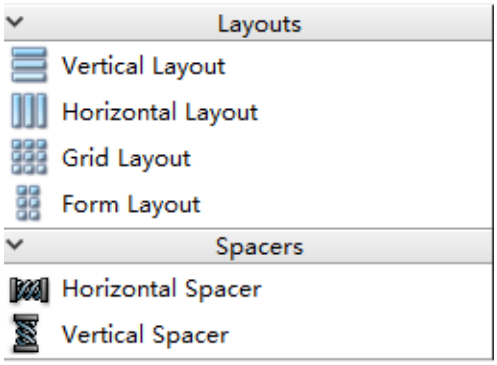
第4章-页面布局

author：岳石磊 copyright：科林明伦 内部资料禁止外泄

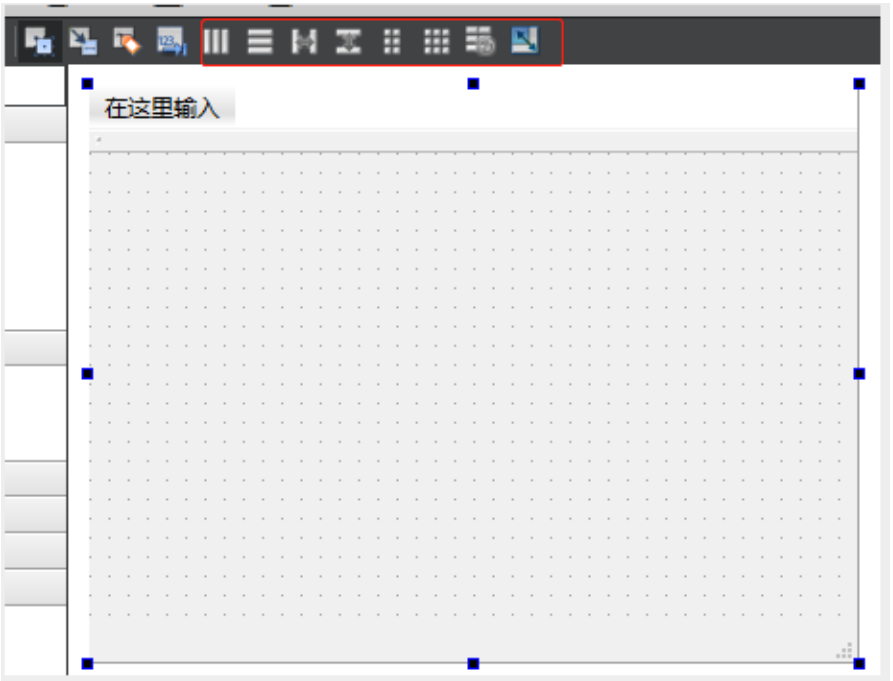
1.布局组件和布局按钮

QT 的UI设计器中提供了丰富的布局管理功能，组件面板里有Layouts（布局） 和 Spacers（间隔器）两个组件面板。



Vertical Layout ： 垂直方向布局，组件自动在垂直方向上分布。
Horizontal Layout ： 水平方向布局，组件自动在水平方向上分布。
Grid Layout：网格状布局（栅格布局），网状布局大小改变时，每个网格的大小都改变。
Form Layout：窗体布局，与网格状布局类似，但只有最右侧一列网格会改变大小。
Horizontal Spacer：一个用于水平分隔的空格。
Vertical Spacer：一个用于垂直分隔的空格。

