Rhymix 운영 메모 · 개정판 v2

APCu 캐시와 CLI/FPM 일관성 문제 정리

게시글 삭제 예제, signed HTTP clear 요청 동작, `apc.enable_cli=0`의 한계, Memcached/Redis 선택 기준, 그리고 HTTP 우회 실패 시 복구 방법

핵심 결론

APCu는 빠르지만 PHP-FPM과 PHP-CLI 사이에서 하나의 외부 저장소처럼 동작하지 않습니다. CLI/cron이 Rhymix 데이터를 수정한다면 Memcached 또는 Redis가 가장 단순하고 안전합니다. APCu를 유지한다면 Rhymix 컨트롤러/모델 경로를 사용하고, signed HTTP clear 요청이 실제로 FPM까지 도달하는지 확인해야 합니다.

작성 기준일: 2026-05-21

기준: Rhymix master branch 및 PHP 공식 문서

0. 문서 목적과 빠른 결론

이 문서는 Rhymix에서 APCu object cache를 사용할 때, PHP-CLI 작업과 PHP-FPM 웹 요청 사이에서 캐시 일관성 문제가 어떻게 생기는지 설명합니다. 특히 게시글 삭제를 예로 들어, 정상 컨트롤러 메서드를 사용했을 때의 장점과 signed HTTP clear 요청이 실패했을 때 벌어질 수 있는 상황을 정리합니다.

> **한 줄 결론:** `apc.enable_cli=0`은 CLI APCu 사용 여부에 관한 설정일 뿐, FPM 쪽 APCu에 남아 있는 stale cache를 지워 주는 해결책이 아닙니다.

질문	정리
APCu를 써도 되나?	웹 요청 중심이고 CLI/cron이 데이터를 거의 수정하지 않는 단순 환경이면 가능. 다만 CLI 작업이 있다면 FPM 캐시 무효화가 실제로 되는지 검증해야 합니다.
Rhymix 정상 메서드를 쓰면?	DB 직접 조작보다 훨씬 안전합니다. 삭제 로직, trigger, 관련 데이터 정리, 썸네일 삭제, 문서 캐시 삭제 호출을 함께 태웁니다.
그럼 HTTP 접속 없이 해결?	게시글 삭제 자체는 CLI 내부 메서드 호출입니다. 그러나 APCu 캐시 무효화는 Rhymix가 자기 사이트로 signed HTTP POST를 보내 FPM 쪽에서 지우게 합니다.
HTTP 우회가 실패하면?	DB에서는 글이 삭제되었는데 FPM APCu에는 삭제 전 document_item 캐시가 남아, 웹에서 삭제된 글이 잠시 보이거나 목록/본문 상태가 불일치할 수 있습니다.
Memcached/Redis가 왜 낫나?	FPM과 CLI가 같은 외부 캐시 서버를 바라보므로, CLI가 지운 키가 웹 요청에도 같은 저장소 기준으로 반영됩니다.

문서 구성

1 APCu에서 문제가 생기는 이유

2 Rhymix 코어의 APCu + CLI 대응 방식

3 signed HTTP clear 요청의 의미

4 CLI에서 정상 컨트롤러 메서드로 게시글 삭제하기

5 HTTP 우회 실패 시 벌어지는 일과 복구 방법

6 apc.enable_cli=0의 한계, Memcached/Redis 선택 기준

7 운영 체크리스트와 참고 소스

1. 왜 APCu에서 문제가 생기나

APCu는 PHP 실행 환경 가까이에 붙어 있는 in-memory object cache입니다. 반면 Memcached/Redis는 별도 서버 또는 데몬으로 동작하는 외부 캐시 저장소입니다. 이 차이 때문에 PHP-FPM과 PHP-CLI가 캐시를 보는 방식이 달라집니다.

APCu 사용 시

PHP-FPM workers

FPM 쪽 APCu 저장 영역

PHP-CLI script

CLI APCu 또는 disabled. FPM 쪽 APCu와 같다고 보면 위험합니다.

Memcached/Redis 사용 시

PHP-FPM workers

외부 캐시 데몬
memcached / redis

PHP-CLI script

FPM과 CLI가 같은 캐시 서버를 바라봅니다.

게시글 삭제 예시: CLI에서 게시글 DB row를 삭제했는데 FPM 쪽 APCu에 문서 캐시가 남아 있으면, 웹 요청은 삭제 전 내용을 계속 볼 수 있습니다. 문제가 되는 것은 “DB 삭제” 자체가 아니라 “FPM 캐시 무효화가 같이 되지 않는 것”입니다.

APCu를 사용할 때 기억할 점

- 웹 요청 안에서 생성된 APCu 캐시는 웹 요청을 처리하는 FPM 환경에서만 안전하게 지울 수 있다고 보는 것이 실무적으로 안전합니다.
- CLI에서 `apcu_delete()`를 호출해도 FPM 쪽 APCu 캐시가 지워진다고 기대하면 안 됩니다.
- CLI/cron/queue가 Rhymix 데이터를 수정한다면, 단일 서버라도 캐시 일관성 검증이 필요합니다.

2. Rhymix 코어의 APCu + CLI 대응 방식

Rhymix의 `Rhymix\Framework\Cache`는 캐시 드라이버가 APC이고 현재 실행 환경이 CLI인 경우를 특별 취급합니다. `delete()`, `clearGroup()`, `clearAll()`에서 `sendClearRequest()`를 호출해 웹 요청을 보내는 구조입니다.

Rhymix 호출	APCu + CLI일 때 추가 동작	의미
`Cache::delete($key)`	`sendClearRequest([$key])`	개별 키 삭제 요청을 FPM 쪽으로 전달
`Cache::clearGroup($group)`	`sendClearRequest([$group . ':*'])`	그룹 버전 증가/무효화 요청을 FPM 쪽으로 전달
`Cache::clearAll()`	`sendClearRequest(['*'])`	전체 캐시 삭제 요청을 FPM 쪽으로 전달

```
// Rhymix\Framework\Cache::sendClearRequest()의 핵심 형태
$keystring = implode('¦', $keys);
$signature = Security::createSignature($keystring);

HTTP::post(Context::getRequestUri(), [
    'module' => 'module',
    'act' => 'procModuleClearCache',
    'keys' => $keys,
    'signature' => $signature,
]);
```

CLI 스크립트에서 Rhymix 로딩

↓

DocumentController::deleteDocument() 호출

↓

문서 삭제 및 관련 데이터 정리

↓

Cache::delete() 또는 clearGroup() 호출

↓

APCu + CLI 조건이면 sendClearRequest()

↓

signed HTTP POST가 PHP-FPM으로 도착

↓

FPM 요청 안에서 APCu key 삭제

주의: 이 흐름은 “게시글 삭제 자체를 HTTP로 호출한다”는 뜻이 아닙니다. 삭제 자체는 CLI 내부 메서드 호출이고, 캐시 삭제 요청만 HTTP로 우회됩니다.

3. signed HTTP clear 요청을 자세히 풀어보기

3.1 “signed”의 의미

signed는 로그인 세션이 있다는 뜻이 아닙니다. 삭제할 캐시 키 목록을 문자열로 만들고, Rhymix 내부 authentication key 기반의 서명을 붙인다는 뜻입니다. FPM 쪽 `procModuleClearCache`는 `keys`와 `signature`를 검증한 뒤 캐시 삭제를 수행합니다.

항목	설명
요청 목적지	`Context::getRequestUri()`가 가리키는 현재 사이트 URL
POST 파라미터	`module=module`, `act=procModuleClearCache`, `keys[]=...`, `signature=...`
CSRF 토큰	CLI에는 웹 세션/CSRF 토큰이 없으므로 일반 CSRF 대신 signature 검증을 사용
무한 루프 방지	FPM 요청에서는 `PHP_SAPI`가 `cli`가 아니므로 `sendClearRequest()`가 다시 호출되지 않음

3.2 로그에서 왜 잘 안 보이나

`sendClearRequest()`는 `act=procModuleClearCache`를 query string이 아니라 POST body로 보냅니다. 그래서 일반 access log에는 보통 `POST /` 또는 `POST /index.php` 정도만 보이고, act 값은 기본 로그에 보이지 않을 수 있습니다.

