Integrate Rails into an Existing IIS Web infrastructure using Mongrel

This article will walk you through the steps of installing Ruby, Gems, Rails, and other important libraries on a Windows 2003 server with IIS.

Microsoft's Internet Information Server is a popular proprietary web and application server. Those who have attempted to run Rails applications on IIS have had mixed results at best. The goal of this article is to show you how to integrate a Rails application into your existing IIS structure.

We're going to walk you through setting up a production environment that will have a URL structure like

http://myserver.mydomain.com/test/
http://myserver.mydomain.com/anothertest/

Revision 1.0 – May 4, 2006

Integrate Rails into an Existing IIS Web infrastructure using Mongrel

Assumptions

This article assumes that you have a working Rails application to test, that you are familiar with how IIS works, and that you have the MySQL Database installed on the local machine. IIS should be running on TCP Port 80.

Shopping List

In order to make this work, you'll need to download the full version of ISAPI Rewrite from a company called Helicon. You can obtain a free 30 day unlimited trial from their web site but they charge $70 per server (or less if you buy more than once license) if you want to use it in production. While there are free rewrite plugins available, this is the only one I know of that provides proxy capabilities for IIS. Visit http://www.isapirewrite.com/ for more information.

Installing Ruby, Rails, and Mongrel

Installing Ruby, Rubygems, and RDoc

1. Download the One-Click Ruby Installer. You can use **Ruby 1.8.2** or **Ruby 1.8.4**. However, you must use **Ruby 1.8.2 if you want to use RMAGICK** as there is a compatibility issue as of the time of this writing.
2. Install the software to the default location **c:\ruby** and accept all defaults.

Install Rails

1. Open a command prompt
2. Install Rails (**gem install rails –include-dependencies**)
3. Install RedCloth (**gem install redcloth**) and any other gems you might find useful.

Install Rmagick

This section is left in because it may eventually work again. Right now, this version of Rmagick **does not work on Windows with Ruby 1.8.4.**

1. Download the special Windows version of Rmagick from http://rubyforge.org/frs/download.php/6276/RMagick-1.9.2-IM-6.2.4-6-win32.zip
2. Unzip this to a temporary location
3. Open a command prompt and navigate into the extracted location
4. Install the gem
   gem install Rmagick-1.9.2-IM-6.2.4-6-win32.gem
5. Run the postinstall.rb script
   ruby postinstall.rb

Install Mongrel

1. Open a command prompt
2. Install mongrel and dependencies (gem install mongrel –include-dependencies)
3. Choose the **win32** option!

Revision 1.0 – May 4, 2006

Serving a Rails Application

Set up a new Rails app

1. Navigate to your **c:\web** folder
   cd\web
2. Create a new Rails application in that folder (or add your own **working** application)
   rails app2
3. Ensure the Rails application works by testing it with WEBrick. Make sure that the database configuration for production is correct. (Sorry, that's not covered here!)
   cd\web\app2
   ruby script/server –e production –p 4001
4. Stop WEBrick with CTRL+Break

Test with Mongrel

To test with Mongrel, simply execute the command **mongrel_rails start –p 4001.**

If you see that, you're good to go! You can now install this application as a service in Production mode!. Of course, you should test it by navigating to http://localhost:4001/

IIS Integration

Installl ISAPI Rewrite

Visit http://www.isapirewrite.com/ and download the trial version of the ISAPI Rewrite plugin.

- Direct download is http://www.isapirewrite.com/download/isapi_rwf_x86_0060.msi
- Launch the installation program and accept all of the default settings.
- You will be prompted to restart IIS and you should allow this.
- If you experience trouble with the installation, you'll need to refer to the developers of this product.

Fixing the ISAPIRewrite association issue

After installing, it may be necessary to "fix" the association of this filter.

- Open a command prompt
- Navigate to **C:\Program Files\Helicon\ISAPI_Rewrite**
- Launch the ProxyCfg.vbs script
  - Proxycfg.vbs –r or cscript proxycfg.vbs –r

- Restart IIS

Configure ISAPI Rewrite

The last step is to modfy the httpd.ini file which resides in **C:\Program Files\Helicon\ISAPI_Rewrite**

Add this line to the bottom of the file.

```
# Proxy requests to Apache on 8080.

# FOR TEST APPLICATION
RewriteProxy /test(.*) http\://localhost:4001$1 [I,U]
```

Save the file. You should not need to restart IIS.

Testing it out!

If all worked well, you can now pull up your Rails application via IIS by navigating to http://localhost/test/

Unfortunately, it's not going to look very good. Read on to find out why.

Reverse Proxy and URLs

The big problem we're faced with now is that the URLs that Rails creates internally, such as stylesheet links, url_for links and other links don't work as we expect… instead, they direct users around the proxy. This is bad because it exposes the proxied server.

IIS has no method to handle reverse proxying. A reverse proxy rewrites the content served from the backend to mask the fact that the request was filtered through a proxy.

Thankfully, there's a way around this… using a simple Rails plugin that modifieds the way Rails creates its URLs. We're going to make Rails prepend our external URL to any URLs it creates through the system. This will force all user requests to come back through the IIS proxy.

Installing the proxy plugin

Execute the command

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

from within your application's root folder. The plugin should install and then ask you for the base url. Enter **http://localhost/app1** and pres 'enter'. If all goes well, the configuration file will be written. If the configuration file can't be modified, you can navigate to **vendor/plugins/reverse_proxy_fix** and change it yourself.

If the installation fails, you can build the plugin yourself if you follow the next section.

