Plug-in application for automatic generation of administration panel

Piotr Jakubowski

December 13, 2010

Contents

1 Introduction **3**
- 1.1 Problem Description . . . 3
- 1.2 Existing Solutions . . . 4
  - 1.2.1 Active Scaffold . . . 4
  - 1.2.2 Typus . . . 6
  - 1.2.3 admin_data . . . 8
  - 1.2.4 rails_admin . . . 9
- 1.3 Solution Proposal . . . 10
  - 1.3.1 Solution Proposal . . . 10
  - 1.3.2 Projected Challenges . . . 10

List of Figures

1.1 Active Scaffold 5
1.2 Automatically generated form in Active Scaffold 6
1.3 Typus main page 7
1.4 Automatically generated form in Typus 7
1.5 Admin data 8
1.6 rails_admin dashboard 9
1.7 Form generated in rails_admin 10

Chapter 1

Introduction

1.1 Problem Description

Along with the huge popularization of the Internet there appeared enormous market for all sorts of web applications. Every month big companies, as well as small startups, try to get as much of this market by bringing in new features and shocking the crowd with innovative functionalities. Surprisingly, the market seems to be far from saturation as brilliant ideas except for fulfilling known demand create new needs as well.

New tools started to emerge in order to meet the needs of Internet population. Dynamic languages started to supersede old and known. Python and Ruby became the new Java and C. The speed of development surpassed the speed of execution as technical limitations and price of servers are not the issue any more. Moreover, web development frameworks made it possible to code the application on the new level of abstraction - we can instantaneously start to implement our business logic, without the need to think about how we should persist our entities into database and how we should handle requests and respond to them. Java developers got Spring and Hibernate, but the noteworthy Ruby on Rails took web development to the whole new level. The threshold of making the ideas happen has been successfully shrunken.

As mentioned before, today's software development has been focused on the effectiveness and speed of development. A lot of work has been done to create developer-oriented tools that try to make developing business ideas effortless.

Of course, there is plenty of room for improvements. One of them is the idea of automating process of creating the administration panel for the application. Very often administration part of the application is mundane collection of CRUD[1] actions. It is the "must do" part.

User-oriented features is the target thing of development. This is what makes money and keeps us all employed. It seems like possibility to be able to automatically generate panel for administering our resources would make development of web application even more dynamic. It would bring plenty of benefits, such as:

- Whole team can focus on what really matters, which is creating functionality that would be an added-value for users of the application
- Adding new features and resources to the application would require less effort - as soon as the user-oriented functionality is ready the administration part is automatically generated.
- Business people connected with the project can instantly browse new resources in the application.

Thereby, the aim of this thesis is to create and demonstrate the process of creation of plugin that would enable automatic generation of administration part of the application. The plugin would be developed for Ruby on Rails framework mentioned before.

1.2 Existing Solutions

Thanks to its rapidly growing popularity, Ruby and Ruby on Rails got very broad community. Not only does it result in a huge number of places you can ask for help, but also in all kind of libraries that would make the life of web developer easier. And as it seems I am not alone stating the issue described above. There are already several implementations attacking the problem of admin interface. Some of them are more successful than others, there are different kinds of approaches and different outcomes.

In order to see what market has to offer I looked through available solutions and chose 4 most popular ones in order to see what kind of functionalities they provide and what flaws they suffer from.

