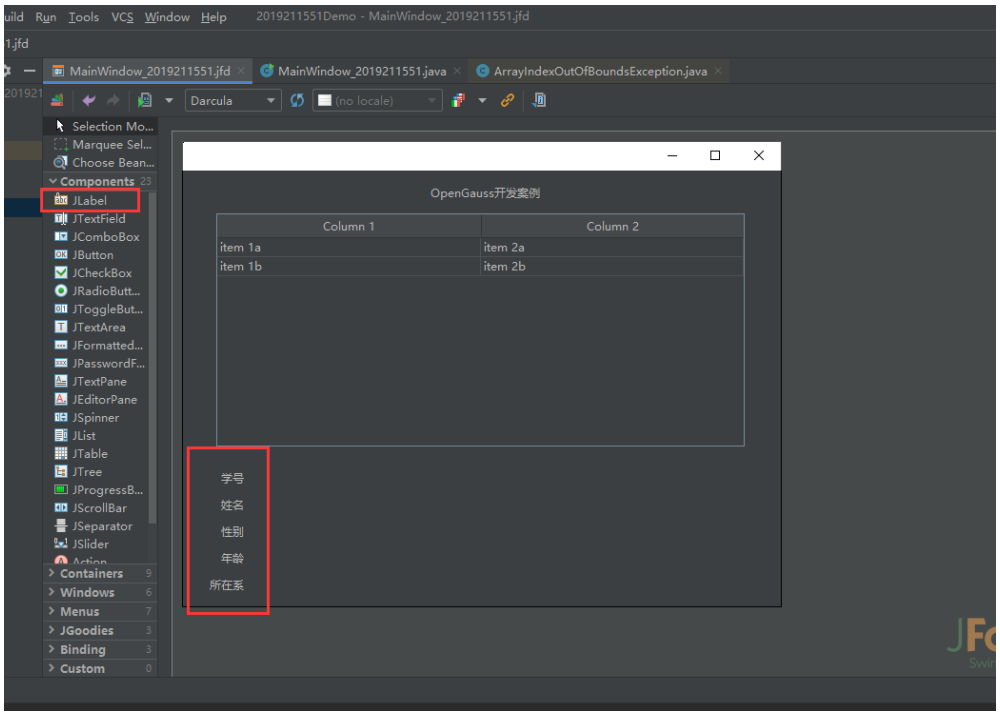
Java+OpenGauss 开发教程（二）

合肥工业大学 计算机与信息学院 张国富

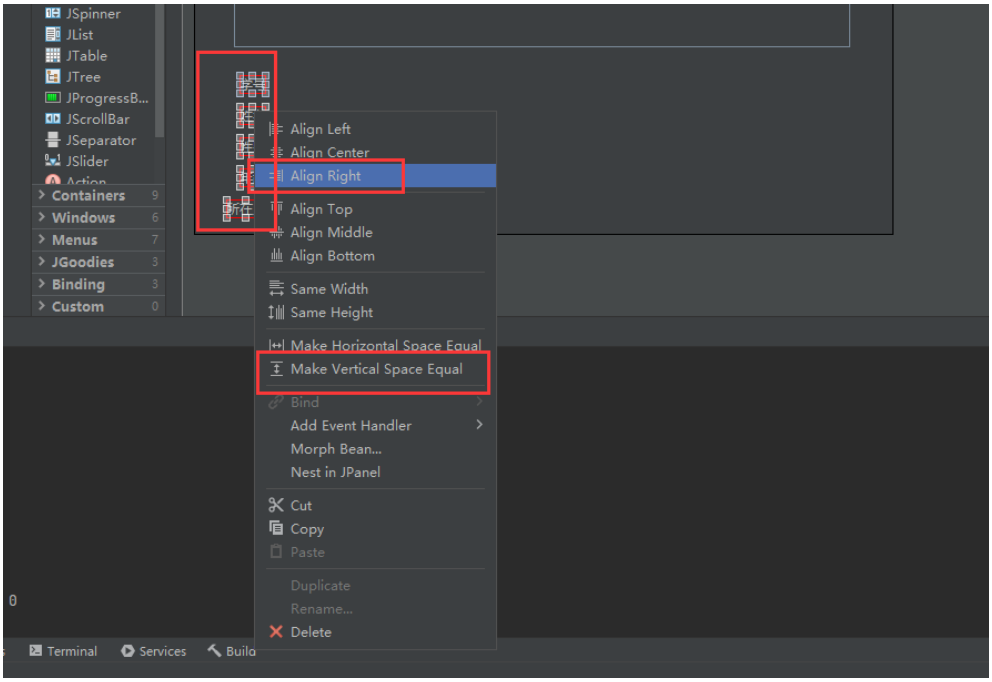
zgf@hfut.edu.cn

一、界面控件的绘制

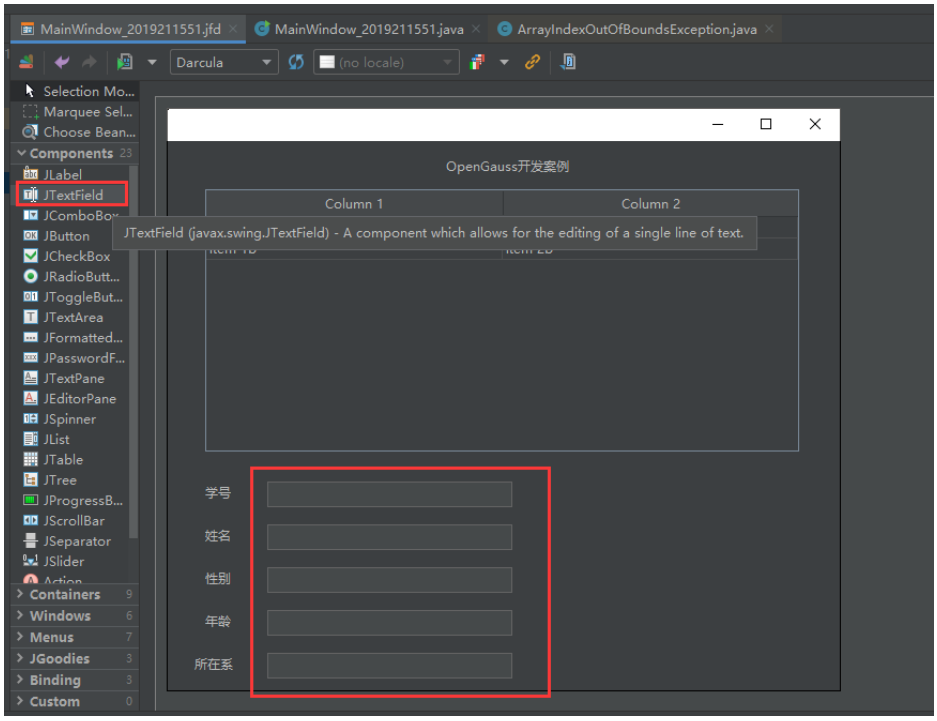
1. 选择左手边的“Components”，点击“JLabel”，创建五个“JLabel”并且双击每一个控件，将其显示字符改为“学号”、“姓名”、“性别”、“年龄”、“所在系”。

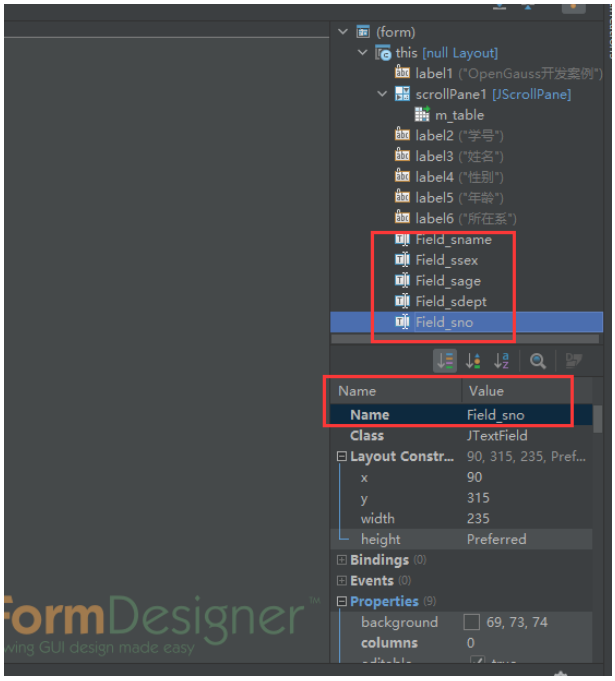


2. 为了使软件效果美观，我们需要按住 Shift 依次点击这些控件，来选中这五个 JLabel 控件，点击右对齐“Align Right”与垂直等间距“Make Vertical Space Equal”，来做到美观处理。



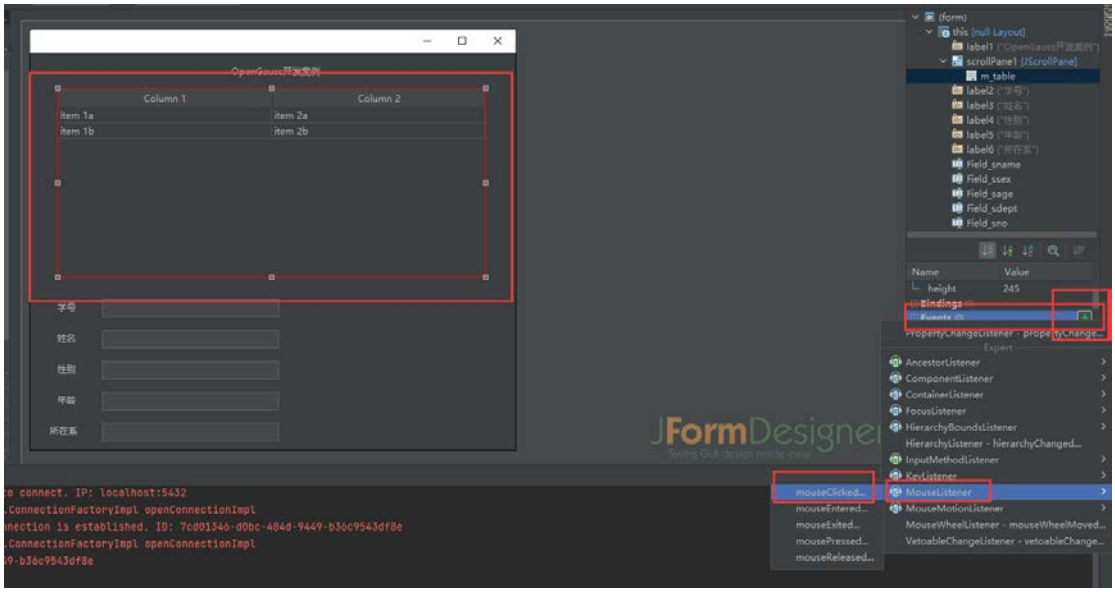
3. 左手边找到“JTextField”，在“学号”后面绘制该控件，并且将其“Name”属性改为“Field_sno”。
左手边找到“JTextField”，在“姓名”后面绘制该控件，并且将其“Name”属性改为“Field_sname”。
左手边找到“JTextField”，在“性别”后面绘制该控件，并且将其“Name”属性改为“Field_ssex”。
左手边找到“JTextField”，在“年龄”后面绘制该控件，并且将其“Name”属性改为“Field_sage”。
左手边找到“JTextField”，在“所在系”后面绘制该控件，并且将其“Name”属性改为“Field_sdept”。

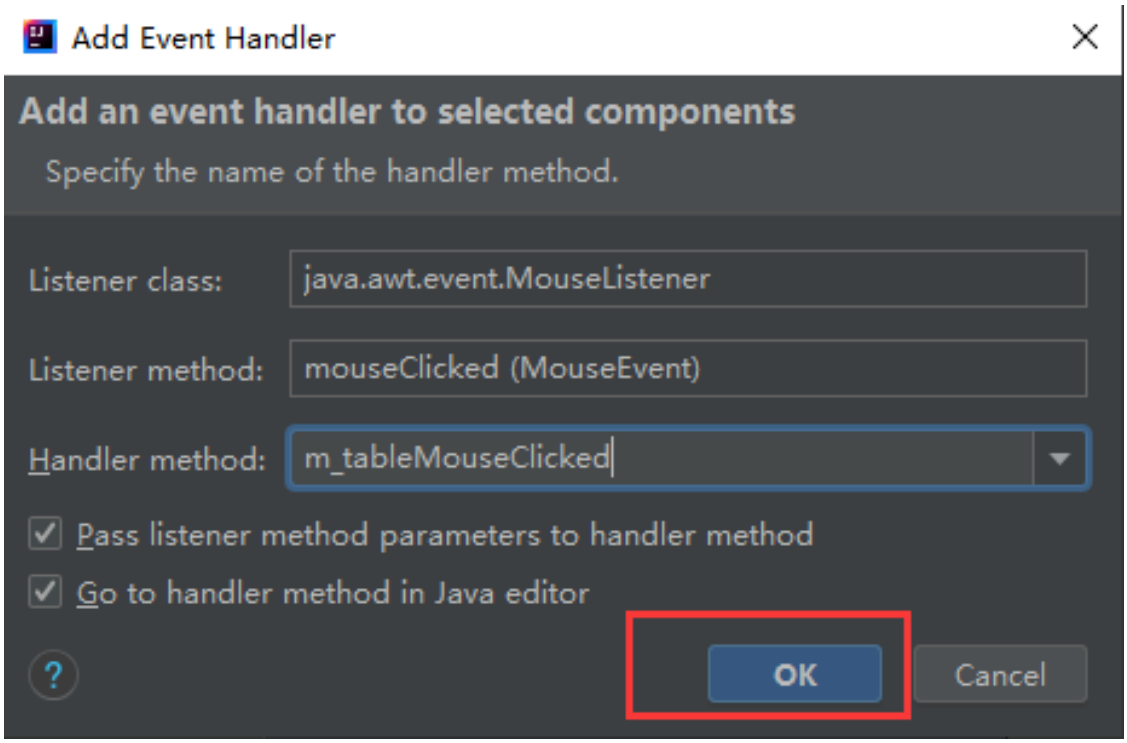




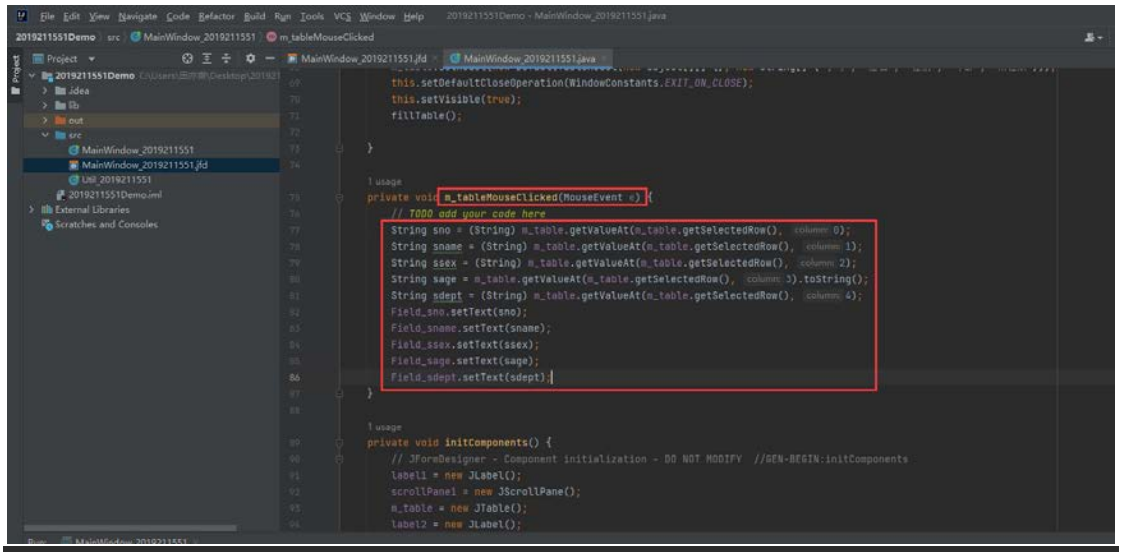
二、实现表格中选中行的数据显示

1. 点击 Table 控件，找到右下属性中的“Events”，点击右边的绿色加号，找到“MouseListener”下的“mouseClicked”事件，点击并且弹出窗口中选择“OK”，跳转至鼠标单击事件的事件处理方法中去。



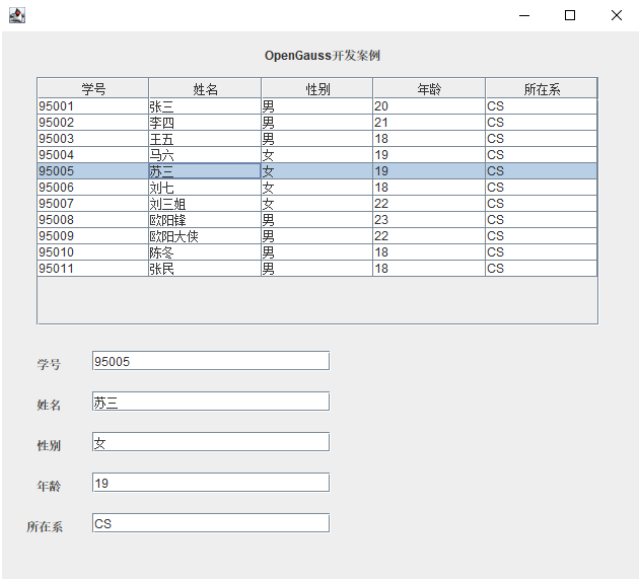


2. 在事件处理方法中添加以下代码。



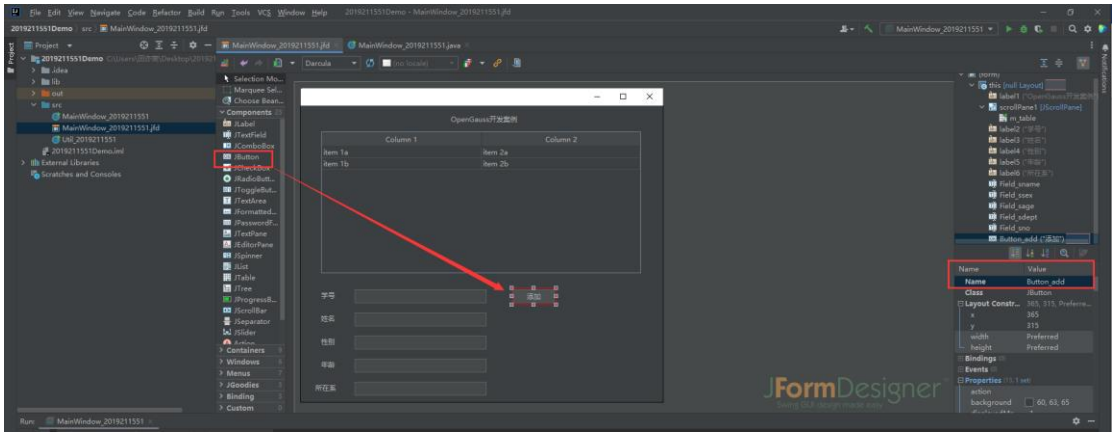
```
String sno = (String) m_table.getValueAt(m_table.getSelectedRow(),
0);
String sname = (String) m_table.getValueAt(m_table.getSelectedRow(),
1);
String ssex = (String) m_table.getValueAt(m_table.getSelectedRow(),
2);
String sage = m_table.getValueAt(m_table.getSelectedRow(),
3).toString();
String sdept = (String) m_table.getValueAt(m_table.getSelectedRow(),
4);
Field_sno.setText(sno);
Field_sname.setText(sname);
Field_ssex.setText(ssex);
Field_sage.setText(sage);
Field_sdept.setText(sdept);
```

3. 执行效果如下。

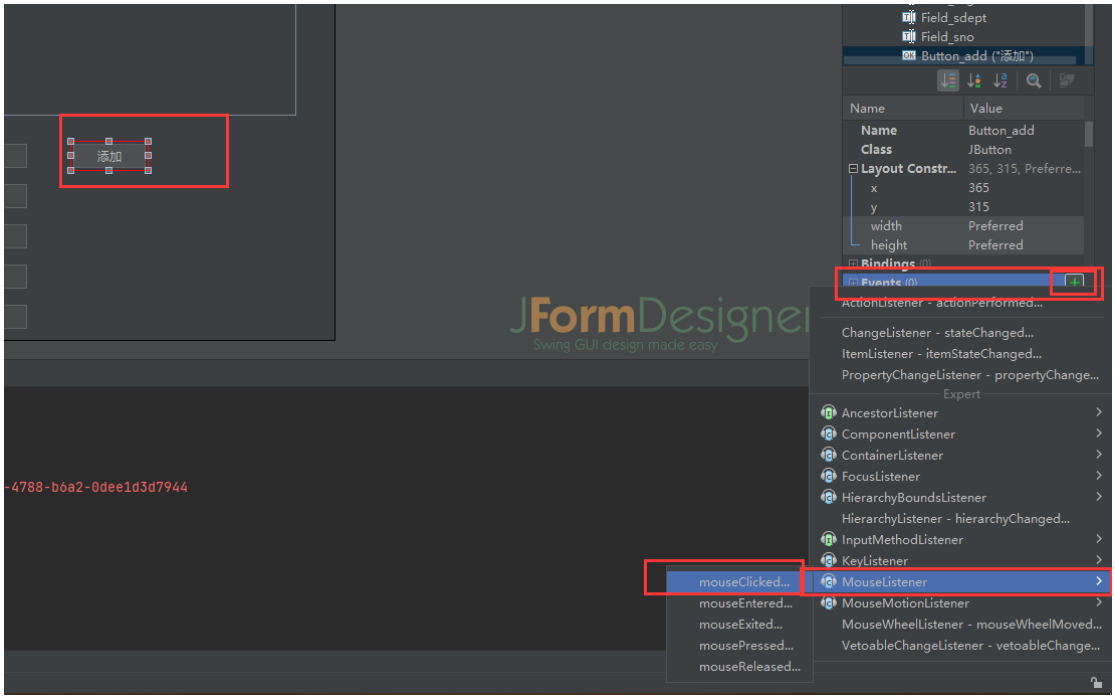


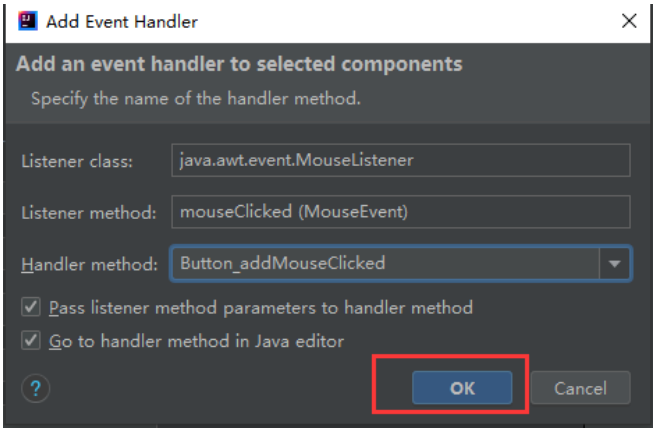
三、实现数据的添加

1. 回到设计视图，找到“Components”目录下的“JButton”控件，将其添加至软件的合适位置，并且在右手边将其“Name”属性改为“Button_add”。

