
目錄

序	1.1
1 luastar简介	1.2
2 luastar安装	1.3
3 api开发	1.4
3.1 luastar结构	1.4.1
3.2 luastar全局变量	1.4.2
3.3 缓存	1.4.3
3.4 luastar_context上下文	1.4.4
3.5 调试与日志	1.4.5
3.6 应用配置	1.4.6
3.7 路由和拦截器	1.4.7
3.8 bean管理	1.4.8
3.9 mysql和redis使用	1.4.9
4 web开发	1.5
4.1 session管理	1.5.1
4.2 页面布局与渲染	1.5.2
9 联系方式	1.6

luastar项目地址：

<https://github.com/luastar/luastar>

luastar简介

luastar是一个基于openresty的高性能高并发开发框架，支持http接口和web开发

luastar使用openresty-1.7.10.2在macOS和centos6.5系统上测试过。

luastar主要特性：

1. request/response封装
2. 缓存管理
3. 配置文件管理
4. 路由/拦截器配置
5. 类似spring bean管理
6. mysql和redis访问封装
7. httpclient等常用工具封装
8. web支持session和页面模板

openresty 安装

请参考官网介绍，<https://openresty.org/cn/installation.html>

建议安装目录：/usr/local/openresty

luastar 安装

1 下载luastar

从[github](#)下载luastar到磁盘上，例如：/data/apps/luastar下。

2 修改luastar配置

替换配置文件『/yourpath/luastar/conf/luastar*.conf』中的openresty安装路径和luastar存放路径，如下：

```
# 设置lua包路径(';;'是默认路径，?.dylib是macos上的库，?.so是centos上的库，ZeroBraneStudio用于调试):
lua_package_path '/Users/zhuminhua/Documents/work-private/luastar/luastar/libs/?.lua;/Users/zhuminhua/Documents/work-private/luastar/luastar/src/?.lua;/Applications/ZeroBraneStudio.app/Contents/ZeroBraneStudio/lualibs/?.lua;/Applications/ZeroBraneStudio.app/Contents/ZeroBraneStudio/lualibs/?.lua;;';
lua_package_cpath '/Users/zhuminhua/Documents/work-private/luastar/luastar/libs/?.dylib;/Users/zhuminhua/Documents/work-private/luastar/luastar/libs/?.so;/Applications/ZeroBraneStudio.app/Contents/ZeroBraneStudio/bin/clibs/?.dylib;;';

#luastar初始化
init_by_lua_file '/Users/zhuminhua/Documents/work-private/luastar/luastar/src/luastar_init.lua';

#设置成一样避免获取request_body时可能会缓存到临时文件
#client_max_body_size 50m;
```

```
#client_body_buffer_size 50m;

server {
    listen 8001;
    #关闭后不用重启nginx即可访问最新代码，生产环境一定要打开
    lua_code_cache off;
    server_name localhost;
    #luastar路径
    set $LUASTAR_PATH '/Users/zhuminghua/Documents/work-private/luastar/luastar';
    #应用名称
    set $APP_NAME 'demo';
    #应用路径
    set $APP_PATH '/Users/zhuminghua/Documents/work-private/luastar/demo';
    #应用使用的配置，可区分开发/生产环境，默认使用app.lua
    set $APP_CONFIG '/config/app_dev.lua';
    #访问日志
    access_log '/Users/zhuminghua/logs/nginx/demo/access.log' main;
    #错误/输出日志
    error_log '/Users/zhuminghua/logs/nginx/demo/error.log' info;
    location / {
        default_type text/html;
        content_by_lua_file '${LUASTAR_PATH}/src/luastar_content.lua';
    }
}

server {
    listen 8002;
    #web项目关闭lua_code_cache后session会失效
    #lua_code_cache off;
    server_name localhost;
    #luastar路径
    set $LUASTAR_PATH '/Users/zhuminghua/Documents/work-private/luastar/luastar';
    #应用名称
    set $APP_NAME 'demo2';
```

```
#应用路径
set $APP_PATH '/Users/zhuminhua/Documents/work-private/luastar/demo2';
#应用使用的配置，可区分开发/生产环境，默认使用app.lua
set $APP_CONFIG '/config/app_dev.lua';
#template模板跟路径，web项目使用
set $template_root '/Users/zhuminhua/Documents/work-private/luastar/demo2/views';
#访问日志
access_log '/Users/zhuminhua/logs/nginx/demo2/access.log' main;
#错误/输出日志
error_log '/Users/zhuminhua/logs/nginx/demo2/error.log' info;
location / {
    default_type text/html;
    content_by_lua_file '${LUASTAR_PATH}/src/luastar_content.lua';
}
#静态文件目录(*.js, *.css...)
location /assets {
    root '/Users/zhuminhua/Documents/work-private/luastar/demo2';
    index index.html index.htm;
}
}
```

luastar/conf/目录下多个文件分别对应不同环境，例如luastar_dev.conf是开发环境的配置，luastar.conf是生产环境的配置

3 修改nginx配置

修改openresty/nginx/conf/nginx.conf，引入luastar项目配置文件：

```
include /Users/zhuminhua/Documents/work-private/luastar/luastar/conf/luastar_dev.conf;
```

4 启动nignx

启动openresty : `openresty/nginx/sbin/nginx -c openresty/nginx/conf/nginx.conf`

测试访问

<http://localhost:8001/api/test/hello>

<http://localhost:8001/api/test/hello?name=haha>

luastar项目结构如下：

/data/apps/（项目存放路径）

```
luastar (luastar)
  conf (nignx配置文件)
  libs (第三方库)
  src (luastar源码)
demo (api应用)
  config (配置目录)
    app*.lua (配置文件)
    bean.lua (bean配置)
    msg.lua (文案配置)
    route.lua (路由/拦截器配置)
  src
    com
      luastar
        demo (包名)
          ctrl (控制类目录)
          interceptor (拦截器)
          service (服务类)
          util (辅助类)
demo2 (web应用)
  config (配置目录)
  src (源码目录)
  views (template视图文件目录)
  assets (静态文件目录)
```

luastar在初始化时，定义了几个常用的全局变量，在项目中可以直接使用，不用require引入。

详情请参看：[luastar/src/luastar_init.lua](#)

