第1章 阿里云lamp环境搭建

1. 参考https://help.aliyun.com/document\_detail/50774.html

#### Apache

访问http://www.apache.org/dyn/closer.cgi，选择最快的镜像下载最新apache安装包

安装依赖包。

yum groupinstall “ Development Tools” -y

yum install libtool

yum install expat-devel pcre pcre-devel openssl-devel -y

下载解压Apache，Apr和Apr-util的源码包。

wget https://mirrors.aliyun.com/apache/httpd/httpd-2.4.39.tar.gz

wget https://mirrors.aliyun.com/apache/apr/apr-1.7.0.tar.gz

wget https://mirrors.aliyun.com/apache/apr/apr-util-1.6.1.tar.gz

tar xvf httpd-2.4.39.tar.gz -C /usr/local/src

tar xvf apr-1.7.0.tar.gz -C /usr/local/src

tar xvf apr-util-1.6.1.tar.gz -C /usr/local/src

mv apr-1.7.0 httpd-2.4.39/srclib/apr

mv apr-util-1.6.1 httpd-2.4.39/srclib/apr-util

cd /usr/local/src/httpd-2.4.

./buildconf

./configure --prefix=/usr/local/apache2 \

--enable-ssl \

--enable-so \

--with-mpm=event \

--with-included-apr \

--enable-cgi \

--enable-rewrite \

--enable-mods-shared=most \

--enable-mpms-shared=all

make && make install

设置PATH环境变量。

echo “export PATH=$PATH:/usr/local/apache2/bin” > /etc/profile.d/httpd.sh

source /etc/profile.d/httpd.sh

输入命令httpd -v可查看Apache的版本号。

添加Apache的启动配置文件。

输入命令vi /usr/lib/systemd/system/httpd.service打开Apache的启动配置文件，按下i键，然后在配置文件中写下如下内容：

[Unit]

Description=The Apache HTTP Server

After=network.target

[Service]

Type=forking

ExecStart=/usr/local/apache2/bin/apachectl -k start

ExecReload=/usr/local/apache2/bin/apachectl -k graceful

ExecStop=/usr/local/apache2/bin/apachectl -k graceful-stop

PIDFile=/usr/local/apache2/logs/httpd.pid

PrivateTmp=true

[Install]

WantedBy=multi-user.target

按下Esc键，然后输入:wq并回车以保存并关闭Apache启动配置文件。

启动Apache服务并设置开机自动启动。

systemctl start httpd

systemctl enable httpd

新建用户

adduser yii6

passwd yii6

cd /home

chmod 755 yii6

修改htdocs根目录

vim /usr/local/apache2/conf/httpd.conf

修改

<Directory />

AllowOverride none

Require all granted

</Directory>

找到 ServerName 参数，添加

#ServerName www.example.com:80

ServerName localhost:80

修改DocumentRoot和紧跟着的<Directory “/usr/local/apache/htdocs”>

把原目录修改为个人目录/home/yii6/public\_html，因为composer安装扩展不能使用root

修改所有相应目录权限为755

修改.htaccess相关的

AllowOverride All

LoadModule rewrite\_module modules/mod\_rewrite.so

在文件最后添加

//PidFile “/var/run/httpd.pid”

AddType application/x-httpd-php .php

AddType application/x-httpd-php-source .phps

定位到DirectoryIndex index.html，修改为 DirectoryIndex index.php index.html

apache配置出错可能出现

1. 域名不能访问，报连接被重置错误，AllowOverride All修改错误
2. php文件不解析，直接显示源码，使用原来配置好的http.conf文件
3. 报DocumentRoot [/usr/local/apache/docs/dummy-host.example.com] does not exist错误

此时比较尴尬，因为是用的老的配置文件，搜索会搜不出来，因为出错地方在/etc/httpd/extra/http-vhosts.conf，注释掉即可

不同二级域名指向不同目录，.htaccess修改失效时可以尝试

vim /etc/httpd/httpd.conf

解除注释Include “/extra/httpd-vhosts.conf”

vim /etc/httpd/extra/httpd-vhosts.conf

两个域名配置之间要有一个空行。

<VirtualHost \*:80>

DocumentRoot “/home/yii6/public\_html/t”

ServerName t.yii6.com

ServerAlias t.yii6.com

ErrorLog “logs/t.yii6.com-error\_log”

CustomLog “logs/t.yii6.com-access\_log” common

</VirtualHost>

<VirtualHost \*:80>

DocumentRoot “/home/yii6/public\_html/”

ServerName yii6.com

ServerAlias www.yii6.com

ErrorLog “logs/yii6.com-error\_log”

CustomLog “logs/yii6.com-access\_log” common

</VirtualHost>

检测是否安装成功

/usr/local/apache2/bin/apachectl start

netstat -tnlp

使.htaccess生效

/usr/local/apache2/bin/apachectl restart

vi /root/.bash\_profile

将 PATH=$PATH:$HOME/bin 修改为 PATH=$PATH:$HOME/bin:/usr/local/apache/bin

运行 source /root/.bash\_profile使修改生效

备用选择：

#### Nginx

centos 8安装

**dnf -y install http://nginx.org/packages/centos/8/x86\_64/RPMS/nginx-1.16.1-1.el8.ngx.x86\_64.rpm**

查看安装路径

**$ rpm -ql nginx**

**/usr/local/nginx/sbin/nginx -t** 测试配置是否正确

**/usr/local/nginx/sbin/nginx -s reload**重启

**/usr/sbin/nginx -t**

**/usr/sbin/nginx -s reload**

参考：

https://help.aliyun.com/document\_detail/173042.html#title-z9a-b9e-4ib

域名配置：

host文件：

127.0.0.1 [www.kidswritedev.com](http://www.kidswritedev.com)

nginx配置：

server\_name [www.kidswritedev.com;](http://www.kidswritedev.com;)

本地访问：



配置文件路径：

/etc/nginx/nginx.conf

修改nginx.conf：

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /usr/local/var/log/nginx/access.log main;

sendfile on;

#tcp\_nopush on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.kidswritedev.com;

access\_log /usr/local/var/log/nginx/host.access.log main;

#修改默认的网页存放路径

root /opt/www/kidswrite\_api/public;

#添加 .php文件格式的首页

index index.php index.html index.htm;

#如果不是请求文件,将路由转写到index.php

if (!-e $request\_filename) {

rewrite ^/(.\*) /index.php?$1 last;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

location ~ \.php$ {

#将所有的php文件转发给9000端口进行处理，否则访问php文件会直接下载

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

#解决报No input file specified 的问题

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

}

include servers/\*;

}

&表示后台执行命令

重启nginx后正常执行phpinfo()

**service nginx restart**

nginx 报错502 bad gateway

可能是 fastcgi.conf没有加载

将配置文件mhdc.xcxweshine.com.conf和fastcgi.conf放在同一个目录，然后修改nginx.conf配置文件，将mhdc.xcxweshine.com.conf包含进来

mac error.log(/usr/local/var/log):

kevent() reported that connect() failed (61: Connection refused) while connecting to upstream

可能php-fpm未启动

上传文件413 Request Entity Too Large错误：

server {

...

client\_max\_body\_size 20m;

...

}

上传文件504 Gateway Time-out

location ~ \.php$ {

…

fastcgi\_index restful.php;

fastcgi\_pass 127.0.0.1:9000;

include fastcgi.conf;

fastcgi\_param PATH\_INFO $path\_info;

fastcgi\_connect\_timeout 300s;

fastcgi\_send\_timeout 300s;

fastcgi\_read\_timeout 300s;

}

#### 安装mysql

rpm -qa | grep mysql

rpm -qa | grep mariadb

如果已经安装，则运行以下任一个命令删除。

rpm -e 软件名

rpm -e --nodeps 软件名 #卸载不成功时使用

准备编译环境。

yum install ncurses-devel bison gnutls-devel -y

yum install cmake3 –y

cmake3需要gcc 5.3以上

yum install devtoolset-4-gcc devtoolset-4-binutils devtoolset-4-gcc-c++

替换系统GCC

export CC=/opt/rh/devtoolset-2/root/usr/bin/gcc

export CPP=/opt/rh/devtoolset-2/root/usr/bin/cpp

export CXX=/opt/rh/devtoolset-2/root/usr/bin/c++

替换系统GCC方案2：

ln -s /opt/rh/devtoolset-2/root/usr/bin/\* /usr/local/bin/

hash -r

gcc –version

准备MySQL数据存放目录。

cd

mkdir /mnt/data

groupadd -r mysql

useradd -r -g mysql -s /sbin/nologin mysql

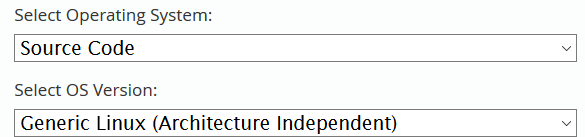
id mysql

更改数据目录属主和属组。

chown -R mysql:mysql /mnt/data

下载稳定版源码包解压编译。

官网<https://dev.mysql.com/downloads/mysql/>



wget https://dev.mysql.com/get/Downloads/MySQL-8.0/mysql-boost-8.0.16.tar.gz

tar xvf mysql-boost-8.0.16.tar.gz -C /usr/local/src

cd /usr/local/src/mysql-8.0.16

新建一个编译用的文件夹

mkdir –p build

cd build

编译命令要把文件目录修改为相对路径

cmake3 .. -DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql \

-DMYSQL\_DATADIR=/mnt/data \

-DSYSCONFDIR=/etc \

-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \

-DWITH\_ARCHIVE\_STORAGE\_ENGINE=1 \

-DWITH\_BLACKHOLE\_STORAGE\_ENGINE=1 \

-DWITH\_READLINE=1 \

-DWITH\_SSL=system \

-DWITH\_ZLIB=system \

-DWITH\_LIBWRAP=0 \

-DMYSQL\_TCP\_PORT=3306 \

-DDEFAULT\_CHARSET=utf8 \

-DMYSQL\_UNIX\_ADDR=/usr/local/mysql/mysql.sock \

-DDEFAULT\_COLLATION=utf8\_general\_ci \

-DWITH\_SYSTEMD=1 \

-DINSTALL\_SYSTEMD\_UNITDIR=/usr/lib/systemd/system \

-DWITH\_BOOST=../boost

make && make install

更改 mysql 安装目录的属性

chown -R mysql:mysql /usr/local/mysql。

运行命令初始化 MySQL 数据库。

/usr/local/mysql/bin/mysqld --initialize-insecure --datadir=/mnt/data/ --user=mysql

复制配置文件

cp /etc/my.cnf /etc/my.cnf.bak

修改配置文件中的安装路径及数据目录存放路径。

vim /etc/my.cnf

basedir = /usr/local/mysql

datadir = /mnt/data

输入命令vi /usr/lib/systemd/system/mysql.service打开MySQL的启动配置文件，按下i键，然后写下如下内容：

[Unit]

Description=MySQL Community Server

After=network.target

After=syslog.target

[Install]

WantedBy=multi-user.target

Alias=mysql.service

[Service]

User=mysql

Group=mysql

PermissionsStartOnly=true

ExecStart=/usr/local/mysql/bin/mysqld

TimeoutSec=600

Restart=always

PrivateTmp=false

设置环境变量。

echo “export PATH=$PATH:/usr/local/mysql/bin” > /etc/profile.d/mysql.sh

source /etc/profile.d/mysql.sh

启动MySQL服务并设置开机启动。

systemctl start mysql

systemctl enable mysql

修改MySQL的root用户密码

mysqladmin -u root password

测试登录数据库（p和密码之间没有空格）

mysql -uroot -p密码

新建需要的数据库yii6

mysql>CREATE DATABASE yzzq\_wshoto\_com;

use yzzq\_wshoto\_com;

CREATE USER 'skf'@'localhost' IDENTIFIED BY 'Cs4K7FhBBY8';

grant all privileges on yii\_skf.\* to skf@localhost;

使操作生效：

mysql> flush privileges;

修改用户密码

在mysql数据库里面

set password for '你的用户名'@'localhost'=password('你的新密码');

报错1：

Starting MySQL.2018-02-12T15:36:17.109370Z mysqld\_safe error: log-error set to '/var/log/mariadb/mariadb.log', however file don't exists. Create writable for user 'mysql'.

解决：

mkdir -p /var/log/mariadb/

cd /var/log/mariadb/

touch mariadb.log

chmod -R 775 mariadb.log

chown -R mysql:mysql mariadb.log

报错2：

mysqld\_safe Directory '/var/lib/mysql' for UNIX socket file don't exists.

解决：

mv /etc/my.cnf /etc/my.cnf.bak

可以cat my.cnf看看，错误原因是读取了错误的配置文件

参考：<https://www.cnblogs.com/ShanFish/p/6652174.html>

报错3：

Starting MySQL.Logging to '/usr/local/mysql/data/iZbp12f0zcyrke28fa59b3Z.err'.

......The server quit without updating PID file

(/usr/local/mysql/data/iZbp12f0zcyrke28fa59b3Z.pid).

查看日志显示：

[ERROR] InnoDB: mmap(137428992 bytes) failed; errno 12

解决：

$ dd if=/dev/zero of=/swapfile bs=1M count=1024

$ mkswap /swapfile

$ swapon /swapfile

vi /etc/fstab 使swap永久保存

/swapfile swap swap defaults 0 0

增加1G的swap

查看日志发现有warning，

修改my.cnf，添加

#开启查询缓存

explicit\_defaults\_for\_timestamp=true

报错4：

error while loading shared libraries: libnuma.so.1: cannot open shared object file: No

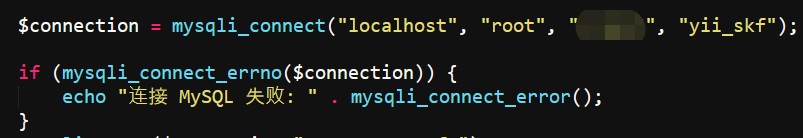
解决：

如果已经安装了libnuma.so.1，先yum remove libnuma.so.1

yum -y install numactl.x86\_64

报错5

mysql数据库连接，no such file or directory，打码的地方是密码



mysql –uroot –pYourpassword

>status



vim /etc/php.ini

mysqli.default\_socket= /usr/local/mysql/mysql.sock

pdo\_mysql.default\_socket=/usr/local/mysql/mysql.sock

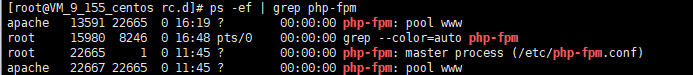
#### 安装php

重启php-fpm

找到进程

ps -ef | grep php-fpm

后面带配置文件的就是进程



kill -USR2 22665

重启进程就能重新加载php的配置和扩展

yum install libmcrypt libmcrypt-devel mhash mhash-devel libxml2 libxml2-devel bzip2 bzip2-devel -y

wget https://www.php.net/distributions/php-7.3.7.tar.gz

tar zxvf php-7.3.7.tar.gz -C /usr/local/src

cd /usr/local/src/php-7.3.7

./configure --prefix=/usr/local/php \

--with-config-file-scan-dir=/etc/php.d \

--with-apxs2=/usr/local/apache2/bin/apxs \

--with-config-file-path=/etc \

--with-pdo-mysql=mysqlnd \

--enable-mbstring \

--with-freetype-dir \

--with-jpeg-dir \

--with-png-dir \

--with-zlib \

--with-libxml-dir=/usr \

--with-openssl \

--enable-xml \

--enable-sockets \

--enable-fpm \

--with-bz2

make && make install

cp php.ini-production /etc/php.ini

cp /usr/local/php/etc/php-fpm.conf.default /usr/local/php/etc/php-fpm.conf

cp /usr/local/php/etc/php-fpm.d/www.conf.default /usr/local/php/etc/php-fpm.d/www.conf

phpsize命令找不到可以搜一下

find / -name phpize

将/etc/php.d里面的ini文件全部重命名为.so文件

启动php-fpm

/usr/local/php/sbin/php-fpm

如果报错：

[20-Aug-2018 09:05:48] ERROR: An another FPM instance seems to already listen on /tmp/php-cgi.sock

说明php-fpm已经启动，而且监听的不是9000端口，而是php-cgi.sock

方式1:  
php-fpm.conf: listen = 127.0.0.1:9000  
nginx.conf: fastcgi\_pass 127.0.0.1:9000;  
方式2:  
php-fpm.conf: listen = /tmp/php-fpm.sock  
nginx.conf: fastcgi\_pass unix:/tmp/php-fpm.sock;

php-fpm.sock也可能是php-cgi.sock

开机启动：

vim /etc/rc.local

加入

/usr/local/php/sbin/php-fpm

用阿里大鱼发短信没有安装curl会报错

undefined function Aliyun\Core\Http\curl\_init()

解决：

cd /usr/local/src/php-7\*/ext/curl

phpize

./configure --with-php-config=/usr/local/php/bin/php-config

make &&make install

出现下面的提示

Installing shared extensions: /usr/local/php/lib/php/extensions/no-debug-zts-20170718/

# cd /usr/local/php/lib/php/extensions/no-debug-zts-20170718/

# ls

文件夹中会有curl.so

vim /etc/php.ini

查找curl，修改或添加下面的一行

extension=curl.so

重启apache curl就可以使用了

yii2的图形验证码报错，tpframe后台验证码不显示：

Either GD PHP extension with FreeType support or ImageMagick PHP extension with PNG support is required

Call to undefined function imagettfbbox()

方案一：

cd /usr/local/src

curl -O ftp://ftp.imagemagick.org/pub/ImageMagick/ImageMagick.tar.gz

tar -zxvf ImageMagick.tar.gz

cd Ima\*

./configure --prefix=/usr/local/ImageMagick

make && make install

-bash: pecl: command not found

yum install php-devel php-pear httpd-devel –y

pecl install imagick

vim /etc/php.ini

加入extension=imagick.so

方案二：

yum install freetype\*

yum install libjpeg\*

find / -name freetype

cd /usr/local/src/php-7\*/ext/gd

make clean(先安装的gd，再安装的freetype则需要make clean)

./configure --with-freetype-dir=/usr/include/freetype2 --with-jpeg-dir=/usr/include

make && make install

安装p7后发现phpinfo的版本是7.3，而php -v查看的却是5.4

mv /usr/bin/php /usr/bin/php.bak

将phpinfo中的php可执行文件软链到/usr/bin/目录下

ln -s /usr/local/php/bin/php /usr/bin/php

重启apache

/usr/local/apache2/bin/apachectl restart

安装微擎Class 'ZipArchive' not found

解决：

cd /usr/local/src/

wget wget <https://nih.at/libzip/libzip-1.2.0.tar.gz>

(最新的不知道为什么没有configure)

cd lib\*

./configure

make && make install

wget <http://pecl.php.net/get/zip-1.15.2.tgz>

（http://pecl.php.net/package/zip）

tar -zxvf zip\*

cd zip\*

phpize

（phpize: command not found

# yum -y install php-devel）

fatal error: zipconf.h: No such file

解决：

find /usr/local -iname 'zipconf.h'

ln –s /usr/local/lib/libzip/include/zipconf.h /usr/local/include

./configure --with-php-config=/usr/local/php/bin/php-config

make && make install

vim /etc/php.ini

加入extension=zip.so

修改环境变量

vi /root/.bash\_profile

PATH=$PATH:$HOME/bin:/usr/local/php/bin

source /root/.bash\_profile

普通用户先cd到主目录

vim .bash\_profile

source .bash\_profile

php -v

报错：

PHP Warning: PHP Startup: Unable to load dynamic library 'curl.so'

