PyStratum SQL Server Documentation

P.R. Water

Oct 26, 2020

Contents:

A stored procedure and function loader and wrapper generator for SQL Server Python.

CHAPTER 1

Licence

This project is licensed under the terms of the MIT-licentie.

CHAPTER 2

API

2.1 pystratum_mssql package

2.1.1 Subpackages

pystratum_mssql.backend package

Submodules

pystratum_mssql.backend.MsSqlBackend module

class pystratum_mssql.backend.MsSqlBackend.**MsSqlBackend**
Bases: pystratum_backend.Backend.Backend

PyStratum Backend for MS SQL Server.

create_constant_worker(*config: configparser.ConfigParser*, *io: pystratum_backend.StratumStyle.StratumStyle*) → Optional[pystratum_backend.ConstantWorker.ConstantWorker]
Creates the object that does the actual execution of the constant command for the backend.

Parameters

- **config** (*ConfigParser*) – The settings from the PyStratum configuration file.
- **io** (*StratumStyle*) – The output object.

Return type ConstantWorker|None

create_routine_loader_worker(*config: configparser.ConfigParser*, *io: pystratum_backend.StratumStyle.StratumStyle*) → Optional[pystratum_backend.RoutineLoaderWorker.RoutineLoaderWorker]
Creates the object that does the actual execution of the routine loader command for the backend.

Parameters

- **config** (*ConfigParser*) – The settings from the PyStratum configuration file.
- **io** (*StratumStyle*) – The output object.

Return type RoutineLoaderWorker|None

create_routine_wrapper_generator_worker(*config: configparser.ConfigParser*, *io: pystratum_backend.StratumStyle.StratumStyle*) → Optional[pystratum_backend.RoutineWrapperGeneratorWorker.Rout

Creates the object that does the actual execution of the routine wrapper generator command for the backend.

Parameters

- **config** (*ConfigParser*) – The settings from the PyStratum configuration file.
- **io** (*StratumStyle*) – The output object.

Return type RoutineWrapperGeneratorWorker|None

pystratum_mssql.backend.MsSqlConstantWorker module

PyStratum

class pystratum_mssql.backend.MsSqlConstantWorker.**MsSqlConstantWorker**(*io: pystratum_backend.StratumStyle.Str* *config: configparser.ConfigParser*)

Bases: *pystratum_mssql.backend.MsSqlWorker.MsSqlWorker*, pystratum_common.backend.CommonConstantWorker.CommonConstantWorker

Class for creating constants based on column widths, and auto increment columns and labels for SQL Server databases.

static derive_field_length(*column: Dict[str, Any]*) → int

Returns the width of a field based based on the data type of column.

Parameters **column** (*dict*) – Info about the column.

Return type int

pystratum_mssql.backend.MsSqlRoutineLoaderWorker module

class pystratum_mssql.backend.MsSqlRoutineLoaderWorker.**MsSqlRoutineLoaderWorker**(*io: pystratum_backend.St* *config: configparser.ConfigPa*

Bases: *pystratum_mssql.backend.MsSqlWorker.MsSqlWorker*, pystratum_common.

backend.CommonRoutineLoaderWorker.CommonRoutineLoaderWorker

Class for loading stored routines into a SQL Server instance from pseudo SQL files.

create_routine_loader_helper(*routine_name: str, pystratum_old_metadata: Optional[Dict[KT, VT]], rdbms_old_metadata: Optional[Dict[KT, VT]]*) → pystratum_mssql.helper.MsSqlRoutineLoaderHelper.MsSqlRoutineLoaderHelper

Creates a Routine Loader Helper object.

Parameters

- **routine_name** (*str*) – The name of the routine.
- **pystratum_old_metadata** (*dict*) – The old metadata of the stored routine from PyStratum.
- **rdbms_old_metadata** (*dict*) – The old metadata of the stored routine from MS SQL Server.