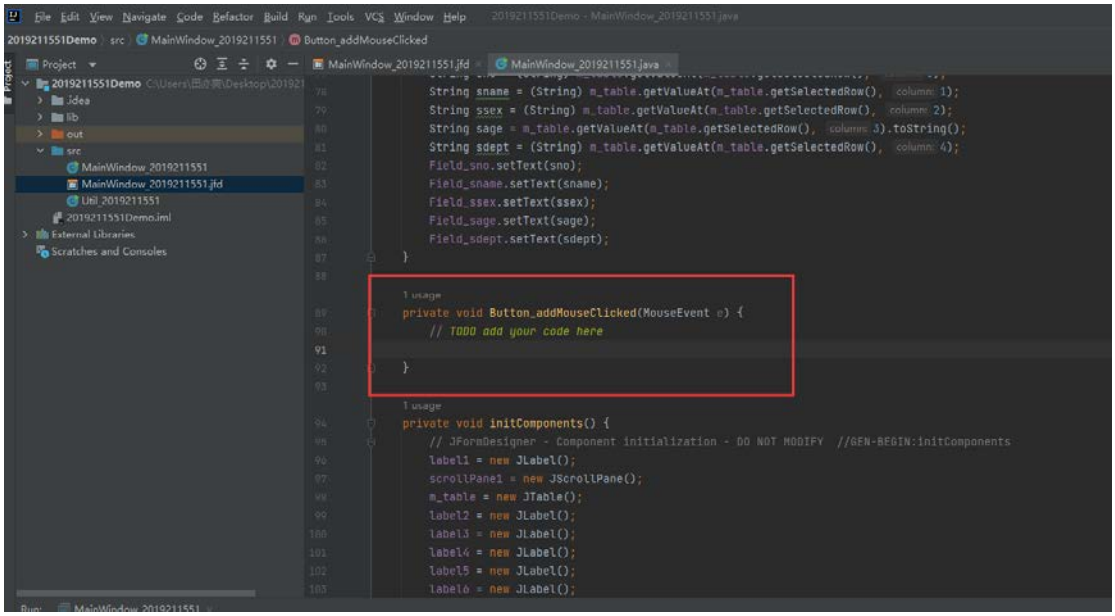


2. 在属性中找到“Events”，点击右边的绿色加号，找到“MouseListener”中的“MouseClicked”，添加一个鼠标单击的事件监听，点击“OK”，跳转至事件处理函数中去。





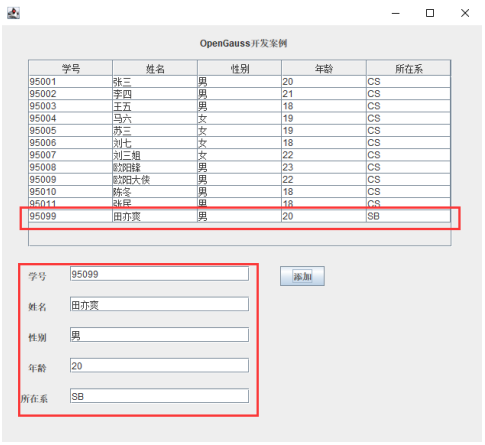
3. 将以下代码添加至 Button_addMouseClicked()函数中去。

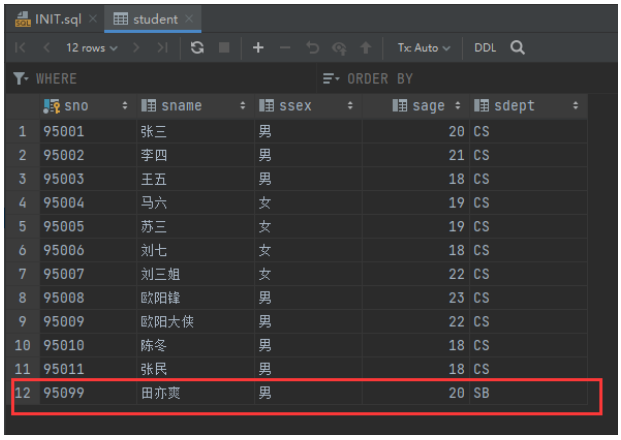



```
String sno = Field_sno.getText();
String sname = Field_sname.getText();
String ssex = Field_ssex.getText();
int sage = 0;
String sdept = Field_sdept.getText();
if (Objects.equals(sno, "")) {
    JOptionPane.showMessageDialog(null, "学号不能为空！");
    return;
}
if (Objects.equals(sname, "")) {
    JOptionPane.showMessageDialog(null, "姓名不能为空！");
    return;
}
if (Objects.equals(ssex, "")) {
    JOptionPane.showMessageDialog(null, "性别不能为空！");
    return;
}
try {
    sage = Integer.parseInt(Field_sage.getText());
} catch (Exception exception) {
    JOptionPane.showMessageDialog(null, "年龄不能为空！");
    exception.printStackTrace();
    return;
}
if (Objects.equals(sdept, "")) {
    JOptionPane.showMessageDialog(null, "院系不能为空！");
    return;
}
```

```
Connection connection = Util_2019211551.getConnection();

String sql = "INSERT INTO public.student (sno, sname, ssex, sage,
sdept) VALUES (?,?,?,?,?)";
PreparedStatement preparedStatement = null;

try {
    preparedStatement = connection.prepareStatement(sql);
    preparedStatement.setString(1, sno);
    preparedStatement.setString(2, sname);
    preparedStatement.setString(3, ssex);
    preparedStatement.setInt(4, sage);
    preparedStatement.setString(5, sdept);
    preparedStatement.executeUpdate();
    preparedStatement.close();
    JOptionPane.showMessageDialog(null, "学生添加成功！");
    System.out.println("学生添加成功");
} catch (SQLException ee) {
    JOptionPane.showMessageDialog(null, "输入有误，学生添加失败！");
    System.out.println("学生添加失败");
    ee.printStackTrace();
}
fillTable();
```

4. 编译运行，点击添加按钮，成功将数据添加至 OpenGauss 中去。





Java+OpenGauss 开发教程（三）

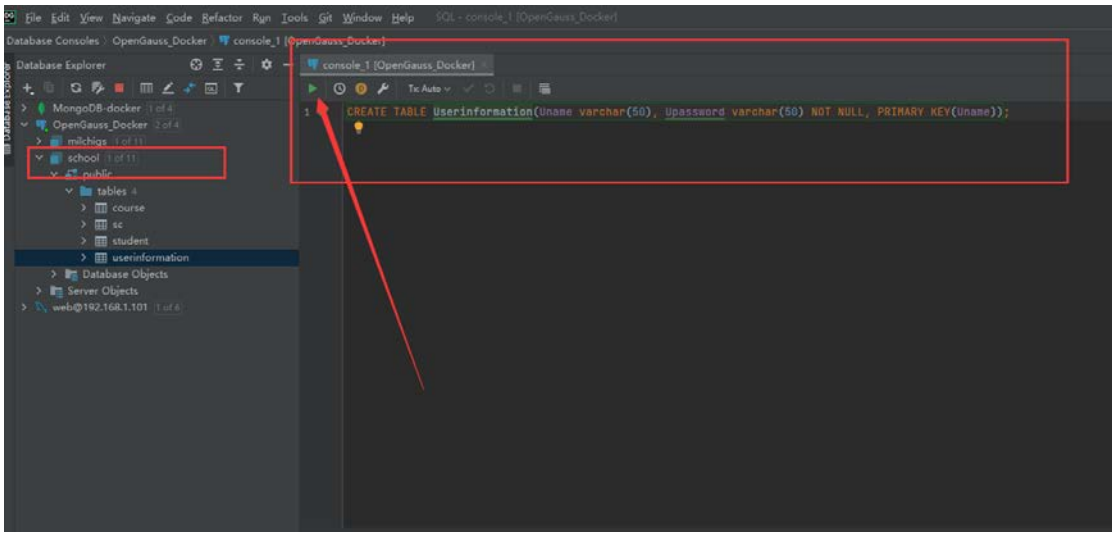
合肥工业大学 计算机与信息学院 张国富

zgf@hfut.edu.cn

一、用户表格的创建

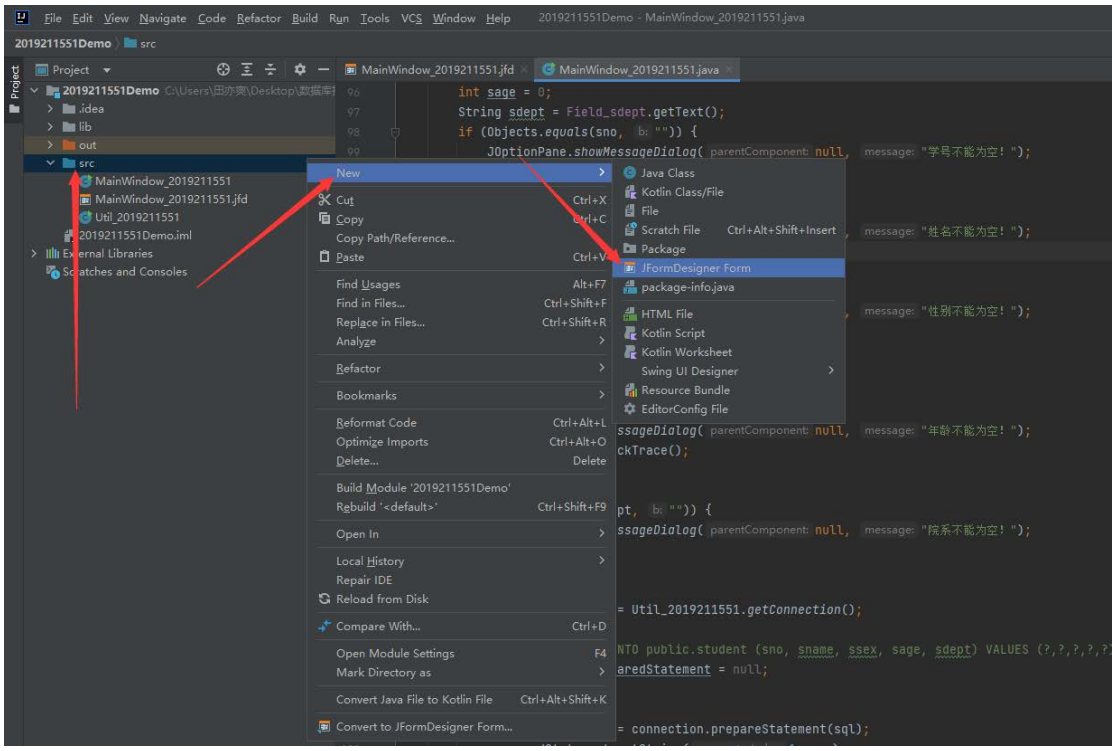
1. 打开 OpenGauss 虚拟机，启动 OpenGauss 数据库，以管理员身份运行 Data Studio，连接上 OpenGauss 虚拟机，连接上 school 数据库，选中 school 数据库，打开新的终端，输入如下 SQL 语言创建：用户表。

```
CREATE TABLE Userinformation(Uname varchar(50), Upassword varchar(50) NOT NULL, PRIMARY KEY(Uname));
```

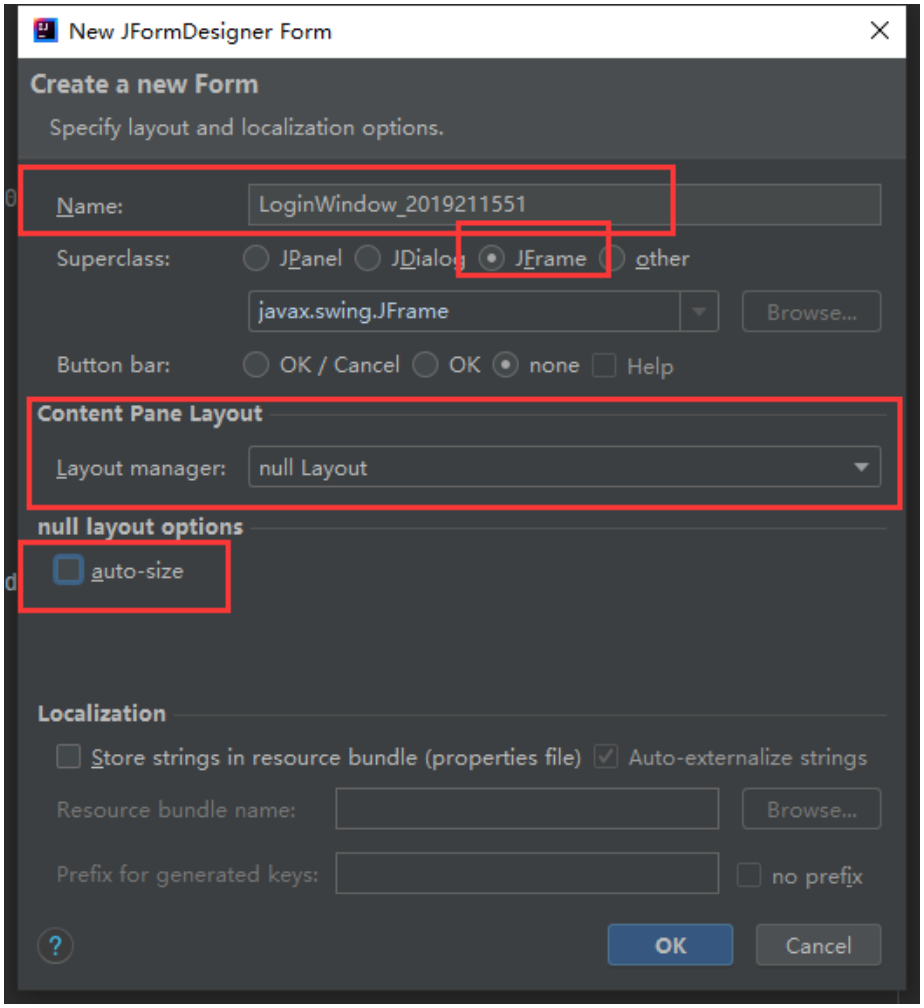


二、登录对话框的创建

2. 右键 src 文件夹，创建一个 JFormDesigner Form 窗体文件。

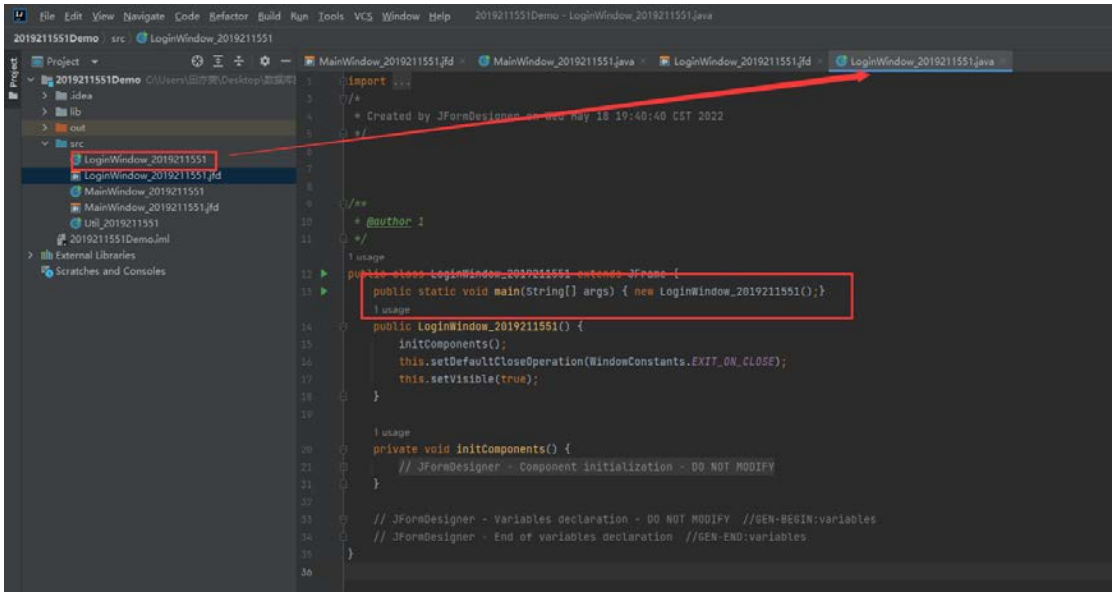


3. 给登录窗口起一个带学号的类名“LoginWindow_2019211551”，选择 JFrame 窗体，不使用任何 Layout 布局，取消勾选 auto-size，点击 OK，完成创建。



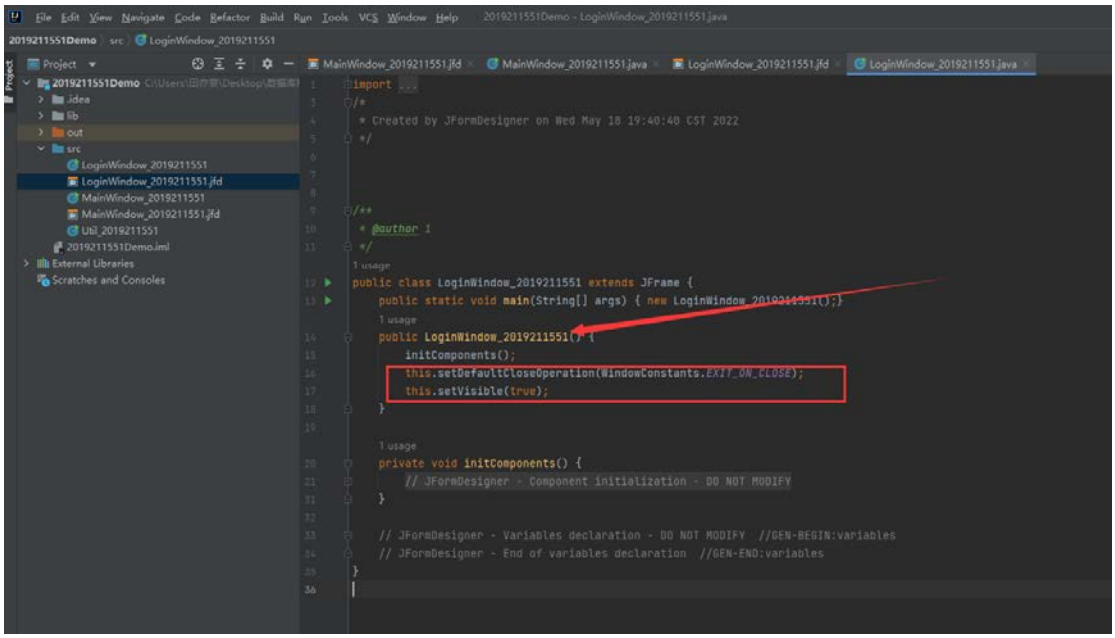
4. 给窗体类代码中添加主函数。找到我们创建的窗体代码，打开在对应位置填入以下代码，让该窗体可以被执行。

```
public static void main(String[] args){new LoginWindow_2019211551();}
```

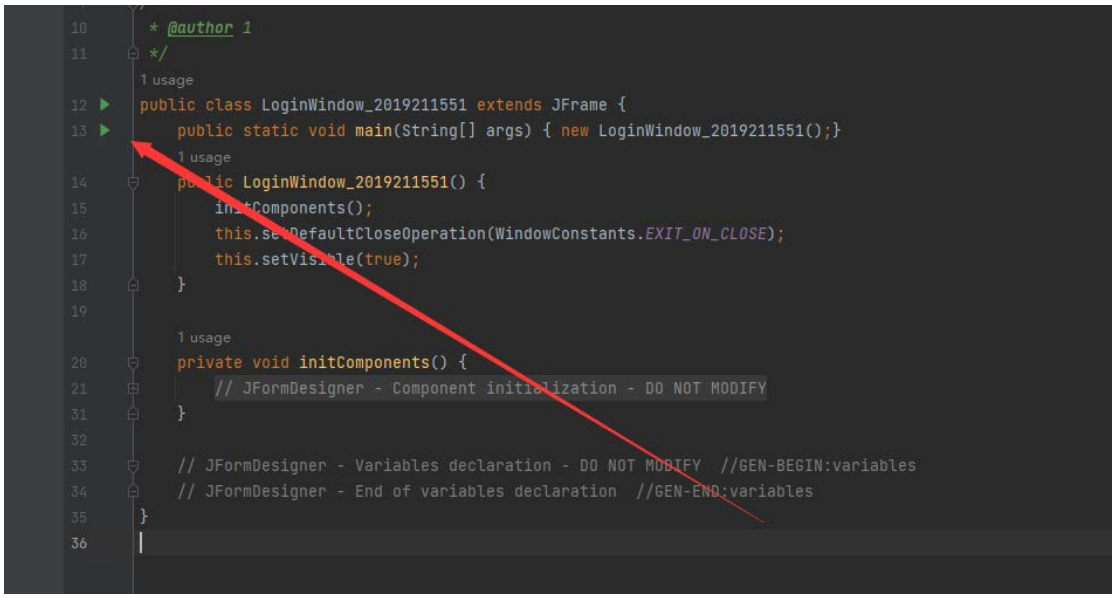


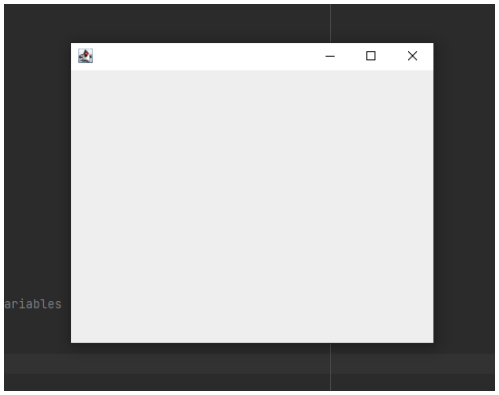
5. 补全构造函数中的代码。配置窗体关闭时的操作与设置窗体可见性为 true。

```
this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
this.setVisible(true);
```