全局变量	说明
Class	luastar中的类定义
cjson	json工具类
_	moses工具类（修改过）
template	html模板类
luastar_cache	luastar缓存
luastar_config	luastar配置
luastar_msg	luastar文案
luastar_context	luastar上下文
logger	日志辅助
session	web session

第三方库

请参考[luastar/libs/](#)，文件头注释中都有源库的引用地址

luastar提供了lua内存缓存，根据openresty机制，每个worker存有一份，所以在使用缓存前，需要先判断是否存在（即使初始化存储过），luastar中使用缓存存储了配置文件信息、bean信息、路由和拦截器信息等等

例如：luastar/src/core/config.lua

```
local _M = {}

local util_file = require("luastar.util.file")

function _M.getConfig(k, default_v)
    -- 从缓存中获取配置信息
    local app_config = luastar_cache.get("app_config")
    if app_config then
        -- 如果配置信息存在，返回
        return app_config[k] or default_v
    end
    -- 如果配置信息不存在，初始化
    ngx.log(ngx.INFO, "init app config.")
    -- 加载配置文件
    local app_config_file = ngx.var.APP_CONFIG or "/config/app.lua"
    local config_file = ngx.var.APP_PATH .. app_config_file
    app_config = util_file.loadlua_nested(config_file) or {}
    -- 缓存配置信息
    luastar_cache.set("app_config", app_config)
    -- 返回结果
    return app_config[k] or default_v
end

return _M
```

说明：内存缓存的好处在于支持所有的lua结构，没有限制。

如果需要缓存的内容比较简单或者可以序列化成json，可以考虑使用[ngx.shared.DICT](#)，在整个nginx共享。

有些内容在init_by_lua阶段无法初始化，需要延后在执行content_by_lua阶段执行，无法放到初始化阶段里作为全局变量直接使用，所以放到了上下文中，luastar主要从上下文中做了以下三件事：

1、初始化项目包路径，

2、获取route

3、获取beanFactory

```
local _M = {}

local src_path = "/src/?.lua"
local route_file = "/config/route.lua"
local bean_file = "/config/bean.lua"

function _M.init_pkg_path()
    local pkg_path_init = luastar_cache.get("pkg_path_init")
    if pkg_path_init then
        return
    end
    package.path = package.path .. ";" .. ngx.var.APP_PATH .. src_path
    luastar_cache.set("pkg_path_init", true)
end

function _M.getRoute()
    local route = luastar_cache.get("route")
    if route then
        return route
    end
    local Route = require("luastar.core.route")
    route = Route(ngx.var.APP_PATH .. route_file)
    luastar_cache.set("route", route)
    return route
end

function _M.getBeanFactory()
    local beanFactory = luastar_cache.get("beanFactory")
    if beanFactory then
```

```
        return beanFactory
    end
    local BeanFactory = require("luastar.core.beanfactory")
    beanFactory = BeanFactory(ngx.var.APP_PATH .. bean_file)
    luastar_cache.set("beanFactory", beanFactory)
    return beanFactory
end

return _M
```

前两项在luastar_content.lua中使用了，可以不用关心，在项目中使用最多的就是从中获取beanFactory以及bean

```
local beanFactory = luastar_context.getBeanFactory()
local mysql_util = beanFactory:getBean("mysql")
local mysql = mysql_util:getConnect()
local res, err, errno, sqlstate = mysql:query(sql)
ngx.log(logger.i(cjson.encode({
    sql = sql,
    res = res,
    err = err,
    errno = errno,
    sqlstate = sqlstate
})))
mysql_util:close(mysql)
```

调试

luastar/openresty可以利用[ZeroBraneStudio](#)工具调用，这个工具非常强大，唯一不足的是在macos上输入中文有问题。

openresty使用ZeroBraneStudio调试步骤可参考链

接：<http://notebook.kulchenko.com/zerobrane/debugging-openresty-nginx-lua-scripts-with-zerobrane-studio>

luastar使用ZeroBranStudio调试步骤如下：

1、在包路径中增加ZeroBranStudio相关库文件，注意macos使用.dylib，centos上使用.so库

```
lua_package_path 'luastar其他库;/Applications/ZeroBraneStudio.app/Contents/ZeroBraneStudio/lualibs/?.lua;/Applications/ZeroBraneStudio.app/Contents/ZeroBraneStudio/lualibs/?.lua;';  
lua_package_cpath 'luastar其他库;/Applications/ZeroBraneStudio.app/Contents/ZeroBraneStudio/bin/clibs/?.dylib;';
```

2、在需要调试的代码前后加上

```
require('mobdebug').start("127.0.0.1")  
-- 调试代码  
require('mobdebug').done()
```

3、断点，按ZeroBranStudio方法启动调试

日志

如果觉得调试起来麻烦，日志就是最好的调试办法，简单高效（熟练后完全可以不需要调试）。

luastar直接使用ngx.log输出，之前也有用过第三方库

<https://github.com/Neopallium/lualogging> 在多worker模式中容易造成日志丢失。

ngx.log的缺点是不能个性化按天输出（可以用脚本定时分割），输出大小有限制，

不过一般也够用了。

luastar只是简单封装了固定输出request_id和简化的方法，不包装起来是为了直观的输出日志的位置

```
ngx.log(logger.info("name=", name))  
-- 或者  
ngx.log(logger.i("name=", name))
```

输出结果：2y6hNDFGd4Nxi7FE9UAP是本次请求的request_id，便于在日志量大的情况下定位一次请求的所有日志。

```
2016/12/19 17:01:50 [info] 14545#0: *553 [lua] hello.lua:12: --[  
2y6hNDFGd4Nxi7FE9UAP]--name=world, try to give a param with name  
., client: 127.0.0.1, server: localhost, request: "GET /api/test  
/hello HTTP/1.1", host: "localhost:8001"
```

一般项目都会有配置文件，在luastar项目中，配置文件放在demo/config/目录下，可以通过在luastar.conf文件中指定，默认使用app.lua文件

```
server {
    listen 8001;
    ...
    set $APP_CONFIG '/config/app_dev.lua';
    ...
}
```

