

API Documentation

API Documentation

April 21, 2007

Contents

Contents	1
1 Package z3c.sqlalchemy	3
1.1 Modules	3
2 Module z3c.sqlalchemy.base	4
2.1 Class BaseWrapper	4
2.1.1 Methods	4
2.1.2 Properties	5
2.1.3 Class Variables	5
2.2 Class DataManager	5
2.2.1 Methods	5
2.2.2 Properties	7
2.2.3 Class Variables	7
2.3 Class ZopeBaseWrapper	7
2.3.1 Methods	7
2.3.2 Properties	8
2.3.3 Class Variables	9
3 Module z3c.sqlalchemy.interfaces	10
3.1 Class ISQLAlchemyWrapper	10
3.1.1 Methods	10
3.1.2 Class Variables	10
3.2 Class IModelProvider	11
3.2.1 Methods	11
3.2.2 Class Variables	11
4 Module z3c.sqlalchemy.mapper	12
4.1 Class MappedClassBase	12
4.1.1 Methods	12
4.1.2 Properties	13
4.2 Class MapperFactory	13
4.2.1 Methods	13
4.2.2 Properties	14
4.3 Class LazyMapperCollection	14
4.3.1 Methods	14
4.3.2 Properties	18

5	Module <code>z3c.sqlalchemy.model</code>	19
5.1	Class <code>Model</code>	19
5.1.1	Methods	19
5.1.2	Properties	23
6	Module <code>z3c.sqlalchemy.postgres</code>	24
6.1	Class <code>PostgresMixin</code>	24
6.1.1	Methods	24
6.1.2	Properties	25
6.1.3	Class Variables	25
6.2	Class <code>PythonPostgresWrapper</code>	25
6.2.1	Methods	25
6.2.2	Properties	27
6.2.3	Class Variables	27
6.3	Class <code>ZopePostgresWrapper</code>	27
6.3.1	Methods	27
6.3.2	Properties	29
6.3.3	Class Variables	29
7	Package <code>z3c.sqlalchemy.tests</code>	30
7.1	Modules	30
8	Module <code>z3c.sqlalchemy.tests.testSQLAlchemy</code>	31
8.1	Functions	31
8.2	Class <code>WrapperTests</code>	31
8.2.1	Methods	31
8.2.2	Properties	35
9	Module <code>z3c.sqlalchemy.util</code>	36
9.1	Functions	36
	Index	37

1 Package z3c.sqlalchemy

1.1 Modules

- **base** (*Section 2, p. 4*)
- **interfaces** (*Section 3, p. 10*)
- **mapper**: Utility methods for SQLAlchemy
(*Section 4, p. 12*)
- **model**: Optional Model support
(*Section 5, p. 19*)
- **postgres** (*Section 6, p. 24*)
- **tests** (*Section 7, p. 30*)
 - **testSQLAlchemy**: Tests, tests, tests.....
(*Section 8, p. 31*)
- **util**: Some helper methods
(*Section 9, p. 36*)

2 Module `z3c.sqlalchemy.base`

2.1 Class `BaseWrapper`

object └─
 `z3c.sqlalchemy.base.BaseWrapper`

Known Subclasses: `z3c.sqlalchemy.base.ZopeBaseWrapper`, `z3c.sqlalchemy.postgres.PythonPostgresWrapper`

2.1.1 Methods

`__init__(self, dsn, model=None, **kw)`
 'dsn' - a RFC-1738-style connection string
 'model' - optional instance of `model.Model`
 'kw' - optional keyword arguments passed to `create_engine()`
 Overrides: `object.__init__`

`registerMapper(self, mapper, name)`

`getMapper(self, tablename, schema='public')`

`getMappers(self, *names)`

`__delattr__(...)`
`x.__delattr__('name') <==> del x.name`

`__getattr__(...)`
`x.__getattr__('name') <==> x.name`

`__hash__(x)`
`hash(x)`

`__new__(T, S, ...)`
Return Value
 a new object with type `S`, a subtype of `T`

`__providedBy__(...)`
 Object Specification Descriptor

`__reduce__(...)`
 helper for pickle

__reduce_ex__(...)

 helper for pickle

__repr__(*x*)

 repr(*x*)

__setattr__(...)

x.__setattr__('name', value) <==> *x*.name = value

__str__(*x*)

 str(*x*)

2.1.2 Properties

Name	Description
metadata	Value: <property object at 0x2aea05336d20>
session	Value: <property object at 0x2aea05336d70>
engine	Value: <property object at 0x2aea05336dc0>
model	Value: <property object at 0x2aea05336e10>
__class__	Value: <attribute '__class__' of 'object' objects>

2.1.3 Class Variables

Name	Description
__implemented__	Value: <implementedBy z3c.sqlalchemy.base.BaseWrapper>
__provides__	Value: <zope.interface.declarations.ClassProvides object at 0x2a...

