API Documentation

API Documentation

May 26, 2007

Contents

C	Contents		
1	Pac	kage z3c.sqlalchemy	3
	1.1	Modules	3
2	Mo	dule z3c.sqlalchemy.base	4
	2.1	Variables	4
	2.2	Class SynchronizedThreadCache	4
		2.2.1 Methods	4
		2.2.2 Properties	5
	2.3	Class BaseWrapper	5
		2.3.1 Methods	5
		2.3.2 Properties	6
		2.3.3 Class Variables	6
	2.4	Class SessionDataManager	7
		2.4.1 Methods	7
		2.4.2 Properties	8
		2.4.3 Class Variables	8
	2.5	Class ConnectionDataManager	8
		2.5.1 Methods	9
		2.5.2 Properties	10
		2.5.3 Class Variables	10
	2.6	Class ZopeBaseWrapper	10
		2.6.1 Methods	10
		2.6.2 Properties	11
		2.6.3 Class Variables	12
3	Mo	dule z3c.sqlalchemy.interfaces	13
	3.1	Class ISQLAlchemyWrapper	13
		3.1.1 Methods	13
		3.1.2 Class Variables	13
	3.2	Class IModelProvider	14
		3.2.1 Methods	14
		3.2.2 Class Variables	14
	3.3	Class IModel	14
		3.3.1 Methods	15
		3.3.2 Class Variables	15

CONTENTS

1	M	dule z3c.sqlalchemy.mapper
4		
	4.1	Class MappedClassBase
		4.1.1 Methods
		4.1.2 Properties
		4.1.3 Class Variables
	4.2	Class MapperFactory
		4.2.1 Methods
		4.2.2 Properties
	4.3	Class LazyMapperCollection
		4.3.1 Methods
		4.3.2 Properties
5	Mo	dule z3c.sqlalchemy.model 24
	5.1	Class Model
		5.1.1 Methods
		5.1.2 Properties
		5.1.3 Class Variables
6	Mo	dule z3c.sqlalchemy.postgres 29
	6.1	Class PostgresMixin
		6.1.1 Methods
		6.1.2 Properties
		6.1.3 Class Variables
	6.2	Class PythonPostgresWrapper
		6.2.1 Methods
		6.2.2 Properties
		6.2.3 Class Variables
	6.3	Class ZopePostgresWrapper
	0.0	6.3.1 Methods
		6.3.2 Properties
		6.3.3 Class Variables
		0.9.9 Class Variables
7	Mo	dule z3c.sqlalchemy.test 35
	7.1	Variables
	7.2	Class HierarchyNode
	•	7.2.1 Methods
		7.2.2 Properties
		7.2.3 Class Variables
	7.3	Class HierarchyNode
	1.5	7.3.1 Methods
		7.3.2 Properties
		1
		7.3.3 Class Variables
8	Dac	kage z3c.sqlalchemy.tests 41
O	8.1	Modules
	0.1	100 tules
9	Mo	dule z3c.sqlalchemy.tests.testSQLAlchemy 42
,	9.1	Functions
	9.2	Class WrapperTests 42
	0.4	9.2.1 Methods 42
		9.2.2 Properties

CONTENTS	CONTENTS

10 Module z3c.sqlalchemy.util	4	17
10.1 Functions	4	17
Index	4	19

1 Package z3c.sqlalchemy

1.1 Modules

- base (Section 2, p. 4)
- interfaces (Section 3, p. 13)
- mapper: Utility methods for SqlAlchemy (Section 4, p. 16)
- model: Optional Model support (Section 5, p. 24)
- postgres (Section 6, p. 29)
- test (Section 7, p. 35)
- tests (Section 8, p. 41)
 - testSQLAlchemy: Tests, tests, tests......... (Section 9, p. 42)
- util: Some helper methods (Section 10, p. 47)

2 Module z3c.sqlalchemy.base

2.1 Variables

Name	Description
session_cache	Value: <z3c.sqlalchemy.base.synchronizedthreadcache< th=""></z3c.sqlalchemy.base.synchronizedthreadcache<>
	object at 0x
connection_cache	Value: <z3c.sqlalchemy.base.synchronizedthreadcache< th=""></z3c.sqlalchemy.base.synchronizedthreadcache<>
	object at 0x

${\bf 2.2}\quad {\bf Class~SynchronizedThreadCache}$

```
\begin{array}{c} \text{object} & \\ \\ \textbf{z3c.sqlalchemy.base.SynchronizedThreadCache} \end{array}
```

2.2.1 Methods

```
__init__(self)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature
Overrides: object.__init__ extit(inherited documentation)
```

set(self, **kw)

get(self, *names)

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

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

 $\frac{-_{\mathbf{hash}}_{-}(x)}{\mathrm{hash}(\mathbf{x})}$

--reduce_-(...)
helper for pickle

reduce_ex()	
helper for pickle	

```
__repr__(x)
repr(x)
```

```
\frac{\text{_--setattr}_{--}(...)}{\text{x._--setattr}_{--}(\text{'name', value}) <==> \text{x.name} = \text{value}}
```

```
\frac{-\mathbf{str}_{--}(x)}{\mathbf{str}(\mathbf{x})}
```

