Serving Multiple Rails Applications on Windows with Apache and Mongrel

This article will walk you through the steps of installing Ruby, Gems, Rails, and other important libraries on a Windows XP or 2003 Server using Apache to serve the static content and Mongrel to serve multiple Rails applications.

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

http://myserver.mydomain.com/app1/
http://myserver.mydomain.com/app2/

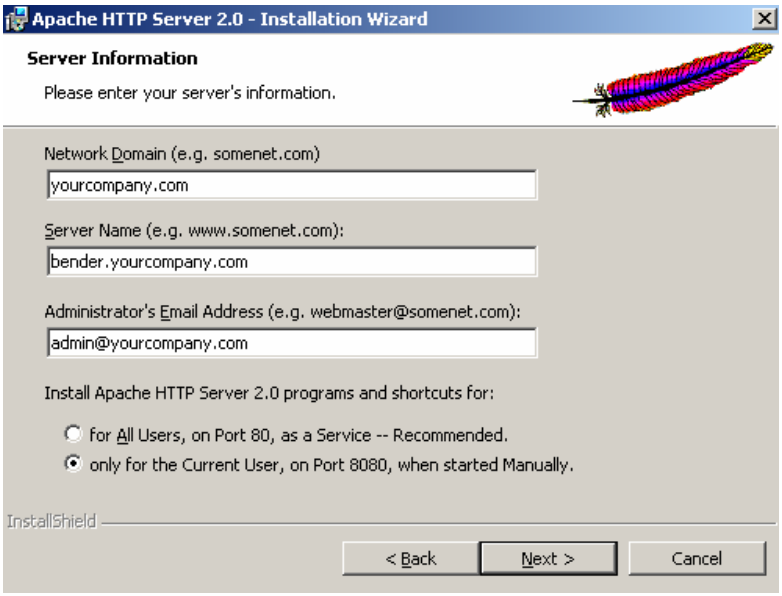
We'll use Apache to proxy requests to WEBrick and Mongrel on higher ports, allowing us to maintain the applications separately, and even move the applications to different servers at some point.

Download and Install Apache

This article will use the 2.0x branch of Apache. At the time of writing, the 2.2x branch is available but there are issues with FastCGI.

1. Download Apache for Windows from here:
   http://apache.cs.utah.edu/httpd/binaries/win32/apache_2.0.55-win32-x86-no_ssl.msi

2. Log in as a member of the **Administrators** group and run the installer.



3. Be sure to select “only for the current user on port 8080”. We’ll install it as a service later but we’ll need to make a lot of configuration changes.

4. Choose **Custom Install** and change the installation folder to **c:\apache** and choose **Next**.

5. The installer will take a minute or so, and you should see a console window appear for some configuration. After this, the installation should complete.

Start Apache as a Service.

1. Open a command prompt and navigate to the folder **c:\apache\apache2\bin**
2. Type **apache –k install** to install Apache as a service.
3. If the Windows Firewall asks you that you need to allow access to Apache, you need to allow it.
4. The service should now appear as a Windows service and should also appear in the Apache monitor.
5. Apache is now running as a service, but it is not started yet. That’s fine because we’ve got a lot more work to do.

Installing Rails

Installing Ruby, Rubygems, and RDoc

1. Download the One-Click Ruby Installer
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**)

Install Rmagick

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!

Configure Apache for Rails Hosting

There are some important steps that need to be performed in order for Apache to start serving out Rails applications efficiently.

Configure Apache

1. Locate the file c:\apache\apache2\conf\httpd.conf and open it in a text editor. For convenience, the Apache installer may have placed an entry in your Start menu for this. You may have to choose to associate this file with an editor. Choose Notepad or a similar text editor.

2. Set Apache to run at port 80 by locating the line **LISTEN** and changing its value to 80
3. Locate the line
   LoadModule alias_module modules/mod_alias.so
   and remove the # character to uncomment the line. We'll need this later!
4. Save the file. **Don't close it!**
5. Start Apache by using the Apache Control or by starting the service from the **Services** panel in Windows.

