Subscribe

Keith R. Bennett's Technical Blog

Search

About

Using Oracle in JRuby with Rails and Sequel

By keithrbennett on January 9th, 2013

The Ruby culture prefers open source technologies, and when it comes to relational data bases, MySQL and Postgres are commonly used. However, there are times when the Rubyist will not be in a position to choose technologies and must inherit legacy decisions. For example, a common issue in the enterprise is the need to integrate with Oracle. In this article, I'll talk about integrating Oracle and JRuby (1), using both Active Record (Ruby on Rails) and the Sequel gem.

JDBC

The JVM typically communicates with data bases using JDBC (Java Data Base Connectivity). The JDBC layer provides an abstraction with which application code can access the data base without (for the most part) needing to know anything about its underlying implementation.

JDBC Jar Files

A JDBC jar (jar = *j*ava *ar*chive) file is provided for each data base product, typically by the vendor, and contains the code needed to implement the JDBC contract for that target data base. Since MySQL and Postgres are open source, their jar files can be freely copied around, and as a result, are included in the respective JDBC gems. This greatly simplifies configuration, since the gem takes care of storing and providing the location of the JDBC jar file.

The activerecord-jdbc-adapter gem includes several JDBC jar files. The complete list can be found by looking at the directory names beginning with "jdbc-" on the activerecord-jdbc-adapter GitHub page here, and, at

the time of this writing, consists of Derby, H2, Hypersonic, jTDS (SQL
Server and Sybase), MySQL, Postgres, and SQLite.

Dealing with the Oracle JDBC Jar File

With Oracle, it's a different story. Oracle does not permit freely coping
their JDBC jar file, and in order to download it, you'll probably need to go
to the Oracle web site and log in first. It would not be legal to write an
Oracle JDBC gem that packaged this jar file, so, unfortunately, extra work
is required.

The solution I chose was to:

1) download it to a directory outside of my project (/opt/oracle-
jdbc/ojdbc6.jar)

2) have an environment variable point to it — I added this to my ~/.zshrc
file (you might use ~/.bashrc instead):

```
1   export ORACLE_JDBC_JAR=/opt/oracle-jdbc/ojdbc6.jar
```

3) use that variable at runtime to locate it (separate solutions for Rails and
Sequel below).

Oracle and Rails – environment.rb and database.yml

For Rails, you'll need this in your config/initializers/environment.rb so
that the JDBC jar file can be found at runtime:

```
1   $CLASSPATH << ENV['ORACLE_JDBC_JAR']
```

Now you'll need to provide the appropriate values in the database.yml file
(test and production groups are omitted for brevity):

```
1   <%
2      require 'socket';
3      host_name = ENV['OJTEST_DB_HOSTNAME']
4      host_ip   = IPSocket.getaddress(host_name)
5      db_name   = ENV['OJTEST_DB_DBNAME']
6      userid    = ENV['OJTEST_DB_USERID']
7      password  = ENV['OJTEST_DB_PASSWORD']
8      dev_url = "jdbc:oracle:thin://@#{host_ip}:1521:
9   %>
10
11  development:
12     adapter: jdbc
13     username: <%= userid %>
14     password: <%= password %>
15     driver: oracle.jdbc.OracleDriver
16     url:  <%= dev_url %>
```

As you can see, I used environment variables beginning with

"OJTEST_DB_" to provide the required values, although that is not important and you can use any approach that works for you.

More importantly, note that I am translating the Oracle host's name to its IP address. This was necessary due to an apparent bug in Oracle's driver.

Oracle and the Sequel Gem

There is also the excellent Sequel gem that can be used for general data base access, even (perhaps especially) in a minimal script. Here's a sample script that worked succesfully for me:

```ruby
#!/usr/bin/env ruby

require 'sequel'
require 'socket';

$CLASSPATH << ENV['ORACLE_JDBC_JAR']

def init_connection

  host_name = ENV['OJTEST_DB_HOSTNAME']
  host_ip   = IPSocket.getaddress(host_name)

  db_name   = ENV['OJTEST_DB_DBNAME']

  userid    = ENV['OJTEST_DB_USERID']

  password  = ENV['OJTEST_DB_PASSWORD']

  dev_url = "jdbc:oracle:thin://@#{host_ip}:1521:

  url = "jdbc:oracle:thin:#{userid}/#{password}@#

  puts "Connecting to #{url}..."
  db = Sequel.connect(url)
  puts 'Initialized connection.'
  db
end

def create_table(db)
  puts 'Creating table'
  # db.run "CREATE TABLE artists (id NUMBER, name
  DB.create_table(:artists) do
    primary_key :id
    String :name
  end
end

def add_records(db)
  puts 'Adding records'
  dataset = db[:artists]
  dataset.insert(name: 'Vincent Van Gogh')
  dataset.insert(name: 'Lino of Cebu')
  dataset
end

def show_records(dataset)
  puts 'Showing records'
  dataset.map { |row| puts row[:name] }
end

# Call this once to set up the table with some re
```

```
52    def setup(db)
53      create_table(db)
54      add_records(db)
55    end
56
57    DB = init_connection
58    # DB.run 'drop table artists'
59    setup(DB)
60    show_records(DB[:artists])
```

Conclusion

This setup took me some time to figure out, but after I did, things went smoothly. I'd like to hear if the approaches described here worked for you or if you have any problems with them.

Footnotes:

(1) an implementation of Ruby that runs on the Java Virtual Machine (aka *JVM*)

Categorized under: Uncategorized.

Tagged with: no tags.

2 Responses to "Using Oracle in JRuby with Rails and Sequel"

January 9, 2013 at 2:20 pm

substars says:

Cool stuff! Just wanted to add that the activerecord-oracle_enhanced-adapter gem supports some additional Oracle-specific features and works on both JRuby (via JDBC) and MRI (with ruby-oci8):

https://github.com/rsim/oracle-enhanced

Log in to Reply

January 14, 2013 at 4:18 pm

keithrbennett says:

Thanks for that information. I'll be sure to check that out if I have to do any serious work with Oracle again.

– Keith

Log in to Reply

Leave a Response

You must be [logged in](#) to post a comment.

← Copying (RVM) Data Between Hosts Using ssh, scp, and netcat

 in Your System Prompt →

META

- Register
- Log in
- Entries RSS
- Comments RSS
- WordPress.org

[Search]

ARCHIVES

- November 2015
- November 2013
- August 2013
- January 2013
- December 2012
- November 2012
- September 2012
- July 2012
- January 2012
- June 2011
- May 2011
- March 2010
- July 2009
- March 2009
- February 2009
- November 2008

RECENT ACTIVITY

Posts	Comments

- The Case for Nested Methods in Ruby
- Ruby's inject/reduce and each_with_object
- in Your System Prompt
- Using Oracle in JRuby with Rails and Sequel
- Copying (RVM) Data Between Hosts Using ssh, scp, and netcat
- Building A Great Ruby Development Environment and Desktop with Linux Mint 13 "Maya" Mate
- Intro to Functional Programming in Ruby
- WordPress Administration with Ruby
- Stealth Conditionals in Ruby
- Hello, Nailgun; Goodbye, JVM Startup Delays

ARCHIVES

Categories	Tags	Dates

Authors
- Uncategorized

Powered by WordPress and the PressPlay Theme   Copyright © 2016 Keith R. Bennett's Technical Blog