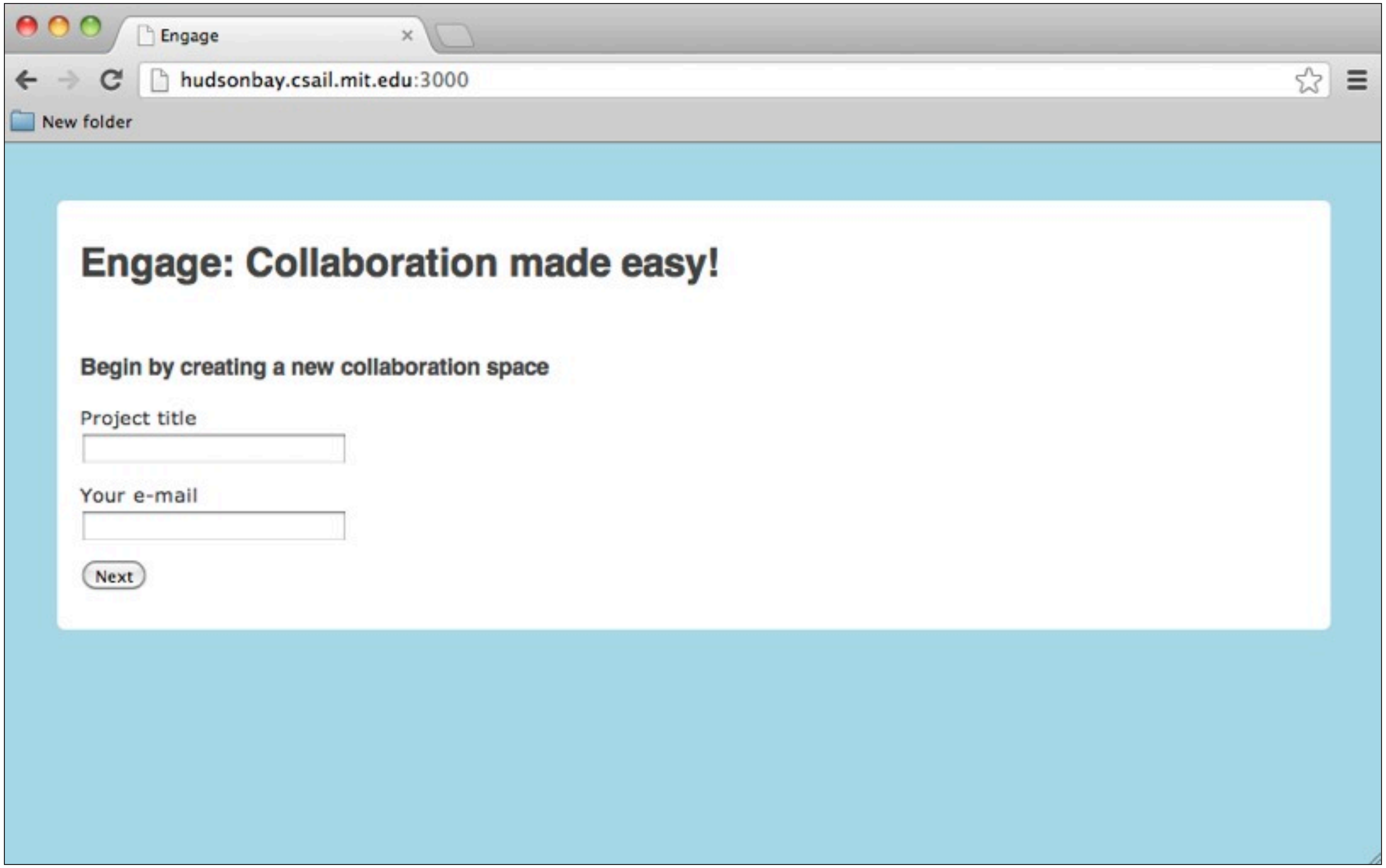
software studio

Example Rails Application

Eunsuk Kang

Demo: An app for collaboration



Engage: Collaboration made easy!

