












































































# RailsAdmin

Overview & Best practices

Base de données  
wordpress (11)

wordpress (11)











- wp\_comments
- wp\_links
- wp\_options
- wp\_postmeta
- wp\_posts
- wp\_terms
- wp\_term\_relationships
- wp\_term\_taxonomy
- wp\_usermeta
- wp\_users

Champ	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
<input type="checkbox"/> comment_ID	bigint(20)		UNSIGNED	Non		auto_increment	    
<input type="checkbox"/> comment_post_ID	int(11)			Non	0		    
<input type="checkbox"/> comment_author	tinytext	utf8_general_ci		Non			    
<input type="checkbox"/> comment_author_email	varchar(100)	utf8_general_ci		Non			    
<input type="checkbox"/> comment_author_url	varchar(200)	utf8_general_ci		Non			    
<input type="checkbox"/> comment_author_IP	varchar(100)	utf8_general_ci		Non			    
<input type="checkbox"/> comment_date	datetime			Non	0000-00-00 00:00:00		    
<input type="checkbox"/> comment_date_gmt	datetime			Non	0000-00-00 00:00:00		    
<input type="checkbox"/> comment_content	text	utf8_general_ci		Non			    
<input type="checkbox"/> comment_karma	int(11)			Non	0		    
<input type="checkbox"/> comment_approved	varchar(20)	utf8_general_ci		Non	1		    
<input type="checkbox"/> comment_agent	varchar(255)	utf8_general_ci		Non			    
<input type="checkbox"/> comment_type	varchar(20)	utf8_general_ci		Non			    
<input type="checkbox"/> comment_parent	bigint(20)			Non	0		    
<input type="checkbox"/> user_id	bigint(20)			Non	0		    

Tout cocher / Tout décocher Pour la sélection :

Version imprimable Suggérer des optimisations quant à la structure de la table

Ajouter 1 champ(s) En fin de table En début de table Après comment\_ID Exécuter

Index					Espace utilisé		Statistiques	
Nom de l'index	Type	Cardinalité	Action	Champ	Type	Espace	Information	Valeur
PRIMARY	PRIMARY	371	 	comment_ID	Données	820,7 Kio	format	dynamique
comment_approved	INDEX	aucune	 	comment_approved	Index	256,0 Kio	Interclassement	utf8_general_ci
comment_post_ID	INDEX	aucune	 	comment_post_ID	Perte	440,5 Kio	Enregistrements	371
comment_approved_date_gmt	INDEX	aucune	 	comment_approved comment_date_gmt	effectif	636,2 Kio	Longueur enr. ø	1 049
comment_date_gmt	INDEX	aucune	 	comment_date_gmt	Total	1 876,7 Kio	Taille enr. ø	2 972 ø

Créer un index sur 1 colonne(s) Exécuter

Optimiser la table

Suivant Autoindex

Création Ven 03 Octobre 2008 à 20:08

Dernière modification Mer 07 Janvier 2009 à 05:59

Acceptable in the 90'

