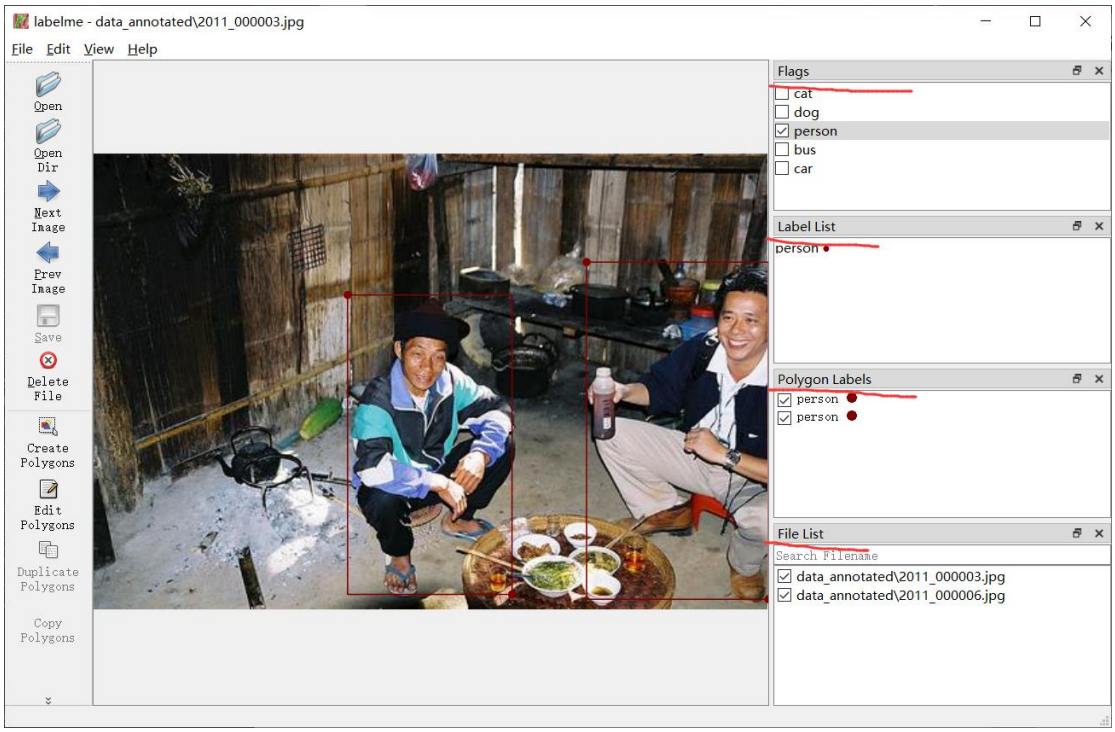
LabelMe 的界面是用 PYQT 工程创建的，在 LabelMe 源代码的窗口类 MainWindow 里，标签、文件列表等视图被分成了四个窗口占在 MainWindow 的右侧




```
self.addDockWidget(Qt.RightDockWidgetArea, self.flag_dock)
self.addDockWidget(Qt.RightDockWidgetArea, self.label_dock)
self.addDockWidget(Qt.RightDockWidgetArea, self.shape_dock)
self.addDockWidget(Qt.RightDockWidgetArea, self.file_dock)
```

目标：实现点击 Flags 的选项，File List 列出所有包含 Flags 的图片。

我们主要对 flag_dock 和 file_dock 进行改动。首先在 flag_dock 列出写在 flag.txt 里的所有标签，其次在 filedock 里列出文件路径下的文件。其次我们需要写出输入一个标签 list，输出这个标签 list 的所有图片的函数，最后我们需要将这个函数关联到鼠标点击选择标签的操作上。

在 flag_dock 列出标签和在 filedock 里列出文件路径下的文件，这两个操作 labelme 是自带的：

```
self.flag_dock = self.flag_widget = None
self.flag_dock = QtWidgets.QDockWidget(self.tr("Flags"), self)
self.flag_dock.setObjectName("Flags")
self.flag_widget = QtWidgets.QListWidget()
if config["flags"]:
    self.loadFlags({k: False for k in config["flags"]})
self.flag_dock.setWidget(self.flag_widget)
self.flag_widget.itemChanged.connect(self.setDirty)
```

```
self.file_dock = QtWidgets.QDockWidget(self.tr("File List"), self)
self.file_dock.setObjectName("Files")
fileListWidget = QtWidgets.QWidget()
fileListWidget.setLayout(fileListLayout)
self.file_dock.setWidget(fileListWidget)
```

接着我们写出输入一个标签 list，列出所有包含标签 list 图片的函数：

```
def importSpecificImages(self, dirpath, pattern=None, load=True):
    if not self.mayContinue() or not dirpath:
        return
    self.lastOpenDir = dirpath
    self.filename = None
    self.fileListWidget.clear()
    for filename in self.scanAllImages(dirpath):
        label_file = osp.splitext(filename)[0] + ".json"
        temp = False
        imgflag = []
        if os.path.exists(label_file):
            data = json.load(open(label_file, 'r'))
            for i in data['shapes']:
                imgflag.append(i['label'])
        if set(pattern) <= set(imgflag):
            temp = True
        if not temp:
            continue
        if self.output_dir:
            label_file_without_path = osp.basename(label_file)
            label_file = osp.join(self.output_dir, label_file_without_path)
        item = QtWidgets.QListWidgetItem(filename)
        item.setFlags(Qt.ItemIsEnabled | Qt.ItemIsSelectable)
        if QtCore.QFile.exists(label_file) and LabelFile.is_label_file(
            label_file
        ):
            item.setCheckState(Qt.Checked)
        else:
            item.setCheckState(Qt.Unchecked)
        self.fileListWidget.addItem(item)
```

函数的关键是红框框出来的部分，他通过输入标签列表 pattern 筛选了所有的图片，判断是否需要被添加到 MainWindow 的 File List 里。

最后，我们需要将这个函数与鼠标点选复选框的操作相关联。
在 PYQT 里，复选框的类会根据鼠标的点选发射信号，如下图：

```
self.boxlist.append(QtWidgets.QCheckBox(key))
self.boxlist[count].stateChanged.connect(self.changebox)
```

途中的 boxlist 为复选框列表，statechanged 为检测复选框选中状态的信号。我们将这个信号通过 connect 操作连接到 self.changebox 的槽函数上。即如果复选框选中状态改变，那么触发后面的槽函数。接着我们就需要后面的槽函数来与上面筛选图片 list 的函数建立联系：

```
def changebox(self):
    a = []
    for i, k in enumerate(self._config["flags"]):
        if self.boxlist[i].isChecked():
            a.append(k)
    self.importSpecificImages(self.rootfilename, a)
```

我们将所有选中的复选框的 flags 添加到 list a 里，然后将 list a 和图片目录 rootfilename 输入上面筛选图片的 list 来改变图片 list。