> **실패 가능 지점:** CLI 환경에서 자기 사이트 URL을 모르는 경우, HTTPS 인증서 오류, hosts/DNS 문제, 방화벽, reverse proxy, basic auth, maintenance mode, 잘못된 `request_uri` 설정 등이 있으면 HTTP clear 요청이 FPM까지 도달하지 못할 수 있습니다.

3.3 이 방식의 한계

- HTTP 요청이 실제로 성공해야 합니다. 실패해도 CLI 스크립트가 이를 확실히 감지하지 못할 수 있습니다.
- 여러 서버/여러 FPM pool에서는 요청을 처리한 FPM/APCu만 정리되고 다른 환경의 APCu가 남을 수 있습니다.
- 직접 DB만 수정하면 `Cache::delete()` 자체가 호출되지 않아 이 우회 경로도 실행되지 않습니다.

4. CLI에서 정상 컨트롤러 메서드로 게시글 삭제하기

정상 메서드를 쓴다는 말은 CLI에서 HTTP로 게시글 삭제 URL을 호출하라는 뜻이 아닙니다. CLI 스크립트가 Rhymix를 부트스트랩한 뒤, 코어 컨트롤러 메서드를 PHP 객체로 직접 호출하라는 뜻입니다.

피해야 할 방식

document 테이블을 직접 DELETE하거나 `document.deleteDocument` query만 실행하면 trigger, 관련 정리, 캐시 삭제 경로가 빠질 수 있습니다.

권장 방식

`DocumentController::deleteDocument($document_srl, true)`처럼 Rhymix 코어 삭제 경로를 호출합니다.

```
// 피해야 할 방식: DB만 직접 건드림
$args = new stdClass();
$args->document_srl = 12345;
executeQuery('document.deleteDocument', $args);

// 또는 더 위험한 raw SQL
DELETE FROM rx_documents WHERE document_srl = 12345;
```

```php
#!/usr/bin/env php
<?php
$root = '/var/www/rhymix';
chdir($root);

// APCu 사용 시 중요: Context::getRequestUri()가 올바른 URL을
// 만들 수 있도록 CLI에서도 서버 정보를 최소한 세팅합니다.
$_SERVER['HTTP_HOST'] = 'example.com';
$_SERVER['SERVER_NAME'] = 'example.com';
$_SERVER['SERVER_PORT'] = 443;
$_SERVER['HTTPS'] = 'on';
$_SERVER['REQUEST_METHOD'] = 'GET';
$_SERVER['REQUEST_URI'] = '/';
$_SERVER['SCRIPT_NAME'] = '/index.php';

require_once $root . '/common/autoload.php';
Context::init();

$document_srl = (int)($argv[1] ?? 0);
$oDocumentController = DocumentController::getInstance();

// 두 번째 인자 true는 권한 검사를 우회합니다.
// 서버 관리자 전용 배치 작업에서만 사용하세요.
$output = $oDocumentController->deleteDocument($document_srl, true);

if ($output instanceof BaseObject && !$output->toBool()) {
    fwrite(STDERR, $output->getMessage() . PHP_EOL);
    exit(1);
}

Context::close();
echo "deleted: {$document_srl}" . PHP_EOL;
```

> **이 방식의 장점:** 문서 삭제 전/후 trigger, 문서 존재/권한 확인, DB 삭제, alias/history 정리, 카테고리 카운트 갱신, after trigger, 썸네일 삭제, `clearDocumentCache()` 호출까지 코어 경로를 태울 수 있습니다.