解决：

将/etc/php.d/目录下的.ini文件全部重命名为.so文件

rename .ini .so \*.ini

将所有匹配\*.ini的文件中的.ini重命名为.so

报错：

Class ‘DomDocument’ not found

yum install php-xml -y

文件上传大小限制：

php.ini

upload\_max\_filesize

post\_max\_size

#### 安装php扩展

编译扩展的时候

make -j12

12是指当前cpu的核心数

查看当前cpu核心数量

grep 'core id' /proc/cpuinfo | sort -u | wc –l

**yaf** :

下载yaf

http://pecl.php.net/package/yaf

tar -zxvf yaf.tar.gz

cd yaf

**mac :**

/usr/local/Cellar/php56/5.6.30\_6/bin/phpize

./configure --with-php-config=/usr/local/Cellar/php56/5.6.30\_6/bin/php-config

make

make install

php.ini里面加上

extension=yaf.so

yaf.environ=product // 本地为 develop

yaf.use\_namespace=1

yaf.use\_spl\_autoload=1

**rabbitmq**

**mac:**

1.brew install rabbitmq

2.brew install rabbitmq-c

3.pecl install amqp

**apcu**

pecl install apcu

**rdkafka**

**mac:**

brew install librdkafka

perl install rdkafka

**centos:**

https://github.com/arnaud-lb/php-rdkafka

https://github.com/edenhill/librdkafka

# unzip librdkafka-master.zip

# cd librdkafka-master

# ./configure

# make && make install

# unzip php-rdkafka-master.zip

# cd php-rdkafka-master

# phpize

# ./configure --with-php-config=/usr/local/php/bin/php-config

# make && make install

**mongodb**

**mac:**

brew tap mongodb/brew

brew install mongodb-community

brew services start mongodb/brew/mongodb-community

**redis**

**mac:**

brew install redis

brew services start redis

pecl install redis

pecl install igbinary

swoole

pecl install swoole

源码安装

https://github.com/swoole/swoole-src/releases

wget https://github.com/swoole/swoole-src/archive/v4.4.7.tar.gz

tar -zxfv v\*

cd s\*

查找openssl的位置

centos:

openssl version -a

mac:

brew info openssl，openssl version -a显示的不一样

phpize

./configure --enable-http2 --enable-openssl --with-openssl-dir=/usr/local/opt/openssl

make && make install

修改php.ini

php -i | grep php.ini

extension=swoole.so

swoole.use\_shortname=‘Off’

#### phpMyAdmin

下载使用yii6的帐号

rm -rf /home/yii6/public\_html/phpmyadmin/

cd /home/yii6/public\_html/

官网https://www.phpmyadmin.net/files/

wget https://files.phpmyadmin.net/phpMyAdmin/4.9.0.1/phpMyAdmin-4.9.0.1-all-languages.zip

unzip phpM\*

mv phpMyAdmin-4.7.8-all-languages phpmyadmin

域名/phpmyadmin不能访问phpmyadmin，直接下载文件，修改

http-vhost.conf

增加一个子域名，DocumentRoot 修改为相应的文件夹

报错1：

配置文件现在需要一个短语密码。

解决：

vim ~/public\_html/phpmyadmin/libraries/config.default.php

$ cfg['blowfish\_secret'] = '最长46位即23个字符';

vim ~/public\_html/phpmyadmin/config.sample.inc.php

$ cfg['blowfish\_secret'] = '32位即16个字符';

报错2：

“Failed to set session cookie. Maybe you are using HTTP instead of HTTPS”.

In some cases, clearing the phpMyAdmin cookies ('pma\*') resolves the issue.

问题没有解决

换了4.7.9版本的好了

报错3：

mysqli\_real\_connect(): (HY000/2002): No such file or directory的错误

解决：

把phpmyadmin目录中的配置文件config.sample.inc.php改成config.inc.php

打开编辑config.inc.php

找到：

$cfg['Servers'][$i]['host'] = 'localhost';

改成：

$cfg['Servers'][$i]['host'] = '127.0.0.1';

1. 1

clon第1.1章 git

1. gitee.com注册
2. 修改ssh公匙
3. 创建项目，如<https://gitee.com/yii6/yi_du_cloth>
4. 安装git，如Git-2.16.2-64-bit.exe

$ git config --global user.name “yii6”

$ git config --global user.email “i@yii6.com”

1. 打开git-bash.exe进入项目所在目录，初始化

git init

添加项目地址

git remote add origin <https://gitee.com/yii6/yi_du_cloth.git>

若报错：

fatal: remote origin already exists.

此时可以先查看远程库信息：

git remote –v

删除已有的远程库再添加：

git remote rm origin

git remote add origin <https://gitee.com/yii6/yi_du_cloth.git>

新建项目时，在码云平台仓库上已经存在 readme 文件，故在提交时可能会存在冲突，这时您需要选择的是保留线上的文件或者舍弃线上的文件，如果您舍弃线上的文件，则在推送时选择强制推送，强制推送需要执行下面的命令：

git push origin master -f

如果您选择保留线上的 readme 文件,则需要先执行：

git pull origin master

执行命令：

$ git push git@gitee.com:yii6/yi\_du\_cloth.git master -f

然后码云上可以看见推送上去的文件

本地删除或者修改后，进入本地仓库目录

git add .

git status

git commit -m “提交”

git push origin master

1. 拉取远程某个分支

git clone -b feature/marketing-plan-priority-MT-13441 ssh://git@git.innotechx.com:7999/x/pt-php-project.git

1. git命名规范

'docs', // 仅仅修改了文档，比如README, CHANGELOG, CONTRIBUTE等等

'chore', // Other changes that don't modify src or test files. 改变构建流程、或者增加依赖库、工具等

'feat', // 新增功能feature

'fix', // 修复bug

'merge', // Merge branch

'perf', // 优化相关，比如提升性能、体验

'refactor', // 代码重构，没有加新功能或者修复bug

'revert', // 回滚到上一个版本

'style', // 仅仅修改了空格、格式缩进、逗号等等，不改变代码逻辑

'test', // 测试用例，包括单元测试、集成测试等

'conf', //配置修改

示例：

git commit -m "feat: MT-13441 小程序首页点击报表"

style: MT-13441 格式化

分支命名规范

开发分支 feature/{key-workds}-{jira-id} , 例如feature/ios-verions-MT-14153

修复分支hotfix/{key-workds}-{jira-id}

发布分支release/{Ymd}, 例如release/20191006

1. 删除分支

首先切换分支

git checkout master

不能切换可以

git checkout .

强制删除分支,本地的文件要先备份

git branch -D feature/marketing-plan-priority-MT-13441

删除远程分支

git push origin --delete feature/marketing-plan-priority-MT-13441-dev

1. master合并到本地

git checkout master

git pull origin master

git checkout feature/marketing-plan-priority-MT-13441

git merge master

在出现的文件里面输入comment,要符合代码规范

创建本地新分支并切换到新分支:

git checkout -b feature/marketing-plan-priority-MT-13441-dev

1. 把一些不提交的改变保存下来

git stash list

git stash save

1. 回退操作1

git revert 'commit-hash'

填写commit

git push上去

回退操作2:

删除远程分支

回退本地分支有3种方式,选择hard这种

git reset --hard c5a8b8238d19ed6e4d0fdae50b4b2adf080c9d92

本地分支推到远程并新建远程分支

git push origin feature/MT-17639-link:feature/MT-17639-link

1. 1
2. 1

第1.2章 linux网络基础

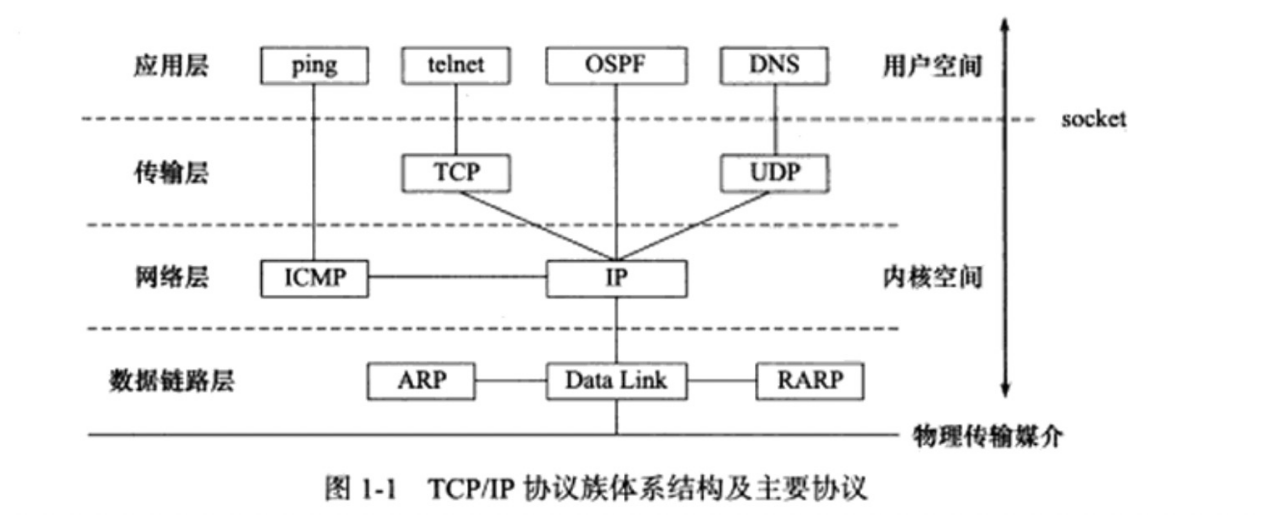
1. 查看软件版本

php --ri swoole

1. 查看php的配置文件的位置

php -i | grep ini

1. 网络协议



1. 查看mac的mac地址

ifconfig | grep ether | head -1

ether f0:18:98:43:b0:cc

1. 1

第2章 php基础

1. 双引号内的变量会被替换为变量的值，单引号内的全部视为字符串
2. 动态变量（一个变量的变量名可以动态的设置和使用）

$var=“apple”;

$$var=5;

echo “I have bought ${$var} {$var}s.”;

输出

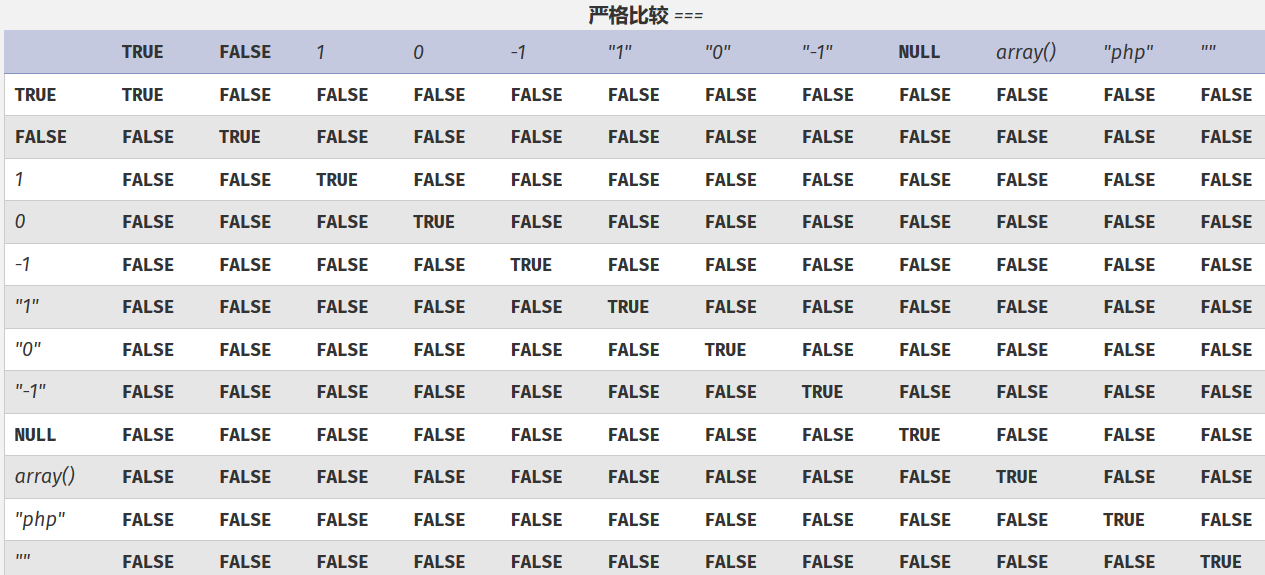
I have bought 5 apples.

1. session保存路径

文件php.ini

session.save\_path=“ “





参考:

https://www.php.net/manual/zh/types.comparisons.php

1. str\_replace(find,replace,string,count)

find 必需。要查找的值

replace 必需。替换 find 中的值的值

string 必需。被搜索的字符串

count 可选。对替换数进行计数的变量

返回值 带有替换值的字符串或数组

1. strip\_tags(string,allow) 将所有html标签删除

string 必需。

allow 可选。不会被删除的标签

返回值 处理后的字符串。

echo strip\_tags(“Hello <b><i>world!</i></b>“,”<b>“);//Hello **world**!

1. string mb\_substr( string $str, int $start[, int $length[, string $encoding]] );

start 开始截取的位置

length 截取的字符串的最大长度;若省略，则截取到末尾

encoding字符编码。若省略，则使用内部编码

1. strrchr() 函数查找字符串在另一个字符串中最后一次出现的位置，并返回从该位置到字符串结尾的所有字符。

<?php

$path = '/www/public\_html/index.html';

$filename = substr(strrchr($path, "/"), 1);//substr返回包括第1个字符在内的之后的字符

echo $filename; // "index.html"

?>

1. explode(separator,string[,limit]),字符串分割函数

limit 规定所返回的数组元素的数目。

> 0 - 返回包含最多 limit 个元素的数组

< 0 - 返回包含除了最后的 -limit 个元素以外的所有元素的数组

=0 - 返回包含一个元素的数组

implode([separator,]array) 返回由数组元素组成的字符串

1. 数组集合操作

合集:array\_merge的时候后面的数组会把前面数组里面相同的键名的值替换掉。

$conf1 = [

'username' => 'yii6',

];

$conf2 = [

'username' => 'yuwuy',

];

var\_dump(array\_merge($conf1,$conf2));

打印：

array(1) {

[“username”] => string(5) “yuwuy”

}

交集:

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");

$a2=array("e"=>"red","f"=>"green","g"=>"blue");

$result=array\_intersect($a1,$a2);

print\_r($result);

$result=array\_diff($a1,$a2);

print\_r($result);

?>

//Array ( [a] => red [b] => green [c] => blue )

差集(在array1中不在array2中):

//Array ( [d] => yellow )

1. 字符串不能和数字0比较，会返回真（如果PHP无法将字符串转换为整数，则假定它为0）

$str=“字符串”;

if($str==0){

echo “返回了true.”;

}

//会打印”返回了true”.

$a ?? 0 等同于 isset($a) ? $a : 0

$a ?: 0 等同于 $a ? $a : 0

empty: 判断一个变量是否为空(null,false,0,'0',"",array()都会返回true)

...表示不定数组

<?php  
//声明时使用  
function sum(...*$numbers*) {  
 $acc = 0;  
 foreach (*$numbers* as $n) {  
 $acc += $n;  
 }  
 return $acc;  
}  
echo sum(1, 2, 3, 4);//10  
//调用时使用  
function add(*$a*, *$b*) {  
 return *$a* + *$b*;  
}  
echo add(...[1, 2])."\n";//3  
$a = [1, 2];  
echo add(...$a);//3

1. 特殊字符导出到csv文件，因为csv文件是逗号分隔的，所以有逗号要处理

特殊字符导出到csv，存入数据库的时候会先转义，先反转义，然后如果有双引号要把双引号替换为2个引号，然后整个字符串加上引号；如果有逗号，整个字符串要加上双引号。

$row['realname'] = htmlspecialchars\_decode($row['realname'],ENT\_QUOTES);

if(strpos($row['realname'],'“')){

$row['realname']=str\_replace('“','““',$row['realname']);

$row['realname']='“'.$row['realname'].'“';

}else{

if(strpos($row['realname'],',')){

$row['realname']='“'.$row['realname'].'“';

}

}

1. ob\_clean();

//这个函数的作用就是清除下json数据前面的bom头?

ob\_clean();

显示操作失败，json前面有个小红点



1. 静态变量属于类，所有相同类的对象共用一个，继承类的静态变量看有没有覆写父类的静态变量。

使用 self:: 或者 \_\_CLASS\_\_ 对当前类的静态引用，取决于定义当前语句所在的类：

Example #1 self:: 用法

<?php  
class A {  
    public static function who() {  
        echo \_\_CLASS\_\_;// 当前语句在A中  
    }  
    public static function test() {  
        self::who();  
    }  
}  
  
class B extends A {  
    public static function who() {  
        echo \_\_CLASS\_\_;  
    }  
}  
  
B::test();  
?>

以上例程会输出：

A

后期静态绑定表示运行时最初调用的类来绕过限制。

Example #2 static:: 简单用法

<?php

class A {

public static function who() {

echo \_\_CLASS\_\_;

}

public static function test() {

static::who(); // 后期静态绑定从这里开始，使用static即表示子类能重写

}

}

class B extends A {

public static function who() {

echo \_\_CLASS\_\_;

}

}

B::test();

?>

以上例程会输出：

B

Example #3 非静态环境下使用 static::

<?php

class A {

private function foo() {

echo \_\_CLASS\_\_.”\nA<br>“;

}

public function test() {

$this->foo();

static::foo();

}

}

class B extends A {

/\* foo() will be copied to B, hence its scope will still be A and

\* the call be successful \*/

}

class C extends A {

private function foo() {

/\* original method is replaced; the scope of the new one is C \*/

echo \_\_CLASS\_\_.”\nC<br>“;

}

}

$b = new B();

$b->test();

$c = new C();

$c->test(); /\*static::foo();报错 Fatal error: Call to private method C::foo() from context 'A'

在A域里面调用C的私有方法出错\*/

?>

Example #4 转发和非转发调用

class A {

public static function foo() {

static::who();

}

public static function who() {

echo \_\_CLASS\_\_.”\n”;

}

}

class B extends A {

public static function test() {

A::foo();//A调用A自己的foo()，static指向调用者当然为A

parent::foo();/\*parent 指向的是定义它的类的父类,

指的是当前类的父类A,调用者依然是C，

parent只是起到了”转发调用者”的作用。

A中的foo内的static解析到C，最后运行的是C的who()，输出的是\_\_CLASS\_\_为C\*/

self::foo();/\*self 指向 定义它的类。也就是你在哪里写了 self，self 就代表哪个类,同样的转发调用者，是C要调用B中的foo() \*/

}

public static function who() {

echo \_\_CLASS\_\_.”\n”;

}

}

class C extends B {

public static function who() {

echo \_\_CLASS\_\_.”\n”;

