OpenMind: Know Your Customer

Contents

[About OpenMind 1](#_Toc204702531)

[Installation 1](#_Toc204702532)

[Create the Database Schema 2](#_Toc204702533)

[Update database.yml 2](#_Toc204702534)

[Update environment.rb 3](#_Toc204702535)

[Configure the mail server 3](#_Toc204702536)

[Populate the Database 3](#_Toc204702537)

[Configure environment.yml 4](#_Toc204702538)

[Configure the Web Service 4](#_Toc204702539)

[Setup the Task Scheduler 4](#_Toc204702540)

# About OpenMind

OpenMind allows product companies to solicit feedback and input from users of their product.

Using the OpenMind application, user can:

* Create ideas that represents features they would like to see in the product
* View ideas that other users have created, and vote for those ideas which they would like to see added to the product
* Engage in online discussions with other users
* Mark ideas of interest to be watched

Using the OpenMind application, product managers can:

* See concrete information about which features users value most
* Move features to scheduled releases on the product roadmap
* Communicate roadmap decisions to the user community

# Installation

Follow the steps outlined below to install and configure the OpenMind application. Unless otherwise noted, all commands are run from the Rails application root directory. For example, if you unzip the openmind.zip file to ~/openmind, then this directory is your rails application root directory. In examples below, we assume the directory is named “openmind”.

There are a wide variety of rails hosting companies available for you to choose from if you opt to use a third party service to host OpenMind. We have used Hosting Rails (<http://www.hostingrails.com>) and have found them to be generally reliable. For your convenience, we have included setup steps specific to the Hosting Rails service highlighted in the instructions below.

## Create the Database Schema

Create a schema in a MySQL database for the OpenMind data. Create a user and password. Make note of the schema name, username and password, as well as the ip\_address or machine name of the machine on which MySQL is running. Ensure that the user you created has read/write privileges to the schema and can create and drop tables, indices, etc.

Setup a database schema using the MySQL Databases icon in the Databases section of the cPanel. Create a schema for the openmind community. Note that hosting rails prefixes your account name to the database name you selected. For example, if you hostingrails account name is bobstur and you selected openmind as your database, then the schema name will be “bobstur\_openmind”.   
  
Note your schema name here: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In the same screen in the database control panel, create a user. Again, hosting rails prepends the username. For example, I created a user called “openmind”. The final username will end up being: bobstur\_openmind.  
  
Note the user information below:  
  
DB Username: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
DB Password: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In same screen , add the user to the schema you created. When prompted, grant your user “All Privileges” for your schema.

## Update database.yml

Edit the file: openmind/config/database.yml  
  
Assuming you are configuring this environment for production purposes, edit the section:

production:

adapter: mysql

database: OpenMind\_production

username: root

password:

host: localhost

If you are configuring for development or test, configure the appropriate section accordingly.

You can leave the value of the host as “localhost”

## Update environment.rb

If you’re configuring a production environment, in your config/environment.rb file you'll just need to uncomment the following line to confirm you're in production mode:

ENV['RAILS\_ENV'] ||= 'production'

## Configure the mail server

If you are configuring a production environment, edit the file: config/environments/production.rb (edit development.rb and test.rb if you are setting up a development or test environment). Add entries to configure the mail server. For example:

ActionMailer::Base.smtp\_settings = {

:address => 'themailserver',

:port => 27,

:authentication => :plain,

:user\_name => 'emailusername',

:password => 'emailpassword'

}

You can use sendmail by appending the following values to the production.rb file:

config.action\_mailer.raise\_delivery\_errors = true

config.action\_mailer.delivery\_method = :sendmail

config.action\_mailer.perform\_deliveries = true

config.action\_mailer.default\_charset = "utf-8"

config.action\_mailer.default\_content\_type = "text/html"

Note: The migration performed in step creates record in the database which will cause observes to send email. Therefore, you must configure the mail server as outlined in this step before you attempt to run the migration. Alternatively, you can uncomment the following line in production.rb (though you will probably want to re-comment the line at a later point):

# config.action\_mailer.raise\_delivery\_errors = false

## Populate the Database

Populating the database by running the following command:

rake db:migrate

Note: if you run into any errors during installation, you can view the log file at logs/production.log for details.

## Configure environment.yml

Edit the file config/environment.yml. Set the options as appropriate. For further explanation on each option, see “” on page .

## Configure the Web Service

At this point, the app is configured, and wait remains is to set up the server of your choice (fast\_cgi, mongrel, etc.). What follows are steps specific to hosting rails to get the application up and running.

Create the symlink so the www directory is your Rails app public folder. Get to your application root directory and:

[~]# mv ~/public\_html ~/public\_html\_backup  
[~]# ln -s ~/yourapp/public ~/public\_html

Notify hosting rails support to configure your apache vhost to use mod\_rails.

## Setup the Task Scheduler

OpenMind includes a task scheduler that runs background tasks – for example, checking for new entries in discussion forums once a day and sending out emails. You can start the task scheduler by issuing the following command:

ruby script/task\_server\_control.rb run

It is strongly suggested that you set up this script to be started automatically via cron or monit.d or another mechanism, so that it will start up automatically in the event of a server reboot.

On unix environments such as hosting rails, you can you cron to ensure that the task scheduler will be started automated if the machine is rebooted.

Enter the following command:

crontab –e

Add the following line to the file:

@reboot cd /home/username/openmind && ruby script/task\_server\_control.rb run

# Configuration Options

OpenMind provides a number of configuration options that you can specify in the config/environment.yml file.

### Host

OpenMind sends email notifications when certain events occur. Many of those emails contains links to the application.

The host name. This is used to construct the url for the link that appears in

# email notifications

host: localhost

# The port number. This is used to construct the url for the link that appears in

# email notifications

port: 3000

# The email address of the administrator. This email will be used as the "from"

# address in emails sent by openmind.

admin\_email: admin@openmind.org

# The subject prefix. This will be the prefix in the subject for emails sent by

# openmind. For example, "OpenMind: Your new account has been created..."

email\_subject\_prefix: OpenMind

# The number of minutes until a vote can no longer be rescinded

rescind\_minutes: 240

# The default number of days before allocations expire

allocation\_expiration\_days: 120

# The url to the directory where the images for emails are stored

email\_image\_url: http://localhost:3000/themes/scribe/images/email

# The name of the theme (e.g., skin). This should correspond to a directory

# under the public/themes directory

app\_theme: openmind

# The number of days within which to warn the user of unused allocations

# about to expire. A value of 0 means never warn the user.

allocation\_expiration\_warning\_days: 45