휴지통 처리까지 웹 UI와 같게 하고 싶다면

게시판 UI는 설정에 따라 휴지통 이동과 영구 삭제를 나눌 수 있습니다. CLI 배치에서 사용자 UI와 동일한 정책까지 재현하려면 board 설정, 댓글 수 제한, 관리자 글 보호, 등록일 기준 보호 등을 별도로 고려해야 합니다. 단순 관리자 배치 작업이면 `deleteDocument($srl, true)`가 실용적이고, 사용자 액션과 같은 정책이 필요하면 게시판 레벨 조건을 함께 구현하는 편이 안전합니다.

5. HTTP 우회 실패 시 벌어지는 일: 게시글 삭제 예제

이 섹션이 이번 개정판의 추가 내용입니다. APCu를 유지하면서 CLI에서 게시글을 삭제했는데, signed HTTP clear 요청이 FPM까지 도달하지 못하면 어떤 상태가 되는지 단계별로 설명합니다.

> **핵심:** 게시글 삭제 DB 처리는 성공했는데 FPM 쪽 APCu에 삭제 전 `document_item` 캐시가 남아 있으면, 웹 사용자는 삭제된 글을 잠시 볼 수 있습니다. 목록에서는 사라졌는데 직접 URL에서는 보이는 식의 불일치도 가능합니다.

5.1 전제 상황

- 웹 사용자가 이미 게시글 `12345`를 조회해서 FPM APCu에 `document_item:...12345` 캐시가 만들어져 있습니다.
- 관리자 CLI 스크립트가 `DocumentController::deleteDocument(12345, true)`를 호출합니다.
- Rhymix는 `clearDocumentCache(12345)`를 통해 문서 캐시를 삭제하려고 합니다.
- 하지만 CLI → 사이트 URL HTTP POST가 SSL, 방화벽, hosts, basic auth, reverse proxy, maintenance mode 등의 이유로 실패합니다.

5.2 실제 상태가 어떻게 갈라지나

영역	상태	사용자가 볼 수 있는 결과
DB	게시글 `12345` row는 삭제됨	목록처럼 DB를 다시 조회하는 화면에서는 글이 사라질 수 있음
부가 데이터	alias/history 정리, 카테고리 카운트 갱신, 썸네일 삭제 등이 이미 수행될 수 있음	일부 화면에서는 글은 보이는데 썸네일/부가 정보는 깨져 보이는 불일치 가능
CLI APCu	`apc.enable_cli=0`이면 의미가 작고, 켜져 있어도 FPM APCu와 같지 않음	웹 사용자에게 직접 도움이 되지 않음
FPM APCu	삭제 전 `document_item:...12345`가 남아 있을 수 있음	직접 URL에서 삭제 전 제목/본문/작성자 등이 보일 수 있음

5.3 흐름도

1. FPM APCu에 document_item:...12345 캐시가 이미 존재

↓

2. CLI에서 Rhymix 정상 메서드로 게시글 12345 삭제

↓

3. DB row 삭제, 관련 데이터 정리, 썸네일 삭제 등은 수행됨

↓

4. clearDocumentCache(12345) → Cache::delete(...) 호출

↓

5. APCu + CLI라서 signed HTTP clear 요청을 보내지만 접속 실패

↓

6. FPM APCu의 오래된 document_item 캐시가 남음

↓

7. 웹 직접 URL에서 삭제 전 글이 보이거나, 목록/본문 상태가 불일치

5.4 왜 CLI 스크립트는 실패를 모를 수 있나

Rhymix의 `sendClearRequest()` 는 내부적으로 HTTP POST를 호출하지만, 그 응답을 엄격하게 검사해 삭제 실패로 되돌리는 구조로 보기 어렵습니다. 따라서 CLI 로그에는 "게시글 삭제 성공"으로 남았는데, 실제 FPM APCu에는 stale cache가 남는 상황이 가능합니다.

> **운영 증상 예:** "CLI 삭제 성공 → DB에는 글 없음 → 게시판 목록에는 글 없음 → 그런데 /board/12345 직접 접속 시 삭제 전 글이 보임." 이 증상이 나오면 FPM APCu stale cache 또는 여러 FPM pool/server 간 캐시 불일치를 의심해야 합니다.

6. HTTP 우회 실패 시 복구와 Rhymix 캐시 초기화

6.1 Rhymix 관리자 "캐시파일 재생성"의 의미

Rhymix 관리자에는 캐시파일 재생성 기능이 있습니다. 현재 코어 기준으로 `procAdminRecompileCacheFile()`는 `files/cache` 디렉터리를 비우거나 이동시키고, 모듈별 `recompileCache()`를 호출하며, object cache 드라이버가 `file`, `sqlite`, `dummy`가 아닐 경우 `Cache::clearAll()`도 호출합니다. APCu 드라이버의 `clear()`는 `apcu_clear_cache()`를 호출합니다.

관리자 캐시파일 재생성에서 하는 일	APCu stale cache에 미치는 영향
`files/cache` 정리 또는 새 디렉터리 생성	파일 캐시/컴파일 캐시 계열 정리
모듈별 `recompileCache()` 호출	모듈이 가진 캐시 재구성 로직 실행
object cache가 file/sqlite/dummy가 아니면 `Cache::clearAll()`	APCu 사용 시 웹/FPM 요청 안에서 전체 object cache 삭제 가능
`opcache_reset()` 가능 시 호출	PHP opcode cache 관련 문제까지 함께 완화 가능

> **정확한 표현:** "전체 캐시 재생성"은 모든 게시글 캐시를 미리 다시 만들어 둔다는 뜻이 아닙니다. 대부분은 "기존 캐시 삭제 → 다음 웹 요청에서 필요한 항목만 DB에서 다시 읽고 캐시 재생성" 흐름입니다. 이미 DB에서 삭제된 글이라면 캐시 초기화 후에는 더 이상 삭제 전 document_item을 재사용하지 않습니다.

6.2 복구 순서

1. **브라우저로 Rhymix 관리자에 접속해서 캐시파일 재생성 실행.** CLI가 아니라 웹/FPM 요청으로 실행하는 것이 중요합니다.
2. **삭제된 글 직접 URL을 재조회.** 더 이상 글이 보이지 않는지 확인합니다.
3. **여전히 보이면 PHP-FPM restart/reload.** APCu는 FPM 환경의 메모리 캐시이므로, FPM 재시작은 강제 초기화 수단입니다. 운영에서는 restart가 가장 확실합니다.
4. **여러 FPM pool/server라면 각 환경 확인.** APCu는 Memcached/Redis처럼 중앙 저장소가 아니므로, 다른 pool/server의 APCu가 남아 있을 수 있습니다.

6.3 CLI에서 Cache::clearAll()만 호출하면 되나?

> **HTTP 우회가 실패하는 환경에서는 CLI의 `Cache::clearAll()`도 충분하지 않을 수 있습니다.** APCu + CLI 조건에서는 전체 삭제도 결국 signed HTTP clear 요청으로 FPM에 전달되어야 하기 때문입니다. 그 경로가 막혀 있다면 CLI에서 전체 삭제를 호출해도 FPM APCu가 그대로 남을 수 있습니다.

6.4 운영에서 확인할 것

- CLI 서버에서 `curl -I https://example.com/` 또는 내부 health URL 접속이 되는지 확인합니다.
- CLI 삭제 후 웹서버 로그에 해당 시각의 POST 요청이 들어오는지 확인합니다. 단, `act=procModuleClearCache`는 POST body에 있으므로 일반 access log에 안 보일 수 있습니다.
- basic auth, 점검 모드, IP 제한, 프록시 헤더, Host 헤더, HTTPS 인증서 검증 문제를 점검합니다.
- staging에서 "웹 조회로 캐시 생성 → CLI 삭제 → 직접 URL 재조회" 테스트를 반복해 stale cache가 재현되는지 확인합니다.

