HITOBITO CHEATSHEET

Version 21-06-08

Hitobito is made up of the core hitobito, plus organization-specific wagons such as hitobito_youth, hitobito_pbs, hitobito_cevi, ...

Core

Normal Ruby on Rails project. All common functionality is defined here.

Wagons

Organization-specific adaptations, such as group and role types, features, and exports

Routes config/routes.rb

Define which URLs are available in the app and connect them to controllers and actions within them. E.g. PeopleController#create

✓ Add more routes
✓ Overwrite core routes (one route at a time)

Controllers app/controllers/*.rb

Define what should happen when a URL is called, and render appropriate HTML through the views.

✓ Adapt core controllers via lib/hitobito_*/wagon.rb
✓ Add new controllers

Views app/views/*.haml

More or less HTML code with placeholders, which are filled with data from the database and with translated texts.

✓ Add HTML in render_extensions points
✓ Add new news
✓ Overwrite core views (one file at a time)

Translations config/locales/*.yml

Each translation has a technical “key” which is structured according to the place of use. E.g. de.events.form_tabs.general

✓ Overwrite core translations (one key at a time)
✓ Neue Übersetzungen hinzufügen

Models app/models/*.rb

Define which objects can be saved in hitobito, which relations between them exist (e.g. a person has roles, a role belongs to a group) and validations (e.g. name may not be empty).

✓ Adapt core models via lib/hitobito_*/wagon.rb
✓ Add new models

Decorators app/decorators/*.rb

“Wrapper” around a model, contains model-specific calculations, e.g. full_name = first_name + last_name

✓ Adapt core decorators via lib/hitobito_*/wagon.rb
✓ Add new decorators

Helpers app/helpers/*.rb

In places where there would be too much logic in the views, we put that into a helper. E.g. reusable role display incl. layer and description.

✓ Add new helpers
✓ Overwrite core helpers (one method at a time)

Domain classes app/domain/*.rb

For more complex calculations and business logic. E.g. PDF export, doublette check, fusing groups, filtering person lists

✓ Adapt core domain classes via lib/hitobito_*/wagon.rb
✓ Add new domain classes

Specs (unit tests) spec/*.rb

Automated tests for anything that might conceivably break in the future

✓ Add new specs

More

Abilities (permissions)	app/abilities/*.rb	✓ Add new abilities and overwrite existing ones
CSS styles	app/javascript/stylesheets/*.css	✓ Add new CSS styles
JavaScript	app/javascript/javascripts/*.coffee	✓ Add new JavaScript scripts
Database structure changes	db/migrate/*.rb	✓ Add new migrations
Config files	config/*.rb	✓ Overwrite intended configuration