14 years ago

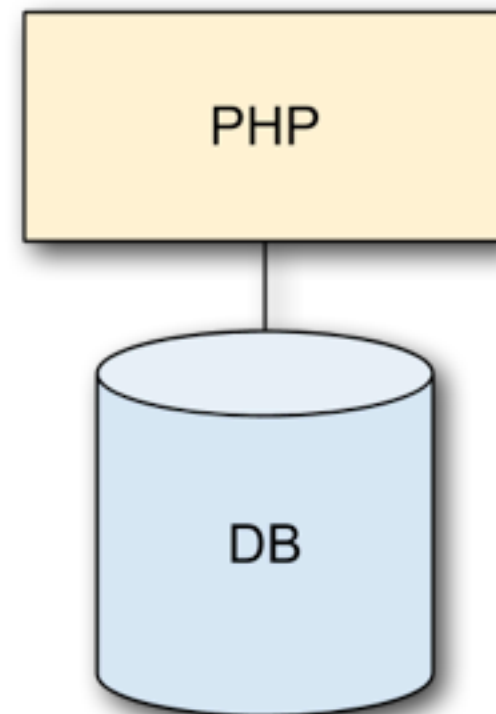
# Data Admin, stand-alone

- **+ Pros**

- Reliable
- Powerful

- **- Cons**

- No business rules whatsoever



CSVimport

Dashboard

Posts

Logout

ADMIN /

Posts

New Post

ID	Title	Content	Tag List	Created At	Updated At	
14	Post 7	Pellentesque at arcu quis diam ultricies viverra a id urna.	purple	June 27, 2012 01:41	June 27, 2012 01:41	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
13	Blog 6	Sed ornare sem at risus vulputate aliquam.	green	June 27, 2012 01:41	June 27, 2012 01:41	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
12	News 5	Praesent id convallis metus.	blue	June 27, 2012 01:41	June 27, 2012 01:41	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
11	Story 4	Etiam sed lorem diam, eu tempor nisi.	red	June 27, 2012 01:41	June 27, 2012 01:41	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
10	Article 3	Maecenas id augue neque, et molestie libero.	green	June 27, 2012 01:41	June 27, 2012 01:41	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
9	Post 2	Pellentesque mattis rhoncus augue sed euismod.	blue, two	June 27, 2012 01:41	June 27, 2012 01:41	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
8	Blog 1	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	red, one	June 27, 2012 01:41	June 27, 2012 01:41	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

Download: [CSV](#), [XML](#), [JSON](#)

Displaying all 7 Posts

Filters

SEARCH TITLE

SEARCH CONTENT

CREATED AT

 -

UPDATED AT

 -


TAGGINGS TAG NAME

☐ RED
 ☐ BLUE
 ☐ GREEN
 ☐ PURPLE
 ☐ ONE
 ☐ TWO

Filter

Clear Filters

dashboard > player



calio@fermaweb.com.br

Log out

rails admin

Dashboard

Division

Draft

Fan

League

Player

+





































select player to edit

history

add new

David

SEARCH

---	TEAM	NAME	POSITION	NUMBER	---	EDIT	DELETE
	New York Mets	David Wright		5		<a href="#">Edit</a>	<a href="#">Delete</a>
	Cincinnati Reds	David Weathers		2		<a href="#">Edit</a>	<a href="#">Delete</a>
	Atlanta Braves	David Ross		9		<a href="#">Edit</a>	<a href="#">Delete</a>
	New York Yankees	David Robertson		7		<a href="#">Edit</a>	<a href="#">Delete</a>
	Milwaukee Brewers	David Riske		5		<a href="#">Edit</a>	<a href="#">Delete</a>
	Toronto Blue Jays	David Purcey		3		<a href="#">Edit</a>	<a href="#">Delete</a>
	Tampa Bay Rays	David Price		8		<a href="#">Edit</a>	<a href="#">Delete</a>
	Baltimore Orioles	David Pauley		5		<a href="#">Edit</a>	<a href="#">Delete</a>
	Chicago Cubs	David Patton		5		<a href="#">Edit</a>	<a href="#">Delete</a>
	Boston Red Sox	David Ortiz		4		<a href="#">Edit</a>	<a href="#">Delete</a>
	Texas Rangers	David Murphy		1		<a href="#">Edit</a>	<a href="#">Delete</a>
	Baltimore Orioles	David Hernandez		6		<a href="#">Edit</a>	<a href="#">Delete</a>
	San Diego Padres	David Eckstein		9		<a href="#">Edit</a>	<a href="#">Delete</a>
	Cleveland Indians	David DeLucci		9		<a href="#">Edit</a>	<a href="#">Delete</a>
	Kansas City Royals	David DeJesus		9		<a href="#">Edit</a>	<a href="#">Delete</a>
	Milwaukee Brewers	David Bush		7		<a href="#">Edit</a>	<a href="#">Delete</a>
	Seattle Mariners	David Anderson		1		<a href="#">Edit</a>	<a href="#">Delete</a>
	Pittsburgh Pirates	Dave Davidson		2		<a href="#">Edit</a>	<a href="#">Delete</a>