在窗体上方的工具栏里还提供了布局管理按钮。

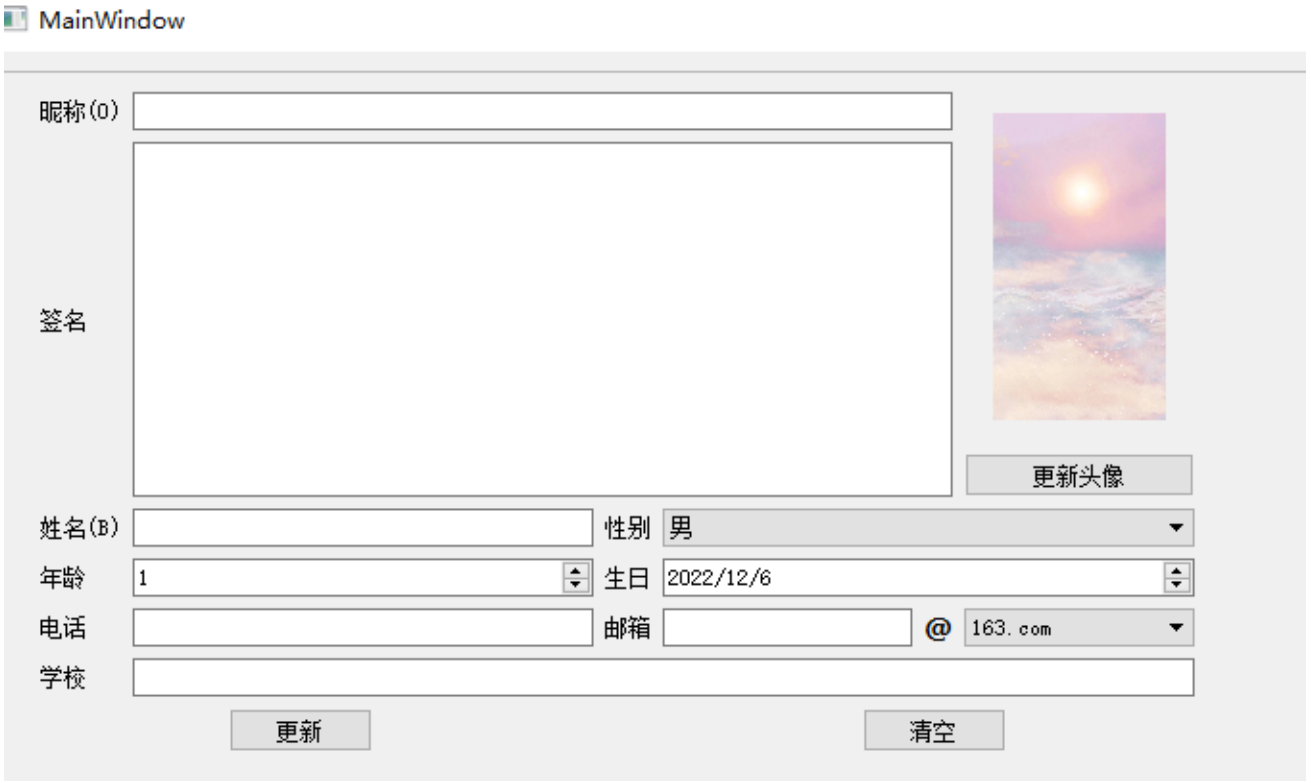


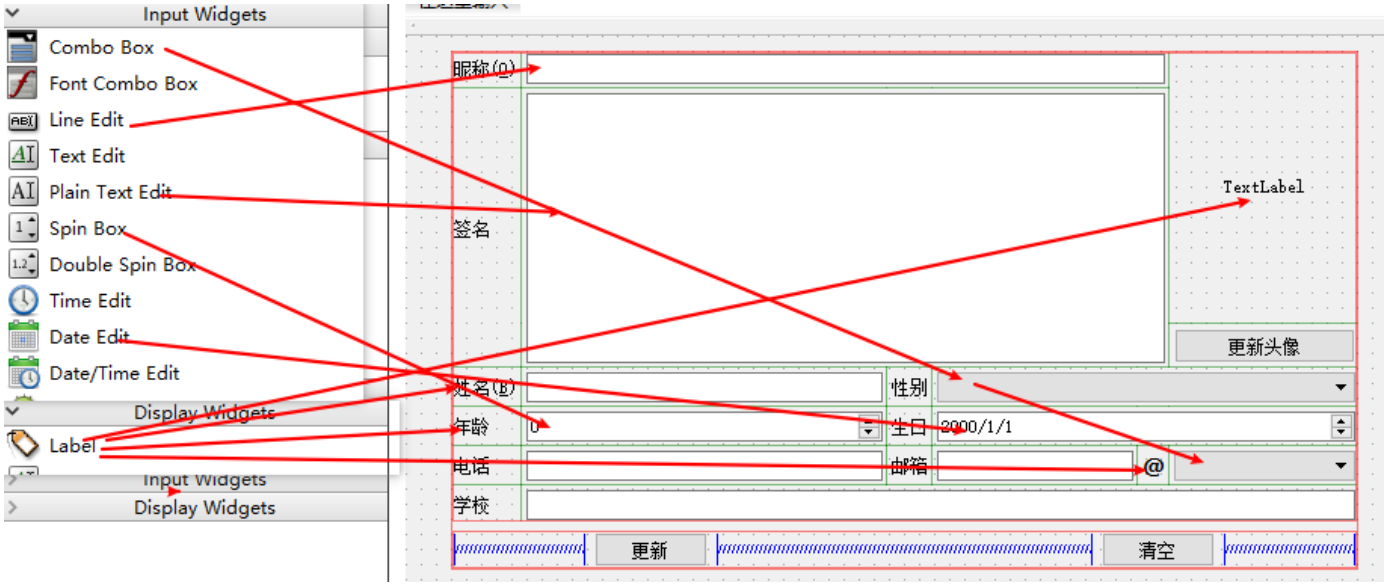
从左到右依次为：
Edit Widget：界面设计进入编辑状态，就是正常的设计状态。
Edit Signals/Slots：进入信号与槽的可视化设计状态。

