Instalação do Redmine 0.8.4 utilizando um Banco de Dados Oracle XE

A instalação do Redmine ocorre, primeiramente, preparando o ambiente Ruby on Rails sobre o qual ele irá executar. Ao se realizar o download da versão 1.8.6 do Ruby, no link abaixo, deve-se seguir as instruções do *wizard* de instalação: http://rubyforge.org/frs/download.php/29263/ruby186-26.exe

Deve ser feito o download do Redmine no endereço do RubyForge, situado em http://rubyforge.org/frs/?group_id=1850. A versão utilizada é a **0.8.4** em sua versão zip. Termindado o download, extraia o conteúdo dentro da pasta "c:\ruby\apps". Antes de migrar o banco, devem ser feitas algumas configurações nos arquivos do Redmine para permitir a conversação com o Oracle e o Subversion de forma adequada.

Primeiramente, dentro do diretório redmine\db\migrate\, deve-se apagar o arquivo de migração 087_change_projects_description_to_text.rb. Logo em seguida, alterar o arquivo 001_setup.rb com a seguinte informação:

```
create_table "enumerations", :force => true do |t|
   t.column "opt", :string, :limit => 4, :default => "", :null => false
     t.column "name", :string, :limit => 40, :default => "", :null => false
end
```

```
  create_table "projects", :force => true do |t|
    t.column "name", :string, :limit => 30, :default => "", :null => false
    t.column "description", :string, :default => "", :null => true
    t.column "homepage", :string, :limit => 60, :default => ""
    t.column "is_public", :boolean, :default => true, :null => false
    t.column "parent_id", :integer
    t.column "projects_count", :integer, :default => 0
    t.column "created_on", :timestamp
    t.column "updated_on", :timestamp
  end
```

```
  create_table "users", :force => true do |t|
    t.column "login", :string, :limit => 30, :default => "", :null => true
    t.column "hashed_password", :string, :limit => 40, :default => "", :null => true
    t.column "firstname", :string, :limit => 30, :default => "", :null => true
    t.column "lastname", :string, :limit => 30, :default => "", :null => false
    t.column "mail", :string, :limit => 60, :default => "", :null => true
    t.column "mail_notification", :boolean, :default => true, :null => false
    t.column "admin", :boolean, :default => false, :null => false
    t.column "status", :integer, :default => 1, :null => false
    t.column "last_login_on", :datetime
    t.column "language", :string, :limit => 2, :default => ""
    t.column "auth_source_id", :integer
    t.column "created_on", :timestamp
    t.column "updated_on", :timestamp
  end
```

Modifique o arquivo 007_create_journals.rb conforme abaixo:

```
  def self.up
    create_table :journals, :force => true do |t|
      t.column "journalized_id", :integer, :default => 0, :null => false
      t.column "journalized_type", :string, :limit => 30, :default => "", :null => false
      t.column "user_id", :integer, :default => 0, :null => false
      t.column "notes", :string
      t.column "created_on", :datetime, :null => false
    end
```

```
def self.down
    drop_table :journal_details
    drop_table :journals

    create_table "issue_histories", :force => true do |t|
      t.column "issue_id", :integer, :default => 0, :null => false
      t.column "status_id", :integer, :default => 0, :null => false
      t.column "author_id", :integer, :default => 0, :null => false
      t.column "notes", :string, :default => ""
      t.column "created_on", :timestamp
    end
```

Em seguida, no arquivo 091_change_changesets_revision_to _string.rb, as seguintes linhas devem ser utilizadas, mudando as originais através da eliminação do termo :null => false:

```
class ChangeChangesetsRevisionToString < ActiveRecord::Migration
  def self.up
    change_column :changesets, :revision, :string, :null => false
  end

  def self.down
    change_column :changesets, :revision, :integer, :null => false
  end
end
```

Deve-se copiar o arquivo "config\database.yml.example" para "config\database.yml". Esse arquivo conterá a configuração necessária para conexão ao banco de dados Oracle, seguindo o formato abaixo:

```
production:
        adapter: oracle_enhanced
        host: <hostname>:<port>:<sid>
        username: railsuser
        password: railspasswd
```

Deve-se criar uma sub-pasta dentro do diretório \config\ nomeada de **initializers**, onde será criado um arquivo de nome oracle_enhanced.rb contendo o seguinte conteúdo:

```
ActiveRecord::ConnectionAdapters::OracleEnhancedAdapter.emulate_dates_by_column_name = true
ActiveRecord::ConnectionAdapters::OracleEnhancedAdapter.instance_eval do
  self.string_to_date_format = "%d/%m/%Y"
  self.string_to_time_format = "%d/%m/%Y %H:%M:%S"
end
```

Deve-se instalar, através do comando **ruby ruby-oci8-1.0.3-mswin32.rb** o conector ruby-oci8, através do seu download e armazenamento na pasta root do ruby, no seguinte endereço:

http://rubyforge.org/frs/download.php/41043/ruby-oci8-1.0.3-mswin32.rb

Em seguida, é necessário instalar o adaptador do banco de dados Oracle usado, sendo realizado através do seguinte comando a partir do diretório ruby:

gem install -v=1.1.8 activerecord-oracle_enhanced-adapter --include-dependencies

O gem é um gerenciador e de pacotes/plugins/engine do ruby que irá instalar o adaptador e suas dependências. Esse adaptador possui o inconveniente de começar a numerar os *tickets* a partir do número 10000. O desejável é que se comece pelo numero 1. Para resolver esse problema, faz-se necessário alterar o arquivo C:\ruby\lib\ruby\gems\1.8\gems\activerecord-oracle_enhanced-adapter-1.1.8\lib\active_record\connection_ adapters\oracle_enhanced_adapter.rb como segue abaixo:

```
def create_table(name, options = {}, &block) #:nodoc:
  create_sequence = options[:id] != false
  if create_sequence
    super(name, options, &block)
  else
    super(name, options) do |t|
      class <<t
        attr_accessor :create_sequence
        def primary_key(*args)
          self.create_sequence = true
          super(*args)
        end
      end
      result = block.call(t)
      create_sequence = t.create_sequence
    end
  end
  seq_name = options[:sequence_name] || "#{name}_seq"
  execute "CREATE SEQUENCE #{seq_name} START WITH 1" if create_sequence
end
```

IMPORTANTE: Copie esse arquivo para o diretório <redmine_dir>\vendor\rails\activerecord\lib \active_record\connection_adapters.

