API Documentation

API Documentation

July 21, 2010

Contents

1 Package iprstats

1.1 Modules

- **core** *(Section 2, p. 3)*
- **exporters** *(Section 3, p. 13)*
- **importers** *(Section 4, p. 16)*
- **iprstats** *(Section 5, p. 19)*
- **standalone** *(Section 6, p. 21)*

1.2 Variables

Name	Description
__package__	**Value:** `None`

2 Module iprstats.core

2.1 Variables

Name	Description
pylab_avail	**Value:** `True`
wx_avail	**Value:** `True`
__package__	**Value:** `'iprstats'`

2.2 Class Cache

Known Subclasses: iprstats.core.SQLiteCache

Object to temporarily store aggregate information retrieved by complex, time-intensive queries. This is the top-level class that stores all information in memory. Subclasses of this class should attempt to preserve memory by caching to disk.

Methods you are required to override when making a subclass are: __open_cache__(self) __create_count_group__(self, app) __insert_count_record__(self, app, name, count) __create_match_group__(self, app) __insert_match_record__(self, app, dbid, name, count, goid, goname) get_one_row(self, app, rownum) get_counts(self, app)

It is also recommended to override the following methods: __commit_records__(self) __close_writing__(self)

2.2.1 Methods

__init__(*self*, *settings*)

Initialize the cache object with a settings object in order for it to get a database connection and the working session directory.

__get_db_conn__(*self*)

Attempt to get a database connection (either MySQL or SQLite based on settings) and return the connection. Defaults to SQLite if it cannot get a MySQL connection.

__get_go_db_conn__(*self*)

Retrieve a connection (conn, cursor) to the gene ontology terms database using connection details specified in the settings if the settings if GO lookup is set as True. Returns (None, None) otherwise.

__get_mysql_conn__(*self*, *dbsettings*)

Generic class for retrieving a MySQL connection object given a DBSettings object.

__populate_cache__(*self*)

Main method for executing queries against the IPRStats database and storing them in memory or on disk.

__open_cache__(*self*)

Method for creating the underlying data structure for storing query results. Subclasses must override this method to open handles to disk-based structures or create memory maps.

__count_query__(*self*, *app*)

Method for executing the query used to retrieve information for making charts.

__create_count_group__(*self*, *app*)

Method used for creating a group or section in the underlying data structure for a particular app to store chart data. Subclasses of Cache must override this method to make a group in the disk-based structure.

__insert_count_record__(*self*, *app*, *name*, *count*)

Method for inserting a chart data record into the underlying cache data structure. Subclasses must override this method.

__match_query__(*self*, *app*)

Method for executing the query that retrieves table data from the IPRStats database.

__create_match_group__(*self*, *app*)

Method used for creating a group or section in the underlying data structure for a particular app to store table data. Subclasses of Cache must override this method to make a group in the disk-based structure.

__insert_match_record__(*self*, *app*, *dbid*, *name*, *count*, *goid*, *goname*)

Method for inserting a table data record into the underlying cache data structure. Subclasses must override this method.

__commit_records__(*self*)

Method for committing or flushing records to disk that have been retrieved so far.

__close_writing__(*self*)

Method for closing the underlying data structure for writing and opening it for reading.

__match_length__(*self*, *app*)

__count_length__(*self*, *app*)

get_match_length(*self*, *app*)

Returns the visible number of rows in the stored table data, limited by the ”max table results” setting or the number of available table rows.

get_count_length(*self*, *app*)

Returns the number of results to be displayed in a graph, limited either by the ”max chart results” setting or the number of available results.

get_one_row(*self*, *app*, *rownum*)

Return one row of table data for the specified app. This should be overridden in any Cache subclasses to access the underlying data storage object.

Returns (dbname, count, goid, dbid, goname)

get_one_cell(*self*, *app*, *rownum*, *colnum*)

Returns a particular table cell for application app. This method likely does not need to be overridden in subclasses.

get_url(*self*, *app*, *rownum*, *go_link*=`False`)

Retrieve the linking URL to a particular app or gene ontology website.