2.2 Class DataManager

object └─
 z3c.sqlalchemy.base.DataManager

Wraps session into transaction context of Zope

2.2.1 Methods

__init__(*self*, *session*)

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

 Overrides: object.__init__ extit(inherited documentation)

tpc_begin(*self*, *trans*)

abort(*self*, *trans*)

commit(*self*, *trans*)

tpc_vote(*self*, *trans*)

tpc_finish(*self*, *trans*)

tpc_abort(*self*, *trans*)

sortKey(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattr__(...)

x.__getattr__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

a new object with type *S*, a subtype of *T*

__providedBy__(...)

Object Specification Descriptor

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

<code>__str__(x)</code>
<code>str(x)</code>

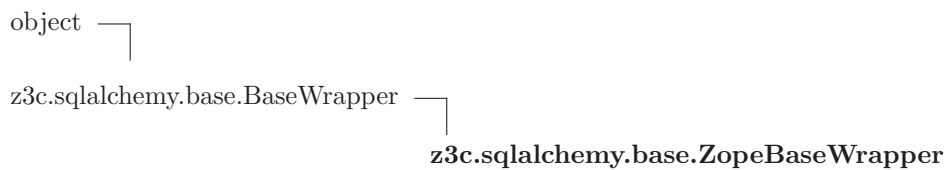
2.2.2 Properties

Name	Description
<code>__class__</code>	Value: <attribute <code>'__class__'</code> of <code>'object'</code> objects>

2.2.3 Class Variables

Name	Description
<code>__implemented__</code>	Value: <implementedBy z3c.sqlalchemy.base.DataManager>
<code>__provides__</code>	Value: <zope.interface.declarations.ClassProvides object at 0x2a...

2.3 Class ZopeBaseWrapper



Known Subclasses: z3c.sqlalchemy.postgres.ZopePostgresWrapper

A wrapper to be used from within Zope. It connects the session with the transaction management of Zope.

2.3.1 Methods

<code>__delattr__(...)</code>
<code>x.__delattr__('name') <==> del x.name</code>

<code>__getattr__(...)</code>
<code>x.__getattr__('name') <==> x.name</code>

<code>__hash__(x)</code>
<code>hash(x)</code>

__init__(*self*, *dsn*, *model*=None, ***kw*)

'dsn' - a RFC-1738-style connection string
 'model' - optional instance of model.Model
 'kw' - optional keyword arguments passed to create_engine()
 Overrides: object.__init__

__new__(*T*, *S*, ...)

Return Value
 a new object with type *S*, a subtype of *T*

__providedBy__(...)

Object Specification Descriptor

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(*x*)

__setattr__(...)

x.__setattr__('name', value) <==> *x*.name = value

__str__(*x*)

str(*x*)

getMapper(*self*, *tablename*, *schema*='public')

getMappers(*self*, **names*)

registerMapper(*self*, *mapper*, *name*)

2.3.2 Properties

Name	Description
session	Value: <property object at 0x2aea0544a190>
__class__	Value: <attribute '__class__' of 'object' objects>
engine	Value: <property object at 0x2aea05336dc0>
metadata	Value: <property object at 0x2aea05336d20>

continued on next page

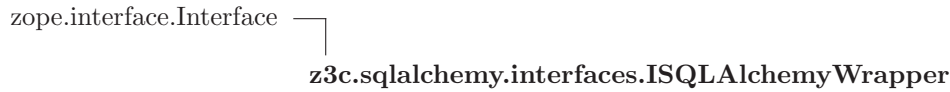
Name	Description
model	Value: <property object at 0x2aea05336e10>

2.3.3 Class Variables

Name	Description
__implemented__	Value: <implementedBy z3c.sqlalchemy.base.BaseWrapper>
__provides__	Value: <zope.interface.declarations.ClassProvides object at 0x2a...

3 Module `z3c.sqlalchemy.interfaces`

3.1 Class `ISQLAlchemyWrapper`



A `SQLAlchemyWrapper` wraps `sqlalchemy` and deals with connection and transaction handling.

3.1.1 Methods

<code>registerMapper</code> (<i>mapper</i> , <i>name</i>)
register your own mapper under a custom name
<code>getMapper</code> (<i>tablename</i> , <i>schema</i> ='public')
return a mapper class for a table given by its 'tablename' and an optional 'schema' name
<code>getMappers</code> (* <i>tablenames</i>)
return a sequence of mapper classes for a given list of table names. ATT: Schema support?

