

Lezione 5: ActiveRecord

Matteo Vaccari

<http://matteo.vaccari.name/>
matteo.vaccari@uninsubria.it



(cc) Matteo Vaccari. Published in Italy.
Attribution – Non commercial – Share alike 2.5

Creare la tabella I

Object-relational mapping before Rails

```
create table people (  
  id int auto_increment primary key,  
  name varchar(255),  
  street varchar(255),  
  city varchar(255),  
  email varchar(255)  
)
```

Creare il *bean* I

```
public class Person {
    private int id;
    private String name;
    private String street;
    private String city;
    private String email;

    public int getId() {
        return id;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getName() {
```

Creare il *bean* II

```
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getStreet() {
        return street;
    }
    public void setStreet(String street) {
        this.street = street;
    }
}
```

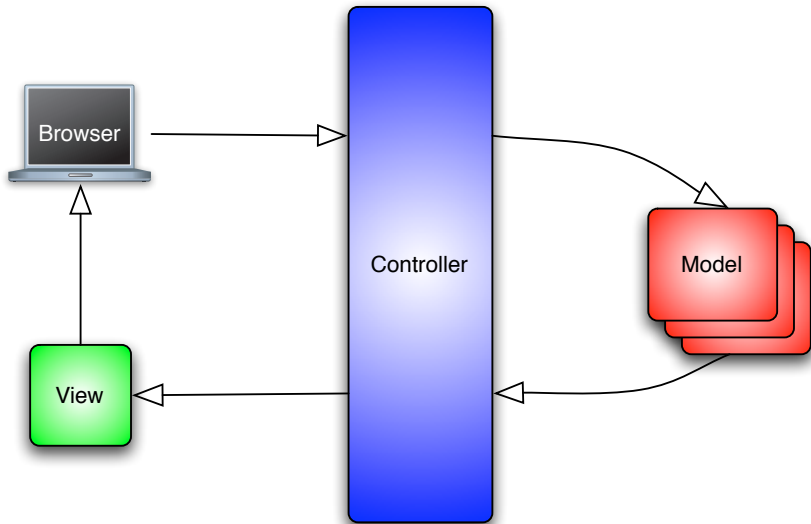
II Data Access Object I

```
public class PersonGateway extends DatabaseGateway {
    public int create(String name, String street, String city, String email) {
        execute("insert into people values (name, street, city, email) " +
            "values (?, ?, ?, ?)",
            list(name, street, city, email));
        return lastInsertId();
    }

    public void update(Person person) {
        execute("update people set name = ?, street = ?, " +
            " city = ?, email = ? where id = ?",
            list(person.getName(), person.getStreet(),
                person.getCity(), person.getEmail(), person.getId()));
    }
}
```

II Data Access Object II

```
public Person find(int id) {
    List rows = select("select * from people where id = ?", list(id));
    if (0 == rows.size()) return null;
    Map row = (Map) rows.get(0);
    Person result = new Person();
    result.setId((Integer) row.get("id"));
    result.setName((String) row.get("name"));
    result.setStreet((String) row.get("street"));
    result.setCity((String) row.get("city"));
    result.setEmail((String) row.get("email"));
    return result;
}
}
```



ActiveRecord

Active Record: “An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data”

—Martin Fowler¹

¹Patterns of Enterprise Application Architecture

config/database.yml

```
development:  
  adapter: mysql  
  database: quotes_development  
  username: matteo  
  password:  
  host: localhost
```

```
test:  
  adapter: mysql  
  database: quotes_test  
  username: matteo  
  password:  
  host: localhost
```

```
production:  
  adapter: mysql  
  database: quotes_production  
  username: root  
  password:  
  host: localhost
```

Creiamo i database

```
$ mysql -uroot -p
```

```
Password:
```

```
Welcome to the MySQL monitor.
```

```
mysql> create database quotes_development;
```

```
Query OK, 1 row affected (0.16 sec)
```

```
mysql> create database quotes_test;
```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql> grant all on quotes_development.* to matteo@localhost;
```

```
Query OK, 0 rows affected (0.25 sec)
```

```
mysql> grant all on quotes_test.* to matteo@localhost;
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> quit
```

```
Bye
```

```
$
```

```
$ script/generate model Quote
  exists  app/models/
  exists  test/unit/
  exists  test/fixtures/
  create  app/models/quote.rb
  create  test/unit/quote_test.rb
  create  test/fixtures/quotes.yml
  create  db/migrate
  create  db/migrate/20100107083453_create_quotes.rb
```

app/models/quote.rb

```
class Quote < ActiveRecord::Base
end
```

```
$ script/generate model Quote
  exists app/models/
  :
  :
  create db/migrate
  create db/migrate/20100107083453_create_quotes.rb
```

db/migrate/20100107083453_create_quotes.rb

```
class CreateQuotes < ActiveRecord::Migration
  def self.up
    create_table :quotes do |t|
      # t.column :name, :string
    end
  end

  def self.down
    drop_table :quotes
  end
end
```

db/migrate/20100107083453_create_quotes.rb

