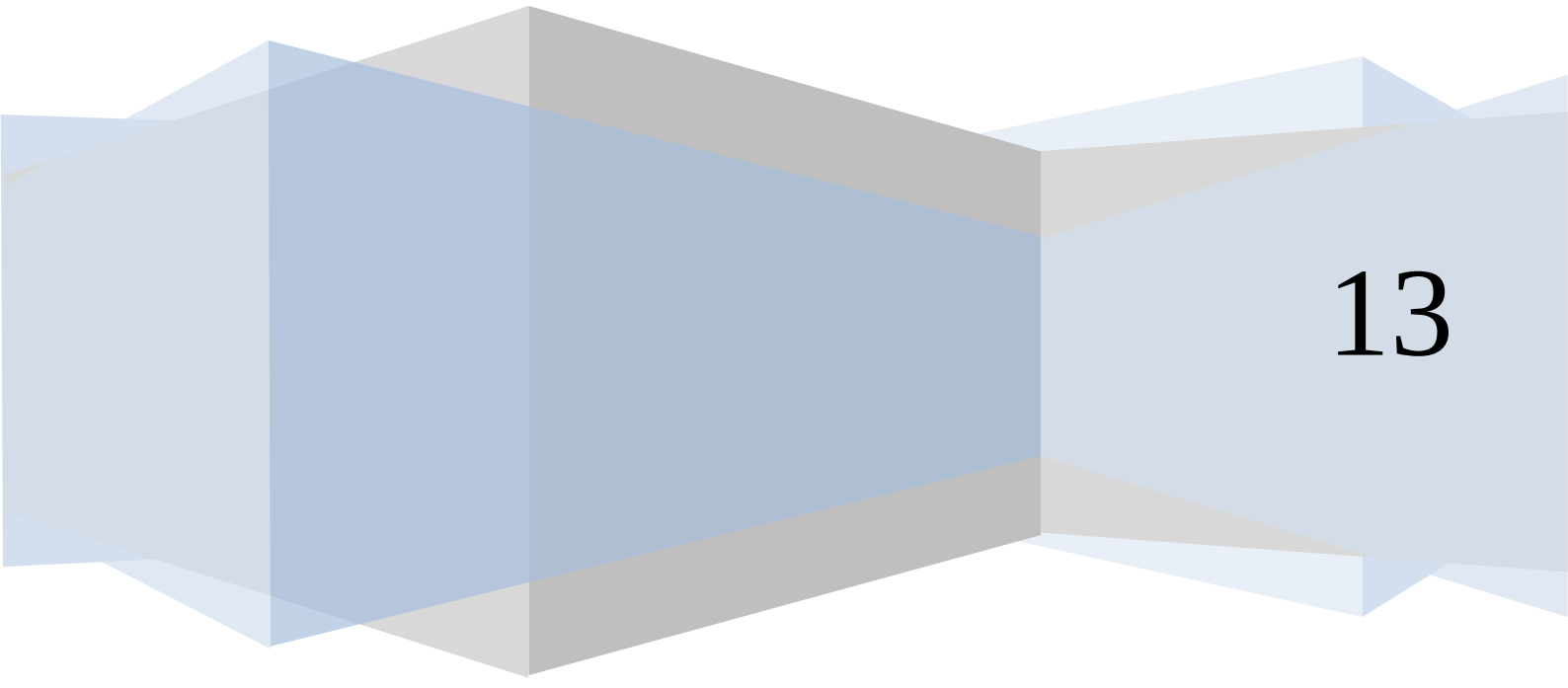
SCNU

zenphpWS3 完全开发手册

暗夜在火星



13

简介

关于 Web Services

根据百度百科定义：Web Service 是基于网络的、分布式的模块化组件，它执行特定的任务，遵守具体的技术规范，这些规范使得 Web Service 能与其他兼容的组件进行互操作。