7. apc.enable_cli=0으로 해결 가능한가

해결책으로 보기는 어렵습니다. PHP 공식 문서 기준으로 `apc.enable_cli`의 기본값은 `0`이고, CLI에서 APC를 활성화하는 것은 주로 testing/debugging 용도입니다. 이 설정은 CLI APCu 사용 여부에 관한 설정이지, FPM APCu를 CLI가 직접 지우도록 만들어 주는 설정이 아닙니다.

설정	효과	stale FPM APCu 해결 여부
`apc.enable_cli=0`	CLI에서 APCu 사용을 비활성화하는 일반적인 설정	아니오. FPM APCu에 남은 캐시를 지우지 못함
`apc.enable_cli=1`	CLI에서도 APCu 함수를 사용할 수 있게 함	아니오. FPM과 하나의 캐시 저장소를 공유하게 만드는 설정이 아님

운영 판단: `apc.enable_cli=0`은 유지할 수 있습니다. 그러나 그것만으로 “CLI에서 바꾼 내용이 웹 APCu 캐시에 반영된다”는 보장은 생기지 않습니다.

8. Memcached 또는 Redis를 쓰는 이유

Memcached와 Redis는 PHP-FPM과 PHP-CLI가 함께 접근하는 외부 캐시 저장소입니다. 따라서 같은 서버 1대라도 CLI와 FPM이 동일한 캐시 서버를 바라보게 만들 수 있습니다. CLI에서 지운 키는 웹 요청에도 같은 저장소 기준으로 반영됩니다.

항목	APCu	Memcached/Redis
저장 위치	PHP SAPI/FPM pool 가까이의 in-memory cache	별도 서버/데몬 형태의 외부 캐시
FPM-CLI 공유	직접 공유된다고 보기 어렵고 우회 무효화 필요	같은 캐시 서버에 접속하므로 구조가 단순
성능	단일 서버 환경에서 빠름	네트워크/소켓 비용이 있지만 일관성 관리가 쉬움
여러 서버	서버별 APCu가 따로 생김	공용 캐시 서버로 묶기 쉬움
추천 상황	웹 요청 중심의 단순 환경	CLI/cron/queue/다중 서버 환경

추천: Rhymix 운영에서 CLI/cron으로 문서, 회원, 설정, 포인트, 캐시 의존 데이터를 수정한다면 Memcached 또는 Redis를 우선 권장합니다. “APCu가 빠르다”보다 “캐시 일관성 문제가 운영 중 재현되지 않는가”가 더 중요합니다.

9. APCu를 계속 쓸 때 점검 체크리스트

9.1 설정 확인

```
php -i | grep -E 'apc.enable_cli|apc.enabled'

php -r 'var_dump(
    PHP_SAPI,
    function_exists("apcu_exists"),
    ini_get("apc.enable_cli")
);'
```

9.2 캐시 clear 요청 확인

- CLI에서 Rhymix를 로딩할 때 `Context::getRequestUri()`가 실제 접속 가능한 사이트 URL을 만들 수 있는지 확인합니다.
- CLI 삭제 후 web server access log에 signed clear 요청이 들어오는지 확인합니다. POST body는 기본 access log에 보이지 않을 수 있습니다.
- HTTPS 인증서, hosts, 방화벽, reverse proxy, basic auth, maintenance mode가 요청을 막지 않는지 확인합니다.
- 가능하면 staging에서 "웹으로 문서 조회 → CLI 삭제 → 웹 재조회" 순서로 stale cache가 재현되는지 테스트합니다.

9.3 코드 작성 원칙

- raw SQL 또는 `executeQuery` 단독으로 핵심 데이터를 삭제/수정하지 않습니다.
- `DocumentController`, `ModuleController`, `MemberController` 등 Rhymix 코어 컨트롤러/모델의 공개 메서드를 우선 사용합니다.
- 직접 캐시를 지워야 한다면 `apcu_delete()`가 아니라 `Rhymix\Framework\Cache::delete()/clearGroup()/clearAll()`을 사용합니다.
- 서버 관리자 전용 CLI 스크립트에서 권한 우회 인자를 사용할 때는 파일 권한과 실행 권한을 엄격히 제한합니다.

