



Introduction to Active Record

A talk on our Object-relational Mapper
by Hunter T. Chapman

Introduction to ActiveRecord

- Just an introduction -- Mostly High Level
- What are the parts and how do they work
- Mini how-to
- This is gonna be awesome

Awesome Docs!

[http://edgeguides.rubyonrails.org/
active_record_migrations.html](http://edgeguides.rubyonrails.org/active_record_migrations.html)

The ActiveRecord Docs are some of the best you will find. Thorough, clean, simple. Lots of sample code to reference. Moral of the story == Use the Docs



Active Record

- Active Record is a gem (primarily) used in Rails for interacting with databases using object-relational mapping (ORM)
- ORM == creating persistent Ruby objects that you can easily manipulate through a database.

Active Record

- Get comfy, you will be using this... A LOT
- NBD - Turns out AR is super easy.

The Broad Strokes



Naming Conventions

- Class names singular, CamelCase
- Table names plural, snake_case

Names that ActiveRecord Can Match	
<i>class name</i>	<i>table name</i>
<i>User</i>	<i>users</i>
<i>LineItem</i>	<i>line_items</i>
<i>Deer</i>	<i>deer</i>
<i>Person</i>	<i>people</i>

Naming Conventions

- Class names singular, CamelCase
- Foreign key names singular, snake_case

Names that ActiveRecord Can Match	
<i>class name</i>	<i>foreign_key</i>
<i>User</i>	<i>user_id</i>
<i>LineItem</i>	<i>line_item_id</i>
<i>Deer</i>	<i>deer_id</i>
<i>Person</i>	<i>person_id</i>

Naming Conventions

Singular, Plural, snake_case, CamelCase

This stuff matters. You will spend hours tracking down issues with your schema only to find you missed an “s” on the end of one line that you’ve looked at a hundred times already. Super Not Cool!



Active Record Sugar

Groovy Stuff AR does for you!

- Managing tables
- Mapping Ruby classes to database tables
- Associations between classes
- Validations

Active Record is BIG

ActiveRecord Source Screen grab

You can, and will learn something new about AR every time you use it for the foreseeable future.

For most of you this is the
Largest code base you have seen so far.

