第一章 TP6 快速入门

1.1 框架简介

框架的作用是解决代码重用以及快速开发的问题。

1.1.1 什么是PHP框架：

PHP框架就是一种可以在项目开发过程中，提高开发效率，创建更为稳定的程序，并减少开发者重复编写代码的基础架构。PHP框架是将不同Web系统开发过程中的共性、通用部分功能进行抽象，形成开发Web程序的基本架构。进行Web系统开发时，开发人员如果在PHP框架基础上进行二次开发，即可大大简化开发过程，快速实现系统功能。PHP框架能促进Web系统的快速开发、节约时间、减少重复代码量，并能帮助初学者创建规范、稳定的Web系统。

1.1.2 PHP框架有哪些：

1. Laravel——巨匠级PHP开发框架

Laravel是一个简单优雅的PHP Web开发框架，可以将开发者从意大利面条式的代码中解放出来，通过简单、高雅、表达式语法开发出很棒的Web应用，Laravel拥有更富有表现力的语法、高质量的文档、丰富的扩展包，被称为“巨匠级PHP开发框架”。

2. Phalcon——最快的PHP框架

Phalcon是一个开源的、全栈的、用C语言编写的PHP5框架，为开发者提供了网站及应用开发所需的大量高级工具，且Phalcon是松耦合的，开发者可以根据需要使用其他组件。Phalcon中的所有函数都以PHP类的方式呈现，开发者无需学习和使用C语言，且无需担心性能问题。（备注：类似的框架还有PHP大神鸟哥的 yaf 框架）

3. Symfony2——开发速度和性能的结合体

Symfony2是一个开源的PHP Web框架，有着开发速度快、性能高等特点。与其他框架相比，Symfony2的优势包括：支持DI（依赖注入）和IoC（控制反转）；扩展性强；文档和社区比较成熟。但是Symfony2的学习曲线也比较陡峭，没有经验的初学者往往需要一些练习才能掌握其特性。

4. ZendFrameWork---最官方的框架

Zend Framework (ZF)是Zend公司推出的一套PHP开发框架。是用 PHP 5 来开发 web程序和服务的开源框架。ZF 用 100% 面向对象编码实现。 ZF 的组件结构独一无二，每个组件几乎不依靠其他组件。这样的松耦合结构可以让开发者独立使用组件。 我们常称此为 “use-at-will”设计。

1.1.3 ThinkPHP 框架:

ThinkPHP 是一款国人开发的非常优秀的框架。

ThinkPHP是为了简化企业级应用开发和敏捷WEB应用开发而诞生的。最早诞生于2006年初，2007年元旦正式更名为ThinkPHP，并且遵循Apache2开源协议发布。ThinkPHP从诞生以来一直秉承简洁实用的设计原则，在保持出色的性能和至简的代码的同时，也注重易用性。并且拥有众多原创功能和特性，在社区团队的积极参与下，在易用性、扩展性和性能方面不断优化和改进。

ThinkPHP是一个快速、兼容而且简单的轻量级国产PHP开发框架，诞生于2006年初，原名FCS，2007年元旦正式更名为ThinkPHP，遵循Apache2开源协议发布，从Struts结构移植过来并做了改进和完善，同时也借鉴了国外很多优秀的框架和模式，使用面向对象的开发结构和MVC模式，融合了Struts的思想和TagLib（标签库）、RoR的ORM映射和ActiveRecord模式。

ThinkPHP可以支持windows/Unix/Linux等服务器环境，正式版需要PHP5.0以上版本支持，支持MySql、PgSQL、Sqlite多种数据库以及PDO扩展，ThinkPHP框架本身没有什么特别模块要求，具体的应用系统运行环境要求视开发所涉及的模块。

作为一个整体开发解决方案，ThinkPHP能够解决应用开发中的大多数需要，因为其自身包含了底层架构、兼容处理、基类库、数据库访问层、模板引擎、缓存机制、插件机制、角色认证、表单处理等常用的组件，并且对于跨版本、跨平台和跨数据库移植都比较方便。并且每个组件都是精心设计和完善的，应用开发过程仅仅需要关注您的业务逻辑。

目前，ThinkPHP 最新的版本是 Thinkphp6.0RC4。

1.2 框架下载以及部署

1.2.1 安装composer:

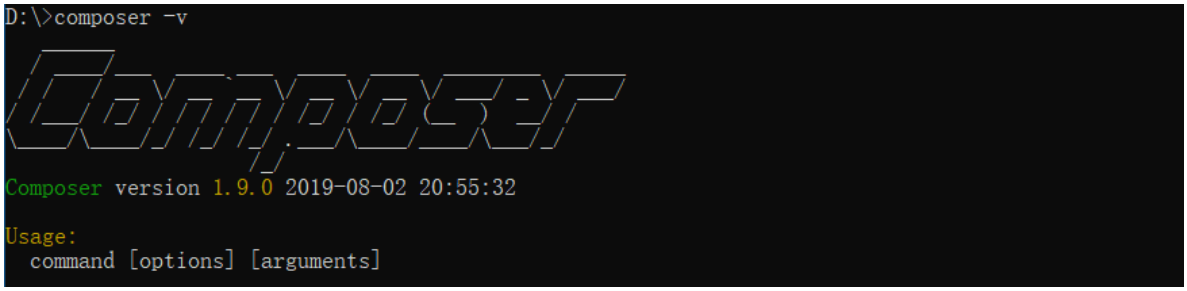
在 Windows 中，你需要下载并运行 Composer-Setup.exe。

link: https://getcomposer.org/download/

如果遇到任何问题或者想更深入地学习 Composer，请参考Composer 文档（英文文档，中文文档）。

link:https://docs.phpcomposer.com/

安装完之后，在命令行运行 composer -v 命令，如果出现如下界面，那恭喜你 composer 安装成功。



1.2.2 安装thinkPHP:

一般情况下，`composer` 安装的是最新的稳定版本，不一定是最新版本，如果你需要安装实时更新的版本（适合学习过程），可以安装`6.0.x-dev`版本。

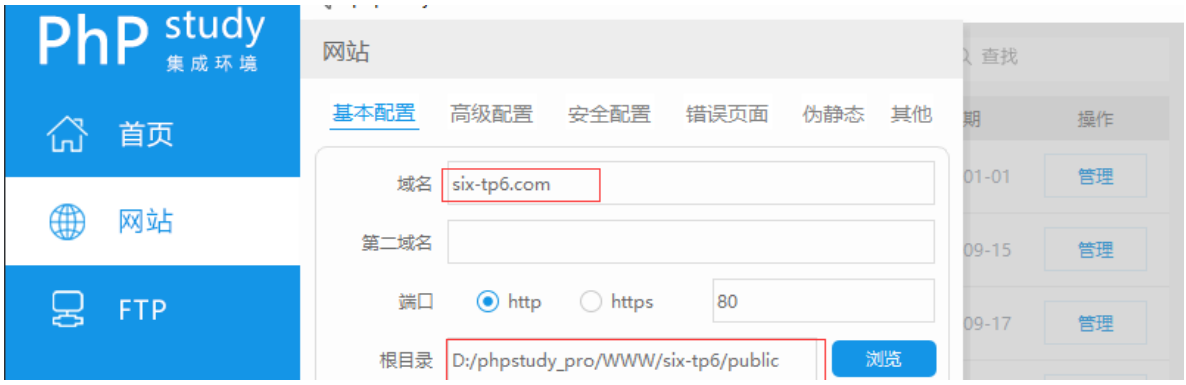
```
composer create-project topthink/think=6.0.x-dev tp
```

ThinkPHP下载需要一定的时间，大家耐心等待一下。。。。下载到最后，出现如下代码意味着ThinkPHP下载成功。

1.2.3 配置域名：

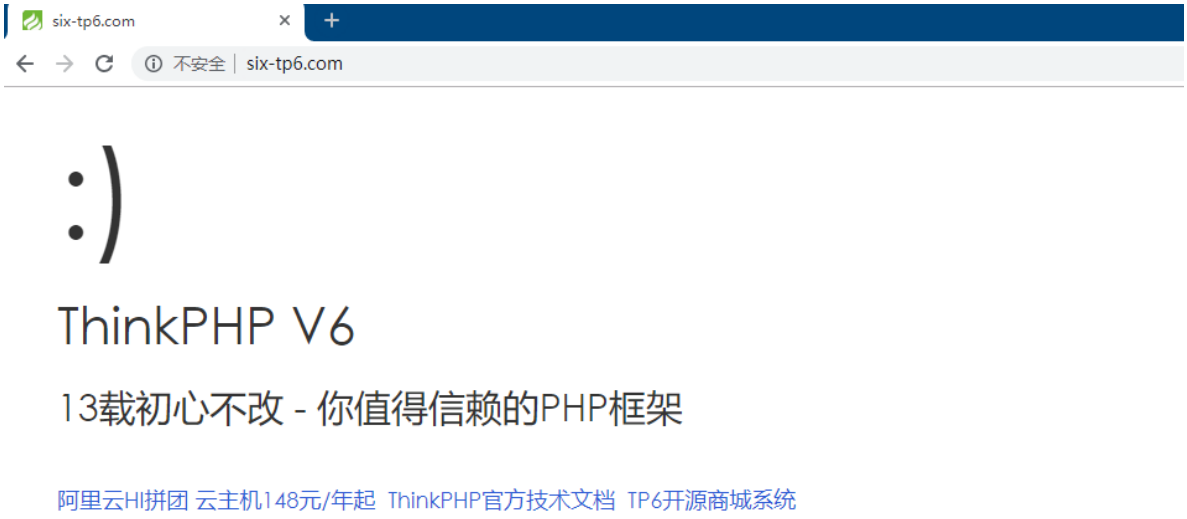
如果是手动搭建的PHP开发环境，配置域名比较复杂。但是，使用集成开发环境，配置域名就变得异常简单，这里使用phpstudy 且是最新版。