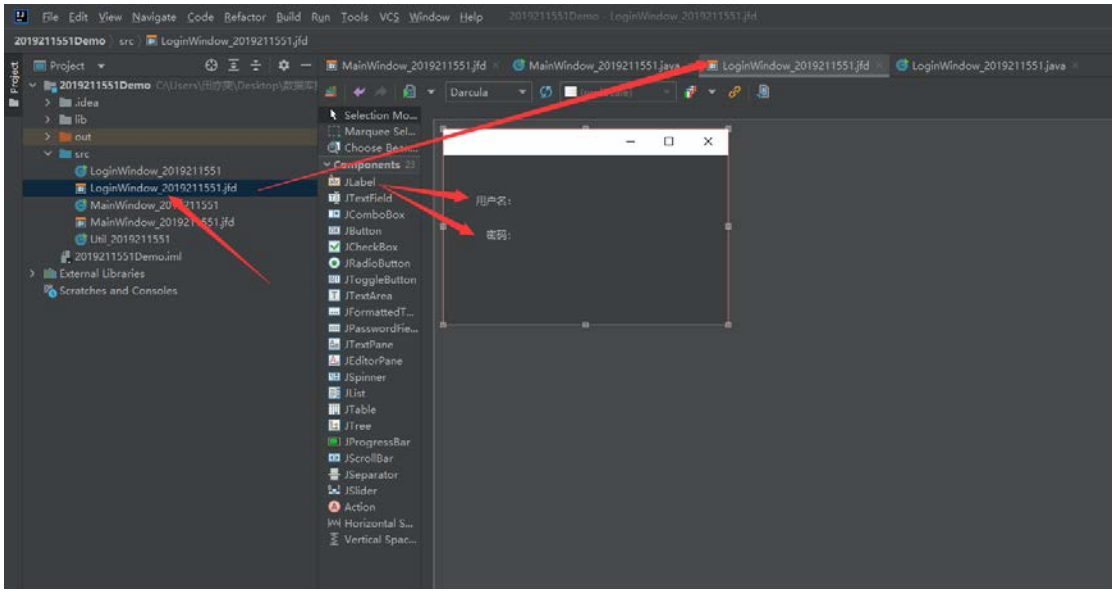
6. 点击绿色小三角，即可执行我们刚刚创建的窗体了。执行结果如下所示。



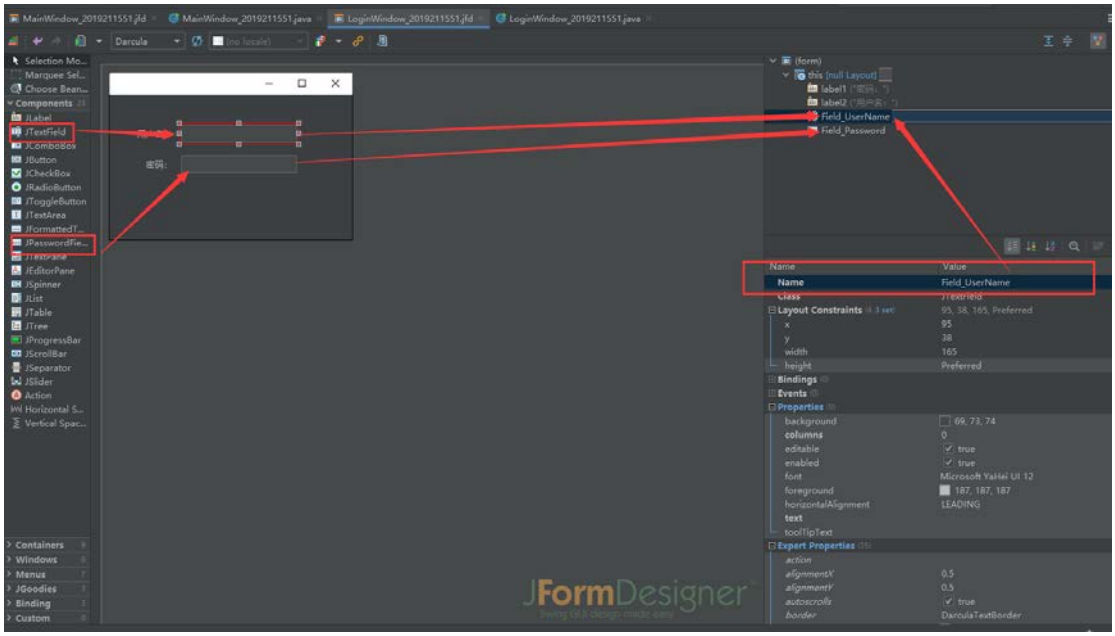


三、登录界面的绘制

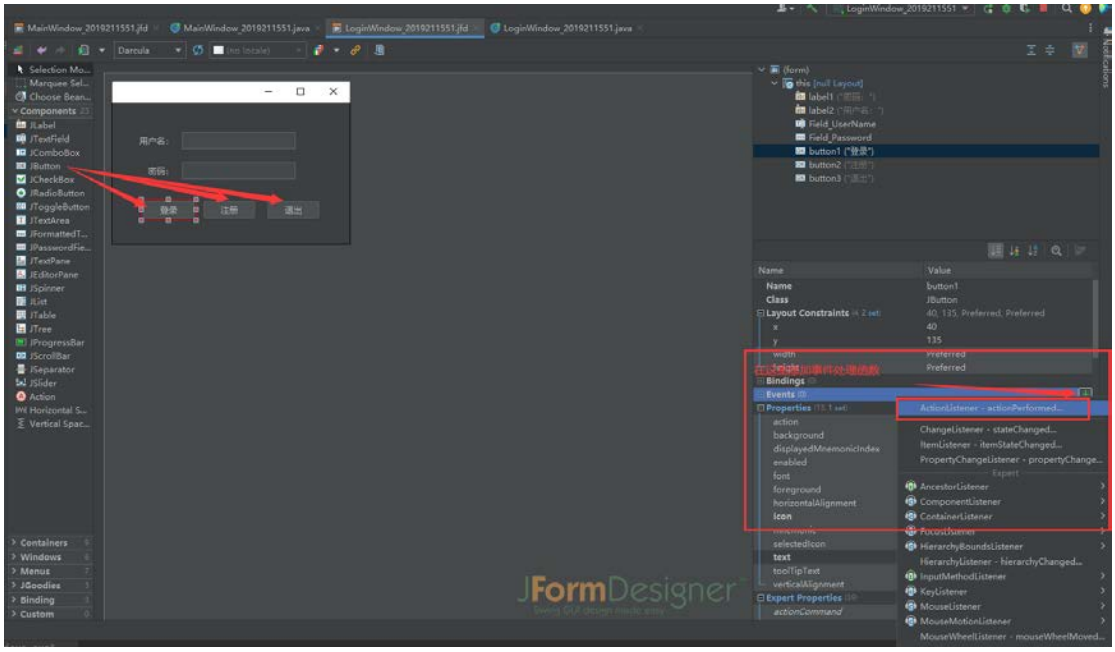
1. 绘制 JLabel，在图形设计窗体下，拖动设计窗口左方的 Components 中的 JLabel 组件，分别双击将里面的文字更改为“用户名:”与“密码:”



2. 分别绘制用户名输入框与，密码输入框。分别使用 Components 下的 JTextField 组件与 JPasswordField 组件。并且将两个组件对应的 Name 属性改成“Field_UserName”与“Field_Password”。



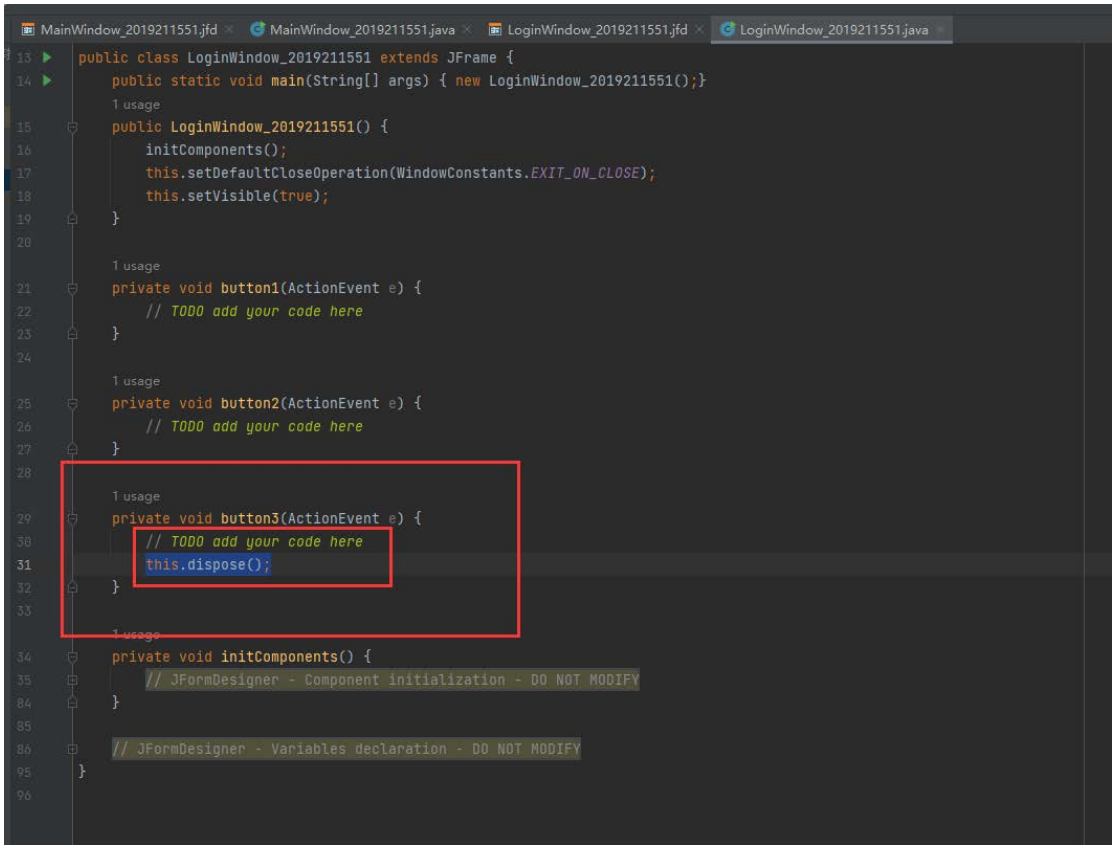
3. 创建登录，注册，退出按钮，并创建对应的单击事件处理。使用 Components 下的 JButton 控件，将其拖入窗体程序，双击更改其对应的文字提示“登录”，“注册”，“退出”。并且在右下方的属性中找到 Events，分别生成按钮对应的单击事件处理函数。



四、退出按钮的实现

1. 找到第三步添加的退出按钮事件处理函数，添加以下代码。即可运行测试窗体程序。

```
this.dispose();
```



2. 测试结果。

五、 关于注册登录按钮的提示

注册按钮：只需要将 SQL 代码送给 OpenGauss 去执行，将用户名和密码输入到表格中，不需要返回数据。

登录按钮：不仅需要将 SQL 代码送给 OpenGauss 去执行，还需要 OpenGauss 返回执行的结果，以检查是否真的存在这个账号。

Pycharm+OpenGuass 开发教程（一）

合肥工业大学 计算机与信息学院 张国富

zgf@hfut.edu.cn

介绍：

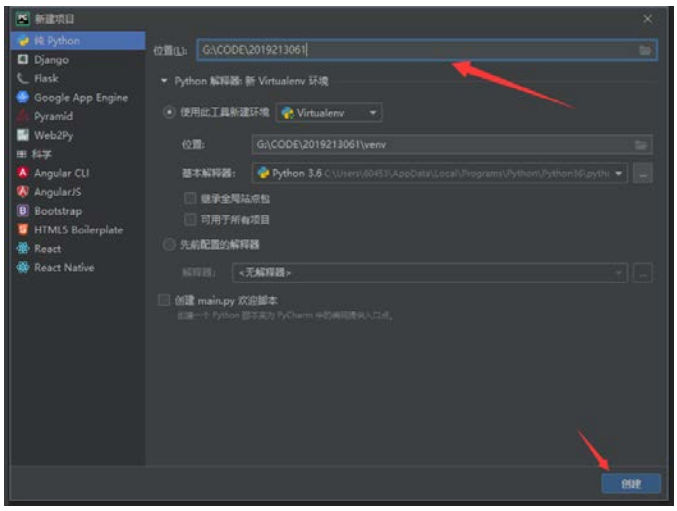
实验环境：WIN10，Python 3.6.5

使用 IDE：Pycharm

OpenGauss 安装环境：VM+centOS linux7

一. 创建一个初始工程

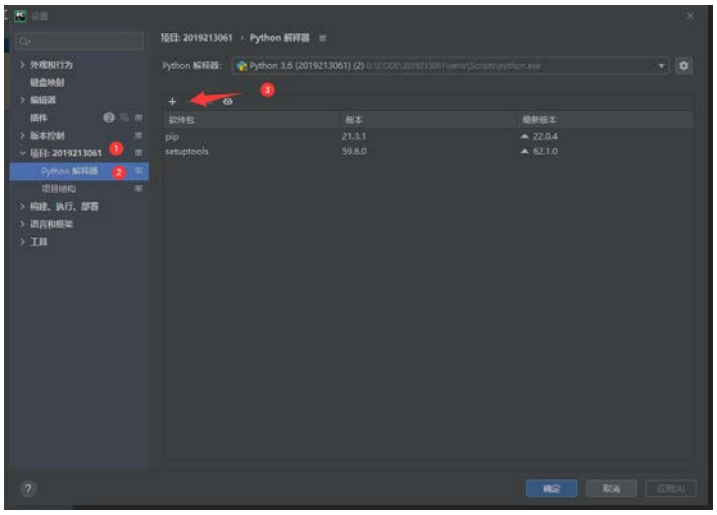
1. 打开 Pycharm，选择左上角“文件”→“新建项目”, 修改项目文件夹的位置和名称，点击“创建”



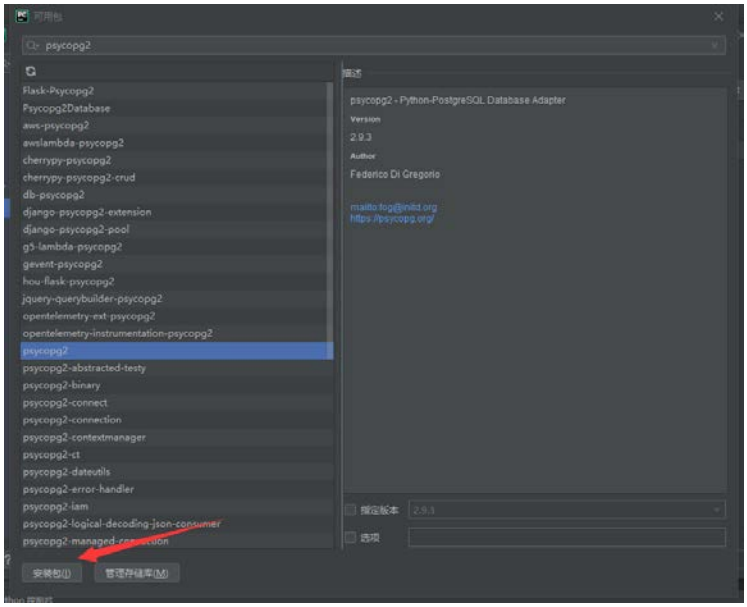
二. 安装连接 OpenGauss 驱动 psycopg2 库

（psycopg2，是 Python 语言的 PostgreSQL 数据库接口）

1. 选择左上角“文件”→“设置”→“项目：（你的项目名）”→“python 解释器”可以看到 Python 当前已安装的软件包也就是各种库，在这里点击加号可以下载第三方库：



2. 搜索 psycopg2 点击安装即可：

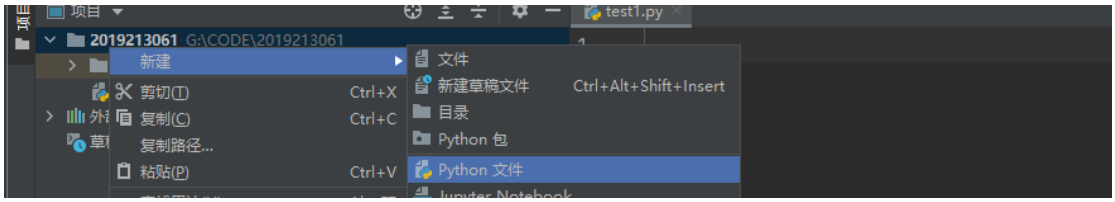


3. 安装完成后可以在刚才的列表中看到新安装的第三方库 psycopg2：

（这里如果直接用 pip 安装会出问题所以建议在 Pycharm 内安装）

4. 测试 psycopg2 与 OpenGauss 连接

新建 python 文件“test1”

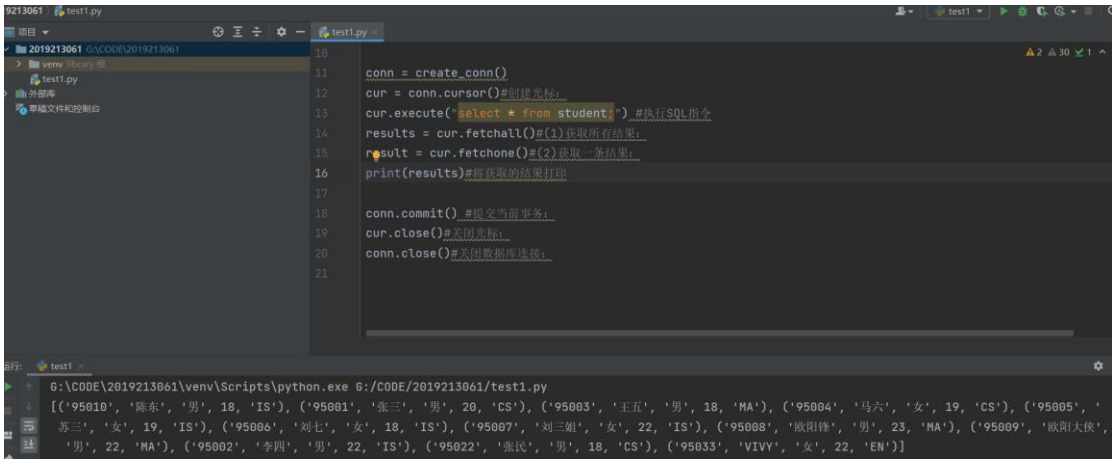


输入以下代码：

```
import psycopg2
def create_conn():
    database='school'#选择数据库名称
    user='dboper'
    password='dboper@123'
    host='192.168.174.134'#数据库 ip
    port='26000'
    conn = psycopg2.connect(database=database, user=user, password=password,
host=host, port=port) #连接数据库
    return conn

conn = create_conn()
cur = conn.cursor()#创建光标:
cur.execute("select * from student;") #执行 SQL 指令
results = cur.fetchall()#(1)获取所有结果:
result = cur.fetchone()#(2)获取一条结果:
print(results)#将获取的结果打印
```

```
conn.commit() #提交当前事务：
cur.close()#关闭光标：
conn.close()#关闭数据库连接：
```

将数据库 IP 改成自己数据库的 IP，然后按下右键执行程序。如果程序正常执行并输出打印了表中的所有数据则，说明与数据库正常连接。



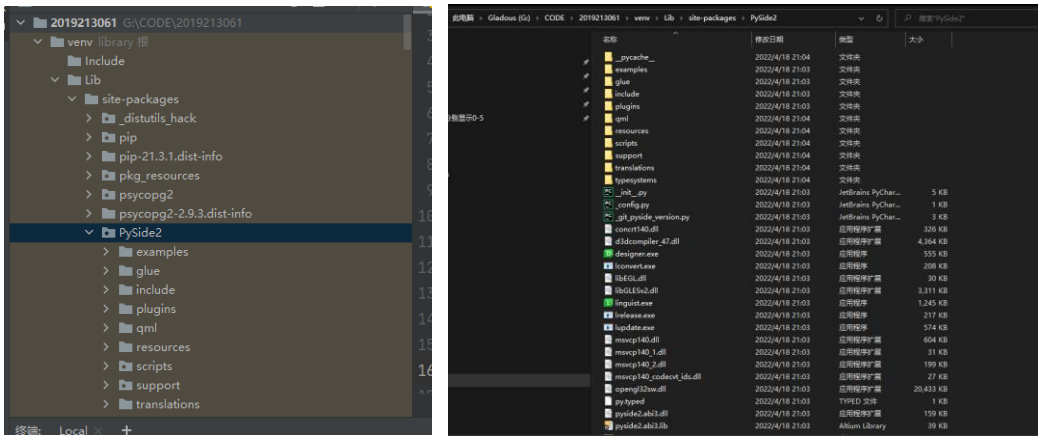
可以在终端看到 student 表中的所有数据，可以看到 fetchall 函数返回的数据是列表，列表中每个元祖内的数据则是每一行的数据。

二. 安装 python 中的 QT 库，Pyside2。

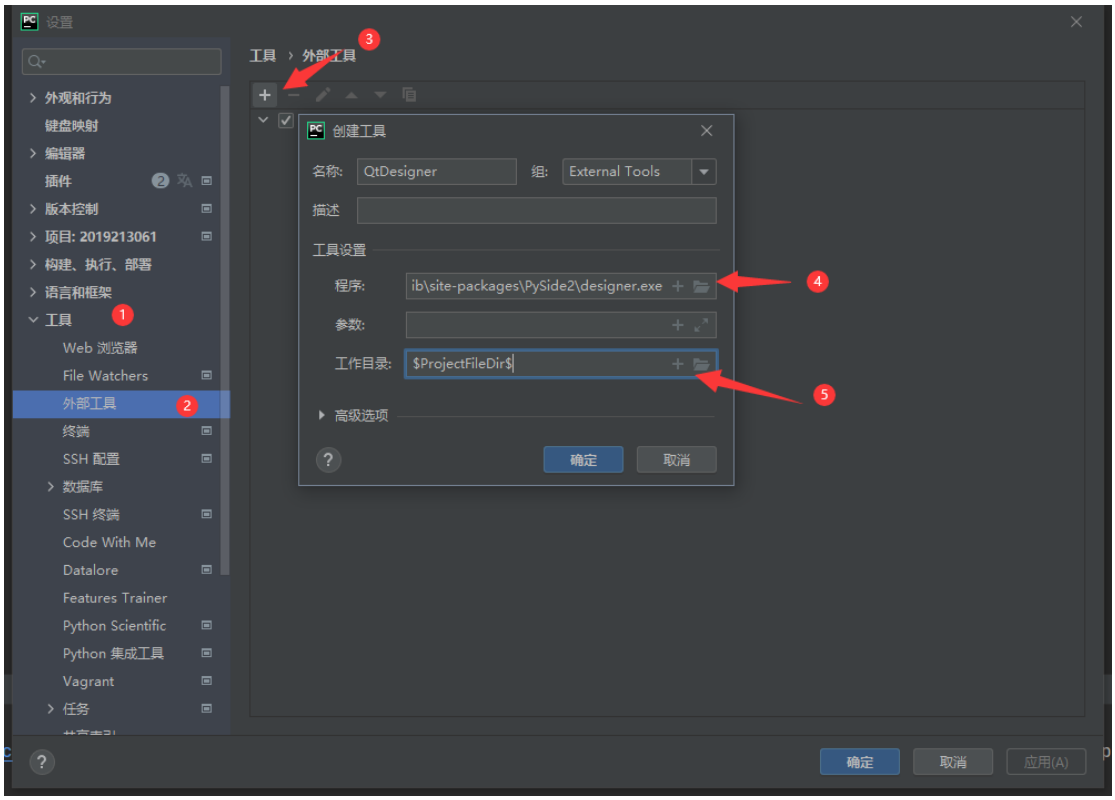
点击左下角终端，复制这条语句进入终端：pip install pyside2 -i https://pypi.tuna.tsinghua.edu.cn/simple 并执行

出现 successfully 代表安装完成

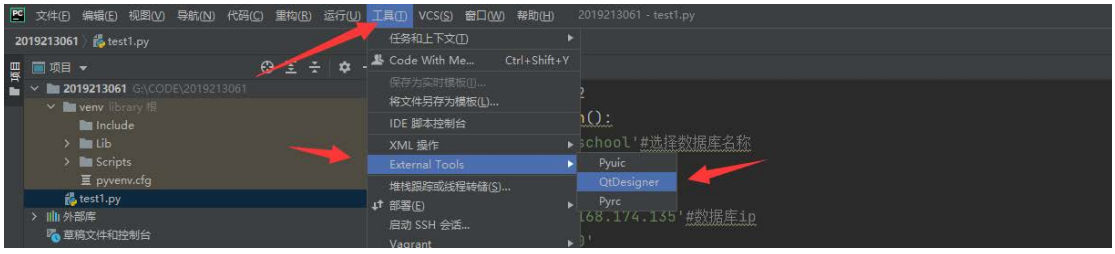
Pyside2 可以用于 GUI 的开发，且自带 designer.exe。在 Pycharm 左边我们可以看到 Pyside2 的安装目录（20xxxxxx\venv\Lib\site-packages\PySide2）



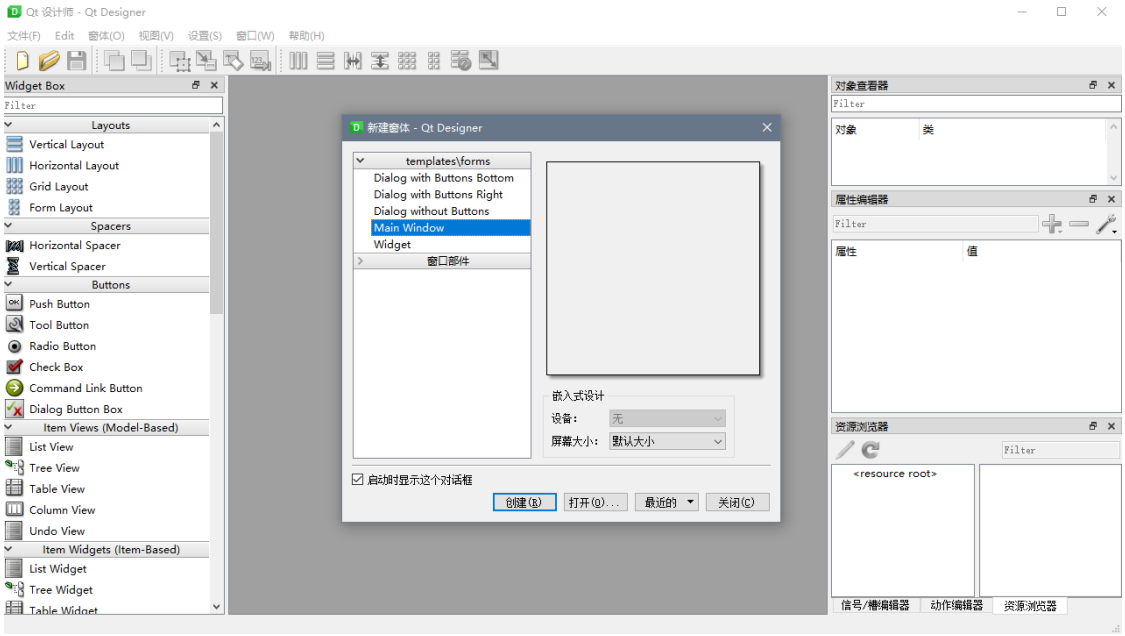
接下来设计图形界面就需要用到 designer.exe 和第三方库 pyside2。为了让 designer 的使用更加方便，可以将 designer 设置到 Pycharm 的外部工具中。选择左上角“文件”→“设置”→“工具”→“ 外部工具”，点击加号创建工具，名称可以为 QtDesigner，程序：刚刚找到的 desginer.exe 的绝对地址，工作目录：$ProjectFileDir$，点击确定。（$ProjectFileDir$的意思是将生成文件保存在项目目录下）