<code>@@associations</code>	Licenseize none / and one / re...	4 days
<code>@@attribute</code>	rm :valueparameter?	9 days
<code>@@attribute set</code>	maintain a consistent order l...	7 days
<code>@@cookers</code>	provide a better error messa...	2 mont.
<code>@@connect...</code>	merge out request #16out l...	20 nov.
<code>@@context set</code>	introduce a context for rend...	4 year
<code>@@cscope</code>	move required error message...	16 nov.
<code>@@docond</code>	change :docond :void to :all...	7 days
<code>@@monitors</code>	merge out request #16out l...	7 mont.
<code>@@names</code>	schema creation doesn't do...	3 days
<code>@@reason</code>	add an action element at t...	8 days
<code>@@scoping</code>	current scope shouldn't do...	8 days
<code>@@serializers</code>	remove most type related b...	17 day.
<code>@@tasks</code>	fix syntax warning	2 mont.
<code>@@type</code>	reorder adapter specific b...	4 day
<code>@@type caster</code>	remove most uses of :look...	1 day
<code>@@validations</code>	rm :valueparameter?	9 days
<code>@@validates</code>	push multi-parameter assign...	9 days
<code>@@validates</code>	infect the :methodparameter ...	2 mont.
<code>@@validates</code>	merge branch master or dr...	3 days
<code>@@attribute :id</code>	Attribute assignment and ty...	16 day.
<code>@@attribute</code>	push multi-parameter assign...	9 days
<code>@@attribute</code>	Attribute assignment and ty...	16 day.
<code>@@attribute</code>	Attribute assignment and ty...	16 day.
<code>@@attribute</code>	conditionally improve the de...	10 day.
<code>@@attributes :id</code>	reorder adapter specific b...	4 day
<code>@@autoeval</code>	Always perform validations o...	18 day.
<code>@@base :id</code>	add has_secure token to A...	8 mont.
<code>@@callbacks :id</code>	deprecate :base as the w...	2 mont.
<code>@@connect...</code>	check for has_many instead ...	2 mont.
<code>@@core :id</code>	mark some methods as :node...	11 day.
<code>@@counter :c</code>	improve consistency of coun...	21 day.
<code>@@dynamic</code>	remove support to activerec...	2 mont.
<code>@@enum :id</code>	mention where not in the s...	3 days
<code>@@errors :id</code>	extract ActiveRecord::RS...	24 day.
<code>@@extend :id</code>	move :id to :comment	7 mont.
<code>@@extend :id</code>	add :id to :mark to :thead...	2 year
<code>@@extend :id</code>	don't try to :CAPLAIN select ...	2 year
<code>@@features :id</code>	use module/include instead ...	16 day.
<code>@@gem vers...</code>	draft Rails 6 development *	3 mont.
<code>@@inherit...</code>	Attribute assignment and ty...	16 day.
<code>@@inherit...</code>	use :mode name on :meth...	8 mont.
<code>@@id :id</code>	remove relationship class...	20 day.
<code>@@model :id</code>	do not check only for the H...	2 mont.
<code>@@model sc...</code>	rm :columnname type	13 day.
<code>@@nested</code>	Always perform validations o...	18 day.
<code>@@no touch...</code>	add :no touch to :touch al...	2 mont.
<code>@@no touch...</code>	Licenseize none / and one / re...	8 days
<code>@@persist...</code>	add destroyed records to th...	10 day.
<code>@@query cac...</code>	remove blank lines in the st...	7 mont.
<code>@@querying :id</code>	Added :for to ActiveRecord...	19 day.
<code>@@rails :id</code>	do not remove AH :idz load f...	3 mont.
<code>@@redefin...</code>	pass symbol as an argument...	3 mont.
<code>@@reflected</code>	remove deprecated support ...	8 mont.
<code>@@reason :id</code>	Licenseize none / and one / re...	4 days
<code>@@result :id</code>	Revert "improve performanc...	3 mont.
<code>@@runtime :r</code>	bioe using method missing t...	4 year
<code>@@serialized</code>	bioe bioinfo a column to d...	8 mont.
<code>@@schemas :id</code>	remove blank lines in the st...	7 mont.
<code>@@schemas :id</code>	improve a dump of the brin...	2 mont.
<code>@@schemas</code>	extract ActiveRecord::Associ...	8 mont.
<code>@@scoping :id</code>	current scope shouldn't do...	8 days
<code>@@secure :id</code>	do not overwrite secret tok...	4 days
<code>@@serializ...</code>	pass symbol as an argument...	3 mont.
<code>@@statement</code>	remove relationship class...	20 day.
<code>@@store :id</code>	fix type case or "validations"	2 mont.
<code>@@store me...</code>	respect custom primary key...	13 day.
<code>@@table :id</code>	fix warning: "uninterpreted ...	2 mont.
<code>@@transac...</code>	add destroyed records to th...	10 day.
<code>@@transac...</code>	don't out most of the ActiveRecord...	3 year
<code>@@type :id</code>	reorder adapter specific b...	4 day
<code>@@type cast...</code>	extract an explicit type case...	2 mont.
<code>@@validates</code>	add specific length validator	2 mont.
<code>@@version :id</code>	introduce Rails gem version ...	4 year

Active Record Parts

```
### Migrations ###  
class CreatePersons < ActiveRecord::Migration  
  create_table "persons" do |t|  
    t.string :first_name  
    t.integer :age  
  end  
  
### Models ###  
class Person < ActiveRecord::Base  
end  
  
### Control ###  
tom = Person.new  
tom.first_name = "Tom"  
tom.age = 28  
tom.save
```

AR: What are the parts?

- Migrations - Used to build tables, a blueprint of DB objects.
- Models - Apply universal methods to table objects. Relationships and Validations live here.
- Control Code - Actual interface code for DB objects.

Basic ActiveRecord Workflow

1. Write Migrations

- Blueprint of your DB

2. Run Migrations

- Once per round of migrations

3. Build Models

- Establish ORM Connection & Set up relationships

4. Implement control code

- Manipulate objects and persist changes to DB

Inherit from ActiveRecord

- Two primary classes in ActiveRecord module

```
module ActiveRecord
  class Migration
    # ...
  end

  class Base
    # ...
  end

  # ...
end
```


Migrations



Introduction to Active Record: Our Object-relational Mapper

ActiveRecord Migrations

File naming conventions

20141020120711_create_users.rb

Three Key Points:

- Timestamp:** This tells the app the order to run migrations.
- Action word:** Describes the primary action of the migration.
- Table word:** Describes the table effected by the migration.

Conventions:

snake_case, Action name is singular, Table name is plural.