18 players

mercredi 7 août 13

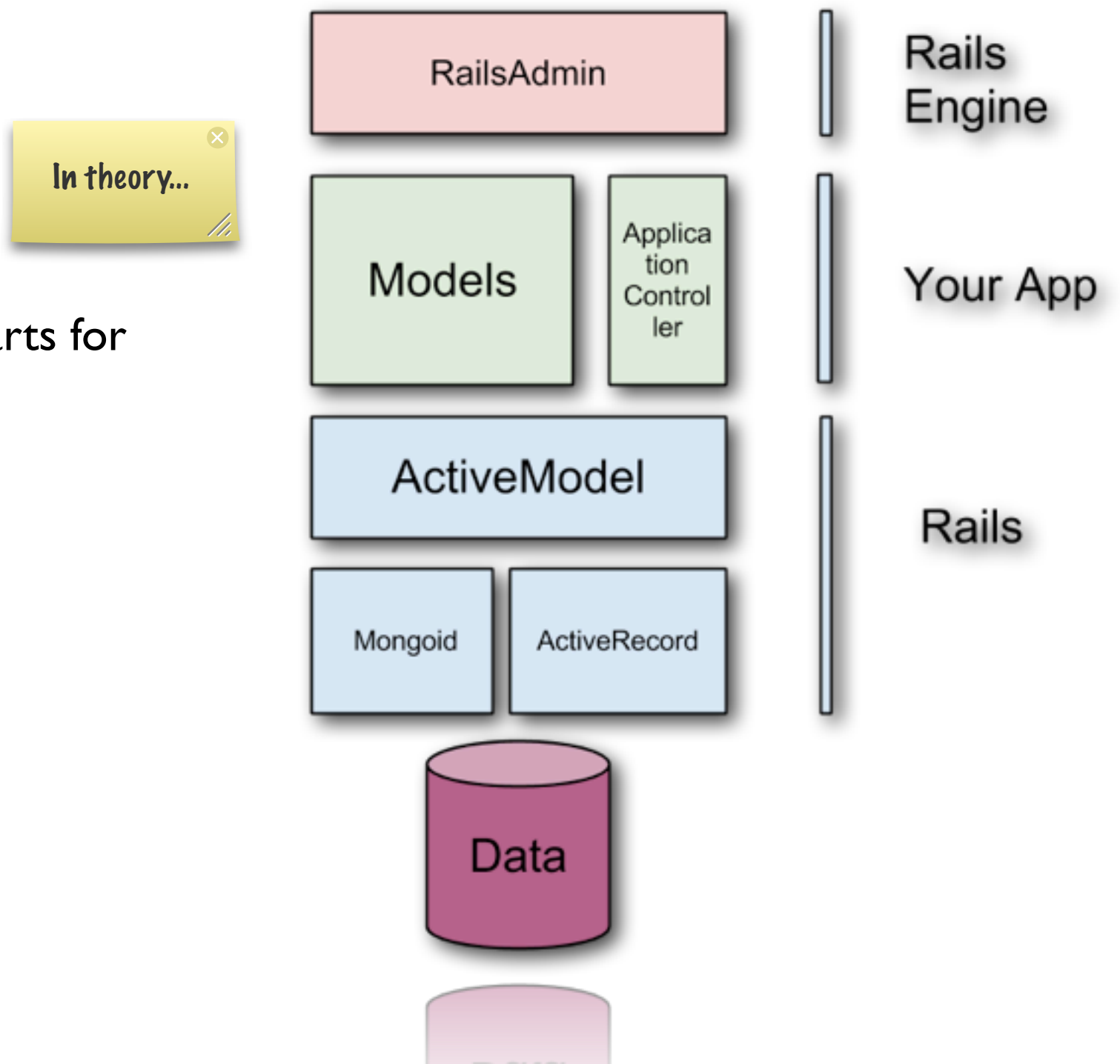
# Data Admin, in the application

- **+ Pros**

- Leverages the whole application or some parts for business rules etc.

- **- Cons**

- No direct (fast) DB administration
- Your code?





# RailsAdmin

- Admin **generator**
- **Immediate** startup
- **Steep** learning curve
- **Extensible**
- Powerful when **leverages Rails** Models and Cancan

# ActiveAdmin

- Admin **builder**
- **Slower** startup
- **Simple** DSL
- **Usable**
- Good for **ad-hoc admin building** when Models/Cancan rules don't infer the UI

Grain of salt, common sense, etc.

