

---

# Abilian Core Documentation

*Release 0.1*

Stefane Fermigier

2019-07-23



# CONTENTS

<b>I</b>	<b>Contents</b>	<b>3</b>
1	About Abilian Core	5
2	Installing Abilian Core	9
3	Contributing to Abilian Core	11
4	Coding standard	13
5	API	15
6	Changelog for Abilian Core	67
7	Credits	81
<b>II</b>	<b>Indices and tables</b>	<b>83</b>
	Index	85





Welcome to Abilian Core's documentation.

Abilian Core is an enterprise application development platform based on the [Flask micro-framework](#), the [SQLAlchemy ORM](#), good intentions and best practices (for some value of "best").

The full documentation is available on <http://docs.abilian.com/>.

It builds on powerful and well documented Python libraries, mainly:

- [Flask](#)
- [SQLAlchemy](#)
- [WTForms](#)

This documentation will assume that a developer already has some knowledge of these libraries.



Part I  
CONTENTS





# ABOUT ABILIAN CORE

Abilian Core is an enterprise application development platform based on the [Flask micro-framework](#), the [SQLAlchemy ORM](#), good intentions and best practices (for some value of “best”).

The full documentation is available on <http://docs.abilian.com/>.

## 1.1 Goals & principles

- Development must be easy and fun (some some definition of “easy” and “fun”, of course)
- The less code (and configuration) we write, the better
- Leverage existing reputable open source libraries and frameworks, such as SQLAlchemy and Flask
- It must lower errors, bugs, project’s time to deliver. It’s intended to be a rapid application development tool
- It must promote best practices in software development, specially Test-Driven Development (as advocated by the [GOOS book](#))

## 1.2 Features

Here’s a short list of features that you may find appealing in Abilian:

### 1.2.1 Infrastructure

- Plugin framework
- Asynchronous tasks (using [Celery](#))
- Security model and service

## 1.2.2 Domain model and services

- Domain object model, based on SQLAlchemy
- Audit

## 1.2.3 Content management and services

- Indexing service
- Document preview and transformation

## 1.2.4 Social

- Users, groups and social graph (followers)
- Activity streams

## 1.2.5 User Interface and API

- Forms (based on [WTForms](#))
- CRUD (Create, Retrieve, Edit/Update, Remove) interface from domain models
- Labels and descriptions for each field
- Various web utilities: view decorators, class-based views, Jinja2 filters, etc.
- A default UI based on [Bootstrap 3](#) and several carefully selected jQuery plugins such as [Select2](#)
- REST and AJAX API helpers
- i18n: support for multi-language via Babel, with multiple translation dictionaries

## 1.2.6 Management and admin

- Initial settings wizard
- Admin and user settings framework
- System monitoring (using [Sentry](#))

## 1.3 Current status

Abilian Core is currently alpha (or even pre-alpha) software, in terms of API stability. It is currently used in several applications that have been developed by [Abilian](#) over the last two years:

- Abilian SBE (Social Business Engine) - an enterprise 2.0 (social collaboration) platform
- Abilian EMS (Event Management System)
- Abilian CRM (Customer / Contact / Community Relationship Management System)
- Abilian Le MOOC - a MOOC prototype
- Abilian CMS - a Web CMS

In other words, Abilian Core is the foundation for a small, but growing, family of business-critical applications that our customers intend us to support in the coming years.

So while Abilian Core APIs, object model and even architecture, may (and most probably will) change due to various refactorings that are expected as we can't be expected to ship perfect software on the first release, we also intend to treat it as a valuable business asset and keep maintaining and improving it in the foreseeable future.

## 1.4 Roadmap & getting involved

We have a [roadmap on Pivotal Tracker](#) that we use internally to manage our iterative delivery process.

For features and bug requests (or is it the other way around?), we recommend that you use the [GitHub issue tracker](#).

## 1.5 Licence

Abilian Core is licensed under the LGPL.

## 1.6 Credits

Abilian Core has been created by the development team at Abilian (currently: Stefane and Bertrand), with financial support from our wonderful customers, and R&D fundings from the French Government, the Paris Region and the European Union.

We are also specially grateful to:

- [Armin Ronacher](#) for his work on Flask.
- [Michael Bayer](#) for his work on SQLAlchemy.
- Everyone who has been involved with and produced open source software for the Flask ecosystem (Kiran Jonnalagadda and the [HasGeek](#) team, Max Countryman, Matt Wright, Matt Good, Thomas Johansson, James Crasta, and probably many others).

- The creators of Django, Pylons, TurboGears, Pyramid and Zope, for even more inspiration.
- The whole Python community.

---

# INSTALLING ABILIAN CORE

---

If you are a Python web developer (which is the primary target for this project), you probably already know about:

- Python 2.7
- Virtualenv
- Pip

So, after you have created and activated a virtualenv for the project, just run:

```
pip install -r requirements.txt
```

To use some features of the library, namely document and images transformation, you will need to install the additional native packages, using our operating system's package management tools (dpkg, yum, brew...):

- A few image manipulation libraries (libpng, libjpeg)
- The poppler-utils, unoconv, LibreOffice, ImageMagick utilities
- [lesscss](#):

For Debian/Ubuntu the package is named *node-less*. If your distribution's package is too old, you may install [node-js](#)  $\geq 0.10$  and [npm](#). Lesscss can then be installed with:

```
$ sudo npm install -g less
npm http GET https://registry.npmjs.org/less
npm http 200 https://registry.npmjs.org/less
...
$ which lessc
/usr/bin/lessc
```

## 2.1 Testing

Abilian Core come with a full unit and integration testing suite. You can run it with `make test` (once your virtualenv has been activated).

Alternatively, you can use tox to run the full test suite in an isolated environment.

# CONTRIBUTING TO ABILIAN CORE

## 3.1 Project on GitHub

The project is hosted on GitHub at: <https://github.com/abilian/abilian-core>.

Participation in the development of Abilian is welcome and encouraged, through the various mechanisms provided by GitHub:

- [Bug reports and feature requests](#).
- [Forks and pull requests](#).

## 3.2 License and copyright

The Abilian code is copyrighted by Abilian SAS, a french company.

It is licenced under the LGPL (Lesser General Public License), which means you can reuse the product as a library

If you contribute to Abilian, we ask you to transfer your rights to your contribution to us.

In case you have questions, you're welcome to contact us.

## 3.3 Build Status

We give a great deal of care to the quality of our software, and try to use all the tools that are at our disposal to make it rock-solid.

This includes:

- Having an exhaustive test suite.
- Using continuous integration (CI) servers to run the test suite on every commit.
- Running tests.
- Using our products daily.

You can check the build status:

- [On Travis CI](#)

You can also check the coverage reports:

- [On coveralls.io](#)

## 3.4 Releasing

We're now using *setuptools\_scm* to manage version numbers.

It comes with some conventions on its own when it comes to releasing.

Here's what you should do to make a new release on PyPI:

1. Check that the `CHANGES.rst` file is correct.
2. Commit.
3. Tag (ex: `git tag 0.3.0`), using numbers that are consistent with semantic versioning.
4. Run `python setup.py sdist upload`.



# CODING STANDARD

We recommend using the PEP8 and Google coding standard, with the following exceptions:

- Indentation should be 2 chars, not 4.

## 4.1 Additional rules

TODO

## 4.2 Notes

### 4.2.1 Line length

We stick to the “no lines longer than 80 characters” rule despite the fact that we’re living in a post VT-220 world.

Here’s [some rationale](#) by user “badsector” on Reddit:

I used to use a 120 character limit or ignore E501 on my pep8 checker (python), but eventually went back to the default 80 character limit. I realized it did more for me than let me fit 4 files side by side on a laptop screen:

- It discouraged me from writing long sprawling if statements and method chains.
- With less space, I thought more assigning about clear and concise names for things.
- I would break out deeply nested ifs and other control statements into separate functions. This is probably the biggest win since smaller code pieces are easier to unit test due to lowered cyclomatic complexity.



## 5.1 Package abilian

### 5.1.1 Module abilian.app

Base Flask application class, used by tests or to be extended in real applications.

**class** `Application`(*name*: *Optional*[*Any*] = *None*, \**args*, \*\**kwargs*)

Base application class.

Extend it in your own app.

**celery\_app\_cls**

alias of `abilian.core.celery.FlaskCelery`

**add\_access\_controller**(*name*: *str*, *func*: *Callable*, *endpoint*: *bool* = *False*) →

*None*  
Add an access controller.

If *name* is *None* it is added at application level, else if is considered as a blueprint name. If *endpoint* is *True* then it is considered as an endpoint.

**add\_static\_url**(*url\_path*: *str*, *directory*: *str*, *endpoint*: *str*, *roles*: *Collection*[*abilian.services.security.models.Role*] = ()) → *None*

Add a new url rule for static files.

#### Parameters

- **url\_path** – subpath from application static url path. No heading or trailing slash.
- **directory** – directory to serve content from.
- **endpoint** – flask endpoint name for this url rule.

Example:

```
app.add_static_url('myplugin',
                  '/path/to/myplugin/resources',
                  endpoint='myplugin_static')
```

With default setup it will serve content from directory `/path/to/myplugin/resources` from url `http://.../static/myplugin`

**add\_url\_rule\_with\_role**(*rule: str, endpoint: str, view\_func: Callable, roles: Collection[abilian.services.security.models.Role] = (), \*\*options*) → None

See `Flask.add_url_rule()`.

If *roles* parameter is present, it must be a `abilian.service.security.models.Role` instance, or a list of `Role` instances.

**check\_instance\_folder**(*create=False*)

Verify instance folder exists, is a directory, and has necessary permissions.

**Param: create** if *True*, creates directory hierarchy

**Raises** `OSError` with relevant `errno` if something is wrong.

**configure**(*config: Optional[type]*) → None

**init\_breadcrumbs**() → None

Insert the first element in breadcrumbs.

This happens during `request_started` event, which is triggered before any `url_value_preprocessor` and `before_request` handlers.

**init\_extensions**() → None

Initialize flask extensions, helpers and services.

**install\_id\_generator**(*sender: flask.app.Flask, \*\*kwargs*) → None

**setup**(*config: Optional[type]*) → None

**setup\_nav\_and\_breadcrumbs**(*app: flask.app.Flask*) → None

Listener for `request_started` event.

If you want to customize first items of breadcrumbs, override `init_breadcrumbs()`

**data\_dir**

**default\_config** = {'ABILIAN\_UPSTREAM\_INFO\_ENABLED': False, 'ADMIN\_PANELS': ('abilian

**default\_view** = None

instance of `web.views.registry.Registry`.

**js\_api** = None

json serializable dict to land in Javascript under `Abilian.api`

**private\_site**

If *True* all views will require by default an authenticated user, unless `Anonymous` role is authorized. Static assets are always public.

**class ServiceManager**

Mixin that provides lifecycle (`register/start/stop`) support for services.

**start\_services**()

**stop\_services**()

```
create_app(config: Optional[type] = None, app_class: type = <class 'abilian.app.Application'>, **kw) → abilian.app.Application
```

## 5.1.2 Module abilian.i18n

I18n.

To mark strings for translation:

```
from abilian.i18n import _
_(u'message to translate')
```

Use `_` for `gettext`, `_l` for `lazy_gettext`, `_n` for `ngettext`.

Babel extension support multiple translation paths. This allows to add more catalogs to search for translations, in LIFO order. This feature can be used to override some translations in a custom application, be providing a catalog with messages to override:

```
current_app.extensions['babel'].add_translations('abilian.core')
```

See `add_translations`.

To extract messages to build the message catalog template (.pot), use the following “-k” parameters:

```
$ pybabel extract -F babel.cfg -k "_n:1,2" -k "_l" -o "msg.pot" "src"
```

This can be made easier by placing in `setup.cfg`:

```
[extract_messages]
mapping_file = babel.cfg
keywords = _n:1,2 _l
output-file = msg.pot
input-dirs = src
```

And just type:

```
$ python setup.py extract_messages
```

```
_ = <function gettext>
    gettext alias
```

```
_l = <function lazy_gettext>
    lazy_gettext alias
```

```
_n = <function ngettext>
    ngettext alias
```

```
class Babel(app=None, default_locale='en', default_timezone='UTC', default_domain='messages', date_formats=None, configure_jinja=True)
    Bases: flask_babel.Babel
```

Allow to load translations from other modules.

**add\_translations**(*module\_name: str, translations\_dir: str = 'translations', domain: str = 'messages'*) → None  
Add translations from external module.

For example:

```
babel.add_translations('abilian.core')
```

Will add translations files from *abilian.core* module.

**init\_app**(*app: flask.app.Flask*) → None  
Set up this instance for use with *app*, if no app was passed to the constructor.

**babel** = <abilian.i18n.Babel object>  
importable instance of Babel

**get\_default\_locale**() → babel.core.Locale

**gettext**(*string, \*\*variables*)  
Translates a string with the current locale and passes in the given keyword arguments as mapping to a string formatting string.

```
gettext(u'Hello World!')  
gettext(u'Hello %(name)s!', name='World')
```

**lazy\_country\_name**(*code*)

**lazy\_gettext**(*string, \*\*variables*)  
Like `gettext()` but the string returned is lazy which means it will be translated when it is used as an actual string.

Example:

```
hello = lazy_gettext(u'Hello World')  
  
@app.route('/')  
def index():  
    return unicode(hello)
```

**localeselector**() → Optional[str]  
Default locale selector used in abilian applications.

**ngettext**(*singular, plural, num, \*\*variables*)  
Translates a string with the current locale and passes in the given keyword arguments as mapping to a string formatting string. The *num* parameter is used to dispatch between singular and various plural forms of the message. It is available in the format string as `%(num)d` or `%(num)s`. The source language should be English or a similar language which only has one plural form.

```
ngettext(u'%(num)d Apple', u'%(num)d Apples', num=len(apples))
```

**render\_template\_i18n**(*template\_name\_or\_list: str, \*\*context*) → str  
Try to build an ordered list of template to satisfy the current locale.

**timezoneselector()** → `datetime.tzinfo`  
Default timezone selector used in abilian applications.

**babel** = `<abilian.i18n.Babel object>`  
importable instance of Babel

**VALID\_LANGUAGES\_CODE** = `frozenset({'af', 'ak', 'am', 'ar', 'as', 'az', 'be', 'bg', 'bm', ...})`  
accepted languages codes

## 5.2 Package `abilian.core`

### 5.2.1 Module `abilian.core.entities`

Base class for entities, objects that are managed by the Abilian framwework (unlike SQLAlchemy models which are considered lower-level).

**class Entity(\*args, \*\*kwargs)**  
Bases: `abilian.core.models.base.Indexable`, `abilian.core.models.BaseMixin`, `abilian.core.models.base.Model`

Base class for Abilian entities.

From Sqlalchemy POV, Entities use [Joined-Table inheritance](#), thus entities sub-classes cannot use inheritance themselves (as of 2013 Sqlalchemy does not support multi-level inheritance)

The metaclass automatically sets up polymorphic inheritance parameters by inserting a mixin class in parent classes. If you need to pass additional parameters to `__mapper_args__`, do it as follow:

```
class MyContent(Entity):

    @sqlalchemy.ext.declarative.declared_attr
    def __mapper_args__(cls):
        # super(Mycontent, cls).__mapper_args__ would be prettier, but
        # `MyContent` is not defined at this stage.
        args = Entity.__dict__['__mapper_args__'].fget(cls)
        args['order_by'] = cls.created_at # for example
        return args
```

**query\_class**  
alias of `EntityQuery`

**\_\_init\_\_(\*args, \*\*kwargs)**

**clone()**  
Copy an entity: copy every field, except the id and sqlalchemy internals, without forgetting about the n-n relationships.

- return: the newly created entity

Example:

```

def clone(self):
    old_attrs = self.__dict__.copy()
    del old_attrs['_sa_instance_state']
    if 'id' in old_attrs:
        del old_attrs['id']
    new = AnEntity(**old_attrs)
    # Needs special treatment for n-n relationship
    new.related_projects = self.related_projects
    new.ancestor = self
    return new

```

**display\_value**(*field\_name*, *value*=<object object>)

Return display value for fields having 'choices' mapping (stored value.

-> human readable value). For other fields it will simply return field value.

*display\_value* should be used instead of directly getting field value.

If *value* is provided, it is "translated" to a human-readable value. This is useful for obtaining a human readable label from a raw value.

**SLUG\_SEPARATOR** = '-'

**\_\_annotations\_\_** = {'\_\_auditable\_\_': typing.FrozenSet, '\_\_editable\_\_': typing.Froze

**\_\_auditable\_\_** = frozenset({})

**\_\_default\_permissions\_\_** = frozenset({})

Permission to roles mapping to set at object creation time.

