

PHP Spider 开发与维护指南 (DZ 风格)

本文档总结了基于 `spider.php` 框架开发、调试、转换 PHP 爬虫源的核心经验与最佳实践。旨在帮助开发者快速上手，并作为后续开发的参考手册。

1. 核心架构与工具

1.1 基础框架 (`lib/spider.php`)

核心框架文件现已移动至 `lib` 目录。

所有源必须包含 `lib/spider.php` 并继承 `BaseSpider` 类（通常在源文件中定义为 `class Spider extends BaseSpider`）。

引用规范：

```
require_once __DIR__ . '/lib/spider.php';
```

核心方法包括：

- `init()`：初始化（可选）。
- `homeContent($filter)`：获取首页分类与筛选配置。
- `categoryContent($tid, $pg, $filter, $extend)`：获取分类列表数据。
- `detailContent($ids)`：获取视频详情与播放列表。
- `searchContent($key, $quick, $pg)`：搜索视频。
- `playerContent($flag, $id, $vipFlags)`：解析真实播放链接。

1.2 文件命名与目录规范

- **源文件命名**：统一使用 `.dz.php` 后缀（注意包含空格），例如 `果果.dz.php`。对于特定类型，建议增加标识：小说使用 `[书]`，漫画使用 `[画]`，例如 `七猫小说.dz[书].php`。
- **系统文件排除**：`config.php` 会自动忽略以下文件：
 - 系统文件 (`index.php`, `test_runner.php` 等)
 - 以 `_` 开头的文件 (如 `_backup.php`)
 - `config` 开头的文件
 - `lib` 目录下的文件

1.3 测试工具 (`test_runner.php`)

用于本地验证源的接口功能。

用法：

```
php test_runner.php "e:\php_work\php\荐片影视.dz.php"
```

(注意：由于文件名包含空格，命令行中路径建议加引号)

测试流程：

1. **首页测试**：检查分类是否获取成功，筛选条件是否解析。
2. **分类测试**：选取第一个分类，获取第一页数据，检查 `vod_id` 和 `vod_name`。

3. **详情测试**: 使用分类接口返回的 `vod_id`，检查详情信息及播放列表解析。
4. **搜索测试**: 使用分类接口获取的名称进行搜索验证。
5. **播放测试**: 选取第一个播放源，尝试解析播放链接。

2. 开发最佳实践

2.1 分页标准化 (`$this->pageResult`)

不要手动拼接复杂的 JSON 返回结构。使用框架内置的辅助方法 `$this->pageResult`。

推荐写法:

```
$videos = [];  
foreach ($items as $item) {  
    $videos[] = [  
        'vod_id' => $item['id'],  
        'vod_name' => $item['name'],  
        'vod_pic' => $item['pic'],  
        'vod_remarks' => $item['remarks']  
    ];  
}  
return $this->pageResult($videos, $page, $total, $pageSize);
```

2.2 数据传递技巧 (`vod_id` 组合)

有时 `categoryContent` 到 `detailContent` 需要传递额外参数 (如 `typeId`)，但 `vod_id` 只能是字符串。

技巧: 使用分隔符组合参数。

```
// 在 categoryContent 中  
'vod_id' => $id . '*' . $typeId  
  
// 在 detailContent 中  
$parts = explode('*', $ids[0]);  
$id = $parts[0];  
$typeId = $parts[1] ?? '';
```

2.3 HTML 解析 (DOMDocument)

处理 HTML 页面时，推荐使用 `DOMDocument` + `DOMXPath`，比正则更稳定。

IDE 爆红修复:

IDE 经常提示 `getAttribute` 方法不存在，因为 `DOMNode` 不一定是 `Element`。

正确写法:

```
$node = $xpath->query('//img')->item(0);  
if ($node instanceof DOMElement) { // 加上类型检查  
    $pic = $node->getAttribute('src');  
}
```

2.4 加密与解密 (JS -> PHP 转换)

遇到 JS 源使用了加密 (如 RSA, AES) , 需要用 PHP 的 `openssl` 扩展对应实现。

案例: RSA 分块解密 (参考 [零度影视 dz.php](#))

