keep it simple

- 中文
- English
- About Me

« Prev 1 2 3 4 5 Next »

cloud_tag plugin for rails (read Chinese version)

2010-02-02

Found this cool flash tag cloud from andy's blog. Readly like it, so create this plugin , and add to my blog.

flash copied from http://www.tag-cloud.de/wpcloud/?r=tag_cloud

use:
1.puts cloud_tag folder into project *vendor/plugins* folder
2.use in view :

```
<%= cloud_tag(tag_link_hash, height = "240", width="200") %>
```

height and **width** are optional.
tag_link_hash should be formatted like:

```
{"tag1" => "www.tag1.com", "tag2" => "www.tag2.com"}
```

example:
http://zhangzhe.heroku.com/example/cloud_tag.html

Github address:
http://github.com/zhangzhe/cloud_tag

tags: cloud_tag, plugin, rails

Comments (3)

rails code evaluation (read Chinese version)

2010-02-01

metric_fu
There are tools like Saikuro, Flog, Flay, Rcov, Reek, Roodi, Churn to evaluate rails code. metric-fu Mix them together.

Install

```
sudo gem install metric_fu
```

create metric_fu.rake file

```
begin
  require "metric_fu"
rescue LoadError
end
```

run

```
rake metrics:all
```

Result of flog

Total Flog score for all methods: 438.7

Average Flog score for all methods: 6.7

File	Total score	Methods	Average score	Highest score
/app/controllers/blogs_controller.rb	86	9	10	41
/app/models/pinyin.rb	49	7	7	37
/app/helpers/application_helper.rb	70	6	12	31
/app/controllers/comments_controller.rb	23	1	23	23
/app/models/blog.rb	21	2	10	18
/app/models/coderay_string.rb	17	1	17	17
/app/models/formatable_string.rb	78	13	6	15
/app/controllers/upload_files_controller.rb	22	2	11	15
/app/controllers/blog_groups_controller.rb	13	2	7	8
/app/controllers/sessions_controller.rb	10	2	5	7
/app/controllers/application_controller.rb	22	6	4	4
/app/models/blog_group.rb	4	1	4	4
/app/controllers/tags_controller.rb	4	1	4	4
/app/models/upload_file.rb	4	1	4	4

rails best practices
Another gem rails best practices gives more advices from code standards, and detail design.
install

sudo gem install rails_best_practices --source http://gemcutter.org

run

rails_best_practices .

The advice to my blog:

./app/views/blog_groups/_show.html.erb:8 - replace instance variable with local variable
./app/views/blog_groups/_show.html.erb:8 - replace instance variable with local variable
./app/views/blogs/_admin_tool.html.erb:2 - replace instance variable with local variable
./app/views/blogs/_form.html.erb:3 - replace instance variable with local variable
./app/views/blogs/_form.html.erb:3 - replace instance variable with local variable
./app/views/blogs/_form.html.erb:13 - replace instance variable with local variable
./app/views/blogs/_show.html.erb:2 - replace instance variable with local variable
./app/views/blogs/_show.html.erb:4 - replace instance variable with local variable
./app/views/blogs/_show.html.erb:6 - replace instance variable with local variable
./app/views/comments/_show.html.erb:5 - replace instance variable with local variable
./app/controllers/blogs_controller.rb:26,33 - use before_filter for new,edit
./app/controllers/upload_files_controller.rb:2,12 - use before_filter for create,index
./app/controllers/blogs_controller.rb:42 - move model logic into model (@blog called_count > 4)
./db/migrate/20100129115908_acts_as_taggable_on_migration.rb:10 - always add db index (taggings => tagger_id)

tags: metric_fu, rails_best_practices, refactor, rails

Comments (0)

set up autotest to use test_notifier (read Chinese version)

2010-01-19

It is a pity that Ubuntu can not set up autotest to use Growl.

