Typecho-1.0-14.10.10版 install.php反序列化漏洞分析

**0x00 说明**

Typecho是由type和echo两个词合成的，来自于开发团队的头脑风暴。Typecho基于PHP5开发，支持多种数据库，是一款内核强健﹑扩展方便﹑体验友好﹑运行流畅的轻量级开源博客程序。选用Typecho，搭建独一无二个人网络日志发布平台，享受创作的快乐。

**0x01 前言**

首先必须先说明一点，本漏洞原作者不是我

源作者: LoRexxar

这个漏洞也是出了很久了，也是很可惜到今天才有时间去看这个漏洞，读者也可以从此次漏洞中学习到开发中的很多注意项与审计中的思路。

注：本文并没有嘲讽此博客系统的意思，只供学习思路

利用文件：

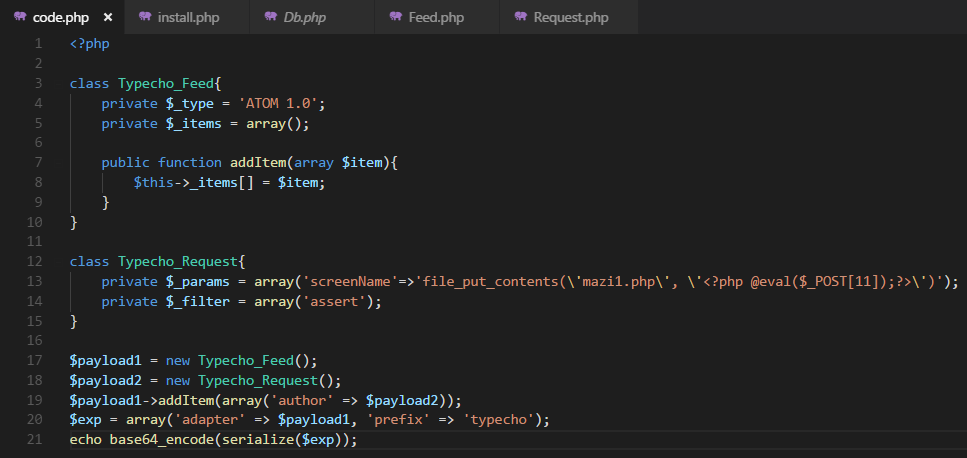


