

In this lecture, we will discuss...

- ✧ Rich Many-to-Many association
- ✧ ActiveRecord Calculations



Rich Many-to-Many Association

- ✧ Sometimes, you need to **keep some data** on the join table or
- ✧ You need to store **grandchild** relationships on a model, like user → articles → comments
- ✧ In our case – all salary ranges for a particular person

Rich Many-to-Many Association

- ✧ ActiveRecord provides a `:through` option for this purpose
- ✧ **Basic idea:** you first **create** a **regular** parent-child relationship and then **use the child model as a “join”** between the parent and grandchild



SalaryRange Model and Migration

```
~/advanced_ar$ rails g model salary_range min_salary:float max_salary:float job:references
  invoke  active_record
  create   db/migrate/20150922165025_create_salary_ranges.rb
  create   app/models/salary_range.rb
  invoke   test_unit
  create    test/models/salary_range_test.rb
  create    test/fixtures/salary_ranges.yml
~/advanced_ar$ rake db:migrate
== 20150922165025 CreateSalaryRanges: migrating =====
-- create_table(:salary_ranges)
   -> 0.0014s
== 20150922165025 CreateSalaryRanges: migrated (0.0014s) =====
```

```
class CreateSalaryRanges < ActiveRecord::Migration
  def change
    create_table :salary_ranges do |t|
      t.float :min_salary
      t.float :max_salary
      t.references :job, index: true, foreign_key: true

      t.timestamps null: false
    end
  end
end
```



Job and SalaryRange Models

job.rb

*



```
1 class Job < ActiveRecord::Base
2   belongs_to :person
3   has_one :salary_range
4 end
```

salary_range.rb

*



```
1 class SalaryRange < ActiveRecord::Base
2   belongs_to :job
3 end
4
```



Person To SalaryRange Pathway

The screenshot shows a code editor with two files open: `person.rb` and `job.rb`. The left sidebar displays a file tree under the 'FOLDERS' section, showing the project structure. The `person.rb` file contains a `Person` class that inherits from `ActiveRecord::Base`. It has several associations: `has_one :personal_info` (dependent: :destroy), `has_many :my_jobs` (class_name: "Job"), `has_and_belongs_to_many :hobbies`, `has_many :jobs`, and `has_many :approx_salaries` (through: :jobs, source: :salary_range). The `job.rb` file contains a `Job` class that inherits from `ActiveRecord::Base`. It has two associations: `belongs_to :person` and `has_one :salary_range`. A red arrow points from the `has_many :approx_salaries` line in `person.rb` to the `has_one :salary_range` line in `job.rb`, indicating the pathway for the association.

FOLDERS

- ▼ advanced_ar
 - ▼ app
 - ▶ assets
 - ▶ controllers
 - ▶ helpers
 - ▶ mailers
 - ▼ models
 - ▶ concerns
 - .keep
 - hobby.rb
 - job.rb
 - person.rb

person.rb

```
1 class Person < ActiveRecord::Base
2   has_one :personal_info, dependent: :destroy
3   has_many :my_jobs, class_name: "Job"
4   has_and_belongs_to_many :hobbies
5   has_many :jobs
6   has_many :approx_salaries, through: :jobs, source: :salary_range
7 end
```