There is a better solution: test_notifier

sudo gem i test_notifier

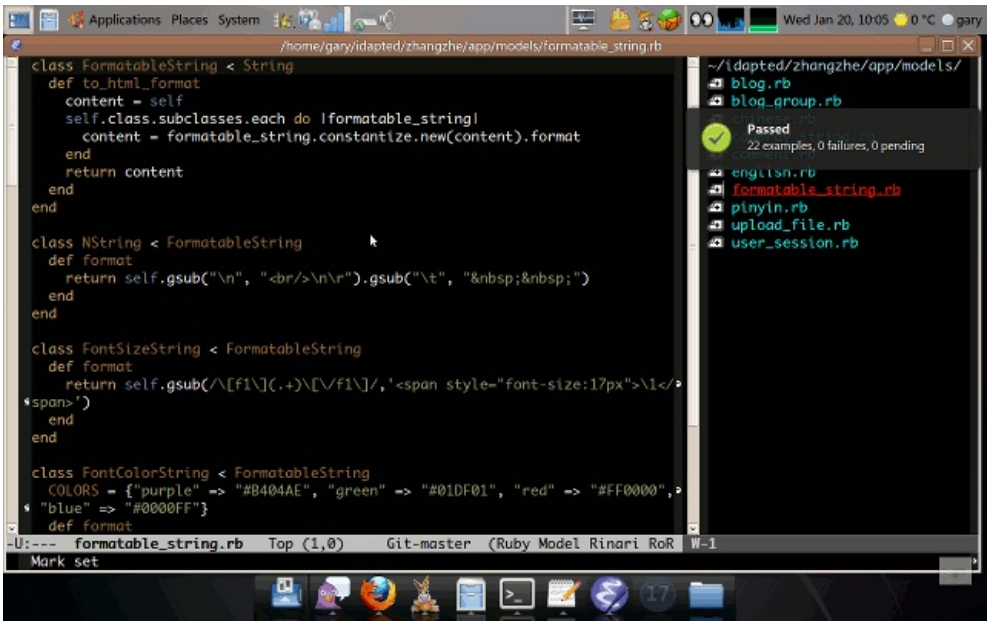
and add to the file *~/.autotest*

```
require "test_notifier/autotest"
```

If do not have notify thing installed, may need libnotify-bin

```
sudo apt-get install libnotify-bin
```

So run autospec, and the test result should be shown on right top of screen.



Nice, isn't it:)

tags: test_notifier, rails, test

Comments (0)

Company book published (read Chinese version)

2010-01-14

Company book published

北京，2010年1月9日 — 外语教学与研究出版社（FLTRP）和EQ英语日前共同推出了一本具有革命性意义的雅思备考教材——《31种雅思口语高分必背公式》，该书适用于全国准备参加雅思考试的考生。

《31种雅思口语高分必背公式》一书中所提到的雅思备考学习方法具有革命性的创新和突破，并且正在受到包括中国著名的海外学习教师徐小平和杨继等在内的整个英语培训界的认可和好评。“我读了这本书后发现，该书中给出的高分公式涵盖了所有的雅思口语考类型，这能够有效地帮助学生学习如何运用英语答题逻辑来组织自己的观点。而且它还能大大提高学习的效率。如果你掌握了这31种高分公式，你的雅思口语分数在1个月内可以提高1分”，徐小平对记者说。

“这本书是英语口语培训的一大突破，” 斯坦福大学英语语言培训主任、菲尔·赫巴德博士说，“学生通过该书将学会如何用西方人的思维方式回答问题。学生从书中获取的知识和帮助越多、成绩越优秀，也就越有可能被斯坦福这样的大学录取，并在学业上取得更大的成功。”

《31种雅思口语高分必背公式》能够指导学生如何回答所有雅思口语考试题型，无论遇到什么样的话题，都可以使用书中的31种高分公式来应对。“高分公式”包括以下几个方面：“答题思路”、教会你“说什么”，以及“高分语言点”、教会你“如何说”。

据悉，“31种高分公式”已经开始在EQ英语在线学习平台上进行应用，加上EQ英语1对1雅思外教的辅导，能够确保学生提高雅思分数。同时校方还表示，如果学习效果不理想还可全额退款。帕里补充道，该学习方法非常行之有效，能够保证学生在一个月内雅思分数至少提高1分。EQ英语也愿意开此先例，为学生提供口语培训的退款承诺。

本书的出版方外研社是中国最大的外语教育出版机构，这次携手雅思专业培训机构EQ英语的专家团队，可谓强强联手，力求打造出令广大读者满意的精品图书。在刚刚结束的新书发布会上，出版方表示对本书的市场前景充满信心。

