**myblog**

**# create pdf**

# add following gem to Gemfile

gem ‘prawn’

gem ‘prawnto’

<http://rubygems.org/gems/prawn>

<http://rubygems.org/gems/prawnto>

**$ rails new ror\_todo**

Add gems to Gemfile

gem ‘nifty-generators’

# make sure this gem get add in order to do the nifty-generators's authentication

gem 'bcrypt-ruby', :require => 'bcrypt'

gem ‘faker’

gem ‘populator’

gem ‘gravtastic’

gem ‘will\_paginate’

gem ‘kaminari’

gem ‘prawn’

gem ‘prawnto’

<https://github.com/ryanb/nifty-generators>

<http://rubygems.org/gems/faker>

1. Create user authentication using nifty-generators
2. Pagination by using will\_paginate or Kaminari gems
3. Create pdf output by using prawn and prawnto gems

#create layout

$ rails g nifty:layout

# create user authentication using nifty-generators

$ rails g nifty:authentication

$ rake db:migrate

# create Posts using scaffolding

$ rails g scaffold post title:string body:text

#migrate post

$ rake db:migrate

#modify routes to point root to post index

rm \public\index.html

# test application

$ rails s

**# while test application, make sure the css backgroup color don't get over write by scaffolds.css.scss**

# creating comments using scaffolding

$ rails g scaffold comment name:string body:text post:references

#post:references is pulling postid to comments table

#Add user to post model

<http://stackoverflow.com/questions/4954969/rails-3-migrations-adding-reference-column>

$rails g migration addUserIdToPosts user\_id:integer

$ rake db:migrate

# adding Login, Logout, and Signup links on the top of page

# we use **‘nifty-generators’, it has sets of variables already set for us to use "current\_user"**

**# as well as path such as logout\_path, login\_path, and signup\_path**

**#** adding **before\_filter :login\_required** on posts\_controller.rb

before\_filter :login\_required, :only => [:create, :new, :destroy, :update, :update]

or

before\_filter :login\_required :except => [:index]

**#add this line to post index before @post.save, this is to assign current user to a post**

@post = Post.new(params[:post])

@post.user = current\_user

#Assign relationship between post and comment models

**Post :**

has\_many : comments

belongs\_to :user

**Comment:**

belongs\_to :post

**User:**

has\_many :posts

#adding validation for post and comment model

Post:

validates :title, :presence => true

validates :body, :presence => true

Comment:

validates :name, :presence => true

posts/show.html.erb

<% @post.comments.each do |comment| %>

<p><b><%= comment.name %> said:</b></p>

<p><%= comment.body %></p>

<p><%= time\_ago\_in\_words comment.created\_at %> ago </p>

<p><%= link\_to "Delete", [@post,comment], :method => :delete, :confirm => "Are you sure?" %></p>

<% end %>

can be replace with partial by passing @post.comments

**<%= render @post.comments %>**

# create comments/\_comment.html.erb

<%= div\_for comment do %>

<p id="notice"><%= notice %></p>

<p><b><%= comment.name %></b> says</p>

<p><b>Body:</b><%= comment.body %></p>

<p><b>Posted: <%= time\_ago\_in\_words(comment.created\_at) %> ago</b></p>

<%= link\_to "Delete", [@post,comment], :method => :delete, :confirm=> "Are you sure?"%>

<% end %>

# prevent current user to modify the post from other user

# add this on show.html.erb

# adding comment section under each post. on show page

Rails Best Practices

<http://rails-bestpractices.com/posts/61-simplify-render-in-views>

#adding "debugger" for debug purpose

# open the console with **--debugger** option

# while open server to test the logic, you can use

$ rails s --debugger

$ list, up, down

$ p action\_name

$ irb

$ @comment.inspect

$ @comment.new\_record?

# check routes

$ rake routes

$ rake db:drop

$ rake db:create # create a database

$ rake db:migrate

$ rake db:reset

#create task by using scaffold command

$ rails g scaffold task name:string status:boolean due\_at:datetime

$ rake db:migrate

**# destroy scaffold model**

**$ rails destroy scaffold task**

**add pagination to list page**

# get all column from Task model

t = Task.column\_names

# string upcase and downcase, titleize

sort = "name"

sort.upcase # "NAME"

sort.downcase # "name"

sort.titleize #"Name"

add bootstrap

add **gem 'twitter-bootstrap-rails'** in assets sectionin the Gemfile

# create bootstrap

$ rails g bootstrap:install

<http://railscasts.com/episodes/145-integrating-active-merchant?autoplay=true>

# create a store controller

$ rails g controller store index

# change root to store\_path

#create a cart model

$ rails g model cart purchased\_at:datetime

#create a lineItem model

$ rails g model lineitem unit\_price:decimal product\_id:integer cart\_id:integer quantity:integer

# create a lineitem controller

$rails g controller lineitems new create

# create order model by using nifty\_scaffold

$ rails g nifty:scaffold order cart\_id:integer ip\_address:string first\_name:string last\_name:string card\_type:string card\_expires\_on:date

# create order\_transaction model

$ rails g model order\_transaction order\_id:integer action:string amount:integer success:boolean authorization:string message:string params:text

# do db:migrate for all of db script

$ rake db:migrate

# create populate.rake to populate data to different tables

# using populator gem, faker gem

**# using or equals(||=) to set variables in Ruby on Rails**

<http://www.paulsturgess.co.uk/articles/70-using-or-equals-to-set-variables-in-ruby-on-rails>

Using 'or equals' is great for creating and setting variables when they don't exist - useful for default values.

>> foo = “hello”

* “hello”

>> foo ||= “goodbye”

* “hello”

# create cart controller to show the cart with lineitems

$ rails g controller carts show