Creating the proxy plugin

If you don't have Subversion installed, you can follow these steps to get the plugin configured properly.

- Create a new Rails plugin called "reverse_proxy_fix"
  ruby/script generate reverse_proxy_fix
- Navigate to your application's vendor/plugins/reverse_proxy_fix folder and edit the init.rb file
  - Add the following code to the file
    Require 'reverse_proxy_fix"
- Edit **vendor/plugins/iis_proxy_fix/lib/reverse_proxy_fix.rb** and replace the contents with the code located in the appendix.
- Modify the first line to match your application's url…
- **BASE_URL = 'http://localhost/app1'**

Using the proxy plugin

Once the plugin is installed, you'll need to restart your Rails application. In this case, you need to restart Mongrel for the changes to take effect.

If all went as expected, any internal links in your application should now be corrected and your users will routed back through the proxy.

The plugin is only active in production mode, so it's safe to keep in your application during the development process.

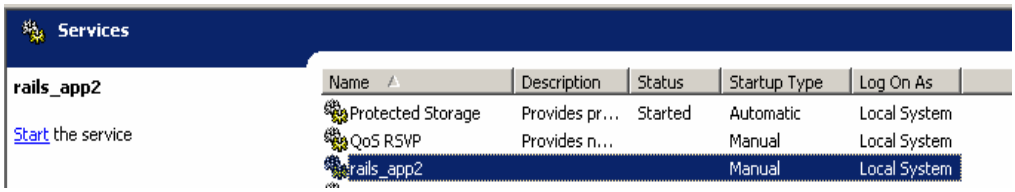
Install Mongrel as a Windows Service

We'll install this application using production mode, so make sure your **database.yml** file points to a working production database if you're using your own application with this tutorial.

1. Make sure Mongrel is stopped by pressing **ctrl+break**. Answer "Yes" to "Terminate Batch Job" if you are prompted.
2. Execute the command below to install the application as a service

   mongrel_rails_service install –n rails_app2 –p 4001

   This will create a new Windows service with the name **rails_app2** which you can view in the Control Panel.



3. You can start the service from Control Panel or you can start it from the command line by executing the command **mongrel_rails_service start –n rails_app2**
   a. To stop the service, you can use **mongrel_rails_service stop –n rails_app2**

   b. If something didn't work right, you can remove the service
      mongrel_rails_service delete –n rails_app2
4. You'll want to set the startup type of the service to **automatic** when you're done so that the service will start when the machine restarts.

Wrapping Up

You now have a working application that should appear to integrate seamlessly with your IIS setup. Adding more applications should be a breeze because you only need to bring up a new instance of Mongrel and apply the reverse_proxy plugin to your Rails application. The Mongrel instance could even reside on another machine.

Requests will always be sent to Mongrel, which means that you won't be able to get the most out of Rails' page caching mechanism. You could solve this by running Lighttpd on Linux in front of Mongrel and proxying requests from IIS to Lighttpd. This will be covered in a future article.

Acknowledgements

- Zed Shaw for developing Mongrel and for pushing me to work out a solution for Windows.
- The Ruby on Rails community for the various postings that led to this article.

Revision 1.0 – May 4, 2006

Appendix

ISAPI Rewrite httpd.ini file

```
[ISAPI_Rewrite]
# 3600 = 1 hour
CacheClockRate 3600

RepeatLimit 32

# Block external access to the httpd.ini and httpd.parse.errors files
RewriteRule /httpd(?:\.ini|\.parse\.errors).* / [F,I,O]
# Block external access to the Helper ISAPI Extension
RewriteRule .*\.isrwhlp / [F,I,O]

# Proxy requests to Apache on 8080.

# TEST APPLICATION
RewriteProxy /test(.*) http\://localhost:4001$1 [I,U]
```

plugin/reverse_proxy_fix/lib/reverse_proxy_fix.rb

```
# Copyright © 2006 Brian Hogan
#
# Permission is hereby granted, free of charge, to any person obtaining
# a copy of this software and associated documentation files (the
# "Software"), to deal in the Software without restriction, including
# without limitation the rights to use, copy, modify, merge, publish,
# distribute, sublicense, and/or sell copies of the Software, and to
# permit persons to whom the Software is furnished to do so, subject to
# the following conditions:
#
# The above copyright notice and this permission notice shall be
# included in all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
# EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
# MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
# NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
# LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
# WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

BASE_URL = ''
module ActionController

  protected
  # Configure the prefix on the url only if we're running in production mode
  # Throws an exception if the BASE_URL constant has not been configured in
  # config.rb
  def self.check_mode_and_base
        if RAILS_ENV == 'production'
                return true
        else
                return false
        end
  end

  # Set the asset host for CSS, JS, and image files if we're in production
  # mode and the base_path has been configured.
  if check_mode_and_base
        ActionController::Base.asset_host = BASE_URL
  end

  # Modifies the original UrlRewriter class, altering how the URLs are created.
  class UrlRewriter

        alias old_rewrite_url rewrite_url

        # Prepends the BASE_URL to all of the URL requests created by the
        # URL rewriter in Rails, stripping off the host, port, etc to ensure that
        # the new URL is exactly what you expect.
        #
        # This method calls check_mode_and_base to ensure that the URL fixing only occurs
        # in production mode and that the BASE_URL variable in config.rb is set.
        def rewrite_url(path, options)
                url = old_rewrite_url(path, options)
                url = url.gsub(@request.protocol + @request.host_with_port, '')
                if ActionController::check_mode_and_base
                        url = BASE_URL + url
                end
                url

        end
  end

end
```