ChatGPT

Rhymix CMS 전체 작동 원리와 게시판 모듈 실행 순서 분석 보고서

Executive summary

Rhymix는 단순한 "게시판 CMS"가 아니라, **레거시 코어 클래스와** `Rhymix\Framework` **계층이 공존하는 하이브리드 구조** 위에서 동작하는 PHP 기반 CMS이자 웹 프레임워크다. 최신 안정판 기준으로 공식 사이트는 **Rhymix 2.1.33**을 배포하고 있으며, 공식 표기상 최소 요구사항은 **PHP 7.4+ / MySQL 또는 MariaDB**다. 웹 요청은 `index.php`에서 시작하여 오토로더 로딩, `Context::init()`에 의한 요청 컨텍스트 초기화, `Router::parseURL()`에 의한 라우팅, `ModuleHandler`를 통한 모듈 인스턴스 생성 및 액션 실행, 마지막으로 `displayContent()`에 의한 응답 렌더링 순으로 흐른다. 이 과정에서 권한, 세션, 언어, 캐시 제어, CSRF 관련 옵션, 프런트엔드 자산 관리가 함께 결합된다. [1]

게시판 모듈은 외형상 하나의 기능처럼 보이지만, 내부적으로는 **보드 모듈이 "오케스트레이션 계층"**으로 동작하고 실제 문서 저장·수정은 `document` 모듈, 댓글은 `comment` 모듈, 첨부파일은 `file` 모듈이 맡는 구조다. 즉, `board.controller.php`는 직접 SQL을 다루기보다 `DocumentController::insertDocument()`, `CommentController::insertComment()` 같은 하위 모듈 API를 호출하여 상태를 조정하고, `board.view.php`는 `DocumentModel::getDocumentList()` 등으로 결과를 조합해 템플릿에 넘긴다. 이런 구조 덕분에 확장성은 높지만, 실제 성능과 보안은 보드 단독이 아니라 **라우터, 보안 필터, 문서/댓글/파일 하위 모듈, DB 추상화 계층**이 함께 보장한다는 점을 이해해야 한다. [2]

분석 기준과 자료 범위

이 보고서는 사용자 요청에 따라 **최신 안정 릴리스 기준**으로 작성했으며, 공식 홈페이지의 다운로드 안내를 기준으로 분석 대상 버전을 **Rhymix 2.1.33**으로 두었다. GitHub 저장소의 기본 브랜치는 `master`로 표시되고 있으며, 소스 코드 분석은 가급적 `2.1.33` **태그 경로**를 기준으로 했다. 환경은 공식 표기인 **PHP 7.4+ / MySQL 또는 MariaDB**를 따르는 일반적인 LAMP/LEMP 구성을 가정했다. 또한 공식 매뉴얼은 rewrite 설정과 nginx 예시를 제공하므로, 실제 URL 라우팅 구조를 제대로 쓰려면 웹서버 rewrite 구성이 전제된다고 보아야 한다. [3]

기준 항목	적용 기준	근거
안정 버전	Rhymix 2.1.33	[4]
기본 브랜치	`master`	[5]
공식 문서	Rhymix 매뉴얼, `rhymix-docs` API/레거시 문서	[6]
소스 기준	`rhymix/rhymix`의 `2.1.33` 태그 경로	[7]
가정 환경	PHP 7.4+, MySQL/MariaDB, Apache/nginx rewrite 활성	[8]

덧붙이면 Rhymix는 2.1 계열에서도 템플릿 시스템과 프레임워크 레이어의 현대화를 지속하고 있으며, 공식 매뉴얼은 템플릿 문법 v2를 2.1.8부터 프리뷰로, 2.2 이상에서는 신설 자료에 권장한다고 설명한다. 따라서 **본 보고서의 요청 흐름과 게시판 모듈 분석은 2.1.33 기준**, 템플릿 계층의 향후 변화 가능성은 별도 주의사항으로 보는 것이 정확하다. [9]

Rhymix 전체 아키텍처 개요

Rhymix의 핵심 특징은 `Rhymix\Framework` **계층과 XE 계열에서 이어진 레거시 클래스가 동시에 중요한 역할을 수행한다는 점** 이다. 공식 문서도 `Context`, `ModuleHandler`, `ModuleObject`, `FrontEndFileHandler`를 "적극 사용 중인 레거시 클래스"로 분류하고 있고, 반대로 `DB`, `Security`, `TemplateHandler`, `CacheHandler` 등은 `Rhymix\Framework` 클래스로 대체되었다고 설명한다. 즉, Rhymix의 실제 구조는 "완전한 신형 프레임워크"도 아니고 "전부 레거시"도 아닌 **혼합형 코어**다. [10]

핵심 컴포넌트	대표 클래스 및 파일 경로	역할
부트스트랩	`index.php`	오토로더 로딩, `Context::init()`, `ModuleHandler` 생성 및 실행, 종료 시 `Context::close()` 호출. [11]
컨텍스트 초기화	`classes/context/Context.class.php` / `Context::init()`	DB 설정 로딩, 요청 메서드 설정, `Router::parseURL()` 호출, 요청 인자 세팅, 업로드 정보 정리, 세션/언어/인증 초기화. [12]
라우터	`common/framework/Router.php` / `Router::parseURL()`	rewrite 레벨 판단, `mid` 또는 모듈 prefix 해석, 액션/변수 추출, route option (`check_csrf`, `enable_session`, `cache_control`, `is_indexable`) 주입. [13]
요청 객체	`common/framework/Request.php`	라우터가 반환하는 요청 표현체. `Context::getCurrentRequest()`로 접근 가능. [14]
모듈 생명주기 관리자	`classes/module/ModuleHandler.class.php`	모듈 초기화, 실행, 출력, 트리거 호출. 공식 문서는 `init()`, `procModule()`, `displayContent()`, `triggerCall()`을 핵심 메서드로 설명한다. [15]
모듈 MVC	예: `modules/board/board.view.php`, `board.controller.php`, `board.model.php`	각 모듈은 보통 `View`, `Controller`, `Model` 클래스로 역할을 분리한다. 게시판은 이 전형적인 구조를 그대로 따른다. [16]
렌더링	`DisplayHandler` 계열, `FrontEndFileHandler`, `Template`	HTML/JSON/XML/raw 응답 렌더링, CSS/JS 자산 정렬·중복 제거, 템플릿 실행. [17]
인증·권한	`Rhymix\Framework\Session`, `MemberController`, 모듈 grant, 객체별 `isGranted()` / `isAccessible()`	세션 기반 로그인 상태 초기화, 모듈 권한(grant) 체크, 문서·댓글 객체별 세부 권한 검사. [18]
DB 추상화	`Rhymix\Framework\DB`	prepared statement와 XML 정의 쿼리(`executeQuery`)를 모두 지원하며, 테이블 prefix 적용과 쿼리 결과 래핑을 담당한다. [19]

