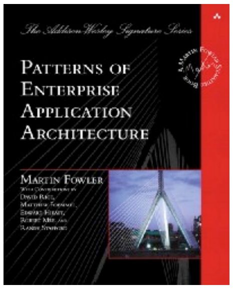
Vital Rails

Ruby on Rails Training

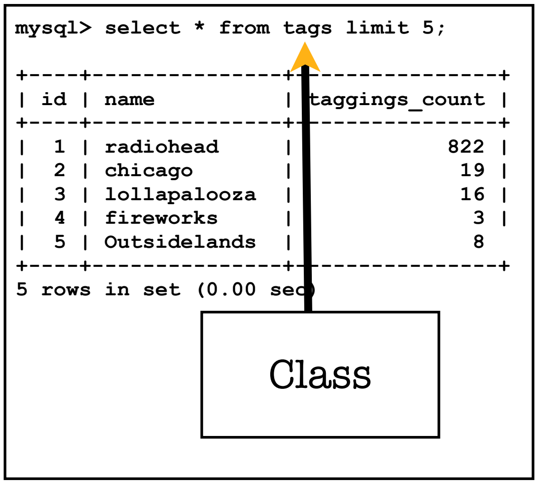
Intro

ActiveRecord

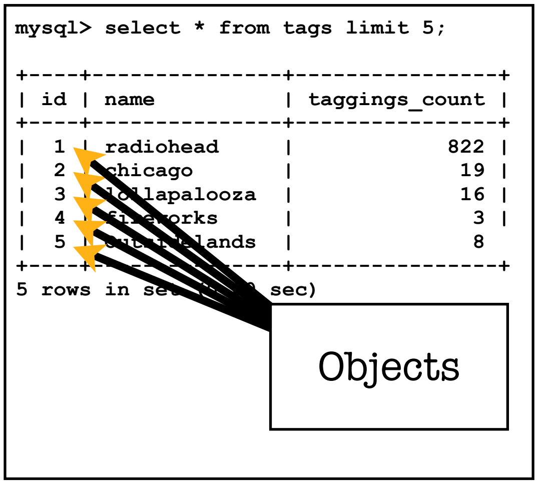
overview


The Addison-Wesley Signature Series
Patterns of Enterprise Application Architecture
Martin Fowler

images from martinfowler.com

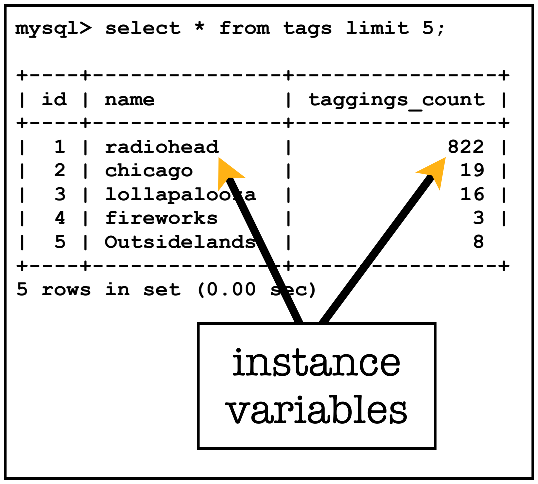


```
mysql> select * from tags limit 5;

+----+----------------+----------------+
| id | name           | taggings_count |
+----+----------------+----------------+
|  1 | radiohead      |            822 |
|  2 | chicago        |             19 |
|  3 | lollapalooza   |             16 |
|  4 | fireworks      |              3 |
|  5 | Outsidelands   |              8
+----+----------------+----------------+
5 rows in set (0.00 sec)
```



```
mysql> select * from tags limit 5;

+----+---------------+----------------+
| id | name          | taggings_count |
+----+---------------+----------------+
|  1 | radiohead     |            822 |
|  2 | chicago       |             19 |
|  3 | lollapalooza  |             16 |
|  4 | fireworks     |              3 |
|  5 | Outsidelands  |              8
+----+---------------+----------------+
5 rows in set (0.00 sec)
```

Objects



```
mysql> select * from tags limit 5;

+----+---------------+----------------+
| id | name          | taggings_count |
+----+---------------+----------------+
|  1 | radiohead     |            822 |
|  2 | chicago       |             19 |
|  3 | lollapalooza  |             16 |
|  4 | fireworks     |              3 |
|  5 | Outsidelands  |              8
+----+---------------+----------------+
5 rows in set (0.00 sec)
```

instance variables

additonal features

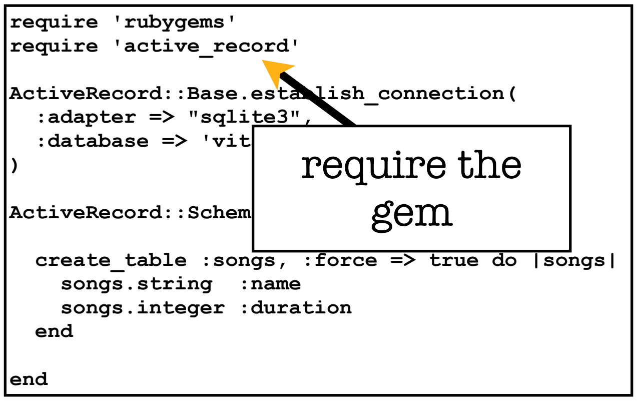
- validations
- associations
- migrations

diving in

init.rb

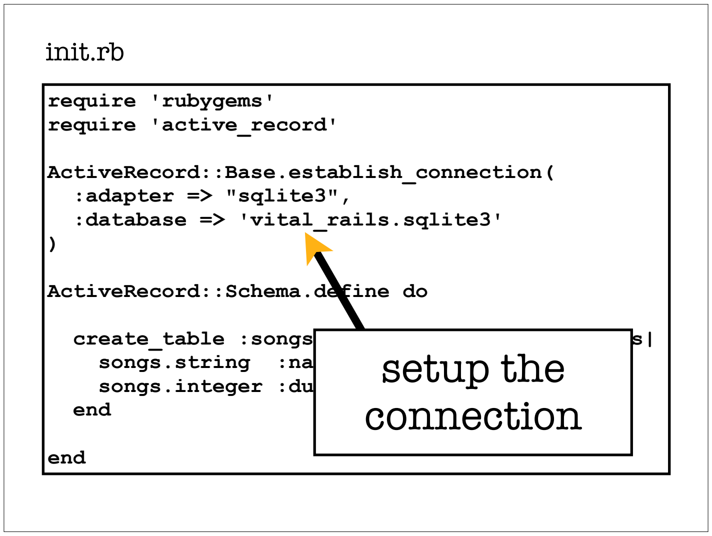
```
require 'rubygems'
require 'active_record'

ActiveRecord::Base.establish_connection(
  :adapter => "sqlite3",
  :database => 'vital_rails.sqlite3'
)

ActiveRecord::Schema.define do

  create_table :songs, :force => true do |songs|
    songs.string  :name
    songs.integer :duration
  end

end
```

init.rb



```
require 'rubygems'
require 'active_record'

ActiveRecord::Base.establish_connection(
  :adapter => "sqlite3",
  :database => 'vital_rails.sqlite3'
)

ActiveRecord::Schema.define do

  create_table :songs, :force => true do |songs|
    songs.string  :name
    songs.integer :duration
  end

end
```

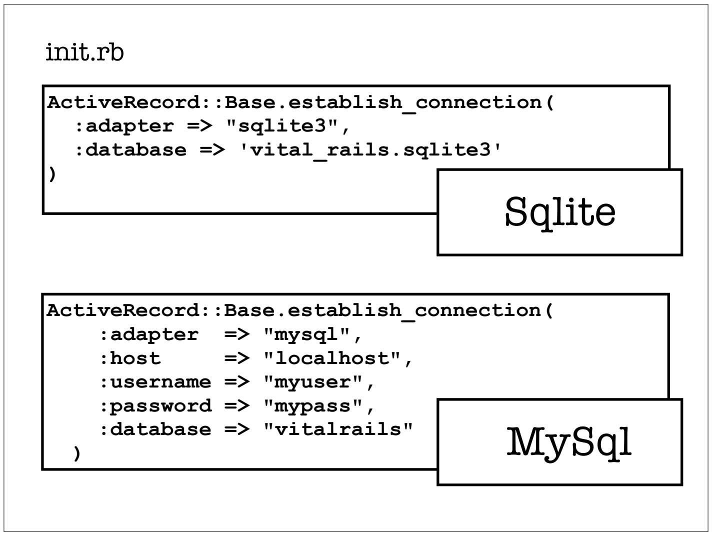
require the gem



init.rb

```
require 'rubygems'
require 'active_record'

ActiveRecord::Base.establish_connection(
  :adapter => "sqlite3",
  :database => 'vital_rails.sqlite3'
)

ActiveRecord::Schema.define do

  create_table :songs                          s|
    songs.string  :na
    songs.integer :du
  end

end
```

setup the connection



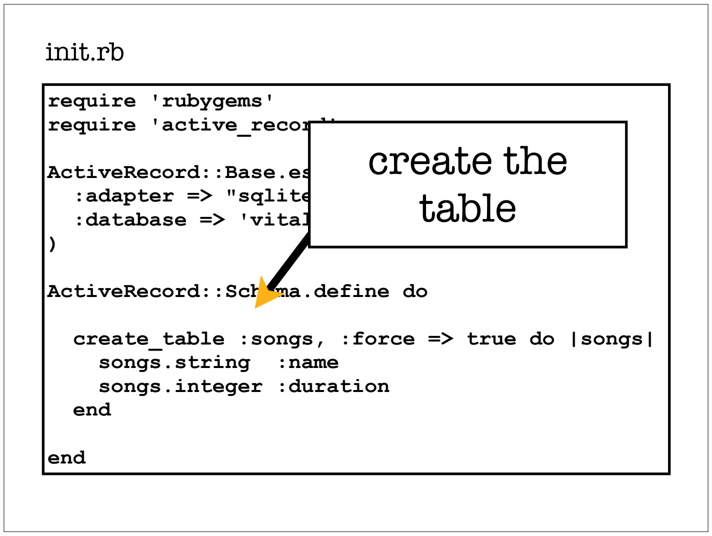
init.rb

```
ActiveRecord::Base.establish_connection(
  :adapter => "sqlite3",
  :database => 'vital_rails.sqlite3'
)
```

Sqlite

```
ActiveRecord::Base.establish_connection(
    :adapter  => "mysql",
    :host     => "localhost",
    :username => "myuser",
    :password => "mypass",
    :database => "vitalrails"
  )
```