操作流程：启动PHPstudy---》选择 “网站” ---》选择 “创建网站”，然后做如下配置；



最后，点击 “确认”。一个域名就配置成功了。

接下来，我们可以在浏览器地址栏输入 six-tp6.com 按回车就会出现如下界面。



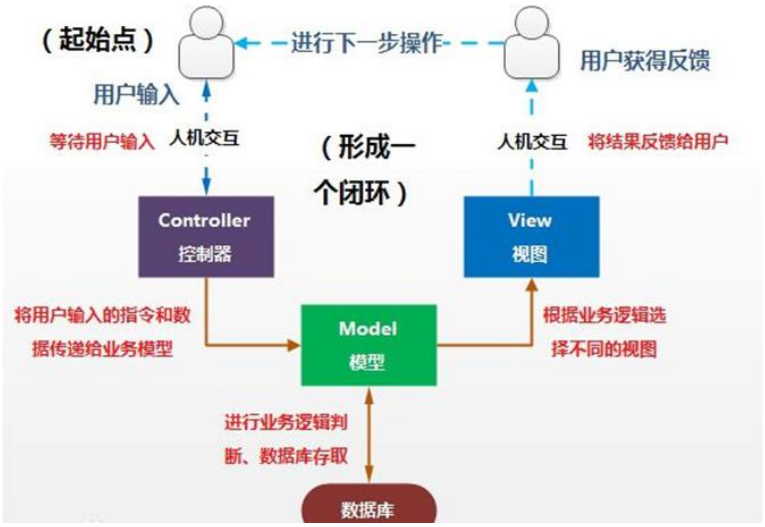
1.3 MVC 模式介绍

MVC要实现的目标是将软件用户界面和业务逻辑分离以使代码可扩展性、可复用性、可维护性、灵活性加强。

1.3.1 mvc 简介

> M：即 model 模型，主要用来实现业务逻辑，在MVC的三个部件中，模型拥有最多的处理任务。被模型返回的数据是中立的，模型与数据格式无关，这样一个模型能为多个视图提供数据，由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。
>
> V：即 view 视图，用户看到并与之交互的界面，比如由html元素组成的网页界面，或者软件的客户端界面。MVC的好处之一在于它能为应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，它只是作为一种输出数据并允许用户操纵的方式。
>
> C：即 controller 控制器，控制器接受用户的输入并调用模型和视图去完成用户的需求，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。

1.3.2 图解mvc:



用户首先在界面中进行人机交互，然后请求发送到控制器，控制器根据请求类型和请求的指令发送到相应的模型，模型可以与数据库进行交互，进行增删改查操作，完成之后，根据业务的逻辑选择相应的视图进行显示，此时用户获得此次交互的反馈信息，用户可以进行下一步交互，如此循环。

1.4 目录介绍和命名规范

1.4.1 框架目录详解:

单应用模式:

```
www  WEB部署目录（或者子目录）
├─app            应用目录
|  ├─controller      控制器目录
|  ├─model           模型目录
|  ├─ ...            更多类库目录
|  |
|  ├─common.php          公共函数文件
|  └─event.php           事件定义文件
|
├─config                 配置目录
|  ├─app.php             应用配置
|  ├─cache.php           缓存配置
|  ├─console.php         控制台配置
|  ├─cookie.php          Cookie配置
|  ├─database.php        数据库配置
|  ├─filesystem.php      文件磁盘配置
|  ├─lang.php            多语言配置
|  ├─log.php             日志配置
|  ├─middleware.php      中间件配置
|  ├─route.php           URL和路由配置
|  ├─session.php         Session配置
|  ├─trace.php           Trace配置
|  └─view.php            视图配置
|
├─view             视图目录
├─route                  路由定义目录
|  ├─route.php           路由定义文件
|  └─ ...
|
├─public                 WEB目录（对外访问目录）
|  ├─index.php           入口文件
|  ├─router.php          快速测试文件
```

```
|   └.htaccess           用于apache的重写
|
├─extend                 扩展类库目录
├─runtime                应用的运行时目录（可写，可定制）
├─vendor                 Composer类库目录
├─.example.env           环境变量示例文件
├─build.example.php      自动生成定义文件（参考）
├─composer.json          composer 定义文件
├─LICENSE.txt            授权说明文件
├─README.md              README 文件
├─think                  命令行入口文件
```

