# shupan001的专栏

断桥残雪，平湖秋月，清风徐来，水波不兴





个人资料

shupan001

+ 加关注 发私信

访问： 195366次
积分： 2578
等级： BLOG 5
排名： 第9659名

原创： 51篇 转载： 92篇
译文： 0篇 评论： 18条

文章搜索

文章分类

android (5)
C (10)
linux (26)
php (19)
其他 (10)
冷笑话 (0)
数据库 (8)
网络 (8)
吉他 (1)
fe (1)
mysql (2)
儿童教育 (0)

文章存档

2013年11月 (1)
2013年09月 (1)
2013年04月 (1)
2013年03月 (2)
2012年02月 (1)

展开

阅读排行

## 转 探究php底层运行机制

标签： php apache variables 工作 extension 服务器

2012-01-04 19:57 4131人阅读 评论(0) 收藏 举报

分类： php（18）

本文转载自:http://www.myext.cn/Article/921.html

概要

简介

先看看下面这个过程:

我们从未手动开启过PHP的相关进程，它是随着Apache的启动而运行的;

PHP通过mod_php5.so模块和Apache相连（具体说来是SAPI，即服务器应用程序编程接口）;

PHP总共有三个模块：内核、Zend引擎、以及扩展层;

PHP内核用来处理请求、文件流、错误处理等相关操作;

Zend引擎（ZE）用以将源文件转换成机器语言，然后在虚拟机上运行它;

扩展层是一组函数、类库和流，PHP使用它们来执行一些特定的操作。比如，我们需要mysql扩展来连接MySQL数据库;

当ZE执行程序时可能会需要连接若干扩展，这时ZE将控制权交给扩展，等处理完特定任务后再返还;

最后，ZE将程序运行结果返回给PHP内核，它再将结果传送给SAPI层，最终输出到浏览器上。

深入探讨

等等，没有这么简单。以上过程只是个简略版，让我们再深入挖掘一下，看看幕后还发生了些什么。

Apache启动后，PHP解释程序也随之启动;

PHP的启动过程有两步;

第一步是初始化一些环境变量，这将在整个SAPI生命周期中发生作用;

第二步是生成只针对当前请求的一些变量设置。

PHP启动第一步

不清楚什么第一第二步是什么？别担心，我们接下来详细讨论一下。让我们先看看第一步，也是最主要的一步。要记住的是，第一步的操作在任何请求到达之前就发生了。

启动Apache后，PHP解释程序也随之启动;

PHP调用各个扩展的MINIT方法，从而使这些扩展切换到可用状态。看看php.ini文件里打开了哪些扩展吧;

MINIT的意思是“模块初始化”。各个模块都定义了一组函数、类库等用以处理其他请求。

一个典型的MINIT方法如下:

```
PHP_MINIT_FUNCTION(extension_name){
/* Initialize functions, classes etc */
```

一位十年软件工程师告诉 (13271)
解决SecureCRT中文编码 (11401)
php中0,'',null,false,true,F (8077)
wireshark抓http包时的过 (7666)
mysql编码设置 (7325)
Android SDK Platforms 找 (5716)
thrift安装记录 (4891)
启用Xdebug使用WinCac (4738)
lucene下载与安装 (4633)
XDebug 配置与使用,Win (4490)

评论排行

一位十年软件工程师告诉 (4)
解决SecureCRT中文编码 (3)
初来淘宝 (1)
解剖MFC自动生成的宏定 (1)
java数据库编程过程 (1)
lucene下载与安装 (1)
给C++初学者的50个忠告 (1)
wireshark抓http包时的过 (1)
thrift安装记录 (1)
史上最全的android开发资 (1)

推荐文章

*Android RocooFix 热修复框架
*笑谈Android图表------MPAndroidChart
*Nginx正反向代理、负载均衡等功能实现配置
* 浅析ZeroMQ工作原理及其特点
*Android开源框架Universal-Image-Loader基本介绍及使用
*Spring Boot 实践折腾记（三）：三板斧，Spring Boot下使用Mybatis

最新评论

给C++初学者的50个忠告
limumu1997: 现在正在学习C的我，看了下 然后对着天空大声说“我绝不会放弃”

一位十年软件工程师告诉你什么样
FLC720: 你好，我想问一个可能比较幼稚的问题，身边有些同学并没有把数据结构和算法学的很好，但是技术不错，照样可...

一位十年软件工程师告诉你什么样
独孤文彬: 学习计算机需要达到一种状态，就是在任何时候脑海里面都会在隐约的思考：对计算机里面某种原理的理解，或者...

启用Xdebug使用WinCacheGrind
空山: 为什么我的 self 大于 cum??

XDebug 配置与使用,WinCacheG
jayxhj: 从昨天搞到今天，目录没有建，配置是成功的，出错信息都是xdebug格式化的输出，在stackover...

lucene下载与安装
q3904175: select * from u

解决SecureCRT中文编码问题
w852821806: 的确是解决了，谢谢你们了，编程因为你们变得更好了，由衷的感谢你们

linux一条命令发邮件

}

PHP启动第二步

当一个页面请求发生时，SAPI层将控制权交给PHP层。于是PHP设置了用于回复本次请求所需的环境变量。同时，它还建立一个变量表，用来存放执行过程中产生的变量名和值。

PHP调用各个模块的RINIT方法，即“请求初始化”。一个经典的例子是Session模块的RINIT，如果在php.ini中启用了Session模块，那在调用该模块的RINIT时就会初始化$_SESSION变量，并将相关内容读入；

RINIT方法可以看作是一个准备过程，在程序执行之间就会自动启动。

一个典型的RINIT方法如下：

```
PHP_RINIT_FUNCTION(extension_name) {
/* Initialize session variables, pre-populate variables, redefine global variables etc */
}
```