6. Navigate to http://localhost and ensure that you do have a server running!
7. Shut down the Apache service

Serving a Rails Application

Preparing a Rails application

1. Make a folder on your server to hold your rails application.
   c:\web
2. Copy a working Rails application into that folder
   c:\web\app1
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\app1
   ruby script/server –e production –p 4000

Configure Apache to Proxy a Rails App

1. Create a new file in **c:\apache\apache2\conf** called **http-proxy.conf**
2. At the top of the file, place the following lines:

```
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>
```

   This sets up the proxy modules and sets up a default proxy policy. Consult the documentation for mod_proxy for more information.

3. Now paste the following so that Apache can find your static files. We'll create an alias that will allow Apache to locate our files at /app1 and then we define **c:\web\app1** as a directory that can be served by Apache.

```
Alias /app1 "c:/web/app1/public"
<Directory "C:/web/app1/public">
    Options Indexes FollowSymLinks
    AllowOverride none
    Order allow,deny
    Allow from all
</Directory>
```

4. Now apply the crucial proxy rules so that Apache can find your application!

```
ProxyPass /app1/images !
ProxyPass /app1/stylesheets !
ProxyPass /app1/javascripts !

ProxyPass /app1/ http://127.0.0.1:4000/
```

```
ProxyPass /app1 http://127.0.0.1:4000/
ProxyPassReverse /app1/ http://127.0.0.1:4000/
```

The first three lines tell Apache **not** to proxy these folders. That way, your images, stylesheets, and javascript files will be served by Apache, improving speed slightly.

The next three lines actually send the request. The line
`ProxyPass /app1 http://127.0.0.1:4000/`
Is actually there to prevent people from viewing the contents of the **public** folder which contains the dispatcher.

Link the configurations

1. Open c:\apache\apache2\conf\httpd.conf (or go back to it if you left it open)
2. At the very bottom of the file, add the following line so that your new proxy configuration file is added when Apache starts.

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

Test the connection

You should still have your Rails application running on http://localhost:4000/ and you should now have an Apache instance running at http://localhost/ so you're ready to see if the proxy actually works.

Navigate to http://localhost/app1/ and you should see your Rails app saying hello to you. However, it might not look quite right and you may have tons of broken links. 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.

The common approach to this is to use some tool to parse the HTML response from the proxy and rewrite the links appropriately. However, that may be more difficult on Windows because you'd need to find a compiled version of **mod_proxy_html** for your specific version of Apache.

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'**
- Finally, restart your Rails application by shutting down WEBrick and restarting it
- If all went as expected, any internal links in your application should now be corrected and routed back through the proxy.

Using Mongrel instead of WEBrick

We've tested our setup with the basic WEBrick server which would work fine for small-scale applications that don't serve many users. But now let's step up and add Mongrel to the mix. We'll do that by actually adding a **second Rails app** to the mix.

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.**

```
** Starting Mongrel in development mode at 0.0.0.0:4001
** Starting Rails in environment development ...
** Rails loaded.
** Loading any Rails specific GemPlugins
** WARNING:  Rails does not support signals on Win32.
** Running 0.0.0.0:4001 listener.
** Mongrel available at 0.0.0.0:4001
```

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/

Install the Reverse Proxy plugin to this new app

Now that we know our app works with Mongrel, we need to apply our reverse_proxy_fix plugin from our first application or just recreate it in this application. Refer to the previous section on creating the plugin.

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.

Modify the Apache Proxy configuration

Add a new section to your **httpd-proxy.conf** file for this new application. You can use the finished httpd-proxy.conf file in the Appendix as a reference.

```
Alias /app2 "c:/web/app2/public"
<Directory "C:/web/app2/public">
    Options Indexes FollowSymLinks
    AllowOverride none
    Order allow,deny
    Allow from all
</Directory>

ProxyPass /app2/images !
ProxyPass /app2/stylesheets !
ProxyPass /app2/javascripts !

ProxyPass /app2/ http://127.0.0.1:4001/
ProxyPass /app2 http://127.0.0.1:4001/
ProxyPassReverse /app2/ http://127.0.0.1:4001/
```

Save the file and restart your Apache server. Navigate to http://localhost/app2/ and you should see your application working properly.

Wrapping Up

You should now have two applications running on higher ports proxied through Apache. Once you have it set up, it pretty much runs without a problem. However, you might want to consider running your own benchmarks, as performance on Windows tends to be slow.

