DXCMS v8.1.0

기술 아키텍처 문서

CMS 아키텍처 — 데이터 흐름 연결 상세 분석

버전: DXCMS v8.1.0
작성일: 2026년 5월 **저작권:** © 2026 designonex [디자인원엑스]
라이선스: GNU Lesser General Public License v3.0 (LGPL-3.0)

1. 데이터 흐름 개요

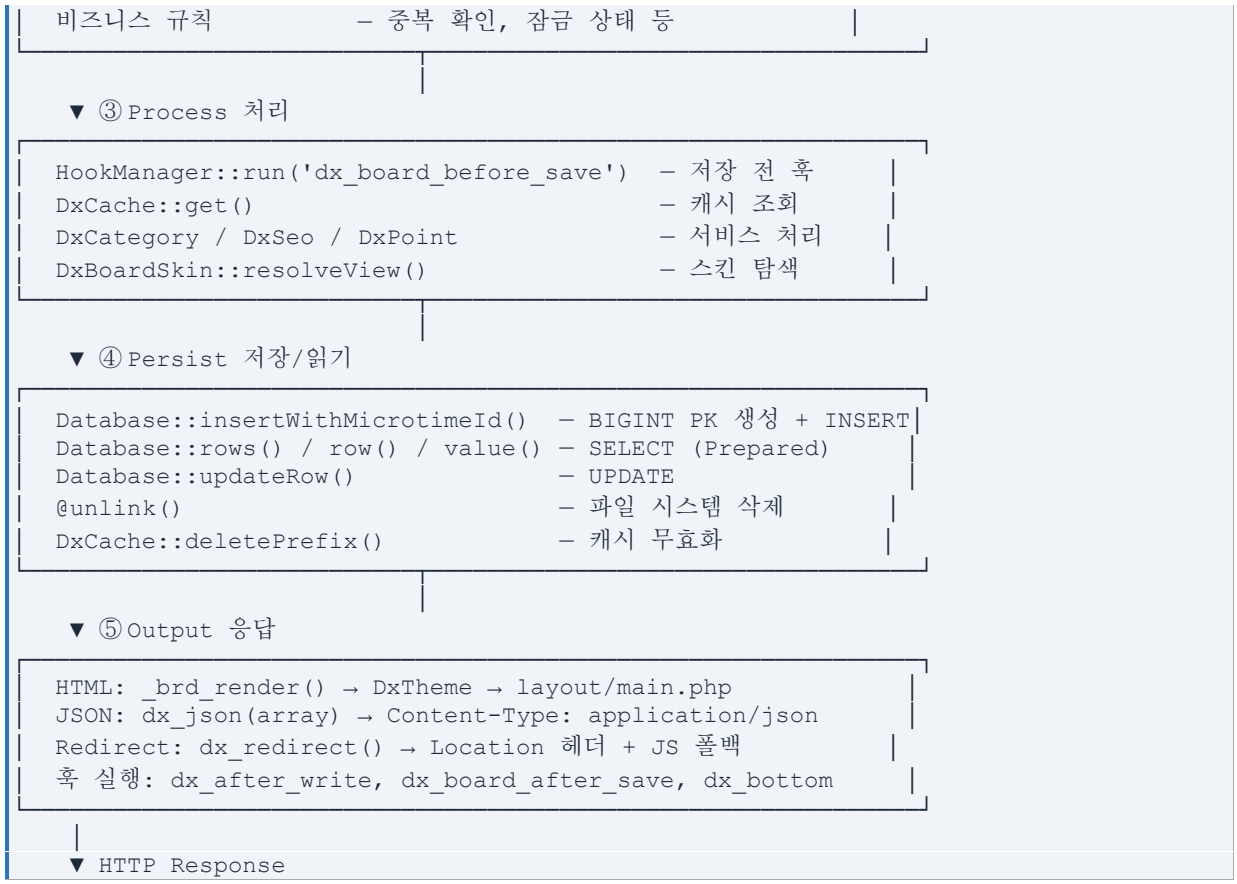
DXCMS의 데이터는 HTTP 요청이 들어오는 순간부터 HTML 응답이 나가는 순간까지 여러 계층을 거쳐 변환됩니다. 이 문서는 입력(Input) → 검증(Validate) → 처리(Process) → 저장(Persist) → 응답(Output)의 5단계 데이터 흐름을 각 시나리오별로 추적합니다.

1.1 데이터 흐름의 5 단계

단계	이름	설명
① Input	입력 수집	HTTP Request에서 데이터 수집. dx_get(), dx_post(), $_FILES, $_SESSION, $_COOKIE. BIGINT ID는 ctype_digit() 검증 후 문자열 유지 (32bit PHP 오버플로우 방지)
② Validate	입력 검증	보안 검증(CSRF, XSS, 권한), 형식 검증(타입 캐스팅, 정규식), 비즈니스 규칙 검증(중복 확인, 잠금 상태). 실패 시 dx_error() 또는 dx_json() 즉시 반환
③ Process	데이터 처리	DxSanitizer로 정제, 비즈니스 로직 실행, DxCache 조회/무효화, 훅(Hook) 발생
④ Persist	저장/읽기	Database(PDO) Prepared Statement로 MySQL 읽기·쓰기. BIGINT PK는 insertWithMicrotimeId()로 생성
⑤ Output	응답 생성	HTML(renderWithLayout), JSON(dx_json), Redirect(dx_redirect), 파일 다운로드. 캐시 무효화 후 응답

1.2 데이터 흐름 전체 지도

```
HTTP Request
    │
    ▼ ① Input 수집
┌──────────────────────────────────────────────────────────────┐
│  dx_get() / dx_post() / $_FILES / $_SESSION / $_COOKIE       │
│  BIGINT: ctype_digit() 검증 후 문자열 유지 (32bit 안전)      │
└──────────────────────────┬───────────────────────────────────┘
                           │
    ▼ ② Validate 검증
┌──────────────────────────────────────────────────────────────┐
│  Secure::csrfCheck()     — CSRF 토큰 검증 (POST 필수)        │
│  Auth::isLoggedIn()      — 인증 상태 확인                    │
│  Auth::isAdmin()         — 권한 레벨 확인                    │
│  DxSanitizer::text()     — 입력 정제 / XSS 방어              │
```



2. 입력(Input) — 데이터 수집 계층

모든 외부 입력은 반드시 헬퍼 함수를 통해 추상화됩니다. raw $_GET/$_POST에 직접 접근하지 않고 타입 캐스팅과 기본값 처리를 함께 수행합니다.

