Dizzy Making business beautiful.

Cheatsheet * Errors

Read methods

◆count

Alias for `size`

◆each

Yields each attribute and associated message per error added.

```
class Company < ActiveRecord::Base
  validates_presence_of :name, :address, :email
  validates_length_of :name, :in => 5..30
end
company = Company.create(:address => '123 First St.')
company.errors.each{|attr,msg| puts "#{attr} - #{msg}" }
  # => name - is too short (minimum is 5 characters)
  # => name - can't be blank
  # => address - can't be blank
```

◆each_full

Yields each full error message added. So `Person.errors.add("first_name", "can't be empty")` will be returned through iteration as `"First name can't be empty"`.

```
class Company < ActiveRecord::Base
  validates_presence_of :name, :address, :email
  validates_length_of :name, :in => 5..30
end
company = Company.create(:address => '123 First St.')
company.errors.each_full{|msg| puts msg }
  # => Name is too short (minimum is 5 characters)
  # => Name can't be blank
  # => Address can't be blank
```

◆full_messages

Returns all the full error messages in an array.

```
class Company < ActiveRecord::Base
  validates_presence_of :name, :address, :email
  validates_length_of :name, :in => 5..30
end
company = Company.create(:address => '123 First St.')
company.errors.full_messages
  # => ["Name is too short (minimum is 5 characters)", "Name can't be blank", "Address can't be blank"]
```

◆empty?

Returns `true` if no errors have been added.

◆length

Alias for `size`

◆on

Returns `nil`, if no errors are associated with the specified attribute. Returns the error message, if one error is associated with the specified attribute. Returns an array of error messages, if more than one error is associated with the specified attribute.

```
class Company < ActiveRecord::Base
  validates_presence_of :name, :address, :email
  validates_length_of :name, :in => 5..30
end
company = Company.create(:address => '123 First St.')
company.errors.on(:name)
  # => ["is too short (minimum is 5 characters)", "can't be blank"]
company.errors.on(:email)
  # => "can't be blank"
company.errors.on(:address)   # => nil
```

This method is also aliased as the shortcut `[]`

◆on_base

Returns errors that have been assigned to the base object through `add_to_base` according to the normal rules of `on(attribute)`.

◆invalid? *(attribute)*

Returns `true` if the specified *attribute* has errors associated with it.

```
class Company < ActiveRecord::Base
  validates_presence_of :name, :address, :email
  validates_length_of :name, :in => 5..30
end
company = Company.create(:address => '123 First St.')
company.errors.invalid?(:name)      # => true
company.errors.invalid?(:address)   # => false
```

◆size

Returns the total number of errors added. Two errors added to the same attribute will be counted as such.

◆to_xml(*options={}*)

Returns an XML representation of this error object.

◆[] *(attribute)*

Alias for on method...

```
company.errors[:email]
```

View Helpers

◆error_message_on

(object, attribute, prepend_text = "", append_text = "", css_class = "formError")

Returns a string containing the error message attached to the attribute of the object if one exists. This error message is wrapped in a `<div>` tag, which can be extended to include a `prepend_text` and/or `append_text` (to properly explain the error), and a `css_class` to style it accordingly. *Object* should either be the name of an instance variable or the actual object itself. As an example, let's say you have a model `@post` that has an error message on the `title` attribute:

```
<%= error_message_on "post", "title" %>
# => <div class="formError">can't be empty</div>
<%= error_message_on @post, "title" %>
# => <div class="formError">can't be empty</div>
<%= error_message_on "post", "title", "Title simply ", " (or it won't work).", "inputError" %>
# => <div class="inputError">Title simply can't be empty (or it won't work).</div>
```

object	the name of an @instance_variable or the actual object
attribute	the attribute you wish to check for errors
prepend_text	text to be prepended to the error message
append_text	text to be appended to the error message
css_class	CSS class of the `<div>` which will wrap the error message

◆error_messages_for(*{hash}*)

Returns a string with a `<div>` containing all of the error messages for the objects located as instance variables by the names given. If more than one object is specified, the errors for the objects are displayed in the order that the object names are provided.

