# Creating gem to Mokio

In this tutorial you will create simple gem to Mokio CMS

# 1. Setup

## 1.1 Create your own gemset and install Rails 4.1.1

```
rvm use <your ruby version>
rvm gemset create <your gemset name>
gem install rails -v 4.1.1
```

`1.2 Create database schema`

Name it i.e. mokio_products

## 1.2 Create your plugin with the command:

`rails plugin new mokio_products –mountable -d <database>`
(<database> options:
`mysql/oracle/postgresql/sqlite3/frontbase/ibm_db/sqlserver/jdbcmysql/jdbcsqlite3/jdbcpostgresql/jdbc`)

I.e. `rails plugin new mokio_products –mountable -d mysql`

It will create standard folders hierarchy.

## 1.3 Configure database connection

In your test/dummy/config/database.yml change configuration

```
development:
  adapter: mysql2
  encoding: utf8
  database: mokio_products
  pool: 5
  username: root
  password: secret_password
  socket: /var/run/mysqld/mysqld.sock
```

## 1.4 Install mokio

First require Mokio in your Gemfile

```
gem 'mokio', :git => "https://github.com/versoft/mokio.git"
```

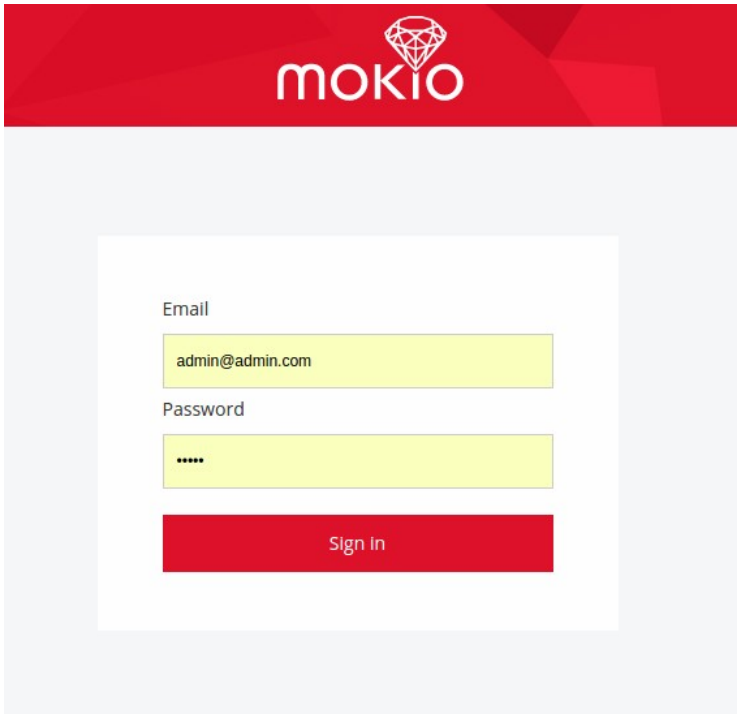
Then run:
`bundle install`
Go to mokio_products/test/dummy and run:
`rake mokio:install` (it will create tables and necessary data)

Open test/dummy/config/routes.rb and edit it to look like this:

```
Rails.application.routes.draw do
  mount Mokio::Engine => "/backend"
end
```

Start your server and go to http://localhost:3000/backend

Welcome!




Your login is: admin@admin.com and password: admin

# 2. Creating model

## 2.1 Migrations (see: http://guides.rubyonrails.org/migrations.html)

We will create table mokio_products with 2 columns: name and part_number

```
rails generate migration CreateMokioProducts name:string part_number:string
active:boolean

Run:
rake db:migrate
```

