

1 Class: AppConfig

AppConfig objects store system specific configuration values for the application.

1.1 Reading

Read any value from the database (or the default config yaml, if it is not yet in the db) by using the `[]` accessor.

```
AppConfig[:language] # => "en"
```

1.2 Storing

Store any value in the database by using the `[] =` accessor.

```
AppConfig[:language] = 'de' # => "de"  
AppConfig[:some_option] = 'some value' # => "some value"
```

1.2.1 Public Class methods

AppConfig[key] # => Object

Returns the value stored for key in the database or its default value.

```
AppConfig[:language] # => "en"
```

AppConfig[key] = value

Stores the value for key in the database.

```
AppConfig[:language] = 'de' # => "de"
```

2 Class: ApplicationController

2.0.2 Private Class methods

login_required(*opts* = {})

Use this in a controller to restrict access.

```
class UsersController < ApplicationController
  login_required :only => [:edit, :update, :show]
end
```

ownership_required(*opts* = {})

Use this in a controller to restrict access to owners.

role_or_ownership_required(*roles*, *opts* = {})

Use this in a controller to restrict access to either users of certain roles (e.g. admins) or the rightful owner of an object.

```
class PostingController < ApplicationController
  role_or_ownership_required [:posting_admin, :administrator]
end
```

role_required(*roles*, *opts* = {})

Use this in a controller to restrict access to either users of certain roles (e.g. admins).

```
class Admin::BaseController < ApplicationController
  role_required :administrator
end
```

same_company_required(*opts* = {})

2.0.3 Private Instance methods

current_company()

Returns the Company object of the currently logged in user or `nil` if no user is logged in.

current_person()

Returns the Person object of the currently logged in user or `nil` if no user is logged in.

current_user()

Returns the User object of the currently logged in user or `nil` if no user is logged in.

demo_mode?()

Returns `true` if the application is running in demo mode.

3 Module: ApplicationHelper

3.0.4 Public Instance methods

clear_both()

Returns a DIV tag that clears floating.

collection_choices(*model*, *attribute_name*, *const = nil*)

Returns the collection of localized choices for a given attribute. Example:

```
collection_choices(Person, :gender)
```

This will look up `Person::GENDER_CHOICES` and return the keys and localized values.

controller?(name) # => boolean

Returns if `c` is the current controller. Example:

```
<%= controller?(:root) %>
# => true
```

format_multiline_input(*text*)

Returns a HTML formatted version of `text`. Example:

```
<%= format_multiline_input("First line.\nSecond Line.") %>
# => "First line.\nSecond line."
```

link_back(*text = t("common.link_back")*)

Returns a link back to the last visited page with a localized caption.

link_to_unless(*condition*, *name*, *options = {}*, *html_options = {}*, *&block*)

TODO: lookup rails3 implementation

localized_info(*obj*, *name*, *lang = I18n.default_locale*)

Returns a formatted string for the associated LocalizedInfo object.

localized_info_field(*f*, *name*, *lang*)

```
TODO: localized_info_field f, :type_of_goods, :en
```

```
BETTA: f.localized_info_field :type_of_goods, :en
```

only_some_attributes_filled?(*ar*)

render_company_info(*company*)

Renders a partial with the contact information for the given company. Example:

```
<%= render_person_info current_company %>
```

render_person_info(*person*)

Renders a partial with the contact information for the given person. Example:

```
<%= render_person_info current_person %>
```

render_table(*arel*)

Renders a table for the given ActiveRecord. Example:

```
<%= render_table User.all %>
```

yes_no(*condition*)

4 Class: CompaniesController

4.0.5 Public Instance methods

create()

dashboard()

new()

The Companies#new action is actually the "Create a new Accountscreen a user sees when he signs up for the freight exchange.

show()

5 Module: CompaniesHelper

5.0.6 Public Instance methods

`registering_new_account?()`

6 Class: Company

Companies are organising Users.

6.0.7 Public Instance methods

ensure_admin()

Ensures there is atleast one :company_admin left. If no admin can be found, the first user of the company is assigned the admin role.