}

}

C::test();//test()来自B。结果为A C C

1. 抽象类不能创建实例，它只能作为父类被继承。抽象类是从多个具体类中抽象出来的父类，它具有更高层次的抽象。从多个具有相同特征的类中抽象出一个抽象类，以这个抽象类作为其子类的模板，从而避免了子类的随意性。

(1) 抽象方法只作声明，而不包含实现，可以看成是没有实现体的虚方法

(2) 抽象类不能被实例化

(3) 抽象类可以但不是必须有抽象属性和抽象方法，但是一旦有了抽象方法，就一定要把这个类声明为抽象类

(4) 具体派生类必须覆盖基类的抽象方法

(5) 抽象派生类可以覆盖基类的抽象方法，也可以不覆盖。如果不覆盖，则其具体派生类必须覆盖它们

1. 接口是什么：

(1) 接口不能被实例化

(2) 接口只能包含方法声明

(3) 接口的成员包括方法、属性、索引器、事件

(4) 接口中不能包含常量、字段(域)、构造函数、析构函数

1. 接口和抽象类的区别：

（1）抽象类可以有构造方法，接口中不能有构造方法。

（2）抽象类中可以有普通成员变量，接口中没有普通成员变量，抽象类和接口中都可以包含静态成员变量，接口中只能且默认是public static final类型

（3）抽象类中可以包含静态方法、构造函数、非抽象的普通方法，接口中不能有静态方法、构造函数、非抽象的普通方法；抽象类中的抽象方法的访问类型可以是public，默认protected , 但接口中的抽象方法只能是public，默认为public abstract

（4）一个类可以实现多个接口，但只能继承一个抽象类，接口也可以继承其他接口

（5）接口可以被多重实现，抽象类只能被单一继承

（6）如果抽象类实现接口，则可以把接口中方法映射到抽象类中作为抽象方法而不必实现，而在抽象类的子类中实现接口中方法

1. 抽象类中的非抽象方法可以有方法体，这样的方法就不是抽象方法了，前面也不能再加abstract修饰，继承它的子类的也不必非要去实现它（根据实际情况而定）。
2. 抽象方法和普通的主要表现区别
3. 抽象方法不具有方法体，方法体是一个方法的一个代码块，即花括号包裹的部分
4. 声明抽象方法的关键字abstract，如

abstract function Demo ();

1. 如果一个类中包含抽象方法，则该类必须声明为抽象类
2. php函数的参数，boolean要求传一个对象，bool要求传一个bool值

<?php

function test(boolean $param) {}

test(true);

?>

以上例程会输出：

Fatal error: Uncaught TypeError: Argument 1 passed to test() must be an instance of boolean, boolean given, called in - on line 1 and defined in -:1

1. compact() 在当前的符号表中查找该变量名并将它添加到输出的数组中，变量名成为键名而变量的内容成为该键的值；如果字符串指向的变量未定义，compact() 会产生 E\_NOTICE 级错误。 在php7.3.0之前，此类问题会静默忽略。

<?php  
$firstname = "Peter";  
$lastname = "Griffin";  
$age = "41";  
$name = array("firstname", "lastname");  
$result = compact($name, "location", "age");  
print\_r($result);  
?>

//Array ( [firstname] => Peter [lastname] => Griffin [age] => 41 )

1. new和make的区别

new(T) 为每个新的类型T分配一片内存，初始化为 0 并且返回类型为\*T的内存地址：这种方法 返回一个指向类型为 T，值为 0 的地址的指针，它适用于值类型如数组、结构体；它相当于 &T{}。

make(T) 返回一个类型为 T 的初始值，它只适用于3种内建的引用类型：切片、map 和 channel，因为切片这些本身是引用类型，所以不需要返回引用的引用.

1. require、include是为了引入文件；namespace 是为了管理类，一般和实际的文件路径相同，方便查找文件；use是引入该命名空间下的某类。

为了使用命名空间必须先将命名空间出现的文件包含进来，而一般框架自动把代码加载了，所以代码里面省略了加载相应命名空间所在文件的代码。

a.php

<?php

namespace ns\a;

class a{

}

b.php

<?php

namespace ns\b;

class a{

}

c.php

<?php

require\_once('a.php');

require\_once('\b.php');

use ns\a\a; //使用命名空间ns\a中的类a

use ns\b\a as b;//使用命名空间ns\b中的类a 并用b替换

1. 北京时间是从1970年01月01日08时00分00秒开始,所以 时间戳/86400 得到的并不是一个整数.
2. 记录脚本内存使用量

print\_r('内存使用：' . memory\_get\_usage() / 1024 / 1024 . " M\n");

设置内存使用上限:

ini\_set('memory\_limit', '10240M');

1. 创建对象时的几个关键应用规则：
2. 避免在循环体中创建对象，即使该对象占用内存空间不大。

一个错误示范:

for(int i=0; i<10000; ++i)

{

Object obj = new Object();

}

一般应该在循环体外创建这个对象,然后操作

（2）尽量及时使对象符合垃圾回收标准。

（3）不要采用过深的继承层次。

1. 访问本地变量优于访问类中的变量。
2. 魔术方法

在读取不可访问（私有或者不存在）的属性时，\_\_get()会被调用，

在给不可访问的属性赋值时，\_\_set()会被调用。

1. usort会把源数组键名去掉，uasort保留键名和键值的对应关系

function my\_sort($a,$b)

{

if ($a==$b)

return 0;

return ($a<$b)?-1:1;

}

$arr=array("a"=>4,"b"=>2,"c"=>8,'d'=>"6");

uasort($arr,"my\_sort");

// Key=b, Value=2

// Key=a, Value=4

// Key=d, Value=6

// Key=c, Value=8

usort($arr,"my\_sort");

// Key=0, Value=2

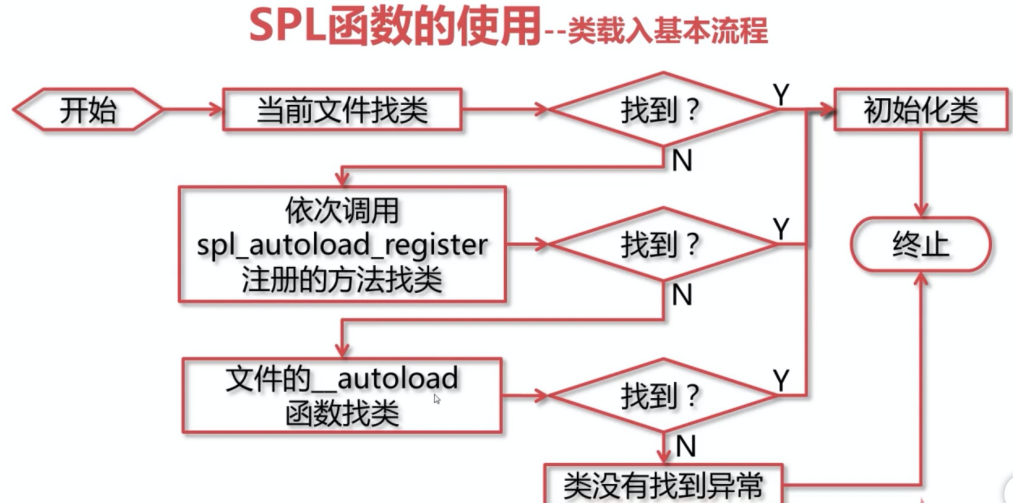
// Key=1, Value=4

// Key=2, Value=6

// Key=3, Value=8

1. 自动加载

<?php  
//设置需要加载的扩展名；先查找前面的扩展名  
*spl\_autoload\_extensions*('.class.php,.php');  
//设置需要自动加载的路径  
*set\_include\_path*(*get\_include\_path*() . *PATH\_SEPARATOR* . "libs/");  
//注册自动加载的方法  
*spl\_autoload\_register*();  
new Test();



1. 查询某个类实现的所有接口

<?php  
interface foo{}  
interface boo{}  
class bar implements foo{}  
class zoo extends bar implements boo{}  
*print\_r*(*class\_implements*(new bar));  
*print\_r*(*class\_implements*('zoo'));  
/\*  
Array  
(  
 [foo] => foo  
 [boo] => boo  
)  
\*/

class\_parents返回所有的父类

1. 查看php内置函数的源码

找到php的安装目录

cd [/usr/local/Cellar/php@7.2/7.2.29](mailto:/usr/local/Cellar/php@7.2/7.2.29)

find /usr/local/Cellar/php@7.2/7.2.29 -name "ext" -type d

进入ext的上层目录

cd /usr/local/Cellar/php@7.2/7.2.29/include/php

grep -rn "PHP\_FUNCTION(uniqid)" ./ext

./ext/standard/uniqid.h:25:PHP\_FUNCTION(uniqid);

去GitHub看源码

https://github.com/php/php-src/blob/master/ext/standard/uniqid.c

1. php扩展

使用高级语言编写PHP扩展

1. 1
2. 1
3. 1
4. 1
5. 1
6. 1
7. 1
8. 1

第2.1章 php编码规范

1. Class names MUST be declared in StudlyCaps.

类名用大小字母开头的驼峰

Class constants MUST be declared in all upper case with underscore separators.

类常量用全大写字母和下划线

Method names MUST be declared in camelCase.

方法用小写字母开头的驼峰

变量可以用驼峰或者$under\_score（小写字母+下划线），但是同一文件中应该统一

1. 函数注释规范

/\*\*

\* 函数的含义说明

\*

\* @access public

\* @param mixed $arg1 参数一的说明

\* @param mixed $arg2 参数二的说明

\* @param mixed $mixed 这是一个混合类型

\* @return array 返回类型

\*/

1. 1

第2.2章 docker

1. yml文件里面,前面是本地,后面是容器里面的,如

ports:

- 80:80

1. 基本docker命令 :

docker-compose up -d ms

up表示上线,d表示deamonize,ms是service的名字

docker-compose exec ms bash

进入ms这个容器，使用bash这个shell

docker ps a



查看当前的docker容器

docker rm -f a2f5d6c25b5a

强制删除container id为a2f5d6c25b5a的容器;也可以stop

1. 挂载的是根据yml文件里面设置的目录

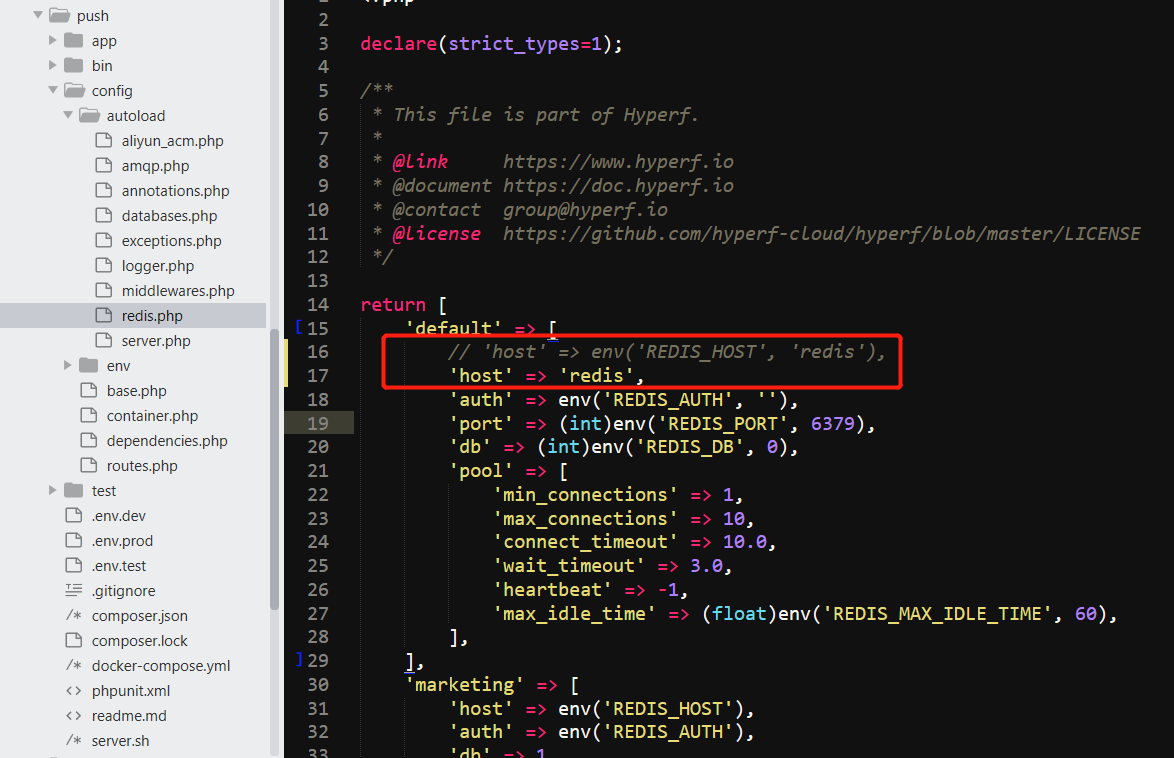
docker-compose up -d mysql redis rabbitmq

启动服务，本地可以用服务名字，配置了本地vpn后用内网ip

php bin/hyperf.php start 报错

redis parameter 1 should be string null given

修改项目里面的redis配置：



下面的marketing、wxmp的也要修改

1. 安装包括hyperf在内的一些插件

composer install

1. 文件修改好了之后重启hyperf的服务，即关闭当前进程，重启hyperf

第3章 阿里大鱼短信

1. 参考https://help.aliyun.com/document\_detail/59210.html
2. 下载php的sdk和demo

ytx-sdk.oss-cn-shanghai.aliyuncs.com/dysms\_php.zip?spm=a2c4g.11186623.2.4.Anqe0n&file=dysms\_php.zip

解压

将aliyun-dysms-php-sdk改名为alisms

将alisms上传到advanced/vendor目录下，msg\_demo和msg\_sdk如果用不到可以删除

修改config/main.php

return [

'aliases'=>[

'@alisms/api\_demo'=>'@vendor/alisms/api\_demo',

],

];

修改SmsDemo.php

在最上方加上命名空间：

namespace alisms\api\_demo;

根据注释修改相应的地方。

调用

use alisms\api\_demo\SmsDemo;

$sms=new SmsDemo();

$response=$sms->sendSms();

充值后即可发送成功。

1. 1
2. 1

第4章 Yii 2

1. 通过composer安装yii的basic版到需要的目录，例如域名根目录(如/home/yuwuy)的子域名e目录下

composer官网有安装方法:

**https://getcomposer.org/download/**

**mv composer.phar /usr/local/bin/composer**

**如果没有root权限就按照官网的方法使用**

**$ composer global require “fxp/composer-asset-plugin:~1.4.5”**

public\_html是www的根目录

**$ composer create-project --prefer-dist yiisoft/yii2-app-advanced advanced**

升级可以使用下面的命令:

**[~/public\_html/e/basic]$ composer update yiisoft/yii2 yiisoft/yii2-composer bower-asset/jquery.inputmask**

使用 Composer 的时候提示输入 Token：

进入https://github.com/settings/tokens 点击 「Generate new token」，复制、粘贴到

Token(hidden)：后回车

00a5d0ec3d0ccb9d02988241fa033aace77868af

检测运行环境：

**[~/public\_html/e/basic]$ php requirements.php**

[~/public\_html/advanced]$ ./init

设置子域名admin.yuwuy.com的根目录:

/public\_html/advanced/backend/web

设置子域名的dns:



1. composer安装图床

**[~/public\_html/e/basic]$ composer require --prefer-dist crazyfd/yii2-qiniu “dev-master”**

国外图床(未搜索测试)：

Photobucket.com 、imageshack.us、 photohost.info

极简图床、sm.ms图床curl上传图片：

https://blog.skyx.in/archives/164/

1. bootstrapTags组件安装

1).下载https://github.com/niluzok/yii2-bootstrap-tags

2).解压重命名为 bootstrapTags，复制到 common\widgets下

3).修改BootstrapTags.php 和 BootstrapTagsAsset.php 的命名空间为namespace common\widgets\bootstrapTags;

4).[~/public\_html/advanced]$ composer require --prefer-dist bower-asset/bootstrap-tags “\*”

5).使用时在views中添加

<?= $form->field($model,'tags')->widget('common\widgets\bootstrapTags\BootstrapTags' ]); ?>

7).在标签表单中输入标签后回车就可以得到标签

1. jstree安装

[~/public\_html/advanced]$ composer require --prefer-dist yii-dream-team/yii2-jstree “\*”

1. sentry安装

composer require --prefer-dist mito/yii2-sentry “\*”

如果报错

The requested package bower-asset/jquery could not be found in any version,

安装一下下面的插件

composer require --prefer-dist fxp/composer-asset-plugin “\*”

报错找不到bower文件

vim htdoc/vendor/yiisoft/yii2/base/Application.php

:456行

1. 如果不是从localhost访问gii，服务器上使用gii时需要把自己的ip加入配置文件

修改basic/config/web.php 或者advanced\backend\config\main-local.php

80 $config['modules']['gii'] = [

81 'class' => 'yii\gii\Module',

82 'allowedIPs'=>['127.0.0.1','::1','117.85.148.198','192.168.1.110'],

83 ];

添加language可以把rules的提示变成中文

$config = [

'language'=>'zh-CN',

]

1. form的hiddenInput还是会产生label

’disabled’=>true则不能编辑，不能post传值

’readonly’=>true则不能编辑，可以post传值

1. ueditor

http://www.yii-china.com/post/detail/3.html

1. 数据分页
2. 数据库数据的分页

controller:

public function actionUserlist()

{

$userdata = User::find()->joinWith('user\_info');

$count = $userdata->count();

$pageSize = yii::$app->params['pageSize']['user'];

$page= new Pagination(['totalCount'=>$count,'pageSize'=> $pageSize]);

$users= $userdata->offset($page->offset)->limit($page->limit)->all();

return $this->render(“userlist”, ['users' => $users, 'page' => $page]);

}

views:

<td>

<?php echo isset($user->city) ? $user->city : '未查询'; ?>

</td>

<td>

<?php echo isset($user->user\_info->id\_no) ? $user->user\_info->id\_no : '未实名验证'; ?>

</td>

<?php

echo yii\widgets\LinkPager::widget(['pagination'=>$page, 'prevPageLabel' => '&#8249;', 'nextPageLabel' => '&#8250;']);

?>

1. 数组数据的分页

controller:

public function actionCatelist()

{

$model = new Cate();

//data是一个数组

$data = $model->getTreelist();

$dataProvider = new ArrayDataProvider([

'allModels' => $data,

'pagination' => [

'pageSize' => 10,

],

]);

$cates = $dataProvider->getModels();

return $this->render('catelist', ['page' => $dataProvider->pagination, 'cates' => $cates]);

}

views:

<?php foreach ($cates as $cateid => $catename): ?>

<tr>

<td>

<?php echo $cateid; ?>

</td>

<td>

<?php echo $catename; ?>

</td>

</tr>

<?php endforeach;?>

1. RBAC

建表

./yii migrate --migrationPath=@yii/rbac/migrations

报错：

You should configure “authManager” component to use database before executing this migration.

