invenio-accounts Documentation

Release 3.4.0

CERN

Aug 30, 2023

CONTENTS

Invenio user management and authentication.

Features:

- User and role management.
- User registration, password reset/recovery and email verification.
- Administration interface and CLI for managing users.
- Session based authentication with session theft protection support.
- Strong cryptographic password hashing with support for migrating password hashes (including Invenio v1.x) to new stronger algorithms.
- Session activity tracking allowing users to e.g. logout of all devices.
- Server-side session management.
- JSON Web Token encoding and decoding support useful for e.g. CSRF-protection in REST APIs.

Invenio-Accounts relies on the following community packages to do all the heavy-lifting:

- Flask-Security
- Flask-Login
- Flask-Principal
- Flask-KVSession
- Passlib

Further documentation is available on https://invenio-accounts.readthedocs.io/

CHAPTER

ONE

USER'S GUIDE

This part of the documentation will show you how to get started in using Invenio-Accounts.

1.1 Installation

Invenio-Accounts is on PyPI so all you need is:

```
$ pip install invenio-accounts
```

1.2 Configuration

Invenio-Accounts depends on many existing community packages, so a large part of the configuration is defined by these packages.

Please refer to the documentation of each package for a full overview over which configration variables that are available:

- Flask-Security
- Flask-Login
- Flask-Principal
- Flask-KVSession

Below we only cover the most important configuration options for Invenio-Accounts.

1.2.1 Secret key

The `SECRET_KEY` (see Flask documentation) is the most important configuration variable. A large part of the security of a web application is based on the secrecy of the value. In case the secret key is leaked, it is imperative that a new secret key is created.

1.2.2 Sessions

Server-side session data can be saved in different data stores (e.g. Redis), you must therefore provide a factory that returns the KV session store object:

invenio_accounts.config.**ACCOUNTS_SESSION_STORE_FACTORY = 'invenio_accounts.sessions:default_session_store_factory'**

Import path or function of factory used to generate the session store object.

When ACCOUNTS_SESSION_REDIS_URL will use redis as cache system otherwise otherwise it will use the in-memory backend simplekv.memory.DictStore.

invenio_accounts.config.**ACCOUNTS_SESSION_REDIS_URL = None**

Redis URL used by the module as a cache system for sessions.

1.2.3 Password hashing

Invenio defaults to use PBKDF2 SHA512 algorithm for password hashing:

invenio_accounts.config.**SECURITY_PASSWORD_HASH = 'pbkdf2_sha512'**

Default password hashing algorithm for new passwords.

Invenio has support for storing hashes using many different algoritms. For instance, by default Invenio also supports Invenio v1.x password hashes to make migration from v1.x easier. Legacy v1.x password hashes will however be automatically migrated to the new stronger algorithm the next time a user login. You can control the supported and deprecated algorithms using the following two configuration variables:

invenio_accounts.config.**SECURITY_PASSWORD_SCHEMES = ['pbkdf2_sha512', 'invenio_aes_encrypted_email']**

Supported password hashing algorithms (for passwords already stored).

You should include both the default, supported and any deprecated schemes.

invenio_accounts.config.**SECURITY_DEPRECATED_PASSWORD_SCHEMES = ['invenio_aes_encrypted_email']**

Deprecated password hashing algorithms.

Password hashes in a deprecated scheme are automatically migrated to the new default algorithm the next time the user login.

1.2.4 Recaptcha

The user registration form has support for recaptcha. All you need to do is to set the following two configuration variables (provided by reCAPTCHA when you register):

invenio_accounts.config.**RECAPTCHA_PUBLIC_KEY = None**

reCAPTCHA public key.

invenio_accounts.config.**RECAPTCHA_PRIVATE_KEY = None**

reCAPTCHA private key.

1.2.5 User tracking

Invenio-Accounts by default comes with user tracking enabled. The user tracking can be disabled using the configuration variables:

`invenio_accounts.config.`**`ACCOUNTS_SESSION_ACTIVITY_ENABLED = True`**

Enable session activity tracking.

`invenio_accounts.config.`**`SECURITY_TRACKABLE = True`**

Enable user tracking on login.

When a user login the following information is tracked:

- IP address (current and previous)
- Timestamp (current and previous)
- Login count

A user **do not** have control over above information as it is logged for security purposes.

In addition Invenio is tracking all active sessions of a user. For each active session we track:

- IP address
- Country of IP address
- Browser (e.g. Chrome)
- Browser version
- Operating system (e.g. MacOS)
- Device type (e.g. iPhone).