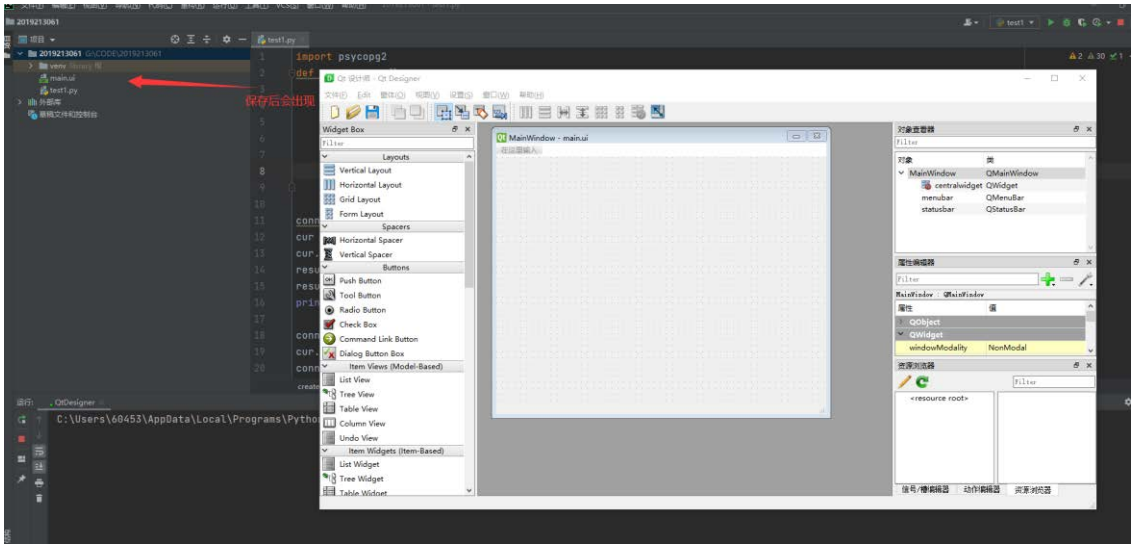
随后我们可以在 PyCharm 上方“工具”中看到我们设置好的 QtDesigner



点击即可进入如下界面：



点击创建即可得到一个基础的窗口，保存后可以在项目文件里看到我们刚生成的.ui 文件。接下来我们设计图形化界面就可以直接用 designer 来操作（与 VS 中的 MFC 类似），然后用 python 调用.ui 文件，写相应的响应函数和后台程序即可，不需要用代码来写界面：



Pycharm+OpenGuass 开发教程（二）

合肥工业大学 计算机与信息学院 张国富

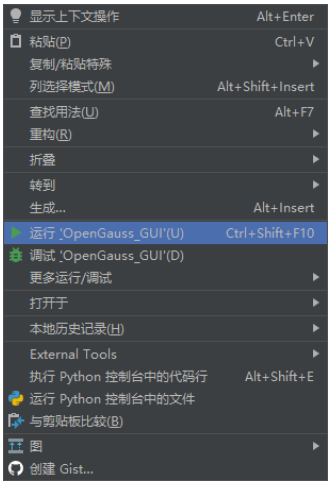
zgf@hfut.edu.cn

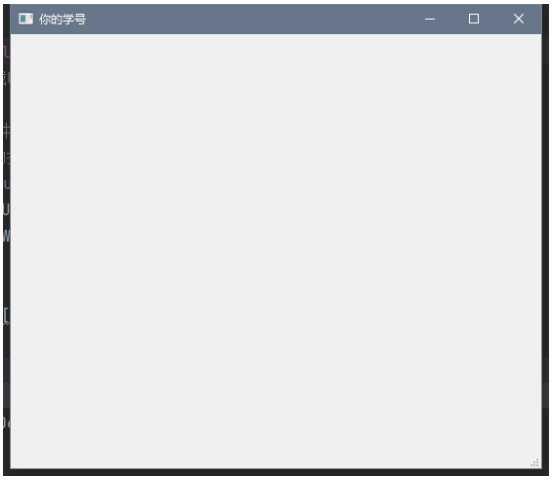
一. 连接 Opengauss 数据库并设置弹窗

1.新建 python 文件“OpenGauss_GUI.py”,然后复制如下代码：

```
from PySide2.QtWidgets import QApplication,QMessageBox,QTableWidgetItem
from PySide2.QtUiTools import QUiLoader
import psycopg2
import sys
# 导入相关库
# 定义一个类专门处理 GUI
class Stats:

    def __init__(self):
        # 从文件中加载 UI 定义
        # 从 UI 定义中动态 创建一个相应的窗口对象

        # 注意：里面的控件对象也成为窗口对象的属性了
        # 比如 self.ui.button , self.ui.textEdit
        self.ui = QUiLoader().load('G:\\CODE\\2019213061\\main.ui')
        self.ui.setWindowTitle('你的学号')#设置窗口名称

# 定义一个连接 OpenGauss 的函数
def create_conn():
    database='school'
    user='dboper'
    password='dboper@123'
    host='192.168.174.134'
    port='26000'
    conn = psycopg2.connect(database=database, user=user, password=password,
host=host, port=port) #连接数据库
    return conn


#程序从这里执行
app = QApplication([])
stats = Stats()
stats.ui.show() # 界面显示


sys.exit(app.exec_())#事件处理循环  要不然程序一闪而过  死循环
```

右键点击“运行”后我们可以看到一个空的窗口，代表程序正常可以正常工作。QUiLoader().load()这里读取了我们之前创建的 main.ui 文件（建议写绝对路径），并生成了一个窗口对象返回到类变量 ui 中，接下来我们对窗口进行操作都需要用到类变量 ui。





2. QMessageBox 控件

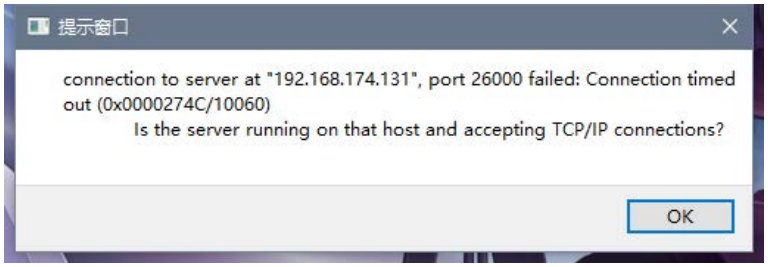
QMessageBox 控件可以用于生产弹窗，接下来我们来实现数据库的连接，在连接成功后给我们弹出一个提示窗口，失败的话将报错信息用弹窗进行显示 stats = Stats()这段代码下加入下述代码：

```
msgBox = QMessageBox()
try:
    conn = create_conn()
except Exception as e:
    msgBox.about(stats.ui, '提示窗口', str(e))

else:
    msgBox.about(stats.ui, '提示窗口', '数据库连接成功  ')#弹窗提示
```

在初始化弹窗对象 msgBox，try 语句下会先执行 conn = create_conn()，如果出现异常则会获得报错信息 e。没有异常则执行 else：中的语句。

其中 About（）函数是 QMessageBox()的类函数，用于弹出窗口，其中第一个参数需要填一个主窗口对象，也就是类变量 ui，第二个参数是弹窗的标题，第三个参数则是弹窗的内容。

右键运行后可以观察到弹窗的出现（如果 IP 有误最终会因为超时报错）



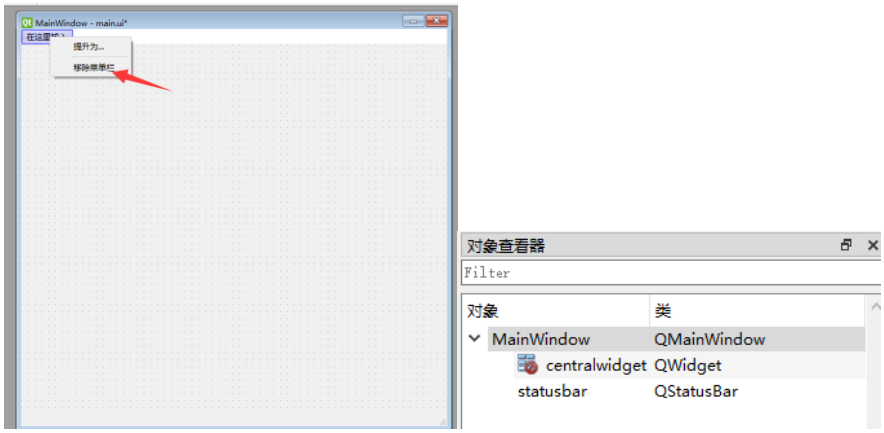
这个时候不管是否连接成功，按下 OK 后会都会显示我们自己 designer 生成的主窗口中，因此我们根据 try 语句的性质将显示界面的那行代码放入 else 里，如下图所示

```
else:
    msgBox.about(stats.ui, '提示窗口', '数据库连接成功   ')#弹窗提示
    stats.ui.show()  # 界面显示



sys.exit(app.exec_())#事件处理循环   要不然程序一闪而过   死循环
```

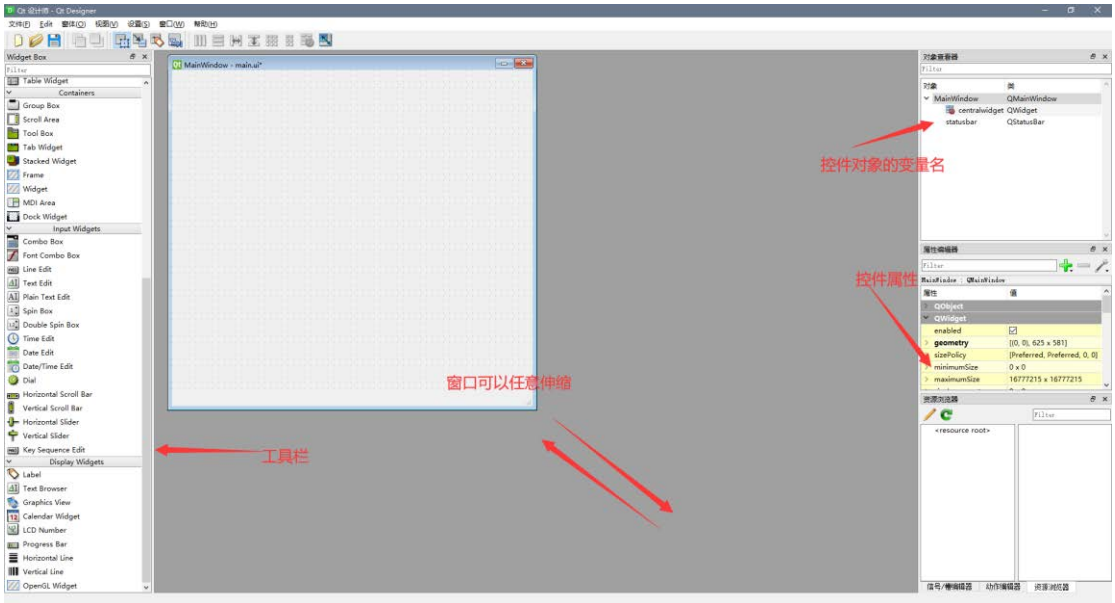
这样只有连接成功时才能进入主窗口中

二. 设计 main.ui

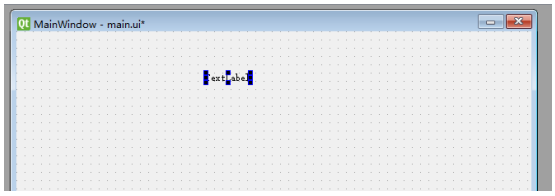
选择“工具”→“工具”→“QtDesigner”启动 designer，开启后我们先将左上角的菜单栏删掉，我们可以发现控件里面的对应菜单的 menubar 已经消失了



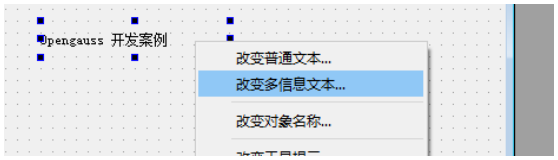
下图为 designer 设计界面的基本介绍：

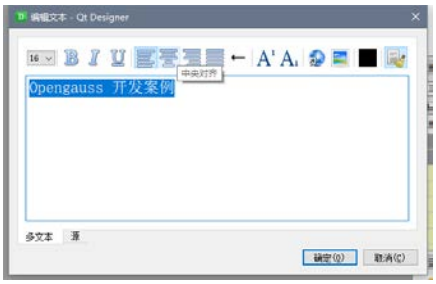


在坐标工具栏中选择 Label 控件，将其拖到主窗口中进行编辑。双击刚刚拖出来的 Label 控件即可对其显示内容进行编辑，将其修改为“Opengauss 开发案例”并将其拖到可以显示全部文字的大小

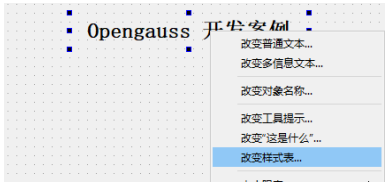


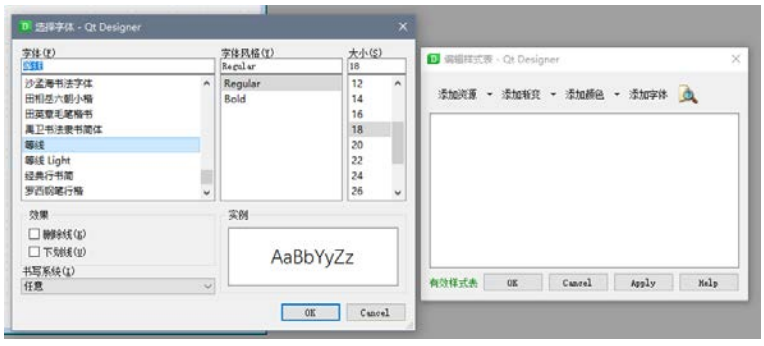
这时如果要修改大小，可以右键点击“改变多信息文本”可以修改文本大小并进行格式编辑

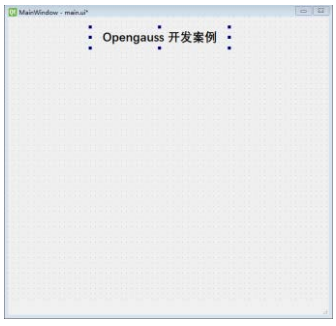




如果要修改字体，则右键点击更改样式，点击添加字体即可。（几乎所有的控件都能进行这样的操作）更改后如右图所示

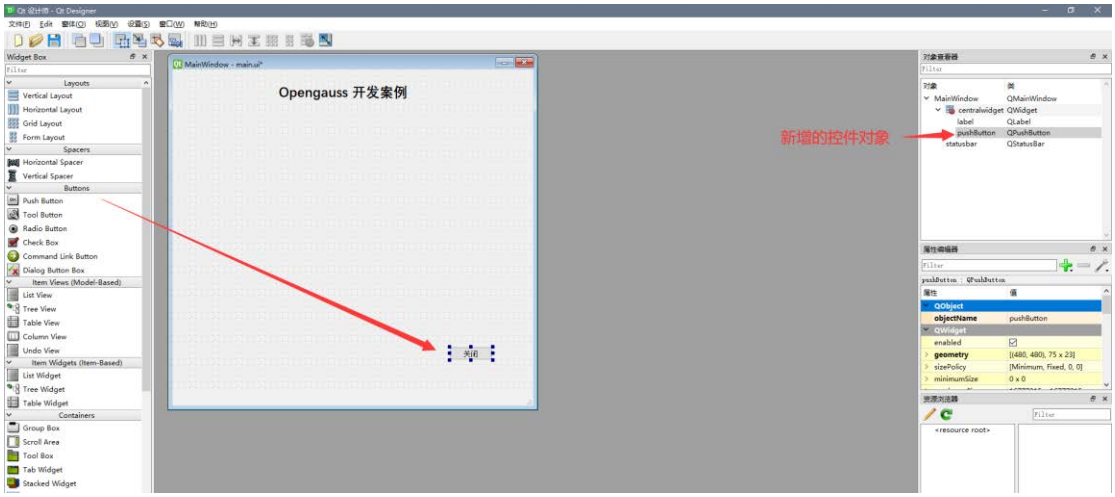






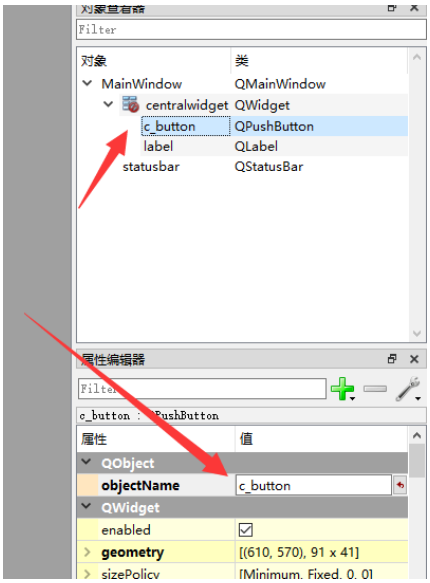
二. PushButton 控件

1. 在左边工具栏找到 PushButton 控件，将其拖入主窗口中，双击修改 Button 的内容为“关闭”，结果如下图所示：



可以根据自己的想法，更改 PushButton 内的文本字体与大小，已经 PushButton 本身的大小

2. 在对象查看器找到 PushButton，双击可以进行编辑，这里我们将其更改为“c_button”，代表该按钮的作用是关闭窗口，



接下来在 Pycharm 中我们就可以通过 self.ui 主窗口对象来调用它的成员变量（c_button 对象）来实现关闭窗口的操作。

3. 保存 main.ui，返回到 Pycharm 中， 在

class Stats 的构造函数 def __init__(self):加入下述语句

```
self.ui.c_button.clicked.connect(self.close_main_window)#按钮按下 执行函数
self.close_main_window
```

并加入类函数：

```
def close_main_window(self):#关闭窗口
    self.ui.close()
```

如下图所示：

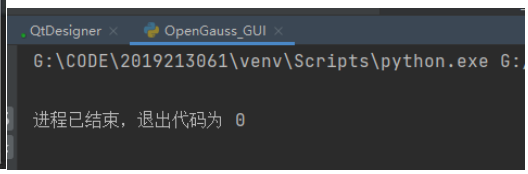
```python
class Stats:

    def __init__(self):
        # 从文件中加载UI定义

        # 从 UI 定义中动态 创建一个相应的窗口对象
        # 注意：里面的控件对象也成为窗口对象的属性了
        # 比如 self.ui.button , self.ui.textEdit
        self.ui = QUiLoader().load('main.ui')
        self.ui.setWindowTitle('你的学号')#设置窗口名称
        self.ui.c_button.clicked.connect(self.close_main_window)#按钮按下 执行函数self.close_main_window

    def close_main_window(self):#关闭窗口
        self.ui.close()
```

关于添加的内容：

（1）self.ui.c_button.clicked.connect(self.close_main_window) 这段可以理解为调用了 ui 中的 c_button 对象启用了一个监听函数，其参数是一个函数名，这里我们填入的参数正是下面我们定义的关闭主窗口的类函数。

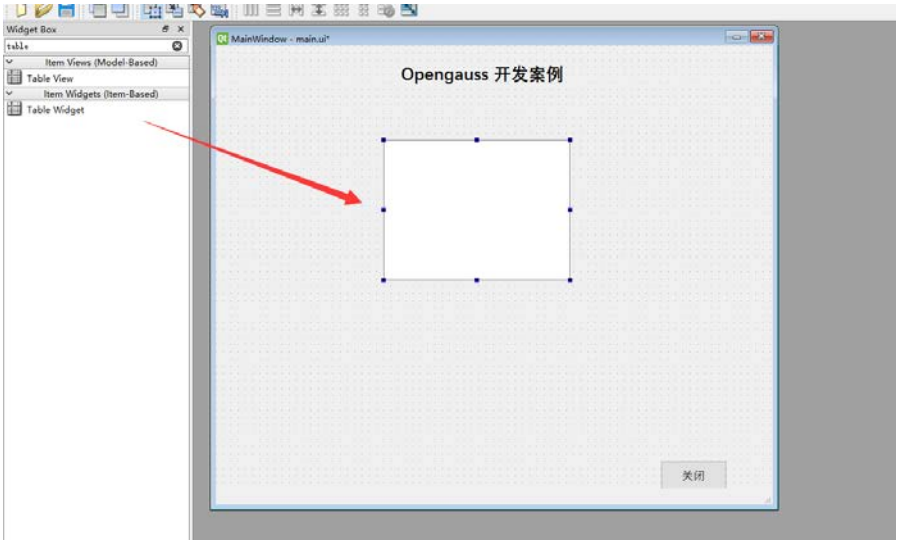
（2）close（）为窗口类的类函数，调用后关闭该窗口，这里 ui.close() 就是只关闭窗口 ui，如果还有其他窗口则不会受到影响

右键运行，可以看到我们刚才添加的 label 以及关闭按钮，按下“关闭后窗口消失”，程序运行结束

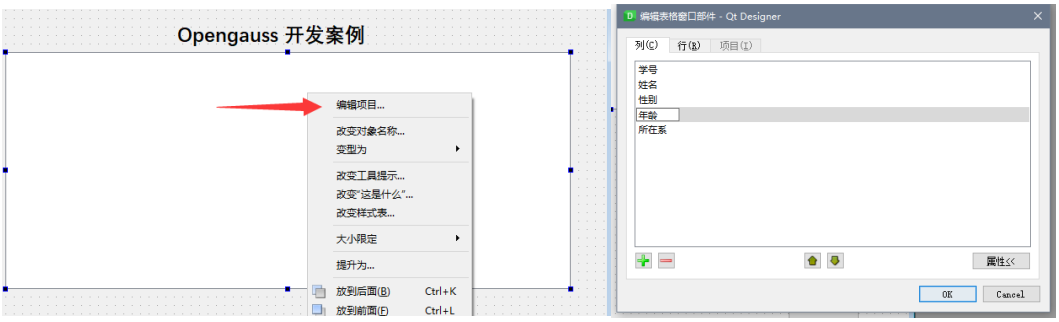


三. 添加表格控件

1. 返回 designer 在工具栏搜索 table，选择 Table Widget 控件

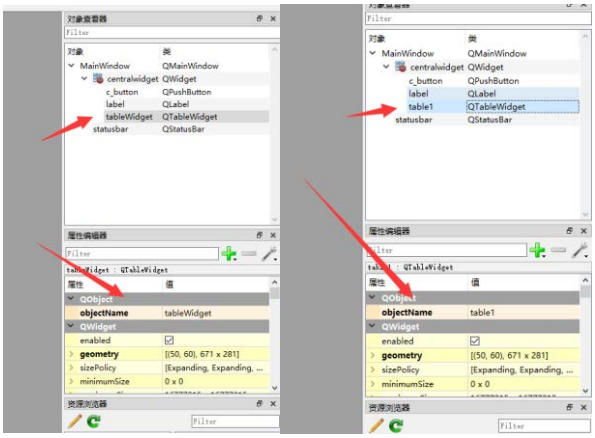


将其调整为合适的大小，右键选择“编辑项目”，即可添加行和列



（右键选择“改变样式”也可修改单元内文本的字体和大小），结果如下：

2. 修改表格对象名，在对象查看器中找到“tableWidget”双击将其修改为“table1”，便于我我们待会再程序内对表格进行操作



三. 数据库中表格数据的加载

保存 main.ui,返回 Pycharm，在 class Stats 下补充类函数:

```
def add_cell(self,row,column,txt): #修改单个单元格的内容
    item = QTableWidgetItem()
    item.setText(txt)
    self.ui.table1.setItem(row,column,item)

def init_table(self,tab_list):
    self.row=int(len(tab_list))#获得行数
    self.ui.table1.setRowCount(self.row) #初始化行数
    for i in range(self.row):
        temp = list(tab_list[i])
        for j in range(5): #循环写入 5 个属性
            self.add_cell(i,j,str(temp[j]))
```