多应用模式：

```
www  WEB部署目录（或者子目录）
├─app            应用目录
|  ├─app_name            应用目录
|  |  ├─common.php       函数文件
|  |  ├─controller       控制器目录
|  |  ├─model            模型目录
|  |  ├─view             视图目录
|  |  └─ ...             更多类库目录
|  |
|  ├─common.php          公共函数文件
|  └─event.php           事件定义文件
|
├─config                 应用配置目录
|  ├─app_name            应用配置目录
|  |  ├─database.php     数据库配置
|  |  ├─cache            缓存配置
|  |  └─ ...
|  |
|  ├─app.php             应用配置
|  ├─cache.php           缓存配置
|  ├─console.php         控制台配置
|  ├─cookie.php          Cookie配置
|  ├─database.php        数据库配置
|  ├─filesystem.php      文件磁盘配置
|  ├─lang.php            多语言配置
|  ├─log.php             日志配置
|  ├─middleware.php      中间件配置
|  ├─route.php           URL和路由配置
|  ├─session.php         Session配置
|  ├─trace.php           Trace配置
|  └─view.php            视图配置
|
├─view                  视图目录
|  ├─app_name            应用视图目录
|  └─ ...
|
├─route                  路由定义目录
|  ├─app_name            应用路由目录
|  |  ├─route.php        路由定义文件
|  |  └─ ...
|
├─public                 WEB目录（对外访问目录）
|  ├─index.php           入口文件
```

```
|   ├─router.php          快速测试文件
|   └─.htaccess           用于apache的重写
|
├─extend                  扩展类库目录
├─runtime                 应用的运行时目录（可写，可定制）
├─vendor                  Composer类库目录
├─.example.env            环境变量示例文件
├─build.example.php       自动生成定义文件（参考）
├─composer.json           composer 定义文件
├─LICENSE.txt             授权说明文件
├─README.md               README 文件
├─think                   命令行入口文件
```

核心目录详解：

app: 应用目录，我们所写的业务逻辑代码都放置在这个目录下，此目录除此之外还包括基础控制类(BaseController.php)、应用异常定义文件(ExceptionHandle.php)、全局公共函数文件(common.php)等等文件。

config: 配置目录，框架所有的配置都放置在这个目录下，配置按照功能定义在不同的配置文件中，文件名即表示配置的功能。此目录包含如下文件，应用配置(app.php)、缓存配置(cache.php)、数据库配置(database.php)、日志配置(log.php)等等。

view: 视图目录，这就是MVC的视图层，存放的是应用的所有的模板(html文件)。

route: 路由目录，在这个目录下系统默认创建 app.php,我们可以把路由规则定义在这个目录下

vendor: 第三方类库目录，框架的核心的代码就放置在这个目录下，另外我们通过composer 安装的第三方包也是放置在这个目录下。

1.4.2 命令规范：

国有国法，家有家规，框架也有自己的规范，遵循这些规范可以避免一些意想不到的错误，也有利于团队之间的相互协作。

目录和文件

.目录使用小写+下划线；
.类库、函数文件统一以.php为后缀；
.类的文件名均以命名空间定义，并且命名空间的路径和类库文件所在路径一致；
.类（包含接口和Trait）文件采用驼峰法命名（首字母大写），其它文件采用小写+下划线命名；
.类名（包括接口和Trait）和文件名保持一致，统一采用驼峰法命名（首字母大写）；

函数和类、属性命名

.类的命名采用驼峰法（首字母大写），例如 User、UserType；
.函数的命名使用小写字母和下划线（小写字母开头）的方式，例如 get_client_ip；
.方法的命名使用驼峰法（首字母小写），例如 getUserName；
.属性的命名使用驼峰法（首字母小写），例如 tableName、instance；
.特例：以双下划线__打头的函数或方法作为魔术方法，例如 __call 和 __autoload

常量和配置

.常量以大写字母和下划线命名，例如 APP_PATH；
.配置参数以小写字母和下划线命名，例如 url_route_on 和url_convert；
.环境变量定义使用大写字母和下划线命名，例如APP_DEBUG；

数据表和字段

.数据表和字段采用小写加下划线方式命名，并注意字段名不要以下划线开头，例如 think_user 表和user_name字段，不建议使用驼峰和中文作为数据表及字段命名。

1.5 框架生命周期

1.5.1 入口文件：

Thinkphp 应用是采用单一入口模式部署，所有的请求都是请求同一个文件 ，也就是入口文件。这个文件里代码量虽然不多，却将整个应用的生命周期描述出来了。

首先，导入自动加载文件，Tp实现的是惰性加载，所有需要的类都是需要的时候才加载进来。

接着，实例化应用类(App.php),在这个过程中会获取应用目录等相关信息。

然后，运行应用，这是应用生命周期中最核心的一部分，像服务注册、路由检测、应用调度分发等等都是再这个阶段完成。

最后，将运行的结果响应输出，并将请求信息写入日志。

```
// public/index.php

// [ 应用入口文件 ]
namespace think;

require __DIR__ . '/../vendor/autoload.php';

// 执行HTTP应用并响应
$http = (new App())->http;

$response = $http->run();

$response->send();

$http->end($response);
```