__go_name__(*self*, *go_id*)

Retrieve the gene ontology name for the given term id. This should only be called if GO lookup is set to True.

get_counts(*self*, *app*)

Retrieve an array of data for chart generation. Returns [(value1, value2, ...), (label1, label2, ...)] This method must be overridden in any subclasses to retrieve the data from the underlying structure and return it in the specified format.

2.3 Class SQLiteCache

```
iprstats.core.Cache ─┐
                     iprstats.core.SQLiteCache
```

Subclass of Cache that uses a SQLite database to store the results of the MySQL query.

2.3.1 Methods

__init__(*self*, *settings*)

Initialize the cache object with a settings object in order for it to get a database connection and the working session directory.

Overrides: iprstats.core.Cache.__init__ extit(inherited documentation)

__open_cache__(*self*)

Open a connection and cursor to the SQLite database located at self.filename

Overrides: iprstats.core.Cache.__open_cache__

__create_count_group__(*self*, *app*)

Create a group (table) in the open SQLite database for storing count query results.

Overrides: iprstats.core.Cache.__create_count_group__

__insert_count_record__(*self*, *app*, *name*, *count*)

Insert a record retrieved by the MySQL query into the SQLite database $APP_counts table.

Overrides: iprstats.core.Cache.__insert_count_record__

__create_match_group__(*self*, *app*)

Create a group (table) in the open SQLite database for storing match query results.

Overrides: iprstats.core.Cache.__create_match_group__

__insert_match_record__(*self*, *app*, *dbid*, *name*, *count*, *goid*, *goname*)

Insert a match record into the SQLite database.

Overrides: iprstats.core.Cache.__insert_match_record__

__commit_records__(*self*)

Commit the statements that have been executed so far.

Overrides: iprstats.core.Cache.__commit_records__

__match_length__(*self*, *app*)

Overrides: iprstats.core.Cache.__match_length__

__count_length__(*self*, *app*)

Overrides: iprstats.core.Cache.__count_length__

get_one_row(*self*, *app*, *rownum*)

Retrieve a row from the match SQLite table for application app. Format: (dbname, count, goid, dbid, goname)

Overrides: iprstats.core.Cache.get_one_row

get_counts(*self*, *app*)

Query the SQLite database for the counts data and return it in the format [(count1, count2, ...), (label1, label2, ...)]

Overrides: iprstats.core.Cache.get_counts

Inherited from iprstats.core.Cache(Section 2.2)

__close_writing__(), __count_query__(), __get_db_conn__(), __get_go_db_conn__(), __get_mysql_conn__(), __go_name__(), __match_query__(), __populate_cache__(), get_count_length(), get_match_length(), get_one_cell(), get_url()

2.4 Class IPRStatsData

This class is the original object used to retrieve aggregate results from the database. It is now fairly useless except to choose between different cache objects.

2.4.1 Methods

__init__(*self*, *settings*)

2.5 Class DBSettings

This class provides database connection details and type checking.

2.5.1 Methods

__init__(*self*, *user*, *passwd*, *db*, *host*=`None`, *port*=`None`)

Initialize with the essential details, including host, user, passwd and db. Make sure they are of the correct type.

gethost(*self*)

Returns: connection host (str) Default: 'localhost'

getuser(*self*)

Returns: connecting user (str)

getpasswd(*self*)

Returns: user's password (str)

getport(*self*)

Returns: connection port (int) Default: 3306

getdb(*self*)

Returns: connection db (str)

2.6 Class ChartSettings

This class provides chart settings, such as the maximum results, chart type, and chart generator.

2.6.1 Methods

__init__(*self*, *max_results*, *chart_type*, *generator*)

Initialize the chart object with max results (int), type (pie or bar), and generator (google or pylab)

getmaxresults(*self*)

Returns: maximum results to appear in the chart (int)

gettype(*self*)

Returns: current chart type (str)

getgenerator(*self*)

Returns: current chart generator (str)

getscale(*self*)

Returns: the chart scale; currently unchangable at 200 (int)

2.7 Class Settings

Class to centralize all the general IPRStats settings

2.7.1 Methods

__init__(*self*, *configpath*=`None`, *installed*=`False`)

Open the specified configuration path and initialize all the settings variables.