下图红色虚线方框内表示 Web Services 所处在的位置和作用。

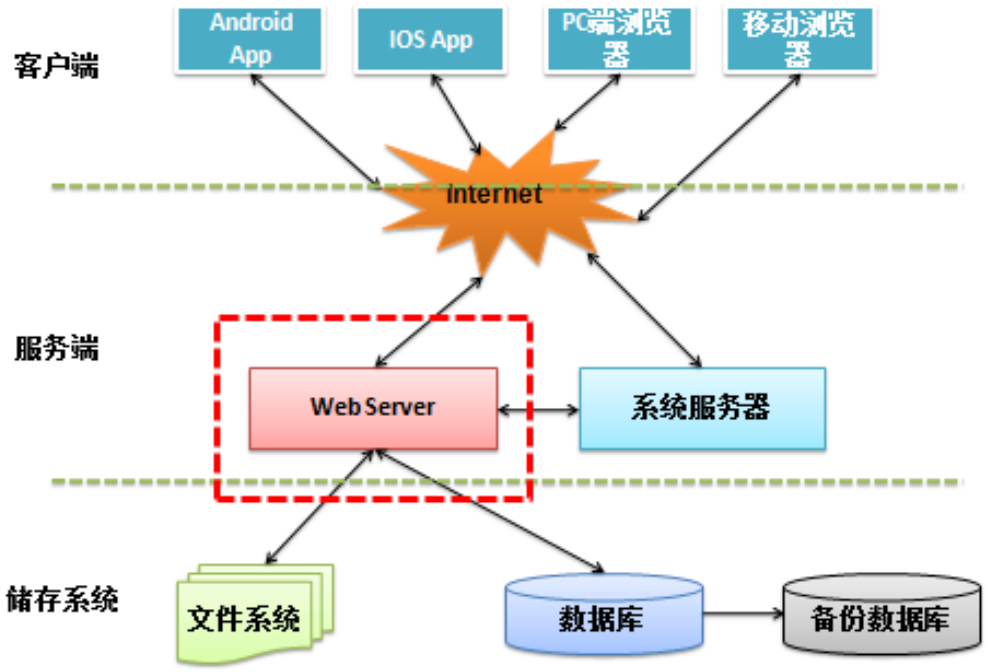


图 0-1 Web Services 所处在的位置和作用

zenphpWS3 命名意义

1) zen 英文意思为禅宗，具有佛家思想，表示集百家精华于一身。
2) php 表示用 PHP 语言实现。
3) WS 表示 Web Services。
4) 3 狭义上表示支持 RPC、SOAP 和 HTTP3 种协议，广义上表示多种功能。

快速说明

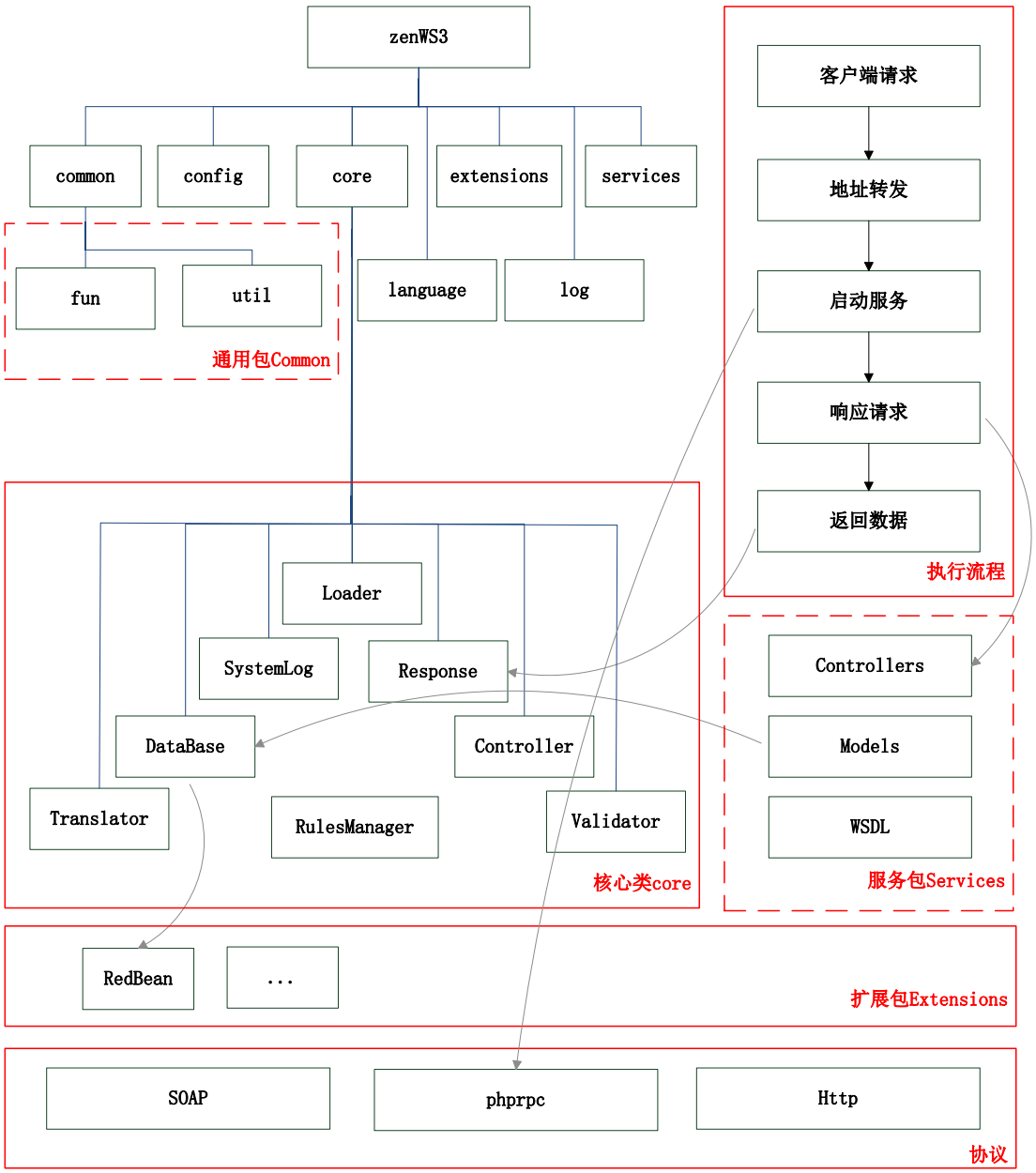
zenWS3 是自主开发、利用 PHP 实现并基于 phprpc 的 Web Services 轻量级开发框架，融合了开源社区的优秀框架精神和相关开发框架、应用框架（如 Yii、ThinkPHP、phpcms、RedBean 等），支持远程调用协调 RPC、简单对象接入协议 SOAP 和 HTTP 协议，具有**便于开发、便于使用、便于扩展**三大特点。