配置文件的内容直接使用lua语法

```
--[[
应用配置文件
--]]
mysql = {
    host = "10.1.1.2",
    port = "3306",
    user = "root",
    password = "lajin2015",
    database = "cms_admin",
    timeout = 30000,
    pool_size = 1000
}
redis = {
    host = "10.1.1.4",
    port = "6382",
    auth = "lajin@2015",
    timeout = 30000,
    pool_size = 1000
}

_include_ = {
    "/config/app_dev_a.lua",
    "/config/app_dev_b.lua"
}
```

`_include_` 是一个特殊的用法，支持配置文件嵌套引入。

配置文件的内容在代码中，可以通过`luastar_config.getConfig`来获取：

```
local mysqlDataSource = luastar_config.getConfig("mysql")

local mysqlDataSourceHost = luastar_config.getConfig("mysql")["host"]
```

配置文件的内容也可以直接在`bean.lua`中使用，

```
mysql = {
  class = "luastar.db.mysql",
  arg = {
    { value = "${mysql}" }
  }
}
```

详情请参考bean的配置用法。

路由和拦截器在demo/config/route.lua文件中配置

```
--[[
应用路由配置
route = {
    {"url1","file1","method"},
    {"url2","file2","method"}
}
拦截器配置，注：拦截器必须实现beforeHandle和afterHandle方法
interceptor = {
    {url="url1", class="file"},
    {url="url2", class="file", excludes={"url1","url2"}}
}
--]]
route = {
    { "/api/test/hello", "com.luastar.demo.ctrl.test.hello", "hello" },
    { "/api/test/pic", "com.luastar.demo.ctrl.test.hello", "pic" },
    { "/api/test/mysql", "com.luastar.demo.ctrl.test.mysql", "mysql" },
    { "/api/test/mysql/transaction", "com.luastar.demo.ctrl.test.mysql", "transaction" },
    { "/api/test/redis", "com.luastar.demo.ctrl.test.redis", "redis" },
    { "/api/test/baidu", "com.luastar.demo.ctrl.test.httpClient", "baidu" },
    { "/api/test/proxy", "com.luastar.demo.ctrl.test.httpClient", "proxy" },
    { "/api/test/form", "com.luastar.demo.ctrl.test.form", "form" }
}

interceptor = {
    {
        url = "/api",
        class = "com.luastar.demo.interceptor.common"
    }
}
```

路由是一个二维数组，每一行表示一个接口地址，第一列表示请求地址（目前只支持全匹配），第二列表示对应的处理类，第三列表示处理类中的方法。

例如：当请求【<http://localhost:8001/api/test/hello>】时，由【com.luastar.demo.ctrl.test.hello】类的【hello】方法处理。

luastar默认给ctrl类请求处理方法传入了request和response对象（也可通过ngx.ctx.request和ngx.ctx.response获取），用于处理输入和输出。

```

--[[

--]]
local _M = {}

--[[
    获取普通参数/文件参数/请求体例子
--]]
function _M.hello(request, response)
    -- request:get_arg 支持获取get,post (含文件) 方式传过来的参数
    local name = request:get_arg("name") or "world, try to give
a param with name."
    ngx.log(logger.i("name=", name))
    -- 获取到的文件类型是table类型，包含filename (文件名) 和value (文件
内容) 属性
    local file = request:get_arg("file")
    if not _.isEmpty(file) then
        local file_save = io.open("/Users/zhuminghua/Downloads/o
utput/"..file["filename"], "w")
        file_save:write(file["value"]);
        file_save:close();
    end
    -- 获取到的request_body类型，注意如果client_max_body_size和clien
t_body_buffer_size不一致，
    -- 请求体超过client_body_buffer_size nginx会缓存到文件中，如果请求
体比较大，建议将两者设置成一致
    local request_body = request:get_request_body()
    -- local file_save = io.open("/Users/zhuminghua/Downloads/out
put/aaa.jpg", "w")
    -- file_save:write(request_body);
    -- file_save:close();
    response:writeLn("hello, " .. name)
end

return _M

```

拦截器与路由稍有不同，每一行表示一个拦截器（优先级取决于数组顺序），url代表拦截的请求，支持lua的模式匹配，class代表拦截器实现，excludes表示该拦截器不处理的请求。

拦截器要实现两个方法beforeHandle和afterHandle，beforeHandle必须返回布尔类型的结果，只要有一个拦截器返回false，则ctrl不会执行，beforeHandle可以返回第二个参数（字符串类型），用于返回false后的输出结果（返回true时忽略）

```
--[[
异常处理拦截器
--]]
local _M = {}

local json_util = require("com.luastar.demo.util.json")

function _M.beforeHandle()
    local request = ngx.ctx.request
    local headParam = {}
    headParam["appkey"] = request:get_header("appkey") or ""
    headParam["appversion"] = request:get_header("apiversion") or ""
    ngx.log(logger.i("request header is ", cjson.encode(headParam)))
    -- 统一校验请求头信息
    -- local hasEmpty = _.any(_.values(headParam), function(v) if
    --   _.isEmpty(v) then return true end end)
    -- return not hasEmpty
    return true
end

function _M.afterHandle(ctrl_call_ok, err_info)
    if not ctrl_call_ok then
        ngx.ctx.response:writeLn(json_util.exp(err_info))
    end
end

return _M
```

luastar实现了简化版的spring bean factory，默认将bean实例化后以单例模式（每个worker一份）存在缓存中。

定义bean

bean在配置文件demo/config/bean.lua文件中配置，注意保证id的唯一性

```
--[[
id = { -- bean id
  class = "", -- 类地址
  arg = { -- 构造参数注入
    {value/ref = ""} -- value赋值，ref引用其他bean
  },
  property = { -- set方法注入，实现set_${name}方法
    {name = "",value/ref = ""}
  },
  init_method = "", -- 初始化方法，默认使用init()
  single = 0 -- 是否单例，默认是1
}
--]]
mysql = {
  class = "luastar.db.mysql",
  arg = {
    { value = "${mysql}" }
  }
}
redis = {
  class = "luastar.db.redis",
  arg = {
    { value = "${redis}" }
  }
}
userService = {
  class = "com.luastar.demo2.service.system.userService"
}
funcService = {
  class = "com.luastar.demo2.service.system.funcService"
}
```

```

roleService = {
    class = "com.luastar.demo2.service.system.roleService"
}
userRoleRelationService = {
    class = "com.luastar.demo2.service.system.userRoleRelationService"
}
_include_ = {
    "/config/bean_uc.lua"
}