PHP 的 `openssl_private_decrypt` 有长度限制 (通常 117 或 128 字节) 。如果密文过长, 必须**分块**解密。

```
private function rsaDecrypt($data) {
    $decoded = base64_decode($data);
    $keyRes = openssl_pkey_get_private($this->privateKey);
    $details = openssl_pkey_get_details($keyRes);
    $keySize = ceil($details['bits'] / 8); // e.g., 128 bytes

    $result = '';
    $chunks = str_split($decoded, $keySize); // 按密钥长度分块

    foreach ($chunks as $chunk) {
        if (openssl_private_decrypt($chunk, $decrypted, $this->privateKey,
            OPENSSL_PKCS1_PADDING)) {
            $result .= $decrypted;
        }
    }
    return $result;
}
```

2.5 CURL Header 空值处理

在 PHP CURL 中, 如果需要发送一个值为空的 Header (如 `Authorization:`) , **不能使用** `"Header: "` (带空格) 或 `"Header:"` (不带值) , 这可能导致 Header 被忽略或发送错误的格式。

正确做法: 使用分号结尾。

```
$headers = [
    'Authorization;', // 发送 "Authorization:" 头, 值为空
    'User-Agent: ...'
];
```

此技巧在移植七猫小说时解决了个别接口 (如章节内容) 验签失败的问题。

2.6 HtmlParser 与 pd 函数的智能 UrlJoin

在使用 `pd()` 函数提取链接 (如图片 `src`、详情页 `href`) 时, 通常需要传入当前页面的 URL 作为 `baseUr1` 以便拼接相对路径。

手动传入 (推荐用于详情页):

```
$pic = $this->pd($html, 'img&&src', $currentUr1);
```

自动识别 (推荐用于列表页):

如果你的 Spider 类定义了 `const HOST` 或 `$HOST` 属性, `pd()` 函数在未传入 `baseUr1` 时会自动使用它作为基准。

```
class Spider extends BaseSpider {
    private const HOST = 'https://www.example.com';
    // ...
    // 这里不需要传 $url, 会自动用 HOST 拼接
    $pic = $this->pd($itemHtml, 'img&&src');
}
```

2.7 IDE 兼容性与反射技巧

在基类中访问子类的私有常量/属性（如 `$this->HOST`）时，直接访问会导致 IDE 报错（Undefined property）。

最佳实践: 使用 `ReflectionClass` 动态获取。

```
$ref = new ReflectionClass($this);
if ($ref->hasConstant('HOST')) {
    return $ref->getConstant('HOST');
}
```

这不仅消除了 IDE 警告，还支持了对 `private/protected` 属性的访问（需配合 `setAccessible(true)`，注意 PHP 8.1+ 已默认支持）。

3. HtmlParser 解析函数指南

为了与 JS 源（Hiker 规则）保持一致，我们在 `BaseSpider` 中内置了 `pdfa`，`pdfh`，`pd` 三个核心函数。它们支持 CSS 选择器风格的解析规则，并自动处理 DOM 操作。

3.1 规则语法 (Rule Syntax)

- **层级:** 使用 `&&` 分隔层级（在 XPath 中对应 `//`）。例如 `div.list&&ul&&li`。
- **属性/选项:** 规则的**最后一部分**指定要获取的内容。
 - `Text`: 获取纯文本（自动去除首尾空格和多余换行）。
 - `Html`: 获取元素的 OuterHTML。
 - `src`, `href`, `data-id`, ...: 获取指定属性值。
- **选择器:**
 - `tag`: 标签名，如 `div`, `a`, `img`。
 - `.class`: 类名，如 `.title`。
 - `#id`: ID，如 `#content`。
 - `:eq(n)`: 索引选择（0 起始）。`:eq(0)` 是第一个，`:eq(-1)` 是最后一个。
 - 组合: `div.item:eq(0)`。

3.2 pdfa (Parse DOM For Array)

用途: 解析列表，返回 HTML 字符串数组。通常用于 `categoryContent` 中解析视频列表。

签名:

```
protected function pdfa(string $html, string $rule): array
```

示例:

```
// 获取所有 ul 下的 li 元素的 HTML
$item = $this->pdfa($html, 'ul.list&&li');
foreach ($items as $itemHtml) {
    // 在循环中继续使用 pdfh/pd 解析具体字段
}
```

3.3 pdfh (Parse DOM For Html/Text)

用途: 解析单个节点的内容（文本、HTML 或属性）。

签名:

```
protected function pdfh(string $html, string $rule, string $baseUrl = ''):
string
```

