<https://github.com/openresty/lua-nginx-module#data-sharing-within-an-nginx-worker>

内Data Sharing within an Nginx Worker

**Data Sharing within an Nginx Worker**

To globally share data among all the requests handled by the same nginx worker process, encapsulate the shared data into a Lua module, use the Lua require builtin to import the module, and then manipulate the shared data in Lua. This works because required Lua modules are loaded only once and all coroutines will share the same copy of the module (both its code and data). Note however that Lua global variables (note, not module-level variables) WILL NOT persist between requests because of the one-coroutine-per-request isolation design.

Here is a complete small example:

-- mydata.lua

local \_M = {}

local data = {

dog = 3,

cat = 4,

pig = 5,

}

function \_M.get\_age(name)

return data[name]

end

return \_M

and then accessing it from nginx.conf:

location /lua {

content\_by\_lua\_block {

local mydata = require "mydata"

ngx.say(mydata.get\_age("dog"))

}

}

The mydata module in this example will only be loaded and run on the first request to the location /lua, and all subsequent requests to the same nginx worker process will use the reloaded instance of the module as well as the same copy of the data in it, until a HUP signal is sent to the Nginx master process to force a reload. This data sharing technique is essential for high performance Lua applications based on this module.

Note that this data sharing is on a *per-worker* basis and not on a *per-server* basis. That is, when there are multiple nginx worker processes under an Nginx master, data sharing cannot cross the process boundary between these workers.

It is usually recommended to share read-only data this way. You can also share changeable data among all the concurrent requests of each nginx worker process as long as there is *no* nonblocking I/O operations (including [ngx.sleep](https://github.com/openresty/lua-nginx-module#ngxsleep)) in the middle of your calculations. As long as you do not give the control back to the nginx event loop and ngx\_lua's light thread scheduler (even implicitly), there can never be any race conditions in between. For this reason, always be very careful when you want to share changeable data on the worker level. Buggy optimizations can easily lead to hard-to-debug race conditions under load.

If server-wide data sharing is required, then use one or more of the following approaches:

1. Use the [ngx.shared.DICT](https://github.com/openresty/lua-nginx-module#ngxshareddict) API provided by this module.
2. Use only a single nginx worker and a single server (this is however not recommended when there is a multi core CPU or multiple CPUs in a single machine).
3. Use data storage mechanisms such as memcached, redis, MySQL or PostgreSQL. [The OpenResty bundle](http://openresty.org/) associated with this module comes with a set of companion Nginx modules and Lua libraries that provide interfaces with these data storage mechanisms.

[Back to TOC](https://github.com/openresty/lua-nginx-module#table-of-contents)