7 Class: Country

8 Class: Freight

8.0.8 Public Instance methods

localized__info(*name*, *lang* = *I18n.default__locale*)

localized__infos=(*array__of__options*)

to__search()

update__localized__infos()

9 Class: FreightController

9.0.9 Public Instance methods

create()

new()

update()

10 Module: FreightsHelper

11 Class: GeneralObserver

12 Class: LoadingSpace

12.0.10 Public Instance methods

localized__info(*name*, *lang* = *I18n.default__locale*)

localized__infos=(*array__of__options*)

to__search()

update__localized__infos()

13 Class: LoadingSpacesController

13.0.11 Public Instance methods

create()

new()

update()

14 Module: LoadingSpacesHelper

15 Class: LocalizedInfo

15.0.12 Public Instance methods

`update_or_destroy!()`

16 Class: Matching::Compare::Base

Compare objects compare two objects A and B based on their type/class.

16.1 Creation

Compare objects accept two constructor parameters for the A and the B object.

```
compare = Compare::String.new('one string', 'another string')
compare.result # => 0.6428...
```

16.2 Conditions

By default, a compare object compares copies of the entire objects it is passed. It is also possible to only compare certain attributes of an object.

```
class UserComparer < Matching::Compare::Base
  compare :gender, :weight
end
```

Thresholds can be used to ensure that only objects who meet certain criteria are considered alike.

```
class UserComparer < Matching::Compare::Base
  compare :weight, :threshold => 10
  # => User A can be 10 kilos heavier or lighter than user B

  compare :weight, :threshold => 0.05
  # => User A can be 5% heavier or lighter than user B

  compare :weight, :threshold => {:up => 0, :down => 0.1}
  # => User A can be 10% lighter than user B, but not any heavier.

  compare :weight, :threshold => :perfect
  # => User A and B have to have the same weight
end
```

All object-pairs not meeting the threshold criteria are automatically assigned a result of 0.0 (not matching at all).

16.3 Overwriting defaults

Blocks can be used to override the default comparisons.

Example:

```
class UserCompanyComparer < Matching::Compare::Base
  # Do not compare the email with the default String processor
  # but compare the email hosts and eliminate the pair if they
  # are not matching.
  compare :email do |a, b|
    email_domain = /[~@]+$/
    a[email_domain] == b[email_domain]
  end
end
```

16.3.1 Public Class methods

compare(*attributes, options = {}, &block)

Specifies one or more attribute(s) that will be compared using the defined options and the block, if given.

Options

- **:as** - A Symbol identifying the Comparer class to be used

(e.g. `:String`, `:Time` etc.)

```
class UserComparer < Matching::Compare::Base
  compare :created_at, :as => :Time
end
```

- **:threshold** - If the attribute of the B object differs more

than the given threshold the comparison fails, resulting in a 0.0 match. `:up` and `:down` options are available as well. Floats are interpreted as relative, Fixnums as absolute thresholds.

```
class UserComparer < Matching::Compare::Base
  compare :weight, :threshold => 10
  # => User A can be 10 kilos heavier or lighter than user B
```

```

    compare :weight, :threshold => 0.05
    # => User A can be 5% heavier or lighter than user B

    compare :weight, :threshold => {:up => 0, :down => 0.1}
    # => User A can be 10% lighter than user B, but not any heavier

    compare :weight, :threshold => :perfect
    # => User A and B have to have the same weight
end

```

Block evaluation If a block is given, the compared attributes are passed and the result of the block is the final result for the comparison (with `true` being interpreted as 1.0).

```

class UserComparer < Matching::Compare::Base
  compare :email do |a, b|
    email_domain = /[~@]+$/
    a[email_domain] == b[email_domain]
  end
end

```

new(a, b)

Create a new Compare object to compare the given objects.

16.3.2 Public Instance methods

result()

Compares two objects and returns a result between 0.0 (not alike) and 1.0 (perfect match).

Examples:

```

Comparer::Base.new(true, false) # => 0.0
Comparer::Base.new(true, true) # => 1.0

```

16.3.3 Protected Instance methods

calc_result(*hsh*)

compare_attribute(*attr*, *opts* = {})

compare_attributes_and_calc_result()

compared_attributes()

comparer_for(*klass*)

in_threshold(*x*, *y*, *result*, *threshold* = {})

Floats are interpreted as relative, Fixnums as absolute thresholds.

17 Class: Matching::Compare::Fixnum

Compares two fixnum objects.