PHP关闭第一步

如同PHP启动一样，PHP的关闭也分两步：

一旦页面执行完毕（无论是执行到了文件末尾还是用exit或die函数中止），PHP就会启动清理程序。它会按顺序调用各个模块的RSHUTDOWN方法。

RSHUTDOWN用以清除程序运行时产生的符号表，也就是对每个变量调用unset函数。

一个典型的RSHUTDOWN方法如下：

```
PHP_RSHUTDOWN_FUNCTION(extension_name) {
/* Do memory management, unset all variables used in the last PHP call etc */
}
```

PHP关闭第二步

最后，所有的请求都已处理完毕，SAPI也准备关闭了，PHP开始执行第二步：

PHP调用每个扩展的MSHUTDOWN方法，这是各个模块最后一次释放内存的机会。

一个典型的RSHUTDOWN方法如下：

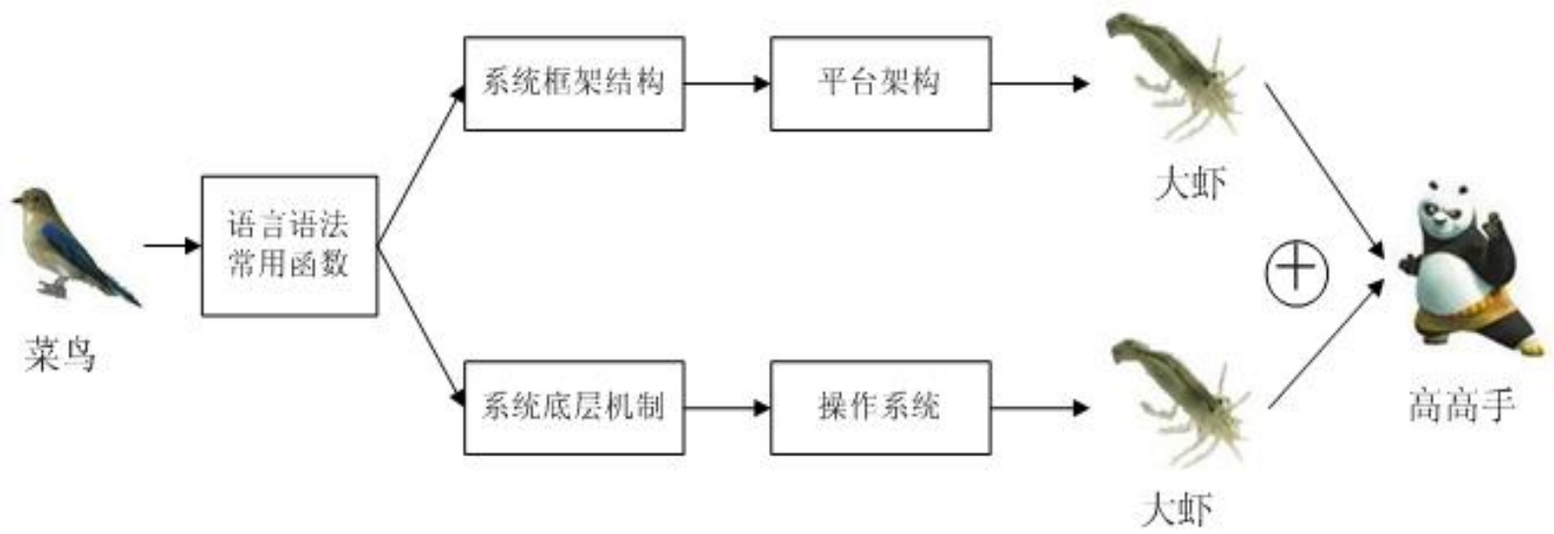
```
PHP_MSHUTDOWN_FUNCTION(extension_name) {
/* Free handlers and persistent memory etc */
}
```

这样，整个PHP生命周期就结束了。要注意的是，只有在服务器没有请求的情况下才会执行“启动第一步”和“关闭第二步”。

一、开篇

在开始这个专题之前，先说一点题外话。大多数人学习编程语言的时候，首先关注的是这种语言的语法及其常用函数。我学习C,Java,Php等语言就是按照这样的方式开始的。一般情况下，这个阶段需要一个月左右的时间就会完全掌握，并能基本熟练地使用。对于已有经验的同学，可能时间更短。其实各种语言的语法和常用函数都差别不大，有很多相通的地方。如果您在学习一种编程语言的时候，拿一些真正的项目任务作为实践，效果更佳，实践远胜于理论。

我们在掌握了一门编程语言之后，又会向两个方向发展：一个方向是向上延伸，从事系统框架结构的探索；另一方向是向下延伸，从事系统底层方面的研究，我大体画了一下这个学习演变过程的示意图。




注：虽然我的形象一直用着“高高手”，但我只是个菜鸟，如有雷同，纯属巧合，欢迎善意拍砖。

php的语法非常简单，正是它的简单性，使它成为了当前互联网第一编程语言。你不需要具备很多的知识就能上手，比如：你学习C语言，就必须非常了解各个变量如何定义，指针如何操作，内存如何创建销毁等等。再比如：你学习Java语言，就必须具有面向对象(OO)的基础,就必须清楚是什么时候需要封装，什么时候需要继承，什么时候需要多态，要做项目，怎么还得懂点SSH。Php的大部分使用者可能根本就没这么多讲究，有的人喜欢面向过程，那你就用面向过程的方式来写代码；有的人喜欢面向对象，那你就用面向对象的方式写代码。Php的产生缘于互联网，目前也是互联网Web2.0第一编程语言。满足用户需求永远是第一位的，可维护性暂且可以放在第二位。我们通常说Web应用永远是β版的，计划远没有变化快。