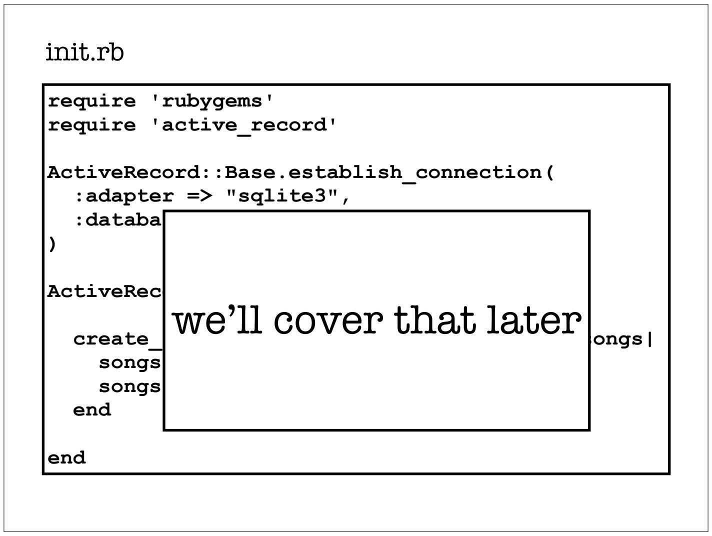
MySql



init.rb

```
require 'rubygems'
require 'active_recor

ActiveRecord::Base.es
  :adapter => "sqlite
  :database => 'vital
)

ActiveRecord::Schema.define do

  create_table :songs, :force => true do |songs|
    songs.string  :name
    songs.integer :duration
  end

end
```

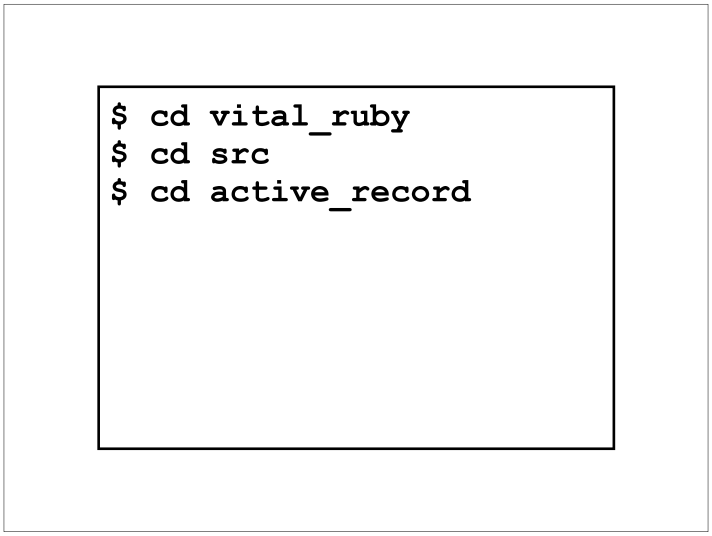
create the table



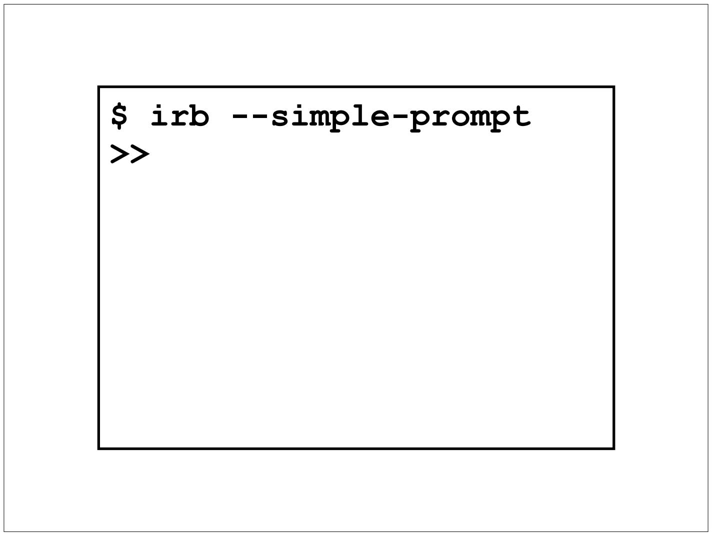
init.rb

```
require 'rubygems'
require 'active_record'

ActiveRecord::Base.establish_connection(
  :adapter => "sqlite3",
  :databa
)

ActiveRec

  create_                                   ongs|
    songs
    songs
  end

end
```

we’ll cover that later



```
$ cd vital_ruby
$ cd src
$ cd active_record
```



```
$ irb --simple-prompt
>>
```

```
>> require 'init'
-- create_table(:songs {:force=>true})
   -> 0.0477s
=> true
>>
```

```
>> require 'init'
-- create_table(:songs {:force=>true})
   -> 0.0477s
=> true
>>
```

CAUTION: drops and recreates table

```
>> require 'init'
-- create_table(:songs {:force=>true})
   -> 0.0477s
=> true
>>
```

if something goes wrong, just quit, and do this again

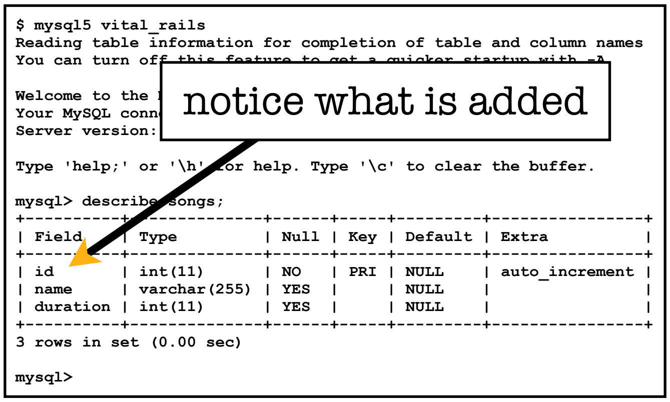
lets take a look under the hood ...

sqlite

```
$ sqlite3 vital_rails.sqlite3
SQLite version 3.4.0
Enter ".help" for instructions
sqlite> .schema songs
CREATE TABLE "songs" ("id" INTEGER
PRIMARY KEY AUTOINCREMENT NOT NULL,
"name" varchar(255), "duration"
integer);
sqlite>
```

MySQL

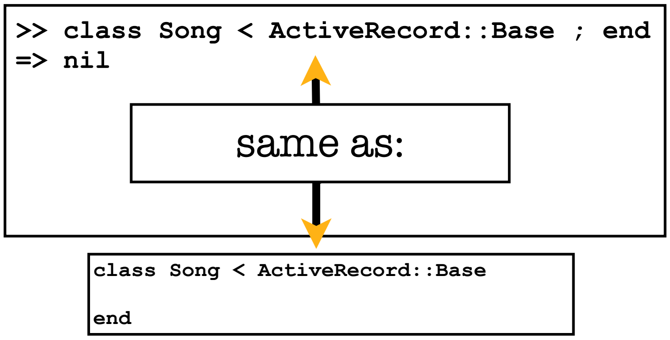
```
$ mysql5 vital_rails
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.1.30 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> describe songs;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| duration | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql>
```



```
$ mysql5 vital_rails
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the
Your MySQL conn
Server version:

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> describe songs;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| duration | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql>
```

notice what is added

creating the class



```
>> class Song < ActiveRecord::Base ; end
=> nil
```

same as:

```
class Song < ActiveRecord::Base

end
```

opinionated software

Rails is VERY opinionated much of the magic is dependent on conventions

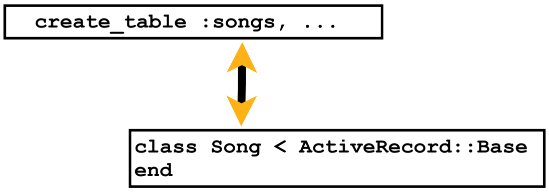
convention over configuration

instead of relying on external configuration, we rely on conventions

ActiveRecord Conventions

1

Table name is plural of class name (lowercase)



```
create_table :songs, ...
```

```
class Song < ActiveRecord::Base
end
```

ActiveRecord Conventions

2

all tables contain primary key (auto incrementing) named id

```
| id       | int(11)      | NO   | PRI | NULL    | auto_increment |
```

class methods for table actions

counting

```
>> Song.count
=> 1
```

retrieving

```
>> Song.find :all
=> [#<Song id: 1, name: "Allentown",
duration: 263>]
```

deleting

```
>> Song.destroy 1
=> #<Song id: 1, name: "Allentown",
duration: 263>
```

creating

```
>> Song.create :name => 'Allentown',
?> :duration => 263
=> #<Song id: 1, name: "Allentown",
duration: 263>
```

this is the one most often used in Rails

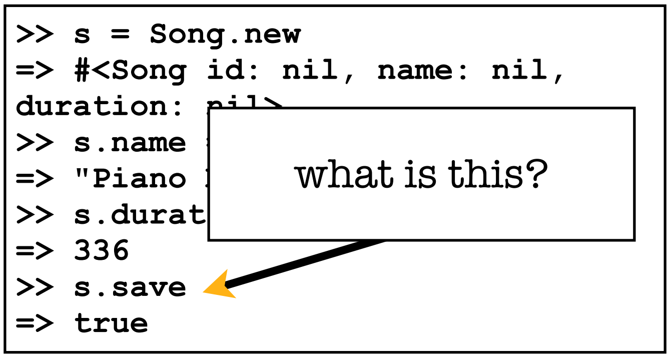
create some songs to play with

```
>> Song.create :name => 'Allentown',
:duration => 263
=> #<Song id: 1, name: "Allentown",
duration: 263>
>> Song.create :name => 'Uptown Girl',
:duration => 189
=> #<Song id: 2, name: "Uptown Girl",
duration: 189>
>> Song.create :name => 'An Innocent Man',
:duration => 369
=> #<Song id: 3, name: "An Innocent Man",
duration: 369>
```

several ways to create objects in ActiveRecord

```
>> Song.create do |s|
?>   s.name = "Baby Grand"
>>   s.duration = 246
>> end
```

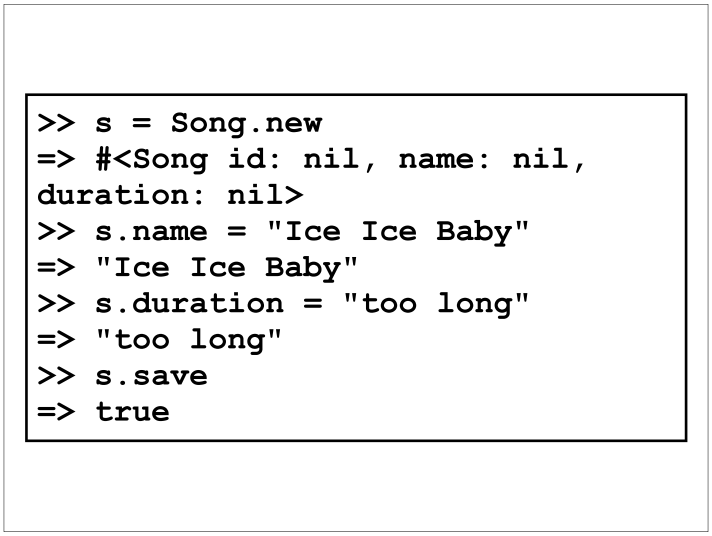
```
>> s = Song.new
=> #<Song id: nil, name: nil,
duration: nil>
>> s.name = "Piano Man"
=> "Piano Man"
>> s.duration = 336
=> 336
>> s.save
=> true
```



