How To Setup Ruby on Rails with Postgres

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: http://faq.asphosthelpdesk.com/article.php?id=228

Introduction

Postgres (or PostgreSQL) is an open source database. Ruby on Rails is an open source web framework written in Ruby. Rails is database agnostic, meaning it can be used with a variety of different databases. By default it assumes that MySQL is being used, but it's quite easy to use with Postgres instead.

This guide will step you through creating a Rails application that uses a Postgres database. You can follow the guide on your local machine or a VPS.

Installing Requirements

Installing Rails using RVM

The easiest way to install Rails is using RVM, which also installs Ruby. To install RVM you will need to ensure your system has curl installed (how you do this depends on your OS). If you already have RVM installed, skip to the next section.

RVM can install Ruby and Rails automatically as part of its installation. To do so, run this command:

```
\curl -L https://get.rvm.io | bash -s stable --rails
```

Note: you should review the RVM install script before running it (or any other remote script that you pipe into bash.

RVM will install itself on your system. You can now use it to manage your Ruby versions. This is useful as you may require different versions of Ruby for different projects. RVM also installed the Rails gem for us.

Installing Rails using RubyGems

If you already have RVM installed, you don't need to re-install it. Instead you can simply install Rails by installing the gem:

```
gem install rails
```

This will install Rails and any other gems it requires.

Installing Postgres

The method of installing Postgres depends on your OS. See postgresql.org/download for a full list. Generally it's easiest to use a package manager such as apt-get on Ubuntu or Homebrew on OS X.

If you are installing Postgres on a local machine you may also want to install a GUI (though this guide assumes command line usage). pgAdmin isn't the prettiest tool in the world, but it does the job.

Finally, you'll want to install the pg gem so that you can interface with Postgres from Ruby code. To do so:

```
gem install pg
```

Setting Up Postgres

Create a Postgres user for the Rails app we'll create in the next step. To do this, switch into the Postgres user:

```
su - postgres
```

Then create a user (or a "role", as Postgres calls it):

```
create role myapp with createdb login password 'password1'
```

Creating Your Rails App

To create a Rails app configured for Postgres, run this command:

```
rails new myapp --database=postgresql
```

This creates a directory called "myapp" which houses an app called "myapp" (you can name it anything you like when running the command). Rails expects the name of the database user to match the name of the application, but you can easily change that if need be.

We will now configure which database Rails will talk to. This is done using the database.yml file, located at:

RAILS_ROOT/config/database.yml

Note: RAILS_ROOT is the Rails root directory. In the above example, it would be at /myapp (relative to your current location).

The database.yml file is used by Rails to connect to the appropriate database for the current Rails environment. It uses YAML, a data serialization standard. There are a few databases listed here for different environments; development, test, and production. By default Rails will expect a different database for each environment. This is handy because, for example, the test database is emptied and rebuilt every time you run Rails tests. For each database, ensure that the username and password match the username and password you gave your Postgres user.

Once configured, your database.yml should contain something like this:

```
development:
  adapter: postgresql
  encoding: unicode
  database: myapp_development
  pool: 5
  username: myapp
  password: password1
test:
  adapter: postgresql
  encoding: unicode
  database: myapp_test
  pool: 5
  username: myapp
  password: password1
```

You can then run:

```
rake db:setup
```

This will create development and test databases, set their owners to the user specified, and create "schema_migrations" tables in each. This table is used to record your migrations to schemas and data.

Running Rails

You should be able to start your Rails app now:

```
rails server
```

If you navigate to localhost:3000 you should see a Rails landing page. This doesn't really do much though. To interact with our database, let's create a scaffold:

```
rails g scaffold Post title:string body:text
rake db:migrate
```

Now navigate to localhost:3000/posts. From here, you can create new posts, edit existing posts, and delete posts. See the Rails getting started guide for more introductory operations.

Your Rails app is now talking to a Postgres database.