Toccata Manual de referencia

v0.1b

Generado por Doxygen 1.5.1

Tue Jun 30 19:14:52 2009

Índice general

Capítulo 1

Toccata Indice de namespace

1.1. Toccata Lista de Paquetes

Aquí van los paquetes con una breve descripción (si etá disponible):

Capítulo 2

Toccata Indice de clases

2.1. Toccata Lista de componentes

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 3

Toccata Documentación de namespace

3.1. Paquetes Highlighter.py

3.1.1. Descripción detallada

La clase encargada de resaltar el texto...

Para Toccata, IDE para Lilypond, Futuro IDE para edición completa de Partituras, casi como WYSIWYG (espero)

Extraída de los ejemplos de PyQt4.

Extendida por Javier Salazar Loyola <jsalazar@elo.utfsm.cl>

3.2. Paquetes MainWindow.py

3.2.1. Descripción detallada

Paquete que implementa la clase de la ventana principal. Incluye todos los menús, acciones y barras de herramientas, además de los módulos de edición de texto como objetos hijos de ésta.

Para Toccata, IDE para Lilypond, Futuro IDE para edición completa de Partituras, casi como WYSIWYG (espero)

Autor:

Javier Salazar Loyola <jsalazar@elo.utfsm.cl>

3.3. Paquetes Subprocesos.py

3.3.1. Descripción detallada

Módulo que define una clase para trabajar con subprocesos según la definición de QtCore.QProcess y que define una clase para visualizar la salida de la compilación con Lilypond, heredera de la clase QtGui.QTextEdit

Para Toccata, IDE para Lilypond, Futuro IDE para edición completa de Partituras, casi como WYSIWYG (espero)

Autor:

Javier Salazar Loyola <jsalazar@elo.utfsm.cl>

Capítulo 4

Toccata Documentación de clases

4.1. Referencia de la Clase Highlighter.Highlighter

Métodos públicos

- def __init__
- def addToDocument
- def addMapping
- def highlight
- def loadHighlight
- def highlightBlock

Atributos públicos

- mappings

4.1.1. Descripción detallada

Clase que se encarga de resaltar el texto. Se basa en los métodos para identificar texto de la clase QtGui.QRegExp

4.1.2. Documentación de las funciones miembro

4.1.2.1. def Highlighter.Highlighter.__init__ (*self*, *parent* = `None`)

El constructor, inicializa el diccionario en que se basa el resaltado

Parámetros:

parent QObject padre de la instancia

4.1.2.2. def Highlighter.Highlighter.addToDocument (*self*, *doc*)

Método para conectar un determinado documento QtGui.QTextDocument con el resaltador, de forma de que la instancia sepa dónde debe resaltar.

Parámetros:

doc QTextDocument al que se resaltará el texto

4.1.2.3. def Highlighter.Highlighter.addMapping (*self*, *pattern*, *format*)

Método que actualiza los resaltados que se utilizarán

Parámetros:

pattern Cadena con la información de los parámetros que se destacarán

format Objeto de la clase QtGui.QTextCharFormat que indica cómo resaltar

4.1.2.4. def Highlighter.Highlighter.highlight (*self*, *position*, *removed*, *added*)

El resaltado es actualizado con cada cambio en el documento, este método realida ese cambio. Debido a que este método se conecta con el cada cambio en el documento, aquella señal es la que entrega los parámetros necesarios para realizar la operación.

Parámetros:

position La posición en que se realizó el cambio

removed Cuánto texto se removió en el cambio

added Cuánto texto se agregó en el cambio

4.1.2.5. def Highlighter.Highlighter.loadHighlight (*self*, *doc*)

Método que se realiza al cargar un archivo nuevo, con el objeto de resaltar el texto con cada carga de un nuevo archivo

Parámetros:

doc El texto cargado (QtGui.QTextDocument)

Devuelve:

True cuando el texto ha sido resaltado completo

4.1.2.6. def Highlighter.Highlighter.highlightBlock (*self*, *block*)

Este método es el que realiza el resaltado en sí.

Parámetros:

block El bloque de texto que se resaltará.

4.1.3. Documentación de los datos miembro

4.1.3.1. Highlighter.Highlighter.mappings

Diccionario con el formato del texto resaltado según cada uno de los patrones que se destacarán

La documentación para esta clase fué generada a partir del siguiente archivo:

- Highlighter.py

4.2. Referencia de la Clase Subprocesos.LogViewer

Métodos públicos

- def __init__
- def salvarLog
- def actualizar

Atributos públicos

- **parent**

4.2.1. Descripción detallada

Clase que extiende QtGui.QTextEdit. Esta clase permite tener el registro de salida de la compilación con Lilypond.