文档说明

此开发手册提纲参考自《ThinkPHP2.1 完全开发手册.pdf》。

1. 架构设计

结合面向对象思想，以及设计模式、分层思想，并参考各大开源框架，将系统设计如下：



到目前为止，该框架具有自动数据验证及生成、统一返回格式设定、日记纪录、自动加载文件、数据库处理、身份验证等功能。

1.1. 目录结构

Common：通用包，包含工具包 Util 和函数库包 Fun。
Config：配置包，包含全部的配置（系统配置和应用配置）。
Core： 核心包，主要的系统实现类。
Extensions：扩展包，系统本身自带 RedBean 包，也可存放用扩展。
Language： 语言包。
Log：日记目录，用于保存系统运行后生成的日记文件。

Phprpc：phprpc 包。
Services：服务包，包含控制器 Controllers、Models 和 WDSL 包。

1.2. 执行流程

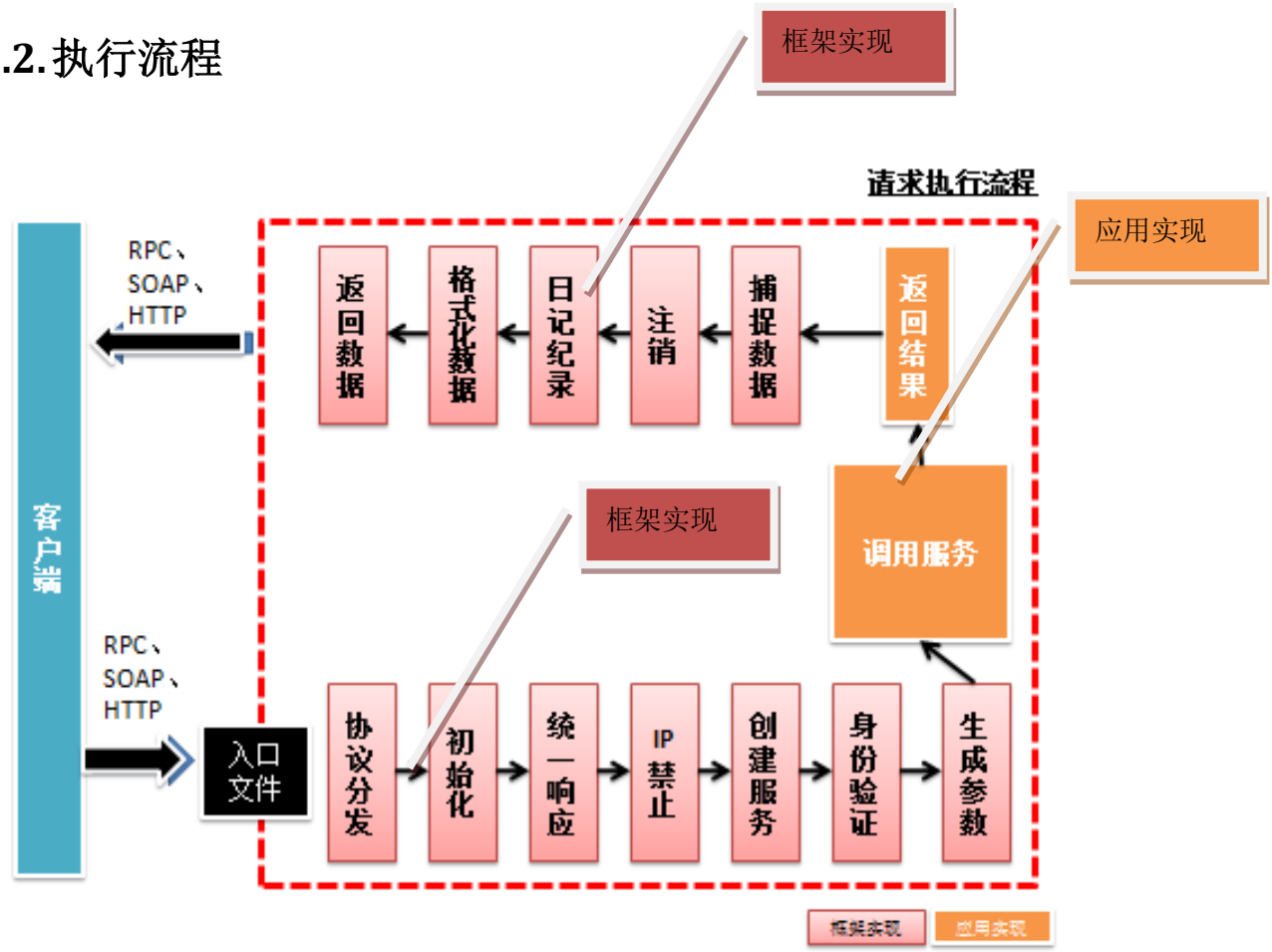


图 1-2-1 服务执行流程

1.3. 支持协议

支持三种协议，分别是 phprpc、PHP 官方自带的 SOAP 和 HTTP 协议。

目前为止，这三种协议支持的客户端开发语言调用分别如下。

表 1-3-1 各种协议支持的客户端语言

协议	支持开发语言	更多
phprpc	根据 phprpc 官方，目前可支持 java, .NET, PHP, Python, Perl, JavaScript 等，详情请见：http://www.phprpc.org/	
soap	由 PHP 官方实现，只支持 PHP 调用。	
http	任意语言，可能过浏览器直接打开查看结果。	

1.4. 返回格式

目前支持三种返回格式，分别是 json/xml/array，而且适用于任何协议。

