Swoole开发指南

流年

目　录

安装Swoole框架和扩展

Swoole扩展

到GitHub首页下载Swoole扩展源码，地址：https://github.com/matyhtf/swoole 下载后按照标准的PHP扩展编译方式进行编译和安装。一般是

```
phpize
./configure
make install
```

编译安装完后，修改php.ini加入extension=swoole.so开启swoole扩展。也可以通过dl('swoole.so')动态载入，推荐修改php.ini。

Swoole框架

下载swoole_framework源码，放置到您的workspace目录中即可。swoole_framework是PHP代码，只需要require/include即可，无需编译和安装。

数据库Model类

model类必须继承自Swoole\Model，必须有一个公共属性$table，标识model对应的数据库表名。

```
class Test extends Swoole\Model
{
    public $table = 'ask_category';
}
```

可选属性

public $primary = "r_id"; 设置主键的字段名称，默认为'id'

model->get

从数据库中获取单条记录

```
function get($object_id=0,$where='')
```

参数为主键ID的值，生成的SQL类似这样

```
select * from table where id = {$id} limit 1
```

如果希望使用另外的字段进行查询，需要传入$where参数，如 $model->get('me', 'name')。则生成的SQL为：

```
select * from table where name = 'me' limit 1
```

返回一个Record对象，在此对象上可以进行更多ORM操作。

model->set

修改单条数据库记录的内容

```
function set($id, $data, $where='')
```

$id 参数为主键ID，$where可以指定其他字段作为查询条件。
$data为修改的内容，必须为键值对应的数组，key将作为数据库字段名称，value为值。

model->del

删除单条数据库表记录。

```
function del($id, $where=null)
```

model($model_name)

创建/获取Model对象

```
$object = model('UserInfo');
$object->get(10001);
```

相当于SQL语句的：

```
select * from user_info where id = 10001 limit 1
```

table($table_name)

使用数据库表名来创建一个Model，这个函数与model不同，它不需要apps/models/目录中有对应的PHP文件。直接可生成Model对象。

```
$object = table('user_info');
$object->get(10001);
```

相当于SQL语句的：

```
select * from user_info where id = 10001 limit 1
```

数据库ORM接口

可以使用createModel('ModelName')->get($id)和createModel('ModelName')->all来生成数据的ORM封装对象。

```
$content = $model->get(1);   //这里返回的是一个Record对象
$content->title = 'hello world';  //Update操作
$content->save();                 //保存操作，这时会执行SQL语句
echo $content->addtime;           //输入值
$content->delete();                 //删除此条数据

$all = $model->all();        //这里返回一个RecordSet对象
$all->filter('userid=2');    //增加一些限定条件，避免读取全部的数据库内容

//遍历符合条件的所有数据库记录
foreach($all as $obj)
{
    $obj->title = 'hello';
    echo $obj->content;
}
```

Socket网络开发

Swoole提供了底层的网络socket服务器实现。普通用户只需要实现协议或基于现有的协议进行二次开发。

底层Driver

- BlockTCP 阻塞的tcp/udp server, 请求按顺序执行，必须处理完一个请求才能继续处理新的请求。
- SelectTCP 使用select做IO复用的异步非阻塞 server，可以同时维持多个TCP连接。select最大只能维持1024个连接，并且性能会随着连接数量增多而下降
- EventTCP 使用libevent做IO复用的异步非阻塞Server，可以同时维持大量TCP连接，性能不会随连接增多而下降
- Server 使用Swoole扩展作为底层的网络驱动，推荐使用

> BlockTCP和SelectTCP可用于Windows

协议Protocol

Protocol决定应用层如何处理数据，如何回应客户端。开发者可以基于Swoole提供的Protocol之上进行开发，也可以自行实现Protocol Swoole框架自带的Protocol有：

- HttpServer 是http协议的实现，提供web server的功能
- WebSocket 是websocket协议的实现
- AppServer 是Web应用服务器的实现
- SOAServer 是SOA协议的实现，（SOA不是标准协议）

运行Server

在你的脚本中只需要选择一种合适的Driver，构造一个Protocol对象。并调用Driver的setProtocol方法，组装到Drvier中即可，最后调用Driver的run方法进入网络事件循环即可。

test.php

```
require __DIR__ . '/../libs/lib_config.php';
$AppSvr = new Swoole\Network\Protocol\HttpServer();

$AppSvr->setDocumentRoot(WEBPATH);
$AppSvr->setLogger(new \Swoole\Log\EchoLog(true)); //Logger

$server = new \Swoole\Network\SelectTCP('0.0.0.0', 8888); //这里选择了selectTCP驱动
$server->setProtocol($AppSvr); //组装协议

$server->run(array('worker_num' => 1, 'max_request' => 5000)); //运行
```

```
php test.php
```

TCP服务器

- Swoole\Network\Server 使用swoole扩展作为底层驱动
- Swoole\Network\SelectTCP 使用PHP提供的stream_select作为事件驱动方式
- Swoole\Network\BlockTCP 阻塞方式的TCP
- Swoole\Network\EventTCP libevent扩展作为底层驱动