Default permissions can be declared as a `dict` on classes, the final datastructure will be changed by metaclass to a `frozenset` of `dict.items()`. This is made to guarantee the immutability of definition on parent classes.

Exemple definition:

```

__default_permissions__ = {
    READ: {Owner, Authenticated},
    WRITE: {Owner},
}

```

To alter inherited default permissions:

```

class Child(Parent):
    __default_permissions__ = dp = dict(ParentClass.__default_
    ↪permissions__)
    dp[READ] = dp[READ] - {Authenticated} + {Anonymous}
    del dp

```

**\_\_editable\_\_** = frozenset({})

**\_\_index\_to\_\_** = (('indexable\_roles\_and\_users', ('allowed\_roles\_and\_users',)), ('\_in

**\_\_indexable\_\_** = False



```
__mapper__ = <Mapper at 0x7f6eecb5f588; Entity>
__mapper_args__ = {'polymorphic_on': '_entity_type'}
__module__ = 'abilian.core.entities'
__permissions__
__searchable__ = frozenset({})
__table__ = Table('entity', MetaData(bind=None), Column('id', Integer(), table=<ent
```

#### **property auto\_slug**

This property is used to auto-generate a slug from the name attribute.

It can be customized by subclasses.

#### **created\_at**

#### **creator**

#### **creator\_id**

#### **deleted\_at**

#### **property entity\_class**

**entity\_type** = ''

#### **id**

#### **meta**

A dictionary of simple values (JSON-serializable) to conveniently annotate the entity.

It is recommended to keep it lightweight and not store large objects in it.

#### **name**

The name is a string that is shown to the user; it could be a title for document, a folder name, etc.

#### **property object\_type**

#### **owner**

#### **owner\_id**

#### **slug**

The slug attribute may be used in URLs to reference the entity, but uniqueness is not enforced, even within same entity type. For example if an entity class represent folders, one could want uniqueness only within same parent folder.

If slug is empty at first creation, its is derived from the name. When name changes the slug is not updated. If name is also empty, the slug will be the friendly entity\_type with concatenated with entity's id.

#### **updated\_at**

**exception ValidationError**

**class Entity(\*args, \*\*kwargs)**  
Base class for Abilian entities.

From Sqlalchemy POV, Entities use **Joined-Table inheritance**, thus entities subclasses cannot use inheritance themselves (as of 2013 Sqlalchemy does not support multi-level inheritance)

The metaclass automatically sets up polymorphic inheritance parameters by inserting a mixin class in parent classes. If you need to pass additional parameters to `__mapper_args__`, do it as follow:

```
class MyContent(Entity):

    @sqlalchemy.ext.declarative.declared_attr
    def __mapper_args__(cls):
        # super(Mycontent, cls).__mapper_args__ would be prettier, but
        # `MyContent` is not defined at this stage.
        args = Entity.__dict__['__mapper_args__'].fget(cls)
        args['order_by'] = cls.created_at # for example
        return args
```

**query\_class**  
alias of EntityQuery

**clone()**  
Copy an entity: copy every field, except the id and sqlalchemy internals, without forgetting about the n-n relationships.

- return: the newly created entity

Example:

```
def clone(self):
    old_attrs = self.__dict__.copy()
    del old_attrs['_sa_instance_state']
    if 'id' in old_attrs:
        del old_attrs['id']
    new = AnEntity(**old_attrs)
    # Needs special treatment for n-n relationship
    new.related_projects = self.related_projects
    new.ancestor = self
    return new
```

**display\_value(field\_name, value=<object object>)**  
Return display value for fields having 'choices' mapping (stored value. -> human readable value). For other fields it will simply return field value.  
*display\_value* should be used instead of directly getting field value.  
If *value* is provided, it is "translated" to a human-readable value. This is useful for obtaining a human readable label from a raw value.

**SLUG\_SEPARATOR = '-'**

**property auto\_slug**

This property is used to auto-generate a slug from the name attribute.

It can be customized by subclasses.

**created\_at****creator****creator\_id****deleted\_at****property entity\_class**

**entity\_type** = ''

**id****meta**

A dictionary of simple values (JSON-serializable) to conveniently annotate the entity.

It is recommended to keep it lightweight and not store large objects in it.

**name**

The name is a string that is shown to the user; it could be a title for document, a folder name, etc.

**property object\_type****owner****owner\_id****slug**

The slug attribute may be used in URLs to reference the entity, but uniqueness is not enforced, even within same entity type. For example if an entity class represent folders, one could want uniqueness only within same parent folder.

If slug is empty at first creation, its is derived from the name. When name changes the slug is not updated. If name is also empty, the slug will be the friendly entity\_type with concatenated with entity's id.

**updated\_at**

```
class EntityQuery(entities, session=None)
```

```
    with_permission(permission: Permission, user: Optional[User] = None) → EntityQuery
```

```
class Indexable
```

Mixin with sensible defaults for indexable objects.

```
    property object_key
```

```
    property object_type
```

## **all\_entity\_classes**

Return the list of all concrete persistent classes that are subclasses of Entity.

## 5.2.2 Module `abilian.core.extensions`

Create all standard extensions.

### **get\_extension**(*name: str*)

Get the named extension from the current app, returning None if not found.

## 5.2.3 Module `abilian.core.logging`

Special loggers

Changing `patch_logger` logging level must be done very early, because it may emit logging during imports. Ideally, it's should be the very first action in your entry point before anything has been imported:

```
import logging
logging.getLogger('PATCH').setLevel(logging.INFO)
```

```
patch_logger = <PatchLoggerAdapter PATCH (WARNING)>
    logger      for      monkey      patches.      use      like      this:
    patch_logger.info(<func>'patched_func')
```

## 5.2.4 Module `abilian.core.signals`

All signals used by Abilian Core.

Signals are the main tools used for decoupling applications components by sending notifications. In short, signals allow certain senders to notify subscribers that something happened.

Cf. <http://flask.pocoo.org/docs/signals/> for detailed documentation.

The main signal is currently activity.

```
activity = <blinker.base.NamedSignal object at 0x7f6ef34d8208; 'activity'>
```

This signal is used by the activity streams service and its clients.

```
components_registered = <blinker.base.NamedSignal object at 0x7f6ef34d81d0; 'app:components'>
```

Triggered at application initialization when all extensions and plugins have been loaded

```
register_js_api = <blinker.base.NamedSignal object at 0x7f6ef34d8160; 'app:register-js-api'>
```

Trigger when JS api must be registered. At this time `flask.url_for()` is usable

```
user_loaded = <blinker.base.NamedSignal object at 0x7f6ef34d8240; 'user_loaded'>
```

This signal is sent when user object has been loaded. `g.user` and `current_user` are available.

## 5.2.5 Module `abilian.core.sqlalchemy`

Additional data types for sqlalchemy.

**class** `JSON(*args, **kwargs)`

Stores any structure serializable with json.

Usage `JSON()` Takes same parameters as `sqlalchemy.types.Text`

**impl**

alias of `sqlalchemy.sql.sqltypes.Text`

**process\_bind\_param**(*value: Any, dialect: sqlalchemy.engine.interfaces.Dialect*)  
→ `Optional[str]`

Receive a bound parameter value to be converted.

Subclasses override this method to return the value that should be passed along to the underlying `TypeEngine` object, and from there to the `DBAPI execute()` method.

The operation could be anything desired to perform custom behavior, such as transforming or serializing data. This could also be used as a hook for validating logic.

This operation should be designed with the reverse operation in mind, which would be the `process_result_value` method of this class.

### Parameters

- **value** – Data to operate upon, of any type expected by this method in the subclass. Can be `None`.
- **dialect** – the `Dialect` in use.

**process\_result\_value**(*value: Optional[str], dialect: sqlalchemy.engine.interfaces.Dialect*)  
→ `Union[Dict[str, Any], List[int], None]`

Receive a result-row column value to be converted.

Subclasses should implement this method to operate on data fetched from the database.

Subclasses override this method to return the value that should be passed back to the application, given a value that is already processed by the underlying `TypeEngine` object, originally from the `DBAPI cursor method fetchone()` or similar.

The operation could be anything desired to perform custom behavior, such as transforming or serializing data. This could also be used as a hook for validating logic.

### Parameters

- **value** – Data to operate upon, of any type expected by this method in the subclass. Can be `None`.
- **dialect** – the `Dialect` in use.

This operation should be designed to be reversible by the “process\_bind\_param” method of this class.

**class JSONUniqueListType(\*args, \*\*kwargs)**

Store a list in JSON format, with items made unique and sorted.

**process\_bind\_param(value, dialect)**

Receive a bound parameter value to be converted.

Subclasses override this method to return the value that should be passed along to the underlying TypeEngine object, and from there to the DBAPI execute() method.

The operation could be anything desired to perform custom behavior, such as transforming or serializing data. This could also be used as a hook for validating logic.

This operation should be designed with the reverse operation in mind, which would be the process\_result\_value method of this class.

#### Parameters

- **value** – Data to operate upon, of any type expected by this method in the subclass. Can be None.
- **dialect** – the Dialect in use.

**property python\_type**

Return the Python type object expected to be returned by instances of this type, if known.

Basically, for those types which enforce a return type, or are known across the board to do such for all common DBAPIs (like int for example), will return that type.

If a return type is not defined, raises NotImplementedError.

Note that any type also accommodates NULL in SQL which means you can also get back None from any type in practice.

**class Locale(\*args, \*\*kwargs)**

Store a babel.Locale instance.

**impl**

alias of sqlalchemy.sql.sqltypes.UnicodeText

**process\_bind\_param(value: Optional[Any], dialect: sqlalchemy.engine.interfaces.Dialect) → Optional[Any]**

Receive a bound parameter value to be converted.

Subclasses override this method to return the value that should be passed along to the underlying TypeEngine object, and from there to the DBAPI execute() method.

The operation could be anything desired to perform custom behavior, such as transforming or serializing data. This could also be used as a hook for validating logic.

This operation should be designed with the reverse operation in mind, which would be the `process_result_value` method of this class.

#### Parameters

- **value** – Data to operate upon, of any type expected by this method in the subclass. Can be None.
- **dialect** – the `Dialect` in use.

`process_result_value`(*value*: `Optional[Any]`, `sqlalchemy.engine.interfaces.Dialect`) → `Optional[Any]` *dialect*: `Optional[Any]`

Receive a result-row column value to be converted.

Subclasses should implement this method to operate on data fetched from the database.

Subclasses override this method to return the value that should be passed back to the application, given a value that is already processed by the underlying `TypeEngine` object, originally from the DBAPI cursor method `fetchone()` or similar.

The operation could be anything desired to perform custom behavior, such as transforming or serializing data. This could also be used as a hook for validating logic.

#### Parameters

- **value** – Data to operate upon, of any type expected by this method in the subclass. Can be None.
- **dialect** – the `Dialect` in use.

This operation should be designed to be reversible by the “`process_bind_param`” method of this class.

#### `property python_type`

Return the Python type object expected to be returned by instances of this type, if known.

Basically, for those types which enforce a return type, or are known across the board to do such for all common DBAPIs (like `int` for example), will return that type.

If a return type is not defined, raises `NotImplementedError`.

Note that any type also accommodates `NULL` in SQL which means you can also get back `None` from any type in practice.

#### `class MutationDict`

Provides a dictionary type with mutability support.

`clear()` → `None`. Remove all items from `D`.

**classmethod** `coerce(key: str, value: Dict) → abilian.core.sqlalchemy.MutationDict`  
Convert plain dictionaries to MutationDict.

**pop**(`k`, `d`) → `v`, remove specified key and return the corresponding value.  
If key is not found, `d` is returned if given, otherwise `KeyError` is raised

**popitem**() → (`k`, `v`), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if `D` is empty.

**setdefault**(`key`, `failobj=None`)  
Insert key with a value of default if key is not in the dictionary.  
Return the value for key if key is in the dictionary, else default.

**update**(`[E]`, `**F`) → `None`. Update `D` from dict/iterable `E` and `F`.  
If `E` is present and has a `.keys()` method, then does: for `k` in `E`: `D[k] = E[k]` If  
`E` is present and lacks a `.keys()` method, then does: for `k`, `v` in `E`: `D[k] = v` In  
either case, this is followed by: for `k` in `F`: `D[k] = F[k]`

### **class MutationList**

Provides a list type with mutability support.

**append**(`item: Any`) → `None`  
Append object to the end of the list.

**classmethod** `coerce(key: str, value: List) → abilian.core.sqlalchemy.MutationList`  
Convert list to MutationList.

**extend**(`other`)  
Extend list by appending elements from the iterable.

**insert**(`idx`, `value`)  
Insert object before index.

**pop**(`i=-1`)  
Remove and return item at index (default last).  
Raises `IndexError` if list is empty or index is out of range.

**remove**(`item`)  
Remove first occurrence of value.  
Raises `ValueError` if the value is not present.

**reverse**()  
Reverse *IN PLACE*.

**sort**(`*args`, `**kwargs`)  
Stable sort *IN PLACE*.

**class SQLAlchemy**(`app=None`, `use_native_unicode=True`, `session_options=None`,  
`metadata=None`)

Base subclass of `flask_sqlalchemy.SQLAlchemy`.

Add our custom driver hacks.



**apply\_driver\_hacks**(*app*: flask.app.Flask, *info*: sqlalchemy.engine.url.URL, *options*: Dict[str, Any]) → None

This method is called before engine creation and used to inject driver specific hacks into the options. The *options* parameter is a dictionary of keyword arguments that will then be used to call the `sqlalchemy.create_engine()` function.

The default implementation provides some saner defaults for things like pool sizes for MySQL and sqlite. Also it injects the setting of `SQLALCHEMY_NATIVE_UNICODE`.

**class Timezone**(\*args, \*\*kwargs)

Store a `pytz.tzfile.DstTzInfo` instance.

**impl**

alias of `sqlalchemy.sql.sqltypes.UnicodeText`

**process\_bind\_param**(*value*: Optional[Any], *dialect*: sqlalchemy.engine.interfaces.Dialect) → Optional[Any]

Receive a bound parameter value to be converted.

Subclasses override this method to return the value that should be passed along to the underlying TypeEngine object, and from there to the DBAPI `execute()` method.

The operation could be anything desired to perform custom behavior, such as transforming or serializing data. This could also be used as a hook for validating logic.

This operation should be designed with the reverse operation in mind, which would be the `process_result_value` method of this class.

#### Parameters

- **value** – Data to operate upon, of any type expected by this method in the subclass. Can be None.
- **dialect** – the Dialect in use.

**process\_result\_value**(*value*: Optional[Any], *dialect*: sqlalchemy.engine.interfaces.Dialect) → Optional[Any]

Receive a result-row column value to be converted.

Subclasses should implement this method to operate on data fetched from the database.

Subclasses override this method to return the value that should be passed back to the application, given a value that is already processed by the underlying TypeEngine object, originally from the DBAPI cursor method `fetchone()` or similar.

The operation could be anything desired to perform custom behavior, such as transforming or serializing data. This could also be used as a hook for validating logic.

### Parameters

- **value** – Data to operate upon, of any type expected by this method in the subclass. Can be None.
- **dialect** – the Dialect in use.

This operation should be designed to be reversible by the “process\_bind\_param” method of this class.

### property python\_type

Return the Python type object expected to be returned by instances of this type, if known.

Basically, for those types which enforce a return type, or are known across the board to do such for all common DBAPIs (like int for example), will return that type.

If a return type is not defined, raises NotImplementedError.

Note that any type also accommodates NULL in SQL which means you can also get back None from any type in practice.

**class UUID(\*args, \*\*kwargs)**

Platform-independent UUID type.

Uses PostgreSQL’s UUID type, otherwise uses CHAR(32), storing as stringified hex values.

From SQLAlchemy documentation.

### impl

alias of sqlalchemy.sql.sqltypes.CHAR

**compare\_against\_backend(dialect, conn\_type)**

Compare this type against the given backend type.

This function is currently not implemented for SQLAlchemy types, and for all built in types will return None. However, it can be implemented by a user-defined type where it can be consumed by schema comparison tools such as Alembic autogenerate.

A future release of SQLAlchemy will potentially implement this method for builtin types as well.

The function should return True if this type is equivalent to the given type; the type is typically reflected from the database so should be database specific. The dialect in use is also passed. It can also return False to assert that the type is not equivalent.

### Parameters

- **dialect** – a Dialect that is involved in the comparison.
- **conn\_type** – the type object reflected from the backend.