我们公司里有很多phper，我曾经问过他们：“php程序到底是如何被执行的？”，多数人似乎很难说得清楚。这种情况，其实并不奇怪，我曾经拿类似的问题问过Javaer，Javaer的回答也是如此。有的同学会问：“研究这样的问题有没有实际意义呢？”我说：“有！”。理解系统的底层，有助于你写出高效健壮的代码，你会更清楚程序的代码到底该怎么去写。另外，如果你有志去做php扩展，那就更不必说，责无旁贷。

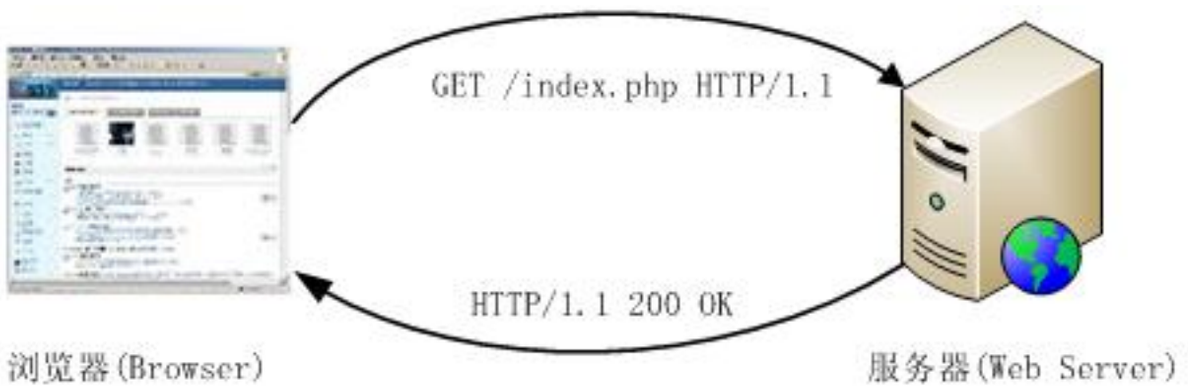
要回答以上问题，我觉得最好的办法是阅读一下php的源码，从“根”上解决。近来我找了点时间，粗读了一遍，愿意与各位共享。

关于php的底层工作原理，一定绕不开webserver，象apache，lighttpd，nginx，iis等。我这里就选择apache为例吧。以下内容将结合apache的源码、工作原理和扩展来逐步切入php的解析过程。

## 二、Apache运行机制剖析

### I　　B/S交互过程

浏览器(Browser)和服务器(Web Server)的交互过程：




1、 浏览器向服务器发出HTTP请求(Request)。

2、 服务器收到浏览器的请求数据，经过分析处理，向浏览器输出响应数据（Response）。

3、 浏览器收到服务器的响应数据，经过分析处理，将最终结果显示在浏览器中。

下图是一份浏览器请求数据和服务器响应数据的快照：


关于浏览器和服务器数据交互过程非常简单，很容易理解。我想从事Web开发的人员都很清楚，在此不再赘述，仅供参考。

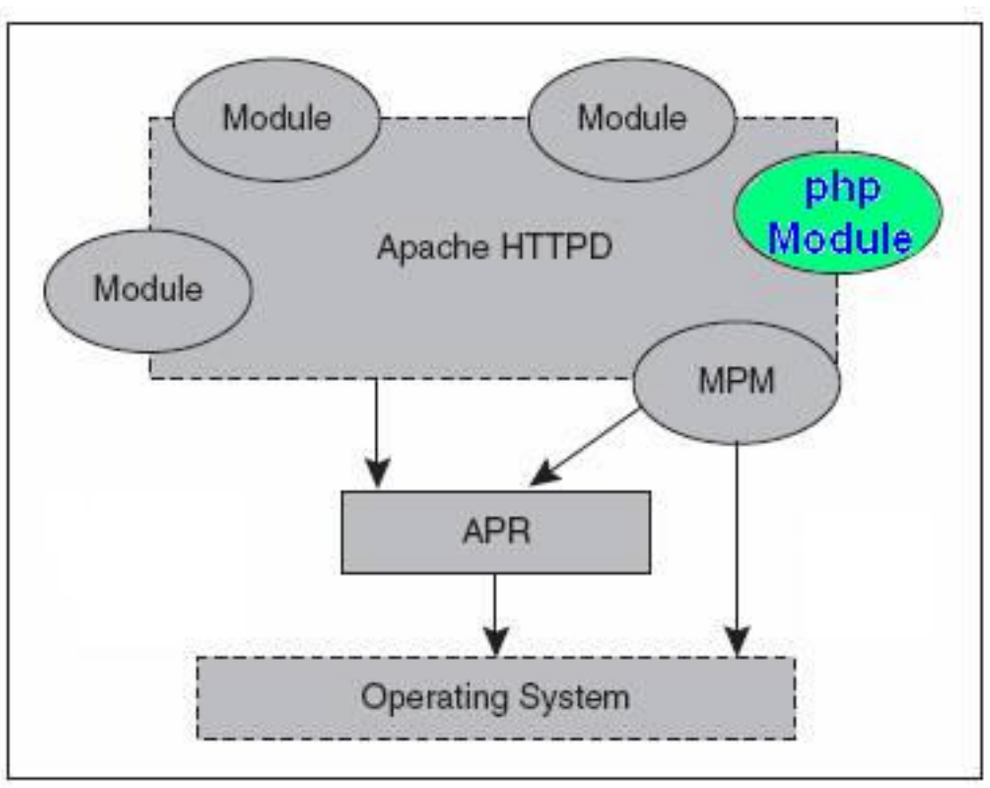
## l Apache概述

Apache是目前世界上使用最为广泛的一种Web Server，它以跨平台、高效和稳定而闻名。按照去年官方统计的数据，Apache服务器的装机量占该市场60%以上的份额。尤其是在X（Unix/Linux）平台上，Apache是最常见的选择。其它的Web Server产品，比如IIS，只能运行在Windows平台上，是基于微软.Net架构技术的不二选择。

