Adhocracy Documentation

Release 2.0dev

Friedrich Lindenberg

September 13, 2012

CONTENTS

Adhocracy is a web-based platform that allows disperse groups to engage in a constructive process to draft proposals which will then represent the group's opinions and eventually its decisions regarding a given subject.

Contents:

CHAPTER

ONE

DEVELOPMENT DOCUMENTATION

1.1 Architecture

1.1.1 Database and model classes

Overview

We have 4 top level content classes user work with:

`page.Page` When the user creates a *Norm* (respectively Paper, or german "Papier") through the web interface, he is internally creating a *Page* with the `page.Page.function` set to `page.Page.NORM`. The text of the page is saved in *Text* objects in `page.Page.texts`, that may have different variants.

Page is a subclass of `delegateable.Delegateable`.

`proposal.Proposal` The actual text of the proposal is saved as a *Page* object with the `Page.function` `page.Page.DESCRIPTIION` in an attribute `page.Page.description`.

In a *Page* a user can propose changes to a *Page (Norm)*. These are saved as `selection.Selection` objects (see below).

Proposal is a subclass of `delegateable.Delegateable`.

`milestone.Milestone` A Milestone is used to represent a date or a deadline. Each Delegtable have a relation to one *Milestone*.

`comment.Comment` Comments can be created for `delegateable.Delegateable` objects. The reference to the *Delegateable* is available as `comment.Comment.topic`. Each comment is associated with a `poll.Poll` to rate content. If it is an answer to another *Comment* this is referenced with `comment.Comment.reply_id`. The text of an comment is stored as a `revision.Revision`. If a comment is edited, a new *Revision* is created. *Revisions* have similar function for *Comments* like *Texts* have for *Pages*.

Some of the more interesting models and support classes are:

`selection.Selection` A *Selection* is a "reference" object for the relation between *Pages (Norms)* and *Proposals* and stores additional information, especially when the relation was created or removed.

`text.Text` `Text` objects are only used to save text for `page.Page` objects. See *Page*.

`revision.Revision` Store the text of a Comment. Similar to *Text*. See *Comment*.

`poll.Poll` A *Poll* is an object representing a particular poll process a :class:Comment, :class:Proposal or :class:Page/Norm Single votes relate to the :class:Poll object.

vote.Vote A *Vote* is a single vote from a user. It knows about the user whose vote it is, and if it was a delegated vote the useris of the delegatee that voted for the user.

delegation.Delegation Created when a user delegates the voting to an delegatee. It knows about the user who (*principal_id*) delegated, who is the delegatee (*agent_id*) for which poll (*scope_id* - the id of the delegateable object).

delegateable.Delegateable Base class for *Pages* and *Proposals* for which a user can delegate the voting. Sqlalchemy's joint table inheritance is used.

tally.Tally A *Tally* saves the linear history of the :class:Poll noting which vote occured and what is the sum number of *for/against/abstains*.

watch.Watch Users can subscribe to content changes and will be notified depending settings in their preferences. A *Watch* object is created in these cases.

meta.Indexable Mixin class for content that should be indexed in Solr. It defines only one method `meta.Indexable.to_index()` that collects data from the models and will be uses automatically when a model object participates in a transaction.

Almost all model classes have a *classmethod* `.create()` to create a new instance of the model and setup the necessary data or relationships. Furthermore methods like `.find()` and `.all()` as convenient query method that support limiting the query to the current `model.instance.Instance` or in-/exclude deleted model instances.

Diagrams

Tables

Updating the diagrams

To update the diagramms install graphviz and easy_install sqlalchemy_schemadisplay into the environment adhocracy is installed in. Then run *python /adhocracy/scripts/generate-db-diagrams.py*. It will create the diagrams as GIF files. Finally replace the GIF files in *adhocracy/docs/development* with the new versions.

Delegateables

A user can delegate his vote to another user for `comment.Comment`, `proposal.Proposal` and `page.Page`. This functionality is enabled by inheriting from `Delegateable`

