Models

Interface to the database

Abstractions for business logic

Aggregation

ORM

object relation mapping

Active Record

The default ORM for rails.

```ruby
class Topic < ActiveRecord::Base
  has_many :votes, dependent: :destroy
end
```

alternatives:
https://www.ruby-toolbox.com/categories/orm

Custom Methods

```
class Topic < ActiveRecord::Base
  has_many :votes, dependent: :destroy

  def self.sorted
    @topics = Topic.all.sort_by {|topic| topic.votes.count}.reverse
  end
end
```

Generator

```
$ rails generate model <name>
$ rake db:migrate
```

Manual Creation

Create the table in the database (migration or in the DB shell directly), then create a file in app/models.

```
class Topic < ActiveRecord::Base
end
```

For this to work, you must follow the model naming conventions.

Naming

Singular of the table name.

Model / Class	Table / Schema
Article	articles
LineItem	line_items
Deer	deers
Mouse	mice
Person	people

http://guides.rubyonrails.org/active_record_basics.html

Attributes

the columns in the table

```
ActiveRecord::Schema.define(version: 20150113071109) do

  create_table "topics", force: :cascade do |t|
    t.string   "title"
    t.text     "description"
    t.datetime "created_at",  null: false
    t.datetime "updated_at",  null: false
  end

  create_table "votes", force: :cascade do |t|
    t.integer  "topic_id"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end

end
"db/schema.rb" 29L, 1155C
```

Over-rides

Why?

Legacy database with non-descript names.

Table Name

```
class Topic < ActiveRecord::Base
  self.table_name = "input"
end
```

Column Name

```
class Topic < ActiveRecord::Base
  def description # alias
    self.title    # name in db table
  end
end
```

(alias)

rails console

interactive shell with your dev environment loaded

Adding DB Rows

new

```ruby
user = User.new do |u|
  u.name = "David"
  u.occupation = "Code Artist"
end
```

http://guides.rubyonrails.org/active_record_basics.html

save

```ruby
user = User.find_by(name: 'David')
user.name = 'Dave'
user.save
```

http://guides.rubyonrails.org/active_record_basics.html

create

```
user = User.create(name: "David", occupation: "Code Artist")
```

http://guides.rubyonrails.org/active_record_basics.html

Queries

all

```
Topic.all
```

http://guides.rubyonrails.org/active_record_basics.html

first

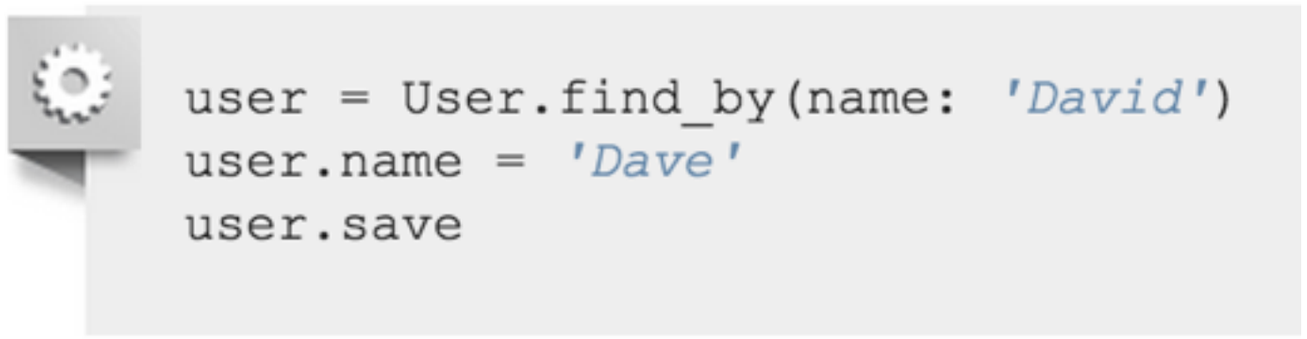
```
Topic.first
```

http://guides.rubyonrails.org/active_record_basics.html

sample

Topic.all.sample

find_by



```
user = User.find_by(name: 'David')
user.name = 'Dave'
user.save
```

http://guides.rubyonrails.org/active_record_basics.html

where

```ruby
# find all users named David who are Code Artists and sort by
created_at in reverse chronological order
users = User.where(name: 'David', occupation: 'Code
Artist').order('created_at DESC')
```

http://guides.rubyonrails.org/active_record_basics.html

order

```ruby
# find all users named David who are Code Artists and sort by
created_at in reverse chronological order
users = User.where(name: 'David', occupation: 'Code
Artist').order('created_at DESC')
```

http://guides.rubyonrails.org/active_record_basics.html

Editing

update

```ruby
user = User.find_by(name: 'David')
user.update(name: 'Dave')
```

http://guides.rubyonrails.org/active_record_basics.html

update_all

```ruby
user = User.find_by(name: 'David')
user.destroy
```

http://guides.rubyonrails.org/active_record_basics.html

Non DB Models

```
class Rando
  attr_accessor :topic

  def self.topic
    Topic.all.sample
  end
end
~
"rando.rb" 7L, 84C written
```

Put the in app/models and make sure the filename matches the class name.

Associations

has_one

```
class Supplier < ActiveRecord::Base
  has_one :account
end
```

has_many

```
class Customer < ActiveRecord::Base
  has_many :orders, dependent: :destroy
end
```

belongs_to

```
class Order < ActiveRecord::Base
  belongs_to :customer
end
```

Reading

http://guides.rubyonrails.org/active_record_basics.html
through section 5

http://guides.rubyonrails.org/association_basics.html
sections 1,2 and 4

http://guides.rubyonrails.org/active_record_migrations.html
through section 3

Homework

Design a model hierarchy for solving a real world problem. If you can't think of anything, you can use: (1) a library checkout system, (2) a patient records system for a hospital, (3) resource distribution in communist russia

Create a repo and upload a picture of your design to a branch called "design". Create a pull request to master and then email the PR url to mel.

When you have design approval, you can merge to master. Then, you will create, from scratch (no generate methods) a skeleton of the system (include empty definitions for any methods you think would be needed in the system). You'll need to write migrations, too. Put all of that in a branch called "models".

When you get approval for that, you'll merge again, and mel will give you UX requirements specific to your domain.