2.1 입력 수집 헬퍼 함수

함수	소스	동작 및 특이사항
`dx_get($key, $default, $type)`	`$_GET`	GET 파라미터 추출. dx_cast()로 타입 캐스팅
`dx_post($key, $default, $type)`	`$_POST`	POST 파라미터 추출. dx_cast()로 타입 캐스팅
`dx_request($key, $default, $type)`	`$_REQUEST`	GET+POST 통합 추출
`dx_cast($val, 'bigint')`	내부 함수	32bit PHP 오버플로우 방지. ctype_digit() 검증 후 문자열 반환. (int) 캐스팅 절대 금지

dx_ip()	$_SERVER	Cloudflare/CDN/리버스프록시 헤더 우선 순위로 실제 IP 추출
$_SESSION[sessionKey()]	세션	Secure::getSessionKey()가 반환하는 동적 키 이름 사용 (사이트별 고유)
$_COOKIE['dx_remember']	쿠키	Remember Me 자동 로그인. member_id:token 형식. 만료 확인 후 토큰 롤링 갱신

2.2 BIGINT ID 처리 — 32bit PHP 안전 패턴

post_id, comment_id 등 BIGINT 컬럼은 16자리 숫자입니다. 32bit PHP에서 (int) 캐스팅 시 오버플로우가 발생하므로 문자열 그대로 사용합니다.

```
// □ WRONG — 32bit PHP에서 16자리 BIGINT가 음수로 변환됨
$id = (int)$_GET['id'];   // → 오버플로우 버그

// □ CORRECT — ctype_digit() 검증 후 문자열 유지
$id = dx_cast($_GET['id'], 'bigint');  // → '1746000000123456' (문자열)

// Router에서의 처리 (Router.php)
$this->current['id'] = ($third && ctype_digit($third)) ? $third : '0';

// handler.php에서의 처리
$_rawId = isset($GLOBALS['dx_route']['id']) ? $GLOBALS['dx_route']['id'] : '0';
$id = (is_string($_rawId) && ctype_digit($_rawId) && $_rawId !== '0') ? $_rawId :
'';

// PDO 바인딩 시 EMULATE_PREPARES=true로 문자열을 BIGINT로 자동 처리
PDO::ATTR_EMULATE_PREPARES  => true,
PDO::ATTR_STRINGIFY_FETCHES => true,  // BIGINT 결과도 문자열로 수신
```

2.3 세션 키 동적 생성 — 예측 불가 구조

세션 키 이름은 설치 시 생성된 64자리 secret_key를 기반으로 동적으로 생성됩니다. 소스코드가 공개되어도 세션 키 이름을 예측할 수 없습니다.

```
// Secure.php — initSecretKeys()
// secret_key(64자리) → SHA-256 → 앞 16자리를 세션 키 이름으로 사용
$this->sessionKey = 'dx_' . substr(hash('sha256', $secret), 0, 16);
// 결과 예: 'dx_a1b2c3d4e5f6a7b8' (사이트마다 다름)

// Auth.php — 세션 읽기/쓰기
```

```
$_SESSION[$this->sessionKey()] = [
    'id'    => $user['id'],
    'token' => $this->makeToken($user),  // HMAC-SHA256
];


// 세션 토큰 생성 (Auth::makeToken)
// user.id + user.join_date(불변값) → secret_key로 HMAC-SHA256
$payload = $user['id'] . '|' . $user['join_date'];
return hash_hmac('sha256', $payload, $secret);
```

3. 검증(Validate) — 보안 및 규칙 검증

입력된 데이터는 세 겹의 검증을 통과해야 합니다. 보안 검증(CSRF·인증·권한), 형식 검증(타입·정규식), 비즈니스 규칙 검증(중복·잠금) 순서로 실행됩니다. 어느 단계에서든 실패하면 즉시 오류 응답이 반환됩니다.

3.1 CSRF 토큰 검증 흐름

```
// 모든 POST 요청의 첫 번째 줄 (API, handler.php 공통)
dx_csrf_check();  // 실패 시 403 exit (Secure::csrfCheck() 위임)


// Secure::csrfCheck() 내부 동작
$token  = $_POST['_csrf'] ?? '';                  // 폼에서 전송된 토큰
$stored = $_SESSION[$this->csrfKey];                // 세션에 저장된 토큰
if (!hash_equals($stored, $token)) {
    http_response_code(403); exit('CSRF 토큰 오류');
}
// 검증 후 새 토큰 재발급 (토큰 교체로 CSRF 재사용 공격 방지)
// v5.2.4: AJAX 응답에 new_csrf 포함하여 JS가 갱신 가능
```

3.2 인증 검증 흐름

검증 함수	내부 동작
`Auth::getInstance()`	생성자에서 loadSession() 호출. 세션 → DB 회원 조회 → 토큰 검증. 실패 시 자동 logout()
`loadSession()`	① 세션 키 확인 → ② DB에서 member status=1 확인 → ③ HMAC 토큰 일치 확인. 셋 중 하나라도 실패 시 로그아웃
`tryRememberMe()`	세션 없을 때 dx_remember 쿠키 확인. member_id:token 파싱 → DB 토

	큰 비교(hash_equals) → 만료 확인 → 성공 시 세션 복구 + 토큰 롤링
`isLoggedIn()`	$this->user !== null 반환. O(1) 확인
`isAdmin()`	$this->user['role'] === 'admin' 확인
`get($field, $default)`	로그인 회원 정보 필드 반환. 비로그인 시 $default 반환

3.3 DxSanitizer — 입력 정제 흐름

모든 사용자 입력은 저장 전에 DxSanitizer를 통해 정제됩니다. 에디터 콘텐츠(HTML)와 일반 텍스트는 다른 정제 방식을 사용합니다.

