**OpenResty学习系列（二）**

从今天开始，我们开始OPenResty的实战，在本文中分享以下两点：

1.OpenResty的下载安装

2． Hello World

一 OpenResty的下载安装

我们进入OpenResty官方进行下载:<https://openresty.org/cn/download.html>,由于我是mac,下面以mac下载安装为例,其他版本参考官网.对于mac用户,原则上是可以用brew install homebrew/nginx/openresty一条命令进行安装,但建议手动安装。我们首先下载最新版的压缩包,进行解压:tar -xzvf openresty-VERSION.tar.gz.示例中的 VERSION替换成 OpenResty的版本号, 比如 1.11.2.1。先安装一些相关依赖：

brew update

brew install pcre openssl

安装好PCRE和OpenSSL之后，要指定下安装路径，不然会报错：

$ ./configure \

--with-cc-opt="-I/usr/local/opt/openssl/include/ -I/usr/local/opt/pcre/include/" \

--with-ld-opt="-L/usr/local/opt/openssl/lib/ -L/usr/local/opt/pcre/lib/" \

-j8

cd打开OpenResty-version目录，之后输入命令

./configure

默认, --prefix=/usr/local/openresty 程序会被安装到/usr/local/openresty目录。可以指定各种选项，比如

./configure --prefix=/opt/openresty \

--with-luajit \

--without-http\_redis2\_module \

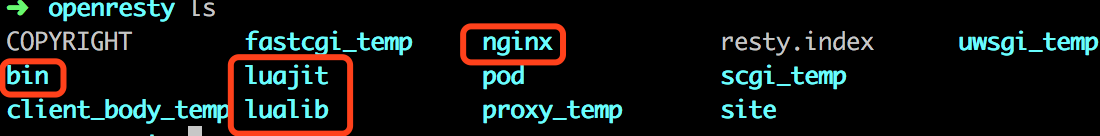
--with-http\_iconv\_module \

--with-http\_postgres\_module

试着使用 ./configure --help 查看更多的选项。

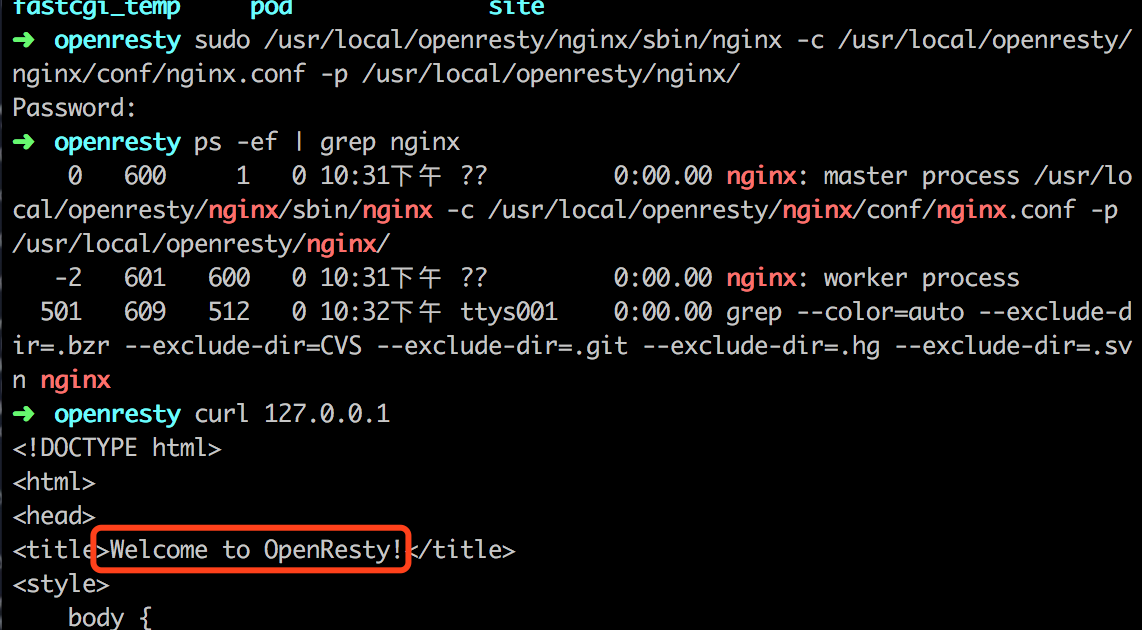
如果没报错的话，会在最后提示让我们进行make和make install，最后一步执行的make install有时候会有报错没有权限，我们可以加上sudo make install，执行命令，静等安装结束。