修改C:\Project\mhdc\addons\ws\_micro\_shop\core\config\console.php

进入advanced目录

导入目录到数据库

./yii rbac/init

修改frontend/controllers/BaseController.php

修改backend/views/layouts/main.php

如果**$auth=yii::$app->authManager;**未获得$auth，那么在config的components中加

'authManager' => [

'class' => 'yii\rbac\DbManager',

'itemTable' => '{{%auth\_item}}',

'itemChildTable' => '{{%auth\_item\_child}}',

'assignmentTable' => '{{%auth\_assignment}}',

'ruleTable' => '{{%auth\_rule}}',

'defaultRoles' => ['default'],

],

1. frontend里面的htaccess是为了pretty url用的
2. 1
3. 1

第4.1章 TP

1. 安装

composer 安装：

git安装（失败）：

git clone --depth=1 [git@github.com:top-think/think.git](mailto:git@github.com:top-think/think.git) tp

报错：

The authenticity of host 'github.com' can't be established

解决1：add an ssh key to your github account

<https://github.com/settings/keys>

1. $\_ENV的值打印不出来

修改/etc/php.ini

将variables\_order = “GPCS”

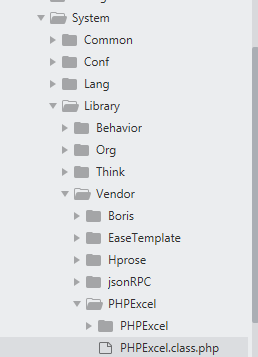
改为variables\_order = “EGPCS”

1. 数据库导出到excel

参考：<https://segmentfault.com/a/1190000009727749>

Github: <https://github.com/PHPOffice/PHPExcel> 下载

把classes拷贝到Library/Vendor目录下，改名为PHPExcel，将PHPExcel.php，重命名成PHPExcel.class.php。



public function settlement()

{

if ($\_GET['settle'] == “批量导出”) {

$xlsCell = array(

array('order\_id', '订单ID'),

array('uid', '用户ID'),

array('mobile', '手机号'),

array('type', '类型'),

array('title', '充值业务'),

array('money', '实付款'),

array('status', '订单状态'),

array('create\_time', '下单时间'),

array('deduction', '佣金抵扣'),

);

$xlsName = '广州瓦伦订单';

$field = null;

foreach ($xlsCell as $key => $value) {

if ($key == 0) {

$field = $value[0];

} else {

$field .= “,” . $value[0];

}

}

$xlsModel = M('zql\_order\_recharge');

unset($\_GET['settle']);

$where['id'] = array('in', $\_GET);

$xlsData = $xlsModel->Field($field)->where($where)->order('id DESC')->select();

foreach ($xlsData as $k => $v) {

$xlsData[$k]['type'] = $v['type'] == 'add\_flow' ? '充流量' : '充话费';

switch ($v['status']) {

case 1:

$xlsData[$k]['status'] = '付款成功，待充值';

break;

case 2:

$xlsData[$k]['status'] = '充值成功';

break;

case 3:

$xlsData[$k]['status'] = '充值失败';

break;

default:

$xlsData[$k]['status'] = '未付款';

}

$xlsData[$k]['create\_time'] = date('Y-m-d H:i:s', $v['create\_time']);

}

$this->exportExcel($xlsName, $xlsCell, $xlsData);

} else {

}

}

/\*\*

\* 数据导出为.xls格式

\* @param string $fileName 导出的文件名

\* @param $expCellName array -> 数据库字段以及字段的注释

\* @param $expTableData Model -> 连接的数据库

\*/

protected function exportExcel($fileName = '广州瓦伦订单', $expCellName, $expTableData)

{

$xlsTitle = iconv('utf-8', 'gb2312', $fileName); //文件名称

$xlsName = $fileName . date(“-Y.m.d”); //or $xlsTitle 文件名称可根据自己情况设定

$cellNum = count($expCellName);

$dataNum = count($expTableData);

import(“Vendor.PHPExcel.PHPExcel”);

import(“Vendor.PHPExcel.Writer.Excel5”);

import(“Vendor.PHPExcel.IOFactory.php”);

$objPHPExcel = new \PHPExcel();

$cellName = array('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'AA', 'AB', 'AC', 'AD', 'AE', 'AF', 'AG', 'AH', 'AI', 'AJ', 'AK', 'AL', 'AM', 'AN', 'AO', 'AP', 'AQ', 'AR', 'AS', 'AT', 'AU', 'AV', 'AW', 'AX', 'AY', 'AZ');

$objPHPExcel->getActiveSheet(0)->mergeCells('A1:' . $cellName[$cellNum - 1] . '1'); //合并单元格

$objPHPExcel->setActiveSheetIndex(0)->setCellValue('A1', $fileName . ' 导出时间:' . date('Y-m-d H:i:s'));

for ($i = 0; $i < $cellNum; $i++) {

$objPHPExcel->setActiveSheetIndex(0)->setCellValue($cellName[$i] . '2', $expCellName[$i][1]);

}

// Miscellaneous glyphs, UTF-8

for ($i = 0; $i < $dataNum; $i++) {

for ($j = 0; $j < $cellNum; $j++) {

$objPHPExcel->getActiveSheet(0)->setCellValue($cellName[$j] . ($i + 3), “ “ . $expTableData[$i][$expCellName[$j][0]]);

}

}

header('pragma:public');

header('Content-type:application/vnd.ms-excel;charset=utf-8;name=“' . $xlsTitle . '.xls”');

header(“Content-Disposition:attachment;filename=$xlsName.xls”); //attachment新窗口打印inline本窗口打印

$objWriter = \PHPExcel\_IOFactory::createWriter($objPHPExcel, 'Excel5');

$objWriter->save('php://output');

exit;

}

1. 1
2. 1

第4.2章 微擎

1. 上传附件大小设置修改

全局搜索：当前 PHP 环境允许最大单个上传文件大小为

找到文件attachment.html

找到关键词：upload\_max\_filesize

找到文件：rlkf.wshoto.com\web\source\system\attachment.ctrl.php

根据提示搜索关键词：以内的视频

找到文件：fileuploader.min.js

找到关键词：fileSingleSizeLimit

找到代码：

**fileNumLimit: l,**

**fileSizeLimit: n,**

**fileSingleSizeLimit: m**

向上看一下，找到定义m大小的地方，修改完成设置

j = “视频”, typeUnit = “个”, k = {

title: “Video”,

extensions: “rm,rmvb,wmv,avi,mpg,mpeg,mp4”,

mimeTypes: “video/\*”

}, l = 30, **m = 50\*1024\*1024**, n = 629145600,

1. 1
2. 1

第4.3.1章 swoole

1. swoole扩展了php脚本的生命周期，无需nginx和fpm就能启动一个web服务。并且从服务启动前，启动后，连接进入，请求到来，请求结束，连接切断，服务终止所有的生命周期都在php脚本的掌控中。
2. swoole三大特性：

常驻进程：传统fpm请求结束把内存释放掉

异步

协程：保存了函数栈

swoole\_set\_process\_name('yii6 process');

在脚本中运行错误：

Exception: swoole\_set\_process\_name(): cli\_set\_process\_title had an error: Not initialized correctly

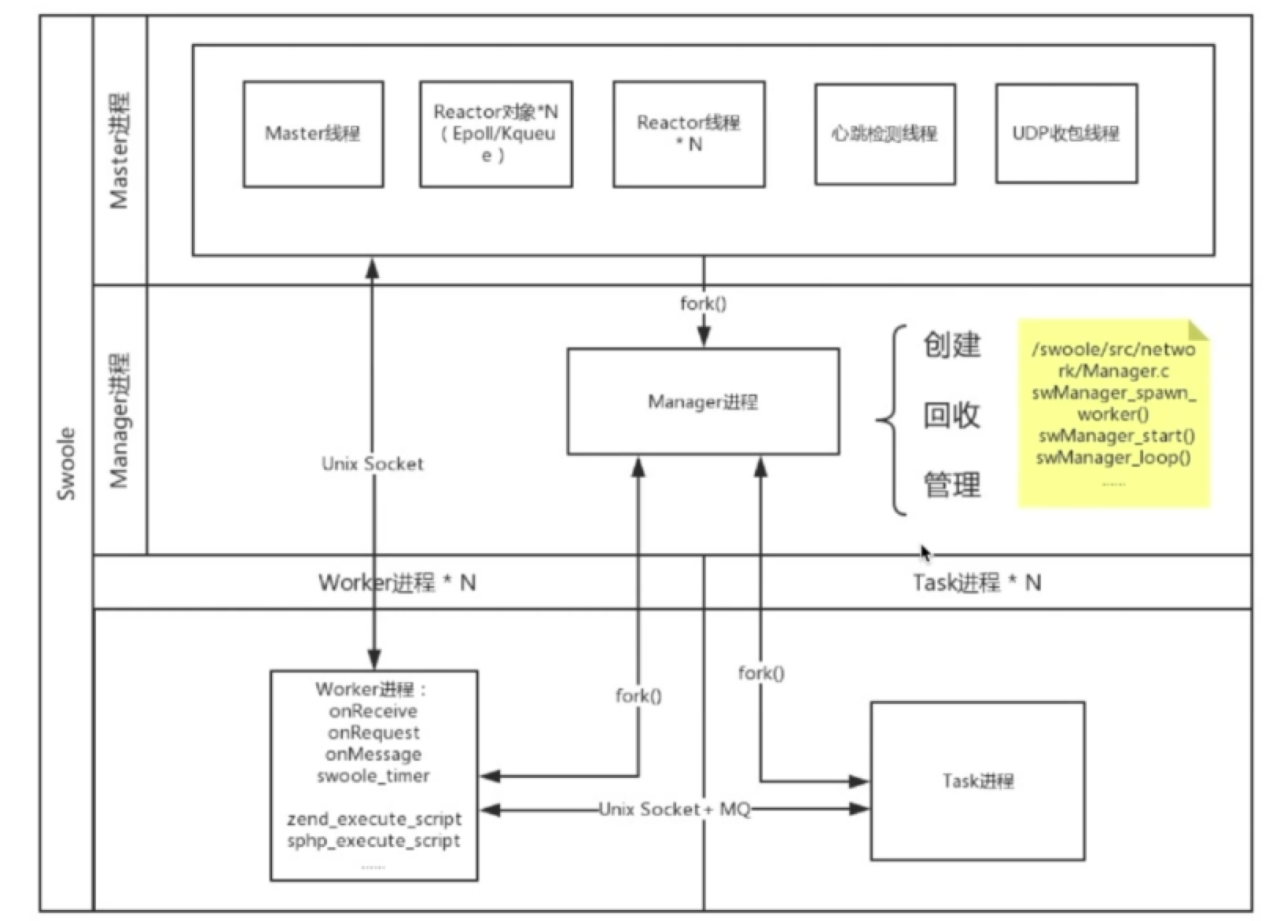
在mac里面因为安全问题不允许

在yii2框架中报错：

//swoole\_set\_process\_name(): swoole\_set\_process\_name only support in CLI mode

在centos中直接运行可以成功

1. swoole运行模式



3种进程：

master进程（运行php脚本创建的进程）：管理连接，

manager进程 ：管理进程的创建、回收和状态

worker进程（task进程是worker进程中的一种）：负责php代码逻辑和脚本执行

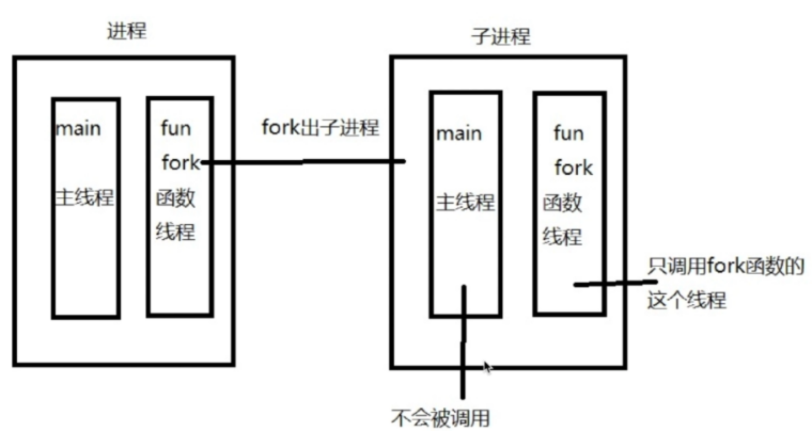
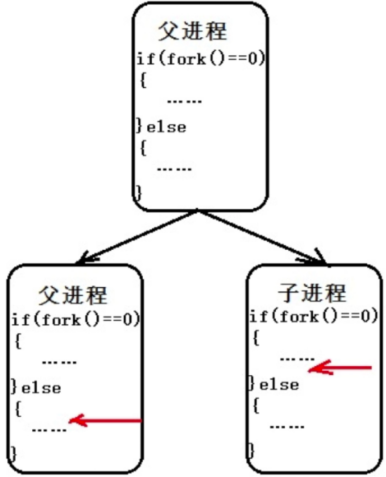
心跳检测：遍历大量的连接筛选出不活跃的连接把它剔除。

线程安全:能正确处理多个线程之间的共享变量。

可重入：在任意的时刻被中断然后再次调用该子程序不会出错。

解决方案一：

多个线程，每个线程处理不同连接上的内容，就不会有数据安全的问题



线程fork的时候没有fork全部线程，例如当前线程拿了一个全局的锁，然后fork了,新的线程里面锁没有线程去释放，那这个资源就成了死锁

1. 协程间通信使用channel
2. 问题

连接已关闭

swFactoryProcess\_finish(errno 1004):send 165 byte failed,because connection[fd=325] is closed

可能原因：上一个响应还没有返回就把连接关闭了

PHP Fatal error: Uncaught Error: Call to undefined function go()

原因：

php.ini配置过 swoole.use\_shortname = "Off" ，注释掉，

然后检验显示如下就可以用了

php --re swoole | grep go

Function [ <internal:swoole> function go ] {

Method [ <internal:swoole> public method goaway ] {

1. Swoole\Coroutine\stats()

coroutine\_peak\_num:协程峰值数量

aio\_worker\_num:aio线程的数量，线程池，数量是动态伸缩的

1. 协程连接池是进程级别的
2. 1

第4.3.2章 Hyperf

1. 项目安装，在项目目录下安装一个最新的名字叫hyperf的项目

composer create-project hyperf/hyperf-skeleton hyperf

然后全部直接回车完成安装

php -v

（NTS not thread safe）

php --ri swoole

没有安装需要下swoole

wget <https://github.com/swoole/swoole-src/archive/v4.4.3.tar.gz>

tar -zxvf v4.4.3.tar.gz

cd swoole-src-4.4.3

编译一下

phpize

./configure --enable-http2 --enable-openssl

--with-openssl-dir=/usr/local/opt/openssl

make && make install

修改php.ini

extension=swoole.so

swoole.use\_shortname = 'Off'

1. 构建一个最基本的路由只需一个URI和一个闭包(Closure)：

<?php

use Hyperf\HttpServer\Router\Router;

Router::get('/hello-hyperf', function () {

return 'Hello Hyperf.';

});

通过 浏览器 或 cURL 命令行来请求 http://host:port/hello-hyperf 来访问该路由。

标准路由指的是由 控制器(Controller) 和 操作(Action) 来处理的路由，使用 请求处理器(Request Handler) 模式也是类似的。

// 下面三种方式的任意一种（也仅可选一种）都可以达到同样的效果

Router::get('/hello-hyperf', 'App\Controller\IndexController::hello');

Router::get('/hello-hyperf', 'App\Controller\IndexController@hello');

Router::get('/hello-hyperf',[App\Controller\IndexController::class, 'hello']);

该定义将 /hello-hyperf 路径绑定到 App\Controller\IndexController 下的 hello 方法。

本框架定义的路由参数必须和控制器参数键名、类型保持一致，否则控制器无法接受到相关参数。

Router::get('/user/{id}', 'App\Controller\UserController::info')

public function index(int $id)

{

$user = User::find($id);

return $user->toArray();

}

通过 {} 来声明必填参数，如 /user/{id}。

通过 [] 来声明可选参数，如 /user/[{id}]。

1. 控制器

<?php

declare(strict\_types=1);

namespace App\Controller;

use Hyperf\HttpServer\Contract\RequestInterface;

use Hyperf\HttpServer\Contract\ResponseInterface;

class IndexController

{

// 在参数上通过定义 RequestInterface 和 ResponseInterface 来获取相关对象，对象会被依赖注入容器自动注入

public function index(RequestInterface $request, ResponseInterface $response)

{

$target = $request->input('target', 'World');

return 'Hello ' . $target;

}

}

我们假设该 Controller 已经定义了路由为 /，通过 cURL 调用该地址，即可看到返回的内容。

$ curl http://127.0.0.1:9501/\?target\=Hyperf

Hello Hyperf.