ActiveRecord Migrations

File Location

Migrations live in:
ROOT/db/migrate



FOLDERS

✓ QuEst

✓ app

> assets

> controllers

> helpers

> mailers

> models

> views

> bin

> config

✓ db

✓ migrate

20141011185442_create_teachers.rb

20141011185502_create_students.rb

20141011185543_create_courses.rb

20141011185641_create_quizzes.rb

ActiveRecord Migrations

- Build database schema by writing migrations

```
# root/db/migrations/201407102947_create_oranges.rb
```

```
class CreateOranges < ActiveRecord::Migration
  def change
    # what do we want to do with the database
  end
end
```

ActiveRecord Migrations

- Add code to the change method to build out a table

```
class CreateOranges < ActiveRecord::Migration
  def change
    create_table :oranges do |t|
      t.integer :diameter
    end
  end
end
```

ActiveRecord Migrations

- Alter database schema by writing and running migrations

```
class AddOrangeTreeIdToOranges < ActiveRecord::Migration
  def change
    add_column :oranges, :orange_tree_id, :integer
  end
end
```

Creating Tables with Migrations

orange_trees
<i>id</i>
<i>age</i>
<i>height</i>
<i>created_at</i>
<i>updated_at</i>

oranges
<i>id</i>
<i>diameter</i>
<i>orange_tree_id</i>
<i>created_at</i>
<i>updated_at</i>

Creating Tables with SQL

```
CREATE TABLE orange_trees (  
  id INTEGER PRIMARY KEY  
    AUTOINCREMENT,  
  age INTEGER,  
  height INTEGER,  
  created_at DATETIME,  
  updated_at DATETIME);
```

orange_trees
<i>id</i>
<i>age</i>
<i>height</i>
<i>created_at</i>
<i>updated_at</i>

Creating Tables with Migrations

```
class CreateOrangeTrees < ActiveRecord::Migration
  def change
    create_table :orange_trees do |t|
      t.integer :age
      t.integer :height

      t.timestamps
    end
  end
end
```

orange_trees
<i>id</i>
<i>age</i>
<i>height</i>
<i>created_at</i>
<i>updated_at</i>

Creating Tables with SQL

```
CREATE TABLE oranges (  
  id INTEGER PRIMARY KEY  
    AUTOINCREMENT,  
  diameter INTEGER,  
  orange_tree_id INTEGER,  
  created_at DATETIME,  
  updated_at DATETIME);
```

oranges
<i>id</i>
<i>diameter</i>
<i>orange_tree_id</i>
<i>created_at</i>
<i>updated_at</i>

Creating Tables with Migrations

```
class CreateOranges < ActiveRecord::Migration
  def change
    create_table :oranges do |t|
      t.integer :diameter
      → t.integer :orange_tree_id
      → t.belongs_to :orange_tree
      → t.references :orange_tree
      t.timestamps
    end
  end
end
```

All the same

oranges
<i>id</i>
<i>diameter</i>
<i>orange_tree_id</i>
<i>created_at</i>
<i>updated_at</i>

Migrations Note

def change is the thing you will use most often.

<http://stackoverflow.com/questions/20890510/rails-migration-change-vs-up-down-methods>

```
class AddOrangeTreeIdToOranges < ActiveRecord::Migration
  def change
  end

  --VS:--

  def self.up
  end

  def self.down
  end
end
```

Common Migration Datatypes

<code>:boolean</code>	<code>:primary_key</code>
<code>:datetime</code>	<code>:string</code>
<code>:decimal</code>	<code>:text</code>
<code>:float</code>	<code>:time</code>
<code>:integer</code>	<code>:timestamp</code>

Sample List only

Consult your docs for accurate data types based on the DB in use.

Models



Models Map Classes to Tables

```
class OrangeTree < ActiveRecord::Base  
end
```

Modeling State	
<i>Ruby</i>	<i>Database</i>
<i>Classes</i>	<i>Tables</i>
<i>Instances of classes</i>	<i>Rows</i>
<i>Instance variables</i>	<i>Fields</i>

ActiveRecord Models

File naming conventions

`student.rb`

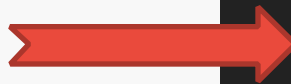
Three Key Points:

- `snake_case`
- **Singular**
- Filename matches applicable table

ActiveRecord Models

File Location

Models live in:
ROOT/app/models

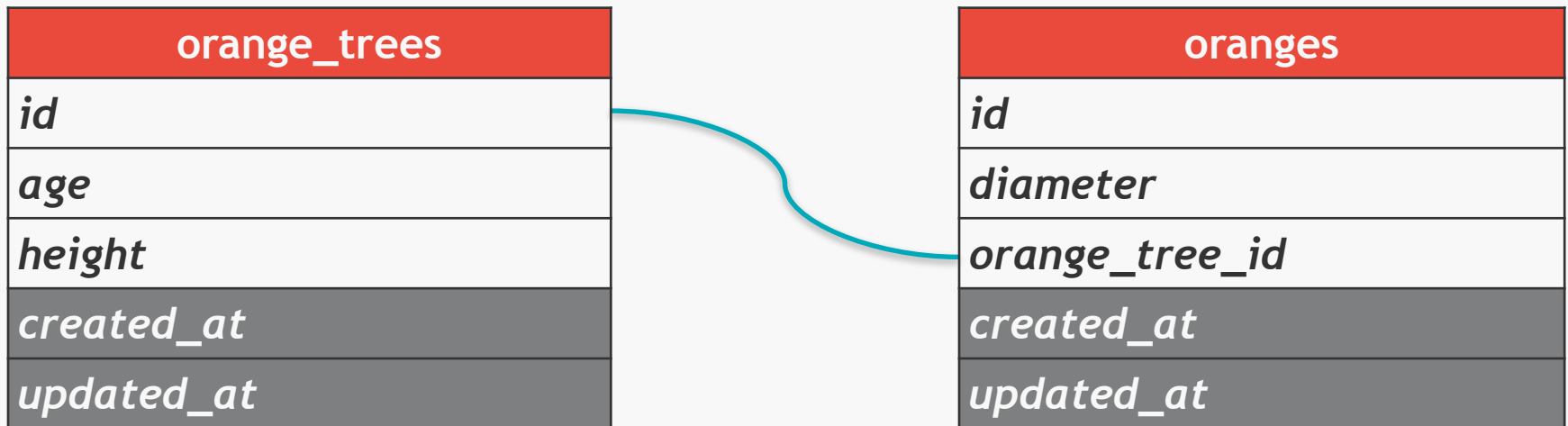


```
▼ QuEst
  ▼ app
    > assets
    > controllers
    > helpers
    > mailers
    ▼ models
      > concerns
      .keep
      choice.rb
      course.rb
      question.rb
      quiz.rb
      student.rb
```

Associations Between Classes

We are dealing with relational databases.

Relationships == Associations



Associations Between Classes

Define associations between classes in their respective models.

```
# root/app/models/orange_tree.rb
```

```
class OrangeTree < ActiveRecord::Base
  has_many :oranges
end
```

```
# root/app/models/orange.rb
```

```
class Orange < ActiveRecord::Base
  belongs_to :orange_tree
end
```

Associations Between Classes

oranges				
<i>id</i>	<i>diameter</i>	<i>orange_tree_</i>	<i>created_at</i>	<i>updated_at</i>
1	2	1	2014-03-22	2014-03-22
2	4	2	2014-03-22	2014-03-22

```
tree = OrangeTree.find(1)
```

```
# => #<OrangeTree:0x007fdd5b9b4a20 @id=1, @age=5, @height=5 ...>
```

```
tree.oranges
```

```
# => [#<Orange:0x003frd5b9t8a24 @id=1, @diameter=2 ...>]
```

Associations Between Classes

oranges				
<i>id</i>	<i>diameter</i>	<i>orange_tree_</i>	<i>created_at</i>	<i>updated_at</i>
1	2	1	2014-03-22	2014-03-22
2	4	2	2014-03-22	2014-03-22

```
orange = Orange.find(1)
```

```
# => #<Orange:0x003frd5b9t8a24 @id=1, @diameter=2 ...>
```

```
orange.orange_tree
```

```
# => #<OrangeTree:0x007fdd5b9b4a20 @id=1, @age=5, @height=5 ...>
```


Validations

- Prevent writing to the database if your validations don't pass.
- **IMPORTANT**— Always make sure your migrations and models are **Rock Solid** before implementing Validations.

Validations

```
CREATE TABLE orange_trees (  
  id INTEGER PRIMARY KEY  
    AUTOINCREMENT,  
  age INTEGER NOT NULL,  
  height INTEGER NOT NULL,  
  created_at DATETIME,  
  updated_at DATETIME);
```

Validations

```
CREATE TABLE orange_trees (  
  id INTEGER PRIMARY KEY  
    AUTOINCREMENT,  
  age INTEGER NOT NULL,  
  height INTEGER NOT NULL,  
  created_at DATETIME,  
  updated_at DATETIME);
```

```
class OrangeTree < ActiveRecord::Base  
  validates :age, presence: true  
  validates :height, presence: true  
end
```

Validations

- ActiveRecord has built in validators
 - presence
 - format
 - uniqueness
 - etc.

