Hotink C0 Coverage Information - RCov

app/controllers/issues_controller.rb

NAME	TOTAL LINES	LINES OF CODE	TOTAL COVERAGE	CODE COVERAGE
app/controllers/issues_controller.rb	102	75	76.47%	69.33%

Key

```
Code reported as executed by Ruby looks like this...
and this: this line is also marked as covered.
Lines considered as run by rcov, but not reported by Ruby, look like this,
and this: these lines were inferred by rcov (using simple heuristics).
Finally, here's a line marked as not executed.
```

Coverage Details

```
1 class IssuesController < ApplicationController
2
3   skip_before_filter :verify_authenticity_token
4   skip_before_filter :login_required, :only => :upload_pdf
5
6   layout 'hotink'
7
8   # GET /issues
9   def index
10     page = (params[:page] || 1).to_i
11     per_page = (params[:per_page] || 15 ).to_i
12     @issues = @account.issues.paginate( :page=>page, :per_page =>per_page, :order => "date DESC")
13
14     respond_to do |format|
15       format.html
16     end
17   end
18
19   # GET /issues/1
20   def show
21     @issue = @account.issues.find(params[:id])
22
23     respond_to do |format|
24       format.html # show.html.erb
25     end
26   end
27
28   # GET /issues/new
29   def new
30     @issue = @account.issues.create(:date => Time.now)
31
32     respond_to do |format|
33       format.html { redirect_to edit_account_issue_url(@account, @issue ) }
34     end
35   end
36
37   # GET /issues/1/edit
38   def edit
39     @issue = @account.issues.find(params[:id])
40
41     respond_to do |format|
42       format.html # edit.html.erb
43       format.js   # edit.js.rjs
44     end
45   end
46
47   # PUT /issues/1
48   def update
```

```
49     @issue = @account.issues.find(params[:id])
50     #date= @issue.date
51
52     respond_to do |format|
53       if @issue.update_attributes(params[:issue])
54         flash[:notice] = 'Issue saved.'
55         format.html { redirect_to account_issues_url(@account) }
56       else
57         #@issue.date = date
58         format.html { render :action => "edit", :status => :bad_request }
59       end
60     end
61   end
62
63   # DELETE /issues/1
64   def destroy
65     @issue = @account.issues.find(params[:id])
66     @issue.destroy
67
68     respond_to do |format|
69       format.html { redirect_to(account_issues_url(@account)) }
70     end
71   end
72
73   # As a limitation of SWFUpload, this action has to skip authntication an reinstitute its own.
74   def upload_pdf
75     user = User.find_by_single_access_token(params['user_credentials'])
76     if user
77       session[:checkpoint_user_id] = user.id
78       user.reset_single_access_token!
79     end
80     if authorized?
81       @issue = @account.issues.find(params[:id])
82       @issue.swfupload_file = params[:Filedata]
83       flash[:notice] = "PDF uploaded" if @issue.save
84       respond_to do |format|
85           format.html do # After an image upload, reload the page with javascript
86             render :update do |page|
87                 page.replace  'issue', :partial => 'issue_form'
88             end
89           end
90       end
91     else
92       render :text => "unauthorized", :status => 401
93     end
94   end
95
96   private
97
98   def single_access_allowed?
99     action_name == 'upload_pdf'
100   end
101
102 end
```