New in version 1.0.3.

**load\_dialect\_impl**(*dialect*: `sqlalchemy.engine.interfaces.Dialect`) → `sqlalchemy.sql.sqltypes.CHAR`

Return a TypeEngine object corresponding to a dialect.

This is an end-user override hook that can be used to provide differing types depending on the given dialect. It is used by the TypeDecorator implementation of `type_engine()` to help determine what type should ultimately be returned for a given TypeDecorator.

By default returns `self.impl`.

**process\_bind\_param**(*value*: `Union[None, str, uuid.UUID]`, *dialect*: `sqlalchemy.engine.interfaces.Dialect`) → `Optional[str]`

Receive a bound parameter value to be converted.

Subclasses override this method to return the value that should be passed along to the underlying TypeEngine object, and from there to the DBAPI `execute()` method.

The operation could be anything desired to perform custom behavior, such as transforming or serializing data. This could also be used as a hook for validating logic.

This operation should be designed with the reverse operation in mind, which would be the `process_result_value` method of this class.

#### Parameters

- **value** – Data to operate upon, of any type expected by this method in the subclass. Can be None.
- **dialect** – the Dialect in use.

**process\_result\_value**(*value*: `Optional[str]`, *dialect*: `sqlalchemy.engine.interfaces.Dialect`) → `Optional[uuid.UUID]`

Receive a result-row column value to be converted.

Subclasses should implement this method to operate on data fetched from the database.

Subclasses override this method to return the value that should be passed back to the application, given a value that is already processed by the underlying TypeEngine object, originally from the DBAPI cursor method `fetchone()` or similar.

The operation could be anything desired to perform custom behavior, such as transforming or serializing data. This could also be used as a hook for validating logic.

#### Parameters

- **value** – Data to operate upon, of any type expected by this method in the subclass. Can be None.
- **dialect** – the Dialect in use.

This operation should be designed to be reversible by the “process\_bind\_param” method of this class.

**JSONDict**(\*args, \*\*kwargs)

Stores a dict as JSON on database, with mutability support.

**JSONList**(\*args, \*\*kwargs)

Stores a list as JSON on database, with mutability support.

If kwargs has a param *unique\_sorted* (which evaluated to True), list values are made unique and sorted.

**filter\_cols**(model, \*filtered\_columns)

Return columnnames for a model except named ones.

Useful for defer() for example to retain only columns of interest

**ping\_connection**(dbapi\_connection: sqlite3.Connection, connection\_record, connection\_proxy) → None

Ensure connections are valid.

From: [http://docs.sqlalchemy.org/en/rel\\_0\\_8/core/pooling.html](http://docs.sqlalchemy.org/en/rel_0_8/core/pooling.html)

In case db has been restarted pool may return invalid connections.

## 5.2.6 Module `abilian.core.models`

**class IdMixin**

`id = Column(None, Integer(), table=None, primary_key=True, nullable=False)`

**class Indexable**

Mixin with sensible defaults for indexable objects.

**property object\_key**

**property object\_type**

**class Info**(\*\*kw)

`copy()` → a shallow copy of D

**class Model**(\*\*kwargs)

**class TimestampedMixin**

`created_at = Column(None, DateTime(), table=None, default=ColumnDefault(<function creation date`

`deleted_at = Column(None, DateTime(), table=None)`

`updated_at = Column(None, DateTime(), table=None, onupdate=ColumnDefault(<function last modification date`

**SYSTEM = {'auditable': False, 'editable': False}**

SYSTEM properties are properties defined by the system and not supposed to be changed manually.

Subject classes (i.e. people, groups, etc.).

See ICOM-ics-v1.0 "Subject Branch".

TODO: I'm not a big fan of the "subject" name. Could be replaced by something else, like "people" or "principal" ?

**class User**(*password=None, \*\*kwargs*)

**query\_class**

alias of UserQuery

**authenticate**(*password: str*) → bool

**display\_value**(*field\_name, value=<object object>*)

Return display value for fields having 'choices' mapping (stored value.

-> human readable value). For other fields it will simply return field value.

*display\_value* should be used instead of directly getting field value.

If *value* is provided, it is "translated" to a human-readable value. This is useful for obtaining a human readable label from a raw value.

**follow**(*followee*)

**is\_admin\_of**(*group*)

**is\_following**(*other*)

**is\_member\_of**(*group*)

**join**(*group*)

**leave**(*group*)

**set\_password**(*password: str*) → None

Encrypts and sets password.

**unfollow**(*followee*)

**can\_login**

**created\_at**

**deleted\_at**

**email**

**entity\_type** = 'abilian.core.models.subjects.User'

**first\_name**

**followees**

**followers**

groups  
id  
property is\_online  
last\_active  
last\_name  
locale  
property name  
password  
photo  
preferences  
property short\_name  
timezone  
updated\_at

**class** Group(\*\*kwargs)

**display\_value**(*field\_name*, *value*=<object object>)

Return display value for fields having 'choices' mapping (stored value.  
-> human readable value). For other fields it will simply return field value.

*display\_value* should be used instead of directly getting field value.

If *value* is provided, it is "translated" to a human-readable value. This is useful for obtaining a human readable label from a raw value.

admins  
created\_at  
deleted\_at  
description  
entity\_type = 'abilian.core.models.subjects.Group'  
id  
members  
members\_count  
name  
photo  
public  
updated\_at

**class Principal**

A principal is either a User or a Group.

**has\_role**(*role*, *context=None*)

**class ClearPasswordStrategy**

Don't encrypt at all.

This strategy should not ever be used elsewhere than in tests. It's useful in tests since a hash like bcrypt is designed to be slow.

**authenticate**(*user*, *password*)

Predicate to tell whether password match user's or not.

**process**(*user*, *password*)

Return a string to be stored as user password.

**property name**

Strategy name.

**gen\_random\_password**(*length=15*)

**create\_root\_user**() → abilian.core.models.subjects.User

**class OwnedMixin**(*\*args*, *\*\*kwargs*)

**creator** = <RelationshipProperty at 0x7f6eeb765548; no key>

**creator\_id** = Column(None, NullType(), ForeignKey('user.id'), table=None)

**property creator\_name**

**owner** = <RelationshipProperty at 0x7f6eeb765f48; no key>

**owner\_id** = Column(None, NullType(), ForeignKey('user.id'), table=None)

**property owner\_name**

Blob.

References to files stored in a on-disk repository

**class Blob**(*value=None*, *\*args*, *\*\*kwargs*)

Model for storing large file content.

Files are stored on-disk, named after their uuid. Repository is located in instance folder/data/files.

**display\_value**(*field\_name*, *value=<object object>*)

Return display value for fields having 'choices' mapping (stored value.

-> human readable value). For other fields it will simply return field value.

*display\_value* should be used instead of directly getting field value.

If *value* is provided, it is "translated" to a human-readable value. This is useful for obtaining a human readable label from a raw value.

**property file**

Return `pathlib.Path` object used for storing value.

**id****property md5**

Return md5 from meta, or compute it if absent.

**meta****property size**

Return size in bytes of value.

**uuid****property value**

Binary value content.

**class SupportTagging****class Tag(\*\*kwargs)**

Tags are text labels that can be attached to entities.Entity.

They are namespaced, so that independent group of tags can be defined in the application. The default namespace is “default”.

**display\_value**(*field\_name*, *value*=<object object>)

Return display value for fields having ‘choices’ mapping (stored value.

-> human readable value). For other fields it will simply return field value.

*display\_value* should be used instead of directly getting field value.

If *value* is provided, it is “translated” to a human-readable value. This is useful for obtaining a human readable label from a raw value.

**entities**

entities attached to this tag

**id****label**

Label visible to the user

**ns**

namespace

**register**(*cls*)

Register an Entity as a taggable class.

Can be used as a class decorator:

```

@tag.register
class MyContent(Entity):
    ...

```

**supports\_tagging**(*obj*)



**Parameters** `obj` – a class or instance

`TAGS_ATTR = '__tags__'`

backref attribute on tagged elements

**class** `Comment(*args, **kwargs)`

A Comment related to an Entity.

`SLUG_SEPARATOR = '-'`

**body**

comment's main content

**created\_at**

**creator**

**creator\_id**

**deleted\_at**

**entity**

Commented entity

**entity\_id**

`entity_type = 'abilian.core.models.comment.Comment'`

**property history**

**id**

**meta**

**name**

**owner**

**owner\_id**

**slug**

**updated\_at**

**class** `Commentable`

`for_entity(obj, check_commentable=False)`

Return comments on an entity.

`is_commentable(obj_or_class: Any) → bool`

**Parameters** `obj_or_class` – a class or instance

`register(cls: type) → type`

Register an Entity as a commentable class.

Can be used as a class decorator:

```
@comment.register
class MyContent(Entity):
    ...
```

**ATTRIBUTE = '\_\_comments\_\_'**  
name of backref on target Entity object

**class Attachment(\*args, \*\*kwargs)**  
An Attachment owned by an Entity.

**SLUG\_SEPARATOR = '-'**

**blob**  
file. Stored in a Blob

**blob\_id**

**created\_at**

**creator**

**creator\_id**

**deleted\_at**

**description**

**entity**  
owning entity

**entity\_id**

**entity\_type = 'abilian.core.models.attachment.Attachment'**

**id**

**meta**

**name**

**owner**

**owner\_id**

**slug**

**updated\_at**

**class SupportAttachment**

**for\_entity(obj, check\_support\_attachments=False)**  
Return attachments on an entity.

**register(cls)**  
Register an Entity as a attachmentable class.

Can be used as a class decorator:

```
@attachment.register
class MyContent(Entity):
    ....
```

**set\_attachment\_name**(*mapper, connection, target*)

**supports\_attachments**(*obj*)

**Parameters** *obj* – a class or instance

**Returns** True is *obj* supports attachments.

**ATTRIBUTE** = '\_\_attachments\_\_'

name of backref on target Entity object

**class** BaseMixin

**to\_dict**()

**to\_json**()

**property** column\_names

## 5.2.7 Module `abilian.core.util`

Various tools that don't belong some place specific.

**class** BasePresenter(*model*)

A presenter wraps a model and adds specific (often, web-centric) accessors.

Subclass to make it useful. Presenters are immutable.

**classmethod** wrap\_collection(*models*)

**class** Pagination(*page, per\_page, total\_count*)

**iter\_pages**(*left\_edge=2, left\_current=2, right\_current=5, right\_edge=2*)

**property** has\_next

**property** has\_prev

**property** next

**property** pages

**property** prev

**class** memoized(*func*)

Decorator that caches a function's return value each time it is called.

If called later with the same arguments, the cached value is returned (not reevaluated).

**class timer**(*f*)  
Decorator that measures the time it takes to run a function.

**encode\_string**(*string*)  
Encode a string to bytes, if it isn't already.  
**Parameters** **string** – The string to encode

**fqn**(*cls: type*) → str  
Fully Qualified Class Name.

**friendly\_fqn**(*cls\_or\_cls\_name: Union[type, str]*) → str  
Friendly name of fully qualified class name.  
**Parameters** **cls\_or\_cls\_name** – a string or a class

**get\_params**(*names*)  
Return a dictionary with params from request.  
TODO: I think we don't use it anymore and it should be removed before someone gets hurt.

**local\_dt**(*dt: datetime.datetime*) → datetime.datetime  
Return an aware datetime in system timezone, from a naive or aware datetime.  
Naive datetime are assumed to be in UTC TZ.

**md5**(*data*)  
md5 function, as in flask-security.

**noproxy**(*obj: Any*)  
Unwrap obj from werkzeug.local.LocalProxy if needed.  
This is required if one want to test *isinstance(obj, SomeClass)*.

**slugify**(*value, separator='-'*)  
Slugify an Unicode string, to make it URL friendly.

**unwrap**(*obj: Any*)  
Unwrap obj from werkzeug.local.LocalProxy if needed.  
This is required if one want to test *isinstance(obj, SomeClass)*.

**utc\_dt**(*dt: datetime.datetime*) → datetime.datetime  
Set UTC timezone on a datetime object.  
A naive datetime is assumed to be in UTC TZ.

**utcnow**() → datetime.datetime  
Return a new aware datetime with current date and time, in UTC TZ.

## 5.3 Package `abilian.services`

### 5.3.1 Module `abilian.services.base`

**exception `ServiceNotRegistered`**

**class `Service`**(*app: Optional[Any] = None*)

Base class for services.

**AppStateClass**

State class to use for this Service

alias of `ServiceState`

**static `if_running`**(*meth: Callable*) → Callable

Decorator for service methods that must be ran only if service is in running state.

**`init_app`**(*app: Application*) → None

**`start`**(*ignore\_state: bool = False*) → None

Starts the service.

**`stop`**(*ignore\_state: bool = False*) → None

Stops the service.

**property `app_state`**

Current service state in current application.

:raise:RuntimeError if working outside application context.

**`name`** = ''

service name in `Application.extensions / Application.services`

**property `running`**

**Returns** *False* if working outside application context, if service is not registered on current application, or if service is halted for current application.

**class `ServiceState`**(*service: abilian.services.base.Service, running: bool = False*)

Service state stored in `Application.extensions`.

**`running`** = `False`

**`service`** = `None`

reference to `Service` instance

### 5.3.2 Module `abilian.services.activity`

**class `ActivityEntry`**(*\*\*kwargs*)

Main table for all activities.

**display\_value**(*field\_name*, *value*=<object object>)

Return display value for fields having 'choices' mapping (stored value.

-> human readable value). For other fields it will simply return field value.

*display\_value* should be used instead of directly getting field value.

If *value* is provided, it is "translated" to a human-readable value. This is useful for obtaining a human readable label from a raw value.

**actor**

**actor\_id**

**happened\_at**

**id**

**object**

**object\_id**

**object\_type**

**target**

**target\_id**

**target\_type**

**verb**

**class** ActivityService(*app*: Optional[Any] = None)

**static entries\_for\_actor**(*actor*: *abilian.core.models.subjects.User*,  
*limit*: *int* = 50) →  
List[abilian.services.activity.models.ActivityEntry]

**log\_activity**(*sender*: None, *actor*: *abilian.core.models.subjects.User*, *verb*: *str*,  
*object*: Any, *target*: Optional[*abilian.core.entities.Entity*] = None)  
→ None

**start**(*ignore\_state*: *bool* = False) → None  
Starts the service.

**stop**(*ignore\_state*: *bool* = False) → None  
Stops the service.

**name** = 'activity'

### 5.3.3 Module *abilian.services.audit*

### 5.3.4 Module *abilian.services.conversion*

Conversion service.

Hardcoded to manage only conversion to PDF, to text and to image series.

Includes result caching (on filesystem).

Assumes poppler-utils and LibreOffice are installed.

TODO: rename Converter into ConversionService ?

**exception ConversionError**

**exception HandlerNotFound**

**class Converter**

**clear()** → None

**get\_image**(*digest, blob, mime\_type, index, size=500*)

Return an image for the given content, only if it already exists in the image cache.

**get\_metadata**(*digest, content, mime\_type*)

Get a dictionary representing the metadata embedded in the given content.

**has\_image**(*digest, mime\_type, index, size=500*)

Tell if there is a preview image.

**init\_app**(*app: flask.app.Flask*) → None

**init\_work\_dirs**(*cache\_dir: pathlib.Path, tmp\_dir: pathlib.Path*) → None

**register\_handler**(*handler: abilian.services.conversion.handlers.Handler*) → None

**to\_image**(*digest: str, blob: bytes, mime\_type: str, index: int, size: int = 500*) → bytes

Convert a file to a list of images.

Returns image at the given index.

**to\_pdf**(*digest: str, blob: bytes, mime\_type: str*) → bytes

**to\_text**(*digest: str, blob: bytes, mime\_type: str*) → str

Convert a file to plain text.

Useful for full-text indexing. Returns a Unicode string.

### 5.3.5 Module `abilian.services.image`

Provides tools (currently: only functions, not a real service) for image processing.

**resize**(*orig: Any, width: int, height: int, mode: str = 'fit'*) → bytes

**get\_format**(*img*)

**SCALE = 'scale'**

resize without retaining original proportions

**FIT = 'fit'**

resize image and retain original proportions. Image width and height will be at most specified width and height, respectively; At least width or height will be equal to specified width and height, respectively.

**CROP = 'crop'**

crop image and resize so that it matches specified width and height.

### 5.3.6 Module `abilian.services.indexing`

**service**

Index documents using whoosh.

**class WhooshIndexService(\*args, \*\*kwargs)**

Index documents using whoosh.