Apache并不是没有缺点，它最为诟病的一点就是变得越来越重，被普遍认为是重量级的WebServer。所以，近年来又涌现出了很多轻量级的替代产品，比如lighttpd,nginx等等，这些WebServer的优点是运行效率很高，但缺点也很明显，成熟度往往要低于Apache，通常只能用于某些特定场合，。

## l Apache组件逻辑图

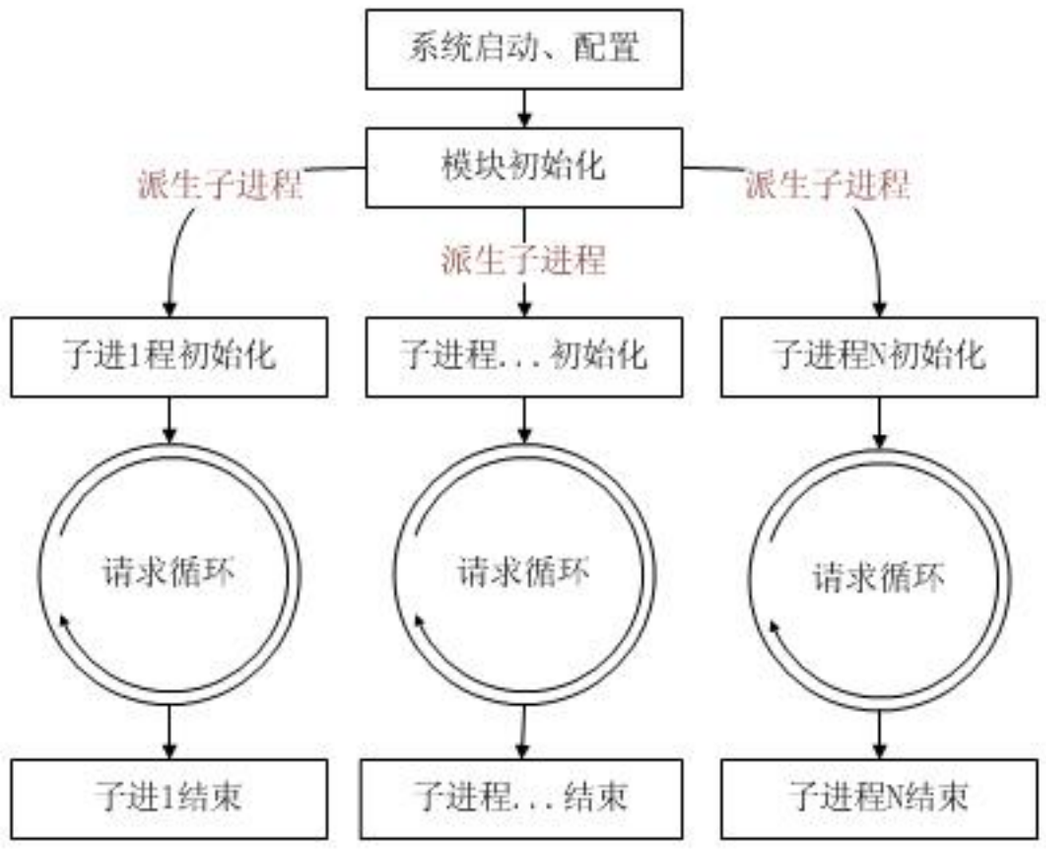
Apache是基于模块化设计的，总体上看起来代码的可读性高于php的代码，它的核心代码并不多，大多数的功能都被分散到各个模块中，各个模块在系统启动的时候按需载入。你如果想要阅读Apache的源代码，建议你直接从main.c文件读起，系统最主要的处理逻辑都包含在里面。MPM（Multi -Processing Modules，多重处理模块）是Apache的核心组件之一，Apache通过MPM来使用操作系统的资源，对进程和线程池进行管理。Apache为了能够获得最好的运行性能，针对不同的平台(Unix/Linux、Window)做了优化，为不同的平台提供了不同的MPM，用户可以根据实际情况进行选择，其中最常使用的MPM有prefork和worker两种。至于您的服务器正以哪种方式运行，取决于安装Apache过程中指定的MPM编译参数,在X系统上默认的编译参数为prefork。由于大多数的Unix都不支持真正的线程，所以采用了预派生子进程(prefork)方式，象Windows或者Solaris这些支持线程的平台，基于多进程多线程混合的worker模式是一种不错的选择。对此感兴趣的同学可以阅读有关资料，此处不再多讲。Apache中还有一个重要的组件就是APR（Apache portable Runtime Library），即Apache可移植运行库，它是一个对操作系统调用的抽象库，用来实现Apache内部组件对操作系统的使用，提高系统的可移植性。Apache对于php的解析，就是通过众多Module中的php Module来完成的。




