Rails Programming CHEATS

Create a New Rails App

The Pragmatic Studio
http://pragmaticstudio.com
Online Rails Programming Course

`rails new` *`app_name`*
creates a new Rails application with a given name **example:** rails new events
`rails new` *`app_name`* `--skip-test-unit (or -T)`
creates a new Rails application with a given name but does not use the Test::Unit framework **example:** rails new events --skip-test-unit rails new events -T
`rails server (or rails s)`
starts a web server on http://localhost:3000
`bundle install`
installs all required gems and their dependencies
`subl .`
opens the Sublime Text editor for the current working directory
`rails console (or rails c)`
starts a Rails console session

ERb Tags

<%=	%>	runs the Ruby code and **substitutes** result into template
<%	%>	runs the Ruby code but **does not** substitute result into template

```
<ul>
  <% @events.each do |event|  %>

    <li><%= event %></li>

  <% end %>
</ul>
```

→

```
<ul>

    <li>BugSmash</li>
    <li>Hackathon</li>

</ul>
```

Create a Controller Class

`rails generate (or g) controller` ***`name_of_controller`***

creates a controller with a given name

example: rails g controller events

Controller names are plural

`rails destroy controller` ***`name_of_controller`***

cleans up what got generated

example: rails destroy controller events

Create a Model and Migration

```
rails g model name_of_model field:type field:type...
```

creates a model with a given name, along with a migration for creating a database table with the specified fields (columns) and types

example: rails g model event name:string location:string price:decimal

Model names are singular

Migrations

The Pragmatic Studio
http://pragmaticstudio.com
Online Rails Programming Course

`rails g migration name_of_migration field:type field:type...`	creates a migration with a given name along with the specified fields and types **tip:** Name your migration using the format AddXXXToYYY, and Rails will assume you want to add the specified columns to the YYY table. **example:** rails g migration AddFieldsToEvents starts_at:datetime description:text
`rake db:migrate`	runs any pending migration files in the db/migrate directory
`rake db:migrate:status`	checks the status of all migrations
`rake db:rollback`	reverses the last applied migration
`rake db:migrate VERSION=XXXX`	rolls back to any previous migration version

Column Types

:binary
:boolean
:date
:datetime
:decimal
:float
:integer
:primary_key
:string
:text
:time
:timestamp

Common Migration Methods

`create_table(name, options={}) do |t|`
`  t.column_type :column_name, options...`
`end`

creates a new table and adds column definitions
`change_table(table_name, options = {})`
uses a block to change columns in the table
`drop_table(name)`
drops the table
`rename_table(table_name, new_name)`
renames a table
`add_column(table_name, column_name, type, options = {})`
adds a new column to the table
`remove_column(table_name, column_name)`
removes the column from the table

Common Migration Methods (cont.)

`change_column(table_name, column_name, type, options = {})`
changes the column's definition according to the options
`rename_column(table_name, column_name, new_column_name)`
renames a column in the table
`add_index(table_name, column_name, options = {})`
adds a new index to the table
`remove_index(table_name, options = {})`
removes the given index from the table
`execute(sql)`
executes arbitrary SQL statements

Rake

`rake -T`
see a list of all Rake tasks
`rake -T db`
see a list of only the database-specific Rake tasks
`rake db:test:prepare`
copy the schema from the development database to the test database
`rake routes`
see a list of all defined routes
`rake db:reset`
resets the database using the migrations (you lose all data) and loads seed data

Generate a Resource

```
rails g resource name_of_resource field:type field:type......
```

generates a resource with a given name along with fields and types

example: rails g resource registration name:string
email:string
how_heard:string
event:references

result: A migration for creating the registrations database table.
A Registration model with a belongs_to declaration.
An empty RegistrationsController.
A set of resource routes.

Define Resource Routes

`resources :`*`name_of_resource`*

dynamically defines all the routes for a resource

example: resources :events

result: 8 defined routes

Name	HTTP Verb	URL Pattern	controller#action
events	GET	/events	events#index
event	GET	/events/:id	events#show
edit_event	GET	/events/:id/edit	events#edit
	PATCH	/events/:id	events#update
	PUT	/events/:id	events#update
	POST	/events	events#create
new_event	GET	/events/new	events#new
	DELETE	/events/:id	events#destroy

REST vs. SQL Actions

Actions	show	create	update	destroy
SQL	select	create	update	delete
REST	get	post	patch	delete

Render a Partial

Step 1

Extract common view code into a partial template

partial file names are always prefixed with an underscore

example: _form.html.erb

example: app/views/layouts/_header.html.erb

Step 2

`<%= render “`*`name of partial without underscore`*`” %>`

render the partial from another view

example: <%= render ‘form’ %>

example: <%= render ‘layouts/header’ %>

Named Route Methods

routename_url
generates the full URL **example:** events_url generates http://www.example.com/events
routename_path
generates just the path part of the URL **example:** events_path generates /events

To try a named route helper method in the console, use the app object:

```
$ rails console

>> app.events_url
=> "http://www.example.com/events"
>> app.events_path
=> "/events"

>> e = Event.find(1)
>> app.event_url(e)
=> "http://www.example.com/events/1"
>> app.event_path(e)
=> "/events/1"
```

