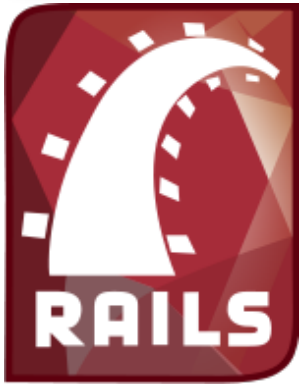

Many-to-Many Association

Many-to-Many association

- One person *can have many* hobbies
- One hobby can be *shared by many* people
- Hmm… Now, what?
- **Simple:** It's a case of `habtm` (has_and_belongs_to_many)
- Need to create an extra (a.k.a. join) table (without a model, i.e. just a migration)

Convention: Plural model names separated by an underscore in alphabetical order

Hobbies and Hobbies_People

```
hazink1:~/advanced_ar$ rails g model hobby name
      invoke  active_record
      create    db/migrate/20140713112446_create_hobbies.rb
      create    app/models/hobby.rb
      invoke    test_unit
      create      test/models/hobby_test.rb
      create      test/fixtures/hobbies.yml
hazink1:~/advanced_ar$ rails g migration create_hobbies_people hobby:references
 person:references
      invoke  active_record
      create    db/migrate/20140713112541_create_hobbies_people.rb
hazink1:~/advanced_ar$ rake db:migrate
== 20140713112446 CreateHobbies: migrating ==================================
-- create_table(:hobbies)
   -> 0.1031s
== 20140713112446 CreateHobbies: migrated (0.1032s) =========================

== 20140713112541 CreateHobbiesPeople: migrating ============================
-- create_table(:hobbies_people, {:id=>false})
   -> 0.0021s
== 20140713112541 CreateHobbiesPeople: migrated (0.0022s) ===================
```

Hobbies and Hobbies_People

20140713112541_create_hobbies_people.rb

```ruby
class CreateHobbiesPeople < ActiveRecord::Migration
  def change
    create_table :hobbies_people, id:false do |t|
      t.references :hobby, index: true
      t.references :person, index: true
    end
  end
end
```

No need for primary key, since this is a join table

Many-to-Many in the DB

```
hazink1:~/advanced_ar$ rails db
SQLite version 3.7.12 2012-04-03 19:43:07
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .schema %hobbies%
CREATE TABLE "hobbies" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
 "name" varchar(255), "created_at" datetime, "updated_at" datetime);
CREATE TABLE "hobbies_people" ("hobby_id" integer, "person_id" integer);
CREATE INDEX "index_hobbies_people_on_hobby_id" ON "hobbies_people" ("ho
bby_id");
CREATE INDEX "index_hobbies_people_on_person_id" ON "hobbies_people" ("p
erson_id");
```

Person and Hobby models

person.rb

```ruby
class Person < ActiveRecord::Base
    has_one :personal_info, dependent: :destroy
    has_many :jobs
    has_many :my_jobs, class_name: "Job"
    has_and_belongs_to_many :hobbies
end
```

hobby.rb

```ruby
class Hobby < ActiveRecord::Base
  has_and_belongs_to_many :persons
end
```

Person and Hobby in action

```
irb(main):002:0> josh = Person.find_by first_name: "Josh"
=> #<Person id: 18, first_name: "Josh", last_name: "Oreck", age: 57, created_at: "
2014-07-11 17:55:16", updated_at: "2014-07-11 17:55:16", login: "josh", pass: "not
_telling">
irb(main):003:0> programming = Hobby.find_by name: "Programming"
=> #<Hobby id: 2, name: "Programming", created_at: "2014-07-13 11:38:48", updated_
at: "2014-07-13 11:38:48">
irb(main):004:0> josh.hobbies << programming
=> #<ActiveRecord::Associations::CollectionProxy [#<Hobby id: 2, name: "Programmin
g", created_at: "2014-07-13 11:38:48", updated_at: "2014-07-13 11:38:48">]>
irb(main):005:0> lebron = Person.find_by first_name: "LeBron"
=> #<Person id: 21, first_name: "LeBron", last_name: "James", age: 30, created_at:
 "2014-07-11 17:55:16", updated_at: "2014-07-11 17:55:16", login: "lebron", pass:
"lejames">
irb(main):006:0> programming.persons << lebron
=> #<ActiveRecord::Associations::CollectionProxy [#<Person id: 18, first_name: "Jo
sh", last_name: "Oreck", age: 57, created_at: "2014-07-11 17:55:16", updated_at: "
2014-07-11 17:55:16", login: "josh", pass: "not_telling">, #<Person id: 21, first_
name: "LeBron", last_name: "James", age: 30, created_at: "2014-07-11 17:55:16", up
dated_at: "2014-07-11 17:55:16", login: "lebron", pass: "lejames">]>
irb(main):007:0> programming.persons
=> #<ActiveRecord::Associations::CollectionProxy [#<Person id: 18, first_name: "Jo
sh", last_name: "Oreck", age: 57, created_at: "2014-07-11 17:55:16", updated_at: "
2014-07-11 17:55:16", login: "josh", pass: "not_telling">, #<Person id: 21, first_
name: "LeBron", last_name: "James", age: 30, created_at: "2014-07-11 17:55:16", up
dated_at: "2014-07-11 17:55:16", login: "lebron", pass: "lejames">]>
```