# Pitfall

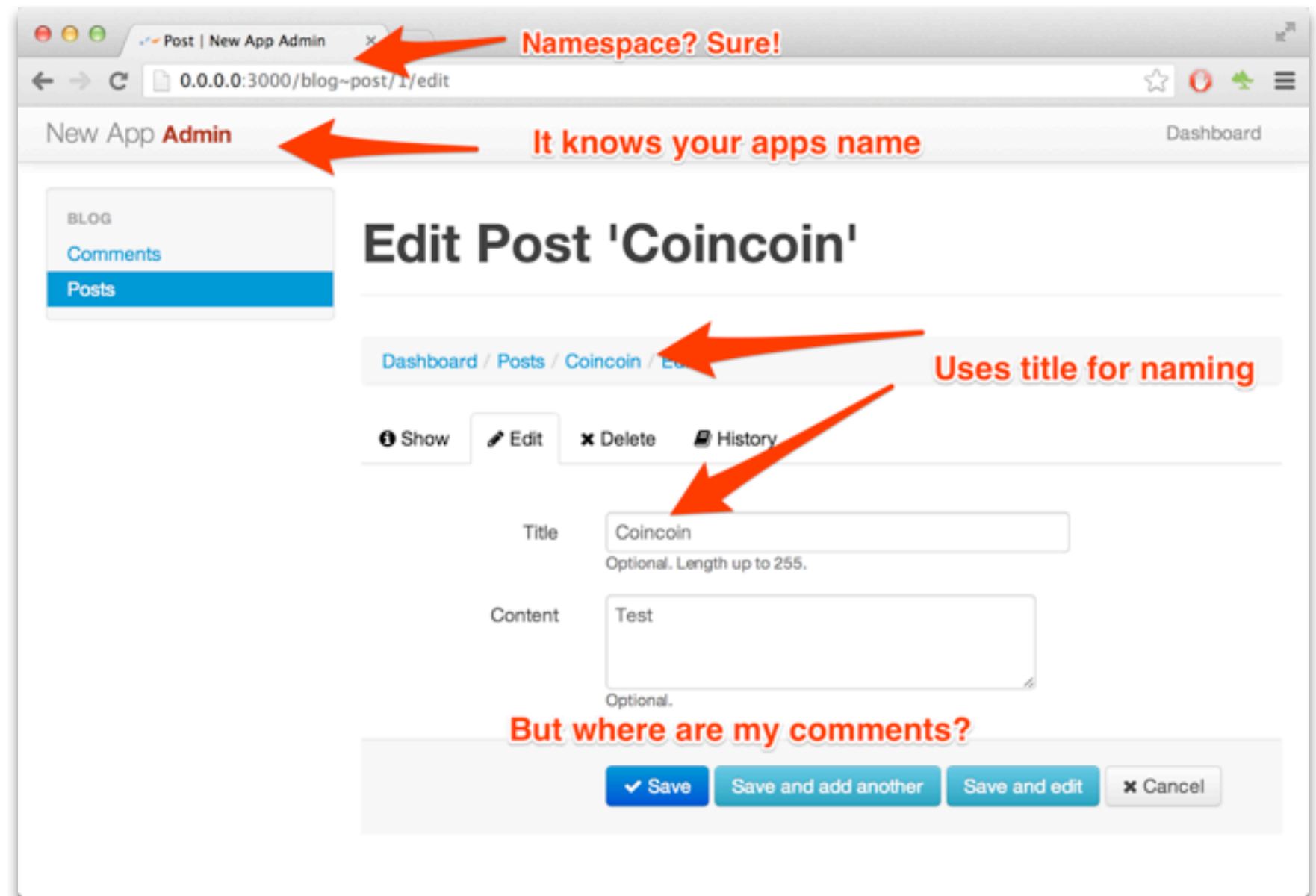
ActiveModel does not provide anything for associations!

- ActiveSupport lets you do the work
  - => with some help from Formtastic :/
- RailsAdmin shortcuts ActiveModel
  - => Complex custom code laying on the ORMs :/
  - But hey! it works®\*

\* Unless you screwed up your models associations definitions

# Minimum installation

- rails new new\_app
- rails g model blog::post  
title:string content:text
- rails g model blog::comment  
content:text post:belongs\_to
- gem 'rails\_admin'
- mount RailsAdmin::Engine => "





# Big picture

- Actions
  - Map routing to templates and controller
- Sections
  - Map field configuration to each action
- Fields
  - Handle fields configurations

# Actions-Sections

## CRUD mapping

- new => new < edit < base
- edit => update < edit < base
- edit/new nested => nested < edit < base
- edit/new modal => modal < edit < base
- index => list < base
- export => export < base

