ActiveRecord

Ruby's Object Relational Mapping (ORM) Tool

Agenda

- ORMs
- ActiveRecord
- Queries
- Validations
- Associations

ORMs

- Object-relational mapping
- ORMs are specific to object oriented programming and achieves two things:
  1. It connects your Ruby classes to your database
  2. It abstracts out the writing of your SQL statements

ActiveRecord

- AR is Ruby's ORM
  - Java's ORM is called Hibernate. JS has an ORM called Sequelize, Python has an ORM called Django ORM.
  - All of these ORMs achieve those same two objectives: connecting your object-oriented framework's classes to your data and abstracting the writing of your SQL statements

Queries in SQL

```
SELECT * FROM posts;
  // Get me all the posts
SELECT * FROM posts WHERE id = 15;
  // Get me all the posts where the ID number is 15
select * FROM posts JOIN authors ON posts.author_id = authors.id WHERE authors.id = 20;
  // Get me all the posts where the author's ID is 20
```

Queries in ActiveRecord

```
Post.all
  # Get me all the posts
Post.find(15)
  # Get me all the posts where the ID number is 15
Post.joins(:authors).where('id' => 20)
  # Get me all the posts where the author's ID is 20
```

- Find all of the queries for ActiveRecord here

Validations

- Everyone is trying to attack your website. Protect your website.
- Validations should be everywhere from HTML/JS on your form to the model layer to the database
- ActiveRecord has validations that ensures that only *valid* data is saved into our database

```ruby
class BasketballPlayer < ActiveRecord::Base
  validates :name, presence: true
  validates :bio, length: { maximum: 500 }
  validates :games_played, numericality: { only_integer: true }
  validates :email, uniqueness: true
end
```

Associations

- Each SQL table is a *plural* noun. Each Ruby class that correlates to the database will inherit from **ActiveRecord::Base** and will be *singular*

```
class Student < ActiveRecord::Base
end
```

- Without the inheritance from AR, your Ruby class is a PORO (Plain Old Ruby Object). With the inheritance, your Ruby class is an extension of ActiveRecord - it becomes an ActiveRecord object
  - What does ActiveRecord give you?

Associations

- Associations is essentially data modeling your database. How are different tables associated with other tables? How are they related? How do they interact?
- As ActiveRecord beginners, we're primarily going to be dealing with 3 different associations: **belongs_to, has_many,** and **has_many through:**
- Let's create a small database for a high school with students and classes!