测试代码：

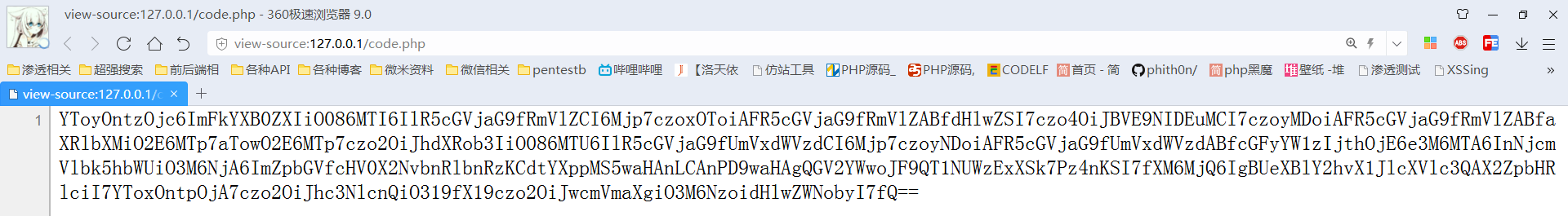


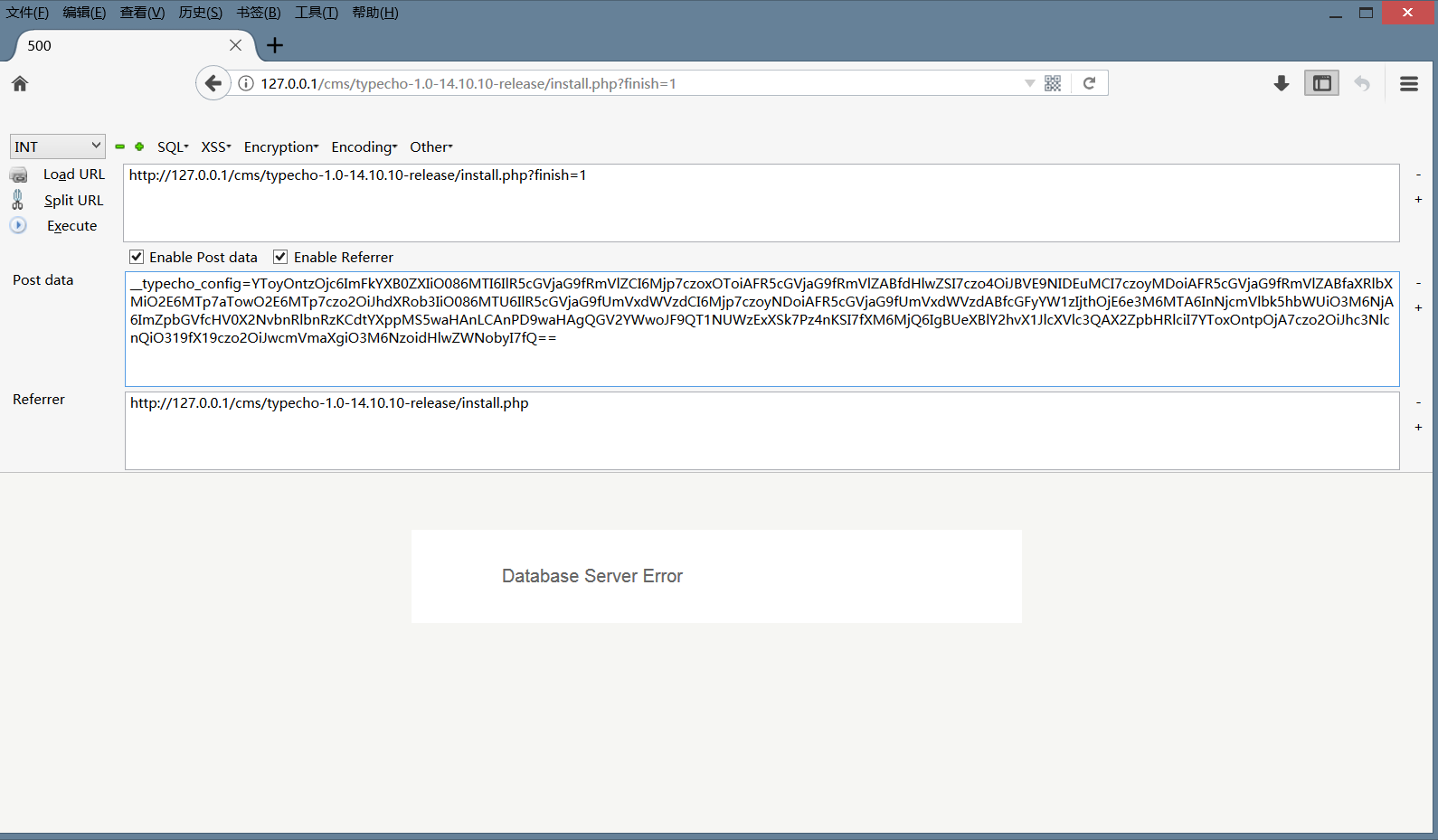
**0x02漏洞利用**

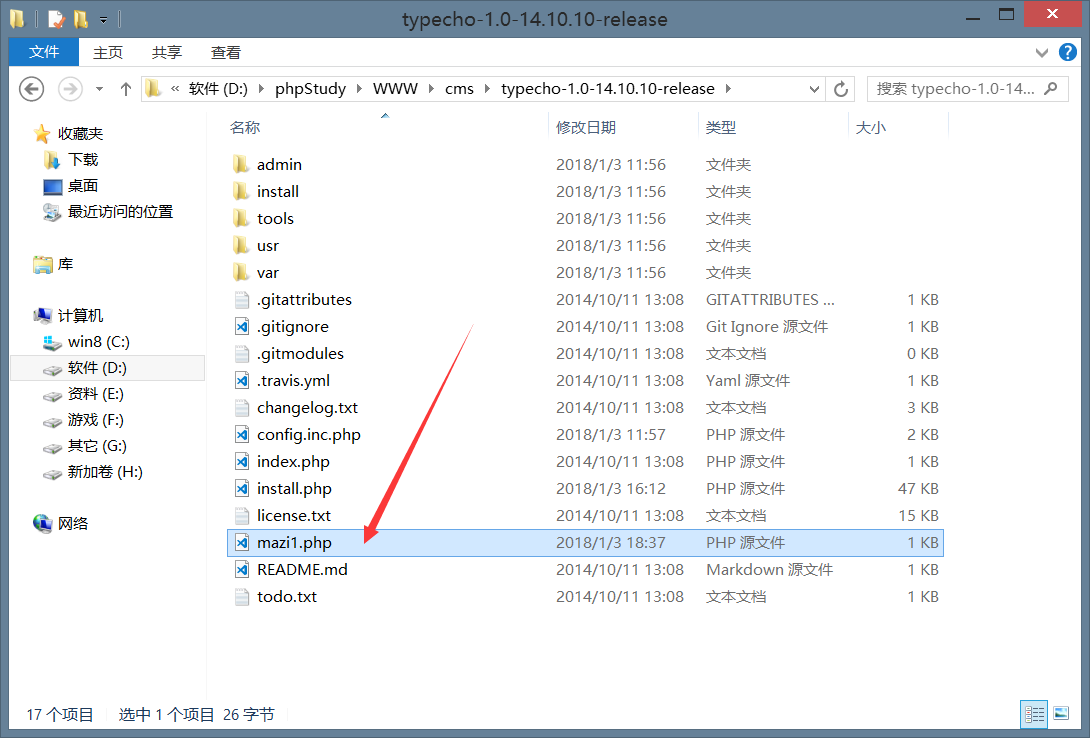
先运行利用脚本



获取参数以后复制他：



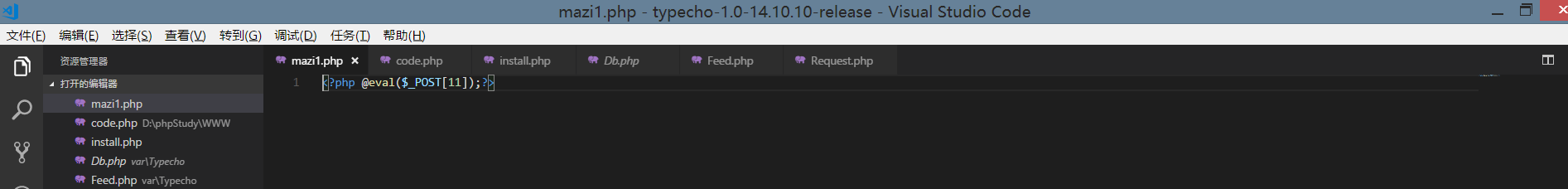


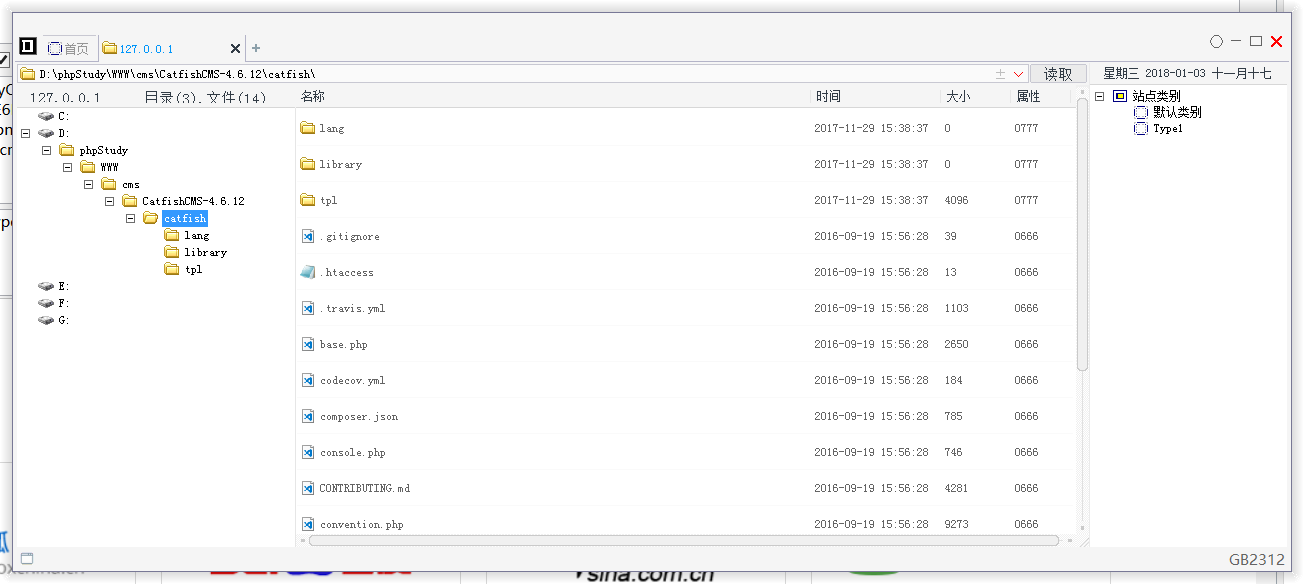


最终就会生成一句话木马：

访问：http://127.0.0.1/cms/typecho-1.0-14.10.10-release/mazi1.php

POST: 11=phpinfo();