1.5.2 运行应用：

这是，生命周期中最核心的部分。

```
/**
     * 执行应用程序
     * @param Request $request
     * @return mixed
     */
    protected function runWithRequest(Request $request)
    {
        $this->initialize();

        // 加载全局中间件
        $this->loadMiddleware();
```

```
        // 设置开启事件机制
        $this->app->event->withEvent($this->app->config->get('app.with_event',
true));

        // 监听HttpRun
        $this->app->event->trigger(HttpRun::class);

        return $this->app->middleware->pipeline()
            ->send($request)
            ->then(function ($request) {
                return $this->dispatchToRoute($request);
            });
    }
```

首先，应用的初始化，在这一阶段会加载环境变量文件 .env 、全局初始化文件、全局公共文件、系统助手函数、全局配置文件、全局事件定义和全局服务定义。

接下来，加载全局中间件，这步操作会把定义在 middleware.php的中间件保存到中间件队列里面。

最后，是路由检测，路由调度，实例化对应控制器，执行对应方法并将运行结果保存返回响应对象。

1.5.3 响应输出：

```
/**
     * 发送数据到客户端
     * @access public
     * @return void
     * @throws \InvalidArgumentException
     */
    public function send(): void
    {
        // 处理输出数据
        $data = $this->getContent();

        if (!headers_sent() && !empty($this->header)) {
            // 发送状态码
            http_response_code($this->code);
            // 发送头部信息
            foreach ($this->header as $name => $val) {
                header($name . (!is_null($val) ? ':' . $val : ''));
            }
        }

        $this->cookie->save();

        $this->sendData($data);

        if (function_exists('fastcgi_finish_request')) {
            // 提高页面响应
            fastcgi_finish_request();
        }
    }
```

首先，获取保存在响应对象里面的内容，接下来设置状态码和设置头部信息，最后将内容输出。

1.6 Tp-think 命令介绍与使用

Think 是 ThinkPHP 自带的命令行接口，他提供了许多使用的命令来帮助你构建 ThinkPHP 应用。

1.6.1 命令详解：

命令列表：

```
D:\phpstudy_pro\WWW\six-tp6>php think list  // 获取命令列表 这里是截取一部分
version 6.0.0RC4

Usage:
  command [options] [arguments]

Options:
  -h, --help            Display this help message
  -V, --version         Display this console version
  -q, --quiet           Do not output any message
      --ansi            Force ANSI output
      --no-ansi         Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
```

生成应用目录：

默认的框架的根目录下面自带了一个`build.example.php`示例参考文件（把该文件修改后改名为`build.php`放入`app`目录下面即可），内容如下：

```
return [
    // 需要自动创建的文件
    '__file__'   => [],
    // 需要自动创建的目录
    '__dir__'    => ['controller', 'model', 'view'],
    // 需要自动创建的控制器
    'controller' => ['Index'],
    // 需要自动创建的模型
    'model'      => ['User'],
    // 需要自动创建的模板
    'view'       => ['index/index'],
];
```

如果使用了多应用模式，可以快速生成一个应用，例如生成`demo`应用的指令如下：

```
php think build demo
```

创建类：

```
php think make:controller index@Blog    // 创建 index应用下的Blog控制器
php think make:model index@Blog         // 创建 index应用下的Blog模型
php think make:middleware Auth          // 生成中间件
php think make:validate index@User      // 生成验证器
```

优化：

```
// 数据表字段信息缓存
php think optimize:schema admin  // 生成admin应用的字段缓存
php think optimize:schema --db demo // 指定数据库
php think optimize:schema --table demo.think_user // 指定数据表

// 路由映射缓存
php think optimize:route index
```

1.6.2 自定义命令：

第一步，创建一个自定义命令类文件，运行指令

```
php think make:command Hello
```

会生成一个`app\command\Hello`命令行指令类，我们修改内容如下：

```
class Hello extends Command
{
    protected function configure()
    {
        // 指令配置
        $this->setName('hello')  // 这是指令的名称
             ->setDescription('the hello command');  // 这是功能的描述
    }

    protected function execute(Input $input, Output $output)
    {
        //业务逻辑代码 指令输出
        $output->writeln('hello');
    }
}
```

第二步，配置`config/console.php`文件

```
<?php
return [
    'commands' => [
        'hello' => 'app\command\Hello',
    ]
];
```

