LabelImg软件开发文档

# 一、加入用户名输入框

1. 在\_\_init\_\_函数中添加用户名标签和输入框

|  |
| --- |
| user = QLabel(u'用户')      self.userTextLine = QLineEdit()      self.connect(self.userTextLine, SIGNAL('textChanged(QString)'), self.user\_onChanged)        userQHBoxLayout = QHBoxLayout()      userQHBoxLayout.addWidget(user)      userQHBoxLayout.addWidget(self.userTextLine)      userContainer = QWidget()  userContainer.setLayout(userQHBoxLayout) |

|  |
| --- |
| listLayout.addWidget(userContainer) |

2. 检测输入框中文字的变化

|  |
| --- |
| def user\_onChanged(self):      b = self.userTextLine.text()  self.setDirty() |

3. 将用户名写入标注文件中

（1）在saveLabels函数中

|  |
| --- |
| self.labelFile.savePascalVocFormat(self.userTextLine.text(), self.phynumTextLine.text(), self.remarkTextLine.text(), annotationFilePath, shapes, self.filePath, self.imageData, self.lineColor.getRgb(), self.fillColor.getRgb()) |

（2）在libs/labelFile.py的savePascalVocFormat函数中

|  |
| --- |
| def savePascalVocFormat(self, user, phynum, remark, filename, shapes, imagePath, imageData, lineColor=None, fillColor=None, databaseSrc=None): |

|  |
| --- |
| writer = PascalVocWriter(user, imgFolderName, imgFileName,  imageShape, localImgPath=imagePath) |

（3）在libs/pascal\_voc\_io.py的PascalVocWriter类的\_init\_函数中

|  |
| --- |
| self.user = user |

（4）在libs/pascal\_voc\_io.py的PascalVocWriter类的genXML函数中

|  |
| --- |
| user = SubElement(top, 'user')  user.text = ustr(self.user) |

# 二、加入实际读数显示和输入框

1. 在\_\_init\_\_函数中添加实际标签的显示和输入框

|  |
| --- |
| phynum = QLabel(u'实际读数')      phynum.setFont(QFont("Roman times",13,QFont.Normal))      self.phynumTextLine = QLineEdit()      self.phynumTextLine.setFont(QFont("Roman times",13,QFont.Normal))      self.connect(self.phynumTextLine, SIGNAL('textChanged(QString)'), self.phynum\_onChanged)      phynumQHBoxLayout = QHBoxLayout()      phynumQHBoxLayout.addWidget(phynum)      phynumQHBoxLayout.addWidget(self.phynumTextLine)      phynumContainer = QWidget()  phynumContainer.setLayout(phynumQHBoxLayout) |

|  |
| --- |
| listLayout.addWidget(phynumContainer) |

2. 检测输入框中文字的变化

|  |
| --- |
| def phynum\_onChanged(self):      b = self.phynumTextLine.text()  self.setDirty() |

3. 读取标注文件中的实际读数并显示

（1）在loadPascalXMLByFilename函数中

|  |
| --- |
| if tVocParseReader.phynum:          self.phynumTextLine.setText(tVocParseReader.phynum)      else:          self.phynumTextLine.setText('')  self.setClean() |

（2）在libs/pascal\_voc\_io.py的PascalVocReader类的\_init\_函数中

|  |
| --- |
| self.phynum = '' |

（3）在libs/pascal\_voc\_io.py的PascalVocReader类的parseXML函数中

|  |
| --- |
| self.phynum = xmltree.find('phynum').text |

4. 将实际读数写入标注文件中

（1）在saveLabels函数中

|  |
| --- |
| self.labelFile.savePascalVocFormat(self.userTextLine.text(), self.phynumTextLine.text(), self.remarkTextLine.text(), annotationFilePath, shapes, self.filePath, self.imageData, self.lineColor.getRgb(), self.fillColor.getRgb()) |

（2）在libs/labelFile.py的savePascalVocFormat函数中

|  |
| --- |
| def savePascalVocFormat(self, user, phynum, remark, filename, shapes, imagePath, imageData, lineColor=None, fillColor=None, databaseSrc=None): |

|  |
| --- |
| writer.phynum = phynum |

（3）在libs/pascal\_voc\_io.py的PascalVocWriter类的\_init\_函数中

|  |
| --- |
| self.phynum = '' |

（4）在libs/pascal\_voc\_io.py的PascalVocWriter类的genXML函数中

|  |
| --- |
| phynum = SubElement(top, 'phynum')  phynum.text = ustr(self.phynum) |

# 三、加入备注显示框

1. 在\_\_init\_\_函数中添加备注的显示框

|  |
| --- |
| remark = QLabel(u'备注')      remark.setFont(QFont("Roman times",13,QFont.Normal))      self.remarkTextLine = QLineEdit()      self.remarkTextLine.setFont(QFont("Roman times",13,QFont.Normal))      self.remarkTextLine.setFocusPolicy(False)      remarkQHBoxLayout = QHBoxLayout()      remarkQHBoxLayout.addWidget(remark)      remarkQHBoxLayout.addWidget(self.remarkTextLine)      remarkContainer = QWidget()  remarkContainer.setLayout(remarkQHBoxLayout) |

|  |
| --- |
| listLayout.addWidget(remarkContainer) |

2. 读取标注文件中的备注并显示

（1）在loadPascalXMLByFilename函数中

|  |
| --- |
| if tVocParseReader.remark:          self.remarkTextLine.setText(tVocParseReader.remark)      else:          self.remarkTextLine.setText('') |

（2）在libs/pascal\_voc\_io.py的PascalVocReader类的\_init\_函数中

|  |
| --- |
| self.remark = '' |

（3）在libs/pascal\_voc\_io.py的PascalVocReader类的parseXML函数中

|  |
| --- |
| self.remark = xmltree.find('remark').text |

3. 将备注写入标注文件（备注不可修改，这里写入备注用于保存时生成新的标注文件）

（1）在saveLabels函数中

|  |
| --- |
| self.labelFile.savePascalVocFormat(self.userTextLine.text(), self.phynumTextLine.text(), self.remarkTextLine.text(), annotationFilePath, shapes, self.filePath, self.imageData, self.lineColor.getRgb(), self.fillColor.getRgb()) |

（2）在libs/labelFile.py的savePascalVocFormat函数中

|  |
| --- |
| def savePascalVocFormat(self, user, phynum, remark, filename, shapes, imagePath, imageData, lineColor=None, fillColor=None, databaseSrc=None): |

|  |
| --- |
| writer.remark = remark |

（3）在libs/pascal\_voc\_io.py的PascalVocWriter类的\_init\_函数中

|  |
| --- |
| self.remark = '' |

（4）在libs/pascal\_voc\_io.py的PascalVocWriter类genXML函数中

|  |
| --- |
| remark = SubElement(top, 'remark')  remark.text = ustr(self.remark) |

# 四、加入从上一张复制标签功能

1. 在\_\_init\_\_函数中添加从上一张复制标签按钮

|  |
| --- |
| self.copyxmlButton = QPushButton(u'从上一张复制标签', self)      self.copyxmlButton.setToolTip("Copy xml from the last image")      self.copyxmlButton.setShortcut('Ctrl+X')  self.copyxmlButton.clicked.connect(self.copy\_xml) |

|  |
| --- |
| copyQHBoxLayout = QHBoxLayout()      copyQHBoxLayout.addWidget(self.copyxmlButton)      copyQHBoxLayout.addWidget(self.copyimgnButton)      copyQHBoxLayout.setStretchFactor(self.copyxmlButton, 2)      copyQHBoxLayout.setStretchFactor(self.copyimgnButton, 1)      copyContainer = QWidget()  copyContainer.setLayout(copyQHBoxLayout) |

|  |
| --- |
| listLayout.addWidget(copyContainer) |

2. 实现从上一张复制标签功能

|  |
| --- |
| def copy\_xml(self):       if self.autoSaving.isChecked():  if self.defaultSaveDir is not None:  if self.dirty is True:  self.saveFile()  else:  self.changeSavedirDialog()  return  if not self.mayContinue():  return  if len(self.mImgList) <= 0:  return  if self.filePath is None:  return  currIndex = self.mImgList.index(self.filePath)  if currIndex - 1 >= 0:  filename = self.mImgList[currIndex - 1]  if filename:  #self.loadFile(filename)          preimgFileName = os.path.basename(filename)          prexmlFileName = os.path.splitext(preimgFileName)[0] + XML\_EXT            if self.defaultSaveDir is not None and len(ustr(self.defaultSaveDir)):           if self.filePath:           imgFileName = os.path.basename(self.filePath)           savedFileName = os.path.splitext(imgFileName)[0] + XML\_EXT           savedPath = os.path.join(ustr(self.defaultSaveDir), savedFileName)           #self.\_saveFile(savedPath)              savedxmlPath = savedPath          else:           imgFileDir = os.path.dirname(self.filePath)           imgFileName = os.path.basename(self.filePath)           savedFileName = os.path.splitext(imgFileName)[0] + XML\_EXT           savedPath = os.path.join(imgFileDir, savedFileName)           #self.\_saveFile(savedPath if self.labelFile           # else self.saveFileDialog())           savedxmlPath = (savedPath if self.labelFile           else self.saveFileDialog())          savedxmlDir = os.path.dirname(savedxmlPath)          prexmlFilePath = os.path.join(savedxmlDir, prexmlFileName)            shutil.copyfile(prexmlFilePath, savedxmlPath)          #set difficult to 0       parser = etree.XMLParser(encoding='utf-8')       xmltree = ElementTree.parse(savedxmlPath, parser=parser).getroot()          objects = xmltree.findall('object')          for object in objects:           object.find('difficult').text = '0'          eletree = xmltree.getroottree()       eletree.write(savedxmlPath, encoding="utf-8")          self.loadFile(self.mImgList[currIndex]) |

# 五、加入复制图片名功能

1. 在\_\_init\_\_函数中添加复制图片名按钮

|  |
| --- |
| self.copyimgnButton = QPushButton(u'复制图片名', self)      self.copyimgnButton.setToolTip("Copy the name of current image")      self.copyimgnButton.setShortcut('Ctrl+Shift+C')  self.copyimgnButton.clicked.connect(self.copy\_imgn) |

|  |
| --- |
| copyQHBoxLayout = QHBoxLayout()      copyQHBoxLayout.addWidget(self.copyxmlButton)      copyQHBoxLayout.addWidget(self.copyimgnButton)      copyQHBoxLayout.setStretchFactor(self.copyxmlButton, 2)      copyQHBoxLayout.setStretchFactor(self.copyimgnButton, 1)      copyContainer = QWidget()  copyContainer.setLayout(copyQHBoxLayout) |

|  |
| --- |
| listLayout.addWidget(copyContainer) |

2. 实现复制图片名功能

|  |
| --- |
| def copy\_imgn(self):       clipboard = QApplication.clipboard()  clipboard.setText(os.path.basename(self.filePath)) |

# 六、加入隐藏标注框功能

1. 在\_\_init\_\_函数中添加不显示标注框选项

|  |
| --- |
| self.showboxesButton = QCheckBox(u'不显示标注框')  self.showboxesButton.setChecked(False) self.showboxesButton.stateChanged.connect(self.showboxes\_btnstate) |

|  |
| --- |
| diffc\_showbQHBoxLayout = QHBoxLayout()      diffc\_showbQHBoxLayout.addWidget(self.diffcButton)      diffc\_showbQHBoxLayout.addWidget(self.showboxesButton)      diffc\_showbContainer = QWidget()      diffc\_showbContainer.setLayout(diffc\_showbQHBoxLayout)  listLayout.addWidget(diffc\_showbContainer) |

2. 实现不显示标注框功能

|  |
| --- |
| def showboxes\_btnstate(self):  """ Function to show or don't show boxes """      self.canvas.setAllShapeVisible(self.showboxesButton.isChecked() == 0) |

3. 在libs/canvas.py中

|  |
| --- |
| def setAllShapeVisible(self, value):      for shape in self.shapes:       self.setShapeVisible(shape, value)  self.repaint() |

4. 加载标签时检查是否需要显示，在loadLabels函数中

|  |
| --- |
| self.showboxes\_btnstate() |

# 七、加入固定缩放比例功能

1. 在\_\_init\_\_函数中添加固定缩放比例按钮

|  |
| --- |
| fixSize = action(u'固定缩放比例', self.setFixSize,               'Ctrl+Shift+X', 'fix-size', u'Fix size',               checkable=True, enabled=False) |

2. 在\_\_init\_\_函数中添加zoomActions

|  |
| --- |
| zoomActions = (self.zoomWidget, zoomIn, zoomOut,  zoomOrg, fitWindow, fitWidth, fixSize) |

3. 在\_\_init\_\_函数中添加actions

|  |
| --- |
| self.actions = struct(save=save, saveAs=saveAs, open=open, close=close, resetAll = resetAll, lineColor=color1, create=create, delete=delete, edit=edit, copy=copy, createMode=createMode, editMode=editMode, advancedMode=advancedMode, shapeLineColor=shapeLineColor, shapeFillColor=shapeFillColor, zoom=zoom, zoomIn=zoomIn, zoomOut=zoomOut, zoomOrg=zoomOrg, fitWindow=fitWindow, fitWidth=fitWidth, fixSize = fixSize, ... |

|  |
| --- |
| addActions(self.menus.view, (  self.autoSaving,  self.singleClassMode,  labels, advancedMode, None,  hideAll, showAll, None,  zoomIn, zoomOut, zoomOrg, None,  fitWindow, fitWidth, fixSize)) |

|  |
| --- |
| self.actions.beginner = (  open, opendir, changeSavedir, openNextImg, openPrevImg, verify, save, None, create, copy, delete, None, zoomIn, zoom, zoomOut, fixSize, fitWindow, fitWidth) |

4. 在\_\_init\_\_函数中设置self.fixsize的初始值为0，代表不固定缩放比例

|  |
| --- |
| self.fixsize = 0 |

5. 实现固定缩放比例功能

|  |
| --- |
| def setFixSize(self, value=True):  self.fixsize = value |

6. 在loadFile函数中，如果不固定缩放比例，则适应窗口大小

|  |
| --- |
| if self.fixsize == 0:  self.adjustScale(initial=True) |

# 八、加入再次打开软件恢复上次标注到的图像、缩放大小和用户名功能

1. 在libs/constants.py中加入变量

|  |
| --- |
| SETTING\_LAST\_OPEN\_FILE = 'lastOpenFile'  SETTING\_LAST\_ZOOM\_VALUE = 'lastZoomValue'  SETTING\_USER = 'user' |

2. 再次打开软件时恢复，在\_init\_函数中

|  |
| --- |
| if os.path.exists(QString.fromUtf8(self.lastOpenDir)):          self.importDirImages(self.lastOpenDir)      self.last\_open\_file = settings.get(SETTING\_LAST\_OPEN\_FILE)      if os.path.exists(QString.fromUtf8(self.last\_open\_file)):          self.loadFile(self.last\_open\_file)      if settings.get(SETTING\_LAST\_ZOOM\_VALUE):          self.setZoom(settings.get(SETTING\_LAST\_ZOOM\_VALUE))        if settings.get(SETTING\_USER):          self.userTextLine.setText(settings.get(SETTING\_USER))          self.setClean() |

3. 切换图像之前记录当前图像名，作为上次标注到的图像

在fileitemDoubleClicked函数中

|  |
| --- |
| self.last\_open\_file = filename |

在openPrevImg函数中

|  |
| --- |
| self.last\_open\_file = filename |

在openNextImg函数中

|  |
| --- |
| self.last\_open\_file = filename |

4. 在关闭软件时将记录写入配置文件中，在closeEvent函数中

|  |
| --- |
| settings[SETTING\_LAST\_OPEN\_FILE] = self.last\_open\_file      settings[SETTING\_LAST\_ZOOM\_VALUE] = self.zoomWidget.value()  settings[SETTING\_USER] = self.userTextLine.text() |

# 九、在“File List”里切换图像自动保存标注

在fileitemDoubleClicked函数中

|  |
| --- |
| if self.autoSaving.isChecked():  if self.defaultSaveDir is not None:  if self.dirty is True:  self.saveFile()  else:  self.changeSavedirDialog()  return |

# 十、改变标注框和填充颜色

1. 改变标注框和填充颜色，在chooseColor1函数中

设置透明度默认为255

|  |
| --- |
| color.setAlpha(255) |

设置颜色并重新加载

|  |
| --- |
| self.settings[SETTING\_LINE\_COLOR] = color  self.loadFile(self.filePath) |

2. 加载标签时设置颜色（包含困难标签），在loadLabels函数中

|  |
| --- |
| if shape.difficult == True:           shape.line\_color = QColor(255, 255, 0, 255)       elif self.settings.get(SETTING\_LINE\_COLOR):           self.lineColor = self.settings[SETTING\_LINE\_COLOR]          shape.line\_color = self.lineColor       else:          self.lineColor = QColor(0, 255, 0, 255)          shape.line\_color = self.lineColor |

|  |
| --- |
| if shape.difficult == True:           shape.fill\_color = QColor(255, 255, 0, 25)       else:          shape.fill\_color = QColor(self.lineColor.red(), self.lineColor.green(), self.lineColor.blue(), 25) |

3. 设置其他对应的颜色

在editLabel函数中

|  |
| --- |
| item.setBackground(self.lineColor) |

在addLabel函数中

|  |
| --- |
| item.setBackground(self.lineColor) |

在labelItemChanged函数中

|  |
| --- |
| shape.line\_color = self.lineColor |

在newShape函数中

|  |
| --- |
| generate\_color = self.lineColor |

# 十一、给困难标签设置特殊的颜色

1. 改变difficult状态后需要保存后重新加载才会改变颜色，在btnstate函数中

|  |
| --- |
| self.saveFile()  self.loadFile(self.filePath) |

2. 加载标签时设置颜色（包含困难标签），在loadLabels函数中

|  |
| --- |
| if shape.difficult == True:           shape.line\_color = QColor(255, 255, 0, 255)       elif self.settings.get(SETTING\_LINE\_COLOR):           self.lineColor = self.settings[SETTING\_LINE\_COLOR]          shape.line\_color = self.lineColor       else:          self.lineColor = QColor(0, 255, 0, 255)          shape.line\_color = self.lineColor |

|  |
| --- |
| if shape.difficult == True:           shape.fill\_color = QColor(255, 255, 0, 25)       else:          shape.fill\_color = QColor(self.lineColor.red(), self.lineColor.green(), self.lineColor.blue(), 25) |

# 十二、支持中文路径

在loadFile函数中

|  |
| --- |
| newWindowTitle = \_\_appname\_\_ + ' ' + filePath  self.setWindowTitle(QString.fromUtf8(newWindowTitle)) |

# 十三、调整拖动标注框边缘时的灵敏度

在libs/canvas.py中

|  |
| --- |
| epsilon = 5.0 |

# 十四、解决微调不能保存的问题

原因是int()默认向下取整，应该用round()，在libs/labelFile.py的convertPoints2BndBox函数中

|  |
| --- |
| return int(round(xmin)), int(round(ymin)), int(round(xmax)), int(round(ymax)) |