3.1.2 Class Variables

Name	Description
<code>dsn</code>	Value: <code>TextLine(title= u'A RFC-1738 style connection string', re...</code>
<code>dbname</code>	Value: <code>TextLine(title= u'Database name', required= True)</code>
<code>host</code>	Value: <code>TextLine(title= u'Hostname of database', required= True)</code>
<code>port</code>	Value: <code>Int(title= u'Port of database', required= True)</code>
<code>username</code>	Value: <code>TextLine(title= u'Database user', required= True)</code>
<code>password</code>	Value: <code>TextLine(title= u'Password of database user', required= T...</code>
<code>echo</code>	Value: <code>Bool(title= u'Echo all SQL statements to the console', re...</code>
<code>__bases__</code>	Value: (<code><InterfaceClass zope.interface.Interface></code>)
<code>__identifier__</code>	Value: <code>'z3c.sqlalchemy.interfaces.ISQLAlchemyWrapper'</code>
<code>__iro__</code>	Value: (<code><InterfaceClass z3c.sqlalchemy.interfaces.ISQLAlchemyWra...</code>
<code>__name__</code>	Value: <code>'ISQLAlchemyWrapper'</code>
<code>__sro__</code>	Value: (<code><InterfaceClass z3c.sqlalchemy.interfaces.ISQLAlchemyWra...</code>

continued on next page

Name	Description
dependents	Value: <WeakKeyDictionary at 47184597930496>

3.2 Class *IModelProvider*

zope.interface.Interface

```

└─ z3c.sqlalchemy.interfaces.IModelProvider

```

A model providers provides information about the tables to be used and the mapper classes.

3.2.1 Methods

getModel()
The model is described as an ordered dictionary. The entries are (tablename, some_dict) where 'some_dict' is a dictionary containing a key 'table' referencing a Table() instance and an optional key 'relationships' referencing a sequence of related table names. An optional mapper class can be specified through the 'class' key (otherwise a default mapper class will be autogenerated).


3.2.2 Class Variables

Name	Description
<code>__bases__</code>	Value: (<InterfaceClass zope.interface.Interface>)
<code>__identifier__</code>	Value: 'z3c.sqlalchemy.interfaces.IModelProvider'
<code>__iro__</code>	Value: (<InterfaceClass z3c.sqlalchemy.interfaces.IModelProvider...
<code>__name__</code>	Value: 'IModelProvider'
<code>__sro__</code>	Value: (<InterfaceClass z3c.sqlalchemy.interfaces.IModelProvider...
dependents	Value: <WeakKeyDictionary at 47184597930280>

4 Module `z3c.sqlalchemy.mapper`

Utility methods for SQLAlchemy

4.1 Class `MappedClassBase`

object 
`z3c.sqlalchemy.mapper.MappedClassBase`

base class for all mapped classes

4.1.1 Methods

`__init__(self, **kw)`

accepts keywords arguments used for initialization of mapped attributes/columns.

Overrides: `object.__init__`

`__delattr__(...)`

`x.__delattr__('name') <==> del x.name`

`__getattr__(...)`

`x.__getattr__('name') <==> x.name`

`__hash__(x)`

`hash(x)`

`__new__(T, S, ...)`

Return Value

a new object with type `S`, a subtype of `T`

`__reduce__(...)`

helper for pickle

`__reduce_ex__(...)`

helper for pickle

`__repr__(x)`

`repr(x)`

`__setattr__(...)`

`x.__setattr__('name', value) <==> x.name = value`

<code>__str__(x)</code>
<code>str(x)</code>

4.1.2 Properties

Name	Description
<code>__class__</code>	Value: <attribute <code>'__class__'</code> of <code>'object'</code> objects>

4.2 Class MapperFactory

object  **z3c.sqlalchemy.mapper.MapperFactory**

a factory for table and mapper objects

4.2.1 Methods

<code>__init__(self, metadata)</code>
<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>x.__class__.__doc__</code> for signature
Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)

<code>__call__(self, table, properties={}, cls=None)</code>
Returns a tuple (<code>mapped_class</code> , <code>table_class</code>). <code>'table'</code> - <code>sqlalchemy.Table</code> to be mapped <code>'properties'</code> - dict containing additional informations about <code>'cls'</code> - (optional) class used as base for creating the mapper class (will be autogenerated if not available).

<code>__delattr__(...)</code>
<code>x.__delattr__('name')</code> <==> <code>del x.name</code>

<code>__getattr__(...)</code>
<code>x.__getattr__('name')</code> <==> <code>x.name</code>

<code>__hash__(x)</code>
<code>hash(x)</code>

<code>__new__(T, S, ...)</code>
Return Value a new object with type <code>S</code> , a subtype of <code>T</code>

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)repr(*x*)**__setattr__**(...)*x*.__setattr__('name', value) <==> *x*.name = value**__str__**(*x*)str(*x*)

4.2.2 Properties

Name	Description
<code>__class__</code>	Value: <attribute <code>'__class__'</code> of <code>'object'</code> objects>

4.3 Class *LazyMapperCollection*



Implements a cache for table mappers

4.3.1 Methods

__init__(*self*, *wrapper*)*x*.__init__(...) initializes *x*; see *x*.__class__.__doc__ for signature**Return Value**

new empty dictionary

Overrides: dict.__init__ extit(inherited documentation)

getMapper(*self*, *name*, *schema*='public')return a (cached) mapper class for a given table *'name'*

__cmp__(*x*, *y*)

cmp(*x*,*y*)

__contains__(*D*, *k*)**Return Value**True if *D* has a key *k*, else False

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*x*, *y*)

del x[*y*]

