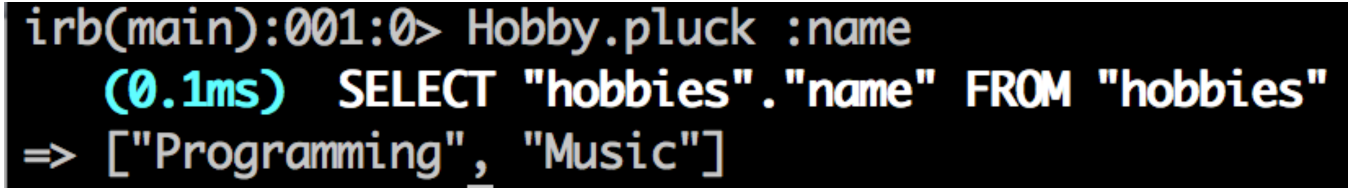
In this lecture, we will discuss…

- Default scope
- Named scopes

Default Scope

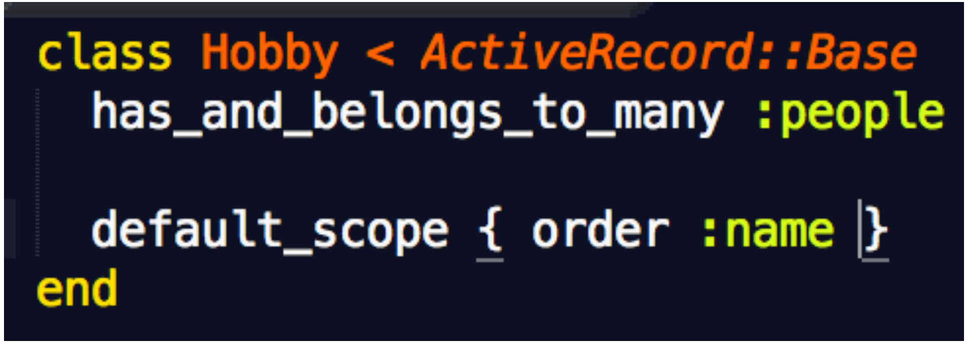
- `default_scope`
  - **Class** “method” for specifying how the records are retrieved by default from the database (instead of relying on the database default)



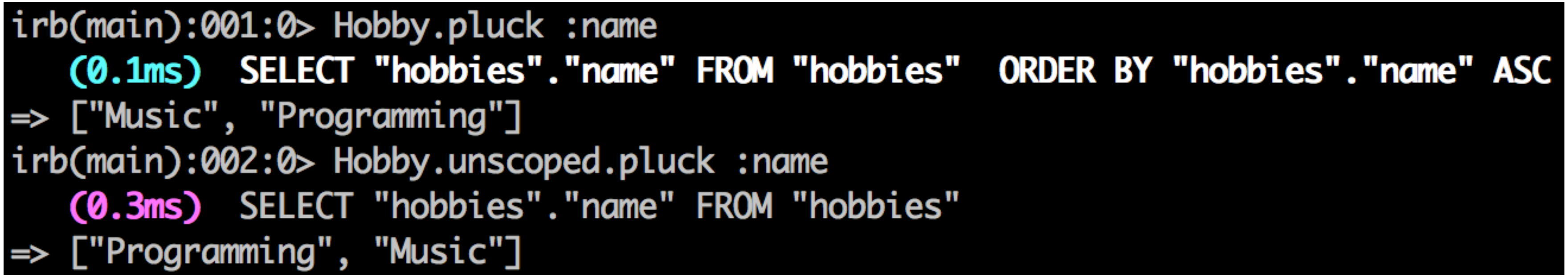
```
irb(main):001:0> Hobby.pluck :name
   (0.1ms)  SELECT "hobbies"."name" FROM "hobbies"
=> ["Programming", "Music"]
```