핵심 컴포넌트	대표 클래스 및 파일 경로	역할
캐시	`Rhymix\Framework\Cache` + 라우터 내부 정적 캐시	범용 키-값 캐시 제공, 라우터는 액션·route 정보를 정적 캐시에 유지한다. [20]
이벤트/훅	`ModuleHandler::triggerCall()`, `ModuleController::addTriggerFunction()`	before/after 트리거 기반 확장 포인트. 게시판은 익명 글 수정 보호에 `document.updateDocument` 트리거를 사용한다. [21]
보안 필터	`Security`, `HTMLFilter`, `FileContentFilter`, `FilenameFilter`	CSRF 검사 API, HTMLPurifier 기반 XSS 방어, 업로드 파일 내용 검사, 파일명/경로 정리 및 직접 다운로드 가능 여부 판정. [22]

이 구조에서 특히 중요한 점은 **"컨트롤러가 모든 것을 직접 처리하지 않는다"** 는 것이다. 예를 들어 게시판 컨트롤러는 글 등록 로직의 상위 조정자이며, 실제 영속화는 `DocumentController`가 맡고, 댓글은 `CommentController`, 첨부파일은 `file` 모듈이 맡는다. 따라서 Rhymix를 분석할 때는 "board만 읽는 것"으로 충분하지 않고, **board ↔ document/comment/file ↔ DB**의 협업 구조를 보는 것이 필수다. [23]

보드 모듈 안에서도 훅 시스템은 실제로 사용된다. 익명 모드에서 기존 익명 정보가 수정 시 유지되도록 게시판 컨트롤러는 `ModuleController::addTriggerFunction('document.updateDocument', 'before', ...)`를 동적으로 추가한다. 즉, 트리거 시스템은 서드파티 확장 전용이 아니라 **코어 모듈 내부 구현에서도 사용되는 1급 메커니즘** 이다. [24]

요청 처리 흐름과 시퀀스

Rhymix의 공통 요청 처리 흐름은 다음과 같이 요약할 수 있다. 웹서버가 들어온 URI를 `index.php`로 전달하면, 엔트리 파일은 오토로더를 포함한 뒤 `Context::init()`으로 환경을 구성하고, `ModuleHandler`를 생성하여 초기화(`init()`), 모듈 실행(`procModule()`), 응답 출력(`displayContent()`)의 세 단계를 수행한다. 이 흐름 자체는 매우 짧지만, 실제 복잡성은 `Context`와 `Router`, 그리고 모듈 액션 메타데이터(`module.xml`)에 숨어 있다. [25]

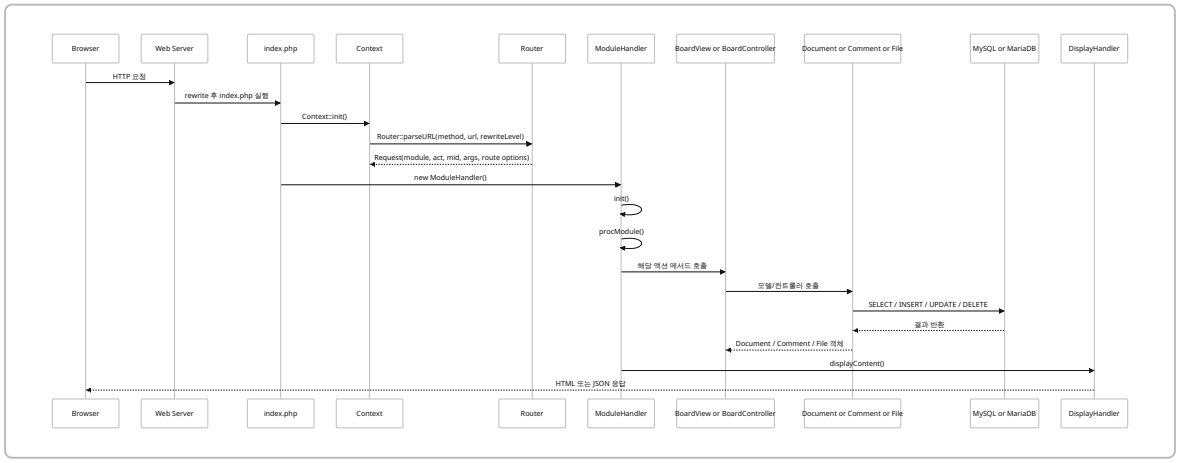
단계	주요 클래스/함수	최신 릴리스 기준 대표 파일 경로	설명
엔트리포인트 진입	`require '/common/autoload.php'`	`index.php`	모든 요청의 시작점. 오토로더를 불러온다. [11]
컨텍스트 초기화	`Context::init()`	`classes/context/Context.class.php`	DB 설정 로딩, 전역 변수 검사, 요청 메서드 설정, 라우터 호출, 업로드 정보 세팅, 세션/언어/인증 초기화. [12]