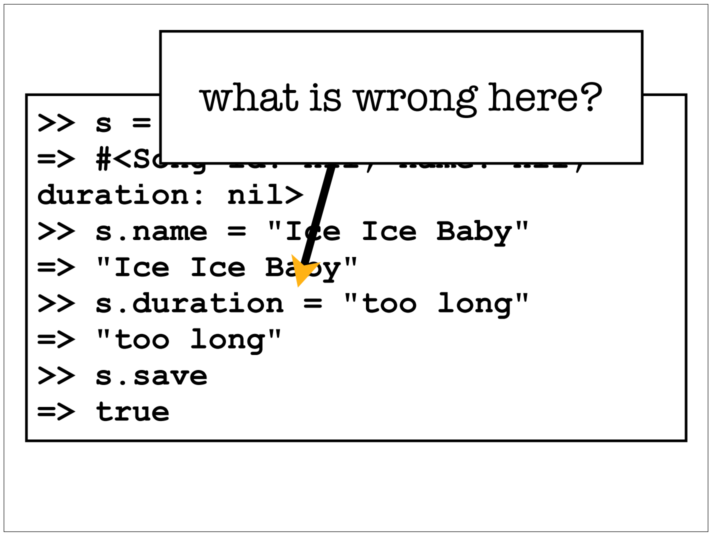
```
>> s = Song.new
=> #<Song id: nil, name: nil,
duration: nil>
>> s.name
=> "Piano
>> s.durat
=> 336
>> s.save
=> true
```

what is this?

validations

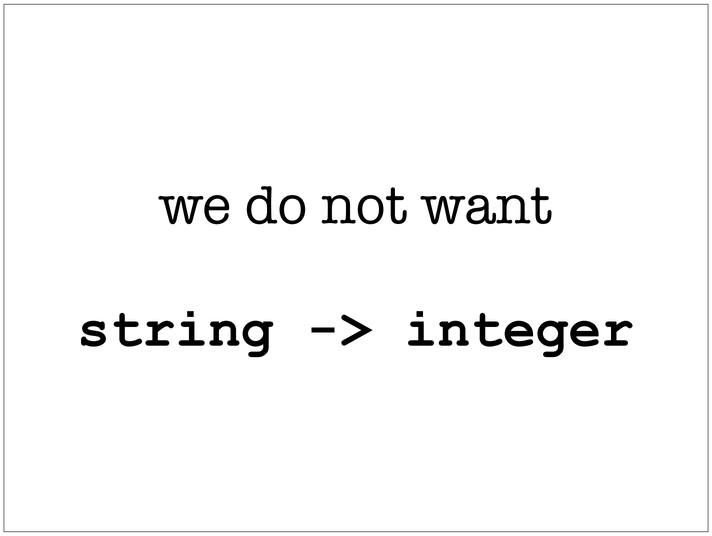


```
>> s = Song.new
=> #<Song id: nil, name: nil,
duration: nil>
>> s.name = "Ice Ice Baby"
=> "Ice Ice Baby"
>> s.duration = "too long"
=> "too long"
>> s.save
=> true
```



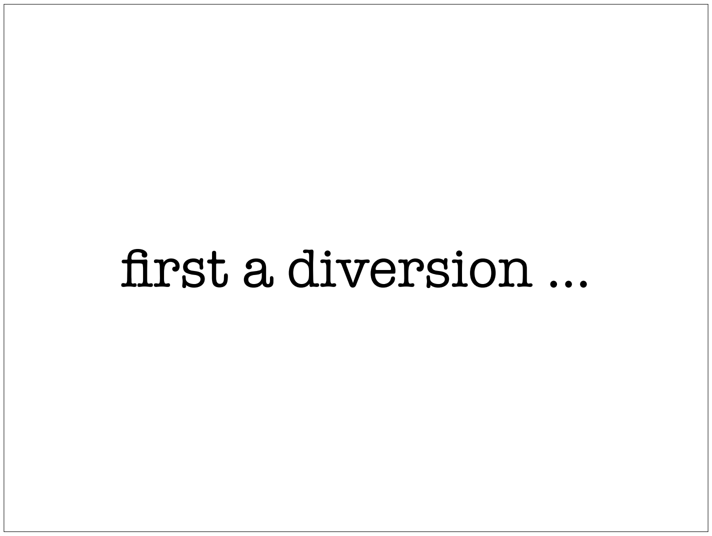
```
>> s =
=> #<Song id: nil, name: nil,
duration: nil>
>> s.name = "Ice Ice Baby"
=> "Ice Ice Baby"
>> s.duration = "too long"
=> "too long"
>> s.save
=> true
```

what is wrong here?

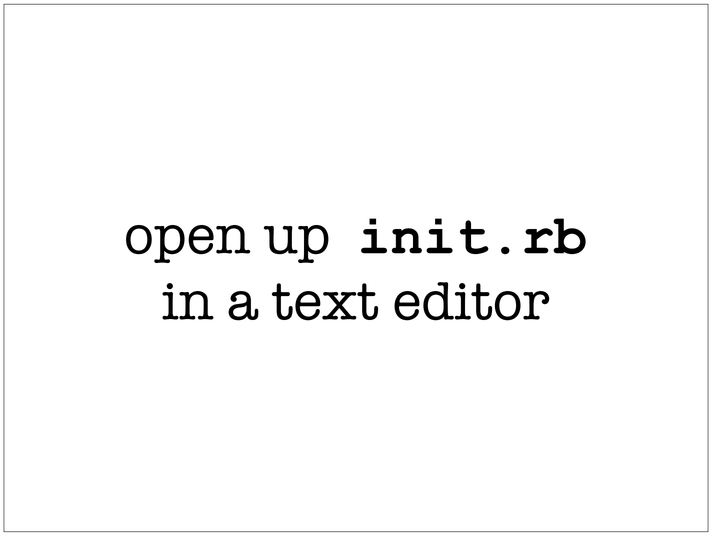


we do not want

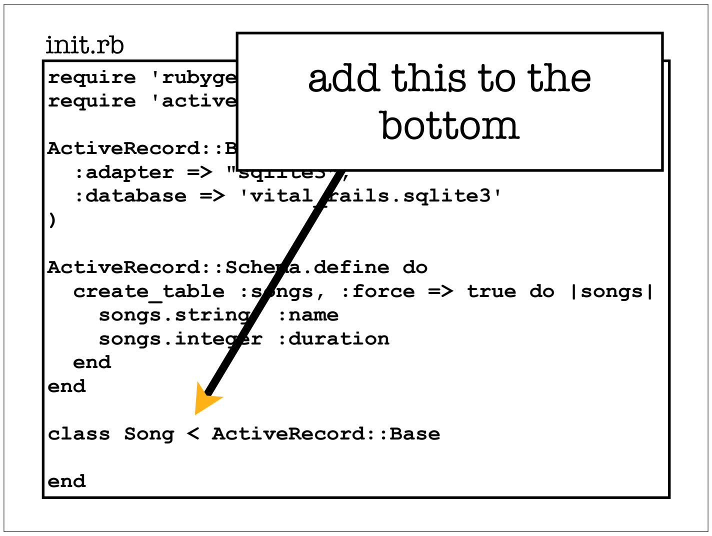
`string -> integer`



first a diversion ...



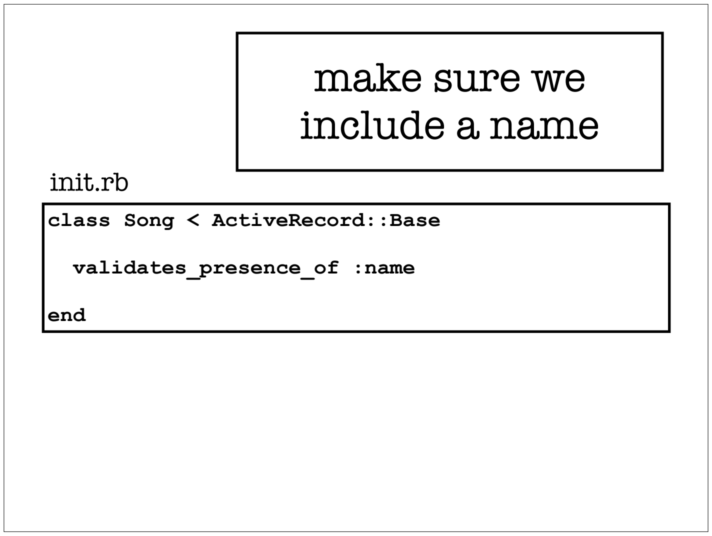
open up **`init.rb`**
in a text editor



init.rb

```
require 'rubyge
require 'active

ActiveRecord::B
  :adapter => "sqlite3",
  :database => 'vital_rails.sqlite3'
)

ActiveRecord::Schema.define do
  create_table :songs, :force => true do |songs|
    songs.string :name
    songs.integer :duration
  end
end

class Song < ActiveRecord::Base

end
```