2.2.2 Properties

Name	Description
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>

2.3 Class BaseWrapper

 $\textbf{Known Subclasses:} \ z3c.sqlalchemy.base.ZopeBaseWrapper, z3c.sqlalchemy.postgres.PythonPostgresWrapper (a) and a subclasses (b) and a subclasses (c) and a subclasses (c) and a subclasse (c) and a subclasses (c) and a subclasse (c) and a subc$

2.3.1 Methods

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

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

```
\frac{-_{\mathbf{hash}}_{--}(x)}{\mathrm{hash}(\mathbf{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

 $__\mathbf{repr}__(x)$

repr(x)

__setattr__(...)

 $x._setattr_('name', value) <==> x.name = value$

 $_{-}\mathbf{str}_{-}(x)$

str(x)

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

getMappers(self, *names)

registerMapper(self, mapper, name)

2.3.2 Properties

Name	Description
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>
engine	Value: <pre><pre><pre>Value: <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>
metadata	Value: <pre><pre><pre>Value: <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>
model	Value: <pre><pre><pre>Value: <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>
session	Value: <pre><pre><pre>Value: <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>

2.3.3 Class Variables

Name	Description
implemented	Value: <implementedby z3c.sqlalchemy.base.basewrapper=""></implementedby>

 $continued\ on\ next\ page$

Name	Description
provides	f Value: < zope.interface.declarations.ClassProvides
	object at 0x2b

2.4Class SessionDataManager

object z3c.sqlalchemy.base.SessionDataManager

Wraps session into transaction context of Zope

2.4.1 Methods

__init__(self, session) x.__init__(...) initializes x; see x.__class__.__doc__ for signature Overrides: object.__init__ extit(inherited documentation) abort(self, trans) commit(self, trans) $\mathbf{tpc_begin}(\mathit{self}, \mathit{trans})$ tpc_vote(self, trans)

tpc_finish(*self*, *trans*)

tpc_abort(self, trans)

sortKey(self)

__delattr__(...) $x._delattr_{-}('name') <==> del x.name$

 $_$ getattribute $_(...)$ $x._getattribute_('name') \le x.name$

 $_{-}\mathbf{hash}_{-}(x)$ 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

 $_{-}\mathbf{repr}_{-}(x)$

repr(x)

 $_$ setattr $_(...)$

 $x._setattr_{-}('name', value) <==> x.name = value$

 $_{-}$ **str** $_{--}(x)$

str(x)

2.4.2 Properties

Name	Description
_class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>

2.4.3 Class Variables

Name	Description
implemented	Value: <implementedby< th=""></implementedby<>
	z3c.sqlalchemy.base.SessionDataManager>
provides	Value: <zope.interface.declarations.classprovides< th=""></zope.interface.declarations.classprovides<>
	object at 0x2b

2.5 Class ConnectionDataManager

object —

 ${\bf z} \\ {\bf 3c. sqlalchemy. base. Connection Data Manager}$

Wraps connection into transaction context of Zope

2.5.1 Methods

__init__(self, connection) x.__init__(...) initializes x; see x.__class__.__doc__ for signature Overrides: object.__init__ extit(inherited documentation) abort(self, trans) commit(self, trans) **tpc_begin**(self, trans) tpc_vote(self, trans) **tpc_finish**(self, trans) tpc_abort(self, trans) $\mathbf{sortKey}(self)$ $_{-}$ delattr $_{-}$ (...) $x._delattr_('name') \le del x.name$ $_$ getattribute $_(...)$ $x._getattribute_('name') <==> x.name$ -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
```

```
\frac{-\mathbf{str}_{-}(x)}{\mathbf{str}(\mathbf{x})}
```

2.5.2 Properties

Name	Description
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>

2.5.3 Class Variables

Name	Description
implemented	Value: <implementedby< th=""></implementedby<>
	z3c.sqlalchemy.base.ConnectionDataManager>
provides	Value: <zope.interface.declarations.classprovides< th=""></zope.interface.declarations.classprovides<>
	object at 0x2b

2.6 Class ZopeBaseWrapper

 ${\bf Known~Subclasses:}~z3c.sqlalchemy.postgres.ZopePostgresWrapper$

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

2.6.1 Methods

```
\frac{\text{--delattr}_{-}(...)}{\text{x.--delattr}_{-}(\text{'name'}) <==> \text{del x.name}}
```

```
\frac{\text{--getattribute}_{-}(...)}{\text{x.-getattribute}_{-}('name') <==> \text{x.name}}
```

 $\frac{--\mathbf{hash}_{--}(x)}{\mathbf{hash}(\mathbf{x})}$

__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

 $_{-}\mathbf{repr}_{-}(x)$

repr(x)

__setattr__(...)

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

 $_{-}\mathbf{str}_{--}(x)$

str(x)

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

 $\mathbf{getMappers}(\mathit{self},\,*\mathit{names})$

registerMapper(self, mapper, name)