Default: it opens '.iprstats/iprstats.cfg' or '$HOME/.iprstats/iprstats.cfg'

__load_file_paths__(*self*, *installed*=`False`, *export_dir*=`None`)

Loads the default paths that IPRStats will used based on whether it's installed (True) or running locally (False)

__get_real_folder__(*self*, *path*)

Tries to create the given path if it doesn't exist and defaults to the temp directory otherwise. Returns: folder path (str)

__open_config_file__(*self*, *config_path*)

Attempts to find the given config file. If it can't, it attempts to copy it to the given location from a standard location. If that fails, it raises an IOError. Returns: config file (ConfigParser)

__load_settings__(*self*)

Read the opened config file (self.config) into settings variables.

__load_db_settings__(*self*, *section*)

Load connections from a given section in the configuration. Returns: dbsettings (DBSettings)

newsession(*self*, *session_id*=`None`)

Create a new session with the provided session_id; creates a random session id if session_id is not provided and deletes the old session files if possible. Returns: the new session id (str)

gethomedir(*self*)

Returns: home directory (str) Default: current working directory or $HOME

getdatadir(*self*)

Returns: data directory (str) Default: '.iprstats' or '$HOME/.iprstats'

getsessiondir(*self*)

```
Returns: current session directory (str)
Default: '.iprstats/sesssion/$SESSIONID' or
         '$HOME/.iprstats/session/$SESSIONID'
```

getsessionsdir(*self*)

Returns: sessions directory (str) Default: '.iprstats/session' or '$HOME/.iprstats/session'

getexportdir(*self*)

Returns: current export directory (str) Default: current working directory or $HOME

getinstalldir(*self*)

```
Returns: module install directory (str)
Default: 'C:\PythonX.X\Lib\site-packages\iprstats' or
         '/usr/share/pyshared/iprstats'
```

setexportdir(*self*, *new_export_dir*)

Sets the new export directory from opening a dialog

setsessiondir(*self*, *session_id*)

Sets the session directory variable to point to the correct session directory and creates the folder if necessary

getconfigparser(*self*)

Get the ConfigParser object. Returns: config (ConfigParser)

getsession(*self*)

Get the current session id. Returns: session id (str)

getmaxtableresults(*self*)

Get the user-specified maximum number of table results. Returns: max table results (int)

getchartsettings(*self*)

Returns: current chart settings (ChartSettings)

getapps(*self*)

Returns: list of supported apps (list of str)

usesqlite(*self*)

Returns: whether to use SQLite (bool)

usegolookup(*self*)

Returns: whether to use GO lookup (bool)

getlocaldb(*self*)

Returns: local MySQL db connection details (DBSettings)

getgodb(*self*)

Returns: Gene Ontology MySQL db connection details (DBSettings)

setgolookup(*self*, *value*)

Sets the GO lookup variable to the specified value

setconfigparser(*self*, *NewConfigParser*=`None`)

Sets a new ConfigParser object and reloads the configuration data. Reloads the current ConfigParser if no arguments

2.8 Class Chart

Object used to define the data, labels, title, filename, etc for generating, saving, and displaying charts.

2.8.1 Methods

__init__(*self*, *app*, *cache*, *chart_title*=`None`)

Initialize a chart with its associated app, the global cache, and a title you'd like to give it.

GetChartTitle(*self*)

Returns the chart title

GetFilename(*self*)

Returns the generated file path that the chart is downloaded to by default.

SetChartTitle(*self*, *chart_title*)

Set the chart title, and by doing so, change default filename for the chart.

CreateGooglePie(*self*, *counts*, *labels*, *settings*)

Create a pie chart using the Google Charting API.

CreateGoogleBar(*self*, *counts*, *labels*, *settings*)

Create a bar chart using the Google Charting API.

CreatePylabPie(*self*, *counts*, *labels*, *settings*)

Create a pie chart using PyLab/Matplotlib.

CreatePylabBar(*self*, *counts*, *labels*, *settings*)

Create a bar chart using PyLab/Matplotlib.

