Using StrixDB with httpd (Apache HTTP Server)

Apache httpd module for SPARQL and SPARQL/Update

Overview

This document explains how to configure **httpd** (Apache HTTP Server) to use the StrixDB RDF store. Configured Apache Server could :

- download and upload RDF graphs into the RDF store (in XML or turtle format),
- respond to **SPARQL** and **SPARQL/Update** HTTP requests with the SPARQL protocol
- respond to HTTP requests with dynamic Lua pages
  - created from **Lua** scripts or
  - created from files with embedded **Lua** code.

Status and License

Current version of **StrixStore** is 0.92
The current version is a beta release and could be used free of charges for any purpose. The current version is ruled by the terms of the **StrixDB license** for beta releases.

Lua API is not final: it could be extended.

History

- 14-07-2010 Initial release v 0.9
- 02-09-2010 release 0.91: improved XML/RDF and RFC 2396 compliance
- 06-11-2010 release 0.92: could be used with **APACHE http server**. StrixServer no more maintained.

Download and installation (for Windows)

Install an **httpd** (Apache HTTP Server) of version **2.2**. The standard Windows installation at **http://httpd.apache.org/** could be used.

Download a **StrixStore** from **http://www.strixDB.com/download.html** (minimum version **0.92**). Installation is easy : just unzip files into a folder.

Copy the file **mod_strixdb.so** into the modules folder of Apache Server (with standard installation, this folder is *C:\Program Files\Apache Software Foundation\Apache2.2\modules*)

Configuration

You have to modify the Apache server configuration file : the default configuration file with standard installation is ***C:/Program Files/Apache Software Foundation/Apache2.2/conf/httpd.conf***

Example of configuration :

```
LoadModule strixdb_module  modules/mod_strixdb.so

<IfModule strixdb_module>
StrixRoot  "C:/Program Files/StrixDB/"
StrixFilename     "D:/RDF/strix.db"
StrixInitfile     "D:/RDF/initDatabase.lua"
StrixDefaultURI "http://mydefault/graph/uri/"
StrixTruncate     true
</IfModule>

<Location /strixdb>
   SetHandler strix-db-handler
</Location>

AddHandler strix-luapage-handler .hlua
AddHandler strix-lua-handler .lua
```

Explanations :

```
LoadModule strixdb_module modules/mod_strixdb.so
```

REQUIRED: load the strixdb module

```
StrixRoot "C:/Program Files/StrixDB/"
```

REQUIRED: defines location of StrixDB installation (where are StrixStore.dll, etc...)

```
StrixFilename     "D:/RDF/strix.db"
```

Defines the path of the persistence file used to store graphs.
NOTES: this parameter is not required (by default it is "./strix.db"). If not specified, file will be located in the directory of Apache httpd.

```
StrixInitfile     "D:/RDF/initDatabase.lua"
```

Use this parameter to specify a file to run at creation (first opening) of the database file (specified with parameter **StrixFilename**).

```
StrixDefaultURI "http://www.myfoaf.com/"
```

Defines the default graph URI. If not specified, use URI returned by **gethostname** C function.

```
StrixTruncate     true
```

If this parameter is set to **true**, the database file is deleted at the server start.

```
AddHandler strix-luapage-handler .hlua
```

Associate an extension file (here .hlua) to the dynamic pages (containing embedded lua code inside **<?lua ?>** tags).

```
AddHandler strix-lua-handler .lua
```

Associate an extension file (here .lua) to full interpreted Lua pages.

Advanced parameters (be carefull)

StrixPoolsize <integer>	the number of pages cached in the poolSize. Each page has 4K. Big poolSize improves speed but consumes	default=100*1024

	memory.	
StrixInitindex <integer>	Size of the inital index (bitmap).	default=8*1024*1024
StrixQuantum <integer>	Size of the new allocated quantum (bitmap) when the allocated file is full.	default=512*1024*1024
StrixSafe <boolean>		default=false
StrixNobuffer <boolean>		default=false
StrixWritethrough <boolean>	if true, wait disk write acknowledge event for each write transaction (safer but slower with SPARQL/Update transactions).	default=false

---

Dynamic Lua Pages

With StrixDB, Lua can be used as a nice alternative to php. We propose 2 uses of Lua:

- as script generating the HTTP response with *print* function.
- as embedded Lua code inside a file (code is inside **<?lua ?>** tags).

First we have to configure Apache httpd.conf :

```
AddHandler strix-luapage-handler .hlua
AddHandler strix-lua-handler .lua
```

Explanations: the **strix-luapage-handler** handler associate file with a **.hlua** extension with the embedded Lua code execution. The **strix-lua-handler** handler associate **.lua** extension with the script generation. You could change the extensions as needed. These association are available for all Apache accessible files.

To test, just copy the folder tests/*scripts* from the StrixDB distribution into your **DocumentRoot** folder specified in *httpd.conf*.

Embedded Lua code

Lua code must be included inside **<?lua ?>** . See **test.hlua** for an example. The MIME type of the file is set by the Lua function *apache.setContent*

```
apache.setContent('text/html')
```

In the sample **testCreatedRdf.hlua**, the content type is set to **application/rdf+xml; charset=utf-8** so that the browser will interpret the response as RDF/XML.

Script generated responses

In this case, all the file is generated by the Lua script. The sample **test.lua** shows how to generate an HTML file. The sample **testCreateRdf.lua** shows how to generate a RDF/turtle file.

As for embedded code, the Lua function *apache.setContent* set the MIME type of the response.

Lua functions reference

With dynamic Lua pages, a Lua table named '**apache**' make the bindings with apache.

apache.host

returns the hostname of the request.

apache.filename

returns the filename requested (this file is the current file executed as a script for **strix-lua-handler** handler and sended for the **strix-luapage-handler** handler)

apache.method

"GET" for a HTTP GET request, "POST" for a HTTP POST request.

apache.uri

the uri of the request.

apache.authentication

returns 2 values : the user and authentication type.

apache.root ()

returns the root folder of Apache (specified with **DocumentRoot** in the Apache configuration file).

apache.datas ()

returns a table of the content datas (only avalaible with a POST request).

apache.headers ()

returns a table of the apache headers_i.

apache.args ()

returns a table of the arguments of the request. **NOTE**: with a POST request, use **apache.datas()** to get the content.

apache.setStatus (<int>)

Set the status of the response (for example, **apache.setStatus(404)** ....)

apache.setContent (<string>)

Set the MIME type of the response.