Edit Buddies：进入伙伴关系编辑状态，可以设置一个Label 与 一个组件成为伙伴关系。
Edit Tab Order：进入 Tab 顺序编辑状态，Tab 顺序是在键盘上按 Tab 热键时，输入焦点在界面各个组件之间的跳动顺序。
Lay Out Horizontally：将窗体上所选择的组件水平布局。
Lay Out Vertically：将窗体上所选择的组件垂直布局。
Lay Out Horizontally in Splitter：将窗体上所选组件用一个分割条进行水平分割布局。
Lay Out Vertically in Splitter：将窗体上所选组件用一个分割条进行垂直分割布局。
Lay Out in a Form Layout：将窗体上所选组件按窗体布局。
Lay Out in a Grid：将窗体上所选组件按网格布局。
Break Layout：解除窗体上所选组件的布局，打散现有布局。
Adjust Size：自动调整所选组件的大小。

2.练习-用户信息

【更新头像】弹出一个弹出框，选择图片文件，缩放到label 上显示头像，表单信息填写完，将用户信息【更新】到文件中，【清空】将表单输入的内容清空，其他重置为初值。





由于组件很多，更改对象的名字方便记忆和使用。

MainWindow	QMainWindow
centralWidget	QWidget
[illegible]	[illegible]
[illegible]	[illegible]
verticalLayout	QVBoxLayout
gridLayout	QGridLayout
comboBox_sex	QComboBox
comboBox_suffix	QComboBox
dateEdit_birth	QDateEdit
label_age	QLabel
label_at	QLabel
label_birth	QLabel
label_email	QLabel
label_pic	QLabel
label_school	QLabel
label_tel	QLabel
lineEdit_email	QLineEdit
lineEdit_name	QLineEdit
lineEdit_nick	QLineEdit
lineEdit_school	QLineEdit
lineEdit_tel	QLineEdit
nick	QLabel
plainTextEdit_signal	QPlainTextEdit
pushButton_pic	QPushButton
signal	QLabel
signal_name	QLabel
signal_sex	QLabel
spinBox_age	QSpinBox
horizontalLayout	QHBoxLayout
horizontalSpacer	Spacer
horizontalSpacer_2	Spacer
horizontalSpacer_3	Spacer
pushButton_clear	QPushButton
pushButton_update	QPushButton

2.1 准备工作

将界面上的【年龄】【性别】【生日】【邮箱后缀】的初始值设定好，此项工作应当在窗口创建之初就应当完成。在MainWindow 的构造函数中setupUi之后进行设定。

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow),m_mysql("utf8")
{
    ui->setupUi(this);


    //开始准备
    ui->comboBox_sex->addItem("男");
    ui->comboBox_sex->addItem("女");