4.2.2. Documentación de las funciones miembro

4.2.2.1. def Subprocesos.LogViewer.__init__ (*self*, *parent* = `None`, *parentOfParent* = `None`)

Constructor

Parámetros:

parent Objeto padre de la instancia de este objeto

parentOfParent Objeto padre del padre de la instancia, que finalmente es el padre de este objeto

4.2.2.2. def Subprocesos.LogViewer.salvarLog (*self*)

Método que permite guardar el registro de salida como archivo de texto (.txt) return False si no se guarda el registro

4.2.2.3. def Subprocesos.LogViewer.actualizar (*self*)

Método que actualiza el estado del registro, agregando el resultado de la compilación. Debe mejorarse para futuras versiones

La documentación para esta clase fué generada a partir del siguiente archivo:

- Subprocesos.py

4.3. Referencia de la Clase MainWindow.MainWindow

Métodos públicos

- def __init__
- def readSettings
- def writeSettings
- def setupEditor
- def crearMenuBar

  Método encargado de crear los menús y ordenarlos.

- def crearToolBar

  Método encargado de crear la barra de herramientas a utilizar.

- def center

  Método que centra en la pantalla la ventana.

- def crearAcciones
- def undo

  Método para deshacer.

- def redo

  Método para rehacer.

- def eventoNuevoArchivo

  Crea un archivo en blanco.

- def eventoAbrir
- def abrirArchivo
- def eventoGuardarComo
- def eventoGuardarArchivo
- def eventoGuardar
- def setCurrentFile
- def crearDock

  Método que crea una subventana que muestra el registro de salida de la compilación.

- def mostrarLog

  Método que muestra de nuevo el registro.

- def about

  Método que muestra información acerca de la aplicación.

- def closeEvent

Atributos públicos

- titulo

  Título de la ventana. Para efectos de actualización del ídem.

- archivoActual

  Nombre del archivo abierto en el momento.

- textEdit

  Bloque editor de texto. Encargado de mostrar el archivo .ly abierto.

- textHighlighter
- procesosHijos
- directorio
- menubar

  La barra de menú.

- archivo

  Menú archivo.

- edicion

  Menú edición.

- compilacion

  Menú compilación.

- ayuda

  Menú ayuda.

- toolbar

  La barra de herramientas.

- salir

  Acción para salir del programa.

- guardar

  Acción para guardar el archivo actual (previamente guardado o no).

- guardarComo

  Acción para guardar el archivo actual eligiendo su nombre.

- abrir

  Acción para abrir un archivo.

- nuevo

  Acción para crear un archivo en blanco.

- deshacer

  Acción para deshacer la última edición.

- rehacer

  Acción para rehacer la última acción deshechada.

- cortar

  Acción para cortar el texto seleccionado.

- copiar

  Acción para copiar el texto seleccionado.

- pegar

  Acción para pegar el texto previamente copiado.

- selAll

  Acción para seleccionar todo el texto en el editor.

- lilypond

  Acción que inicia la compilación del archivo actual.

- verPdf

  Acción para el pdf creado.

- verPs

  Acción para el ps creado.

- verLog

  Acción para mostrar el registro.

- saveLog

  Acción para guardar el registro de compilación.

- acercaDe

  Información acerca del programa.

- acercaDeQt

  Información acerca de Qt.

- dock

  Subventana que contiene el visor de registro de salida.

- LogView

  El registro de salida.

4.3.1. Descripción detallada

Clase heredera de QtGui.QMainWindow que contiene todos los menús, editores y demases que hacen funcionar el programa, la instancia de ésta clase es la instancia madre de todos los demás objetos.

4.3.2. Documentación de las funciones miembro

4.3.2.1. def MainWindow.MainWindow.__init__ (*self*, *parent* = `None`, *fileName* = `QtCore.QString("")`)

Método constructor

Parámetros:

parent objeto padre

fileName Archivo que se cargará al momento de abrir el programa

4.3.2.2. def MainWindow.MainWindow.readSettings (*self*)

Método que se encarga de cargar las configuraciones del programa. Por ahora, está soportado sólamente el tamaño de la ventana y si es que al momento de cerrarse (ultima configuración guardada) estaba maximizada o no, además del último directorio de trabajo

4.3.2.3. def MainWindow.MainWindow.writeSettings (*self*)

Método que escribe la configuración del programa. Escribe el tamaño de la pantalla, si está maximizada y el directorio de trabajo actual

4.3.2.4. def MainWindow.MainWindow.setupEditor (*self*)

Método que configura el editor de texto y agrega los patrones y configuraciones de color al objeto resaltador

4.3.2.5. def MainWindow.MainWindow.crearAcciones (*self*)

Método que crea las acciones del programa y las conecta con sus correspondientes métodos

4.3.2.6. def MainWindow.MainWindow.eventoAbrir (*self*)

Abre un archivo existente.