메서드	처리 내용
`DxSanitizer::text($val)`	trim + 허용 태그 모두 제거. 일반 텍스트 입력(제목, 태그, 카테고리 등)에 사용
`DxSanitizer::editorContent($html)`	에디터 HTML 정제. 허용 태그 화이트리스트, on* 이벤트 핸들러 제거, javascript: 프로토콜 차단. CKEditor/Jodit/TinyMCE 출력물에 적용
`dx_esc($str)`	htmlspecialchars(ENT_QUOTES, UTF-8). 출력 시 XSS 방어. 테마 파일에서 모든 동적 출력에 사용
`dx_safe_url($url)`	javascript:/vbscript:/data: 프로토콜 차단. 서브디렉토리 설치 시 절대경로 자동 변환
`DxSanitizer::filename($name)`	이중확장자(shell.php.jpg) 차단. 특수문자 제거. 업로드 파일명 정규화

3.4 권한 검증 데이터 흐름

```
// handler.php — 게시글 작성 권한 검증
$wl = (int)$board['write_level'];  // 0=전체, 1=회원, 9=관리자
if ($wl===1 && !$auth->isLoggedIn()) {
    dx_redirect(dx_base_url('auth/login') . '?redirect=' .
urlencode(dx_current_url()));
    exit;
}
if ($wl===9 && !$auth->isAdmin()) {
    dx_error('관리자만 글쓰기 가능합니다.', 403);
}
```

```
// 게시글 수정 권한 검증 (작성자 또는 관리자)
$ok = $auth->isAdmin()
   || ($auth->isLoggedIn() && $auth->user()['id'] == $editPost['member_id']);
if (!$ok) dx_error('수정 권한이 없습니다.', 403);


// 비밀글 접근 권한
if (!empty($post['is_secret'])) {
    $ok = $auth->isAdmin()
       || ($auth->isLoggedIn() && $auth->user()['id'] == $post['member_id']);
    if (!$ok) dx_error('비밀글입니다.', 403);
}
```

4. Database — 데이터베이스 읽기/쓰기 흐름

모든 DB 접근은 Database 클래스의 PDO Prepared Statement를 통해 이루어집니다. SQL Injection을 원천 차단하며, BIGINT 타입 안전을 보장합니다.

4.1 쿼리 실행 파이프라인

```
// Database::execute() — 모든 쿼리의 공통 실행 경로
$this->queryCount++;


if (DX_DEBUG) {
    $this->queryLog[] = ['sql' => $sql, 'params' => $params];
}


try {
    $stmt = $this->pdo->prepare($sql);   // Prepared Statement 생성
    $stmt->execute((array)$params);       // 바인딩 + 실행
    return $stmt;
} catch (PDOException $e) {
    if ($GLOBALS['DX_SHUTTING_DOWN']) throw $e;  // shutdown 중 재throw
    if (DX_DEBUG) dx_error('DB 오류: ' . $sql);
    else          dx_error('데이터베이스 오류가 발생했습니다.');
}


// PDO 연결 옵션 (32bit 안전)
PDO::ATTR_EMULATE_PREPARES  => true,   // BIGINT 오버플로우 방지
PDO::ATTR_STRINGIFY_FETCHES => true,   // 모든 숫자를 문자열로 수신
PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
```

4.2 BIGINT PK 생성 — insertWithMicrotimeId()

게시글(posts), 댓글(comments) 등 BIGINT PK 테이블은 AUTO_INCREMENT 대신 microtime 기반 고유 ID를 생성합니다. 분산 환경에서 충돌을 방지하고, 삽입 순서를 시간순으로 유지합니다.

```
// Database::generateMicrotimeId()
// 13자리 밀리초 타임스탬프 + 랜덤 3자리 = 최대 16자리
$mt  = microtime(true);
$sec = floor($mt);
$ms  = round(($mt - $sec) * 1000);
$msStr = sprintf('%010d%03d', $sec, $ms);   // 13자리 문자열
$rndStr = sprintf('%03d', rand(0, 999));     // 3자리 랜덤
$id = $msStr . $rndStr;                       // 16자리 문자열 (절대 (int) 금지!)

// 충돌 확인 후 재시도 (최대 10회)
$exists = $db->value("SELECT COUNT(*) FROM `{$tbl}` WHERE id = ?", [$id]);
if (!$exists) return $id;  // 문자열 반환 — 32bit PHP 안전
usleep(1000);  // 1ms 대기 후 재시도
```

4.3 Database 메서드 호출 패턴

메서드	SQL	사용 시나리오
`row($sql, $params)`	`SELECT … LIMIT 1`	게시글 단건, 회원 조회, 게시판 설정 로드
`rows($sql, $params)`	`SELECT …`	목록 조회 (게시글 목록, 댓글 목록, 파일 목록)
`value($sql, $params)`	`SELECT 단일값`	COUNT(*), 인기점수 계산값, 단일 컬럼
`insertRow($table, $data)`	`INSERT`	알림, 방문자 로그, 설정 저장 등 AUTO_INCREMENT 테이블
`insertWithMicrotimeId($table, $data)`	`INSERT (BIGINT PK)`	게시글, 댓글 저장. microtime ID 자동 생성
`updateRow($table, $data, $where)`	`UPDATE`	조회수 +1, 좋아요 수 갱신, 회원 정보 수정
`deleteRow($table, $where)`	`DELETE`	단순 조건 삭제 (편의 메서드)
`query($sql, $params)`	`DML (영향 행 수)`	복잡한 UPDATE/DELETE, popular_score 갱신
`begin() / commit() / rollback()`	`트랜잭션`	다중 테이블 원자적 처리
`tableExists($name)`	`SHOW TABLES LIKE`	마이그레이션 전 컬럼/테이블 존재 여부 확인