Apache的逻辑构成以及与操作系统的关系

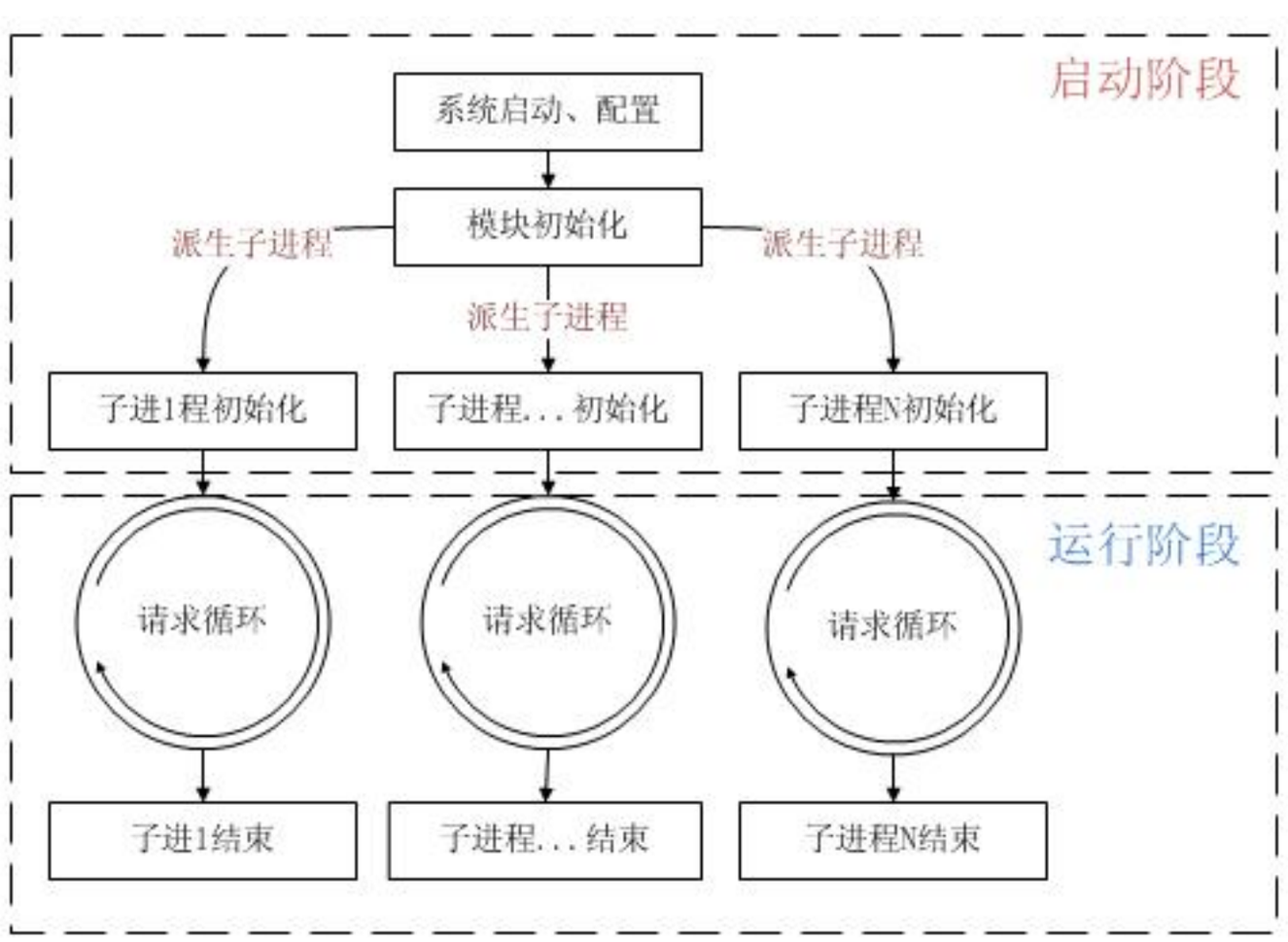
## l Apache的生命周期

这一节的内容会与php模块的载入有关，您可以略微关注一下。以下是Apache的生命周期（prefork模式）示意图。




l Apache的生命周期

这一节的内容将会阐述php模块的载入过程，请参考Apache的生命周期示意图（prefork模式下）。




Apache的运行分为启动阶段和运行阶段。

1. 启动阶段

在启动阶段，Apache主要进行配置文件解析(例如http.conf以及Include指令设定的配置文件等)、模块加载(例如mod_php.so,mod_perl.so等)和系统资源初始化（例如日志文件、共享内存段等）工作。

在这个阶段，Apache为了获得系统资源最大的使用权限，将以特权用户root（X系统）或超级管理员administrator(Windows系统)完成启动。

Apache和“php处理机”的装配过程就是在这个阶段完成的。

“php处理机”就是负责解释和执行你的php代码的系统模块。这个名字是我特意创造的，目的是为了帮助你理解

本节的内容，后面的章节还会给出更专业的名称。

你单独做过php的安装配置吗?

如果你做过类似的工作，下面的内容很容易理解；如果你没有做过，可以尝试安装一下，有助于加深你的理解。不过，我的文章向来深入浅出，我会尽量把这个过程讲得更浅显一些。其实php的安装非常简单，如果你很感兴趣的话，可以到网上随便搜一篇安装指南，按步骤照做就可以了。

把php最终集成到Apache系统中，还需要对Apache进行一些必要的设置。这里，我们就以php的mod_php5 SAPI运行模式为例进行讲解，至于SAPI这个概念后面我们还会详细讲解。

假定我们安装的版本是Apache2 和 Php5，那么需要编辑Apache的主配置文件http.conf，在其中加入下面的几行内容:

Unix/Linux环境下:

LoadModule php5_module modules/mod_php5.so

AddType application/x-httpd-php .php

注：其中modules/mod_php5.so 是X系统环境下mod_php5.so文件的安装位置。

Windows环境下:

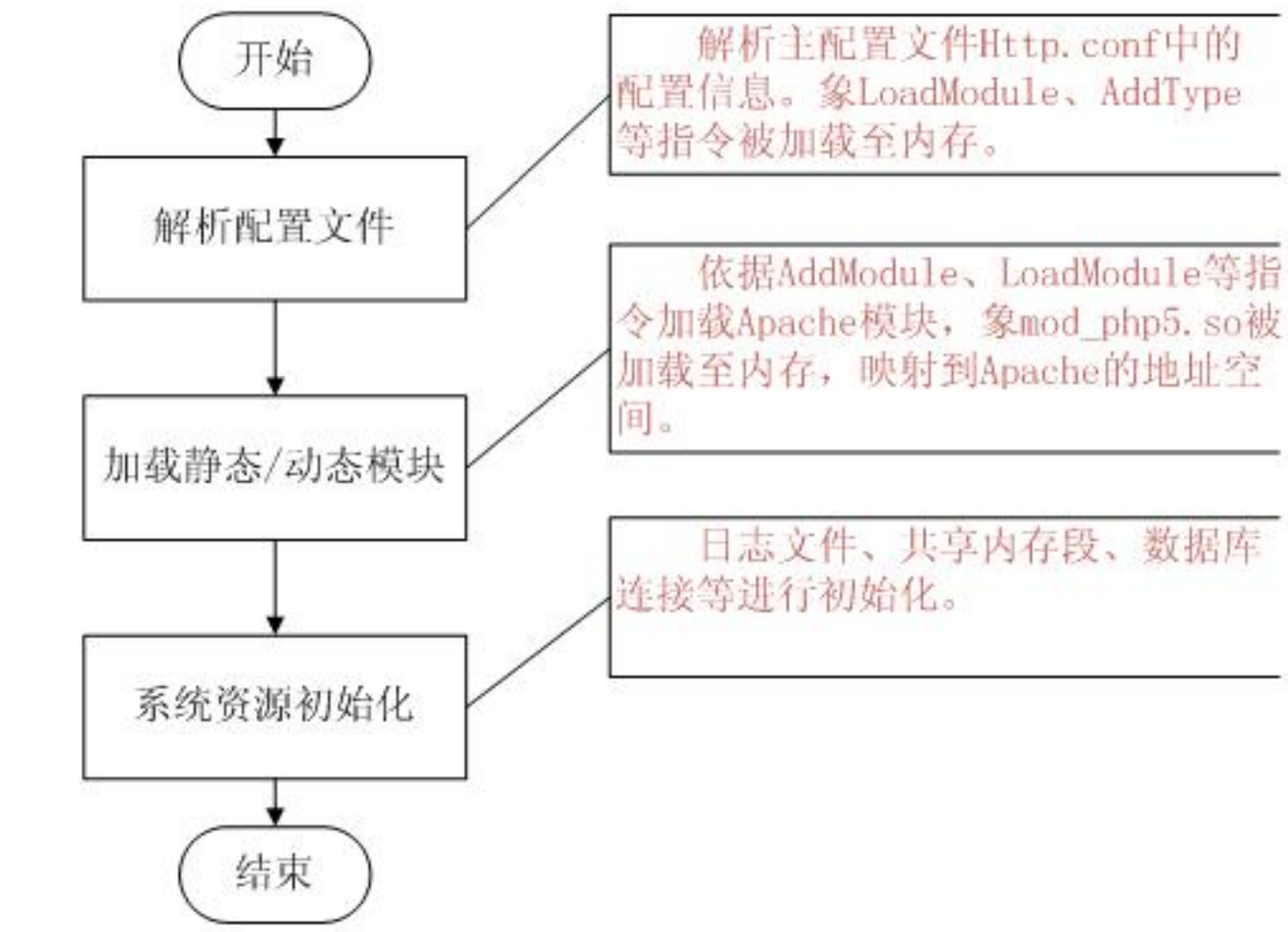
LoadModule php5_module d:/php/php5apache2.dll

AddType application/x-httpd-php .php

注：其中d:/php/php5apache2.dll 是在Windows环境下php5apache2.dll文件的安装位置。

这两项配置就是告诉Apache Server，以后收到的Url用户请求，凡是以php作为后缀，就需要调用php5_module模块（mod_php5.so/ php5apache2.dll）进行处理。

这个过程可以参考以下的示意图:




Apache启动阶段的源码包含在server/main.c中，我整理了一下源码中的对应关系:

```
.....................................
.....................................
//读取apache配置文件，进行分析
    ap_server_root = def_server_root;
    if (temp_error_log) {
        ap_replace_stderr_log(process->pool, temp_error_log);
    }
    server_conf = ap_read_config(process, ptemp, confname, &ap_conftree);
    if (!server_conf) {
        destroy_and_exit_process(process, 1);
    }
.....................................
.....................................
//改动配置文件后，信号通知apache重新加载
    signal_server = APR_RETRIEVE_OPTIONAL_FN(ap_signal_server);
    if (signal_server) {
        int exit_status;

        if (signal_server(&exit_status, pconf) != 0) {
            destroy_and_exit_process(process, exit_status);
        }

    for (;;) {
        apr_hook_deregister_all();
        apr_pool_clear(pconf);
.....................................
.....................................
//加载动/静态模块，注册各模块的hook handler

        for (mod = ap_prelinked_modules; *mod != NULL; mod++) {
            ap_register_hooks(*mod, pconf);
        }

.....................................
.....................................
//MPM管理，系统资源初始化管理

        if (ap_mpm_run(pconf, plog, server_conf))
            break;

        apr_pool_lock(pconf, 0);
    }
.....................................
.....................................
```

不熟悉unix/linux的同学可能会问so文件（mod_php5.so)是个什么样的文件?

unix/linux下,so后缀文件是一个DSO文件，DSO与windows系统下的dll是等价概念，都是把一堆函数封装在一个二进制文件中。调用它们的进程把它们装入内存后，会将其映射到自己的地址空间。

DSO全称为Dynamic Shared Object，即动态共享对象。DLL全称为Dynamic Link Library 即动态链接库。

Apache 服务器的体系结构的最大特点，就是高度模块化。如果你为了追求处理效率，可以把这些dso模块在apache编译的时候静态链入，这样会提高Apache 5%左右的处理性能。