    //ui->comboBox_sex->setCurrentIndex(1);  //通过下标设置，哪一个下拉框项显示到编辑框上

    QStringList strList;
    strList.push_back("qq.com");
    strList.push_back("163.com");
    strList.push_back("gmail.com");
    strList.push_back("hotmail.com");

    ui->comboBox_suffix->addItems(strList);
    ui->comboBox_suffix->setCurrentText("163.com");   //通过字符串，哪一个下拉框项显示到编辑框上


    ui->spinBox_age->setValue(1);  //年龄默认为1岁


    QDate date = QDate::currentDate();  //获取当前日期
    ui->dateEdit_birth->setDate(date);  //设置当前日期到日期编辑框上
```

对于 ComboBox 组件，设置编辑框上的文本，可以通过两种方式：setCurrentIndex （下标从0开始，顺序递增）和 setCurrentText。如果不设置，默认则是下拉框中的第一个选项。

2.2 设置头像

点击【更新头像】，弹出一个文件对话框，选择一个图片文件并将图片设置到上方的 Label 组件上。

添加一个自定义槽函数，并绑定【更新头像】按钮的 clicked信号上。

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;

public slots:
    void slots_updatePic();
```

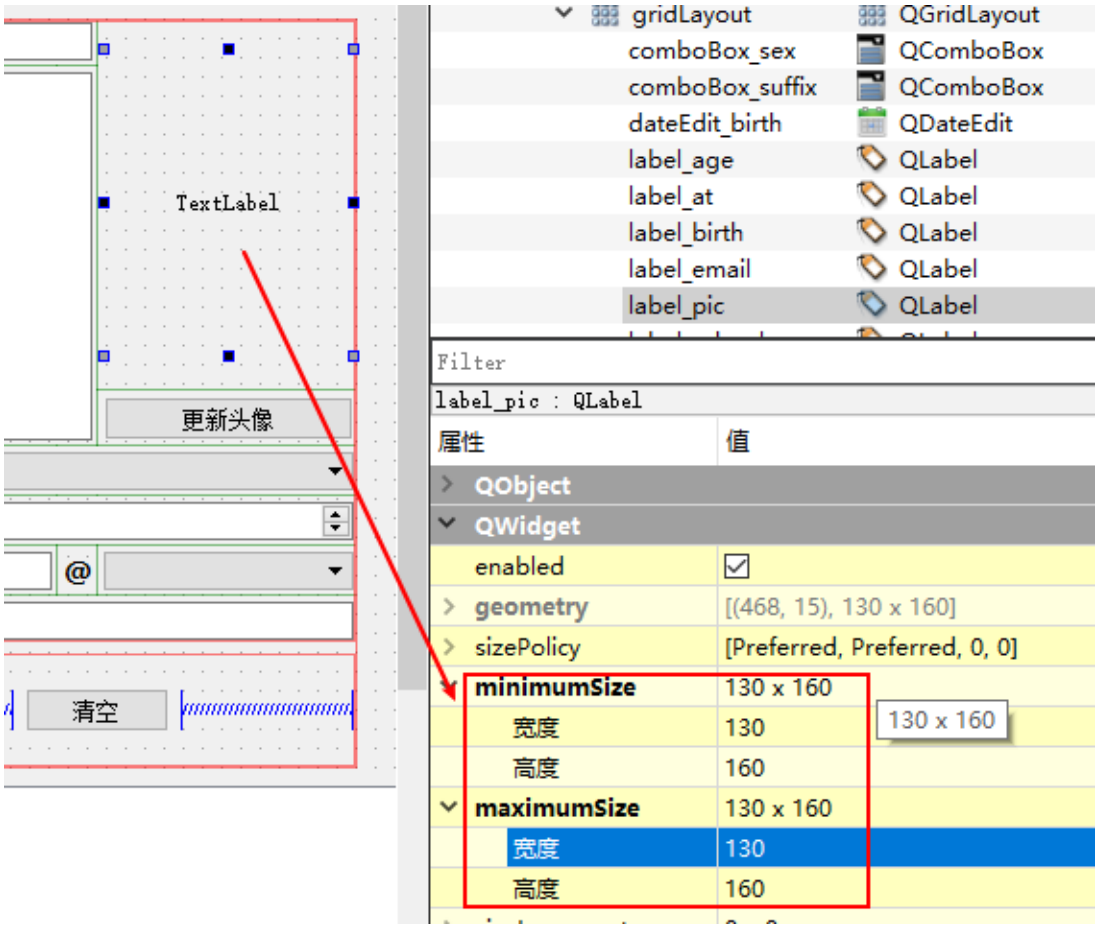
在MainWindow 的构造函数中进行连接:

```
1  //绑定连接
2  QObject::connect(ui->pushButton_pic,SIGNAL(clicked()),this,SLOT(slots_u
   pdatePic()));
```

