# Работа с моделями в Rails

Active Model, Active Record

# Превращение класса в модель

Active Model − библиотека, представляющая собой набор модулей с функционалом, представленным в Active Record. Позволяет многократно использовать функционал для валидации, сериализации объектов.

```ruby
class Product
  attr_reader :id
  attr_accessor :name, :amount, :price

  def initialize(id:, name:, amount:, price:)
    @id = id
    @name = name
    @amount = amount
    @price = price
  end

  def valid?
    id && name && amount.to_i > 0 && price.to_i > 100 && price.to_i < 2000
  end
end

i_phone = Product.new(id: 1, name: 'iPhone 13 Pro', amount: 0, price: 1000)
mac_book = Product.new(id: 2, name: 'MacBook Pro 13', amount: 5, price: 1300)

puts i_phone.id, i_phone.valid? #=> 1, false
puts mac_book.name, mac_book.valid? #=> MacBook Pro 13, true
```

```ruby
require 'active_model'

class Product
  include ActiveModel::Model

  attr_accessor :name, :amount, :price

  def valid?
    name && amount.to_i > 0 && price.to_i > 100 && price.to_i < 2000
  end

  def to_s
    { name:, amount:, price: }.to_s # Ruby 3.1.0+ syntax
  end
end

i_phone = Product.new(name: 'iPhone 13 Pro', amount: 0, price: 1000)
mac_book = Product.new(name: 'MacBook Pro 13', amount: 5, price: 1300)

puts i_phone, i_phone.valid?
#=> {:name=>"iPhone 13 Pro", :amount=>0, :price=>1000}, false
puts mac_book, mac_book.valid?
#=> {:name=>"MacBook Pro 13", :amount=>5, :price=>1300}, true
```

# Валидация полей модели

```ruby
require 'active_model'

class Product
  include ActiveModel::Model

  attr_accessor :name, :amount, :price

  validates :name, :amount, :price, presence: true
  validates :name, length: { in: 4..40 }
  validates :price, numericality: { greater_than_or_equal_to: 100 }
  validates :amount, numericality: { greater_than: 0, only_integer: true }

  def to_s
    { name:, amount:, price: }.to_s # Ruby 3.1.0+ syntax
  end
end

i_phone = Product.new(name: 'iPhone 13 Pro', amount: 0, price: 1000)
mac_book = Product.new(name: 'MacBook Pro 13', amount: 5, price: 1300)

puts i_phone, i_phone.valid?
#=> {:name=>"iPhone 13 Pro", :amount=>0, :price=>1000}, false
puts mac_book, mac_book.valid?
#=> {:name=>"MacBook Pro 13", :amount=>5, :price=>1300}, true

mac_book.name = 'Mac'
puts mac_book.valid?, mac_book.invalid? #=> false, true
```

```ruby
class Person
  include ActiveModel::Validations

  attr_accessor :first_name, :last_name

  validates_each :first_name, :last_name do |record, attr, value|
    record.errors.add attr, "starts with z." if value.start_with?("z")
  end
end
```

Which provides you with the full standard validation stack that you know from Active Record:

```ruby
person = Person.new
person.valid?                  # => true
person.invalid?                # => false

person.first_name = 'zoolander'
person.valid?                  # => false
person.invalid?                # => true
person.errors.messages         # => {first_name:["starts with z."]}
```

# Сериализация модели

```ruby
require 'active_model'

class Product
  include ActiveModel::Model
  include ActiveModel::Serializers::JSON

  attr_accessor :name, :amount, :price

  validates :name, :amount, :price, presence: true
  validates :name, length: { in: 4..40 }
  validates :price, numericality: { greater_than_or_equal_to: 100 }
  validates :amount, numericality: { greater_than: 0, only_integer: true }

  def attributes
    { name: nil, price: nil }
  end
end

i_phone = Product.new(name: 'iPhone 13 Pro', amount: 0)
mac_book = Product.new(name: 'MacBook Pro 13', amount: 5, price: 1300)

puts i_phone.as_json #=> {"name"=>"iPhone 13 Pro", "price"=>nil}
puts mac_book.as_json #=> {"name"=>"MacBook Pro 13", "price"=>1300}
```

# Шаблон Active Record. Подключение

Active Record – паттерн проектирования, используемый для описания доступа к данным реляционных баз данных. Впервые описан Мартином Фаулером. Иногда считается анти-шаблоном за счет того, что нарушает *Принцип единой ответственности*.