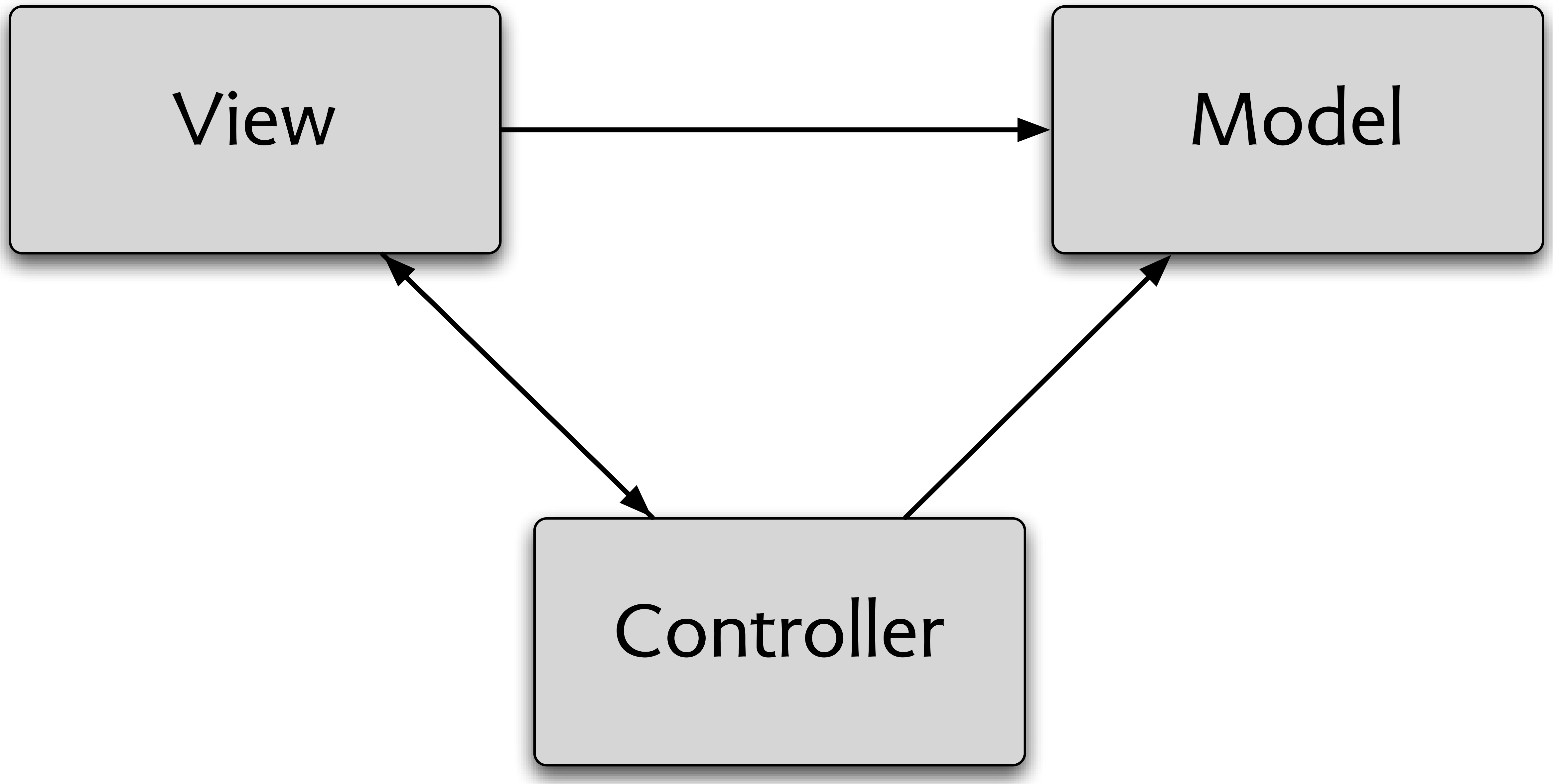
Begin by creating a new collaboration space

Project title

Your e-mail

Next

MVC Design



Exercise: What's in a Model?