2.　　运行阶段

在运行阶段，Apache主要工作是处理用户的服务请求。

在这个阶段，Apache放弃特权用户级别，使用普通权限，这主要是基于安全性的考虑，防止由于代码的缺陷引起的安全漏洞，象微软的IIS就曾遭受“红色代码（Code Red）”和“尼姆达（Nimda）”等恶意代码的溢出攻击。

2、运行阶段

2.1 运行阶段概述

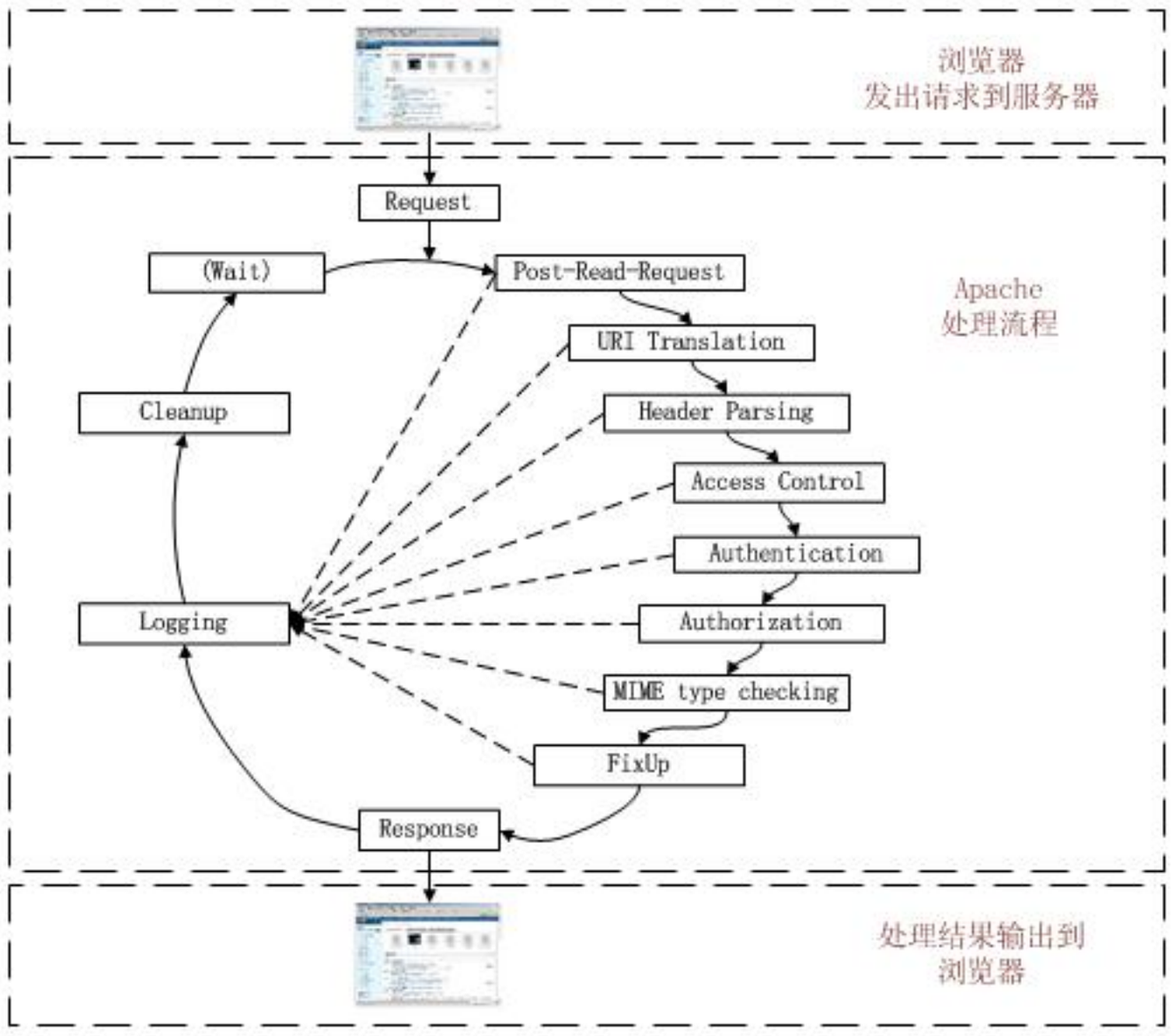
在运行阶段，Apache主要工作是处理用户的服务请求。

在这个阶段，Apache放弃特权用户级别，使用普通权限，这主要是基于安全性的考虑，防止由于代码的缺陷引起的安全漏洞。象微软的IIS就曾遭受“红色代码（Code Red）”和“尼姆达（Nimda）”等恶意代码的溢出攻击。

? 2.2 运行阶段流程

Apache将请求处理循环分为11个阶段，依次是：Post-Read-Request，URI Translation，Header Parsing，Access Control，Authentication，Authorization，MIME Type Checking，FixUp，Response，Logging，

CleanUp。




Apache Hook机制

Apache的Hook机制是指：Apache 允许模块(包括内部模块和外部模块，例如mod_php5.so,mod_perl.so等)将自定义的函数注入到请求处理循环中。换句话说，模块可以在Apache的任何一个处理阶段中挂接(Hook)上自己的处理函数，从而参与Apache的请求处理过程。

mod_php5.so/ php5apache2.dll就是将所包含的自定义函数，通过Hook机制注入到Apache中，在Apache处理流程的各个阶段负责处理php请求。

关于Hook机制在Windows系统开发也经常遇到，在Windows开发既有系统级的钩子，又有应用级的钩子。常见的翻译软件（例如金山词霸等等）的屏幕取词功能，大多数是通过安装系统级钩子函数完成的，将自定义函数替换gdi32.dll中的屏幕输出的绘制函数。

Apache请求处理循环详解

Apache请求处理循环的11个阶段都做了哪些事情呢?