4.4 QueryBuilder — 라라벨 스타일 체이닝

기존 Database 클래스와 완전히 하위호환하면서 라라벨 스타일의 체이닝 쿼리를 지원합니다. dx_db() 전역 함수로 어디서나 접근 가능합니다.

```
// 기본 사용법
$posts = dx_db('posts')
    ->where('status', 1)
    ->where('board_id', $boardId)
    ->orderBy('id', 'desc')
    ->limit(10)
    ->get();


// LEFT JOIN (회원 이름 포함)
$posts = dx_db('posts')
    ->leftJoin('members', 'posts.member_id', '=', 'members.id')
    ->select(['posts.*', 'members.name AS member_name'])
    ->where('posts.status', 1)
    ->paginate(20);  // ['data' => rows, 'total' => count, 'pages' => n]


// 복수 조건
$member = dx_db('members')
    ->where('login_id', $loginId)
    ->orWhere('email', $loginId)
    ->where('status', 1)
    ->first();
```

5. DxCache — 캐시 데이터 흐름

DxCache는 APCu(메모리) 또는 파일 기반 캐시를 자동 선택합니다. 캐시는 READ 경로에서 DB 조회를 건너뛰게 하고, WRITE 경로에서 즉시 무효화됩니다.

5.1 캐시 READ 흐름

```
// 게시판 목록 캐시 (handler.php)
$_brdCacheKey = 'board_list_' . $board['board_key'] . '_p' . $page;


// 캐시 적용 조건 (비로그인 + 검색/카테고리/태그 없음 + action=list)
$_brdCacheOk = (
    class_exists('DxCache') &&
    !$search && !$cat && !$tag &&
    !$auth->isLoggedIn() &&
    $action !== 'search'
);


if ($_brdCacheOk) {
    $_brdCached = DxCache::get($_brdCacheKey, null);
    if (is_array($_brdCached)) {
```

```
        extract($_brdCached, EXTR_SKIP);  // 캐시에서 변수 복원
        _brd_render('list', $ctx);
        break;  // DB 쿼리 완전 생략
    }
}

// 전역 공지는 실시간 확인 필요 → 캐시에서 제외, 항상 DB 조회
```

5.2 캐시 WRITE/무효화 흐름

무효화 시점	무효화 대상	코드
게시글 작성/수정	board_list_{key}_*	DxCache::deletePrefix('board_list_'.$key.'_')
게시글 완전 삭제	board_list_{key}_*	_dx_delete_post_completely()에서 호출
설정 저장	전체 캐시	DxCache::flush() (관리자 설정 저장 시)
카테고리 변경	cat_board_{id}	DxCache::deletePrefix('cat_board_')
사이트 설정 저장	site_{md5(domain)}	DxCache::deletePrefix('site_')
댓글 삭제	(목록 캐시 미영향)	comment_count 감소 + popular_score 재계산만

6. 게시글 CRUD — 전체 데이터 흐름

가장 핵심적인 게시판 기능의 데이터 흐름을 단계별로 추적합니다. 각 단계에서 어떤 데이터가 어느 클래스를 거쳐 어디에 저장되는지 명확히 보여줍니다.

6.1 게시글 작성(Write) 전체 데이터 흐름

```
① HTTP POST /free/write
   │
   ▼ ② 입력 수집 (handler.php)
   $data = [
       'board_id' => (int)$board['id'],
       'title'    => DxSanitizer::text(dx_post('title')),      // XSS 정제
       'content'  => DxSanitizer::editorContent($_POST['content']),  // HTML 정제
       'category' => DxSanitizer::text(dx_post('category')),
       'tags'     => DxSanitizer::text(dx_post('tags')),
       'is_notice'=> $auth->isAdmin() && !empty($_POST['is_notice']) ? 1 : 0,
       'is_secret'=> !empty($_POST['is_secret']) ? 1 : 0,
       'member_id'=> (int)$auth->user()['id'],
       'ip'       => dx_ip(),
```

```
    'status'   => 1,
];
│
▼ ③검증
dx_csrf_check();              // CSRF 토큰 검증
Auth::isLoggedIn()            // 로그인 확인
DxCaptcha::verify() (비회원)  // 캡챠 검증
tableExists('posts.link')     // 컬럼 존재 확인 (마이그레이션 호환)
│
▼ ④저장 전 훅
dx_run_hook('dx_board_before_save', ['data' => &$data, 'board' => $board]);
// 플러그인이 $data를 수정 가능
│
▼ ⑤DB INSERT (Persist)
$postId = $db->insertWithMicrotimeId('posts', $data);
// microtime ID: 예) '1746123456789012'  (16자리 문자열)
│
▼ ⑥파일 업로드 처리 (조건부)
if ($board['use_file'] && isset($_FILES['files']))
    dx_board_upload_files($postId, $board, $_FILES['files']);
│
▼ ⑦갤러리 썸네일 자동 생성 (조건부)
if ($board['board_type'] === 'gallery' && class_exists('DxThumb'))
    DxThumb::autoFromPost($postId, $board['board_key'], $thumbOpts);
│
▼ ⑧저장 후 훅 실행
dx_run_hook('dx_after_write',      ['post_id'=>$postId, 'board'=>$board]);
// → _dx_register_point_hooks: DxPoint::add('write'), addExp('write')
dx_run_hook('dx_board_after_save', ['post_id'=>$postId, 'board'=>$board]);
// → 플러그인: 인덱싱, 알림, 외부 API 연동 등
│
▼ ⑨실시간 소켓 브로드캐스트 (조건부)
$_SESSION['dx_post_live_broadcast'] = ['post_id'=>$postId, 'board_key'=>$key];
// view.php 로드 시 JS가 소켓서버에 post_live 이벤트 전송
│
▼ ⑩캐시 무효화
DxCache::deletePrefix('board_list_' . $board['board_key'] . '_');
│
▼ ⑪Redirect
dx_redirect($board['board_key'] . '/view/' . $postId);
```