Also, you'll notice that the method outlined here does not allow Apache to cache pages created by Rails. Requests will always be sent to Mongrel. To solve this, you could use Apache as the proxy to a Lighttpd instance on a higher port that in turn invokes Mongrel. I think that's too many layers and isn't necessary, as Mongrel seems to do reasonably well when serving static content. You could use rewrite rules to handle some of these issues as well, but that's another topic.

Acknowledgements

- Zed Shaw for developing Mongrel and for giving me a hard time about Rails on Windows.
- Gael Martin for Rails Prod (http://sourceforge.net/projects/rails-prod-win/) which gave me some insight into how proxying should work.
- The Ruby on Rails community for the various postings that led to this article.

Appendix

Apache httpd.conf

```
ServerRoot "C:/apache/Apache2"

PidFile logs/httpd.pid
Timeout 300
```

```
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15

<IfModule mpm_winnt.c>
        ThreadsPerChild 250
        MaxRequestsPerChild  0
</IfModule>

Listen 80

LoadModule access_module modules/mod_access.so
LoadModule actions_module modules/mod_actions.so
LoadModule alias_module modules/mod_alias.so
LoadModule asis_module modules/mod_asis.so
LoadModule auth_module modules/mod_auth.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule cgi_module modules/mod_cgi.so
LoadModule dir_module modules/mod_dir.so
LoadModule env_module modules/mod_env.so
LoadModule imap_module modules/mod_imap.so
LoadModule include_module modules/mod_include.so
LoadModule isapi_module modules/mod_isapi.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule mime_module modules/mod_mime.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule userdir_module modules/mod_userdir.so
#LoadModule ssl_module modules/mod_ssl.so


ServerAdmin myserver.mydomain.com

ServerName myserver.mydomain.com:80

UseCanonicalName Off

DocumentRoot "C:/apache/Apache2/htdocs"

<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

<Directory "C:/apache/Apache2/htdocs">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

DirectoryIndex index.html index.html.var
AccessFileName .htaccess

<FilesMatch "^\.ht">
    Order allow,deny
    Deny from all
</FilesMatch>

TypesConfig conf/mime.types
DefaultType text/plain

<IfModule mod_mime_magic.c>
    MIMEMagicFile conf/magic
</IfModule>

HostnameLookups Off

ErrorLog logs/error.log
LogLevel warn
```

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
CustomLog logs/access.log common

ServerTokens Full
ServerSignature On

Alias /icons/ "C:/apache/Apache2/icons/"

<Directory "C:/apache/Apache2/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

AliasMatch ^/manual(?:/(?:de|en|es|fr|ja|ko|ru))?(/.*)?$ "C:/apache/Apache2/manual$1"

<Directory "C:/apache/Apache2/manual">
    Options Indexes
    AllowOverride None
    Order allow,deny
    Allow from all

    <Files *.html>
        SetHandler type-map
    </Files>

   SetEnvIf Request_URI ^/manual/(de|en|es|fr|ja|ko|ru)/ prefer-language=$1
   RedirectMatch 301 ^/manual(?:/(de|en|es|fr|ja|ko|ru)){2,}(/.*)?$ /manual/$1$2
</Directory>

ScriptAlias /cgi-bin/ "C:/apache/Apache2/cgi-bin/"

<Directory "C:/apache/Apache2/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>


IndexOptions FancyIndexing VersionSort

AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core
AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
```

```
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

DefaultIcon /icons/unknown.gif

ReadmeName README.html
HeaderName HEADER.html

IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t

AddLanguage ca .ca
AddLanguage cs .cz .cs
AddLanguage da .dk
AddLanguage de .de
AddLanguage el .el
AddLanguage en .en
AddLanguage eo .eo
AddLanguage es .es
AddLanguage et .et
AddLanguage fr .fr
AddLanguage he .he
AddLanguage hr .hr
AddLanguage it .it
AddLanguage ja .ja
AddLanguage ko .ko
AddLanguage ltz .ltz
AddLanguage nl .nl
AddLanguage nn .nn
AddLanguage no .no
AddLanguage pl .po
AddLanguage pt .pt
AddLanguage pt-BR .pt-br
AddLanguage ru .ru
AddLanguage sv .sv
AddLanguage zh-CN .zh-cn
AddLanguage zh-TW .zh-tw