示例:

```
// 获取标题文本
$title = $this->pdfh($itemHtml, '.title&&Text');

// 获取描述（可能包含 HTML 标签）
$desc = $this->pdfh($itemHtml, '.desc&&Html');

// 获取自定义属性
$dataId = $this->pdfh($itemHtml, 'a&&data-id');
```

3.4 pd (Parse DOM for Url)

用途: 解析链接（图片、跳转链接），并自动进行 URL 拼接（UrlJoin）。

签名:

```
protected function pd(string $html, string $rule, string $baseUrl = ''): string
```

特点:

- 等同于 pdfh + urlJoin。
- 如果规则末尾是属性（如 src, href），会自动基于 \$baseUrl 转换为绝对路径。
- 如果未传入 \$baseUrl，会自动尝试读取类常量 HOST。

示例:

```
// 自动拼接 HOST（假设类中定义了 const HOST）
$pic = $this->pd($itemHtml, 'img&&src');

// 手动指定 baseUrl（如详情页解析推荐列表）
$link = $this->pd($html, 'a.next&&href', 'https://m.example.com/list/');
```

4. 常见问题排查

- Q: 为什么搜不到结果？
 - A: 检查 searchContent 的 URL 参数是否正确编码。特别是中文关键词，部分站点需要 URL 编码，部分不需要。

- **Q: 详情页没有章节?**
 - A: 很多小说/漫画源的详情页接口 (`/detail`) 返回的信息不全, 通常需要额外调用章节列表接口 (`/chapter-list` 或类似)。务必抓包确认。
- **Q: 图片加载失败?**
 - A: 检查图片链接是否为相对路径。如果是, 请确保在 `pd()` 或手动处理时进行了完整的 URL 拼接。
- **Q: 验签失败?**
 - A: 仔细比对 Python/JS 源的签名逻辑。注意参数排序 (`ksort`)、空值处理、特殊字符编码差异。PHP 的 `md5` 输出默认是小写 hex。

5. Flutter 环境适配与 PHP 8.5+ 兼容性

在将 PHP 源部署到 Flutter 环境 (如 TVBox 及其变种) 并使用高版本 PHP (如 8.5.1) 时, 可能会遇到类型严格性导致的兼容问题。

5.1 核心报错: `type 'String' is not a subtype of type 'int' of 'index'`

现象:

本地 `test_runner.php` 测试一切正常, 但在 Flutter 端运行时报错, 提示 String 类型无法作为 List 的索引。

原因:

PHP 脚本执行结束后没有输出任何内容。

- `test_runner.php` 是手动实例化类并调用方法, 所以能拿到结果。
- Flutter 端通过 CLI 调用 PHP 脚本, 如果脚本末尾没有主动调用运行逻辑, 输出为空字符串。
- 适配层收到空字符串后, 可能默认处理为 `[]` (空 List)。后续逻辑尝试以 Map 方式 (如 `['class']`) 访问这个 List 时, 就会触发 Dart 的类型错误。

解决方案:

确保每个源文件末尾都包含自动运行指令:

```
// 必须在文件末尾加入此行
(new Spider())->run();
```

5.2 严格类型处理 (JSON 空对象)

现象:

PHP 的空数组 `[]` 在 `json_encode` 时默认为 `[]` (List)。如果在 PHP 8.5+ 环境下, 客户端期望的是 Map `{}` (Object), 可能会导致解析错误或类型不匹配。

解决方案:

对于明确应该是对象的字段 (如 `header`, `filters`, `ext`), 如果为空, 必须强制转换为 Object。

```
// 错误 (输出 [])
'header' => []

// 正确 (输出 {})
'header' => (object)[]
```

5.3 HTTPS 与 SSL 证书验证

现象：

在某些 Flutter 环境或 Android 设备上，cURL 请求 HTTPS 站点失败，无返回或报错。这是 because 系统证书库可能不完整或 curl 配置过严。

解决方案：

显式关闭 SSL 证书校验。BaseSpider 的 `fetch` 方法已默认处理，但在重写 `fetch` 或使用原生 cURL 时需注意：

```
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);  
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
```

5.4 健壮性建议

1. **JSON 解析容错**： `json_decode($str ?: '{}', true)`，避免对空字符串解析报错。
2. **空 ID 容错**：在 `detailContent` 或 `playerContent` 中，检查 ID 是否为空，避免向 API 发送非法请求导致崩溃。

本文档更新于 2026/01/25，基于 Trae IDE 协作环境。