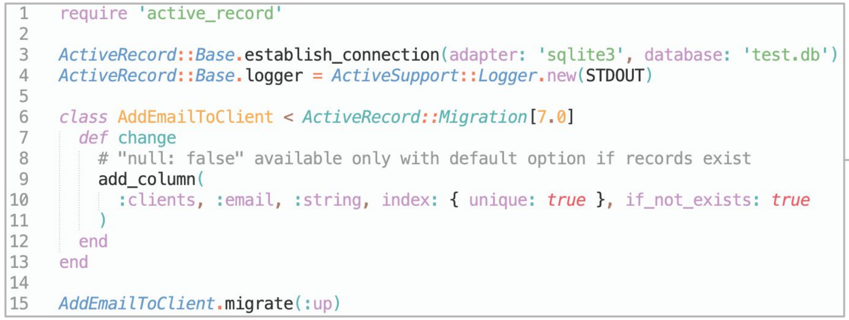
Жизненный цикл моделей

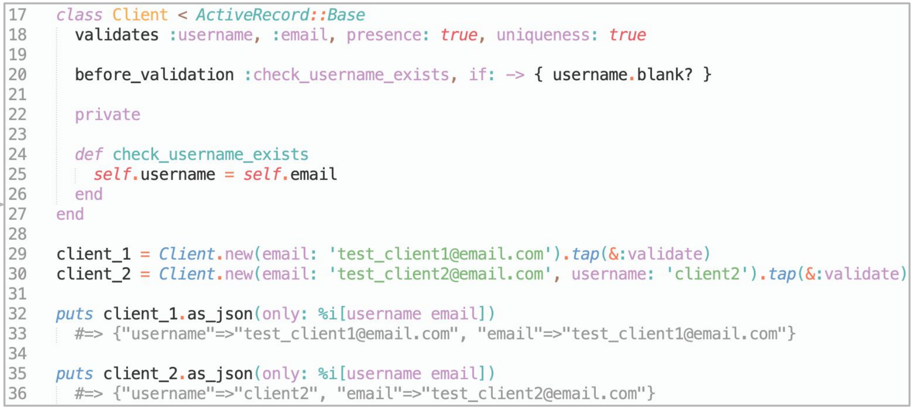
Жизненный цикл, создание callback-ов

Жизненный цикл объекта

В Rails приложении жизненный цикл объекта модели состоит из операций валидации, создания, обновления и удаления данного объекта. Для выполнения дополнительных операций используются callback методы.



```ruby
require 'active_record'

ActiveRecord::Base.establish_connection(adapter: 'sqlite3', database: 'test.db')
ActiveRecord::Base.logger = ActiveSupport::Logger.new(STDOUT)

class AddEmailToClient < ActiveRecord::Migration[7.0]
  def change
    # "null: false" available only with default option if records exist
    add_column(
      :clients, :email, :string, index: { unique: true }, if_not_exists: true
    )
  end
end

AddEmailToClient.migrate(:up)
```



```ruby
class Client < ActiveRecord::Base
  validates :username, :email, presence: true, uniqueness: true

  before_validation :check_username_exists, if: -> { username.blank? }

  private

  def check_username_exists
    self.username = self.email
  end
end

client_1 = Client.new(email: 'test_client1@email.com').tap(&:validate)
client_2 = Client.new(email: 'test_client2@email.com', username: 'client2').tap(&:validate)

puts client_1.as_json(only: %i[username email])
  #=> {"username"=>"test_client1@email.com", "email"=>"test_client1@email.com"}

puts client_2.as_json(only: %i[username email])
  #=> {"username"=>"client2", "email"=>"test_client2@email.com"}
```

Доступные callback методы

Создание объекта	Обновление объекта	Удаление объекта
1.before_validation	1.before_validation	1.before_destroy
2.after_validation	2.after_validation	2.around_destroy
3.before_save	3.before_save	3.after_destroy
4.around_save	4.around_save	4.after_commit / after_rollback
5.before_create	5.before_update	
6.around_create	6.around_update	
7.after_create	7.after_update	
8.after_save	8.after_save	
9.after_commit / after_rollback	9.after_commit / after_rollback	

Также доступны такие callback-и как ***after_initialize***, ***after_find*** и ***after_touch***.