__eq__(*x*, *y*)

x==y

__ge__(*x*, *y*)

x>=y

__getattr__(...)

x.__getattr__('name') <==> x.name

Overrides: object.__getattr__

__getitem__(*x*, *y*)

x[*y*]

__gt__(*x*, *y*)

x>y

__hash__(*x*)

hash(*x*)

Overrides: object.__hash__

__iter__(*x*)

iter(*x*)

__le__(*x*, *y*)

x<=y

__len__(*x*)len(*x*)**__lt__**(*x*, *y*)*x* < *y***__ne__**(*x*, *y*)*x* != *y***__new__**(*T*, *S*, ...)**Return Value**a new object with type *S*, a subtype of *T*

Overrides: object.__new__

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)repr(*x*)

Overrides: object.__repr__

__setattr__(...)*x*.__setattr__('name', value) <==> *x*.name = value**__setitem__**(*x*, *i*, *y*)*x*[*i*] = *y***__str__**(*x*)str(*x*)**clear**(*D*)Remove all items from *D*.**Return Value**

None

copy(*D*)**Return Value**a shallow copy of *D*

fromkeys(*dict*, *S*, *v=...*)

v defaults to None.**Return Value**New dict with keys from *S* and values equal to *v*

get(*D*, *k*, *d=...*)

d defaults to None.**Return Value***D*[*k*] if *k* in *D*, else *d*

has_key(*D*, *k*)**Return Value**True if *D* has a key *k*, else False

items(*D*)**Return Value**list of *D*'s (key, value) pairs, as 2-tuples

iteritems(*D*)**Return Value**an iterator over the (key, value) items of *D*

iterkeys(*D*)**Return Value**an iterator over the keys of *D*

itervalues(*D*)**Return Value**an iterator over the values of *D*

keys(*D*)**Return Value**list of *D*'s keys

pop(*D*, *k*, *d=...*)

If key is not found, *d* is returned if given, otherwise *KeyError* is raised**Return Value***v*, remove specified key and return the corresponding value

popitem(*D*)

2-tuple; but raise *KeyError* if *D* is empty**Return Value**

(k, v), remove and return some (key, value) pair as a

setdefault(*D*, *k*, *d*=...)

Return Value

D.get(*k*,*d*), also set *D*[*k*]=*d* if *k* not in *D*

update(*D*, *E*, ***F*)

Update *D* from *E* and *F*: for *k* in *E*: *D*[*k*] = *E*[*k*] (if *E* has keys else: for (*k*, *v*) in *E*: *D*[*k*] = *v*) then: for *k* in *F*: *D*[*k*] = *F*[*k*]

Return Value

None

values(*D*)

Return Value

list of *D*'s values

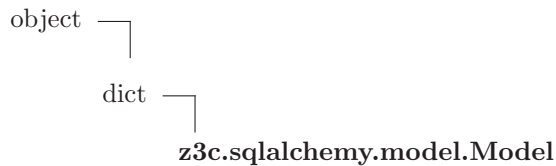
4.3.2 Properties

Name	Description
<code>__class__</code>	Value: <attribute ' <code>__class__</code> ' of 'object' objects>

5 Module *z3c.sqlalchemy.model*

Optional Model support

5.1 Class Model



The Model is an optional helper class that can be passed to the constructor of a SQLAlchemy wrapper in order to provide hints for the mapper generation.

5.1.1 Methods

`__init__(self, *args)`

The constructor can be called with a series of dict. Each dict represents a single table and its data (see `add()` method).

Return Value

new empty dictionary

Overrides: `dict.__init__`

`add(self, name, table=None, mapper_class=None, relations=None, autodetect_relations=False, table_name=None)`

'name' – name of table (no schema support so far!)

'table' – a sqlalchemy.Table instance (None, for autoloading)

'mapper_class' – an optional class to be used as mapper class for 'table'

'relations' – an optional list of table names referencing 'table'. This is used for auto-constructing the relation properties of the mapper class.

'autodetect_relations' – try to autodetect the relationships between tables and auto-construct the relation properties of the mapper if 'relations' is omitted (set to None)

'table_name' – optional full name of a table (e.g. 'someschema.sometable') if you want to use 'name' as alias for the table.

`items(self)`

return items in insertion order

Return Value

list of D's (key, value) pairs, as 2-tuples

Overrides: `dict.items`

`__cmp__(x, y)`

`cmp(x,y)`

`__contains__`(*D*, *k*)

Return Value

True if *D* has a key *k*, else False

`__delattr__`(...)

`x.__delattr__('name') <==> del x.name`

`__delitem__`(*x*, *y*)

`del x[y]`

`__eq__`(*x*, *y*)

`x==y`

`__ge__`(*x*, *y*)

`x>=y`

`__getattr__`(...)