2.6.2 Properties

continued on next page

Name	Description
Name	Description
_class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>
connection	Value: <property 0x2b57c55bb730="" at="" object=""></property>
engine	Value: <property 0x2b57c55bb460="" at="" object=""></property>
metadata	Value: <property 0x2b57c55bb3c0="" at="" object=""></property>
model	Value: <property 0x2b57c55bb4b0="" at="" object=""></property>
session	Value: <property 0x2b57c55bb6e0="" at="" object=""></property>

2.6.3 Class Variables

Name	Description	
implemented	Value: <implementedby z3c.sqlalchemy.base.basewrapper=""></implementedby>	
provides	Value: <zope.interface.declarations.classprovides< th=""></zope.interface.declarations.classprovides<>	
	object at 0x2b	

3 Module z3c.sqlalchemy.interfaces

3.1 Class ISQLAlchemyWrapper

zope.interface.Interface	
	z3c.sqlalchemy.interfaces.ISQLAlchemyWrapper

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

3.1.1 Methods

registerMapper(mapper, name)
register your own mapper under a custom name

$\mathbf{getMappers}(*tablenames)$	
return a sequence of mapper classes for a given list of table names.	ATT: Schema support?

3.1.2 Class Variables

Name	Description
dsn	Value: TextLine(title= u'A RFC-1738 style connection
	string', re
dbname	Value: TextLine(title= u'Database name', required=
	True)
host	Value: TextLine(title= u'Hostname of database',
	required= True)
port	Value: Int(title= u'Port of database', required= True)
username	Value: TextLine(title= u'Database user', required=
	True)
password	Value: TextLine(title= u'Password of database user',
	required= T
echo	Value: Bool(title= u'Echo all SQL statements to the
	console', re
bases	Value: (<interfaceclass zope.interface.interface="">)</interfaceclass>
identifier	Value: 'z3c.sqlalchemy.interfaces.ISQLAlchemyWrapper'
iro	Value: (<interfaceclass< td=""></interfaceclass<>
	z3c.sqlalchemy.interfaces.ISQLAlchemyWra
_name	Value: 'ISQLAlchemyWrapper'
sro	Value: (<interfaceclass< th=""></interfaceclass<>
	z3c.sqlalchemy.interfaces.ISQLAlchemyWra
dependents	Value: <weakkeydictionary 47655970883560="" at=""></weakkeydictionary>

3.2 Class IModelProvider

 $\begin{tabular}{ll} zope.interface. Interface & \\ & z3c. sqlalchemy.interfaces. IModel Provider \\ \end{tabular}$

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

3.2.1 Methods

getModel(metadata=None)

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
_bases	Value: (<interfaceclass zope.interface.interface="">)</interfaceclass>
identifier	Value: 'z3c.sqlalchemy.interfaces.IModelProvider'
iro	Value: (<interfaceclass< th=""></interfaceclass<>
	z3c.sqlalchemy.interfaces.IModelProvider
_name	Value: 'IModelProvider'
sro	Value: (<interfaceclass< th=""></interfaceclass<>
	z3c.sqlalchemy.interfaces.IModelProvider
dependents	Value: <weakkeydictionary 47655970884208="" at=""></weakkeydictionary>

3.3 Class IModel

 $\begin{tabular}{ll} {\bf zope.interface.Interface} & & \\ & &$

A model represents a configuration hint for SQLAlchemy wrapper instances in order to deliver mappers for a given name.

3.3.1 Methods

 $\mathbf{add}(name,\ table = \mathtt{None},\ mapper_class = \mathtt{None},\ relations = \mathtt{None},\ autodetect_relations = \mathtt{False},\ table_name = \mathtt{None})$

'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()
return items in insertion order

3.3.2 Class Variables

Name	Description
_bases	Value: (<interfaceclass zope.interface.interface="">)</interfaceclass>
identifier	Value: 'z3c.sqlalchemy.interfaces.IModel'
iro	Value: (<interfaceclass< td=""></interfaceclass<>
	z3c.sqlalchemy.interfaces.IModel>, <inte< td=""></inte<>
_name	Value: 'IModel'
sro	Value: (<interfaceclass< td=""></interfaceclass<>
	z3c.sqlalchemy.interfaces.IModel>, <inte< td=""></inte<>
dependents	Value: <weakkeydictionary 47655973196576="" at=""></weakkeydictionary>

4 Module z3c.sqlalchemy.mapper

Utility methods for SqlAlchemy

4.1 Class MappedClassBase

object —

z3c.sqlalchemy.mapper.MappedClassBase

Known Subclasses: z3c.sqlalchemy.test.HierarchyNode

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__

clone(self)

Create a pristine copy. Use this method if you need to reinsert a copy of the current mapper instance back into the database.

getMapper(self, name)

Return a mapper associated with the current mapper. If this mapper represents a table A having a relationship to table B then the mapper for B can be obtained through self.getMapper('B'). This method is useful if you don't want to

pass the wrapper around this the wrapper is officially the only way to get hold of a mapper by name. See also http://groups.google.com/group/sqlalchemy/browse_thread/thread/18fb2e2818bdc032/5c2dfd71679925cb#5c2dfd71679925

 $__{\mathbf{delattr}_(...)}$

 $x._delattr_('name') \le = > del x.name$

 $_$ getattribute $_$ (...)