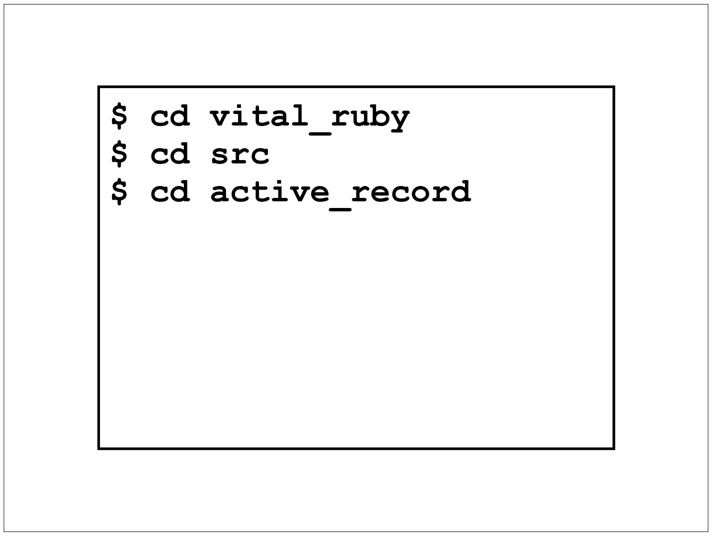
add this to the bottom



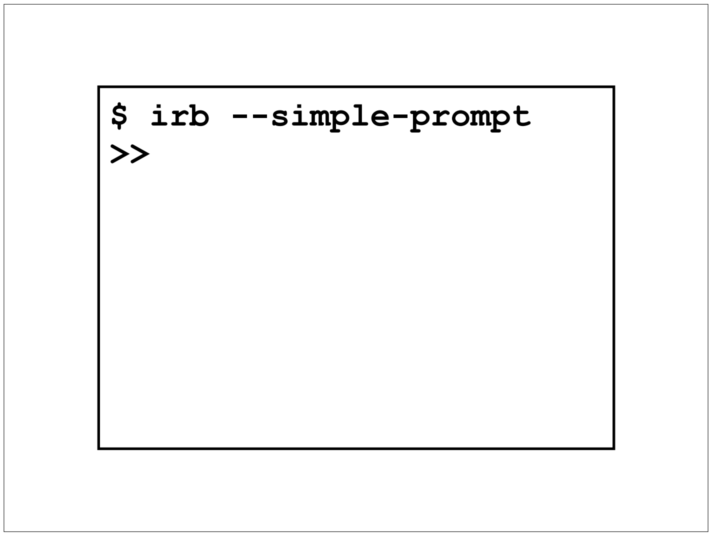
make sure we include a name

init.rb

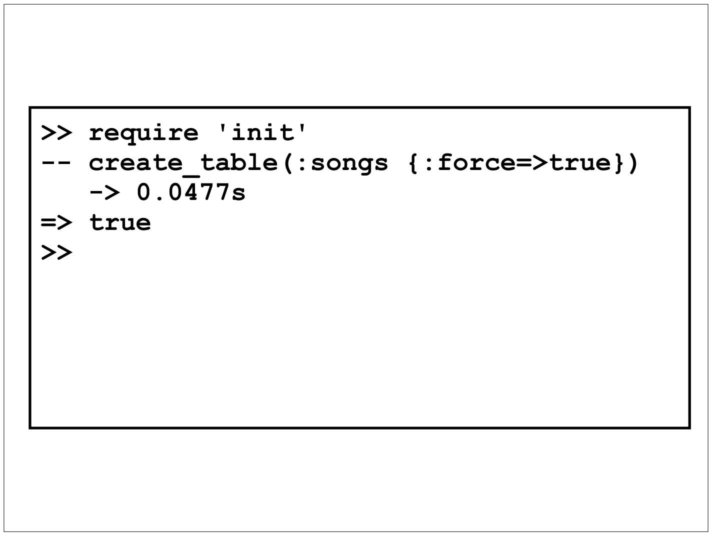
```
class Song < ActiveRecord::Base

  validates_presence_of :name

end
```



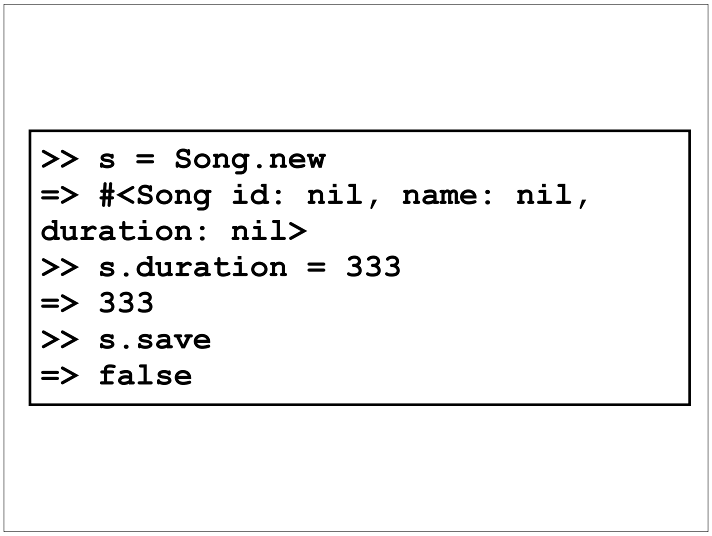
```
$ cd vital_ruby
$ cd src
$ cd active_record
```



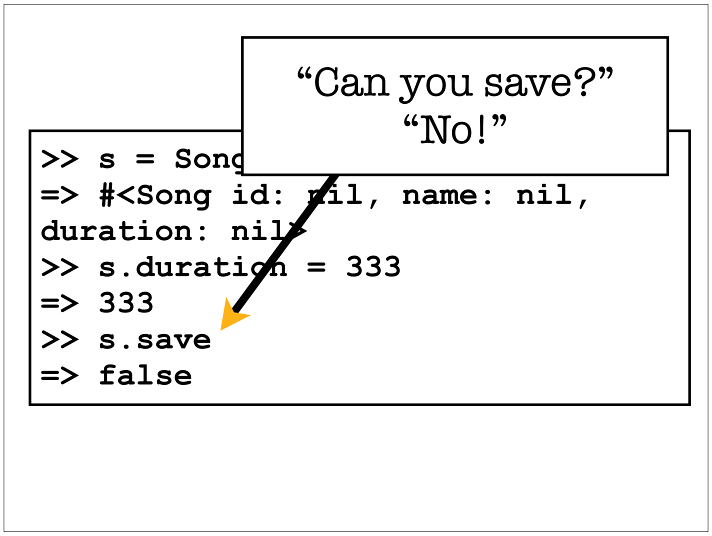
```
$ irb --simple-prompt
>>
```



```
>> require 'init'
-- create_table(:songs {:force=>true})
   -> 0.0477s
=> true
>>
```



```
>> s = Song.new
=> #<Song id: nil, name: nil,
duration: nil>
>> s.duration = 333
=> 333
>> s.save
=> false
```

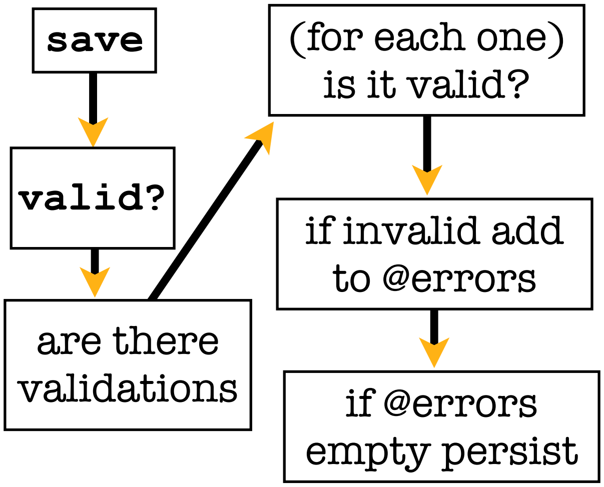


"Can you save?"
"No!"

```
>> s = Song
=> #<Song id: nil, name: nil,
duration: nil>
>> s.duration = 333
=> 333
>> s.save
=> false
```

```
>> s.errors
=> #<ActiveRecord::Errors:  ...
>> s.errors.full_messages
=> ["Name can't be blank"]
>>
```

We can inspect the errors



different ways to save

did the save succeed?

```
>> s.save
=> false
```

I'm confident it will work

```
>> s.save!
ActiveRecord::RecordInvalid: Validation
failed: Name can't be blank
        from /Library/Ruby/...
```

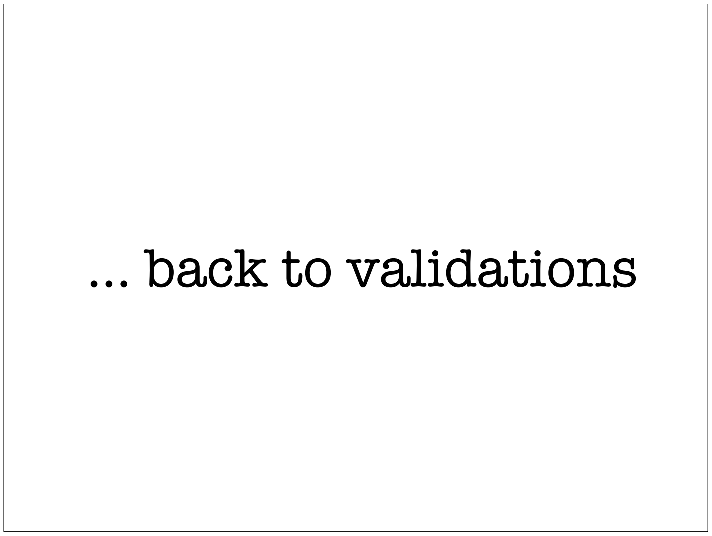
same with create

did the create succeed?

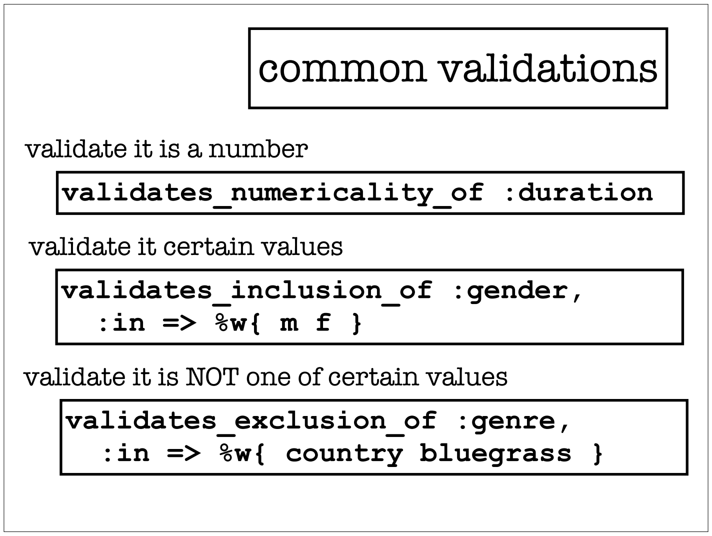
```
>> s = Song.create :duration => 333
=> #<Song id: nil, name: nil,
duration: 333>
>> s.save
=> false
```

I'm confident it will work

```
>> Song.create! :duration => 333
ActiveRecord::RecordInvalid: Validation
failed: Name can't be blank
     from /Library/Ruby/Gems/1.8/...
```



… back to validations



common validations

validate it is a number

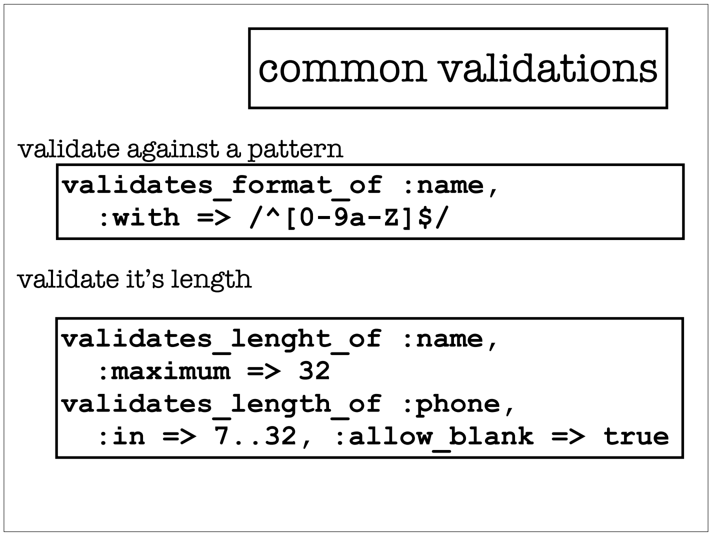
```
validates_numericality_of :duration
```

validate it certain values

```
validates_inclusion_of :gender,
  :in => %w{ m f }
```

validate it is NOT one of certain values

```
validates_exclusion_of :genre,
  :in => %w{ country bluegrass }
```