Шаблон реализуется таким образом, что выполняются следующие условия:

- каждый экземпляр класса представляет запись из таблицы;
- при создании нового экземпляра создается новая запись в таблице;
- при удалении объекта удаляется и запись из таблицы.

```ruby
require 'active_record'

ActiveRecord::Base.establish_connection(
  adapter: 'sqlite3',
  database: 'test.db'
)
```

```ruby
require 'active_record'

ActiveRecord::Base.establish_connection(
  adapter: 'mysql2', # postgresql
  database: 'localhost',
  username: 'db_user',
  password: 'p@ssw0rd',
  database: 'test_db'
)
```

# Active Record миграции

```ruby
require 'active_record'

ActiveRecord::Base.establish_connection(
  adapter: 'sqlite3',
  database: 'test.db'
)

class Product < ActiveRecord::Base
  validates :name, :amount, :price, presence: true
  validates :name, length: { in: 4..40 }
  validates :price, numericality: { greater_than_or_equal_to: 100 }
  validates :amount, numericality: { greater_than: 0, only_integer: true }
end

# Product.last #=> no such table: products (ActiveRecord::StatementInvalid)

class CreateProductTable < ActiveRecord::Migration[7.0]
  def change
    create_table :products do |t|
      t.string :name, null: false
      t.decimal :price, precision: 6, scale: 2, null: false
      t.integer :amount, null: false
      t.timestamps
    end
  end
end

CreateProductTable.migrate(:up)
```

```ruby
ReversibleAndIrreversibleMethods = [
  :create_table, :create_join_table, :rename_table, :add_column, :remove_column,
  :rename_index, :rename_column, :add_index, :remove_index, :add_timestamps, :remove_timestamps,
  :change_column_default, :add_reference, :remove_reference, :transaction,
  :drop_join_table, :drop_table, :execute_block, :enable_extension, :disable_extension,
  :change_column, :execute, :remove_columns, :change_column_null,
  :add_foreign_key, :remove_foreign_key,
  :change_column_comment, :change_table_comment,
  :add_check_constraint, :remove_check_constraint,
  :add_exclusion_constraint, :remove_exclusion_constraint,
  :create_enum, :drop_enum,
]
```

```ruby
define_column_methods :bigint, :binary, :boolean, :date, :datetime, :decimal,
  :float, :integer, :json, :string, :text, :time, :timestamp, :virtual

alias :blob :binary
alias :numeric :decimal
```

# Чтение и запись данных

```ruby
# Product.create!(name: 'iPhone') #=> Validation failed (ActiveRecord::RecordInvalid)
Product.destroy_all

Product.create!(name: 'iPhone 13', amount: 10, price: 1000)
Product.create!(name: 'MacBook Pro 13', amount: 15, price: 1300)
Product.create!(name: 'Lenovo Legion 15', amount: 7, price: 700.50)
puts Product.count #=> 3
```

```ruby
ActiveRecord::Base.logger = ActiveSupport::Logger.new(STDOUT)

puts Product.first.as_json
 #=> {"id"=>13, "name"=>"iPhone 13", "price"=>"1000.0", "amount"=>10,
 #    "created_at"=>"2022-08-27T14:39:42.006Z", "updated_at"=>"2022-08-27T14:39:42.006Z"}

Product.last
puts Product.find_by(price: 400..).as_json(only: %i[name price])
 #=> {"name"=>"iPhone 13", "price"=>"1000.0"}

p Product.where(price: 400..).as_json(only: :name)
 #=> [{"name"=>"iPhone 13"}, {"name"=>"MacBook Pro 13"}, {"name"=>"Lenovo Legion 15"}]

puts Product.all.count #=> 3
```

```
  Product Load (0.0ms)  SELECT "products".* FROM "products" ORDER BY "products"."id" ASC LIMIT ?  [["LIMIT", 1]]
{"id"=>13, "name"=>"iPhone 13", "price"=>"1000.0", "amount"=>10, "created_at"=>"2022-08-27T14:39:42.006Z", "updated_at"=>"2022-08-27T14:39:42.006Z"}
  Product Load (0.0ms)  SELECT "products".* FROM "products" ORDER BY "products"."id" DESC LIMIT ?  [["LIMIT", 1]]
  Product Load (0.0ms)  SELECT "products".* FROM "products" WHERE "products"."price" >= ? LIMIT ?  [["price", 400.0], ["LIMIT", 1]]
{"name"=>"iPhone 13", "price"=>"1000.0"}
  Product Load (0.0ms)  SELECT "products".* FROM "products" WHERE "products"."price" >= ?  [["price", 400.0]]
[{"name"=>"iPhone 13"}, {"name"=>"MacBook Pro 13"}, {"name"=>"Lenovo Legion 15"}]
  Product Count (0.0ms)  SELECT COUNT(*) FROM "products"
```