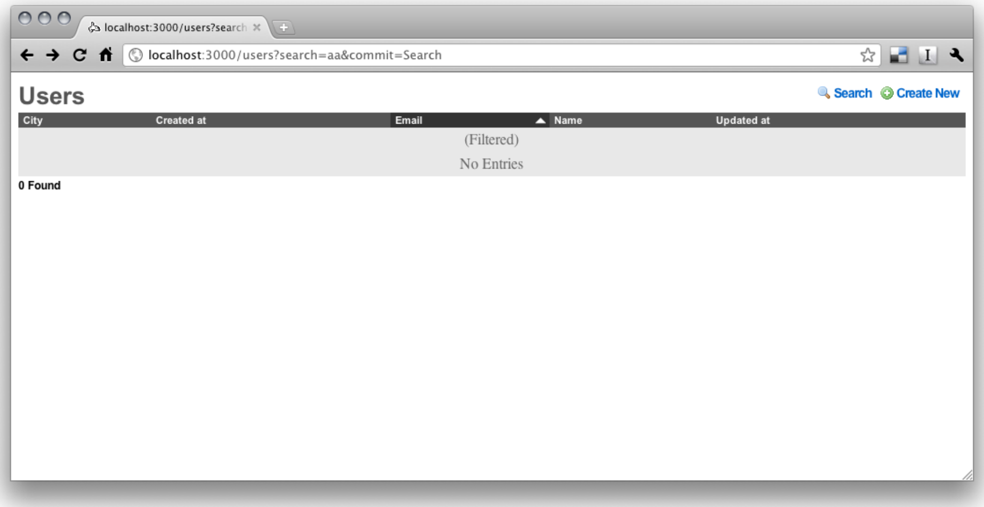
1.2.1 Active Scaffold[2]

According to Ruby Toolbox[3] it is the most popular plugin for admin interface generation. Unfortunately, it seems like the days of glory for this gem are long

[1]Create Read Update Delete
[2]http://activescaffold.com/
[3]http://ruby-toolbox.com/categories/rails_admin_interfaces.html

gone. The plugin does not support Rails in version 3, which despite being released very recently already became a standard for new applications.

Technicalities aside, it seems to be quite functional. The almost out-of-the-box look is presented on figures 1.1 and 1.2


localhost:3000/users?search
localhost:3000/users?search=aa&commit=Search
Search
Create New

Users

City	Created at	Email	Name	Updated at
(Filtered)				
No Entries				

0 Found

Figure 1.1: Active Scaffold

Active Scaffold ships with comprehensive API for customizing its behavior. User can change which columns are visible in which view or how the system behaves during particular actions. Moreover, it uses Asynchronous JavaScript and XML (AJAX) for interaction with user which gives nice and responsive user interface.

Unfortunately, it has big disadvantage. It is not entirely automatic. It requires developer to take following action to set it up:

- Add command to the layout of the application that will include styles and javascript for Active Scaffold.
- For every resource he wants to administer, he needs to create controller and add code that will set it up.
- Set up url for Active Scaffold actions for given controller by adding code to routes file.

This results in following inconveniences:

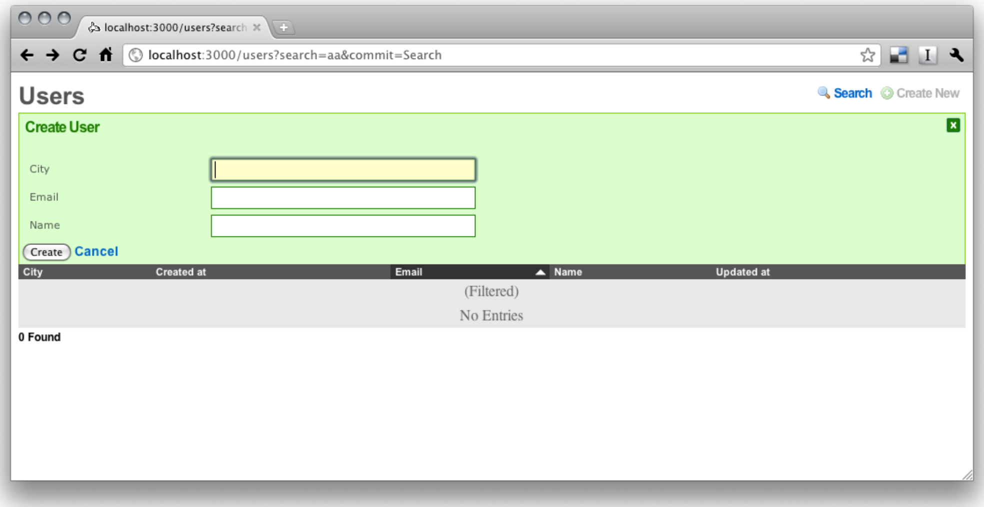


Figure 1.2: Automatically generated form in Active Scaffold

- Active Scaffold mixes with our application code.
- Every time we create some model we need to take care to set up Active Scaffold for it.