Return type *MsSqlRoutineLoaderHelper*

pystratum_mssql.backend.MsSqlRoutineWrapperGeneratorWorker module

class pystratum_mssql.backend.MsSqlRoutineWrapperGeneratorWorker.**MsSqlRoutineWrapperGenerat**

Bases: *pystratum_mssql.backend.MsSqlWorker.MsSqlWorker*, pystratum_common.backend.CommonRoutineWrapperGeneratorWorker.CommonRoutineWrapperGeneratorWorker

Class for generating a class with wrapper methods for calling stored routines in a SQL Server database.

pystratum_mssql.backend.MsSqlWorker module

class pystratum_mssql.backend.MsSqlWorker.**MsSqlWorker**(*io: pystratum_backend.StratumStyle.StratumStyle, config: configparser.ConfigParser*)

Bases: object

connect() → None
Connects to the database.

disconnect() → None
Disconnects from the database.

Module contents

pystratum_mssql.helper package

Submodules

pystratum_mssql.helper.MsSqlDataTypeHelper module

class pystratum_mssql.helper.MsSqlDataTypeHelper.**MsSqlDataTypeHelper**
Bases: pystratum_common.helper.DataTypeHelper.DataTypeHelper

Utility class for deriving information based on a SQL Server data type.

column_type_to_python_type(*data_type_info: Dict[str, Any]*) → str
Returns the corresponding Python data type of a SQL Server data type.

Parameters **data_type_info** (*dict*) – The SQL Server data type metadata.

Return type str

column_type_to_python_type_hint(*data_type_info: Dict[str, Any]*) → str
Returns the corresponding Python data type hinting of a SQL Server data type.

Parameters **data_type_info** (*dict*) – The PostgreSQL data type metadata.

Return type str

pystratum_mssql.helper.MsSqlRoutineLoaderHelper module

class pystratum_mssql.helper.MsSqlRoutineLoaderHelper.**MsSqlRoutineLoaderHelper**(*io: pystratum_backend.StratumStyle.StratumStyle*, *dl: pystratum_mssql.MsSqlDataLayer.MsSqlDataLayer*, *routine_filename*, *routine_file_encoding*, *pystratum_old_metadata*, *replace_pairs*, *rdbms_old_metadata*)
Bases: pystratum_common.helper.RoutineLoaderHelper.RoutineLoaderHelper

Class for loading a single stored routine into a SQL Server instance from a (pseudo) SQL file.

Module contents

pystratum_mssql.wrapper package

Submodules

pystratum_mssql.wrapper.MsSqlFunctionsWrapper module

class pystratum_mssql.wrapper.MsSqlFunctionsWrapper.**MsSqlFunctionsWrapper**(*routine: Dict[str, Any], lob_as_string_flag: bool*)

Bases: *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*, pystratum_common.wrapper.FunctionsWrapper.FunctionsWrapper

Wrapper method generator for stored functions.

pystratum_mssql.wrapper.MsSqlLogWrapper module

class pystratum_mssql.wrapper.MsSqlLogWrapper.**MsSqlLogWrapper**(*routine: Dict[str, Any], lob_as_string_flag: bool*)

Bases: *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*, pystratum_common.wrapper.LogWrapper.LogWrapper

Wrapper method generator for stored procedures with designation type log.

pystratum_mssql.wrapper.MsSqlNoneWrapper module

class pystratum_mssql.wrapper.MsSqlNoneWrapper.**MsSqlNoneWrapper**(*routine: Dict[str, Any], lob_as_string_flag: bool*)

Bases: *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*, pystratum_common.wrapper.NoneWrapper.NoneWrapper

Wrapper method generator for stored procedures without any result set.

pystratum_mssql.wrapper.MsSqlRow0Wrapper module

class pystratum_mssql.wrapper.MsSqlRow0Wrapper.**MsSqlRow0Wrapper**(*routine: Dict[str, Any], lob_as_string_flag: bool*)

Bases: *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*, pystratum_common.wrapper.Row0Wrapper.Row0Wrapper

Wrapper method generator for stored procedures that are selecting 0 or 1 row.

pystratum_mssql.wrapper.MsSqlRow1Wrapper module

class pystratum_mssql.wrapper.MsSqlRow1Wrapper.**MsSqlRow1Wrapper**(*routine: Dict[str, Any], lob_as_string_flag: bool*)

Bases: *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*, pystratum_common.wrapper.Row1Wrapper.Row1Wrapper

Wrapper method generator for stored procedures that are selecting 1 row.

