

## PDF에 포함된 파일 경로 목록

- adapters/BaseAdapter.php
- adapters/DbAdapter.php
- adapters/TaskDfToJsonAdapter.php
- adapters/TaskJsonAdapter.php
- adapters/TaskJsonToMermaidAdapter.php
- classes/App.php
- classes/Array/ArrayHelper.php
- classes/Auth/Session.php
- classes/Cache/CacheMem.php
- classes/Cache/CachedMem.php
- classes/Calendars/Calendars.php
- classes/Cipher/AES256Hash.php
- classes/Cipher/Base64UrlEncoder.php
- classes/Cipher/CipherGeneric.php
- classes/Cipher/HashEncoder.php
- classes/Cipher/PasswordHash.php
- classes/Cipher/ROT13Encoder.php
- classes/Date/DateTimez.php
- classes/Date/DateTimezPeriod.php
- classes/Db/CipherMysqlAes256Cbc.php
- classes/Db/CipherPgsqlAes.php
- classes/Db/CipherPgsqlAes256Cbc.php
- classes/Db/CipherPgsqlBasic.php
- classes/Db/DbCipherGeneric.php
- classes/Db/DbCipherInterface.php
- classes/Db/DbCouch.php
- classes/Db/DbInterface.php
- classes/Db/DbManager.php
- classes/Db/DbMySql.php
- classes/Db/DbPgSql.php
- classes/Db/DbResultCouch.php
- classes/Db/DbResultSql.php
- classes/Db/QueryBuilderAbstractCouch.php
- classes/Db/QueryBuilderAbstractSql.php
- classes/Db/WhereCouch.php
- classes/Db/WhereHelper.php
- classes/Db/WhereInterface.php
- classes/Db/WhereSql.php
- classes/Dir/DirInfo.php
- classes/Dir/DirObject.php
- classes/Enum/EnumValueStorage.php
- classes/File/Download.php
- classes/File/FileRemove.php
- classes/File/FileSize.php
- classes/File/Storage.php
- classes/File/Upload.php
- classes/Ftp/Ftp.php
- classes/Ftp/FtpObject.php
- classes/Html/XssChars.php

- classes/Http/HttpRequest.php
- classes/Http/HttpResponse.php
- classes/Http/HttpUrlFilter.php
- classes/Image/ImageExif.php
- classes/Image/ImageGDS.php
- classes/Image/ImageViewer.php
- classes/Json/JsonDecoder.php
- classes/Json/JsonEncoder.php
- classes/Log.php
- classes/Mail/MailSend.php
- classes/Model.php
- classes/Paging/Relation.php
- classes/R.php
- classes/Random/Random.php
- classes/Request/FormValidation.php
- classes/Request/Request.php
- classes/Request/Validation.php
- classes/Route/DbRoute.php
- classes/Route/JsonRoute.php
- classes/Route/RouteLoader.php
- classes/Sharding/ConsistentHashing.php
- classes/Sharding/JumpHashing.php
- classes/Sharding/ShardManager.php
- classes/Strings/StringTools.php
- classes/TaskFlow.php
- classes/Text/TextKeyword.php
- classes/Text/TextUtil.php
- classes/Uuid/UuidGenerator.php
- composer.json
- interfaces/BaseAdapterInterface.php
- interfaces/DeleteInterface.php
- interfaces/DoInterface.php
- interfaces/EditInterface.php
- interfaces/EditUpdateInterface.php
- interfaces/EnumInterface.php
- interfaces/EnumValueInterface.php
- interfaces/ImageCompressorInterface.php
- interfaces/InsertInterface.php
- interfaces/ListInterface.php
- interfaces/PostInsertInterface.php
- interfaces/PostInterface.php
- interfaces/ReplInterface.php
- interfaces/ReplyInterface.php
- interfaces/ReplyReplInterface.php
- interfaces/RouteInterface.php
- interfaces/ShardingStrategyInterface.php
- interfaces/UpdateInterface.php
- interfaces/ViewInterface.php
- res/sysmsg.json
- res/sysmsg\_en.json
- res/sysmsg\_jp.json
- res/sysmsg\_ko.json
- res/sysmsg\_zh.json

- task/EnumImportBasicTask.php
- task/ExceptionBasicTask.php
- task/HttpRequestTask.php
- task/PagingRelationBasicTask.php
- task/QueryDeleteBasicTask.php
- task/QueryInsertBasicTask.php
- task/QueryReplyBasicTask.php
- task/QuerySelectBasicTask.php
- task/QueryUpdateBasicTask.php
- task/QueryWhereCaseBasicTask.php
- task/RequestedFetchBasicTask.php
- task/SortByFidBasicTask.php
- task/TotalRecordBasicTask.php
- task/ValidationBasicTask.php
- traits/DelimitedStringTrait.php
- traits/EntryArrayTrait.php
- traits/EnumInstanceTrait.php
- traits/FidTrait.php
- traits/ImageCompoessorEditjsTrait.php
- traits/ImageCompressorBase64Trait.php
- traits/NullableValidationTrait.php
- traits/PasswordHashTrait.php
- traits/TimeZoneTrait.php
- traits/UniqueldTrait.php
- utils/Requested.php

--- 파일 경로: adapters/BaseAdapter.php ---

```
<?php
namespace Flex\Banana\Adapters;

use Flex\Banana\Interfaces\BaseAdapterInterface;

class BaseAdapter implements BaseAdapterInterface{
    public const __version = '0.1';

    public function getVersion(): string
    {
        return static::__version;
    }
}
```

--- 파일 경로: adapters/DbAdapter.php ---

```
<?php
namespace Flex\Banana\Adapters;

use Flex\Banana\Classes\Db\DbManager;
use Flex\Banana\Adapters\BaseAdapter;

class DbAdapter extends BaseAdapter{
    public function __construct(
        public DbManager $db
    ){
    }
}
```

--- 파일 경로: adapters/TaskDfToJsonAdapter.php ---

```
<?php
namespace Flex\Banana\Adapters;

use DOMDocument;
use DOMXPath;
use Flex\Banana\Classes\TaskFlow;
use Flex\Banana\Classes\Log;

final class TaskDfToJsonAdapter
{
    public const __version = '0.1.0';
    public function __construct(
        private TaskFlow $task
    ){
    }

    public function execute(array $posts): array
    {
        $result = [];
        $flow_json = json_decode($posts['contents'],true);
        // print_r($flow_json);
        foreach ($flow_json as $node) {
            if (!isset($node['data'])) continue;
            $result[] = $node['data'];
        }
        // print_r($result);

        $posts['flow'] = json_encode($result,JSON_UNESCAPED_UNICODE);
        return $posts;
    }
}
```

--- 파일 경로: adapters/TaskJsonAdapter.php ---

```

<?php
namespace Flex\Banana\Adapters;

use Flex\Banana\Classes\TaskFlow;
use Flex\Banana\Classes\Log;

final class TaskJsonAdapter
{
    public const __version = '0.8.0';
    private array $workflow;
    private array $bannedEnvVars = [];
    private array $bannedConstants = [];

    // 무한루프 처리값 설정
    private const MAX_EXECUTION_STEPS = 100;

    public function __construct(array $workflow)
    {
        $this->workflow = $workflow;
    }

    // 금지할 환경 변수 목록을 배열로 한번에 설정하는 메소드
    public function setBannedEnvVars(array $bannedVars): self
    {
        $this->bannedEnvVars = $bannedVars;
        return $this;
    }

    // 금지할 상수 목록을 배열로 한번에 설정하는 메소드
    public function setBannedConstants(array $bannedConstants): self
    {
        $this->bannedConstants = $bannedConstants;
        return $this;
    }

    public function process(TaskFlow $flow): TaskFlow
    {
        Log::d("====> JsonAdapter: 워크플로우 처리 시작.");

        $index = 0;
        $count = count($this->workflow);

        // SECURITY: DoS 방지를 위한 스텝 카운터 초기화
        $stepCounter = 0;

        // idMap: id => step index (for fast lookup)
        $idMap = [];
        foreach ($this->workflow as $i => $step) {
            if (isset($step['id'])) {
                $idMap[$step['id']] = $i;
            }
        }

        while ($index < $count)
        {
            $step = $this->workflow[$index];
            Log::d("JsonAdapter: 스텝 실행 중 - " . json_encode($step));

            // 무한루프 처리
            if (++$stepCounter > self::MAX_EXECUTION_STEPS) {
                throw new \Exception("****최대 실행 스텝(" . self::MAX_EXECUTION_STEPS . "회)을 초과****");
            }

            try {
                $type = $step['type'] ?? 'class';

                // switch/if 조건 분기 처리
                if ($type === 'switch' || $type === 'if') {
                    $newIndex = $this->handleConditionalStep($flow, $step, $idMap);
                    if ($newIndex !== null) {
                        $index = $newIndex;
                        continue;
                    }
                }

                // 일반 실행 처리
                match ($type) {
                    'method' => $this->handleMethodStep($flow, $step),
                }
            }
        }
    }
}

