Computer and Information Science
www.ccsenet.org/journal.html
Vol. 1, No. 1
February 2008

A Study on the Issue of Chinese Character in Ruby on Rails and Relevant Solving Scheme

Ying Li
School of Management, Tianjin Polytechnic University, Tianjin 300387, China
E-mail: liying@smtjpu.net

Abstract

As Ruby on rails springs out in China, it wins greater attentions from developers at once. Focusing on the Chinese character issue appeared in the Ruby on rails development frame, this paper puts forward specific solving scheme.

Keywords: Ruby, Rails, Chinese character

1. Introduction

Although Ruby has already emerged for a period of time, it has not won popular except in Japan before the year 2000 because it does not have the English file. Since the appearance of Rails frame in 2000, the Ruby has become popular in Web development. The Rails is a frame edited by pure Ruby. It serves as a powerful support for Web development, such as data reflection, MVC mode, Web services, and safety. And these functions can be realized easier than similar products. But the development of Ruby on rails is still at its first step in China. Up till now, there is still no perfect scheme for solving the Chinese character issue. It is a critical weakness for domestic programmers who are engaged in developing Chinese interface.

Since we have already determined to use Ruby on rails to finish some projects, we should try our best to achieve this goal. As a matter of fact, most programmers are persistent and never give up. Therefore, focusing on the Chinese character issue emerged in program development, this paper puts forward specific solving scheme, with the hope of being helpful references for all programmers.

2. The code of rails

As the base of rails, although the ruby language supports the Unicode character string, its class libraries do not support the Unicode. In other words, in dealing with one Chinese character, Ruby fails to identify that two bytes under the Unicode format mean one character.

2.1 The relationship between utf8 and Unicode

The utf8 means Unicode Translation Format. Unicode has many different formats, such as utf8 and utf16, in transmission. In fact, the utf8 is not a friendly format for Chinese character. Why? In the utf8, one Chinese character has three bytes. But one English ascii character has one bytes. As a result, Chinese characters will occupy a large space and their numbers can not be deduced merely by their bytes. In contrast, the utf16 is better. Each character has two bytes. However, the utf16 and even the utf32 are seldom used. The utf8 is generally recognized.

2.2 Converse the code

If we require for the ‘iconv’ database, we can do:

Iconv. conv (“utf8”, “GBK”, “……”). Turn the characters under the GBK mode into the ones under the utf8 mode. As the source characters have illegal code, we ask iconv to neglect it. Otherwise, iconv will order to stop.

2.3 The Chinese characters under the utf8 mode

Calculate the number of the characters.

In the ruby environment, we can do:

Require ‘jcode’

$KCODE='u' # or $KCODE='UTF8'

They are equal.

Get certain characters.

“How do you do (in Chinese)” [0,1] # random code

“How do you do (in Chinese)”. Scan (/./) [0,1]. join # “you”

3. The Chinese character issue in the rails database

Under the common situation, change the MySQL file C:\Program Files\MySQL\MySQL Server 5.0\my.ini. Make two changes. default-character-set=utf8. Then restart MySQL (windows service). This step can also be realized by instance wizard. By this way, the Chinese character issue existed in most rails databases can be solved. But there are still some special situations.

As database mysql uses utf-8 to store Chinese characters, the SQLyog Enterprise shows a series of random codes. Because the Chinese characters are turned into utf8 mode as we store them. And they are still in this mode as they are on the screen. At this time, we can use this order as follow.

Set names ‘gbk’;

Here, Chinese characters are changed into gbk mode from utf8 mode. We can read the Chinese characters at last.

For another instance, if we set source file code as UTF-8 mode, the database content is normal in web pages. But if we require the database content by phpMyAdmin and mysql-front, it shows random code. And under the control platform mode, the database content still shows random code. In practice, we find the solving method as follow.

(1) For MySql: set the character as utf8 mode

(2) Write the following codes into application.rb

before filter: configure charsets

def configure charsets

@response.headers["Content-Type"] = "text/html; charset=utf-8"

Set connection charset. MySQL 4.0 doesn’t support this so it

will throw an error, MySQL 4.1 needs this

suppress(ActiveRecord::StatementInvalid) do

ActiveRecord::Base.connection.execute 'SET NAMES UTF8'

end

end

(3) Write following codes into environment.rb

$KCODE = 'u'

require 'jcode'

4. The random code issue in uploading and downloading files in rails

For Chinese users, the default code of files is gbk. In other words, the file name in utf8 mode is random code for us. That is why as we upload or download files, we usually come across random code. Therefore, we should converse the random code.

Generally speaking, ruby is not good at providing with support for Unicode, which in a sense blocks the development. Therefore, this issue has to turn to other tools. In general, we are merely familiar with the utf8 mode and the gbk mode. As long as the database, connection way, source code, and page files are in utf8 mode, the work will become easy. But recently as we download files, we still find the random code issue. Then we can use the additional database Iconv in ruby. It is simple.

Iconv. new (to, from)

It is a class method. Generate and back from to the new converter. To and from respectively refers to the after-conversion character and the before-conversion character.

Iconv#iconv(str)

Start to converse characters and return the result.

Firstly, we define two global variable in environment.rb

UTF8_TO_GBK = Iconv.new "gbk", "utf-8"

GBK_TO_UTF8 = Iconv.new "utf-8", "gbk"

Then as we download files, we use “send_file”. The file name can be shown in Chinese character. But we still have to turn the utf8 mode into the gbk mode. We use:

:filename => UTF8_TO_GBK.iconv(filename)

As a result, it shows Chinese character. As far as uploading files with Chinese names is concerned, the main problem is the random code of local files stored in the service. We have to ensure that the local files are in GBK mode as we read or store them.

(1) Uploading files

filename=file.original_filename

File.open("#{RAILS_ROOT}/documents/#{UTF8_TO_GBK.iconv(filename)}", "wb")

By this way, the file names uploaded on the service are shown in Chinese characters.

(2) Downloading files

filename=Document.find(params[:id]).name

send_file(“#{RAILS_ROOT}/documents/#{UTF8_TO_GBK.iconv(filename)}”,:filename=>UTF8_TO_GBK.iconv(filename))

By this way, the file names as we read or store files are shown in Chinese characters.

5. Conclusion

Here this paper puts forward a way of solving the Chinese character issue in programming by rails. Comparing with JAVA, Ruby on rails is a new frame and it is still immature. Especially in China, it is still on its initial stage and few articles can serve as valuable reference. As a matter of fact, the Chinese character issue in rails is originated from the unpopularity of Chinese and the lag-behind technology of China. Rails and ruby have taken references from java in many ways. But they sustain their own special advantages. It is believed that more and more people will use rails to write web programs.

References

Fan, Kai. (2006). The miracle created by Ruby. *Software World*. No.21.

Fikko. (2005). Ruby on Rails. *Programmer*. No.9.

Meng, Yan. (2005). Ruby in Rails: the wonderful new Web development technology. *Programmer*. No.9.

Meng, Yan & Zhu, Haiyan. (2006). How will Java meet new challenges. *Programmer*. No.11.

Zhang, Xin & Kui, Gang. (2002). Initial Ruby language. *Internet Information World*. No.12.