**Access Labs Applications on Rails+AngularJs**

**About**

**Local environment**

We chose to install Ruby on Rails via rvm with which dependencies will installed as well.

1. Install rvm

Note: Single-User installations (recommended) - For an isolated install within a user's $HOME, not for root.

$curl -L get.rvm.io | bash -s stable

2. Load RVM

$source ~/.rvm/scripts/rvm

3. Display a list of all "known" rubies

$rvm list known

4. Install a version of Ruby (e.g. 1.9.3)

$rvm install 1.9.3

5. Install Rails with gem

$gem install rails

**An ROR Demo**

1. Creating an app or cloning one.

You can create an app on your own or copy a repository locally by passing the git URL to 'git clone'.

* Creating an app

Rails projects and modules are created by rails commands.

$rails new rordemo

Any other operations related to the project should be done in the local directory.

$cd rordemo

Routes are defined in the file config/routes.rb. This file is created when you first create your Rails application. It comes with a few routes already written and in most cases you'll want to change and/or add to the routes defined in it.

The routing system has to find a pattern match for a URL it's trying to recognize, or a parameters match for a URL it's trying to generate.

$vim config/routes.rb

|  |
| --- |
| scope $lab\_context do  require Rails.root.join('config', 'initializers', 'labs.rb')  get "/", to: 'application#index'  end |

In this file, labs.rb is made accessible from routes.rb and the root route is defined to refer to app/views/application/index.html.erb via $lab\_cotext/

The initializer labs.rb is defined as follow.

$vim config/initializers/labs.rb

|  |
| --- |
| Rails.application.config.before\_initialize do  $lab\_id = 'rordemo'  $lab\_name = 'Ruby on Rails demo'  $lab\_context = "/labs/#{$lab\_id}"  $is\_prod\_gear = ENV['HOME'].eql? "labsprod"  Rails.application.config.assets.prefix = "#{$lab\_context}/assets"  Rails.application.config.action\_controller.relative\_url\_root = $lab\_context  ENV['RAILS\_RELATIVE\_URL\_ROOT'] = $lab\_context  end |

Copy labs\_key.txt to directory <appName>/public and make sure all your css/js files are put in public/labs/<appName>/js(css)

$mkdir -p public/labs/rordemo/js(css)

The app must be chromed to look like the rest of the Customer Portal (See <https://mojo.redhat.com/docs/DOC-951401>).

$vim public/labs/rordemo/js/app.js

|  |
| --- |
| /\*global document, window, angular\*/  (function () {  "use strict";  var private\_functions = {},  public\_funtions = {};  window.chrometwo\_require(['jquery', 'chrome\_lib', '/webassets/avalon/j/lib/angular/1.2.8/angular.min.js'], function (jq, lib) {  var app = window.angular.module('rordemo', []);  app.controller('AppCtrl', function ($scope, $location, $http) {  $scope.info = “My Rails+AngularJs app”;  });  window.angular.bootstrap(document, ['rordemo']);  jq('#main').css('visibility', 'visible');  });  }()); |

Rails will use the layout defined in the file app/views/layouts/application.html.erb as a default for rendering any page.

It contains a title for your web pages.

The stylesheet\_link\_tag and javascript\_include\_tag elements add CSS stylesheets and JavaScript from the Rails asset pipeline. The data-turbolinks-track attribute supports Rails Turbolinks.

The csrf\_meta\_tags element is a view helper that prevents cross-site request forgeries when users input data in forms.

The content from a Rails view is inserted where you place the yield keyword. Without any arguments, yield will render the template of the current controller/action (app/views/application/index.html.erb in this demo).

$vim app/views/layouts/application.html.erb

|  |
| --- |
| <!DOCTYPE html>  <html class="chrometwo bootstrap3 next" xmlns:ng="http://angularjs.org">  <head>  <title><%= $lab\_name %></title>  <link href="/chrome\_themes/umbra/s/chrometwo.css" rel="stylesheet" />  <link href="<%= $lab\_context %>/css/app.css?bust=1" rel="stylesheet" />  <link rel="stylesheet" type="text/css" media="all" href="/chrome\_themes/umbra/s/labs.css">  <script type="text/javascript" src="/webassets/avalon/j/lib/require.js"></script>  <script type="text/javascript" src="<%= $lab\_context %>/js/app.js?bust=1"></script>  <%= csrf\_meta\_tags %>  </head>  <body class="chrometwo">  <div id="chrometwo">  <div id="main" style="visibility: hidden;">  <%= yield %>  <script type="text/javascript">  window.breadcrumbs = [["Labs", "/labs/"], ["<%= $lab\_name %>", "<%= $lab\_context %>"]];  </script>  </div>  </div>  </body>  </html> |

$mkdir -p app/views/application

$vim app/views/application/index.html.erb

|  |
| --- |
| <div ng-controller="AppCtrl">  <h1><%= $lab\_name %></h1>  <div class="row">  <div class="app-block">  <h2 class="title">{{info}}</h2>  </div>  </div> <!-- /row -->  </div> |

* Cloning a demo

Rails+AngularJS demo app quickstart | Access Labs guide

<https://mojo.redhat.com/docs/DOC-955023>

2. Setting up an environment that mimics the Access Labs development space by following steps below:

Step 1:

Install Apache2.

Step 2:

[Create a DDNS Entry](https://mojo.redhat.com/docs/DOC-13143)

Step 3:

Set up a virtual host and some proxy passes so that the JS chroming services work in /etc/httpd/conf/httpd.conf.

Note: make sure ssl has been installed and modules/mod\_ssl.so has been loaded.

|  |
| --- |
| <VirtualHost \*:443>  SSLEngine On  SSLProxyEngine On  SSLCertificateFile /etc/httpd/ssl/httpd.pem  SSLCertificateKeyFile /etc/httpd/ssl/httpd.key  ServerName localhost.localdomain  ProxyPass /labs/ http://localhost:3000/labs/  </VirtualHost>  <IfModule mod\_proxy.c>  ProxyRequests on  RewriteEngine On  SSLEngine on  SSLProxyEngine on  SSLCertificateFile /etc/httpd/ssl/httpd.pem  SSLCertificateKeyFile /etc/httpd/ssl/httpd.key  ProxyPass /webassets/ https://access.redhat.com/webassets/  ProxyPass /chrome\_themes/ https://access.redhat.com/chrome\_themes/  ProxyPass /services/ https://access.redhat.com/services/  ProxyPass /click/ https://access.redhat.com/click/  ProxyPass /suggest/ https://access.redhat.com/suggest/  </IfModule> |

Step 4:

Execute command “service httpd start” as root and test the proxy using the url <https://localhost/webassets/avalon/j/lib/angular/1.2.8/angular.min.js> or https://<localhost.localdomain>/webassets/avalon/j/lib/angular/1.2.8/angular.min.js.

If some contents of js are shown, the configuration works.

3. Running the rails server

$rails server

Refer to the app via https://<localhost.localdomain>/labs/<demo $lab\_id>

**Deploying apps to openshift**

Before pushing the app to openshift, make sure “config.assets.compile” is “true” in file <app dir>/config/environments/production.rb.

$vim config/environments/production.rb

|  |
| --- |
| config.assets.compile = true |

# Ruby On Rails on OpenShift

https://mojo.redhat.com/docs/DOC-965910