The user **do** have full control over the active sessions, meaning they can browse and revoke active session resulting in that the information is removed. The session activity tracking feature is used to allow users to logout from all their active sessions, but also allow administrators to ban a user and ensure they are logged out of all active sessions in the application.

Cleaning session activity table

If the session activity tracking is enabled you should also ensure that you regularly clean the session tracking tables for expired sessions. You do this by configuring a Celery Beat schedule similar to this:

```
from datetime import timedelta
CELERYBEAT_SCHEDULE = {
    'session_cleaner': {
        'task': 'invenio_accounts.tasks.clean_session_table',
        'schedule': timedelta(days=1),
    },
    'delete_login_ips': {
        'task': 'invenio_accounts.tasks.delete_ips',
        'schedule': timedelta(days=30),
    }
}
```

1.2.6 Templates

You can customize many of the templates used to render user registration, login, logout, email confirmations etc. Here are some few of the possiblities:

invenio_accounts.config.**SECURITY_LOGIN_USER_TEMPLATE = 'invenio_accounts/login_user.html'**

Default template for login.

invenio_accounts.config.**SECURITY_REGISTER_USER_TEMPLATE = 'invenio_accounts/register_user.html'**

Default template for user registration.

invenio_accounts.config.**SECURITY_RESET_PASSWORD_TEMPLATE = 'invenio_accounts/reset_password.html'**

Default template for password recovery (reset of the password).

invenio_accounts.config.**SECURITY_CHANGE_PASSWORD_TEMPLATE = 'invenio_accounts/change_password.html'**

Default template for change password.

invenio_accounts.config.**SECURITY_FORGOT_PASSWORD_TEMPLATE = 'invenio_accounts/forgot_password.html'**

Default template for password recovery (asking for email).

invenio_accounts.config.**SECURITY_SEND_CONFIRMATION_TEMPLATE = 'invenio_accounts/send_confirmation.html'**

Default template for email confirmation.

invenio_accounts.config.**SECURITY_SEND_LOGIN_TEMPLATE = 'invenio_accounts/send_login.html'**

Default template for email confirmation.

1.2.7 URLs

You can also customize the URLs under which you register and login in case you e.g. do not like the current naming:

invenio_accounts.config.**SECURITY_LOGIN_URL = '/login/'**

URL endpoint for login.

invenio_accounts.config.**SECURITY_LOGOUT_URL = '/logout/'**

URL endpoint for logout.

invenio_accounts.config.**SECURITY_REGISTER_URL = '/signup/'**

URL endpoint for user registation.

invenio_accounts.config.**SECURITY_RESET_URL = '/lost-password/'**

URL endpoint for password recovery.

1.2.8 Feature flags

A lot of the behaviour of Invenio-Accounts can be enabled/disabled depending on your current needs. Here are some of the feature flag options:

`invenio_accounts.config.SECURITY_REGISTERABLE = True`
: Allow users to register.

`invenio_accounts.config.SECURITY_RECOVERABLE = True`
: Allow password recovery by users.

`invenio_accounts.config.SECURITY_CONFIRMABLE = True`
: Allow user to confirm their email address.

`invenio_accounts.config.SECURITY_CHANGEABLE = True`
: Allow password change by users.