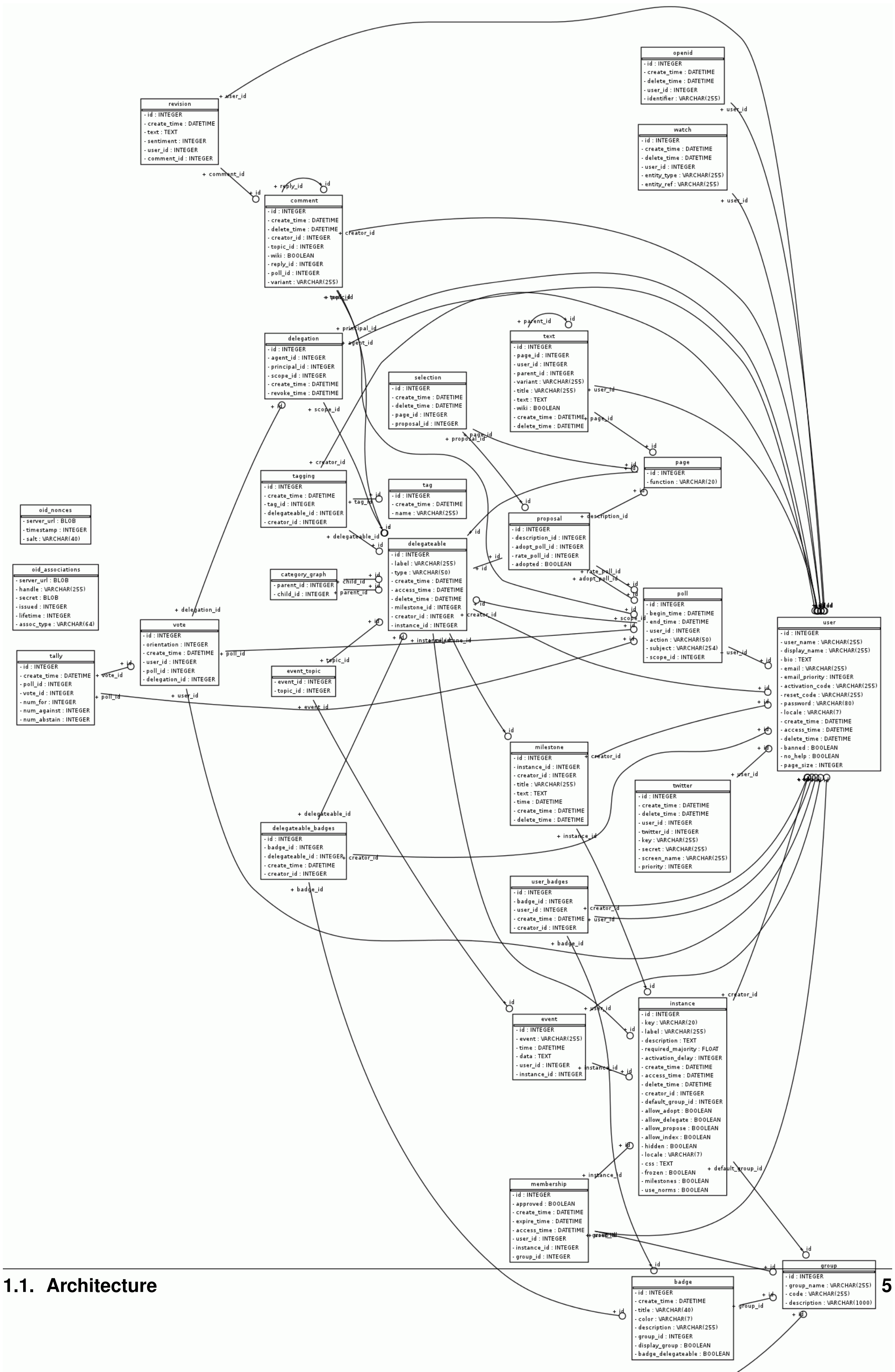
Inheritance is done with sqlalchemy's joint table inheritance where the *delegateable* table is polymorphic on `delegateable.Delegateable.type`

Page Models

The model class `page.Page` has 2 uses in adhocracy that are differentiated by the value of `page.Page.function`.

- A *Page* represents a *Norm* if `page.Page.function` is `page.Page.NORM`. This is the primary usage of Page.
- For every `proposal.Proposal` a page is created and available as `proposal.Proposal.description` to manage the text and text versions of the *proposal*. `page.Page.function` is `page.Page.DESCRIPTION` in this case.

Pages are *delegateable* and inherit from `delegateable.Delegateable`.