1.5. 命名规范

使用结合 Yii 框架和 ThinkPHP 命名规范，以及驼峰法命名法进行编码命名。

主要有：

1) 类文件名：统一以.class.php 为后缀。
2) 方法与变量：利用驼峰命名法，开头小写，后面用大写字母分隔。
3) 注释：统一的文件、类、方法注释风格。

1.6. 入口文件

采用单一入口。

下面是一个标准入口文件的写法：

```
<?php
// 加载框架公共入口类文件
require(dirname(__FILE__).'/ZenWebService.class.php');

//实例化一个 Web Server 应用实例
ZenWebService::run();
?>
```

1.7. 身份验证

系统身份验证与服务身份验证相结合。

1.8. 数据库抽象层

通过 RedBean 实现。框架通过 DataBase（外观模式）负责对数据库的连接和断开等操作，以及各种的数据库操作。便于日后改用其它类库框架实现对数据库的操作。

关于 RedBean，请见：http://www.redbeanphp.com/

1.9. WDSL 查询

1.10. 函数库

采用了 phpcms 的部分核心函数库，如密码加密，获取 IP 地址，判断是否为合法用户名等通用操作函数。

1.11. 扩展

系统默认扩展使用了 RedBean 扩展，以实现对数据库的操作。

2. 构建服务

2.1. 入口文件

下面是一个标准入口文件的写法：

```php
<?php
// 加载框架公共入口类文件
require(dirname(__FILE__).'/ZenWebService.class.php');

//实例化一个Web Server应用实例
ZenWebService::run();
?>
```