Save(*self*, *settings*=`None`, *filename*=`None`)

Generate the chart using the given ChartSettings object and download it to chart_filename.

GetChart(*self*, *settings*=`None`)

Return a wxPython object to display in the GUI. Right now it only saves the chart and returns a Bitmap, but eventually it could draw and return charts drawn in a wxCanvas object.

3 Module iprstats.exporters

3.1 Variables

Name	Description
__package__	**Value:** `'iprstats'`

3.2 Class html

Class for exporting all pages at one time

3.2.1 Methods

__init__(*self*, *iprstatsdata*)

Initialize the object with the required cache object

__get_link__(*self*, *link_name*, *link_url*, *target*=`None`, *link_title*=`None`)

Create a string containing a generic html link

link_name - the name of the link to be displayed to the user link_url - the website address of the link target - target window for the link to launch in; "_blank", "_parent", "_self" (default), or "_top" link_title - hover-over text of the link

returns a string containing the link HTML

__get_menu__(*self*, *menu_links*, *current_page*=`None`, *menu_id*=`None`)

Generate an HTML list of links to be styled with CSS into a menu

Requires a list of links of form [(title1, url1), (title2, url2), ...] current_page - if specified, will add class="selected" to that link menu_id - list id for identification and styling

__get_chart__(*self*, *app*, *directory*=`None`)

Saves the application chart to the given directory

chart_data of form [(label1, label2, ...), (value1, value2, ...)] chart_title - title to be displayed above the chart chart_filename - location to save the chart chart_type (optional) - 'google' or 'pylab'

Returns: an HTML img tag (str) or None

__get_table__(*self*, *app*, *file*)

Prints the generated link table to the supplied handle using the cache object for data.

export_page(*self*, *app*, *directory*=`None`)

Export an HTML page using the cache from initialization

app - specific app to export directory - output directory or sys.stdout if None

export(*self*, *directory*=`None`)

Export all pages as HTML

directory - HTML output directory (default=sys.stdout)

3.3 Class xls

Class for exporting IPRStatsData as a spreadsheet.

Each application will be generated in its own worksheet.

3.3.1 Methods

__init__(*self*, *cache*)

Initialize the exporter with the initialized IPRStatsData object

__generate_sheet__(*self*, *app*, *xls_doc*)

Generate a single worksheet with a single app

app - app to create a worksheet for xls_doc - xlwt.Workbook object to append to

export(*self*, *app*=`None`, *filename*=`None`)

Exports IPRStatsData as a spreadsheet

app - export only a single app (default: all) filename - save as file (default: iprstats.xls in current directory)

3.4 Class ips

Class used to save a session

This class zips the session directory into a tar.bz2 so that it can be unzipped and opened later or on a different machine.

3.4.1 Methods

__init__(*self*, *sessiondir*)

Initialize with session directory to save

export(*self*, *outputdir*)

Save session directory to outputdir file

4 Module iprstats.importers

4.1 Variables

Name	Description
__package__	**Value:** `'iprstats'`

4.2 Class EBIXML

xml.sax.handler.ContentHandler
  iprstats.importers.EBIXML

4.2.1 Methods

__init__(*self*, *settings*)

Overrides: xml.sax.handler.ContentHandler.__init__

startElement(*self*, *name*, *attrs*)

Signals the start of an element in non-namespace mode.

The name parameter contains the raw XML 1.0 name of the element type as a string and the attrs parameter holds an instance of the Attributes class containing the attributes of the element.

Overrides: xml.sax.handler.ContentHandler.startElement extit(inherited documentation)

endElement(*self*, *name*)

Signals the end of an element in non-namespace mode.

The name parameter contains the name of the element type, just as with the startElement event.

Overrides: xml.sax.handler.ContentHandler.endElement extit(inherited documentation)

endDocument(*self*)

Receive notification of the end of a document.

The SAX parser will invoke this method only once, and it will be the last method invoked during the parse. The parser shall not invoke this method until it has either abandoned parsing (because of an unrecoverable error) or reached the end of input.