slots_updatePic 函数的定义实现:

```
void MainWindow::slots_updatePic(){
    QString wrkPath =   QDir::currentPath();  //获取当前项目的工作路径

    //如果选择了一张图，则返回绝对路径，否则返回空字符串
    QString picPath = QFileDialog::getOpenFileName(this,  //父窗口
                            "请选择一张图片", //标题
                            wrkPath,   //初始的默认路径
                            "Images (*.png *.bmp *.jpg)"// 文件后缀过滤
                            );

    qDebug()<<"picPath="<<picPath;
    if(picPath!=""){   //选择了一张图片，路径+图片文件名的形式
        QPixmap pic(picPath);  //定义对象即和图片资源进行绑定
        //pic.load(picPath);  //pic 变量和 具体的图片资源进行绑定

        QRect rect = ui->label_pic->geometry();

        //图片进行缩放（宽度，高度，模式（等比例缩放））
        pic = pic.scaled(rect.width(),rect.height(),Qt::KeepAspectRatio);
        ui->label_pic->setPixmap(pic);  //将图片设置到label 组件上
    }else{
        qDebug()<<"并未选择任何图片";
    }
}
```

其中用于显示图片的Label 会根据图片的大小而改变自身的大小，于是我们将其大小固定。



对于图片的缩放，有三种模式： 而我们选择的是保持宽高比。

- If *aspectRatioMode* is Qt::IgnoreAspectRatio, the pixmap is scaled to *size*.
- If *aspectRatioMode* is Qt::KeepAspectRatio, the pixmap is scaled to a rectangle as large as possible inside *size*, preserving the aspect ratio.
- If *aspectRatioMode* is Qt::KeepAspectRatioByExpanding, the pixmap is scaled to a rectangle as small as possible outside *size*, preserving the aspect ratio.

2.3 创建文件

在MainWindow 类中增加创建文件的公共接口 `bool CreateFile(QString filePath,QString pathfile)`。

```
namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;

public:
    bool CreateFile(QString filePath,QString pathfile);
```

在函数的实现中，先创建路径在创建文件，创建路径是选择函数 mkpath 而不是mkdir，mkdir是用来创建文件夹的。

在创建文件时使用函数 open 虽然名为打开，但实则包含创建。

```
bool MainWindow::CreateFile(QString filePath,QString pathfile){
    if(filePath==""){
        filePath = QDir::currentPath();
    }

    if(pathfile==""){
        QMessageBox::warning(nullptr,"警告","文件名为空，创建文件失败");
        return false;
    }

    QDir dir;
    if( !dir.exists(filePath)){  //不存在路径
        if( !dir.mkpath(filePath)){  //创建路径
            QMessageBox::critical(nullptr,"错误","创建路径失败");
            return false;
        }

    }
    //---路径存在，或者是创建成功----------------------

    //路径和文件名拼接，创建文件

    dir.setPath(filePath);  //dir 对象 和 路径绑定
    QString absPathFile =   dir.absoluteFilePath(pathfile);   //使用上一步绑定的路径和参数中的文件名做一个拼接

    QFile file(absPathFile);   //定义文件对象和某一个文件绑定

    if( !file.exists()){  //如果是不存在，则创建文件

        if( !file.open(QIODevice::ReadWrite|QIODevice::Text)){  //创建文件 读写文本
            QMessageBox::critical(nullptr,"错误","文件创建失败");
            return false;
        }

    }
    qDebug()<<"文件创建成功";
    file.close();  //关闭文件
    return true;
}
```

2.4 写入文件

我们约定，用户信息文件名为：userinfo.txt，此文件应当是处于一个“常开的”状态，当程序创建的时候打开文件，直到程序退出才关闭文件。所以我们需要一个用于绑定此文件的 QFile 的变量。

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;

public:
    QFile m_file;
```

在MainWindow构造函数中添加如下代码：

```
QString pathfile = QDir::currentPath()+"/userinfo.txt";
m_file.setFileName(pathfile);   //和具体的某个文件绑定

//判断文件是否存在，不存在则创建
if( !m_file.exists() ){
    if(!CreateFile(QDir::currentPath(),"userinfo.txt")){
        QMessageBox::critical(nullptr,"错误","创建文件失败");
        exit(0);
        return ;
    }
}

if( !m_file.open(QIODevice::ReadWrite|QIODevice::Text)){
    QMessageBox::critical(nullptr,"错误","文件打开失败");
    exit(0);
    return ;
}
```

点击【更新】按钮，将获取的表单信息写入文件，添加自定义槽函数与按钮的clicked信号进行绑定关联，


