
Sorcery: Magical Authentication



Magical Authentication for Rails. Supports ActiveRecord, DataMapper, Mongoid and MongoMapper.

Inspired by Restful Authentication, Authlogic and Devise. Crypto code taken almost unchanged from Authlogic. OAuth code inspired by OmniAuth and Ryan Bates's Railscast about it.

Philosophy

Sorcery is a stripped-down, bare-bones authentication library, with which you can write your own authentication flow. It was built with a few goals in mind:

- Less is more - less than 20 public methods to remember for the entire feature-set make the lib easy to 'get'.
- No built-in or generated code - use the library's methods inside *your own* MVC structures, and don't fight to fix someone else's.
- Magic yes, Voodoo no - the lib should be easy to hack for most developers.
- Configuration over Confusion - Centralized (1 file), Simple & short configuration as possible, not drowning in syntactic sugar.
- Keep MVC cleanly separated - DB is for models, sessions are for controllers. Models stay unaware of sessions.

Table of Contents

1. Useful Links
2. API Summary
3. Installation
4. Configuration
5. Full Features List by Module
6. Planned Features
7. Contributing
8. Contact
9. License

Useful Links

- [Documentation](#)

-
- Railscast
 - Simple tutorial
 - Example Rails app

Check out the tutorials in the wiki for more:

- DataMapper Support
- DelayedJob Integration
- Simple Password Authentication
- Single Table Inheritance Support
- Upgrading

API Summary

Below is a summary of the library methods. Most method names are self explaining and the rest are commented:

Core

```
1 require_login # This is a before action
2 login(email, password, remember_me = false)
3 auto_login(user) # Login without credentials
4 logout
5 logged_in? # Available in views
6 current_user # Available in views
7 redirect_back_or_to # Use when a user tries to access a page while
   logged out, is asked to login, and we want to return him back to the
   page he originally wanted
8 @user.external? # Users who signed up using Facebook, Twitter, etc.
9 @user.active_for_authentication? # Add this method to define behaviour
   that will prevent selected users from signing in
10 @user.valid_password?('secret') # Compares 'secret' with the actual
   user's password, returns true if they match
11 User.authenticates_with_sorcery!
```

HTTP Basic Auth

```
1 require_login_from_http_basic # This is a before action
```

External

```
1 login_at(provider) # Sends the user to an external service (Facebook,
   Twitter, etc.) to authenticate
2 login_from(provider) # Tries to login from the external provider's
   callback
3 create_from(provider) # Create the user in the local app database
4 build_from(provider) # Build user instance using user_info_mappings
```

Remember Me

```
1 auto_login(user, should_remember = false) # Login without credentials,
   optional remember_me
2 remember_me!
3 forget_me!
4 force_forget_me! # Forgets all sessions by clearing the token, even if
   remember_me_token_persist_globally is set to true
```

Reset Password

```
1 User.load_from_reset_password_token(token)
2 @user.generate_reset_password_token! # Use if you want to send the
   email by yourself
3 @user.deliver_reset_password_instructions! # Generates the token and
   sends the email
4 @user.change_password(new_password)
5 @user.change_password!(new_password) # Same as change_password but
   raises exception on save
```

Session Timeout

```
1 invalidate_active_sessions! #Invalidate all sessions with a login_time
   or last_action_time before the current time. Must Opt-in
```

User Activation

```
1 User.load_from_activation_token(token)
2 @user.setup_activation
3 @user.activate!
```

Please see the tutorials in the github wiki for detailed usage information.