It seems like Active Scaffold instead of being "stand-alone" solution for administration is a set of commands, that speeds up set up of administration interface. It gives flexibility, but is not as hassle-free as could be and as developers would like it to be.

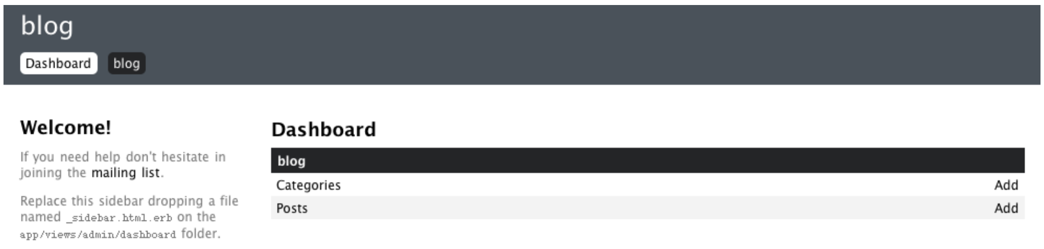
1.2.2 Typus[4]

Typus is reported to be the second most popular Rails admin interface generator. As opposed to Active Scaffold, its development team is still active. Thanks to that, it has been adapted to Rails 3 already.

Installation and set up is quick and effortless. We just need to install Typus and run one command. That way we get to the /admin path in our application. Its user interface is presented on figures 1.3 and 1.4.

Documentation provides thorough description of possibilities of Typus' customization. We can decide which columns will be visible, how admins can search our models and so on. Most of the configuration is done by specifying

[4]http://core.typuscms.com/



Welcome!

If you need help don't hesitate in joining the mailing list.

Replace this sidebar dropping a file named `_sidebar.html.erb` on the `app/views/admin/dashboard` folder.