6.2 게시글 목록(List) 데이터 흐름

```
①HTTP GET /free/list?page=1
│
▼ ②캐시 확인 (비로그인 + 검색없음 + 기본 페이지만)
$cacheKey = 'board_list_free_p1';
$cached = DxCache::get($cacheKey, null);
if ($cached) { _brd_render('list', ctx); break; }  // DB 조회 생략!
│
▼ ③DB 쿼리 (캐시 미스 시)
// 공지글 별도 조회
SELECT * FROM posts WHERE board_id=? AND status=1 AND is_notice=1
// 일반 글 페이지네이션
SELECT p.*, m.name AS member_name FROM posts p
LEFT JOIN members m ON p.member_id=m.id
WHERE p.board_id=? AND p.status=1 AND p.is_notice=0
ORDER BY p.id DESC LIMIT 20 OFFSET 0
// 전체 공지 (실시간 기간 체크, 캐시 제외)
SELECT * FROM global_notices WHERE status=1 AND start_at<=? AND end_at>=?
```

```
   │
   ▼ ④ 카테고리 목록 (캐시)
   DxCategory::getByBoard($boardId, 'list');
   // 캐시키: cat_board_{id}  TTL: 300초
   │
   ▼ ⑤ 컨텍스트 구성 및 훅
   $ctx = compact('board','posts','notices','total','page','perPage',...);
   dx_run_hook('dx_board_list_context', ['context' => &$ctx, 'board' => $board]);
   │
   ▼ ⑥ 캐시 저장
   DxCache::set($cacheKey, [...$ctx], 60);  // TTL 60초
   // 전체 공지는 캐시에서 제외 (실시간 기간 체크 필요)
   │
   ▼ ⑦ 렌더링
   _brd_render('list', $ctx);  // DxBoardSkin 폴백 체인 → layout/main.php
```

6.3 게시글 보기(View) 데이터 흐름

```
① HTTP GET /free/view/1746000000123456
   │
   ▼ ② 입력 수집
   $id = '1746000000123456';  // 문자열 (32bit 안전)
   │
   ▼ ③ DB 조회 (LEFT JOIN으로 회원명 포함)
   SELECT p.*, m.name AS member_name, m.profile_img
   FROM posts p LEFT JOIN members m ON p.member_id=m.id
   WHERE p.id='1746000000123456' AND p.board_id=? AND p.status=1
   │
   ▼ ④ 비밀글 접근 권한 확인
   if ($post['is_secret']) { auth->isAdmin() || 작성자만 허용 }
   │
   ▼ ⑤ 조회수 중복 방지
   $vk = 'viewed_post_' . $id;  // 세션 키
   if (!$_SESSION[$vk]) {
       UPDATE posts SET view_count=view_count+1 WHERE id=?
       // popular_score 재계산: (조회×1 + 좋아요×5 + 댓글×3) × 시간감쇠
       $decay = max(0.1, 1.0 - floor($daysPassed/7) * 0.1);
       UPDATE posts SET popular_score=? WHERE id=?
       $_SESSION[$vk] = 1;  // 중복 방지 플래그
   }
   │
   ▼ ⑥ 관련 데이터 조회
   // 첨부파일
   SELECT * FROM post_files WHERE post_id=? ORDER BY id ASC
   // 댓글 (회원명 포함)
   SELECT c.*, m.name AS member_name FROM comments c
   LEFT JOIN members m ON c.member_id=m.id
   WHERE c.post_id=? AND c.status=1 ORDER BY c.id ASC
   // 이전글/다음글
   SELECT id,title FROM posts WHERE board_id=? AND status=1 AND id<? ORDER BY id
DESC LIMIT 1
   SELECT id,title FROM posts WHERE board_id=? AND status=1 AND id>? ORDER BY id ASC
LIMIT 1
   │
   ▼ ⑦ SEO 메타 생성
   DxSeo::build('board_view', ['post'=>$post, 'board'=>$board]);
   // title, description(본문앞150자), og:image(첫번째img), JSON-LD Article
   // 비밀글 → noindex=true
   │
```

```
   ▼ ⑧ 컨텍스트 구성 및 훅
   dx_run_hook('dx_board_view_context', ['context'=>&$ctx, 'post'=>$post]);
   │
   ▼ ⑨ 렌더링
   _brd_render('view', $ctx);  // DxBoardSkin 폴백 → layout/main.php
```

6.4 게시글 완전 삭제(Delete) 데이터 흐름

게시글 삭제는 연관 데이터를 모두 정리하는 cascade 삭제입니다. 첨부파일 실제 삭제, 댓글 좋아요, 댓글, 게시글 좋아요, 스크랩, 포인트 로그 순서로 처리됩니다.

```
// _dx_delete_post_completely($db, $postId, $boardKey)
// PDO 직접 사용 — Database::execute()의 dx_error/exit 우회


① 첨부파일 삭제
   SELECT save_path FROM post_files WHERE post_id=?
   → @unlink($realPath)  // 실제 파일 삭제
   → DELETE FROM post_files WHERE post_id=?


② 댓글 좋아요 삭제
   SELECT id FROM comments WHERE post_id=?
   → DELETE FROM likes WHERE target_type='comment' AND target_id=?


③ 댓글 hard delete
   DELETE FROM comments WHERE post_id=?


④ 게시글 좋아요 삭제
   DELETE FROM likes WHERE target_type='post' AND target_id=?


⑤ 스크랩 삭제
   DELETE FROM scraps WHERE post_id=?


⑥ 포인트 로그 삭제
   DELETE FROM point_logs WHERE ref_type='post' AND ref_id=?


⑦ 게시글 hard delete
   DELETE FROM posts WHERE id=?


⑧ 캐시 무효화
   DxCache::deletePrefix('board_list_' . $boardKey . '_');
```

7. API — AJAX 데이터 흐름

AJAX 요청은 core/api/ 폴더의 PHP 파일로 라우팅됩니다. 모든 API는 JSON을 반환하며, POST 요청은

CSRF 토큰 검증이 필수입니다.