《31种雅思口语高分必背公式》的作者是首席运营官和合伙创办人Jonathan Palley、首席执行官和合伙创办人李国栋，主编是EQ英语内容主任Oliver Davies。本书由外语教学与研究出版社出版，在全国各地书店有售。要想获得关于本书的更多信息，请登录www.EQEnglish.com/book。

外研社综合英语分社社长张黎新、本书作者之一李国栋、EQ英语内容主任Oliver Davies在发布会现场

《31种雅思口语高分必备公式》作者之一李国栋在发布会现场

关于EQ英语

EQ英语是一家基于互联网的实时1对1英语学习服务公司，所属公司为美国最大的在线1对1语言指导供应商—— 一德有限公司。

《31种雅思口语高分必背公式》

作者简介

Jonathan Palley，美国人，在斯坦福大学主修物理学和戏剧表演。他始终致力于教育与技术的双重领域，将互联网技术和培训完美结合，创建了世界领先的语音交流平台，使在线语言培训实现了革命化巨变。同时也是31种雅思口语高分公式的创始人之一。

Adrian Li，英国人，曾获剑桥大学经济学学士和斯坦福大学MBA硕士学位。曾供职于世界知名企业摩根大通（JPMorgan）管理层，并在语言学习领域多年不辍研究。2006年来到中国创建了EQ英语，联合多位语言学专家，独创EQ英语逻辑学习法，是31种雅思口语高分公式的创始人之一。

Oliver Davies，英国人，曾获威尔士大学宗教与哲学学士学位，拥有英国伦敦圣三一学院英语教学师资证书。从事英语教学工作长达10年，先后在多个国家的大学和国际学校里任教。多元文化下的工作经历以及对语言教学理念的创新，让他在EQ英语成功地把语言教学搬上了互联网。

tags: company

Comments (0)

my rails test environment (read Chinese version)

2010-01-09

Rspec and TDD are great.
Would like to share my configuration and some experience, enjoy:)
1.generate test doc
use params *--format specdoc* will output test doc in term

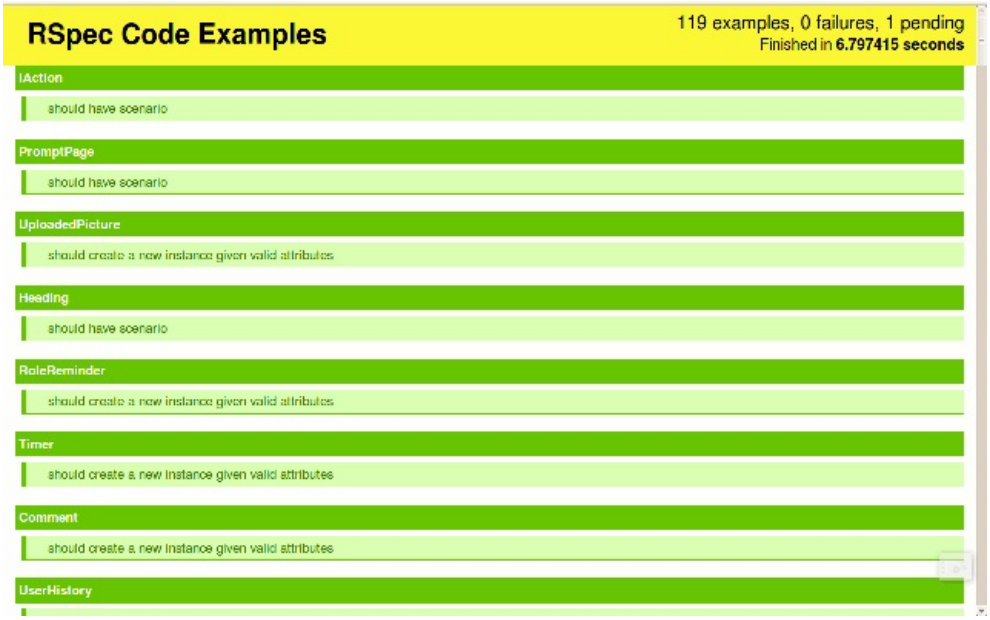
spec spec/models/blog_spec.rb --format specdoc