2.2. 系统配置

```
//调试
        'DEBUG' => true,                            //是否开启调试

        //数据库
        'DB_ON' => true,                            //是否开启数据库
        'DB_TYPE' => 'mysql',                           //数据库类型
        'DB_HOST' => 'localhost',                       //数据库域名
        'DB_NAME' => '',                    //数据库名字
        'DB_USER' => 'root',                        //数据库用户名
        'DB_PWD' => '',                             //数据库密码
        'DB_PORT' => '3306',                        //数据库端口
```

```
        'DB_PREFIX' => '',                          //数据库表前缀

        //系统默认参数
        'SYS_DEFAULT_FORMAT' => 'json',             //默认返回数据格式：
json/xml/array
        'SYS_DEFAULT_PROTOCL' => 'http',            //默认请求协议：
rpc/soap/http
        'SYS_DEFAULT_CONTROLLER' => 'Default',      //默认控制器
        'SYS_DEFAULT_ACTION' => 'action',           //默认方法
        'SYS_DEFAULT_LANGUAGE' => 'CN',             //默认语言包

        //验证
        'VERIFY_APP_KEY' => false,                  //是否开启App Key验证
        'VERIFY_TOKEN' => true,                     //是否开启Token验证
        'APP_KEY' => 'quCSUczKBHrKGvjMHIjRurWt',    //App Key
```

2.3. 服务定义

需要继续 Controller 类，并存放在 Services/Controllers 目录下面。

2.4. 业务逻辑

需要存放在 Services/Models 目录下面。

2.5. WSDL 建立

3. 开发指南

3.1. 配置

3.1.1. 获取配置

使用：Config::get(配置键值); 不存在时，返回 null。

示例

```
$value =    Config::get('VERIFY_APP_KEY');    //获取是否需要验证 AppKey
```

3.1.2. 动态设置配置

使用：Config::set(配置键值，配置值);

示例

```
Config::set('VERIFY_APP_KEY', false);          //取消验证 AppKey
```

3.2.服务（控制器）定义

服务组是 Web Services 的核心部分，主要负责响应各种请求，并返回相应的数据。

3.2.1. 编写服务类

需要继承服务基类 Controller，如定义用户服务类，其中包含注册操作：

```
<?php
class UserController extends Controller
{
    public function register()
    {
        //未实现。。。
        return false;
    }

}
```

3.2.2. 初始化（更改配置）

服务类需要进行初始化工作，可以将初始化的操作放在构造函数内，如更改配置。

```
<?php
class UserController extends Controller
{
    public function __construct()
    {
        Config::set('VERIFY_APP_KEY', false);
        parent::__construct();
    }
}
```

值得注意的是，需要调用父类的构造函数，否则不能正确初始化系统所需要的操作。

3.2.3. 设置参数规则

通过重定义父类的 getRules()函数，可以设置参数规则，可以对请求提供的参数进行过滤并按指定的规则创建参数，避免了开发人员重新处理获取参数的操作。

其中**'*'**所对应的规则，适用于此服务类的全部操作，避免同一条规则重复定义。如果具体方法定义了重复的规则，会覆盖通用（'*'）规则。

如设置用户服务类的注册操作所需要的参数：

```
<?php
class UserController extends Controller
{
    protected function getRules()
    {
        return array(
          '*' => array(
                      'username' => array('type' => 'string',
                                'default' => '',
                                'require' => true,
                                ),
                  ),
            'register' => array(
                'password' => array('type' => 'string',
                            'default' => '',
                            'require' => true,
                            ),
            ),
        );
    }
}
```

这样就可以为注册操作定义了用户名 username 和密码 password 这两个参数。
关于参数规则，请见 **3.7 参数规则**。

3.2.4. 实现服务操作

3.2.4.1. 参数获取

3.2.4.1.1. 类成员变量获取

通过设置规则（请见 3.2.3. 设置参数规则），系统将会把生成的参数保存在类的成员变

量里面。

如获取已定义 好的用户名，直接使用`$this->username` 即可。

3.2.4.1.2. 手动获取

步骤如下：

1) 设置规则
2) 通过参数生成器生成参数
3) 检查错误信息
4) 使用参数

示例代码：

```
//设置规则
    $rules = array(
        'username' => array('type' => 'string',
                            'default' => '',
                            'require' => true,
                            ),
        'password' => array('type' => 'string',
                            'default' => '',
                            'require' => true,
                            ),
        );

    //通过参数生成器生成参数
    $error = '';
    $params = Validator::getParams($rules, $error);

    if(!empty($error))
    {
        //参数获取失败，$error 纪录了错误的提示信息。。。
    }else{
        //获取 成功，通过$params['username'],$params['password']访问。。。。
    }
```