第三步，测试-命令帮助-命令行下运行

```
php think list
```

第四步，运行`hello`命令

```
 php think hello
```

1.7 配置phptorm配置xdebug调试

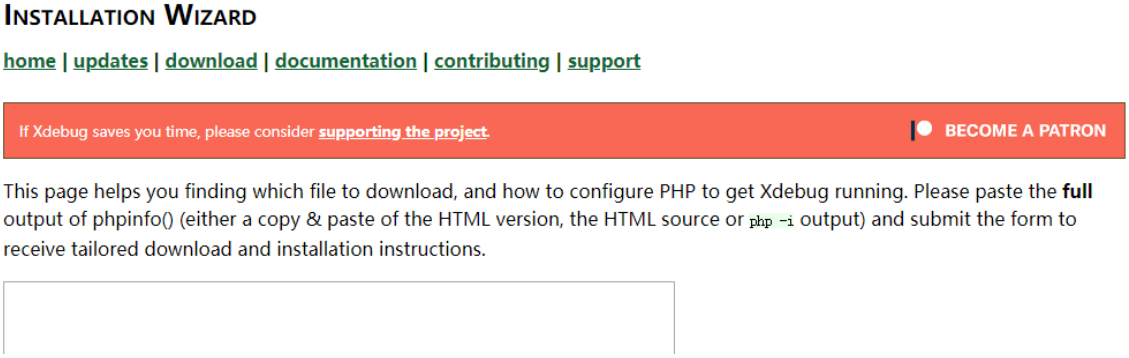
xdebug调试对于PHP项目排错有非常重要的意义，能帮助我们快速定位到问题，从而迅速解决问题。接下来向大家详细介绍phpstrom 配置xdebug.

1.7.1 下载对应的版本的xdebug：

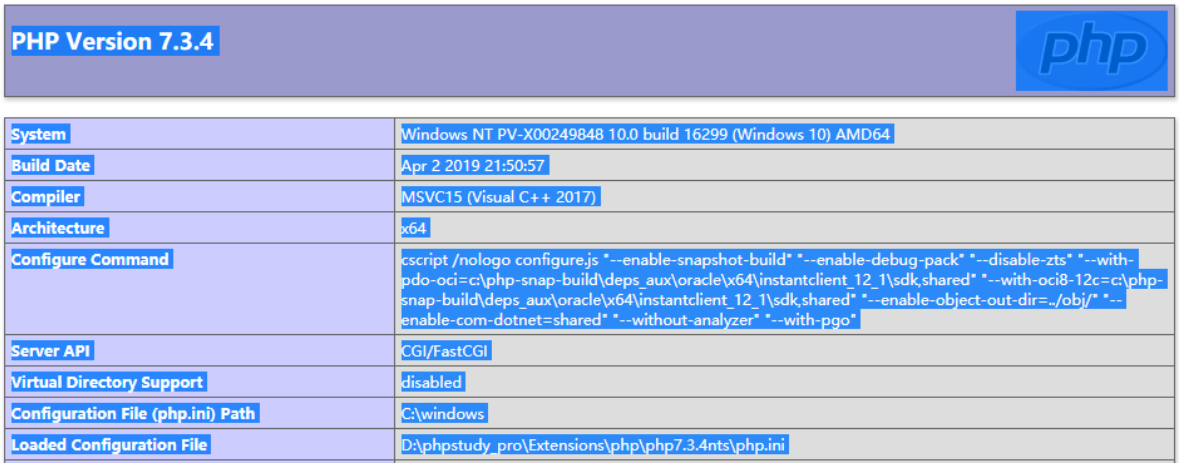
xdebug官网下载地址：https://xdebug.org/download.php

你需要仔细分析和选择要下载的对应版本，否则无法调试。由于非常容易出错，建议采用下面这种简单方法：

xdebug网站提供一个自动分析你系统对应的xdebug版本的页面，网址是 https://xdebug.org/wizard.php



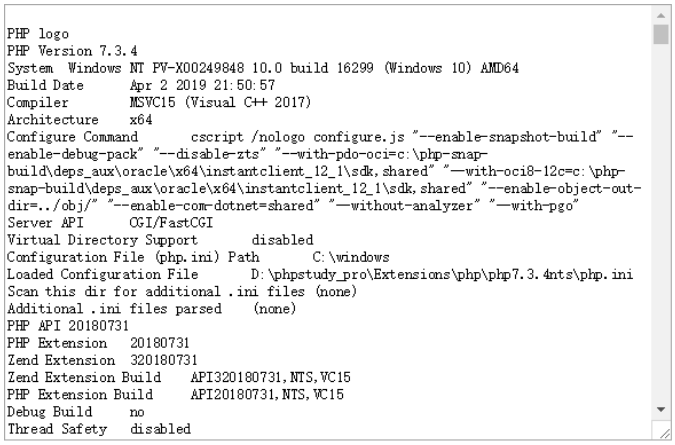
在页面中需要粘贴进去php版本信息，也就是phpinfo()函数的信息，如下图：



PHP Version 7.3.4

System	Windows NT PV-X00249848 10.0 build 16299 (Windows 10) AMD64
Build Date	Apr 2 2019 21:50:57
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x64
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--disable-zts" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\windows
Loaded Configuration File	D:\phpstudy_pro\Extensions\php\php7.3.4nts\php.ini

