Earth GUI Plug-in Development Documentation

Contents

List of Figures

1 Introduction

Plugins are already existent in the Earth's daemon. However, this project had identified that this is a potential improvement towards the front-end. Thus, the concept of plugins for Earth's daemon was ported to the front-end. However, the concept needed some changes in order to make them work.

Instead of following the daemon plugins' standards, the plugins for the front-end, dubbed as GUI plugins, were developed based on the Rails' plugin framework. The reason is to allow an easy to access portal to create, update and maintain these GUI plugins. Another reason why the Rails plugin framework was chosen is because Earth was built using the Rails web-application framework.

The following section will present how do these plugins work in Models, Controllers and Views. Then, the rest of the sections will present how the plugins can be created and deployed.

2 Investigation

It was observed that it is better to make the front-end as plug-able as the daemon, so that users can easily add or remove GUI related functionalities. The problem with GUI plugins is that these plugins may not be as simple as the plugins created for the daemon. Their content, or rather codes, can be scattered over the three sections of the Rails development paradigm, which are the Models, Controllers and Views. However, the Rails plugin framework solves the problem.

A Rails plugin is either an extension or a modification of the core framework. Such plugins can do almost anything that a Rails application can, plus a little more. The provider plugin `generator` script is able to copy the respective files into their respective `app` sub-directories. In addition, the `generator` script can process an `ERB` file and have it copied into the `migration` folder through the use of the following migration templates:

- **Models** Put a model in the plugins `lib` folder and use `generator` to copy it to `app/models` folder.
- **View Helpers** A helper method can be included, or mixed, into the rest of the application. (This is also known as "mixin".)
- **Controllers** Copies a controller into the `app/controllers` directory.
- **rake Tasks** Creates a `rake` file into the tasks folder.
- **Images, Stylesheets, Javascripts** The `generator` copies these into the public directory.
- **Test assertions** To be mixed-in to the tests cases.
- **Unit and Functional tests** Can be generated like controllers.

From the investigation above, it is clear that the Rails plugin framework can be used to create Earth GUI plugins.

3 Preparation

To create a GUI plugin, one first needs to know what feature or functionality one wants to add, and how the result should be presented on the front-end. To do so, one needs to create necessary files in a directory of their preference. Then, one will need to have these plugin files copied into the `plugin` directory. Finally, use the `generator` script to process and copy the files into the Earth's `app` directory. The following sections will discuss the mentioned steps in detailed.

4 Implementation

The GUI plugin is implemented using the Rails plugin framework. However, some code changes were needed in the original View codes. This is to allow addition functionality, such as more tag or search field, to be added. The Rails plugin framework is used for the rest of the MVC model. The structure of the GUI plugin framework is shown Figure 1.

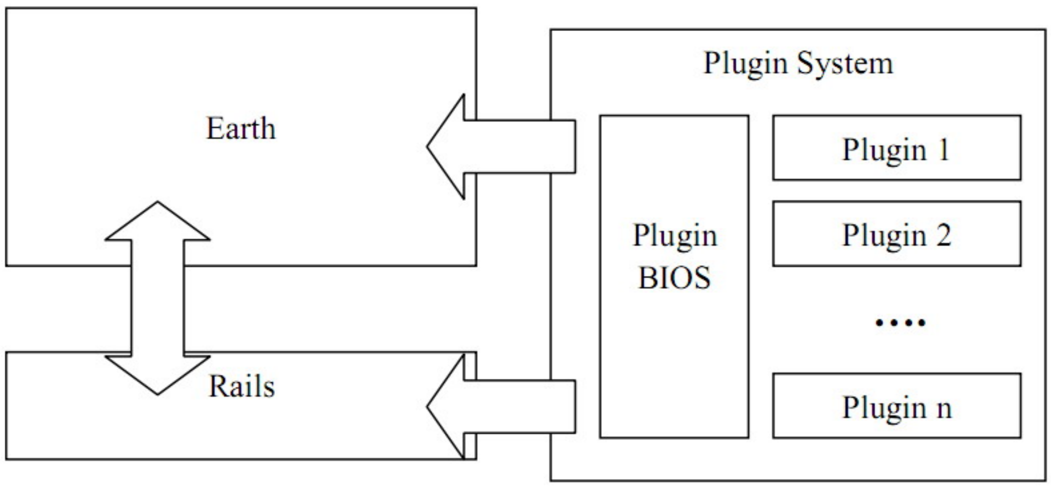


Figure 1: The structure of the GUI plugin framework.