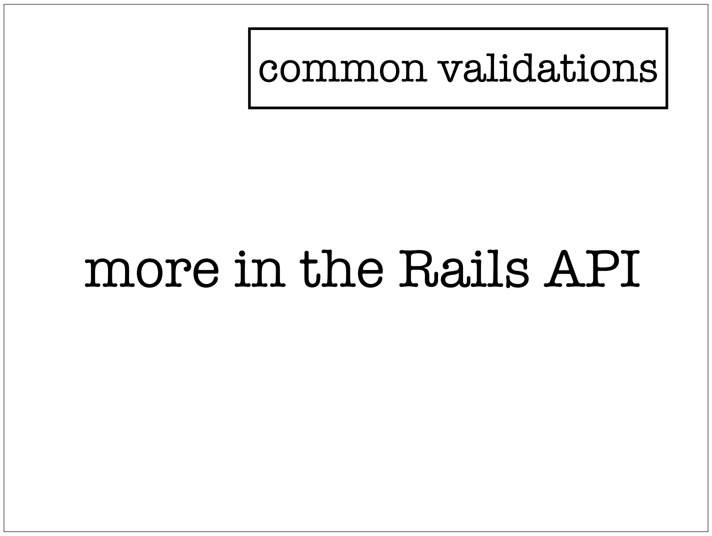
common validations

validate against a pattern

```
validates_format_of :name,
  :with => /^[0-9a-Z]$/
```

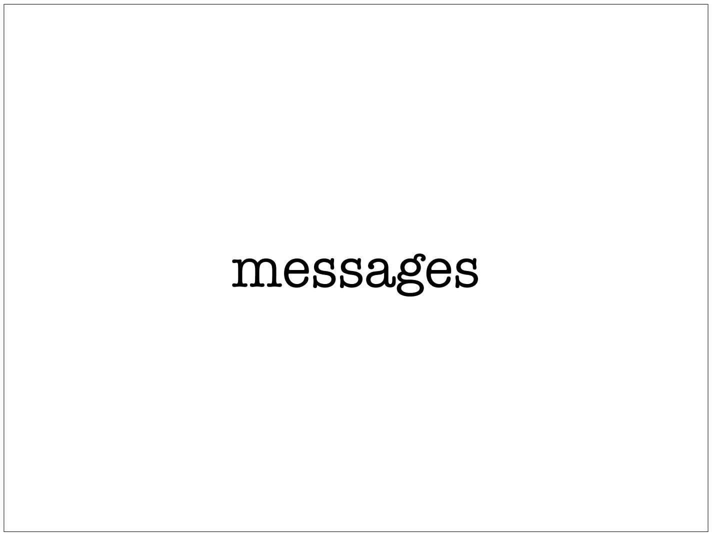
validate it’s length

```
validates_lenght_of :name,
  :maximum => 32
validates_length_of :phone,
  :in => 7..32, :allow_blank => true
```

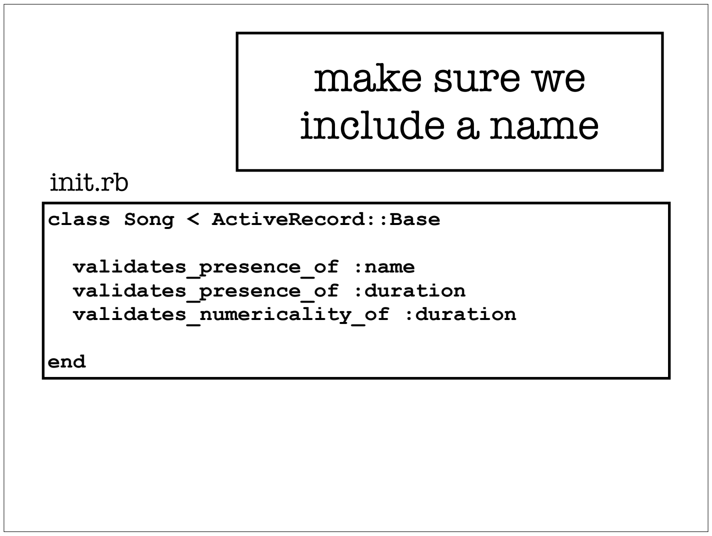


common validations

more in the Rails API



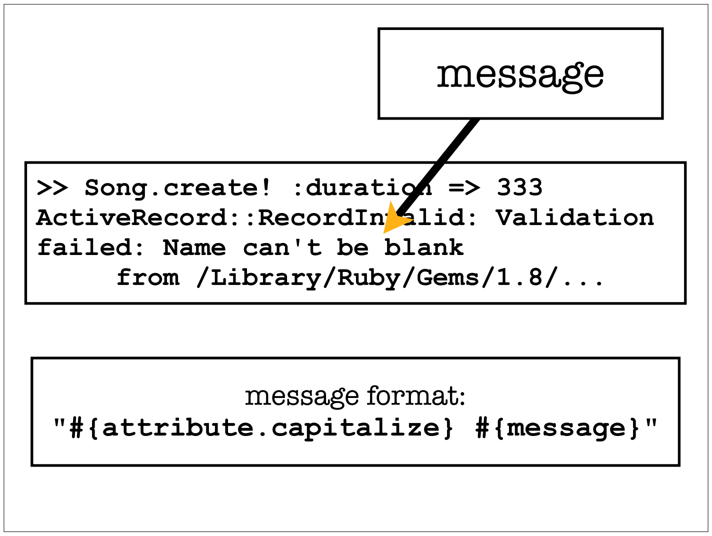
messages



make sure we include a name

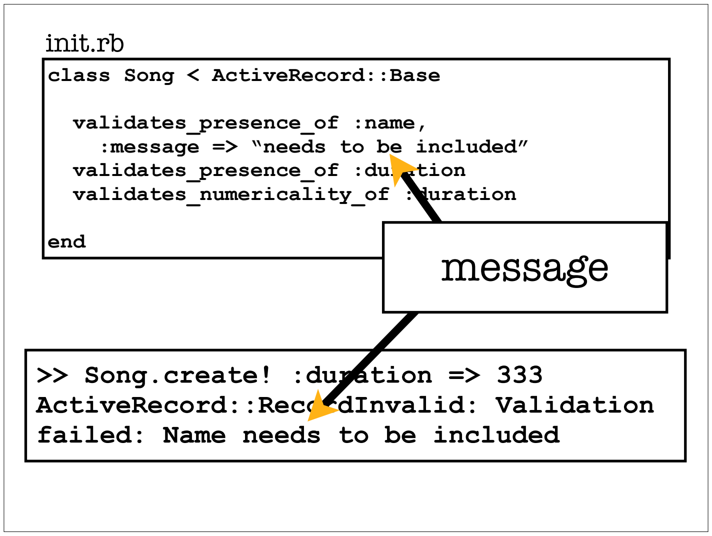
init.rb

```
class Song < ActiveRecord::Base

  validates_presence_of :name
  validates_presence_of :duration
  validates_numericality_of :duration

end
```



```
>> Song.create! :duration => 333
ActiveRecord::RecordInvalid: Validation
failed: Name can't be blank
     from /Library/Ruby/Gems/1.8/...
```

message format:
`"#{attribute.capitalize} #{message}"`

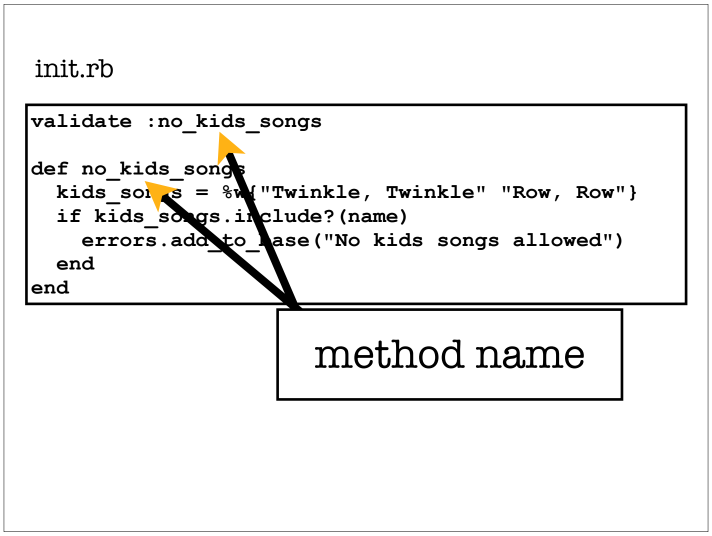


init.rb

```
class Song < ActiveRecord::Base

  validates_presence_of :name,
    :message => “needs to be included”
  validates_presence_of :duration
  validates_numericality_of :duration

end
```

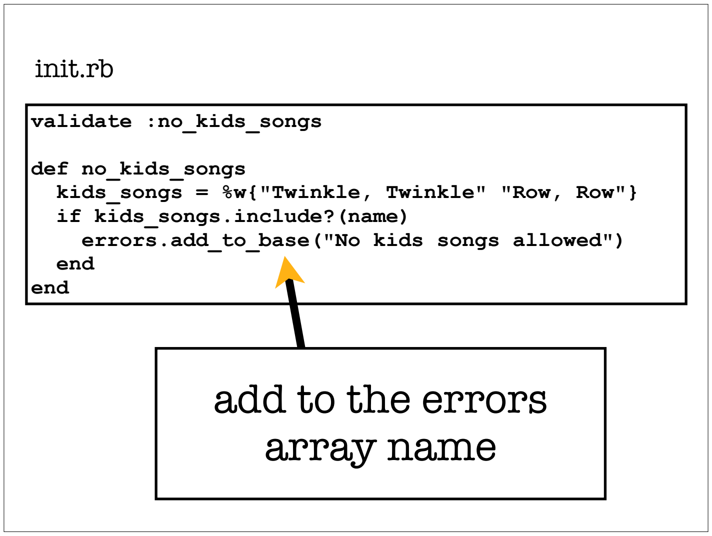
```
>> Song.create! :duration => 333
ActiveRecord::RecordInvalid: Validation
failed: Name needs to be included
```

rolling your own



init.rb

```
validate :no_kids_songs

def no_kids_songs
  kids_songs = %w{"Twinkle, Twinkle" "Row, Row"}
  if kids_songs.include?(name)
    errors.add_to_base("No kids songs allowed")
  end
end
```



init.rb

```
validate :no_kids_songs

def no_kids_songs
  kids_songs = %w{"Twinkle, Twinkle" "Row, Row"}
  if kids_songs.include?(name)
    errors.add_to_base("No kids songs allowed")
  end
end
```