## 2.2 Model class

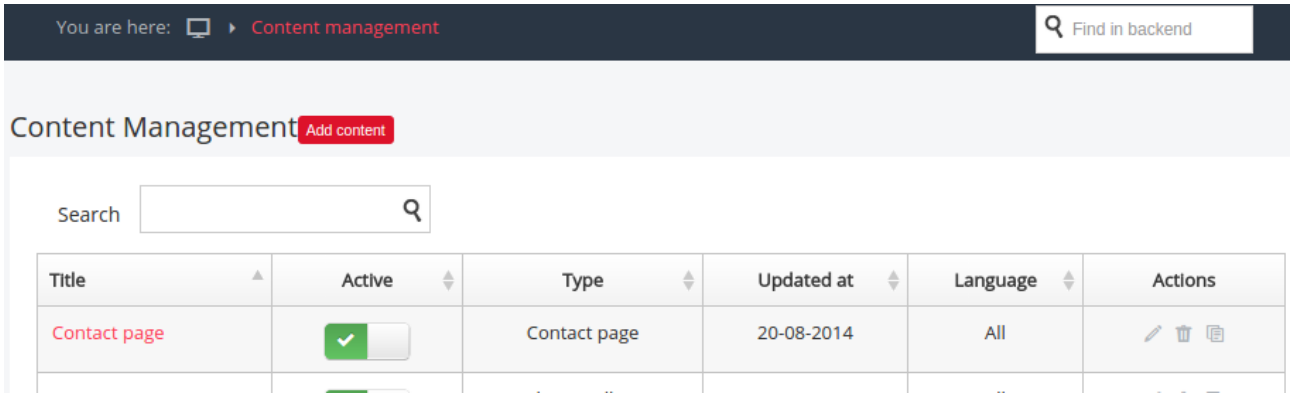
Create your model class in `app/model/mokio/products.rb` as following:

```
module Mokio
  class Products < ActiveRecord::Base
    include Mokio::Concerns::Models::Common
  end
end
```

Common module add's methods commonly used in our models. I.e. standard scopes, methods, validators. In your model you must define columns editable and deletable or define methods with these names. You also have to override `self.columns_for_table` method which defines columns displayed in index action.

```
module Mokio
  class Products < ActiveRecord::Base
    include Mokio::Concerns::Models::Common
  end

  def editable
    true
  end

  def deletable
    true
  end

  def self.columns_for_table
    %w(name active part_number)
  end

end
```

If you set these methods to true, you will see edit and delete buttons in your model record actions.




# 3. Creating controllers

Add app/controlers/mokio/products_controller.rb
Extend Mokio::CommonController to add standard actions to your controller (index, new, edit, create, update, destroy) and some other methods

```
module Mokio
  class ProductsController < Mokio::CommonController
    include ActiveSupport::Concern
```

```
  end
end
```

# 4. Creating views

As you can see in backend there is no button to your Products model. Mokio CMS has mechanism to extend standard views.

## 4.1 Routes

Change your app/config/routes.rb:

```
Mokio::Engine.routes.draw do
  resources :products
end
```

## 4.2 Locales

Create your locale file `i.e app/config/locales/en.yml:`

```
en:
  products:
    products: Awesome products
    index_title: Products
    index_add_new: Add new product
```

## 4.3 Extend Mokio views

### 4.3.1 Template

Create file with your button template (slim, haml, erb are allowed)

app/views/my_btn.html.slim:

```
li
  a href=products_path
    span.icon16.entypo-icon-images
    = bt("products", Mokio::Product)
```

Info: `bt("products", Mokio::Product)` is equal to `t("products.products ")`

### 4.3.2 Config

Then create `app/config/views.yml:`

```
-
  original_view: "mokio/layout/sidebar"
  override_path: "my_btn"
  element_path: "#sidebar-ul"
  type: "css"
```

You can use type: "xpath" to use Nokogiri selectors syntax in element path.

-
original_view: "mokio/layout/sidebar"
override_path: "overrides/sidebar_account"
html_element_id: "#sidebar-ul"

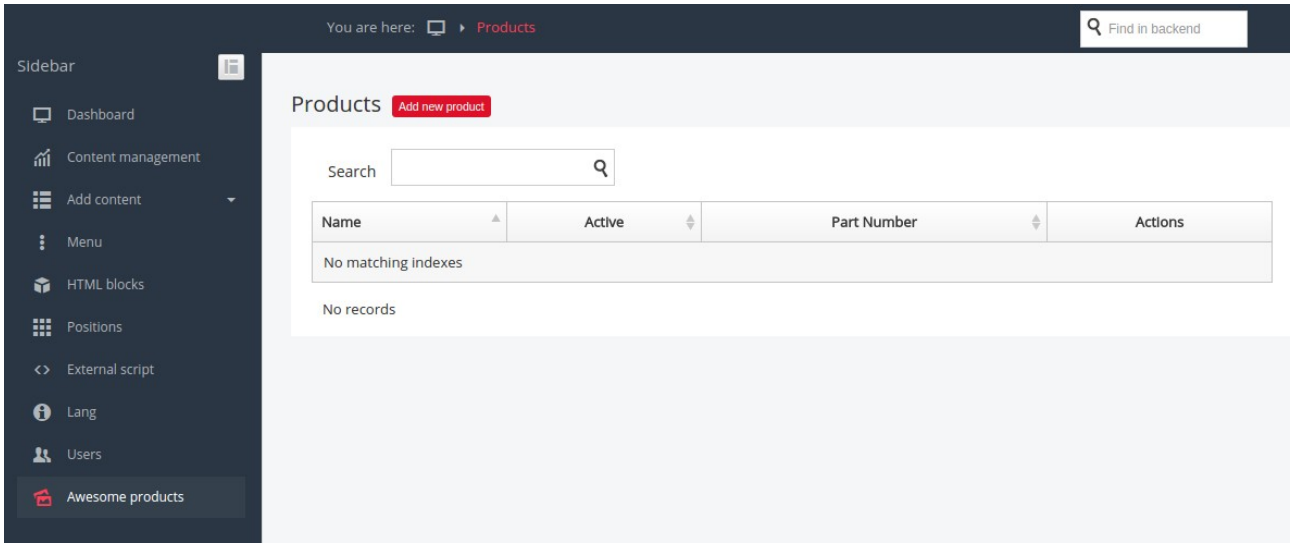
**REMEMBER:**

If you want to override partial view, the override_path should be:
**"path/to/view/my_partial"**

but the file should have a name: "path/to/view/_my_partial"

Remember to restart your server after changing views.yml file

Start server and go to localhost:3000/backend
Now you will see button to your products and working index view




## 4.4 Handle create action

To handle [Add new product] button you have to create partial template app/views/mokio/_form.html.slim

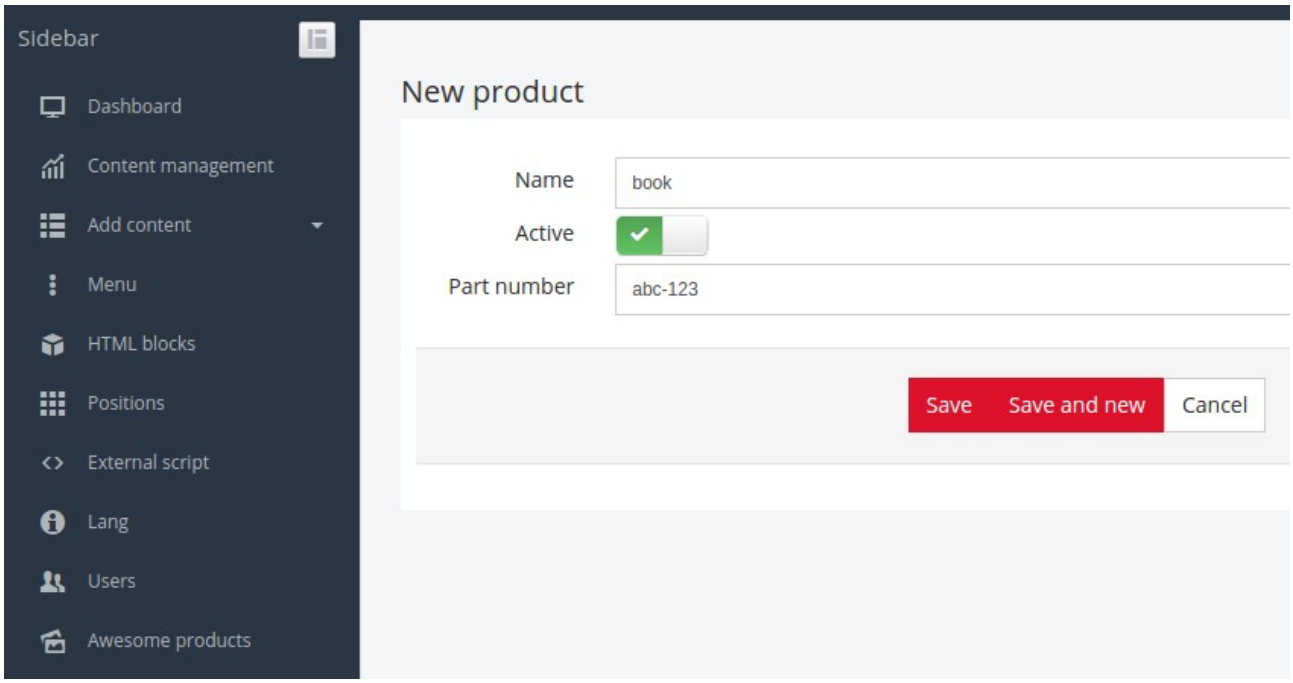
```
- cache [:form, obj] do
  = f.input :name
  = f.input :active, :wrapper => :active_checkbox, disabled: !
obj.display_editable_field?('active')
  = f.input :part_number
```