结果如图所示:

```
class Stats:

    def __init__(self):
        # 从文件中加载UI定义

        # 从 UI 定义中动态 创建一个相应的窗口对象
        # 注意，里面的控件对象也成为窗口对象的属性了
        # 比如 self.ui.button , self.ui.textEdit
        self.ui = QUiLoader().load('main.ui')
        self.ui.setWindowTitle('你的学号')#设置窗口名称
        self.ui.c_button.clicked.connect(self.close_main_window)#按钮按下 执行函数self.close_main_window

    def close_main_window(self):#关闭窗口
        self.ui.close()


    def add_cell(self,row,column,txt): #修改单个单元格的内容
        item = QTableWidgetItem()
        item.setText(txt)
        self.ui.table1.setItem(row,column,item)

    def init_table(self,tab_list):
        self.row=int(len(tab_list))
        self.ui.table1.setRowCount(self.row) #初始化行数
        for i in range(self.row):
            temp = list(tab_list[i])
            for j in range(5): #循环写入5个属性
                self.add_cell(i,j,str(temp[j]))
```

这里类函数 init_table 需要输入含有每一行数据的列表作为参数，然后通过循环写入整个表的数据，因此我们需要先用 SQL 语句获取 Student 表的数据，找到下图 else：所在的位置

```
#程序从这里执行
app = QApplication([])
stats = Stats()
msgBox = QMessageBox()
try:
    conn = create_conn()
except Exception as e:
    msgBox.about(stats.ui, '提示窗口', str(e))

else:
    msgBox.about(stats.ui, '提示窗口', '数据库连接成功  ')#弹窗提示

    stats.ui.show()  # 界面显示

sys.exit(app.exec_())#事件处理循环  要不然程序一闪而过  死循环
```

在 msgBox.about(stats.ui, '提示窗口', '数据库连接成功 ')后加入下述语句

```
######      获得 student 表的全部内容        ######
cur = conn.cursor() #创建光标
cur.execute("select * from student;") #执行 SQL 指令
results = cur.fetchall()#获取所有结果
conn.commit()
######        将数据写入 UI                  ######
stats.init_table(results)
```

结果如下：

```
else:
    msgBox.about(stats.ui, '提示窗口', '数据库连接成功  ')#弹窗提示

    ######      获得student表的全部内容        ######
    cur = conn.cursor() #创建光标
    cur.execute("select * from student;") #执行SQL指令
    results = cur.fetchall()#获取所有结果
    conn.commit()
    ######        将数据写入UI                  ######
    stats.init_table(results)

    stats.ui.show()  # 界面显示
```

这里的 conn 和 cur 相当于全局变量，可以在类函数里面直接调用（后面会用到）

同时，创建了 conn 与 cur，我们需要在程序结束时释放掉资源，所以我们要在关闭主窗口的类函数下加入这两句，cur.close()和 conn.close()。如下图所示：

```
def close_main_window(self):#关闭窗口
    self.ui.close()
    cur.close()
    conn.close()
```

右键运行，可以看到，数据库 student 表中的所有数据均已显示在主窗口的表格中：


2019213061
Opengauss 开发案例
学号 姓名 性别 年龄 所在系
1 95010 陈东 男 18 IS
2 95001 张三 男 20 CS
3 95003 王五 男 18 MA
4 95004 马六 女 19 CS
5 95005 苏三 女 19 IS
6 95006 刘七 女 18 IS
7 95007 刘三姐 女 22 IS
8 95008 欧阳锋 男 23 MA
关闭

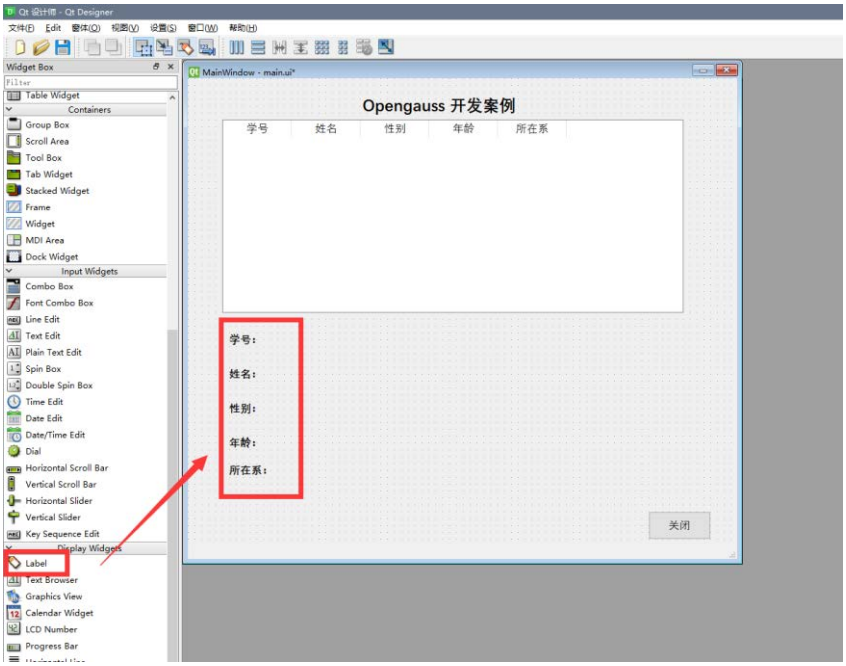
Pycharm+OpenGuass 开发教程（三）

合肥工业大学 计算机与信息学院 张国富

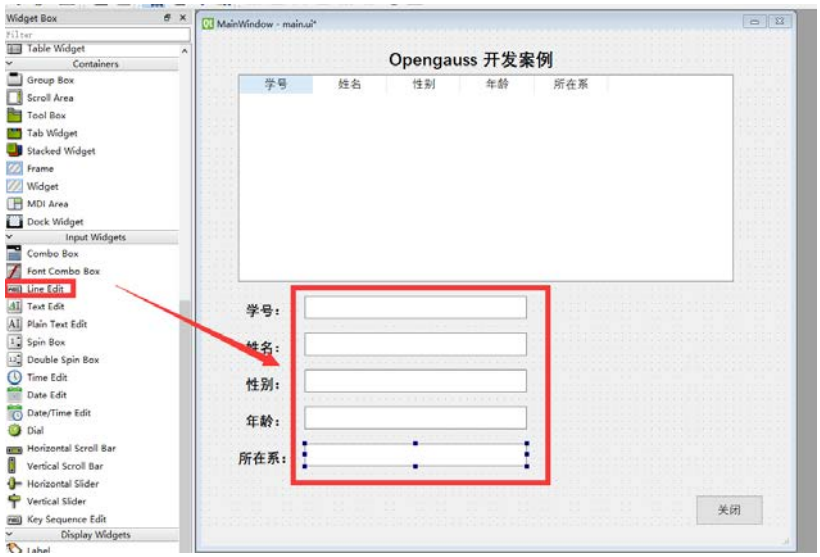
zgf@hfut.edu.cn

一．界面控件的绘制

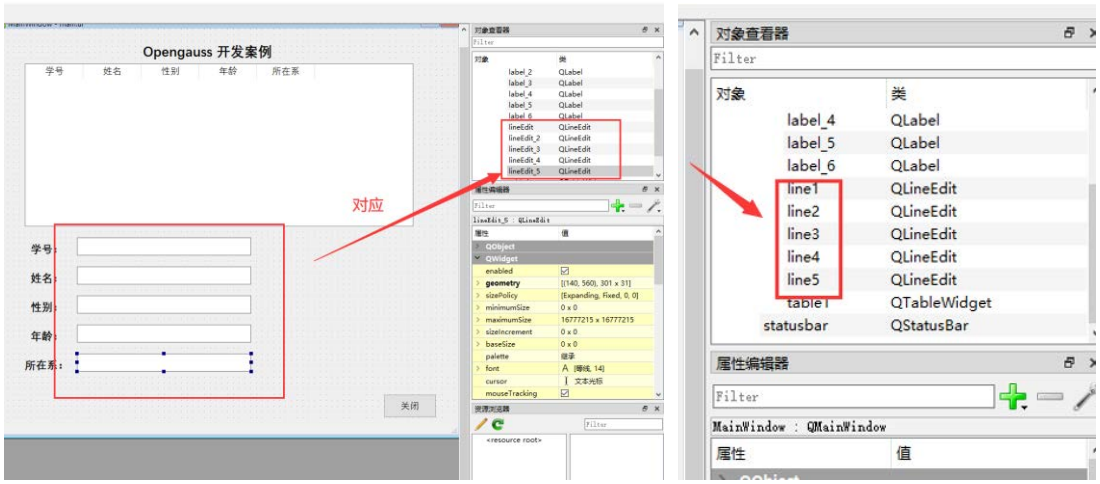
选择“工具”→“External Tools”→“QtDesigner”启动 designer，在左边工具栏选择 label 控件，双击将控件名改为“学号”，用于之前相同的方法修改文本至合适的大小，依次绘制“姓名”、“年龄”、“性别”、“所在系”。完成后点击保存。



2. 从工具箱选择“line Edit”控件，在“学号”后面绘制该控件，与先前一样右键后选择“更改样式表”，可以改变其显示的字体与大小。完成后复制控件，依次粘贴在“姓名”、“年龄”、“性别”、“所在系”之后：



3.观察右边的对象查看器，可以看到多出来了 5 个 LineEdit 对象，双击后可以进行编辑，将其依次修改为 line1，line2，line3，line4，line5：



点击保存:

二．实现表格中选中行的数据显示

1.切换回 Pycharm 中,在 Stats 类构造函数下添加点击表格的事件函数:

```
self.ui.table1.itemClicked.connect(self.show_data)
```

再在类中添加类函数 show_data

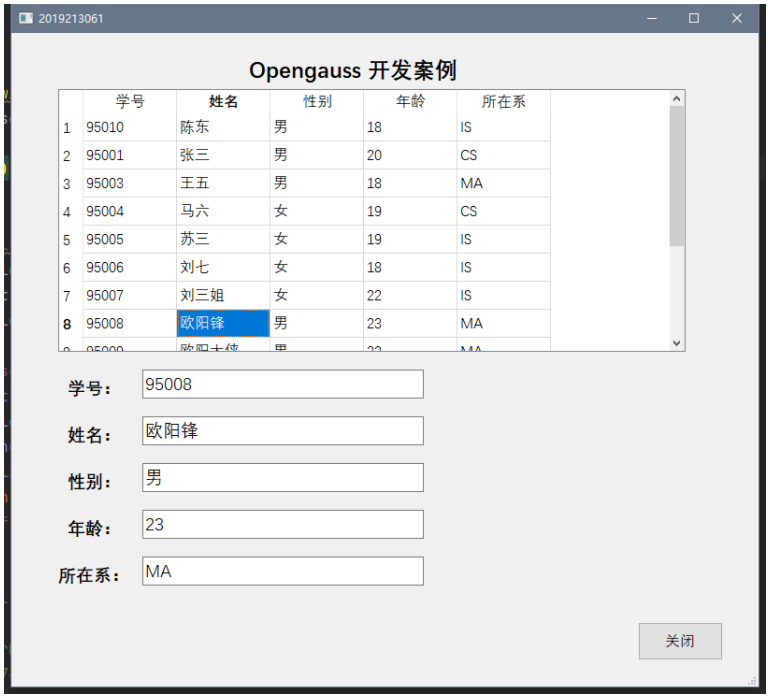
```
def show_data(self, Item=None):
    # 如果单元格对象为空
    if Item is None:
        return
    else:
        row = Item.row()  # 获取行数
        sno = self.ui.table1.item(row,0).text()#获取该行内容
        sname = self.ui.table1.item(row, 1).text()
        ssex = self.ui.table1.item(row, 2).text()
        ssage= self.ui.table1.item(row, 3).text()
        sdept  = self.ui.table1.item(row, 4).text()
        self.ui.line1.setText(sno) #显示到 5 个编辑框中
        self.ui.line2.setText(sname)
        self.ui.line3.setText(ssex)
        self.ui.line4.setText(ssage)
        self.ui.line5.setText(sdept)
```

添加后的效果如下图：

```python
class Stats:

    def __init__(self):
        # 从文件中加载UI定义

        # 从 UI 定义中动态 创建一个相应的窗口对象
        # 注意：里面的控件对象也成为窗口对象的属性了
        # 比如 self.ui.button , self.ui.textEdit
        self.ui = QUiLoader().load('main.ui')
        self.ui.setWindowTitle('2019213061')#设置窗口名称
        self.ui.c_button.clicked.connect(self.close_main_window)#按钮按下 执行函数self.close_main_window
        self.ui.table1.itemClicked.connect(self.show_data)

    def show_data(self, Item=None):
        # 如果单元格对象为空
        if Item is None:
            return
        else:
            row = Item.row()  # 获取行数
            sno = self.ui.table1.item(row,0).text()#获取该行内容
            sname = self.ui.table1.item(row, 1).text()
            ssex = self.ui.table1.item(row, 2).text()
            ssage= self.ui.table1.item(row, 3).text()
            sdept  = self.ui.table1.item(row, 4).text()
            self.ui.line1.setText(sno) #显示到5个编辑框中
            self.ui.line2.setText(sname)
            self.ui.line3.setText(ssex)
            self.ui.line4.setText(ssage)
            self.ui.line5.setText(sdept)
```

这样在点击表格中的单元格时，会调用 show_data 函数并以当前单元格信息作为参数，通过获取当前的行数，来得到当前行内的所有信息，再显示在 line1-line5 这 5 个控件上

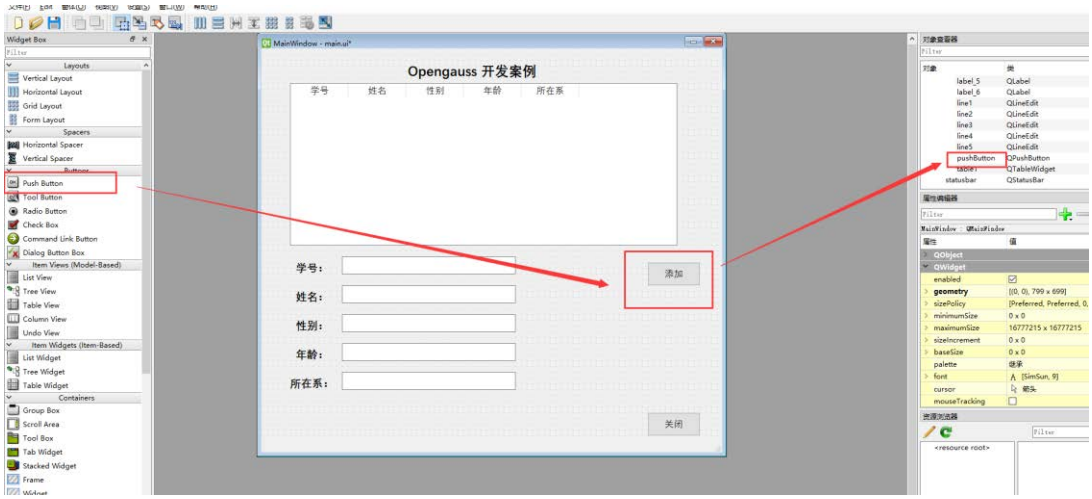
2. 右键后选择“运行”结果如下：

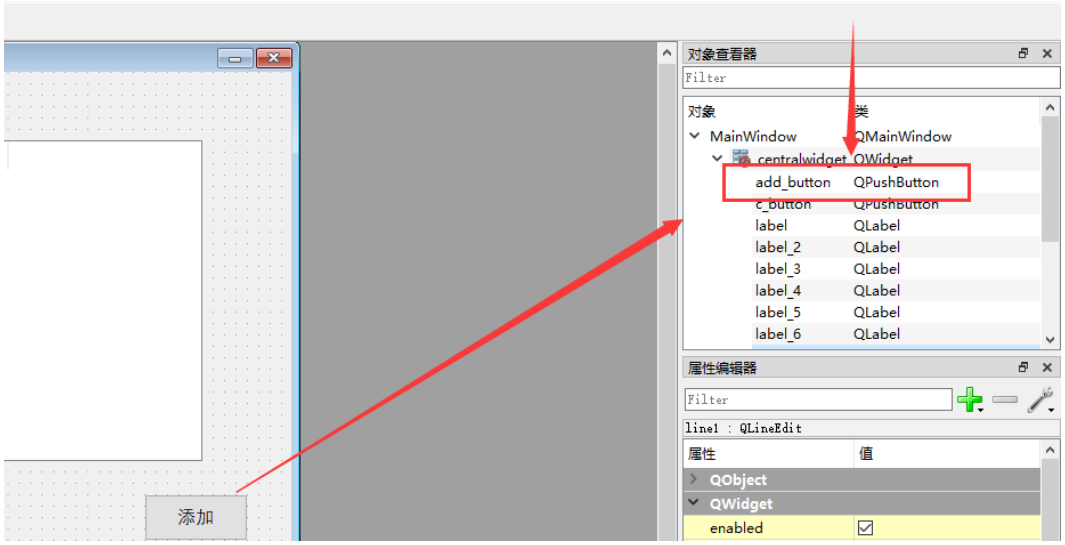


三. 实现数据的添加

1. 回到 Designer 中，在工具栏中找到 pushbutton，将其绘制到主窗口中，双击进行编辑，将其更改为“添加”，右键选择“更改样式”将文本更改到合适的字体和大小，同时在对象查看器中，将此时刚出现的对象名“pushbutton”修

改为“add_button”。





2.切换回 Pycharm 中,在 Stats 类构造函数下添加点击 button 的事件函数:

```
self.ui.add_button.clicked.connect(self.add_Line)
```

并添加类函数 add_Line:

```
def add_Line(self):  # 关闭窗口
    sno = self.ui.line1.text()  # 获取输入框的内容
    sname = self.ui.line2.text()
    ssex = self.ui.line3.text()
    sage = self.ui.line4.text()
    sdept = self.ui.line5.text()

    cur.execute(f'select * from student where sno = \'%s\' ;' % (sno))
    result = cur.fetchall()
    if result != []:
        QMessageBox.about(self.ui, '提示', '该学号已存在，无法添加')
        return 0
    self.ui.table1.insertRow(self.row)  # 增加新行
    self.add_cell(self.row, 0, sno)
    self.add_cell(self.row, 1, sname)
    self.add_cell(self.row, 2, ssex)
```

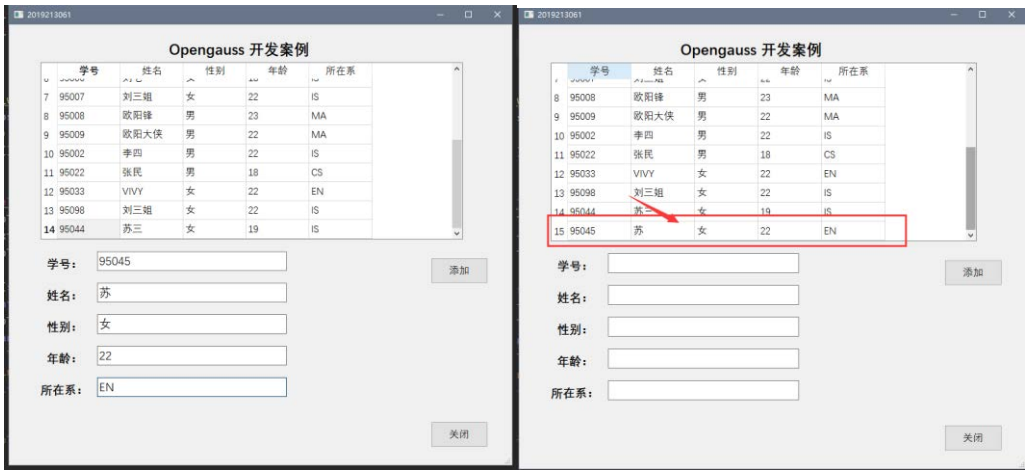
```
    self.add_cell(self.row, 3, sage)
    self.add_cell(self.row, 4, sdept)
    self.row += 1
    val = (sno, sname, ssex, int(sage), sdept)
    cur.execute("insert into student(Sno, Sname, Ssex, Sage, Sdept) VALUES
(%s,%s,%s,%s,%s);", val)
    conn.commit()
    self.ui.line1.clear()  # 清理编辑框的内容
    self.ui.line2.clear()
    self.ui.line3.clear()
    self.ui.line4.clear()
    self.ui.line5.clear()
    QMessageBox.about(self.ui, '提示', '添加成功')
```

添加后结果如下：

```
class Stats:

    def __init__(self):
        # 从文件中加载UI定义

        # 从 UI 定义中动态 创建一个相应的窗口对象
        # 注意：里面的控件对象也成为窗口对象的属性了
        # 比如 self.ui.button , self.ui.textEdit
        self.ui = QUiLoader().load('main.ui')
        self.ui.setWindowTitle('2019213061')#设置窗口名称
        self.ui.c_button.clicked.connect(self.close_main_window)#按钮按下 执行函数self.close_main_window
        self.ui.table1.itemClicked.connect(self.show_data)
        self.ui.add_button.clicked.connect(self.add_Line)

    def add_Line(self):  # 关闭窗口
        sno = self.ui.line1.text()  # 获取输入框的内容
        sname = self.ui.line2.text()
        ssex = self.ui.line3.text()
        sage = self.ui.line4.text()
        sdept = self.ui.line5.text()

        cur.execute(f'select * from student where sno = \'%s\' ;' % (sno))
        result = cur.fetchall()
        if result != []:
            QMessageBox.about(self.ui, '提示', '该学号已存在，无法添加')
            return 0
        self.ui.table1.insertRow(self.row)  # 增加新行
        self.add_cell(self.row, 0, sno)
```

这样在点击“添加”按键时，会自动调用 add_Line 类函数将编辑框的内容显示出来，并用 SQL 语句将数据添加进数据库，这里需要事先判断主码是否重复，并添加相应的提示框。

3. 右键点击运行：



可以看到上图已经添加成功。

Pycharm+OpenGuass 开发教程（四）

合肥工业大学 计算机与信息学院 张国富

zgf@hfut.edu.cn

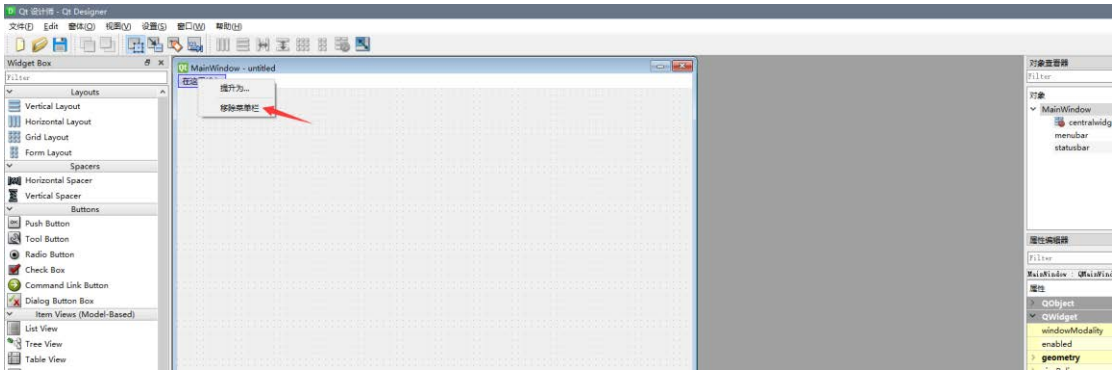
一．用户表格的创建

打开 Opengauss 虚拟机，启动 opengauss 数据库，以管理员身份运行 Data Studio，连接上 Opengauss 虚拟机，连接上 school 数据库，选中 school 数据库，鼠标右键，打开新的终端，输入如下 SQL 语言创建：用户表。