Запуск callback методов

Методы, которые запускают callback-и

- create
- create!
- destroy
- destroy!
- destroy_all
- destroy_by
- save
- save!
- save(validate: false)
- toggle!
- touch
- update_attribut
- update
- update!
- valid?

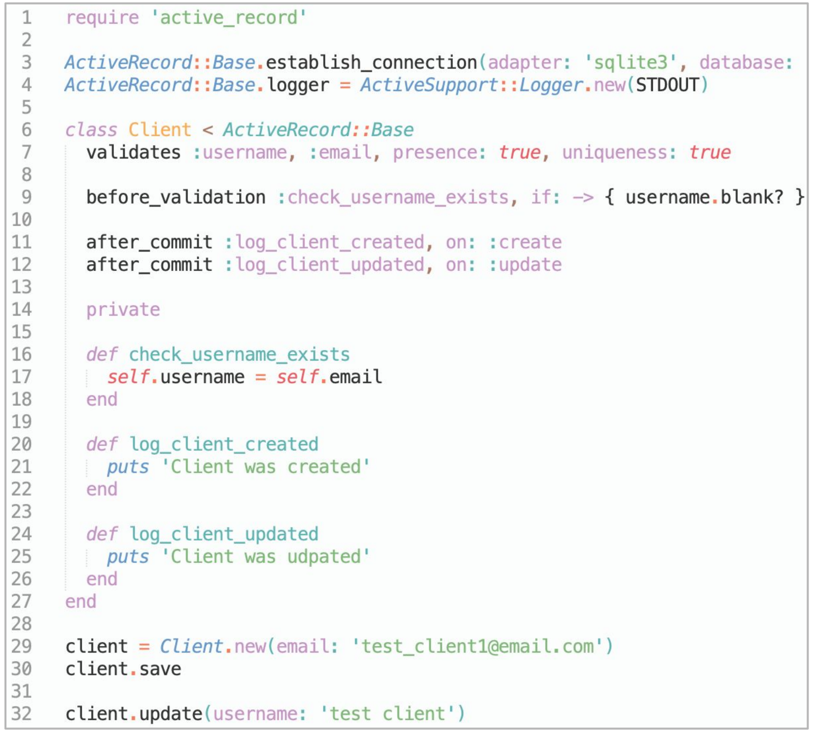
Методы, которые запускают *after_find* callback

- all
- first
- find
- find_by
- last

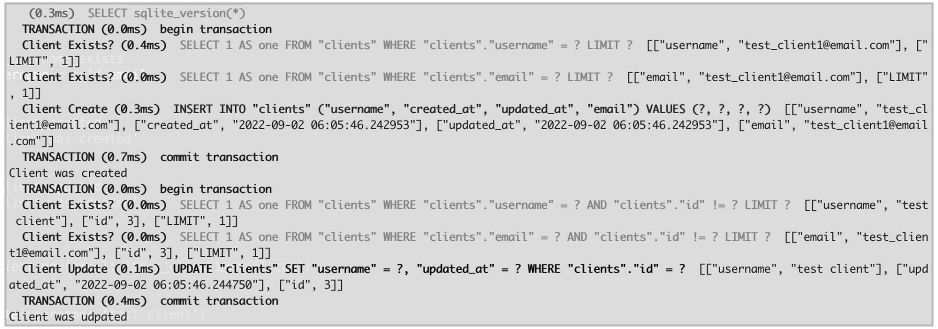
Методы, которые пропускают callback-и

- decrement!
- decrement_counter
- delete
- delete_all
- delete_by
- increment!
- increment_counter
- insert
- insert!
- insert_all
- insert_all!
- touch_all
- update_column
- update_columns
- update_all
- update_counters
- upsert
- upsert_all