Figure 1.3: Typus main page

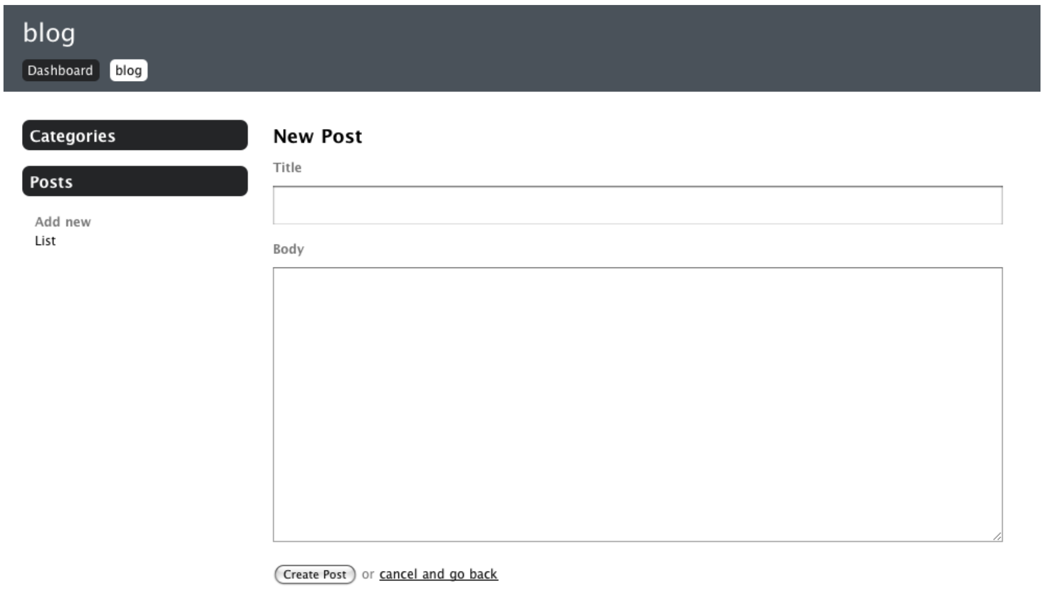


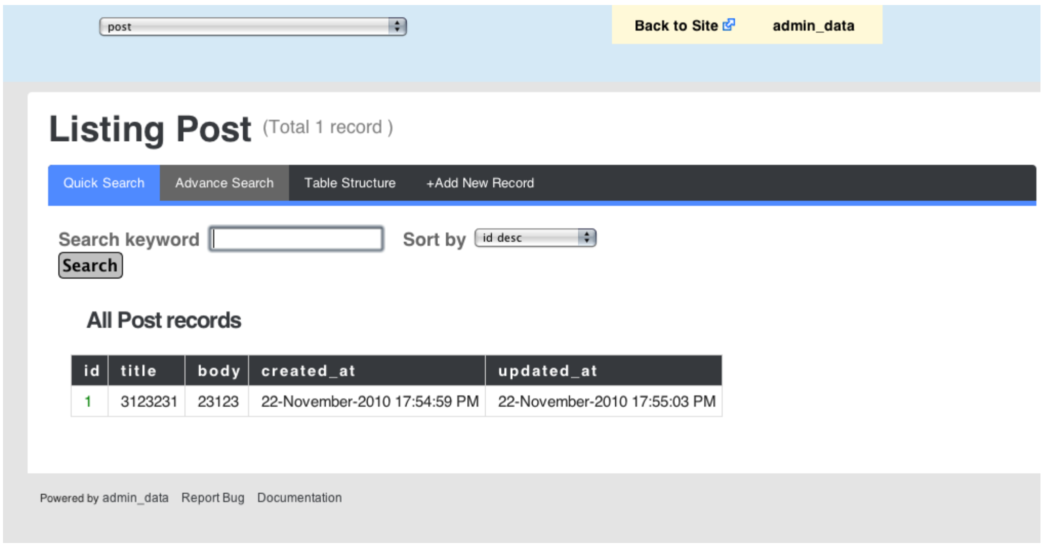
Figure 1.4: Automatically generated form in Typus

appropriate YAML files. In addition, Typus is supposed to handle file uploads in your application, which is very convenient. However, it will happen only if your application is using particular plugin for files upload called Paperclip.

But again, similiar as it was in Active Scaffold, Typus does not work entirely "on-the-fly". The generator we have to run in order to set Typus up, except for copying needed HTML, javascript and image files, creates controllers that serve particular Models of our application. As said before, this means that we can get full control on how given controller behaves - we can override Typus implementation with our own - but such approach is not as dynamic in terms of adapting to upcoming resources in the application. Of course, now it is just the sake of running simple command that would take care of everything once we introduce some new Model into application, but surely it is not something we just install and can forget about.

1.2.3 admin_data

Admin_data is next one on the list. Again, the project seems to be still maintained and has been upgraded to Rails 3 already. Installation is very much alike to the one of Typus. But, after installation we do not need to run any generators or create any kind of files. We instantly get access to /admin_data path and figure 1.5 presents what more or less we would see after trying it out.



id	title	body	created_at	updated_at
1	3123231	23123	22-November-2010 17:54:59 PM	22-November-2010 17:55:03 PM

Figure 1.5: Admin data

Of course, it provides us with some level of customizability, though it is not as wide as in two other projects. But, admin_data has the level of dynamism I have been looking for. It creates the list of models on the fly, so while developing our application we will constantly have the most current version of our models lists, no question asked. Moreover, we can also see table's structure for given model, which may come in handy at times.

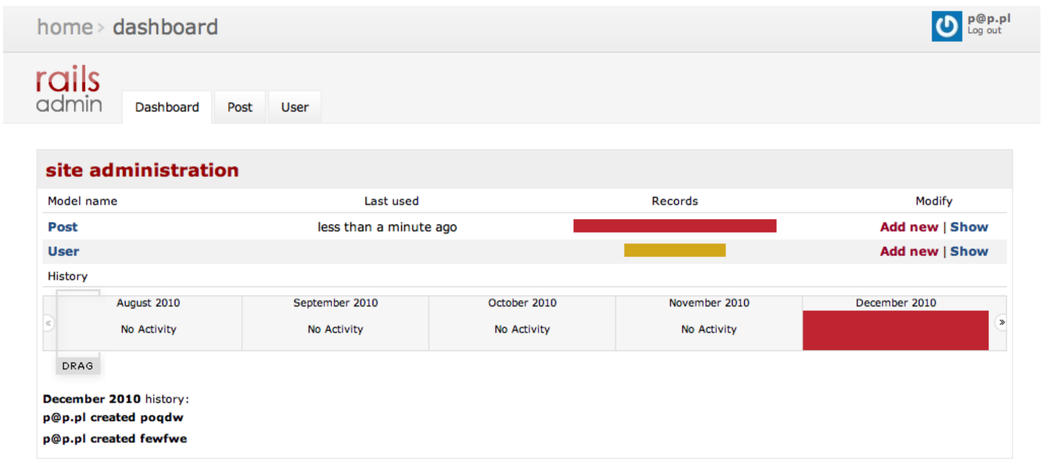
On the other hand, you can clearly see that admin_data has been created with programmers in mind. The user interface can be very hard to comprehend for people that do not know what is SQL or do not care how developers organized tables in the database. Therefore, it could be very hard to convince our business people that it is something they can use.

