- Models are simple Ruby classes that represent real-world things
- In Rails, models are expected to be under **/app/models**
- Database-backed models should derive from ActiveRecord::Base
- **The "ez" gem takes care of all of this for you**
- Use **rails console** to load your Rails app and interact with your database
- **CRUD** with the console: **`create`**, **`read`**, **`update`**, and **`delete`**

Create

```
rocky = Movie.new
rocky.name = 'Rocky'
rocky.year_released = 1976
rocky.summary = 'Adrian!!!!'
rocky.save
```

```
rocky = Movie.create(name: 'Rocky',
year_released: 1976, summary = 'Adrian!!!!')
```

Read (a single record)

```
rocky = Movie.find_by(id: 1)
rocky.name
=> "Rocky"
```

Read (all records)

```
all_the_movies = Movie.all
```

Update

```
rocky = Movie.find_by(id: 1)
rocky.summary = ‘Boxing and stuff’
rocky.save
```

```
rocky = Movie.find_by(id: 1)
rocky.update(summary: ’Boxing and stuff’)
```

Delete

```
rocky = Movie.find_by(id: 1)
rocky.delete
```

- Rails gives us a way to interact with databases using **Ruby instead of SQL.**
- We still have all the powerful filtering, sorting, and limiting powers of SQL at our disposal.

Filtering & Limiting

```
Movie.where(year_released: 1978)
```

```
Movie.where(year_released: 1978).limit(10)
```

Sorting

```
Movie.order('name')
```

```
Movie.where(year_released: 1978).order('name')
```

- Things get really interesting when the ability to CRUD spans multiple models.
- **.find_by** returns one record
- **.where** returns multiple records

```
toy_story = Movie.find_by(name: “Toy Story”)
pixar = Studio.find_by(id:
toy_story.studio_id)
```

```
pixar = Studio.find_by(name: “Pixar”)
pixar_movies = Movie.where(studio_id:
pixar.id)
```