openid
- id : INTEGER
- create_time : DATETIME
- delete_time : DATETIME
- user_id : INTEGER
- identifier : VARCHAR(255)
watch
- id : INTEGER
- create_time : DATETIME
- delete_time : DATETIME
- user_id : INTEGER
- entity_type : VARCHAR(255)
- entity_ref : VARCHAR(255)
revision
- id : INTEGER
- create_time : DATETIME
- text : TEXT
- sentiment : INTEGER
- user_id : INTEGER
- comment_id : INTEGER
comment
- id : INTEGER
- create_time : DATETIME
- delete_time : DATETIME
- creator_id : INTEGER
- topic_id : INTEGER
- wiki : BOOLEAN
- reply_id : INTEGER
- poll_id : INTEGER
- variant : VARCHAR(255)
delegation
- id : INTEGER
- agent_id : INTEGER
- principal_id : INTEGER
- scope_id : INTEGER
- create_time : DATETIME
- revoke_time : DATETIME
text
- id : INTEGER
- page_id : INTEGER
- user_id : INTEGER
- parent_id : INTEGER
- variant : VARCHAR(255)
- title : VARCHAR(255)
- text : TEXT
- wiki : BOOLEAN
- create_time : DATETIME
- delete_time : DATETIME
selection
- id : INTEGER
- create_time : DATETIME
- delete_time : DATETIME
- page_id : INTEGER
- proposal_id : INTEGER
page
- id : INTEGER
- function : VARCHAR(20)
tagging
- id : INTEGER
- create_time : DATETIME
- tag_id : INTEGER
- delegateable_id : INTEGER
- creator_id : INTEGER
tag
- id : INTEGER
- create_time : DATETIME
- name : VARCHAR(255)
proposal
- id : INTEGER
- description_id : INTEGER
- adopt_poll_id : INTEGER
- rate_poll_id : INTEGER
- adopted : BOOLEAN
oid_nonces
- server_url : BLOB
- timestamp : INTEGER
- salt : VARCHAR(40)
oid_associations
- server_url : BLOB
- handle : VARCHAR(255)
- secret : BLOB
- issued : INTEGER
- lifetime : INTEGER
- assoc_type : VARCHAR(64)
delegateable
- id : INTEGER
- label : VARCHAR(255)
- type : VARCHAR(50)
- create_time : DATETIME
- access_time : DATETIME
- delete_time : DATETIME
- milestone_id : INTEGER
- creator_id : INTEGER
- instance_id : INTEGER
category_graph
- parent_id : INTEGER
- child_id : INTEGER
poll
- id : INTEGER
- begin_time : DATETIME
- end_time : DATETIME
- user_id : INTEGER
- action : VARCHAR(50)
- subject : VARCHAR(254)
- scope_id : INTEGER
user
- id : INTEGER
- user_name : VARCHAR(255)
- display_name : VARCHAR(255)
- bio : TEXT
- email : VARCHAR(255)
- email_priority : INTEGER
- activation_code : VARCHAR(255)
- reset_code : VARCHAR(255)
- password : VARCHAR(80)
- locale : VARCHAR(7)
- create_time : DATETIME
- access_time : DATETIME
- delete_time : DATETIME
- banned : BOOLEAN
- no_help : BOOLEAN
- page_size : INTEGER
vote
- id : INTEGER
- orientation : INTEGER
- create_time : DATETIME
- user_id : INTEGER
- poll_id : INTEGER
- delegation_id : INTEGER
tally
- id : INTEGER
- create_time : DATETIME
- poll_id : INTEGER
- vote_id : INTEGER
- num_for : INTEGER
- num_against : INTEGER
- num_abstain : INTEGER
event_topic
- event_id : INTEGER
- topic_id : INTEGER
milestone
- id : INTEGER
- instance_id : INTEGER
- creator_id : INTEGER
- title : VARCHAR(255)
- text : TEXT
- time : DATETIME
- create_time : DATETIME
- delete_time : DATETIME
twitter
- id : INTEGER
- create_time : DATETIME
- delete_time : DATETIME
- user_id : INTEGER
- twitter_id : INTEGER
- key : VARCHAR(255)
- secret : VARCHAR(255)
- screen_name : VARCHAR(255)
- priority : INTEGER
delegateable_badges
- id : INTEGER
- badge_id : INTEGER
- delegateable_id : INTEGER
- create_time : DATETIME
- creator_id : INTEGER
user_badges
- id : INTEGER
- badge_id : INTEGER
- user_id : INTEGER
- create_time : DATETIME
- creator_id : INTEGER
event
- id : INTEGER
- event : VARCHAR(255)
- time : DATETIME
- data : TEXT
- user_id : INTEGER
- instance_id : INTEGER
instance
- id : INTEGER
- key : VARCHAR(20)
- label : VARCHAR(255)
- description : TEXT
- required_majority : FLOAT
- activation_delay : INTEGER
- create_time : DATETIME
- access_time : DATETIME
- delete_time : DATETIME
- creator_id : INTEGER
- default_group_id : INTEGER
- allow_adopt : BOOLEAN
- allow_delegate : BOOLEAN
- allow_propose : BOOLEAN
- allow_index : BOOLEAN
- hidden : BOOLEAN
- locale : VARCHAR(7)
- css : TEXT
- frozen : BOOLEAN
- milestones : BOOLEAN
- use_norms : BOOLEAN
membership
- id : INTEGER
- approved : BOOLEAN
- create_time : DATETIME
- expire_time : DATETIME
- access_time : DATETIME
- user_id : INTEGER
- instance_id : INTEGER
- group_id : INTEGER
group
- id : INTEGER
- group_name : VARCHAR(255)
- code : VARCHAR(255)
- description : VARCHAR(1000)
badge
- id : INTEGER
- create_time : DATETIME
- title : VARCHAR(40)
- color : VARCHAR(7)
- description : VARCHAR(255)
- group_id : INTEGER
- display_group : BOOLEAN
- badge_delegateable : BOOLEAN
+ user_id
+ comment_id
+ reply_id
+ id
+ creator_id
+ topic_id
+ principal_id
+ agent_id
+ scope_id
+ parent_id
+ page_id
+ proposal_id
+ description_id
+ tag_id
+ delegateable_id
+ rate_poll_id
+ adopt_poll_id
+ child_id
+ delegation_id
+ poll_id
+ vote_id
+ event_id
+ instance_id
+ milestone_id
+ badge_id
+ default_group_id
+ group_id