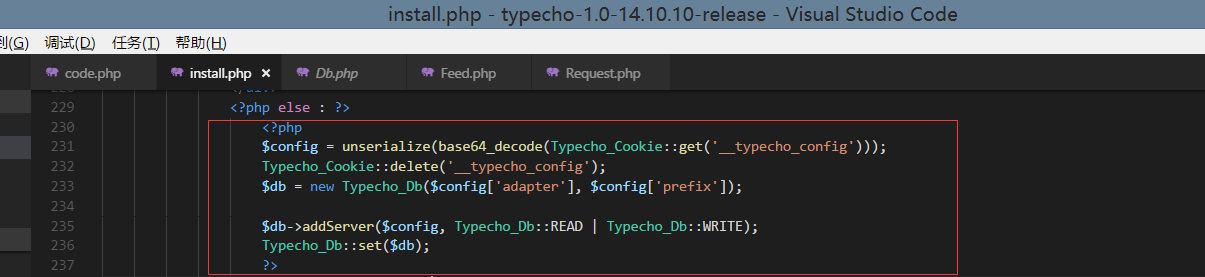


**0x03漏洞原理**

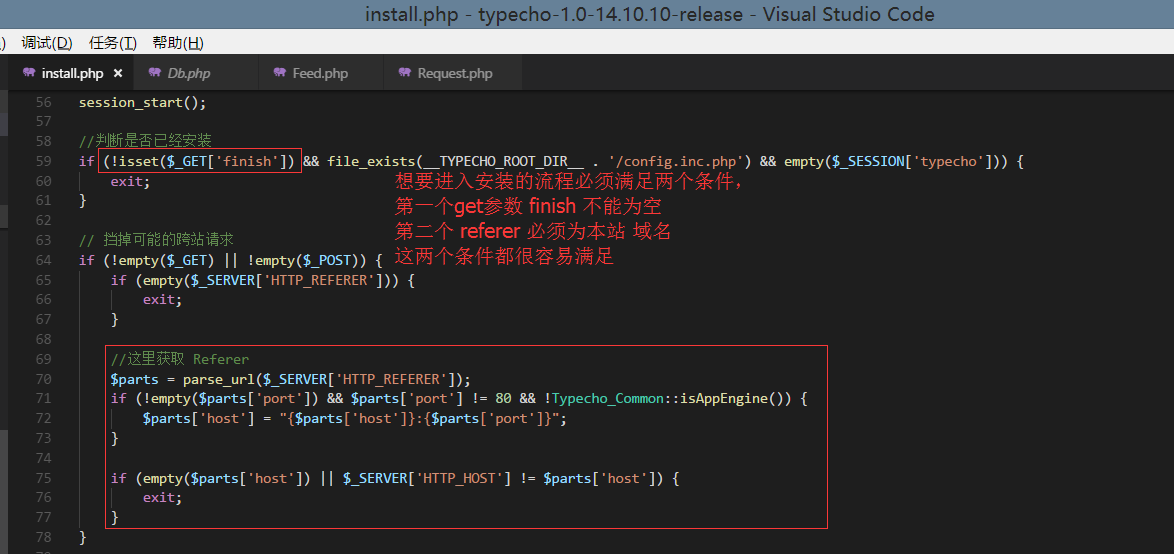
首先看看文件

文件地址： typecho-1.0-14.10.10-release\install.php

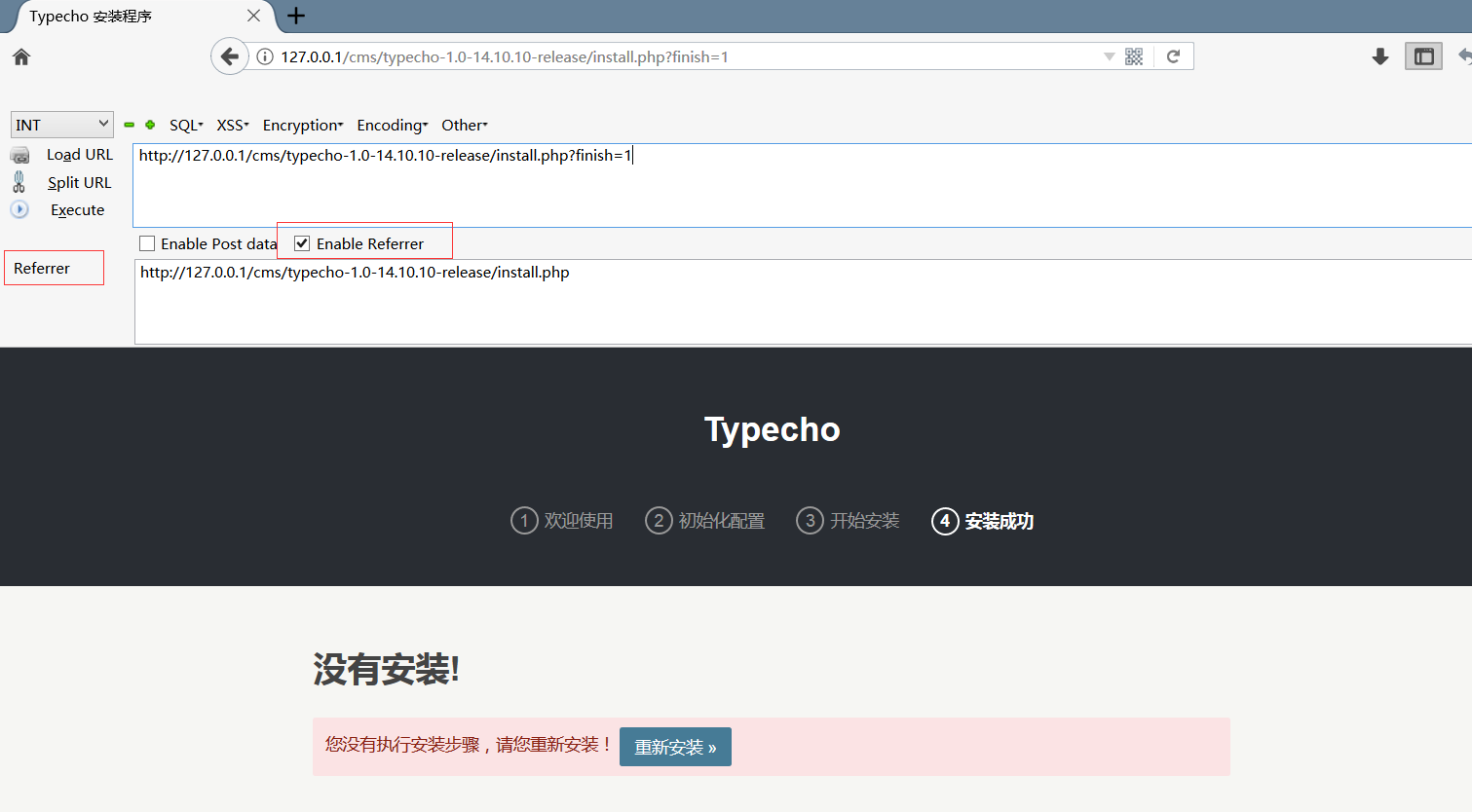
问题行数：230-237



想要进入这个流程，需要满足两个条件

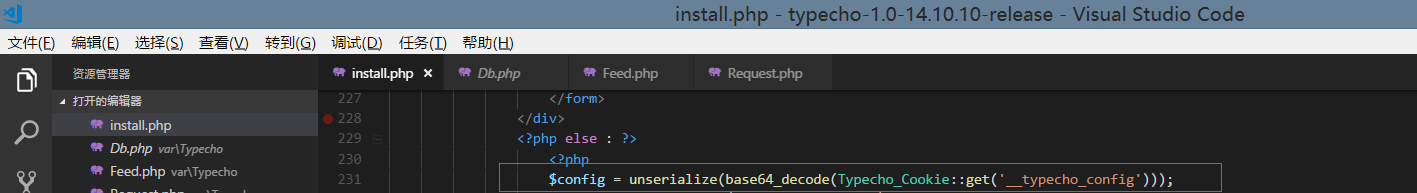


具体方法的话，我们可以这样干

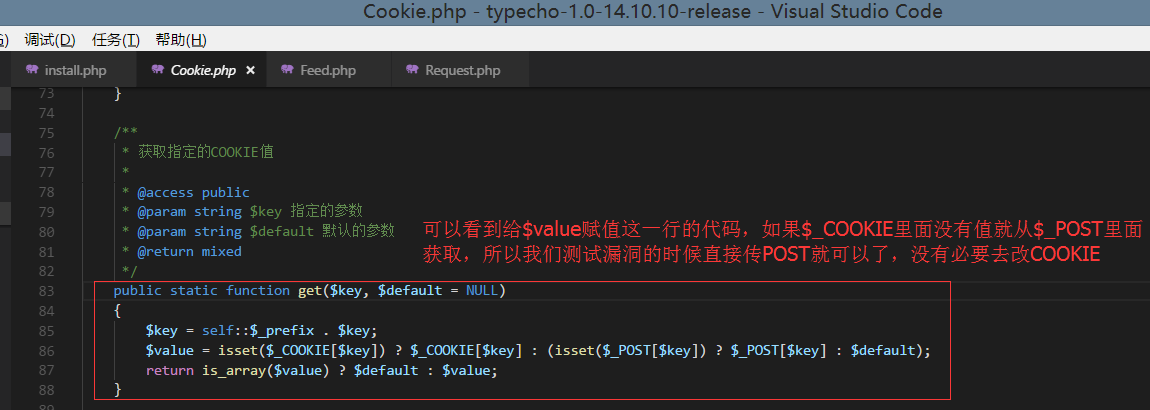


看到这个就说明进入了安装流程了。

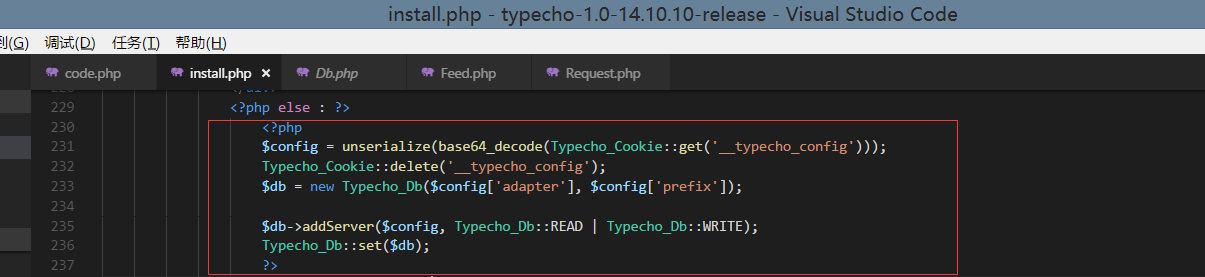
继续跟反序列化的地方：



首先使用Typecho\_Cookie的get方法获取\_\_typecho\_config，get方法如下：



那么我们继续



第231行获取cookie中的\_\_typecho\_config值base64解码，然后在进行反序列化。想要执行成功，只需$\_GET['finish']不为空，并且\_\_typecho\_config存在值。反序列化成功后233行把$config['adapter']和$config['prefix']传入Typecho\_Db进行实例化。

反序列化漏洞想要成功利用势必离不开PHP面向对象的魔术方法。

这里的话，笔者整理了一份反序列化相关的魔术方法

1.构造方法： \_\_construct()

参数：自定义

触发时机：new的一瞬间自动调用

作用：初始化成员属性

2.析构方法：\_\_destruct()

参数：无

触发时机：在对象释放的一瞬间自动调用

作用：做收尾工作，数据库关闭连接等

3.\_\_get()

参数：一个，成员属性名称