将函数信息 全选 复制到 xdebug 的方框里面 点击 analyse my phpinfo() output 按钮



```
PHP logo
PHP Version 7.3.4
System  Windows NT PV-X00249848 10.0 build 16299 (Windows 10) AMD64
Build Date      Apr 2 2019 21:50:57
Compiler        MSVC15 (Visual C++ 2017)
Architecture    x64
Configure Command       cscript /nologo configure.js "--enable-snapshot-build" "--
enable-debug-pack" "--disable-zts" "--with-pdo-oci=c:\php-snap-
build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-
snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-
dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API      CGI/FastCGI
Virtual Directory Support       disabled
Configuration File (php.ini) Path       C:\windows
Loaded Configuration File       D:\phpstudy_pro\Extensions\php\php7.3.4nts\php.ini
Scan this dir for additional .ini files (none)
Additional .ini files parsed    (none)
PHP API 20180731
PHP Extension   20180731
Zend Extension  320180731
Zend Extension Build    API320180731,NTS,VC15
PHP Extension Build     API20180731,NTS,VC15
Debug Build     no
Thread Safety   disabled
```

The information that you upload will not be stored. The script will only use a few regular expressions to analyse the output and provide you with instructions. You can see the code here.

Analyse my phpinfo() output

我们将获得如下信息：

You're already running the latest Xdebug version

But here are the instructions anyway:

1. Download php_xdebug-2.7.2-7.3-vc15-nts-x86_64.dll
2. Move the downloaded file to D:\phpstudy_pro\Extensions\php\php7.3.4nts\ext
3. Update `D:\phpstudy_pro\Extensions\php\php7.3.4nts\php.ini` and change the line
   `zend_extension = D:\phpstudy_pro\Extensions\php\php7.3.4nts\ext\php_xdebug-2.7.2-7.3-vc15-nts-x86_64.dll`
4. Restart the webserver

我们依照上面的步骤操作就可以了，至此xdebug的下载和启用就完成了，重新运行 phpinfo.php 文件，在打开的页面中出现如下标识说明安装正确：

xdebug

xdebug support	enabled
Version	2.7.2
IDE Key	PHPSTORM

1.7.2 phpstorm中使用xdebug：

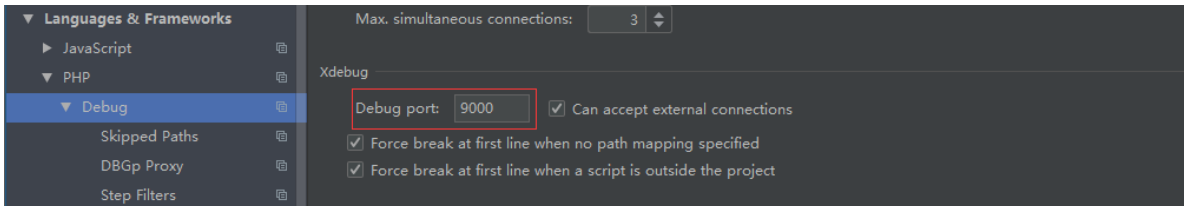
修改PHP配置文件

在php.ini文件的末尾追加如下代码

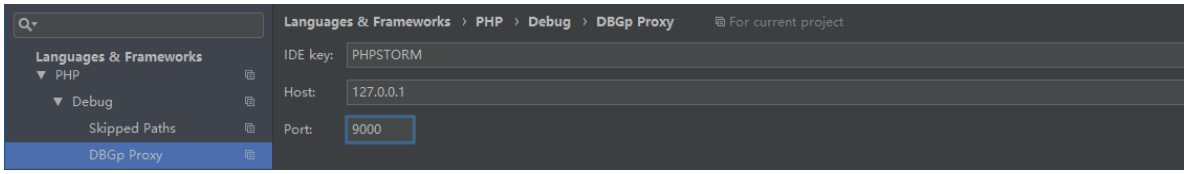
```
[xdebug]
; 扩展文件
zend_extension = D:\phpstudy_pro\Extensions\php\php7.3.4nts\ext\php_xdebug-2.7.2-7.3-vc15-nts-x86_64.dll
; 开启远程调试
xdebug.remote_enable = on
; 调试协议
xdebug.remote_handler = dbgp
; 远程地址
xdebug.remote_host= 127.0.0.1
; 远程调试端口
xdebug.remote_port = 9000
; idekey 区分大小写
xdebug.idekey = PHPSTORM
```