10. 운영 선택 기준

상황	권장 선택	이유
웹 요청만 있고 CLI 작업이 거의 없음	APCu 가능	구조가 단순하고 캐시 무효화 경로가 대부분 웹 안에서 끝남
CLI/cron이 게시글, 설정, 회원 등을 수정함	Memcached/Redis 권장	FPM-CLI 캐시 일관성 문제가 줄어듦
PHP-FPM 서버가 여러 대	Memcached/Redis 강력 권장	APCu는 서버별로 따로 존재하므로 일관성 관리가 어려움
APCu를 유지해야 함	Rhymix 메서드 + HTTP clear 검증	삭제 로직과 캐시 무효화 로직을 빠뜨리지 않아야 함
HTTP 우회 실패가 의심됨	웹 관리자 캐시파일 재생성 또는 PHP-FPM 재시작	FPM 쪽 APCu를 실제로 비워야 stale cache가 제거됨
`apc.enable_cli=0`만 적용하고 싶음	불충분	CLI 사용 여부 설정일 뿐 FPM cache invalidation이 아님

최종 요약: 1) `apc.enable_cli=0`은 해결책이 아닙니다. 2) CLI에서 Rhymix 데이터를 바꿀 때는 코어 컨트롤러/모델 경로를 사용해야 합니다. 3) APCu에서는 캐시 clear가 signed HTTP 요청으로 FPM에 전달됩니다. 4) 그 HTTP 요청이 실패하면 게시물 삭제 후 stale cache가 남을 수 있습니다. 5) 운영 안정성을 우선하면 Memcached/Redis가 더 단순합니다.

11. 참고 소스

아래 소스는 이 문서 작성 시점의 확인 기준입니다. GitHub master branch는 이후 변경될 수 있으므로, 실제 운영 반영 전에는 사용 중인 Rhymix 버전의 동일 파일을 다시 확인하는 것이 좋습니다.

번호	소스	확인 내용	URL
[S1]	Rhymix common/framework/Cache.php	`Cache::delete()`, `clearGroup()`, `clearAll()`, `sendClearRequest()`의 APCu + CLI 처리	https://github.com/rhymix/rhymix/blob/master/common/framework/Cache.php
[S2]	Rhymix modules/module/module.controller.php	`procModuleClearCache`의 keys/signature 검증 및 캐시 삭제 처리	https://github.com/rhymix/rhymix/blob/master/modules/module/module.controller.php
[S3]	Rhymix modules/document/document.controller.php	문서 삭제, 휴지통 이동, `clearDocumentCache()` 경로	https://github.com/rhymix/rhymix/blob/master/modules/document/document.controller.php
[S4]	Rhymix modules/document/document.item.php	`document_item` 캐시 조회 및 캐시 hit 시 카운트 컬럼 재조회 흐름	https://github.com/rhymix/rhymix/blob/master/modules/document/document.item.php
[S5]	Rhymix admin CacheReset.php	관리자 캐시파일 재생성: `files/cache` 정리, 모듈별 recompile, object cache clear, opcache reset	https://github.com/rhymix/rhymix/blob/master/modules/admin/controllers/maintenance/CacheReset.php
[S6]	Rhymix APC cache driver	APCu 드라이버의 `apcu_fetch/store/delete/clear_cache` 사용	https://github.com/rhymix/rhymix/blob/master/common/framework/drivers/cache/apc.php
[S7]	PHP APCu Runtime Configuration	`apc.enable_cli` 기본값과 CLI 활성화 설명	https://www.php.net/manual/en/apcu.configuration.php
[S8]	PHP Memcached Manual	Memcached의 외부 memory object cache 성격	https://www.php.net/manual/en/book.memcached.php