Web服务器

Swoole框架提供的WebServer有3种使用方法

一、直接使用HttpServer

HttpServer支持静态文件和include file。业务代码不需要写任何Server的代码，只需要设置document_root，并编写对应php文件。这种使用方法与Apache/Nginx+FPM类似。

server.php

```
$AppSvr = new Swoole\Network\Protocol\HttpServer();
$AppSvr->loadSetting("./swoole.ini"); //加载配置文件
$AppSvr->setDocumentRoot(__DIR__.'/webdocs/'); //设置document_root

$server = new \Swoole\Network\Server('0.0.0.0', 9501);
$server->setProtocol($AppSvr);
//$server->daemonize(); //作为守护进程
$server->run(array('worker_num' => 2, 'max_request' => 1000));
```

webdocs/index.php

```
<?php
echo "hello world";
```

在浏览器中打开http://localhost:9501/index.php

二、继承HttpServer

业务代码只需要继承此类，并自行实现onRequest方法即可。

```
    /**
     * 处理请求
     * @param $request
     * @return Swoole\Response
     */
    function onRequest(Swoole\Request $request)
```

onRequest方法参数为解析好的Request对象

- $request->post : $_POST
- $request->get : $_GET
- $request->cookie : $_COOKIES
- $request->file $_FILES

onRequest方法必须返回一个Response对象

- $response->body 返回的HTML内容
- $response->head HTTP头信息

三、使用AppServer

基于AppServer类开发，就必须遵循Swoole MVC规范。具体可以查看examples/和apps/中的示例程序。 apps/目录中存放应用代码。

目录	说明
apps/controllers	控制器代码
apps/models	数据模型代码
apps/teamplets	模板文件
apps/config	配置文件

WebSocket

Swoole框架提供了WebSocket协议的实现。具体代码可以参考 examples/websocket_server.php和examples/websocket_client.hml。

如何使用

应用程序代码只需要继承 Swoole\Network\Protocol\WebSocket，并实现onMessage方法即可。onMessage方法在服务器端收到客户端消息时回调。Swoole框架已经处理好了connect/accept，打包解包等工作。应用层无需关心。 在onMessage方法中，可调用

```
$this->close($client_id) //关闭此连接
$this->send($client_id, $response_string) //向某个客户端发送数据
```

如果你的代码中没有阻塞，建议使用Swoole扩展的SWOOLE_BASE模式，如果业务代码中存在阻塞操作，请使用SWOOLE_PROCESS模式，并根据实际处理时间，设置worker_num参数，启用多进程。

Nginx+Swoole服务器配置

nginx配置：

```
server {
    listen  80;
    server_name  www.swoole.com;
    root  /data/wwwroot/www.swoole.com;

    location / {
        if (!-e $request_filename){
            proxy_pass http://127.0.0.1:9501;
        }
    }
}
```

9501就是swoole服务器监听的地址。root设置为静态文件的目录。当请求静态文件是由Nginx直接处理，当请求的文件不存在时，发送给Swoole服务器，来进行处理。

Apache+Swoole服务器配置

Apache可以使用代理指令或Rewrite来实现。

```
<VirtualHost *:80>
    ServerName www.swoole.com
    DocumentRoot /data/webroot/www.swoole.com
    DirectoryIndex index.html index.php

    <Directory "/data/webroot/www.swoole.com">
        Options Indexes FollowSymLinks
            AllowOverride None
            Require all granted
    </Directory>

#   ProxyPass /admin !
#   ProxyPass /index.html !
#   ProxyPass /static !
#   ProxyPass / http://127.0.0.1:9501/

    <IfModule mod_rewrite.c>
        RewriteEngine On
        RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} !-f
        RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} !-d
        RewriteRule ^(.*)$ http://127.0.0.1:9501$1 [L,P]
    </IfModule>
</VirtualHost>
```

控制器Controller

命名空间

SwooleFramework使用了完全命名空间化的代码管理。

- Swoole命名空间下的类是library类，对应目录是 {WEBPATH}/libs/Swoole/
- App命名空间下是用户类，对应目录是{WEBPATH}/apps/classes/

如果你希望增加另外的命名空间，可以使用

```
Swoole\Loader::addNameSpace($root, $path);
```

$root就是命名空间的根名称，$path是对应的目录绝对路径。

文件上传组件

swoole提供了文件上传模块，可以自动处理来自HTTP POST的文件上传。在Controller中调用

```
$this->upload->save('Upfile_key');
//需要生成缩略图
$this->upload->thumb_width = 136; //缩略图宽度
$this->upload->thumb_height = 136; //缩略图高度
$this->upload->thumb_qulitity = 100; //缩略图质量

//自动压缩图片
$this->upload->max_width = 600; //约定图片的最大宽度
$this->upload->max_height = 600; //约定图片的最大高度
$this->upload->max_qulitity = 90; //图片压缩的质量
```