LAB 1

Conference Submission Model

Conference Submission

- You are organizing a major conference and you decide to write some software that will help the selection committee select the best presentations from those submitted.
- Each speaker will be allowed to submit a talk

Conference Submission

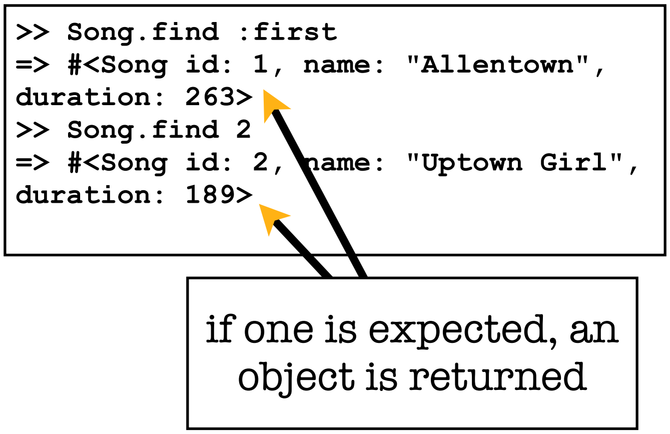
- Create a Presentation object. It should have:
  - The title and presenter's name
  - A large description (abstract)
  - email address for the speaker

Conference Submission

- use **`init.rb`** as a guide
- name it **`talks.rb`**
- add validations in there (see api.rubyonrails.org for more validations)
- create at least 10 objects at the bottom of your file

finding things

`find` is used to retrieve items from the database



```
>> Song.find :first
=> #<Song id: 1, name: "Allentown",
duration: 263>
>> Song.find 2
=> #<Song id: 2, name: "Uptown Girl",
duration: 189>
```

if one is expected, an object is returned

```
>> Song.find :all
=> [#<Song id: 1, name: "Allentown",
duration: 263>, #<Song id: 2, name:
"Uptown Girl", duration: 189>, #<Song
id: 3, name: "An Innocent Man",
duration: 369>, #<Song id: 4, name:
"Angry Young Man", ...
```

if **more** than one is expected, an Array is returned

```
>> Song.find :all, :conditions =>
{ :name => "Allentown"}
=> [#<Song id: 1, name: "Allentown",
duration: 263>]
```

even if only returning one object

common finds

find all of them

```
Song.find :all
```

find the first one

```
Song.find :first
```

find by a particular id

```
Song.find 4
Song.find 8
```

:conditions

three types

with a string

```
Song.find :all,
  :conditions => "name = #{name}"
```

with an array

```
Song.find :all,
  :conditions => ['name = ?', name]
```

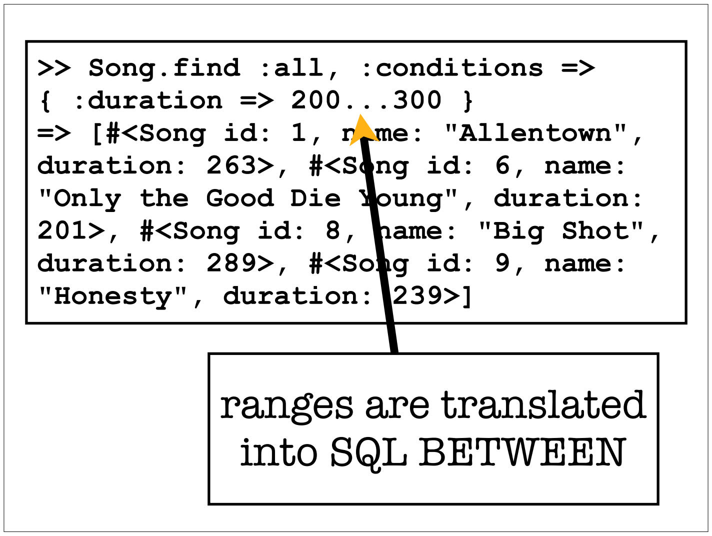
with a hash

```
Song.find :all,
  :conditions => { :name => name }
```

with a string

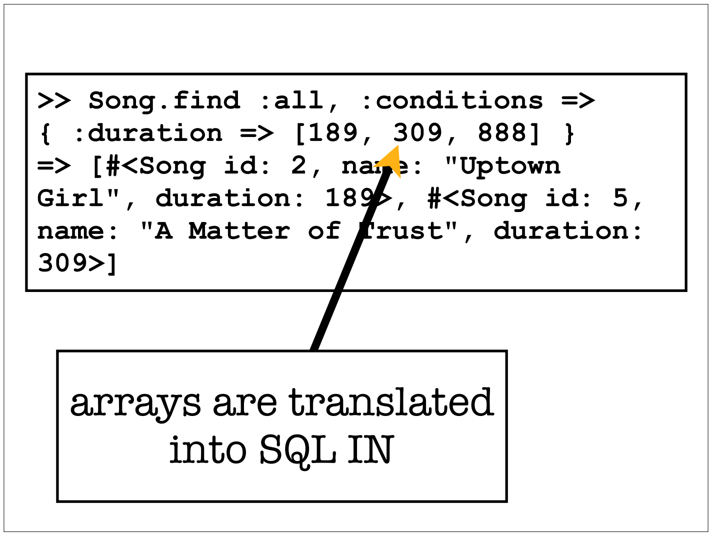
```
Song.find :all,
  :conditions => "name = #{name}"
```

why is this not a good idea?



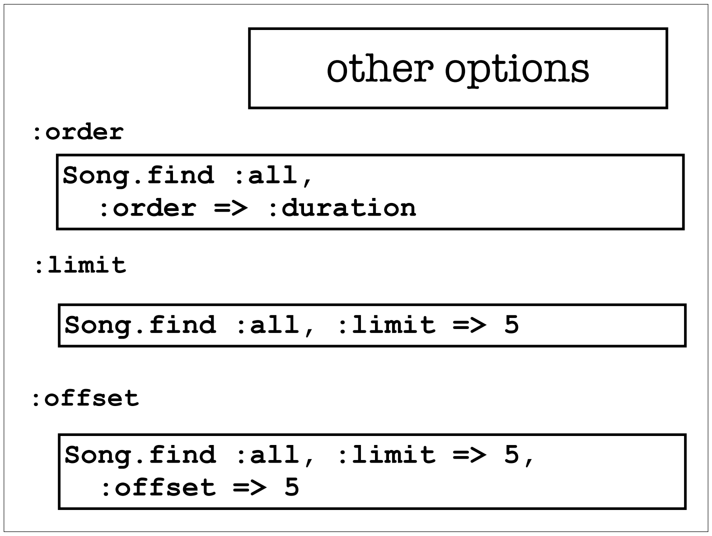
```
>> Song.find :all, :conditions =>
{ :duration => 200...300 }
=> [#<Song id: 1, name: "Allentown",
duration: 263>, #<Song id: 6, name:
"Only the Good Die Young", duration:
201>, #<Song id: 8, name: "Big Shot",
duration: 289>, #<Song id: 9, name:
"Honesty", duration: 239>]
```

ranges are translated into SQL BETWEEN



```
>> Song.find :all, :conditions =>
{ :duration => [189, 309, 888] }
=> [#<Song id: 2, name: "Uptown
Girl", duration: 189>, #<Song id: 5,
name: "A Matter of Trust", duration:
309>]
```

arrays are translated into SQL IN



other options

:order

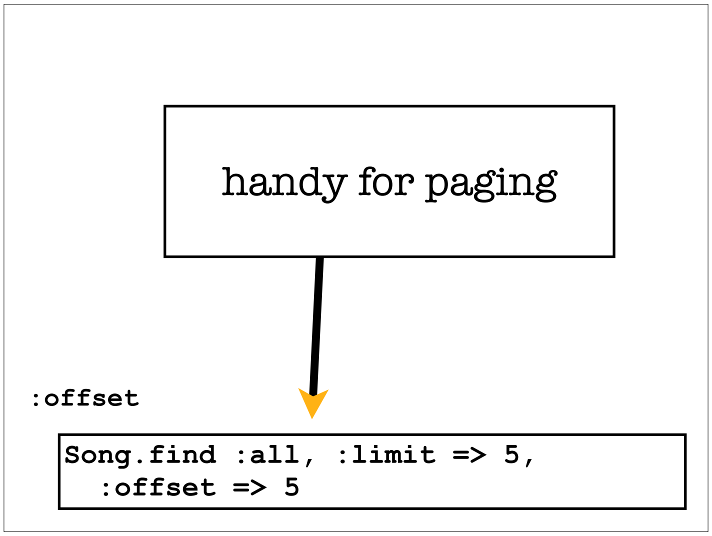
```
Song.find :all,
  :order => :duration
```

:limit

```
Song.find :all, :limit => 5
```

:offset

```
Song.find :all, :limit => 5,
  :offset => 5
```

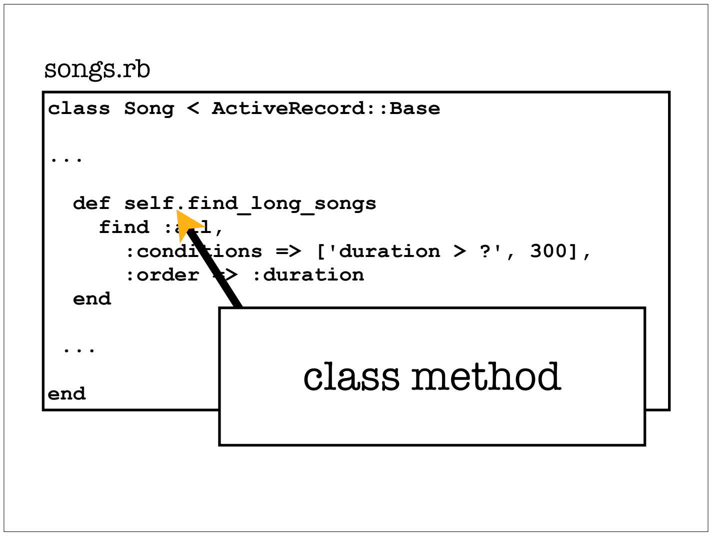


handy for paging

:offset

```
Song.find :all, :limit => 5,
  :offset => 5
```

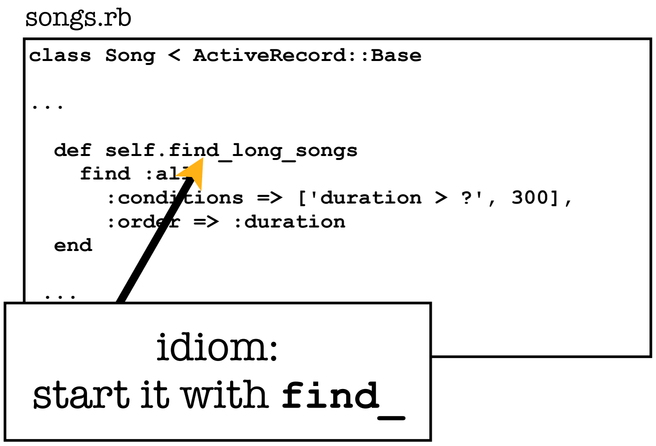
rolling our own finders



songs.rb

```
class Song < ActiveRecord::Base

...

  def self.find_long_songs
    find :all,
      :conditions => ['duration > ?', 300],
      :order => :duration
  end

 ...

end
```

class method



songs.rb

```
class Song < ActiveRecord::Base

...

  def self.find_long_songs
    find :all,
      :conditions => ['duration > ?', 300],
      :order => :duration
  end

 ...
```

idiom:
start it with **`find_`**

refactor

songs.rb

```
class Song < ActiveRecord::Base

  LONG_SONG = 300

  ...

  def self.find_long_songs
    find :all,
      :conditions => ['duration > ?', LONG_SONG],
      :order => :duration
  end

end
```

keep business knowledge in the business objects

```
>> Song.find_long_songs
=> [#<Song id: 3, name: "An Innocent
Man", duration: 369>, #<Song id: 4,
name: "Angry Young Man", duration:
323>, #<Song id: 5, name: "A Matter
of Trust", duration: 309>, #<Song id:
7, name: "Baby Grand", duration:
369>, #<Song id: 10, name: "Goodnight
Saigon", duration: 440>]
```

LAB 2

Custom Finders

Conference Submission

- play with the finders on your demo data
- add at least one custom finder
- add a finder that returns the submissions by a speaker whose name starts with a given letter (if I ask for all with O, it will return all speakers whose last name begins with an O).