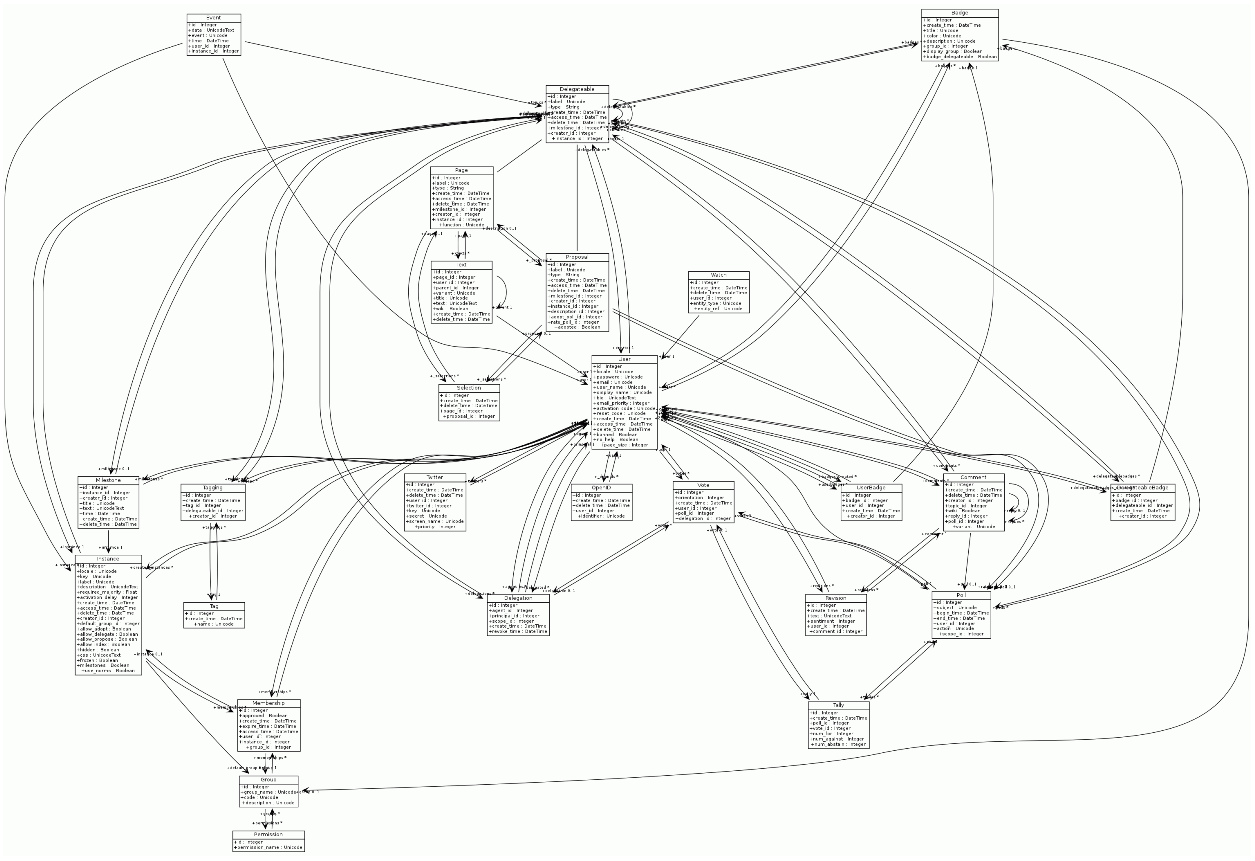


Figure 1.2: Diagramm of all model classes

Variants and Versions

The text of the *Page* is not saved in the page table but created as a `text.Text` object. A page can contain different variants of the text, and for each variant an arbitrary number of versions, e.g.:

- initial text
  - version 1
  - version 2
  - ...
- other text variant
  - version 1
  - version 2
  - ...
- ...

Text variants are used for *Norms*. For the initial text, *variant* is set to `text.Text.HEAD`. This text variant is handled special in the UI and labeled *Status Quo* in *Norms*. Other variants can be freely named. All text variants are listed in `Page.variants`.

Each variant can have a arbitrary number of versions. The newest version of the text is called the *head* (not to confuse with the default text variant *Text.HEAD*). You can get the newest version of a specific variant with `Page.variant_head()`. The newest versions of all variants is available as `Page.heads`. A shortcut to obtain the newest version of the *HEAD variant* is `Page.head`.

Text variants are not used for pages that are used as the description of `proposal.Proposal` objects (`proposal.Proposal.description`).

The poll tally of a variant or all variant can be optained with `Page.variant_tally()` or `Page.variant_tallies()`

Polls are set up per variant, not for the Page object.

Page Hierarchies

`Page` objects (that have the funciton *Norm*) can be organized in a tree stucture by setting another *Page (Norm)* object as one of the `Page.parents` of the current page. *Parents* can be an arbitrary number of `delegateable.Delegateable` objects, but only one, not already deleted Page with the `Page.function` `Page.NORM` is allowed. *Parents* are taken into account when we compute a *delegation graph*.

The subpages of a page are available as `Page.subpages`.

Other functionality

Beside that *Pages* have functions and attributes to handle purging, deleting, renaming, Selections (*Page (Norm)* <-> *Proposal* relationships) and other things. See the api documentation for `Page`

1.1.2 Indexing/Searching

Indexing and searching is done with *sql(alchemy)* and *solr*. Indexing with *solr* is done asyncronously most of the time while updates of the rdbm is done syncronously most of the time. The asyncronous indexing is done throug a *rabbitmq* job queue.

Types of search indexes

Beside rdbm intern indexes adhocracy maintains application specific indexes (that partly act as audit trails too):

- *solr' is used for full text and tag searches. It is an document oriented index. The index schema can be found in 'adhocracy/solr/schema.xml*
- new `tally.Tally` objects are created with every new or changed vote and provide the current total of votes.

Update application layer indexes

adhocracy implements an sqlalchemy *Mapperextension* with `hooks.HookExtension` that provides hook method to sqlalchemy that will be called before and after *insert*, *update* and *delete* operations for every model instance that is part of a commit. To determinate what to do it will inspect the model instance for fitting hook methods.

The asyncronous system roughly works like this:

- `hooks` defines a list of event hook methods names that are also used as event identifiers (:const:.PREINSERT, :const:.PREDELETE, :const:.PREUPDATE, :const:.POSTINSERT, :const:.POSTDELETE, :const:.POSTUPDATE)
- All model classes defined in `adhocracy.models.refs.TYPES` are patched by `init_queue_hooks()`. A function that posts a message message to the job queue (*_handle_event*) is patched in as all the method names listed above. The post to the job queue contains the the entity (model class) and the event identifier.
- A number of callback functions is registered by `adhocracy.lib.democracy.init_democracy()` with the help of `register_queue_callback()` in the hooks `REGISTRY`
- Everytime one of the patched models is inserted, updated, or deleted, a generic job is inserted into the jobqueue that contains the changed model instance and the event identifier.
- The background thread (*paster background <ini-file>*) picks up the jobs and calls `handle_queue_message()` which calls all registered callbacks.

To have indexing and searching working propperly you need:

- a working *rabbitmq*
- a working *solr*
- a running background process to process the jobs pushed into the *rabbitmq* job queue (*paster background <ini-file>*)

Read the install documentation for setup information.

1.1.3 Authentication and Permissions

1.2 Internal API Documentation

Adhocracy does not have a public programming API. There is a plan to develop a common *Kernel* interface for *Liquid Democracy* projects. Once such an API becomes available, Adhocracy will be modified to implement that API.

1.2.1 Core polling logic

1.2.2 Delegation management and traversal

1.2.3 Database models and helper classes

Badge

Comment

Delegateable

Delegation

Event

Group

Instance

Membership

Milestone

Openid

Page

Permission

Poll

Proposal

Revision

Selection

Tag

Tagging

Tally

Text

Twitter

User

Vote

Watch

1.2.4 Template Variables

Pylons provides a thread local variable `pylons.tmpl_context` that is available in templates a *c*. The following variables are commonly or always available in templates:

c.instance A `adhocracy.model.Instance` object or *None*. It is set by `adhocracy.lib.base.BaseController` from a value determinated by `adhocracy.lib.instance.DescriminatorMiddleware` from the host name.

c.user A `adhocracy.model.User` object or *None* if unauthenticated. It is set by `adhocracy.lib.base.BaseController` from a value determinated by the `repoze.who` middleware.

c.active_global_nav A *str* naming the current active top navigation item. It is set to 'instance' in `adhocracy.lib.base.BaseController` if the request is made to an instance and can be overridden in any controller.