What are resources?

What are their attributes? Constraints?

What about relationships between them?

Resources

What are resources?

Resources

What are resources?

› Projects, comments

Attributes

What are resources?

- Projects, comments

What are their attributes? Constraints?

Attributes

What are resources?

- Projects, comments

What are their attributes? Constraints?

- Each project has a title & owner's e-mail
- Each project is accessible with a secret key
- Each comment has a commenter and a body

Relationships

What are resources?

› Projects, comments

What are their attributes? Constraints?

› Each project has a title & owner's e-mail
› Each project is accessible with a secret key
› Each comment has a commenter and a body

What about relationships between them?

Relationships

What are resources?

› Projects, comments

What are their attributes? Constraints?

› Each project has a title & owner's e-mail

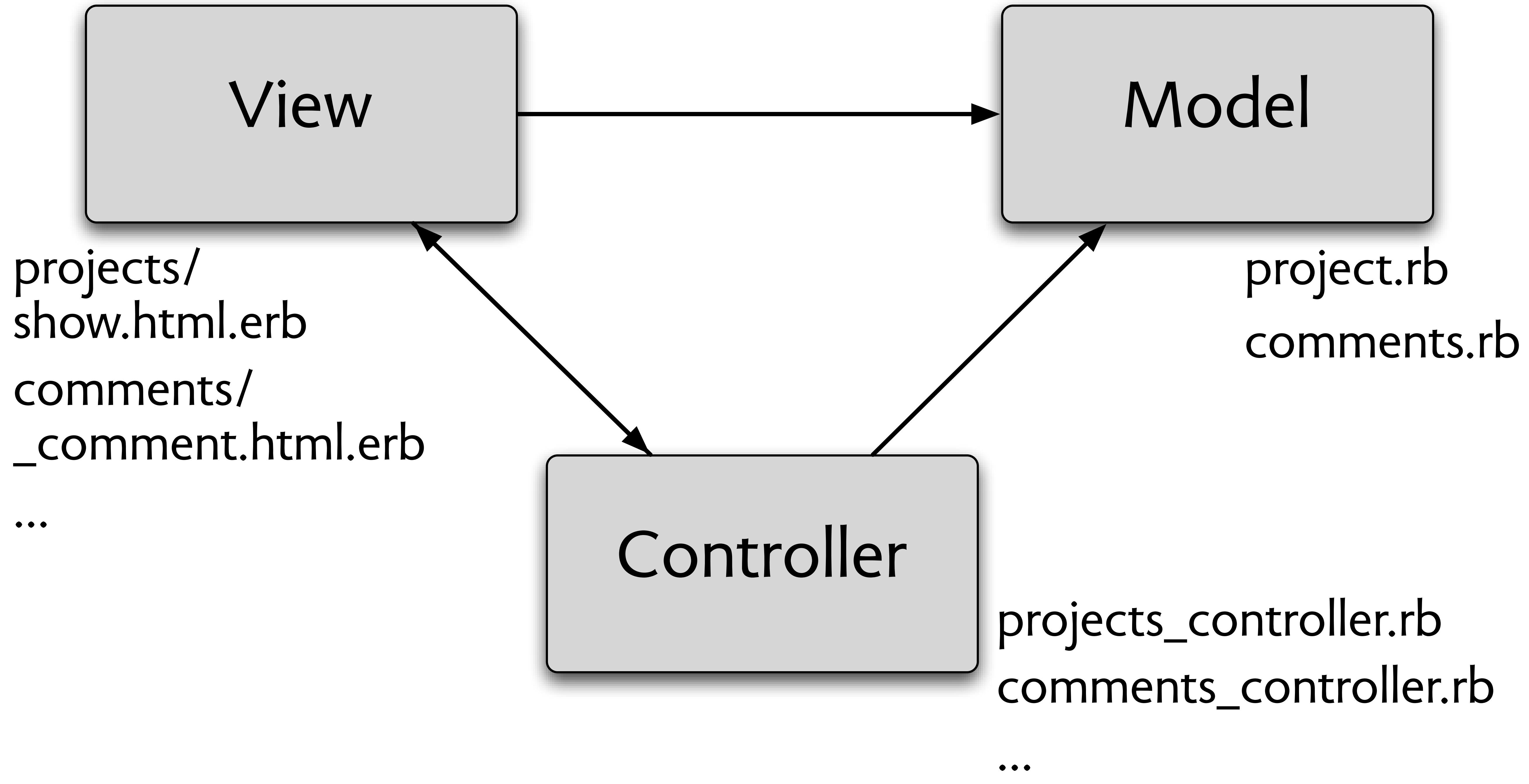
› Each project is accessible with a secret key