Also can output test doc for all tests using commend

rake spec:doc

But my favourite way is use *--format html* to output doc into a html file

spec spec/models/blog_spec.rb --format html:test.html



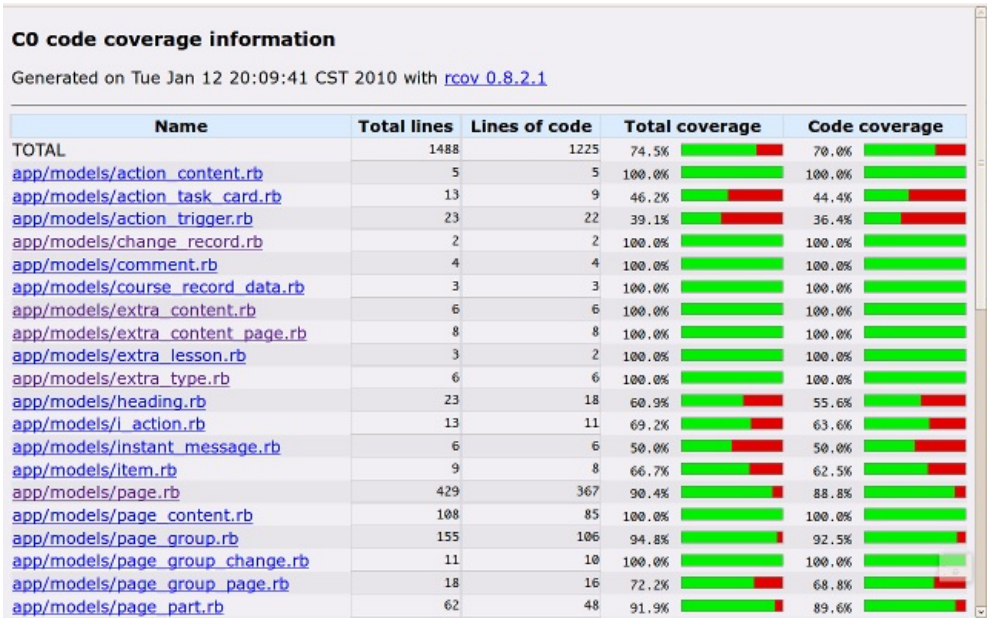
When code coverage and organization are good enough, this doc even can use as documentation for code.

A more convenient way is automate generate this doc after test or rcov, detail see 4 bellow.

2.generate code coverage
rcov

rake spec:rcov

will generate html format code coverage.



C0 code coverage information

Generated on Tue Jan 12 20:09:41 CST 2010 with rcov 0.8.2.1

Name	Total lines	Lines of code	Total coverage	Code coverage
TOTAL	1488	1225	74.5%	70.0%
app/models/action_content.rb	5	5	100.0%	100.0%
app/models/action_task_card.rb	13	9	46.2%	44.4%
app/models/action_trigger.rb	23	22	39.1%	36.4%
app/models/change_record.rb	2	2	100.0%	100.0%
app/models/comment.rb	4	4	100.0%	100.0%
app/models/course_record_data.rb	3	3	100.0%	100.0%
app/models/extra_content.rb	6	6	100.0%	100.0%
app/models/extra_content_page.rb	8	8	100.0%	100.0%
app/models/extra_lesson.rb	3	2	100.0%	100.0%
app/models/extra_type.rb	6	6	100.0%	100.0%
app/models/heading.rb	23	18	60.9%	55.6%
app/models/i_action.rb	13	11	69.2%	63.6%
app/models/instant_message.rb	6	6	50.0%	50.0%
app/models/item.rb	9	8	66.7%	62.5%
app/models/page.rb	429	367	90.4%	88.8%
app/models/page_content.rb	108	85	100.0%	100.0%
app/models/page_group.rb	155	106	94.8%	92.5%
app/models/page_group_change.rb	11	10	100.0%	100.0%
app/models/page_group_page.rb	18	16	72.2%	68.8%
app/models/page_part.rb	62	48	91.9%	89.6%

spec:rcov by default will generate code coverage for all code under app/ and /lib, even there is no test file for them. It is not good in some condition, we can config *spec/rcov.opts* to change this.
Just generate code coverage for models:

--exclude "spec/*,gems/*,app/controllers/*,app/helpers/*,app/sweepers/*,app/workers/*,lib/*"

Code coverage is great way for checking the quality of code, but 100% coverage does not mean your code is perfect. As says in The Rspec Book:
So while low code coverage is a clear indicator that your specs need some work, high coverage does not necessarily indicate that everything is honky-dory.