단계	주요 클래스/함수	최신 릴리스 기준 대표 파일 경로	설명
URL 해석	`Router::getRewriteLevel()`, `Router::parseURL()`	`common/framework/Router.php`	rewrite 수준(0/1/2) 판정 후, `mid` / module prefix와 route를 해석하여 `Request` 객체에 `module`, `act`, `mid`, args를 채운다. [26]
액션 메타 주입	`Router::_fillActionProperties()`	`common/framework/Router.php`	`check_csrf`, `is_indexable`, `enable_session`, `cache_control` 같은 route option을 요청 객체에 채운다. [27]
모듈 핸들러 생성	`new ModuleHandler()`	`index.php`	실제 모듈 실행 수명주기를 담당할 객체 생성. [11]
모듈 초기화	`ModuleHandler::init()`	`classes/module/ModuleHandler.class.php`	현재 요청에 맞는 모듈과 액션 준비. 공식 문서상 lifecycle의 첫 단계다. [15]
모듈 실행	`ModuleHandler::procModule()`	`classes/module/ModuleHandler.class.php`	모듈 인스턴스를 만들고 액션 메서드를 실행한다. [15]
모델/컨트롤러/뷰 실행	예: `BoardView::dispBoardContent()`, `BoardController::procBoardInsertDocument()`	`modules/board/*.php`	요청 종류에 따라 목록/보기/쓰기/댓글 처리로 분기된다. [28]
데이터 접근	`DocumentModel`, `DocumentController`, `CommentController`, `DB`	`modules/document/*`, `modules/comment/*`, `common/framework/DB*`	조회는 model, 변경은 controller, 실제 DB I/O는 DB 추상화 계층이 수행한다. [29]
응답 렌더링	`ModuleHandler::displayContent()` / `DisplayHandler` / `FrontEndFileHandler`	`classes/module/*`, `classes/display/*`	템플릿을 통해 HTML/JSON/XML/raw 응답을 생성하고 CSS/JS 자산을 정리한다. [30]

단계	주요 클래스/함수	최신 릴리스 기준 대표 파일 경로	설명
종료 처리	`Context::close()`	`classes/context/Context.class.php`	디버그 정보 저장, 세션 종료 등 자원 정리. [31]

Rhymix의 라우터는 “단순히 URI를 해석하는 수준”을 넘어서, **모듈 액션 정의를 읽어 요청 객체에 보안/세션/인덱싱 옵션까지 주입** 한다. `parseURL()` 은 우선 query string을 분리하고 UTF-8 검사를 수행한 뒤, rewrite level 2에서는 `/mid/나머지경로` 형태를 기준으로 prefix를 해석한다. 그 다음 `module.xml` 에서 읽어온 라우트 정보와 현재 HTTP method를 대조해 액션을 확정하고, 마지막에 `_fillActionProperties()` 가 `check_csrf` , `enable_session` , `cache_control` 등의 옵션을 세팅한다. 이 설계 때문에 보안과 캐시 정책이 “컨트롤러 코드만”이 아니라 **액션 메타데이터 수준** 에서도 관리된다. [32]

공식 매뉴얼의 “라우터 사용법” 문서도 Rhymix의 짧은주소가 기본적으로 `http://example.com/mid/나머지/부분` 형태를 취하며, 각 모듈이 `module.xml` 의 `route` 속성으로 후속 경로를 선언한다고 설명한다. 게시판 모듈이 대표 사례로 제시되며, 실제 `board/conf/module.xml` 도 `dispBoardContent` , `dispBoardWrite` , 댓글 관련 액션들에 route를 매핑하고 있다. [33]



위 시퀀스에서 눈여겨볼 부분은 **Context 단계에서 이미 Request 객체와 업로드 정보, 세션, 인증 상태가 만들어지고**, 이후 `ModuleHandler` 는 “이미 정규화된 요청”을 받아 액션 실행에 집중한다는 점이다. 따라서 Rhymix에서 디버깅을 할 때는 “컨트롤러 이전 단계”인 `Context::init()` 과 `Router::parseURL()` 을 함께 봐야 문제를 빨리 찾을 수 있다. [34]

게시판 모듈 구조와 데이터 모델

게시판 모듈의 디렉터리는 전형적인 Rhymix 모듈 구조를 보여준다. `conf/` , `queries/` , `ruleset/` , `tpl/` 폴더와 함께, 실동작 클래스는 `board.class.php` , `board.view.php` , `board.controller.php` , `board.model.php` , 그리고 관리자용 `board.admin.*` 파일들로 나뉘어 있다. 즉, **설정 메타데이터와 실행 클래스를 분리하는 구조** 다. [35]

파일/폴더	역할	GitHub
`modules/board/conf/module.xml`	액션, permission, route, ruleset 선언	[36]
`modules/board/board.class.php`	게시판 공통 기본 클래스, 기본 검색·정렬 옵션 보유	[37]

파일/폴더	역할	GitHub
`modules/board/board.view.php`	목록/보기/쓰기/댓글·첨부 표시 등 읽기 흐름	38
`modules/board/board.controller.php`	글/댓글 등록·수정·삭제, 비밀번호 인증, 추천 처리	39
`modules/board/board.model.php`	모듈 설정 기본값과 보정 로직	40
`modules/board/tpl/`	기본 스킨 템플릿	41
`modules/board/queries/`	XML 정의 쿼리 저장소	42

구조적으로 보면 게시판 모듈은 `Board` 기본 클래스 위에 `BoardView`, `BoardController`, `BoardModel`이 각각 상속하는 전형적인 모듈이다. `Board` 클래스는 기본 검색 옵션과 정렬 컬럼을 유지하는데, 검색 대상은 `title_content`, `title`, `content`, `comment`, `user_name`, `nick_name`, `user_id`, `regdate`, `tag` 등을 포함하고 정렬 대상은 `list_order`, `update_order`, `regdate`, `voted_count`, `readed_count`, `comment_count`, `title` 등이다. 즉, 게시판이 제공하는 검색/정렬 기능의 범위는 **보드 모듈 기본 클래스에 명시적으로 선언** 되어 있다. 37

게시판 모듈의 라우팅과 권한 메타데이터

`board/conf/module.xml`은 게시판 모듈의 실질적인 "계약서"다. `dispBoardContent`는 게시판의 인덱스 액션이며, `document_srl`, `category`, `page`를 route 변수로 받는다. `dispBoardWrite`는 `write` 또는 `$document_srl/edit` 경로와 연결되고, `procBoardInsertDocument`, `procBoardDeleteDocument`, `procBoardInsertComment`, `procBoardDeleteComment`는 각각 controller 액션으로 선언되어 있다. 또한 `write_document`, `write_comment`, `view`, `list` 같은 grant가 액션 단위로 걸려 있어 **초기 권한 필터링이 메타데이터 수준에서 1차로 수행** 된다. 43

데이터베이스 스키마

보드 모듈은 "자체 전용 게시글 테이블"을 따로 가지는 방식이 아니라, **문서/댓글/파일/모듈 인스턴스 테이블을 조합해서 동작** 한다. 게시판 하나를 식별하는 인스턴스 정보는 `modules` 테이블의 `module_srl`, `module`, `mid`로 관리되고, 실제 글 본문은 `documents`, 댓글은 `comments`, 첨부는 `files`에 저장된다. 44

테이블	대표 컬럼	용도
`modules`	`module_srl`, `module`, `mid`, `domain_srl`	게시판 인스턴스 식별. 어떤 `mid`가 어떤 모듈 인스턴스인지 결정한다. 45
`documents`	`document_srl`, `module_srl`, `category_srl`, `is_notice`, `title`, `content`, `readed_count`, `comment_count`, `uploaded_count`, `member_srl`, `nick_name`, `regdate`, `last_update`, `list_order`, `update_order`	게시글 본문 및 메타데이터 저장. 목록·정렬·조회수·댓글수·첨부수까지 한 테이블에서 관리한다. 46

테이블	대표 컬럼	용도
`comments`	`comment_srl`, `module_srl`, `document_srl`, `parent_srl`, `is_secret`, `content`, `voted_count`, `notify_message`, `password`, `user_name`	댓글과 대댓글 저장. `parent_srl`을 통해 트리/답글 관계를 표현한다. [47]
`files`	`file_srl`, `upload_target_srl`, `upload_target_type`, `module_srl`, `member_srl`, `download_count`, `direct_download`, `source_filename`, `uploaded_filename`, `file_size`	첨부파일 저장. 문서/댓글과의 연결은 `upload_target_srl`로 관리한다. [48]

이 스키마를 보면 게시판 목록 성능에 중요한 필드가 무엇인지도 자연스럽게 보인다. `documents`에는 `module_srl`, `category_srl`, `is_notice`, `readed_count`, `comment_count`, `list_order`, `update_order`, `regdate` 등에 인덱스가 붙어 있다. 이는 목록 조회가 보통 “게시판 인스턴스 + 정렬 컬럼 + 카테고리” 조합으로 이루어진다는 것을 보여준다. [46]

게시판 모듈의 권한, 검증, 보안 처리

게시판 읽기/쓰기는 **두 단계 권한 구조** 로 이루어진다. 먼저 `module.xml`의 액션 permission이 모듈 grant를 거르고, 그 다음 실제 문서/댓글 수준에서는 `$oDocument->isGranted()`, `$oDocument->isAccessible()`, `$comment->isGranted()` 같은 객체 수준 검사가 수행된다. 예를 들어 글 삭제는 `procBoardDeleteDocument()`에서 단순히 `write_document` grant만 보는 것이 아니라, 실제 대상 문서 객체를 가져와 존재 여부와 개별 권한까지 재검사한다. 댓글도 마찬가지로 대상 문서 접근성부터 확인한 뒤 처리한다. [49]

입력 검증은 보드 레벨과 프레임워크 레벨이 함께 담당한다. `BoardView::init()`은 `document_srl`, `comment_srl`, `mid`, `page`, `category`, `search_target`, `search_keyword` 같은 요청 변수를 `Security()->encodeHTML()`로 인코딩하여 템플릿 반사 출력 시 XSS 가능성을 줄인다. 글/댓글 등록 컨트롤러는 내용이 비었는지(`is_empty_html_content`), 길이가 너무 긴지, 본문에 과도한 data URL이 있는지, 카테고리가 필요한데 누락되지 않았는지 등을 검사한다. 프레임워크 차원에서는 `HTMLFilter`가 HTMLPurifier 기반으로 HTML 콘텐츠를 정제하고, `FileContentFilter`는 `Context` 초기화 과정에서 업로드된 파일 내용을 검사한다. [50]

CSRF 측면에서는 라우터가 액션 메타데이터를 해석할 때 `check_csrf` route option을 기본적으로 활성화하도록 설계되어 있고, 프레임워크는 `Security::checkCSRF()` API를 제공한다. 게시판의 controller 액션들(`procBoardInsertDocument`, `procBoardInsertComment`, `procBoardDeleteDocument` 등)은 별도 비활성화 선언이 없으므로, 설계상 CSRF 검사 대상이 되는 흐름으로 이해하는 것이 맞다. 다만 실제 검사 시점은 요청 라이프사이클의 공통 처리 계층에 있으므로, 보드 컨트롤러 안에서 직접 `checkCSRF()`를 호출하는 구조는 아니다. [51]

파일 업로드와 다운로드 정책도 흥미롭다. `FileContentFilter`는 확장자가 아니라 **실제 파일 내용** 을 검사하고, `FilenameFilter`는 위험한 문자와 경로 조작 요소를 정리한다. 또한 `FilenameFilter::isDirectDownload()` 문서는 이미지·멀티미디어처럼 직접 다운로드 가능한 파일은 `./files/attach/images/` 계열에 저장될 수 있고, 그 외 이진 파일은 `file` 모듈의 `procFileDownload` 액션을 통해 내려받도록 설계된다고 설명한다. 보드 뷰에서는 실제 첨부 목록을 `oDocument->getUploadedFiles()`로 가져와 화면에 표시한다. [52]