When you need to reload the environment in your console, use: **reload!**

Frequently Used Built-In View Helpers

`truncate(text, options = {})`
truncates the given text to a default of 30 characters
`pluralize(count, singular, plural = nil)`
pluralizes the singular word unless the count is 1
`number_to_currency(number, options = {})`
1234567890.50 => $1,234,567,890.50
`number_to_percentage(number, options = {})`
100 => 100.000%
`number_to_phone(number, options = {})`
3035551212 => 303-555-1212
`time_ago_in_words(from_time)`
Time.now + 50.minutes => “about 1 hour”
`content_tag(name, content, options = {})`
returns a safe HTML tag of type name surrounding the content

To try a built-in view helper in the console, use the helper object:

```
$ rails console

>> helper.number_to_currency(12.5)
=> "$12.50"

>> helper.pluralize(1, "person")
=> "1 person"

>> helper.pluralize(2, "person")
=> "2 people"

>> helper.time_ago_in_words(Time.now + 50.minutes)
=> "about 1 hour"
```

Common Validation Methods

`validates :name, presence: true`
validates that the value of the specified attribute is not blank
`validates :description, length: { minimum: 25 }`
validates that the value of the specified attribute matches the specified length restrictions
`validates :price, numericality: { greater_than_or_equal_to: 0 }`
validates that the value of the specified attribute is numeric
`validates :email, format: { with: /(\S+)@(\S+)/ }`
validates that the value of the specified attribute has a format that matches the regular expression
`validates :password, :confirmation => true`
validates that the value of the specified attribute matches a confirmation value
`validates :how_heard, inclusion: { in: ["Blog", "Newsletter"] }`
validates that the value of the specified attribute is available in the specified enumerable (array)
`validates :role, :exclusion => { :in => ["Admin", "SuperUser"] }`
validates that the value of the specified attribute is not available in the specified enumerable (array)
`valid?`
runs all the validations; automatically called when you try to create or save a model object

Run all the validations from the console like so:

```
$ rails console

>> e = Event.new
=> => #<Event id: nil, ...>
>> e.save
=> false
>> e.valid?
=> false
>> e.invalid?
=> true
>> e.errors.full_messages
=> ["Name can't be blank", ...]
```

Seeding Data

It's often handy to populate the database with example data.

Create records in the db/seeds.rb file

example:

```
Event.create!(
  name: 'BugSmash',
  location: 'Denver',
  price: 0.00,
  starts_at: 10.days.from_now,
  description: 'A fun evening of bug smashing!'
)
```

Step 1

`rake db:seed`

loads the data into the current environment's database

Step 2

Setting up RSpec

Step 1

```
group :test, :development do
 gem "rspec-rails"
end

group :test do
 gem "capybara"
end
```

add these lines to the bottom of the Gemfile

Step 2

```
bundle install
```

installs the gems

Step 3

```
rails generate rspec:install
```

runs the RSpec install generator

shortcut: rails g rspec:install

Running Specs

`rspec`
runs all the specs
`rspec --format doc`
runs all the specs and formats the output to include group and example names **shortcut:** rspec -f d
`rspec` ***`path/to/directory`***
runs all the spec files in the specified directory **example:** rspec spec/features
`rspec` ***`path/to/file`***
runs the specified spec file **example:** rspec spec/features/list_events_spec.rb
`rspec` ***`path/to/file:linenumber`***
runs the single code example at the specified line number in the specified spec file **example:** rspec spec/features/list_events_spec.rb:9

Common Form Helper Methods

`<%= form_for(@event) do |f| %>`
generates a form for a single resource
`<%= form_for([@event, @registration]) do |f| %>`
generates a form for a nested resource
`<%= f.label :name %>`
returns a label tag for labelling an input field for the specified attribute
`<%= f.text_field :name, autofocus: true %>`
returns an input tag of the “text” type for accessing the specified attribute
`<%= f.password_field :password %>`
returns an input tag of the “password” type (masked) for accessing the specified attribute
`<%= f.text_area :description, cols: 40, rows: 7 %>`
returns a textarea tag set for accessing the specified attribute
`<%= f.number_field :price %>`
returns an input tag of type “number” for accessing the specified attribute
`<%= f.datetime_select :starts_at %>`
returns a set of select tags pre-selected for accessing the specified datetime-based attribute
`<%= f.select :how_heard, ["Blog", "Newsletter"] %>`
returns a select tag and a series of contained option tags for the specified attribute

Example has_many Methods

`event.registrations`
returns an array of the event's registrations (may be empty)
`event.registrations.new(attributes = {})`
instantiates a new registration for the event (event_id = event.id), but does **not** save it
`event.registrations.create(attributes = {})`
instantiates and saves a new registration for the event (event_id = event.id)
`event.registrations << registration`
associates the registration with the event (event_id = event.id) and saves the registration
`event.registrations.size`
returns the number of registrations associated with the event
`event.registrations.delete(registration)`
removes the registration from the event (if :dependent => :destroy, it will also delete the registration)
`event.registrations.clear`
removes all registrations from the event (if :dependent => :destroy, it will also delete the registrations)

Example belongs_to Methods

`registration.event`
returns the event associated with the registration (may be nil)
`registration.event = an_event`
assigns the event to the registration

Example belongs_to Methods

	Single Resource	Nested Resource
Index:	/events	/events/:event_id/registrations
	events_path	event_registrations_path(event_id)
New:		/events/:event_id/registrations/new
	new_event_path	new_event_registration_path(event_id)