job.rb

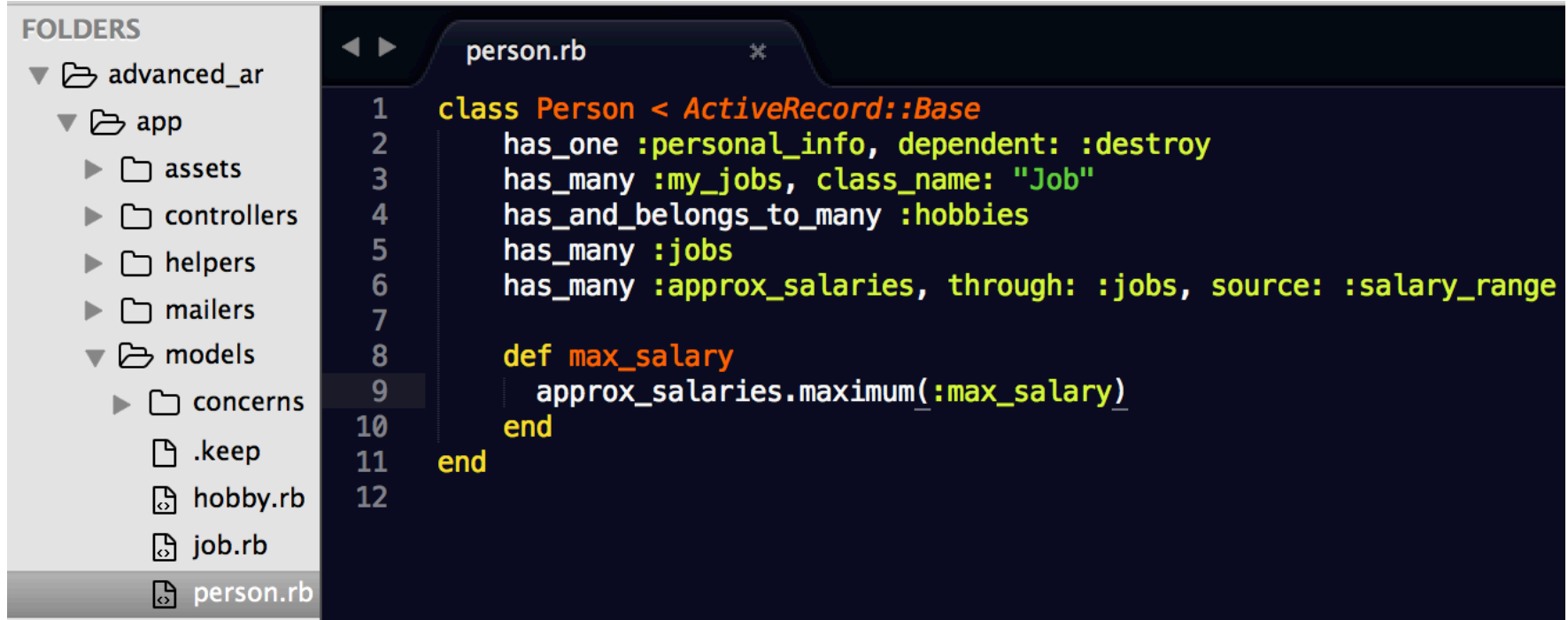
```
1 class Job < ActiveRecord::Base
2   belongs_to :person
3   has_one :salary_range
4 end
```

Person's Salary Ranges

```
irb(main):002:0> lebron = Person.find_by(first_name: "LeBron")
=> #<Person id: 35, first_name: "LeBron", age: 30, last_name: "James", created_at: "2015-09-22 15:06:15", updated_at: "2015-09-22 15:06:15", ass: "need more rings">
irb(main):003:0> lebron.jobs.count
=> 2
irb(main):004:0> lebron.jobs.pluck(:id)
=> [12, 13]
irb(main):005:0> Job.find(12).create_salary_range(min_salary: 10000.00, max_salary: 20000.00)
=> #<SalaryRange id: 1, min_salary: 10000.0, max_salary: 20000.0, job_id: 12, created_at: "2015-09-22 17:25:01", updated_at: "2015-09-22 17:25:01">
irb(main):006:0> Job.find(13).create_salary_range(min_salary: 15000.00, max_salary: 35000.00)
=> #<SalaryRange id: 2, min_salary: 15000.0, max_salary: 35000.0, job_id: 13, created_at: "2015-09-22 17:25:33", updated_at: "2015-09-22 17:25:33">
irb(main):007:0> lebron.approx_salaries
=> #<ActiveRecord::Associations::CollectionProxy [#<SalaryRange id: 1, min_salary: 10000.0, max_salary: 20000.0, job_id: 12, created_at: "2015-09-22 17:25:01", updated_at: "2015-09-22 17:25:01">, #<SalaryRange id: 2, min_salary: 15000.0, max_salary: 35000.0, job_id: 13, created_at: "2015-09-22 17:25:33", updated_at: "2015-09-22 17:25:33">]>
```



Person's Salary Ranges – Max Salary



```
1 class Person < ActiveRecord::Base
2   has_one :personal_info, dependent: :destroy
3   has_many :my_jobs, class_name: "Job"
4   has_and_belongs_to_many :hobbies
5   has_many :jobs
6   has_many :approx_salaries, through: :jobs, source: :salary_range
7
8   def max_salary
9     approx_salaries.maximum(:max_salary)
10  end
11 end
12
```

Average, minimum and sum also available...
(<http://api.rubyonrails.org/classes/ActiveRecord/Calculations.html>)



Person's Salary Ranges – Max Salary

```
irb(main):001:0> lebron = Person.find_by last_name: "James"
  Person Load (0.2ms) SELECT "people".* FROM "people" WHERE "people"."last_name" = ? LIMIT 1
[["last_name", "James"]]
=> #<Person id: 35, first_name: "LeBron", age: 30, last_name: "James", created_at: "2015-09-22 1
5:06:15", updated_at: "2015-09-22 15:06:15", login: "bron", pass: "need more rings">
irb(main):002:0> lebron.max_salary
  (0.2ms) SELECT MAX("salary_ranges"."max_salary") FROM "salary_ranges" INNER JOIN "jobs" ON "
salary_ranges"."job_id" = "jobs"."id" WHERE "jobs"."person_id" = ? [["person_id", 35]]
=> 35000.0
```



Summary

- ✧ Can get some pretty cool results with `:through`
- ✧ ActiveRecord calculations make your DB do the hard work (a good thing)

What's Next?

- ✧ Scopes