게시판 모듈 요청별 상세 순서

아래 표의 SQL은 **개념적 예시** 다. Rhymix는 실제로 `Rhymix\Framework\DB`의 prepared statement와 XML 정의 쿼리(`executeQuery`)를 함께 사용할 수 있으며, 실제 실행 SQL은 driver, prefix, query builder, 하위 모듈 구현에 따라 달라질 수 있다. 따라서 아래 SQL은 **테이블 스키마와 호출 체인을 바탕으로 재구성한 대표 예시** 로 이해하는 것이 정확하다. [53]

요청 시나리오	URL/액션	대표 호출 순서	권한/검증
글 목록 조회	`dispBoardContent` → 내부에서 `dispBoardContentList()`	`index.php` → `Context::init()` → `Router::parseURL()` → `ModuleHandler::procModule()` → `BoardView::dispBoardContent()` → `BoardView::dispBoardContentList()` → `DocumentModel::getDocumentList()` → 템플릿 렌더. [54]	`grant->access`, `grant->list` 확인. 카테고리·검색 옵션 구성, page/list_count/page_count 세팅. [55]
글 보기	`dispBoardContentView`	`BoardView::dispBoardContentView()` → `DocumentModel::getDocument(document_srl, false, true)` → 댓글/첨부 보조 호출 → 템플릿 표시. [56]	문서 존재 여부, 현재 게시판/포함 모듈 소속 여부, 접근 가능 여부를 확인. 비밀글·포함 게시판 케이스 처리. [57]
글 쓰기 폼	`dispBoardWrite`	`BoardView::dispBoardWrite()` → 기존 문서면 `DocumentModel::getDocument()`/`setDocument()` → 수정 가능성 검사 → 쓰기 템플릿 렌더. [58]	`grant->write_document` 확인. 등록일 보호, 댓글 존재 시 수정 보호 등 게시판 설정 반영. [58]
새 글 등록	`procBoardInsertDocument`	`BoardController::procBoardInsertDocument()` → 요청 변수 수집 → 특수문자 정리 → 빈 본문/길이/data URL/카테고리 검사 → 상태/익명모드/공지 처리 → `DocumentController::insertDocument()` 호출 → 세션 grant 부여 → 리다이렉트. [59]	`grant->write_document` 확인, 익명 모드면 사용자 정보 제거 및 가명 부여, 관리자 권한 없으면 공지/스타일 필드 제거. [60]
글 수정	`procBoardInsertDocument` 동일 액션 내부 update 분기	`procBoardInsertDocument()` → 기존 문서 로드 → `oDocument->isGranted()` 검사 → 관리자 문서 보호/등록일 보호 검사 → `DocumentController::updateDocument()` 호출 → 리다이렉트. [61]	수정 권한은 모듈 grant뿐 아니라 문서 객체 grant가 최종 판단. 익명 글은 트리거를 사용해 익명 정보 유지. [24]

요청 시나리오	URL/액션	대표 호출 순서	권한/검증
글 삭제	`procBoardDeleteDocument`	`BoardController::procBoardDeleteDocument()` → 문서 조회 → 존재/권한 검사 → 보호 설정 검사 → `DocumentController::deleteDocument()` 또는 휴지통 로직 → 목록으로 리다이렉트. 62	`oDocument->isGranted()` 재검사, 댓글 수 기준 보호, 관리자 글 삭제 보호. 63
댓글 작성	`procBoardInsertComment`	`BoardController::procBoardInsertComment()` → 대상 문서 조회 → 접근성 확인 → 특수문자 정리 → 빈 본문/길이/data URL 검사 → 익명/비밀 상태 결정 → 부모 댓글 검사 → `CommentController::insertComment()` → 세션 grant 부여 → 댓글 anchor로 리다이렉트. 64	`grant->write_comment`, 문서 접근성, 부모 댓글이 같은 문서에 속하는지, 부모가 비밀댓글이면 비밀 상속. 65
댓글 수정	`procBoardInsertComment` 동일 액션 내부 update 분기	기존 댓글 로드 → 보호 설정 검사 → `comment->isGranted()` 확인 → `CommentController::updateComment()` → 리다이렉트. 66	등록일/대댓글 수 기반 보호, 관리자 댓글 보호. 67
댓글 삭제	`procBoardDeleteComment`	댓글 조회 → `comment->isGranted()` → 삭제 보호/관리자 보호 검사 → 설정에 따라 삭제문구 치환 또는 실제 삭제/휴지통 이동 → 문서로 리다이렉트. 68	자식 댓글 수 보호, 관리자 댓글 보호, 삭제문구 정책 적용. 69

목록 조회의 내부 동작

실제 목록 조회는 `dispBoardContent()`가 직접 SQL을 치지 않고, 보조 메서드들을 호출해 화면 구성요소를 단계적으로 조립하는 방식이다. 먼저 access/list grant를 확인하고, 카테고리 목록을 구성한 뒤 검색 옵션을 템플릿에 주입한다. 이후 `dispBoardContentList()`에서 `module_srl`, `page`, `list_count`, `page_count`, `include_days` 등을 담은 `$args`를 만들고, 최종적으로 `DocumentModel::getDocumentList($args, $this->except_notice, TRUE, $this->columnList)`를 호출한다. 반환값에는 `data`, `total_count`, `total_page`, `page`, `page_navigation`이 포함되어 화면에 그대로 세팅된다. 즉, **페이징 책임은 보드-뷰가 아니라 하위 문서 모델과 페이지네이션 계층에 분산** 되어 있다. 70

```
$output = DocumentModel::getDocumentList(
    $args, $this->except_notice, TRUE, $this->columnList
);
```

이 짧은 호출 한 줄이 게시판 목록 처리의 핵심이다. 보드 모듈은 목록 화면의 정책과 파라미터를 만들고, 실제 목록 조립과 페이지 객체 생성은 문서 모델이 담당한다. 71