```
class CreateQuotes < ActiveRecord::Migration
  def self.up
    create_table :quotes do |t|
      t.column :author, :string
      t.column :body, :text
      t.column :created_at, :datetime
    end
  end
end
```

```
$ rake migrate
```

```
(in /Users/matteo/didattica/aw/slides2006/quotes_03)
```

```
== CreateQuotes: migrating =====
```

```
- create_table(:quotes)
```

```
-> 0.1357s
```

```
== CreateQuotes: migrated (0.1359s) =====
```

```
$
```

```
class CreateQuotes < ActiveRecord::Migration
  def self.up
    create_table :quotes do |t|
      t.column :author, :string
      t.column :body, :text
      t.column :created_at, :datetime
    end
  end
end
```

```
CREATE TABLE quotes (
  id          int NOT NULL auto_increment,
  author      varchar(255) default NULL,
  body        text,
  created_at  datetime default NULL,
  PRIMARY KEY (id)
);
```

```
$ mysql quotes_development
Welcome to the MySQL monitor.
```

```
mysql> show tables;
```

```
+-----+
| Tables_in_quotes_development |
+-----+
| quotes                        |
| schema_migrations            |
+-----+
2 rows in set (0.00 sec)
```

```
mysql> describe quotes;
```

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |
| author     | varchar(255)  | YES  |     | NULL    |                |
| body       | text          | YES  |     | NULL    |                |
| created_at | datetime      | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.26 sec)
```

```
mysql>
```

Come fa rake “db:migrate” a sapere quali migrazioni applicare?

```
mysql> select * from schema_migrations;
+-----+
| version      |
+-----+
| 20100107083453 |
+-----+
1 row in set (0.00 sec)
```

Questo corrisponde al nome del file di migrazione

20100107083453_create_quotes.rb

Le migrazioni supportano lo sviluppo evolutivo

```
$ script/generate migration add_title_to_quotes
  exists db/migrate
  create db/migrate/20100107085255_add_title_to_quotes.rb
```

```
db/migrate/20100107085255_add_title_to_quotes.rb
```

```
class AddTitleToQuotes < ActiveRecord::Migration
  def self.up
    add_column :quotes, :title, :string
  end

  def self.down
    remove_column :quotes, :title
  end
end
```

```
$ rake db:migrate
(in /Users/matteo/work/.../quotes)
== AddTitleToQuotes: migrating =====
== AddTitleToQuotes: migrated (0.0000s) =====
```

Le migrazioni si possono fare e disfare

```
$ rake db:rollback # disfa l'ultima migrazione
(in /Users/matteo/work/.../quotes)
== AddTitleToQuotes: reverting =====
== AddTitleToQuotes: reverted (0.0000s) =====
```

```
$ rake db:migrate # riapplica l'ultima migrazione
(in /Users/matteo/work/.../quotes)
== AddTitleToQuotes: migrating =====
== AddTitleToQuotes: migrated (0.0000s) =====
```

```
$ rake db:migrate:redo # disfa e riapplica l'ultima migrazione
(in /Users/matteo/work/.../quotes)
== AddTitleToQuotes: reverting =====
== AddTitleToQuotes: reverted (0.0000s) =====
```

```
== AddTitleToQuotes: migrating =====
== AddTitleToQuotes: migrated (0.0000s) =====
```

```
$
```

Giocare con la console

```
$ script/console
Loading development environment.
>> Quote.find(:all)
=> []
>> q = Quote.new(:author => 'Jack', :body => " Il mattino ha l'oro in bocca")
=> #<Quote:0x27216d0
      @new_record=true,
      @attributes='body'=>'Il mattino ha l'oro in bocca', 'author'=>'Jack',
      'created_at'=>nil>
>> q.author
=> 'Jack'
>> q.body
=> 'Il mattino ha l'oro in bocca'
>> q.save
=> true
>> q
=> #<Quote:0x27216d0
      @new_record=false,
      @errors=#<...@errors=...>,
      @attributes='body'=>'Il mattino ha l'oro in bocca', 'author'=>'Jack',
      'id'=>1, 'created_at'=>Mon Oct 23 15:37:44 CEST 2006>
>>
```

Object-relational mapping (ORM)

DB	Ruby
table	class
row	instance
column	attribute

Convenzione: il nome della classe è singolare, quello della tabella plurale

SQL	Ruby
quotes	Quote
line_items	LineItem

Primary keys

Convenzione ogni tabella ha come *chiave primaria* una colonna **id** di tipo intero

```
class orders (  
  id          int          auto_increment,  
  name        varchar(200) not null,  
  total       float        not null,  
  primary key (id)  
);
```

```
class Order < ActiveRecord::Base  
end
```

```
order = Order.find(123) # find by id  
order.name = 'Dave Thomas'  
order.save
```