# Example



# The issue...

```
config.model 'PlayerType' do
  list do
    field :name
    field :description
  end

  show do
    field :name
    field :description
    field :_id
    field :created_at
    field :updated_at
  end

  edit do
    field :name
    field :description
    field :product_types
  end
end

config.model 'PlayerVersion' do
  list do
    field :name
    field :description
    field :updated_at
  end

  show do
    field :name
    field :description
    field :_id
    field :created_at
    field :updated_at
  end

  edit do
    field :name
    field :description
  end
end

config.model 'ProductType' do
  list do
    field :name
  end
end
```

- Never-ending list of section/fields definitions

# Solutions

- Use **configure** as much as you can (doesn't force the field list in opt-in mode)
- Use **order**
- Move each model configuration to a **rails\_admin block** in class definition
- **Hide fields** you don't need instead of listing them (less overhead usually)

# Associations

```
class Blog::Post < ActiveRecord::Base
  has_many :comments
end
```

Title

Optional. Length up to 255.

Content

Optional.

Comments

Blog::Comment #1

Choose all

Optional.

Clear all

+ Add a new Comment

I don't want the #1 thing :( I don't want to see other posts comments :(



# The associations

- `has_many => nested`
- `has_one => nested`
- `belongs_to => modal`
- `has_many :through => modal`
- `has_and_belong_to_many => modal`
- `It makes sense®`

# Back to our models...

```
class Blog::Comment < ActiveRecord::Base
  belongs_to :post

  def title
    content.try(:[], 0..20)
  end
end

class Blog::Post < ActiveRecord::Base
  has_many :comments

  accepts_nested_attributes_for :comments
end
```

The form displays nested attributes for a **Post** and its **Comments**. The **Post** section includes fields for **Title** (First post) and **Content** (Optional). The **Comments** section includes a button to **+ Add a new Comment** and a list of existing comments. Each comment has its own **Content** field. At the bottom, the **Post** is selected via a dropdown menu (First post), with buttons to **+ Add a new Post** and **Edit this Post**.

**I don't need the post here, I'm editing it!**

# You tell your ORM about it

```
class Blog::Comment < ActiveRecord::Base
  belongs_to :post, inverse_of: :comments

  def title
    content.try(:[], 0..20)
  end
end
```

The diagram illustrates the relationship between the Ruby code and a web form. A vertical blue line separates the code on the left from the form on the right. The form is divided into two sections. The top section, labeled 'Title' and 'Content', corresponds to the `def title` method in the code. It contains a text input field with the value 'First post' and a label 'Optional. Length up to 255.' below it. The bottom section, labeled 'Comments', corresponds to the `content` attribute in the code. It contains a dropdown menu with a plus sign and the text '+ Add a new Comment', a label 'Optional.', and two tabs: 'First comment' and 'Comment (new)'. Below the tabs is a text input field with a vertical cursor and a label 'Optional.' below it.

Title	<input type="text" value="First post"/>
	Optional. Length up to 255.
Content	<input type="text"/>
	Optional.
Comments	<div><div>▼</div><div>+ Add a new Comment</div></div>
	Optional.
	<div><div>First comment</div><div>Comment (new)</div></div>
Content	<input type="text"/>
	Optional.

# Stay on the Golden path

- **Convention** over configuration!
- RailsAdmin gets as much intel as it can from your models:
  - Fields
  - Associations
  - Validations
- Don't configure RailsAdmin when you can tweak your app to Rails' best standards!



# But wait! Admin!

## Passwords and stuff!

- `gem 'devise'`
- `rails g devise:install`
- `rails g devise user`
- `rails g migration add_user_id_to_blog_posts user:belongs_to`

```
class Blog::Post < ActiveRecord::Base
  has_many :comments
  belongs_to :user

  accepts_nested_attributes_for :comments
end
```

- `config/rails_admin.rb:`

```
RailsAdmin.config do |config|
  config.current_user_method &:current_user
end
```

Signed in successfully.



# Who am I?

```
def name
  email.try(:split, '@').try(:first).try(:humanize)
end
```

Dashboard username@example.com [Log out](#)

## Edit User 'Username'

Dashboard / Users / Username / Edit

[Show](#) [Edit](#) [Delete](#) [History](#)

Email

username@example.com

Required. Length up to 255.

Password

Required. Length of 8-128.

