
TheSortableTree

 maintainability 

 maintainability 

tested with rails 4.1.0rc2 + haml 4

Nested Set + Drag&Drop GUI. Very fast! Best render helper! **2000 nodes/sec.** Ready for rails 4 (RubyGems)
















Dummy Application

- spec/dummy_app
- spec/dummy_app/README.md

Sortable tree. Drag&Drop GUI

Ruby Language

Table of contents

| | |
|---|---|
|  Program |   |
|  Lexical Structure |   |
|  Comments |   |
|  Documentation |   |
|  Whitespace |   |
|  Literals |   |
|  Identifiers |   |

[New Page](#)

Render tree

Ruby Language

Table of contents

| |
|--------------------------|
| <u>Program</u> |
| <u>Lexical Structure</u> |
| <u>Comments</u> |
| <u>Documentation</u> |
| <u>Whitespace</u> |
| <u>Literals</u> |
| <u>Identifiers</u> |

Render Nested Options

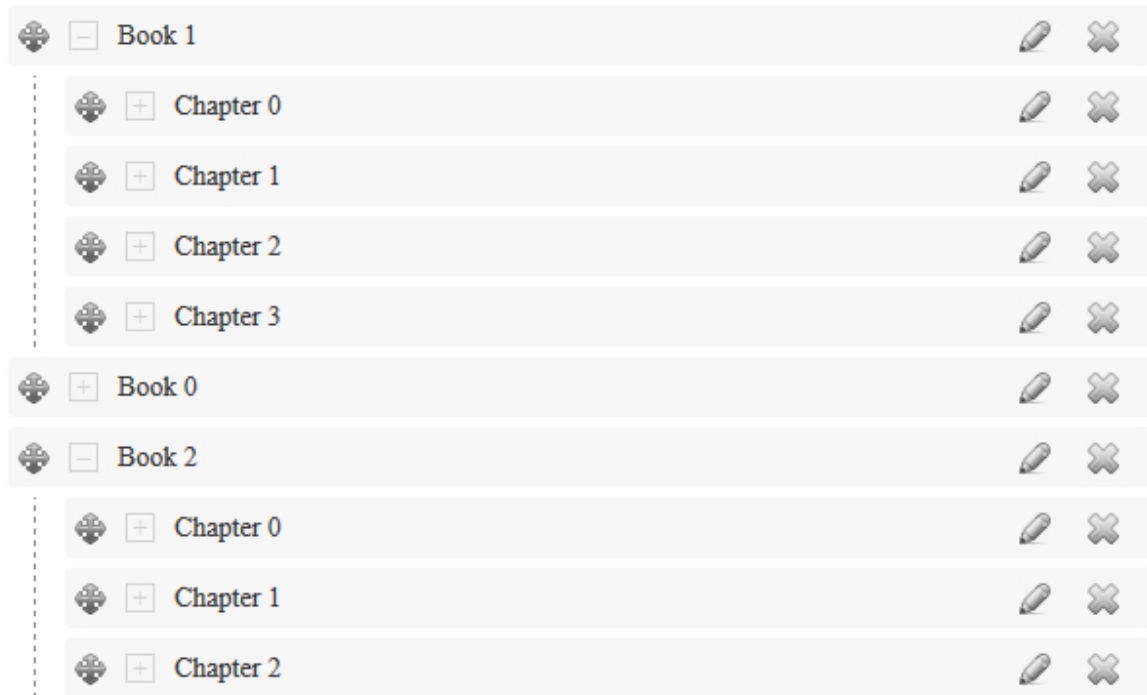
Ruby Language

Table of contents

| |
|-------------------|
| Program |
| Program |
| Lexical Structure |
| Documentation |
| Comments |
| Whitespace |
| Identifiers |

Expandable tree

/pages/expand#TST_Page|22;43;59;64



Keywords

Awesome nested set, Nested set, Ruby, Rails, Nested set view helper, Sortable nested set, Drag&Drop GUI for nested set, View helper for nested set, render tree

Install

Gemfile (Rails 3, Rails 4)

```
1 gem 'awesome_nested_set' # or any similar gem (gem 'nested_set')
2 gem "the_sortable_tree", "~> 2.5.0"
```

Console

```
1 bundle
```

Using

JQuery and JavaScripts `app/assets/javascripts/application.js`

Sortable GUI require JQuery libs

```
1 //= require jquery
2 //= require jquery-ui
3 //= require jquery_ujs
```

Add next JS only for Sortable GUI

```
1 //= require the_sortable_tree/jquery.ui.nestedSortable
2 //= require the_sortable_tree/sortable_ui/base
3
4 // with Turbolinks
5 $ ->
6   TheSortableTree.SortableUI.init()
7
8 // with Turbolinks
9 $(document).on "ready page:load", ->
10   TheSortableTree.SortableUI.init()
```