```

bean配置文件也支持_include_引入其他配置的语法。

在类中定义的方法最好使用类的模式，存储私有变量，可以使用luastar框架中的class类定义：

```

--[[
    系统用户服务类
--]]
local UserService = Class()

local sql_util = require("luastar.util.sql")
local date_util = require("luastar.util.date")

function UserService:init()
    self.queryCondition = {
        [[ID=#{id}]],
        [[and LOGIN_NAME=#{loginName}]],
        [[and PAZZWORD=#{pazzword}]],
        [[and USER_NAME=#{userName}]],
        [[and IS_EFFECTIVE=#{isEffective}]],
        [[and CREATED_TIME=#{createdTime}]],
        [[and UPDATED_TIME=#{updatedTime}]],
        [[and (
            LOGIN_NAME like concat('%',{keyword},%') or USER_NAME like concat('%',{keyword},%')
        )]]
    }
end

```

```
function UserService:getEmptyUser()  
    return {  
        id = 0,  
        loginName = "",  
        pazzword = "",  
        userName = "",  
        isEffective = 1,  
        createTime = "",  
        updateTime = ""  
    }  
end  
  
function UserService:userResultMap(result)  
    if _.isEmpty(result) then  
        return nil  
    end  
    return {  
        id = result["ID"],  
        loginName = result["LOGIN_NAME"],  
        pazzword = result["PAZZWORD"],  
        userName = result["USER_NAME"],  
        isEffective = result["IS_EFFECTIVE"],  
        createTime = result["CREATED_TIME"],  
        updateTime = result["UPDATED_TIME"]  
    }  
end  
  
function UserService:insert(user)  
    local sql_table = {  
        sql = [[  
            insert into SYS_USER (  
                LOGIN_NAME, PAZZWORD, USER_NAME, IS_EFFECTI  
VE, CREATED_TIME, UPDATED_TIME  
            ) values (  
                #{loginName}, #{pazzword}, #{userName}, #{i  
sEffective}, #{createTime}, #{updateTime}  
            )  
        ]]  
    }  
    user["createTime"] = date_util.get_time()
```



```
local sql = sql_util.getsql(sql_table, user)
local beanFactory = luastar_context.getBeanFactory()
local mysql_util = beanFactory.getBean("mysql")
local mysql = mysql_util:connect()
local res, err, errno, sqlstate = mysql:query(sql)
ngx.log(logger.i(cjson.encode({
    sql = sql,
    res = res,
    err = err,
    errno = errno,
    sqlstate = sqlstate
})))
mysql_util:close(mysql)
if _.isEmpty(res) then
    return nil
end
user["id"] = res["insert_id"]
return user["id"]
end

function UserService:update(user)
    local sql_table = {
        sql = [[
            update SYS_USER
            @{set}
            where ID=#{id}
        ]],
        set = {
            [[LOGIN_NAME=#{loginName}]],
            [[PAZZWORD=#{pazzword}]],
            [[USER_NAME=#{userName}]],
            [[IS_EFFECTIVE=#{isEffective}]],
            [[CREATED_TIME=#{createdTime}]],
            [[UPDATED_TIME=#{updatedTime}]]
        }
    }
    local sql = sql_util.getsql(sql_table, user)
    local beanFactory = luastar_context.getBeanFactory()
    local mysql_util = beanFactory.getBean("mysql")
    local mysql = mysql_util:connect()
```

```
local res, err, errno, sqlstate = mysql:query(sql)
ngx.log(logger.i(cjson.encode({
    sql = sql,
    res = res,
    err = err,
    errno = errno,
    sqlstate = sqlstate
})))
mysql_util:close(mysql)
if _.isEmpty(res) then
    return 0
end
return res["affected_rows"]
end

function UserService:getUserById(id)
    if _.isEmpty(id) then
        return nil
    end
    local sql_table = {
        sql = [[
            select * from SYS_USER
            where ID = #{id}
        ]]
    }
    local data = { id = id }
    local sql = sql_util.getsql(sql_table, data)
    local beanFactory = luastar_context.getBeanFactory()
    local mysql_util = beanFactory.getBean("mysql")
    local mysql = mysql_util:connect()
    local res, err, errno, sqlstate = mysql:query(sql)
    ngx.log(logger.i(cjson.encode({
        sql = sql,
        res = res,
        err = err,
        errno = errno,
        sqlstate = sqlstate
    })))
    mysql_util:close(mysql)
    if _.isEmpty(res) then
```

```
        return nil
    end
    return self:userResultMap(res[1])
end

function UserService:getUserByName(loginName)
    if _.isEmpty(loginName) then
        return nil
    end
    local sql_table = {
        sql = [[
            select * from SYS_USER
            where LOGIN_NAME = #{loginName}
            order by ID desc
            limit 1
        ]]
    }
    local data = { loginName = loginName }
    local sql = sql_util.getsql(sql_table, data)
    local beanFactory = luastar_context.getBeanFactory()
    local mysql_util = beanFactory:getBean("mysql")
    local mysql = mysql_util:getConnect()
    local res, err, errno, sqlstate = mysql:query(sql)
    ngx.log(logger.i(cjson.encode({
        sql = sql,
        res = res,
        err = err,
        errno = errno,
        sqlstate = sqlstate
    })))
    mysql_util:close(mysql)
    if _.isEmpty(res) then
        return nil
    end
    return self:userResultMap(res[1])
end

function UserService:countUser(data)
    if data == nil then
        data = {}
    end
end
```

```
end
local sql_table = {
    sql = [[
        select count(*) num
        from SYS_USER
        @where
    ]],
    where = self.queryCondition
}
local sql = sql_util.getsql(sql_table, data)
local beanFactory = luastar_context.getBeanFactory()
local mysql_util = beanFactory.getBean("mysql")
local mysql = mysql_util:connect()
local res, err, errno, sqlstate = mysql:query(sql)
ngx.log(logger.i(cjson.encode({
    sql = sql,
    res = res,
    err = err,
    errno = errno,
    sqlstate = sqlstate
}))))
mysql_util:close(mysql)
if _.isEmpty(res) then
    return 0
end
return tonumber(res[1]["num"])
end

function UserService:getUserList(data)
    if data == nil then
        data = {}
    end
    local sql_table = {
        sql = [[
            select * from SYS_USER
            @where
            order by ID desc
            @limit
        ]],
        where = self.queryCondition,
```

```

        limit = {
            start = "#{start}",
            limit = "#{limit}"
        }
    }
    local sql = sql_util.getsql(sql_table, data)
    local beanFactory = luastar_context.getBeanFactory()
    local mysql_util = beanFactory.getBean("mysql")
    local mysql = mysql_util:connect()
    local res, err, errno, sqlstate = mysql:query(sql)
    ngx.log(logger.i(cjson.encode({
        sql = sql,
        res = res,
        err = err,
        errno = errno,
        sqlstate = sqlstate
    })))
    mysql_util:close(mysql)
    if _.isEmpty(res) then
        return nil
    end
    return _.mapArray(res, function(i, v)
        return self:userResultMap(v)
    end)
end

function UserService:existUser(userId, loginName)
    if _.isEmpty(loginName) then
        return true
    end
    local userList = self:getUserList({ loginName = loginName })
    if _.isEmpty(userList) then
        return false
    end
    -- 如果新增，有记录则存在
    if _.isEmpty(userId) or tonumber(userId) == 0 then
        return true
    end
    -- 如果是修改，则多于一条记录则存在
    if #userList > 1 then