安装完成后，我们可以看下OpenResty的目录：有四个子目录是必备的——bin、nginx、luajit、lualib。



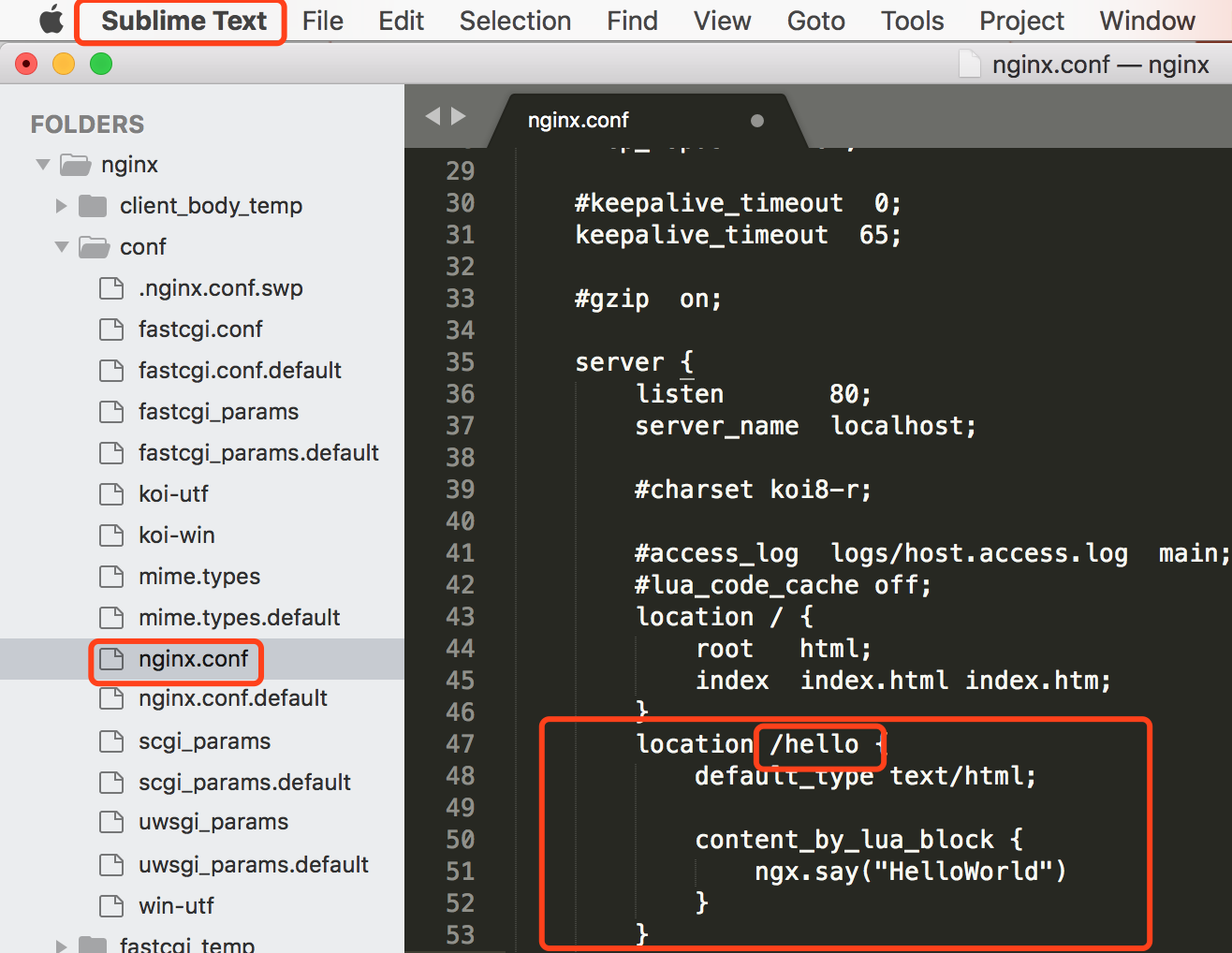
为检查下是否真的安装好了，我们启动下Nginx：

* 依次执行启动指令；检查下nginx是否真的启动了；用curl验证下。可以看到已经安装成功：

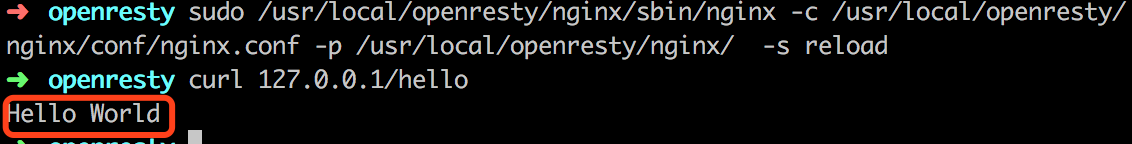


二 Hello World

安装好后，该怎么写Hello World？我们可以Google搜索官方文档，网上有现成的例子，我们可以用Sublime打开openresty/nginx，在配置文件nginx.conf里进行修改,这里要注意的是，我们在代码官网location后加了子目录/hello，不然会和配置文件里的‘location/’重复引发错误示意图如下：

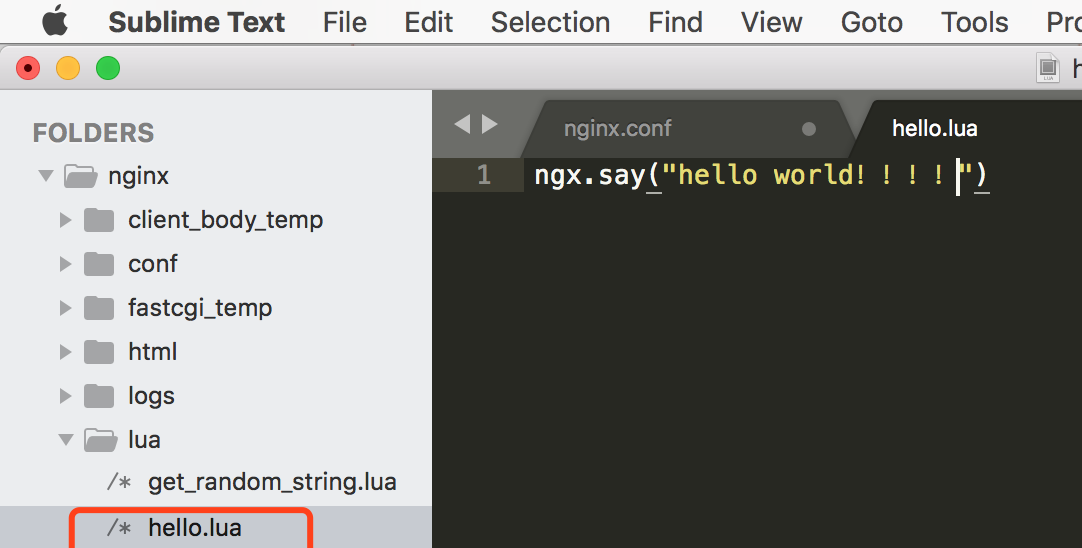


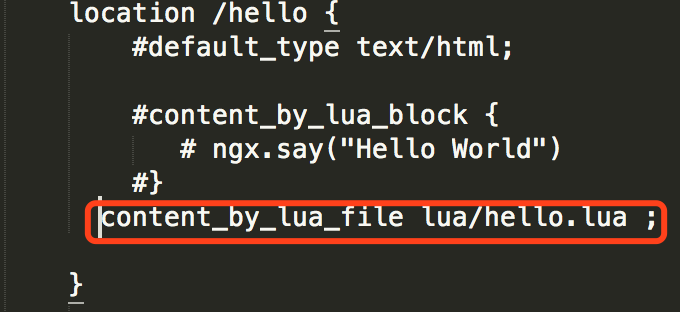
ok，由于我们对配置文件进行了修改，我们重置一下nginx后用curl检验一下：我们发现，出现了Hello World。



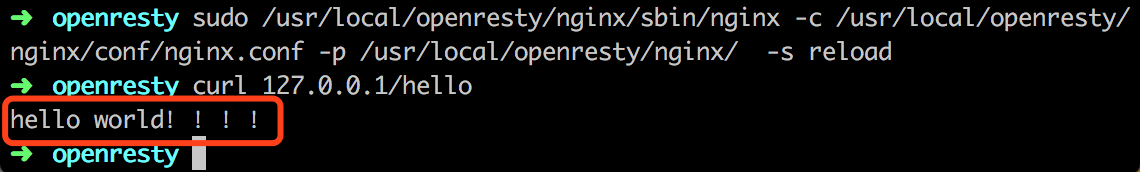
我们可以再来看下代码，OPenResty神奇的地方就是可以在Nginx里编写Lua代码来调用她，直接实现我们的需求。