Stylesheets `app/assets/stylesheets/application.css`

```
1 *= require tree
2 *= require sortable_tree
3 *= require nested_options
4
5 *= the_sortable_tree/sortable_ui/base
```

```
1 ol.the-sortable-tree.the-sortable-tree--list(data={ max_levels: 5,
2   rebuild_url: rebuild_hubs_url })
3   = sortable_tree @hubs
```

Extend your Routes for Sortable GUI

```
1 resources :pages do
2   collection do
3     get :manage
4
5     # required for Sortable GUI server side actions
6     post :rebuild
7   end
8 end
```

manage - just page, where you want render Sortable tree.

Extend your Model

```
1 class Page < ActiveRecord::Base
2   acts_as_nested_set scope: :user
3   include ::TheSortableTree::Scopes
4
5   # any code here
6 end
```

Extend your controller and find your tree

```
1 class PagesController < ApplicationController
2   include ::TheSortableTreeController::Rebuild
3
4   def manage
5     @pages = Page.nested_set.select('id, title, content, parent_id').
      all
6   end
7
8   # any code here
9 end
```

Basic Render Method

```
1 build_server_tree(tree, options)
```

Render Tree

app/views/pages/manage.html.haml

```
1 %ol.tree= just_tree @pages
```

just_tree is just alias of **build_server_tree(tree, type: :tree)**

Render Sortable Tree

```
1 %ol.sortable_tree{ data: { max_levels: 5, rebuild_url:
  rebuild_pages_url } }
2   = sortable_tree @pages
```

sortable_tree is just alias of **build_server_tree(tree, type: :sortable)**

Render Nested Options Tree

```
1 = select_tag :page_id, nested_options(@pages, :selected => Page.last),  
  class: :nested_options
```

nested_options is just alias of **build_server_tree(tree, type: :nested_options)**

build_server_tree options

Client side:

Required params for sortable GUI! Must be defined at root element of tree.

1. **max_levels** - maximum depth of nesting
2. **rebuild_url** - URL to rebuild method on server side

```
1 %<ol.sortable_tree{ data: { max_levels: 3, rebuild_url:  
  rebuild_pages_url } }
```

Server side:

define

```
1 build_server_tree(pages, {:option_1 => :value_1, :option_2 => :value_2  
  })
```

use

```
1 options[:NAME]
```

Options list:

1. **id** - id field of node
2. **title** - title field of node
3. **type** - type of tree [tree|sortable|nested_options]
4. **namespace** - for example: **:admin.** [] - by default
5. **render_module** - your own Render Tree Helper

Rendering runtime params (see code of render helpers):

You can use next options, when rendering run:

1. **level** - level number
2. **root** - root flag [true|false]
3. **klass** - class name

-
4. **has_children** - has children flag [true|false]
 5. **children** - array of children

Customization

Try to run next view generators:

Render helpers for HTML tree generation

```
1 bundle exec rails g the_sortable_tree:views tree
2 bundle exec rails g the_sortable_tree:views sortable
3 bundle exec rails g the_sortable_tree:views nested_options
```

Base Render helper of gem

```
1 bundle exec rails g the_sortable_tree:views helper
```

Assets of gem

```
1 bundle exec rails g the_sortable_tree:views assets
```

I want to know more

1. How to change HTML code of tree?
2. How to create new tree HTML Builder helper?
3. I need to render reversed tree
4. Gem can't correctly define a Name of your Model
5. ChangeLog

Is it fast? Yes, it is!

BANCHMARK:

tree params: 16.000 nodes, 3 levels

- Views: 7999.6ms | ActiveRecord: 79.2ms
- WebInspector full time ~ 9.64s

total: ~ **2000 nodes/sec**

Looking for maintainers

Do you want to be open source contributor? There are some ideas:

Try to create view helpers for:

1. Mongoid NestedSet
2. acts_as_ordered_tree
3. Expand tree via AJAX
4. Comments Tree gem
5. gem Ancestry (???)

I want to try! I need tests!

1. I'm develop gem with test app the_sortable_tree_test_app. You can clone it and see some testcase-pages for gem
2. Sorry, but I have not tests for this gem. Gem is so simple. It's easy to test only with test app.
3. You can write some tests, if your need. I will be happy certainly.
4. No! I know RSpec. I can write tests. But I have not reasons to write tests here.

Acknowledgment

1. mjsarfatti/nestedSortable
2. iconza

this line fot testing with submodules

The MIT License (MIT)

Copyright 2009-2013 Ilya N. Zykin (the-teacher), Mikhail Dieterle (Mik-die), Matthew Clark

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.