3.2.4.2. 添加错误

通过父类的`addError($newError, $isNeedTran = true)`方法，可以添加错误信息。

示例：

```
$this->addError('Username not exsit!');    //需要翻译
```

或者

```
$this->addError('用户名不存在！', false);        //不需要翻译
```

3.2.4.3. 添加调试信息

通过父类的 addDebug($key, $value)方法，可以添加调试信息。

3.2.4.4. 设置成功状态

在成功响应后，需要手动设置成功状态，调用：$this->succeed();

注意，此成功表示操作成功，如登录失败不属于操作成功。也就是说在没有错误信息情况下，视为成功，否则视为失败。默认操作返回状态为 FAIL， 需要手动设置成功状态。

3.3. 模型

3.4. WDSL

3.5. 文件加载

通过使用 ZenLoader:: *load*($path, $type = '');可以加载项目文件。

其中，$type 是为了方便书写文件路径，表示如下：

给 3-4-1 文件路径简化标记说明

$type	表示包	代表路径
E	扩展包	PRE_PATH.'extensions/'.$path
C	服务（控制器）包	PRE_PATH.'services/Controllers/'.ucfirst($path).'Controller.class.php'
M	模型包	PRE_PATH.'services/Models/'.ucfirst($path).'Model.class.php'
U	工具包	PRE_PATH.'common/util/'.$path
空		不表示简化路径。

示例：

如加载用户模型类文件，

可以用

```
ZenLoader::load('services/Models/UserModel.class.php');
```

或者

```
ZenLoader::load('User', 'M');
```

另外，对于模型类，不需要手动导入，系统会在需要时候自动导致。

3.6. 参数生成器

通过 `Validator::getParams($rules, &$error = '', $data = null)` 可以根据提供的参数规则和数据源（默认为`$_REQUEST`），生成所需要的参数，并把错误信息保存在`$error`。

3.7. 参数规则

对于参数规则，提供了一个用于管理参数规则的容器 `RulesManager`，并提供了转认的系统参数所对应的规则（请见 `4.2.系统参数说明`）。

规则为一数组，对应各个字段的说明如下：

给 3-7-1 参数规则字段说明

字段	说明	备注
name	参数返回变量名字，为空时使用标记名字。	
type	转换变量类型。目前支持：string/int/float/	
default	缺省默认值。	
require	是否必须，为 true/false。	

3.8. 翻译

可以通过客户端提供的请求参数获取所需要的语言包，如果没有则使用系统默认的语言包。

翻译使用 Translator::get($key);

示例：

//Language/CN/common.php

```
<?php
return array(
    'Username not exsit!' => '用户名不存在！',
);
?>
```

//其他 php 文件调用

$message = Translator::get(‘Username not exsit!’);

如果设置了中文（CN）语言包，并提供了对应的翻译文件，刚返回对应的翻译。

3.9. 日记

由系统自动纪录在本地文件，以每小时生成一个新的文件做纪录，应用可以定义需要保存的错误级别，默认保存全部请求状态下的纪录。

3.10. 安全

3.10.1. 身份验证

使用服务端验证（token）和客户端验证（md5 加密的 App Key）相结合方案，同时支持手动设置或取消验证机制，统一但不失灵活。

3.10.2. 数据加密

使用 phpcms 核心的加密方法。

3.10.3. SQL 注入

通过 RedBean 实现。

3.10.4. IP 段禁止

可以通过在配置文件配置多个需要禁止的 IP 段，让系统实现自动过滤。

对应的配置如下：

```
'BAN_IPS' => array(                         //禁止的IP段
        array( 'min' => '',                 //起始IP地址
               'max' => '',                 //结束IP地址
               'reason' => '',              //原因
        ),
```

4. 服务调用

4.1. 客户端调用

4.1.1. 通过 RPC 协议调用

前置条件：

需要使用 phprpc 包，相关下载链接：http://www.phprpc.org/zh_CN/download/

```
<?php
require (dirname(__FILE__)."phprpc/phprpc_client.php");

$appKey = 'quCSUczKBHrKGvjMHIjRurWt';          //应用KEY值，由服务器提供

$t = @time();
$k = md5($t.$appKey);

$host = "http://localhost/zenws3/index.php?"; //服务器域名
$sysParam = "c=User&a=login&t=$t&k=$k";           //系统参数
$appParam = "&username=test&password=1234";       //应用参数

$client = new PHPRPC_Client();
$client->useService($host.$sysParam.$appParam);          //
$data = $client->response();        //请求响应

if($data instanceof  PHPRPC_Error)
{
    //异常处理。。。
    echo $data->toString();
}

//处理返回的数据
var_dump($data);
?>
```