但现在有个问题，如果我们有几百上千甚至上万行代码，直接写在配置文件里会显得不太合适，这里我们可以在Google里搜索ngx lua打开ngx官方文档进行搜索，我们后面每个系列基本都会用到这个文档，在OpenResty里有个指令content\_by\_lua\_file，我们可以在文件里写入我们的Lua代码，然后使用content\_by\_lua\_file调用这个文件：我们首先在nginx目录下创建lua/hello.lua，将lua代码写入hello.lua文件中然后调用文件，示意如下图

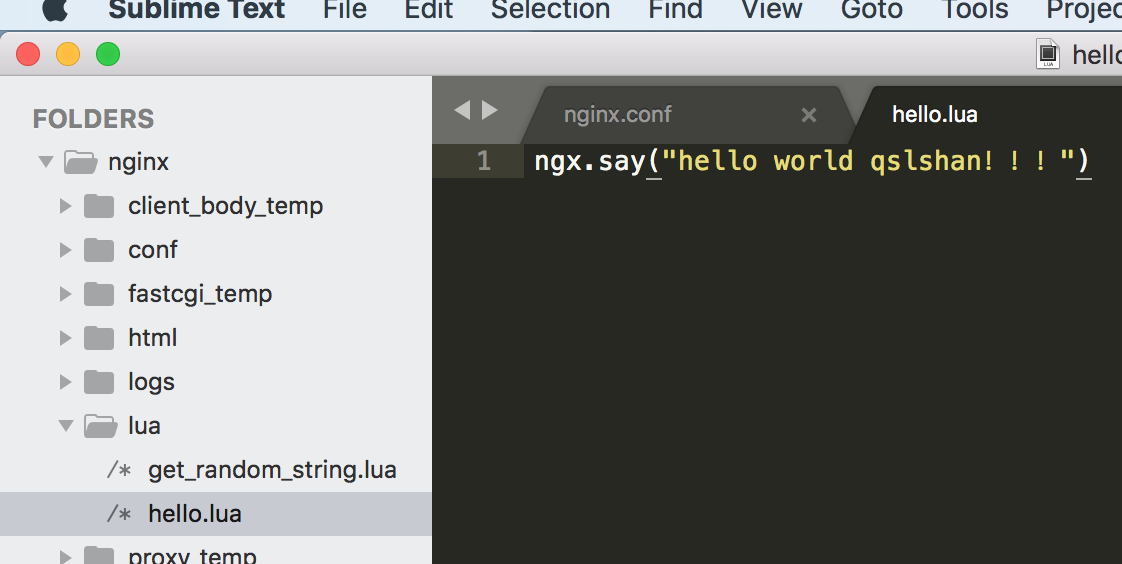


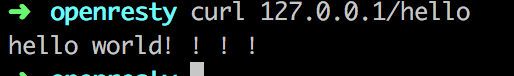


重置nginx后执行结果如下：我们看到了想要的结果。

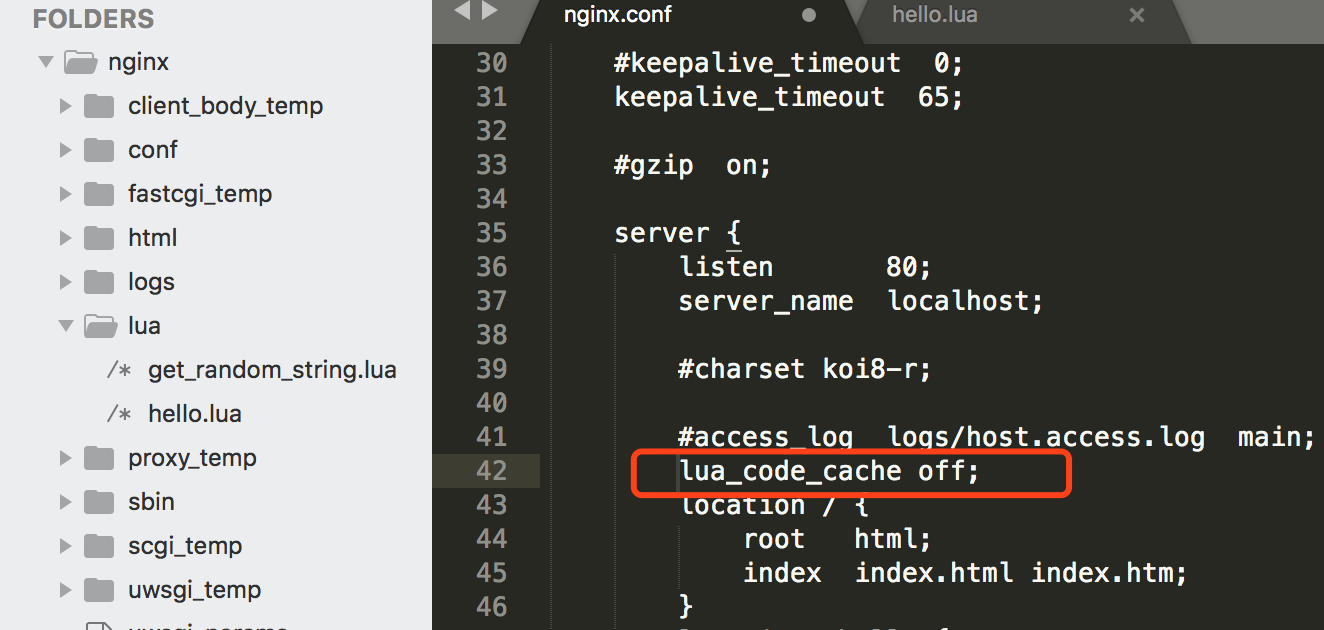


我们修改一下文件中代码再次执行：

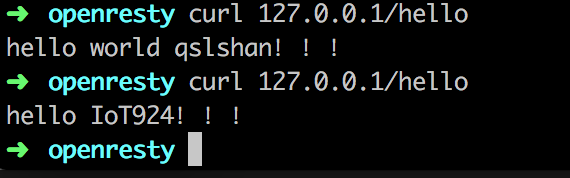


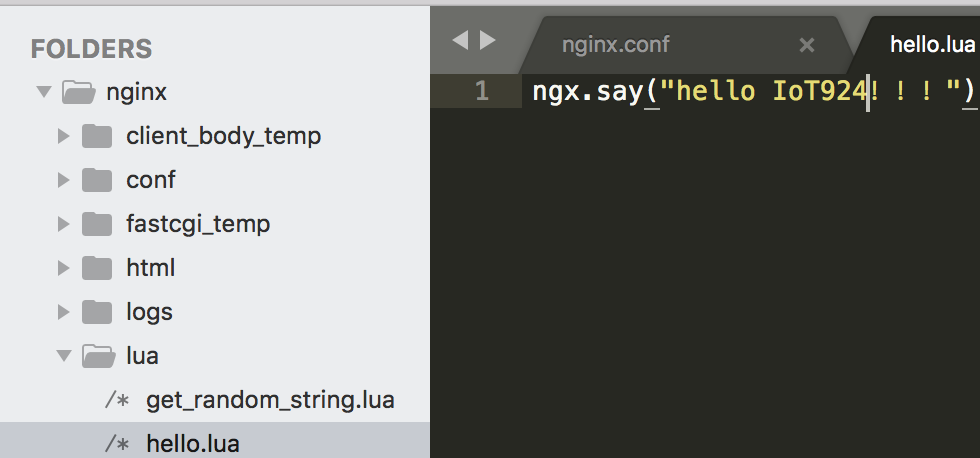


结果和上次相同，一拍脑门发现原来是我们忘记重置了，但是我们在编程的过程中，改写是很频繁的，每次都要重置会很麻烦，该怎么办呢？我们再次搜索ngx，发现里面有个指令lua\_code\_cache，它有on/off两个值，分别表示打开/关闭缓存，我们在测试的时候可以将lua\_code\_cache设置成off状态，这样我们每次执行时都会默认重置，但在开发环境下一定要将其打开，这会对性能有很大的提高，我们可以将lua\_code\_cache放在server下，进行下面的尝试：。



首次需要启动，很显然我们得到我们想要的结果，我们再次对hello.lua进行修改，这次直接测试结果,结果很显然。





Ok,到这里我们就实现了我们的第一个Hello World，过程很简单，实现这个Demo的目的是帮大家了解OpenResty开发是什么样子的：在OpenResty里，开发分成两步，一个是对Nginx配置文件的修改，另一个则是Lua代码的开发，这是和其他开发语言很不一样的一个点。希望大家在后面的系列中慢慢熟悉这样的开发流程，在OpenResty学习系列（三）中我会用一个更复杂的例子，来分享一下Nginx的API。

欢迎一起交流学习！