people instead of persons

hobby.rb

```ruby
class Hobby < ActiveRecord::Base
  has_and_belongs_to_many :persons
  has_and_belongs_to_many :people, class_name: "Person"
end
```

```
hazink1:~/advanced_ar$ rails c
Loading development environment (Rails 4.1.1)
irb(main):001:0> programming = Hobby.last
  Hobby Load (0.1ms)  SELECT  "hobbies".* FROM "hobbies"   ORDER BY "hobbies"."id" DESC
 LIMIT 1
=> #<Hobby id: 2, name: "Programming", created_at: "2014-07-13 11:38:48", updated_at: "
2014-07-13 11:38:48">
irb(main):002:0> programming.people.find_by first_name: "LeBron"
  Person Load (0.2ms)  SELECT  "people".* FROM "people" INNER JOIN "hobbies_people" ON
"people"."id" = "hobbies_people"."person_id" WHERE "hobbies_people"."hobby_id" = ? AND
"people"."first_name" = 'LeBron' LIMIT 1  [["hobby_id", 2]]
=> #<Person id: 21, first_name: "LeBron", last_name: "James", age: 30, created_at: "201
4-07-11 17:55:16", updated_at: "2014-07-11 17:55:16", login: "lebron", pass: "lejames">
```

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

```
hazink1:~/advanced_ar$ rails g model salary_range min_salary:float max_salary:float job:references
      invoke  active_record
      create    db/migrate/20140713120044_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
```

20140713120044_create_salary_ranges.rb

```ruby
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

      t.timestamps
    end
  end
end
```

```
hazink1:~/advanced_ar$ rake db:migrate
== 20140713120044 CreateSalaryRanges: migrating =========
-- create_table(:salary_ranges)
   -> 0.0144s
== 20140713120044 CreateSalaryRanges: migrated (0.0144s)
```

Job and SalaryRange Models

job.rb

```ruby
class Job < ActiveRecord::Base
  belongs_to :person
  has_one :salary_range
end
```

salary_range.rb

```ruby
class SalaryRange < ActiveRecord::Base
  belongs_to :job
end
```

Person to SalaryRange pathway

person.rb

```ruby
class Person < ActiveRecord::Base
    has_one :personal_info, dependent: :destroy
    has_many :my_jobs, class_name: "Job"
    has_and_belongs_to_many :hobbies
    has_many :jobs
    has_many :approx_salaries, through: :jobs, source: :salary_range
end
```

job.rb

```ruby
class Job < ActiveRecord::Base
  belongs_to :person
  has_one :salary_range
end
```

Person's salary ranges

```
hazink1:~/advanced_ar$ rails c
Loading development environment (Rails 4.1.1)
irb(main):001:0> ActiveRecord::Base.logger = nil
=> nil
irb(main):002:0> kalman = Person.find_by first_name: "Kalman"
=> #<Person id: 15, first_name: "Kalman", last_name: "Smith", age: 33, created_at: "2014-07-11 17:55:16",
 updated_at: "2014-07-11 17:55:16", login: "kalman", pass: "abc123">
irb(main):003:0> kalman.jobs.count
=> 2
irb(main):004:0> kalman.jobs.pluck(:id)
=> [3, 4]
irb(main):005:0> Job.find(3).create_salary_range min_salary: 20000.00, max_salary: 40000.00
=> #<SalaryRange id: 1, min_salary: 20000.0, max_salary: 40000.0, job_id: 3, created_at: "2014-07-13 12:5
2:47", updated_at: "2014-07-13 12:52:47">
irb(main):006:0> Job.find(4).create_salary_range min_salary: 80000.00, max_salary: 90000.00
=> #<SalaryRange id: 2, min_salary: 80000.0, max_salary: 90000.0, job_id: 4, created_at: "2014-07-13 12:5
3:21", updated_at: "2014-07-13 12:53:21">
irb(main):007:0> kalman.approx_salaries
=> #<ActiveRecord::Associations::CollectionProxy [#<SalaryRange id: 1, min_salary: 20000.0, max_salary: 4
0000.0, job_id: 3, created_at: "2014-07-13 12:52:47", updated_at: "2014-07-13 12:52:47">, #<SalaryRange i
d: 2, min_salary: 80000.0, max_salary: 90000.0, job_id: 4, created_at: "2014-07-13 12:53:21", updated_at:
 "2014-07-13 12:53:21">]>
```

