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 (`spider.php`)

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

```php
$videos = [];
foreach ($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 `LingDu_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 播放链接解析 (二次解析)

有些源的播放链接不是直接的 mp4/m3u8，需要再次请求 API。
在 `playerContent` 中：

1. 解析传入的 `$id` (可能是 JSON 字符串)。
2. 请求解密接口获取初步 URL。
3. 如果需要，进行二次请求（如 `analysisMovieUrl`）获取最终直链。

---

3. JS 源转 PHP 流程

1. **分析 JS 逻辑**:
   - 寻找网络请求部分 (`req`, `request`, `fetch`)。
   - 分析 Headers 构造（特别是 `token`, `deviceId`, `sign` 等字段）。
   - 识别加密算法（搜索 `RSA`, `AES`, `MD5`, `Base64`）。
2. **PHP 实现**:
   - **Headers**: 复制 User-Agent 等固定头，动态生成 Token。
   - **Crypto**: 将 JS 的 crypto-js 逻辑映射到 PHP `openssl_*` 或 `hash_*` 函数。
   - **Random**: JS `Math.random()` -> PHP `mt_rand() / mt_getrandmax()`.
3. **接口映射**:
   - JS `home()` -> PHP `homeContent()`
   - JS `category()` -> PHP `categoryContent()`
   - JS `detail()` -> PHP `detailContent()`
   - JS `play()` -> PHP `playerContent()`
   - JS `search()` -> PHP `searchContent()`

---

4. 调试常见问题

问题现象	可能原因	解决方案
`test_runner` 报错 "Class 'Spider' not found"	文件名命名问题或未定义类	检查类名是否为 `Spider`，文件名是否正确。
`vod_play_url` 为空	解密失败或解析逻辑错误	检查 RSA 密钥格式；检查是否需要分块解密；打印中间变量调试。
IDE 提示 `Call to undefined method`	DOM 操作未做类型检查	添加 `instanceof DOMElement` 判断。
返回数据结构臃肿	手动构建了复杂的 JSON	改用 `$this->pageResult` 简化代码。
接口超时或 403	Headers 缺失或 Token 过期	检查 `getHeaders()` 方法，确保 `deviceId`/`token` 生成逻辑正确。

---

5. 经验沉淀 (Core Memory)

- **腾腾.php**: 修复了 `DOMElement` 类型检查缺失导致的 IDE 报错。
- **yjm 系列**: 将复杂的分类/搜索返回重构为 `$this->pageResult`。

- **零度影视 (LingDu_DZ)**:
  - 实现了完整的 JS -> PHP 转换。
  - **关键点**: `deviceId` 自动生成、RSA 大数据分块解密、播放链接二次解析。

本文档由 Trae AI 生成，用于记录开发经验以供后续快速恢复上下文。