Rails GUI Development with Ext JS

martin.rehfeld@glnetworks.de @ RUG-B 10-Jan-2008



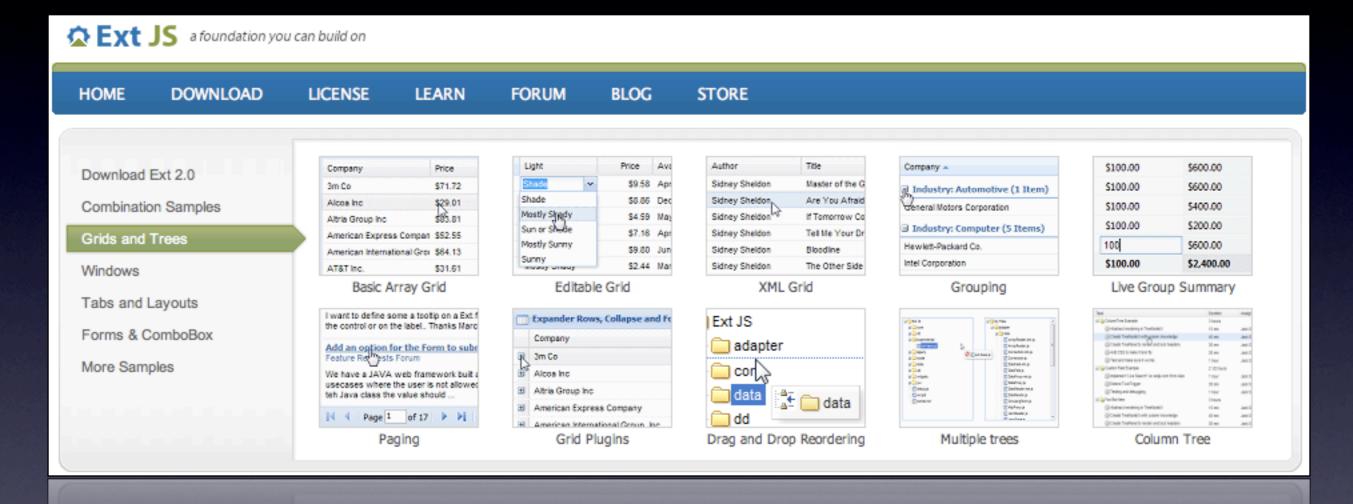
What's wrong with ActionView::Helpers?

- Nothing!
- But...
 - often feels like assembling a spacecraft molecule by molecule
 - which is fine for building the one GUI that no one else has, but get's boring for standard apps

Wouldn't it be nice, if...

- ... you had rich GUI components like
 - Data Grids
 - Tree Views
 - Tabs
 - Toolbars / Menus

Maybe Ext JS is for you



http://www.extjs.com

Ext JS

2.0

Ext 2.0 Final Released

The Ext team is proud to announce that the official release of Ext v2.0 is available for download. This new version of the Ext framework is the culmination of many long hours of work and dedication by the Ext Core team as well as our community of testers and

many long hours of work and dedication by the Ext

- JavaScript Framework
- Dual-License: LGPL + Commercial
 - rich GUI components
 - interface to JSON or XML based data sources (~ web services)
 - keeps state in session, i.e. cookies
 - cross-browser cross-browser



Segregation of Duties

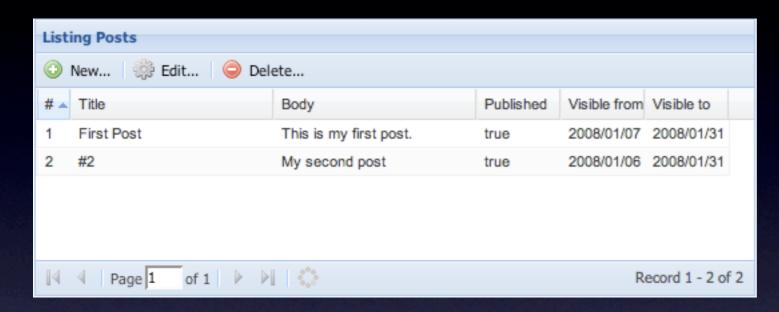


- Model: business as usual
- Controller: REST web service providing special :ext_json format, handle page flow / redirects
- View: provide layout,
 DOM structure and include JavaScript