**Nailed it!**

**This is me.**



# This is called authentication.

## Do you have authorization?

- `gem 'cancan'`
- `rails g migration add_profile_to_users profile:string`
- `config/rails_admin.rb:`

```
config.authorize_with :cancan
```

- `rails g cancan:ability`

```
def initialize(user)
  if user.profile == 'admin'
    can :manage, :all
  end
end
```

**CanCan::AccessDenied in RailsAdmin::MainController#edit**

**You are not authorized to access this page.**

# Some other profiles



```
def initialize(user)
  case user.profile
  when 'admin'
    can :manage, :all
  when 'writer'
    can :access, :rails_admin
    can :dashboard
    can :manage, Blog::Post, { user_id: user.id }
  end
end
```

**Safe..** ←

**Own posts only** ←

- Admin can do anything
- Writer can
  - access rails\_admin
  - access the dashboard
  - CRUD Blog::Post that he owns
  - for new posts, user\_id will be forced to his id

# Ok, what do I get for my writers ?

Title	<input type="text"/>	
		Optional. Length up to 255.
Content	<div></div>	
		Optional.
User	<div><input type="text" value="Search"/>▼</div>	 <b>I don't need that then?</b>
		Optional.
Comments	<div> <b>+ Add a new Comment</b></div>	 <b>What if I don't want my writers to see ANY comments ?</b>
		Optional.

# At last, some RailsAdmin field configuration!

```
class Blog::Post < ActiveRecord::Base
  has_many :comments
  belongs_to :user

  accepts_nested_attributes_for :comments

  rails_admin do
    configure :user do
      visible do
        bindings[:controller].current_ability.can? :edit, User
      end
    end

    configure :comments do
      visible do
        bindings[:controller].current_ability.can? :edit, Blog::Comment
      end
    end
  end
end
```

# Deal with serialized fields

- rails g migration add\_metadata\_to\_blog\_posts metadata:text
- models/blog/post.rb `serialize :metadata, Hash`

Title	<input type="text" value="I need a title"/> Optional. Length up to 255.
Content	<div><div>Some content too good to print</div><div></div></div> Optional.
Metadata	<div><div>--- !ruby/hash:ActionController::Parameters block: front_page subtitle: some subtitle needed as well</div><div></div></div> Optional.

# Now what?

- It's **error** prone :(
- It's **ugly** :(
- It's not user **friendly** :(
- It's **dangerous** :(
- **What should I do?**



# Use some ‘virtual’ fields?

```
configure :block do
  def value
    bindings[:object].metadata[method_name]
  end

  def allowed_methods
    'metadata'
  end

  def parse_input(params)
    params['metadata'] ||= bindings[:object].metadata
    params['metadata'][method_name] = params.delete(method_name)
  end
end
```

- You don't need your object to respond to 'block' !
- You can create RailsAdmin fields out of the blue
- You just need to define **value** and **parse\_input** methods (in & out the form)
- **allowed\_methods** lets you modify the mass-assignable params on the object (safety)

# Say I need another one like 'block'?

- Let's create a metadata RailsAdmin field type, in our application
- lib/rails\_admin/metadata.rb

```
require 'rails_admin/config/fields/base'

module RailsAdmin
  class Metadata < RailsAdmin::Config::Fields::Base
    RailsAdmin::Config::Fields::Types::register(self)

    def value
      raise 'No metadata!' unless bindings[:object].respond_to?(:metadata)
      bindings[:object].metadata[method_name]
    end

    def allowed_methods
      'metadata'
    end

    def parse_input(params)
      params['metadata'] ||= bindings[:object].metadata
      params['metadata'][method_name] = params.delete(method_name)
    end
  end
end
```

# Time to leverage

```
configure :block, :metadata
configure :sticky, :metadata do
  view_helper :checkbox
end
configure :published, :metadata do
  view_helper :checkbox
end
configure :priority, :metadata do
  view_helper :number_field
  default_value 0
end
```

Block	<input type="text" value="front_page"/> Optional.
Sticky	<input checked="" type="checkbox"/> Optional.
Published	<input checked="" type="checkbox"/> Optional.
Priority	<input type="text" value="0"/> Optional.

No pollution in the model!

# Some actions?

Dashboard Log out

## List of Posts

Dashboard / Posts

List Add new Export History

Add filter Selected items

Filter Refresh

Export found Posts

	Id	Title	Content	Created at	Updated at	Block	
<input type="checkbox"/>	4	I need a title	Some content too go...	August 03, 2013 18:08	August 04, 2013 14:16	front_page	
<input type="checkbox"/>	3	Test	content	August 03, 2013 17:12	August 03, 2013 17:12	-	...