› Each comment has a commenter and a body

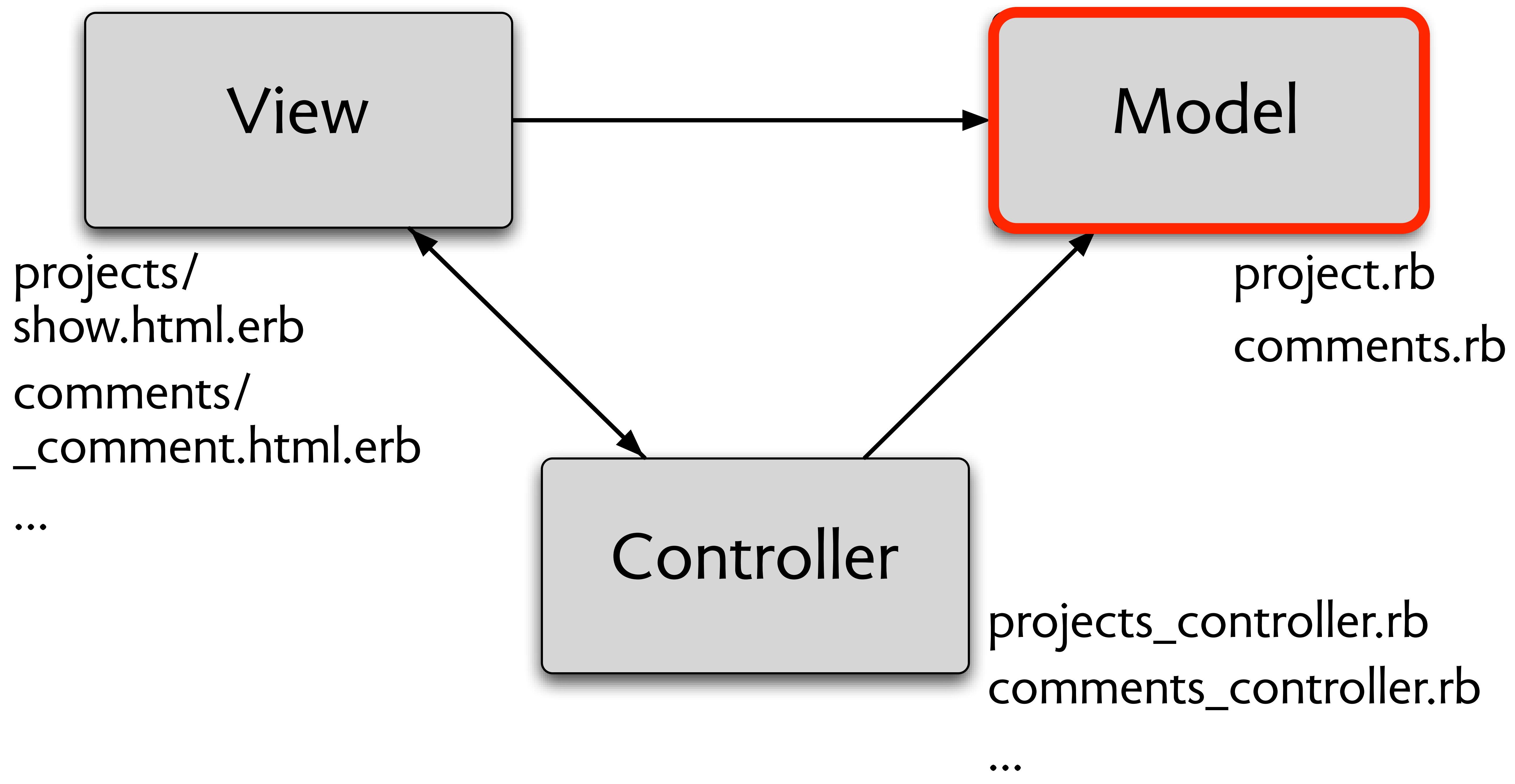
What about relationships between them?

› A project has many comments

Rails Convention



Model


View
Model
Controller
projects/
show.html.erb
comments/
_comment.html.erb
...
project.rb
comments.rb
projects_controller.rb
comments_controller.rb
...

ActiveRecords

app/models/project.rb

```ruby
class Project < ActiveRecord::Base
  KEY_LENGTH = 10
  VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i

  attr_accessible :title, :owner_email
  before_create :generate_access_key

  validates :title, :presence => true
  validates :owner_email, :presence => true,
            :format => { :with => VALID_EMAIL_REGEX }

  has_many :comments

  def generate_access_key
    self.access_key = SecureRandom.hex(KEY_LENGTH)
  end
end
```

Validation

Enforce constraints over your data!

```
class Project < ActiveRecord::Base
  KEY_LENGTH = 10
  VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i

  attr_accessible :title, :owner_email
  before_create :generate_access_key

  validates :title, :presence => true
  validates :owner_email, :presence => true,
            :format => { :with => VALID_EMAIL_REGEX }

  has_many :comments

  def generate_access_key
    self.access_key = SecureRandom.hex(KEY_LENGTH)
  end
end
```

Callbacks

Methods called at various points

```
class Project < ActiveRecord::Base
  KEY_LENGTH = 10
  VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i

  attr_accessible :title, :owner_email
  before_create :generate_access_key

  validates :title, :presence => true
  validates :owner_email, :presence => true,
            :format => { :with => VALID_EMAIL_REGEX }

  has_many :comments


  def generate_access_key
    self.access_key = SecureRandom.hex(KEY_LENGTH)
  end
end
```

Associations

Declare relationships between objects

```
class Project < ActiveRecord::Base
  ...
  has_many :comments
  ...
end
```

```
class Comment < ActiveRecord::Base
  belongs_to :project
  ...
end
```

To create a new comment for a project:

```
@comment = @project.comments.create(
              :commenter => ‘Alex’,
              :body => ‘Hello World!’)
```

DB Schema

db/schema.rb

```
ActiveRecord::Schema.define(:version => 20130203203925) do

  create_table "projects", :force => true do |t|
    t.string   "title"
    t.string   "owner_email"
    t.string   "access_key"
  end

  create_table "comments", :force => true do |t|
    t.string   "commenter"
    t.text     "body"
    t.string   "project_id"
  end

end
```

Migration

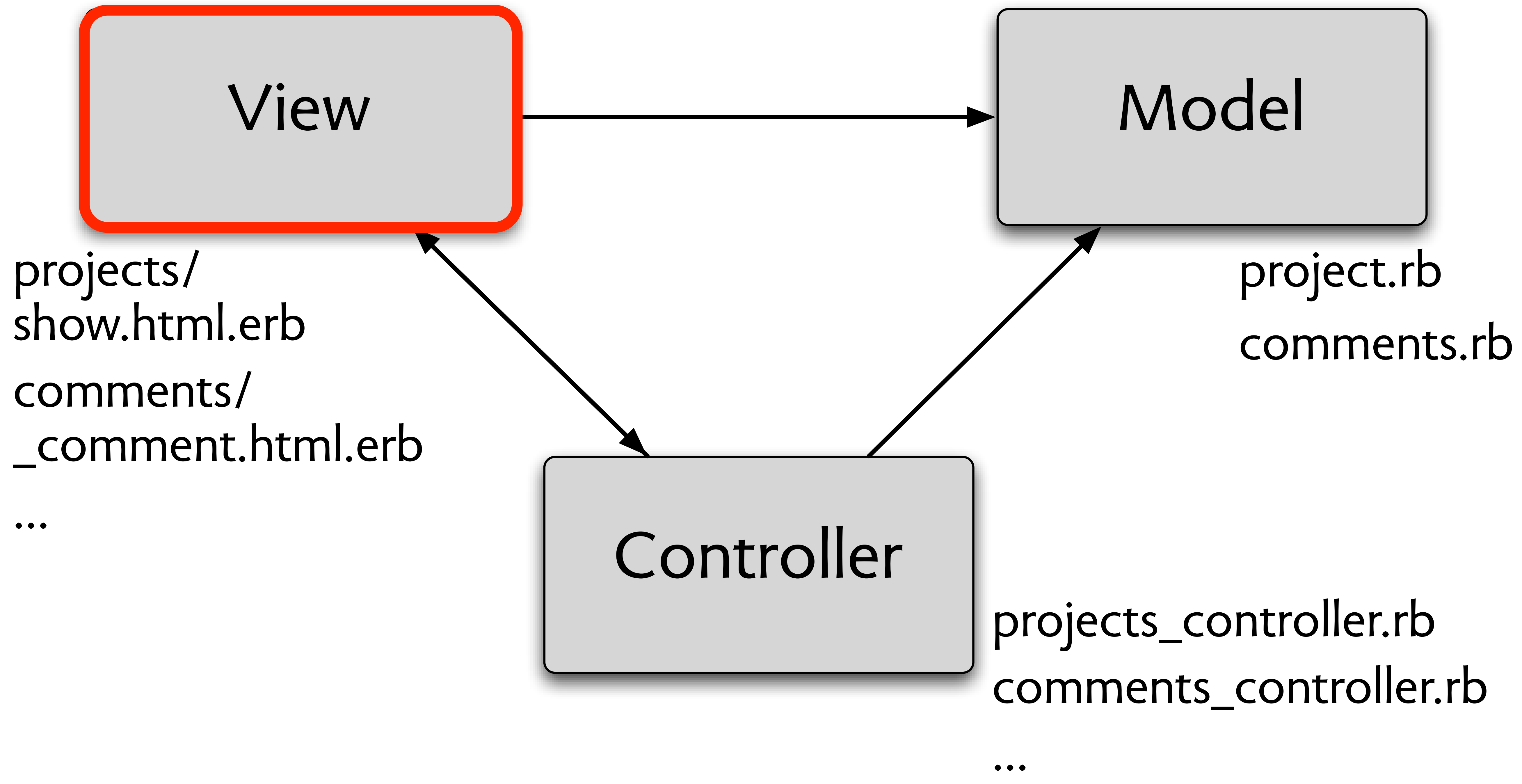
Create DB table from schema definition

```
ActiveRecord::Schema.define(:version => 20130203203925) do

  create_table "projects", :force => true do |t|
    t.string   "title"
    t.string   "owner_email"
    t.string   "access_key"
  end
  ...
end
```

id	title	owner_email	access_key
1	“6.170 Homework #1”	instructor@mit.edu	e5056b6f653214f4
2	“House utility bills”	joe@mit.edu	24fa027bdb6794e

View


View
Model
Controller
projects/
show.html.erb
comments/
_comment.html.erb
...
project.rb
comments.rb
projects_controller.rb
comments_controller.rb
...

View Templates

app/views/projects/show.html.erb

```
<table>
<tr>
<td>
<div id="comments">
  <div class="comment">
    <i>Welcome to <%= @project.title %>!</i><br>
    <i>Your comment goes here...</i><br><br>
  </div>
  <%= render @project.comments %>
</div>
</td>
</tr>
</table>
```

Partials

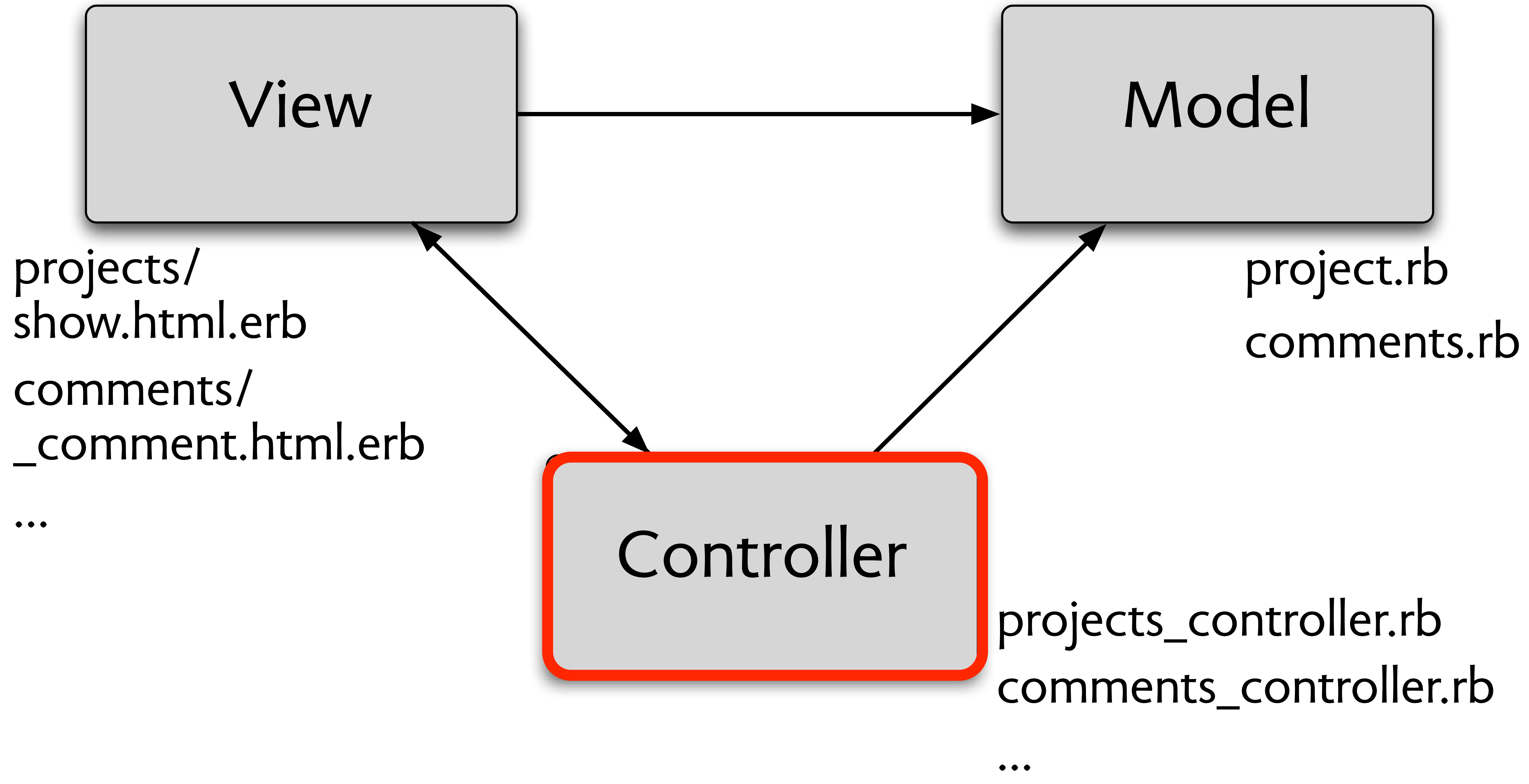
Factor out & reuse a partial view (‘Don’t Repeat Yourself’)

```
<table>
<tr>
<td>
<div id="comments">
  ...
  <%= render @project.comments %>
</div>
</td>
</tr>
</table>
```