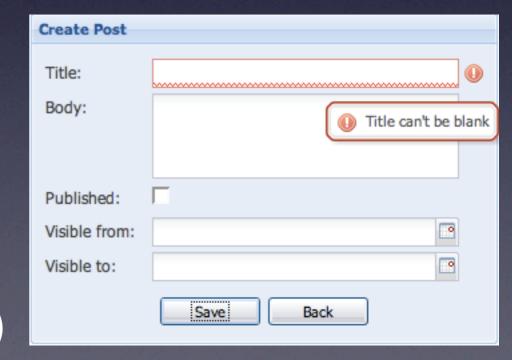
- program GUI and interactions in JavaScript
- use Rails backend to load and persist data as needed

Test Drive: Scaffolding



Data Grid with toolbar and pagination

Forms with validation (client side + server side)



Demo

Ext Scaffold Generator

- Create Rails apprails ext_demo
- Download & extract Ext to public/ext
 http://extjs.com/deploy/ext-2.0.zip
- Install ext_scaffold plugin: script/plugin install http://rug-b.rubyforge.org/svn/ext_scaffold
- Generate Demo resource

```
script/generate ext_scaffold Post title:string body:text
published:boolean visible_from:datetime visible_to:date;
rake db:migrate
```

• Enjoy :-)
script/server -> http://localhost:3000/posts

Code

Model

Nothing special :-)

```
class Post < ActiveRecord::Base
  validates_presence_of :title, :message => "Title can't be blank"
end
```

- Validation errors will be presented in forms just as Ext's own (client side) validation failures
- Ext Scaffold will return JSON error structure with per-field messages

Controller

Support ext_json format -> render :json

```
def index
  respond_to do IformatI
    format.html # index.html.erb (no data required)
    format.ext_json { render :json => Post.find(:all).to_ext_json }
  end
end
def create
  @post = Post.new(params[:post])
  respond_to do IformatI
    if @post.save
      flash[:notice] = 'Post was successfully created.'
      format.ext_json { render(:update) {| page| page.redirect_to posts_url } }
    else
      format.ext_json { render :json => @post.to_ext_json(:success => false) }
    end
  end
end
```

#to_ext_json?

- Implemented as extension to Array and ActiveRecord::Base respectively
 - Array form

Single (valid) record

View

Supported by (some tiny) helpers

```
<% javascript_tag do -%>
  var post_form_items = [
     <%= ext_field(:field_label => 'Title', :name => 'post[title]',
                      :xtype => :text_field) %>,
     <%= ext_field(:field_label => 'Body', :name => 'post[body]',
                      :xtype => :text_area) %>,
     <%= ext_field(:field_label => 'Published', :name => 'post[published]',
                      :xtype => :check_box) %>,
     <%= ext_field(:field_label => 'Visible from', :name => 'post[visible_from]',
                      :xtype => :datetime_select) %>,
     <%= ext_field(:field_label => 'Visible to', :name => 'post[visible_to]',
                      :xtype => :date_select) %>
  ];
<% end -%>

<p
                     :element => 'post-form',
                     :form_items => :post_form_items,
                     :mode => :new %>
<div id="post-form"></div>
```

Resulting "HTML"

```
<script type="text/javascript">
 var post_form_items = [
   { fieldLabel: 'Title', xtype: 'textfield',
                                                                name: 'post[title]'},
   { fieldLabel: 'Body', xtype: 'textarea',
                                                                name: 'post[body]'},
   { fieldLabel: 'Published', xtype: 'checkbox', inputValue: '1', name: 'post[published]'},
                           { xtype: 'hidden', value: '0', name: 'post[published]' },
 ];
 Ext.onReady(function(){
   var panel = new Ext.FormPanel({
       url:
                  '/posts',
       title: 'Create Post',
       renderTo: 'post-form',
       baseParams: {authenticity_token: 'my_secret'},
       items:
                  post_form_items,
                   [ { text: 'Save', type: 'submit',
       buttons:
                       handler: function(){
                         panel.form.submit({url: '/posts?format=ext_json',
                                            waitMsg:'Saving...'}); }},
                     { text: 'Back', handler: function(){ window.location.href = '/posts'; } }
   });
 });
</script>
<div id="post-form"></div>
```

Conclusion

- Using Ext gives
 - a very slim Rails backend,
 just as we like it :-)
 - lots of JavaScript in the view :-(
 - then again:
 loads of functionality "for free" and less involvement of the backend in user interactions
 :-))))

Q & A

Martin Rehfeld martin.rehfeld@glnetworks.de http://inside.glnetworks.de