1.3 Update translations

1.3.1 Translations for contributors

We manage our translations in a Transifex project. If you want to change a translation you can go to the project page, choose your language and click on the resource "Adhocracy". You will get a menu where you can download the .po file to edit it on your computer with an application like poedit ("Download for use"). After you translated the file you can go to the menu and upload the file. From the menu you can also use the transifex online editor (Button: " Translate now")

It would be nice to drop us a note before you start to translate, to adhocracy-dev@lists.liqd.net or info@liqd.net. You can also contact us to set up a new language on transifex.

1.3.2 Translations for developers

Adhocracy uses Babel together with Transifex to manage translations. Both are preconfigured in setup.cfg and .tx/config.

Preperations

CAUTION:: If you're using the adhocracy.buildout (highly recommended) you need to use the `--distribute` option to `bootstrap.py` to work with the preconfigured babel commands. This document assumes that you installed the buildout in a virtualenv "adhocracy".

Install the transifex client on your system. Then add your username and password for transifex.net to *~/.transifexrc*:

```
[https://www.transifex.net]
hostname = https://www.transifex.net
username = <your transifex username>
password = <your transifex password>
token =
```

Translation workflow

All .po and .pot files should go through transifex before they are committed. This might be annoying but unifies the formatting and makes it easier to review commits.

Extract new messages

1. Extract new messages with `extract_messages`. This will update `adhocracy/i18n/adhocracy.pot`:

   ```
   (adhocracy)/src/adhocracy$ ../../bin/adhocpy setup.py extract_messages
   ```

2. Push that to transifex:

   ```
   (adhocracy)/src/adhocracy$ tx push --source
   ```

3. Pull all files from transifex:

   ```
   (adhocracy)/src/adhocracy$ tx pull
   ```

   If it skips languages the files on transifex are older than the files on your system. See Troubleshooting.

4. Commit adhocracy.pot:

   ```
   (adhocracy)/src/adhocracy$ hg ci adhocracy/i18n/adhocracy.pot \
   > -m 'i18n: extract new messages'
   ```

Update the translations

1. Go to the transifex project and use the the online translation editor to translate and continue with 4.

   Or translate it locally. To do that make sure you have pulled the most recent translations from transifex:

   ```
   $ (adhocracy)/src/adhocracy$ tx pull  # pulls all languages or
   $ (adhocracy)/src/adhocracy$ tx pull -l <language>
   ```

2. Edit the `.po` files for your language(s).

   INFO:: The prefered way to edit `.po` files is to use an application like poedit. It will highlight untranslated messages and messages that were created with fuzzy matching and will automatically update or remove markers like `, fuzzy` and update the header of the `.po` file.

3. Push the translation to transifex:

   ```
   (adhocracy)/src/adhocracy$ tx push -l <language>
   ```

4. Pull the translation back:

   ```
   (adhocracy)/src/adhocracy$ tx pull -l <language>
   ```

5. Compile the catalogs with `compile_catalog`:

   ```
   (adhocracy)/src/adhocracy$ ../../bin/adhocpy setup.py compile_catalog
   ```

   This will also show you errors in the `.po` files and statistics about the translation.

6. Commit the .po and .mo files of the language(s) you translated, e.g.:

   ```
   (adhocracy)/src/adhocracy$ hg ci adhocracy/i18n/de' -m 'i18n: ...'
   ```

Troubleshooting

If tx skips the languages you want to pull from the server, the local file is most likely newer than the file on transifex.net. You can add *-d* to the command to get debug output, e.g.:

```
tx -d pull -l de
```

Than you have to check which of the files to use. Copy the local file and pull the language (with *-f/–force*) from transifex...:

```
(adhocracy)/src/adhocracy$ cd adhocracy/i18n/de/LC_MESSAGES
(adhocracy) .../de/LC_MESSAGES$ cp adhocracy.po local.po
(adhocracy) .../de/LC_MESSAGES$ tx  pull -f -l de
```

..and compare them. A good tool to compare is podiff from the Python GetText Translation Toolkit (which you can install from source of from their ubuntu ppa). It contains several other tools to work with po-files. You might have to give the *-r* (relax) option to podiff.

```
(adhocracy) .../de/LC_MESSAGES$ podiff local.po adhocracy.po
```

(There is also another podiff package on pypi.)

Babel command

`(adhocracy)/src/adhocracy$ ../../bin/adhocpy setup.py extract_messages` Extract the messages from the python files and templates into `adhocracy/i18n/adhocracy.pot`

`(adhocracy)/src/adhocracy$ ../../adhocpy setup.py compile_catalog` Compile the `.po` files for all languages to `.mo` files.

The babel command *update_catalog* should not be used anymore. Use the tx client instead.

1.4 Add and run tests

Before every commit you have to run all tests. Every new feature has to have a good test coverage. You should use Test Driven Develpment (http://en.wikipedia.org/wiki/Test-driven_development) and Acceptance Test Driven Develpment. Acceptance Tests correspondence to user stories (http://en.wikipedia.org/wiki/User_story). They use TestBrowser sessions and reside inside the functional tests directory.

1.4.1 Add a new test

‘‘**Go to (adhocracy)/src/adhocracy/adhocracy/tests and add you test** (http://pylonsbook.com/en/1.1/testing.html).

