河北师范大学 软件学院
Software College of Hebei Normal University

Linux平台PHP服务端开发——

第十讲 简单的Web聊天室程序

实现方式

- 类型：很简单的聊天室程序。先不做验证功能，仅仅基于连接。
- 使用Swoole + Websocket协议可以实现服务器实时推送。
- 使用Memcached缓存连接信息。
- 使用一个简单的通信协议。
- 服务器接收到数据，直接转发给其他成员。

设计简单的通信协议

- 通信数据使用JSON格式。前端JS可以直接处理JSON数据，PHP使用json扩展也可以很方便的处理JSON数据。

客户端发送：

```
{
  “msg”:”I am a php programmer”
}
```

客户端接收：

```
{
  “from_id” : “12”,
  “msg”:”I am a php programmer”,
  “msg_time”:”1456690889”
}
```

服务器推送：

```
{
  “msg”:”you are login as 23 id”,
  “msg_source”:”server”
}
```

为何要使用Memcached缓存连接数据

- Swoole使用的是多进程。仅仅通过一个变量保存连接数据是不能保证同步的。
- 管理进程会把请求按照指定规则分配给每个进程，而这样每个进程获得的连接信息是有差异的。
- 避免多进程的数据不能共享问题，使用一个公共的数据缓存。

设计实现分析

- 因为服务要在后台运行，所以要创建守护进程。
- 要能够处理SIGTERM信号，捕获信号以后，清理memcached缓存数据，关闭连接。
- 服务端程序仅仅是做转发处理。

初始化操作

- 开启Memcached连接，并初始化websocket_server对象。

```
function __construct()
{
    $this->mcache = new Memcached('websocket_pool');
    $this->mcache->addServer('localhost',11211);
    $this->server = new swoole_websocket_server('localhost',9876);
    $this->server->set([
        'daemonize' => 1
    ]);
}
```

握手连接建立后：onOpen事件

- 连接建立后要把连接信息加入到Memcached缓存。并返回一条提示信息。

```
public function on_open($server, $req) {
    $this->mcache->set($this->conn_head.$req->fd, $req->fd);
    $sys_msg = [
        'msg_source'=>'server',
        'msg'=>'you are login at '.$req->fd
    ];

    $server->push($req->fd,json_encode($sys_msg));
}
```

收到信息：onMessage事件

- 获取用户发送的数据。并判断如果数据为空则不转发。
- 生成要发送的消息。
- 从Memcached获取保存的所有连接，并判断如果不是当前连接则进行转发。

收到信息：onMessage事件代码示例

```
public function on_message($server, $cnn) {
    $data = json_decode($cnn->data,true);
    $msg = (isset($data['msg'])?$data['msg']:'');
    if (empty($msg)){return ;}
    $send_msg = [
        'from_id'=>$cnn->fd,
        'msg'=>$msg,
        'time'=>time(),
    ];
    $keys = $this->mcache->getAllKeys();
    $this->mcache->getDelayed($keys);
    $key_vals = $this->mcache->fetchAll();
    foreach ($key_vals as $kv) {
        if ($kv['value']==$cnn->fd) {
            continue;
        }
        $server->push($kv['value'],json_encode($send_msg));
    }
}
```

连接关闭：onClose事件

- 关闭连接要从Memcached去除缓存数据。

```
public function on_close($server,$fd) {
    $this->mcache->delete($this->conn_head.$fd,0);
}
```

处理SIGTERM信号：onShutdown事件

- 进程捕获到SIGTERM信号要做后续处理。

```
public function on_shutdown($server) {
    $this->mcache->deleteMulti($this->mcache->getAllKeys());
    $this->mcache->quit();
}
```

前端页面JS如何进行Websocket连接

```
var ws = new WebSocket("ws://127.0.0.1:9876");
ws.onopen = function() {
    //连接建立执行的函数，通过onOpen事件触发
};
ws.onclose = function() {
    //连接关闭执行的函数，onClose事件触发
}
ws.onmessage = function (evt) {
    //onMessage事件，通过evt.data获取收到的数据。
};
```