```

```

        'function' => $this->handleFunctionStep($flow, $step),
        'class' => $this->handleClassStep($flow, $step),
        default => throw new \Exception("Unknown step type: " . $type),
    };

    $newIndex = $this->handleGoStep($step, $idMap);
    if ($newIndex !== null) {
        $index = $newIndex;
        continue;
    }
} catch (\Throwable $e) {
    Log::e("JsonAdapter: 스텝 처리 중 예외 발생 - ", $e->getMessage());
    throw new \Exception($e->getMessage());
}

$index++;
}

Log::d("<---- JsonAdapter: 워크플로우 처리 완료".PHP_EOL);
return $flow;
}

private function handleMethodStep(TaskFlow $flow, array $step): void
{
    $resolve = fn($v) => $this->resolveContextReference($flow, $v);

    $objectName = $step['object'];
    $method      = $step['method'];
    $params       = array_map($resolve, $step['params'] ?? []);

    if (str_starts_with($objectName, 'enum::')) {
        $enumKey = substr($objectName, strlen('enum::'));

        if (enum_exists($enumKey)) {
            $target = $enumKey::cases()[0];
        }
        // 기본 네임 스페이스 폴백
        elseif (enum_exists("\Columns\\" . $enumKey)) {
            $target = ("\Columns\\" . $enumKey)::cases()[0];
        } else {
            throw new \Exception("Enum class {$enumKey} not found");
        }
    } else {
        $target = $flow->$objectName ?? null;
    }
    Log::d("handleMethodStep: {$objectName}->{$method} 호출 중, 파라미터: " . json_encode($params));

    if (!is_object($target) || (!is_callable([$target, $method]) && !method_exists($target, '__call'))) {
        throw new \Exception("Method {$method} not callable on object {$objectName}");
    }

    $result = call_user_func_array([$target, $method], $params);
    Log::d("handleMethodStep: 결과: " . json_encode($result));

    foreach (($step['outputs'] ?? []) as $ctxKey => $resultKey) {
        $flow->$ctxKey = $resultKey === 'self' ? $target : ($resultKey === '@return' ? $result : null);
    }
}

private function handleFunctionStep(TaskFlow $flow, array $step): void
{
    $resolve = function($v) use (&$resolve, $flow) {
        if (is_array($v)) {
            return array_map($resolve, $v);
        }
        return $this->resolveContextReference($flow, $v);
    };

    $function = $step['function'] ?? null;
    if (!$function) {
        throw new \Exception("Function '{$function}' not callable.");
    }

    // 열거적인 방법 호출 지원
    if (!empty($step['method']) && str_contains($function, '::')) {
        [$cls, $case] = explode('::', $function);
        if (enum_exists($cls)) {
            $enumInstance = constant("\{$cls}::{$case}");
            $method = $step['method'];

```

```

        if (method_exists($enumInstance, $method)) {
            $params = array_map($resolve, $step['params'] ?? []);
            $result = call_user_func_array([$enumInstance, $method], $params);
            Log::d('handleFunctionStep (enum): {${cls}}->{${method}} 호출 결과: " . json_encode($result));
            foreach (($step['outputs'] ?? []) as $ctxKey => $resultKey) {
                $flow->$ctxKey = $resultKey === '@return' ? $result : null;
            }
            return;
        } else {
            throw new \Exception("Method {${method}} not found on enum {${cls}}");
        }
    }
}

if (!is_callable($function)) {
    throw new \Exception("Function '{${function}}' not callable.");
}

$params = array_map($resolve, $step['params'] ?? []);
Log::d("handleFunctionStep: 함수 {${function}} 호출 중, 파라미터: " . json_encode($params));

// 변수 참조해야 하는 함수들 지원 (확장됨)
if (in_array($function, ['array_push', 'array_unshift', 'array_shift', 'array_pop', 'sort', 'rsort', 'asort', 'ksort', 'usort', 'array_reverse'], true)) {
    if (!empty($params) && is_array($params[0])) {
        $ref = &$params[0]; // 참조

        if (in_array($function, ['sort', 'rsort', 'asort', 'ksort', 'usort'], true)) {
            $function($ref);
            $result = $ref;
        } else {
            $result = $function($ref);
        }

        // 이 라인 중요: 결과 배열을 다시 flow에 반영
        foreach (($step['outputs'] ?? []) as $ctxKey => $resultKey) {
            if ($resultKey === '@return') {
                $flow->$ctxKey = $result;
            } elseif ($resultKey === 'self') {
                $flow->$ctxKey = $ref; // 참조 배열로 덮어쓰기
            }
        }
    } else {
        throw new \Exception("{${function}} requires the first parameter to be an array.");
    }
} else {
    $result = call_user_func_array($function, $params);
}
Log::d("handleFunctionStep: 결과: " . json_encode($result));

// '속성'이 설정된 경우 결과 오브젝트에서 속성 추출 지원
if (!empty($step['property']) && is_object($result) && property_exists($result, $step['property'])) {
    $result = $result->{$step['property']};
}

foreach (($step['outputs'] ?? []) as $ctxKey => $resultKey) {
    if ($resultKey === '@return') {
        // 항상 컨텍스트에 최신 결과 반영
        $flow->$ctxKey = $result;
    } elseif (is_string($resultKey) && str_starts_with($resultKey, '@')) {
        $resolvedKey = substr($resultKey, 1);
        $flow->$ctxKey = $flow->$resolvedKey ?? null;
    } elseif (is_array($result) && isset($result[$resultKey])) {
        $flow->$ctxKey = $result[$resultKey];
    }
}

}

private function handleClassStep(TaskFlow $flow, array $step): void
{
    $resolve = function($v) use (&$resolve, $flow) {
        if (is_array($v)) {
            return array_map($resolve, $v);
        }
        return $this->resolveContextReference($flow, $v);
    };

    try {
        $flow->do(function(TaskFlow $task) use ($step, $resolve) {
            try {

```

```

$class = $step['class'] ?? null;
$method = $step['method'] ?? null;

if ($class && $class[0] === '@') {
    $class = $task->{substr($class, 1)} ?? null;
}

if (!class_exists($class)) {
    throw new \Exception("Class {$class} not found");
}

$constructArgs = [];
if (empty($step['inputs']['@construct'])) {
    $constructArgs = array_map($resolve, $step['inputs']['@construct']);
    unset($step['inputs']['@construct']);
} elseif (empty($step['inputs']['construct'])) {
    $constructArgs = array_map($resolve, $step['inputs']['construct']);
    unset($step['inputs']['construct']);
}
Log::d("handleClassStep: {$class} 인스턴스 생성 중, 생성자 인자: " . json_encode($constructArgs));

$instance = new $class(...$constructArgs);
$params = array_map($resolve, $step['params'] ?? []);
if (empty($step['inputs'])) {
    foreach ($step['inputs'] as $key => $ref) {
        if (isset($task->{$ref})) {
            $params[$key] = $task->{$ref};
        }
    }
}
Log::d("handleClassStep: {$class}->{$method} 호출 중, 파라미터: " . json_encode($params));

$result = call_user_func_array([$instance, $method], $params);
if (empty($step['property'])) {
    if (is_object($result) && property_exists($result, $step['property'])) {
        $result = $result->{$step['property']};
    } elseif (is_array($result) && isset($result[$step['property']])) {
        $result = $result[$step['property']];
    } elseif (is_object($instance) && property_exists($instance, $step['property'])) {
        $result = $instance->{$step['property']};
    }
}
Log::d("handleClassStep: 결과: " . json_encode($result));

// @return 해상도에 사용되는 경우 컨텍스트로 해결된 결과를 주입.
foreach ($step['outputs'] ?? [] as $ctxKey => $resultKey) {
    if (is_string($resultKey) && str_starts_with($resultKey, '@')) {
        $resolvedKey = substr($resultKey, 1);
        if (isset($task->{$resolvedKey})) {
            $step['outputs'][$ctxKey] = $task->{$resolvedKey};
        }
    }
}

foreach ((($step['outputs'] ?? [])) as $ctxKey => $resultKey) {
    if ($resultKey === '@class') {
        $task->{$ctxKey} = $instance;
    } elseif ($resultKey === '@return') {
        $task->{$ctxKey} = $result;
    } elseif (is_array($result) && isset($result[$resultKey])) {
        $task->{$ctxKey} = $result[$resultKey];
    }
}

return $task;
} catch (\Throwable $e) {
    Log::e($e->getMessage());
    throw new \Exception($e->getMessage());
}
});
} catch (\Throwable $e) {
    Log::e("handleClassStep 오류: " . $e->getMessage());
    throw new \Exception($e->getMessage());
}
}

private function resolveContextReference(TaskFlow $flow, $value)
{
    if ($value === '@task') {