Rails

name of your application.

```
$ cd vital_rails
$ cd src
$ rails rtunes
      create
      create  app/controllers
      create  app/helpers
      create  app/models
      create  app/views/layouts
      create  config/environments
      create  config/initializers
      create  config/locales
      create  db
      create  doc ...
```

open this file in an editor

```
...
      create  config/database.yml
...
```

config/database.yml

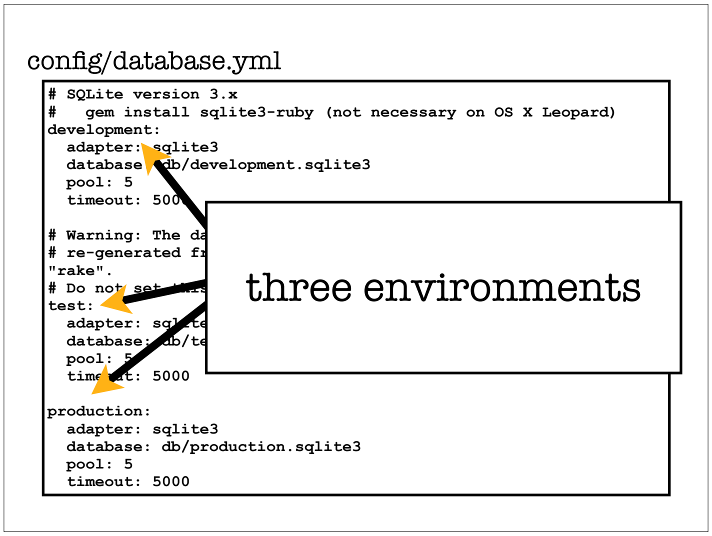
```
# SQLite version 3.x
#   gem install sqlite3-ruby (not necessary on OS X Leopard)
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

# Warning: The da
# re-generated fr
"rake".
# Do not set this
test:
  adapter: sqlite
  database: db/te
  pool: 5
  timeout: 5000

production:
  adapter: sqlite3
  database: db/production.sqlite3
  pool: 5
  timeout: 5000
```

if you are using sqlite, no changes are needed

config/database.yml

```
# SQLite version 3.x
#   gem install sqlite3-ruby (not necessary on OS X Leopard)
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

# Warning:
# re-gener
"rake".
# Do not s
test:
  adapter:
  database
  pool: 5
  timeout:

production
  adapter:
  database: db/production.sqlite3
  pool: 5
  timeout: 5000
```

mysql users, change the adapter, database, and add username and password

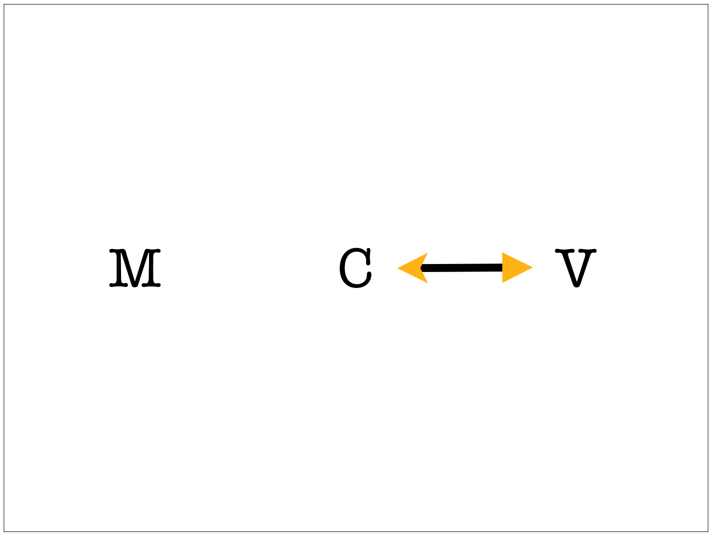


config/database.yml

```
# SQLite version 3.x
#   gem install sqlite3-ruby (not necessary on OS X Leopard)
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

# Warning: The da
# re-generated fr
"rake".
# Do not set this
test:
  adapter: sqlite
  database: db/te
  pool: 5
  timeout: 5000

production:
  adapter: sqlite3
  database: db/production.sqlite3
  pool: 5
  timeout: 5000
```

three environments

M V C



M odel

Represents the business model the "things" in your application

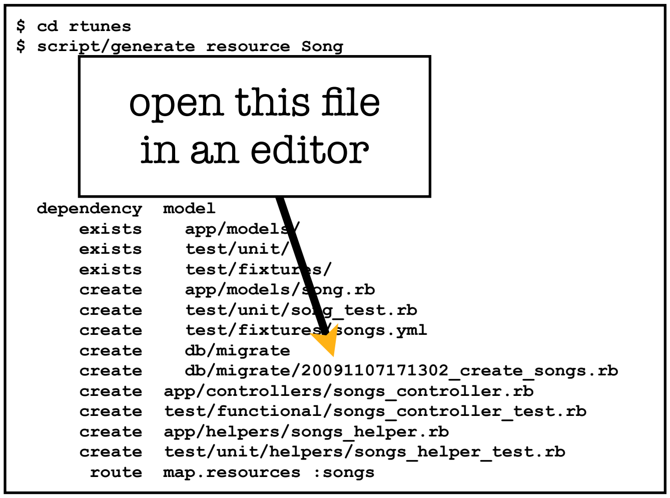
C ontroller

stage director in your application. What goes where

V iew

How things are represented in the application lets get our first model created

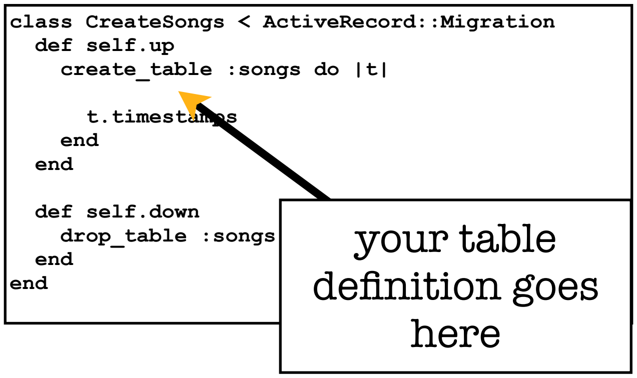
```
$ cd rtunes
$ script/generate resource Song
      exists  app/models/
      exists  app/controllers/
      exists  app/helpers/
      create  app/views/songs
      exists  test/functional/
      exists  test/unit/
      create  test/unit/helpers/
  dependency  model
      exists    app/models/
      exists    test/unit/
      exists    test/fixtures/
      create    app/models/song.rb
      create    test/unit/song_test.rb
      create    test/fixtures/songs.yml
      create    db/migrate
      create    db/migrate/20091107171302_create_songs.rb
      create  app/controllers/songs_controller.rb
      create  test/functional/songs_controller_test.rb
      create  app/helpers/songs_helper.rb
      create  test/unit/helpers/songs_helper_test.rb
       route  map.resources :songs
```



```
$ cd rtunes
$ script/generate resource Song
  dependency  model
      exists    app/models/
      exists    test/unit/
      exists    test/fixtures/
      create    app/models/song.rb
      create    test/unit/song_test.rb
      create    test/fixtures/songs.yml
      create    db/migrate
      create    db/migrate/20091107171302_create_songs.rb
      create  app/controllers/songs_controller.rb
      create  test/functional/songs_controller_test.rb
      create  app/helpers/songs_helper.rb
      create  test/unit/helpers/songs_helper_test.rb
       route  map.resources :songs
```

open this file in an editor

db/migrate/[timestamp]_create_songs.rb

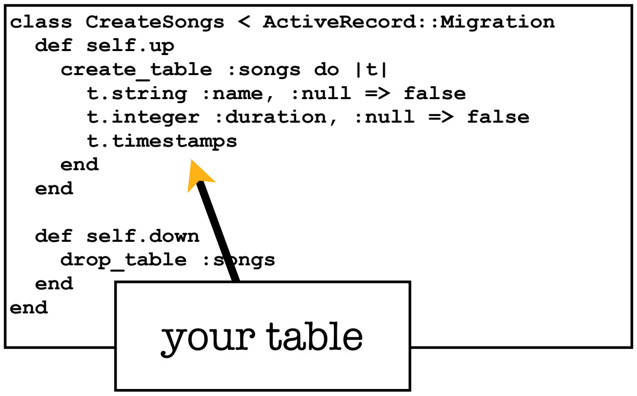


```
class CreateSongs < ActiveRecord::Migration
  def self.up
    create_table :songs do |t|

      t.timestamps
    end
  end

  def self.down
    drop_table :songs
  end
end
```

your table definition goes here

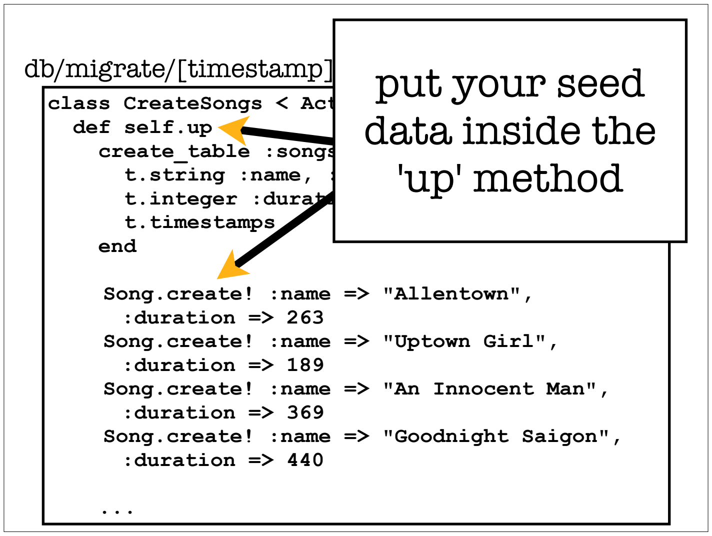
migrations allow you to incrementally migrate your database

db/migrate/[timestamp]_create_songs.rb



```
class CreateSongs < ActiveRecord::Migration
  def self.up
    create_table :songs do |t|
      t.string :name, :null => false
      t.integer :duration, :null => false
      t.timestamps
    end
  end

  def self.down
    drop_table :songs
  end
end
```