1. DI(Dependency Injection)容器是负责管理对象的创建和对象的依赖管理的

DI容器创建出来的对象是个单例，是长生命周期对象

希望通过DI来管理,但是又希望创建短生命周期的对象,

通过$container->make()或make()函数创建短生命周期对象

通过new来实例化的对象注解不会生效，依赖需自行管理

为什么用Inject?因为这样可以把精力只集中在和自身功能相关的操作,而不用去一个一个的new依赖的对象.比如你需要发送邮件,可能要先new User,然后new Mail.用容器管理依赖之后,你就可以只需要注入Mail

在hperf/config/autoload/dependencies.php里面进行接口类与实现类的绑定

DI管理的对象是单例

class Test

{

private static $obj = null;

private function \_\_construct()

{

//禁止创建对象

}

//后门

public static function getObject()

{

if (self::$obj === null) {

self::$obj = new self();

}

return self::$obj;

}

}

工厂模式注入:

<?php

namespace App\Service;

use Hyperf\Contract\ConfigInterface;

use Psr\Container\ContainerInterface;

class UserServiceFactory

{

// 实现一个 \_\_invoke() 方法来完成对象的生产，方法参数会自动注入一个当前的容器实例

public function \_\_invoke(ContainerInterface $container)

{

$config = $container->get(ConfigInterface::class);

// 我们假设对应的配置的 key 为 cache.enable

$enableCache = $config->get('cache.enable', false);

// make(string $name, array $parameters = []) 方法等同于 new ，使用 make() 方法是为了允许 AOP 的介入，而直接 new 会导致 AOP 无法正常介入流程

return make(UserService::class, compact('enableCache'));

}

}

<?php

namespace App\Service;

class UserService implements UserServiceInterface

{

/\*\*

\* @var bool

\*/

private $enableCache;

public function \_\_construct(bool $enableCache)

{

// 接收值并储存于类属性中

$this->enableCache = $enableCache;

}

public function getInfoById(int $id)

{

return (new Info())->fill($id);

}

}

在 config/autoload/dependencies.php 调整绑定关系：

<?php

return [

\App\Service\UserServiceInterface::class => \App\Service\UserServiceFactory::class

];

这样在注入 UserServiceInterface 的时候容器就会交由 UserServiceFactory 来创建对象了。

注意：不要把模型注入service层，因为注入之后会常驻内存，数据查询出来后也会常驻内存，在首次查询之后，数据是正确的，但是第二次之后数据可能是污染了的。

1. 注解

参数传递(必须用双引号,单引号无效)：

传递单个参数 @DemoAnnotation("value")

传递字符串参数 @DemoAnnotation(key1="value1",key2="value2")

传递数组参数 @DemoAnnotation(key={"value1","value2"})

注解类的 @Annotation 和 @Target 注解为全局注解，无需 use

其中 @Target 有如下参数：

METHOD 注解允许定义在类方法上

PROPERTY 注解允许定义在类属性上

CLASS 注解允许定义在类上

ALL 注解允许定义在任何地方

1. 获取指定输入值

通过 input(string $key, $default = null) 和 inputs(array $keys, $default = null): array 获取 一个或多个任意形式的输入值：

// 存在则返回，不存在则返回 null

$name = $request->input('name');

// 存在则返回，不存在则返回默认值 Hyperf

$name = $request->input('name', 'Hyperf');

如果传输表单数据中包含「数组」形式的数据，那么可以使用「点」语法来获取数组：

$name = $request->input('products.0.name');

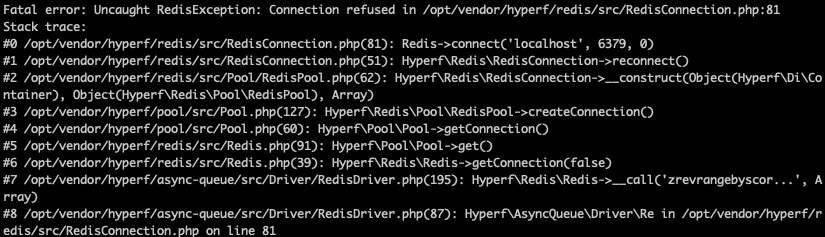
$names = $request->input('products.\*.name');

1. 路由;

404 没找到路由,看方法有没有注册

405 方法不允许访问,看注册的是get还是post方法,只能用相应的方法访问

1. 报错:



检查一下有没有.env文件

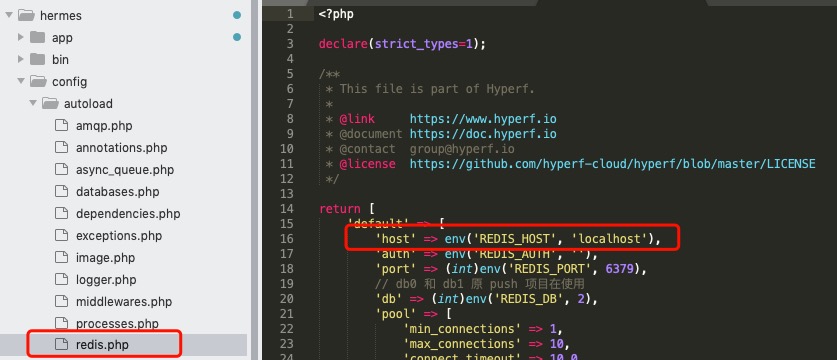
切换到项目目录

cd /Users/admin/opt/label-dynamic-group/service/hermes

cp .env.dev .env

还有redis的报错,可以尝试修改项目下面的redis配置,

进到docker容器里面ping一下redis的host,看看报错



'host'=>'redis'

1. 切面AOP

模块与模块之间，方法与方法之间的这个gap就是切面

AOP应用场景：

参数校验、日志、无侵入埋点、安全控制、性能统计、事务处理、异常处理、缓存、无侵入监控、资源池、连接池管理等。

通过DI得到的类为代理类,保存在

runtime/container/proxy

是一个继承原类的子类,原类方法会被重写;这个类只会生成一次,所以如果想重新生成,需要把这个文件夹删除,然后Hyperf启动的时候就会把代理类全部重新生成了

1. 错误日志

StdoutLoggerInterface

1. 协程上下文
2. 单元测试

/\*\*

\* @dataProvider getAdd

\* @param $a

\* @param $b

\* @param $expected

\*/

public function testGetGivenChannelMarketingPlans(int $a, int $b, int $expected)

{

$c = $this->call\_private\_method(new PlanService(), 'add', [$a, $b]);

$this->assertSame($expected, $c);

}

public function getAdd()

{

return [

[2, 3, 5],

[1, 2, 3],

];

}

#### amqp

#### model

php bin/hyperf.php gen:model --pool=pt\_config --path=app/MallMarketing/Model short\_data

1. exchange types，和routing key相关

direct:完全匹配

fanout:类似于广播

topic:#匹配多个词，\*匹配单个词

1. #可以匹配a.b和a.b.c

a.\*只会匹配到a.b

1. 环境配置

http://km.innotechx.com/pages/viewpage.action?pageId=82256420

前置条件

brew install node(会自动安装npm)

前端：

git clone ssh://git@git.innotechx.com:7999/x/pt-rock-admin.git

cd pt-rock-admin

npm i

npm run build:test

npm run dev

后端：

git clone ssh://git@git.innotechx.com:7999/x/pt-php-project.git

cd pt-php-project/service/omsapi

composer i

composer watch

1. 1

第4.4章 yaf

1. 安装，

报错1：

Class 'Yaf\_Application' not found in

解决：

yaf.use\_namespace打开了，但是new那个类的时候不是命名空间的写法，所以在php.ini里面修改yaf的配置

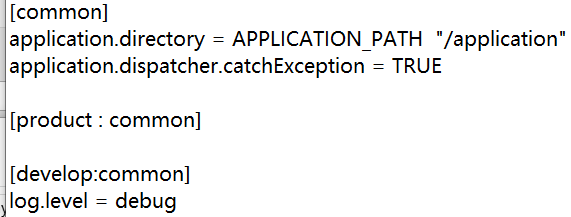
报错2：

Uncaught Yaf\_Exception: There is no section 'develop'

conf/application.ini里面没有develop的配置，加上就可以了

报错3：

Uncaught Yaf\_Exception\_TypeError: Expected an array of application configure



看配置的意思就很容易明白，develop后面要用单词common，而不是别的单词

1. 自动加载的类库路径

全局共享的类库在php.ini中可以配置路径

yaf.library=’/home/yii6/public\_html/lib’

项目自身的类库在application.ini配置文件中可以设置

application.library.namespace=“Tools,Db”

表示加载yaf\application\library下面的Tools和Db文件夹

手动导入类

//绝对路径

Yaf\Loader::import("/home/yii6/public\_html/library/Tools/Http.php");

//相对路径，会在APPLICATION\_PATH.”/library”下加载

Yaf\Loader::import(“plugins/foo.php”);

1. 1，

第4.5章 Laravel

1. 遍历集合,不修改集合内容,返回false中断循环

$collection = collect(['a' => 'a', 'b']);

$collection->each(function ($item, $key) {

print\_r($item . $key);

if (/\* some condition \*/) {

return false;

}

});

//aab0

eachSpread 方法用于循环集合项，将每个嵌套集合项的值传递给回调函数

$collection = collect([['John Doe', 35], ['Jane Doe', 33]]);

$collection->eachSpread(function ($name, $age) {

print\_r($name . $age);

});

//John Doe35Jane Doe33

every用于验证集合中的每一个元素是否通过指定的条件测试,如果有一个不通过返回false

$collection = collect([]);

$collection->every(function($value, $key) {

return $value > 2;

});

// true

1. Laravel 中的模型提供了 fillable 和 guarded专门用来解决批量注入的问题。两者是互斥关系，如果同时存在，fillable 优先级较高。

fillable 为字段修改白名单，比如：

protected $fillable = ['title', 'body', 'category\_id'];

而 guarded 变量存储不允许自动填充的模型字段，是修改字段的黑名单，如：

protected $guarded=['id'];

1. 本地作用域允许定义通用的约束集合以便在程序中重复使用。要定义这样一个范围，只需要在对应的 Eloquent 模型方法前添加 scope 前缀。

/\*\*

\* @param \Illuminate\Database\Eloquent\Builder $query

\* @return \Illuminate\Database\Eloquent\Builder

\*/

public function scopeDefaultSelect($query)

{

return $query->select([

'id', 'name', 'user\_group\_id', 'channel\_template', 'start\_date', 'is\_push\_all',

'end\_date', 'execution\_time', 'status', 'remark', 'is\_cycle', 'cycle',

'update\_at', 'is\_ab\_test', 'priority', 'create\_by'

]);

}

1. append 访问器和修改器

<https://learnku.com/docs/laravel/6.x/eloquent-mutators/5179>

$tpl->append('channel\_desc');

这句代码会调用tpl model里下面的方法：

public function getChannelDescAttribute()

{

return MarketingConf::$channel[$this->attributes['channel\_id']] ?? '';

}

1. 1
2. 1
3. 1

第5章 Redis

1. 访问官网redis.io/download或中国官网<http://www.redis.cn/download.html>

cd /usr/local/src

wget download.redis.io/releases/redis-5.0.4.tar.gz

tar xzf redis-5.0.4.tar.gz

mkdir -p /usr/local/redis

mv redis-5.0.4/\* /usr/local/redis/

cd /usr/local/redis

make

报错1：gcc command not fund

yum  install  gcc -y

报错2：zmalloc.h:50:31: error: jemalloc/jemalloc.h: No such file or directory

make命令加参数：

make MALLOC=libc

启动：

src/redis-server

src/redis-cli

设置开机启动：

vim /etc/rc.d/rc.local

添加

/usr/local/redis/src/redis-server /usr/local/redis/redis.conf

设置后台启动

vim /usr/local/redis/redis.conf

修改

daemonize yes

查看启动

**ps -ef | grep redis**

使用客户端

**>setex name 10 Alice** 设置缓存时间为10s，同时可以修改值

**>del name**

关闭客户端

**redis-cli shutdown**

yii2 安装redis

**composer require --prefer-dist yiisoft/yii2-redis**

(error) MISCONF Redis is configured to save RDB snapshots, but is currently not able to persist on disk.

解决方案：

[~/redis-4.0.2]$ ./src/redis-cli

127.0.0.1:6379> config set stop-writes-on-bgsave-error no

1. Hyperf里面使用redis

有序集合添加键值

public function zAdd(string $key, $score, $value){}

$this->redis->zAdd($key\_name, time(), $user\_group['id']);

如果某个成员已经是有序集的成员，那么更新这个成员的分数值，并通过重新插入这个成员元素，来保证该成员在正确的位置上。

删除集合里面的单个元素

$this->redis->zDelete($key\_name, $user\_group\_id);

返回整个集合,第一个元素是0,第二个元素是1,最后一个元素是-1

$this->redis->zRange($key\_name, 0, -1);

获取元素的值

$score = $this->redis->zScore($set\_name, $uid);

无序集合添加键值,返回成功添加的数量

$redis = Redis::connection();

$redis->sAdd(self::IS\_COUNTING\_KEY, $user\_group->id);

如果某个成员已经是无序集的成员，那么sAdd插入返回0。

删除集合里面的单个元素

$redis->sRem(self::IS\_COUNTING\_KEY, '85');

返回整个集合的数组

$this->redis->sMembers($key\_name);

删除集合所有元素(删除键名):

$this->redis->del($key\_name);

如果不清空集合可以设置过期时间,然后到时间自动清空

$this->redis->expire($key\_name,3600\*24);

1. 求集合的交集

# 有序集 mid\_test

redis 127.0.0.1:6379> ZADD mid\_test 70 "Li Lei"

(integer) 1

redis 127.0.0.1:6379> ZADD mid\_test 70 "Han Meimei"

(integer) 1

redis 127.0.0.1:6379> ZADD mid\_test 99.5 "Tom"

(integer) 1

# 另一个有序集 fin\_test

redis 127.0.0.1:6379> ZADD fin\_test 88 "Li Lei"

(integer) 1

redis 127.0.0.1:6379> ZADD fin\_test 75 "Han Meimei"

(integer) 1

redis 127.0.0.1:6379> ZADD fin\_test 99.5 "Tom"

(integer) 1

# 交集

redis 127.0.0.1:6379> ZINTERSTORE sum\_point 2 mid\_test fin\_test

(integer) 3

# 显示有序集内所有成员及其分数值

redis 127.0.0.1:6379> ZRANGE sum\_point 0 -1 WITHSCORES

1) "Han Meimei"

2) "145"

3) "Li Lei"

4) "158"

5) "Tom"

6) "199"

1. 删除单个元素

redis 127.0.0.1:6379> ZREM page\_rank google.com

1. Mac:

brew install redis

brew services start redis

redis-cli

切换数据库

127.0.0.1:6379> select 2

OK

127.0.0.1:6379[2]> keys \*

1) "mc:default:m:wechat\_auth:id:27"

1. 1

第6章 JavaScript

1. 类型判断

typeof 适合基本类型和函数的判断,返回字符串

typeof 100 “number” typeof NaN “number”

typeof true “boolean”

typeof function “function”

typeof (undefined) “undefined”

typeof [1,2] “object” typeof null===“object”

instanceof 基于原型链判断,不同window或iframe间的对象不能使用

[1,2] instanceof Array===true

1. 对象原型方法

function testType(element){

return element===null?

'[object Null]':Object.prototype.toString.apply(element);

//ie 6、7、8会将null返回[object Object]

}

function Person(name,age){

this.name=name;

this.age=age;

}

Person.prototype.hi=function(){

console.log(“Hi,my name is “+this.name);

}

Person.prototype.LEGS\_NUM=2;

function Student(name,age,className){

Person.call(this,name,age);

this.className=className;

}

Student.prototype=Object.create(Person.prototype);

Student.prototype.constructor=Student;

Student.prototype.x=100;

var bob=new Student('Bob',3,'Class 2');

bob.x;//100

Student.prototype={y:2};

bob.y;//undefined

bob.x;//100,修改对象原型不会影响已经实例化的对象

var alice=new Student('Alice',3,'Class 1');

alice.x;//undefined

alice.y;//2

1. 对象属性的设置，默认都为false

var person={};

Object.defineProperty(person, 'name',{ //person为对象名，name为属性名

configurable:false,//能否删除

writable:false, //能否修改

enumerable:true, //有的属性不能被枚举出来

value:”yuwuy”

});

Object.getOwnPropertyDescriptor(person,'name');

//Object{value:”yuwuy”,writable:false,enumerable:false,configurable:false}

var Dave=new person();

'name' in Dave;//true

Dave.hasOwnProperty('name');//false

Object.defineProperty(person, {

salary:{value:50000,enumerable:true,writable:true},

luck:{

get:function(){

return Math.random()>0.5? 'good': 'bad';

}

},

promote:{

set:function(level){

this.salary\*=1+level\*0.1;

}

}

});

person.salary; //50000

person.promote=2;

person.salary; //60000

属性get/set方法

var man={name: 'yuwuy',

get age(){

return new Date().getFullYear()-1992;

},

set age(val){

console.log('Age can\'t be set to '+val);

}

}

console.log(man.age); //25

man.age=100; // Age can't be set to 100

1. 浏览器兼容

ReferenceError:ActiveXobject is not defined

只有IE支持ActiveXObject，其余的浏览器并不支持

1. iife与闭包

闭包视为对变量域的控制，可以设置私有变量，即在函数内可以访问外部变量，但是外部不能访问函数内的变量。闭包提供了外部访问、修改内部变量的桥梁。

iife的常用写法：

(function(){ /\* code \*/ }());

(function(){ /\* code \*/ })();

函数后面加上括号表示立即调用，当解析器对代码进行解释的时候，先碰到了()，然后碰到function关键字就会自动将()里面的代码识别为函数表达式而不是函数声明。

function f1(){

　　　　var n=999;

　　　　nAdd=function(){n+=1}

　　　　function f2(){

　　　　　　alert(n);

　　　　}

　　　　return f2;

}

var result=f1();

result(); // 999

nAdd();

result(); // 1000

//f1是f2的父函数，f1返回f2,f2被赋给了一个全局变量result，这导致f2始终在内存中，而f2的存在依赖于f1，因此f1也始终在内存中

< script type = “text/javascript” >

for (var i = 0; i < 3; i++) {

setTimeout(function () {

console.log(i); //普通函数，结果为3 3 3

}, 0);

};

for (var i = 0; i < 3; i++) {

setTimeout((function (x) {

console.log(x); //iife，结果为0 1 2

})(i), 0);

};

< / script >

//setTimeout后面的值可以不为0更好理解一点，但是视为先执行for循环，再执行内部function

var elems = document.getElementsByTagName('a');

for (var i = 0; i < elems.length; i++) {

(function (lockedInIndex) {

elems[i].addEventListener('click', function (e) {

e.preventDefault();

alert('I am link #' + lockedInIndex);

}, 'false'); //false表示从最内层开始向外冒泡执行事件,true表示从最外层开始向内捕获事件

})(i);

}

//在iife内部，i的值传给了lockedIndex，并且被锁在内存中，尽管for循环结束后i的值已经改变，但是立即执行函数内部lockedIndex的值并不会改变

1. navigator.userAgent

可以判断浏览器信息

1. 对象和字符串转换

var obj={“name”:null,”u”:undefined,a:NaN,b:Infinity,

d:new Date(),

//自定义toString()方法

o:{x:1,y:2,toJSON:function(){

return this.x+this.y;

}

}

};

var objToStr=JSON.stringify(obj);

//' {“name”:null,”a”:null,”b”:null,”d”:”2017-07-03T08:07:40.610Z”,”o”:3}'

var str='{“name”:”Mike”,”sex”:”女”,”age”:29}';

var strToObj=JSON.parse(str);

var o={x:1,y:2};

o.toString();//[object Object]

//自定义对象的toString方法

o.toString=function(){return this.x+this.y};

“Result “+o; //”Result 3”,字符串拼接

+o;//3, +运算符尝试把对象转化为数字,调用toString()

//valueOf()是尝试把对象转为基本类型的时候调用的方法

o.valueOf=function(){return this.x+this.y+100};

+o; //103,尝试把对象转化为数字,定义了valueOf就调用valueOf方法

“Result “+o; //”Result 103”,valueOf和toString同时存在,尝试拼接时调用valueOf方法

1. JavaScript数组方法

|  |  |
| --- | --- |
| 数组被修改 | 原数组未被修改 |
| sort  reverse  splice 数组元素替换 | join 数组元素拼接成字符串  concat 数组元素拼接为新数组  slice 数组元素切割  filter |

数组创建

var arr=new Array(100);//undefined\*100,稀疏数组

var arr=new Array(true,null,1,2,”hi”);

数组元素读写

delete arr[0];

arr[0];//undefined

数组元素增删

var arr=[1,2];

arr.push(3); //尾部添加元素,返回新数组的长度3

arr;//[1,2,3]

arr.unshift(0); //头部添加元素,返回新数组的长度4

arr;//[0,1,2,3]

arr.length-=1;

arr;//[0,1,2]

arr.pop();//返回删除的尾部元素2

arr;//[0,1]

arr.shift();//返回删除的头部元素0

arr;//[1]