```



```

return $flow;
}

if (is_string($value) && str_starts_with($value, '@'))
{
    $ref = substr($value, 1);

    // ENV::
    if (str_starts_with($ref, 'ENV::')) {
        $envVar = substr($ref, 5);
        // SECURITY CHECK: 금지된 환경 변수인지 확인
        if (in_array($envVar, $this->bannedEnvVars, true)) {
            throw new \Exception("보안 오류: 금지된 환경 변수('@ENV::{$envVar}')에 대한 접근이 차단되었습니다.");
        }
        return getenv($envVar) ?: null;
    }

    // DEFINE::
    if (str_starts_with($ref, 'DEFINE::')) {
        $const = substr($ref, 8);
        // SECURITY CHECK: 금지된 상수인지 확인
        if (in_array($const, $this->bannedConstants, true)) {
            throw new \Exception("보안 오류: 금지된 상수('@DEFINE::{$const}')에 대한 접근이 차단되었습니다.");
        }
        return defined($const) ? constant($const) : null;
    }

    // R::method.key1.key2
    if (preg_match('/^R::([a-zA-Z_]+)((?!\.[a-zA-Z0-9_]+)*)$/', $ref, $match)) {
        $method = $match[1];
        $path = ltrim($match[2], '.');
        $args = explode('.', $path);
        $base = call_user_func(["\Flex\Banana\Classes\R", $method], $args[0] ?? null);

        foreach (array_slice($args, 1) as $key) {
            if (is_array($base) && array_key_exists($key, $base)) {
                $base = $base[$key];
            } else {
                return null;
            }
        }
        return $base;
    }

    // @enums::EnumName()
    if (preg_match('/^@([^\:]+)::([^\(]+)\(\)$/ ', $ref, $matches)) {
        [$_, $ctxKey, $enumKey] = $matches;
        $ctx = $flow->$ctxKey ?? null;
        if (is_array($ctx) && isset($ctx[$enumKey]) && $ctx[$enumKey] instanceof \BackedEnum) {
            return $ctx[$enumKey]->value;
        }
    }

    // @enums::EnumName || @object::property
    $parts = explode(':', $ref, 2);
    if (count($parts) === 2) {
        [$ctxKey, $enumKey] = $parts;
        $ctx = $flow->$ctxKey ?? null;
        if (is_array($ctx) && array_key_exists($enumKey, $ctx)) {
            $resolved = $ctx[$enumKey];
            if (is_string($resolved) && enum_exists($resolved)) {
                return $resolved::cases()[0];
            }
            if ($resolved instanceof \UnitEnum) {
                return $resolved;
            }
        }
        return $resolved;
    }

    // @object.property.subkey
    $parts = explode('.', $ref);
    $ctx = $flow->{$parts[0]} ?? null;
    if ($ctx === null) {
        Log::w("resolveContextReference:", $value, "→ null (not found)");
    }
    foreach (array_slice($parts, 1) as $key) {
        if ($ctx === null) {
            Log::w("resolveContextReference:", $value, "→ null (not found)");
        }
    }
}

```

```

        break;
    }
    $ctx = is_array($ctx) && isset($ctx[$key]) ? $ctx[$key] : null;
    if ($ctx === null) {
        Log::w("resolveContextReference:", $value, "→ null (not found)");
    }
}
return $ctx;

// @enum::IdEnum.value 또는 @enum::IdEnum()
if (preg_match('/^enum::([a-zA-Z0-9_\\]+)(?\\.\\(w+))?$/', $ref, $match)) {
    $enumClass = $match[1];
    $enumProp = $match[2] ?? null;

    // 네임스페이스 보완
    if (!enum_exists($enumClass) && enum_exists("\\Columns\\{$enumClass}")) {
        $enumClass = "\\Columns\\{$enumClass}";
    }

    if (!enum_exists($enumClass)) {
        Log::w("resolveContextReference:", $ref, "→ enum class not found");
        return null;
    }

    $enumInstance = $enumClass::cases()[0];

    if ($enumProp === 'value') {
        return $enumInstance->value;
    } elseif ($enumProp && method_exists($enumInstance, $enumProp)) {
        return $enumInstance->{$enumProp}(); // 메서드 실행
    } elseif ($enumProp) {
        return $enumInstance->{$enumProp} ?? null; // 속성
    }

    return $enumInstance; // 기본 객체 반환
}

return $value;
}

private function handleConditionalStep(TaskFlow $flow, array $step, array $idMap): ?int
{
    $condition = (string) $this->resolveContextReference($flow, $step['condition'] ?? '');
    $outputs = $step['outputs'] ?? [];
    Log::d('condition', $condition);
    // print_r($outputs);

    // □ 명시적으로 키 존재 여부 확인
    if (!array_key_exists($condition, $outputs) && !array_key_exists('default', $outputs)) {
        throw new \Exception("조건 {$condition}에 해당하는 분기 또는 기본(default) 분기가 없습니다.");
    }

    $nextId = $outputs[$condition] ?? $outputs['default'];
    Log::d('nextId', $nextId);
    if ($nextId && isset($idMap[$nextId])) {
        $nextIndex = $idMap[$nextId];
        return $nextIndex;
    }

    throw new \Exception("다음 단계 ID가 유효하지 않음: " . json_encode($nextId));
}

private function handleGoStep(array $step, array $idMap): ?int
{
    if (!array_key_exists('go', $step) || $step['go'] === null || $step['go'] === '') {
        return null;
    }

    $nextId = $step['go'];
    if (isset($idMap[$nextId])) {
        $nextIndex = $idMap[$nextId];
        return $nextIndex;
    } else {
        throw new \Exception("Go step id '{$nextId}' not found");
    }
}
}

```

--- 파일 경로: adapters/TaskJsonToMermaidAdapter.php ---

```
<?php
namespace Flex\Banana\Adapters;

use Flex\Banana\Classes\TaskFlow;
use Flex\Banana\Classes\Log;

final class TaskJsonToMermaidAdapter
{
    public const __version = '0.3.1';
    private array $workflow;

    public function process(array $workflow): string
    {
        $tasks = [];
        if (isset($workflow['tasks']) && is_array($workflow['tasks'])) {
            $tasks = $workflow['tasks'];
        } else {
            $tasks = $workflow;
        }

        if (empty($tasks)) {
            throw new \Exception("Mermaid로 변환할 Task 데이터가 없습니다.");
        }

        $this->workflow = $tasks;

        $nodes = [];
        $edges = [];
        $taskCount = count($this->workflow);

        // 노드(작업) 정의
        foreach ($this->workflow as $index => $task)
        {
            $taskId = !empty($task['id']) ? $task['id'] : "step{$index}";
            $title = htmlspecialchars($task['title'] ?? $taskId, ENT_QUOTES, 'UTF-8');
            $taskType = $task['type'] ?? null;

            if ($taskType === 'if' || $taskType === 'switch') {
                $nodes[] = "    {$taskId}{{\"{$title}\"}}";
            } else {
                $nodes[] = "    {$taskId}[\"{$title}\"]";
            }
        }

        // 연결(엣지) 정의
        foreach ($this->workflow as $index => $task)
        {
            $currentId = !empty($task['id']) ? $task['id'] : "step{$index}";
            $taskType = $task['type'] ?? null;
            $isException = isset($task['class']) && str_contains($task['class'], 'ExceptionBasicTask');

            if ($taskType === 'if' || $taskType === 'switch')
            {
                if (isset($task['outputs']) && is_array($task['outputs'])) {
                    foreach ($task['outputs'] as $condition => $target) {
                        // [수정] target 값도 비어있지 않은지 확인
                        if(!empty($target)) {
                            $label = $condition === 'default' ? 'else' : $condition;
                            $edges[] = "    {$currentId} -- \"{$label}\" --> {$target}";
                        }
                    }
                }
            }
            // [수정] 'go'의 값이 존재하는지와 비어있지 않은지를 함께 확인
            elseif (isset($task['go']) && !empty($task['go'])) {
                $edges[] = "    {$currentId} --> {$task['go']}";
            }
            elseif ($isException || $index === $taskCount - 1) {
                // 종료 노드는 연결 없음
            } else {
                $nextIndex = (int)$index + 1;
            }
        }
    }
}
```

```

        if (isset($this->workflow[$nextIndex])) {
            $nextTask = $this->workflow[$nextIndex];
            $nextId = !empty($nextTask['id']) ? $nextTask['id'] : 'step' . $nextIndex;
            $edges[] = "    {$currentId} --> {$nextId}";
        }
    }
}

// 최종 Mermaid 문자열 조합
$mermaidParts = ["graph TD"];
$mermaidParts = array_merge($mermaidParts, $nodes, array_unique($edges));

return implode("\n", $mermaidParts);
}
}