app/views/comments/_comment.html.erb

```
<div class="comment">
  <b><%= comment.commenter %></b>: <%= comment.body %>
</div>
```

Controller


View
Model
Controller
projects/
show.html.erb
comments/
_comment.html.erb
...
project.rb
comments.rb
projects_controller.rb
comments_controller.rb
...

Routes

config/routes.rb

```
Engage::Application.routes.draw do
  ...
  match 'projects/:id' => 'projects#show', :via => :get
  ...
end
```

Example

› Request   `GET /projects/bce7cca9ee32`

› Controller Method   `ProjectsController.show`

(with id = bce7cca9ee32)

Controllers

app/controllers/projects_controller.rb

```
class ProjectsControlle  < ApplicationController

  def show
    # Look up project
    @project   Project.find(params[:id])

    if no  @project.blank?
      @username = cookies[:current_username]

      respond_to do |format|
        format.html # show.html.erb
        format.json { render :json =  @project }
      end
    else
      render_404
    end
  end
  ...
end
```

DB Access

No SQL queries; use Rails helpers

```
class ProjectsControlle  < ApplicationController

  def show
    # Look up project
    @project   Project.find(params[:id])

    i  not @project.blank?
      @username = cookies[:current_username]

      respond_to do |format|
        format.html # show.html.erb
        format.json { render :json => @project }
      end
    else
      render_404
    end
  end
  ...
end
```

DB Access

Look up by access_key, not ID!

```
class ProjectsControlle  < ApplicationController

  def show
    # Look up project by access_key
    @project   Project.find_by_access_key(params[:id])

    i  not @project.blank?
      @username = cookies[:current_username]

      respond_to do |format|
        format.html # show.html.erb
        format.json { render :json => @project }
      end
    else
      render_404
    end
  end
  ...
end
```

Response

Multiple response formats

```
class ProjectsControlle  < ApplicationController

  def show
    # Look up project by access_key
    @project   Project.find_by_access_key(params[:id])

    if no  @project.blank?
      @username = cookies[:current_username]

      respond_to do |format|
        format.html # show.html.erb
        format.json { render :json =  @project }
      end
    else
      render_404
    end
  end
  ...
end
```

Default Routes

Rails generates CRUD routes (think REST!)

```
Engage::Application.routes.draw do
  ...
  # match 'projects/:id' => 'projects#show', :via => :get
  resources :projects
  ...
end
```

console> `rake routes`

```
GET    /projects                                    projects#index
POST   /projects                                    projects#create
GET    /projects/new                                projects#new
GET    /projects/:id/edit                           projects#edit
GET    /projects/:id                                projects#show
PUT    /projects/:id                                projects#update
DELETE /projects/:id                                projects#destroy
```

Nested Routes

```
Engage::Application.routes.draw do
  ...
  resources :projects do
    resources :comments
  end
  ...
end
```

console> `rake routes`

```
...
GET    /projects/:project_id/comments              comments#index
POST   /projects/:project_id/comments              comments#create
GET    /projects/:project_id/comments/new          comments#new
GET    /projects/:project_id/comments/:id/edit comments#edit
...
```

Code on Github

New features added over term

- Collaborative editing, uploads, user accounts, etc.
- Released as separate branches