ThinkPHP Framework 1.0
Run Flow

ThinkPHP 1.0
执行流程

编写：ThinkPHP 文档组

最后更新：2008-07-27

目录

1 概述

本文描述了 ThinkPHP 的执行流程和各个子流程。

2 执行流程分析

我们对用户的第一次 URL 访问 http://<serverIp>/My/index.php/Index/show/ 所执行的流程进行详细的分析，用户的 URL 访问首先是定位到了 My 项目的 index.php 入口文件（注意：如果使用了 URL_REWRITE，可能 index.php 已经被隐藏了），项目的入口文件所做的其实是实例化一个 App 应用实例，并且执行这个应用。

2.1 加载公共入口文件

在实例化 App 类之前，我们需要首先加载系统的公共入口文件 ThinkPHP.php，这个文件是 ThinkPHP 的总入口，让我们来一探究竟。在加载 ThinkPHP.php 文件的过程中，其实完成了下面的操作：

- ✧ 记录开始执行时间 $GLOBALS['_beginTime']；
- ✧ 检测 THINK_PATH 定义，如果没有则创建；
- ✧ 检测项目名称 APP_NAME，如果没有则按照一定规则自动定义；
- ✧ 检测项目编译缓存目录定义，没有则取项目的 Temp 目录；
- ✧ 加载系统定义文件 defines.php 和公共函数文件 functions.php；
- ✧ 如果项目编译缓存目录不存在，则自动创建项目目录结构；
- ✧ 加载系统核心类库（包括 Base、App、Action、Model、View、ThinkException、Log）；
- ✧ 如果 PHP 版本低于 5.2.0 则加载兼容函数库 compat.php；
- ✧ 生成核心编译缓存~runtime.php；
- ✧ 记录加载文件时间 $GLOBALS['_loadTime']；

2.2 项目初始化 init

在加载完成 ThinkPHP 的公共入口文件之后 ,我们就开始执行应用了 ,而首先应该是初始化 App 应用。

- 设定错误和异常处理机制（set_error_handler 和 set_exception_handler）;
- 项目预编译并载入；
- 设置时区支持；
- Session 过滤器检查；
- session 初始化；
- 检查并加载插件；
- URL 分析和调度；
- 获取当前执行的模块和操作名；
- 加载模块配置文件；
- 页面防刷新机制检查；
- 语言检查并读取对应的语言文件；
- 模板检查并定义相关的模板变量；
- RBAC 权限检测；
- 如果开启静态写入则读取静态缓存文件；
- 应用初始化过滤插件 app_init；
- 记录应用初始化时间 $GLOBALS['_initTime']

2.3 项目预编译

- 加载系统惯例配置文件 convention.php；
- 加载项目配置文件 config.php；

- ✧ 加载项目公共文件 common.php；
- ✧ 如果是调试模式加载系统调试配置文件 debug.php；
- ✧ 如果定义了项目的调试配置文件则载入 debug.php；
- ✧ 生成项目编译缓存文件~app.php；

2.4 URL 分析 Dispatcher

- ✧ 检查当前 URL 模式 URL_MODEL；
- ✧ 如果存在$_GET 变量，则根据当前的 URL 模式和设置进行重定向；
- ✧ 进行路由定义检测；
- ✧ 分析 PATH_INFO 的 URL 信息到数组；
- ✧ 把 PATH_INFO 得到的值和$_GET 合并；

2.5 获取模块和操作名

- ✧ 检查 VAR_MODULE 变量（包括 GET 和 POST），如果未定义，则获取默认模块名；
- ✧ 检查组件模块；
- ✧ 检查模块伪装；
- ✧ 检查 VAR_ACTION 变量（包括 GET 和 POST），如果未定义，则获取默认操作名；
- ✧ 检查操作链；
- ✧ 检查操作伪装；

2.6 项目执行 exec

- ✧ AUTO_LOAD_CLASS 检查 如果有则导入公共类;
- ✧ 实例化当前模块的 Action 控制器类;

- 如果 Action 控制器不存在则检查空模块 EmptyAction;
- 检查操作链，如果有执行操作链；
- 检查前置操作方法 _before_操作名；
- 执行模块的操作方法，调度转移给 Action 控制器；
- 执行后置操作方法 _after_操作名；
- 执行应用结束过滤器 app_end；
- 如果开启日志记录，写入错误日志；

2.7 执行控制器的操作

- 实例化视图类 View；
- 取得当前控制器名称；
- 控制器初始化_initialize；
- 如果操作方法不存在检查空操作 _empty；
- 如果空操作没有定义则检查对应的模板文件；
- 调用模型获取数据；
- 渲染视图进行输出；

2.8 调用模型获取数据 find

- 实例化模型类；
- 模型初始化 _initialize；
- 判断当前模型名称和对应数据表；
- 实例化数据库操作对象；

- 数据表字段检测并缓存；
- 查询需要的数据；
- 判断是否视图模型；
- 如果是延时查询返回 ResultIterator 对象；
- 取出数据对象的时候记录乐观锁；
- 获取文本字段数据；
- 获取关联数据；
- 对数据对象自动编码转换；
- 记录当前数据对象；
- 返回定义的数据格式（数组或者 stdClass 对象）

2.9 输出视图

- 模板变量赋值；
- 检测是否是布局输出；
- 检测页面输出编码；
- 缓存初始化过滤 ob_init;
- 页面缓存开启 ob_start;
- 缓存开启后执行的过滤;
- 模版文件名过滤 template_file;
- 定位当前输出的模板文件；
- 模版变量过滤 template_var；
- 根据不同模版引擎进行处理；

- ✧ 如果是 PHP 模板引擎，直接载入模板文件；
- ✧ 使用内置模板引擎，检测缓存有效期；
- ✧ 缓存无效则重新编译模板文件；
- ✧ 载入模板缓存文件；
- ✧ 获取并清空缓存；
- ✧ 输出编码转换；
- ✧ 输出过滤 ob_content；
- ✧ 开启静态写入则写入静态文件；
- ✧ 如果输出则获取视图运行时间；
- ✧ 如果是 display 则渲染模板输出信息；
- ✧ 开启页面 Trace 则显示页面 Trace 信息；
- ✧ 如果是 fetch 则返回模板输出信息；