글 보기, 댓글, 첨부의 결합 방식

글 보기에서는 `dispBoardContentView()`가 `DocumentModel::getDocument($document_srl, false, true)`로 문서 객체를 가져온다. 이후 댓글 목록은 `oDocument->getComments()`, 첨부 목록은 `oDocument->getUploadedFiles()` 식으로 문서 객체를 통해 접근한다. 즉, 보드 화면은 문서 객체를 중심 허브로 사용하며, 댓글/첨부는 **문서 객체의 파생 관계** 로 연결된다. 첨부파일 테이블이 `upload_target_srl`을 사용한다는 점과 정확히 맞물리는 설계다. [72]

익명글과 비밀댓글의 처리

게시판 설정에서 익명 모드가 켜져 있으면, 댓글/글 등록 시 사용자 식별 정보는 제거되고 `member_srl`은 음수 값으로 바뀌며, `createAnonymousName()`으로 가명이 생성된다. 수정 시에도 익명 정보가 유지되도록 게시판 컨트롤러는 `document.updateDocument` before 트리거를 추가한다. 댓글 답글의 경우 부모 댓글이 비밀이고 게시판에서 비밀 상태를 허용하면, 자식 댓글도 자동으로 `is_secret='Y'`가 된다. 이 동작은 대댓글 계층에서 비밀글 접근 모델을 일관되게 유지하려는 설계로 볼 수 있다. [73]

```
if ($parent_comment->isSecret() && $this->module_info->secret === 'Y')
{
  $obj->is_secret = 'Y';
}
```

이 짧은 조건문은 “비밀댓글에 대한 답글은 비밀을 상속한다”는 정책을 직접 보여준다. [74]

성능과 보안 고려사항 및 개선 포인트

성능 관점

Rhymix는 프레임워크 차원에서 캐시와 라우트 캐시를 제공한다. `Rhymix\Framework\Cache`는 TTL과 그룹 버전 관리가 가능한 범용 캐시를 제공하고, 라우터는 `_action_cache_prefix`, `_action_cache_module`, `_route_cache` 같은 정적 캐시를 가진다. 즉, 동일 요청 패턴이 반복될 때 **“URL 해석 → 액션 매핑” 비용을 줄이는 설계** 가 이미 들어가 있다. [75]

다만 게시판 목록에서는 구조적으로 비용이 커질 수 있는 부분이 보인다. `board.view.php`는 현재 보고 있는 문서가 목록 몇 페이지에 속하는지를 찾는 로직을 “매우 resource-intensive”하다고 주석으로 직접 밝히며, 크롤러라면 이 과정을 생략하도록 구현하고 있다. 이는 대형 게시판에서 “본문 하단 목록(bottom list)”이 예상보다 비싼 기능이라는 의미다. 따라서 트래픽이 많고 게시글 수가 많은 사이트라면 **하단 목록 비활성화, page-hint 캐시, crawler 최적화 강화** 가 성능상 유의미하다. [76]

문서 스키마를 보면 `module_srl`, `category_srl`, `is_notice`, `list_order`, `update_order`, `regdate` 등엔 인덱스가 있지만, 현재 공개 스키마는 주로 **단일 컬럼 인덱스 중심** 이다. 여기서부터는 보고서 작성자의 추론이지만, 대형 커뮤니티라면 다음과 같은 복합 인덱스를 검토할 가치가 크다. 현재 보드 목록이 `module_srl + 정렬컬럼 + 카테고리` 조합으로 자주 필터링되기 때문이다. [77]

개선 포인트	이유	판단 근거
`documents(module_srl, list_order)` 복합 인덱스 검토	기본 목록 정렬이 `list_order` 계열일 때 유리	현재 스키마에 `module_srl`, `list_order` 단일 인덱스만 공개됨. 46
`documents(module_srl, category_srl, list_order)` 검토	카테고리 게시판 목록 최적화	카테고리 필터와 정렬이 동시에 자주 쓰임. 78
검색 전용 전략 분리	`title/content/comment/tag` 검색은 범위가 넓음	보드 기본 검색 컬럼 범위가 매우 넓다. 37
하단 목록 캐시 또는 비활성화	현재 문서의 목록상 페이지 계산이 비쌈	코드 주석이 직접 resource-intensive라고 명시. 76

검색은 기본 컬럼 범위가 넓고, 댓글 검색까지 포함될 수 있으므로 데이터가 커질수록 DB 부하가 가파르게 증가할 가능성이 높다. 따라서 대형 사이트에서는 FULLTEXT, 별도 검색 인덱스, 또는 최소한 "제목/작성자 위주 간이 검색과 본문/댓글 통합검색 분리" 같은 정책적 조정이 필요하다. 이 부분은 공식 소스에 직접 구현되어 있다기보다, 현재 기본 검색 옵션 범위를 바탕으로 도출한 개선 의견이다. 37

보안 관점

보안 측면에서 Rhymix 보드 스택은 한 겹이 아니라 여러 겹이다. 먼저 라우터 단계에서 액션별 `check_csrf`, `enable_session`, `cache_control`이 주입되고, 보드 뷰 초기화는 주요 요청 변수를 HTML-encode한다. HTML 본문 자체는 `HTMLFilter`가 HTMLPurifier로 정리할 수 있고, 파일 업로드는 `FileContentFilter`가 `Context` 초기화 시점에 검사한다. 파일명과 경로는 `FilenameFilter`가 정리하며, 직접 다운로드 가능 여부에 따라 파일 모듈을 경유하도록 분기할 수 있다. 즉, 보안은 **route metadata → request encoding → content filtering → file filtering → object grant** 순으로 중첩되어 있다. 79