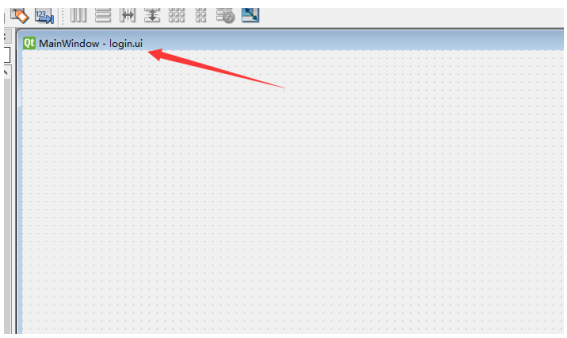
```
create table Userinformation(Uname varchar(50), Upassword varchar(50) not null, Primary key (Uname))
```

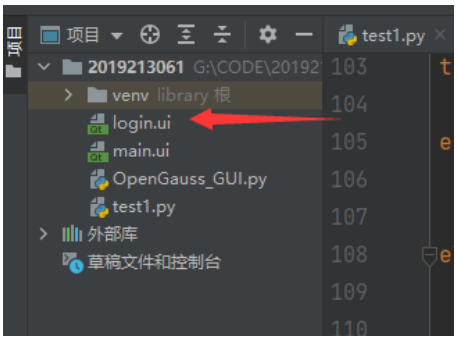
二．登录对话框的创建

选择“工具”→“External Tools”→“QtDesigner”启动 designer，关闭当前窗口，点击“新建”，创建新的窗口，并将左上角的菜单删掉，如下图所示：



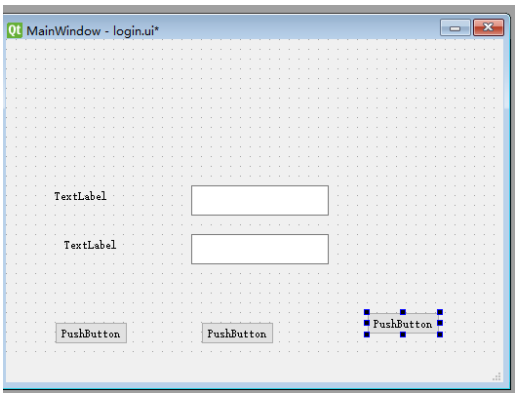
随后保存该窗口，命名为 login.ui：





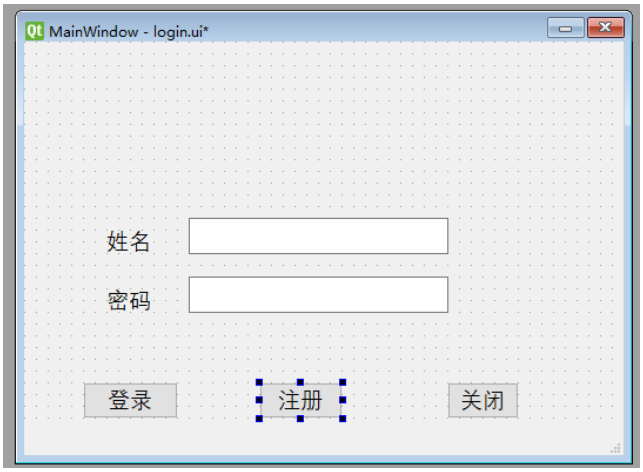
二．登录界面的绘制

1. 在 designer 中，选择 Lable，Line Edit,PushButton 这三个控件，将其拖入当前窗口，如下图所示：

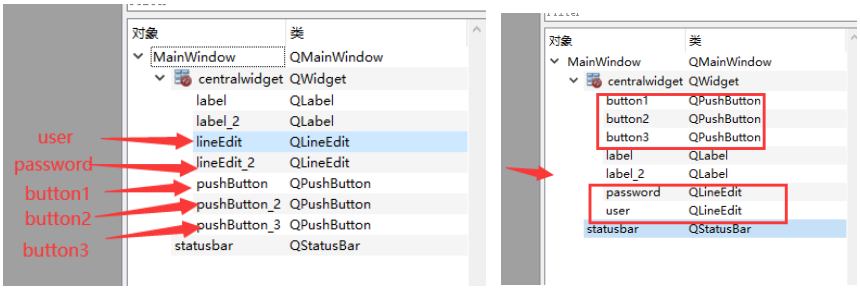


2. 双击 label 和 button 控件编辑其内容，然后修改控件文本的至合适的大

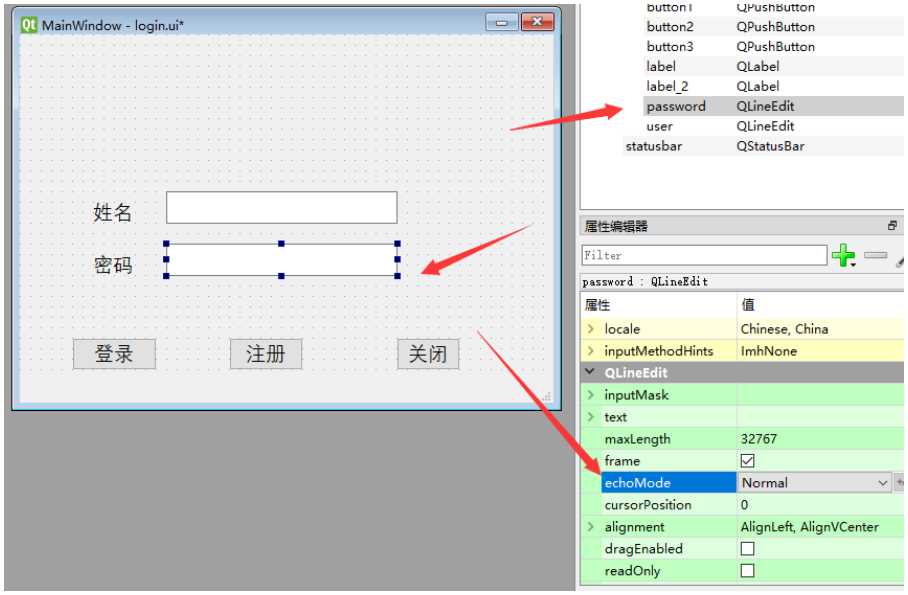
小：



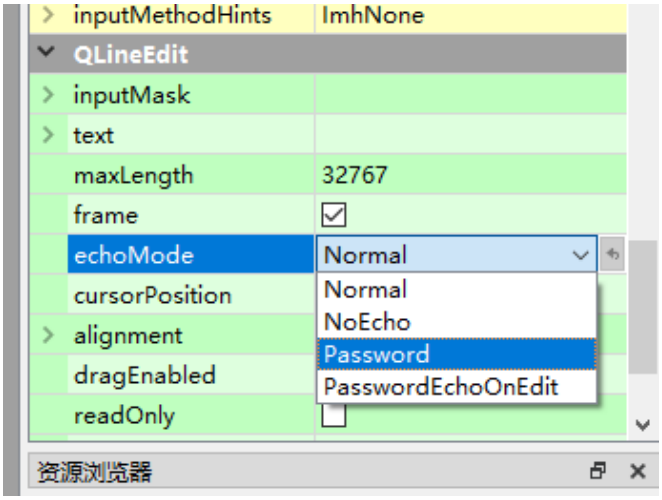
3. 接下来在对象查看器中，双击修改控件对象变量名称，将 lineEdit，lineEdit_2 修改为 user，和 password，3 个 Pushbutton 修改为 button1，button2，button3。方便我们接下来在程序中调用这些控件。



4. 选中 password，可以看到它对应的是第二行的输入栏，由于是输入密码，我们需要对其属性就行修改，让其不可见，在对象查看器下面的属性编辑器中，找到 echoMode 选项：



更改其属性为 password，这样密码在输入是就不是直接可见的了：



二．界面的导入与关闭按钮的实现

返回 Pycharm，在 Status 类中导入新建的 login.ui，并添加关闭按键响应函数。将下列代码放入 Status 类的构造函数中：

```
self.login = QUiLoader().load('G:\\CODE\\2019213061\\login.ui')
self.login.button3.clicked.connect(self.close_login_window)
```

然后加入关闭登录窗口的类函数 close_login_window：

```
def close_login_window(self):#关闭窗口
    self.login.close()
    cur.close()
    conn.close()
```

如下图所示：（要注意缩进）

```
7   class Stats:
8
9       def __init__(self):
10          # 从文件中加载UI定义
11
12          # 从 UI 定义中动态 创建一个相应的窗口对象
13          # 注意：里面的控件对象也成为窗口对象的属性了
14          # 比如 self.ui.button , self.ui.textEdit
15          self.ui = QUiLoader().load('main.ui')
16          self.ui.setWindowTitle('2019213061')#设置窗口名称
17          self.ui.c_button.clicked.connect(self.close_main_window)#按钮按下 执行函数self.close_main_window
18          self.ui.table1.itemClicked.connect(self.show_data)
19          self.ui.add_button.clicked.connect(self.add_Line)
20          # 登录窗口
21          self.login = QUiLoader().load('G:\\python_study\\try4.ui')
22          self.login.Button3.clicked.connect(self.close_login_window)
23
24      def close_login_window(self):#关闭窗口
25          self.login.close()
26          cur.close()
27          conn.close()
28
29      def add_Line(self):  #添加一行数据
```

然后在下面的执行程序中，将 status.ui.show()，改为 status.login.show()，这样程序执行时就可以看到登录界面。

```
else:
    msgBox.about(stats.ui, '提示窗口', '数据库连接成功   ')#弹窗提示

    ######      获得student表的全部内容        ######
    cur = conn.cursor() #创建光标
    cur.execute("select * from student;") #执行SQL指令
    results = cur.fetchall()#获取所有结果
    conn.commit()
    ######        将数据写入UI                ######
    stats.init_table(results)

    stats.ui.show()  # 界面显示
```

```
    stats.init_table(results)

    stats.login.show()  # 界面显示
```

login Stats

执行结果：

这个时候是没法进入主界面的，因为并没有显示 ui 的代码，因此我们需要添加一个登录函数，在密码正确时，关闭 login 窗口进入主窗口 ui。

三．登录按键与注册按键的实现

与上述同理，先加入登录按键和注册按键的事件函数，再加入相应的登录函数，在构造函数下加入如下代码：

```
self.login.button1.clicked.connect(self.Sign_in)
self.login.button2.clicked.connect(self.Register)
```

然后加入对应的类函数 Sign_in 与 Register：

```
def Sign_in(self):
    user = self.login.user.text()  # 获取输入框的内容
    password = self.login.password.text()
```

```
    val = (user, password)
    cur.execute('select * from userinformation where uname = %s and upassword
= %s;', val)
    result = cur.fetchall()  # 获取所有结果
    conn.commit()
    if result == []:
        QMessageBox.about(self.login,'提示','密码错误或用户不存在')
    else:
        QMessageBox.about(self.login, '提示', '登录成功')
        self.login.close()
        self.ui.show()


def Register(self):
    user = self.login.user.text()  # 获取输入框的内容
    password = self.login.password.text()
    cur.execute(f'select * from userinformation where uname = \'%s\' ;'%(user))
    result = cur.fetchall()  # 获取所有结果
    conn.commit()
    val = (user,password)
    if result != []:
        QMessageBox.about(self.login, '提示', '用户已存在')
    else:
        cur.execute("insert into userinformation VALUES (%s,%s);", val)
        conn.commit()
        QMessageBox.about(self.login, '提示', '注册成功')
```

（1)关于登录函数,先获取当前输入框内容,然后用 SQL 语句在 userinformation 中查找是否有对应的值，如果存在，则返回对应的值，如果不存在，则返回空列表“[]”借此判断是否存在该用户，如果用户存在，密码正确则，关闭 login 显示主窗口 ui。

（2)关于注册函数,先获取当前输入框内容,然后用 SQL 语句在 userinformation 中查找是否有对应的值，如果存在的话则提示已有用户存在，如果返回空列表，则证明用户不存在，将输入框的用户名与密码写入数据库。

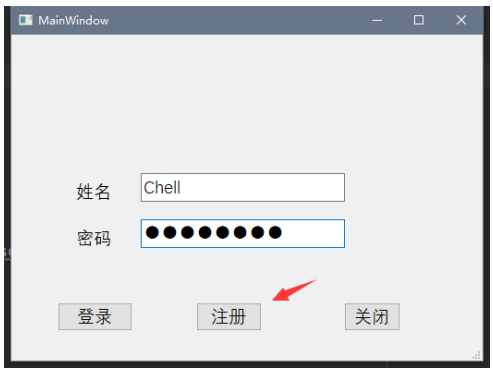
添加后效果如下：

```
49
50      def Sign_in(self):
51          user = self.login.user.text()  # 获取输入框的内容
52          password = self.login.password.text()
53          val = (user, password)
54          cur.execute('select * from userinformation where uname = %s and upassword = %s;', val)
55          result = cur.fetchall()  # 获取所有结果
56          conn.commit()
57          if result == []:
58              QMessageBox.about(self.login,'提示','密码错误或用户不存在')
59          else:
60              QMessageBox.about(self.login, '提示', '登录成功')
61              self.login.close()
62              self.ui.show()
63
64
65      def Register(self):
66          user = self.login.user.text()  # 获取输入框的内容
67          password = self.login.password.text()
68          cur.execute(f'select * from userinformation where uname = \'%s\' ;'%(user))
69          result = cur.fetchall()  # 获取所有结果
70          conn.commit()
71          if result != []:
72              QMessageBox.about(self.login, '提示', '用户已存在')
73          else:
74              cur.execute("insert into userinformation VALUES (%s,%s);", val)
75              conn.commit()
76              QMessageBox.about(self.login, '提示', '注册成功')
77
78
79
80      def add_cell(self,row,column,txt): #修改单个单元格的内容
```

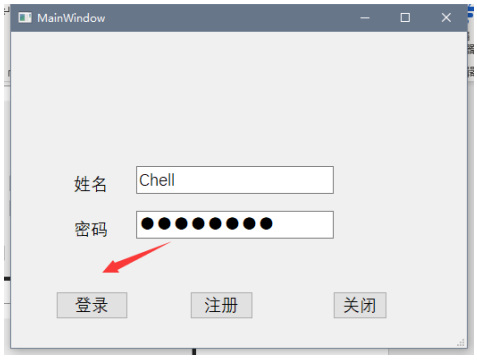
注意缩进

运行当前程序：

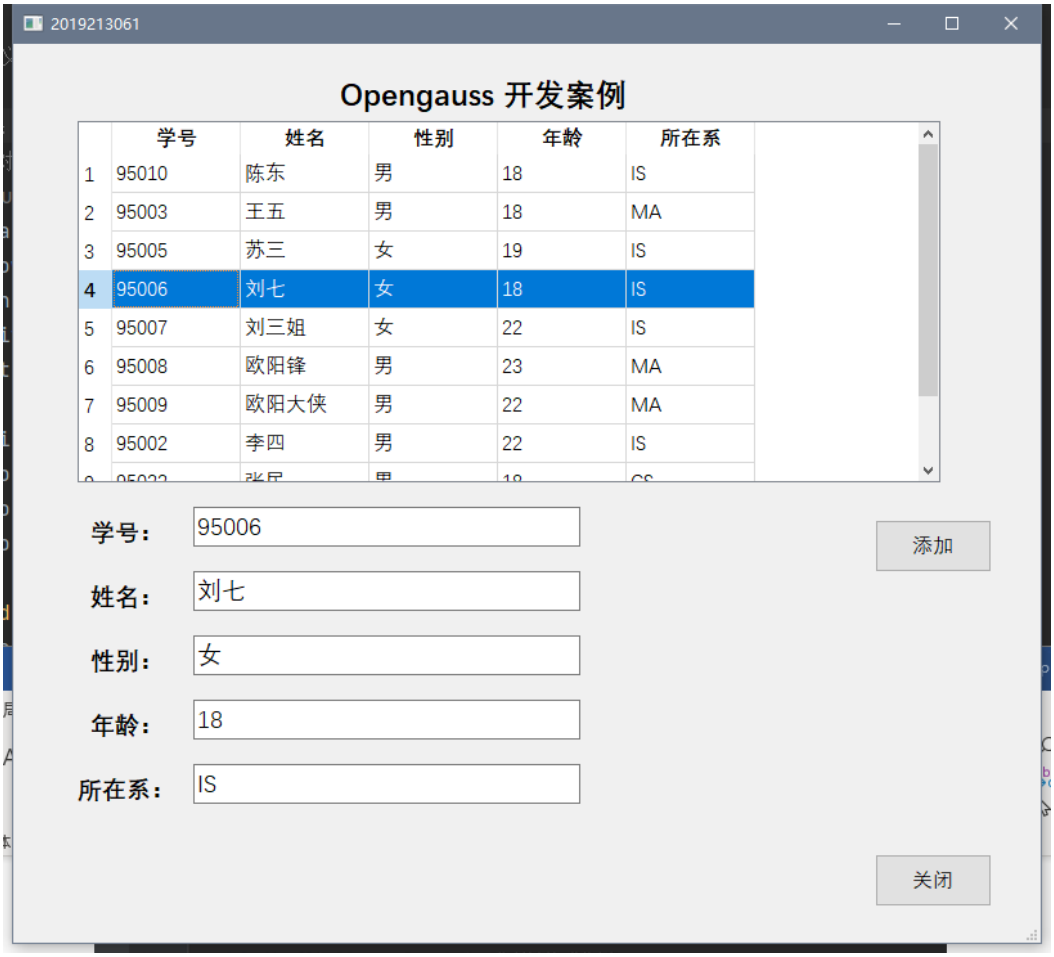
注册：



登录：



随后会进入主窗口：

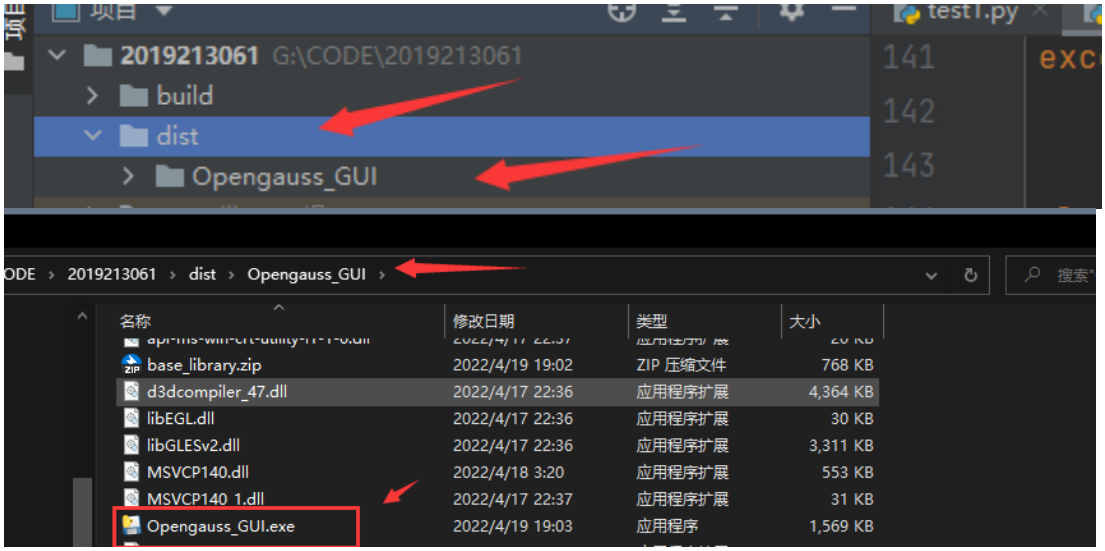


四．生成 exe 执行文件：

在终端输入下述代码 ：

pyinstaller Opengauss_GUI.py --noconsole --hidden-import PySide2.QtXml

运行完成后会在项目目录生成 dist 文件夹,在里面可以找到我们生成的 exe 文件。



Pycharm+OpenGuass 开发教程（五）

合肥工业大学 计算机与信息学院 张国富

zgf@hfut.edu.cn

界面优化

一.修改生成程序的图标：

1.先准备好一个.ico 文件

2.导入相关库

```
from PySide2.QtWidgets import QApplication,QMessageBox,QTableWidgetItem
from PySide2.QtUiTools import QUiLoader
from PySide2.QtGui import QIcon
import psycopg2
import sys
# 导入相关库
# 定义一个类专门处理GUI
```

3.添加如下代码，将其引入程序：

app.setWindowIcon(QIcon('G:\\CODE\\2019213061\\logo.ico'))

```
#程序从这里执行
app = QApplication([])
stats = Stats()
msgBox = QMessageBox()
```

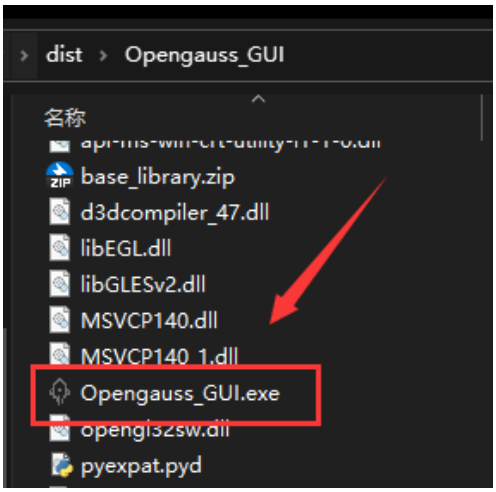
```
135
136 #程序从这里执行
137 app = QApplication([])
138 app.setWindowIcon(QIcon('G:\\CODE\\2019213061\\logo
139 stats = Stats()
140 msgBox = QMessageBox()
```

4.最后在生成 exe 文件引用图标

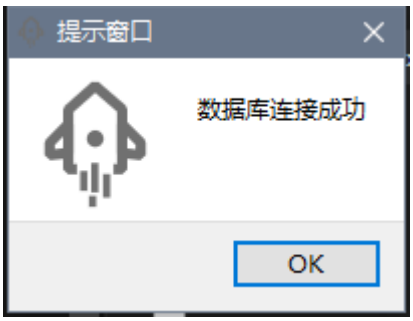
pyinstaller Opengauss_GUI.py --noconsole --hidden-import PySide2.QtXml --icon="logo.ico"

```
33789 INFO: Building COLLECT COLLECT-00.toc
35587 INFO: Building COLLECT COLLECT-00.toc completed successfully.

(venv) G:\CODE\2019213061>pyinstaller Opengauss_GUI.py --noconsole --hidden-import PySide2.QtXml --icon="logo.ico"
```

