

ActiveRecord

Notatki do prezentacji do prezentacji na 4tym spotkaniu spotkanie KRUG (www.ruby.org.pl)
Autor: Maciej Książek www.DrClick.pl

Wprowadzenie

Cel twórców: możliwość stworzenia modelu domeny aplikacji przy użyciu jak najmniejszej ilości kodu.

ActiveRecord realizuje:

- mapowanie obiektów na relacyjną bazę danych.
- pozwala stworzyć model aplikacji udostępniając dane jak i logikę aplikacji

Konwencja ponad konfigurację (Convention over Configuration)

```
# plik User.rb w katalogu app/models
```

```
class User < ActiveRecord::Base  
end
```

```
CREATE TABLE `users` (  
  `id` int(11) NOT NULL auto_increment,  
  `login` varchar(255) NOT NULL,  
  `password` varchar(255) NOT NULL,  
  `name` varchar(255) NOT NULL,  
  `role_id` int(11) NOT NULL,  
  PRIMARY KEY (`id`)  
)
```

```
u = User.find 1 # SELECT * FROM users WHERE  
id = 1  
u.name = "Jan Kowalski"  
u.save # zapisuje rekord w bazie (UPDATE users  
SET ... WHERE ... )
```

```
u.destroy # usuwa rekord z bazy i zamraża  
instancję obiektu
```

```
users = User.find :all, :conditions =>  
{ :first_name => "Maciej" }
```

```

teenagers = User.find :all,
  :select => "name, id ",
  :conditions => ["age < ? AND age > ?", 20, 10],
  :order => "age DESC",
  :limit => 5

```

```

SELECT name, id
FROM users
WHERE age < 20 AND age > 10
ORDER BY age DESC
LIMIT 5

```

```

h = { :name => "Jan Nowak",
      :email => "x@test.pl",
      :password => "123" }

```

```

User.find_by_name("Jan Kowalski").update_attributes(h)

```

```

jk = User.find_all_by_name_and_email("Jan Kowalski", "x@test.pl")

```

Relacje



```

class User < ActiveRecord::Base

```

```

  has_many :activities

```

```

  has_many :projects, :through => :activities, :group => "project_id"

```

```

end

```

```

class Activity < ActiveRecord::Base

```

```

  belongs_to :project

```

```

  belongs_to :user

```

```

end

```

```

class Project < ActiveRecord::Base

```

```

  has_many :activities, :dependent => :destroy

```

```

  has_many :recent_activities, :conditions => "updated_at > NOW( ) - INTERVAL 7 DAY"

```

```

end

```

Relacje: użycie:



```
u = User.find 1          # wyszukuje rekord o id=1
u.activities            # zwraca Array z activities należącymi do danego usera
u.activities.first.project.name # nazwa projektu przypisanego do pierwszej aktywności użytkownika
```

```
wspolpracownicy = u.activities.first.project.users.collect { |u| [u.first_name,u.last_name ] }
```

```
p.activities.collect {|a| a.project.name }.uniq
```

Walidacje

```
class Project < ActiveRecord::Base
```

```
  validates_presence_of :name, :description
  validates_uniqueness_of :name
```

```
end
```

```
p = Project.new
p.save          # => false
```

```
p.errors.size      # => 2
p.errors.on "name" # => "can't be blank"
p.valid?           # => false
```

```
p.name = "37 signals"
p.description = "Ruby on Rails tu powstał."
p.save          # => true
```

```
class Project < ActiveRecord::Base
```

```
  def validate_on_update
    errors.add("Wystaw fakturę zanim zamkniesz projekt.") if invoices.blank? and status ==
STATUS[:closed]
  end
```

```
end
```

```
validates_acceptance_of
validates_associated
validates_confirmation_of
validates_each
validates_exclusion_of
validates_format_of
validates_inclusion_of
validates_length_of
validates_numericality_of
validates_presence_of
```

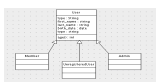
```
validates_size_of  
validates_uniqueness_of
```

Callbacks

```
class Image < ActiveRecord::Base  
  after_destroy :delete_image_files  
  
  def delete_image_files  
    # usuwa wszystkie pliki związane z kasowanym rekordem  
  end  
  
  def before_save  
    raise "Brak zdefiniowanej nazwy pliku !" if file_name.blank?  
  end  
  
  before_create 'logger.info "New image created: #{file_name}" '  
  
end
```

```
after_create  
after_destroy  
after_save  
after_update  
after_validation  
after_validation_on_create  
after_validation_on_update  
before_create  
before_destroy  
before_save  
before_update  
before_validation  
before_validation_on_create  
before_validation_on_update
```

Single Table Inheritance



```
class User < ActiveRecord::Base  
  
  def age  
    (Date.today - birth_date.to_date) / 365  
  end  
  
end
```

```
end
```

```
class Member < User  
end
```

```
class Admin < User  
end
```

```
class UnregisteredUser < User  
end
```

```
# ----- sposób użycia
```

```
# szuka po całej tabeli users (dowolny typ użytkownika)  
users = User.find :all  
users[0].class # => Member  
users[1].class # => Admin
```

```
# szuka w tabeli users z WHERE type="Member" AND first_name = "Maciej"  
Member.find :all, :conditions => {:first_name=> "Maciej"}
```

```
# szuka w tabeli users z WHERE type="Admin"  
Admin.find :all
```

Relacje polimorficzne

```
class Address < ActiveRecord::Base  
  belongs_to :addressable, :polymorphic => true  
end
```

```
class Company < ActiveRecord::Base  
  has_many :addresses, :as => :addressable  
end
```

```
class User < ActiveRecord::Base  
  has_many :addresses, :as => :addressable  
end
```

W tabeli addresses potrzebne pola:

```
addressable_type  
addressable_id
```

Przykład - adres należący do użytkownika (klasa User) o id =1

```
addressable_type = "User"  
addressable_id = 1
```

Transakcje

```
# Obtain an exclusive lock on person 1 so we can safely increment visits.
Person.transaction do
  # SELECT * FROM people WHERE id=1 FOR UPDATE
  person = Person.find(1, :lock => true)
  person.visits += 1
  person.save!
end
```

Inne

```
u = User.find :all
u.to_xml # => zwraca XML z danymi użytkowników
u.to_yaml # => dane w formacie yaml
```