Add some translations:

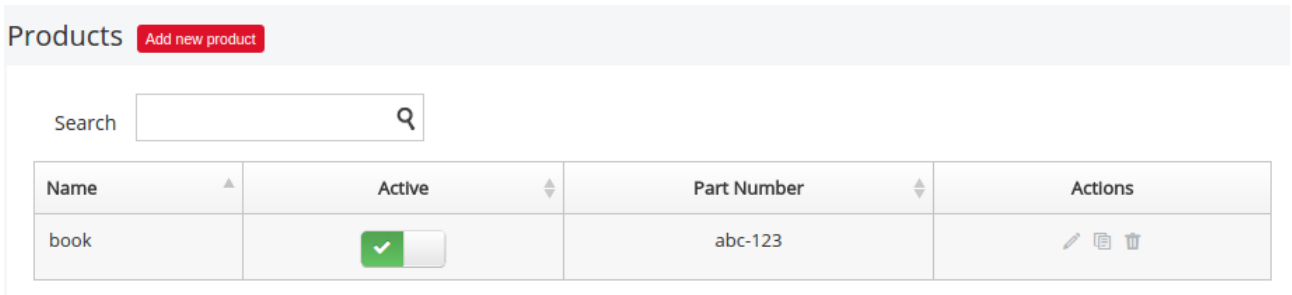
```
en:
  backend:
    breadcrumbs:
      products: Products
      products_new: Creating new product
```

```
  products:
    products: Awesome products
    index_title: Products
    index_add_new: Add new product
    new_title: New product
  activerecord:
    attributes:
      mokio/product:
        name: Name
        active: Active
        part_number: Part number
```

Now you can create your product.




After save you will see it on product list.