`x.__getattr__('name') <==> x.name`

Overrides: `object.__getattr__`

`__getitem__`(*x*, *y*)

`x[y]`

`__gt__`(*x*, *y*)

`x>y`

`__hash__`(*x*)

`hash(x)`

Overrides: `object.__hash__`

`__iter__`(*x*)

`iter(x)`

`__le__`(*x*, *y*)

`x<=y`

`__len__`(*x*)

`len(x)`

__lt__(*x*, *y*)

x < *y*

__ne__(*x*, *y*)

x != *y*

__new__(*T*, *S*, ...)

Return Value

a new object with type *S*, a subtype of *T*

Overrides: `object.__new__`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

`repr(x)`

Overrides: `object.__repr__`

__setattr__(...)

x.`__setattr__('name', value)` <==> *x*.*name* = *value*

__setitem__(*x*, *i*, *y*)

x[*i*] = *y*

__str__(*x*)

`str(x)`

clear(*D*)

Remove all items from *D*.

Return Value

None

copy(*D*)

Return Value

a shallow copy of *D*

fromkeys(*dict*, *S*, *v*=...)

v defaults to None.**Return Value**New dict with keys from *S* and values equal to *v*

get(*D*, *k*, *d*=...)

d defaults to None.**Return Value***D*[*k*] if *k* in *D*, else *d*

has_key(*D*, *k*)**Return Value**True if *D* has a key *k*, else False

iteritems(*D*)**Return Value**an iterator over the (key, value) items of *D*

iterkeys(*D*)**Return Value**an iterator over the keys of *D*

itervalues(*D*)**Return Value**an iterator over the values of *D*

keys(*D*)**Return Value**list of *D*'s keys

pop(*D*, *k*, *d*=...)

If key is not found, *d* is returned if given, otherwise `KeyError` is raised**Return Value***v*, remove specified key and return the corresponding value

popitem(*D*)

2-tuple; but raise `KeyError` if *D* is empty**Return Value**

(k, v), remove and return some (key, value) pair as a

setdefault(*D*, *k*, *d*=...)**Return Value***D*.get(*k*,*d*), also set *D*[*k*]=*d* if *k* not in *D*

update(*D*, *E*, *******F*)

Update *D* from *E* and *F*: for *k* in *E*: *D*[*k*] = *E*[*k*] (if *E* has keys else: for (*k*, *v*) in *E*: *D*[*k*] = *v*) then: for *k* in *F*: *D*[*k*] = *F*[*k*]

Return Value

None

values(*D*)

Return Value

list of *D*'s values

5.1.2 Properties

Name	Description
<code>__class__</code>	Value: <attribute <code>'__class__'</code> of <code>'object'</code> objects>

6 Module `z3c.sqlalchemy.postgres`

6.1 Class `PostgresMixin`

object 
`z3c.sqlalchemy.postgres.PostgresMixin`

Known Subclasses: `z3c.sqlalchemy.postgres.PythonPostgresWrapper`, `z3c.sqlalchemy.postgres.ZopePostgresWrapper`
 Mixin class for Postgres aspects

6.1.1 Methods

`findDependentTables(self, schema='public', ignoreErrors=False)`

Returns a mapping `tablename -> [list of referencing table(names)]`. ATT: this method is specific to Postgres databases! ATT: This method is limited to a particular schema.

`__delattr__(...)`

`x.__delattr__('name') <==> del x.name`

`__getattr__(...)`

`x.__getattr__('name') <==> x.name`

`__hash__(x)`

`hash(x)`

`__init__(...)`

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

`__new__(T, S, ...)`

Return Value

a new object with type `S`, a subtype of `T`

`__providedBy__(...)`

Object Specification Descriptor

`__reduce__(...)`

helper for pickle

`__reduce_ex__(...)`

helper for pickle

`__repr__(x)``repr(x)``__setattr__(...)``x.__setattr__('name', value) <==> x.name = value``__str__(x)``str(x)`

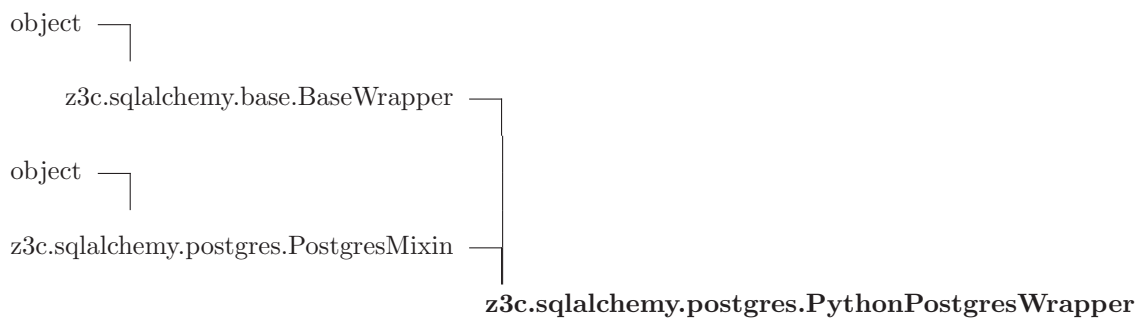
6.1.2 Properties

Name	Description
<code>__class__</code>	Value: <attribute <code>'__class__'</code> of <code>'object'</code> objects>

6.1.3 Class Variables

Name	Description
<code>__implemented__</code>	Value: <implementedBy z3c.sqlalchemy.postgres.PostgresMixin>
<code>__provides__</code>	Value: <zope.interface.declarations.ClassProvides object at 0x2a...>

6.2 Class PythonPostgresWrapper



Wrapper to be used with Python with extended Postgres functionality.