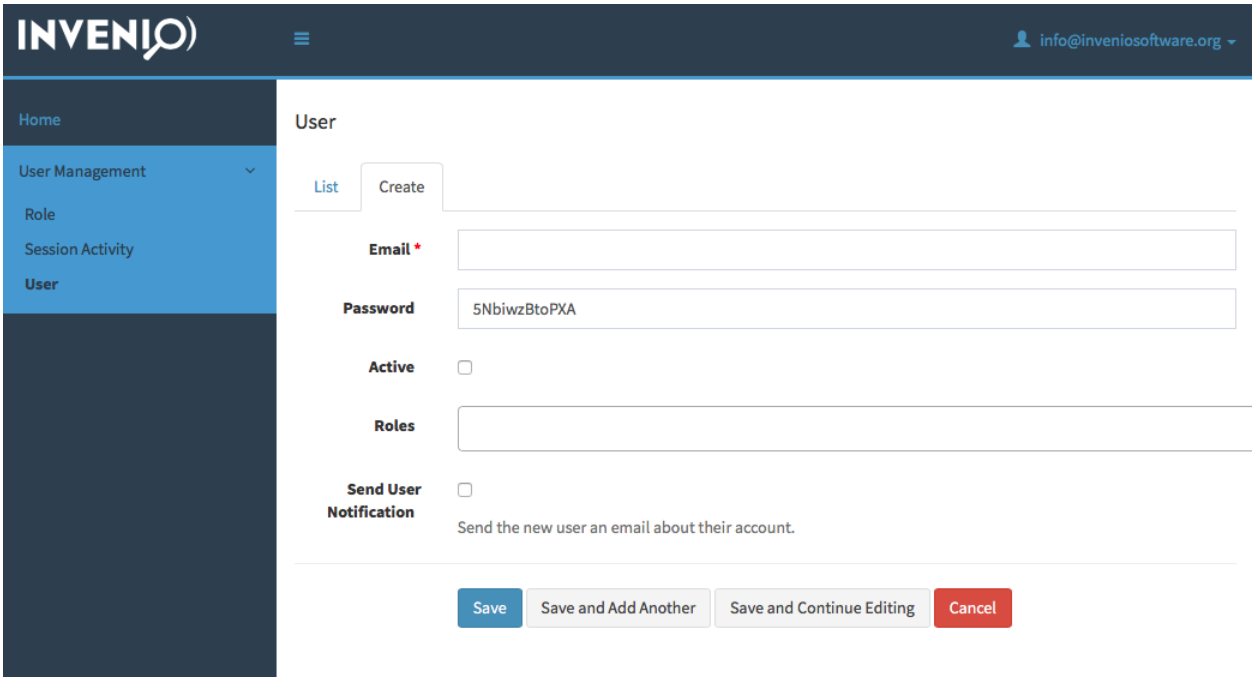
`invenio_accounts.config.SECURITY_LOGIN_WITHOUT_CONFIRMATION = True`
: Allow users to login without first confirming their email address.

1.3 Usage

Invenio user management and authentication.

1.3.1 Adminstration interface

You can view and manage users and roles via the administration interface. Below is a screenshot from the user creation:



1.3.2 Command-line interface

Users and roles can be created via the CLI. Below is a simple example of creating a user, a role and assining the user to the role:

```
$ flask users create --active info@inveniosoftware.org
$ flask roles create admins
$ flask roles add info@inveniosoftware.org admins
```

You can also e.g. deactive users:

```
$ flask users deactivate info@inveniosoftware.org
```

CHAPTER

TWO

API REFERENCE

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.1 API Docs

2.1.1 Extension

Invenio user management and authentication.

class invenio_accounts.ext.**InvenioAccounts**(*app=None*, *sessionstore=None*)

Invenio-Accounts extension.

Extension initialization.

Parameters

- **app** – The Flask application.
- **sessionstore** – store for sessions. Passed to `flask-kvsession`. Defaults to redis.

check_configuration_consistency(*app*)

Check if the config is consistent and issue a warning if not.

init_app(*app*, *sessionstore=None*, *register_blueprint=True*)

Flask application initialization.

The following actions are executed:

1. Initialize the configuration.
2. Monkey-patch Flask-Security.
3. Create the user datastore.
4. Create the sessionstore.
5. Initialize the extension, the forms to register users and confirms their emails, the CLI and, if `ACCOUNTS_USE_CELERY` is `True`, register a celery task to send emails.
6. Override Flask-Security's default login view function.
7. Warn if inconsistent configuration is detected

Parameters

- **app** – The Flask application.

- **sessionstore** – store for sessions. Passed to `flask-kvsession`. If `None` then Redis is configured. (Default: `None`)
- **register_blueprint** – If `True`, the application registers the blueprints. (Default: `True`)

init_config(*app*)

Initialize configuration.

Parameters **app** – The Flask application.

property jwt_creation_factory

Load default JWT creation factory.

property jwt_decode_factory

Load default JWT veryfication factory.

static monkey_patch_flask_security()

Monkey-patch Flask-Security.

register_anonymous_identity_loader(*state*)

Registers a loader for AnonymousIdentity.

Additional loader is necessary for applying a need 'any-user' to AnonymousUser in the invenio-access module

class invenio_accounts.ext.**InvenioAccountsREST**(*app=None*, *sessionstore=None*)

Invenio-Accounts REST extension.

Extension initialization.

Parameters

- **app** – The Flask application.
- **sessionstore** – store for sessions. Passed to `flask-kvsession`. Defaults to redis.

init_app(*app*, *sessionstore=None*, *register_blueprint=False*)

Flask application initialization.