触发时机：在类外部访问私有的受保护的或者未定义的成员属性时自动调用

作用：控制私有的受保护的未定义的成员属性的访问

4.\_\_set()

参数：两个，成员属性的名称和值

触发时机：在类外部设置私有的受保护的未定义的成员属性时自动调用

作用：对私有的受保护的未定义的成员属性进行赋值控制

5.\_\_isset()

参数：一个参数，成员属性名称

触发时机：在类外部调用isset和empty方法验证私有的受保护的未定义的成员属性时自动调用

作用：对私有的受保护的未定义成员属性进行isset和empty的判断控制

6.\_\_unset()

参数：一个参数，成员属性名称

触发时机：在类外部使用unset方法对私有的受保护的成员属性进行销毁的一瞬间自动调用

作用：对私有的受保护的成员属性进行unset销毁控制

7.\_\_clone()

参数：无

触发时机：在类外部使用clone关键字对对象进行克隆时自动调用

作用：对克隆体进行标示

8.\_\_toString()

参数：无

触发时机：在类外部echo对象的时候自动调用

作用：快速输出对象的内容

注意：本方法必须返回一个字符串类型的内容

9.\_\_call()

参数：两个参数，调用的不存在方法名称，第二个参数为方法的参数（数组）

触发时机：使用对象调用一个不存在的方法自动调用

作用：屏蔽显示的错误

10.\_\_autoload()

参数：一个参数，new的类的名称

触发时机：当new一个找不到的类或者继承的父类或者在方法体中new的找不到的类的时候自动调用

作用：自动加载找不到的类文件

注意：本方法是在类外部定义的

11.\_\_sleep()

参数：无

触发时机：当在类外部对对象进行串行化的一瞬间自动调用serialize()

作用：可以对串行化的成员属性的选择进行控制

注意：必须返回一个数组（元素：成员属性名称）

12.\_\_wakeup()

参数：无

触发时机：反串行化的一瞬间自动调用unserialize()