Devuelve:

False si no se seleccionó archivo, en caso contrario, devuelve lo que devuelve el método abrirArchivo

4.3.2.7. def MainWindow.MainWindow.abrirArchivo (*self*, *fileName*)

El método que realiza la apertura efectiva del archivo.

Parámetros:

fileName El nombre del archivo que se abrirá return True si se abre el archivo, False si no se puede

4.3.2.8. def MainWindow.MainWindow.eventoGuardarComo (*self*)

Método que pregunta el nombre del archivo a guardar y luego realiza el salvado del archivo.

Devuelve:

False si no se elige nombre para guardar, en otro caso, el valor de retorno de eventoGuardarArchivo

4.3.2.9. def MainWindow.MainWindow.eventoGuardarArchivo (*self*, *fileName*)

Método que realiza el guardado del archivo. Si no se especifica, se añade automáticamente la extensión .ly al nombre del archivo.

Parámetros:

fileName El nombre del archivo a guardar

Devuelve:

True si se guarda, False en otro caso

4.3.2.10. def MainWindow.MainWindow.eventoGuardar (*self*)

Método llamado con la acción guardar: verifica que el archivo tenga o no nombre.

Devuelve:

lo que devuelve el eventoGuardarComo si es que no hay archivo abierto, o lo que devuelve eventoGuardarArchivo si es que hay ya un archivo abierto

4.3.2.11. def MainWindow.MainWindow.setCurrentFile (*self*, *fileName*)

Coloca el título de la ventana y actualiza el archivo actual.

Parámetros:

fileName El nombre del archivo que se colocará en el título

4.3.2.12. def MainWindow.MainWindow.closeEvent (*self*, *event*)

Define un evento de cierre diferente. Pregunta si sale en caso de haber modificado el documento actual. El objetivo es dejarlo como para que si hay cambios sin salvar, que pregunte

Parámetros:

event el evento de cierre que se llama

Devuelve:

False si es que no se acepta el evento de cierre; en otro caso, no retorna nada

4.3.3. Documentación de los datos miembro

4.3.3.1. MainWindow.MainWindow.textHighlighter

Instancia del resaltador de texto. El objeto que se encarga de mostrarlo todo en colores

Ver también:

Highlighter.py

4.3.3.2. MainWindow.MainWindow.procesosHijos

Objeto encargado de manejar los subprocesos requeridos. Por ahora, sólo están soportado Lilypond, evince y xpdf (futuras versiones serán configurables)

4.3.3.3. MainWindow.MainWindow.directorio

Directorio de trabajo. El directorio que se abrirá por defecto con los cuadros de diálogo y el directorio de trabajo de los procesos hijos

La documentación para esta clase fué generada a partir del siguiente archivo:

- MainWindow.py

4.4. Referencia de la Clase Subprocesos.ProcessHandler

Métodos públicos

- def __init__
- def compilar
- def verPDF

  Método que crea un proceso hijo que muestra el archivo PDF generado.

- def verPS

  Método que crea un proceso hijo que muestra el archivo PS generado.

Atributos públicos

- parent
- compilador

  El proceso que compila.

- **visorPDF**
- **visorPS**
- programas

  Diccionario con los nombres de los programas a utilizar.

4.4.1. Descripción detallada

Clase que maneja los subprocesos del programa, siendo éstos el compilador (Lilypond) y los visores de PDF y PS.

4.4.2. Documentación de las funciones miembro

4.4.2.1. def Subprocesos.ProcessHandler.__init__ (*self*, *parent*, *nombreCompilador* = `"lilypond"`, *nombreVisorPDF* = `"xpdf"`, *nombreVisorPS* = `"evince"`)

Crea un nuevo objeto con los programas requeridos.

Parámetros:

parent Objeto padre de la instancia, de la clase MainWindow

nombreCompilador el comando a correr para compilar

nombreVisorPDF el comando a correr para visualizar el PDF

nombreVisorPS el comando a correr para mostrar el PS generado

4.4.2.2. def Subprocesos.ProcessHandler.compilar (*self*)

Método que realiza la compilación. Pregunta primero si se debe guardar el archivo, para evitar problemas de archivo inexistente

Devuelve:

False si no se puede compilar por problemas de guardado de archivo

4.4.3. Documentación de los datos miembro

4.4.3.1. Subprocesos.ProcessHandler.parent

El objeto padre. Debe ser una ventana principal de la clase MainWindow, de lo contrario, se generarán problemas por llamados a métodos que otras clases no tienen

Ver también:

MainWindow.py

La documentación para esta clase fué generada a partir del siguiente archivo:

- Subprocesos.py

Índice alfabético