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

 $-\mathbf{hash}_{-}(x)$

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
```

```
\frac{-\mathbf{repr}_{--}(x)}{\mathrm{repr}(\mathbf{x})}
```

```
\frac{\text{--setattr}_{-}(...)}{\text{x.--setattr}_{-}(\text{'name', value}) <==> \text{x.name} = \text{value}}
```

```
\frac{-\mathbf{str}_{--}(x)}{\mathbf{str}(\mathbf{x})}
```

4.1.2 Properties

Name	Description
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>

4.1.3 Class Variables

Name	Description
_allow_access_to_unprotected_s-	Value: 1
ubobjects	

4.2 Class MapperFactory

 $\begin{array}{c} \text{object} \ \ \, \begin{array}{c} -\\ \\ \textbf{z3c.sqlalchemy.mapper.MapperFactory} \end{array}$

a factory for table and mapper objects

4.2.1 Methods

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

Overrides: object.__init__ extit(inherited documentation)
```

__call__(self, table, properties={}, cls=None)

Returns a tuple (mapped_class, table_class). 'table' - sqlalchemy. Table to be mapped 'properties' - dict containing additional informations about

'cls' - (optional) class used as base for creating the mapper class (will be autogenerated if not available).

 $_{-}$ delattr $_{-}$ (...)

 $x._delattr_('name') \le > del x.name$

 $_$ getattribute $_(...)$

 $x._getattribute_('name') \le x.name$

-hash-(x)

hash(x)

 $_{-}$ **new** $_{-}$ (T, S, ...)

Return Value

a new object with type S, a subtype of ${\tt T}$

 $_$ 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
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>

4.3 Class LazyMapperCollection

object	
	dict —
	z3c.sglalchemy.mapper.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'
```

```
\frac{-\text{cmp}_{--}(x, y)}{\text{cmp}(x,y)}
```

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

```
\begin{array}{c}
--\mathbf{eq}_{--}(x, y) \\
\mathbf{x} = = \mathbf{y}
\end{array}
```

```
\frac{--\mathbf{g}\mathbf{e}_{--}(x, y)}{x > = y}
```

```
\_getattribute\_(...)
x._getattribute_('name') \le x.name
Overrides: object.__getattribute__
_{-}getitem_{-}(x, y)
x[y]
_{-}\mathbf{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)
-1t_{-}(x, y)
x < y
-\mathbf{ne}_{-}(x, y)
x!=y
__new__( T, S, ...)
Return Value
      a new object with type {\tt S}, a subtype of {\tt 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

 $_{\mathbf{L}}\mathbf{str}_{\mathbf{L}}(x)$

str(x)

 $\mathbf{clear}(D)$

Remove all items from D.

Return Value

None

 $\mathbf{copy}(D)$

Return Value

a shallow copy of ${\tt D}$

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

v defaults to None.

Return Value

New dict with keys from S and values equal to \boldsymbol{v}

 $\mathbf{get}(D, k, d = \dots)$

d defaults to None.

Return Value

D[k] if k in D, else d

 $\mathbf{has}_{\mathbf{key}}(D, k)$

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 $\ensuremath{\mathsf{D}}$

itervalues(D)

Return Value

an iterator over the values of D

 $\mathbf{keys}(D)$

Return Value

list of D's keys

 $\mathbf{pop}(D, k, d = \dots)$

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

 $\mathbf{update}(D, E, **F)$

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
_class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>

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__

 ${\tt add}(self, name, table = {\tt None}, mapper_class = {\tt None}, relations = {\tt None}, autodetect_relations = {\tt False}, table_name = {\tt None}, cascade = {\tt None})$

'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.

'cascade' - optional cascade parameter directly passed to the relation() call

items(self)

return items in insertion order

Return Value

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

Overrides: dict.items

```
\frac{-\operatorname{cmp}_{-}(x, y)}{\operatorname{cmp}(x, y)}
```

 $_$ contains $_$ (D, k) Return Value

True if D has a key k, else False

__delattr__(...)

 $x._delattr_('name') \le del x.name$

 $_$ delitem $_(x, y)$

del x[y]

 $-\mathbf{eq}_{--}(x, y)$

x==y

 $_{--}$ **ge** $_{--}(x, y)$

x>=y

 $__\mathbf{getattribute}__(...)$

 $x._getattribute_('name') <==> x.name$

Overrides: object.__getattribute__

 $_$ getitem $_(x, y)$

x[y]

 $_{-}\mathbf{gt}_{-}(x, y)$

x>y

 $_$ hash $_$ (x)

hash(x)

Overrides: object._hash__

 $_{-}$ iter $_{-}(x)$

iter(x)

 $-\mathbf{le}_{-}(x, y)$

x < =y

 $_{-}$ len $_{-}(x)$

len(x)

 $\frac{-\mathbf{lt}_{--}(x, y)}{\mathbf{x} < \mathbf{y}}$

 $\underline{\mathbf{x}!=\mathbf{y}}$

__new__(T, S, ...)

Return Value

a new object with type S, a subtype of T

Overrides: object.__new__

 $__\mathbf{providedBy}__(...)$

Object Specification Descriptor

__reduce__(...)
helper for pickle

__reduce_ex__(...)
helper for pickle

 $\frac{__\mathbf{repr}__(x)}{\mathrm{repr}(\mathbf{x})}$ Overrides: object. $_$ repr $_$

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

 $\frac{\text{__setitem}_{\text{_}}(x, i, y)}{x[i] = y}$

 $\frac{-\mathbf{str}_{-}(x)}{\mathbf{str}(\mathbf{x})}$

 $\mathbf{clear}(D)$

Remove all items from D.