Creating new rows

```
class Order < ActiveRecord::Base
end
```

```
an_order = Order.new                # creo un oggetto in ram
an_order.name      = 'Dave Thomas'  # assegna gli attributi
an_order.email     = 'dave@pragprog.com'
an_order.address   = '123 Main St'
an_order.pay_type  = 'check'
an_order.save      # crea la riga su db
```

```
# metodo alternativo:
```

```
an_order = Order.new(                # passa una hash
  :name      => 'Dave Thomas',        # nel costruttore
  :email     => 'dave@pragprog.com',
  :address   => '123 Main St',
  :pay_type  => 'check')
an_order.save      # crea la riga su DB
```

Reading existing rows

By primary key:

```
an_order    = Order.find(42)
many_orders = Order.find([19, 23, 29])
```

By arbitrary conditions:

```
n = "Dave"

# NO !!!
o = Order.find(:first,
               :conditions => "name = '#{n}' ")

# OK !!!
o = Order.find(:first,
               :conditions => ["name = ?", n])

# Ancora meglio:
o = Order.find(:first,
               :conditions => ["name = :name", {:name => n}])
```

Tre tipi di find

lancia eccezione se non lo trova

```
Order.find(42)
```

restituisce nil se non lo trova

```
Order.find(:first, :conditions => ...)
```

restituisce tutti quelli che trova in un'array

```
Order.find(:all, :conditions => ...)
```

Updating

Metodo 0

```
o = Order.find(42)
o.name = "Donaldus Anas"
if o.save
  # ordine salvato correttamente
else
  # l'ordine non è stato salvato
end
```

Metodo 1

```
o = Order.find(42)
o.update_attributes(:name => "Mickey", :city => 'Topolinia')
```

Metodo 2

```
Order.update(42, :name, "Mickey")
```

Deleting

```
Order.delete(42)
```

```
Order.delete([42, 43, 4444])
```

```
Order.delete_all("name like ?", pattern)
```

Che ne è dei nostri motti?

app/controllers/quotes_controller.rb

```
class QuotesController < ApplicationController
  def index
    @quote = Quote.find(:first, :order => 'rand()')
  end
end
```

app/views/index.html.erb

```
<h2>Il motto:</h2>
<div class='quotation'>
  <div class='quote'>
    <%= @quote.body %>
    <div class='author'>&mdash; <%= @quote.author %></div>
  </div>
</div>
```

app/controllers/quotes_controller.rb

```
class QuotesController < ApplicationController
  def index
    @quote = Quote.find(:first, :order => 'rand()')
  end
  def list
    @quotes = Quote.find(:all)
  end
end
```

app/views/list.html.erb

```
<h2>Tutti i motti</h2>
<% for quote in @quotes %>
  <div class='quotation'>
    <div class='quote'>
      <%= quote.body %>
      <div class='author'>&mdash; <%= quote.author %></div>
    </div>
  </div>
<% end %>
```

Don't Repeat Yourself!

app/views/index.html.erb

```
<h2>Il motto:</h2>
<div class='quotation'>
  <div class='quote'>
    <%= @quote.body %>
    <div class='author'>&mdash; <%= @quote.author %></div>
  </div>
</div>
```

app/views/list.html.erb

```
<h2>Tutti i motti</h2>
<% for quote in @quotes %>
  <div class='quotation'>
    <div class='quote'>
      <%= quote.body %>
      <div class='author'>&mdash; <%= quote.author %></div>
    </div>
  </div>
<% end %>
```

Use partial templates to avoid repetitions

app/views/index.html.erb

```
<h2>Il motto:</h2>
<%= render :partial => 'quote' %>
```

app/views/list.html.erb

```
<h2>Tutti i motti</h2>
<% for quote in @quotes %>
  <%= render :partial => 'quote', :object => quote %>
<% end %>
```

app/views/_quote.html.erb (a **partial template**)

```
<div class='quotation'>
  <div class='quote'>
    <%= quote.body %>
    <div class='author'>&mdash; <%= quote.author %></div>
  </div>
</div>
```

Even better

app/views/list.html.erb

```
<h2>Tutti i motti</h2>
<% for quote in @quotes %>
  <%= render :partial => 'quote', :object => quote %>
<% end %>
```

app/views/list.html.erb

```
<h2>Tutti i motti</h2>
<%= render :partial => 'quote', :collection => @quotes %>
```

Esercizio: un servizio citazioni

Rifare l'esercizio usando Active Record

1. `http://example.com/`
ridirige su `http://example.com/quotes?random`
2. `http://example.com/quotes?random`
produce una citazione a caso
3. `http://example.com/quotes`
produce le prime 10 citazioni
4. `http://example.com/quotes/show/123`
produce la citazione numero 123
5. `http://example.com/quotes?count=4`
produce le prime 4 citazioni
6. `http://example.com/quotes?start=12`
produce 10 citazioni a partire dalla dodicesima

Continuare l'esercizio

Rifare l'esercizio usando Active Record

1. <http://example.com/quotes?q=pippo>
produce una citazione che contiene la parola pippo
2. <http://example.com/quotes?o=alpha>
produce 10 citazioni in ordine alfabetico (invece che in ordine di id)
3. <http://example.com/quotes?o=date>
produce 10 citazioni in ordine di data di ultima modifica