Parameters

- **app** – The Flask application.
- **sessionstore** – store for sessions. Passed to `flask-kvsession`. If `None` then Redis is configured. (Default: `None`)
- **register_blueprint** – If `True`, the application registers the blueprints. (Default: `True`)

class invenio_accounts.ext.**InvenioAccountsUI**(*app=None*, *sessionstore=None*)

Invenio-Accounts UI extension.

Extension initialization.

Parameters

- **app** – The Flask application.
- **sessionstore** – store for sessions. Passed to `flask-kvsession`. Defaults to redis.

init_app(*app*, *sessionstore=None*, *register_blueprint=True*)

Flask application initialization.

Parameters

- **app** – The Flask application.
- **sessionstore** – store for sessions. Passed to `flask-kvsession`. If `None` then Redis is configured. (Default: `None`)
- **register_blueprint** – If `True`, the application registers the blueprints. (Default: `True`)

make_session_permanent(*app*)

Make session permanent by default.

Set *PERMANENT_SESSION_LIFETIME* to specify time-to-live

2.1.2 Administration

2.1.3 Datastore

Session-aware datastore.

class invenio_accounts.datastore.**SessionAwareSQLAlchemyUserDatastore**(*db*, *user_model*, *role_model*)

Datastore which deletes active session when a user is deactivated.

commit()

Commit a user to its session.

create_role(***kwargs*)

Creates and returns a new role from the given parameters.

deactivate_user(*user*)

Deactivate a user.

Parameters **user** – A *invenio_accounts.models.User* instance.

Returns The datastore instance.

find_role_by_id(*role_id*)

Fetches roles searching by id.

mark_changed(*sid*, *uid=None*, *rid=None*)

Save a user to the changed history.

update_role(*role*)

Updates roles.

2.1.4 Errors

Exception classes.

exception invenio_accounts.errors.**AlreadyLinkedError**(*user*, *external_id*)

Signifies that an account was already linked to another account.

Initialize exception.

exception invenio_accounts.errors.**JWTDecodeError**

Exception raised when decoding is failed.

exception invenio_accounts.errors.**JWTExpiredToken**

Exception raised when JWT is expired.

exception invenio_accounts.errors.**JWTExtendedException**

Base exception for all JWT errors.

2.1.5 Forms

Additional non-userprofile fields used during registration.

Currently supported: recaptcha

class invenio_accounts.forms.**RegistrationFormRecaptcha**(**args*, ***kwargs*)

Form for editing user profile.

Parameters

- **formdata** – Input data coming from the client, usually `request.form` or equivalent. Should provide a "multi dict" interface to get a list of values for a given key, such as what Werkzeug, Django, and WebOb provide.
- **obj** – Take existing data from attributes on this object matching form field attributes. Only used if `formdata` is not passed.
- **prefix** – If provided, all fields will have their name prefixed with the value. This is for distinguishing multiple forms on a single page. This only affects the HTML name for matching input data, not the Python name for matching existing data.
- **data** – Take existing data from keys in this dict matching form field attributes. `obj` takes precedence if it also has a matching attribute. Only used if `formdata` is not passed.
- **meta** – A dict of attributes to override on this form's `meta` instance.
- **extra_filters** – A dict mapping field attribute names to lists of extra filter functions to run. Extra filters run after filters passed when creating the field. If the form has `filter_<fieldname>`, it is the last extra filter.
- **kwargs** – Merged with `data` to allow passing existing data as parameters. Overwrites any duplicate keys in `data`. Only used if `formdata` is not passed.

class invenio_accounts.forms.**RevokeForm**(**args*, ***kwargs*)

Form for revoking a session.

Parameters

- **formdata** – Input data coming from the client, usually `request.form` or equivalent. Should provide a "multi dict" interface to get a list of values for a given key, such as what Werkzeug, Django, and WebOb provide.
- **obj** – Take existing data from attributes on this object matching form field attributes. Only used if `formdata` is not passed.
- **prefix** – If provided, all fields will have their name prefixed with the value. This is for distinguishing multiple forms on a single page. This only affects the HTML name for matching input data, not the Python name for matching existing data.
- **data** – Take existing data from keys in this dict matching form field attributes. `obj` takes precedence if it also has a matching attribute. Only used if `formdata` is not passed.
- **meta** – A dict of attributes to override on this form's `meta` instance.

- **extra_filters** – A dict mapping field attribute names to lists of extra filter functions to run. Extra filters run after filters passed when creating the field. If the form has `filter_<fieldname>`, it is the last extra filter.
- **kwargs** – Merged with `data` to allow passing existing data as parameters. Overwrites any duplicate keys in `data`. Only used if `formdata` is not passed.