4.1.2. 通过 SOAP 调用

```
<?php
```

```
$appKey = 'quCSUczKBHrKGvjMHIjRurWt';          //应用KEY值，由服务器提供

$t = @time();
$k = md5($t.$appKey);

$host = "http://localhost/zenws3/index.php?"; //服务器域名
$sysParam = "c=User&a=login&t=$t&k=$k";           //系统参数
$appParam = "&username=test&password=1234";       //应用参数

try {
    $client = new SoapClient(null,
        array('location' => $host.$sysParam.$appParam,'uri' =>
"http://localhost/",)
    );

    $data = $client->response();

    //处理返回的数据。。。
    var_dump($data);
}catch(SoapFault $fault){
    echo "Error: ".$fault->faultcode.", string: ".$fault->faultstring;
}
?>
```

4.1.3. 通过 HTTP 协议调用

可以通过在浏览器打开链接查看结果。

以 IOS 客户端为例：

```
NSString *requestStr =
$str(@"%@&c=User&a=register&username=%@&password=1%@&p=http",_generateCommon
Param(),_usernameFiled.text,_password2Filed.text);
[[MyHttpClient sharedInstance]commandWithPath:requestStr onCompletion:^(NSDictionary
*dict)
{
    NSLog(@"dict:%@",dict);
    if ([[dict objectForKey:@"status"] isEqualToString:@"OK"]) {
        [SVProgressHUD dismissWithSuccess:@"success"];
        [self.view removeFromSuperview];
    }
} failure:^(NSError *error) {
```

```
        NSLog(@"dict:%@",error);
        [SVProgressHUD dismissWithError:[error description]];
}];
```

4.2. 系统参数说明

表 4-2-1 系统参数说明

名字	说明	备注
a	（必须）方法名字	可配置默认值
b		
c	（必须）控制器名字	可配置默认值
d		
e		
f	（可选）返回的数据格式，有 josn/xml/default，default 表示按默认格式返回，即数组。	可配置默认值
g		
h		
i		
j		
k	（必须）k=md5(t+APP_KEY)生成的字符串，用于验证 App Key，拦截非法请求。	
l	（可选）语言包，有 CN/EN，默认为 CN，中文。	可配置默认值
m		
n		
o		
p	（可选）客户端请求协议，有 rpc/soap/http，默认为 rpc。	可配置默认值
q		
r		
s		
t	（必须）时间戳，用于生成 k。	
u		
v	客户端平台版本标记	
w		
x		
y		
z	（部分必须）token 值，用于验证用户成功登录后的身份。	通过登录成功后，由服务返回，用于标志当前会话。

温馨提示：26 个小写字母（区分大小写），都是系统保留的参数。

4.3. 请求链接

服务器域名 + 系统参数 + [应用参数]

示例：
http:// localhost/zenWS3/index.php?c=User&a=login&t=1357826899&k=acf18657239b2b9dba910dd2629d52946&p=http&username=test&password=1234

正常情况下会返回如下数据：
{"status":"OK","data":"sFZWlWMoKTknnoktmLZwpJEH","error":"","debug":[]}

4.4. 返回数据

正常系统响应情况下，返回的数据有四大部分：

1) data: 客户所需要获取的数据。
2) error: 全部错误提示信息，包括系统错误信息和应用错误信息。
3) status: 系统响应的状态，见**表 返回状态类型说明表**。
4) debug：在开启调试状态下会有此部分数据，正式发布时不存在。

4.4.1. 数据 data

根据具体不同的的请求，设定对应的返回数据。也就是请求服务后返回的数据。

4.4.2. 错误信息 error

为字符串，拼接了全部的错误信息，包括系统和应用，以及客户所触发的错误提示信息。

4.4.3. 返回状态 status

分为应用层状态和系统层状态，有以下几个类型。

表 4-4-3-1 返回状态类型说明表

	类型	说明	示例
应 用 层	OK	表示正确返回数据，并正确获取数据。	正常情况下。
	FAIL	表示正确响应，但由于提交不成功导致	正常情况下。如登录失

状态		失败。	败。
系统层状态	WRONG	用户操作错误	如缺少必须的系统参数，或者 API 所需参数，或者调用不存在的服务等。
	ERROR	系统错误	如包含不存在的文件。
	UNKNOW	未知错误	开发中未定义的错误。

