Ruby & Rails Naming Conventions

Ruby_Rails_Naming_Conventions.md

Alex's Rails Cheat Sheet

I think the most confusing thing that I have found about Ruby on Rails so far has been the transition from (trying to) write code myself to the use of the fabled "Rails Magic". So, to help my own understanding of a few core Ruby on Rails concepts, I have decided to write something on what I think is a ***CRITICAL*** topic... the idea of **Convention over Configuration** and why (in my mind) it is the most important thing that helps Rails become magic!

(This may be a topic that we cover in more detail in class but as I said, I'm writing this for my own understanding... I hope it helps someone else understand things too... Perhaps you can give me a hand when I'm crying next week!)

##Convention over configuration ###What does this "actually" mean...

Wikipedia says:

> "Convention over configuration (also known as coding by convention) is a software design paradigm which seeks to decrease the number of decisions that developers need to make, gaining simplicity, but not necessarily losing flexibility."

Rails' homepage says:

> "Ruby on Rails is an open-source web framework that's optimized for programmer happiness and sustainable productivity. It lets you write beautiful code by favoring convention over configuration."

Before having a web application framework as powerful as Rails with tools that can do things like scaffold things for you, developers had to make LOTS more decisions about how they called each variable, controller, model, database table etc that they put into their application. Although there were things like lower_snake_case, cameBack and UpperCamelCase there was still a lot of room for personal decision...

In Rails, there are different conventions for the:

1. **File name** (outside) e.g. orders_controller.rb
2. **File definition** (inside) e.g. OrdersController
3. **File location** (inside) e.g. /app/controllers

##Rails Naming Convention

Rails use the same naming convention as Ruby (for a list of the Ruby naming conventions scroll down) with some additions:

Variable - *e.g. order_amount, total*
Variables are named where all letters are lowercase and words are separated by underscores.

Class and Module - *e.g. InvoiceItem*
Classes and modules use MixedCase and have no underscores, each word starts with a uppercase letter.

Database Table - *e.g. invoice_items, orders* Table names have all lowercase letters and underscores between words, also all table names need to be plural.

Model - *e.g. Order*
The model is named using the class naming convention of unbroken MixedCase and is always the singular of the table name. e.g.

- Table name might be orders (plural)
- The model name would be Order (singular)
- Rails will then look for the class definition in a file called order.rb in the /app/models directory.
- If the model class name has multiple capitalised words, the table name is assumed to have underscores between these words.

Controller - *e.g. OrdersController*
Controller class names are pluralized, such that **OrdersController would be the controller class for the orders table**. Rails will then look for the class definition in a file called orders_controller.rb in the /app/controllers directory.

Files, Directories and other pluralization
Files are named using lowercase and underscores. Assuming we have an Orders controller then the following other conventions will apply:

- That there is a helper module named **OrdersHelper** in the **orders_helper.rb** found in the app/helpers directory.
- Rails will look for view template files for the controller in the **app/views/orders** directory.

- Output from this view will then be used in the layout defined in the **orders.html.erb** in the **app/views/layouts** directory.
- Test files including **order_test.rb** will be created in the **/test/unit** directory, a file will be created in the **/test/fixtures** directory called **orders.yml** and finally a file called **orders_controller_test.rb** will be created in the **/test/functional directory**

Primary Key
The primary key of a table is assumed to be named id.

Foreign Key
The foreign key is named with the singular version of the target table name with _id appended to it, e.g. order_id in the items table where we have items linked to the orders table.

Many to Many Link Tables
Tables used to join two tables in a many to many relationship is named using the table names they link, with the table names in alphabetical order, for example items_orders.

Automated Record Timestamps
You can get ActiveRecord to automatically update the create and update times of records in a database table. To do this create two specially named columns created_at and updated_at to your table, i.e. t.datetime :created_at and t.datetime :updated_at. If you only want to store the date rather than a date and time, use :created_on and :updated_on.

##Naming Convention Summary

#####Model Naming Convention

```
Table:                  orders
Class:                  Order
File:                   /app/models/order.rb
Primary Key:    id
Foreign Key:    customer_id
Link Tables:    items_orders
```

#####Controller Naming Convention

```
Class:                  OrdersController
File:                   /app/controllers/orders_controller.rb
Layout:                 /app/layouts/orders.html.erb
```

#####View Naming Convention

```
Helper:                 /app/helpers/orders_helper.rb
Helper Module:  OrdersHelper
Views:                  /app/views/orders/… (list.html.erb for example)
```

####Tests Naming Convention

```
Unit:                   /test/unit/order_test.rb
Functional:     /test/functional/orders_controller_test.rb
Fixtures:               /test/fixtures/orders.yml
```

##The Ruby Naming Conventions

As Jon would say, ProTip:

> Ruby uses the first character of the name to help it determine it's intended use.

Local Variables
These should be a lowercase letter followed by other characters, naming convention states that it is better to use underscores rather than camelBack for multiple word names, **e.g. mileage, variable_xyz**

Instance Variables
Instance variables are defined using the single "at" sign (@) followed by a name. It is suggested that a lowercase letter should be used after the @, **e.g. @colour**

Instance Methods
Method names should start with a lowercase letter, and may be followed by digits, underscores, and letters, e.g. paint, close_the_door

Class Variables
Class variable names start with a double "at" sign (@@) and may be followed by digits, underscores, and letters, **e.g. @@colour**

Constant
Constant names start with an uppercase letter followed by other characters. Constant objects are by convention named using all uppercase letters and underscores between words, **e.g. THIS_IS_A_CONSTANT**

Class and Module
Class and module names starts with an uppercase letter, by convention they are named using MixedCase, **e.g. module Encryption, class MixedCase**

Global Variables
Starts with a dollar ($) sign followed by other characters, **e.g. $global**

Some info borrowed from - http://itsignals.cascadia.com.au/?p=7