```

### --- 파일 경로: classes/App.php ---

```

<?php
namespace Flex\Banana\Classes;

# 접속에 따른 디바이스|브라우저등 정보
final class App
{
    public const __version    = '1.2';
    public static $platform  = 'Nan';
    public static $browser   = 'Nan';
    public static $host;
    public static $language  = 'ko';
    public static $locale    = 'ko_KR';
    public static $http_referer = null;
    public static $ip_address = '';
    public static $protocol   = 'Nan';
    public static $version    = '1.0';

    public static function init() : void
    {
        self::detectPlatformAndBrowser();
        self::setHttpReferer();
        self::setLanguageAndLocale();
        self::setProtocolAndHost();
        self::$ip_address = self::getClientIp();
    }

    private static function detectPlatformAndBrowser() : void
    {
        $agent = $_SERVER['HTTP_USER_AGENT'] ?? '';

        $platforms = [
            'Linux' => 'Linux',
            'iPod' => 'iPod',
            'iPhone' => 'iPhone',
            'iPad' => 'iPad',
            'Windows Phone' => 'Windows Phone',
            'Windows CE' => 'Windows CE',
            'Igtelecom' => 'Igtelecom',
            'Android' => 'Android',
            'Macintosh' => 'Mac',
            'mac os x' => 'Mac',
            'Windows' => 'Windows',
            'Win32' => 'Windows'
        ];

        foreach ($platforms as $key => $value) {
            if (striestr($agent, $key)) {
                self::$platform = $value;
                break;
            }
        }

        $browsers = [
            'MSIE' => 'Explorer',
            'Firefox' => 'Firefox',
            'Chrome' => 'Chrome',

```

```

        'Safari' => 'Safari',
        'Opera' => 'Opera',
        'Netscape' => 'Netscape'
    ];

    foreach ($browsers as $key => $value) {
        if (strpos($agent, $key) && ($key !== 'MSIE' || !strpos($agent, 'Opera'))) {
            self::$browser = $value;
            break;
        }
    }
}

private static function setHttpReferer() : void
{
    self::$http_referer = $_SERVER['HTTP_REFERER'] ?? null;
}

private static function setLanguageAndLocale() : void
{
    if (isset($_SERVER['HTTP_ACCEPT_LANGUAGE'])) {
        $hal = explode(',', strtolower($_SERVER['HTTP_ACCEPT_LANGUAGE'], [';' => ';', '-' => '_']));
        foreach ($hal as $v) {
            if (strpos($v, self::$language . '_') !== false) {
                self::$locale = $v;
                break;
            }
        }
        self::$language = substr($_SERVER['HTTP_ACCEPT_LANGUAGE'], 0, 2);
    }
}

private static function setProtocolAndHost() : void
{
    self::$protocol = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === "on") ? 'https' : 'http';
    self::$host = isset($_SERVER['HTTP_HOST']) ? self::$protocol . "://" . $_SERVER['HTTP_HOST'] : "";
}

public static function getClientIp() : string
{
    $ipSources = [
        'HTTP_CLIENT_IP',
        'HTTP_X_FORWARDED_FOR',
        'HTTP_X_FORWARDED',
        'HTTP_FORWARDED_FOR',
        'HTTP_FORWARDED',
        'REMOTE_ADDR'
    ];

    foreach ($ipSources as $source) {
        if (isset($_SERVER[$source])) {
            return $_SERVER[$source];
        }
    }

    return "";
}
}

```

--- 파일 경로: classes/Array/ArrayHelper.php ---

```

<?php
namespace Flex\Banana\Classes\Array;

# 배열 사용에 도움을 주는 클래스
class ArrayHelper
{
    public const __version = '1.3.2';
    public function __construct(
        private array $value
    ){
    }

    # 멀티배열 키의 값으로 소팅 [{}, {}]
    # sort : asc | desc

```

```

# key : 소팅 비교할 키네임
public function sorting(string $key, string $sorting = 'ASC') : ArrayHelper
{
    $sorting = strtoupper($sorting);
    usort($this->value, function($a, $b) use ($key,$sorting) {
        return match($sorting){
            'DESC' => $this->desc($a[$key],$b[$key]),
            'ASC' => $this->asc($a[$key],$b[$key])
        };
    });
    return $this;
}

# 멀티배열 중 원하는 값의 첫번째 키를 찾아낸다
public function find(string $key, mixed $val) : ArrayHelper
{
    $index = $this->findIndex($key, $val);
    if($index > -1){
        $this->value = $this->value[$index];
    }else $this->value = [];
    return $this;
}

# 멀티배열 중 원하는 값의 전체를 찾아 낸다
public function findAll(string $key,...$params) : ArrayHelper
{
    $values = $params;

    # 배열로 들어왔는지 체크
    if(is_array($params[0])){
        $values = $params[0];
    }

    $result = [];
    $argv = array_column($this->value, $key);
    foreach($argv as $idx => $val){
        foreach($values as $fval){
            if($val == $fval){
                $result[] = $this->value[$idx];
            }
        }
    }
    $this->value = $result;
    return $this;
}

# select 여러키 중에서 원하는 키만 뽑아서 배열에 담기
public function select(...$keys) : ArrayHelper
{
    $this->value = array_map(function($item) use ($keys) {
        return array_intersect_key($item, array_flip($keys));
    }, $this->value);

    return $this;
}

# 멀티 키 => 밸류 값 찾기 OR
public function findWhere (array $params, string $operator='AND') : ArrayHelper
{
    $result = [];
    $find_cnt = count($params);
    $sup_operator = strtoupper($operator);
    foreach ($this->value as $key => $value)
    {
        if($sup_operator == 'AND')
        {
            $find_cnt = 0;
            foreach ($params as $fk => $fv) {
                if (isset($value[$fk])){
                    if(is_array($fv)){
                        $condition = $fv[0];
                        $fvalue = $fv[1];
                        switch($condition){
                            case '>': if($value[$fk] > $fvalue) $find_cnt++;break;
                            case '>=': if($value[$fk] >= $fvalue) $find_cnt++;break;
                            case '<': if($value[$fk] < $fvalue) $find_cnt++; break;
                            case '<=': if($value[$fk] <= $fvalue) $find_cnt++; break;
                            case '=': if($value[$fk] = $fvalue) $find_cnt++;break;
                            case '!=': if($value[$fk] != $fvalue) $find_cnt++;break;
                        }
                    }
                }
            }
            if($find_cnt == $find_cnt) $result[] = $value;
        }
    }
    return $result;
}

```

```

        case 'LIKE':
            if(strpos($value[$fk],$fvalue) !==false) $find_cnt++;
            break;
        case 'LIKE-R':
            if (preg_match('/^' . preg_quote($fvalue, '/') . '/', $value[$fk])) { $find_cnt++; }
            break;
        case 'LIKE-L':
            if (preg_match('/^.*' . preg_quote($fvalue, '/') . '$/', $value[$fk])) { $find_cnt++; }
            break;
    }
    }else if($value[$fk] == $fv){
        $find_cnt++;
    }
    }

    if($find_cnt == $find_mcnt){
        $result[] = $value;
    }
}
}else{
    foreach ($params as $fk => $fv) {
        if (isset($value[$fk]) && $value[$fk] == $fv){
            $result[] = $value;
        }
    }
}
}

$this->value = $result;
return $this;
}

```

```

# 멀티 키 => 밸류 값 찾기
public function findWhereIndex (array $params) : int
{
    $result = -1;
    $find_mcnt = count($params);
    foreach ($this->value as $key => $value)
    {
        $find_cnt = 0;
        foreach ($params as $fk => $fv) {
            if (isset($value[$fk]) && $value[$fk] == $fv){
                $find_cnt++;
            }

            if($find_cnt == $find_mcnt){
                $result = $key;
                break;
            }
        }

        if($result>-1){
            break;
        }
    }
    return $result;
}

```

```

# 중복 데이터 제거
public function unique(string $column_name) : ArrayHelper
{
    $result = [];
    $fd_args = array_unique(array_column($this->value, $column_name));
    foreach($fd_args as $idx => $val){
        $result[] = $this->value[$idx];
    }
    $this->value = $result;
    return $this;
}

```

```

# 빈데이터가 있는 배열 찾기
public function isnull(...$params) : ArrayHelper
{
    $result = [];

    # 지정된 카들이 있는지 체크
    if(count($params) > 0){
        foreach($this->value as $idx => $arg){
            foreach ($params as $key) {