```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;

public slots:
    void slots_updateToFile();
```

构造函数添加如下代码:

```
//更新按钮  和 槽函数进行绑定关联
QObject::connect(ui->pushButton_update,SIGNAL(clicked()),this,SLOT(slots_updateToFile()));
```

slots_updateToFile 函数实现如下:

```
void MainWindow::slots_updateToFile(){
    m_file.resize(0);  //在每次写入文件之前清空文件

    //获取表单信息
    QString  strNick =   ui->lineEdit_nick->text();

    if( !m_file.write((strNick.toStdString()+"\n").c_str())){

        QMessageBox::critical(nullptr,"错误","昵称写入文件失败");
    }


```

```
13      if( !m_file.write((ui->plainTextEdit_signal->toPlainText().toStdStr
    ing()+"\n").c_str())){
14
15          QMessageBox::critical(nullptr,"错误","签名写入文件失败");
16      }
17
18      if( !m_file.write((ui->lineEdit_name->text().toStdString()+"\n").c_
    str())){
19
20          QMessageBox::critical(nullptr,"错误","姓名写入文件失败");
21      }
22
23      if( !m_file.write((ui->comboBox_sex->currentText().toStdString()+
    "\n").c_str())){
24
25          QMessageBox::critical(nullptr,"错误","性别写入文件失败");
26      }
27
28      QString strAge = QString::number( ui->spinBox_age->value());
29
30      //ui->spinBox_age->text();直接获取文本
31
32      if( !m_file.write((strAge.toStdString()+"\n").c_str())){
33
34          QMessageBox::critical(nullptr,"错误","年龄写入文件失败");
35      }
36
37      if( !m_file.write((ui->dateEdit_birth->text().toStdString()+"\n").c
    _str())){
38
39          QMessageBox::critical(nullptr,"错误","生日写入文件失败");
40      }
41
42      if( !m_file.write((ui->lineEdit_tel->text().toStdString()+"\n").c_s
    tr())){
43
44          QMessageBox::critical(nullptr,"错误","电话写入文件失败");
45      }
46
47      QString strEmail = ui->lineEdit_email->text()+"@"+ui->comboBox_suff
    ix->currentText();
48
49      if( !m_file.write((strEmail.toStdString()+"\n").c_str())){
50
```

```
51          QMessageBox::critical(nullptr,"错误","邮箱写入文件失败");
52      }
53
54      if( !m_file.write((ui->lineEdit_school->text().toStdString()+"\n").
    c_str())){
55
56          QMessageBox::critical(nullptr,"错误","学校写入文件失败");
57      }
58
59      //刷新文件缓冲区
60      m_file.flush();
61  }
```

2.5 清空表单信息

在MainWindow中增加自定义槽函数 void slots_clearFrom()，并在构造中进行绑定关联。

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;

public slots:
    void slots_clearFrom();
```

```
1   //清空按钮 和 槽函数进行绑定关联
2   QObject::connect(ui->pushButton_clear,SIGNAL(clicked()),this,SLOT(slots
    _clearFrom()));
```