Return Value None $\mathbf{copy}(D)$

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 \boldsymbol{v}

 $\mathbf{get}(D, k, d = \dots)$

d defaults to None.

Return Value

D[k] if k in D, else d

 $\mathbf{has}_{\mathbf{key}}(D, k)$

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 ${\tt D}$

 $\mathbf{keys}(D)$

Return Value

list of D's keys

 $\mathbf{pop}(D, k, d = \dots)$

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

 $\mathbf{update}(D, E, **F)$

 $\label{eq:continuous} Update \ D \ from \ E \ and \ F: \ for \ k \ in \ E: \ D[k] = E[k] \ (if \ E \ has \ keys \ else: \ for \ (k, \ v) \ in \ E: \ D[k] = v) \ then: \ for \ k \ in \ F: \ D[k] = F[k]$

Return Value

None

values(D)

Return Value

list of D's values

5.1.2 Properties

Name	Description
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>

5.1.3 Class Variables

Name	Description
implemented	Value: <implementedby z3c.sqlalchemy.model.model=""></implementedby>
provides	Value: <zope.interface.declarations.classprovides< th=""></zope.interface.declarations.classprovides<>
	object at 0x2b

6 Module z3c.sqlalchemy.postgres

6.1 Class PostgresMixin

object	
	z3c.sqlalchemy.postgres.PostgresMixir

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

6.1.1 Methods

```
\mathbf{findDependentTables}(\mathit{self}, \mathit{schema} \texttt{='public'}, \mathit{ignoreErrors} \texttt{=} \texttt{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.

```
\frac{\text{--delattr}_{-}(...)}{\text{x.--delattr}_{-}(\text{'name'}) <==> \text{del x.name}}
```

```
__getattribute__(...)

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

```
\frac{-\mathbf{hash}_{-}(x)}{\mathbf{hash}(\mathbf{x})}
```

```
__init__(...)

x.__init__(...) initializes x; see x.__class____doc__ for signature
```

```
--providedBy__(...)
Object Specification Descriptor
```

```
__reduce__(...)
helper for pickle
```

```
__reduce_ex__(...)
helper for pickle
```

```
\frac{-\mathbf{repr}_{-}(x)}{\mathrm{repr}(\mathbf{x})}
```

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

```
\frac{-\mathbf{str}_{-}(x)}{\mathbf{str}(\mathbf{x})}
```

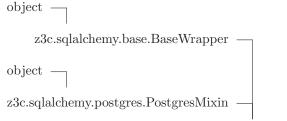
6.1.2 Properties

Name	Description
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>

6.1.3 Class Variables

Name	Description
implemented	Value: <implementedby< th=""></implementedby<>
	z3c.sqlalchemy.postgres.PostgresMixin>
provides	Value: <zope.interface.declarations.classprovides< th=""></zope.interface.declarations.classprovides<>
	object at 0x2b

${\bf 6.2 \quad Class\ Python Postgres Wrapper}$



z3c. sqlalchemy. postgres. Python Postgres Wrapper

Wrapper to be used with Python with extended Postgres functionality.

6.2.1 Methods

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

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

 $_{-}\mathbf{repr}_{-}(x)$

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)

 $\mathbf{registerMapper}(\mathit{self}, \mathit{mapper}, \mathit{name})$

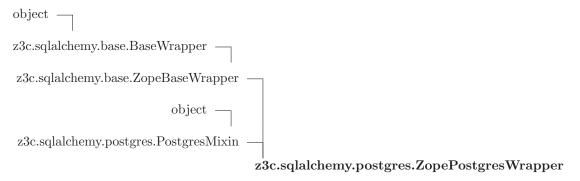
6.2.2 Properties

Name	Description
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>
engine	Value: <pre><pre><pre>value: <pre><pre>property object at 0x2b57c55bb460></pre></pre></pre></pre></pre>
metadata	Value: <pre><pre><pre>value: <pre><pre>property object at 0x2b57c55bb3c0></pre></pre></pre></pre></pre>
model	Value: <pre><pre><pre>value: <pre><pre>property object at 0x2b57c55bb4b0></pre></pre></pre></pre></pre>
session	Value: <pre><pre><pre>value: <pre><pre>property object at 0x2b57c55bb410></pre></pre></pre></pre></pre>

6.2.3 Class Variables

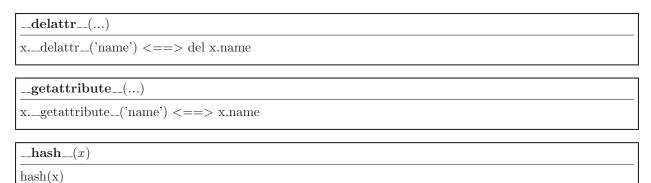
Name	Description	
implemented	Value: <implementedby z3c.sqlalchemy.base.basewrapper=""></implementedby>	
provides	Value: <zope.interface.declarations.classprovides< th=""></zope.interface.declarations.classprovides<>	
	object at 0x2b	