3.rspec options
most useful options for me

--colour	Show coloured (red/green) output
--format o	Specifies format for output: Profiling of 10 slowest examples
--format html:test.html	Specifies format for output: A nice HTML report
--diff	Show diff of objects that are expected to be equal when they are not
--help	Help

diff need gem diff-lcs

sudo gem install diff-lcs

All options are asstable, for example

spec spec/models/formatable_string_spec.rb --format specdoc --color

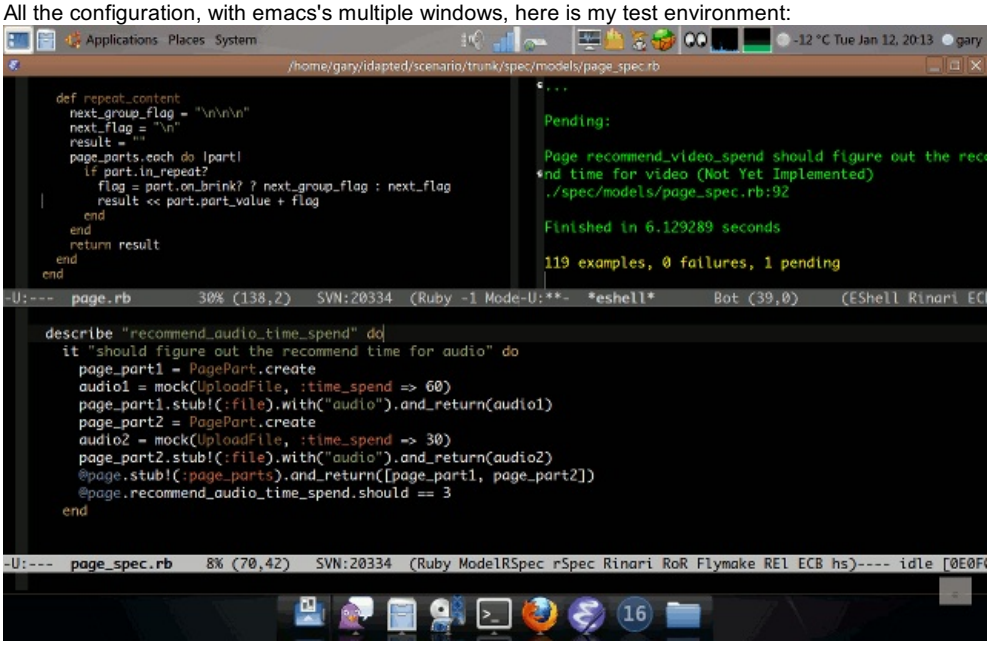
4.opts
Do not need to add options every time when test, just do configuration to *spec/spec.opts*

--colour
--format html:test.html
--diff

Combine the way of "generating test doc" in 1 , add *--format html:test.html* to *spec/spec.opts* , will automate generate test doc after each test or rcov , the doc will output into test.html.

5.Autotest
Autotest is very cleaver, every time you save a test file, autotest will run that test file; and every time you save a library file, autotest will run the corresponding test file. If it sees that the previous failures are now passing, it loads up the entire suite and runs all of the examples again.
When tried autotest on company project, there was a problem. Solution see solve autotest always rerun problem.

All the configuration, with emacs's multiple windows, here is my test environment:



(code top left, autotest top right, test code underneath)

Test is good, but as says in Code Complete:
Even considering the numerous kinds of testing available, testing is only one part of a good software-quality program. High-quality development methods, including minimizing defects in requirements and design, are at least as important.

Do not excessive relay on test.

tags: test, rcov, autotest, rails, rspec

Comments (0)

- **RSS**

  Blog List

Recent Articles

- cloud_tag plugin for rails
- rails code evaluation
- set up autotest to use te...
- Company book published
- my rails test environment
- solve autotest always rer...
- my 2009
- "find in project" in emacs