pystratum_mssql.wrapper.MsSqlRowsWithIndexWrapper module

class pystratum_mssql.wrapper.MsSqlRowsWithIndexWrapper.**MsSqlRowsWithIndexWrapper**(*routine: Dict[str, Any], lob_as_strin bool*)

Bases: pystratum_common.wrapper.RowsWithIndexWrapper.RowsWithIndexWrapper, *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*

Wrapper method generator for stored procedures whose result set must be returned using tree structure using a combination of non-unique columns.

pystratum_mssql.wrapper.MsSqlRowsWithKeyWrapper module

class pystratum_mssql.wrapper.MsSqlRowsWithKeyWrapper.**MsSqlRowsWithKeyWrapper**(*routine: Dict[str, Any], lob_as_string_flag bool*)

Bases: pystratum_common.wrapper.RowsWithKeyWrapper.RowsWithKeyWrapper, *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*

Wrapper method generator for stored procedures whose result set must be returned using tree structure using a combination of unique columns.

pystratum_mssql.wrapper.MsSqlRowsWrapper module

class pystratum_mssql.wrapper.MsSqlRowsWrapper.**MsSqlRowsWrapper**(*routine: Dict[str, Any], lob_as_string_flag: bool*)

Bases: *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*, pystratum_common.wrapper.RowsWrapper.RowsWrapper

Wrapper method generator for stored procedures that are selecting 0, 1, or more rows.

pystratum_mssql.wrapper.MsSqlSingleton0Wrapper module

class pystratum_mssql.wrapper.MsSqlSingleton0Wrapper.**MsSqlSingleton0Wrapper**(*routine: Dict[str, Any], lob_as_string_flag: bool*)

Bases: *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*, pystratum_common.wrapper.Singleton0Wrapper.Singleton0Wrapper

Wrapper method generator for stored procedures that are selecting 0 or 1 row with one column only.

pystratum_mssql.wrapper.MsSqlSingleton1Wrapper module

class pystratum_mssql.wrapper.MsSqlSingleton1Wrapper.**MsSqlSingleton1Wrapper**(*routine: Dict[str, Any], lob_as_string_flag: bool*)

Bases: *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*, pystratum_common.wrapper.Singleton1Wrapper.Singleton1Wrapper

Wrapper method generator for stored procedures that are selecting 1 row with one column only.

pystratum_mssql.wrapper.MsSqlTableWrapper module

class pystratum_mssql.wrapper.MsSqlTableWrapper.**MsSqlTableWrapper**(*routine: Dict[str, Any], lob_as_string_flag: bool*)

Bases: *pystratum_mssql.wrapper.MsSqlWrapper.MsSqlWrapper*, pystratum_common.wrapper.TableWrapper.TableWrapper

Wrapper method generator for printing the result set of stored procedures in a table format.

pystratum_mssql.wrapper.MsSqlWrapper module

class pystratum_mssql.wrapper.MsSqlWrapper.**MsSqlWrapper**(*routine: Dict[str, Any], lob_as_string_flag: bool*)

Bases: pystratum_common.wrapper.Wrapper.Wrapper, abc.ABC

Parent class for wrapper method generators for stored procedures and functions.

is_lob_parameter(*parameters: List[Dict[str, Any]]*) → bool

Returns True of one of the parameters is a BLOB or CLOB. Otherwise, returns False.

Parameters **parameters** – The parameters of a stored routine.

Return type bool:

Module contents

pystratum_mssql.wrapper.**create_routine_wrapper**(*routine*, *lob_as_string_flag*)
A factory for creating the appropriate object for generating a wrapper method for a stored routine.

Parameters

- **routine** (*dict[str,str]*) – The metadata of the sored routine.
- **lob_as_string_flag** (*bool*) – If True BLOBs and CLOBs must be treated as strings.

Return type pystratum.mssql.wrapper.MsSqlWrapper.MsSqlWrapper

2.1.2 Submodules

2.1.3 pystratum_mssql.MsSqlConnector module

class pystratum_mssql.MsSqlConnector.**MsSqlConnector**
Bases: object

Interface for classes for connecting to a MS SQL Server instances.

connect() → Any
Connects to a MS SQL Server instance.

disconnect() → None
Disconnects from a MS SQL Server instance.

2.1.4 pystratum_mssql.MsSqlDataLayer module

class pystratum_mssql.MsSqlDataLayer.**MsSqlDataLayer**(*connector: pystratum_mssql.MsSqlConnector.MsSqlConnector*)
Bases: object

Class for connecting to a SQL Server instance and executing SQL statements. Also, a parent class for classes with static wrapper methods for executing stored procedures and functions.

autocommit(*status: bool*) → None
Sets auto commit mode. See http://pymssql.org/en/stable/ref/pymssql.html#pymssql.Connection.autocommit.

Parameters **status** (*bool*) – True: Auto commit on. False: Auto commit off.

commit() → None
Commits the current transaction. See http://pymssql.org/en/stable/ref/pymssql.html#pymssql.Connection.commit.

connect() → None
Connects to a MS SQL Server instance.

disconnect() → None
Disconnects from the MS SQL Server instance. See http://pymssql.org/en/stable/ref/pymssql.html#pymssql.Connection.close.

execute_csv(*sql: str*, *filename: str*, *dialect: str = 'unix'*, *encoding: str = 'utf-8'*) → int

execute_log(*sql: str*, **params*) → int
Executes a query with log statements. Returns the number of lines in the log.

Parameters

- **sql** (*str*) – The SQL statement.
- **params** (*iterable*) – The parameters.

Return type int

execute_none (*sql: str*, **params*) → None
Executes a query that does not select any rows.

Parameters

- **sql** (*str*) – The SQL statement.
- **params** (*iterable*) – The parameters.

Return type None

execute_row0 (*sql*, **params*) → Optional[Dict[str, Any]]
Executes a query that selects 0 or 1 row. Returns the selected row or None.

Parameters

- **sql** (*str*) – The SQL statement.
- **params** (*iterable*) – The parameters.

Return type Noneldict[str,*]

execute_row1 (*sql: str*, **params*) → Dict[str, Any]
Executes a query that selects 1 row. Returns the selected row.

Parameters

- **sql** (*str*) – The SQL statement.
- **params** (*iterable*) – The parameters.

Return type dict[str,*]

execute_rows (*sql: str*, **params*) → List[Dict[str, Any]]
Executes a query that selects 0 or more rows. Returns the selected rows (an empty list if no rows are selected).

Parameters

- **sql** (*str*) – The SQL statement.
- **params** (*iterable*) – The parameters.

Return type list[dict[str,*]]

execute_singleton0 (*sql: str*, **params*) → Any
Executes a query that selects 0 or 1 row with 1 column. Returns the value of selected column or None.

Parameters

- **sql** (*str*) – The SQL statement.
- **params** (*iterable*) – The parameters.

Return type

-

execute_singleton1 (*sql: str*, **params*) → Any
Executes a query that selects 1 row with 1 column. Returns the value of the selected column.

:param str sql:The SQL statement. :param iterable params: The parameters.

Return type

-

execute_sp_none (*sql: str*, **params*) → None
Executes a stored routine that does not select any rows.

Parameters

- **sql** (*str*) – The SQL calling the stored procedure.
- **params** (*iterable*) – The parameters for the stored procedure.

Return type None

execute_sp_row0 (*sql: str*, **params*) → Optional[Dict[str, Any]]
Executes a stored procedure that selects 0 or 1 row. Returns the selected row or None.

Parameters

- **sql** (*str*) – The SQL call the the stored procedure.
- **params** (*iterable*) – The parameters for the stored procedure.

Return type None|dict[str,*]

execute_sp_row1 (*sql: str*, **params*) → Dict[str, Any]
Executes a stored procedure that selects 1 row. Returns the selected row.

Parameters

- **sql** (*str*) – The SQL calling the the stored procedure.
- **params** (*iterable*) – The parameters for the stored procedure.

Return type dict[str,*]

execute_sp_rows (*sql: str*, **params*) → List[Dict[str, Any]]
Executes a stored procedure that selects 0 or more rows. Returns the selected rows (an empty list if no rows are selected).

Parameters

- **sql** (*str*) – The SQL calling the the stored procedure.
- **params** (*iterable*) – The parameters for the stored procedure.

Return type list[dict[str,*]]

execute_sp_singleton0 (*sql: str*, **params*) → Any
Executes a stored procedure that selects 0 or 1 row with 1 column. Returns the value of selected column or None.