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



 $_$ 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

 $_{-}\mathbf{repr}_{-}(x)$

repr(x)

 $_{-}$ setattr $_{-}(...)$

 $x._setattr_('name', value) <==> x.name = value$

 $_{\mathbf{L}}\mathbf{str}_{\mathbf{L}}(x)$

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
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>
connection	Value: <pre><pre><pre><pre>Value: <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>
engine	Value: <pre><pre><pre>volue:</pre></pre></pre>
metadata	Value: <pre><pre><pre>volue:</pre></pre></pre>
model	Value: <pre><pre>cproperty object at 0x2b57c55bb4b0></pre></pre>
session	Value: <pre><pre>cproperty object at 0x2b57c55bb6e0></pre></pre>

6.3.3 Class Variables

Name	Description	
implemented	Value: <implementedby z3c.sqlalchemy.base.basewrapper=""></implementedby>	
provides	Value: <zope.interface.declarations.classprovides< th=""></zope.interface.declarations.classprovides<>	
	object at 0x2b	

7 Module z3c.sqlalchemy.test

7.1 Variables

Name	Description
dsn	Value:
	'postgres://postgres:postgres@cmsdb/Toolbox2Test'
е	Value: create_engine(dsn)
metadata	Value: BoundMetaData()
HierarchyTable	Value:
	Table('hierarchy',BoundMetaData(),Column(u'id',PGInteger(.
m	Value: {'hierarchy': {'name': 'hierarchy',
	'autodetect_relations
wrapper	Value: <z3c.sqlalchemy.postgres.pythonpostgreswrapper< th=""></z3c.sqlalchemy.postgres.pythonpostgreswrapper<>
	object at
session	Value: wrapper.session
rows	Value: [<z3c.sqlalchemy.test.hierarchynode at<="" object="" th=""></z3c.sqlalchemy.test.hierarchynode>
	0x2b57c5790
EXT_PASS	Value: <object 0x2b57c26c10a0="" at="" object=""></object>
NULLTYPE	Value: NullTypeEngine()
default_metadata	Value: DynamicMetaData()
func	Value: <sqlalchemy.sqlfunctiongateway at<="" object="" th=""></sqlalchemy.sqlfunctiongateway>
	0x2b57c4a80d10>

7.2 Class HierarchyNode

object — $z3c.sqlalchemy.mapper.MappedClassBase \quad — \\ z3c.sqlalchemy.test.HierarchyNode$

7.2.1 Methods

$_$ _delattr $_$ ()	
xdelattr('name') <==> del x.name	

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

-hash (x)	
hash(x)	

 $_$ init $_$ (self, *args, **kwargs)

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

Overrides: z3c.sqlalchemy.mapper.MappedClassBase.__init__

 $_{-}$ **new** $_{-}$ (T, S, ...)

Return Value

a new object with type ${\tt S}$, a subtype of ${\tt T}$

__reduce__(...)

helper for pickle

 $_$ reduce $_$ ex $_$ (...)

helper for pickle

 $_{-}\mathbf{repr}_{-}(x)$

repr(x)

__setattr__(...)

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

 $_{-}$ str $_{-}(x)$

str(x)

clone(self)

Create a pristine copy. Use this method if you need to reinsert a copy of the current mapper instance back into the database.

getMapper(self, name)

Return a mapper associated with the current mapper. If this mapper represents a table A having a relationship to table B then the mapper for B can be obtained through self.getMapper('B'). This method is useful if you don't want to

pass the wrapper around this the wrapper is officially the only way to get hold of a mapper by name. See also http://groups.google.com/group/sqlalchemy/browse_thread/thread/18fb2e2818bdc032/5c2dfd71679925cb#5c2dfd71679925

7.2.2 Properties

Name	Description		
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>		

7.2.3 Class Variables

Name	Description
_allow_access_to_unprotected_s-	Value: 1
ubobjects	
aedat	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
benutzer	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
bezeichnung	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
С	Value: <sqlalchemy.orm.mapper.lorderedprop 0x2b57c5784<="" at="" object="" td=""></sqlalchemy.orm.mapper.lorderedprop>
children	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
comment	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
deleted	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
id	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
idhierarchy_share	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
idprodukt	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
linkindex	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
neudat	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
parent	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
parentid	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
pos	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
produktkuerzel	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
show_gattung_in_bauplan	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
sortierung	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
sorting	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
visible	Value: <sqlalchemy.orm.unitofwork.uowproperty 0x2b57c5<="" at="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>

7.3 Class HierarchyNode

object — $z3c.sqlalchemy.mapper.MappedClassBase — \\ & z3c.sqlalchemy.test.HierarchyNode$

7.3.1 Methods

 $\frac{-\text{delattr}_{-}(...)}{\text{x._delattr}_{-}(\text{'name'}) <==> \text{del x.name}}$