`invenio_accounts.forms.`**confirm_register_form_factory**(*Form*, *app*)

Return confirmation for extended registration form.

`invenio_accounts.forms.`**login_form_factory**(*Form*, *app*)

Return extended login form.

`invenio_accounts.forms.`**register_form_factory**(*Form*, *app*)

Return extended registration form.

`invenio_accounts.forms.`**send_confirmation_form_factory**(*Form*, *app*)

Return extended login form.

2.1.6 Hash

Legacy Invenio hash support.

class `invenio_accounts.hash.`**InvenioAesEncryptedEmail**(*salt=None*, ***kwds*)

Invenio AES encryption of user email using password as secret key.

Invenio 1.x was AES encrypting the users email address with the password as the secret key and storing it in a blob column. This e.g. caused problems when a user wanted to change email address. This hashing engine, differs from Invenio 1.x in that it sha256 hashes the encrypted value as well to produce a string in the same length instead of a binary blob. It is not done for extra security, just for convenience of migration to using passlib's sha512. An upgrade recipe is provided to migrated existing binary password hashes to hashes of this engine.

classmethod from_string(*hash*, ***context*)

Parse instance from configuration string in Modular Crypt Format.

to_string()

Render instance to configuration string in Modular Crypt Format.

2.1.7 Models

Database models for accounts.

class `invenio_accounts.models.`**LoginInformation**(***kwargs*)

Login information for a user.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

current_login_at

When user logged into the current session.

current_login_ip

Current user IP address.

last_login_at

When the user logged-in for the last time.

last_login_ip

Last user IP address.

login_count

Count how many times the user logged in.

user

User to whom this information belongs.

user_id

ID of user to whom this information belongs.

validate_ip(*key*, *value*)

Hack untrackable IP addresses.

class invenio_accounts.models.**Role**(***kwargs*)

Role data model.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

created

description

Role description.

is_managed

True when the role is managed by Invenio, and not externally provided.

name

Role name.

updated

version_id

Used by SQLAlchemy for optimistic concurrency control.

class invenio_accounts.models.**SessionActivity**(***kwargs*)

User Session Activity model.

Instances of this model correspond to a session belonging to a user.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

browser

User browser.

browser_version

Browser version.

country
Country name.

created

device
User device.

ip
IP address.

classmethod is_current(*sid_s*)
Check if the session is the current one.

os
User operative system name.

classmethod query_by_expired()
Query to select all expired sessions.

classmethod query_by_user(*user_id*)
Query to select user sessions.

sid_s
Serialized Session ID. Used as the session's key in the kv-session store employed by *flask-kvsession*. Named here as it is in *flask-kvsession* to avoid confusion.

updated

user_id
ID of user to whom this session belongs.

class invenio_accounts.models.**User**(**args*, ***kwargs*)
User data model.

Constructor.

active
Flag to say if the user is active or not .

confirmed_at
When the user confirmed the email address.

created

property current_login_at
When user logged into the current session.

property current_login_ip
Current user IP address.

email
User email.

property last_login_at
When the user logged-in for the last time.

property last_login_ip
Last user IP address.

property login_count

Count how many times the user logged in.

password

User password.

preferences

Get the user preferences.

roles

List of the user's roles.

updated

user_profile

Get the user profile.

username

Get username.

version_id

Used by SQLAlchemy for optimistic concurrency control.

class invenio_accounts.models.**UserIdentity**(***kwargs*)

Represent a UserIdentity record.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

classmethod create(*user*, *method*, *external_id*)

Link a user to an external id.

Parameters

- **user** – A *invenio_accounts.models.User* instance.
- **method** – The identity source (e.g. orcid, github)
- **method** – The external identifier.

Raises *AlreadyLinkedError* – Raised if already exists a link.

created

classmethod delete_by_external_id(*method*, *external_id*)

Unlink a user from an external id.

classmethod delete_by_user(*method*, *user*)

Unlink a user from an external id.

classmethod get_user(*method*, *external_id*)

Get the user for a given identity.

updated

invenio_accounts.models.**userrole = Table('accounts_userrole', MetaData(), Column('user_id', Integer(), ForeignKey('accounts_user.id'), table=<accounts_userrole>), Column('role_id', String(length=80), ForeignKey('accounts_role.id'), table=<accounts_userrole>), schema=None)**

Relationship between users and roles.