效果如下图所示：



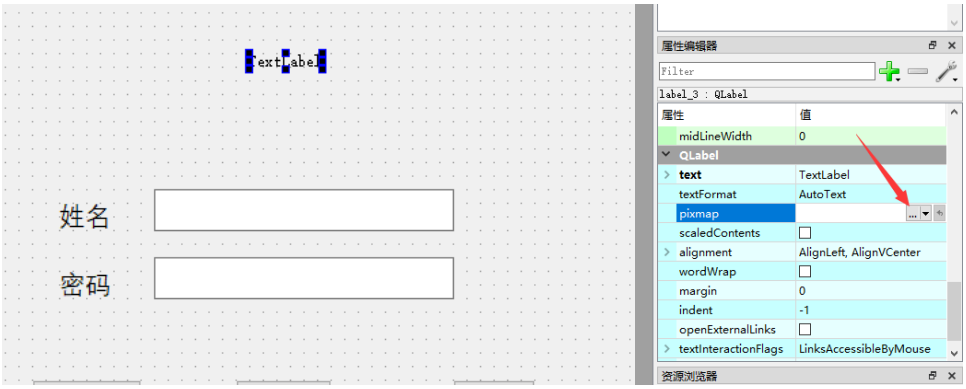
点击后可以发现弹窗现在也会拥有图标



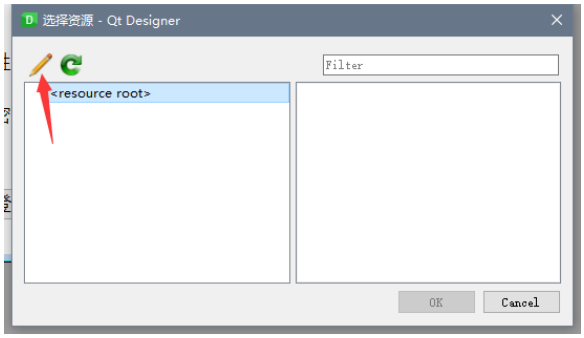
二.加入图片：

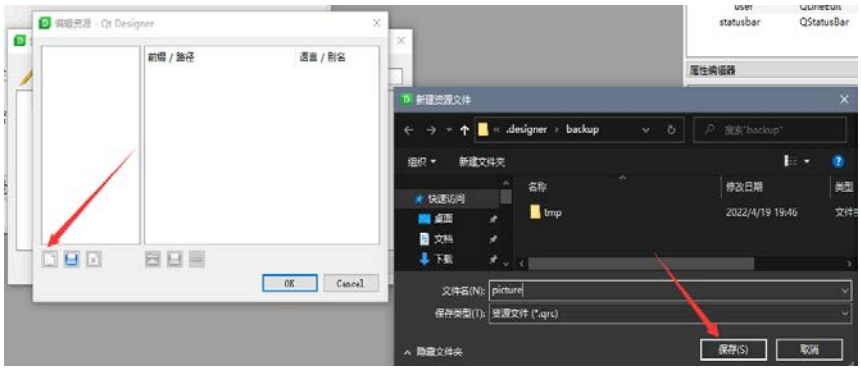
1.准备一张图片

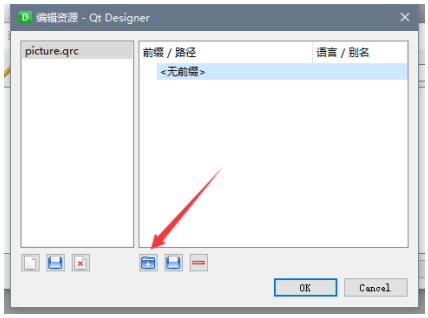
2.开启 designer，进入 login.ui，选择 label 控件拖入主窗口，在属性编辑器找到 pixmap 属性，点击浏览。



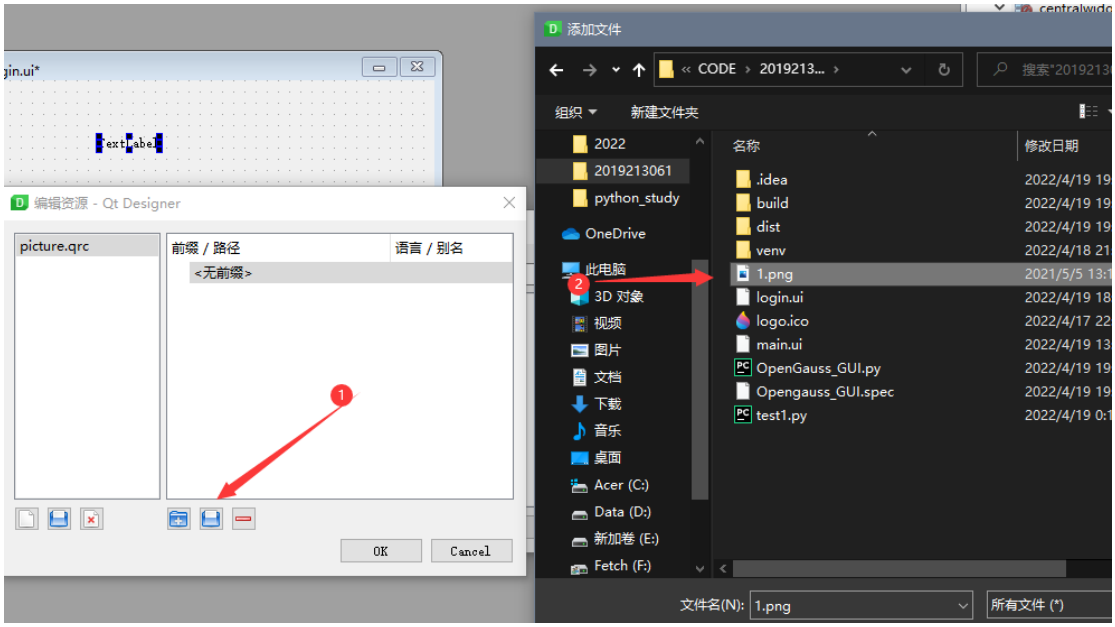
点击编辑：



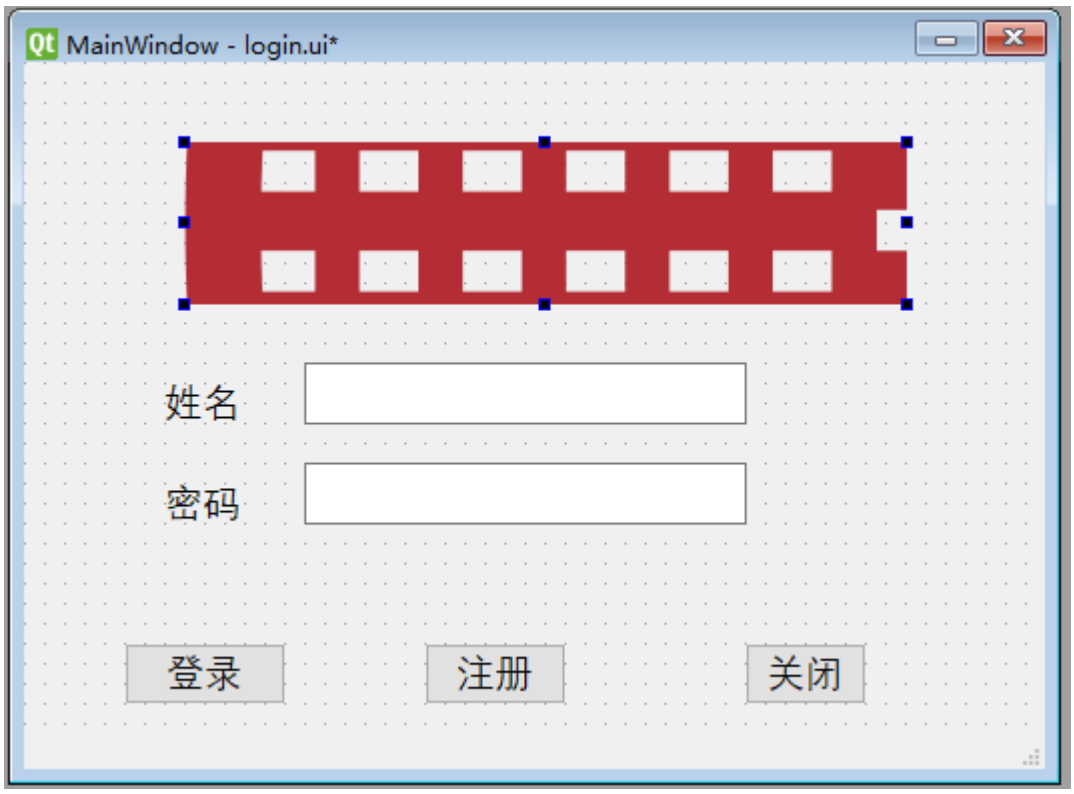




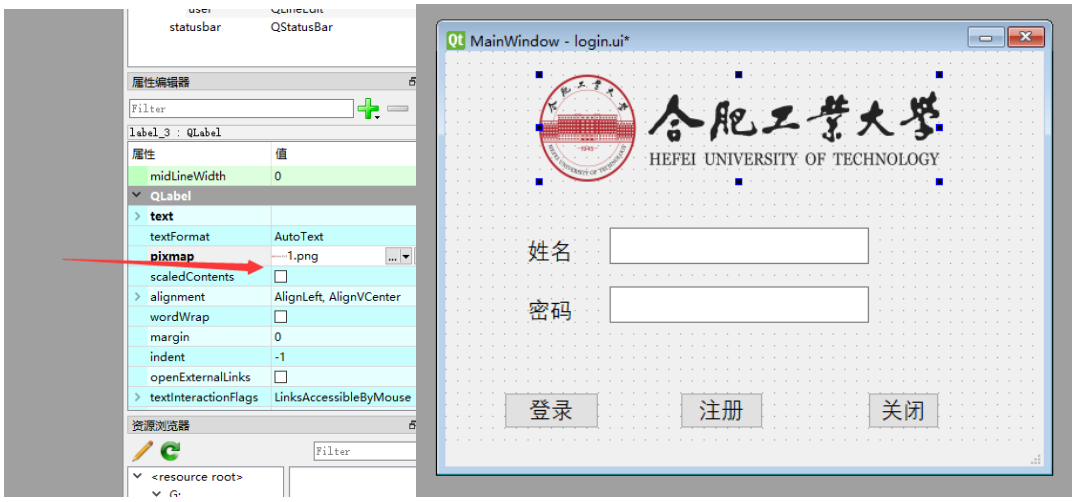
选中图片，点击 OK



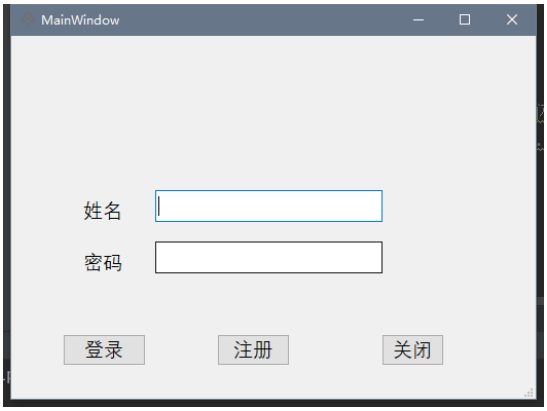
把 label 拉大可以发现，图片并没有正常显示：



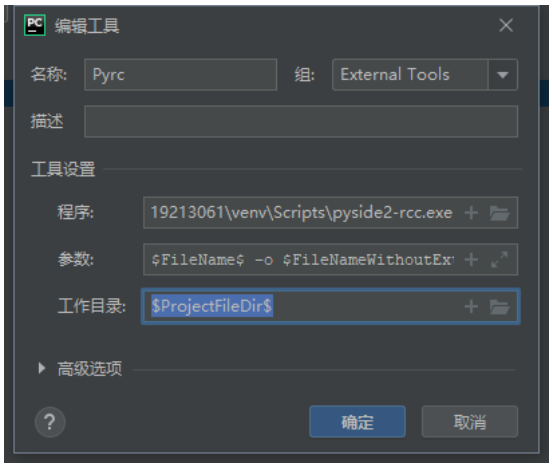
找到 scaledContents 属性，打上√后可以发现，图片正常显示:



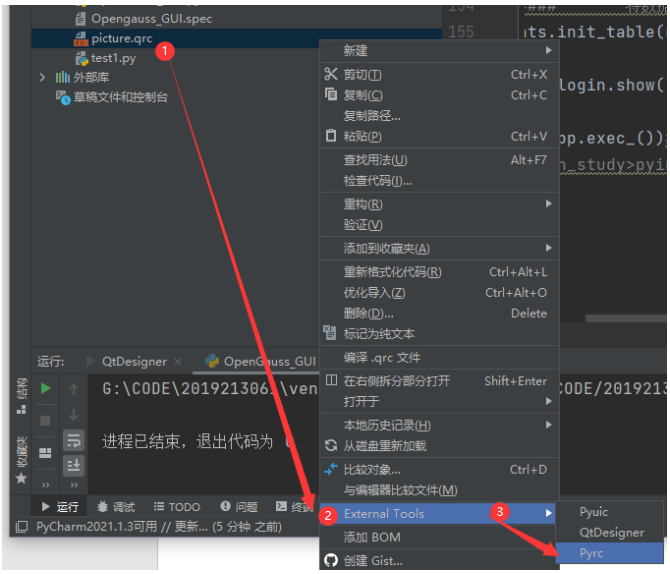
保存 login.ui 文件，返回 Pycharm，点击运行发布图片并没有显示，这是因为需要将 qc 文件变为.py 文件才能正常显示

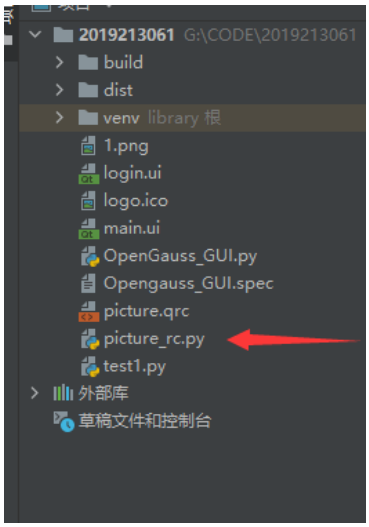


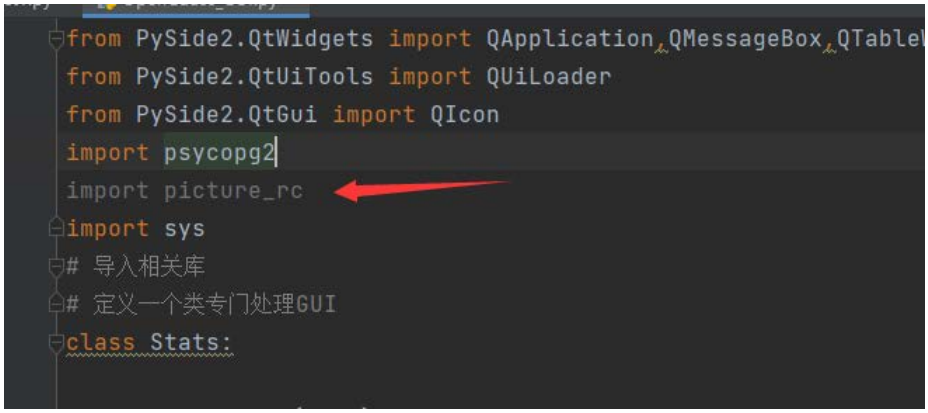
这里需要配置再为 Pycharm 配置一个“外部工具”，选择左上角“文件”→“设置”→“工具”→“ 外部工具”，
程序：G:\CODE\2019213061\venv\Scripts\pyside2-rcc.exe
参数：$FileName$ -o $FileNameWithoutExtension$_rc.py
工作目录：$ProjectFileDir$



接下来利用 Pyrc 即可将.qrc 文件转化为.py 文件，可以看到生成的 picture_rc.py 文件，然后需要将用 import 导入







再次点击运行，可以发现图片正常显示：


MainWindow
合肥工業大學
HEFEI UNIVERSITY OF TECHNOLOGY
姓名
密码
登录
注册
关闭

VS2013+Socket 开发教程-TCP 篇

合肥工业大学 计算机与信息学院 张国富

zgf@hfut.edu.cn

一．创建一个初始工程-Server 端

1. 打开 VS2013，选择“文件”→新建→项目；
2. 在“新建项目”中，左手边选择“Virtual C++”，右手边选择“MFC 应用程序”，在下面的“名称”中输入你的工程名，工程名要包含学号，例如：2015211TCPSever。注意查看工程保存的“位置”，以便后期查找这个工程。
   点击右下角“确定”。
3. 点击左手边“应用程序类型”，选择“基于对话框”；
   再点击左手边的“高级功能”，勾选“Windows 套接字”。最后点击“完成”，一个新的工程就创建好了。

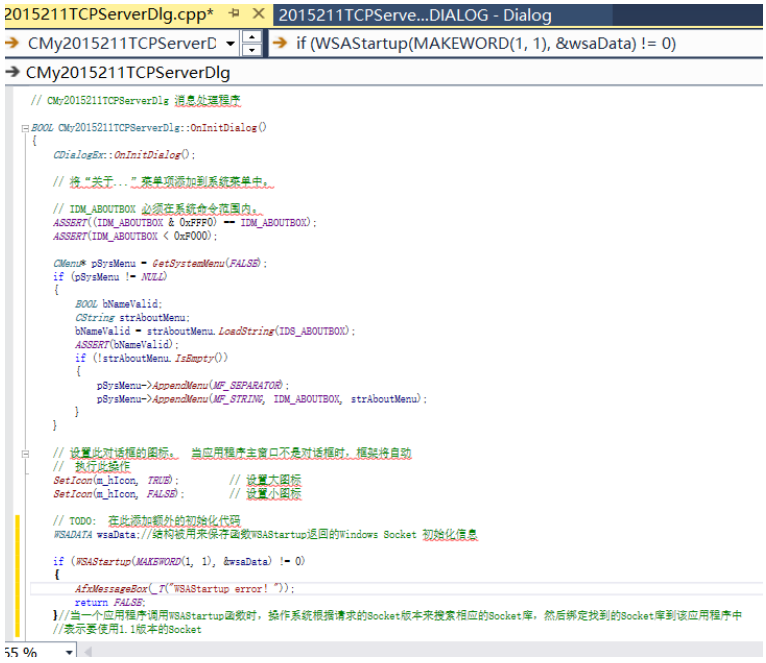
二．Socket 版本初始化

1. 在右手边中间的“解决方案资源管理器”，找到“*TCPServerDlg.cpp”，在对话框初始化函数 *BOOL* CMy2015211TCPServerDlg::OnInitDialog()中添加如下代码。

****

```
WSADATA wsaData;//结构被用来保存函数WSAStartup返回的Windows Socket  初始化信息
    if (WSAStartup(MAKEWORD(1, 1), &wsaData) != 0)
    {
        AfxMessageBox(_T("WSAStartup error！ "));
        return FALSE;
    }//当一个应用程序调用WSAStartup函数时，操作系统根据请求的Socket版本来搜索相应的Socket库，然后绑定找到的Socket库到该应用程序中
    //表示要使用1.1版本的Socket
```

****



2. 选择“资源视图”，调出主界面。选中“取消”按钮，将其 Caption 部分修改为“退出”，选中这个按钮，点击鼠标右键，“添加事件处理程序”，选择“添加编辑”。在 void CMy2015211TCPServerDlg::OnBnClickedCancel()函数中添加如下代码。

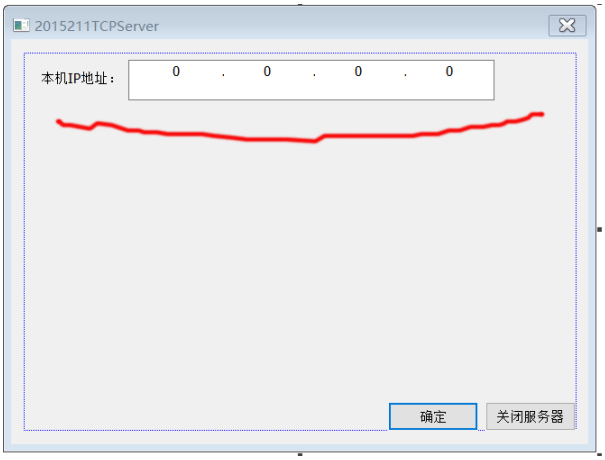
****

```
WSACleanup();//释放资源
```

****

三．获取主机 IP 地址

1. 选择“资源视图”，调出主界面。选中默认的静态文本控件，将其 Caption 部分修改为“本机 IP 地址：”。Align Text 修改为“right”。选择最上面的“格式”--->“按内容调整大小”。
2. 打开工具箱，选择“IP Address Control”，并在界面上绘制此控件。将其 ID 改为“IDC_IPADDRESS_HOST”。选中这个控件，鼠标右键，“添加变量”，输入变量名“HostIP”，点击“完成”。



3. 回到“类视图”，点击“CMy2015211TCPServerDlg”，点击鼠标右键，“添加”->“添加函数”。返回类型改为“void”，“函数名”输入“GetLocalIP”。在 void CMy2015211TCPServerDlg::GetLocalIP()中添加如下代码。

```
**********************************************************************
char HostName[128];//记录主机名
char *IPAddress;//记录IP地址
if (gethostname(HostName, 128) == 0)//获取主机名成功
{
    struct hostent *pHost;
    pHost = gethostbyname(HostName);//用域名或主机名获取IP地址
    IPAddress = pHost->h_addr_list[0];//取第一个地址
    HostIP.SetAddress(*IPAddress, *(IPAddress + 1), *(IPAddress + 2), *(IPAddress + 3));//设置IP
地址
    UpdateData(FALSE);//刷新界面
} **********************************************************************
```

4. 在对话框初始化函数 BOOL CMy2015211TCPServerDlg::OnInitDialog()中调用 GetLocalIP。

```
**********************************************************************
GetLocalIP();//获取 IP 地址
**********************************************************************
```

2015211TCPServer.cpp | 2015211TCPServerDlg.cpp* | 2015211TCPServe...DIA

2015211TCPServerDlg.cp | C:\Users\hfut\Documents\Visual Studio 2013\Projects

CMy2015211TCPServerDlg

```
BOOL CMy2015211TCPServerDlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();

    // 将“关于...”菜单项添加到系统菜单中。

    // IDM_ABOUTBOX 必须在系统命令范围内。
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        BOOL bNameValid;
        CString strAboutMenu;
        bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
        ASSERT(bNameValid);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // 设置此对话框的图标。  当应用程序主窗口不是对话框时，框架将自动
    //  执行此操作
    SetIcon(m_hIcon, TRUE);         // 设置大图标
    SetIcon(m_hIcon, FALSE);        // 设置小图标

    // TODO:  在此添加额外的初始化代码
    WSADATA wsaData;//结构被用来保存函数WSAStartup返回的Windows Socket 初始化信息

    if (WSAStartup(MAKEWORD(1, 1), &wsaData) != 0)
    {
        AfxMessageBox(_T("WSAStartup error! "));
        return FALSE;
    }//当一个应用程序调用WSAStartup函数时，操作系统根据请求的Socket版本来搜索相应的Socket库，然后绑定找到的Socket库到该应用程序中
    //表示要使用1.1版本的Socket

    GetLocalIP();//获取IP地址

    return TRUE;  // 除非将焦点设置到控件，否则返回 TRUE
}
```

四．启动服务

1. 在“解决方案资源管理器”中找到“*TCPServerDlg.cpp”并打开，申明如下全局变量。

```
**********************************************************************
#define DEFAULT_PORT 2000//服务器监听端口
SOCKET m_Listening;//监听socket
HANDLE m_hListenThread;//线程句柄
**********************************************************************
```

