1801210682

ThinkPHP学习笔记

1801210682

一. 配置环境

1. 在终端中输入ssh root@140.143.148.177，并输入密码登录至云服务器

2. 输入yum install screen安装screen，screen是可以在多个进程之间多路复用一个物理终端的窗口管理器，安装完成后使用screen -S lnmp打开新窗口

3. 输入wget http://soft.vpser.net/lnmp/lnmp1.5.tar.gz -cO lnmp1.5.tar.gz下载lnmp并使用tar zxf lnmp1.5.tar.gz解压缩

4. 输入cd lnmp1.5.tar.gz进入目录并输入./install.sh lnmp安装lnmp

5. 设置MYSQL数据库密码并将所有的配置设为默认

6. 访问ip地址出现以下页面即为安装成功，默认访问的地址为/home/wwwroot/default

7. 在http://www.thinkphp.cn/donate/download/id/1278.html中下载thinkphp，并使用scp将其上传到/home/wwwroot/default/test文件夹下

8. 输入sudo chmod 777 -R application为目录增加权限，访问http://140.143.148.177/test/public/index.php出现以下界面即为框架安装成功

二. 目录结构

1. ThinkPHP的目录结构如下图所示：

```
project  应用部署目录
├─application           应用目录（可设置）
|  ├─common            公共模块目录（可更改）
|  ├─index             模块目录(可更改)
|  |  ├─config.php     模块配置文件
|  |  ├─common.php     模块函数文件
|  |  ├─controller     控制器目录
|  |  ├─model          模型目录
|  |  ├─view           视图目录
|  |  └─ ...           更多类库目录
|  ├─command.php       命令行工具配置文件
|  ├─common.php        应用公共（函数）文件
|  ├─config.php        应用（公共）配置文件
|  ├─database.php      数据库配置文件
|  ├─tags.php          应用行为扩展定义文件
|  └─route.php         路由配置文件
├─extend                扩展类库目录（可定义）
├─public                WEB 部署目录（对外访问目录）
|  ├─static            静态资源存放目录(css,js,image)
|  ├─index.php         应用入口文件
|  ├─router.php        快速测试文件
|  └─.htaccess         用于 apache 的重写
├─runtime               应用的运行时目录（可写，可设置）
├─vendor                第三方类库目录（Composer）
├─thinkphp              框架系统目录
|  ├─lang              语言包目录
|  ├─library           框架核心类库目录
|  |  ├─think          Think 类库包目录
|  |  └─traits         系统 Traits 目录
|  ├─tpl               系统模板目录
|  ├─.htaccess         用于 apache 的重写
|  ├─.travis.yml       CI 定义文件
|  ├─base.php          基础定义文件
|  ├─composer.json     composer 定义文件
|  ├─console.php       控制台入口文件
|  ├─convention.php    惯例配置文件
|  ├─helper.php        助手函数文件（可选）
|  ├─LICENSE.txt       授权说明文件
|  ├─phpunit.xml       单元测试配置文件
|  ├─README.md         README 文件
|  └─start.php         框架引导文件
├─build.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
```

2. public目录是作为web目录访问内容，public目录下的index.php为应用入口文件，会转到application路径

3. public/static是静态存放目录，用来存放css和js，并使用<link rel='stylesheet' href='…/css.css' type='text/css'>和<script type="text/javascript" src="…/other.js"></script>导入存储在static中存放的css和js文件

4. runtime是运行时的目录，必须确保有写权限

5. application目录是web页面存放的目录，其中的common模块为公共模块，可以将所有公用的方法存放在common中

6. common模块是默认禁止直接访问的

7. database.php是关于数据库的配置文件，可以在该文件中修改服务器地址、数据库名、用户名、密码和端口等参数

三. MVC架构

1. ThinkPHP是基于MVC（模型-视图-控制器）的方式来组织，将应用程序分成模型（M）、视图（V）、控制器（C），它们各自处理自己的任务，这种方法可以让业务逻辑、数据、界面显示分离，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC 分层同时也简化了分组开发。不同的开发人员可同时开发视图、控制器逻辑和业务逻辑

2. 应用是由多个模块组成，每个模块是应用目录下的一个子目录，拥有独立的MVC库和配置文件

3. 模块目录全部采用小写和下划线命名

4. 控制器是应用程序中处理用户交互的部分，负责相应请求，存放在模块的controller目录下，每个控制器是一个独立的控制器类，主要负责视图中请求的接收，并调用相关的模型处理，并最终通过视图输出