Overrides: xml.sax.handler.ContentHandler.endDocument extit(inherited documentation)

Inherited from xml.sax.handler.ContentHandler

characters(), endElementNS(), endPrefixMapping(), ignorableWhitespace(), processingInstruction(), setDocumentLocator(), skippedEntity(), startDocument(), startElementNS(), startPrefixMapping()

4.3 Class ParseXMLFile

```
object ─┐
        │
threading._Verbose ─┐
                    │
     threading.Thread ─┐
                       │
                  iprstats.importers.ParseXMLFile
```

4.3.1 Methods

__init__(*self*, *filename*, *settings*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

run(*self*)

Overrides: threading.Thread.run

Inherited from threading.Thread

__repr__(), getName(), isAlive(), isDaemon(), is_alive(), join(), setDaemon(), setName(), start()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

4.3.2 Properties

Name	Description
Inherited from threading.Thread daemon, ident, name	
Inherited from object __class__	

4.4 Class ips

Opens a session file (.ips) by extracting it as a tar.bz2 into the sessions folder and populating the GUI elements.

4.4.1 Methods

__init__(*self*, *sessionsdir*)

Initialize the object with the directory to extract to – Settings.getsessionsdir()

open(*self*, *filename*)

Extract the provided filename (typically ending in .ips) to the directory that this object was initialized with. Return the session id (first membername/folder in the archive) or None if invalid.

5 Module iprstats.iprstats

5.1 Class IPRStats

Top level application for calling GUI

Handles menu actions, linking, and close events

5.1.1 Methods

__init__(*self*, *installed*=`False`)

Initialize the top-level frame

Initializes top-level grid, establishes a home directory, and reads in the configuration file.

OnOpen(*self*, *event*)

Open an XML file

Launches a file-chooser dialog for the user to select the InterProScan output XML file or IPRStats file to be opened. It then calls the corresponding function to open it and refresh the GUI data.

OpenXMLFile(*self*, *filename*)

Create a new session, parse the XML file, and retrieve the results.

OpenSession(*self*, *filename*)

Create a session importer instance and refresh the data.

OnSaveAs(*self*, *event*)

Saves the current session.

Compresseses the current session folder into a .tar.bz2 so that the user doesn't have to deal with parsing and querying an XML file every time.

OnExportHTML(*self*, *event*)

Exports current session as HTML

Takes the parsed and queried data from an opened XML file and writes it as HTML. Each app from InterProScan gets its own static HTML page.

OnExportXLS(*self*, *event*)

Save current session as a spreadsheet

Uses python-xlwt to write the parsed XML file as a spreadsheet file compatible with OpenOffice.org and Microsoft Excel

OnExit(*self*, *event*)

Closes the frame, any open files, and database connections.

OnProperties(*self*, *event*)

Opens the properties dialog box so the user can change settings

When the user clicks ”OK” on the launched properties dialog, the settings are written to disk and the GUI table is refreshed.

OnAbout(*self*, *event*)

Displays an about box

EnableExportOptions(*self*)

Enables previously disabled export menu items.

OnCellLeftClick(*self*, *event*)

Launches web browser when user clicks on certain grid cells

Retrieves the correct URL from the IPRStatsData object and launches the user’s default web browser.

6 Module iprstats.standalone

6.1 Class MainFrame

wx.Frame ─┐
iprstats.standalone.MainFrame

Top level frame

Handles menu actions, linking, and close events

6.1.1 Methods

__init__(*self*, *apps*, **args*, ***kwds*)

Initialize the top-level frame

Initializes top-level grid, establishes a home directory, and reads in the configuration file.

6.2 Class TabPanel

wx.Panel ─┐
iprstats.standalone.TabPanel

Custom wx.Panel containing a chart and table.

6.2.1 Methods

__init__(*self*, *parent*, *app*)

Initialize with a parent (Listbook) object and its specified app.

6.3 Class TabBook

wx.Listbook ─┐
iprstats.standalone.TabBook

Object for containing all the tabs and displaying them.

6.3.1 Methods

__init__(*self*, *parent*, *apps*)

Parent is the main frame (wx.Frame) and apps is a list of the available apps/tabs.

UpdateTabs(*self*, *iprstat*=`None`)

Assuming the values have changed, update each tab to display the new data. Return if no new data.