```

```

        if (isset($arg[$key]) && ($arg[$key] === "" || $arg[$key] === null)) {
            $result[$idx] = $arg;
            break;
        }
    }
}
}else{
    foreach($this->value as $idx => $arg){
        if (in_array("", $arg, true) || in_array(null, $arg, true)) {
            $result[$idx] = $arg;
        }
    }
}
$this->value = $result;
return $this;
}

# 빈데이터가 있는 배열 제거
public function dropnull(...$params) : ArrayHelper
{
    $result = [];
    # 지정된 카들이 있는지 체크
    if(count($params) > 0){
        foreach($this->value as $idx => $arg){
            foreach ($params as $key) {
                if (isset($arg[$key]) && ($arg[$key] !== "" || $arg[$key] !== null)) {
                    $result[] = $arg;
                    break;
                }
            }
        }
    }
}else{
    foreach($this->value as $idx => $arg){
        if (!in_array("", $arg, true) || !in_array(null, $arg, true)) {
            $result[] = $arg;
        }
    }
}
$this->value = $result;
return $this;
}

# 빈데이터 있는 배열에 데이터 채우기
public function fillnull(mixed $filldata) : ArrayHelper
{
    $is_arr = (is_array($filldata)) ? true : false;
    foreach($this->value as $idx => $arg){
        $cur_keys = array_keys($arg,null);
        foreach($cur_keys as $nkey){
            if($is_arr){
                if(isset($filldata[$nkey])){
                    $this->value[$idx][$nkey] = $filldata[$nkey];
                }
            }else{
                $this->value[$idx][$nkey] = $filldata;
            }
        }
    }
}
return $this;
}

# fill
public function fill(int $start=0, ?int $length=null, mixed $value=null) : ArrayHelper
{
    if ($length === null || $length < $start) {
        $length = $start;
    }

    // 원래의 배열 값을 유지하면서 새로운 범위만 변경
    $args = array_fill($start, $length, $value);
    $this->value = $this->value + $args;
    return $this;
}

# 배열 끝에 추가
public function append(array $args) : ArrayHelper
{
    $this->value[] = $args;
    return $this;
}

```



```

}

# 특정 배열의 int 값 sum 하기
public function sum(string $key = "") : int|float
{
    $result = ($key) ? $this->find_numeric($key) : $this->value;
    $sum = ($key) ? array_sum($result) : count($result);
    return $sum;
}

# 특정 배열의 int 값 min 값
public function min(string $key = "") : int|float
{
    $result = ($key) ? $this->find_numeric($key) : array_keys($this->value);
    $min = 0;
    if(count($result)){
        $min = min($result);
    }
    return $min;
}

# 특정 배열의 int 값 max 값
public function max(string $key = "") : int|float
{
    $result = ($key) ? $this->find_numeric($key) : array_keys($this->value);
    $max = 0;
    if(count($result)){
        $max = max($result);
    }
    return $max;
}

# 특정 배열의 int 값의 평균 값
public function avg(string $key) : int|float
{
    $result = $this->find_numeric($key);
    $avg = 0;
    $cnt = count($result);
    if($cnt>0){
        $avg = array_sum($result) / $cnt;
    }
    return $avg;
}

# union
public function union (array $params) : ArrayHelper
{
    $temp = [];

    # params columns
    $columns = [];
    foreach($params as $uikey => $pvalue){
        $columns[$uikey] = explode(',', $pvalue);
    }

    $sarr = [];
    foreach($this->value as $uikey => $args)
    {
        $index = 0;
        foreach($args as $cidx => $cargs)
        {
            foreach($columns[$uikey] as $column_name){
                if(isset($temp[$column_name][$cidx])){
                    $sarr[$index][$column_name] = $cargs[$column_name];
                }
            }
            $index++;
        }
    }
    $this->value = $sarr;
    return $this;
}

# union All
public function unionAll(...$params) : ArrayHelper
{
    $this->value = call_user_func_array('array_merge', $params);
    return $this;
}

```

```

# index key number
public function findIndex(string $key, mixed $val) : int
{
    $result = -1;
    $index = array_search($val, array_column($this->value, $key));
    if($index !== false){
        $result = $index;
    }
    return $result;
}

# split 배열 여러개씩 잘라서 묶음으로 배열화 하기
public function split(int $length = 2) : ArrayHelper {
    $this->value = array_chunk($this->value, $length );
    return $this;
}

# slice 배열 자르기
public function slice(...$params) : ArrayHelper {
    $result = [];
    if(count($params) > 1){
        $result = array_slice($this->value, $params[0], $params[1]);
    }else {
        $result = array_slice($this->value, $params[0]);
    }

    $this->value = $result;

    return $this;
}

# 모든배열의 키를 새롭게 바꿈
public function changeKeys (...$keys) : ArrayHelper | null
{
    $result = [];

    # key 만 뽑기
    $arrKeys = (isset($keys[0]) && is_array($keys[0])) ? array_values($keys[0]) : array_values($keys);

    # 키배열크기와 값크기가 일치하는지 체크 및 부족한 키 밸류키에서 넣기
    $kCnt = count($arrKeys);
    $vCnt = count($this->value[0]);
    $valueKeys = array_keys($this->value[0]);
    if($kCnt < $vCnt){
        for($i=$kCnt; $i < $vCnt; $i++){
            $arrKeys[] = $valueKeys[$i];
        }
    }else if($kCnt > $vCnt){
        $arrKeys = array_slice($arrKeys, 0, $vCnt);
    }

    # change keys
    foreach($this->value as $index => $args){
        $result[] = array_combine($arrKeys, array_values($args));
    }
    $this->value = $result;

    return $this;
}

public function map(callable $callback): self
{
    $this->value = array_map($callback, $this->value);
    return $this;
}

public function reduce(callable $callback, mixed $initial = null): mixed
{
    return array_reduce($this->value, $callback, $initial);
}

# 원하는 키만 뽑아서 1차원 배열로 출력하기
public function pluck(string $key) : ArrayHelper
{
    $this->value = array_column($this->value, $key);
    return $this;
}

private function find_numeric (string $key) : array

```

```

{
    $result = [];
    foreach($this->value as $a){
        if(is_numeric($a[$key])){
            $result[] = $a[$key];
        }
    }
    return $result;
}

private function asc ($a, $b ): mixed {
    return $a <=> $b;
}

private function desc ($a, $b ): mixed {
    return $a <= $b;
}

public function __get(string $propertyName){
    $result = [];
    if(property_exists($this,$propertyName)){
        $result = $this->{$propertyName};
    }
    return $result;
}

public function __set(string $propertyName, array $args) : void{
    if(property_exists($this,$propertyName)){
        if(is_array($args))
            $this->{$propertyName} = $args;
    }
}
}

```

--- 파일 경로: classes/Auth/Session.php ---

```

<?php
namespace Flex\Banana\Classes\Auth;

final class Session
{
    public const __version = '1.1';

    # 세션 항목
    private $auth_args = [];
    private $authinfo = [];

    # run
    public function __construct(?Array $args){
        if (!is_null($args)){
            if(is_array($args) && count($args)>0){
                $this->auth_args = $args;
            }
        }
    }

    # 세션 키와 값 생성
    public function __set(string $k, mixed $v) : void{
        $this->authinfo[$k] = $v;
        if(!isset($_SESSION[$k]) && $v){
            $_SESSION[$k] = $v;
        }
    }

    # 세션 키값
    public function __get(string $k){
        if(array_key_exists($k, $this->authinfo))
            return $this->authinfo[$k];
    }

    # 세션생성된 전체 값
    public function fetch(): array
    {
        return $this->authinfo;
    }
}

```

```

    }

    # 세션등록
    public function regiAuth(array $data_args) : void
    {
        if(is_array($data_args))
        {
            @session_start();
            foreach($this->auth_args as $k => $session_key){
                if(isset($data_args[$k]) && $data_args[$k]!=""){
                    #Flex\Banana\Log::d($session_key, $data_args[$k]);
                    $_SESSION[$session_key] = $data_args[$k];
                }
            }
        }
    }

    # 세션스타트 및 배열에 담기
    public function sessionStart() : void
    {
        if(is_array($_SESSION)){
            foreach($this->auth_args as $k => $session_key){
                if(isset($_SESSION[$session_key])){
                    $this->authinfo[$session_key] = $_SESSION[$session_key];
                }
            }
        }
    }

    #void
    # 세션비우기
    public function unregiAuth() : void{
        foreach($this->auth_args as $k=>$v)
        {
            $this->authinfo = [];
            if(isset($_SESSION[$v])){
                unset($_SESSION[$v]);
            }
        }
    }
}

```

--- 파일 경로: classes/Cache/CacheMem.php ---

```

<?php
namespace Flex\Banana\Classes\Cache;

use \Memcache;

class CacheMem
{
    public const __version = '0.5.1';
    private string $cache_key;

    private Memcache $memcache;

    public function __construct(
        private string $host='localhost',
        private int $port=11211
    ){
        $this->memcache = new Memcache;
        if(!$this->memcache->connect($this->host, $this->port)){
            throw new \Exception("Memcache connect fail...");
        }
    }

    public function __invoke(string $cache_key): CacheMem
    {
        $this->cache_key = $cache_key;

        return $this;
    }

    public function __call(string $method,array $params) : mixed