Installation

Add this line to your application's Gemfile:

```
1 gem 'sorcery'
```

And then execute:

```
1 $ bundle
```

Or install it yourself as:

```
1 $ gem install sorcery
```

Configuration

Run the following command to generate the core migration file, the initializer file and the `User` model class.

```
1 $ rails generate sorcery:install
```

Run the following command generate the migrations files for `remember_me` and `reset_password` sub-modules and will create the initializer file (and add submodules to it), and create the `User` model class.

```
1 $ rails generate sorcery:install remember_me reset_password
```

Run the following command to generate the core migration file, the initializer and change the model class (in the initializer and migration files) to the class `Person` (and its pluralized version, 'people')

```
1 $ rails generate sorcery:install --model Person
```

Run the following command to generate only the migration files for the specified submodules and will add them to the initializer file.

```
1 $ rails generate sorcery:install http_basic_auth external remember_me  
   --only-submodules
```

Inside the initializer, the comments will tell you what each setting does.

Full Features List by Module

Core (see `lib/sorcery/model.rb` and `lib/sorcery/controller.rb`):

-
- Login / logout, optional return user to requested url on login, configurable redirect for non-logged-in users.
 - Password encryption, algorithms: bcrypt (default), MD5, SHA-1, SHA-256, SHA-512, AES or custom. Configurable stretches and salt.
 - Configurable attribute names for username, password and email.
 - Allow multiple fields to serve as username.

User Activation (see lib/sorcery/model/submodules/user_activation.rb):

- User activation by email with optional success email
- Configurable attribute names
- Configurable mailer, method name, and attribute name
- Configurable temporary token expiration
- Optionally prevent non-active users to login

Reset Password (see lib/sorcery/model/submodules/reset_password.rb):

- Reset password with email verification
- Configurable mailer, method name, and attribute name
- Configurable temporary token expiration
- Configurable time between emails (hammering protection)

Remember Me (see lib/sorcery/model/submodules/remember_me.rb):

- Remember me with configurable expiration
- Configurable attribute names
- Configurable to persist globally (supporting multiple browsers at the same time), or starting anew after each login

Session Timeout (see lib/sorcery/controller/submodules/session_timeout.rb):

- Configurable session timeout
- Optionally session timeout will be calculated from last user action
- Optionally enable a method to clear all active sessions, expects an `invalidate_sessions_before` datetime attribute.

Brute Force Protection (see lib/sorcery/model/submodules/brute_force_protection.rb):

- Brute force login hammering protection
- configurable logins before lock and lock duration

Basic HTTP Authentication (see lib/sorcery/controller/submodules/http_basic_auth.rb):

- A before action for requesting authentication with HTTP Basic

-
- Automatic login from HTTP Basic
 - Automatic login is disabled if session key changed

Activity Logging (see `lib/sorcery/model/submodules/activity_logging.rb`):

- Automatic logging of last login, last logout, last activity time and IP address for last login
- Configurable timeout by which to decide whether to include a user in the list of logged in users

External (see `lib/sorcery/controller/submodules/external.rb`):

- OAuth1 and OAuth2 support (currently: Twitter, Facebook, Github, Google, Heroku, LinkedIn, VK, LiveID, Xing, Salesforce)
- Configurable database column names
- Authentications table

Planned Features

- Passing a block to encrypt, allowing the developer to define his own mix of salting and encrypting
- Forgot username, maybe as part of the `reset_password` module
- Scoping logins (to a subdomain or another arbitrary field)
- Allowing storing the salt and encrypted password in the same DB field for extra security
- Other reset password strategies (security questions?)
- Other brute force protection strategies (captcha)

Have an idea? Let us know, and it might get into the gem!

Contributing

Bug reports and pull requests are welcome on GitHub at <https://github.com/Sorcery/sorcery>.

- Git Workflow
- Running the specs

Contact

Feel free to ask questions using these contact details:

Current Maintainers:

- Josh Buker (@joshbuker) | Email

Past Maintainers:

- Noam Ben-Ari (@NoamB) | Email | Twitter
- Kir Shatrov (@kirs) | Email | Twitter
- Grzegorz Witek (@arnvald) | Email | Twitter
- Chase Gilliam (@Ch4s3) | Email

License

The gem is available as open source under the terms of the MIT License.