Full Stack Notes

Database CRUD with ActiveRecord

“Active Record is the M in MVC - the model - which is the layer of the system responsible for representing business data and logic.”

Table of Contents

ActiveRecord the Pattern

Active Record was first described by Martin Fowler in his book Patterns of Enterprise Application Architecture. In Active Record, objects carry both persistent data and behavior which operates on that data.

ActiveRecord in Rails is an example of an ORM or Object Relational Mapper, a technique that connects the Model objects of your application to tables in your relational database management system.

ActiveRecord Naming Conventions

ActiveRecord models should be nouns. Rails will automatically associate a model with a table by pluralizing the model name.

Some example Model names with their associated table names:

Model Name	Associated Table
Article	articles
Ghost	ghosts
LineItem	line_items
Person	people
Category	categories

Notes that Rails understands how to properly pluralize words like person and category.

Also note that CamelCase model names are converted to plural snake_case table names.

A Basic AR Model

Here’s what a basic model would look like for an `articles` table:

```
class Article < ActiveRecord::Base
end
```

Even with no class content we inherite a bunch of functionality from `ApplicationRecord`:

```
Article.count # RUNS: SELECT COUNT(*) FROM articles
```

ActiveRecord is For CRUD

ActiveRecord allows us to perform CRUD actions on our database without writing system. For example:

```
# CREATE:
Article.create(title: 'First Post', content: 'This is a witty article. har har har.')
```

```ruby
# READ:

article = Article.first

# UPDATE

article.update(title: 'My First Post!')

# DELETE

article.destroy
```

ActiveRecord Notes as Code

The remainder of the AR notes are presented as executable Ruby source code in this Github repo.

Each one of these following scripts depends on the provided ar.rb file to load up the database connection. The following four scripts show how to perform CRUD on data in a customers table using a Customer mode.

- Read: active_record_read.rb
- Create: active_record_create.rb
- Update: active_record_update.rb
- Delete: active_record_delete.rb

ActiveRecord Associations

One-to-many and many-to-many associations between your database tables can be facilitated by ActiveRecord.

Here’s an example of an association where each row in the customers table “belongs to” a row in the provinces table by way of a `province_id` foreign key:

```ruby
class Province < ActiveRecord::Base
  # This AR class is linked with the provinces table.
  # A province has a one to many relationship with customers.
  # In other words, a province:
  has_many :customers
end

class Customer < ActiveRecord::Base
  # This AR class is linked with the customers table.
  # This table has a province_id foreign key so a customer:
  belongs_to :province
end
```

Explore the following file to see how to work with this customer/province association:

- AR Associations: active_record_assoc.rb

ActiveRecord Resources

Everything else you've wanted to know about ActiveRecord but were afraid to ask is covered in the official guides:

- Active Record Basics
- Active Record Query Interface Guide
- Active Record Validations Guide
- Active Record Associations Guide