1.2.4 rails_admin

Although rails_admin is not present on the ruby-toolbox list it has been very popular lately. It has been developed during Ruby Summer of Code[5]. It has been developed strictly for Rails 3 so it uses all the best features and the best practices from the newest version of the framework.

Installation is very quick, although we need to go through few steps. Moreover, it depends on devise[6] gem, which takes care of users authentication, as rails_admin's default option is to make people to log in to the administration panel.

What we get after installation is very nice and user-friendly panel(figure 1.6). In addition, it tracks changes in records, what gives us functionality of seeing whole history of activities on records as well as people that made them. Additionally, we have that data presented in form of nice graph that gives us an idea of what was happening in the system throughout the month or year.



site administration

Model name	Last used	Records	Modify
Post	less than a minute ago	🟥	Add new | Show
User		🟨	Add new | Show

History

August 2010	September 2010	October 2010	November 2010	December 2010
No Activity	No Activity	No Activity	No Activity	🟥

December 2010 history:
p@p.pl created poqdw
p@p.pl created fewfwe

Figure 1.6: rails_admin dashboard

Rails_admin seems to have real potential and although I have not seen it in use in production yet, soon it maybe become real power of Rails applications.

[5]http://rubysoc.org/
[6]https://github.com/plataformatec/devise

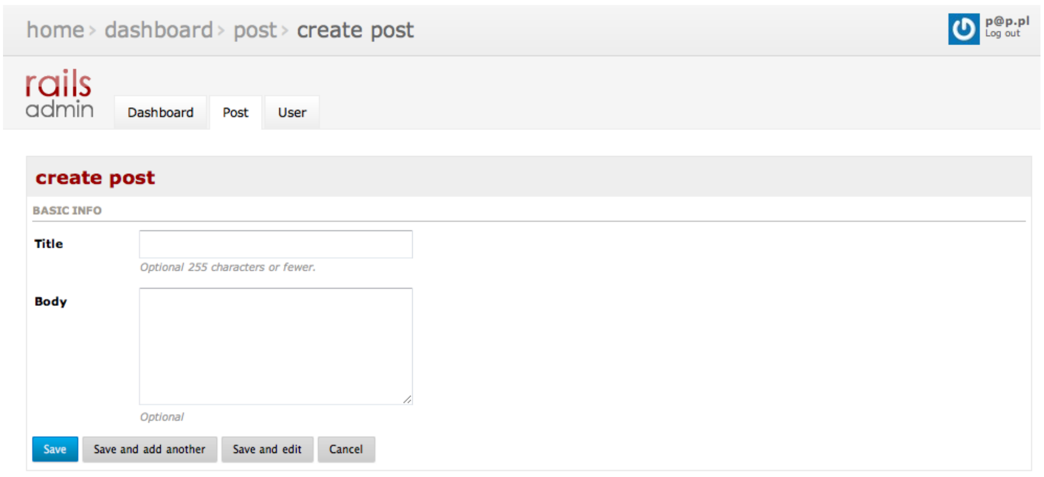


Figure 1.7: Form generated in rails_admin

Definitely, ideas used in that project are very interesting and may be very useful.

1.3 Solution Proposal

1.3.1 Solution Proposal

TBD

1.3.2 Projected Challenges

TBD