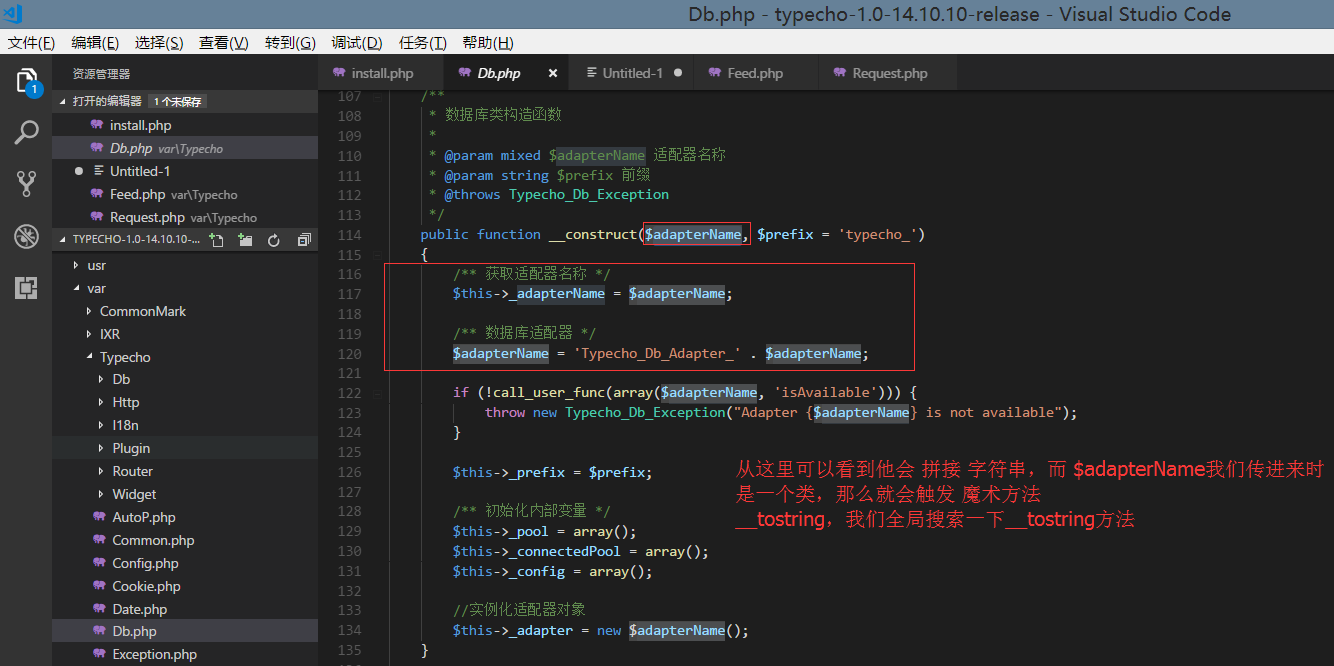
作用：在唤醒时对成员属性进行控制

注意：在反串行化时必须有类在场

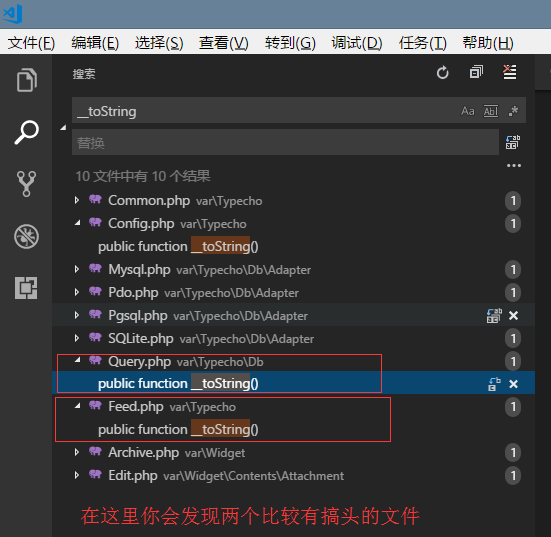
随后我们打开文件 DB.php

文件地址： var\Typecho\Db.php

文件行数：116-120



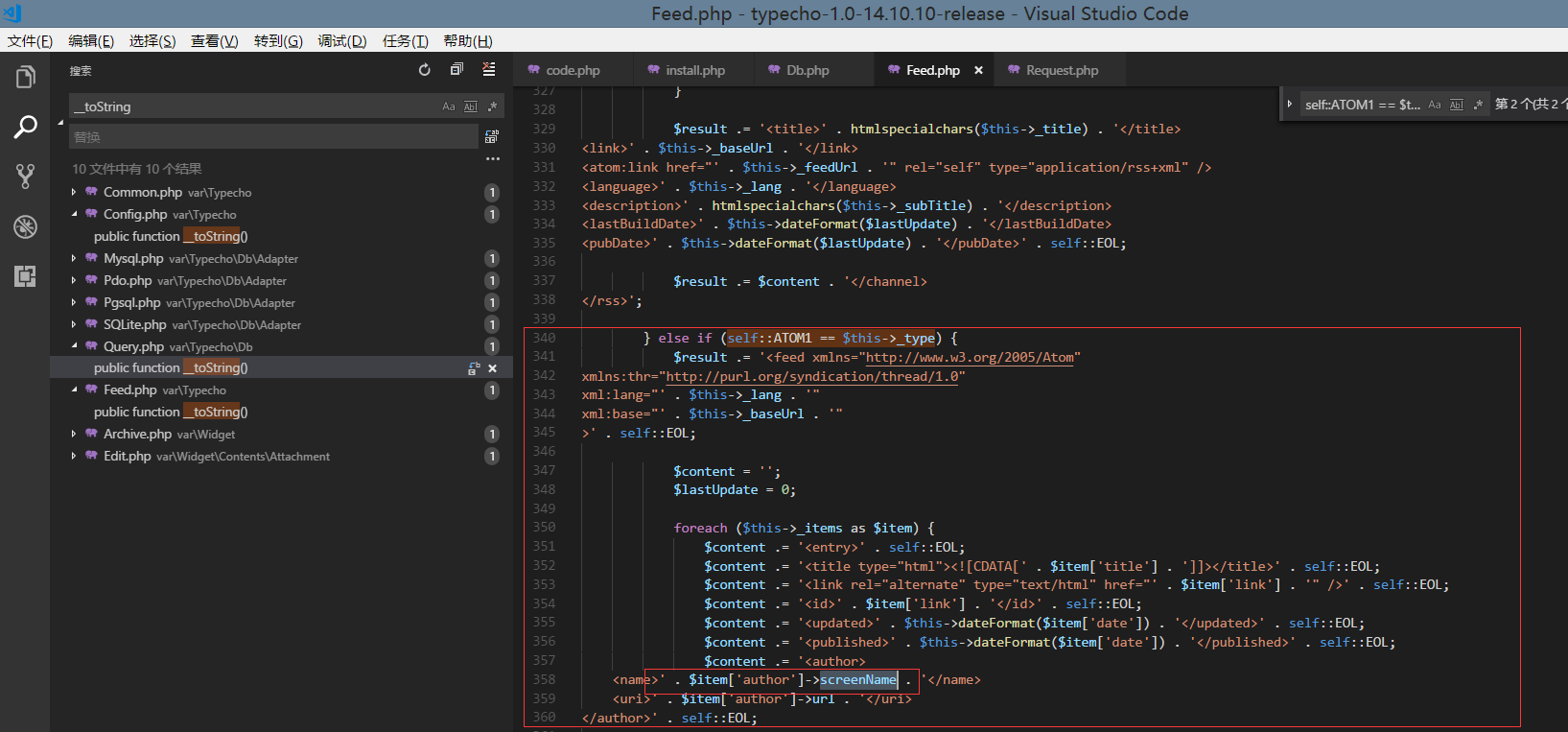
那么如果我们在反序列化中构造一个数组，其中adapter设置为一个类，那么就可以自动触发这个类的\_\_toString()方法，全局搜索一下。