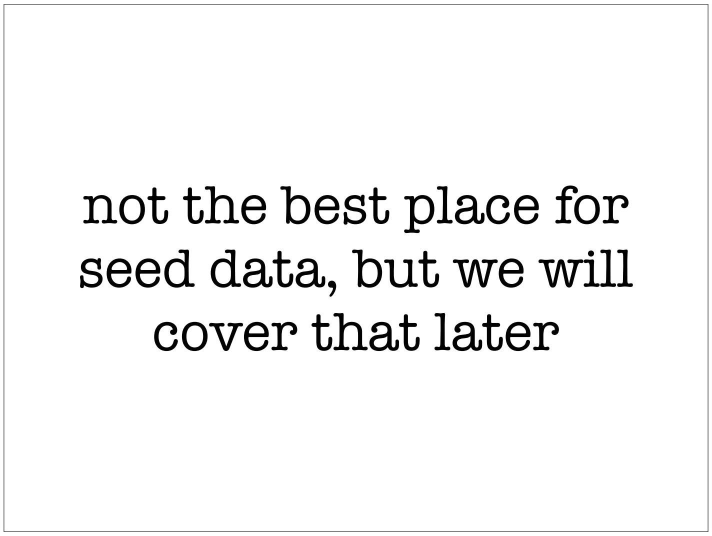
your table



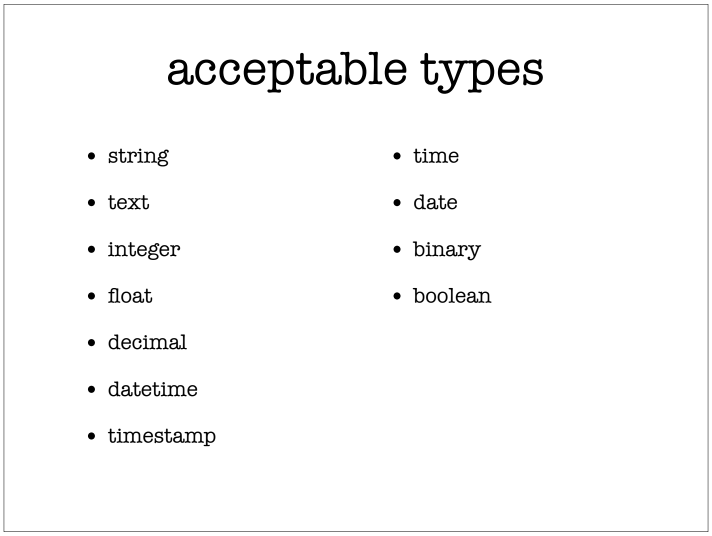
db/migrate/[timestamp]

```
class CreateSongs < Act
  def self.up
    create_table :songs
      t.string :name, :
      t.integer :durat
      t.timestamps
    end

     Song.create! :name => "Allentown",
       :duration => 263
     Song.create! :name => "Uptown Girl",
       :duration => 189
     Song.create! :name => "An Innocent Man",
       :duration => 369
     Song.create! :name => "Goodnight Saigon",
       :duration => 440

     ...
```

put your seed data inside the 'up' method

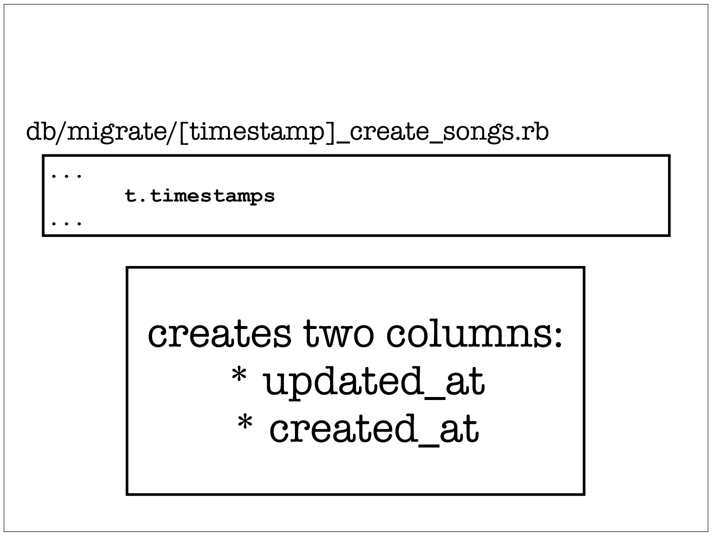


not the best place for seed data, but we will cover that later



acceptable types

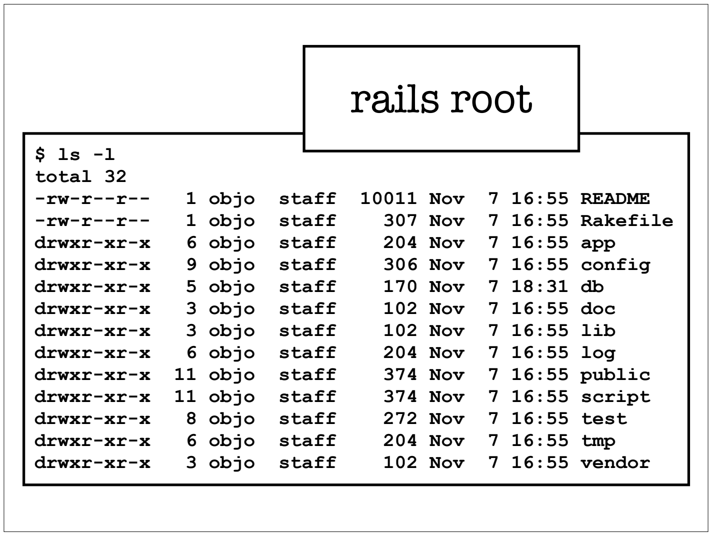
- string
- text
- integer
- float
- decimal
- datetime
- timestamp
- time
- date
- binary
- boolean



db/migrate/[timestamp]_create_songs.rb

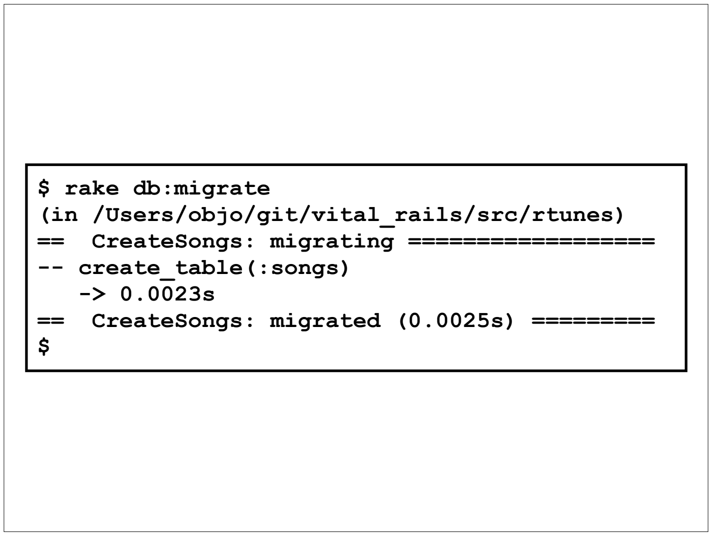
```
...
      t.timestamps
...
```

creates two columns:
* updated_at
* created_at

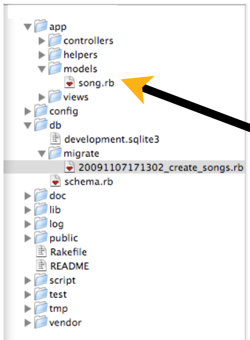


rails root

```
$ ls -l
total 32
-rw-r--r--   1 objo  staff  10011 Nov  7 16:55 README
-rw-r--r--   1 objo  staff    307 Nov  7 16:55 Rakefile
drwxr-xr-x   6 objo  staff    204 Nov  7 16:55 app
drwxr-xr-x   9 objo  staff    306 Nov  7 16:55 config
drwxr-xr-x   5 objo  staff    170 Nov  7 18:31 db
drwxr-xr-x   3 objo  staff    102 Nov  7 16:55 doc
drwxr-xr-x   3 objo  staff    102 Nov  7 16:55 lib
drwxr-xr-x   6 objo  staff    204 Nov  7 16:55 log
drwxr-xr-x  11 objo  staff    374 Nov  7 16:55 public
drwxr-xr-x  11 objo  staff    374 Nov  7 16:55 script
drwxr-xr-x   8 objo  staff    272 Nov  7 16:55 test
drwxr-xr-x   6 objo  staff    204 Nov  7 16:55 tmp
drwxr-xr-x   3 objo  staff    102 Nov  7 16:55 vendor
```



```
$ rake db:migrate
(in /Users/objo/git/vital_rails/src/rtunes)
==  CreateSongs: migrating ==================
-- create_table(:songs)
   -> 0.0023s
==  CreateSongs: migrated (0.0025s) =========
$
```



```
app
  controllers
  helpers
  models
    song.rb
  views
config
db
  development.sqlite3
  migrate
    20091107171302_create_songs.rb
  schema.rb
doc
lib
log
public
Rakefile
README
script
test
tmp
vendor
```

open the model file

app/models/song.rb

```
class Song < ActiveRecord::Base
end
```

copy the class you defined earlier and paste it in here

app/models/song.rb

```
class Song < ActiveRecord::Base

  LONG_SONG = 300

  validates_presence_of :name,
    :message => "needs to be included"
  validates_presence_of :duration
  validates_numericality_of :duration
  validate :no_kids_songs

  def self.find_long_songs
    find :all,
      :conditions => ['duration > ?', LONG_SONG],
      :order => :duration
  end

  ...
end
```

and now we can play

```
$ script/console
Loading development environment (Rails 2.3.2)
>> Song.count
=> 10
>> Song.create! :name => "Allentown",
:duration => 263
=> #<Song id: 1, name: "Allentown", duration:
263, created_at: "2009-11-07 18:54:46",
updated_at: "2009-11-07 18:54:46">
```

LAB 4

Initial Rails Application

Conference Submission

- create the submit_it application
- create a Presentation resource
- copy your code from `lab1` into your app
- include the sample data
- make sure it works