各种返回状态发生的阶段

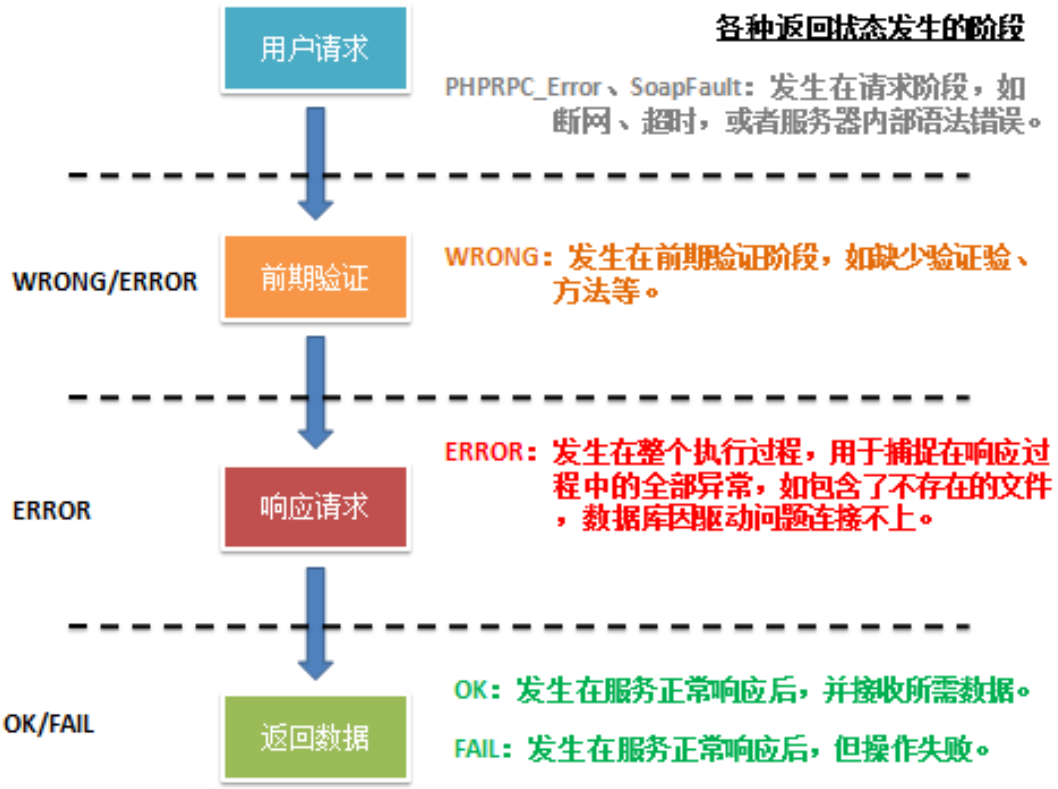


图 4-4-3-1 各种返回状态发生的阶段

4.4.4. 调试信息 debug

在开启调试的情况下，显示此部分数据，为一数组，以方便开发人员获取更多的相关辅助信息。由开发人员自动添加。

4.4.5. 返回数据示例

以请求默认服务为例，并假设服务已关闭 AppKey 验证与 Token 验证。

4.4.5.1. JSON 格式

请求链接为：http://localhost/zenws3/index.php?f=xml

{"status":"OK","data":"Welcome to use zenWS3!","error":"","debug":{"msg":"This is default service!"}}

4.4.5.2. XML 格式

请求链接为：http://localhost/zenws3/index.php?f=xml

```
<?xml version="1.0" encoding="utf-8" ?>
 <xmlData>
   <status>OK</status>
   <data>Welcome to use zenWS3!</data>
   <error />
   <debug>
     <msg>This is default service!</msg>
   </debug>
 </xmlData>
```

4.4.5.3. 数组格式

请求链接为：http://localhost/zenws3/index.php?f=array

Array ([status] => OK [data] => Welcome to use zenWS3! [error] => [debug] => Array ([msg] => This is default service!))

5. 附录

5.1. 参考资料

- 《Web Services 原理与研发实践》 顾宁，刘家茂，柴晓路等编著
- 《深入 PHP:面向对象、模式与实践》 (美) Matt Zandstra 著 陈浩 ... [等] 译
- 《高性能 PHP 应用开发》 (美) Armando Padilla, Tim Hawkins 著
- 《PHP 6 高级编程》 (美) Ed Lecky-Thompson, Steven D. Nowicki, Thomas Myer 著
- 《领域驱动设计与模式实战》 (瑞典) Jimmy Nilsson 著
- 《修改代码的艺术》 (美) Michael C. Feathers 著

- 《设计模式解析第 2 版》 (美) Alan Shalloway, James R. Trott 著