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

 $\frac{--\mathbf{hash}_{--}(x)}{\mathbf{hash}(\mathbf{x})}$

__init__(self, *args, **kwargs)
accepts keywords arguments used for initialization of mapped attributes/columns.
Overrides: z3c.sqlalchemy.mapper.MappedClassBase.__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

 $\frac{_{-\mathbf{repr}_{--}}(x)}{\operatorname{repr}(\mathbf{x})}$

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

 $\frac{-_{\mathbf{str}_{--}}(x)}{\operatorname{str}(\mathbf{x})}$

clone(self)

Create a pristine copy. Use this method if you need to reinsert a copy of the current mapper instance back into the database.

getMapper(self, name)

Return a mapper associated with the current mapper. If this mapper represents a table A having a relationship to table B then the mapper for B can be obtained through self.getMapper('B'). This method is useful if you don't want to

pass the wrapper around this the wrapper is officially the only way to get hold of a mapper by name. See also http://groups.google.com/group/sqlalchemy/browse_thread/18fb2e2818bdc032/5c2dfd71679925cb#5c2dfd71679925

7.3.2 Properties

Name	Description	
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>	

7.3.3 Class Variables

Name	Description
_allow_access_to_unprotected_s-	Value: 1
$ubobjects_{}$	
aedat	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
benutzer	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
bezeichnung	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
С	Value: <sqlalchemy.orm.mapper.lorderedprop at<="" object="" td=""></sqlalchemy.orm.mapper.lorderedprop>
	0x2b57c5784
children	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
comment	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
deleted	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
id	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
idhierarchy_share	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
idprodukt	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
linkindex	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
neudat	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
parent	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
parentid	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
pos	Value: <sqlalchemy.orm.unitofwork.uowproperty at<="" object="" td=""></sqlalchemy.orm.unitofwork.uowproperty>
	0x2b57c5
	0x2b57c5

continued on next page

Name	Description		
produktkuerzel	Value: <sqlalchemy.orm.unitofwork.uowproperty< td=""><td>object</td><td>at</td></sqlalchemy.orm.unitofwork.uowproperty<>	object	at
	0x2b57c5		
show_gattung_in_bauplan	Value: <sqlalchemy.orm.unitofwork.uowproperty< td=""><td>object</td><td>at</td></sqlalchemy.orm.unitofwork.uowproperty<>	object	at
	0x2b57c5		
sortierung	Value: <sqlalchemy.orm.unitofwork.uowproperty< td=""><td>object</td><td>at</td></sqlalchemy.orm.unitofwork.uowproperty<>	object	at
	0x2b57c5		
sorting	Value: <sqlalchemy.orm.unitofwork.uowproperty< td=""><td>object</td><td>at</td></sqlalchemy.orm.unitofwork.uowproperty<>	object	at
	0x2b57c5		
visible	Value: <sqlalchemy.orm.unitofwork.uowproperty< td=""><td>object</td><td>at</td></sqlalchemy.orm.unitofwork.uowproperty<>	object	at
	0x2b57c5		

8 Package z3c.sqlalchemy.tests

8.1 Modules

• testSQLAlchemy: Tests, tests, tests......... (Section 9, p. 42)

9 Module z3c.sqlalchemy.tests.testSQLAlchemy

Tests, tests, test	ts.							
--------------------	-----	--	--	--	--	--	--	--

9.1 Functions

test_suite()

9.2 Class WrapperTests

testWrapperRegistration(self)

9.2.1 Methods

setUp(self)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

testIFaceBaseWrapper(self)

testIFacePythonPostgres(self)

testIFaceZopePostgres(self)

testIModel(self)

testSimplePopulation(self)

testMapperWithCustomModel(self)

testCustomMapperClassWithWrongType(self)

testGetMappers(self)

testModelWeirdParameters(self)

testModelWeirdRelationsParameters(self)

testModelNonExistingTables(self)

 ${\bf testWrapperRegistrationFailing} (self)$

testWrapperDirectRegistration(self)

 ${\bf testMapperGetMapper}(\mathit{self})$

 $_$ call $_$ (self, *args, **kwds)

 $_{-}$ delattr $_{-}$ (...)

 $x._delattr_{-}('name') \le = > del x.name$

 $_$ getattribute $_(...)$

 $x._getattribute_('name') \le x.name$

 $_{-}\mathbf{hash}_{-}(x)$

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

 $__\mathbf{reduce}__(...)$

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).

${\bf assertAlmostEquals}(\textit{self}, \textit{first}, \textit{second}, \textit{places}{=}7, \textit{msg}{=}\texttt{None})$

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.

$\mathbf{countTestCases}(self)$

$\mathbf{debug}(self)$

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(self)

run(self, result=None)

shortDescription(self)

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.

tearDown(self)

Hook method for deconstructing the test fixture after testing it.

9.2.2 Properties

Name	Description		
class	Value: <attribute 'class'="" 'object'="" objects="" of=""></attribute>		

10 Module z3c.sqlalchemy.util

Some helper methods

10.1 Functions

createSAWrapper(dsn, model=None, forZope=False, name=None, **kw)

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

'dsn' - something like 'postgres://user:password@host/dbname'

'model' - None or an instance of model. Model or a string representing a named utility implementing IModel Provider or a method/callable returning an instance of model. Model.

'forZope' - set this to True in order to obtain a Zope-transaction-aware wrapper.

'name' can be set to register the wrapper automatically in order to avoid a dedicated registerSAWrapper() call.

createSQLAlchemyWrapper(dsn, model=None, forZope=False, name=None, **kw)

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

'dsn' - something like 'postgres://user:password@host/dbname'

'model' - None or an instance of model. Model or a string representing a named utility implementing IModel Provider or a method/callable returning an instance of model. Model.

'forZope' - set this to True in order to obtain a Zope-transaction-aware wrapper.

'name' can be set to register the wrapper automatically in order to avoid a dedicated registerSAWrapper() call.

registerSAWrapper(wrapper, name)

deferred registration of the wrapper as named utility

registerSQLAlchemyWrapper(wrapper, name)

deferred registration of the wrapper as named utility

getSAWrapper(name)

return a SQLAlchemyWrapper instance by name

getSQLAlchemyWrapper(name)

return a SQLAlchemyWrapper instance by name

allRegisteredSAW rappers()

return a dict containing information for all registered wrappers.

${\bf allRegistered SQLAlchemy Wrappers}()$

return a dict containing information for all registered wrappers.

${\bf all SAW rapper Names}()$

return list of all registered wrapper names

Index

```
dict._cmp__ (function), 19, 24
                                                        unittest.TestCase.countTestCases (function), 45
dict._contains_ (function), 19, 24
                                                         unittest. Test Case. debug (function), 45
dict._delitem__ (function), 19, 25
                                                        unittest.TestCase.defaultTestResult (function), 45
dict.__eq__ (function), 19, 25
                                                        unittest.TestCase.fail (function), 45
dict.__ge__ (function), 19, 25
                                                        unittest.TestCase.failIf (function), 44, 45
dict._getitem_ (function), 20, 25
                                                        unittest.TestCase.failIfAlmostEqual (function), 44, 45
dict.__gt__ (function), 20, 25
                                                        unittest.TestCase.failIfEqual (function), 44, 45
dict.__iter__ (function), 20, 25
                                                        unittest.TestCase.failUnless (function), 45
                                                        unittest.TestCase.failUnlessAlmostEqual (function), 43-
dict.__le__ (function), 20, 25
dict._len__ (function), 20, 25
dict.__lt__ (function), 20, 25
                                                        unittest.TestCase.failUnlessEqual (function), 44, 45
dict.__ne__ (function), 20, 26
                                                        unittest.TestCase.failUnlessRaises (function), 44, 46
dict._setitem_ (function), 21, 26
                                                        unittest.TestCase.id (function), 46
dict.clear (function), 21, 26
                                                        unittest.TestCase.run (function), 46
dict.copy (function), 21, 26
                                                        unittest. Test Case. short Description (function), 46
dict.fromkeys (function), 21, 27
                                                        unittest.TestCase.tearDown (function), 46
dict.get (function), 21, 27
                                                        z3c (package)
dict.has_key (function), 21, 27
                                                             z3c.sqlalchemy (package), 3
dict.items (function), 21
                                                               z3c.sqlalchemy.base (module), 4–12
dict.iteritems (function), 21, 27
                                                                z3c.sqlalchemy.interfaces (module), 13–15
dict.iterkeys (function), 22, 27
                                                                z3c.sqlalchemy.mapper (module), 16–23
dict.itervalues (function), 22, 27
                                                                z3c.sqlalchemy.model (module), 24–28
dict.keys (function), 22, 27
                                                                z3c.sqlalchemy.postgres (module), 29–34
dict.pop (function), 22, 27
                                                                z3c.sqlalchemy.test (module), 35–40
dict.popitem (function), 22, 27
                                                                z3c.sqlalchemy.tests (package), 41
dict.setdefault (function), 22, 27
                                                                z3c.sqlalchemy.util (module), 47-48
dict.update (function), 22, 28
dict.values (function), 22, 28
object._delattr_ (function), 4, 5, 7, 9, 10, 16, 18, 19,
         25, 29, 30, 32, 35, 38, 43
object._getattribute_ (function), 4, 5, 7, 9, 10, 16,
         18, 29, 30, 32, 35, 38, 43
object._hash__ (function), 4, 5, 7, 9, 10, 16, 18, 29,
         30, 32, 35, 38, 43
object.__init__ (function), 29
object._new_ (function), 4, 5, 7, 9, 11, 16, 18, 29,
         31, 33, 36, 38, 43
object._reduce_ (function), 4, 6, 8, 9, 11, 16, 18, 20,
         26, 29, 31, 33, 36, 38, 43
object._reduce_ex_ (function), 4, 6, 8, 9, 11, 17, 18,
         20, 26, 29, 31, 33, 36, 38, 43
object._repr_ (function), 5, 6, 8, 9, 11, 17, 18, 29,
         31, 33, 36, 38
object._setattr_ (function), 5, 6, 8, 10, 11, 17, 18,
         21, 26, 30, 31, 33, 36, 38, 43
object._str_ (function), 5, 6, 8, 10, 11, 17, 18, 21,
         26, 30, 31, 33, 36, 38
```

unittest.TestCase.__call__ (function), 43