```

```
        return true
    end
    -- 如果是修改，只有一条记录且ID与自己不同，则存在
    if userList[1]["id"] ~= tonumber(userId) then
        return true;
    end
    return false;
end

return UserService
```

使用bean

在代码中先获取bean工厂，再获取bean

```
function _M.list(request, response)
    local param = {
        draw = request:get_arg("draw"),
        start = tonumber(request:get_arg("start")) or 0,
        limit = tonumber(request:get_arg("length")) or 10,
        keyword = request:get_arg("query_username")
    }
    -- 查询结果
    local beanFactory = luastar_context.getBeanFactory()
    local userService = beanFactory:getBean("userService")
    local num = userService:countUser(param);
    local data = {}
    if num > 0 then
        data = userService:getUserList(param);
    end
    -- 返回结果
    local result = {
        draw = param["draw"],
        recordsTotal = num,
        recordsFiltered = num,
        data = data
    }
    response:writeLn(json_util.toJson(result, true))
end
```

luastar中对mysql和redis的操作基于openresty官方提供的组件 [LuaRestyMySQLLibrary](#) 和 [LuaRestyRedisLibrary](#)

luastar中对mysql和redis提供了以下功能：

1. 数据源配置
2. 获取连接
3. 关闭连接（使用连接池）
4. mysql事务
5. sql语句动态拼装
6. 未关闭连接监控

1 使用方法：

1.1 app.lua中配置数据源

```
mysql = {  
    host = "localhost",  
    port = "3306",  
    user = "admin",  
    password = "xxx",  
    database = "xxx",  
    timeout = 30000,  
    pool_size = 1000  
}  
redis = {  
    host = "localhost",  
    port = "6379",  
    auth = "xxx",  
    timeout = 30000,  
    pool_size = 1000  
}
```

1.2 bean.lua中配置bean

多数据源可以配置多个，id不一样即可


```
mysql = {  
    class = "luastar.db.mysql",  
    arg = {  
        { value = "${mysql}" }  
    }  
}  
redis = {  
    class = "luastar.db.redis",  
    arg = {  
        { value = "${redis}" }  
    }  
}
```

1.3 代码中使用

```
-- 获取封装类  
local beanFactory = luastar_context.getBeanFactory()  
local mysql_util = beanFactory.getBean("mysql")  
local redis_util = beanFactory.getBean("redis")  
  
-- 对于单次请求操作，可直接使用下列语句，不用获取和关闭连接  
mysql_util.query("sql")  
redis_util.hgetall("key")  
  
-- 对于多次请求操作，需要先获取到连接，依次执行，最后关闭连接  
local mysql = mysql_util:connect()  
local res1, err1, errno1, sqlstate1 = mysql:query(sql1)  
local res2, err2, errno2, sqlstate2 = mysql:query(sql2)  
mysql_util:close(mysql)  
  
local redis = redis_util:connect()  
local userinfo = table_util.array_to_hash(redis:hgetall("user:info:" .. uid))  
redis_util:close(redis)
```

1.4 动态sql语句

完整使用方式如下：

```
#{}，如果值为字符串，则增加单引号防sql注入，如果为空，处理为null
${}，直接替换，如果为空，处理为null
@{}，引用其他语句
sql_table = {
    sql = [[
        update SYS_USER @{}@{}@{}
    ]],
    set = {
        "USER_NAME = #{}userName", -- userName为nil时忽略
        "UPDATED_TIME = #{}updatedAt"
    },
    where = {
        "LOGIN_NAME = #{}loginName", -- loginName为nil时该语句忽略
        [[
            and USER_NAME like concat('%',#{}userName,'%') -- userName
为nil时该语句忽略
        ]]
    }
    limit = {
        start = "${start}", -- start和limit为nil时忽略
        limit = "${limit}"
    }
}
```

mysql使用示例（普通查询和事务回滚）：

```
--[[
--]]

local _M = {}

local sql_util = require("luastar.util.sql")

function _M.mysql(request, response)
    local name = request:get_arg("name") or ""
    local sql_table = {
```

```

        sql = [[
            select * from SYS_USER
            @{where}
            order by ID desc
            limit #{start},#{limit}
        ]],
        where = {
            "LOGIN_NAME = #{loginName}",
            [[
                and USER_NAME like concat('%',#{userName},'%')
            ]]
        }
    }
    local data = { userName = name, start = 0, limit = 10 }
    local sql = sql_util.getsql(sql_table, data)
    local beanFactory = luastar_context.getBeanFactory()
    local mysql_util = beanFactory:getBean("mysql")
    local mysql = mysql_util:connect()
    local res, err, errno, sqlstate = mysql:query(sql)
    mysql_util:close(mysql)
    response:write(cjson.encode({
        sql = sql,
        res = res,
        err = err,
        errno = errno,
        sqlstate = sqlstate
    }))
end

function _M.transaction(request, response)
    local beanFactory = luastar_context.getBeanFactory()
    local mysql_util = beanFactory:getBean("mysql")
    local sqlArray = {
        "update SYS_USER set USER_NAME='管理员1' where ID=1",
        "update SYS_USER set USER_NAME_A='管理员2' where ID=1" --
        USER_NAME_A not exists
    }
    local result_table = mysql_util:queryTransaction(sqlArray)
    response:write(cjson.encode(result_table))
end