6.2.1 Methods

`__delattr__(...)``x.__delattr__('name') <==> del x.name`

__getattr__(...) $x._\text{getattr__}(\text{'name'}) \iff x.\text{name}$ **__hash__**(*x*)hash(*x*)**__init__**(*self*, *dsn*, *model*=None, ****kw**)

'dsn' - a RFC-1738-style connection string

'model' - optional instance of model.Model

'kw' - optional keyword arguments passed to create_engine()

Overrides: object.__init__

__new__(*T*, *S*, ...)**Return Value**a new object with type *S*, a subtype of *T***__providedBy__**(...)

Object Specification Descriptor

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)repr(*x*)**__setattr__**(...) $x._\text{setattr__}(\text{'name'}, \text{value}) \iff x.\text{name} = \text{value}$ **__str__**(*x*)str(*x*)**findDependentTables**(*self*, *schema*='public', *ignoreErrors*=False)

Returns a mapping tablename -> [list of referencing table(names)]. ATT: this method is specific to Postgres databases! ATT: This method is limited to a particular schema.

getMapper(*self*, *tablename*, *schema*='public')**getMappers**(*self*, ***names**)

<code>registerMapper(<i>self</i>, <i>mapper</i>, <i>name</i>)</code>
--

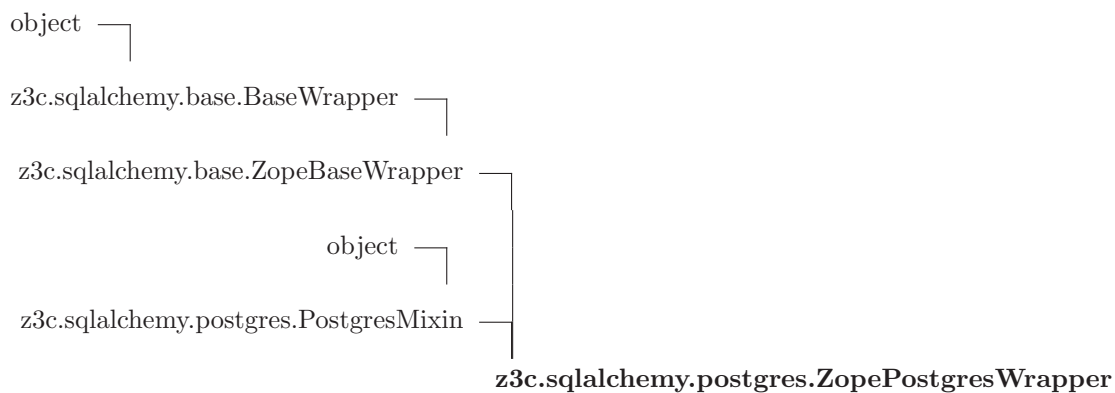
6.2.2 Properties

Name	Description
<code>__class__</code>	Value: <attribute ' <code>__class__</code> ' of 'object' objects>
<code>engine</code>	Value: <property object at 0x2aea05336dc0>
<code>metadata</code>	Value: <property object at 0x2aea05336d20>
<code>model</code>	Value: <property object at 0x2aea05336e10>
<code>session</code>	Value: <property object at 0x2aea05336d70>

6.2.3 Class Variables

Name	Description
<code>__implemented__</code>	Value: <implementedBy z3c.sqlalchemy.base.BaseWrapper>
<code>__provides__</code>	Value: <zope.interface.declarations.ClassProvides object at 0x2a...>

6.3 Class ZopePostgresWrapper



A wrapper to be used from within Zope. It connects the session with the transaction management of Zope.

6.3.1 Methods

<code>__delattr__(...)</code>
<code>x.__delattr__('name') <==> del x.name</code>

<code>__getattr__(...)</code>
<code>x.__getattr__('name') <==> x.name</code>

__hash__(*x*)

hash(x)

__init__(*self*, *dsn*, *model*=None, ***kw*)

'dsn' - a RFC-1738-style connection string

'model' - optional instance of model.Model

'kw' - optional keyword arguments passed to create_engine()

Overrides: object.__init__

__new__(*T*, *S*, ...)**Return Value**a new object with type *S*, a subtype of *T***__providedBy__**(...)

Object Specification Descriptor

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

findDependentTables(*self*, *schema*='public', *ignoreErrors*=False)

Returns a mapping tablename -> [list of referencing table(names)]. ATT: this method is specific to Postgres databases! ATT: This method is limited to a particular schema.