文件地址：var\Typecho\Feed.php

这里的话，我选择查看Feed.php，查看Feed.php的\_\_toString() 方法，其中第340-360行：

调用了$item['author']->screenName，$item是$this->\_items的foreach循环出来的，并且$this->\_items是Typecho\_Feed类中的一个private属性。



那么我们就可以利用这个$item来调用某个类中的\_\_get()方法,

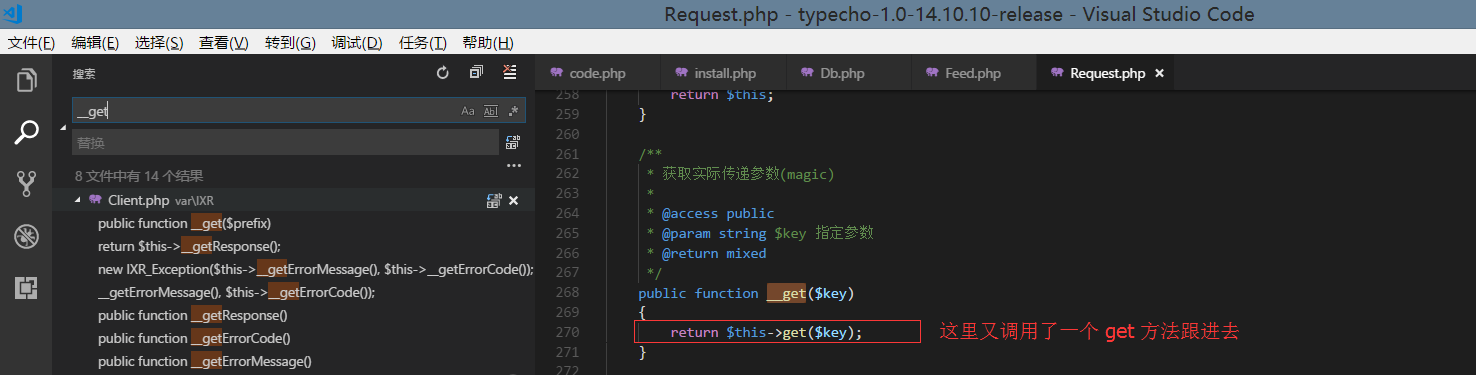
\_\_get()在类外部访问私有的受保护的或者未定义的成员属性时自动调用

全局搜索：\_\_get()