LanguagePriority en ca cs da de el eo es et fr he hr it ja ko ltz nl nn no pl pt pt-BR ru
sv zh-CN zh-TW

ForceLanguagePriority Prefer Fallback

AddCharset ISO-8859-1  .iso8859-1 .latin1
AddCharset ISO-8859-2  .iso8859-2 .latin2 .cen
AddCharset ISO-8859-3  .iso8859-3 .latin3
AddCharset ISO-8859-4  .iso8859-4 .latin4
AddCharset ISO-8859-5  .iso8859-5 .latin5 .cyr .iso-ru
AddCharset ISO-8859-6  .iso8859-6 .latin6 .arb
AddCharset ISO-8859-7  .iso8859-7 .latin7 .grk
AddCharset ISO-8859-8  .iso8859-8 .latin8 .heb
AddCharset ISO-8859-9  .iso8859-9 .latin9 .trk
AddCharset ISO-2022-JP .iso2022-jp .jis
AddCharset ISO-2022-KR .iso2022-kr .kis
AddCharset ISO-2022-CN .iso2022-cn .cis
AddCharset Big5        .Big5       .big5
AddCharset WINDOWS-1251 .cp-1251   .win-1251
AddCharset CP866       .cp866
AddCharset KOI8-r      .koi8-r .koi8-ru
AddCharset KOI8-ru     .koi8-uk .ua
AddCharset ISO-10646-UCS-2 .ucs2
AddCharset ISO-10646-UCS-4 .ucs4
AddCharset UTF-8       .utf8
AddCharset GB2312      .gb2312 .gb
AddCharset utf-7       .utf7
AddCharset utf-8       .utf8
AddCharset big5        .big5 .b5
AddCharset EUC-TW      .euc-tw
AddCharset EUC-JP      .euc-jp
AddCharset EUC-KR      .euc-kr
AddCharset shift_jis   .sjis
```

```
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz

AddHandler type-map var

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
BrowserMatch "Microsoft Data Access Internet Publishing Provider" redirect-carefully
BrowserMatch "MS FrontPage" redirect-carefully
BrowserMatch "^WebDrive" redirect-carefully
BrowserMatch "^WebDAVFS/1.[0123]" redirect-carefully
BrowserMatch "^gnome-vfs" redirect-carefully
BrowserMatch "^XML Spy" redirect-carefully
BrowserMatch "^Dreamweaver-WebDAV-SCM1" redirect-carefully


Include conf/httpd-proxy.conf
```

Apache conf/http-proxy.conf

```
##################### MONGREL ########################
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 /app1 "c:/web/app1/public"
<Directory "C:/web/app1/public">
    Options Indexes FollowSymLinks
    AllowOverride none
    Order allow,deny
    Allow from all
</Directory>



ProxyPass /app1/images !
ProxyPass /app1/stylesheets !
ProxyPass /app1/javascripts !

ProxyPass /app1/ http://127.0.0.1:4000/
ProxyPass /app1 http://127.0.0.1:4000/
ProxyPassReverse /app1/ http://127.0.0.1:4000/


################################

Alias /app2 "c:/web/app2/public"
<Directory "C:/web/app2/public">
    Options Indexes FollowSymLinks
    AllowOverride none
    Order allow,deny
    Allow from all
</Directory>
```

```
ProxyPass /app2/images !
ProxyPass /app2/stylesheets !
ProxyPass /app2/javascripts !

ProxyPass /app2/ http://127.0.0.1:4001/
ProxyPass /app2 http://127.0.0.1:4001/
ProxyPassReverse /app2/ http://127.0.0.1:4001/



######################################################
```

plugin/reverse_proxy_fix/lib/reverse_proxy_fix.rb

```
BASE_URL = ''
module ActionController
ActionController::Base.asset_host = BASE_URL

   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.
     def rewrite_url(path, options)
       url = old_rewrite_url(path, options)
       url = url.gsub(@request.protocol + @request.host_with_port, '')
       url = BASE_URL + url
       url
     end
   end
end
```