2. 选择“资源视图”，调出主界面。删除“确定”按钮。在“工具箱”中选择“Button”并在界面上绘制控件。将其 ID 改为“IDC_BUTTON_START”，Caption 改我“启动服务”。选中这个按钮，点击鼠标右键，“添加事件处理程序”，选择“添加编辑”。在 void CMy2015211TCPServerDlg:: OnBnClickedButtonStart ()函数中添加如下代码。

```
**********************************************************************
    sockaddr_in local;//使用sockaddr_in结构指定IP地址和端口信息
    local.sin_family = AF_INET;
    local.sin_port = htons(DEFAULT_PORT);//设置的端口为DEFAULT_PORT。将整型变量从主机字节顺序转变成网络字节顺序
    char IPAddress[MAX_PATH];//记录IP地址
    BYTE add1, add2, add3, add4;
    HostIP.GetAddress(add1, add2, add3, add4);
    sprintf_s(IPAddress, "%d.%d.%d.%d", add1, add2, add3, add4);
    local.sin_addr.S_un.S_addr = inet_addr(IPAddress);//将一个点分十进制的IP转换成一个长整数型数(u_long类型),返回的整数形式是网络字节序
    m_Listening = socket(AF_INET, SOCK_STREAM, 0);//初始化Socket，采用TCP方式-SOCK_STREAM
    if (m_Listening == INVALID_SOCKET)//创建失败
    {
        AfxMessageBox(_T("初始化socket失败！"));
        return;
    }

    if (bind(m_Listening, (LPSOCKADDR)& local, sizeof(local)) == SOCKET_ERROR)//将IP地址和端口绑定到所创建的套接字上
    {
        AfxMessageBox(_T("绑定失败！"));
        closesocket(m_Listening);
        return;
    }

    //创建监听线程
    DWORD dwThreadID = 0;
    m_hListenThread = ::CreateThread(NULL, 0, ListenThread, this, 0, &dwThreadID);//ListenThread为线程函数，this为当前主对话框的指针
```

****

3. 选择“资源视图”，调出主界面。在“工具箱”中选择“Button”并在界面上绘制控件。将其 ID 改为“IDC_BUTTON_STOP”，Caption 改我“停止服务”。选中这个按钮，点击鼠标右键，“添加事件处理程序”，选择“添加编辑”。在 void CMy2015211TCPServerDlg:: OnBnClickedButtonStop ()函数中添加如下代码。

****

```
TerminateThread(m_hListenThread, 0);//关闭线程
CloseHandle(m_hListenThread);//释放资源
shutdown(m_Listening, 2);//关闭读写通道
closesocket(m_Listening);//释放socket
```

****

4. 回到“资源视图”，打开主界面。选中“启动服务”按钮，鼠标右键，添加变量，输入变量名“m_btnStart”，点击完成。选中“停止服务”按钮，鼠标右键，添加变量，输入变量名“m_btnStop”，点击完成。

   在 void CMy2015211TCPServerDlg::OnBnClickedButtonStart()函数中输入如下代码。

****

```
m_btnStart.EnableWindow(FALSE);
m_btnStop.EnableWindow(TRUE);
```

****

在 void CMy2015211TCPServerDlg:: OnBnClickedButtonStop()函数中输入如下代码。

****

```
m_btnStart.EnableWindow(TRUE);
m_btnStop.EnableWindow(FALSE);
```

****

5. 回到“资源视图”，打开主界面，在“工具箱”中选择“Edit Control”并在界面上绘制。将其 ID 号改为“IDC_EDIT_MESSAGE”，选择这个控件，鼠标右键，“添加变量”，输入变量名“m_edtMessage”。

6. 回到“解决方案资源管理器”，找到“*TCPServerDlg.cpp”并打开，申明如下全局函数。

****

```
char * CSt2char(CString str)
{
    //注意：以下n和len的值大小不同,n是按字符计算的，len是按字节计算的
    int n = str.GetLength(); // n = 14, len = 18
```

```
    //获取宽字节字符的大小，大小是按字节计算的
    int len = WideCharToMultiByte(CP_ACP, 0, str, str.GetLength(), NULL, 0, NULL, NULL);

    //为多字节字符数组申请空间，数组大小为按字节计算的宽字节字节大小
    char * buff = new char[len + 1]; //以字节为单位

    //宽字节编码转换成多字节编码
    WideCharToMultiByte(CP_ACP, 0, str, str.GetLength() + 1, buff, len + 1, NULL, NULL);

    buff[len + 1] = '\0'; //多字节字符以'/0'结束

    return buff;
}

CString char2CSt(char * szBuf)
{
    //计算char *数组大小，以字节为单位，一个汉字占两个字节
    int charLen = strlen(szBuf);
    //计算多字节字符的大小，按字符计算。
    int len = MultiByteToWideChar(CP_ACP, 0, szBuf, charLen, NULL, 0);
    //为宽字节字符数组申请空间，数组大小为按字节计算的多字节字符大小
    TCHAR *buf = new TCHAR[len + 1];
    //多字节编码转换成宽字节编码
    MultiByteToWideChar(CP_ACP, 0, szBuf, charLen, buf, len);
    buf[len] = '\0'; //添加字符串结尾，注意不是len+1
    //将TCHAR数组转换为CString
    CString pWideChar;
    pWideChar.Append(buf);
    //删除缓冲区
    delete[]buf;

    return pWideChar;
}

DWORD WINAPI ListenThread(LPVOID lpparam)
{
    CMy2015211TCPServerDlg* pDlg = (CMy2015211TCPServerDlg*)lpparam;
    if (pDlg == NULL) return 0;//获取当前主对话框的指针

    SOCKET ConnectSocket;//记录捕捉到的连接
    sockaddr_in ClientAddr;//记录数据包的IP地址
    int nLen = sizeof(sockaddr);

    if (listen(m_Listening, 40) == SOCKET_ERROR)//开始监听是否有客户端连接。
```

```
    {

        return 0;
    }

    char szBuf[MAX_PATH];//数据缓冲区
    memset(szBuf, 0, MAX_PATH);//初始化缓冲区

    while (1)
    {
        ConnectSocket = accept(m_Listening, (sockaddr*)&ClientAddr, &nLen);//阻塞直到有客户端连
接，不然多浪费CPU资源
        char *pAddrname = inet_ntoa(ClientAddr.sin_addr);//得到客户端的IP地址

        memset(szBuf, 0, MAX_PATH);//每次接收数据前清空缓冲区

        if (recv(ConnectSocket, szBuf, sizeof szBuf, 0) != SOCKET_ERROR)
        {

            pDlg->m_edtMessage.SetWindowTextW(char2CSt(szBuf));


            char buff[MAX_PATH] = "我是勘探队，我是勘探队，收到请回答！(";
            strcat_s(buff, pAddrname);
            strcat_s(buff, "）");

            send(ConnectSocket, buff, sizeof buff, 0);

        }
    }

    return 0;
}
**********************************************************************
```

五．创建一个初始工程-Client 端

1. 打开 VS2013，选择“文件”→新建→项目；
2. 在“新建项目”中，左手边选择“Virtual C++”，右手边选择“MFC 应用程序”，在下面的“名称”中输入你的工程名，工程名要包含学号，例如：2015211TCPClient。注意查看工程保存的“位置”，以便后期查找这个工程。
3. 点击左手边“应用程序类型”，选择“基于对话框”；再点击左手边的“高级功能”，勾选“Windows 套接字”。最后点击“完成”，一个新的工程就创建好了。
4. 在右手边中间的“解决方案资源管理器”，找到“*TCPClientDlg.cpp”，在对话框初始化函数 BOOL CMy2015211TCPClientDlg::OnInitDialog()中添加如下代码。

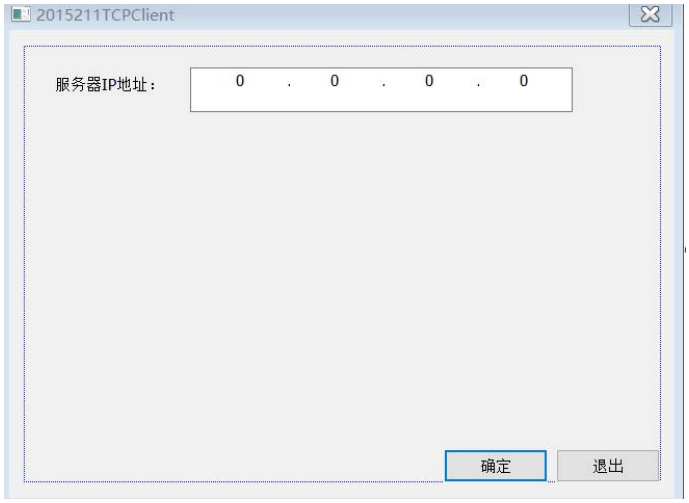
```
**********************************************************************
WSADATA wsaData;//结构被用来保存函数 WSAStartup 返回的 Windows Socket 初始化信息
 if (WSAStartup(MAKEWORD(1, 1), &wsaData) != 0)
 {
     AfxMessageBox(_T("WSAStartup error！"));
     return FALSE;
 }//当一个应用程序调用 WSAStartup 函数时，操作系统根据请求的 Socket 版本来搜索相应的 Socket 库，然后绑定找到的 Socket 库到该应用程序中
 //表示要使用 1.1 版本的 Socket
**********************************************************************
```

5. 选择“资源视图”，调出主界面。选中“取消”按钮，将其 Caption 部分修改为“退出”，选中这个按钮，点击鼠标右键，“添加事件处理程序”，选择“添加编辑”。在 void CMy2015211TCPClientDlg::OnBnClickedCancel()函数中添加如下代码。

```
**********************************************************************
WSACleanup();//释放资源
**********************************************************************
```

六．设置服务器 IP 地址

1. 选择“资源视图”，调出主界面。选中默认的静态文本控件，将其 Caption 部分修改为“服务器 IP 地址：”。Align Text 修改为“right”。选择最上面的“格式”--->“按内容调整大小”。
2. 打开工具箱，选择“IP Address Control”，并在界面上绘制此控件。将其 ID 改为“IDC_IPADDRESS_SERVER”。选中这个控件，鼠标右键，“添加变量”，输入变量名“ServerIP”，点击“完成”。



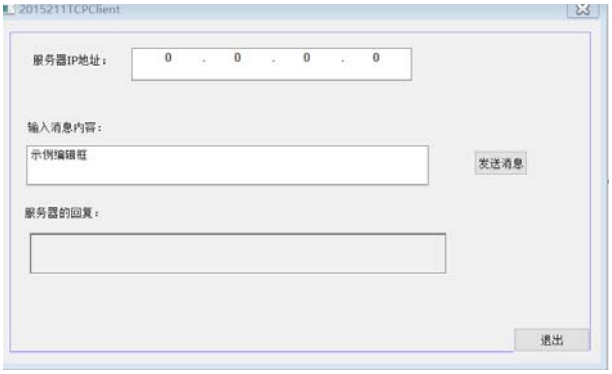
3. 回到“解决方案资源管理器”，找到“*TCPClientDlg.cpp”，在对话框初始化函数 BOOL CMy2015211TCPClientDlg::OnInitDialog()中添加如下代码。

```
**********************************************************************
ServerIP.SetAddress(127, 0, 0, 1);
UpdateData(FALSE);
**********************************************************************
```

七．实现消息的发送

1. 选择“资源视图”，调出主界面。选中工具箱中的“Static Text”控件并在界面上绘制，将其 Caption 部分修改为“输入消息内容：”。Align Text 修改为“right”。选择最上面的“格式”--->“按内容调整大小”。
2. 选中工具箱中的“Edit Control”控件并在界面上绘制，将其 ID 修改为“IDC_EDIT_MESSAGE”。选中这个控件，鼠标右键，添加变量，把“类别”修改为“value”，输入变量名“m_strSendMessage”，点击完成。
3. 回到“解决方案资源管理器”，找到“*TCPClientDlg.cpp”，在对话框构造函数 CMy2015211TCPClientDlg::CMy2015211TCPClientDlg(*CWnd** pParent /*=NULL*/)中修改 m_strSendMessage 的初始值。如下：

   m_strSendMessage(_T("研究所呼叫勘探队！研究所呼叫勘探队！请回答.."))
4. 选择“资源视图”，调出主界面。选中工具箱中的“Static Text”控件并在界面上绘制，将其 Caption 部分修改为“服务器的回复：”。Align Text 修改为“right”。选择最上面的“格式”--->“按内容调整大小”。
5. 再选中工具箱中的“Static Text”控件并在界面上绘制，将其 Caption 部分清空。将其 ID 修改为“IDC_STATIC_RECV”，“Static Edge”和“Border”均改为 True，选中这个控件，点击鼠标右键，添加变量，把“类别”修改为“value”，输入变量名“m_strRecvMessage”，点击完成。



6. 回到“解决方案资源管理器”，找到“*TCPClientDlg.cpp”，声明如下全局变量和函数：

```
**********************************************************************
#define DEFAULT_PORT 2000//服务器监听端口，注意与服务器端保持一致

char * CSt2char(CString str)
{
    //注意：以下n和len的值大小不同,n是按字符计算的，len是按字节计算的
    int n = str.GetLength(); // n = 14, len = 18

    //获取宽字节字符的大小，大小是按字节计算的
    int len = WideCharToMultiByte(CP_ACP, 0, str, str.GetLength(), NULL, 0, NULL, NULL);

    //为多字节字符数组申请空间，数组大小为按字节计算的宽字节字节大小
    char * buff = new char[len + 1]; //以字节为单位
```

```
    //宽字节编码转换成多字节编码
    WideCharToMultiByte(CP_ACP, 0, str, str.GetLength() + 1, buff, len + 1, NULL, NULL);

    buff[len + 1] = '\0'; //多字节字符以'/0'结束

    return buff;
}

CString char2CSt(char * szBuf)
{
    //计算char *数组大小，以字节为单位，一个汉字占两个字节
    int charLen = strlen(szBuf);
    //计算多字节字符的大小，按字符计算。
    int len = MultiByteToWideChar(CP_ACP, 0, szBuf, charLen, NULL, 0);
    //为宽字节字符数组申请空间，数组大小为按字节计算的多字节字符大小
    TCHAR *buf = new TCHAR[len + 1];
    //多字节编码转换成宽字节编码
    MultiByteToWideChar(CP_ACP, 0, szBuf, charLen, buf, len);
    buf[len] = '\0'; //添加字符串结尾，注意不是len+1
    //将TCHAR数组转换为CString
    CString pWideChar;
    pWideChar.Append(buf);
    //删除缓冲区
    delete[]buf;

    return pWideChar;
}
```

****

7. 回到主界面，删除“确定”按钮。在“工具箱”中选择“Button”并在界面上绘制控件。将其 ID 改为“IDC_BUTTON_SEND”，Caption 改我“发送消息”。选中这个按钮，点击鼠标右键，“添加事件处理程序”，选择“添加编辑”。在 void CMy2015211TCPClientDlg:: OnBnClickedButtonSend ()函数中添加如下代码。

****

```
sockaddr_in local;
local.sin_family = AF_INET;//必须是AF_INET,表示该socket在Internet域中进行通信
local.sin_port = htons(DEFAULT_PORT);//端口号

char IPAddress[MAX_PATH];
BYTE add1, add2, add3, add4;
ServerIP.GetAddress(add1, add2, add3, add4);
sprintf_s(IPAddress, "%d.%d.%d.%d", add1, add2, add3, add4);
local.sin_addr.S_un.S_addr = inet_addr(IPAddress);//服务器的IP地址。

SOCKET socketTmp = socket(AF_INET, SOCK_STREAM, 0);///初始化Socket，与服务器端保持一致
```

```
if (connect(socketTmp, (LPSOCKADDR)& local, sizeof(local)) != 0)//连接服务器
{
	closesocket(socketTmp);
	AfxMessageBox(_T("连接服务器失败"));
	return;
}

char szText[MAX_PATH];//数据缓冲区
memset(szText, 0, MAX_PATH);
UpdateData(TRUE);
sprintf_s(szText, "%s", CSt2char(m_strSendMessage));
send(socketTmp, szText, sizeof szText, 0);//发送消息

memset(szText, 0, MAX_PATH);//清空缓冲区
if (recv(socketTmp, szText, sizeof szText, 0) != SOCKET_ERROR)//读取服务器端返回的数据
{
	m_strRecvMessage = char2CSt(szText);
	UpdateData(FALSE);//刷新界面
}

shutdown(socketTmp, 2);//关闭读写通道
closesocket(socketTmp);
**********************************************************************
```

VS2013+Socket 开发教程-UDP 篇

合肥工业大学 计算机与信息学院 张国富

zgf@hfut.edu.cn

一．Server 端

1. 打开前面创建的 TCP Server 端的工程，在右手边中间的“解决方案资源管理器”，打开“*TCPServerDlg.cpp”。
2. 找到 void CMy2015211TCPServerDlg::OnBnClickedButtonStart()函数，修改代码如下：
   m_Listening = *socket*(*AF_INET*, ***SOCK_DGRAM***, 0);//初始化 Socket，采用 UDP 方式-SOCK_DGRAM
3. 找到 *DWORD WINAPI* ListenThread(*LPVOID* lpparam) 函数，修改代码如下：

```
CMy2015211TCPServerDlg* pDlg = (CMy2015211TCPServerDlg*)lpparam;
    if (pDlg == NULL) return 0;//获取当前主对话框的指针

    sockaddr_in ClientAddr;//记录数据包的IP地址
    int nLen = sizeof(sockaddr);

    char szBuf[MAX_PATH];//数据缓冲区
    memset(szBuf, 0, MAX_PATH);//初始化缓冲区

    while (1)
    {
        if (recvfrom(m_Listening, szBuf, sizeof szBuf, 0, (struct sockaddr*)&ClientAddr, &nLen) !=
SOCKET_ERROR)
        {
            pDlg->m_edtMessage.SetWindowTextW(char2CSt(szBuf));

            char *pAddrname = inet_ntoa(ClientAddr.sin_addr);//得到客户端的IP地址

            memset(szBuf, 0, MAX_PATH);//每次接收数据前清空缓冲区

            char buff[MAX_PATH] = "我是勘探队，我是勘探队，收到请回答！(";
            strcat_s(buff, pAddrname);
            strcat_s(buff, "）");

            sendto(m_Listening, buff, sizeof buff, 0, (struct sockaddr*)&ClientAddr, sizeof
ClientAddr);

        }
    }
```

二. Client 端

1. 打开前面创建的 TCP Client 端的工程，在右手边中间的“解决方案资源管理器”，打开“*TCPClientDlg.cpp”。
2. 找到 void CMy2015211TCPClientDlg:: OnBnClickedButtonSend ()函数，修改代码如下：

```
**********************************************************************
sockaddr_in local;
    int nlen = sizeof local;
    local.sin_family = AF_INET;//必须是AF_INET,表示该socket在Internet域中进行通信
    local.sin_port = htons(DEFAULT_PORT);//端口号

    char IPAddress[MAX_PATH];
    BYTE add1, add2, add3, add4;
    ServerIP.GetAddress(add1, add2, add3, add4);
    sprintf_s(IPAddress, "%d.%d.%d.%d", add1, add2, add3, add4);
    local.sin_addr.S_un.S_addr = inet_addr(IPAddress);//服务器的IP地址。

    SOCKET socketTmp = socket(AF_INET, SOCK_DGRAM, 0);///初始化Socket，与服务器端保持一致


    char szText[MAX_PATH];//数据缓冲区
    memset(szText, 0, MAX_PATH);
    UpdateData(TRUE);
    sprintf_s(szText, "%s", CSt2char(m_strSendMessage));

    sendto(socketTmp, szText, sizeof szText, 0, (struct sockaddr*)&local, sizeof local);//发送消息

    memset(szText, 0, MAX_PATH);//清空缓冲区

    if (recvfrom(socketTmp, szText, sizeof szText, 0, (struct sockaddr*)&local, &nlen) !=
SOCKET_ERROR)//读取服务器端返回的数据
    {
        m_strRecvMessage = char2CSt(szText);
        UpdateData(FALSE);//刷新界面
    }

    shutdown(socketTmp, 2);//关闭读写通道
    closesocket(socketTmp);
**********************************************************************
```

VS2013+Socket 开发教程-文件传输

合肥工业大学 计算机与信息学院 张国富

zgf@hfut.edu.cn

一．Server 端

1. 打开前面创建的 TCP Server 端的工程，在右手边中间的“解决方案资源管理器”，打开“*TCPServerDlg.cpp”。添加全局常量。

```
#define MAX_LENGTH 1024*100//每次读取的数据块大小
```

2. 找到 *DWORD WINAPI* ListenThread(*LPVOID* lpparam)全局函数，修改代码如下：

```
DWORD WINAPI ListenThread(LPVOID lpparam)
{
	CMy2015211TCPServerDlg* pDlg = (CMy2015211TCPServerDlg*)lpparam;
	if (pDlg == NULL) return 0;//获取当前主对话框的指针

	SOCKET ConnectSocket;//记录捕捉到的连接
	sockaddr_in ClientAddr;//记录数据包的IP地址
	int nLen = sizeof(sockaddr);

	if (listen(m_Listening, 40) == SOCKET_ERROR)//开始监听是否有客户端连接。
	{
		return 0;
	}

	ConnectSocket = accept(m_Listening, (sockaddr*)&ClientAddr, &nLen);//阻塞直到有客户端连接，不然多浪费CPU资源
	char *pAddrname = inet_ntoa(ClientAddr.sin_addr);//得到客户端的IP地址

	//读取mp3文件
	FILE *fp = NULL;
	if (fopen_s(&fp, "大悲咒1.mp3", "rb")!=0)
	{
		AfxMessageBox(_T("Open Mp3 File Failed1!"));
		return 0;
	}

	char dataBuf[MAX_LENGTH];//数据缓冲区
	int i = 1;
	while (!feof(fp))
	{
		memset(dataBuf, 0, MAX_LENGTH);//初始化缓冲区
		int iBytesRead = fread(dataBuf, 1, MAX_LENGTH, fp);
		int iRet = send(ConnectSocket, dataBuf, iBytesRead, 0);
```

```
            if (iRet <=0)
            {
                break;
            }

            char szBuf[MAX_PATH];//数据缓冲区
            memset(szBuf, 0, MAX_PATH);//初始化缓冲区
            sprintf_s(szBuf, "Send packet %d length: %d to", i++, iBytesRead);
            strcat_s(szBuf, pAddrname);
            pDlg->m_edtMessage.SetWindowTextW(char2CSt(szBuf));

            Sleep(10);

        }

        fclose(fp);

        AfxMessageBox(_T("Send mp3 file successfully!"));

        return 0;
}
```

二． Client 端

1. 打开前面创建的 TCP Client 端的工程，在右手边中间的“解决方案资源管理器”，打开“*TCPClientDlg.cpp”。
2. 找到 void CMy2015211TCPClientDlg:: OnBnClickedButtonSend ()函数，修改代码如下：

****

```
void CMy2015211TCPClientDlg::OnBnClickedButtonSend()
{
	sockaddr_in local;
	local.sin_family = AF_INET;//必须是AF_INET,表示该socket在Internet域中进行通信
	local.sin_port = htons(DEFAULT_PORT);//端口号

	char IPAddress[MAX_PATH];
	BYTE add1, add2, add3, add4;
	ServerIP.GetAddress(add1, add2, add3, add4);
	sprintf_s(IPAddress, "%d.%d.%d.%d", add1, add2, add3, add4);
	local.sin_addr.S_un.S_addr = inet_addr(IPAddress);//服务器的IP地址。

	SOCKET socketTmp = socket(AF_INET, SOCK_STREAM, 0);///初始化Socket，与服务器端保持一致

	if (connect(socketTmp, (LPSOCKADDR)& local, sizeof(local)) != 0)//连接服务器
	{
		closesocket(socketTmp);
		AfxMessageBox(_T("连接服务器失败"));
		return;
	}

	//读取mp3文件
	FILE *fp = NULL;
	if (fopen_s(&fp, "大悲咒2.mp3", "wb") != 0)
	{
		AfxMessageBox(_T("Open Mp3 File Failed2!"));
		return;
	}

	int len = 0;//接收到的数据长度
	int i = 1;
	char dataBuff[MAX_LENGTH];//数据缓冲区

	while (1)
	{
		memset(dataBuff, 0, MAX_LENGTH);
		len = recv(socketTmp, dataBuff, sizeof(dataBuff), 0);
		if (len<=0)
```

```
            {
                //AfxMessageBox(_T("Recv error!"));

                shutdown(socketTmp, 2);//关闭读写通道
                closesocket(socketTmp);

                break;
            }else
                fwrite(dataBuff, 1, len, fp);

            Sleep(10);

        }

        fclose(fp);
        AfxMessageBox(_T("Recv finished!"));



}
```