```

```
return _M
```

2 连接监控

luastar默认开启了mysql和redis的未关闭连接监控，如果有没有关闭的连接，会输出错误日志：

```
2016/12/20 16:34:23 [error] 40144#0: *45 [lua] monitor.lua:42: check(): check info +...luastar/db/mysql.lua:73, client: 127.0.0.1, server: localhost, request: "GET /api/test/mysql/transaction HTTP/1.1", host: "localhost:8001"
```

加号代表开启了连接的位置，减号代表关闭了连接的位置，如果有不匹配的+和-，则能定位到未关闭的位置，如果一次请求中开启和关闭的次数太多，日志可能输出不全（ngx.log的限制）。

luastar中session的管理使用的是第三方库：[lua-resty-session](#)

session已放入到全局变量中，可以在代码中直接使用，支持cookie、shm、memcache和redis持久化方式。

session保存

```
-- session保存
ngx.log(logger.i("保存session: ", cJSON.encode(userInfo)))
session.save("user", userInfo)
```

session校验

```
function _M.beforeHandle()
    -- session校验
    if session.check() then
        local data = session.getData("user")
        ngx.log(logger.i("用户session验证通过", cJSON.encode(data)))
    )
    return true
end
ngx.log(logger.i("用户session验证不通过"))
template.render("login.html", { message = "login timeout!" })
)
return false
end
```

session数据获取

```
-- 登录用户信息
local userInfo = session.getData("user")
```

session销毁

```
-- 销毁session
session.destroy()
```

其他用法可参考 [lua-resty-session](#)

luastar中页面布局和渲染使用第三方库 [lua-resty-template](#)

配置web相关目录，相比api应用，需要额外配置template模板根路径和静态文件访问


```
server {
    listen 8002;
    #web项目关闭lua_code_cache后session会失效
    #lua_code_cache off;
    server_name localhost;
    #luastar路径
    set $LUASTAR_PATH '/Users/zhuminhua/Documents/work-private/luastar/luastar';
    #应用名称
    set $APP_NAME 'demo2';
    #应用路径
    set $APP_PATH '/Users/zhuminhua/Documents/work-private/luastar/demo2';
    #应用使用的配置，可区分开发/生产环境，默认使用app.lua
    set $APP_CONFIG '/config/app_dev.lua';
    #template模板根路径，web项目使用
    set $template_root '/Users/zhuminhua/Documents/work-private/luastar/demo2/views';
    #访问日志
    access_log '/Users/zhuminhua/logs/nginx/demo2/access.log' main;
    #错误/输出日志
    error_log '/Users/zhuminhua/logs/nginx/demo2/error.log' info;
    location / {
        default_type text/html;
        content_by_lua_file '${LUASTAR_PATH}/src/luastar_content.lua';
    }
    #静态文件目录(*.js,*.css...)
    location /assets {
        root '/Users/zhuminhua/Documents/work-private/luastar/demo2';
        index index.html index.htm;
    }
}
```