Default Scope Example

- Use *unscoped* to break out of the default



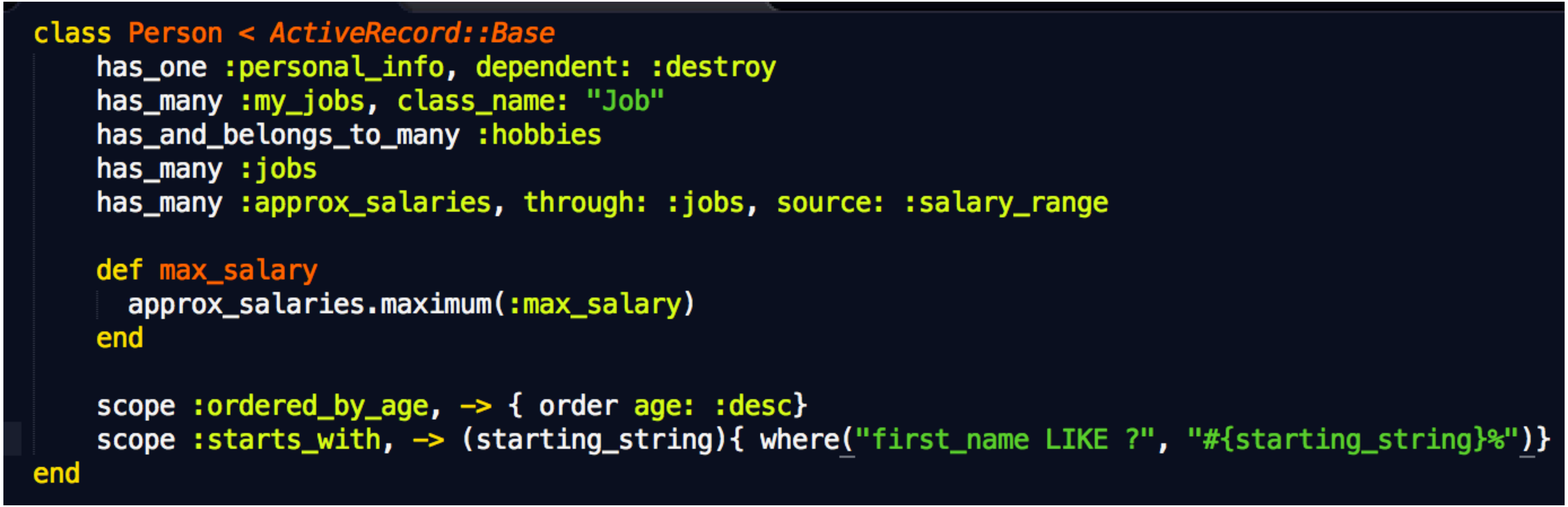
```ruby
class Hobby < ActiveRecord::Base
  has_and_belongs_to_many :people

  default_scope { order :name }
end
```



```
irb(main):001:0> Hobby.pluck :name
   (0.1ms)  SELECT "hobbies"."name" FROM "hobbies"  ORDER BY "hobbies"."name" ASC
=> ["Music", "Programming"]
irb(main):002:0> Hobby.unscoped.pluck :name
   (0.3ms)  SELECT "hobbies"."name" FROM "hobbies"
=> ["Programming", "Music"]
```

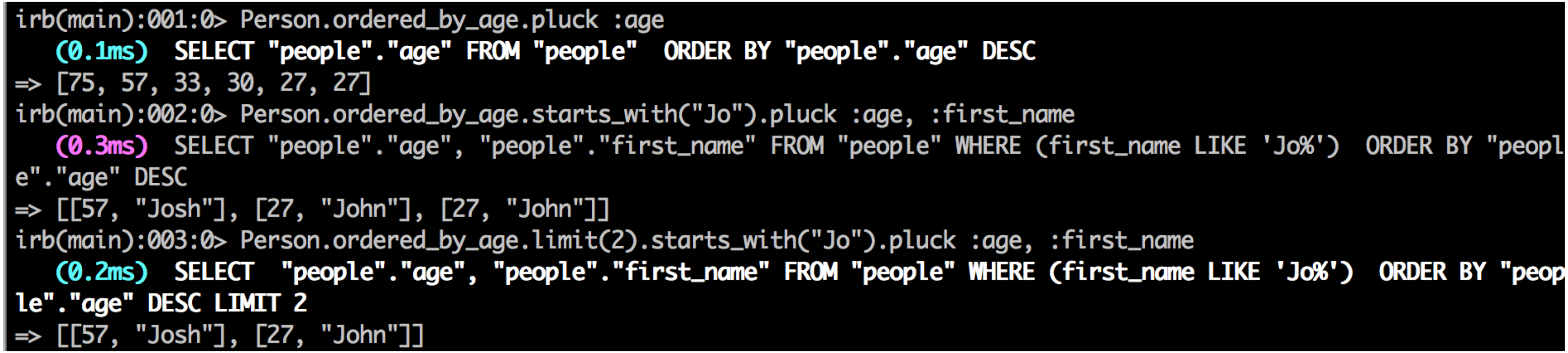
Named Scopes

- scope :name, lambda



```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, -> (starting_string){ where("first_name LIKE ?", "#{starting_string}%")}
end
```

Named Scopes - Chaining



```
irb(main):001:0> Person.ordered_by_age.pluck :age
   (0.1ms)  SELECT "people"."age" FROM "people"  ORDER BY "people"."age" DESC
=> [75, 57, 33, 30, 27, 27]
irb(main):002:0> Person.ordered_by_age.starts_with("Jo").pluck :age, :first_name
   (0.3ms)  SELECT "people"."age", "people"."first_name" FROM "people" WHERE (first_name LIKE 'Jo%')  ORDER BY "peopl
e"."age" DESC
=> [[57, "Josh"], [27, "John"], [27, "John"]]
irb(main):003:0> Person.ordered_by_age.limit(2).starts_with("Jo").pluck :age, :first_name
   (0.2ms)  SELECT  "people"."age", "people"."first_name" FROM "people" WHERE (first_name LIKE 'Jo%')  ORDER BY "peop
le"."age" DESC LIMIT 2
=> [[57, "Josh"], [27, "John"]]
```

Summary

- Scopes always return `ActiveRecord::Relation`
- Can make your ActiveRecord queries very expressive

What's Next?

- Validations