게시판 컨트롤러 자체도 입력을 상당히 엄격히 검사한다. 글과 댓글 모두 빈 본문, 과도한 길이, 과도한 inline data URL을 제한하고, 관리자 권한이 없으면 공지·스타일 필드를 제거한다. 삭제와 수정은 반드시 실제 문서/댓글 객체의 grant를 재검사하며, 관리자 작성 글·댓글 보호 옵션, 댓글 수 기반 수정/삭제 보호 옵션도 적용한다. 비회원 문서/댓글은 `procBoardVerificationPassword()`를 통해 비밀번호가 확인되면 세션 grant를 부여하는 방식으로 편집 권한을 잠시 연다. 이는 무작정 password 비교만 하는 것이 아니라 **세션 기반 grant 부여** 로 이어진다는 점에서 비교적 정돈된 설계다. 80

다만 개선 포인트도 있다. 아래 내용은 현재 코드와 공개 문서를 바탕으로 한 **감사 관점 제안** 이다. 81

개선 제안	이유
CSRF 토큰 적용 지점을 운영 문서에 더 명확히 노출	라우터 옵션 기반 보안은 강력하지만, 운영자 관점에서는 "어디서 토큰이 강제되는지"가 눈에 잘 보이지 않는다.
댓글/문서 삭제 정책 로그 강화	현재 보호 옵션이 많아 정책은 풍부하지만, 실제 운영 중 "왜 삭제가 거부되었는지" 추적 로그가 더 있으면 감사가 쉬워진다.
익명 모드 감사 포인트 분리	`member_srl` 음수 변환과 트리거 기반 익명 정보 보존은 영리하지만, 추후 데이터 분석·마이그레이션 시 혼동 소지가 있다.
대용량 data URL 제한 기본값 재검토	인라인 이미지 남용은 DB/응답 크기 증가와 직접 연결되므로 대형 사이트에서는 더 보수적 기본값이 유리할 수 있다.

핵심 코드 발췌와 참고 소스 안내

핵심 함수와 동작 해설

아래 발췌는 동작 원리를 설명하기 위한 **짧은 핵심 구문** 만 실었다.

구문	의미	파일 경로
`Context::init();`	요청, 세션, 업로드, 언어, 인증, 라우팅 정보를 모두 초기화한다.	`index.php`
`$request = Router::parseURL(...)`	URL을 `module`, `act`, `mid`, args로 바꾼다.	`classes/context/Context.class.php` `common/framework/Router.php`
`$oModuleHandler->displayContent($oModuleHandler->procModule())`	실행과 렌더링을 분리한 Rhymix 핵심 호출.	`index.php`
`$output = DocumentModel::getDocumentList(...)`	게시판 목록은 실제로 문서 모듈 모델이 가져온다.	`modules/board/board.view.php`
`$output = $oDocumentController->insertDocument(...)`	글 저장은 보드가 아니라 문서 컨트롤러에 위임된다.	`modules/board/board.controller`
`$output = $oCommentController->insertComment(...)`	댓글 저장도 댓글 컨트롤러에 위임된다.	`modules/board/board.controller`

구문	의미	파일 경로
`ModuleController::addTriggerFunction('document.updateDocument', 'before', ...)`	익명 글 수정 시 메타데이터를 유지하기 위해 트리거를 사용한다.	`modules/board/board.controller`
`$oSecurity->encodeHTML(...)`	request 파라미터를 화면 반사 전 HTML 인코딩한다.	`modules/board/board.view.php`

```
require __DIR__ . '/common/autoload.php';
Context::init();

$oModuleHandler = new ModuleHandler();
$oModuleHandler->init() && $oModuleHandler->displayContent($oModuleHandler->procModule());
```

위 네 줄은 Rhymix 전체 요청 라이프사이클의 압축판이다. 오토로드 → 컨텍스트 초기화 → 모듈 실행 → 응답 렌더링이라는 전체 구조가 이 안에 다 들어 있다. 11

```
$output = $oDocumentController->insertDocument(
    $obj, $manual, false, $obj->document_srl ? false : true
);
```

이 호출은 “게시판이 글을 직접 저장한다”는 오해를 바로잡아준다. 게시판 컨트롤러는 정책 검증과 파라미터 조립을 맡고, 실제 문서 저장은 하위 `document` 모듈로 위임한다. 84

참고한 공식 문서와 소스코드

아래 목록은 이번 보고서에서 핵심적으로 사용한 **공식 사이트·공식 문서·공식 GitHub 소스** 이다. 인용 표시는 각각 해당 페이지로 이동하는 링크 역할을 한다.

구분	자료명	활용 목적
공식 사이트	Rhymix 홈페이지 및 다운로드 안내 4	최신 안정판, 요구환경, 릴리스 시점 확인
공식 소개	Rhymix 소개 페이지 88	CMS/프레임워크 정체성 확인
공식 매뉴얼	매뉴얼 홈 89	문서 체계와 한국어 공식 자료 범위 확인

구분	자료명	활용 목적
공식 매뉴얼	라우터 사용법 [90]	`mid` 기반 짧은주소, `module.xml` route 규칙 확인
공식 API 문서	`legacy.md` [91]	레거시/프레임워크 혼합 구조 확인
공식 API 문서	`ModuleHandler` 문서 [15]	요청 lifecycle와 trigger 시스템 확인
공식 API 문서	`FrontEndFileHandler` 문서 [92]	CSS/JS 자산 관리 구조 확인
공식 API 문서	`Router` 문서 [93]	rewrite level과 URL 파싱 API 확인
공식 API 문서	`DB` 문서 [19]	prepared statement / XML query 구조 확인
공식 API 문서	`Cache` 문서 [94]	캐시 계층 확인
공식 API 문서	`Security`, `HTMLFilter`, `FileContentFilter`, `FilenameFilter` 문서 [22]	CSRF/XSS/업로드 보안 구조 확인
GitHub 소스	`index.php` [11]	부트스트랩 확인
GitHub 소스	`Context.class.php` [12]	초기화·라우팅·세션·인증 흐름 확인
GitHub 소스	`Router.php` [95]	URL 파싱과 route option 확인
GitHub 소스	`modules/board/conf/module.xml` [96]	게시판 액션/권한/route 정의 확인
GitHub 소스	`board.view.php`, `board.controller.php`, `board.model.php` [16]	게시판 실제 동작 순서 분석
GitHub 소스	`documents.xml`, `comments.xml`, `files.xml`, `modules.xml` [97]	게시판 관련 DB 스키마 확인