Person's salary ranges – Max salary

person.rb

```
class Person < ActiveRecord::Base
    has_one :personal_info, dependent: :destroy
    has_many :my_jobs, class_name: "Job"
    has_and_belongs_to_many :hobbies
    has_many :jobs
    has_many :approx_salaries, through: :jobs, source: :salary_range

    def max_salary
      approx_salaries.maximum(:max_salary)
    end
end
```

Person's salary ranges – Max salary

```
hazink1:~/advanced_ar$ rails c
Loading development environment (Rails 4.1.1)
irb(main):001:0> kalman = Person.find_by first_name: "Kalman"
  Person Load (0.1ms)  SELECT  "people".* FROM "people"  WHERE "people"."first_name" = 'Kalman' LIMIT 1
=> #<Person id: 15, first_name: "Kalman", last_name: "Smith", age: 33, created_at: "2014-07-11 17:55:16",
 updated_at: "2014-07-11 17:55:16", login: "kalman", pass: "abc123">
irb(main):002:0> kalman.max_salary
   (0.2ms)  SELECT MAX("salary_ranges"."max_salary") AS max_id FROM "salary_ranges" INNER JOIN "jobs" ON
"salary_ranges"."job_id" = "jobs"."id" WHERE "jobs"."person_id" = ?  [["person_id", 15]]
=> 90000.0
```

Default Scope

- `default_scope block`
  - **Class** method for specifying how the records are retrieved by default from the database (instead of relying on the database default)

```
irb(main):001:0> Hobby.pluck(:name)
   (0.1ms)  SELECT "hobbies"."name" FROM "hobbies"
=> ["Soccer", "Programming"]
```

Default Scope example

- Use *unscoped* to break out of the default

hobby.rb

```
class Hobby < ActiveRecord::Base
  has_and_belongs_to_many :persons
  has_and_belongs_to_many :people, class_name: "Person"

  default_scope { order name: :asc }
end
```

```
irb(main):001:0> Hobby.pluck(:name)
   (0.1ms)  SELECT "hobbies"."name" FROM "hobbies"   ORDER BY "hobbies"."name" ASC
=> ["Programming", "Soccer"]
irb(main):002:0> Hobby.unscoped.pluck(:name)
   (0.2ms)  SELECT "hobbies"."name" FROM "hobbies"
=> ["Soccer", "Programming"]
```

Named Scopes

```
scope :name, lambda
```

person.rb

```ruby
class Person < ActiveRecord::Base
    has_one :personal_info, dependent: :destroy
    has_many :my_jobs, class_name: "Job"
    has_and_belongs_to_many :hobbies
    has_many :jobs
    has_many :approx_salaries, through: :jobs, source: :salary_range

    def max_salary
      approx_salaries.maximum(:max_salary)
    end

    scope :ordered_by_age, -> { order age: :desc }
    scope :starts_with, -> (search_string) { where("first_name LIKE ?", "#{search_string}%") }
end
```

Named Scopes - chaining

```
irb(main):001:0> Person.starts_with("Jo").ordered_by_age
  Person Load (1.1ms)  SELECT "people".* FROM "people"  WHERE (first_name LIKE 'Jo%')
  ORDER BY "people"."age" DESC
=> #<ActiveRecord::Relation [#<Person id: 18, first_name: "Josh", last_name: "Oreck",
 age: 57, created_at: "2014-07-11 17:55:16", updated_at: "2014-07-11 17:55:16", login
: "josh", pass: "not_telling">, #<Person id: 16, first_name: "John", last_name: "What
ever", age: 27, created_at: "2014-07-11 17:55:16", updated_at: "2014-07-11 17:55:16",
 login: "johnw", pass: "123abc">, #<Person id: 19, first_name: "John", last_name: "Sm
ith", age: 27, created_at: "2014-07-11 17:55:16", updated_at: "2014-07-11 17:55:16",
login: "john", pass: "secret">]>
irb(main):002:0> Person.starts_with("Jo").ordered_by_age.count
   (0.2ms)  SELECT COUNT(*) FROM "people"  WHERE (first_name LIKE 'Jo%')
=> 3
irb(main):003:0> Person.starts_with("Jo").ordered_by_age.pluck(:first_name)
   (0.3ms)  SELECT "people"."first_name" FROM "people"  WHERE (first_name LIKE 'Jo%')
  ORDER BY "people"."age" DESC
=> ["Josh", "John", "John"]
```