PHPstorm 中的配置

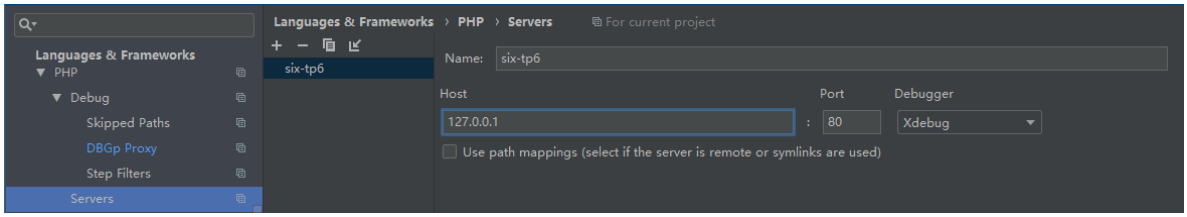
在文件->设置->语言与框架中->PHP->debug 设置端口



在DBGp Proxy中配置你的idekey，idekey就是你在配置文件中最后一项，host是你的服务器ip或者是已经可以解析的域名。

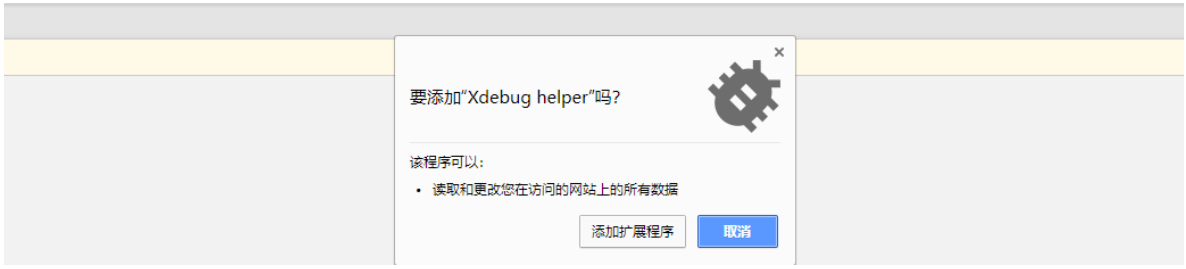


自己手动添加一个，Host填服务器ip，然后port是默认80端口

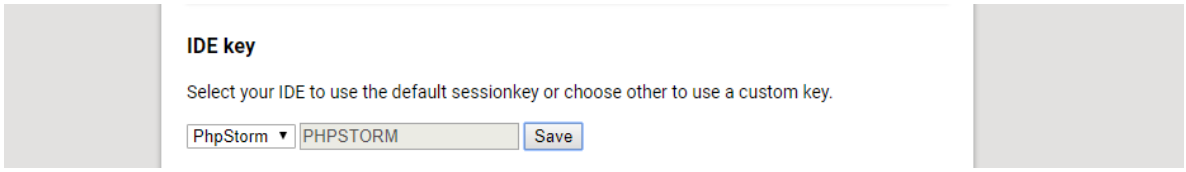


1.7.3 安装浏览器调试插件 (xdebug helper，这里使用星愿浏览器作为演示浏览器）。

首先将从网上下载的插件拖拽至浏览器的首页，会弹出如下窗口，点击“添加扩展程序”



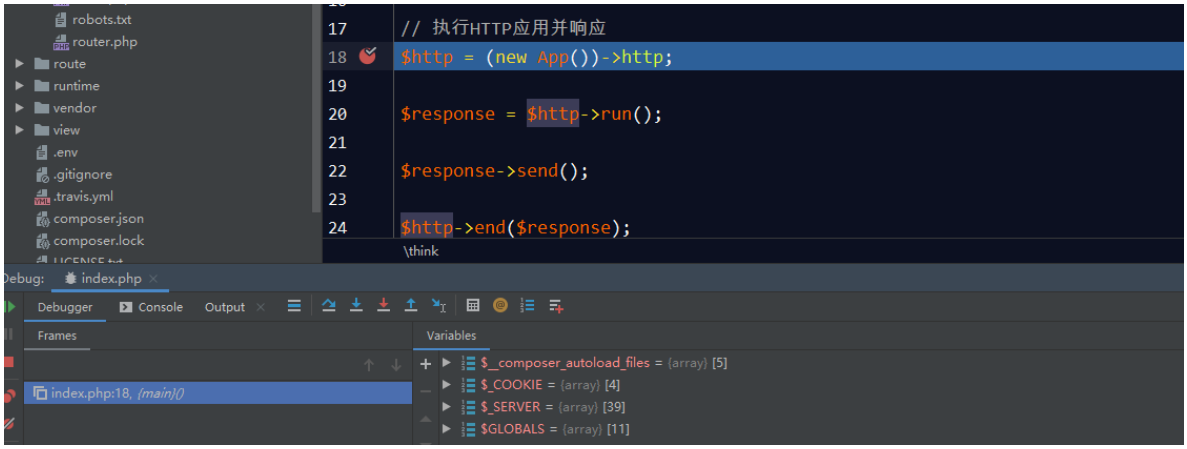
插件安装之后，在浏览器的右上角会出现一个 “七星瓢虫” 的图标，右击图标，选择 “选项”，在弹出的界面中将IDE key 选择 phpstorm



最后，单击 “七星瓢虫” 图标，激活插件。到这里，整个phptorm 配置xdebug调试就全部结束了。

1.7.4 断点调试

我们可以在任意想要断住得地方下断点，在某个地方下了断点之后，程序运行到这个地方就会停下来，并且会把变量信息展示出来。



各位同学们，以上就是tp快速入门得全部内容。