2 posts

Application (root) level actions

Instance (member) actions

collection level actions

# I need to publish my posts!

```
require 'rails_admin/config/actions/base'
```

```
module RailsAdmin
```

```
  class Publish < RailsAdmin::Config::Actions::Base
    RailsAdmin::Config::Actions.register(self)
```

```
    register_instance_option :member do
      true
    end
```

```
    register_instance_option :http_methods do
      [:get, :post]
    end
```

```
    register_instance_option :controller do
```

```
      Proc.new do
```

```
        if request.post?
```

```
          @object.metadata['published'] = true
```

```
          if @object.save && false
```

```
            redirect_to_on_success
```

```
          else
```

```
            flash[:error] = 'You probably screwed up smtg again'
```

```
          end
```

```
        end
```

```
      end
```

```
    end
```

```
  end
```

```
end
```

```
publish do
  only 'Blog::Post'
  link_icon 'icon-check'
end
```

Define an action, then use it

# Then the view part, simple stuff

```
h4 Publish "#{@object.title}"?
= form_tag :url => publish_path(:model_name => @abstract_model.to_param, :object => @object), :method => 'post' do
  .form-actions
    button.btn.btn-danger type="submit" data-disable-with="#{t('admin.form.confirmation')}}"
      i.icon-white.icon-ok
      = t("admin.form.confirmation")
    | &nbsp;
    button.btn type="submit" name="_continue" data-disable-with="#{t('admin.form.cancel')}}"
      i.icon-remove
      = t("admin.form.cancel")
```

```
en:
  hello: "Hello world"
```

```
admin:
  actions:
    publish:
      title: "Publish %{model_label}"
      menu: "Publish"
      breadcrumb: "Publish"
      link: "Publish %{model_label}"
      done: "published"
```

The view goes to rails\_admin/  
main/publish.slim

Then add some translations for  
proper integration

 Show  Edit  Delete  History  Publish

Publish "I need a title"?

✓ Yes, I'm sure

✗ Cancel



# I'm in trouble

- Check that your models are ok (**rails c**)
- **bundle open rails\_admin**
- **trace @abstract\_model, @object** in **actions**
- **trace @abstract\_model, bindings[:object]** in **fields**

# Someone said I need tests, I probably need tests, did I say I need tests?

```
mount RailsAdmin::Engine => '', as: 'rails_admin'
```

*## Go somewhere*

```
visit rails_admin.dashboard_path
visit rails_admin.new_path(model_name: 'blog~post')
visit rails_admin.edit_path(model_name: 'blog~post', id: post.id)
```

[https://github.com/sferik/rails\\_admin/wiki/Rspec-with-capybara-examples](https://github.com/sferik/rails_admin/wiki/Rspec-with-capybara-examples)

*## Assert you landed somewhere*

```
expect(current_path).to eq rails_admin.dashboard_path
expect(current_path).to eq rails_admin.new_path(model_name: 'blog~post')
expect(current_path).to eq rails_admin.edit_path(model_name: 'blog~post', id: post.id)
```

*## Click links*

*# From anywhere*

```
page.find('.dashboard_root_link a').click # to a root action
page.find('[data-model=blog~post] a').click # to any model index
```

*# From any collection action*

```
page.find('.new_collection_link a').click # to another collection action
```

*# From any member action*

```
page.find('.edit_member_link a').click # to another member action
```

*# From the dashboard*

```
page.find('.blog_post_links .index_collection_link a').click # to any collection action
```

*# From the index page of a model*

```
page.find('.blog_post_row[1] .show_member_link a').click # to any row's member action
```

*## Assert the content*

```
expect(page.find('#blog_post_title')).to have_content 'Blog Post Title' # of an edit/new form input
expect(page.find('.blog_post_row[1] .title_field')).to have_content 'Blog Post Title' # of an index table row cell
expect(page.find('.alert')).to have_content 'Post successfully created' # of a flash message
```

# Thanks for listening!

Full example code

[https://github.com/bbenezech/blog\\_admin](https://github.com/bbenezech/blog_admin)

[benoit.benezech@gmail.com](mailto:benoit.benezech@gmail.com)

@bbenezech github/Skype/twitter

Freelance