Lua FreeSWITCH Dbh

About

The FreeSWITCH Database Handler (freeswitch_Db) allows you to connect to databases from your Lua script. The advantage of this method is that it makes use of connection pooling provided by FreeSWITCH which gives a nice increase in speed when compared to creating a new TCP connection for each LuaSQL env:connect().

Here are some examples for using freeswitch_Db in a mod_lua script.

---

Example Dialplan App

Here is a simple way to get a few variables from your db and set them as channel variables:

Assume you have a table did_users in your db with only 2 columns: did and user with a unique key over both.

**XML Dialplan**

```
<extension name="map_did_to_user" continue="true">
  <condition field="destination_number" expression="^\+\d+$"> 
    <action inline="true" application="lua" data="map_did_to_user.lua $2"/>
  </condition>
</extension>

<extension name="if_user_then_">
  <condition field="${user}" expression="...
``` 

Note that the lua script must be run inline, so that the retrieved channel variables are available immediately.

**Lua Script**

```
Map DID to user

-- map_did_to_user.lua
-- takes DID as first argument

local did = argv[1]

local dbh = freeswitch_Db("dbname://datasourcename:username:password")

local function set_session_variables(row)
    -- Sets session variables with the same names as the columns from the database
    for key, val in pairs(row) do
        if session then
            session:setVariable(key, val)
        end
        freeswitch.consoleLog("DEBUG", string.format("set(%s=%s)\n", key, val))
    end
end

assert(dbh:connected())

local sql_query = "SELECT user FROM did_users WHERE did = ". did

assert(dbh:query(sql_query, set_session_variables))
```
The script checks if a session is available. If it is, the key/value will be set as a channel variable. The result will always be printed to your console in the debug level.

Example User Directory XML

The following script is a simple way to use an SQL query to generate User Directory XML on the fly inside FreeSWITCH.

It's assumed you have a table named `users` in your database containing the following columns: `domain`, `id`, `mailbox`, `number-alias`, `password`, `dial-string` and `user_context` (all `varchar`s).

Lua Configuration

You can enable sending XML directory lookups through Lua by adding the following lines to your `lua.conf.xml`.

```xml
<param name="xml-handler-script" value="gen_dir_user_xml.lua"/>
<param name="xml-handler-bindings" value="directory"/>
```

Lua Script

Based on the mailing list: [http://lists.freeswitch.org/pipermail/freeswitch-users/2012-January/079296.html](http://lists.freeswitch.org/pipermail/freeswitch-users/2012-January/079296.html) there are problems with Voicemail Inject when using Lua to serve configs (as well as personal experience), i will elaborate on this more in depth - Destreyf
MySQL Example

```lua
-- gen_dir_user_xml.lua
-- example script for generating user directory XML

-- comment the following line for production:
freeswitch.consoleLog("notice", "Debug from gen_dir_user_xml.lua, provided params:
.. params:serialize() .. 
"n")

local req_domain = params:getHeader("domain")
local req_key    = params:getHeader("key")
local req_user   = params:getHeader("user")

assert (req_domain and req_key and req_user,
"This example script only supports generating directory xml for a single user !\n")

local dbh = freeswitch.Dbh("odbc://datasourcename:username:password")
if dbh:connected() == false then
    freeswitch.consoleLog("notice", "gen_dir_user_xml.lua cannot connect to database" .. dsn .. 
"\n")
    return
end

-- it's probably wise to sanitize input to avoid SQL injections!
local my_query = string.format("select * from users where domain = '%s' and '%%s'=%%%s' limit 1",
    req_domain, req_key, req_user)

assert (dbh:query(my_query, function(u) -- there will be only 0 or 1 iteration (limit 1)
XML_STRING =
[<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="freeswitch/xml">
  <section name="directory">
    <domain name="%s">
      <user id="%s" mailbox="%s" cidr="%s" number-alias="%s">
        <params>
          <param name="password" value="%s"/>
          <param name="dial-string" value="%s"/>
        </params>
        <variables>
          <variable name="user_context" value="%s"/>
        </variables>
    </user>
  </domain>
</section>
</document>]

-- comment the following line for production:
freeswitch.consoleLog("notice", "Debug from gen_dir_user_xml.lua, generated XML:
.. XML_STRING .. 
"n")
end))
```