Um usuário **ruby** com uma senha definida deve estar criado na base de dados Oracle, para poder dar início ao processo de migração de dados. Nesse momento, deve-se criar a estrutura do banco de dados 'redmine', digitando o comando no diretório do redmine:

rake db:migrate RAILS_ENV=production

É altamente recomendável que se faça o carregamento dos valores padrões do banco de dados ora criado. Isso será feito com o comando:

rake redmine:load_default_data RAILS_ENV=production

O servidor padrão para ambientes de desenvolvimento/teste em rails é o WEBrick, que possui um desempenho muito aquém do desejável para ambientes de produção. Para remediar isso, torna-se necessário instalar um servidor que cumpra bem o seu papel em ambientes destinados a uso intenso. Seu nome é Mongrel e pode ser instalado através dos comandos (use sempre as opções envolvendo a plataforma *i386-mswin32*):

gem install -v=1.1.3 mongrel --include-dependencies

A partir daí, o servidor do Redmine já pode ser iniciado pelo comando:

ruby script/server –e production

Certifique-se que a mensagem abaixo apareça e que seja realmente o ambiente de produção aquele que está iniciando:

```
=> Booting Mongrel (use 'script/server webrick' to force WEBrick)
=> Rails 2.1.2 application starting on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
** Starting Mongrel listening at 0.0.0.0:3000
** Starting Rails with production environment...
```

[**OPICIONAL**] Para automatizar esse processo de start do servidor, no momento que o Windows inicializar, é preciso instalar um pacote gem que permita criar serviços no Windows. Esse pacote pode ser instalado através do comando (na plataforma *mswin32*):

gem install -v=0.5.2 win32-service --include-dependencies

Em seguida, instalar o serviço mongrel, utilizando a plataforma *mswin32*:

gem install -v=0.3.3 mongrel_service --include-dependencies

Finalmente, adicionar o serviço ao Windows, através do comando:

mongrel_rails service::install -N REDSERV -c c:\ruby\apps\redmine -p 3000 -e production

O servidor Apache será utilizado para receber as requisições dos usuários do sistema através de uma configuração prévia. Nessa situação, não se está visando balanceamento de carga no servidor, devido ao fato de a aplicação Redmine ser acessada por uma equipe pequena. Mas caso haja o aumento de quantitativo de acessos e se perceba deficiências no desempenho das respostas às requisições, pode-se, de acordo com [http://nlakkakula.wordpress.com/2008/11/24/10-steps-for-deploying-your-ruby-on-rails-application-on-a-windows-server-2008-apache-mongrel-cluster/], configurar um *cluster*[1] para balanceamento de carga, fugindo do escopo deste trabalho.

1 – Máquinas interligadas em rede para processamento paralelo da informação, com vistas a aumentar o poder de cômputo das informações e, por conseguinte, o desempenho das aplicações.

Com o servidor Apache instalado, deve-se criar um arquivo na pasta DIR\Apache2.2\conf nomeado de httpd-proxy.conf, contendo o seguinte código:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
ProxyRequests Off

<Proxy *>
  Order deny,allow
  Allow from all
</Proxy>

Alias /redmine "c:/ruby/apps/redmine"
<Directory "C:/ruby/apps/redmine">
  Options Indexes FollowSymLinks
  AllowOverride none
  Order allow,deny
  Allow from all
</Directory>

ProxyPass /redmine/ http://127.0.0.1:3000/
ProxyPass /redmine/ http://127.0.0.1:3000/
ProxyPassReverse /redmine/ http://127.0.0.1:3000/
```

Finalmente, no arquivo já existente na pasta acima, chamado de httpd.conf assegurar que *LoadModule alias_module modules/mod_alias.so* não está comentado (precedido por #) e inserir a seguinte linha no topo deste arquivo:

```
Include conf/httpd-proxy.conf
```

Até o momento, podemos ter acesso à página inicial do Redmine através do Apache. Contudo, a estrutura interna da aplicação, criada pelo rails, não se torna acessível para o proxy do servidor Apache, fazendo com que as folhas de estilos e os links estejam indisponíveis. Para resolver esse problema, é preciso utilizar um proxy reverso que intercepte as requisições e corrija o endereço dos links. Esse papel sera desempenhado com a instalação de um plugin **reverse_proxy_fix** que deverá ser executado a partir do diretório da aplicação:

ruby script/plugin install http://svn.napcsweb.com/public/reverse_proxy_fix

Ao se pedir o formato da URL, informar http://localhost/redmine. Com o comando services.msc deve-se tornar o processo REDSERV, recém criado, automático ao Windows iniciar. E por meio da URL acima, será feito o acesso ao sistema, que no primeiro acesso será feito por:

login: admin

senha: admin

A configuração da aplicação deverá ser feita conforme a necessidade do fluxo existente na organização de desenvolvimento de software. Essa etapa será detalhada na próxima seção.