```

```

{
    if(method_exists($this->memcache,$method))
    {
        $is_flag = match($method){
            'set','get','delete','flush','close' => false,
            default => true
        };
        return ($is_flag) ?
            call_user_func_array(array($this->memcache,$method),$params) :
            throw new \Exception("Memcache not exists method: {$method}");
    } else throw new \Exception("Memcache not exists method: {$method}");
}

# 서버 상태 체크 : 서버가 실패하면 0, 그렇지 않으면 0이 아님
public function _serverStatus() : mixed
{
    if(empty($this->memcache->getServerStatus($this->host, $this->port)){
        return null;
    }
}
return true;
}

/**
 * 캐시에 키-값 쌍을 설정
 *
 * @param mixed $data 캐싱될 데이터
 * @param int $expiration 캐시 만료 시간 = 0 - 만료되지 않음, 초 단위 >0
 */
public function _set(mixed $data, int $expiration = 0): CacheMem
{
    if (!$this->memcache->set($this->cache_key, $data, 0, $expiration)) {
        throw new \Exception("Memcache set failed for key: {$this->cache_key}");
    }
    return $this;
}

# 캐시에서 값을 가져오기
public function _get(): mixed
{
    $data = $this->memcache->get($this->cache_key);
    if ($data === false) {
        $data = null; // 캐시 값이 없는 경우
    }

    return $data;
}

# 캐시에서 키를 삭제
public function _delete(int $time = 0): void
{
    if(!$this->memcache->delete($this->cache_key, $time)){
        throw new \Exception("Memcache delete failed for key: {$this->cache_key}");
    }
}

# 캐시 비우기
public function _clear(): void
{
    if(!$this->memcache->flush()){
        throw new \Exception("Memcache clear failed");
    }
}

# 캐시 접속 종료
public function _close() : void
{
    if(!$this->memcache->close()){
        throw new \Exception("Memcache close failed");
    }
}

# 자동소멸
public function __destruct(){
    $this->_close();
}
}

```

--- 파일 경로: classes/Cache/CachedMem.php ---

```
<?php
namespace Flex\Banana\Classes\Cache;

use \Memcached;

class CachedMem
{
    public const __version = '0.1.1';
    private string $cache_key;

    private Memcached $memcached;

    public function __construct(
        private string $host = 'localhost',
        private int $port = 11211
    ){
        $this->memcached = new Memcached();
        $this->memcached->addServer($this->host, $this->port);

        if (!$this->_serverStatus()) {
            throw new \Exception("Memcached connect fail...");
        }
    }

    public function __invoke(string $cache_key): CachedMem
    {
        $this->cache_key = $cache_key;

        return $this;
    }

    public function __call(string $method, array $params): mixed
    {
        if (!method_exists($this->memcached, $method)) {
            throw new \Exception("Method '{$method}' does not exist in Memcached.");
        }

        try {
            return call_user_func_array([$this->memcached, $method], $params);
        } catch (\Throwable $e) {
            throw new \Exception("Error calling method '{$method}': " . $e->getMessage(), 0, $e);
        }
    }

    # 서버 상태 체크 : 서버가 실패하면 false 반환, 성공하면 true
    public function _serverStatus(): bool
    {
        $stats = $this->memcached->getStats();
        return isset($stats["{$this->host};{$this->port}"]) && $stats["{$this->host};{$this->port}"]["pid"] > 0;
    }

    /**
     * 캐시에 키-값 쌍을 설정
     *
     * @param mixed $data 캐싱될 데이터
     * @param int $expiration 캐시 만료 시간 = 0 - 만료되지 않음, 초 단위 >0
     */
    public function _set(mixed $data, int $expiration = 0): CachedMem
    {
        if (!$this->memcached->set($this->cache_key, $data, $expiration)) {
            throw new \Exception("Memcached set failed for key: {$this->cache_key}");
        }

        return $this;
    }

    # 캐시에서 값을 가져오기
    public function _get(): mixed
    {
        $data = $this->memcached->get($this->cache_key);
        if ($data === false && $this->memcached->getResultCode() !== Memcached::RES_SUCCESS) {
```

```

        $data = null; // 캐시 값이 없는 경우
    }

    return $data;
}

# 캐시에서 키를 삭제
public function _delete(): void
{
    if (!$this->memcached->delete($this->cache_key)) {
        throw new \Exception("Memcached delete failed for key: {$this->cache_key}");
    }
}

# 캐시 비우기
public function _clear(): void
{
    if (!$this->memcached->flush()) {
        throw new \Exception("Memcached clear failed");
    }
}

# 캐시 접속 종료
public function _close(): void
{
    $this->memcached->quit();
}

# 자동소멸
public function __destruct()
{
    $this->_close();
}
}

```

--- 파일 경로: classes/Calendars/Calendars.php ---

```

<?php
namespace Flex\Banana\Classes\Calendars;

use \DateTime;

# Parent Class : DateTime::;
class Calendars extends DateTime
{
    public const __version = '2.5.1';
    # 년
    private $year = 0;

    # 이달
    # @ monthname : August
    # @ shortmonthname : Aug
    # @ lastdaydow : 그달마지막날이 무슨요일(3) 값
    # @ lastday : 이달의 마지막 날짜
    # @ firstdaydow : 이달의 첫째날이 속한 요일(3)
    private $month = 0;
    private $lastday, $lastdaydow;
    private $firstdaydow;
    private $monthname, $shortmonthname;

    # 오늘(일)
    # @ dow : (일=0,월=1,화=2,수=3,목=4,금=5,토=6)
    # @ dayname : Saturday
    # @ shortdayname : Sat
    # @ cur_week : 오늘 날짜가 속한주(2)째주
    private $day = 0;
    private $daydow, $dayname, $shortdayname;
    private $cur_week;

    # 이전 년월
    private $pre_year;
    private $pre_month;
    private $pre_week;
    # 다음 년월