数组遍历

var arr=[3,2];

for(i in arr){

console.log(i);//0 1

console.log(arr[i]);//0 3 1 2

}

数组和字符串转换

var arr=[1,2,3];

arr.join();//”1,2,3”,原数组没有被修改

arr.join(“\_”);//”1\_2\_3”

function repeatString(str,n){

return new Array(n+1).join(str); //str是分隔符,undefined会被忽略掉,即n+1个空字符之间的有n个str

}

repeatString(“a”,3);//”aaa”

reverse数组

var arr=[1,2,3];

arr.reverse();//[3,2,1]

arr;//[3,2,1],原数组被修改

数组排序

var arr=[“a”,”d”,”c”,”b”];

arr.sort();//[“a”,”b”,”c”,”d”]

arr;// [“a”,”b”,”c”,”d”],原数组被修改

var arr=[13,23,51,3];

arr.sort();//[13,23,3,51],默认安装字典顺序排序

arr.sort(function(a,b){return a-b;}); //[3,13,23,51]

arr=[{age:25},{age:23},{age:42}];

arr.sort(function(a,b){return a.age-b.age;});

arr.forEach(function(i){console.log('age:',i.age);});

/\*

age:23

age:25

age:42 \*/

var arr=[1,2,3,4,5];

arr.forEach(function(x,index,a){

console.log(x+'|'+index+'|'+(a===arr));});

/\* 第一个为元素值,第二个为索引,第三个为数组本身

1|0|true

2|1|true

3|2|true

4|3|true

5|4|true\*/

数组拼接

var arr=[1,2,3];

arr.concat(4,5);//[1,2,3,4,5]

arr;//[1,2,3],原数组没有被修改

arr.concat([10,11],13);//[1,2,3,10,11,13],和数组合并会和数组的元素进行合并

arr.concat([1,[2,3]]);//[1,2,3,1,[2,3]]

数组切割

var arr=[1,2,3,4,5];

arr.slice(1,3);//[2,3],左闭右开区间,不会修改原数组

arr.slice(1);//[2,3,4,5]

arr.slice(1,-1);//[2,3,4],负数表示从后往前数,最后一个元素下标为-1

var copy=arr.slice(-4,-3);//[2]

console.log(copy);//[2]

var domNodes=

Array.protype.slice.call(document.getElementsByTagName(“div”));

//getElementsByTagName,document.childNodes返回的nodeList对象不是数组,用slice方法复制后可以转换为数组,使用Array的所有方法

数组替换

var arr=[1,2,3,4,5];

arr.splice(2);//[3,4,5]

arr;//[1,2]

var arr=[1,2,3,4,5];

arr.splice(2,2);//[3,4]

arr;//[1,2,5]

var arr=[1,2,3,4,5];

arr.splice(1,1,'a','b','c');//[2]

arr;//[1,'a','b','c',3,4,5]

数组映射

var arr=[1,2,3];

arr.map(function(x){return x+10;});//[11,12,13]

arr;//[1,2,3],不会修改原数组

数组元素过滤

var arr=[1,2,3,4,5,6,7,8,9];

arr.filter(function(x,index){return x>8||index%3===0;});//[1,4,7,9]

arr;//[1,2,3,4,5,6,7,8,9],不会修改原数组

数组元素判断

var arr=[1,2,3,4,5];

arr.every(function(x){return x>8;});//flase所有元素

arr.some(function(x){return x===3;});//true某个元素

数组元素处理

var arr=[1,2,3];

var sum=arr.reduce(function(x,y){ return x+y;},0);//6,0带入x,数组第一个元素带入y,求和的结果带入x,数组第二个元素带入y,两两进行操作

arr;//[1,2,3];

var max=arr.reduce(function(x,y){

console.log(x+”|”+y);

return x>y?x:y;}

);

//1|2

//2|3

max;//3

//reduceRight从右向左进行操作

max=arr.reduceRight(function(x,y){

console.log(x+”|”+y);

return x>y?x:y;}

);

//3|2

//3|1

max;//3

数组检索

var arr=[1,2,3,2,1];

arr.indexOf(2);//1

arr.indexOf(4);//-1

arr.indexOf(2,3);//3,查找2,从第4个元素开始查找

arr.indexOf(1,-3);//4

arr.lastIndexOf(2);//3,从后向前查找2

arr.lastIndexOf(2,-2);//3

1. 函数

function构造器(没有函数名,方法名大写,没有空对象的创建和返回)

var func =new Function('a','b','console.log(a+b);'); //new可以省略

func(1,2);//3

var globalVal='a';

(function(){

var localVal='b';

Function('console.log(typeof localVal,typeof globalVal);')();

})();//undefined string,function构造器能访问到全局变量,不能访问到localVal

Function('var localVal=“a”;console.log(localVal);')();

console.log(typeof localVal);//a undefined

function foo(x,y,z){

arguments.length;//2,实际传入参数的个数

arguments[0];//1

arguments[0]=10;

x;//10

arguments[2]=100;

z;//undefined

arguments.callee===foo;//true

}

function foo(x,y,z){

'use strict';

arguments[0]=10;

x;//1

//arguments.callee===foo;严格模式下不能使用

}

foo(1,2);

foo.length;//3,形参个数

foo.name;//”foo”

1. this

console.log(this.document===document);//true

console.log(this===window);//true

function f(){

“use strict”;//使用严格模式

return this;

}

f()===undefined; //true

var o={

prop:37,

f:function(){

return this.prop;

}

};

console.log(o.f());//37

var o={prop:37};

function independent(){

return this.prop;

}

o.f=independent;

console.log(o.f());//37

function modulus(){

return Math.sqrt(this.r\*this.r+this.i\*this.i);

}

var o={

r:1,

i:1,

get phase(){

return Math.atan2(this.i,this.r);

}

};

Object.defineProperty(o,'modul',{get:modulus,enumerable:true,configurable:true});//o是对象名,modul是属性名,属性modul的get方法使用的函数是上面定义的modulus()

console.log(o.phase,o.modul);

call/apply方法与this

function add(c,d){

return this.a+this.b+c+d;

}

var o={a:1,b:3};

add.call(o,5,7); //1+3+5+7=16

add.apply(o,[10,20]); //1+3+10+20=34

bind方法与this

function f(){

return this.a;

}

var g=f.bind({a:”test”});

console.log({g()});//test

var o={a:37,f:f,g:g};

console.log(o.f(),o.g());//37,test

The simplest use of bind() is to make a function that, no matter how it is called, is called with a particular this value.

this.x = 9; // this refers to global “window” object here in the browser

var module = {

x: 81,

getX: function() { return this.x; }

};

module.getX(); // 81

var retrieveX = module.getX;

retrieveX(); // 9 - The function gets invoked at the global scope

var boundGetX = retrieveX.bind(module);

boundGetX(); // 81

The next simplest use of bind() is to make a function with pre-specified initial arguments. These arguments (if any) follow the provided this value and are then inserted at the start of the arguments passed to the target function, followed by the arguments passed to the bound function.

function list() {

return Array.prototype.slice.call(arguments);

}

var list1 = list(1, 2, 3); // [1, 2, 3]

// Create a function with a preset leading argument

var leading = list.bind(null, 37);

var list2 = leading (); // [37]

var list3 = leading(1, 2, 3);// [37, 1, 2, 3]

var leading2 = leading.bind(null, 21);

var list4 = leading2(1,2,3); // [37,21, 1, 2, 3]

function foo(){

this.b=100;

return this.a; //返回的不是一个对象

}

var func =foo.bind({a:1});

func();//1

new func();//{b:100},除非返回的是一个对象否则返回空对象,且对象的值被设置为100

1. 正则表达式

分组

(x) /(abc)/.test('abc');

\n 使用前面的第n个分组 /(abc)\1/.test('abcabc');

(?:x) 该分组不能被\n使用 /(?:abc)(def)\1/.test('abcdefdef');

重复

|  |  |  |
| --- | --- | --- |
| 贪婪算法 | 非贪婪算法 |  |
| x\* | x\*? | 重复>=0 |
| x+ | x+? | 重复>=1 |

x? 出现0或1次

三个flag

|  |  |
| --- | --- |
| global,匹配所有 | g |
| ignoreCase | i |
| multiline,换行匹配 | m |

/abc/gim.test(“ABC”);//true

RegExp(“abc”,”mgi”);

RegExp对象方法

/abc/.exec(“abcdd”);//”abc”

/abc/.toString();//”/abc/”

var reg=/abc/;

reg.compile(“def”);//改变正则表达式的内容

reg.test(“def”);//true

String类型与正则表达式相关的方法

“abcabcdef”.search(/(abc)\1/);//0

“aabbcc”.replace(/b+?/,”1”);//aa1bcc

“aabbcc”.match(/b+/);//[“bb”]

“aabbbbccbbaa”.match(/b+/g);//[“bbbb”,”bb”]

“aabbbbccbbaa”.split(/b+/);//[“aa”,”cc”,”aa”],split用来分割字符串,参数是分隔符,可选参数是需要的个数,如果空字符串作为分隔符,则分割每个字符

1. 节点

var x=document.getElementById(“con”);

// document.write(x.childNodes[0]);

// document.write(x.childNodes[x.childNodes.length-1]);

document.write(x.firstChild);

document.write(x.lastChild);

document.write(x.parentNode.nodeName);

document.write(x. nextSibling.nodeName);

document.write(x. previousSibling.nodeName);

var new1=document.createElement('li');

//var txt=document.createTextNode('PHP');//创建文本节点

//new1.appendChild(txt);

new1.innerHTML='PHP';

x.appendChild(new1); //插入子节点

var new2=document.createElement(“li”);

new2.innerHTML='php';

x.insertBefore(new2,new1); //子节点之前插入子节点

x.replaceChild(new2,new1); //用new2替换new1

var tmp=x.removeChild(x.childNodes[0]);

tmp=null; //子节点不在DOM树中，但还在内存中，若要完全删除，使x=null

1. 网页尺寸

scrollHeight和scrollWidth，网页内容高度和宽度,大致等于clientHeight,与浏览器相关

offsetHeight = clientHeight + 滚动条 + 边框

1. 事件

|  |  |
| --- | --- |
| onfocus | 光标聚焦 |
| onblur | 失焦 |
| onselect | 内容选中 |
| onchange | 文本框内容改变 |
| onload | 页面加载完成后 |
| onunload | 页面关闭或刷新 |

1. 禁止输入数字和小数点以外的字符

onkeyup=“this.value=this.value.replace(/[^0-9\.]/g,'');”

第7章 jQuery

1. 环境搭建

官方网站获取最新的版本 [http://jquery.com/download/](http://jquery.com/download/" \t "_blank)

<script type=“text/javascript”

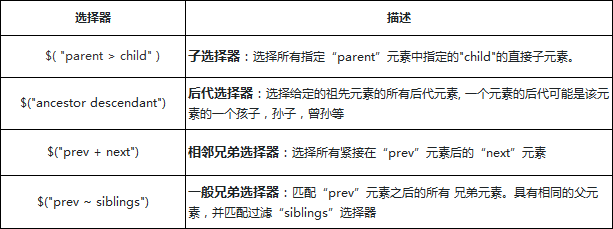
src=“ http://libs.baidu.com/jquery/1.9.1/jquery.js “>

</script>

1. 选择器

|  |  |
| --- | --- |
| id选择器 | $(“#id”) |
| class选择器 | $(“.class”) |
| tag选择器 | $(“tagName”) |

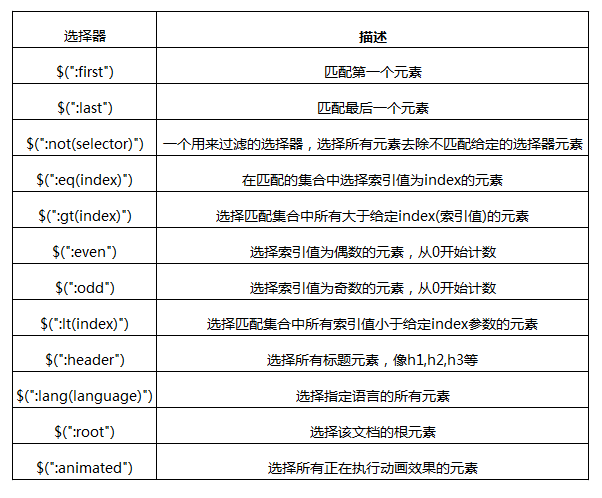
**层选择器**



1. 层级选择器都有一个参考节点
2. 后代选择器包含子选择器的选择的内容
3. 一般兄弟选择器包含相邻兄弟选择的内容
4. 相邻兄弟选择器和一般兄弟选择器所选择到的元素，必须在同一个父元素下

$('div>p').css(“border”, “1px groove red”);//选择所有div里面的子元素P

**基本筛选选择器**

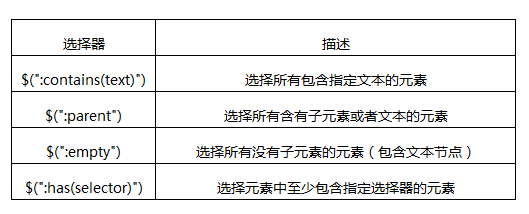


//选中所有紧接在(没有checked属性的)input元素后的第一个p元素,相邻兄弟选择器

$(“input:not(:checked)+p”)

$(“.div:last”).css(“color”, “#CD00CD”);//找到最后一个class为div的元素

**内容筛选选择器**

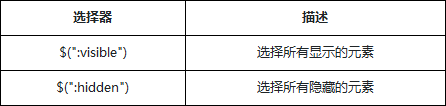


1. :contains与:has都有查找的意思，但是contains查找包含”指定文本”的元素，has查找包含”指定元素”的元素
2. 如果:contains匹配的文本包含在元素的子元素中，同样认为是符合条件的。
3. :parent与:empty是相反的，两者所涉及的子元素，包括文本节点

//查找所有class='div'中DOM元素中包含”contains”的元素节点并且设置颜色

$(“.div:contains('contains')”).css(“color”, “#CD00CD”);

**可见性筛选选择器**



除了3以外让元素在页面不显示的方法:

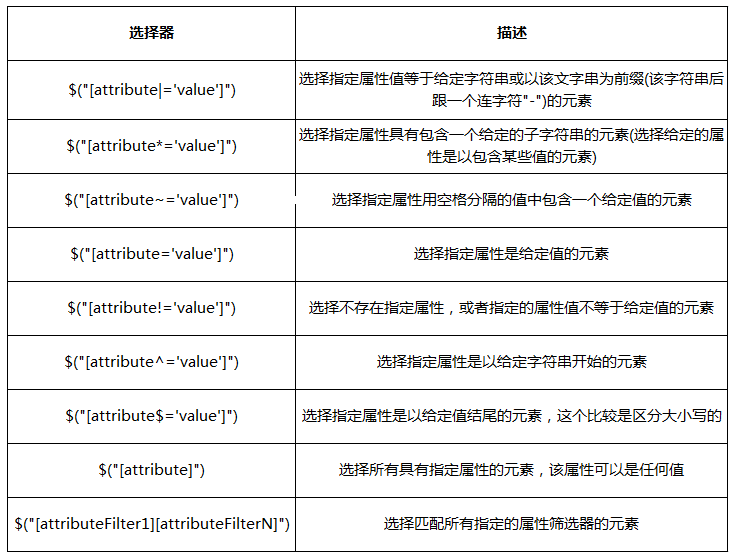
1. CSS display的值是none
2. type=“hidden”的表单元素。
3. 宽度和高度都显式设置为0。
4. 元素的祖先元素是隐藏的
5. CSS visibility的值是hidden
6. CSS opacity的值是0

如果元素占据一定的空间(宽度或高度大于零),元素被认为是可见的。visibility: hidden 或 opacity: 0被选择器认为是可见的，因为仍然占用布局空间。

//检测id = div1的DOM元素是否可见

$('#div1:visible').length==1;

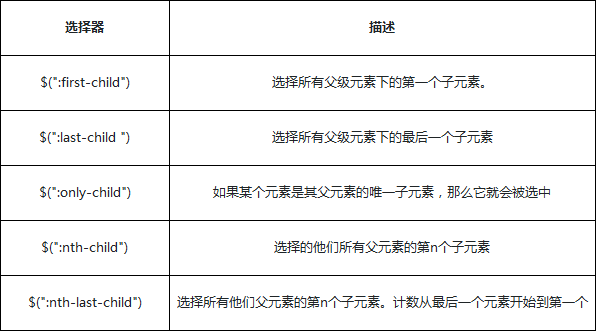
**属性筛选选择器**



//查找所有div中，有属性name的值包含一个空格和”a”的元素

$('div[name~=“a”]').css(“border”, “3px groove #668B8B”);

**子元素筛选选择器**

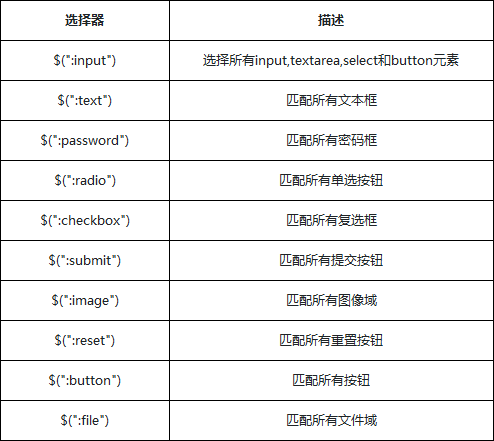


1. nth-child(index)从1开始计数，而eq(index)从0开始(是一个数组)
2. :first只匹配一个元素，但是:first-child可以匹配多个：即匹配每个父级元素的第一个子元素

//查找所有class=“last-div”下的倒数第二个a元素

$('.last-div a:nth-last-child(2)').css(“color”, “red”);

**表单元素选择器**



//查找所有 input, textarea, select 和 button 元素

$(':input').css(“border”, “1px groove red”);

//匹配所有input元素中类型为text的input元素,层选择器加表单元素选择器

$('input:text').css(“background”, “#A2CD5A”);



1. 选择器适用于复选框和单选框，对于下拉框元素, 使用 :selected 选择器
2. 在某些浏览器中，选择器:checked可能会错误选取到<option>元素，所以保险起见换用选择器input:checked，确保只会选取<input>元素
3. Attribute就是dom节点自带的属性,如id、class、title、align等：

<div id=“vider” title=“good”></div>

而Property是这个DOM元素作为对象，其附加的内容，如tagName, nodeName, nodeType,defaultChecked, 和 defaultSelected

获取Attribute用attr，获取Property用prop

1. attr(传入属性名)：获取属性值
2. attr(属性名, 属性值)：设置属性值
3. attr(属性名,函数值)：设置属性的函数值
4. attr(attributes)：给指定元素设置多个值

//$(selector).attr(attribute,function(index,oldvalue)), index是选择器的

<img src=“/i/eg\_smile.gif” width=“128” height=“128” />

$(“img”).attr(“width”,function(n,v){

return v-50;

});//减少图像的宽度50

$(“img”).attr({width:”50”,height:”80”});

1. 属性与样式

|  |  |
| --- | --- |
| .html() | 获取集合中第一个匹配元素的HTML内容 |
| .html( htmlString ) | 设置匹配元素的html内容 |
| .html( function(index, oldhtml) ) | 返回设置html内容的一个函数 |

.text() .val()与.html()的方法是一样的，只是.html处理的对象是html元素(包括标签)，.text处理的是文本, .val()处理表单元素.

.val()和.html()如果其应用在多个元素上，只读取第一个元素，但.text()应用在多个元素上将会读取所有选中元素的文本内容; 3个方法都会替换所有选中元素的内容

用.val()处理select元素，当没有选项被选中，返回null,若select元素有multiple（多选）属性且至少一个选项被选中，返回一个包含每个选中项值的数组.

