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

View  Controller  Model

Controller
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

View

projects/show.html.erb
comments/_comment.html.erb
...

Controller

projects_controller.rb
comments_controller.rb
...

Model

project.rb
comments.rb
Model

View

projects/show.html.erb
comments/_comment.html.erb
...

Controller

projects_controller.rb
comments_controller.rb
...

Model

project.rb
comments.rb

class Project < ActiveRecord::Base
  KEY_LENGTH = 10
  VALID_EMAIL_REGEX = /\A[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

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

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

To create a new comment for a project:

```ruby
@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

```ruby
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
```

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>owner_email</th>
<th>access_key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“6.170 Homework #1”</td>
<td><a href="mailto:instructor@mit.edu">instructor@mit.edu</a></td>
<td>e5056b6f653214f4</td>
</tr>
<tr>
<td>2</td>
<td>“House utility bills”</td>
<td><a href="mailto:joe@mit.edu">joe@mit.edu</a></td>
<td>24fa027bdb6794e</td>
</tr>
</tbody>
</table>
View

Controller

Model

projects/show.html.erb
comments/_comment.html.erb
...
View Templates

app/views/projects/show.html.erb

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

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

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

app/views/comments/_comment.html.erb

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

View

projects/show.html.erb
comments/_comment.html.erb
...

Controller

projects_controller.rb
comments_controller.rb

Model

project.rb
comments.rb

...
Routes

cfg/routes.rb

```ruby
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)
class ProjectsController < ApplicationController

  def show
    @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 ProjectsController < 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
Look up by access_key, not ID!

class ProjectsController < 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
...
Response

Multiple response formats

class ProjectsController < 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

cronos> rake routes

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/projects</td>
<td>projects#index</td>
</tr>
<tr>
<td>POST</td>
<td>/projects</td>
<td>projects#create</td>
</tr>
<tr>
<td>GET</td>
<td>/projects/new</td>
<td>projects#new</td>
</tr>
<tr>
<td>GET</td>
<td>/projects/:id/edit</td>
<td>projects#edit</td>
</tr>
<tr>
<td>GET</td>
<td>/projects/:id</td>
<td>projects#show</td>
</tr>
<tr>
<td>PUT</td>
<td>/projects/:id</td>
<td>projects#update</td>
</tr>
<tr>
<td>DELETE</td>
<td>/projects/:id</td>
<td>projects#destroy</td>
</tr>
</tbody>
</table>
Nested Routes

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

```bash
console> rake routes
```

```text
... 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