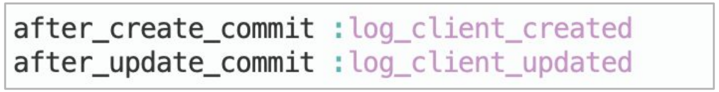
Пример after_commit callback-а



```ruby
require 'active_record'

ActiveRecord::Base.establish_connection(adapter: 'sqlite3', database:
ActiveRecord::Base.logger = ActiveSupport::Logger.new(STDOUT)

class Client < ActiveRecord::Base
  validates :username, :email, presence: true, uniqueness: true

  before_validation :check_username_exists, if: -> { username.blank? }

  after_commit :log_client_created, on: :create
  after_commit :log_client_updated, on: :update

  private

  def check_username_exists
    self.username = self.email
  end

  def log_client_created
    puts 'Client was created'
  end

  def log_client_updated
    puts 'Client was udpated'
  end
end

client = Client.new(email: 'test_client1@email.com')
client.save

client.update(username: 'test client')
```

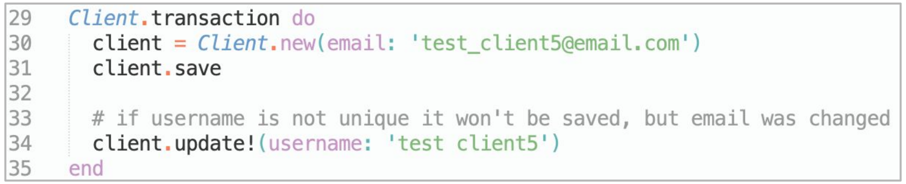


```
   (0.3ms)  SELECT sqlite_version(*)
  TRANSACTION (0.0ms)  begin transaction
  Client Exists? (0.4ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."username" = ? LIMIT ?  [["username", "test_client1@email.com"], ["
LIMIT", 1]]
  Client Exists? (0.0ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."email" = ? LIMIT ?  [["email", "test_client1@email.com"], ["LIMIT"
, 1]]
  Client Create (0.3ms)  INSERT INTO "clients" ("username", "created_at", "updated_at", "email") VALUES (?, ?, ?, ?)  [["username", "test_cl
ient1@email.com"], ["created_at", "2022-09-02 06:05:46.242953"], ["updated_at", "2022-09-02 06:05:46.242953"], ["email", "test_client1@email
.com"]]
  TRANSACTION (0.7ms)  commit transaction
Client was created
  TRANSACTION (0.0ms)  begin transaction
  Client Exists? (0.0ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."username" = ? AND "clients"."id" != ? LIMIT ?  [["username", "test
 client"], ["id", 3], ["LIMIT", 1]]
  Client Exists? (0.0ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."email" = ? AND "clients"."id" != ? LIMIT ?  [["email", "test_clien
t1@email.com"], ["id", 3], ["LIMIT", 1]]
  Client Update (0.1ms)  UPDATE "clients" SET "username" = ?, "updated_at" = ? WHERE "clients"."id" = ?  [["username", "test client"], ["upd
ated_at", "2022-09-02 06:05:46.244750"], ["id", 3]]
  TRANSACTION (0.4ms)  commit transaction
Client was udpated
```

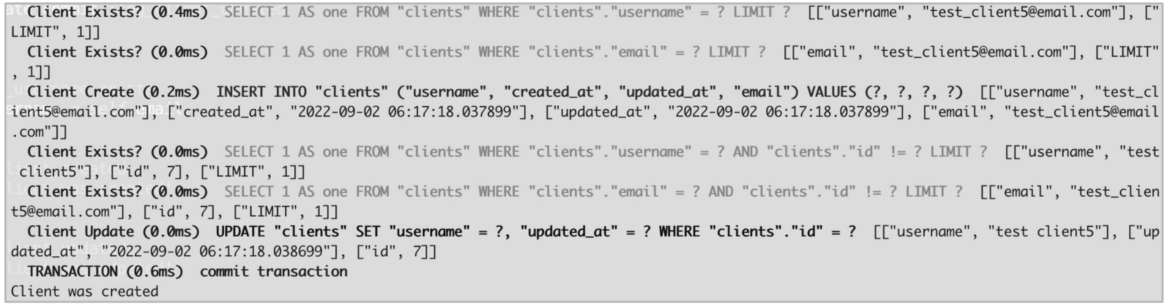


```ruby
after_create_commit :log_client_created
after_update_commit :log_client_updated
```