2.1.8 Utils

Utility function for ACCOUNTS.

invenio_accounts.utils.**change_user_password**(*_reset_password_link_func=None*, ***user_data*)

Change user password.

invenio_accounts.utils.**default_confirmation_link_func**(*user*)

Return the confirmation link that will be sent to a user via email.

invenio_accounts.utils.**default_reset_password_link_func**(*user*)

Return the reset password link that will be sent to a user via email.

invenio_accounts.utils.**jwt_create_token**(*user_id=None*, *additional_data=None*)

Encode the JWT token.

Parameters

- **user_id** (*int*) – Addition of user_id.
- **additional_data** (*dict*) – Additional information for the token.

Returns The encoded token.

Return type str

Note: Definition of the JWT claims:

- exp: ((Expiration Time) expiration time of the JWT.
- sub: (subject) the principal that is the subject of the JWT.
- jti: (JWT ID) UID for the JWT.

---

invenio_accounts.utils.**jwt_decode_token**(*token*)

Decode the JWT token.

Parameters **token** (*str*) – Additional information for the token.

Returns The token data.

Return type dict

invenio_accounts.utils.**obj_or_import_string**(*value*, *default=None*)

Import string or return object.

Params value Import path or class object to instantiate.

Params default Default object to return if the import fails.

Returns The imported object.

invenio_accounts.utils.**register_user**(*_confirmation_link_func=None*, *send_register_msg=True*, ***user_data*)

Register a user.

invenio_accounts.utils.**set_session_info**(*app*, *response*, ***extra*)

Add X-Session-ID and X-User-ID to http response.

invenio_accounts.utils.**validate_username**(*username*)

Validate the username.

Parameters **username** – The username to validate.

Raises ValueError – If validation fails.

2.1.9 Testing utils

Invenio-Accounts utility functions for tests and testing purposes.

Warning: DO NOT USE IN A PRODUCTION ENVIRONMENT.

Functions within accessing the datastore will throw an error if called outside of an application context. If pytest-flask is installed you don't have to worry about this.

invenio_accounts.testutils.**client_authenticated**(*client*, *test_url=None*)

Attempt to access the change password page with the given client.

Parameters **test_url** – URL to attempt to get. Defaults to the current application's "change password" page.

Returns True if the client can get the test_url without getting redirected and flask_login.current_user is not anonymous after requesting the page.

invenio_accounts.testutils.**create_test_user**(*email*, *password='123456'*, ***kwargs*)

Create a user in the datastore, bypassing the registration process.

Accesses the application's datastore. An error is thrown if called from outside of an application context.

Returns the created user model object instance, with the plaintext password as *user.password_plaintext*.

Parameters

- **email** – The user email.
- **password** – The user password. (Default: 123456)

Returns A *invenio_accounts.models.User* instance.

invenio_accounts.testutils.**login_user_via_session**(*client*, *user=None*, *email=None*)

Login a user via the session.

Parameters

- **client** – The CLI test client.
- **user** – The *invenio_accounts.models.User* instance. Optional. (Default: None)
- **email** – Load the user by the email. Optional. (Default: None)

invenio_accounts.testutils.**login_user_via_view**(*client*, *email=None*, *password=None*, *user=None*, *login_url=None*)

Attempt to log the given user in via the 'login' view on the client.

Parameters

- **client** – client to send the request from.
- **email** – email of user account to log in with.
- **password** – password of user account to log in with.
- **user** (*invenio_accounts.models.User* (with the addition of a `password_plaintext` field)) – If present, `user.email` and `user.password_plaintext` take precedence over the *email* and *password* parameters.
- **login_url** – URL to post login details to. Defaults to the current application's login URL.

Returns The response object from the POST to the login form.

invenio_accounts.testutils.**unserialize_session**(*sid_s*)

Return the unserialized session.

Parameters **sid_s** – The session ID.

Returns The unserialized version.

invenio_accounts.testutils.**webdriver_authenticated**(*webdriver*, *test_url=None*)

Attempt to get the change password page through the given webdriver.

Similar to *client_authenticated*, but for selenium webdriver objects.

2.1.10 Tasks

()invenio_accounts.tasks.**send_security_email**(*data*)

Celery task to send security email.

Parameters **data** – Contains the email data.

()invenio_accounts.tasks.**clean_session_table**()

Automatically clean session table.