# Создание связанных таблиц

```ruby
require 'active_record'

ActiveRecord::Base.establish_connection(
  adapter: 'sqlite3',
  database: 'test.db'
)

ActiveRecord::Base.logger = ActiveSupport::Logger.new(STDOUT)

class CreateUsersTable < ActiveRecord::Migration[7.0]
  def change
    create_table :users do |t|
      t.string :username, null: false
      t.integer :raing, null: false, default: 0
      t.timestamps
    end
  end
end

class RenameUsersToClients < ActiveRecord::Migration[7.0]
  def change
    rename_table :users, :clients
    rename_column :clients, :raing, :rating
    add_column :clients, :has_purchases, :boolean, null: false, default: false
    add_index :clients, :username, unique: true
  end
end

CreateUsersTable.migrate(:up)
RenameUsersToClients.migrate(:up)

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

```ruby
require 'active_record'

ActiveRecord::Base.establish_connection(
  adapter: 'sqlite3',
  database: 'test.db'
)

ActiveRecord::Base.logger = ActiveSupport::Logger.new(STDOUT)

class CreateClientProductTable < ActiveRecord::Migration[7.0]
  def change
    create_table :client_products do |t|
      t.references :client, foreign_key: true, null: false
      t.belongs_to :product, foreign_key: true, null: false
      t.timestamps
    end
  end
end

unless ActiveRecord::Base.connection.table_exists?('client_products')
  CreateClientProductTable.migrate(:up)
end
```

# Указание связей в модели

```ruby
require 'active_record'

ActiveRecord::Base.establish_connection(
  adapter: 'sqlite3', database: 'test.db'
)

ActiveRecord::Base.logger = ActiveSupport::Logger.new(STDOUT)

class Product < ActiveRecord::Base
  has_and_belongs_to_many :clients, join_table: 'client_products'

  validates :name, :amount, :price, presence: true
  validates :name, length: { in: 4..40 }
  validates :price, numericality: { greater_than_or_equal_to: 100 }
  validates :amount, numericality: { greater_than: 0, only_integer: true }
end

class Client < ActiveRecord::Base
  has_and_belongs_to_many :products, join_table: 'client_products'

  validates :username, presence: true, uniqueness: true
end

Client.destroy_all

client = Client.new(username: 'first_client')
client.save

client.products
```

```ruby
class ClientProduct < ActiveRecord::Base
  belongs_to :product
  belongs_to :client
end

class Product < ActiveRecord::Base
  has_many :client_products, dependent: :destroy
  has_many :clients, through: :client_products

  validates :name, :amount, :price, presence: true
  validates :name, length: { in: 4..40 }
  validates :price, numericality: { greater_than_or_equal_to: 100 }
  validates :amount, numericality: { greater_than: 0, only_integer: true }
end

class Client < ActiveRecord::Base
  has_many :client_products, dependent: :destroy
  has_many :products, through: :client_products

  validates :username, presence: true, uniqueness: true
end

Client.destroy_all

client = Client.new(username: 'first_client')
client.save!

client.client_products.create!(product: Product.first)
ClientProduct.create!(client: client, product: Product.second)

p client.client_products.map(&:product).as_json(only: :name)
 #=> [{"name"=>"iPhone 13"}, {"name"=>"MacBook Pro 13"}]

p client.products.as_json(only: :name)
 #=> [{"name"=>"iPhone 13"}, {"name"=>"MacBook Pro 13"}]
```

# Полезные ссылки

https://github.com/rails/rails/tree/main/activemodel/lib/active_model − Ссылка на гем Acive Model в GitHub;

https://edgeguides.rubyonrails.org/active_model_basics.html − Описание основных модулей Active Model;

https://github.com/rails/rails/tree/main/activerecord − Ссылка на гем Active Record в GitHub;

https://www.devdungeon.com/content/ruby-activerecord-without-rails-tutorial − Описание работы с Active Record вне Rails.

Конец! Спасибо!