getMapper(*self*, *tablename*, *schema*='public')**getMappers**(*self*, **names*)**registerMapper**(*self*, *mapper*, *name*)

6.3.2 Properties

Name	Description
<code>__class__</code>	Value: <attribute ' <code>__class__</code> ' of 'object' objects>
<code>engine</code>	Value: <property object at 0x2aea05336dc0>
<code>metadata</code>	Value: <property object at 0x2aea05336d20>
<code>model</code>	Value: <property object at 0x2aea05336e10>
<code>session</code>	Value: <property object at 0x2aea0544a190>

6.3.3 Class Variables

Name	Description
<code>__implemented__</code>	Value: <implementedBy z3c.sqlalchemy.base.BaseWrapper>
<code>__provides__</code>	Value: <zope.interface.declarations.ClassProvides object at 0x2a...

7 Package `z3c.sqlalchemy.tests`

7.1 Modules

- **testSQLAlchemy**: Tests, tests, tests.....
(Section 8, p. 31)

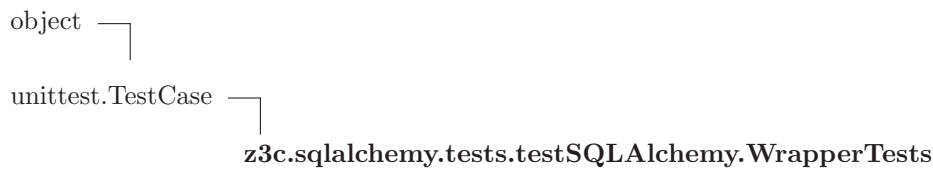
8 Module `z3c.sqlalchemy.tests.testSQLAlchemy`

Tests, tests, tests.....

8.1 Functions

<code>test_suite()</code>

8.2 Class `WrapperTests`



8.2.1 Methods

<code>setUp(self)</code> Hook method for setting up the test fixture before exercising it. Overrides: <code>unittest.TestCase.setUp</code> <code>exitit</code> (inherited documentation)

<code>testIFaceBaseWrapper(self)</code>
--

<code>testIFacePythonPostgres(self)</code>

<code>testIFaceZopePostgres(self)</code>

<code>testSimplePopulation(self)</code>
--

<code>testMapperWithCustomModel(self)</code>

<code>testGetMappers(self)</code>
--

<code>testModelWeirdParameters(self)</code>
--

<code>testModelNonExistingTables(self)</code>
--

<code>testWrapperRegistration(self)</code>

<code>testWrapperRegistrationFailing(self)</code>
--

<code>__call__(self, *args, **kws)</code>
--

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattr__(...)

x.__getattr__('name') <==> x.name

__hash__(x)

hash(x)

__init__(self, methodName='runTest')

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

Overrides: object.__init__

__new__(T, S, ...)**Return Value**

a new object with type S, a subtype of T

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(self)

repr(x)

Overrides: object.__repr__ extit(inherited documentation)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(self)

str(x)

Overrides: object.__str__ extit(inherited documentation)

assertAlmostEqual(self, first, second, places=7, msg=None)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

assertAlmostEquals(*self*, *first*, *second*, *places*=7, *msg*=None)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

assertEqual(*self*, *first*, *second*, *msg*=None)

Fail if the two objects are unequal as determined by the '==' operator.

assertEquals(*self*, *first*, *second*, *msg*=None)

Fail if the two objects are unequal as determined by the '==' operator.

assertFalse(*self*, *expr*, *msg*=None)

Fail the test if the expression is true.

assertNotAlmostEqual(*self*, *first*, *second*, *places*=7, *msg*=None)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

assertNotAlmostEquals(*self*, *first*, *second*, *places*=7, *msg*=None)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

assertNotEqual(*self*, *first*, *second*, *msg*=None)

Fail if the two objects are equal as determined by the '==' operator.

assertNotEquals(*self*, *first*, *second*, *msg*=None)

Fail if the two objects are equal as determined by the '==' operator.

assertRaises(*self*, *excClass*, *callableObj*, **args*, ***kwargs*)

Fail unless an exception of class *excClass* is thrown by *callableObj* when invoked with arguments *args* and keyword arguments *kwargs*. If a different type of exception is thrown, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

assertTrue(*self*, *expr*, *msg*=None)

Fail the test unless the expression is true.

assert_(self, expr, msg=None)

Fail the test unless the expression is true.

countTestCases(self)

debug(self)

Run the test without collecting errors in a TestResult

defaultTestResult(self)

fail(self, msg=None)

Fail immediately, with the given message.

failIf(self, expr, msg=None)

Fail the test if the expression is true.

failIfAlmostEqual(self, first, second, places=7, msg=None)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

failIfEqual(self, first, second, msg=None)

Fail if the two objects are equal as determined by the '==' operator.

failUnless(self, expr, msg=None)

Fail the test unless the expression is true.

failUnlessAlmostEqual(self, first, second, places=7, msg=None)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

failUnlessEqual(self, first, second, msg=None)