1.4.2 Run all tests

In an adhocracy.buildout you have `bin/test`. Alternatively you can call:

```
(adhocracy)$ bin/nosetests --with-pylons=src/adhocracy/test.ini src/adhocracy/adhocracy/tests``
```

1.4.3 Run one test file

```
(adhocracy)/src/adhocracy/$ ../../bin/nosetest -s adhocracy.tests.test_module
```

The -s option enables stdout, so you can use pdb/ipdb statements in your code.

CHAPTER

TWO

REST INTERFACE (OUTDATED)

Warning: This is the documentation for the REST interface. Unfortunately it is outdated and parts of the interface may not work anymore.

Warning: This is the documentation for the REST interface. Unfortunately it is outdated and parts of the interface may not work anymore.

2.1 REST API Conventions

Adhocracy provides a REST-inspired API for client applications to remotely gather or submit data or to synchronize Adhocracy sign-ups and voting processes with other sites.

While at the moment only JSON and RSS is produced and only JSON is processed by the software, future support for other formats, such as XML (i.e. StratML, EML) is planned.

2.1.1 Data Submission

All data submitted is expected to be either URL-encoded (for GET requests) or `application/x-www-form-urlencoded` (i.e. formatted as an HTML form, for either POST and PUT requests). Accept/Content-type based submission of JSON/XML data will be implemented in a later release.

A meta parameter called `_method` is evaluated for each request to fake a request method if needed. This is useful for older HTTP libraries or JavaScript clients which cannot actually perform any of the more exotic HTTP methods, such as PUT and DELETE.

2.1.2 Authentication and Security

Authentication can take place either via form-based cookie creation (POST `login` and `password` to `/perform_login`) or via HTTP Basic authentication (i.e. via HTTP headers).

Please note that for any write action using a cookie-based session, the site will expect an additional request parameter, `_tok`, containing a session ID. This is part of Adhocracy's CSRF filter and it will not apply to requests made using HTTP Basic authentication. Since the value of `_tok` is not returned by the API, it is recommended to use HTTP Basic for any API applications.

OAuth-based authorization is planned for a future release and will allow for token-based access to specific resources or operations.

2.1.3 Pagination

Many listings in Adhocracy are powered by a common pager system. Each pager has a specific prefix (e.g. `proposals_`) and a set of request parameters that can be used to influence the pager:

- `[prefix]_page`: The page number to retrieve, i.e. page offset.
- `[prefix]_count`: Number of items to retrieve per page.
- `[prefix]_sort`: Sorting key. These are somewhat erratically numbered and need to be redone in the future.
- `[prefix]_q` (in some cases): A search query used to filter the items.

Warning: This is the documentation for the REST interface. Unfortunately it is outdated and parts of the interface may not work anymore.

2.2 REST Resources

2.2.1 `/instance` - Group/Organization Instances

index

- List all existing and non-hidden instances.
- URL: `http://[instance].adhocracy.cc/instance[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- Pager prefix: `instances_`

create

- Create a new instance for a group or organization.
- URL: `http://[instance].adhocracy.cc/instance[.format]`
- Method: `POST`
- Formats: `html, json`
- Authentication: yes
- Parameters:
  - `key`: A unique identifier for the instance. Short lower-case alpha-numeric text. This cannot be edited after the instance creation.
  - `label`: A title for the instance.
  - `description`: Short description for the instance, e.g. a mission statement.

show

- View an instance's home page or base data
- URL: `http://[instance].adhocracy.cc/instance/[key][.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- *Note*: If no instance subdomain has been specified, this will 302 to the actual instance.

update

- Update some of an instance's properties.
- URL: `http://[instance].adhocracy.cc/instance/[key][.format]`
- Method: `PUT`
- Formats: `html, json`
- Authentication: yes
- Parameters:
  - `label`: A title for the instance.
  - `description`: Short description for the instance, e.g. a mission statement.
  - `required_majority`: The percentage of voters required for the adoption of a proposal (e.g. 0.66 for 66%).
  - `activation_delay`: Delay (in days) that a proposal needs to maintain a majority to be adopted.
  - `allow_adopt`: Whether to allow adoption polls on proposals (`bool`).
  - `allow_delegate`: Whether to enable delegated voting (`bool`).
  - `allow_index`: Allow search engine indexing (via robots.txt, `bool`).
  - `hidden`: Show instance in listings.
  - `default_group`: Default group for newly joined members (one of: `observer`, `advisor`, `voter`, `supervisor`).

delete

- Delete an instance and all contained entities.
- URL: `http://[instance].adhocracy.cc/instance/[key][.format]`
- Method: `DELETE`
- Formats: `html, json`
- Authentication: yes *(requires global admin rights)*
- *Note*: This will also delete all contained items, such as proposals, delegations, polls or comments.

activity

- Retrieve a list of the latest events relating to this instance.
- URL: `http://[instance].adhocracy.cc/instance/[key]/activity[.format]`
- Method: `GET`
- Formats: `html, rss`
- Authentication: no
- Pager prefix: `events_`

join

- Make the authenticated user a member of this `Instance`.
- URL: `http://[instance].adhocracy.cc/instance/[key]/join[.format]`
- Method: `GET`
- Formats: `html`
- Authentication: yes
- *Note*: Fails if the user is already a member of the instance.

leave

- Terminate the authenticated user's membership in this `Instance`.
- URL: `http://[instance].adhocracy.cc/instance/[key]/leave[.format]`
- Method: `POST`
- Formats: `html`
- Authentication: yes
- *Note*: Fails if the user is not a member of the instance.

2.2.2 `/user` - User Management

index

- List all users with an active membership in the specified instance.
- URL: `http://[instance].adhocracy.cc/user[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- Pager prefix: `users_`
- Parameters:
  - `users_q`: A search query to filter with.
  - `users_filter`: Filter by membership group (only in an instance context).