**AppStateClass**

alias of `IndexServiceState`

**after\_commit**(*session: sqlalchemy.orm.session.Session*) → None

Any db updates go through here.

We check if any of these models have `__searchable__` fields, indicating they need to be indexed. With these we update the whoosh index for the model. If no index exists, it will be created here; this could impose a penalty on the initial commit of a model.

**after\_flush**(*session: sqlalchemy.orm.session.Session, flush\_context: sqlalchemy.orm.unitofwork.UOWTransaction*) → None

**clear**() → None

Remove all content from indexes, and unregister all classes.

After `clear()` the service is stopped. It must be started again to create new indexes and register classes.

**clear\_update\_queue**(*app: Optional[flask.app.Flask] = None*) → None

**get\_document**(*obj: abilian.core.entities.Entity, adapter: abilian.services.indexing.adapter.SAAdapter = None*) → Dict[str, Any]

**index**(*name: str = 'default'*) → `whoosh.index.Index`

**index\_objects**(*objects, index='default'*)

Bulk index a list of objects.

**init\_app**(*app: Application*) → None

**init\_indexes**() → None

Create indexes for schemas.



**register\_class**(*cls*: *type*, *app\_state*: *abilian.services.indexing.service.IndexServiceState* = *None*) → *None*  
Register a model class.

**register\_classes**() → *None*

**register\_search\_filter**(*func*)

Register a function that returns a query used for filtering search results. This query is And'ed with other filters.

If no filtering should be performed the function must return *None*.

**register\_value\_provider**(*func*)

Register a function that may alter content of indexable document.

It is used in `get_document()` and called after adapter has built document.

The function must accept (*document*, *obj*) as arguments, and return the new document object.

**search**(*q*: *str*, *index*: *str* = *'default'*, *fields*: *Optional[Dict[str, float]]* = *None*, *Models*: *Tuple[Type[abilian.core.models.base.Model]]* = *()*, *object\_types*: *Tuple[str]* = *()*, *prefix*: *bool* = *True*, *facet\_by\_type*: *None* = *None*, *\*\*search\_args*)

Interface to search indexes.

#### Parameters

- **q** – unparsed search string.
- **index** – name of index to use for search.
- **fields** – optionnal mapping of field names -> boost factor?
- **Models** – list of Model classes to limit search on.
- **object\_types** – same as *Models*, but directly the model string.
- **prefix** – enable or disable search by prefix
- **facet\_by\_type** – if set, returns a dict of *object\_type*: results with a max of *limit* matches for each type.
- **search\_args** – any valid parameter for `whoosh.searching.Search.search()`. This includes *limit*, *groupedby* and *sortedby*

**search\_for\_class**(*query*, *cls*, *index*=*'default'*, *\*\*search\_args*)

**searchable\_object\_types**() → *List*

List of (*object\_types*, friendly name) present in the index.

**start**(*ignore\_state*: *bool* = *False*) → *None*

Starts the service.

**property default\_search\_fields**

Return default field names and boosts to be used for searching.

Can be configured with `SEARCH_DEFAULT_BOOSTS`

```
name = 'indexing'
```

**indexable\_role**(*principal: abilian.services.security.models.Role*) → str

Return a string suitable for query against *allowed\_roles\_and\_users* field.

**Parameters principal** – It can be Anonymous, Authenticated, or an instance of User or Group.

### 5.3.7 Module `abilian.services.security`

**service**

Security service, manages roles and permissions.

**class SecurityServiceState**(*service: abilian.services.base.Service, running: bool = False*)

**needs\_db\_flush = False**

True if security has changed

**use\_cache = True**

**Anonymous**

Defines role by name. Roles instances are unique by name.

**Parameters assignable** – true if this role is can be assigned through security service. Non-assignable roles are roles automatically given depending on context (ex: Anonymous/Authenticated).

**Authenticated**

Defines role by name. Roles instances are unique by name.

**Parameters assignable** – true if this role is can be assigned through security service. Non-assignable roles are roles automatically given depending on context (ex: Anonymous/Authenticated).

## 5.4 Package `abilian.web`

### 5.4.1 Module `abilian.web.attachments`

**register\_plugin**(*app: flask.app.Flask*) → None

### 5.4.2 Module `abilian.web.comments`

**register\_plugin**(*app: flask.app.Flask*) → None

### 5.4.3 Module `abilian.web.filters`

Add a few specific filters to Jinja2.

**abbrev**(*s: str, max\_size: int*) → str

**age**(*dt: Optional[datetime.datetime], now: Optional[datetime.datetime] = None, add\_direction: bool = True, date\_threshold: Optional[Any] = None*) → str

#### Parameters

- **dt** – datetime instance to format
- **now** – datetime instance to compare to *dt*
- **add\_direction** – if *True*, will add “in” or “ago” (example for *en* locale) to time difference *dt - now*, i.e “in 9 min.” or “9min. ago”
- **date\_threshold** – above threshold, will use a formatted date instead of elapsed time indication. Supported values: “day”.

**autoescape**(*filter\_func: Callable*) → Callable

Decorator to autoescape result from filters.

**babel2datepicker**(*pattern: babel.dates.DateTimePattern*) → str

Convert date format from babel (<http://babel.pocoo.org/docs/dates/#date-fields>) to a format understood by bootstrap-datepicker.

**bool2check**(*val, true='✓', false=''*)

Filter value as boolean and show check mark (✓) or nothing.

**date\_age**(*dt: Optional[datetime.datetime], now: Optional[datetime.datetime] = None*) → str

**date\_fmt**(*value, format='EE, d MMMM y'*)

@deprecated: use flask\_babel’s `dateformat` filter instead.

**datetimeparse**(*s*) → Optional[datetime.datetime]

Parse a string date time to a datetime object.

Suitable for dates serialized with `.isoformat()`

**Returns** None, or an aware datetime instance, tz=UTC.

**filesize**(*d: Union[int, str]*) → markupsafe.Markup

**init\_filters**(*env: jinja2.environment.Environment*) → None

**labelize**(*s: str*) → str

**linkify**(*s: str*) → markupsafe.Markup

**n12br**(*value: str*) → markupsafe.Markup

Replace newlines with `<br />`.

**obj\_to\_url**(*obj*)

Find url for obj using `url_for()`, return empty string is not found.

`url_for()` is also provided in jinja context, the filtering version is forgiving when *obj* has no default view set.

**paragraphs**(*value: str*) → str

Blank lines delimitates paragraphs.

**roughsize**(*size: int, above: int = 20, mod: int = 10*) → str

6 -> '6' 15 -> '15' 134 -> '130+'.

**to\_timestamp**(*dt*)

#### 5.4.4 Module `abilian.web.action`

**class ActionRegistry**

The Action registry.

This is a Flask extension which registers Action sets. Actions are grouped by category and are ordered by registering order.

From your application use the instanciated registry actions.

The registry is available in jinja2 templates as *actions*.

**actions**(*context: Optional[Any] = None*) → Dict[str, Any]

Return a mapping of category => actions list.

Actions are filtered according to `Action.available()`.

if *context* is None, then current action context is used (*context*).

**for\_category**(*category: str, context: Any = None*) → List[`abilian.web.action.Action`]

Returns actions list for this category in current application.

Actions are filtered according to `Action.available()`.

if *context* is None, then current action context is used (*context*)

**init\_app**(*app: flask.app.Flask*) → None

**installed**(*app: Optional[flask.app.Flask] = None*) → bool

Return *True* if the registry has been installed in current applications.

**register**(\**actions*) → None

Register *actions* in the current application. All *actions* must be an instance of `Action` or one of its subclasses.

If *overwrite* is *True*, then it is allowed to overwrite an existing action with same name and category; else *ValueError* is raised.

**property context**

Return action context (dict type).

Applications can modify it to suit their needs.

```
class Action(category: str, name: str, title: Union[flask_babel.speaklater.LazyString,
str] = "", description: str = "", icon: Union[str, abilian.web.action.Icon,
None] = None, url: Union[str, Callable] = "", endpoint: Op-
tional[abilian.web.action.Endpoint] = None, condition: Op-
tional[Callable] = None, status: Optional[Any] = None, template:
Optional[Any] = None, template_string: Optional[Any] = None,
button: Optional[Any] = None, css: Optional[Any] = None)
```

Action interface.

```
class Endpoint(name: str, *args, **kwargs)
```

```
    get_kwargs() → Dict[str, str]
```

Hook for subclasses.

The key and values in the returned dictionary can be safely changed without side effects on self.kwargs (provided you don't alter mutable values, like calling list.pop()).

```
    available(context: Dict[str, Any]) → bool
```

Determine if this actions is available in this *context*.

**Parameters context** – a dict whose content is left to application needs; if condition is a callable it receives *context* in parameter.

```
    get_render_args(**kwargs) → Dict[str, Any]
```

```
    pre_condition(context: Dict[str, Any]) → bool
```

Called by available() before checking condition.

Subclasses may override it to ease creating actions with repetitive check (for example: actions that apply on a given content type only).

```
    render(**kwargs) → markupsafe.Markup
```

```
    url(context: Dict[str, Any] = None) → str
```

```
    CSS_CLASS = 'action action-{category} action-{category}-{name}'
```

```
    property description
```

```
    property enabled
```

```
    property endpoint
```

```
    property icon
```

```
    property status
```

```
    template_string = '<a class="{ action.css_class }" href="{ url }">{% if action
```

```
    property title
```

```
class ActionDropDown(category, name, items=(), *args, **kwargs)
```

Renders as a button dropdown.

```
    template_string = '\n <div class="btn-group">\n <button type="button" class="{ act
```

```

class ActionGroup(category, name, items=(), *args, **kwargs)
    A group of single actions.
    get_render_args(**kwargs)
    template_string = '<div class="btn-group" role="group" aria-label="{{ action.name}}>'

class ActionGroupItem(category, name, divider=False, *args, **kwargs)

    divider = False
        if True, add a divider in dropdowns

class ButtonAction(category: str, name: str, submit_name: str = '__action',
                   btn_class: str = 'default', *args, **kwargs)

    btn_class = 'default'
    template_string = '<button type="submit" class="btn btn-{{ action.btn_class }}">'

class FAIcon(name: str = "")
    Renders markup for FontAwesome icons.
    template = <Template memory:7f6eeea2dfd0>

class DynamicIcon(endpoint: Union[str, Callable, None] = None, width: int = 12,
                  height: int = 12, css: str = "", size: Optional[int] = None, url_args:
                  Optional[Callable] = None, **fixed_url_args)

    get_url_args() → Dict[str, str]
    template = <Template memory:7f6eee9bd470>

class StaticIcon(filename: str, endpoint: str = 'static', width: int = 12, height: int =
                  12, css: str = "", size: Optional[int] = None)
    Renders markup for icon located in static folder served by endpoint.
    Default endpoint is application static folder.

class ModalActionMixin

    template_string = '<a class="{{ action.css_class }}" href="{{ url }}" data-toggle="...'

class Endpoint(name: str, *args, **kwargs)

    get_kwargs() → Dict[str, str]
        Hook for subclasses.

        The key and values in the returned dictionary can be safely changed with-
        out side effects on self.kwargs (provided you don't alter mutable values,
        like calling list.pop()).

class Glyphicon(name: str = "")
    Renders markup for bootstrap's glyphicons.
    template = <Template memory:7f6eeea59cf8>

```

**getset**(*f: Callable*) → property  
Shortcut for a custom getter/ standard setter.

Usage:

```
@getset
def my_property(self, value=None):
    if value is None:
        return getter_value
    set_value(value)
```

Default value for *value* should be any marker that helps distinguish between getter or setter mode. If None is not appropriate a good approach is to use a unique object instance:

```
MARK = object()
# test like this
if value is MARK:
    # getter mode
```

**ENABLED** = Status('enabled')  
default action status: show in UID, usable, not marked “current”

**ACTIVE** = Status('active')  
action is “active” or “current”. For example the current navigation item.

**DISABLED** = Status('disabled')  
action should be shown in a disabled state

## 5.4.5 Module `abilian.web.nav`

Navigation elements.

Abilian define these categories: *section*: Used for navigation elements relevant to site  
*section user*: User for element that should appear in user menu

```
class BreadcrumbItem(label: Union[flask_babel.speaklater.LazyString, str] = "", url:
    Union[str, abilian.web.action.Endpoint] = '#', icon: Op-
    tional[str] = None, description: Optional[Any] = None)
```

A breadcrumb element has at least a label or an icon.

**render**() → markupsafe.Markup

**description** = None  
Additional text, can be used as tooltip for example

**icon** = None  
Icon to use.

**label** = None  
Label shown to user. May be an i18n string instance

```
template_string = '{%- if url %}<a href="{ url }">{%- endif %}{%- if item.icon %}
```

**property url**

**class NavGroup**(*category: str, name: str, items: Tuple[()] = (), \*args, \*\*kwargs*)

A navigation group renders a list of items.

**append**(*item: abilian.web.nav.NavItem*) → None

**get\_render\_args**(*\*\*kwargs*) → Dict[str, Any]

**insert**(*pos: int, item: abilian.web.nav.NavItem*) → None

**property status**

**template\_string** = '\n <ul class="nav navbar-nav {{ action.css\_class }}">\n <li clas

**class NavItem**(*category: str, name: str, divider: bool = False, \*args, \*\*kwargs*)

A single navigation item.

**divider** = False

**property path**

**property status**

## 5.4.6 Module `abilian.web.forms`

## 5.4.7 Module `abilian.web.views`

**class View**

Base class to use for all class based views.

The view instance is accessible in `g` and is set in actions context.

**classmethod as\_view**(*name: str, \*class\_args, \*\*class\_kwargs*) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

**dispatch\_request**(*\*args, \*\*kwargs*) → str

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

**prepare\_args**(*args, kwargs*)

If view arguments need to be prepared it can be done here.

A typical use case is to take an identifier, convert it to an object instance and maybe store it on view instance and/or replace identifier by object in arguments.

**redirect**(*url*)

Shortcut all call stack and return response.



usage: *self.response(url\_for(...))*

### class JSONView

Base view for JSON GET.

Renders as JSON when requested by Ajax, renders as HTML when requested from browser.

**data**(\*args, \*\*kwargs) → Dict

This method should return data to be serialized using JSON.

**get**(\*args, \*\*kwargs) → str

**prepare\_args**(args: Tuple, kwargs: Dict[Any, Any]) → Tuple[Tuple, Dict[Any, Any]]

If view arguments need to be prepared it can be done here.

A typical use case is to take an identifier, convert it to an object instance and maybe store it on view instance and/or replace identifier by object in arguments.

**methods** = {'GET'}

### class Registry(\*args, \*\*kwargs)

Registry for default (canonical) views for entities.

There is one registry per application instance.

**register**(entity: Union[abilian.core.entities.Entity, Type[abilian.core.entities.Entity]], url\_func: Callable) → None

Associate a *url\_func* with entity's type.

**Param:entity** an abilian.core.extensions.db.Model class or instance.

**Param:url\_func** any callable that accepts an entity instance and return an url for it.

**url\_for**(entity: Union[flask\_sqlalchemy.Model, whoosh.searching.Hit, Dict, None] = None, object\_type: Optional[str] = None, object\_id: Optional[int] = None, \*\*kwargs) → str

Return canonical view URL for given entity instance.

If no view has been registered the registry will try to find an endpoint named with entity's class lowercased followed by '.view' and that accepts *object\_id=entity.id* to generates an url.

#### Parameters

- **entity** – a instance of a subclass of abilian.core.extensions.db.Model, whoosh.searching.Hit or dict
- **object\_id** – if *entity* is not an instance, this parameter must be set to target id. This is usefull when you know the type and id of an object but don't want to retrieve it from DB.

**Raises KeyError** – if no view can be found for the given entity.

```
class default_view(app_or_blueprint: Union[Application,
    flask.blueprints.Blueprint], entity: abilian.core.entities.Entity,
    id_attr: str = 'object_id', endpoint: Optional[Any] = None,
    kw_func: Optional[Any] = None)
```

Decorator to register a view as default view for given entity class.

#### Parameters

- **id\_attr** – url parameter name for object id.
- **endpoint** – endpoint to use, defaults to view function’s name.
- **kw\_func** – function to process keywords to be passed to url\_for. Useful for additional keywords. This function receives: kw, obj, obj\_type, obj\_id, \*\*kwargs. It must return kw.

```
class BaseObjectView(Model=None, pk=None, base_template=None, *args,
    **kwargs)
```

Base class common to all database objects views.

#### breadcrumb()

Return nav.BreadcrumbItem instance for this object.

This method may return a list of BreadcrumbItem instances. Return *None* if nothing.