To enable a periodically clean of the session table, you should configure the task as a celery periodic task.

```
from datetime import timedelta
CELERYBEAT_SCHEDULE = {
    'session_cleaner': {
        'task': 'invenio_accounts.tasks.clean_session_table',
        'schedule': timedelta(days=1),
    },
}
```

See Invenio-Celery documentation for further details.

2.1.11 Views

Invenio user management and authentication.

invenio_accounts.views.security.**revoke_session**()

Revoke a session.

invenio_accounts.views.security.**security**()

View for security page.

Invenio user management and authentication.

invenio_accounts.views.settings.**check_security_settings**()

Warn if session cookie is not secure in production.

invenio_accounts.views.settings.**init_menu**()

Initialize menu before first request.

CHAPTER

THREE

ADDITIONAL NOTES

Notes on how to contribute, legal information and changes are here for the interested.

3.1 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

3.1.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/inveniosoftware/invenio-accounts/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

Invenio-Accounts could always use more documentation, whether as part of the official Invenio-Accounts docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/inveniosoftware/invenio-accounts/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.1.2 Get Started!

Ready to contribute? Here's how to set up *invenio-accounts* for local development.

1. Fork the *inveniosoftware/invenio-accounts* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/invenio-accounts.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv invenio-accounts
$ cd invenio-accounts/
$ pip install -e .[all]
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass tests:

```
$ ./run-tests.sh
```

   The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8 as well as build the Sphinx documentation and run doctests.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -s
    -m "component: title without verbs"
    -m "* NEW Adds your new feature."
    -m "* FIX Fixes an existing issue."
    -m "* BETTER Improves and existing feature."
    -m "* Changes something that should not be visible in release notes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

3.1.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests and must not decrease test coverage.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
3. The pull request should work for Python 2.7, 3.3, 3.4 and 3.5. Check https://github.com/inveniosoftware/invenio-accounts/actions?query=event%3Apull_request and make sure that the tests pass for all supported Python versions.

3.2 Changes

Version 3.4.0 (released 2023-08-30)

- templates: refactor send confirmation template

Version 3.3.1 (released 2023-08-23)

- config: set *ACCOUNTS_DEFAULT_USERS_VERIFIED* to False by default

Version 3.3.0 (released 2023-08-21)

- models: add *verified_at* column in User model. The default value is controlled by a new config variable called *ACCOUNTS_DEFAULT_USERS_VERIFIED*. If True, then a date is generated, otherwise is set to *None*.

Version 3.2.1 (released 2023-08-17)

- alembic: fix sqlalchemy op.execute statements due to latest sqlalchamy-continuum

Version 3.2.0 (released 2023-08-02)

- users: add blocket_at and verified_at data model fields

Version 3.1.0 (released 2023-07-31)

- templates: Improve accessibility and layout
- pulled translations

Version 3.0.3 (released 2023-06-15)

- models: fix autogeneration of role id

Version 3.0.2 (released 2023-06-14)

- alembic: adapt recipe to mysql

Version 3.0.1 (released 2023-06-14)

- alembic: fix upgrade recipes

Version 3.0.0 (released 2023-06-14)

- models: add managed field to groups
- models: alter primary key type of group (id)
- cli: pass id on create role action

Version 2.2.0 (released 2023-04-25)

- models: add support for locale in user preferences

Version 2.1.0 (released 2023-03-01)

- global: replace deprecated babelex imports
- update invenio-i18n

Version 2.0.2 (released 2022-12-14)

- cli: add *–confirm* flag when creating a user
- new config variables to set the default user and email visibility
- register_user: method accepts new argument, *send_register_msg*, to control programmatically the send of registration email independently of the global configuration.

Version 2.0.1 (released 2022-11-18)

- Add translation workflow
- Add pulled translations
- Add black
- Fix icons not appearing

Version 2.0.0 (released 2022-05-23)

- Adds customizable user profiles and user preferences fields to the user data model.
- Adds version counter to the user table to enable optimistic concurrency control on the user table.
- Moves login information fields from user table to a separate login information table.
- Moves the external user identity table from Invenio-OAuthclient to Invenio-Accounts.
- Adds support for tracking changed users within a transaction to allow for updating the related indexes.
- Changes from using Flask-Security to using a private fork named Flask-Security-Invenio. Flask-Security-Too was evaluated but was found to have significantly increased scope with features not needed.

Version 1.4.9 (released 2021-12-04)

- Fixed issue with account creation via CLI due to issue with changed API in Flask-WTF.

Version 1.4.8 (released 2021-10-18)

- Unpin Flask requirement.