- *Note*: If no instance is specified, all registered users will be returned.

create

- Create a new user.
- URL: `http://[instance].adhocracy.cc/user[.format]`
- Method: `POST`
- Formats: `html, json`
- Authentication: no
- Parameters:
  - `user_name`: A unique user name for the new user.
  - `email`: An email, must be validated.
  - `password`: A password, min. 3 characters.
  - `password_confirm`: Must be identical to `password`.
- *Note*: Does not require an instance to be specified. If an instance is selected, the user will also become a member of that instance.

show

- View an user's home page and activity stream,
- URL: `http://[instance].adhocracy.cc/user/[user_name][.format]`
- Method: `GET`
- Formats: `html, json, rss`
- Authentication: no
- *Note*: Also available outside of instance contexts.

update

- Update the user's profile and settings.
- URL: `http://[instance].adhocracy.cc/user/[user_name][.format]`
- Method: `PUT`
- Formats: `html, json`
- Authentication: yes *(either to own user or with user management permissions)*
- Parameters:
  - `display_name`: Display name, i.e. the real name to be shown in the application.
  - `email`: E-Mail address. Must be re-validated when changed.
  - `locale`: A locale, currently: `de_DE`, `en_US` or `fr_FR`.
  - `password`: A password, min. 3 characters.
  - `password_confirm`: Must be identical to `password`.

  - `bio`: A short bio, markdown-formatted.
  - `email_priority`: Minimum priority level for E-Mail notifications to be sent (0-6).
  - `twitter_priority`: Minimum priority level for Twitter direct message notifications to be sent (0-6).

delete

- Delete an user. **Not implemented**

votes

- Retrieve a list of the decisions that were made by this user.
- URL: `http://[instance].adhocracy.cc/user/[user_name]/votes[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- Pager prefix: `decisions_`
- *Note*: Does not include rating polls, limited to adoption polls.

delegations

- Retrieve a list of the delegations that were created by this user.
- URL: `http://[instance].adhocracy.cc/user/[user_name]/delegations[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- Pager prefix: `delegations_` *(‘‘json‘‘ view only)*
- *Note*: In `html`, lists both incoming and outgoing delegations. When rendered as `json`, this only includes outgoing delegations.

instances

- A list of all non-hidden instances in which the user is a member.
- URL: `http://[instance].adhocracy.cc/user/[user_name]/instances[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- Pager prefix: `instances_`

proposals

- A list of all proposals that the user has introduced.
- URL: `http://[instance].adhocracy.cc/user/[user_name]/proposals[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- Pager prefix: `proposals_`

groupmod

- Modify a user's membership in the current instance
- URL: `http://[instance].adhocracy.cc/user/[user_name]/proposals[.format]`
- Method: `GET`
- Formats: `html`
- Authentication: yes *(requires instance admin privileges)*
- Parameters:
  - `to_group`: Target group (one of: `observer`, `advisor`, `voter`, `supervisor`).

kick

- Terminate a user's membership in the current instance
- URL: `http://[instance].adhocracy.cc/user/[user_name]/proposals[.format]`
- Method: `GET`
- Formats: `html`
- Authentication: yes *(requires instance admin privileges)*
- *Note*: Since the user can re-join at any time, this is largely a symbolic action.

2.2.3 `/proposal` - Proposal drafting

index

- List all existing proposals in the given instance.
- URL: `http://[instance].adhocracy.cc/proposal[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- Pager prefix: `proposals_`
- Parameters:
  - `proposals_q`: A search query to filter with.

  - `proposals_state`: Filter by state (one of: `draft`, `polling`, `adopted`). Only available if adoption polling is enabled in the selected instance.

create

- Create a new proposal.
- URL: `http://[instance].adhocracy.cc/proposal[.format]`
- Method: `POST`
- Formats: `html, json`
- Authentication: yes
- Parameters:
  - `label`: A title for the proposal.
  - `text`: Goals of the proposal.
  - `tags`: Comma-separated or space-separated tag list to be applied to the proposal.
  - `alternative` (multiple values): IDs of any proposals that should be marked as an alternative to this proposal.

show

- View an proposals's goal page
- URL: `http://[instance].adhocracy.cc/proposal/[id][.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no

update

- Update some of a proposal's properties.
- URL: `http://[instance].adhocracy.cc/proposal/[id][.format]`
- Method: `PUT`
- Formats: `html, json`
- Authentication: yes
- Parameters:
- `label`: A title for the proposal.
- `alternative` (multiple values): IDs of any proposals that should be marked as an alternative to this proposal.
- *Note*: The goal description and tag list are edited separately.

delete

- Delete a proposal and any contained entities.
- URL: `http://[instance].adhocracy.cc/proposal/[id][.format]`
- Method: `DELETE`
- Formats: `html, json`
- Authentication: yes *(requires instance admin rights)*
- *Note*: This will also delete all contained items, such as comments and delegations.

delegations

- Retrieve a list of the delegations that exist regarding this proposal.
- URL: `http://[instance].adhocracy.cc/proposal/[id]/delegations[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- Pager prefix: `delegations_`

canonicals

- Retrieve a list of canonical comments regarding the proposal. Canonical comments are listed as “provisions” in the UI.
- URL: `http://[instance].adhocracy.cc/proposal/[id]/delegations[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- *Note*: No pager.

alternatives

- Retrieve a list of the alternatives that exist regarding this proposal.
- URL: `http://[instance].adhocracy.cc/proposal/[id]/alternatives[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- Pager prefix: `proposals_`

activity

- Retrieve a list of events within the scope of the given proposal.
- URL: `http://[instance].adhocracy.cc/proposal/[id]/activity[.format]`
- Method: `GET`
- Formats: `html, rss`
- Authentication: no
- Pager prefix: `events_`

adopt

- Trigger an adoption poll regarding this proposal.
- URL: `http://[instance].adhocracy.cc/proposal/[id]/adopt[.format]`
- Method: `POST`
- Formats: `html`
- Authentication: yes
- *Note*: Requires at least one canonical comment. Adoption polls must be enabled on the instance level.

tag

- Apply an additional tag to a proposal (or support an existing tag).
- URL: `http://[instance].adhocracy.cc/proposal/[id]/tag[.format]`
- Method: `GET`
- Formats: `html`
- Authentication: yes
- Parameters:
  - `text`: Comma-separated or space-separated tag list to be applied to the proposal.

untag

- Remove a tag association (tagging) from a proposal.
- URL: `http://[instance].adhocracy.cc/proposal/[id]/untag[.format]`
- Method: `GET`
- Formats: `html`
- Authentication: yes
- Parameters:
  - `tagging`: ID of the tagging association to be removed.
- *Note*: Only taggings created by the user can be removed.