#### get(\*args, \*\*kwargs)

#### init\_object(args, kwargs)

This method is responsible for setting obj.

It is called during prepare\_args().

#### prepare\_args(args, kwargs)

If view arguments need to be prepared it can be done here.

A typical use case is to take an identifier, convert it to an object instance and maybe store it on view instance and/or replace identifier by object in arguments.

#### Model = None

Model class

#### base\_template = 'base.html'

default templates inherit from “base\_template”. This allows to use generic templates with a custom base

#### methods = {'GET'}

#### obj = None

object instance for this view

#### object\_id = None

object id

#### pk = 'object\_id'

primary key name to look for in url arguments

**template = None**

template to render

**property template\_kwargs**

Get template render arguments.

You may override *base\_template* for instance. Only *view* cannot be overridden.

**title = None**

form title

**class ObjectView**(*Model=None, pk=None, Form=None, template=None, \*args, \*\*kwargs*)

View objects.

**get\_form\_kwargs()**

**index\_url()**

**prepare\_args**(*args, kwargs*)

form is initialized here. See also *View.prepare\_args()*.

**redirect\_to\_index()**

**Form = None**

View form class. Form object used to show objects fields

**form = None**

form instance for this view

**methods = {'GET'}**

**permission = Permission('read')**

required permission. Must be an instance of *abilian.services.security.Permission*

**template = 'default/object\_view.html'**

html template

**property template\_kwargs**

Provides form to templates.

**class ObjectEdit**(*Model=None, pk=None, Form=None, template=None, view\_endpoint=None, message\_success=None, \*args, \*\*kwargs*)

Edit object.

**after\_populate\_obj()**

Called after *self.obj* values have been updated, and *self.obj* attached to an ORM session.

**before\_populate\_obj()**

This method is called after form has been validated and before calling *form.populate\_obj()*.

Sometimes one may want to remove a field from the form because it's nonsense to store it on edited object, and use it in a specific manner, for example:

```
image = form.image
del form.image
store_image(image)
```

**cancel()**

**commit\_success()**

Called after object has been successfully saved to database.

**edit(redirect\_to=None)**

**form\_csrf\_invalid()**

Called when a form doesn't validate *only* because of csrf token expiration.

This works only if form is an instance of flask\_wtf.form.SecureForm. Else default CSRF protection (before request) will take place.

It must return a valid Flask.Response instance. By default it returns to edit form screen with an informative message.

**form\_invalid()**

When a form doesn't validate this method is called.

It may return a Flask.Response instance, to handle specific errors in custom screens.

Else the edit form screen is returned with error(s) highlighted.

This method is useful for detecting edition conflict using hidden fields and show a specific screen to help resolve the conflict.

**form\_valid(redirect\_to=None)**

Save object.

Called when form is validated.

**Parameters redirect\_to** – real url (created with url\_for) to redirect to, instead of the view by default.

**get\_form\_buttons(\*args, \*\*kwargs)**

**handle\_action(action)**

**handle\_commit\_exception(exc)**

Hook point to handle exception that may happen during commit.

It is the responsibility of this method to perform a rollback if it is required for handling *exc*. If the method does not handle *exc* it should do nothing and return None.

**Returns**

- a valid Response if exception is handled.
- *None* if exception is not handled. Default handling happens.

**message\_success()**

**post**(\*args, \*\*kwargs)

**prepare\_args**(args, kwargs)  
form is initialized here. See also `View.prepare_args()`.

**put**()

**redirect\_to\_view**()

**send\_activity**()

**validate**()

**view\_url**()

**action = None**  
action name from form data

**property activity\_target**  
Return *target* to use when creating activity.

**activity\_verb = 'update'**  
verb used to describe activity

**button = None**  
button clicked, corresponding to action.

**property buttons**

**data = None**  
submitted form data

**decorators = (<function support\_graceful\_failure>,)**

**methods = {'GET', 'POST', 'PUT'}**

**permission = Permission('write')**

**template = 'default/object\_edit.html'**

**view\_endpoint = None**

**class ObjectCreate**(chain\_create\_allowed=None, \*args, \*\*kwargs)  
Create a new object.

**breadcrumb**()  
Return `nav.BreadcrumbItem` instance for this object.  
  
This method may return a list of `BreadcrumbItem` instances. Return *None* if nothing.

**cancel**()

**chain\_create**()

**create**()

**get\_form\_buttons**(\*args, \*\*kwargs)

**get\_form\_kwargs**()

**init\_object**(*args, kwargs*)  
This method is responsible for setting obj.  
It is called during `prepare_args()`.

**prepare\_args**(*args, kwargs*)  
form is initialized here. See also `View.prepare_args()`.

**activity\_verb** = 'post'

**chain\_create\_allowed** = False  
set to *True* to show 'Save and add new' button

**methods** = {'GET', 'POST', 'PUT'}

**permission** = `Permission('create')`

**class ObjectDelete**(*Model=None, pk=None, Form=None, template=None, view\_endpoint=None, message\_success=None, \*args, \*\*kwargs*)  
Delete object.

Supports the DELETE verb.

**delete**()

**get\_form\_buttons**(\**args, \*\*kwargs*)

**init\_object**(*args, kwargs*)  
This method is responsible for setting obj.  
It is called during `prepare_args()`.

**activity\_verb** = 'delete'

**methods** = ['POST']

**permission** = `Permission('delete')`

**class JSONBaseSearch**(\**args, \*\*kwargs*)

**data**(*q, \*args, \*\*kwargs*) → Dict  
This method should return data to be serialized using JSON.

**get\_item**(*obj*)  
Return a result item.

**Parameters** *obj* – Instance object

**Returns** a dictionary with at least *id* and *text* values

**get\_results**(*q, \*args, \*\*kwargs*)

**prepare\_args**(*args, kwargs*)  
If view arguments need to be prepared it can be done here.

A typical use case is to take an identifier, convert it to an object instance and maybe store it on view instance and/or replace identifier by object in arguments.

```

Model = None
methods = {'GET'}
minimum_input_length = 2
class JSONModelSearch(*args, **kwargs)
    Base class for json sqlalchemy model search.
    As used by select2 widgets for example.
    filter(query, q, **kwargs)
    get_item(obj)
        Return a result item.
        Parameters obj – Instance object
        Returns a dictionary with at least id and text values
    get_label(obj)
    get_results(q, *args, **kwargs)
    options(query)
    order_by(query)
    methods = {'GET', 'OPTIONS'}
class JSONWhooshSearch(*args, **kwargs)
    Base class for JSON Whoosh search, as used by select2 widgets for example.
    get_item(hit)
        Return a result item.
        Parameters hit – Hit object from Whoosh
        Returns a dictionary with at least id and text values
    get_results(q, *args, **kwargs)
    methods = {'GET'}

```

#### 5.4.8 Module `abilian.web.frontend`

Front-end for a CRM app.

This should eventually allow implementing very custom CRM-style application.

```

class BaseEntityView(module: abilian.web.frontend.Module, *args, **kwargs)

    breadcrumb()
    check_access()
    init_object(args, kwargs)
    prepare_args(args, kwargs)

```

```

    redirect_to_index()
    property can_create
    property can_delete
    property can_edit
    pk = 'entity_id'
    property single_view
class CRUDApp(app: abilian.app.Application, modules: None = None, name: None =
                None)

    add_module(module: abilian.web.frontend.Module) → None
    create_blueprint(module: abilian.web.frontend.Module) →
                        flask.blueprints.Blueprint
    get_module(module_id)

class DefaultRelatedView(label, attr, column_names, options=None,
                        show_empty=False)
    Default view used by Module for items directly related to entity.

    render(entity)
        Return a dict with keys 'label', 'attr_name', 'rendered', 'size', 'show_empty',
        'default_collapsed'.

class EntityCreate(module: abilian.web.frontend.Module, *args, **kwargs)

    breadcrumb()
        Return nav.BreadcrumbItem instance for this object.

        This method may return a list of BreadcrumbItem instances. Return None if
        nothing.

    check_access()

    prepare_args(args, kwargs)
        form is initialized here. See also View.prepare_args().

    methods = {'GET', 'POST', 'PUT'}

    mode = 'create'

    template = 'default/single_view.html'

    property template_kwargs
        Provides form to templates.

class EntityDelete(module: abilian.web.frontend.Module, *args, **kwargs)

    methods = {'DELETE', 'GET', 'POST', 'PUT'}

```



```

class EntityEdit(module: abilian.web.frontend.Module, *args, **kwargs)

    methods = {'GET', 'POST', 'PUT'}
    mode = 'edit'
    template = 'default/single_view.html'
    property template_kwargs
        Provides form to templates.

class EntityView(module: abilian.web.frontend.Module, *args, **kwargs)

    methods = {'GET'}
    mode = 'view'
    property object_actions
    template = 'default/single_view.html'
    property template_kwargs
        Provides form to templates.

class ListJson(module: abilian.web.frontend.Module, *args, **kwargs)
    JSON endpoint, for AJAX-backed table views.

    data(*args, **kwargs) → Dict
        This method should return data to be serialized using JSON.

    methods = {'GET'}

class Module

    create_cls
        alias of EntityCreate

    delete_cls
        alias of EntityDelete

    edit_cls
        alias of EntityEdit

    json_search_cls
        alias of abilian.web.views.object.JSONWhooshSearch

    view_cls
        alias of EntityView

    create_blueprint(crud_app: abilian.web.frontend.CRUDApp) →
        flask.blueprints.Blueprint
        Create a Flask blueprint for this module.

    get_component(name)

    get_grouped_actions() → collections.OrderedDict

```

**init\_related\_views()** → None

**is\_current()**

**list\_json2()**

Other JSON endpoint, this time used for filling select boxes dynamically.

You can write your own search method in `list_json2_query_all`, that returns a list of results (not json).

**list\_json2\_query\_all(q)**

Implements the search query for the `list_json2` endpoint.

May be re-defined by a Module subclass in order to customize the search results.

- Return: a list of results (not json) with an 'id' and a 'text' (that will be displayed in the `select2`).

**list\_query**(*request*: *flask.wrappers.Request*) → *abilian.core.entities.EntityQuery*

Return a filtered query based on request args, for listings.

Like *query*, but subclasses can modify it to remove costly joined loads for example.

**list\_view()** → str

**ordered\_query**(*request*: *flask.wrappers.Request*, *query*: *Optional[abilian.core.entities.EntityQuery]* = None) → *abilian.core.entities.EntityQuery*

Order query according to request args.

If *query* is None, the query is generated according to request args with `self.query(request)`

**query**(*request*: *flask.wrappers.Request*)

Return filtered query based on request args.

**register\_actions()** → None

**JSON2\_SEARCH\_LENGTH** = 50

**property action\_category**

**property base\_query**

Return a query instance for `managed_class`.

**base\_template** = 'base.html'

**blueprint** = None

**components** = ()

**edit\_form\_class** = None

**endpoint** = None

**id** = None

**label** = None

**list\_view\_columns** = []

**property listing\_query**

Like *read\_query*, but can be made lightweight with only columns and joins of interest.

*read\_query* can be used with exports for example, with lot more columns (generally it means more joins).

**managed\_class** = None

**name** = None

**property read\_query**

Return a query instance for *managed\_class* filtering on *READ* permission.

**related\_views** = []

**search\_criteria** = (<TextSearchCriterion name=name>,)

**single\_view** = None

**static\_folder** = None

**tableview\_options** = {}

**url** = None

**view\_form\_class** = None

**view\_new\_save\_and\_add** = False

**view\_options** = None

**view\_template** = None

```
class ModuleAction(module: abilian.web.frontend.Module, group: str, name: str,  
                  *args, **kwargs)
```

Base action class for Module actions.

Basic condition is simple: category must match the string *'module:{module.endpoint}'*

```
pre_condition(context: Dict[str, Module]) → bool
```

Called by *available()* before checking condition.

Subclasses may override it to ease creating actions with repetitive check (for example: actions that apply on a given content type only).

```
class ModuleActionDropDown(module: abilian.web.frontend.Module, group: str,  
                           name: str, *args, **kwargs)
```

```
    template_string = '\n <div class="btn-group">\n <button type="button" class="{act
```

```
class ModuleActionGroup(module: abilian.web.frontend.Module, group: str, name:  
                       str, *args, **kwargs)
```

```

    template_string = '<div class="btn-group" role="group" aria-label="{ { action.name}}'
class ModuleActionGroupItem(module: abilian.web.frontend.Module, group: str,
                             name: str, *args, **kwargs)
class ModuleComponent(name=None)
    A component that provide new functions for a Module
    get_actions()
    init(*args, **kwargs)
        Implements this in components.
    init_module(module)
    name = None
class ModuleMeta(classname: str, bases: Tuple, fields: Dict[str, Any])
    Module metaclass.
    Does some precalculations (like getting list of view methods from the class) to
    avoid calculating them for each view class instance.
class ModuleView(module: abilian.web.frontend.Module, *args, **kwargs)
    Mixin for module base views.
    Provide module.
    module = None
        Module instance
class RelatedView
    A base class for related views.
    render(entity)
        Return a dict with keys 'label', 'attr_name', 'rendered', 'size', 'show_empty',
        'default_collapsed'.
add_to_recent_items(entity, type='ignored')
expose(url: str = '/', methods: Tuple[str] = ('GET', )) → Callable
    Use this decorator to expose views in your view classes.
    url Relative URL for the view methods Allowed HTTP methods. By default only
    GET is allowed.
labelize(s: str) → str
make_single_view(form: wtforms.form.Form, **options) → abil-
ian.web.forms.widgets.SingleView

```

#### 5.4.9 Module `abilian.web.tags`

```

class TagsExtension(app: flask.app.Flask)
    API for tags, installed as an application extension.
    It is also available in templates as tags.

```

**add**(*entity: abilian.core.entities.Entity, tag: abilian.core.models.tag.Tag = None, ns: Any = None, label: Any = None*) → *abilian.core.models.tag.Tag*

**entity\_default\_ns**(*entity*)

**entity\_tags**(*entity*)

**entity\_tags\_form**(*entity, ns=None*)  
Construct a form class with a field for tags in namespace *ns*.

**get**(*ns, label=None*)  
Return tags instances for the namespace *ns*, ordered by label.  
If *label* is not *None* the only one instance may be returned, or *None* if no tags exists for this label.

**get\_form\_context**(*obj, ns=None*)  
Return a dict: form instance, action button, submit url...  
Used by macro `m_tags_form(entity)`

**remove**(*entity, tag=None, ns=None, label=None*)

**supports\_tagging**(*entity*)

**tags\_from\_hit**(*tag\_ids*)  
**Parameters** *tag\_ids* – indexed ids of tags in hit result. Do not pass hit instances.  
**Returns** an iterable of Tag instances.

**class TagCriterion**(*\*args, \*\*kwargs*)  
Filter entities with selected tag(s).

**filter**(*query, module, request, searched\_text, \*args, \*\*kwargs*)

**get\_request\_values**(*request*)

**form\_default\_value** = ''

**property form\_filter\_args**

**property form\_filter\_type**

**property form\_unset\_value**

**property model**

**property valid\_tags**

#### 5.4.10 Module `abilian.web.util`

A few utility functions.

See <https://docs.djangoproject.com/en/dev/topics/http/shortcuts/> for more ideas of stuff to implement.

**get\_object\_or\_404**(*cls, \*args*)

Shorthand similar to Django's *get\_object\_or\_404*.

**send\_file\_from\_directory**(*filename, directory, app=None*)

Helper to add static rules, like in *abilian.app.app*.

Example use:

```
app.add_url_rule(
    app.static_url_path + '/abilian/<path:filename>',
    endpoint='abilian_static',
    view_func=partial(send_file_from_directory,
                      directory='/path/to/static/files/dir'))
```

**url\_for**(*obj: Any, \*\*kw*) → str

Polymorphic variant of Flask's *url\_for* function.