7.1 댓글 등록 데이터 흐름

```
① HTTP POST /api/comment (AJAX)
   Content-Type: application/x-www-form-urlencoded
   Params: _csrf, post_id, content, parent_id(선택)
   │
   ▼ ② 검증
   dx_csrf_check();
   $postId = dx_post('post_id', '0', 'bigint');  // BIGINT 안전
   $content = DxSanitizer::editorContent($_POST['content']);
   // 게시판 댓글 레벨 확인, 로그인 여부
   │
   ▼ ③ DB INSERT
   $commentId = $db->insertWithMicrotimeId('comments', [
       'post_id'    => $postId,
       'member_id'  => $auth->get('id'),
       'content'    => $content,
       'parent_id'  => $parentId,
       'status'     => 1,
   ]);
   // 게시글 comment_count +1, popular_score 재계산
   UPDATE posts SET comment_count=comment_count+1 WHERE id=?
   │
   ▼ ④ 훅 실행
   dx_run_hook('dx_after_comment', ['comment_id'=>$commentId]);
   // → DxPoint::add('comment'), addExp('comment')
   // → DxNotification::add(게시글 작성자에게 알림)
   │
   ▼ ⑤ JSON 응답
   dx_json([
       'success'   => true,
       'comment_id'=> $commentId,
       'new_csrf'  => Secure::csrfToken(),  // v5.2.4 CSRF 갱신
   ]);
```

7.2 좋아요 토글 데이터 흐름

```
① HTTP POST /api/like
   Params: _csrf, post_id
   │
   ▼ ② 검증
   dx_csrf_check();
   $postId = dx_post('post_id', '0', 'bigint');
   // 내 글 좋아요 차단
   $postOwner = $db->value("SELECT member_id FROM posts WHERE id=?", [$postId]);
   if ((int)$postOwner === (int)$memberId) { dx_json(['success'=>false, ...]); }
   │
   ▼ ③ 중복 확인
   // 로그인: member_id로 확인 (IP 오판 방지)
   // 비로그인: ip로만 확인
   $exists = $db->row("SELECT id FROM likes WHERE target_type='post'",
       "AND target_id=? AND member_id=?", [$postId, $memberId]);
   │
```

```
   ▼ ④토글
   if ($exists) {  // 좋아요 취소
       DELETE FROM likes WHERE id=?
       UPDATE posts SET like_count=GREATEST(0, like_count-1) WHERE id=?
   } else {        // 좋아요 추가
       INSERT INTO likes (target_type, target_id, member_id, ip) VALUES(...)
       UPDATE posts SET like_count=like_count+1 WHERE id=?
       // DxNotification::add(게시글 작성자에게 알림)
       // dx_run_hook('dx_after_like', ['post_id', 'owner_id'])
       // → DxPoint::add(owner, 'like_recv')
   }
   │
   ▼ ⑤popular_score 재계산
   $raw = view_count*1 + like_count*5 + comment_count*3;
   $decay = max(0.1, 1.0 - floor($daysPassed/7) * 0.1);
   UPDATE posts SET popular_score=? WHERE id=?
   │
   ▼ ⑥JSON 응답
   dx_json(['success'=>true, 'like_count'=>$newCount, 'liked'=>$liked]);
```

7.3 파일 업로드 데이터 흐름

```
①HTTP POST /api/upload (또는 게시글 저장 시 멀티파트)
   Content-Type: multipart/form-data
   │
   ▼ ②검증 (Secure.php가 전담)
   // MIME 타입 + 확장자 이중 검증
   $ext = strtolower(pathinfo($filename, PATHINFO_EXTENSION));
   $mime = mime_content_type($tmpName);
   // 이중 확장자 차단: shell.php.jpg → 차단
   if (preg_match('/\.php/i', $filename)) → 거부
   // 실제 이미지 검증 (이미지 확장자인 경우)
   $imgInfo = @getimagesize($tmpName);
   if (!$imgInfo) → 거부 (가짜 이미지)
   │
   ▼ ③저장 경로 생성
   $savePath = DX_DATA . '/uploads/' . $boardKey . '/' . date('Y/m/');
   @mkdir($savePath, 0755, true);
   $savedName = uniqid() . '.' . $ext;  // 원본 파일명 대신 고유명 사용
   │
   ▼ ④파일 이동 + DB 저장
   move_uploaded_file($tmpName, $savePath . $savedName);
   $db->insertRow('post_files', [
       'post_id'   => $postId,
       'orig_name' => $originalName,
       'save_path' => $savePath . $savedName,
       'file_size' => $fileSize,
       'mime_type' => $mime,
   ]);
   │
   ▼ ⑤응답 (에디터 인라인 삽입용)
   dx_json(['url' => dx_base_url('api/download?id=' . $fileId)]);
```