2.2.4 `/poll` - Poll data and voting

show

- View a poll, listing the current decisions and offering a chance to vote.
- URL: `http://[instance].adhocracy.cc/poll/[id][.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no

delete

- End a poll and close voting.
- URL: `http://[instance].adhocracy.cc/poll/[id][.format]`
- Method: `DELETE`
- Formats: `html, json`
- Authentication: yes
- *Note*: This will only work for adoption polls, rating polls cannot be terminated.

votes

- Retrieve a list of the decisions that were made regarding this poll.
- URL: `http://[instance].adhocracy.cc/poll/[id]/votes[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no
- Pager prefix: `decisions_`
- Parameters:
  - `result`: Filter for a specific decision, i.e. -1 (No), 1 (Yes), 0 (Abstained).

rate

- Vote in the poll via rating.
- URL: `http://[instance].adhocracy.cc/poll/[id]/rate[.format]`
- Method: `POST`
- Formats: `html, json`
- Authentication: yes
- *Note*: This implements relative voting, i.e. if a user has previously voted -1 and now votes 1, the result will be 0 (a relative change). Used for comment up-/downvoting. Unlike `vote`, this will also trigger an automated tallying of the poll. It is thus slower, especially for large polls.

vote

- Vote in the poll.
- URL: `http://[instance].adhocracy.cc/poll/[id]/vote[.format]`
- Method: `POST`
- Formats: `html, json`
- Authentication: yes
- *Note*: This does not trigger tallying. Thus a subsequent call to `show` might yield an incorrect tally until a server background job has run.

2.2.5 `/comment` - Commenting and comment history

index

- List all existing comments.
- URL: `http://[instance].adhocracy.cc/comment[.format]`
- Method: `GET`
- Formats: `json`
- Authentication: no
- Pager prefix: `comments_`

create

- Create a new comment within a specified context.
- URL: `http://[instance].adhocracy.cc/comment[.format]`
- Method: `POST`
- Formats: `html, json`
- Authentication: yes
- Parameters:
  - `topic`: ID of the Delegateable to which this comment is associated.
  - `reply`: A parent comment ID, if applicable.
  - `canonical` (bool): Specify whether this is part of the implementation description of the proposal to which it will be associated.
  - `text`: The comment text, markdown-formatted.
  - `sentiment`: General tendency of the comment, i.e. -1 for negative, 0 for neutral and 1 for a supporting argument.

show

- View a comment separated out of their context.
- URL: `http://[instance].adhocracy.cc/comment/[id][.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: no

update

- Create a new revision of the given comment.
- URL: `http://[instance].adhocracy.cc/comment/[id][.format]`
- Method: `PUT`
- Formats: `html, json`
- Authentication: yes
- Parameters:
  - `text`: The comment text, markdown-formatted.
  - `sentiment`: General tendency of the comment, i.e. -1 for negative, 0 for neutral and 1 for a supporting argument.

delete

- Delete a comment.
- URL: `http://[instance].adhocracy.cc/comment/[id][.format]`
- Method: `DELETE`
- Formats: `html, json`
- Authentication: yes
- *Note*: Comments can only be deleted by non-admins if they have not yet been edited.

history

- List all revisions of the specified comment.
- URL: `http://[instance].adhocracy.cc/comment/[id]/history[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: yes
- Pager prefix: `revisions_`

revert

- Revert to an earlier revision of the specified comment.
- URL: `http://[instance].adhocracy.cc/comment/[id]/revert[.format]`
- Method: `GET`
- Formats: `html, json`
- Authentication: yes
- Parameters:
  - `to`: Revision ID to revert to.
- *Note*: This will actually create a new revision containing the specified revision's text.

2.2.6 `/delegation` - Vote delegation management

index

- List all existing delegations (instance-wide).
- URL: `http://[instance].adhocracy.cc/delegation[.format]`
- Method: `GET`
- Formats: `json, dot`
- Authentication: no
- Pager prefix: `delegations_`
- *Note*: The `dot` format produces a graphviz file.

create

- Create a new delegation to a specified principal in a given scope.
- URL: `http://[instance].adhocracy.cc/delegation[.format]`
- Method: `POST`
- Formats: `html, json`
- Authentication: yes
- Parameters:
  - `scope`: ID of the `Delegateable` which will be the delegation's scope.
  - `agent`: User name of the delegation recipient.
  - `replay`: Whether or not to re-play all of the agents previous decisions within the scope.

show

- View the delegation.
- URL: `http://[instance].adhocracy.cc/delegation/[id][.format]`
- Method: `GET`

- Formats: `html, json`
- Authentication: no
- Pager prefix: `decisions_`
- *Note*: For `json` this will return a tuple of the actual serialized delegation and a list of decisions.

delete

- Revoke a the delegation.
- URL: `http://[instance].adhocracy.cc/delegation/[id][.format]`
- Method: `DELETE`
- Formats: `html, json`
- Authentication: yes
- *Note*: Can only be performed by the delegation's principal.

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*