6.4 Class Menu

wx.MenuBar ─┐
iprstats.standalone.Menu

Main menu bar object used by iprstats.py

6.4.1 Methods

__init__(*self*, **args*, ***kwds*)

6.5 Class LinkTable

object ─┐
wx._core.Object ─┐
wx.grid.GridTableBase ─┐
wx.grid.PyGridTableBase ─┐
iprstats.standalone.LinkTable

Class underlying wx.grid.Grid that stores only the data from visible cells in memory at a given time

6.5.1 Methods

__init__(*self*, *app*, *data*=`None`)

Initialize the table with a specified app and an initialized cache object

Return Value

PyGridTableBase

Overrides: object.__init__

GetAttr(*self*, *row*, *col*, *kind*)

Return styling attributes for a cell

Makes columns 0 and 2 look like links if their value isn't None; make everything readonly

Overrides: wx.grid.GridTableBase.GetAttr

GetColLabelValue(*self*, *col*)

Define column labels

Overrides: wx.grid.GridTableBase.GetColLabelValue

GetNumberRows(*self*)

Return the current length of the table

Return Value

int

Overrides: wx.grid.GridTableBase.GetNumberRows

GetNumberCols(*self*)

Return the width of the table

Return Value

int

Overrides: wx.grid.GridTableBase.GetNumberCols

IsEmptyCell(*self*, *row*, *col*)

No visible cells should be empty

Overrides: wx.grid.GridTableBase.IsEmptyCell

GetValue(*self*, *row*, *col*)

Retrieve cell value from the cache object

Overrides: wx.grid.GridTableBase.GetValue

SetValue(*self*, *row*, *col*, *value*)

Cell editing is disabled

Overrides: wx.grid.GridTableBase.SetValue

Update(*self*, *data*=`None`)

Update the table length based on the underlying cache object (for if the max_tables_result value is changed).

Inherited from wx.grid.PyGridTableBase

Destroy(), __repr__(), base_AppendCols(), base_AppendRows(), base_CanGetValueAs(), base_CanHaveAttributes(), base_CanSetValueAs(), base_Clear(), base_DeleteCols(), base_DeleteRows(), base_GetAttr(), base_GetColLabelValue(), base_GetRowLabelValue(), base_GetTypeName(), base_InsertCols(), base_InsertRows(), base_SetAttr(), base_SetColAttr(), base_SetColLabelValue(), base_SetRowAttr(), base_SetRowLabelValue()

Inherited from wx.grid.GridTableBase

AppendCols(), AppendRows(), CanGetValueAs(), CanHaveAttributes(), CanSetValueAs(), Clear(), DeleteCols(), DeleteRows(), GetAttrProvider(), GetRowLabelValue(), GetTypeName(), GetValueAsBool(), GetValueAsDouble(), GetValueAsLong(), GetView(), InsertCols(), InsertRows(), SetAttr(), SetAttrProvider(), SetColAttr(), SetColLabelValue(), SetRowAttr(), SetRowLabelValue(), SetValueAsBool(), SetValueAsDouble(), SetValueAsLong(), SetView()

Inherited from wx._core.Object

GetClassName()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

6.5.2 Properties

Name	Description
Inherited from object	
__class__	

6.6 Class PropertiesDlg

wx.Dialog ─┐
iprstats.standalone.PropertiesDlg

Defines the properties dialog for changing settings

6.6.1 Methods

__init__(*self*, *parent*, *id*, *title*, *settings*)

Requires a parent wx object, an id, a title and a ConfigParser object for reading and writing settings

PopulateDialog(*self*)

Create control elements and initialize their values with the settings from self.config

OnUseSqlite(*self*, *event*)

Prevents user from disabling SQLite if MySQLdb is not installed

OnGoLookup(*self*, *event*)

Disables gene ontology lookup if MySQLdb is not installed

SaveProperties(*self*)

Writes all the dialog settings back to self.config

6.7 Class About

wx.AboutDialogInfo ─┐
iprstats.standalone.About

Defines information to display in the ”about” dialog

6.7.1 Methods

__init__(*self*, **args*, ***kwds*)

Index