```

```

private $next_year;
private $next_month;
private $next_week;

# 총 달력 데이터
/**
days_of_month[0] =array(1,2,3,4,5,6,7);
days_of_month[1] =array(8,9,10,11,12,13,14);
*/
private $days_of_month = [];

# 기념일 및 휴일설정
private $memorial_arg = [];

# 음력 관련
private $sunargs= array(20000101,20000107,20000201,20000205,20000301,20000306,20000401,20000405,
20000501,20000504,20000601,20000602,20000701,20000702,20000731,20000801,20000829,20000901,
20000928,20001001,20001027,20001101,20001126,20001201,20001226,20010101,20010124,20010201,
20010223,20010301,20010325,20010401,20010424,20010501,20010523,20010601,20010621,20010701,
20010721,20010801,20010819,20010901,20010917,20011001,20011017,20011101,20011115,20011201,
20011215,20020101,20020113,20020201,20020212,20020301,20020314,20020401,20020413,20020501,
20020512,20020601,20020611,20020701,20020710,20020801,20020809,20020901,20020907,20021001,
20021006,20021101,20021105,20021201,20021204,20030101,20030103,20030201,20030301,20030303,
20030401,20030402,20030501,20030531,20030601,20030630,20030701,20030729,20030801,20030828,
20030901,20030926,20031001,20031025,20031101,20031124,20031201,20031223,20040101,20040122,
20040201,20040220,20040301,20040321,20040401,20040419,20040501,20040519,20040601,20040618,
20040701,20040717,20040801,20040816,20040901,20040914,20041001,20041014,20041101,20041112,
20041201,20041212,20050101,20050110,20050201,20050209,20050301,20050310,20050401,20050409,
20050501,20050508,20050601,20050607,20050701,20050706,20050801,20050805,20050901,20050904,
20051001,20051003,20051101,20051102,20051201,20051202,20051231,20060101,20060129,20060201,
20060228,20060301,20060329,20060401,20060428,20060501,20060527,20060601,20060626,20060701,
20060725,20060801,20060824,20060901,20060922,20061001,20061022,20061101,20061121,20061201,
20061220,20070101,20070119,20070201,20070218,20070301,20070319,20070401,20070417,20070501,
20070517,20070601,20070615,20070701,20070714,20070801,20070813,20070901,20070911,20071001,
20071011,20071101,20071110,20071201,20071210,20080101,20080108,20080201,20080207,20080301,
20080308,20080401,20080406,20080501,20080505,20080601,20080604,20080701,20080703,20080801,
20080831,20080901,20080929,20081001,20081029,20081101,20081128,20081201,20081227,20090101,
20090126,20090201,20090225,20090301,20090327,20090401,20090425,20090501,20090524,20090601,
20090623,20090701,20090722,20090801,20090820,20090901,20090919,20091001,20091018,20091101,
20091117,20091201,20091216,20100101,20100115,20100201,20100214,20100301,20100316,20100401,
20100414,20100501,20100514,20100601,20100612,20100701,20100712,20100801,20100810,20100901,
20100908,20101001,20101008,20101101,20101106,20101201,20101206,20110101,20110104,20110201,
20110203,20110301,20110305,20110401,20110403,20110501,20110503,20110601,20110602,20110701,
20110731,20110801,20110829,20110901,20110927,20111001,20111027,20111101,20111125,20111201,
20111225,20120101,20120123,20120201,20120222,20120301,20120322,20120401,20120421,20120501,
20120521,20120601,20120620,20120701,20120719,20120801,20120818,20120901,20120916,20121001,
20121015,20121101,20121114,20121201,20121213,20130101,20130112,20130201,20130210,20130301,
20130312,20130401,20130410,20130501,20130510,20130601,20130609,20130701,20130708,20130801,
20130807,20130901,20130905,20131001,20131005,20131101,20131103,20131201,20131203,20140101,
20140131,20140201,20140301,20140331,20140401,20140429,20140501,20140529,20140601,20140627,
20140701,20140727,20140801,20140825,20140901,20140924,20141001,20141024,20141101,20141122,
20141201,20141222,20150101,20150120,20150201,20150219,20150301,20150320,20150401,20150419,
20150501,20150518,20150601,20150616,20150701,20150716,20150801,20150814,20150901,20150913,
20151001,20151013,20151101,20151112,20151201,20151211,20160101,20160110,20160201,20160209,
20160301,20160309,20160401,20160407,20160501,20160507,20160601,20160605,20160701,20160704,
20160801,20160803,20160901,20161001,20161031,20161101,20161129,20161201,20161229,20170101,
20170128,20170201,20170227,20170301,20170328,20170401,20170426,20170501,20170526,20170601,
20170624,20170701,20170723,20170801,20170822,20170901,20170920,20171001,20171020,20171101,
20171118,20171201,20171218,20180101,20180117,20180201,20180216,20180301,20180317,20180401,
20180416,20180501,20180515,20180601,20180614,20180701,20180713,20180801,20180811,20180901,
20180910,20181001,20181009,20181101,20181108,20181201,20181207,20190101,20190106,20190201,
20190205,20190301,20190307,20190401,20190405,20190501,20190505,20190601,20190603,20190701,
20190703,20190801,20190830,20190901,20190929,20191001,20191028,20191101,20191127,20191201,
20191226,20200101,20200125,20200201,20200224,20200301,20200324,20200401,20200423,20200501,
20200523,20200601,20200621,20200701,20200721,20200801,20200819,20200901,20200917,20201001,
20201017,20201101,20201115,20201201,20201215,20210101,20210113,20210201,20210212,20210301,
20210313,20210401,20210412,20210501,20210512,20210601,20210610,20210701,20210710,20210801,
20210808,20210901,20210907,20211001,20211006,20211101,20211105,20211201,20211204,20220101,
20220103,20220201,20220301,20220303,20220401,20220501,20220530,20220601,20220629,20220701,
20220729,20220801,20220827,20220901,20220926,20221001,20221025,20221101,20221124,20221201,
20221223,20230101,20230122,20230201,20230220,20230301,20230322,20230401,20230420,20230501,
20230520,20230601,20230618,20230701,20230718,20230801,20230816,20230901,20230915,20231001,
20231015,20231101,20231113,20231201,20231213,20240101,20240111,20240201,20240210,20240301,
20240310,20240401,20240409,20240501,20240508,20240601,20240606,20240701,20240706,20240801,
20240804,20240901,20240903,20241001,20241003,20241101,20241201,20241231,20250101,20250129,
20250201,20250228,20250301,20250329,20250401,20250428,20250501,20250527,20250601,20250625,
20250701,20250725,20250801,20250823,20250901,20250922,20251001,20251021,20251101,20251120,
20251201,20251220,20260101,20260119,20260201,20260217,20260301,20260319,20260401,20260417,
20260501,20260517,20260601,20260615,20260701,20260714,20260801,20260813,20260901,20260911,