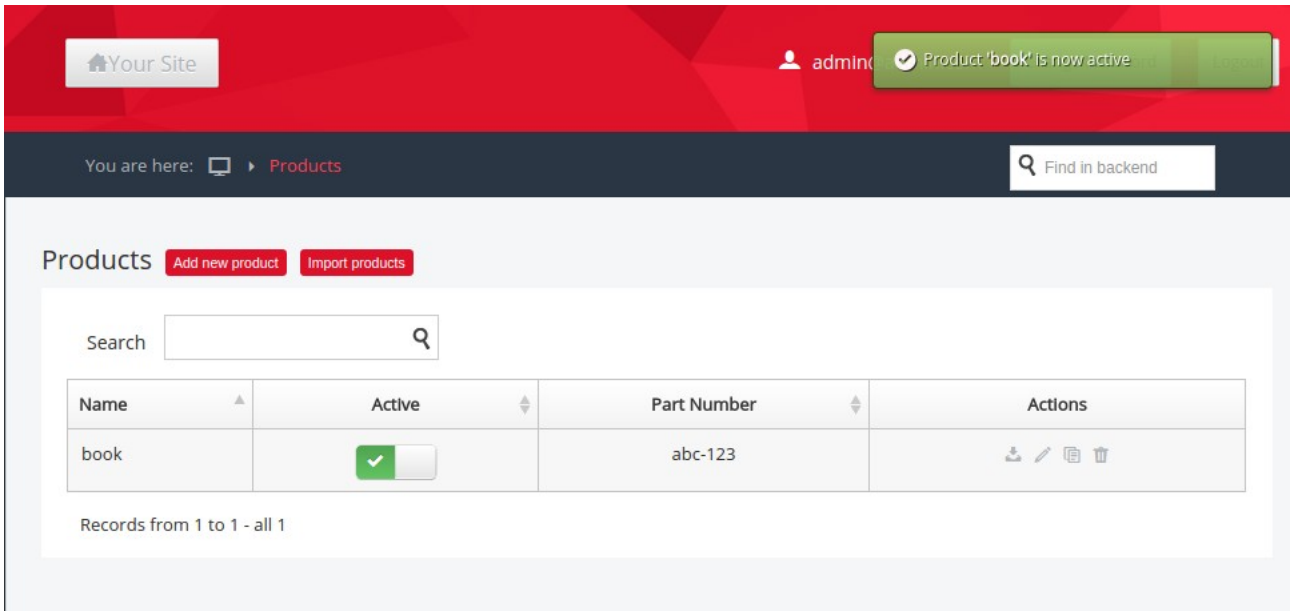

To handle active button add to en.yml

```
en:
  products:
    update_active_true: "Product <strong>'%{title}'</strong> is now active"
    update_active_false: "Product <strong>'%{title}'</strong> is now
inactive"
```

Add to routes.rb:

```
resources :products do
```

```
  member do
    post :update_active
  end
end
```




## 4.5 Additional index buttons

Mokio has mechanism to simple add own action buttons. Lets add [Import products] button next to [Add new product]

### 4.5.1 Routes

Change routes.rb:

```
resources :products do
  collection do
    get '/import', to: 'products#import'
  end
end
```

### 4.5.2 View

Create app/views/mokio/products/_import_btn.html.slim

```
= btn_new( bt("index_import", @obj_class), import_products_path)
```

### 4.5.3 Translations

Add:

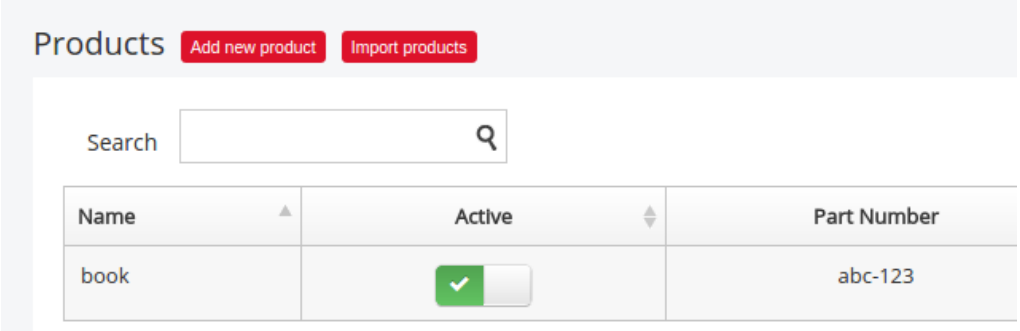
```
en:
  products:
    index_import: Import products
```

### 4.5.4 Controller

Add import action and override method `additional_index_buttons`.

```
def import
  #some logic
end

def additional_index_buttons
  "mokio/products/import_btn"
end
```

Now you can see our [Import products] button.




## 4.6 Additional action buttons

Now lets add our own button in actions i.e. download button.

### 4.5.1 Routes

Change routes.rb:

```
resources :products do
  collection do
    get '/import', to: 'products#import'
    get '/download', to: 'products#download'
  end
end
```

### 4.5.2 View

Create json.slim view with our action button.
app/views/mokio/products/_download_btn.json.slim

```
= link_to import_products_path(:product_id => obj.id) do
  span class="icon12 entypo-icon-download"
```

### 4.5.4 Controller

Override method `additional_action_buttons`.

```
def additional_action_buttons
  "mokio/products/download_btn"
end
```

Now you can see our [download] action button.