Behaves like the original function when the first argument is a string. When it's an object, it

## 5.5 Package *abilian.testing*

# CHANGELOG FOR ABILIAN CORE

## 6.1 v0.11.2 (2019-06-28)

- Add flake8-mypy. [Stefane Fermigier]
- Add type annotations. [Stefane Fermigier]
- Better variable naming. [Stefane Fermigier]
- Class BlobQuery is not needed. [Stefane Fermigier]
- Cleanup imports. [Stefane Fermigier]
- Couple of typing fixes. [Stefane Fermigier]
- Fix incomplete refactoring. [Stefane Fermigier]
- Format + typing. [Stefane Fermigier]
- Make more robust. [Stefane Fermigier]
- Py3k. [Stefane Fermigier]
- Refactor caching. [Stefane Fermigier]
- Refactor conversion service. [Stefane Fermigier]
- Refactor: extract variable. [Stefane Fermigier]
- Set up CI with Azure Pipelines. [Stefane Fermigier]
- Skip test when soffice not available. [Stefane Fermigier]
- Typing. [Stefane Fermigier]

## 6.2 v0.11.1 (2019-05-02)

- A couple of typing fixes. [Stefane Fermigier]
- Dont run flake8-mypy for now. [Stefane Fermigier]

### 6.3 0.11.0 (2019-04-15)

- Drop Python 2 support.
- Rewrite code to be Python 3 only.
- Various fixes.

### 6.4 0.10.34 (2019-01-17)

- Simplify indexing control DSL: `__indexation_params__` -> `__index_to__`.

### 6.5 0.10.34 (2019-01-17)

- Simplify indexing control DSL: `__indexation_params__` -> `__index_to__`.

### 6.6 0.10.32 (2019-01-02)

- Switched dependency management to poetry
- Py3k migration and fixes.

### 6.7 0.10.29 (2018-12-26)

- Cleanup, small fixes related to updated dependencies.

### 6.8 0.10.29 (2018-12-26)

- Cleanup, small fixes related to updated dependencies.

### 6.9 0.10.20 (2018-07-19)

- Clean up audit objects by removing null values on init

### 6.10 0.10.15 (2018-07-05)

- Unpin pillow, small cleanups.



## 6.11 0.10.14 (2018-06-11)

- pin wtforms because 2.2 breaks our tests

## 6.12 0.10.12 (2018-04-27)

- Fix for Flask 1.0

## 6.13 0.10.11 (2018-04-15)

- Fix install under pip 10

## 6.14 0.10.8 (2018-04-04)

- Refactor pytest fixtures. API has changed.

## 6.15 0.10.3 (2018-02-22)

- Cleanup JS

## 6.16 0.10.2 (2018-02-21)

- Refactor tests (use pytest fixtures)
- Refactor Application class

## 6.17 0.10.2 (2018-02-15)

- Fix Py3k compatibility.

## 6.18 0.10.0 (2018-02-12)

Breaking changes:

- Removed deprecated plugin loader
- Renamed *is\_support\_attachments* to *supports\_attachments*

Other:

- Refactoring tests to use pytest's function-based tests instead of unittest's class-based tests.

## 6.19 0.9.30 (2018-01-11)

- Don't depend on psycopg2, so you can use your favorite driver (ex: pg8000).

## 6.20 0.9.19-0.9.29

- Cleanup
- Bug fixes
- Python 3 compatibility
- Dependencies updates

## 6.21 0.9.18 (2017-10-06)

- Relax dependency constraint on Bleach to allow upgrade of other deps.

## 6.22 0.9.17 (2017-10-02)

- Cleanup
- Fix some warnings.

## 6.23 0.9.16 (2017-09-08)

- JS cleanup and linting
- Deps updates

## 6.24 0.9.15 (2017-09-04)

- Revert some buggy JS "clean up".
- Deps updates

## 6.25 0.9.12 (2017-08-28)

- Code clean up.

## 6.26 0.9.11 (2017-08-03)

- Workaround bug in Babel related to Python 3.

## 6.27 0.9.10 (2017-08-02)

- Cleanup and prepare for Python 3.

## 6.28 0.9.9 (2017-08-01)

- Cleanup and prepare for Python 3.
- Use headless libreoffice for conversion instead of unoconv.

## 6.29 0.9.3 (2017-07-03)

- Add “impersonate” admin panel.

## 6.30 0.9.3 (2017-06-30)

- Fix bug on *form\_valid*

## 6.31 0.7.24 (2017-01-10)

- Downgrade Ravenjs :(

## 6.32 0.7.21 (2017-01-09)

- Ravenjs update
- Update deps

### 6.33 0.7.10 (2016-08-30)

- Fix issue with raven-js logging

### 6.34 0.7.9 (2016-08-29)

- More robust reindex command.
- Pytest > 3.0 compat

### 6.35 0.7.8 (2016-08-04)

- Use *bcrypt* library instead of *py-bcrypt*.
- Work on Py3k compatibility (not done yet)
- Update dependencies.

### 6.36 0.7.7 (2016-07-13)

- Work on Py3k compatibility (not done yet)
- Remove unneeded dependencies.
- Update dependencies.
- Harder linting.

### 6.37 0.7.0 (2016-05-31)

- Made compatible with Flask 0.11, SQLAlchemy 1.0 and a few other recent releases.
- General cleanup.

### 6.38 0.6.5 (2016-05-10)

Workaround some regression by not generating less source map.

## 6.39 0.6.2 (2016-05-09)

- Fix import error.

## 6.40 0.6.1 (2016-05-09)

- Allow SQLAlchemy 0.9.x for now
- Allow application/x-pdf mime type.

## 6.41 0.6.0 (2016-04-29)

- Upgrade SQLAlchemy to 1.0+.
- Dump config in sysinfo admin panel

Cleanup:

- Upgrade deps
- Reformat code using Google style rule

## 6.42 0.5.3-0.5.6 (2016-03-17)

Features:

- dynamic row widget options to add controls at the bottom (23 hours ago)<yvon>

Fixes:

- fix datatable optionalcriterion filter (2 days ago)<yvon>
- fix jquery datable jqmigrate warning (2 days ago)<yvon>
- fix search criterion outerjoin (6 days ago)<yvon>
- textsearch criterion mysterious onclause fix (9 days ago)<yvon>

Cleanup:

- Upgrade deps
- Reformat code using Google style rule

## 6.43 0.5.2 (2016-02-16)

- Fix IPv6 / GeoIP issue

- Improve debug toolbar
- Improve dashboard
- Celery: expire task before next run scheduled

## 6.44 0.5.1 (2016-01-29)

- add security debug panel: shows permissions and roles assignments
- faster `query_with_permission()`
- Fix: user administration could remove non-assignable roles
- Subforms (Form used in `FormFields` / `ListFormFields` / etc) can filter their fields according to permission passed to top Form.

## 6.45 0.5.0 (2015-11-20)

- Editable comments
- Upgrade SQLAlchemy to 0.9
- Admin: add Tag panels

## 6.46 0.4.5 (2015-10-15)

### 6.46.1 Improvements and updates

- Breaking: minor schemas changes. Migrations needed for existing applications
- tags in 'default' namespace are indexed in document's text for full text search on tag label
- age filter has a new option to show full date when date is not today
- run command: add `-ssl` option
- admin: manage groups membership from user page
- updated requirements to ensure sane minimum versions
- Role based access control makes more permissions checks againsts roles and less simple role check

## 6.46.2 Fixes

- fixes for celery workers
- fix: check user has role on object with global role
- fix: check user has roles through group membership

## 6.47 0.4.4 (2015-08-07)

### 6.47.1 Design / UI

- Navbar is now non-fluid.

### 6.47.2 Updates

- Upgrade Jinja to 2.8 and Babel to 2.0

### 6.47.3 Fixes

- Fixed image cropping.

## 6.48 0.4.3 (2015-07-29)

Another release because there was a version number issue with the previous one.

## 6.49 0.4.2 (2015-07-29)

### 6.49.1 Bugfixes / cleanup

- Replace Scribe by CKEditor for better IE compatibility.
- Smaller bug fixes and code cleanups

## 6.50 0.4.1 (2015-07-21)

### 6.50.1 Bugfixes / cleanup

- permission: no-op when service not running

- JS fixes
- CSS fixes
- <https://github.com/mitsuhiko/flask/issues/1135>

## 6.51 0.4.0 (2015-07-15)

### 6.51.1 Features

- Object level permissions
- Add “meta” properties to entities
- Attached files to entities
- More flexible search filters
- Avatars
- Tag engine (alpha)

### 6.51.2 Fixes / cleanup

- JS: Update ravenjs, requirejs, bootbox, jquery, scribe

## 6.52 0.3.6 (2015-05-27)

### 6.52.1 Fixes

- security service: fix exception on has\_role()

## 6.53 0.3.5 (2015-05-27)

### 6.53.1 Features

- default user avatar is now a circle with their last name initial (#12)
- add PRIVATE\_SITE, app, blueprint and endpoint access controller registration
- Better handling of CSRF failures
- add dynamic row widget js
- js: add datatable advanced search



## 6.53.2 Fixes

- CSS (Bootstrap) fixes
- Permissions fixes

## 6.53.3 Updates

- Updated Bootstrap to 3.3.4
- Updated flask-login to 0.2.11
- Updated Sentry JS code to 1.1.18

## 6.54 0.3.4 (2015-04-14)

- updated Select2 to 3.5.2
- enhanced fields and widgets
- set default SQLALCHEMY\_POOL\_RECYCLE to 30 minutes
- Users admin panel: fix roles not set; fix all assignable roles not listed; fix cannot set password during user creation.

## 6.55 0.3.3 (2015-03-31)

### 6.55.1 Features

- Use ravenjs to monitor JS errors with Sentry
- Vocabularies

## 6.56 0.3.2 (2014-12-23)

- Minor bugfixes

## 6.57 0.3.1 (2014-12-23)

- Minor bugfixes

## 6.58 0.3.0 (2014-12-23)

### 6.58.1 Features

- Added a virus scanner.
- Changed the WYSIWYG editor to Scribe.
- Vocabularies

### 6.58.2 API changes

- Deprecated the @templated decorator (will be removed in 0.4.0).

### 6.58.3 Building, tests

- Build: Use pbr to simplify setup.py.
- Dependencies: moved deps to ./requirements.txt + cleanup / update.
- Testing: Tox and Travis config updates.
- Testing: Run tests under Vagrant.
- QA: Fixed many pyflakes warnings.

## 6.59 0.2.0 (2014-08-07)

- Too long to list.

## 6.60 0.1.4 (2014-03-27)

- refactored abilian.core.entities, abilian.core.subjects. New module abilian.core.models containing modules: base, subjects, owned.
- Fixed or cleaned up dependencies.
- Fixed setupwizard.
- added config value: BABEL\_ACCEPT\_LANGUAGES, to limit supported languages and change order during negotiation
- Switched CSS to LESS.
- Updated to Bootstrap 3.1.1

## 6.61 0.1.3 (2014-02-03)

- Update some dependencies
- Added login/logout via JSON api
- Added 'createuser' command

## 6.62 0.1.2 (2014-01-11)

- added jinja extension to collect JS snippets during page generation and put them at end of document ("deferred")
- added basic javascript to prevent double submission
- Added Flask-Migrate

## 6.63 0.1.1 (2013-12-26)

- Redesigned indexing:
  - single whoosh index for all objects
  - search results page do not need anymore to fetch actual object from database
  - index security information, used for filtering search results
  - Added "reindex" shell command

## 6.64 0.1 (2013-12-13)

- Initial release.



## CREDITS

### 7.1 Design, programming

Abilian development team: 2012-2013.

### 7.2 Art (images, icons)

See links for copyright and licences (usually Creative Commons).

- Background image(s) for login: Kevin Dooley <http://www.flickr.com/photos/pagedooley/>.
- File types icons: [http://www.splitbrain.org/projects/file\\_icons](http://www.splitbrain.org/projects/file_icons).
- Animal pictures: [http://en.wikipedia.org/wiki/Wikipedia:Featured\\_pictures/Animals#Animals](http://en.wikipedia.org/wiki/Wikipedia:Featured_pictures/Animals#Animals).
- Group pictures: [http://en.wikipedia.org/wiki/File:Estudiante\\_INTEC.jpg](http://en.wikipedia.org/wiki/File:Estudiante_INTEC.jpg), [http://en.wikipedia.org/wiki/File:Salesman\\_-beach\\_-\\_bikini-\\_sun-27Dec2008.jpg](http://en.wikipedia.org/wiki/File:Salesman_-beach_-_bikini-_sun-27Dec2008.jpg), [http://en.wikipedia.org/wiki/File:Cocacola-5cents-1900\\_edit1.jpg](http://en.wikipedia.org/wiki/File:Cocacola-5cents-1900_edit1.jpg).
- Group Icon (CC Attribution-Noncommercial-No Derivate 3.0): <http://www.iconarchive.com/show/soft-scraps-icons-by-deleket/User-Group-icon.html>.



## Part II

# INDICES AND TABLES

- genindex
- modindex
- search





# INDEX

## Symbols

- `_` (in module *abilian.i18n*), 17
  - `__annotations__` (Entity attribute), 20
  - `__auditable__` (Entity attribute), 20
  - `__default_permissions__` (Entity attribute), 20
  - `__editable__` (Entity attribute), 20
  - `__index_to__` (Entity attribute), 20
  - `__indexable__` (Entity attribute), 20
  - `__init__()` (Entity method), 19
  - `__mapper__` (Entity attribute), 20
  - `__mapper_args__` (Entity attribute), 21
  - `__module__` (Entity attribute), 21
  - `__permissions__` (Entity attribute), 21
  - `__searchable__` (Entity attribute), 21
  - `__table__` (Entity attribute), 21
  - `_l` (in module *abilian.i18n*), 17
  - `_n` (in module *abilian.i18n*), 17
- ## A
- `abbrev()` (in module *abilian.web.filters*), 47
  - abilian.app* (module), 15
  - abilian.core.entities* (module), 19
  - abilian.core.extensions* (module), 24
  - abilian.core.logging* (module), 24
  - abilian.core.models* (module), 39
  - abilian.core.models.attachment* (module), 38
  - abilian.core.models.base* (module), 32
  - abilian.core.models.blob* (module), 35
  - abilian.core.models.comment* (module), 37
  - abilian.core.models.owned* (module), 35
  - abilian.core.models.subjects* (module), 33
  - abilian.core.models.tag* (module), 36
  - abilian.core.signals* (module), 24
  - abilian.core.sqlalchemy* (module), 25
  - abilian.core.util* (module), 39
  - abilian.i18n* (module), 17
  - abilian.services.activity* (module), 41
  - abilian.services.audit* (module), 42
  - abilian.services.base* (module), 41
  - abilian.services.conversion* (module), 42
  - abilian.services.image* (module), 43
  - abilian.testing* (module), 66
  - abilian.web.action* (module), 48
  - abilian.web.attachments* (module), 46
  - abilian.web.comments* (module), 46
  - abilian.web.filters* (module), 47
  - abilian.web.forms* (module), 52
  - abilian.web.frontend* (module), 59
  - abilian.web.nav* (module), 51
  - abilian.web.tags* (module), 64
  - abilian.web.util* (module), 65
  - abilian.web.views* (module), 52
  - Action (class in *abilian.web.action*), 48
  - action (ObjectEdit attribute), 57
  - Action.Endpoint (class in *abilian.web.action*), 49
  - action\_category() (Module property), 62
  - ActionDropDown (class in *abilian.web.action*), 49
  - ActionGroup (class in *abilian.web.action*), 49
  - ActionGroupItem (class in *abilian.web.action*), 50
  - ActionRegistry (class in *abilian.web.action*), 48
  - actions() (ActionRegistry method), 48
  - ACTIVE (in module *abilian.web.action*), 51
  - activity (in module *abilian.core.signals*), 24
  - activity\_target() (ObjectEdit property), 57
  - activity\_verb (ObjectCreate attribute), 58
  - activity\_verb (ObjectDelete attribute), 58