Fail if the two objects are unequal as determined by the '==' operator.

failUnlessRaises(self, excClass, callableObj, *args, **kwargs)

Fail unless an exception of class excClass is thrown by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is thrown, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

id (<i>self</i>)

run (<i>self</i> , <i>result</i> =None)

shortDescription (<i>self</i>)

<p>Returns a one-line description of the test, or None if no description has been provided. The default implementation of this method returns the first line of the specified test method's docstring.</p>

tearDown (<i>self</i>)

<p>Hook method for deconstructing the test fixture after testing it.</p>
--

8.2.2 Properties

Name	Description
<code>__class__</code>	Value: <attribute ' <code>__class__</code> ' of 'object' objects>

9 Module `z3c.sqlalchemy.util`

Some helper methods

9.1 Functions

<code>createSQLAlchemyWrapper</code> (<i>dsn</i> , <i>model</i> =None, <i>forZope</i> =False, <i>**kw</i>)
--

Convenience method to generate a wrapper for a DSN and a model. This method hides all database related magic from the user. Set 'forZope' to True for a Zope related wrapper.

<code>registerSQLAlchemyWrapper</code> (<i>wrapper</i> , <i>name</i>)
--

deferred registration of the wrapper as named utility

<code>getSQLAlchemyWrapper</code> (<i>name</i>)
--

<code>allRegisteredSQLAlchemyWrappers</code> ()
--

return a dict containing information for all registered wrappers.

Index

- dict.__cmp__ (function), 14, 19
- dict.__contains__ (function), 15, 19
- dict.__delitem__ (function), 15, 20
- dict.__eq__ (function), 15, 20
- dict.__ge__ (function), 15, 20
- dict.__getitem__ (function), 15, 20
- dict.__gt__ (function), 15, 20
- dict.__iter__ (function), 15, 20
- dict.__le__ (function), 15, 20
- dict.__len__ (function), 15, 20
- dict.__lt__ (function), 16, 20
- dict.__ne__ (function), 16, 21
- dict.__setitem__ (function), 16, 21
- dict.clear (function), 16, 21
- dict.copy (function), 16, 21
- dict.fromkeys (function), 16, 21
- dict.get (function), 17, 22
- dict.has_key (function), 17, 22
- dict.items (function), 17
- dict.iteritems (function), 17, 22
- dict.iterkeys (function), 17, 22
- dict.itervalues (function), 17, 22
- dict.keys (function), 17, 22
- dict.pop (function), 17, 22
- dict.popitem (function), 17, 22
- dict.setdefault (function), 17, 22
- dict.update (function), 18, 22
- dict.values (function), 18, 23

- object.__delattr__ (function), 4, 6, 7, 12, 13, 15, 20, 24, 25, 27, 31
- object.__getattr__ (function), 4, 6, 7, 12, 13, 24, 25, 27, 32
- object.__hash__ (function), 4, 6, 7, 12, 13, 24, 26, 27, 32
- object.__init__ (function), 24
- object.__new__ (function), 4, 6, 8, 12, 13, 24, 26, 28, 32
- object.__reduce__ (function), 4, 6, 8, 12, 13, 16, 21, 24, 26, 28, 32
- object.__reduce_ex__ (function), 4, 6, 8, 12, 14, 16, 21, 24, 26, 28, 32
- object.__repr__ (function), 5, 6, 8, 12, 14, 24, 26, 28
- object.__setattr__ (function), 5, 6, 8, 12, 14, 16, 21, 25, 26, 28, 32
- object.__str__ (function), 5, 6, 8, 12, 14, 16, 21, 25, 26, 28

- unittest.TestCase.__call__ (function), 31
- unittest.TestCase.countTestCases (function), 34
- unittest.TestCase.debug (function), 34
- unittest.TestCase.defaultTestResult (function), 34
- unittest.TestCase.fail (function), 34
- unittest.TestCase.failIf (function), 33, 34
- unittest.TestCase.failIfAlmostEqual (function), 33, 34
- unittest.TestCase.failIfEqual (function), 33, 34
- unittest.TestCase.failUnless (function), 33, 34
- unittest.TestCase.failUnlessAlmostEqual (function), 32, 34
- unittest.TestCase.failUnlessEqual (function), 33, 34
- unittest.TestCase.failUnlessRaises (function), 33, 34
- unittest.TestCase.id (function), 34
- unittest.TestCase.run (function), 35
- unittest.TestCase.shortDescription (function), 35
- unittest.TestCase.tearDown (function), 35

- z3c (package)
 - z3c.sqlalchemy (package), 3
 - z3c.sqlalchemy.base (module), 4–9
 - z3c.sqlalchemy.interfaces (module), 10–11
 - z3c.sqlalchemy.mapper (module), 12–18
 - z3c.sqlalchemy.model (module), 19–23
 - z3c.sqlalchemy.postgres (module), 24–29
 - z3c.sqlalchemy.tests (package), 30
 - z3c.sqlalchemy.util (module), 36