template页面渲染语法在这里不多介绍了，请参考 [lua-resty-template](#)，这里说一下页面布局



该页面由以下几部分组成：

layout.html 布局文件，其他界面都通过该布局文件输出。

```
<!DOCTYPE html>
<!--
Template Name: Metronic - Responsive Admin Dashboard Template bu
ild with Twitter Bootstrap 3.3.6
Version: 4.5.6
Author: KeenThemes
Website: http://www.keenthemes.com/
Contact: support@keenthemes.com
Follow: www.twitter.com/keenthemes
Dribbble: www.dribbble.com/keenthemes
Like: www.facebook.com/keenthemes
Purchase: http://themeforest.net/item/metronic-responsive-admin-
dashboard-template/4021469?ref=keenthemes
Renew Support: http://themeforest.net/item/metronic-responsive-a
dmin-dashboard-template/4021469?ref=keenthemes
License: You must have a valid license purchased only from theme
forest(the above link) in order to legally use the theme for you
r project.
-->
<!--[if IE 8]>
<html lang="en" class="ie8 no-js"> <![endif]-->
<!--[if IE 9]>
```

```
<html lang="en" class="ie9 no-js"> <![endif]-->
<!--[if !IE]><!-->
<html lang="en">
<!--<![endif]-->
<!-- BEGIN HEAD -->
<head>
  <meta charset="utf-8"/>
  {* blocks.page_title *}
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta content="width=device-width, initial-scale=1" name="viewport"/>
  <meta content="" name="description"/>
  <meta content="" name="author"/>
  <!-- BEGIN GLOBAL MANDATORY STYLES -->
  <link href="http://fonts.googleapis.com/css?family=Open+Sans:400,300,600,700&subset=all" rel="stylesheet" type="text/css" />
  <link href="/assets/global/plugins/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css" />
  <link href="/assets/global/plugins/simple-line-icons/simple-line-icons.min.css" rel="stylesheet" type="text/css" />
  <link href="/assets/global/plugins/bootstrap/css/bootstrap.min.css" rel="stylesheet" type="text/css" />
  <link href="/assets/global/plugins/bootstrap-switch/css/bootstrap-switch.min.css" rel="stylesheet" type="text/css" />
  <!-- END GLOBAL MANDATORY STYLES -->
  <!-- BEGIN COMMON PLUGIN STYLES -->
  <link href="/assets/global/plugins/bootstrap-toastr/toastr.min.css" rel="stylesheet" type="text/css" />
  <link href="/assets/global/plugins/bootstrap-modal/css/bootstrap-modal-bs3patch.css" rel="stylesheet" type="text/css" />
  <link href="/assets/global/plugins/bootstrap-modal/css/bootstrap-modal.css" rel="stylesheet" type="text/css" />
  <link href="/assets/global/plugins/datatables/datatables.min.css" rel="stylesheet" type="text/css" />
  <link href="/assets/global/plugins/datatables/plugins/bootstrap/datatables.bootstrap.css" rel="stylesheet" type="text/css" />

  <!-- END COMMON PLUGIN STYLES -->
  <!-- BEGIN PAGE LEVEL PLUGINS -->
  {* blocks.page_css *}
```

```
<!-- END PAGE LEVEL PLUGINS -->
<!-- BEGIN THEME GLOBAL STYLES -->
<link href="/assets/global/css/components.min.css" rel="stylesheet" id="style_components" type="text/css" />
<link href="/assets/global/css/plugins.min.css" rel="stylesheet" type="text/css" />
<!-- END THEME GLOBAL STYLES -->
<!-- BEGIN THEME LAYOUT STYLES -->
<link href="/assets/layouts/layout/css/layout.min.css" rel="stylesheet" type="text/css" />
<link href="/assets/layouts/layout/css/themes/darkblue.min.css" rel="stylesheet" type="text/css" id="style_color" />
<link href="/assets/layouts/layout/css/custom.min.css" rel="stylesheet" type="text/css" />
<!-- END THEME LAYOUT STYLES -->
<link rel="shortcut icon" href="favicon.ico" /> </head>
</head>
<!-- END HEAD -->

<body class="page-header-fixed page-sidebar-closed-hide-logo page-content-white">
{{layouts/header.html}}
<!-- BEGIN HEADER & CONTENT DIVIDER -->
<div class="clearfix"></div>
<!-- END HEADER & CONTENT DIVIDER -->
<!-- BEGIN CONTAINER -->
<div class="page-container">
    {{layouts/sidebar.html}}
    <!-- BEGIN CONTENT -->
    <div class="page-content-wrapper">
        <!-- BEGIN CONTENT BODY -->
        <div class="page-content">
            {* blocks.page_content *}
        </div>
        <!-- END CONTENT BODY -->
    </div>
    <!-- END CONTENT -->
</div>
<!-- END CONTAINER -->
{{layouts/footer.html}}
```

```
<!--[if lt IE 9]>
<script src="/assets/global/plugins/respond.min.js"></script>
<script src="/assets/global/plugins/excanvas.min.js"></script>
<![endif]-->
<!-- BEGIN CORE PLUGINS -->
<script src="/assets/global/plugins/jquery.min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/bootstrap/js/bootstrap.min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/js.cookie.min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/bootstrap-hover-dropdown/bootstrap-hover-dropdown.min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/jquery-slimscroll/jquery.slimscroll.min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/jquery.blockui.min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/bootstrap-switch/js/bootstrap-switch.min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/underscore/underscore-min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/seajs/sea.js" type="text/javascript"></script>
<script src="/assets/global/plugins/seajs/seajs-text.js" type="text/javascript"></script>
<!-- END CORE PLUGINS -->
<!-- BEGIN COMMON PLUGINS -->
<script src="/assets/global/plugins/bootstrap-toastr/toastr.min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/bootstrap-modal/js/bootstrap-modalmanager.js" type="text/javascript"></script>
<script src="/assets/global/plugins/bootstrap-modal/js/bootstrap-modal.js" type="text/javascript"></script>
<script src="/assets/global/scripts/datatable.js" type="text/javascript"></script>
<script src="/assets/global/plugins/datatables/datatables.min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/datatables/plugins/bootstrap/datatables.bootstrap.js" type="text/javascript"></script>
<script src="/assets/global/plugins/select2/js/select2.full.min.js"></script>
```

```

js" type="text/javascript"></script>
<script src="/assets/global/plugins/jquery-validation/js/jquery.
validate.min.js" type="text/javascript"></script>
<script src="/assets/global/plugins/jquery-validation/js/additio
nal-methods.min.js" type="text/javascript"></script>
<!-- END COMMON PLUGINS -->
<!-- BEGIN PAGE LEVEL PLUGINS -->
{* blocks.page_plugins *}
<!-- END PAGE LEVEL PLUGINS -->
<!-- BEGIN THEME GLOBAL SCRIPTS -->
<script src="/assets/global/scripts/app.min.js" type="text/javas
cript"></script>
<!-- END THEME GLOBAL SCRIPTS -->
<!-- BEGIN PAGE LEVEL SCRIPTS -->
{* blocks.page_js *}
<!-- END PAGE LEVEL SCRIPTS -->
<!-- BEGIN THEME LAYOUT SCRIPTS -->
<script src="/assets/layouts/layout/scripts/layout.min.js" type=
"text/javascript"></script>
<script src="/assets/layouts/layout/scripts/demo.min.js" type="t
ext/javascript"></script>
<!-- END THEME LAYOUT SCRIPTS -->
</body>

</html>

```

header.html 头部内容，需要输出登录用户信息

```

<!-- BEGIN HEADER -->
<div class="page-header navbar navbar-fixed-top">
  <!-- BEGIN HEADER INNER -->
  <div class="page-header-inner ">
    <!-- BEGIN LOGO -->
    <div class="page-logo">
      <a href="javascript:;">
        
      </a>
      <div class="menu-toggler sidebar-toggler">

```

```

        <span></span>
    </div>
</div>
<!-- END LOGO -->
<!-- BEGIN RESPONSIVE MENU TOGGLER -->
    <a href="javascript:;" class="menu-toggler responsive-to
ggler" data-toggle="collapse" data-target=".navbar-collapse">
        <span></span>
    </a>
<!-- END RESPONSIVE MENU TOGGLER -->
<!-- BEGIN TOP NAVIGATION MENU -->
<div class="top-menu">
    <ul class="nav navbar-nav pull-right">
        <!-- BEGIN USER LOGIN DROPDOWN -->
        <!-- DOC: Apply "dropdown-dark" class after belo
w "dropdown-extended" to change the dropdown styte -->
        <li class="dropdown dropdown-user">
            <a href="javascript:;" class="dropdown-toggl
e" data-toggle="dropdown" data-hover="dropdown" data-close-others
="true">

                
                <span class="username username-hide-on-m
obile"> {*username*} </span>
                <i class="fa fa-angle-down"></i>
            </a>
            <ul class="dropdown-menu dropdown-menu-defau
lt">

                <li>
                    <a href="javascript:;">
                        <i class="icon-user"></i> My Pro
file </a>

                </li>
                <li class="divider"></li>
                <li>
                    <a href="/system/logout">
                        <i class="icon-key"></i> Log Out

                </li>
            </ul>
        </li>
    </ul>

```

```

        </li>
        <!-- END USER LOGIN DROPDOWN -->
    </ul>
</div>
<!-- END TOP NAVIGATION MENU -->
</div>
<!-- END HEADER INNER -->
</div>
<!-- END HEADER -->

```

sidebar.html 左侧菜单，需要输出菜单信息

```

<!-- BEGIN SIDEBAR -->
<div class="page-sidebar-wrapper">
    <!-- BEGIN SIDEBAR -->
    <!-- DOC: Set data-auto-scroll="false" to disable the sidebar from auto scrolling/focusing -->
    <!-- DOC: Change data-auto-speed="200" to adjust the sub menu slide up/down speed -->
    <div class="page-sidebar navbar-collapse collapse">
        <!-- BEGIN SIDEBAR MENU -->
        <!-- DOC: Apply "page-sidebar-menu-light" class right after "page-sidebar-menu" to enable light sidebar menu style(without borders) -->
        <!-- DOC: Apply "page-sidebar-menu-hover-submenu" class right after "page-sidebar-menu" to enable hoverable(hover vs accordion) sub menu mode -->
        <!-- DOC: Apply "page-sidebar-menu-closed" class right after "page-sidebar-menu" to collapse("page-sidebar-closed" class must be applied to the body element) the sidebar sub menu mode -->
        <!-- DOC: Set data-auto-scroll="false" to disable the sidebar from auto scrolling/focusing -->
        <!-- DOC: Set data-keep-expanded="true" to keep the sub menus expanded -->
        <!-- DOC: Set data-auto-speed="200" to adjust the sub menu slide up/down speed -->
        <ul class="page-sidebar-menu page-header-fixed " data-keep-expanded="false" data-auto-scroll="true" data-slide-speed="200">

```



```
00">
    <!-- DOC: To remove the sidebar toggler from the sidebar you just need to completely remove the below "sidebar-toggler-wrapper" LI element -->
    <li class="sidebar-toggler-wrapper hide">
        <!-- BEGIN SIDEBAR TOGGLER BUTTON -->
        <div class="sidebar-toggler">
            <span></span>
        </div>
        <!-- END SIDEBAR TOGGLER BUTTON -->
    </li>
    {*sidebar*}
</ul>
<!-- END SIDEBAR MENU -->
</div>
<!-- END SIDEBAR -->
</div>
<!-- END SIDEBAR -->
```

footer.html 底部内容

```
<!-- BEGIN FOOTER -->
<div class="page-footer">
    <div class="page-footer-inner">
        2016 &copy; Luastar.
    </div>
    <div class="scroll-to-top">
        <i class="icon-arrow-up"></i>
    </div>
</div>
<!-- END FOOTER -->
```

user.html 用户列表

```
{-page_title-}
<title>用户管理</title>
{-page_title-}

{-page_css-}
```

```

{-page_css-}

{-page_content-}
<!-- BEGIN 用户列表 -->
<div id="content_user_list" class="row" data-sign="content">
  <div class="col-md-12">
    <div class="portlet box blue">
      <div class="portlet-title">
        <div class="caption">
          <i class="fa fa-list"></i>用户管理
        </div>
      </div>
      <div class="portlet-body">
        <div class="table-toolbar">
          <!-- BEGIN 操作按钮-->
          <div class="row margin-bottom-10">
            <div class="col-md-12">
              <div class="btn-group">
                <button id="btn_user_new" class=
"btn green">新增</button>
              </div>
            </div>
          </div>
          </div>
          <!-- END 操作按钮 -->
          <!-- BEGIN 查询表单 -->
          <div class="row">
            <div class="col-md-12">
              <form id="form_user_query" class="fo
rm-inline" role="form">
                <div class="form-group">
                  <div class="input">
                    <input type="text" class=
"form-control" name="query_username" placeholder="登录名或用户名">
                  </div>
                </div>
                <button id="btn_user_query" type=
"button" class="btn default">查询</button>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

```

```

        <!-- END 查询表单 -->
    </div>
    <table id="table_user_list" class="table table-s
triped table-bordered table-hover">
        <thead>
        </thead>
        <tbody>
        </tbody>
    </table>
</div>
</div>
</div>
</div>
<!-- END 用户列表 -->
<!-- BEGIN 用户编辑 -->
<div id="content_user_edit" class="row" data-sign="content">
    <div class="col-md-12">
        <div class="portlet box blue">
            <div class="portlet-title">
                <div class="caption"><i class="fa fa-edit"></i>
用户信息</div>
            </div>
            <div id="div_user_edit" class="portlet-body form">
            </div>
        </div>
    </div>
</div>
<!-- END 用户编辑 -->
{-page_content-}

{-page_plugins-}
{-page_plugins-}

{-page_js-}
<script src="/assets/pages/scripts/app_init.js" type="text/javas
cript"></script>
<script>
    jQuery(document).ready(function () {
        AppInit.init();
        seajs.use("pages/scripts/system/user/user", function (Us

```

```
er) {  
    new User().init();  
});  
});  
</script>  
{-page_js-}
```

输出页面

```
-- 输出  
local funcId = request:get_arg("funcId")  
local view = template.new("system/user/user.html", layout_util.g  
etLayout(funcId))  
view:render()  
  
function _M:getLayout(funcId)  
    -- 登录用户信息  
    local userInfo = session.getData("user")  
    -- 功能菜单  
    local beanFactory = luastar_context.getBeanFactory()  
    local funcService = beanFactory.getBean("funcService")  
    local sidebar = funcService:getUserSidebar(userInfo["id"], f  
uncId)  
    -- 布局  
    local layout = template.new("layouts/layout.html")  
    layout.username = userInfo["userName"]  
    layout.sidebar = sidebar  
    return layout  
end
```

luastar由个人开发维护，完全开源，对于使用到的第三方开源软件，有侵权的地方麻烦及时告知。

QQ群：545501138

Email：19102630@163.com