Parameters

- **sql** (*str*) – The SQL calling the stored procedure.
- **params** (*iterable*) – The parameters for the stored procedure.

Return type

-

execute_sp_singleton1 (*sql: str*, **params*) → Any
Executes a stored routine with designation type "table", i.e a stored routine that is expected to select 1 row with 1 column.

Parameters

- **sql** (*str*) – The SQL calling the the stored procedure.
- **params** (*iterable*) – The parameters for the stored procedure.

Return type

- The value of the selected column.

line_buffered = True
If True log messages from stored procedures with designation type 'log' are line buffered (Note: In python sys.stdout is buffered by default).

Type bool

rollback() → None
Rolls back the current transaction. See http://pymssql.org/en/stable/ref/pymssql.html#pymssql.Connection.rollback.

static stratum_msg_handler(*msgstate: str*, *severity: int*, *srvname: str*, *procname: str*, *line: int*, *msgtext: bin*) → None
Custom message handler suppressing some superfluous messages.

2.1.5 pystratum_mssql.MsSqlDefaultConnector module

class pystratum_mssql.MsSqlDefaultConnector.**MsSqlDefaultConnector**(*params: Dict[str, Union[str, int]]*)

Bases: *pystratum_mssql.MsSqlConnector.MsSqlConnector*

Connects to a MySQL instance using user name and password.

connect() → Any
Connects to the MySQL instance.

disconnect() → None
Disconnects from the MySQL instance.

2.1.6 pystratum_mssql.MsSqlMetadataDataLayer module

class pystratum_mssql.MsSqlMetadataDataLayer.**MsSqlMetadataDataLayer**(*io: pystratum_backend.StratumStyle.Stratu connector: pystratum_mssql.MsSqlConnector.MsSq*

Bases: pystratum_common.MetadataDataLayer.MetadataDataLayer

Data layer for retrieving metadata and loading stored routines.

commit() → None
Connects to a SQL Server instance.

connect() → None
Connects to a SQL Server instance.

disconnect() → None
Disconnects from the SQL Server instance.

drop_stored_routine(*routine_type: str*, *schema_name: str*, *routine_name: str*) → None
Drops a stored routine if it exists.

Parameters

- **routine_type** (*str*) – The type of the routine (i.e. procedure or function).
- **schema_name** (*str*) – The name of the schema.
- **routine_name** (*str*) – The name of the routine.

drop_temporary_table(*table_name: str*) → None
Drops a temporary table.

Parameters **table_name** (*str*) – The name of the table.

execute_none(*query: str*) → None
Executes a query that does not select any rows.

Parameters **query** (*str*) – The query.

Return type int

execute_rows(*query: str*) → List[Dict[str, Any]]
Executes a query that selects 0 or more rows. Returns the selected rows (an empty list if no rows are selected).

Parameters **query** (*str*) – The query.

Return type list[dict[str,*]]

get_all_table_columns() → List[Dict[str, Any]]
Selects metadata of all columns of all tables.

Return type list[dict[str,*]]

get_label_tables(*regex: str*) → List[Dict[str, Any]]
Selects metadata of tables with a label column.

Parameters **regex** (*str*) – The regular expression for columns which we want to use.

Return type list[dict[str,*]]

get_labels_from_table(*database_name: str*, *schema_name: str*, *table_name: str*, *id_column_name: str*, *label_column_name: str*) → List[Dict[str, Any]]
Selects all labels from a table with labels.

Parameters

- **database_name** (*str*) – The name of the database.
- **schema_name** (*str*) – The name of the schema.
- **table_name** (*str*) – The name of the table.
- **id_column_name** (*str*) – The name of the auto increment column.
- **label_column_name** (*str*) – The name of the column with labels.

Return type list[dict[str,*]]

get_routine_parameters(*schema_name: str*, *routine_name: str*) → List[Dict[str, Any]]
Selects metadata of the parameters of a stored routine.

Parameters

- **schema_name** (*str*) – The name of the schema.

- **routine_name** (*str*) – The name of the routine.

Return type list[dict[str,*]]

get_routines() → List[Dict[str, Any]]
Selects metadata of all routines.

Return type list[dict[str,*]]

2.1.7 Module contents

Python Module Index

p

Index

G

I

L

M

P

R

S