需要在apps/configs中配置upload.php，base_dir 就是上传文件的根目录。

```
$upload = array(
    'base_dir' => WEBPATH.'/uploads/',
);
return $upload;
```

即可自动处理上传的文件。

Redis

修改 apps/configs/redis.php ，加入配置。

```
$redis['master'] = array(
    'host' => '172.19.104.157',
    'port' => 6379, //默认为6379
    //'database' => 16, //数据库ID，可选
);
```

- 默认取master项作为redis配置，$this->redis->get，框架会读取$this->config['redis']['master']作为配置
- 使用$this->redis('master2')->get 实现多实例访问，框架会读取$this->config['redis']['master2']作为配置

Database

修改 apps/configs/db.php ，加入配置。

```
$db['master'] = array(
    'type'    => Swoole\Database::TYPE_MYSQLi,
    'host'    => "127.0.0.1",
    'port'    => 3306,
    'dbms'    => 'mysql',
    'engine'  => 'MyISAM',
    'user'    => "root",
    'passwd'  => "root",
    'name'    => "db_live",
    'charset' => "utf8",
    'setname' => true,
    'persistent'  => true,
);
return $db;
```

使用方式

单数据库，默认读取master配置

```
$this->db->query("select * from test");
```

多数据库，读取制定的数据库配置

```
$this->db('other_db_config')->query("select * from test");
```

驱动类型

swoole框架支持3种驱动类型：

1. Swoole\Database::TYPE_MYSQL，使用mysql扩展
2. Swoole\Database::TYPE_MYSQLi，使用mysqli扩展
3. Swoole\Database::TYPE_PDO，使用PDO扩展

配置选项

- charset 制定数据库字符集
- setname 在连接服务器成功后发送set names $charset
- persistent 启用MySQL数据库长连接

Swoole\Database->insert

将一个数组插入到数据库表中。

```
function Swoole\Database->insert(array $record, string $table_name);
$db->insert(array('field1' => 12345, 'field2' => 'hello'), 'test');
```

- $record是一个键值对应数组，键务必要一一对应数据库表的字段
- $table_name 指定数据库表名
- 操作成功返回true，失败返回false

框架规范

目录规范

假设根目录为$ROOT。

$ROOT/apps

应用程序代码，此目录中的代码是公用的，包括类，配置，模板，控制器，Model等。此目录不得放置静态文件，如js, css, jpg, html等，必须全部为.php文件。此目录不允许http直接访问。

- $ROOT/apps/controllers Web应用的控制器类代码
- $ROOT/apps/models 数据模型封装类代码
- $ROOT/apps/configs 配置文件，通过$php->config['db']['master'] 这样来访问
- $ROOT/apps/classes 类库，这里存放所有用户定义的类，必须符合psr-0规范，文件名必须为 {类名}.php，顶层命名空间必须为App。
- $ROOT/apps/templates 模板文件目录

> 命名空间：如 new App\Hello\Test 类，会映射到 $ROOT/apps/classes/Hello/Test.php
> 配置文件：如 $php->config['db']['master'] 或 Swoole::getInstance()->config['db']['master'] 会映射到$ROOT/apps/configs/db.php文件，db.php中必须返回数组，key为master。
> 数据模型：model('UserInfo')或者$php->model->UserInfo 会映射到 $ROOT/apps/models/UserInfo.php

$ROOT/static

静态文件目录，比如js, css, jpg, html等。此目录允许http直接访问。

$ROOT/index.php

web网站单一入口文件，可直接放到根目录，或者单独建立目录存放，如$ROOT/webroot/index.php

$ROOT/server.php

服务器程序启动入口。

自定义路由

在Swoole::$php->runMVC() 调用之前可以修改默认的路由函数。

```
Swoole::$php->router(function(){
    return array('controller' => 'YourController', 'view' => 'YourView');
});
Swoole::$php->runMVC();
```

设置后将使用制定的函数作为路由，路由函数务必要返回一个数组，包含控制器和视图的名称。

```
 array('controller' => 'YourController', 'view' => 'YourView');
```

URL映射规则

swoole框架使用强规则来做URL映射。如下面的URL

```
http://127.0.0.1/hello/index/
```

将会映射到 apps/controllers/Hello.php 中的 Hello::index 方法。

自定义URL

修改 `apps/configs/rewrite.php` ，增加正则配置。具体使用方法请看示例。

```
$rewrite[] = array(
    'regx' => '^/content/([a-z]+)\.html$',
    'mvc'  => array('controller' => 'content', 'view' => 'getlist'),
    'get'  => 'app',
);
```

- $regx需要传入一个正则表达式，符合该正则表达式就会进入此条URL路由
- $mvc指定对应的controller, view名称
- $get可以将正则表达式中的子表达式（括号中的表达式）赋值到$_GET参数中

魔法参数

swoole提供了自动参数处理规则。

- /hello/index/100 ，自动赋值给 $_GET['id'] = 100
- /hello/index/cid-1-name-rango ，自动赋值到 $_GET['cid'] = 1， $_GET['name'] = 'rango'