1、Post-Read-Request阶段

在正常请求处理流程中，这是模块可以插入钩子的第一个阶段。对于那些想很早进入处理请求的模块来说，这个阶段可以被利用。

2、URI Translation阶段

Apache在本阶段的主要工作：将请求的URL映射到本地文件系统。模块可以在这阶段插入钩子，执行自己的映射逻辑。mod_alias就是利用这个阶段工作的。

3、Header Parsing阶段

Apache在本阶段的主要工作：检查请求的头部。由于模块可以在请求处理流程的任何一个点上执行检查请求头部的任务，因此这个钩子很少被使用。mod_setenvif就是利用这个阶段工作的。

4、Access Control阶段

Apache在本阶段的主要工作：根据配置文件检查是否允许访问请求的资源。Apache的标准逻辑实现了允许和拒

绝指令。mod_authz_host就是利用这个阶段工作的。

5、Authentication阶段

Apache在本阶段的主要工作：按照配置文件设定的策略对用户进行认证，并设定用户名区域。模块可以在这阶段插入钩子，实现一个认证方法。

6、Authorization阶段

Apache在本阶段的主要工作：根据配置文件检查是否允许认证过的用户执行请求的操作。模块可以在这阶段插入钩子，实现一个用户权限管理的方法。

7、MIME Type Checking阶段

Apache在本阶段的主要工作：根据请求资源的MIME类型的相关规则，判定将要使用的内容处理函数。标准模块mod_negotiation和mod_mime实现了这个钩子。

8、FixUp阶段

这是一个通用的阶段，允许模块在内容生成器之前，运行任何必要的处理流程。和Post_Read_Request类似，这是一个能够捕获任何信息的钩子，也是最常使用的钩子。

9、Response阶段

Apache在本阶段的主要工作：生成返回客户端的内容，负责给客户端发送一个恰当的回复。这个阶段是整个处理流程的核心部分。

10、Logging阶段

Apache在本阶段的主要工作：在回复已经发送给客户端之后记录事务。模块可能修改或者替换Apache的标准日志记录。

11、CleanUp阶段

Apache在本阶段的主要工作：清理本次请求事务处理完成之后遗留的环境，比如文件、目录的处理或者Socket的关闭等等，这是Apache一次请求处理的最后一个阶段。

模块的注入Apache的过程可以参考源码中server/core.c文件：

```
static void register_hooks(apr_pool_t *p)
{
    /* create_connection and install_transport_filters are
     * hooks that should always be APR_HOOK_REALLY_LAST to give other
     * modules the opportunity to install alternate network transports
     * and stop other functions from being run.
     */
    ap_hook_create_connection(core_create_conn, NULL, NULL,
                              APR_HOOK_REALLY_LAST);
    ap_hook_pre_connection(core_pre_connection, NULL, NULL,
                           APR_HOOK_REALLY_LAST);

    ap_hook_post_config(core_post_config,NULL,NULL,APR_HOOK_REALLY_FIRST);
    ap_hook_translate_name(ap_core_translate,NULL,NULL,APR_HOOK_REALLY_LAST);
    ap_hook_map_to_storage(core_map_to_storage,NULL,NULL,APR_HOOK_REALLY_LAST);
    ap_hook_open_logs(ap_open_logs,NULL,NULL,APR_HOOK_REALLY_FIRST);
    ap_hook_child_init(ap_logs_child_init,NULL,NULL,APR_HOOK_MIDDLE);
    ap_hook_handler(default_handler,NULL,NULL,APR_HOOK_REALLY_LAST);
/* FIXME: I suspect we can eliminate the need for these do_nothings - Ben */
    ap_hook_type_checker(do_nothing,NULL,NULL,APR_HOOK_REALLY_LAST);
    ap_hook_fixups(core_override_type,NULL,NULL,APR_HOOK_REALLY_FIRST);
    ap_hook_access_checker(do_nothing,NULL,NULL,APR_HOOK_REALLY_LAST);
    ap_hook_create_request(core_create_req, NULL, NULL, APR_HOOK_MIDDLE);
    APR_OPTIONAL_HOOK(proxy, create_req, core_create_proxy_req, NULL, NULL,
                      APR_HOOK_MIDDLE);
    ap_hook_pre_mpm(ap_create_scoreboard, NULL, NULL, APR_HOOK_MIDDLE);
```

mod_php5.so/ php5apache2.dll注入到Apache的函数中，最重要的就是Response阶段的处理函数。

上一篇 mysql与事务
下一篇 PHP SAPI---CLI

## 我的同类文章

### php（18）

- 五种常见的 php设计模式 2012-01-19 阅读 344
- PHP SAPI---CLI 2012-01-04 阅读 2646
- XDebug 配置与使用,WinCa… 2011-12-29 阅读 4488
- smarty语法 2011-10-25 阅读 340
- PHP的ob_start();用法 2011-10-11 阅读 252
- 什么是CGI、FastCGI、PHP… 2012-01-05 阅读 3826
- 启用Xdebug使用WinCache… 2011-12-29 阅读 4729
- php中0,'',null,false,true,FLA… 2011-12-23 阅读 8077
- linux下php实现C/C++扩展… 2011-10-11 阅读 265
- php中的SERVER变量 2011-10-10 阅读 315

更多文章

## 猜你在找



### 查看评论

暂无评论

### 发表评论

用 户 名: sinat_20684939

评论内容:

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

### 核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity

Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC
coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
Angular Cloud Foundry Redis Scala Django Bootstrap



公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved