# 软件开发文档

目录

[软件开发文档 1](#_Toc30151651)

[一、整体架构 2](#_Toc30151652)

[1、app/ 2](#_Toc30151653)

[2、docs/ 2](#_Toc30151654)

[3、images/ 2](#_Toc30151655)

[4、lib/ 2](#_Toc30151656)

[5、qss/ 2](#_Toc30151657)

[二、新增不同类别的Widget步骤 2](#_Toc30151658)

[三、Widget Id 3](#_Toc30151659)

[1、what 3](#_Toc30151660)

[2、how 3](#_Toc30151661)

[四、Func类 3](#_Toc30151662)

[generateWidgetId(widget\_type: int) -> int 3](#_Toc30151663)

[generateWidgetName(widget\_type: int) -> str: 3](#_Toc30151664)

[checkWidgetNameValidity(widget\_name: str) -> (bool, str): 3](#_Toc30151665)

[getImage(image\_path: str, type: int = 0, size: QSize = None) -> QPixmap or QIcon: 4](#_Toc30151666)

[print(information: str, information\_type: int = 0) -> None: 4](#_Toc30151667)

[changeWidgetName(widget\_id: int, widget\_name: str): 4](#_Toc30151668)

[getTrackingPix(text: str) -> QPixmap: 4](#_Toc30151669)

[getWidgetAttributes(widget\_id: int) -> dict: 4](#_Toc30151670)

[getWidgetProperties(widget\_id: int) -> dict: 4](#_Toc30151671)

[isWidgetType(widget\_id: int, widget\_type: int) -> bool: 5](#_Toc30151672)

[五、Widget 5](#_Toc30151673)

[1、创建widget 5](#_Toc30151674)

[2、widget的必要信号连接 5](#_Toc30151675)

[六、五大主要窗口 5](#_Toc30151676)

[1、Center 5](#_Toc30151677)

[2、Structure 5](#_Toc30151678)

[3、Properties 5](#_Toc30151679)

[4、Attributes 5](#_Toc30151680)

[5、Output 5](#_Toc30151681)

[七、各类widget 5](#_Toc30151682)

[1、timeline 6](#_Toc30151683)

[2、Cycle 6](#_Toc30151684)

## 一、整体架构

### 1、app/

Info (app/info.py) 类存放配置信息

Kernel (app/kernel.py) 类存放数据信息

Func (app/func.py) 类存放通用函数

Psy（main.py）类为主类，其中包含了各个控件之间的信号连接和信号处理。

其余文件夹按模块划分

### 2、**docs**/

存放文档

### 3、images/

按类别存放图片

### 4、lib/

存放项目通用模块

### 5、qss/

存放qss文件

## 二、新增不同类别的Widget步骤

1、继承lib.TabItemWidget类，完成对应模块

2、 在Info中增加改类别xxx = int\_value，并新增WidgetType[int\_value] = “type\_name”

3、在Kernel中修改WidgetTypeCount和WidgetNameCount

4、在app/main.py中导入相应类，并修改Psy类的的createWidget函数

5、如果有特殊的信号需要增加，修改Psy类的linkWidgetSignals函数

6、在images/widgets中放入图标，格式为png

7、在app/center/timeline/icon\_bar.py中修改\_\_init\_\_函数，新增图标

## 三、Widget Id

### 1、what

Widget id是用来和相应widget进行绑定的关键字，通过Func.generateWidgetId (widget\_type)来生成。

### 2、how

在timeline控件中，从上方的icon bar拖入下方区域便会生成对于widget id并创建widget。

当发生复制行为时，生成对于widget id并复制控件（控件内部提供复制函数）

当发生引用行为时，生成对于widget id并引用控件（通过将不同的widget id指向同一widget实现）

## 四、Func类

generateWidgetId(widget\_type: int) -> int:

it' used to generate widget id which is used to discern different widget.

@param widget\_type: the add\_type of widget, such as timeline.

@return: new widget id

### generateWidgetName(widget\_type: int) -> str:

it' used to generate widget id which is used to discern different widget.

@param widget\_type: the add\_type of widget, such as timeline.

@return: new widget id

### checkWidgetNameValidity(widget\_name: str) -> (bool, str):

check the validity of widget name.

It should be unique, unless it's a reference.

@param widget\_name:

@return: validity

### getImage(image\_path: str, type: int = 0, size: QSize = None) -> QPixmap or QIcon:

get image from its relative path, return qt image object, include QPixmap or QIcon.

@param image\_path: its relative path

@param type: 0: pixmap (default),

1: icon

@return: Qt image object

### print(information: str, information\_type: int = 0) -> None:

print information in output.

@param information:

@param information\_type: 0 none

1 success

2 fail

### changeWidgetName(widget\_id: int, widget\_name: str):

change widget.widget\_name

@param widget\_id:

@param widget\_name:

### getTrackingPix(text: str) -> QPixmap:

generate pixmap from text

@param text:

@return:

### getWidgetAttributes(widget\_id: int) -> dict:

get widget's attributes and attributes' layer

@param widget\_id:

@return: {attribute: layer}

### getWidgetProperties(widget\_id: int) -> dict:

get widget's properties

@param widget\_id:

@return:

### isWidgetType(widget\_id: int, widget\_type: int) -> bool:

judge widget\_id's widget type

@param self:

@param widget\_id:

@param widget\_type:

@return: true or false

## 五、Widget

### 1、创建widget

调用Func类的createWidget函数，需提供widget\_id和widget\_name（二者可自行提供或先通过Func.generateWidgetId函数和Func.generateWidgetName函数生成）

### 2、widget的必要信号连接

调用Func类的linkWidgetSignals，这个函数实际是调用了Psy（主窗口类）的linkWidgetSignals

## 六、五大主要窗口

### 1、Center

### 2、Structure

### 3、Properties

### 4、Attributes

### 5、Output

七、各类widget

### 1、timeline

#### a）构造方法

首先继承lib.TabItemWidget，timeline包含两个主要部分，提供图标的icon bar和 图标拖拽区域的timeline area。

Icon bar是继承QTabWidget后每个tab包含一个QListView，将QListView设置为icon模式后，放入图标。Icon需要可拖拽，对其进行相关设置

Timeline area是继承QFrame后，放入一个QTableWidget，通过table对放入的item进行管理和控制。

Timeline area中的箭头标志是通过在table中设置图片对象后产生的。

为了对点击事件和修改时间作出更好的控制，重写了TimelineItem（QLabel）和TimelineNameItem（QTableItemWidget）。

Timeline通过addItem函数添加新item，deleteItem删除item，其实都是一层层下放到timeline area和timeline table，其中addItem返回新增item的widget id， widget name和插入timeline table的index。

#### b）信号量

itemNameChanged (widget id: int, widget name: str)

itemClicked (widget id: int)

itemDoubleClicked (widget id: int)

itemMoved (widget id: int, origin index: int, new index: int)

itemDeleted(widget id: int)

itemAdded(parent\_widget\_id, widget\_id, widget\_name, index, add\_type)

#### c）可用API

暂无

### 2、Cycle

#### a）构造方法

#### b）信号量

#### c）可用API

#### d）关于引用(reference)

引用就是将不同的widget id指向同一个widget达到一种引用的效果，需要注意以下几点

1. Timeline不能直接引用，可以通过引用cycle间接引用
2. 引用需要有相同的属性环境，即父节点存在相同的cycle（引用的cycle也算相同）
3. 在timeline中修改名称后，是断开引用，但是由于cycle可以引用，