Rails plugin framework can be used as an extension add more functionality development in Earth. It is helpful for development of Earth. In the Plugin System, the Plugin BIOS was created. It is an intermediary communication layer between the plugins, Earth and Rails. Currently, this layer is only a simple communication layer. It is expected in the future this layer will be extended to include dependance checks and have features to have sub-classing ability. Each plugin, a configuration file named "plugin_cfg" is created. This file contains the information about the plugin to be used as a manifest of the plugin package.

5 Instruction

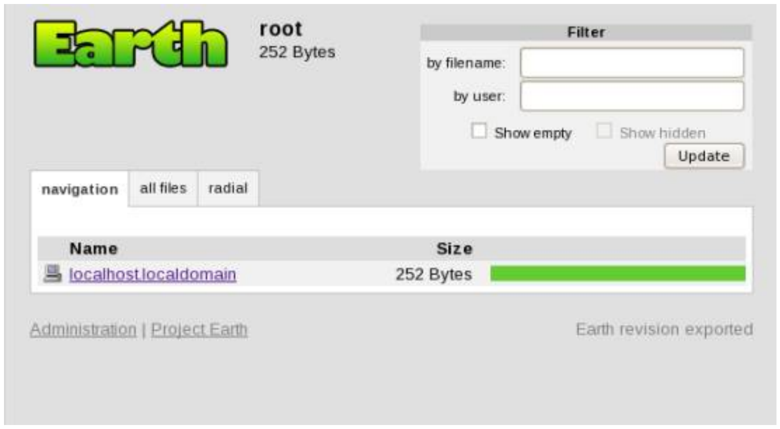
In this section, the steps of how to develop, install and uninstall a GUI plugin is presented. To ease explanation, a blank plugin, called "mr_bogus", is used as an example. To begin, Figure 2 shows the original Earth front-end, before mr_bogus is installed.

5.1 Development

As mentioned before, the `generate` script is first executed to create some basic necessary structures and files. To do so, the following command can be used:

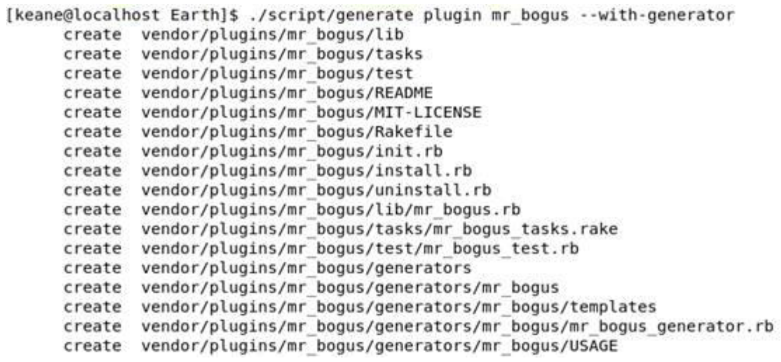
```
./script/generate plugin mr_bogus --with generator
```

A blank plugin in Rails have been created after the previous command was executed successfully. At this stage, one can safely edit the affected files to have the final results displayed. Views can be added by creating new directories and files in the `./vendor/plugins/mr_bogus/generators/mr_bogus/templates` directory. For example, the action controller needs the directory called "controllers". Finally, update the file called "mr_bogus_generator.rb", which can be executed to create particular file wherever one wants. More instructions can be found in this web-site `http://wiki.rubyonrails.com/rails/pages/HowTosPlugins`.



Name	Size
localhost.localdomain	252 Bytes

Figure 2: Original Earth front-end.



```
[keane@localhost Earth]$ ./script/generate plugin mr_bogus --with-generator
      create  vendor/plugins/mr_bogus/lib
      create  vendor/plugins/mr_bogus/tasks
      create  vendor/plugins/mr_bogus/test
      create  vendor/plugins/mr_bogus/README
      create  vendor/plugins/mr_bogus/MIT-LICENSE
      create  vendor/plugins/mr_bogus/Rakefile
      create  vendor/plugins/mr_bogus/init.rb
      create  vendor/plugins/mr_bogus/install.rb
      create  vendor/plugins/mr_bogus/uninstall.rb
      create  vendor/plugins/mr_bogus/lib/mr_bogus.rb
      create  vendor/plugins/mr_bogus/tasks/mr_bogus_tasks.rake
      create  vendor/plugins/mr_bogus/test/mr_bogus_test.rb
      create  vendor/plugins/mr_bogus/generators
      create  vendor/plugins/mr_bogus/generators/mr_bogus
      create  vendor/plugins/mr_bogus/generators/mr_bogus/templates
      create  vendor/plugins/mr_bogus/generators/mr_bogus/mr_bogus_generator.rb
      create  vendor/plugins/mr_bogus/generators/mr_bogus/USAGE
```

Figure 3: Running the `generate` script to create mr_bogus plugin.

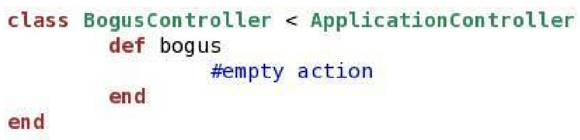

```
class BogusController < ApplicationController
        def bogus
                #empty action
        end
end
```

Figure 4: The bogus_controller.rb file.

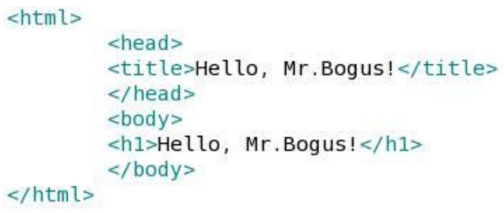

```
<html>
        <head>
        <title>Hello, Mr.Bogus!</title>
        </head>
        <body>
        <h1>Hello, Mr.Bogus!</h1>
        </body>
</html>
```

Figure 5: The bogus.rhtml file.

Here an action controller and a webpage were created for views:

```
./vendor/plugins/mr_bogus/generators/mr_bogus/templates/controllers/bogus_controller.rb
./vendor/plugins/mr_bogus/generators/mr_bogus/templates/views/bogus.rhtml
```

Figure 4 shows the content of the bogus_controller.rb file, and Figure 5 shows the content of the bogus.rhtml file.

The file named mr_bogus_generator.rb would be modified for preparation of moving files to particular folders, as shown in Figure 6.

Once done, the following command can be executed to have the plugin implemented into Rails:

```
./script/generator mr_bogus bogus
```

To test the plugin, simply point the web-browser of choice to the following URL:

```
http://localhost:3000/bogus/bogus
```

However, the plugin should be implemented into the existing Earth system instead. To do so, a configuration file has to be created. The file is named "plugin_cfg" in the ./vendor/plugins/mr_bogus directory. Figure 7 shows the content of the configuration file.

Now, point the web-browser to the homepage of Earth. The result is shown in Figure 8. Figure 9 shows the implementation of the plugin.

So far, the plugin had included a new tab in Earth. However, to add a new text field in the filter section, the following can statements can be added into the configuration file:

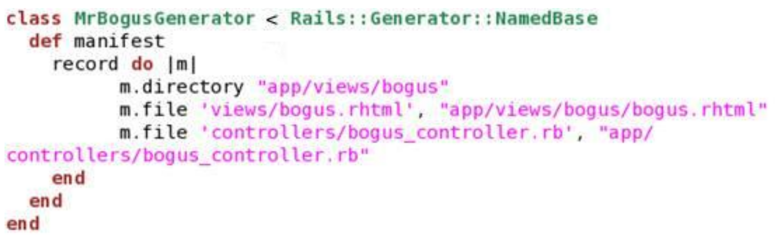
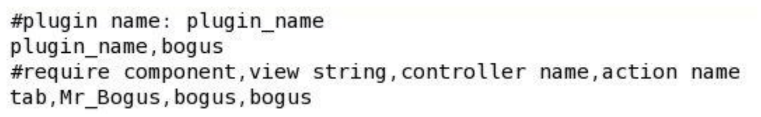
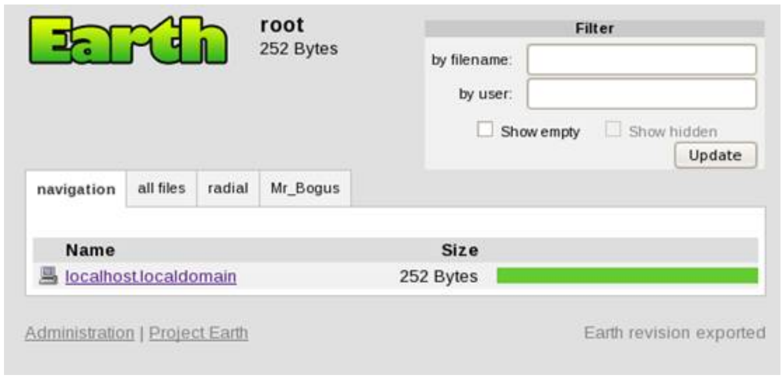

```
class MrBogusGenerator < Rails::Generator::NamedBase
  def manifest
    record do |m|
          m.directory "app/views/bogus"
          m.file 'views/bogus.rhtml', "app/views/bogus/bogus.rhtml"
          m.file 'controllers/bogus_controller.rb', "app/
controllers/bogus_controller.rb"
    end
  end
end
```

Figure 6: The bogus_controller.rb file.



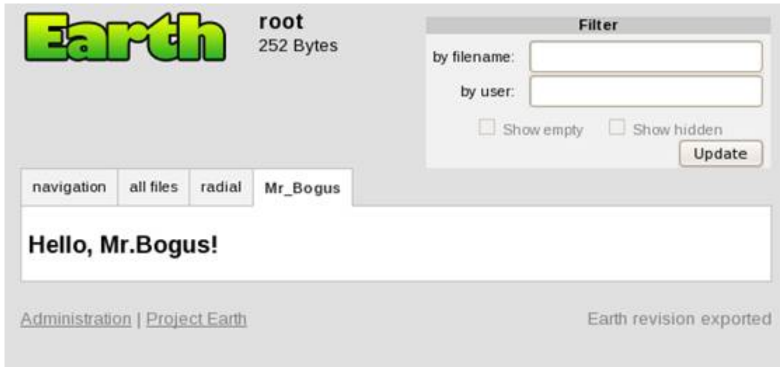
```
#plugin name: plugin_name
plugin_name,bogus
#require component,view string,controller name,action name
tab,Mr_Bogus,bogus,bogus
```

Figure 7: The `plugin_cfg` file.



Name	Size
localhost.localdomain	252 Bytes

Figure 8: `mr_bogus` implemented in Earth.



Hello, Mr.Bogus!

Figure 9: The `mr_bogus` tab.

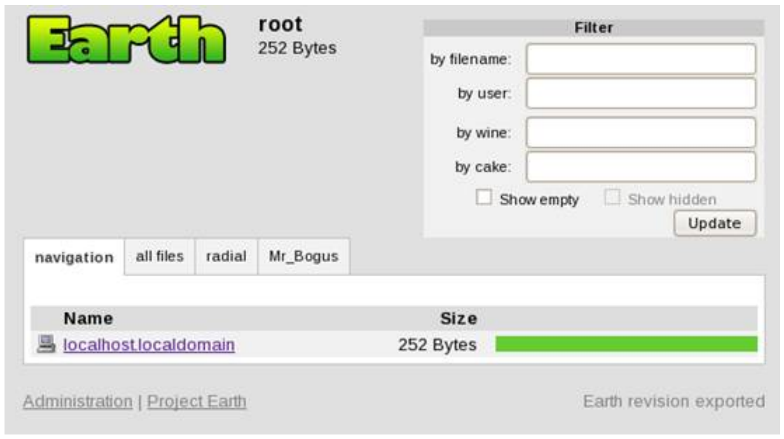


Figure 10: New text fields.

```
#require componenet, view string, parameter name
field_tag, by wine, wine
field_tag, by cake, cake
```

Figure 10 shows the refreshed the homepage of Earth.