관련 표준과 해석 보조 자료

Rhymix 공식 자료만으로도 구현을 충분히 분석할 수 있지만, 라우터와 HTTP 요청 모델을 더 엄밀하게 이해하려면 URI/HTTP의 표준 문맥도 함께 보면 좋다. 보조 표준으로는 **URI generic syntax의 RFC 3986**과 **HTTP semantics의 RFC 9110**이 적합하다. 본 보고서에서는 검색 결과를 통해 이 두 RFC를 관련 표준으로 식별했으며, Rhymix Router의 URL 파싱과 HTTP method 처리 해석의 배경 문헌으로 삼았다. [98]

종합 결론

정리하면, Rhymix 게시판 모듈은 “독립된 소형 앱”이 아니라 **Rhymix 전체 요청 처리 파이프라인 위에 얹힌 조정 계층**이다. 요청은 `index.php → Context → Router → ModuleHandler → BoardView/BoardController → Document/Comment/File → DB → DisplayHandler`로 흐르고, 보드 모듈은

이 중간에서 **권한, 게시판 정책, 익명/비밀글 규칙, 카테고리, 하단 목록, 댓글/첨부 조합** 을 관리한다. 실제 저장과 조회의 많은 핵심은 하위 모듈에 위임되므로, Rhymix 게시판을 깊이 이해하려면 보드 소스만이 아니라 **문서/댓글/파일 스키마와 라우터/컨텍스트 계층을 함께 읽어야 한다** 는 것이 이 보고서의 핵심 결론이다. [99]

---

[1] [3] [4] [8] https://rhymix.org/
https://rhymix.org/

[2] [23] [24] [39] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [73] [74] [80] [84] [85] [86] https://github.com/rhymix/rhymix/blob/2.1.33/modules/board/board.controller.php
https://github.com/rhymix/rhymix/blob/2.1.33/modules/board/board.controller.php

[5] https://github.com/rhymix/rhymix
https://github.com/rhymix/rhymix

[6] [89] https://rhymix.org/manual
https://rhymix.org/manual

[7] [11] [25] [31] [54] [82] [99] https://github.com/rhymix/rhymix/blob/2.1.33/index.php
https://github.com/rhymix/rhymix/blob/2.1.33/index.php

[9] https://rhymix.org/manual/theme/intro
https://rhymix.org/manual/theme/intro

[10] [17] [91] https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/legacy.md
https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/legacy.md

[12] [34] [83] https://github.com/rhymix/rhymix/blob/2.1.33/classes/context/Context.class.php
https://github.com/rhymix/rhymix/blob/2.1.33/classes/context/Context.class.php

[13] [26] [93] https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/Router.md
https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/Router.md

[14] https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework.md
https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework.md

[15] [21] [30] https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/legacy/ModuleHandler.md
https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/legacy/ModuleHandler.md

[16] [28] [29] [38] [50] [55] [56] [57] [58] [70] [71] [72] [76] [87] https://github.com/rhymix/rhymix/blob/2.1.33/modules/board/board.view.php
https://github.com/rhymix/rhymix/blob/2.1.33/modules/board/board.view.php

[18] https://raw.githubusercontent.com/rhymix/rhymix/2.1.33/classes/context/Context.class.php
https://raw.githubusercontent.com/rhymix/rhymix/2.1.33/classes/context/Context.class.php

[19] [53] https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/DB.md
https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/DB.md

[20] [75] [94] https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/Cache.md
https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/Cache.md

[22] https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/Security.md
https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/Security.md

[27] [32] [51] [79] [81] [95] https://github.com/rhymix/rhymix/blob/2.1.33/common/framework/Router.php
https://github.com/rhymix/rhymix/blob/2.1.33/common/framework/Router.php

33 90 https://rhymix.org/manual/plugin/router/router
https://rhymix.org/manual/plugin/router/router

35 41 42 https://github.com/rhymix/rhymix/tree/2.1.33/modules/board
https://github.com/rhymix/rhymix/tree/2.1.33/modules/board

36 43 49 96 https://github.com/rhymix/rhymix/blob/2.1.33/modules/board/conf/module.xml
https://github.com/rhymix/rhymix/blob/2.1.33/modules/board/conf/module.xml

37 https://github.com/rhymix/rhymix/blob/2.1.33/modules/board/board.class.php
https://github.com/rhymix/rhymix/blob/2.1.33/modules/board/board.class.php

40 https://github.com/rhymix/rhymix/blob/2.1.33/modules/board/board.model.php
https://github.com/rhymix/rhymix/blob/2.1.33/modules/board/board.model.php

44 45 https://github.com/rhymix/rhymix/blob/2.1.33/modules/module/schemas/modules.xml
https://github.com/rhymix/rhymix/blob/2.1.33/modules/module/schemas/modules.xml

46 77 78 97 https://github.com/rhymix/rhymix/blob/2.1.33/modules/document/schemas/documents.xml
https://github.com/rhymix/rhymix/blob/2.1.33/modules/document/schemas/documents.xml

47 https://github.com/rhymix/rhymix/blob/2.1.33/modules/comment/schemas/comments.xml
https://github.com/rhymix/rhymix/blob/2.1.33/modules/comment/schemas/comments.xml

48 https://github.com/rhymix/rhymix/blob/2.1.33/modules/file/schemas/files.xml
https://github.com/rhymix/rhymix/blob/2.1.33/modules/file/schemas/files.xml

52 https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/filters/FileContentFilter.md
https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/filters/FileContentFilter.md

88 https://rhymix.org/about
https://rhymix.org/about

92 https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/legacy/FrontEndFileHandler.md
https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/legacy/FrontEndFileHandler.md

98 https://en.wikipedia.org/wiki/Uniform_Resource_Identifier
https://en.wikipedia.org/wiki/Uniform_Resource_Identifier