<select id=“multiple” multiple=“multiple”>

<option selected=“selected”>imocc</option>

<option>慕课网</option>

<option selected=“selected”>博客园</option>

</select>

//返回[“imocc”, “博客园”],打印imocc,博客园

$(“p:last”).text( $(“#multiple”).val() );

1. each()根据每次调用callback的返回值来决定后续动作,若返回false，则停止循环

遍历数组

$.each(['Aa', 'Bb'], function(i, item) {

$aaron.append(“索引=“ + i + “; 元素=“ + item);

});

遍历对象

$.each({name: “张三”,age: 18},

function(property, value) {

$aaron.append(“属性名=“ + property + “; 值=“ + value);

}

);

1. 增加样式.addClass()

$(“div”).addClass(function(index,className) {

if(-1 !== className.indexOf('imooc')){

$(this).addClass('imoocClass') //this指向匹配元素集合中的当前元素

}

});

.removeClass( [className ] )：每个匹配元素移除一个或多个用空格隔开的class, 如果没有参数，那么所有的class将被移除

<div class=“right”>

<div class=“aa bb imoocClass”>

<article>

<p>imoocClass</p>

</article>

</div>

<div>

</div>

</div>

$('.right > div:first').removeClass(function(index,className){

//把className = aa bb imoocClass赋给下一个兄弟div作为它的class

$(this).next().addClass(className);

return 'imoocClass'; //删除自己本身的imoocClass

})

.toggleClass()判断对应的元素上是否存在指定的Class名，有就删除，没有就增加

//true，那么样式类将被添加; false，样式类将被移除

$(“#table tr:even”).toggleClass(“c”, true);

$(“p”).each(function() {

var $thisParagraph = $(this);

var count = 0;

$thisParagraph.click(function() {

++count;

$thisParagraph.find(“span”).text('clicks: ' + count);

//第三次点击时添加 “highlight” 类, 第一 二次移除

$thisParagraph.toggleClass(“highlight”, count % 3 == 0);

});

});

1. 样式操作

获取

|  |  |
| --- | --- |
| .css(propertyName) | 获取匹配元素中的第一个的属性 |
| .css(propertyNames) | 传入一个数组,返回一个对象结果 |

设置

|  |  |
| --- | --- |
| .css(propertyName,value) |  |
| .css(propertyName,function) |  |
| .css(properties) | 传入一个对象,同时设置多个样式 |

当一个数只被作为value的时候， jQuery会将其转换为一个字符串，并添在字符串的结尾添加px，如 .css(“width”,50}) 与 .css(“width”,”50px”})一样.

$('.sixth').css(“width”,function(index,value){

value=value.split('px');

return (Number(value[0])+50)+”px”;

});

1. 数据存储

$('.left').click(function() {

var ele = $(this);

//通过$.data方式存储数据

$.data(ele, “a”, “data test”);

$.data(ele, “b”, {

name : “Bob”

});

//读取数据

var reset = $.data(ele, “a”) + “</br>“ + $.data(ele, “b”).name;

//find()获得当前元素集合中每个元素的后代, append()在被选元素的结尾插入指定内容

ele.find('span').append(reset);

})

$('.right').click(function() {

var ele = $(this);

ele.data(“a”, “data test”);

ele.data(“b”, {

name: “Alice”

}) ;

var reset = ele.data(“a”) + “</br>“ + ele.data(“b”).name;

ele.find('span').append(reset);

})

1. 节点插入

$(“#bt1”).on('click', function() {

//添加子元素

//在父元素的内容尾部

$(“.content”).append('<div class=“append”>添加的元素</div>');

})

$(“#bt2”).on('click', function() {

$('<div class=“ap”>txt</div>').appendTo($(“.content”));

})

//在父元素的内容之前

$(“#bt1”).on('click', function() {

$('.aaron1').prepend('<p>元素</p>');

$('<p>元素</p>').prependTo($('.aaron2'));

})

//添加兄弟元素before,after,insertBefore,insertAfter

$(“#bt2”).on('click', function() {

$(“.test2”).after('<p style=“color:blue”>after,在匹配元素之后增加</p>', '<p style=“color:blue”>多参数</p>');

$('<p style=“color:red”></p>','<p>多参数</p>').insertBefore($(“.test1”));

});

**节点删除**

删除所有子节点

<div class=“hello”><p>网</p></div>

$('.hello').empty();// <div class=“hello”></div>

删除元素及子节点

$(“p”).remove(“:contains('3')”);

删除节点但保留数据和事件

$(“#bt1”).click(function() {

if (!$(“p”).length) return;

//节点被删除了,页面上暂时不可见

p = $(“p”).detach();

});

$(“#bt2”).click(function() {

//节点重新加回到页面,原来节点上的事件还在

$(“body”).append(p);

});

**节点复制**

$(“.left”).append( $(this).clone().css('color','red')); //只复制结构,事件丢失

$(“.left”).append( $(this).clone(true).css('color','blue')); //复制结构和事件

clone()只能处理通过jQuery绑定的事件与数据;元素数据内对象和数组不会被复制

**节点替换**

$(“.right > div:first p:eq(1)”).replaceWith('<a style=“color:red”>替换第二段的内容</a>');

$('<a style=“color:red”>内容</a>').replaceAll('.right > div:last p:last');

替换会删除与节点相关联的所有数据和事件

**节点wrap和unwrap**

$('a').wrap(function() {

//返回的是一个字符串,包含+连接的3部分

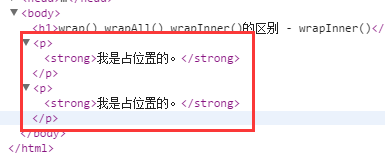
return '<div class=“'+$(this).text()+'“ />'; });

$('p').wrap('<div></div>');

$('p').unwrap();

|  |  |
| --- | --- |
| wrap | 将匹配的每个元素分别打包 |
| wrapAll | 将匹配的所有元素整体打包到一个父元素中 |
| wrapInner | 将匹配的每个元素分别打包出一个子元素 |

$(“p”).wrapInner(“<strong></strong>“);



1. 节点遍历

|  |  |
| --- | --- |
| children() | 只查找子节点 |
| parent() | 父元素 |
| next() | 下一个兄弟元素 |
| prev() | 前一个兄弟元素 |

//可以加入选择器

$('.div').children(':last').css('border', '3px solid blue');

// parents()查找每个元素的所有祖先元素,sibilings()匹配每个元素的所有兄弟元素

$('.item-b').parents('.first-div').css('border', '2px solid blue');

// .closest()开始于当前元素,向上查找直到找到一个匹配的就停止

$('li.item-1').closest('ul.level-2').css('border','3px solid red');

//find()方法必须带选择器参数,查找所有后代节点

$('.left').find(“li:last”).css('border',”3px solid red”);

//this指向当前迭代的dom元素,index是li的索引,element是对应的li节点

$(“li”).each(function(index, element) {

if (index % 2) {

$(this).css('color','blue')

}

})

1. 将元素加入集合中

//将文档中p标签加入li标签形成集合,然后修改新集合的颜色;li并不在p的集合中

$('li').add('p').css('background', 'red');

$('li').add(document.getElementsByTagName('p')[0]);

//动态创建p标签加入到集合,但是并不会显示在页面里;然后作为一个整体重新加入到页面中

$('li').add('<p>新的p元素2</p>').appendTo($('.right'));

1. 事件

|  |  |
| --- | --- |
| focusin | 鼠标聚焦 |
| focusout | 失焦 |

当元素的子元素触发了focus事件时,元素产生focusin事件,即focus()事件在元素本身产生，focusin()在元素包含的元素中产生

click()监听单击,dbclick()监听双击,包括mousedown和mouseup

//不带参数

$(“button:eq(0)”).click(function() {

alert(this);

})

//绑定指定的元素

$('p').click(function(e) {

alert(e.target.textContent);

})

$(“button:eq(1)”).click(function() {

$('p').click();

})

//绑定指定的元素并可以接受一个数据参数

$(“button:eq(2)”).click(1111, da); //da()是一个函数

function da(e) {

alert(e.data); //1111,e.data是元素属性

}

//mouseup,mousedown,mousemove,mouseover,mouseout,focusin与click类似的3种方式

$(“button:eq(0)”).mousedown(function(e) {

alert('e.which: ' + e.which); //敲左键时which=1，中键2，右键3

})

mousemove事件是当鼠标指针在元素范围内移动时触发;如果处理器做任何重大的处理，或者如果该事件存在多个处理函数，可能造成浏览器严重的性能问题;

mouseover,mouseout触发后往上找祖先元素，如果有事件,就会触发;

mouseenter,mouseleave只会在绑定它的元素上被调用

$(selector).hover(handlerIn, handlerOut)可以简便的同时处理鼠标进入和离开的事件

$(“p”).hover(

function() {

$(this).css(“background”, 'red');

},

function() {

$(this).css(“background”, '#bbffaa');

}

);

change事件

|  |  |
| --- | --- |
| input | 当value值改变时,失焦后触发;对于选择框,鼠标选择时立即触发 |
| select | 鼠标选择时立即触发 |
| textarea | 失焦后触发 |

select事件

当textarea或文本类型的input元素被选中时触发,会调用执行绑定的所有函数,包括浏览器的默认行为.传统的方式是调用事件对象e.preventDefault()来处理, jQuery中可以通过在函数中返回false来防止触发浏览器的默认行为.

$(“input”).select(function(e){

alert(e.target.value);

return false;

})

form元素有默认提交表单的行为.

<form id=“target2” action=“destination.html”>

回车键或者点击提交表单,禁止浏览器默认跳转：

<input type=“text” value=“输入新的值” />

<input type=“submit” value=“Go” />

</form>

$('#target2').submit(function() {

alert('捕获提交表达动作,阻止页面跳转');

return false;

});

1. 键盘事件顺序keydown->keypress->keyup

keypress不能捕获组合键,不响应某些功能键(Shift,Ctrl,Alt,F1-F12,arrow keys),不支持中文

1. on的多事件绑定

$(“#test1”).on('click', function(e) {

$(this).text('触发事件：' + e.type);

})

//多个事件绑定同一个函数

$(“#test2”).on('mousedown mouseup', function(e) {

$(this).text('触发事件：' + e.type);

})

//多个事件绑定不同函数

$(“#test3”).on({

mousedown: function(e) {

$(this).text('触发事件：' + e.type);

},

mouseup: function(e) {

$(this).text('触发事件：' + e.type);

}

})

事件委托

<div class=“aaron”>

<a>点击这里</a>

<a>点击这里</a>

</div>

//如果有多个子节点需要绑定同一个函数,可以把事件绑定到某个祖先节点,在子元素触发,冒泡到祖先元素

$('body').on('click', 'a', function(e) {

alert(e.target.textContent);

})

事件删除

$(“button:first”).click(function() {

$(“.aaron:first”).off('mousedown mouseup');

})

$(“button:last”).click(function() {

$(“.aaron:last”).off();

})

1. 事件对象的属性和方法

$(“a”).click(function(event) {

alert(event.type); // “click”事件

});

event.pageX 和 event.pageY：获取鼠标当前相对于页面左上角的坐标

event.preventDefault() ：阻止默认行为

event.stopPropagation() ：阻止事件冒泡

event.which：获取鼠标单击时的键

event.currentTarget : 在事件冒泡过程中的当前DOM元素, 等同于this

1. trigger事件

一般事件需要和用户交互才能触发,trigger()方法不需要和用户交互直接触发浏览器事件.

$( “#foo” ).on( “click”, function() {

alert( $( this ).text() );

});

$( “#foo” ).trigger( “click” );

自定义事件传递参数

$( “#foo” ).on( “custom”, function( event, param1, param2 ) {

alert( param1 + “\n” + param2 );

});

$( “#foo”).trigger( “custom”, [ “Custom”, “Event” ] );

triggerHandler()

* 不会引起事件（如表单提交）的默认行为
* 只影响第一个匹配元素,trigger() 会操作 jQuery 对象匹配的所有元素
* 不会在 DOM 树中冒泡；如果目标元素不直接处理它们，则不发生任何事情
* 返回最后一个函数的返回值;如果没有事件被触发，返回undefined

<div id=“accident”>

<a>triggerHandler事件</a>

<input type=“text”>

</div>

$(“input”).on(“focus”,function(event,title) {

$(this).val(title); //设置input的value

});

//点击div的所有子元素会触发事件

$(“#accident”).on(“click”,function() {

alert(“trigger触发的事件会在 DOM 树中向上冒泡”);

});

$(“button:first”).click(function() {

$(“a”).trigger(“click”);

//focus是原生事件,触发默认聚焦,此时不能更改input的value值

$(“input”).trigger(“focus”);

});

$(“button:last”).click(function() {

$(“a”).triggerHandler(“click”);

//没有触发默认的聚焦事件,可以传入参数修改input的value

$(“input”).triggerHandler(“focus”,'没有触发默认聚焦事件');

});

1. 隐藏,显示元素

$(“button:first”).click(function() {

$(“#a1”).hide()

});

$(“button:last”).click(function() {

$(“#a2”).hide({

duration: 3000,

complete: function() {

alert('执行3000ms动画完毕');

}

})

});

$(“#a1”).hide(3000).show(3000); //3秒动画隐藏元素后显示

$(“.right”).toggle(3000); //在隐藏和显示之间切换

$(“p”).toggle(true); //true表示显示所有元素,false表示隐藏所有元素

//当方法完成时执行函数

$(“button”).click(function(){

$(“p”).toggle(1000,function(){

alert(“The paragraph is now hidden”);

});

});

//上卷(下拉),需要先将元素隐藏

$(“button:last”).click(function() {

$(“#a2”).slideDown(3000,function(){//slideUp()隐藏元素,slideToggle()切换

alert('动画执行结束');

})

});

//淡出(淡入),fadeIn(),fadetoggle()

$(“p”).fadeOut(2000, function() {

alert(“隐藏完毕!”);

});

//某一中间值的淡入淡出效果

$(““).fadeTo(1000, 0.6, function() {

alert('完成');

});

1. animate

//直接设定数值

$aaron.animate({

width :300,

height: “toggle”,

fontSize: “+=1em”

},

2000,

function(){

alert(“动画 fontSize执行完毕!”);

}

);

animate的options参数

* easin : 过渡使用哪种缓动函数
* step：动画的每一步完成之后要执行的函数
* progress：动画调用的时候会执行这个回调，进度的概念
* complete：动画完成回调

$aaron.animate({

height: '50'//动画改变的元素

}, {

duration :2000,

step: function(now, fx) {

$aaron.text('高度现在的值:'+now); //now为元素现在的高度

}

})

$aaron.animate({

height: '50'

}, {

duration :2000,

progress: function(now, fx) {

$aaron.text('进度:'+fx) //arguments[1]和fx等价,为动画的进度,范围0~1

}

})

动画停止

* .stop(); 停止元素当前的动画并开始下一个动画
* .stop(true); 停止元素所有的动画
* .stop(true,true); 停止元素所有的动画，元素的属性立刻修改成当前动画的目标值

$(“#aaron”).animate({

height: 300

},

5000

)

$(“#aaron”).animate({

width: 300

},

5000

)

$aaron.stop(true,true);

1. 提高和取消样式优先级

.right {

display: none !important;

}

$(“.right”).attr('style','display: block !important')

1. 查找值索引

var index = $.inArray('a',['a','b','c','d','a','c'],2); //从索引2开始查找

1. 去除字符串首尾的空格

**var** str = “ lots of spaces before and after “;

$( “#trimmed” ).html( “$.trim()'ed: '“ + $.trim(str) + “'“);

//$.trim()'ed: 'lots of spaces before and after'

1. 获取元素

$(“#aaron a”).get(1).style.color = “blue”;

$(“#aaron a”)get(-1).style.color = “#8A2BE2”;//倒数第一个

$(“#aaron a”)get().style.color = “#8A2BE2”;//没有参数则获取全部

查找元素位置

<ul>

    <a></a>

    <li id=“test1”>1</li>

    <li id=“test2”>2</li>

</ul>

$(“li”).index($(“#test2”)); //1, 返回给定元素在li元素数组里的索引

$(“li”).index();//1,返回第一个li元素在兄弟元素里的索引

1. get():Load data from the server using a HTTP GET request

//Get the test.php page contents, which has been returned in json format (<?php echo json\_encode( array( “name”=>“John”,”time”=>“2pm” ) ); ?>), and add it to the page.

$.get( “test.php”, function( data ) {

$( “body” )

.append( “Name: “ + data.name ) // John

.append( “Time: “ + data.time ); // 2pm

}, “json” );

$.get('<?php echo yii\helpers\Url::to(['order/getExpress']) ?>',

{'expressNo':a.attr('data')},

function(rd){

var str=““;

if(rd.message=='ok'){

for(var i=0;i<res.data.length;++i)

str+=“<p>“+res.data[i].context+” “+res.data[i].time+”<p>“;

}

a.find(“.expressshow”).html(str);

a.attr('data','ok');

},

'json');

1. 图像预览

<input type=“file” id=“xdaTanFileImg” accept=“image/\*” name=“avatar”>

<img id=“xmTanImg” src=“{$user['avatar']}”>

<script type=“text/javascript”>

$('#xdaTanFileImg').change(function(){

var t = this;

xmTanUploadImg(t);

})

//判断浏览器是否支持FileReader接口

if (typeof FileReader == 'undefined') {

document.getElementById(“xmTanDiv”).InnerHTML =

“<h1>当前浏览器不支持FileReader接口</h1>“;

//使选择控件不可操作

document.getElementById(“xdaTanFileImg”).setAttribute(“disabled”, “disabled”);

}

//选择图片，马上预览

function xmTanUploadImg(obj) {

var file = obj.files[0];

// console.log(obj);console.log(file);

// console.log(“file.size = “ + file.size); //file.size 单位为byte

var reader = new FileReader();

//读取文件过程方法

reader.onloadstart = function (e) {

// console.log(“开始读取....”);

}

reader.onprogress = function (e) {

// console.log(“正在读取中....”);

}

reader.onabort = function (e) {

// console.log(“中断读取....”);

}

reader.onerror = function (e) {

// console.log(“读取异常....”);

}

reader.onload = function (e) {

// console.log(“成功读取....”);

var img = document.getElementById(“xmTanImg”);

img.src = e.target.result;

img.title=file.name;

//或者 img.src = this.result; //e.target == this

}

reader.readAsDataURL(file)

}

</script>

1. 1
2. 1
3. 1
4. 1

第8章 ssl

1. 1

第8.1章 第三方登录

1. QQ互联
2. 申请加入

https://connect.qq.com/intro/login/

身份证需要手持身份证的照片。

1. 1
2. 1
3. 1

第9章 服务器相关

1. jdk下载后格式不对服务器不能解压,其实是文件下载失败

wget --no-check-certificate --no-cookies --header “Cookie: oraclelicense=accept-securebackup-cookie” http://download.oracle.com/xxxxxx

1. cron定时任务执行时找不到脚本路径,修改yii的第一行为主机空间上php的路径

#!/usr/bin/env /usr/local/bin/php

1. 不能远程连接mysql,报错

2003 can't connect to mysql server on 113

关闭centos防火墙,如果iptables设置了也要修改

# systemctl stop firewalld.service

1. 上传文件413 Request Entity Too Large错误：

server {

...

client\_max\_body\_size 20m;

...