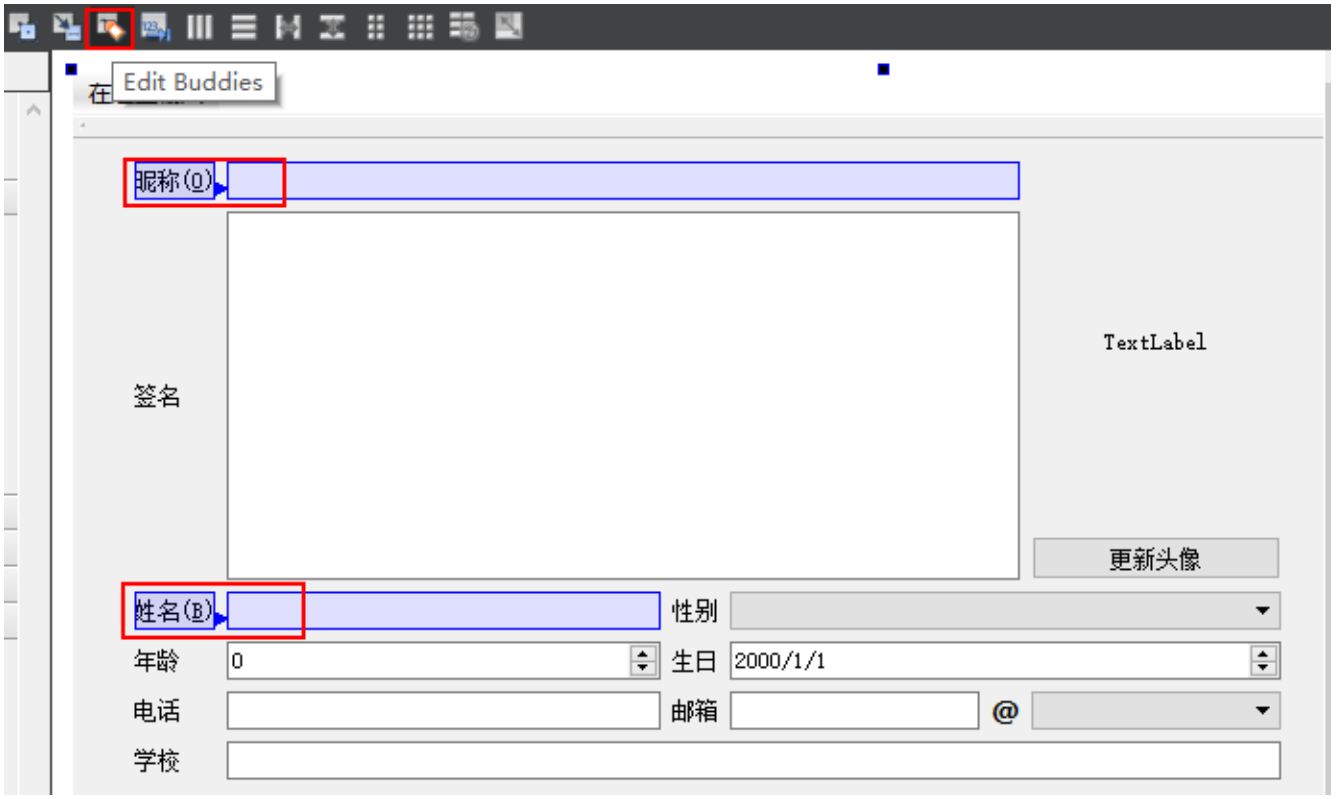
slots_clearFrom 函数定义如下：

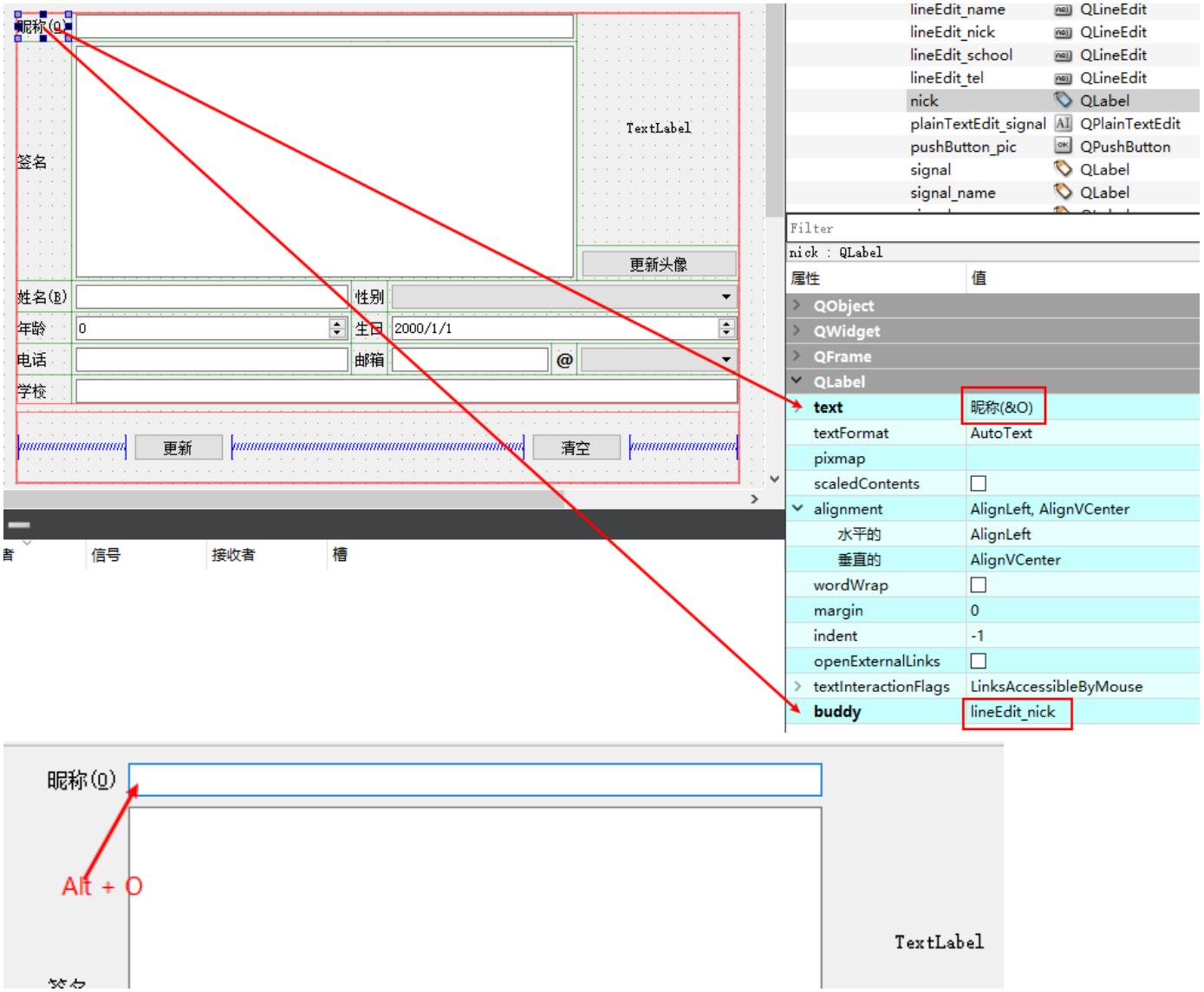
```
void MainWindow::slots_clearFrom(){
    ui->lineEdit_nick->setText("");  //置为空串
    //ui->lineEdit_nick->clear();    //清空

    ui->plainTextEdit_signal->clear();
    ui->lineEdit_name->clear();
    ui->comboBox_sex->setCurrentText("男");
    ui->spinBox_age->setValue(1);
    ui->dateEdit_birth->setDate(QDate::currentDate());
    ui->lineEdit_tel->clear();
    ui->lineEdit_email->clear();
    ui->comboBox_suffix->setCurrentIndex(1);
    ui->lineEdit_school->clear();
}
```