5. 一个控制器包含多个方法，方法是一个URL访问的最小单元

6. 模型表示应用程序核心，通常完成实际的业务逻辑和数据封装，并返回和格式无关的数据，连接数据库和对数据库进行的操作都需要在模型中完成

7. 视图控制器调用模型类后返回的数据通过视图组装成不同格式的输出。视图根据不同的需求，来决定调用模板引擎进行内容解析后输出还是直接输出

8. 命名空间是用来组织和重用代码的，为了解决库文件中的同名变量或函数。通过使用 namespace xxx；使用库函数或变量就是在该名字空间中定义的，从而避免冲突

9. URL的典型访问规则是：http://serverName/index.php（或者其它应用入口文件）/模块/控制器/操作/[参数名/参数值…]，可以通过配置伪静态隐藏入口文件从而加强安全性

10. 入口文件是用户请求的PHP文件，负责处理一个请求（注意，不一定是URL请求）的生命周期，最常见的入口文件是index.php

四. 控制器

1. 下图为一个典型的控制器：

```
namespace app\index\controller;

class Index
{
    public function index()
    {
        return 'index';
    }
}
```

2. 如果继承think\Controller，可以调用$this->fetch和$this->assign等方法

3. 默认情况下，控制器的输出采用return方式

4. 可以通过['except' => '方法名,方法名']和['only' => '方法名,方法名']控制某些方法的前置方法

5. $this->success('新增成功', 'User/list');和 $this->error('新增失败');可以用来进行带提示信息的页面跳转

6. $this->redirect()可以进行页面重定向，$this->redirect('News/category', ['cate_id' => 2], 302, ['data' => 'hello']);可以在重定向时在SESSION中存储数据

7. redirect('News/category')->remember();可以在跳转时记录当前的url，并使用redirect()->restore();恢复到上次记录的url

8. $request = Request::instance();可以用于获取当前的请求信息，使用Request::instance()->post()可以获取通过post传递到后端的数据

9. 各命令的功能如下图所示：

```
$request = Request::instance();
// 获取当前域名
echo 'domain: ' . $request->domain() . '<br/>';
// 获取当前入口文件
echo 'file: ' . $request->baseFile() . '<br/>';
// 获取当前URL地址 不含域名
echo 'url: ' . $request->url() . '<br/>';
// 获取包含域名的完整URL地址
echo 'url with domain: ' . $request->url(true) . '<br/>';
// 获取当前URL地址 不含QUERY_STRING
echo 'url without query: ' . $request->baseUrl() . '<br/>';
// 获取URL访问的ROOT地址
echo 'root:' . $request->root() . '<br/>';
// 获取URL访问的ROOT地址
echo 'root with domain: ' . $request->root(true) . '<br/>';
// 获取URL地址中的PATH_INFO信息
echo 'pathinfo: ' . $request->pathinfo() . '<br/>';
// 获取URL地址中的PATH_INFO信息 不含后缀
echo 'pathinfo: ' . $request->path() . '<br/>';
// 获取URL地址中的后缀信息
echo 'ext: ' . $request->ext() . '<br/>';
```

五. 模型

1. 对于数据库的操作都在model中进行，在定义model时同样需要指定namespace和use，通常是namespace app\index\model;和use think\Model;

2. 数据库的查询操作为Db::table('think_user')->where('id',1)->find();（查询一个数据，结果不存在返回null），Db::table('think_user')->where('status',1)->select();（查询所有符合条件的数据，结果不存在返回空数组）

3. 数据库的添加操作为Db::table('think_user')->insert($data);（insert 方法添加数据成功返回添加成功的条数，insert 正常情况返回 1），Db::name('user')->insertAll($data);（insertAll 方法添加数据成功返回添加成功的条数）

4. 数据库的数据更新操作为Db::table('think_user')->where('id', 1)->update(['name' => 'thinkphp']);（update 方法返回影响数据的条数，没修改任何数据返回 0）

5. 数据库的数据自增操作为Db::table('think_user')->where('id', 1)->setInc('score');自减操作为Db::table('think_user')->where('id', 1)->setDec('score');

6. 数据库的删除操作为Db::table('think_user')->where('id',1)->delete();还可以使用Db::table('think_user')->where('id','<',10)->delete();进行条件删除

7. 可以使用Db::transaction进行事务操作，当发生异常会自动回滚，防止数据库操作问题，如下图所示：

```
Db::transaction(function(){
    Db::table('think_user')->find(1);
    Db::table('think_user')->delete(1);
});
```

六. 视图

1. 使用extends继承\think\Controller，可以直接调用控制器基础类封装的相关视图类的方法，如果使用use，就先需要实例化类才可以使用类中的方法

2. 渲染模版输出的方法是return $this->fetch('hello',['name'=>'thinkphp']);，其他可以直接调用的方法如下图所示：

方法	说明
fetch	渲染模板输出
display	渲染内容输出
assign	模板变量赋值
engine	初始化模板引擎

3. $this->assign可以为前端的变量赋值，前端使用{$value}既可以生成动态的内容