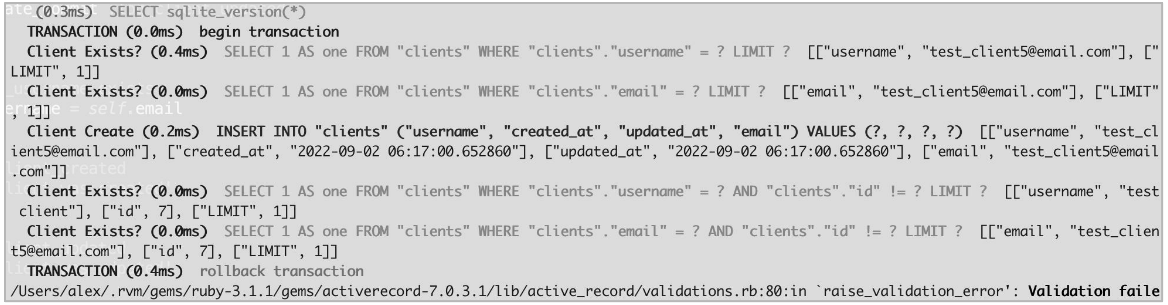
Транзакция и callback-и



```ruby
Client.transaction do
  client = Client.new(email: 'test_client5@email.com')
  client.save

  # if username is not unique it won't be saved, but email was changed
  client.update!(username: 'test client5')
end
```



```
  Client Exists? (0.4ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."username" = ? LIMIT ?  [["username", "test_client5@email.com"], ["
LIMIT", 1]]
  Client Exists? (0.0ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."email" = ? LIMIT ?  [["email", "test_client5@email.com"], ["LIMIT"
, 1]]
  Client Create (0.2ms)  INSERT INTO "clients" ("username", "created_at", "updated_at", "email") VALUES (?, ?, ?, ?)  [["username", "test_cl
ient5@email.com"], ["created_at", "2022-09-02 06:17:18.037899"], ["updated_at", "2022-09-02 06:17:18.037899"], ["email", "test_client5@email
.com"]]
  Client Exists? (0.0ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."username" = ? AND "clients"."id" != ? LIMIT ?  [["username", "test
 client5"], ["id", 7], ["LIMIT", 1]]
  Client Exists? (0.0ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."email" = ? AND "clients"."id" != ? LIMIT ?  [["email", "test_clien
t5@email.com"], ["id", 7], ["LIMIT", 1]]
  Client Update (0.0ms)  UPDATE "clients" SET "username" = ?, "updated_at" = ? WHERE "clients"."id" = ?  [["username", "test client5"], ["up
dated_at", "2022-09-02 06:17:18.038699"], ["id", 7]]
  TRANSACTION (0.6ms)  commit transaction
Client was created
```



```
   (0.3ms)  SELECT sqlite_version(*)
  TRANSACTION (0.0ms)  begin transaction
  Client Exists? (0.4ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."username" = ? LIMIT ?  [["username", "test_client5@email.com"], ["
LIMIT", 1]]
  Client Exists? (0.0ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."email" = ? LIMIT ?  [["email", "test_client5@email.com"], ["LIMIT"
, 1]]
  Client Create (0.2ms)  INSERT INTO "clients" ("username", "created_at", "updated_at", "email") VALUES (?, ?, ?, ?)  [["username", "test_cl
ient5@email.com"], ["created_at", "2022-09-02 06:17:00.652860"], ["updated_at", "2022-09-02 06:17:00.652860"], ["email", "test_client5@email
.com"]]
  Client Exists? (0.0ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."username" = ? AND "clients"."id" != ? LIMIT ?  [["username", "test
 client"], ["id", 7], ["LIMIT", 1]]
  Client Exists? (0.0ms)  SELECT 1 AS one FROM "clients" WHERE "clients"."email" = ? AND "clients"."id" != ? LIMIT ?  [["email", "test_clien
t5@email.com"], ["id", 7], ["LIMIT", 1]]
  TRANSACTION (0.4ms)  rollback transaction
/Users/alex/.rvm/gems/ruby-3.1.1/gems/activerecord-7.0.3.1/lib/active_record/validations.rb:80:in `raise_validation_error': Validation faile
```

Полезные ссылки

https://guides.rubyonrails.org/active_record_callbacks.html – Описание callback методов;

https://www.honeybadger.io/blog/database-transactions-rails-activerecord/ – Примеры работы с транзакциями.

Конец! Спасибо!