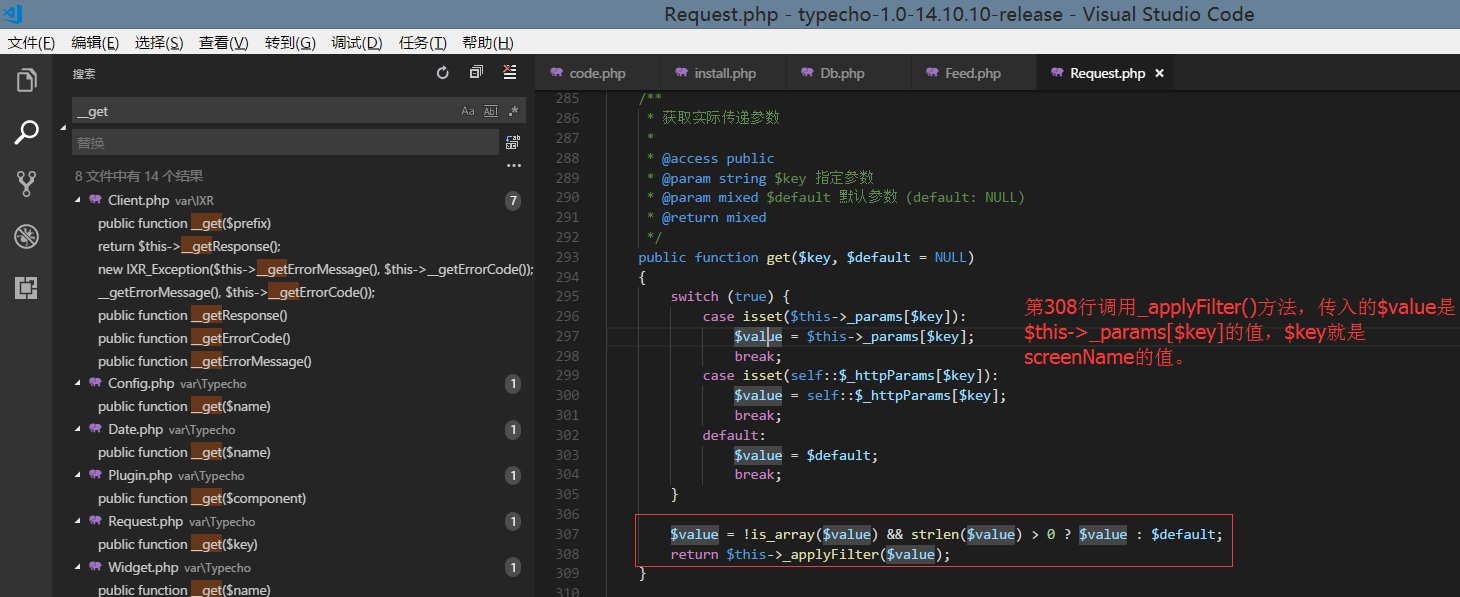
最终发现有搞头的文件

文件地址：var\Typecho\Request.php

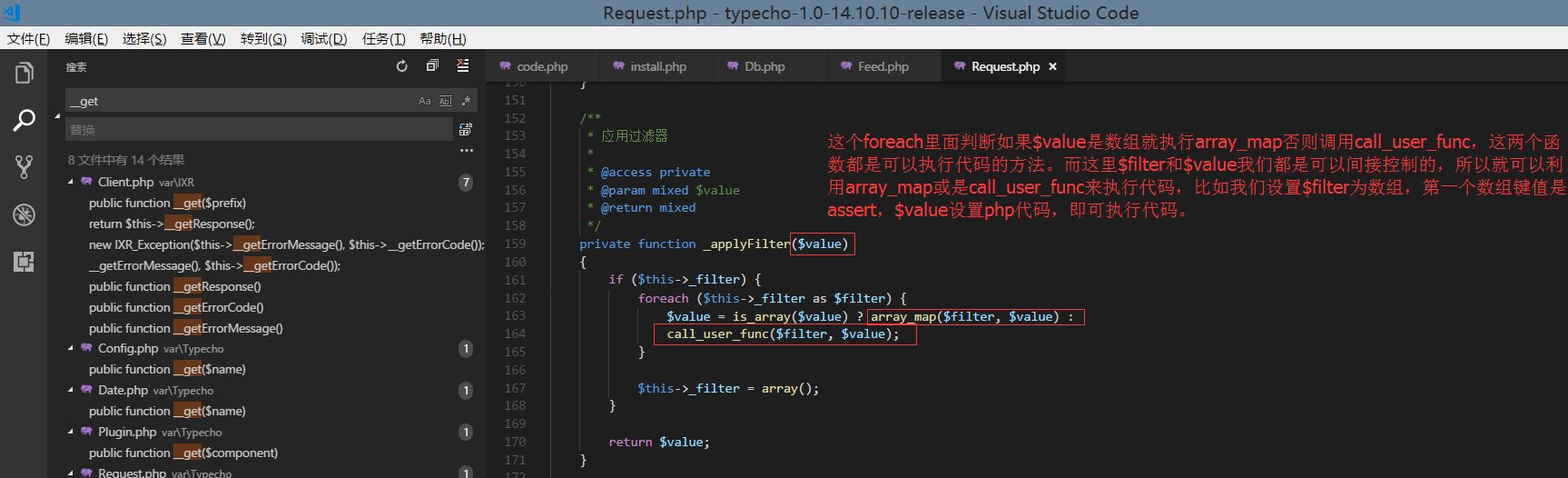
函数：\_\_get();



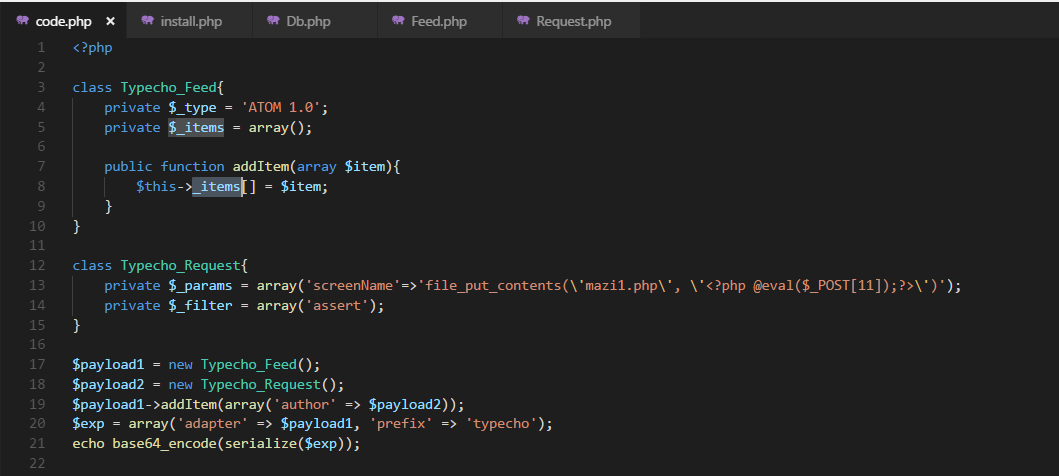
第308行调用\_applyFilter()方法，传入的$value是$this->\_params[$key]的值，$key就是screenName的值。



跟进\_applyFilter()，Request.php 159-171行：



最终的exp



整个漏洞的利用就这样结束了。

问题一：我怎么知道db.php 类中可以调用 \_\_toString() 方法？

$db = new Typecho\_Db($config['adapter'], $config['prefix']);

首先我们要知道一件事情，那么就是首先我们是可以控制

install.php 文件中的$config 的而当我们的 $config['adapter'] 传进去的是一个类时

因为db.php有析构方法\_\_construct 所以会先执行\_\_construct 方法 而\_\_construct方法中$adapterName 进行过了字符串拼接

也就是这句代码

/\*\* 数据库适配器 \*/

$adapterName = 'Typecho\_Db\_Adapter\_' . $adapterName;

那么根据我上面提供的 魔术方法的 事例就可以得知

\_\_toString() 在类外部echo对象的时候自动调用 而这里已经满足了这个文件，那么自然的也就可以调用\_\_toString() 方法

当然拉，使用这些方法的重点是，你操控的类，也就是传输进入的 反序列化的代码必须要设置了对应的 魔术方法 如果没有设置当然也是调用不成功的。

问题二：我怎么知道Feed.php类中 $item['author']->screenName 为什么就可以调用\_\_get()？

这里可能是有点难理解的，我就重点说明一下下这里是什么意思呢？

我们这里经过前面的传递已经是可以任意的new系统中的类了，那么 我们这一块的

$item['author'] 假如传入的是一个类

例如：

$item['author'] = new Typecho\_Request();

$item['author']->screenName

然后$item['author'] 调用screenName 而Typecho\_Request类中并没有screenName方法或是参数 然后就会自动调用 \_\_get() 方法，当然前提是Typecho\_Request 类中有设置\_\_get() 方法才行 : )