Convention over Configuration

Lots of really smart people have already cranked through the vast majority of problems you are going to face. They were kind enough to hook you up with the fruits of their labor, or what we affectionately call Best Practices.

- **Convention:** Write less code, straight forward and clean approach to make your life easier.
- **Configuration:** Breaks from standard practice when absolutely required to get the job done.

Custom Validations

- You can also write your own validators

```
class Orange < ActiveRecord::Base
  validate :legit_diameter

  def legit_diameter
    errors.add_to_base("Not Legit") unless orange.diameter > 2
  end
end
```

Failed Validations add Errors

- Before writing to the database, ActiveRecord runs validations from the model
- If a validation fails:
 - An error is added to the object
 - ActiveRecord will not write to the database

Callbacks

Callbacks are methods that will run at a given point in your objects life cycle within the DB.

Callbacks

A few of the many callbacks available

before_validation
before_save
before_update
before_create
before_destroy

after_validation
after_create
after_save
after_update
after_destroy

Callbacks

```
class Orange < ActiveRecord::Base

  belongs_to :orange_tree

  after_initialize do |orange|
    puts "I made an Orange!!!"
  end

end
```

Control Code



Control Code

MODEL:

```
class OrangeTree < ActiveRecord::Base  
  has_many :oranges  
End
```

Control Code accesses this model:

```
tree = OrangeTree.new(age: 6, height: 15)
```

Now your program has access to this DB object in local memory

Mapping Classes to Tables

- Class methods:
Retrieve records from table
- Instance methods:
Read and write values

Class Methods

orange_trees				
<i>id</i>	<i>age</i>	<i>height</i>	<i>created_at</i>	<i>updated_at</i>
1	5	5	2014-03-22	2014-03-22
2	6	6	2014-03-22	2014-03-22

OrangeTree.all

```
# => [#<OrangeTree:0x007fdd5b9b4a20 @id=1, @age=5, @height=5 ...>,  
      #<OrangeTree:0x007fdd5b22b268 @id=2, @age=6, @height=6 ...>]
```

OrangeTree.find(1)

```
# => #<OrangeTree:0x007fdd5b9b4a20 @id=1, @age=5, @height=5 ...>
```

Instance Methods Match Fields

orange_trees				
<i>id</i>	<i>age</i>	<i>height</i>	<i>created_at</i>	<i>updated_at</i>
1	5	5	2014-03-22	2014-03-22
2	6	6	2014-03-22	2014-03-22

```
tree = OrangeTree.find(1)
# => #<OrangeTree:0x007fdd5b9b4a20 @id=1, @age=5, @height=5 ...>
```

```
tree.id      # => 1
tree.age     # => 5
tree.height  # => 5
```

Instance Methods Match Fields

orange_trees				
<i>id</i>	<i>age</i>	<i>height</i>	<i>created_at</i>	<i>updated_at</i>
1	5	5	2014-03-22	2014-03-22
2	6	6	2014-03-22	2014-03-22

```
tree = OrangeTree.find(1)
# => #<OrangeTree:0x007fdd5b9b4a20 @id=1, @age=5, @height=5 ...>

tree.age = 9 # => 9
tree
# => #<OrangeTree:0x007fdd5b9b4a20 @id=1, @age=9, @height=5 ...>
```


Instance Methods Match Fields

orange_trees				
<i>id</i>	<i>age</i>	<i>height</i>	<i>created_at</i>	<i>updated_at</i>
1	9	5	2014-03-22	2014-03-22
2	6	6	2014-03-22	2014-03-22

```
tree = OrangeTree.find(1)
# => #<OrangeTree:0x007fdd5b9b4a20 @id=1, @age=5, @height=5 ...>

tree.age = 9 # => 9
tree
# => #<OrangeTree:0x007fdd5b9b4a20 @id=1, @age=9, @height=5 ...>
tree.save    # => true
```

CRUD

```
### C.R.U.D. ###
```

```
### CREATE ###
```

```
p1 = Person.create(first_name: "Tami", age: 26)
```

```
p2 = Person.new(first_name: "Bill", age: 46)
```

```
p2.save
```

```
### READ ###
```

```
p3 = Person.find(1)
```

```
p4 = Person.find_by_first_name("Bill")
```

```
### UPDATE ###
```

```
p1.update_attributes(age: 27)
```

```
### Destroy ###
```

```
p2.destroy
```



THE END

Introduction to Active Record: Our Object-relational Mapper