This `<div>` can be tailored by the following options:

`:header_tag`	Used for the header of the error `<div>` (default is `h2`)
`:id`	The class of the error `<div>` (default is `errorExplanation`)
`:class`	The id of the error `<div>` (default is `errorExplanation`)
`:object`	The object (or array of objects) for which to display errors, if you need to escape the instance variable convention
`:object_name`	The object name to use in the header, or any text that you prefer. If `:object_name` is not set, the name of the first object will be used
`:header_message`	The message in the header of the error `<div>`. Pass `nil` or an empty string to avoid the header message altogether (default message is "*X errors prohibited this object from being saved*")
`:message`	The explanation message after the header message and before the error list. Pass `nil` or an empty string to avoid the explanation message altogether (default message is *"There were problems with the following fields:"*)

To specify the display for one object, you simply provide its name as a parameter. For example, for the `@user` model:

```
error_messages_for :user
```

To specify more than one object, you simply list them: optionally, you can add an extra :object_name parameter, which will be the name used in the header message:

```
error_messages_for :user_common, :user, :object_name => :user
```

If the objects cannot be located as instance variables, you can add an extra :object paremeter which gives the actual object (or array of objects to use):

```
error_messages_for :user, :object => @question.user
```

This is a pre-packaged presentation of the errors with embedded strings and a certain HTML structure. If what you need is significantly different from the default presentation, it makes plenty of sense to access the *object.errors* instance yourself and set it up.

Default error messages

These error messages are stored in a Rails class variable, `@@default_error_messages` and can be changed or added to as follows:

```
ActiveRecord::Errors.default_error_messages[:blank] = "Your custom message here"
```

These default error messages are used by Rails' built in validation class methods and some of the Errors write methods such as **add_on_blank**. You may find it useful to change them if, for example, you require your error messages in a different language.

`:inclusion`	"is not included in the list"
`:exclusion`	"is reserved"
`:invalid`	"is invalid"
`:confirmation`	"doesn't match confirmation"
`:accepted`	"must be accepted"
`:empty`	"can't be empty"
`:blank`	"can't be blank"
`:too_long`	"is too long (maximum is %d characters)"
`:too_short`	"is too short (maximum is %d characters)"
`:wrong_length`	"is the wrong length (should be %d characters)"
`:taken`	"has already been taken
`:not_a_number`	"is not a number
`:greater_than`	"must be greater than %d"
`:greater_than_or_equal_to`	"must be greater than or equal to %d"
`:equal_to`	"must be equal to %d"
`:less_than`	"must be less than %d"
`:less_than_or_equal_to`	"must be less than or equal to %d"
`:odd`	"must be odd"
`:even`	"must be even"

Write methods

◆add (*attribute, msg = @@default_error_messages[:invalid]*)

Adds an error message *msg* to the *attribute*, which will be returned on a call to `on(attribute)` for the same attribute and ensure that this error object returns `false` when asked if `empty?`. More than one error can be added to the same attribute in which case an array will be returned on a call to `on(attribute)`. If no *msg* is supplied, `"invalid"` is assumed.

◆add_on_blank *([attributes], msg = @@default_error_messages[:blank])*

Will add an error message to each of the attributes in *[attributes]* that is blank (for example, an empty string).

◆add_on_empty *(attributes, msg = @@default_error_messages[:empty])*

Will add an error message to each of the attributes in *attributes* that is empty.

◆add_to_base *(attributes, msg = @@default_error_messages[:empty])*

Adds an error to the base object instead of any particular attribute. This is used to report errors that don't tie to any specific attribute, but rather to the object as a whole. These error messages don't get prepended with any field name when iterating with `each_full`, so they should be complete sentences.

◆clear

Removes all the errors that have been added to the object.