对于编辑文本类的组件，清空操作可以 设置空串 或 clear。

3. Buddy(伙伴)关系

伙伴关系是指界面上的一个 label 和一个组件关联，在程序运行时，在窗体上按 快捷键 快速将焦点切换到label的伙伴关系组件上。
在伙伴关系工具下，选中label 标签拖动到后面的组件上，那么两者就成了伙伴关系，label 的buddy 属性，显示了关联的伙伴。
label 的 text 属性改为 xx(&S) ，括号为英文，其中&用来指定快捷键，并不会显示到窗口上，指定的快捷键为 O，当程序运行后，按下 Alt + O 即可将光标焦点切换到昵称的伙伴组件上。





4. Tab 顺序

tab 顺序说的在窗口上按下 tab 键，各个控件焦点的顺序。可以按照我们的习惯来指定这些控件焦点顺序，而不是一个乱的顺序。
注意：没有输入焦点的组件是没有tab顺序的。如果我们想更改顺序，只需要点击上面的数字，自动更改顺序，点击的顺序为自然数顺序。

控件上tab顺序序号有三种颜色：
绿色：代表已经点击过了，排好顺序了。
红色：正在指定顺序的数字下标。
蓝色：还未指定的数字顺序。

也可以选中数字【右键】
【从这里开始】：代表从选中数字之后的顺序开始调整，之前的数字顺序为原来的顺序。
【重新开始】：所有的控件重新从1开始指定。

如果我们想按照数字反向顺序切换，【shift】+【tab】