8. 인증 — 로그인/로그아웃/회원가입 데이터 흐름

8.1 로그인 데이터 흐름

```
① HTTP POST /auth/login
   Params: login_id, password, remember(선택)
   │
   ▼ ② Auth::login($loginId, $password)
   // login_id 또는 email로 조회
   SELECT * FROM members
   WHERE (login_id=? OR email=?) AND status=1 LIMIT 1
   │
   ▼ ③ 비밀번호 검증
   password_verify($password, $user['password'])  // bcrypt
   // 실패 시: login_fail++ (10회 초과 → 계정 잠금)
   UPDATE members SET login_fail=login_fail+1 WHERE id=?
   │
   ▼ ④ 로그인 성공 처리
   UPDATE members SET login_fail=0, last_login=?, last_ip=? WHERE id=?
   // 세션 생성
   $_SESSION[$sessionKey] = [
       'id'    => $user['id'],
       'token' => hash_hmac('sha256', id+join_date, secret_key),
   ];
   // Remember Me 쿠키 발급 (remember_token 컬럼 존재 시)
   UPDATE members SET remember_token=?, remember_expires=? WHERE id=?
   setcookie('dx_remember', $userId.':'.randomHex(64), time()+86400, ...);
   │
   ▼ ⑤ 훅 실행
   dx_run_hook('dx_after_login', ['user' => $user]);
   // → DxPoint::add('login') (오늘 첫 로그인 시만)
   // → DxMemberMonitor::onLogin($userId)
   │
   ▼ ⑥ Redirect → 홈 또는 ?redirect= URL
```

8.2 회원가입 데이터 흐름

```
① HTTP POST /auth/register
   Params: login_id, password, password_confirm, name, email, ...
   │
   ▼ ② 검증 (Auth::register())
   // 아이디 형식: /^[a-zA-Z0-9_]{4,20}$/
   // 비밀번호: 8자 이상
   // 중복: $db->exists('members', ['login_id' => $loginId])
   //       $db->exists('members', ['email' => $email])
   │
   ▼ ③ 데이터 준비
   $data['password'] = password_hash($password, PASSWORD_BCRYPT);
   // 화이트리스트: SHOW COLUMNS로 실제 컬럼만 INSERT (SQL Injection 방지)
   $cols = array_column($db->rows("SHOW COLUMNS FROM members"), 'Field');
   $safeData = array_intersect_key($data, array_flip($cols));
   // 가입 경로 수집: IP, UA, device, OS, browser
   │
   ▼ ④ DB INSERT
   $id = $db->insertRow('members', $safeData);
   // 가입과 동시에 자동 로그인 세션 생성
```

```
   |
   ▼ ⑤ 훅 실행
   dx_run_hook('dx_after_register', ['user_id' => $id]);
   // → DxPoint::add('signup', 10), addExp('signup', 20)
```

8.3 Remember Me — 자동 로그인 흐름

```
// Auth::loadSession() → 세션 없을 때 tryRememberMe() 호출


① dx_remember 쿠키 파싱: "userId:token"
② DB 조회: SELECT * FROM members WHERE id=? AND status=1
③ PHP에서 토큰 비교:
   hash_equals($user['remember_token'], $token)
   // SQL WHERE 절 비교 금지 → 타이밍 공격 방지
④ 만료 확인:
   if (strtotime($user['remember_expires']) < time()) → 쿠키 삭제
⑤ 자동 로그인 성공 → 세션 복구
⑥ 토큰 롤링 갱신:
   $newToken = Secure::randomHex(64);  // 새 토큰 생성
   UPDATE members SET remember_token=?, remember_expires=? WHERE id=?
   setcookie('dx_remember', $userId.':'.newToken, ...);  // 30일 연장
// 매 요청마다 토큰 갱신 → 탈취된 토큰의 사용 기회 최소화
```

9. 훅(Hook) — 데이터 흐름 연결 메커니즘

훅은 CMS의 각 데이터 처리 단계를 서로 연결하는 핵심 메커니즘입니다. 핵심 파일을 수정하지 않고도 데이터 흐름에 새로운 처리를 삽입할 수 있습니다.

9.1 데이터 흐름 내 훅 발생 지점

훅 이름	유형	데이터 흐름 연결
dx_board_before	Action	게시판 핸들러 진입 시. 접근 제한, 로깅 삽입 가능
dx_board_before_save	Action (ref)	저장 전 $data 수정 가능. 플러그인이 추가 필드 처리
dx_after_write	Action	글 저장 후. → DxPoint::add(write), 외부 인덱싱
dx_board_after_save	Action	글 저장 후. 리다이렉트 URL도 변경 가능(ref)
dx_board_list_context	Action (ref)	목록 컨텍스트 수정 가능. 광고, 추가 데이터 주입
dx_board_view_context	Action (ref)	보기 컨텍스트 수정 가능. 관련글, 추천글 주입

dx_board_before_delete	**Action**	삭제 전. 취소 가능 (exit)
dx_board_after_delete	**Action**	삭제 후. 외부 인덱스 제거
dx_after_comment	**Action**	댓글 저장 후. → DxPoint, DxNotification
dx_after_like	**Action**	좋아요 후. → DxPoint(like_recv), DxNotification
dx_after_login	**Action**	로그인 후. → DxPoint(login), DxMemberMonitor
dx_after_logout	**Action**	로그아웃 후. → DxMemberMonitor
dx_after_register	**Action**	가입 후. → DxPoint(signup)
dx_head	**Action**	<head> 내. CSS/JS 추가
dx_body_bottom	**Action**	</body> 직전. DxPopup::render(), 스크립트
dx_editor_init	**Action**	에디터 초기화 요청 → dx_editor_render 호출
dx_editor_render	**Action**	에디터 플러그인이 등록. 실제 에디터 HTML 출력