activity\_verb (*ObjectEdit* attribute), 57  
 ActivityEntry (class in *abilian.services.activity*), 41  
 ActivityService (class in *abilian.services.activity*), 42  
 actor (*ActivityEntry* attribute), 42  
 actor\_id (*ActivityEntry* attribute), 42  
 add() (*TagsExtension* method), 64  
 add\_access\_controller() (*Application* method), 15  
 add\_module() (*CRUDApp* method), 60  
 add\_static\_url() (*Application* method), 15  
 add\_to\_recent\_items() (in module *abilian.web.frontend*), 64  
 add\_translations() (*Babel* method), 17  
 add\_url\_rule\_with\_role() (*Application* method), 16  
 admins (*Group* attribute), 34  
 after\_commit() (*WhooshIndexService* method), 44  
 after\_flush() (*WhooshIndexService* method), 44  
 after\_populate\_obj() (*ObjectEdit* method), 55  
 age() (in module *abilian.web.filters*), 47  
 all\_entity\_classes (in module *abilian.core.entities*), 23  
 Anonymous (in module *abilian.services.security*), 46  
 app\_state() (*Service* property), 41  
 append() (*MutationList* method), 28  
 append() (*NavGroup* method), 52  
 Application (class in *abilian.app*), 15  
 apply\_driver\_hacks() (*SQLAlchemy* method), 28  
 AppStateClass (*Service* attribute), 41  
 AppStateClass (*WhooshIndexService* attribute), 44  
 as\_view() (*View* class method), 52  
 Attachment (class in *abilian.core.models.attachment*), 38  
 ATTRIBUTE (in module *abilian.core.models.attachment*), 39  
 ATTRIBUTE (in module *abilian.core.models.comment*), 38  
 authenticate() (*ClearPasswordStrategy* method), 35  
 authenticate() (*User* method), 33  
 Authenticated (in module *abilian.services.security*), 46  
 auto\_slug() (*Entity* property), 21, 22  
 autoescape() (in module *abilian.web.filters*), 47  
 available() (*Action* method), 49  
**B**  
 Babel (class in *abilian.i18n*), 17  
 babel (in module *abilian.i18n*), 18, 19  
 babel2datepicker() (in module *abilian.web.filters*), 47  
 base\_query() (*Module* property), 62  
 base\_template (*BaseObjectView* attribute), 54  
 base\_template (*Module* attribute), 62  
 BaseEntityView (class in *abilian.web.frontend*), 59  
 BaseMixin (class in *abilian.core.models*), 39  
 BaseObjectView (class in *abilian.web.views*), 54  
 BasePresenter (class in *abilian.core.util*), 39  
 before\_populate\_obj() (*ObjectEdit* method), 55  
 blob (*Attachment* attribute), 38  
 Blob (class in *abilian.core.models.blob*), 35  
 blob\_id (*Attachment* attribute), 38  
 blueprint (*Module* attribute), 62  
 body (*Comment* attribute), 37  
 bool2check() (in module *abilian.web.filters*), 47  
 breadcrumb() (*BaseEntityView* method), 59  
 breadcrumb() (*BaseObjectView* method), 54  
 breadcrumb() (*EntityCreate* method), 60  
 breadcrumb() (*ObjectCreate* method), 57  
 BreadcrumbItem (class in *abilian.web.nav*), 51  
 btn\_class (*ButtonAction* attribute), 50  
 button (*ObjectEdit* attribute), 57  
 ButtonAction (class in *abilian.web.action*), 50  
 buttons() (*ObjectEdit* property), 57  
**C**  
 can\_create() (*BaseEntityView* property), 60  
 can\_delete() (*BaseEntityView* property), 60  
 can\_edit() (*BaseEntityView* property), 60  
 can\_login (*User* attribute), 33

cancel() (*ObjectCreate* method), 57  
cancel() (*ObjectEdit* method), 56  
celery\_app\_cls (*Application* attribute), 15  
chain\_create() (*ObjectCreate* method), 57  
chain\_create\_allowed (*ObjectCreate* attribute), 58  
check\_access() (*BaseEntityView* method), 59  
check\_access() (*EntityCreate* method), 60  
check\_instance\_folder() (*Application* method), 16  
clear() (*Converter* method), 43  
clear() (*MutationDict* method), 27  
clear() (*WhooshIndexService* method), 44  
clear\_update\_queue() (*WhooshIndexService* method), 44  
ClearPasswordStrategy (class in *abilian.core.models.subjects*), 35  
clone() (*Entity* method), 19, 22  
coerce() (*MutationDict* class method), 27  
coerce() (*MutationList* class method), 28  
column\_names() (*BaseMixin* property), 39  
Comment (class in *abilian.core.models.comment*), 37  
Commentable (class in *abilian.core.models.comment*), 37  
commit\_success() (*ObjectEdit* method), 56  
compare\_against\_backend() (*UUID* method), 30  
components (*Module* attribute), 62  
components\_registered (in module *abilian.core.signals*), 24  
configure() (*Application* method), 16  
context() (*ActionRegistry* property), 48  
ConversionError, 43  
Converter (class in *abilian.services.conversion*), 43  
copy() (*Info* method), 32  
create() (*ObjectCreate* method), 57  
create\_app() (in module *abilian.app*), 17  
create\_blueprint() (*CRUDApp* method), 60  
create\_blueprint() (*Module* method), 61  
create\_cls (*Module* attribute), 61  
create\_root\_user() (in module *abilian.core.models.subjects*), 35  
created\_at (*Attachment* attribute), 38  
created\_at (*Comment* attribute), 37  
created\_at (*Entity* attribute), 21, 23  
created\_at (*Group* attribute), 34  
created\_at (*TimestampedMixin* attribute), 32  
created\_at (*User* attribute), 33  
creator (*Attachment* attribute), 38  
creator (*Comment* attribute), 37  
creator (*Entity* attribute), 21, 23  
creator (*OwnedMixin* attribute), 35  
creator\_id (*Attachment* attribute), 38  
creator\_id (*Comment* attribute), 37  
creator\_id (*Entity* attribute), 21, 23  
creator\_id (*OwnedMixin* attribute), 35  
creator\_name() (*OwnedMixin* property), 35  
CROP (in module *abilian.services.image*), 44  
CRUDApp (class in *abilian.web.frontend*), 60  
CSS\_CLASS (*Action* attribute), 49

## D

data (*ObjectEdit* attribute), 57  
data() (*JSONBaseSearch* method), 58  
data() (*JSONView* method), 53  
data() (*ListJson* method), 61  
data\_dir (*Application* attribute), 16  
date\_age() (in module *abilian.web.filters*), 47  
date\_fmt() (in module *abilian.web.filters*), 47  
datetimeparse() (in module *abilian.web.filters*), 47  
decorators (*ObjectEdit* attribute), 57  
default\_config (*Application* attribute), 16  
default\_search\_fields() (*WhooshIndexService* property), 45  
default\_view (*Application* attribute), 16  
default\_view (class in *abilian.web.views*), 53  
DefaultRelatedView (class in *abilian.web.frontend*), 60  
delete() (*ObjectDelete* method), 58  
delete\_cls (*Module* attribute), 61  
deleted\_at (*Attachment* attribute), 38  
deleted\_at (*Comment* attribute), 37  
deleted\_at (*Entity* attribute), 21, 23  
deleted\_at (*Group* attribute), 34  
deleted\_at (*TimestampedMixin* attribute), 32  
deleted\_at (*User* attribute), 33

description (*Attachment attribute*), 38  
description (*BreadcrumbItem attribute*), 51  
description (*Group attribute*), 34  
description() (*Action property*), 49  
DISABLED (*in module abilian.web.action*), 51  
dispatch\_request() (*View method*), 52  
display\_value() (*ActivityEntry method*), 41  
display\_value() (*Blob method*), 35  
display\_value() (*Entity method*), 20, 22  
display\_value() (*Group method*), 34  
display\_value() (*Tag method*), 36  
display\_value() (*User method*), 33  
divider (*ActionGroupItem attribute*), 50  
divider (*NavItem attribute*), 52  
DynamicIcon (*class in abilian.web.action*), 50

**E**  
edit() (*ObjectEdit method*), 56  
edit\_cls (*Module attribute*), 61  
edit\_form\_class (*Module attribute*), 62  
email (*User attribute*), 33  
ENABLED (*in module abilian.web.action*), 51  
enabled() (*Action property*), 49  
encode\_string() (*in module abilian.core.util*), 40  
Endpoint (*class in abilian.web.action*), 50  
endpoint (*Module attribute*), 62  
endpoint() (*Action property*), 49  
entities (*Tag attribute*), 36  
entity (*Attachment attribute*), 38  
Entity (*class in abilian.core.entities*), 19, 21  
entity (*Comment attribute*), 37  
entity\_class() (*Entity property*), 21, 23  
entity\_default\_ns() (*TagsExtension method*), 65  
entity\_id (*Attachment attribute*), 38  
entity\_id (*Comment attribute*), 37  
entity\_tags() (*TagsExtension method*), 65  
entity\_tags\_form() (*TagsExtension method*), 65  
entity\_type (*Attachment attribute*), 38  
entity\_type (*Comment attribute*), 37  
entity\_type (*Entity attribute*), 21, 23  
entity\_type (*Group attribute*), 34  
entity\_type (*User attribute*), 33  
EntityCreate (*class in abilian.web.frontend*), 60

EntityDelete (*class in abilian.web.frontend*), 60  
EntityEdit (*class in abilian.web.frontend*), 60  
EntityQuery (*class in abilian.core.entities*), 23  
EntityView (*class in abilian.web.frontend*), 61  
entries\_for\_actor() (*ActivityService static method*), 42  
expose() (*in module abilian.web.frontend*), 64  
extend() (*MutationList method*), 28

**F**  
FAIcon (*class in abilian.web.action*), 50  
file() (*Blob property*), 35  
filesize() (*in module abilian.web.filters*), 47  
filter() (*JSONModelSearch method*), 59  
filter() (*TagCriterion method*), 65  
filter\_cols() (*in module abilian.core.sqlalchemy*), 32  
first\_name (*User attribute*), 33  
FIT (*in module abilian.services.image*), 43  
follow() (*User method*), 33  
followees (*User attribute*), 33  
followers (*User attribute*), 33  
for\_category() (*ActionRegistry method*), 48  
for\_entity() (*in module abilian.core.models.attachment*), 38  
for\_entity() (*in module abilian.core.models.comment*), 37  
Form (*ObjectView attribute*), 55  
form (*ObjectView attribute*), 55  
form\_csrf\_invalid() (*ObjectEdit method*), 56  
form\_default\_value (*TagCriterion attribute*), 65  
form\_filter\_args() (*TagCriterion property*), 65  
form\_filter\_type() (*TagCriterion property*), 65  
form\_invalid() (*ObjectEdit method*), 56  
form\_unset\_value() (*TagCriterion property*), 65  
form\_valid() (*ObjectEdit method*), 56

fqcn() (in module *abilian.core.util*), 40  
friendly\_fqcn() (in module *abilian.core.util*), 40

## G

gen\_random\_password() (in module *abilian.core.models.subjects*), 35  
get() (*BaseObjectView* method), 54  
get() (*JSONView* method), 53  
get() (*TagsExtension* method), 65  
get\_actions() (*ModuleComponent* method), 64  
get\_component() (*Module* method), 61  
get\_default\_locale() (in module *abilian.i18n*), 18  
get\_document() (*WhooshIndexService* method), 44  
get\_extension() (in module *abilian.core.extensions*), 24  
get\_form\_buttons() (*ObjectCreate* method), 57  
get\_form\_buttons() (*ObjectDelete* method), 58  
get\_form\_buttons() (*ObjectEdit* method), 56  
get\_form\_context() (*TagsExtension* method), 65  
get\_form\_kwargs() (*ObjectCreate* method), 57  
get\_form\_kwargs() (*ObjectView* method), 55  
get\_format() (in module *abilian.services.image*), 43  
get\_grouped\_actions() (*Module* method), 61  
get\_image() (*Converter* method), 43  
get\_item() (*JSONBaseSearch* method), 58  
get\_item() (*JSONModelSearch* method), 59  
get\_item() (*JSONWhooshSearch* method), 59  
get\_kwargs() (*Action.Endpoint* method), 49  
get\_kwargs() (*Endpoint* method), 50  
get\_label() (*JSONModelSearch* method), 59  
get\_metadata() (*Converter* method), 43  
get\_module() (*CRUDApp* method), 60  
get\_object\_or\_404() (in module *abilian.web.util*), 65

get\_params() (in module *abilian.core.util*), 40  
get\_render\_args() (*Action* method), 49  
get\_render\_args() (*ActionGroup* method), 50  
get\_render\_args() (*NavGroup* method), 52  
get\_request\_values() (*TagCriterion* method), 65  
get\_results() (*JSONBaseSearch* method), 58  
get\_results() (*JSONModelSearch* method), 59  
get\_results() (*JSONWhooshSearch* method), 59  
get\_url\_args() (*DynamicIcon* method), 50  
getset() (in module *abilian.web.action*), 50  
gettext() (in module *abilian.i18n*), 18  
Glyphicon (class in *abilian.web.action*), 50  
Group (class in *abilian.core.models.subjects*), 34  
groups (User attribute), 33

## H

handle\_action() (*ObjectEdit* method), 56  
handle\_commit\_exception() (*ObjectEdit* method), 56  
HandlerNotFound, 43  
happened\_at (*ActivityEntry* attribute), 42  
has\_image() (*Converter* method), 43  
has\_next() (*Pagination* property), 39  
has\_prev() (*Pagination* property), 39  
has\_role() (*Principal* method), 35  
history() (*Comment* property), 37

## I

icon (*BreadcrumbItem* attribute), 51  
icon() (*Action* property), 49  
id (*ActivityEntry* attribute), 42  
id (*Attachment* attribute), 38  
id (*Blob* attribute), 36  
id (*Comment* attribute), 37  
id (*Entity* attribute), 21, 23  
id (*Group* attribute), 34  
id (*IdMixin* attribute), 32  
id (*Module* attribute), 62  
id (*Tag* attribute), 36  
id (*User* attribute), 34  
IdMixin (class in *abilian.core.models.base*), 32

if\_running() (*Service static method*), 41  
 impl (JSON attribute), 25  
 impl (Locale attribute), 26  
 impl (Timezone attribute), 29  
 impl (UUID attribute), 30  
 index() (*WhooshIndexService method*), 44  
 index\_objects() (*WhooshIndexService method*), 44  
 index\_url() (*ObjectView method*), 55  
 Indexable (*class in abilian.core.entities*), 23  
 Indexable (*class in abilian.core.models.base*), 32  
 indexable\_role() (*in module abilian.services.indexing*), 46  
 Info (*class in abilian.core.models.base*), 32  
 init() (*ModuleComponent method*), 64  
 init\_app() (*ActionRegistry method*), 48  
 init\_app() (*Babel method*), 18  
 init\_app() (*Converter method*), 43  
 init\_app() (*Service method*), 41  
 init\_app() (*WhooshIndexService method*), 44  
 init\_breadcrumbs() (*Application method*), 16  
 init\_extensions() (*Application method*), 16  
 init\_filters() (*in module abilian.web.filters*), 47  
 init\_indexes() (*WhooshIndexService method*), 44  
 init\_module() (*ModuleComponent method*), 64  
 init\_object() (*BaseEntityView method*), 59  
 init\_object() (*BaseObjectView method*), 54  
 init\_object() (*ObjectCreate method*), 57  
 init\_object() (*ObjectDelete method*), 58  
 init\_related\_views() (*Module method*), 61  
 init\_work\_dirs() (*Converter method*), 43  
 insert() (*MutationList method*), 28  
 insert() (*NavGroup method*), 52  
 install\_id\_generator() (*Application method*), 16  
 installed() (*ActionRegistry method*), 48  
 is\_admin\_of() (*User method*), 33  
 is\_commentable() (*in module abilian.core.models.comment*), 37  
 is\_current() (*Module method*), 62  
 is\_following() (*User method*), 33  
 is\_member\_of() (*User method*), 33  
 is\_online() (*User property*), 34  
 iter\_pages() (*Pagination method*), 39  
 J  
 join() (*User method*), 33  
 js\_api (*Application attribute*), 16  
 JSON (*class in abilian.core.sqlalchemy*), 25  
 JSON2\_SEARCH\_LENGTH (*Module attribute*), 62  
 json\_search\_cls (*Module attribute*), 61  
 JSONBaseSearch (*class in abilian.web.views*), 58  
 JSONDict() (*in module abilian.core.sqlalchemy*), 32  
 JSONList() (*in module abilian.core.sqlalchemy*), 32  
 JSONModelSearch (*class in abilian.web.views*), 59  
 JSONUniqueListType (*class in abilian.core.sqlalchemy*), 26  
 JSONView (*class in abilian.web.views*), 53  
 JSONWhooshSearch (*class in abilian.web.views*), 59  
 L  
 label (*BreadcrumbItem attribute*), 51  
 label (*Module attribute*), 62  
 label (*Tag attribute*), 36  
 labelize() (*in module abilian.web.filters*), 47  
 labelize() (*in module abilian.web.frontend*), 64  
 last\_active (*User attribute*), 34  
 last\_name (*User attribute*), 34  
 lazy\_country\_name() (*in module abilian.i18n*), 18  
 lazy\_gettext() (*in module abilian.i18n*), 18  
 leave() (*User method*), 33  
 linkify() (*in module abilian.web.filters*), 47  
 list\_json2() (*Module method*), 62  
 list\_json2\_query\_all() (*Module method*), 62  
 list\_query() (*Module method*), 62  
 list\_view() (*Module method*), 62  
 list\_view\_columns (*Module attribute*), 63