Note that the query in the example is formatted for MySQL; use double-quotes (") instead of backticks (`) for PostgreSQL.

db_connect.lua rewrite

Chapter 7 of the FreeSWITCH book explains a db_connect.lua script. This is a rewrite not using LuaSQL. I used a simple MS-Access database (freeswitch.accdb) to test the script.

```lua
-- db_connect.lua
-- Connects to a database using freeswitch.Dbh connection pooling, checks PIN, reads balance
-- A hangup function makes the code a bit cleaner

local dbh = freeswitch.Dbh("odbc://datasourcename:username:password")
local row = {}
function hangup_call ()
    session:streamFile("ivr/ivr-thank_you.wav")
    session:sleep(250)
    session:streamFile("voicemail/vm-goodbye.wav")
    session:hangup()
end

ifdbh:connected() == false then
    freeswitch.consoleLog("notice", "gen_dir_user_xml.lua cannot connect to database" .. dsn .. ":\n")
    hangup_call()
end

-- Set invalid entry file
invalid = "ivr/ivr-that_was_an_invalid_entry.wav"

-- Greet caller
session:answer()
session:streamFile("ivr/ivr-hello.wav")
tries = 0

while (tries < 3) do
    -- Collect account number
    acct = session:playAndGetDigits(3, 5, 3, 7000, "+", "phrase:enter_message_number", invalid, "+")
    if (acct) then
        -- Pull account from database -> assumes that acct is unique
        my_query = "select * from users where acct = "+ acct
        assert(dbh:query(my_query, function(qrow)
            for key, val in pairs(qrow) do
                row[key] = val
            end
        end))
    end
    -- Confirm that we received the record
    if (row.pin == nil) then
        -- We have an account, now collect PIN and check
        tries = 0
        while (tries < 3) do
            pin = session:playAndGetDigits(3, 5, 3, 7000, "+", "ivr/ivr-please_enter_pin_followed_by_pound.wav", invalid, "+")
            if (row.pin == pin) then
                user_repeat = true
                while(user_repeat == true) do
                    session:streamFile("voicemail/vm-you_have.wav")
                    session:sleep(200)
                    session:say(row.balance, "en", "currency", "pronounced")
                    session:sleep(200)
                    -- "to repeat these options, please press 1"
                    digits = session:playAndGetDigits(1, 1, 3, 7000, "+", "ivr/ivr-to_repeat_these_options.wav!ivr/ivr-please.wav!voicemail/vm-press.wav!digits/1.wav!
                        "file_string://ivr/ivr-to_repeat_these_options.wav!ivr/ivr-please.wav!voicemail/vm-press.wav!digits/1.wav",
                        invalid,"\d+")
                    -- repeat y/n
                    freeswitch.consoleLog("INFO", "User entered " .. digits .. "\n")
                    if (digits == "1") then
                        user_repeat = true
                    else
                        hangup_call()
                        break
                    end
                end
            else
                -- Callerr entered wrong PIN
                session:streamFile("ivr/ivr-that_was_an_invalid_entry.wav")
                tries = tries + 1;
            end
        end
    end
end

if (tries > 2) then
    -- Too many failed attempts to enter PIN
    session:streamFile("voicemail/vm-abort.wav")
    hangup_call()
    break
else
-- We did not find this account
session:streamFile(invalid)
tries = tries + 1;
end
end -- while (tries < 3)

if (tries > 2) then
    session:streamFile("voicemail/vm-abort.wav")
hangup_call()
end

See Also

Lua Database