9.2 포인트 데이터 흐름 연결

포인트는 훅을 통해 각 이벤트에 자동으로 연결됩니다. _dx_register_point_hooks()에서 모든 포인트 훅이 일괄 등록됩니다.

```
// STEP 4에서 일괄 등록 (index.php)
_dx_register_point_hooks();


// 각 이벤트 → 포인트/경험치 지급 흐름
dx_after_login    → DxPoint::add(login, +1pt, +2exp)   // 오늘 첫 로그인만
dx_after_write    → DxPoint::add(write, +5pt, +10exp)
dx_after_comment  → DxPoint::add(comment, +2pt, +5exp)
dx_after_like     → DxPoint::add(like_recv, +1pt, +2exp)  // 글 작성자에게
dx_after_register → DxPoint::add(signup, +10pt, +20exp)


// DxPoint::add() 내부 흐름
INSERT INTO point_log (member_id, type, point, ref_type, ref_id, ...)
UPDATE members SET point=point+?, exp=exp+? WHERE id=?
// 레벨업 확인
$newLevel = DxPoint::calcLevel($totalExp);
if ($newLevel > $currentLevel) {
    UPDATE members SET level=? WHERE id=?
}
```

10. 응답(Output) — 데이터 출력 흐름

처리된 데이터는 세 가지 방식으로 출력됩니다. HTML 렌더링, JSON 응답, Redirect가 있으며 각각 다른 경로를 따릅니다.

10.1 HTML 렌더링 흐름 (_brd_render)

```
// _brd_render($skinAction, $context)  내부 흐름

① DxBoardSkin::resolveView($skin, $skinAction)
   // 6단계 폴백 체인으로 스킨 파일 탐색

② 스킨 파일 안전 실행 (에러 격리)
   set_error_handler(...);  // notice/warning → 로그만, 계속 실행
   ob_start();
   try {
       extract($context, EXTR_SKIP);  // $context를 변수로 풀기
       include $skinFile;              // 스킨 파일 실행
   } catch (Exception $e) {
       echo '<div class="error">...';  // 에러 박스 표시
   }
   $dx_content = ob_get_clean();  // 스킨 출력 캡처
   restore_error_handler();

③ 레이아웃 파일 실행
   $layoutFile = DxTheme::getInstance()->resolve('layout/main.php');
   extract($context, EXTR_SKIP);  // 레이아웃에 컨텍스트 전달
   require $layoutFile;
   // layout/main.php 내부에서 echo $dx_content; 로 본문 삽입


// layout/main.php 내 데이터 흐름
DxSeo::render()            → <head> SEO 메타 출력
dx_run_hook('dx_head')     → CSS/JS 삽입
echo $dx_content           → 스킨 본문 삽입
dx_run_hook('dx_body_bottom') → 팝업, 분석 스크립트
```

10.2 JSON 응답 흐름 (dx_json)

```
// dx_json($data, $code=200)
while (ob_get_level() > 0) ob_end_clean();  // 버퍼 비우기
http_response_code($code);
header('Content-Type: application/json; charset=utf-8');
echo json_encode($data, JSON_UNESCAPED_UNICODE);
exit;


// 표준 응답 구조
// 성공: {success:true, data:..., new_csrf:'...'(POST 후 CSRF 갱신)}
// 실패: {success:false, message:'...'}, HTTP 4xx/5xx
```

10.3 Redirect 흐름 (dx_redirect)

```
// dx_redirect($url, $code=302)
$GLOBALS['DX_SHUTTING_DOWN'] = true;  // DB 오류 시 exit 방지 플래그


// 출력 버퍼 전체 비우기 (카페24 등 공유호스팅 호환)
while (ob_get_level() > 0) ob_end_clean();


// HTTP Location 헤더
if (!headers_sent()) {
    header('Location: ' . $url, true, $code);
}


// 항상 HTML 폴백도 출력 (header() 실패 환경 대응)
echo '<meta http-equiv="refresh" content="0;url=' . $safeUrl . '">';
echo '<script>location.replace(' . json_encode($url) . ');</script>';
exit;
```

11. 데이터 흐름 연결 전체 매트릭스

주요 시나리오별로 데이터가 어떤 컴포넌트를 거쳐 흐르는지 한눈에 보여주는 매트릭스입니다.

시나리오	Input	CSRF	Auth	Sanitize	DB	Cache	Output
게시글 목록 조회	`dx_get()`	—	isLoggedIn()	—	`rows(list,notices)`	**READ**	HTML
게시글	`dx_get(bigint)`	—	isLoggedIn()	—	`row(post)+JOIN`	—	HTML

보기							
게시글 작성	dx_post()	✔	isLoggedIn()	text+editor	insertWithId()	DELETE	Redirect
게시글 삭제	dx_post()	✔	isAdmin/owner	—	cascade DELETE	DELETE	Redirect
댓글 등록	dx_post()	✔	isLoggedIn()	editorContent	insertWithId()	—	JSON
좋아요 토글	dx_post(bigint)	✔	선택	—	INSERT/DELETE	—	JSON
파일 업로드	$_FILES	✔	isLoggedIn()	filename	insertRow()	—	JSON
로그인	dx_post()	✔	—	text	row(member)	—	Redirect
회원가	dx_post()	✔	—	text	insertRow()	—	Redirect

입							
자동 로그인	$_COOKIE	—	tryRememberMe	–	row+UPDATE	READ	세션복구

□ 데이터 흐름 핵심 원칙 요약

1) BIGINT는 문자열로: 모든 BIGINT ID는 (int) 캐스팅 금지, 문자열 유지
2) 모든 POST는 CSRF 먼저: dx_csrf_check()가 첫 번째 줄
3) 저장 전 Sanitize: text() 또는 editorContent()로 반드시 정제
4) 캐시는 Write 즉시 무효화: deletePrefix()로 관련 캐시 일괄 삭제
5) 훅으로 연결: 포인트·알림·모니터링은 훅으로만 연결, 핵심 파일 수정 없음
6) 삭제는 cascade: 게시글 삭제 시 파일·댓글·좋아요·스크랩 모두 정리
7) 에러 격리: 스킨/extend 오류가 전체 응답을 죽이지 않도록 ob_start + try/catch
8) Redirect 후 DX_SHUTTING_DOWN: 리다이렉트 후 발생하는 DB 오류가 exit로 이어지지 않게 보호