17.0.4 Public Instance methods

result()

18 Class: Matching::Compare::Hash

Compares to two hashes by comparing all values of hash A with their counterparts in hash B.

18.0.5 Public Instance methods

result()

19 Class: Matching::Compare::String

Compares two strings using Levenshtein distance.

19.0.6 Public Instance methods

result()

20 Class: Matching::Compare::Time

Compares two time objects.

20.0.7 Public Instance methods

result()

21 Module: Matching::Compare

The Compare module provides a set of classes and methods to match objects like Strings, Numbers and Dates.

22 Module: Matching

The Matching module provides a set of classes and methods to match objects. On top of this, it provides an extendable generic API for matching Freight and LoadingSpace objects (see `compare_freight_and_loading_space` method).

23 Class: Object

23.0.8 Public Instance methods

obj.full? obj.full? { |f| ... }

Returns wheter or not the given obj is not blank?. If a block is given and the obj is full?, the obj is yielded to that block.

```
salary = nil
salary.full? { |s| "#{s} $" } # => nil
salary = 100
salary.full? { |s| "#{s} $" } # => "100 $"
```

With Rails' implementation of Symbol`#to_proc` it is possible to write:

```
current_user.full?(&:name) # => "Dave"
```

24 Class: PeopleController

25 Module: PeopleHelper

26 Class: Person

Person objects contain personal information about a User.

26.0.9 Public Instance methods

name()

TODO: Anrede?

27 Class: Posting

27.0.10 Public Instance methods

`to_search()`

`validate()`

28 Class: PostingsController

29 Module: PostingsHelper

30 Class: Recording

31 Class: Region

32 Class: RootController

32.0.11 Public Instance methods

about()

index()

welcome()

This action decides what to do with a freshly logged in user.

33 Module: RootHelper

34 Class: SearchController

34.0.12 Public Instance methods

`index()`

35 Module: SearchHelper

36 Class: SiteInfo

SiteInfo objects contain information about loading and unloading sites, such as name of the site, address of the site, name of the contractor etc.

37 Class: Station

37.0.13 Public Instance methods

`to_search()`

38 Class: StationsController

39 Module: StationsHelper

40 Class: User

User objects represent a user of the system and are used to authenticate users upon login (using `acts_as_authentic` plugin) and handle permission handling via assigned `UserRole` objects.

Data concerning the actual, human user (like company, gender, language etc.) is stored in associated `Person` and `Company` objects.

40.0.14 Public Instance methods

`user.has_role?(role_name) # => boolean`

Returns true if a user has a `UserRole` with the given `name`.

```
user.has_role?(:administrator) # => true
```

`is?(name)`

Alias for `has_role?`

`user.roles # => array`

Returns an array of role names.

```
user.roles # => ["administrator", "company_admin"]
```

41 Class: UserRole

UserRoles grant a logged in User access to certain parts of the application.

41.1 Creation

UserRoles are created and identified via their `:name` attribute.

```
UserRole.create(:name => 'employee_of_the_month')
```

41.2 Find by name

UserRoles can be found via their `:name` attribute using the `[]` accessor.

```
UserRole[:employee_of_the_month]
```

41.3 Assigning

Finally, UserRoles can be assigned to a User with the `<<` operator.

```
user.user_roles << UserRole[:employee_of_the_month]
```

To access the backend e.g. a user must have administrator privileges:

```
user.user_roles << UserRole[:administrator]
```

This is also used in the frontend to restrict the privileges of users in companies.

```
user.user_roles << UserRole[:company_admin]
```

42 Class: UsersController

42.0.1 Public Instance methods

create()

This creates a new user inside the current company. For the original sign up screen, see `Companies#new`.

index()

Lists all users in the current company.

43 Class: UserSession

43.0.2 Public Class methods

UserSession.login(user) # => boolean

Authenticates a user and logs him in.

```
UserSession.login(User.first) # => true
```


44 Class: UserSessionsController

The UserSessionsController handles all requests regarding logging in and out.

44.0.3 Public Instance methods

create()

Authenticates a User by creating and saving a UserSession.

demo_login()

This action is only available if the application is running in demo mode. It creates a UserSession for a given user without any authentication.

destroy()

Logs a user out.

new()

45 Module: UserSessionsHelper

46 Module: UsersHelper