*listing\_query()* (*Module property*), 63  
*ListJson* (*class in abilian.web.frontend*), 61  
*load\_dialect\_impl()* (*UUID method*), 30  
*local\_dt()* (*in module abilian.core.util*), 40  
*Locale* (*class in abilian.core.sqlalchemy*), 26  
*locale* (*User attribute*), 34  
*localeselector()* (*in module abilian.i18n*), 18  
*log\_activity()* (*ActivityService method*), 42

## M

*make\_single\_view()* (*in module abilian.web.frontend*), 64  
*managed\_class* (*Module attribute*), 63  
*md5()* (*Blob property*), 36  
*md5()* (*in module abilian.core.util*), 40  
*members* (*Group attribute*), 34  
*members\_count* (*Group attribute*), 34  
*memoized* (*class in abilian.core.util*), 39  
*message\_success()* (*ObjectEdit method*), 56  
*meta* (*Attachment attribute*), 38  
*meta* (*Blob attribute*), 36  
*meta* (*Comment attribute*), 37  
*meta* (*Entity attribute*), 21, 23  
*methods* (*BaseObjectView attribute*), 54  
*methods* (*EntityCreate attribute*), 60  
*methods* (*EntityDelete attribute*), 60  
*methods* (*EntityEdit attribute*), 61  
*methods* (*EntityView attribute*), 61  
*methods* (*JSONBaseSearch attribute*), 59  
*methods* (*JSONModelSearch attribute*), 59  
*methods* (*JSONView attribute*), 53  
*methods* (*JSONWhooshSearch attribute*), 59  
*methods* (*ListJson attribute*), 61  
*methods* (*ObjectCreate attribute*), 58  
*methods* (*ObjectDelete attribute*), 58  
*methods* (*ObjectEdit attribute*), 57  
*methods* (*ObjectView attribute*), 55  
*minimum\_input\_length* (*JSONBaseSearch attribute*), 59  
*ModalActionMixin* (*class in abilian.web.action*), 50  
*mode* (*EntityCreate attribute*), 60  
*mode* (*EntityEdit attribute*), 61  
*mode* (*EntityView attribute*), 61  
*Model* (*BaseObjectView attribute*), 54  
*Model* (*class in abilian.core.models.base*), 32  
*Model* (*JSONBaseSearch attribute*), 58  
*model()* (*TagCriterion property*), 65  
*Module* (*class in abilian.web.frontend*), 61  
*module* (*ModuleView attribute*), 64  
*ModuleAction* (*class in abilian.web.frontend*), 63  
*ModuleActionDropDown* (*class in abilian.web.frontend*), 63  
*ModuleActionGroup* (*class in abilian.web.frontend*), 63  
*ModuleActionGroupItem* (*class in abilian.web.frontend*), 64  
*ModuleComponent* (*class in abilian.web.frontend*), 64  
*ModuleMeta* (*class in abilian.web.frontend*), 64  
*ModuleView* (*class in abilian.web.frontend*), 64  
*MutationDict* (*class in abilian.core.sqlalchemy*), 27  
*MutationList* (*class in abilian.core.sqlalchemy*), 28

## N

*name* (*ActivityService attribute*), 42  
*name* (*Attachment attribute*), 38  
*name* (*Comment attribute*), 37  
*name* (*Entity attribute*), 21, 23  
*name* (*Group attribute*), 34  
*name* (*Module attribute*), 63  
*name* (*ModuleComponent attribute*), 64  
*name* (*Service attribute*), 41  
*name* (*WhooshIndexService attribute*), 45  
*name()* (*ClearPasswordStrategy property*), 35  
*name()* (*User property*), 34  
*NavGroup* (*class in abilian.web.nav*), 52  
*NavItem* (*class in abilian.web.nav*), 52  
*needs\_db\_flush* (*SecurityServiceState attribute*), 46  
*next()* (*Pagination property*), 39  
*ngettext()* (*in module abilian.i18n*), 18  
*n12br()* (*in module abilian.web.filters*), 47  
*noproxy()* (*in module abilian.core.util*), 40  
*ns* (*Tag attribute*), 36

## O

*obj* (*BaseObjectView attribute*), 54

obj\_to\_url() (in module *abilian.web.filters*), 47  
 object (*ActivityEntry* attribute), 42  
 object\_actions() (*EntityView* property), 61  
 object\_id (*ActivityEntry* attribute), 42  
 object\_id (*BaseObjectView* attribute), 54  
 object\_key() (*Indexable* property), 23, 32  
 object\_type (*ActivityEntry* attribute), 42  
 object\_type() (*Entity* property), 21, 23  
 object\_type() (*Indexable* property), 23, 32  
 ObjectCreate (class in *abilian.web.views*), 57  
 ObjectDelete (class in *abilian.web.views*), 58  
 ObjectEdit (class in *abilian.web.views*), 55  
 ObjectView (class in *abilian.web.views*), 55  
 options() (*JSONModelSearch* method), 59  
 order\_by() (*JSONModelSearch* method), 59  
 ordered\_query() (*Module* method), 62  
 OwnedMixin (class in *abilian.core.models.owned*), 35  
 owner (*Attachment* attribute), 38  
 owner (*Comment* attribute), 37  
 owner (*Entity* attribute), 21, 23  
 owner (*OwnedMixin* attribute), 35  
 owner\_id (*Attachment* attribute), 38  
 owner\_id (*Comment* attribute), 37  
 owner\_id (*Entity* attribute), 21, 23  
 owner\_id (*OwnedMixin* attribute), 35  
 owner\_name() (*OwnedMixin* property), 35  
**P**  
 pages() (*Pagination* property), 39  
 Pagination (class in *abilian.core.util*), 39  
 paragraphs() (in module *abilian.web.filters*), 48  
 password (*User* attribute), 34  
 patch\_logger (in module *abilian.core.logging*), 24  
 path() (*NavItem* property), 52  
 permission (*ObjectCreate* attribute), 58  
 permission (*ObjectDelete* attribute), 58  
 permission (*ObjectEdit* attribute), 57  
 permission (*ObjectView* attribute), 55  
 photo (*Group* attribute), 34  
 photo (*User* attribute), 34  
 ping\_connection() (in module *abilian.core.sqlalchemy*), 32  
 pk (*BaseEntityView* attribute), 60  
 pk (*BaseObjectView* attribute), 54  
 pop() (*MutationDict* method), 28  
 pop() (*MutationList* method), 28  
 popitem() (*MutationDict* method), 28  
 post() (*ObjectEdit* method), 56  
 pre\_condition() (*Action* method), 49  
 pre\_condition() (*ModuleAction* method), 63  
 preferences (*User* attribute), 34  
 prepare\_args() (*BaseEntityView* method), 59  
 prepare\_args() (*BaseObjectView* method), 54  
 prepare\_args() (*EntityCreate* method), 60  
 prepare\_args() (*JSONBaseSearch* method), 58  
 prepare\_args() (*JSONView* method), 53  
 prepare\_args() (*ObjectCreate* method), 58  
 prepare\_args() (*ObjectEdit* method), 57  
 prepare\_args() (*ObjectView* method), 55  
 prepare\_args() (*View* method), 52  
 prev() (*Pagination* property), 39  
 Principal (class in *abilian.core.models.subjects*), 34  
 private\_site (*Application* attribute), 16  
 process() (*ClearPasswordStrategy* method), 35  
 process\_bind\_param() (*JSON* method), 25  
 process\_bind\_param() (*JSONUniqueList-Type* method), 26  
 process\_bind\_param() (*Locale* method), 26  
 process\_bind\_param() (*Timezone* method), 29  
 process\_bind\_param() (*UUID* method), 31  
 process\_result\_value() (*JSON* method), 25  
 process\_result\_value() (*Locale* method), 27  
 process\_result\_value() (*Timezone* method), 29  
 process\_result\_value() (*UUID* method), 31  
 public (*Group* attribute), 34  
 put() (*ObjectEdit* method), 57  
 python\_type() (*JSONUniqueListType* prop-



erty), 26  
python\_type() (Locale property), 27  
python\_type() (Timezone property), 30

## Q

query() (Module method), 62  
query\_class (Entity attribute), 19, 22  
query\_class (User attribute), 33

## R

read\_query() (Module property), 63  
redirect() (View method), 52  
redirect\_to\_index() (BaseEntityView method), 60  
redirect\_to\_index() (ObjectView method), 55  
redirect\_to\_view() (ObjectEdit method), 57  
register() (ActionRegistry method), 48  
register() (in module *abilian.core.models.attachment*), 38  
register() (in module *abilian.core.models.comment*), 37  
register() (in module *abilian.core.models.tag*), 36  
register() (Registry method), 53  
register\_actions() (Module method), 62  
register\_class() (WhooshIndexService method), 44  
register\_classes() (WhooshIndexService method), 45  
register\_handler() (Converter method), 43  
register\_js\_api (in module *abilian.core.signals*), 24  
register\_plugin() (in module *abilian.web.attachments*), 46  
register\_plugin() (in module *abilian.web.comments*), 46  
register\_search\_filter() (WhooshIndexService method), 45  
register\_value\_provider() (WhooshIndexService method), 45  
Registry (class in *abilian.web.views*), 53  
related\_views (Module attribute), 63  
RelatedView (class in *abilian.web.frontend*), 64  
remove() (MutationList method), 28

remove() (TagsExtension method), 65  
render() (Action method), 49  
render() (BreadcrumbItem method), 51  
render() (DefaultRelatedView method), 60  
render() (RelatedView method), 64  
render\_template\_i18n() (in module *abilian.i18n*), 18  
resize() (in module *abilian.services.image*), 43  
reverse() (MutationList method), 28  
roughsize() (in module *abilian.web.filters*), 48  
running (ServiceState attribute), 41  
running() (Service property), 41

## S

SCALE (in module *abilian.services.image*), 43  
search() (WhooshIndexService method), 45  
search\_criteria (Module attribute), 63  
search\_for\_class() (WhooshIndexService method), 45  
searchable\_object\_types() (WhooshIndexService method), 45  
SecurityServiceState (class in *abilian.services.security.service*), 46  
send\_activity() (ObjectEdit method), 57  
send\_file\_from\_directory() (in module *abilian.web.util*), 66  
Service (class in *abilian.services.base*), 41  
service (in module *abilian.services.indexing*), 44  
service (in module *abilian.services.security*), 46  
service (ServiceState attribute), 41  
ServiceManager (class in *abilian.app*), 16  
ServiceNotRegistered, 41  
ServiceState (class in *abilian.services.base*), 41  
set\_attachment\_name() (in module *abilian.core.models.attachment*), 39  
set\_password() (User method), 33  
setdefault() (MutationDict method), 28  
setup() (Application method), 16  
setup\_nav\_and\_breadcrumbs() (Application method), 16  
short\_name() (User property), 34  
single\_view (Module attribute), 63

single\_view() (*BaseEntityView* property), 60  
 size() (*Blob* property), 36  
 slug (*Attachment* attribute), 38  
 slug (*Comment* attribute), 37  
 slug (*Entity* attribute), 21, 23  
 SLUG\_SEPARATOR (*Attachment* attribute), 38  
 SLUG\_SEPARATOR (*Comment* attribute), 37  
 SLUG\_SEPARATOR (*Entity* attribute), 20, 22  
 slugify() (in module *abilian.core.util*), 40  
 sort() (*MutationList* method), 28  
 SQLAlchemy (class in *abilian.core.sqlalchemy*), 28  
 start() (*ActivityService* method), 42  
 start() (*Service* method), 41  
 start() (*WhooshIndexService* method), 45  
 start\_services() (*ServiceManager* method), 16  
 static\_folder (*Module* attribute), 63  
 StaticIcon (class in *abilian.web.action*), 50  
 status() (*Action* property), 49  
 status() (*NavGroup* property), 52  
 status() (*NavItem* property), 52  
 stop() (*ActivityService* method), 42  
 stop() (*Service* method), 41  
 stop\_services() (*ServiceManager* method), 16  
 SupportAttachment (class in *abilian.core.models.attachment*), 38  
 supports\_attachments() (in module *abilian.core.models.attachment*), 39  
 supports\_tagging() (in module *abilian.core.models.tag*), 36  
 supports\_tagging() (*TagsExtension* method), 65  
 SupportTagging (class in *abilian.core.models.tag*), 36  
 SYSTEM (in module *abilian.core.models.base*), 32

**T**  
 tableview\_options (*Module* attribute), 63  
 Tag (class in *abilian.core.models.tag*), 36  
 TagCriterion (class in *abilian.web.tags*), 65  
 TAGS\_ATTR (in module *abilian.core.models.tag*), 37  
 tags\_from\_hit() (*TagsExtension* method), 65  
 TagsExtension (class in *abilian.web.tags*), 64  
 target (*ActivityEntry* attribute), 42  
 target\_id (*ActivityEntry* attribute), 42  
 target\_type (*ActivityEntry* attribute), 42  
 template (*BaseObjectView* attribute), 54  
 template (*DynamicIcon* attribute), 50  
 template (*EntityCreate* attribute), 60  
 template (*EntityEdit* attribute), 61  
 template (*EntityView* attribute), 61  
 template (*FAIcon* attribute), 50  
 template (*Glyphicon* attribute), 50  
 template (*ObjectEdit* attribute), 57  
 template (*ObjectView* attribute), 55  
 template\_kwargs() (*BaseObjectView* property), 55  
 template\_kwargs() (*EntityCreate* property), 60  
 template\_kwargs() (*EntityEdit* property), 61  
 template\_kwargs() (*EntityView* property), 61  
 template\_kwargs() (*ObjectView* property), 55  
 template\_string (*Action* attribute), 49  
 template\_string (*ActionDropDown* attribute), 49  
 template\_string (*ActionGroup* attribute), 50  
 template\_string (*BreadcrumbItem* attribute), 51  
 template\_string (*ButtonAction* attribute), 50  
 template\_string (*ModalActionMixin* attribute), 50  
 template\_string (*ModuleActionDropDown* attribute), 63  
 template\_string (*ModuleActionGroup* attribute), 63  
 template\_string (*NavGroup* attribute), 52  
 timer (class in *abilian.core.util*), 39  
 TimestampedMixin (class in *abilian.core.models.base*), 32  
 Timezone (class in *abilian.core.sqlalchemy*), 29  
 timezone (*User* attribute), 34  
 timezoneselector() (in module *abilian.i18n*), 18  
 title (*BaseObjectView* attribute), 55

title() (*Action property*), 49  
to\_dict() (*BaseMixin method*), 39  
to\_image() (*Converter method*), 43  
to\_json() (*BaseMixin method*), 39  
to\_pdf() (*Converter method*), 43  
to\_text() (*Converter method*), 43  
to\_timestamp() (*in module abilian.web.filters*), 48

## U

unfollow() (*User method*), 33  
unwrap() (*in module abilian.core.util*), 40  
update() (*MutationDict method*), 28  
updated\_at (*Attachment attribute*), 38  
updated\_at (*Comment attribute*), 37  
updated\_at (*Entity attribute*), 21, 23  
updated\_at (*Group attribute*), 34  
updated\_at (*TimestampedMixin attribute*), 32  
updated\_at (*User attribute*), 34  
url (*Module attribute*), 63  
url() (*Action method*), 49  
url() (*BreadcrumbItem property*), 51  
url\_for() (*in module abilian.web.util*), 66  
url\_for() (*Registry method*), 53  
use\_cache (*SecurityServiceState attribute*), 46  
User (*class in abilian.core.models.subjects*), 33  
user\_loaded (*in module abilian.core.signals*), 24  
utc\_dt() (*in module abilian.core.util*), 40  
utcnow() (*in module abilian.core.util*), 40  
uuid (*Blob attribute*), 36  
UUID (*class in abilian.core.sqlalchemy*), 30

## V

VALID\_LANGUAGES\_CODE (*in module abilian.i18n*), 19  
valid\_tags() (*TagCriterion property*), 65  
validate() (*ObjectEdit method*), 57  
ValidationError, 21  
value() (*Blob property*), 36  
verb (*ActivityEntry attribute*), 42  
View (*class in abilian.web.views*), 52  
view\_cls (*Module attribute*), 61  
view\_endpoint (*ObjectEdit attribute*), 57  
view\_form\_class (*Module attribute*), 63

view\_new\_save\_and\_add (*Module attribute*), 63  
view\_options (*Module attribute*), 63  
view\_template (*Module attribute*), 63  
view\_url() (*ObjectEdit method*), 57

## W

WhooshIndexService (*class in abilian.services.indexing.service*), 44  
with\_permission() (*EntityQuery method*), 23  
wrap\_collection() (*BasePresenter class method*), 39