The plugin can now be tested and debugged. Finally, a set of installation and uninstallation scripts can be created from the following steps:

1. Find files and folders created.
2. Modify the installation in root of plugin to have the appropriate files copied to their appropriate destination directories.
3. Build the plugin as a package including the installation file.

The following is the content of installation of “mr_bogus” named “install.rb”:

```
require 'fileutils'
plugin_name = ARGV[1] #mr_bogus
earth_root = ARGV[0]
puts "Creating the plugin"
system "ruby #{earth_root}/script/generate plugin #{plugin_name} --with-generator"
RAILS_ROOT = earth_root
FileUtils.cp File.join(File.dirname(__FILE__), 'init.rb'),File.join
(RAILS_ROOT, 'vendor','plugins',plugin_name,'init.rb')
FileUtils.cp File.join(File.dirname(__FILE__), 'install.rb'),File.join
(RAILS_ROOT, 'vendor','plugins',plugin_name,'install.rb')
FileUtils.cp File.join(File.dirname(__FILE__), 'uninstall.rb'),File.join
(RAILS_ROOT, 'vendor','plugins',plugin_name,'uninstall.rb')
FileUtils.cp File.join(File.dirname(__FILE__), 'plugin_cfg'),File.join
(RAILS_ROOT, 'vendor','plugins',plugin_name,'plugin_cfg')
FileUtils.cp File.join(File.dirname(__FILE__),'lib','mr_bogus.rb'),File.join
(RAILS_ROOT, 'vendor','plugins',plugin_name,'lib','mr_bogus.rb')
FileUtils.cp File.join(File.dirname(__FILE__),'generators',plugin_name,
'mr_bogus_generator.rb'),File.join(RAILS_ROOT,'vendor','plugins',plugin_name,
'generators',plugin_name,'mr_bogus_generator.rb')
Dir.mkdir("#{RAILS_ROOT}/vendor/plugins/mr_bogus/generators/mr_bogus
/templates/controllers") unless File.directory?("#{RAILS_ROOT}/vendor/plugins
/mr_bogus/generators/mr_bogus/templates/controllers")
Dir.mkdir("#{RAILS_ROOT}/vendor/plugins/mr_bogus/generators/mr_bogus
/templates/views") unless File.directory?("#{RAILS_ROOT}/vendor/plugins/mr_bogus/generators
/mr_bogus/templates/views")
FileUtils.cp File.join(File.dirname(__FILE__),'generators',plugin_name,'templates',
'controllers','bogus_controller.rb'),File.join(RAILS\_ROOT,'vendor','plugins',
plugin_name,'generators',plugin_name,'templates','controllers','bogus_controller.rb')
FileUtils.cp File.join(File.dirname(__FILE__),'generators',plugin_name,'templates','views',
'bogus.rhtml'),File.join(RAILS_ROOT,'vendor','plugins',plugin_name,'generators',
plugin_name,'templates','views','bogus.rhtml')
```

An uninstallation file named “uninstall.rb” for removing the plugin created. Here is an example:

```
require 'fileutils'
plugin_name = ARGV[1] #mr_bogus
earth_root = ARGV[0]
puts "Uninstalling the generator"
system "ruby #{earth_root}/script/destroy #{plugin_name} #{plugin_name}"
puts "Uninstalling the plugin"
system "ruby #{earth_root}/script/destroy plugin #{plugin_name} --with-generator"
system "rm -r #{earth_root}/vendor/plugins/#{plugin_name}"
```

5.2 Installation

To install and activate the plugin, simply execute the following commands:

```
ruby install.rb <root_earth> <plugin_name>
ruby <root_earth>/script/generate <plugin_name> <plugin_name>
```

5.3 Uninstallation

To uninstall, execute the following command:

```
ruby uninstall.rb <root_earth> <plugin_name>
```