}

上传文件504 Gateway Time-out

location ~ \.php$ {

…

fastcgi\_index restful.php;

fastcgi\_pass 127.0.0.1:9000;

include fastcgi.conf;

fastcgi\_param

PATH\_INFO $path\_info;

fastcgi\_connect\_timeout 300s;

fastcgi\_send\_timeout 300s;

fastcgi\_read\_timeout 300s;

}

1. 设置默认时区为日本东京

date\_default\_timezone\_set(“Asia/Tokyo”);

1. 修改用户权限

# whereis sudoers

方法二： vim /etc/sudoers

## Allow root to run any commands anywhere

root ALL=(ALL) ALL

testuser ALL=(ALL) ALL

修改完毕，然后用命令 sudo su - ，即可获得root权限进行操作。

方法三： vim /etc/passwd，把用户ID修改为 0：

testuser:x:500:500:tommy:/home/tommy:/bin/bash

修改后如下

testuser:x:0:500:tommy:/home/tommy:/bin/bash

1. 配置config文件免密登录:

Host shuhu-test

HostName 47.101.54.133

Port 22

User root

将本地公匙上传到服务器

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@47.101.54.133

查看authorized\_keys文件里面添加的公钥

1. 压测：

ab -c 20 -n 10000 -k 127.0.0.1:9501/test/firstCache

ab -c 256 -n 10000 -r -k http://127.0.0.1:9501/marketing/template/scenes

报错：

[2020-05-07 11:12:23 #1714.1] WARNING swServer\_master\_onAccept(:115): accept() failed, Error: Too many open files[24]

硬限制：值只能比这个值少

sudo launchctl limit maxfiles 65536 unlimited

修改的是硬最大限制

ulimit -n 2048这个值必须小于硬最大限制

修改了之后还不行，修改-c的值，改小一点

1. fpm下的所有连接都是短连接（pconnect除外），单例也是短连接

短连接：

优点：简单，无状态对lb（load balance）友好

缺点：多余传输（三次握手，4次握手解除连接）；tcp慢启动，每次请求都会测试网络承载能力（速率一点一点往上涨）；如果网络抖动，握手阶段丢包，则重传间隔时间较长（秒级别）；对连接的占用约等于长连接

长连接问题：

1）连接失效：

mysql:wait\_timeout&interactive\_timeout(has gone away)

解决：定时发心跳维持连接

2）过多的内存占用，尤其是阻塞的情况下；

3）过多的端口占用、fd占用

4）传输效率的问题，多开连接效率低

综上：在限制了最大连接数的情况下，尽可能提高利用率，共享连接

1. 多协程共享连接
2. 多进程共享连接
3. 多机器（ip）共享连接
4. centos7 安装FFmpeg

yum install -y epel-release

rpm -v --import http://li.nux.ro/download/nux/RPM-GPG-KEY-nux.ro

rpm -Uvh <http://li.nux.ro/download/nux/dextop/el7/x86_64/nux-dextop-release-0-5.el7.nux.noarch.rpm>

yum install ffmpeg ffmpeg-devel -y

第10章 html+css

1. bootstrap

navbar和下面元素的间距，修改

.navbar {

position: relative;

min-height: 50px;

margin-bottom: 0px;

border: 1px solid transparent;

}

1. 谷歌浏览器找到要改的元素，

勾选hover后样式改变的元素，

然后在子元素里面找样式

1. 元素分类

常用的块状元素有：

<div>、<p>、<h1>...<h6>、<ol>、<ul>、<dl>、<table>、<address>、<blockquote> 、<form>

* 每个块级元素都从新的一行开始，并且其后的元素也另起一行
* 元素的height、width、line-height以及margin-top和margin-bottom都可设置
* 元素宽度在不设置的情况下，是它本身父容器的100%

常用的内联元素有：

<a>、<span>、<br>、<i>、<em>、<strong>、<label>、<q>、<var>、<cite>、<code>

* 和其他元素都在一行上
* 元素的宽度就是它包含的文字或图片的宽度，不可改变

常用的内联块状元素有：

<img>、<input>

* 和其他元素都在一行上
* 元素的height、width、line-height以及margin-top和margin-bottom都可设置

1. 层模型

**position:absolute**将元素从文档流中拖出来，然后使用left、right、top、bottom属性相对于其最接近的一个具有定位属性的父包含块进行绝对定位。如果不存在这样的包含块，则相对于body元素，即相对于**浏览器窗口**。

**position:relative**通过left、right、top、bottom属性确定元素在**正常文档流中**的偏移位置。相对定位完成的过程是首先按static(float)方式生成一个元素(并且元素像层一样浮动了起来)，然后相对于以前的位置移动，偏移前的位置保留不动（即出现了2个元素，之前的元素不显示但还是占了位子）。

**position:fixed**的相对移动的坐标是视图（**屏幕内的网页窗口**）本身，因此固定定位的元素会始终位于浏览器窗口内视图的某个位置，不会受文档流动影响。

1. 自动换行

自动换行

p

{

white-space: nowrap

}

normal默认。空白会被浏览器忽略。pre空白会被浏览器保留。其行为方式类似 HTML 中的 <pre> 标签。nowrap文本不会换行，文本会在在同一行上继续，直到遇到 <br> 标签为止。pre-wrap保留空白符序列，但是正常地进行换行。pre-line合并空白符序列，但是保留换行符。inherit规定应该从父元素继承 white-space 属性的值。

1. 去掉默认type为number的input框的箭头

<style>

input::-webkit-outer-spin-button,

input::-webkit-inner-spin-button {

-webkit-appearance: none;

}

input[type=“number”]{

-moz-appearance: textfield;

}

</style>

第11章 mysql

1. 更改数据库编码

ALTER DATABASE ‘user’ CHARACTER SET utf8

1. 外键约束

1.父表和子表

CREATE TABLE provinces(

id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

pname VARCHAR(20) NOT NULL

);

CREATE TABLE users(

id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(20) NOT NULL

pid

);

FOREIGN KEY (pid) REFERENCES provinces (id)

1. 依都服装（czty）

地区数据表

ims\_ws\_micro\_shop\_district

快递数据表

ims\_ws\_micro\_shop\_express

1. text类型不能直接存储int类型

mysql text类型不能直接存储数字，会报错，如remarksaler是text类型

//pdo\_update('ws\_retailshop\_rz\_order', array('remarksaler' => (string)$this->type), array('id' => 261));

1. 两个字段联合唯一

alter table `ims\_ws\_retailshop\_bz\_member` add unique uniacid\_openid(uniacid,openid);

1. elequent里面的with用法

在MarketingPlanTask的model里面定义

/\*\*

\* @return \Hyperf\Database\Model\Relations\BelongsTo

\*/

public function marketingPlan()

{

return $this->belongsTo(MarketingPlan::class, 'marketing\_plan\_id', 'id');

}

在别的方法里面

return MarketingPlanTask::query()

->select(['id', 'user\_group\_id', 'marketing\_plan\_id'])

->where([

['status', '=', MarketingPlanTask::STATUS\_NOT\_START],

['execution\_time', 'like', "{$current\_date}%"],

])

->with(

array(

'marketingPlan' => function ($query) {

$query->select(['priority','id'])

->where([

['priority', '>', 0],

['priority', '<', 100],

])

->orderBy('priority', 'desc');

}

)

)

->get()->toArray();

注意：foreign key和local key都要select到才会有结果

返回的结果:

Array

(

[0] => Array

(

[id] => 52

[user\_group\_id] => 31

[marketing\_plan\_id] => 70

[marketing\_plan] => Array

(

[priority] => 22

[id] => 70

)

)

[2] => Array

(

[id] => 54

[user\_group\_id] => 31

[marketing\_plan\_id] => 72

[marketing\_plan] =>

)

)

1. es相关

GET /reviews/info/\_search

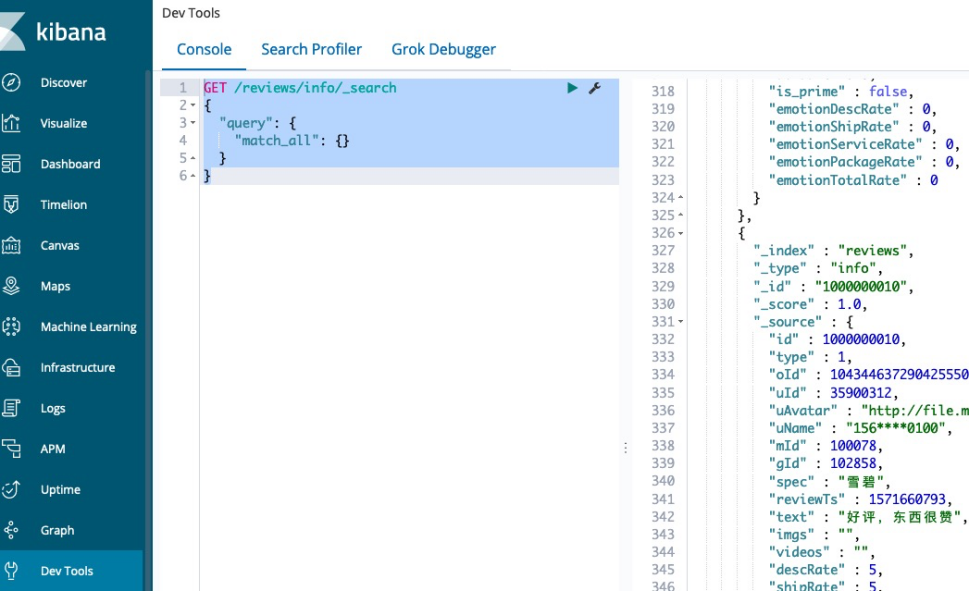
{

"query": {

"match\_all": {}

}

}



1. 1
2. 1

第12章 安装vue.js

1. cmd中，安装了node.js之后

npm -v

npm install -g cnpm --registry=https://registry.npm.taobao.org

这个很卡，多试几次

安装在了C:\Users\yii6\AppData\Roaming\npm

复制cnpm 、cnpm.cmd到C:\Program Files\nodejs（和npm.cmd同级）

复制C:\Users\yii6\AppData\Roaming\npm\node\_modules\cnpm到

C:\Program Files\nodejs\node\_modules\cnpm

升级npm

cnpm install npm -g

cnpm install vue

这个提示没权限，用管理员的cmd

# 全局安装 vue-cli

$ cnpm install --global vue-cli

1. vue项目在网页中打开

首先在文件夹最外层打开Terminal

命令行输入

cnpm i

npm start

1. 1

第13章 API相关

1. 谷歌地图

地址转经纬度：

//传入详细地址，返回-1表示没结果，否则返回经纬度字符串；服务器在国外才能使用

public function latitude($address)

{

$res =

file\_get\_contents(“https://maps.googleapis.com/maps/api/geocode/json?address=“ . $address . “&key=AIzaSyDE5zPr2gtSLCQAi7hOeX76wDw-XE12Cq0”);

$arr = json\_decode($res, true);

if (isset($arr['results']['0']['geometry']['location'])) {

return implode(“,”, $arr['results']['0']['geometry']['location']);

} else {

return -1;

}

}

1. 1

第14章 htaccess

1. 0-n条cond后有一条rule
2. http\_host域名

request\_uri除了域名之外的部分

1. 1
2. 1
3. 1
4. mac相关
5. 修改谷歌显示的语言

defaults write com.google.Chrome AppleLanguages '(en-US)'

defaults write com.google.Chrome AppleLanguages '(zh-CN)'

207. 安装软件

安装brew

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"

brew info openssl

获得当前openssl的路径

Pecl install swoole

enable openssl support? [no] : yes --with-openssl-dir=

只开启了http2和openssl

Percl install redis

Pecl install mongodb

Brew install librdkafka

Pecl install rdkafka

brew install protobuf

pecl install protobuf

brew install zookeeper

pecl install zookeeper

pecl install grpc

1. phpstorm 快捷键

多行编辑：alt+鼠标左键点击需要编辑的地方

1. 安装pecl

curl -O https://pear.php.net/go-pear.phar

sudo php -d detect\_unicode=0 go-pear.phar

Configure and Install PEAR

You should now be at a prompt to configure PEAR.

Type 1 and press return.

Enter: /usr/local/pear

Type 4 and press return.

Enter: /usr/local/bin

Press return

pecl命令可以使用了但是安装扩展还是不行

1. Configure and build php扩展 on macOS Catalina 10.15

mkdir ~/php-private

cp /usr/bin/phpize ~/php-private/

cp /usr/bin/php-config ~/php-private/

下载2个补丁包：

<https://profilingviewer.com/catalina-php-patches/phpize-catalina.patch.zip>

<https://profilingviewer.com/catalina-php-patches/php-config-7.3.11-catalina.patch.zip>

unzip Downloads/phpize-catalina.patch.zip -d ~/Downloads/

unzip Downloads/php-config-7.3.11-catalina.patch.zip -d ~/Downloads/

patch ~/php-private/phpize < ~/Downloads/phpize-catalina.patch

patch ~/php-private/php-config < ~/Downloads/php-config-7.3.11-catalina.patch

进入扩展文件包

cd redis-\*

~/php-private/phpize

./configure --with-php-config=/Users/yu/php-private/php-config

make

 sudo mkdir -p /usr/local/php/extensions

sudo cp modules/redis.so /usr/local/php/extensions/

1. 1

第16章 调试

1. gdb

b zif\_var\_dump

断点断在var\_dump函数里面

z:zend

i:internal

f:function

m:method

1. 调试协程

<?php  
use Swoole\Coroutine as Co;  
use Swoole\Timer;  
*co*::set([  
 'log\_level' => *SWOOLE\_LOG\_INFO*,  
 'trace\_flags' => 0,  
]);  
function stackTraceFormat(array *$stackTrace*): string  
{  
 $r = *str\_repeat*("=", 50) . *PHP\_EOL*;  
 $i = 1;  
 foreach (*$stackTrace* as $node) {  
 $r .= "#$i:" . *basename*($node['file']) . ":" . $node['function'] . "(" . $node['line'] . ")" . *PHP\_EOL*;  
 ++$i;  
 }  
 return $r;  
}  
  
*Co*\run(function () {  
 go(function () {  
 *Co*::sleep(1);  
 });  
 go(function () {  
 *Co*::gethostbyname('www.baidu.com');  
 });  
 go(function () {  
 *Co*::statvfs(*\_\_FILE\_\_*);  
 });  
 go(function () {  
 *Co*::readFile(*\_\_FILE\_\_*);  
 });  
 Timer::after(1, function () {  
 });  
  
 //debug  
 *Co*::create(function () {  
 *var\_dump*(*Co*::stats());  
 foreach (*Co*::list() as $cid) {  
 $backtrace = *Co*::getBackTrace($cid);  
 echo stackTraceFormat($backtrace);  
 }  
  
 echo *PHP\_EOL*;  
  
 *var\_dump*(Timer::stats());  
 foreach (Timer::list() as $tid) {  
 *var\_dump*(Timer::info($tid));  
 }  
 });  
});

远程调试：

<?php  
  
use Swoole\Coroutine as Co;  
  
*co*::set([  
 'log\_level' => *SWOOLE\_LOG\_INFO*,  
 'trace\_flags' => 0,  
]);  
  
*Co*\run(function () {  
 $socket = new *Co*\Socket(*AF\_INET*, *SOCK\_STREAM*, *IPPROTO\_IP*);  
 if (!$socket->bind('127.0.0.1', 9876)) {  
 throw new Exception('bind error');  
 }  
 if (!$socket->listen()) {  
 throw new Exception('listen error');  
 }  
 while (1) {  
 $conn = $socket->accept();  
 if (!$conn) {  
 throw new Exception('accept error');  
 }  
 *Co*::create(function () use ($conn) {  
 $conn->setProtocol([  
 'open\_eof\_check' => true,  
 'package\_eof' => "\n"  
 ]);  
 while (1) {  
 $data = $conn->recvPacket();  
 if (!$data) {  
 break;  
 }  
 $info = *var\_dump*(*Co*::stats()) . "\n";  
 if ($conn->sendAll($info) != *strlen*($info)) {  
 break;  
 }  
 }  
 });  
 }  
});

1. 1
2. 1

1

缩写

**html:**

**a**：archor

**href**：hypertext reference

**td**：table data

**tr**：table row

**br**: break

**css:**

**padding**:内边距

clears an area around the content (inside the border) of an element.

material, as cotton or straw, used to pad something.

**margin**:外边距

set the size of the white space outside the border.

the edge or border of something.

**cellpadding**:

specifies the space between the cell wall and the cell content.

**cellspacing**:

specifies the space between cells.

border-width

border-style

border-color

border-bottom-width

border-top-width

**list-style**:列表图标的显示

**line-height**：行高，两行之间的距离。

**mysql:**

**desc** descending

**asc** ascending

其它：

**API** ：application programming interfaces

**ajax**：Asynchronous JavaScript and XML

**BOM** : Byte Order Mark，隐藏字符。类似WINDOWS自带的记事本等软件，在保存UTF-8文件时，会在文件开始的地方插入三个不可见的字符（0xEF 0xBB 0xBF，即BOM），用于让记事本等编辑器识别这个文件是否以UTF-8编码。

PHP并不会忽略BOM，所以在读取、包含或者引用这些文件时，会把BOM作为该文件开头正文的一部分。这串字符将被直接执行（显示）出来。由此造成即使页面的 top padding 设置为0，也无法让整个网页紧贴浏览器顶部。

受COOKIE送出机制的限制，在这些文件开头已经有BOM的文件中，COOKIE无法送出（因为在COOKIE送出前PHP已经送出了文件头），所以登入和登出功能失效。一切依赖COOKIE、SESSION实现的功能全部无效。

**CLI** : command-line interface

**PHP-FPM**：PHP FastCGI Process Manager

**CGI** : Common Gateway Interface

**CMS**: content management system

**QPS**：Queries Per Second

**TPS** : Transactions Per Second,包括

1）用户请求服务器

2）服务器自己的内部处理

3）服务器返回给用户

**AMQP**：advanced message queuing protocol 高级消息队列协议

**AOP**：aspect oriented programming 面向切面编程

**RPC**：remote procedure call 远程过程调用

**RFC**：request for comments,是一系列以编号排定的文件。文件收集了有关互联网相关信息，以及UNIX和互联网社区的[软件](https://baike.baidu.com/item/%25E8%25BD%25AF%25E4%25BB%25B6/12053)文件。基本的互联网通信协议都有在RFC文件内详细说明。RFC文件还额外加入许多在标准内的论题，例如对于互联网新开发的协议及发展中所有的记录。

**PSR** : PHP Standards Recommendation, php标准规范

**IPC** ：Interprocess communication 进程内通信

**IoC**:Inversion of Control，控制反转

**DI:**Dependency Injection,依赖注入

**fd** : file descriptor文件描述符

**OSPF**:open shortest path first,开放最短路径优先，动态路由更新协议，用于路由器之间通信

**MAC**:media access control address,也称为局域网地址（LAN Address），以太网地址（Ethernet Address）或物理地址（Physical Address）

**LAN**:local area network