Version 1.4.7 (released 2021-10-06)

- Adds celery task to remove IP addresses from user table after a specified retention period (defaults to 30 days).

Version 1.4.6 (released 2021-07-12)

- Adds german translations

Version 1.4.5 (released 2021-05-21)

- Removes config entrypoint.
- Bump module versions.

Version 1.4.4 (released 2021-05-11)

- Enables login view function overridability.
- Allows to disable local login via configuration variable.

Version 1.4.3 (released 2020-12-17)

- Adds theme dependent icons.

Version 1.4.2 (released 2020-12-11)

- Fixes logout from security view.

Version 1.4.1 (released 2020-12-10)

- Fixes styling of forgot password form in semantic ui theme.

Version 1.4.0 (released 2020-12-09)

- Major: adds new Semantic UI theme.
- Adds Turkish translations.
- Fixes `next` parameter being used in the sign-up form.
- Fixes issue with translation files causing translations not to be picked up.
- Fixes wording from sign in to log in.
- Removes password length validation during login.

Version 1.3.0 (released 2020-05-15)

- Refreshes the CSRF token on login and logout.
- Removes the example app.
- Migrate from *Flask-KVSession* to *Flask-KVSession-Invenio*, fork of the former.

Version 1.2.2 (released 2020-05-13)

This release was removed from PyPI on 2020-05-15 due to issues with the release.

Version 1.2.1 (released 2020-04-28)

- Fixes issue with the latest WTForms v2.3.x release which now requires an extra library for email validation.

Version 1.2.0 (released 2020-03-09)

- Replaces Flask dependency with centrally managed invenio-base

Version 1.1.4 (released 2020-04-28)

- Fixes issue with the latest WTForms v2.3.x release which now requires an extra library for email validation.

Version 1.1.3 (released 2020-02-19)

- Replaces Flask-CeleryExt to invenio-celery due to version incompatibilities with celery, kombu. Removes Flask-BabelExt already provided by invenio-i18n

Version 1.1.2 (released 2020-02-12)

- Fixes requirements for Flask, Werkzeug and Flask-Login due to incompatibilities of latest released modules.

Version 1.1.1 (released 2019-03-10)

- Fixes an issue where the HTTP headers X-Session-ID and X-User-ID are added even if the value is not known. This causes 'None' to be logged in Nginx, instead of simply '-'.

Version 1.1.0 (released 2019-02-15)

- Added support for for adding the user id and session id of the current user into the HTTP headers (`X-User-ID` and `X-Session-ID`) for upstream servers to use. For instance, this way current user/session ids can be logged by Nginx into the web server access logs. The feature is off by default and can be enabled via the `ACCOUNTS_USERINFO_HEADERS` configuration variable. Note: The upstream server should strip the two headers

from the response returned to the client. The purpose is purely to allow upstream proxies like Nginx to log the user/session id for a specific request.

- Changed token expiration from 5 days to 30 minutes for the password reset token and email confirmation token. Using the tokens will as a side-effect login in the user, which means that if the link is leaked (e.g. forwarded by the users themselves), then another person can use the link to access the account. Flask-Security v3.1.0 addresses this issue, but has not yet been released.
- Fixes issue that could rehash the user password in the adminstration interface.

Version 1.0.2 (released 2018-10-31)

- Added AnonymousIdentity loader to app initialisation to fix the `any_user` Need in Invenio-Access.

Version 1.0.1 (released 2018-05-25)

- Bumped Flask-CeleryExt from v0.3.0 to v0.3.1 to fix issue with Celery version string not being parsable and thus causing problems with installing Celery.

Version 1.0.0 (released 2018-03-23)

- Initial public release.

3.3 License

3.4 Contributors

- Alexander Ioannidis
- Alizee Pace
- Bruno Cuc
- Chiara Bigarella
- Diego Rodriguez
- Dinos Kousidis
- Eamonn Maguire
- Esteban J. G. Gabancho
- Harri Hirvonsalo
- Harris Tzovanakis
- Jacopo Notarstefano
- Javier Delgado
- Javier Martin Montull
- Jiri Kuncar
- Krzysztof Nowak
- Lars Holm Nielsen
- Leonardo Rossi
- Liam Kirsh
- Nicolas Harraudeau
- Odd Magnus Trondrud
- Orestis Melkonian
- Rémi Ducceschi
- Sami Hiltunen
- Sebastian Witowski
- Tibor Simko
- Maximilian Moser
- Mojib Wali

PYTHON MODULE INDEX

i

INDEX

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

W