```



20261001,20261011,20261101,20261109,20261201,20261209,20270101,20270108,20270201,20270207,  
20270301,20270308,20270401,20270407,20270501,20270506,20270601,20270605,20270701,20270704,  
20270801,20270802,20270901,20270930,20271001,20271029,20271101,20271128,20271201,20271228,  
20280101,20280127,20280201,20280225,20280301,20280326,20280401,20280425,20280501,20280524,  
20280601,20280623,20280701,20280722,20280801,20280820,20280901,20280919,20281001,20281018,  
20281101,20281116,20281201,20281216,20290101,20290115,20290201,20290213,20290301,20290315,  
20290401,20290414,20290501,20290513,20290601,20290612,20290701,20290712,20290801,20290810,  
20290901,20290908,20291001,20291008,20291101,20291106,20291201,20291205,20300101,20300104,  
20300201,20300203,20300301,20300304,20300401,20300403,20300501,20300502,20300601,20300701,  
20300730,20300801,20300829,20300901,20300927,20301001,20301027,20301101,20301125,20301201,  
20301225,20310101,20310123,20310201,20310222,20310301,20310323,20310401,20310422,20310501,  
20310521,20310601,20310620,20310701,20310719,20310801,20310818,20310901,20310917,20311001,  
20311016,20311101,20311115,20311201,20311214,20320101,20320113,20320201,20320211,20320301,  
20320312,20320401,20320410,20320501,20320509,20320601,20320608,20320701,20320707,20320801,  
20320806,20320901,20320905,20321001,20321004,20321101,20321103,20321201,20321203,20330101,  
20330131,20330201,20330301,20330331,20330401,20330429,20330501,20330528,20330601,20330627,  
20330701,20330726,20330801,20330825,20330901,20330923,20331001,20331023,20331101,20331122,  
20331201,20331222,20340101,20340120,20340201,20340219,20340301,20340320,20340401,20340419,  
20340501,20340518,20340601,20340616,20340701,20340716,20340801,20340814,20340901,20340913,  
20341001,20341012,20341101,20341111,20341201,20341211,20350101,20350110,20350201,20350208,  
20350301,20350310,20350401,20350408,20350501,20350508,20350601,20350606,20350701,20350705,  
20350801,20350804,20350901,20350902,20351001,20351031,20351101,20351130,20351201,20351229,  
20360101,20360128,20360201,20360227,20360301,20360328,20360401,20360426,20360501,20360526,  
20360601,20360624,20360701,20360723,20360801,20360822,20360901,20360920,20361001,20361019,  
20361101,20361118,20361201,20361217,20370101,20370116,20370201,20370215,20370301,20370317,  
20370401,20370416,20370501,20370515,20370601,20370614,20370701,20370713,20370801,20370811,  
20370901,20370910,20371001,20371009,20371101,20371107,20371201,20371207,20380101,20380105,  
20380201,20380204,20380301,20380306,20380401,20380405,20380501,20380504,20380601,20380603,  
20380701,20380702,20380801,20380830,20380901,20380929,20381001,20381028,20381101,20381126,  
20381201,20381226,20390101,20390124,20390201,20390223,20390301,20390325,20390401,20390423,  
20390501,20390523,20390601,20390622,20390701,20390721,20390801,20390820,20390901,20390918,  
20391001,20391018,20391101,20391116,20391201,20391216,20400101,20400114,20400201,20400212,  
20400301,20400313,20400401,20400411,20400501,20400511,20400601,20400610,20400701,20400709,  
20400801,20400808,20400901,20400907,20401001,20401006,20401101,20401105,20401201,20401204,  
20410101,20410103,20410201,20410301,20410303,20410401,20410430,20410501,20410530,20410601,  
20410628,20410701,20410728,20410801,20410827,20410901,20410925,20411001,20411025,20411101,  
20411124,20411201,20411223,20420101,20420122,20420201,20420220,20420301,20420322,20420401,  
20420420,20420501,20420519,20420601,20420618,20420701,20420717,20420801,20420816,20420901,  
20420914,20421001,20421014,20421101,20421113,20421201,20421213,20430101,20430111,20430201,  
20430210,20430301,20430311,20430401,20430410,20430501,20430509,20430601,20430607,20430701,  
20430707,20430801,20430805,20430901,20430903,20431001,20431003,20431101,20431102,20431201,  
20431231,20440101  
);

```
private $moonargs = array (  
19991125,19991201,19991226,20000101,20000126,20000201,20000227,20000301,20000327,20000401,20000429,  
20000501,20000530,20000601,20000701,20000702,20000801,20000804,20000901,20000904,20001001,20001006,  
20001101,20001106,20001201,20001207,20010101,20010109,20010201,20010207,20010301,20010308,20010401,  
20010408,20010401,20010410,20010501,20010511,20010601,20010612,20010701,20010714,20010801,20010815,  
20010901,20010916,20011001,20011017,20011101,20011118,20011201,20011220,20020101,20020118,20020201,  
20020219,20020301,20020319,20020401,20020421,20020501,20020521,20020601,20020623,20020701,20020724,  
20020801,20020825,20020901,20020927,20021001,20021027,20021101,20021129,20021201,20030101,20030129,  
20030201,20030230,20030301,20030401,20030501,20030502,20030601,20030602,20030701,20030704,20030801,  
20030805,20030901,20030906,20031001,20031008,20031101,20031108,20031201,20031210,20040101,20040111,  
20040201,20040211,20040201,20040212,20040301,20040313,20040401,20040414,20040501,20040514,20040601,  
20040616,20040701,20040717,20040801,20040818,20040901,20040919,20041001,20041020,20041101,20041121,  
20041201,20041223,20050101,20050121,20050201,20050223,20050301,20050323,20050401,20050425,20050501,  
20050525,20050601,20050627,20050701,20050728,20050801,20050828,20050901,20050930,20051001,20051030,  
20051101,20051201,20051202,20060101,20060104,20060201,20060202,20060301,20060304,20060401,20060404,  
20060501,20060506,20060601,20060606,20060701,20060708,20060701,20060709,20060801,20060810,20060901,  
20060911,20061001,20061011,20061101,20061113,20061201,20061214,20070101,20070112,20070201,20070214,  
20070301,20070315,20070401,20070416,20070501,20070517,20070601,20070619,20070701,20070720,20070801,  
20070821,20070901,20070922,20071001,20071022,20071101,20071123,20071201,20071225,20080101,20080124,  
20080201,20080225,20080301,20080326,20080401,20080428,20080501,20080528,20080601,20080701,20080801,  
20080802,20080901,20080903,20081001,20081004,20081101,20081104,20081201,20081206,20090101,20090107,  
20090201,20090205,20090301,20090306,20090401,20090407,20090501,20090509,20090501,20090509,20090601,  
20090611,20090701,20090713,20090801,20090813,20090901,20090915,20091001,20091015,20091101,20091117,  
20091201,20091218,20100101,20100116,20100201,20100217,20100301,20100318,20100401,20100419,20100501,  
20100520,20100601,20100621,20100701,20100723,20100801,20100824,20100901,20100925,20101001,20101026,  
20101101,20101127,20101201,20101229,20110101,20110127,20110201,20110228,20110301,20110329,20110401,  
20110430,20110501,20110601,20110701,20110702,20110801,20110804,20110901,20110905,20111001,20111006,  
20111101,20111107,20111201,20111208,20120101,20120110,20120201,20120209,20120301,20120311,20120301,  
20120311,20120412,20120501,20120512,20120601,20120614,20120701,20120715,20120801,20120816,  
20120901,20120918,20121001,20121018,20121101,20121120,20121201,20121221,20130101,20130120,20130201,  
20130221,20130301,20130322,20130401,20130423,20130501,20130523,20130601,20130625,20130701,20130726,  
20130801,20130827,20130901,20130928,20131001,20131029,20131101,20131201,20140101,20140102,20140201,  
20140301,20140302,20140401,20140403,20140501,20140504,20140601,20140605,20140701,20140706,20140801,  
20140808,20140901,20140908,20140901,20140909,20141001,20141010,20141101,20141111,20141201,20141213,  
20150101,20150111,20150201,20150213,20150301,20150313,20150401,20150415,20150501,20150516,20150601,
```

```
#@ void
# Y-m-d H:i:s
public function __construct($times){
    parent::__construct($times);
    self::resetTodayDate();
}

#@ void
#날짜 리셋
public function resetTodayDate() : void{
    $ymd_args = explode("-", $this->format("Y-m-d"));
    $this->year = $ymd_args[0];
    $this->month = $ymd_args[1];
    $this->day = $ymd_args[2];

    self::fromJd();
}
```

```

self::set_pre_next_date();
}

#@ void
# 오늘날짜에 속한 정보들을 얻는다
public function fromJd() :void{
    if(function_exists('unixtojd'))
    {
        $today_mktime =unixtojd(mktime(0,0,0,$this->month,$this->day, $this->year));
        $today_args =cal_from_jd($today_mktime, CAL_GREGORIAN);
        if(is_array($today_args))
        {
            #month
            $this->monthname    = $today_args['monthname'];
            $this->shortmonthname = (isset($today_args['abbrevmonthname'])) ? $today_args['abbrevmonthname'] : substr($today_args['monthname'],0,3);

            #day
            $this->daydow      = $today_args['dow'];
            $this->dayname     = $today_args['dayname'];
            $this->shortdayname = $today_args['abbrevdayname'];
            $this->firstdaydow = date('w',mktime(0,0,0,$this->month,1,$this->year));
        }
        $this->lastday  = cal_days_in_month(CAL_GREGORIAN, $this->month, $this->year);
        $this->lastdaydow = date('w',mktime(0,0,0,$this->month,$this->lastday,$this->year));
    }else{
        $this->daydow      = date('w',mktime(0,0,0,$this->month,$this->day,$this->year));
        $this->firstdaydow = date('w',mktime(0,0,0,$this->month,1,$this->year));
        $this->lastday     = date("t",mktime(0,0,1,$this->month,1,$this->year));
        $this->lastdaydow  = date('w',mktime(0,0,0,$this->month,$this->lastday,$this->year));
    }
}

#@ void
# 음력, 양력 기념일 데이터를 양력으로 통일 시킴
/**
= 입력 데이터 형식
[0]=array(
    'date'      =>[0000-01-01|2012-01-01],  #날짜
    'smtype'    =>[s|m],                    #양력|음력
    'repeat'     =>[y|m|n],                  #반복(년,월,없음)
    'holiday'    =>[0|1],                    #휴일여부
    'holiday_plus' =>[0|1],                  #설날,추석같이 특이한 휴일설정일경우(추가휴일여부)
    'title'      =>신청                      #제목
)
*/
public function set_memorials(Array $memorials=array()) : void
{
    $count=count($memorials);
    for($i=0; $i<$count; $i++)
    {
        if(isset($memorials[$i]))
        {
            $m      =&$memorials[$i];
            $date_args = explode('-', $m['date']);
            $this_int_date = $this->year.$date_args[1].$date_args[2];
            $int_date      = $date_args[0].$date_args[1].$date_args[2];
            $holiday_plus  = $m['holiday_plus'];

            # 반복아닌기념일
            # y : 년, m : 월, n : 반복없음
            if($m['smtype']=='m'){ # 음력
                switch($m['repeat']){
                    case 'y' :
                        self::set_memorials_holiday(self::get_moon2sun($this_int_date), $m['holiday'], $m['title'], $holiday_plus);
                        break;
                    case 'm':
                        for($ssi=1; $ssi<13; $ssi++)
                            self::set_memorials_holiday(self::get_moon2sun($date_args[0].sprintf("%02d",$ssi).$date_args[2]), $m['holiday'], $m['title'],$holiday_plus);
                        break;
                    default :
                        self::set_memorials_holiday(self::get_moon2sun($int_date), $m['holiday'], $m['title'],$holiday_plus);
                }
            }
        }
    }
    switch($m['repeat']){
        case 'y' :
            self::set_memorials_holiday($this_int_date, $m['holiday'], $m['title'], $holiday_plus);
            break;
        case 'm':
            for($ssi=1; $ssi<13; $ssi++)

```

```

        self::set_memorials_holiday($this->year.sprintf("%02d",$si).$date_args[2], $m['holiday'], $m['title'],$holiday_plus);
        break;
        default:
            self::set_memorials_holiday($this_int_date, $m['holiday'], $m['title'],$holiday_plus);
    }
}
}
}
}

#@ void
# 기념일 데이터를 입력
private function set_memorials_holiday($int_date, $holiday, $holiday_title,$holiday_plus) : void{
    $this->memorial_arg[$int_date] = ['holiday'=>$holiday,'title'=>$holiday_title];
    if($holiday_plus==1){
        $this->memorial_arg[$int_date-1] = ['holiday'=>$holiday,'title'=>"];
        $this->memorial_arg[$int_date+1] = ['holiday'=>$holiday,'title'=>"];
    }
}

#현재달력 구하기
public function set_days_of_month() : void
{
    $x=0;

    # 이전달
    if(function_exists('cal_days_in_month')){
        $pre_lastday = cal_days_in_month(CAL_GREGORIAN, $this->pre_month, $this->pre_year);
    }else{
        $pre_lastday = date("t",mktime(0,0,1,$this->pre_month,1,$this->pre_year));
    }

    if($this->firstdaydow==0) $s_pre_day=$pre_lastday-6;
    else $s_pre_day=$pre_lastday-($this->firstdaydow-1);

    for($i=$s_pre_day; $i<=$pre_lastday; $i++)
    {
        $tmp_date = sprintf("%04d-%02d-%02d", $this->pre_year,$this->pre_month,$i);
        $int_date = intval(str_replace('-',",$tmp_date));
        $this->days_of_month[$x][] =array(
            'date'      => $tmp_date,
            'day'       => $i,
            'moon'      => "",
            'holiday'   => "",
            'event_title' => "",
            'this_month' => ""
        );
        $num=date('w',mktime(0,0,0,$this->pre_month,$i,$this->pre_year));
        if($num== 6) $x++;
    }

    # 현재달
    for($j=1; $j<=$this->lastday; $j++)
    {
        $tmp_date = sprintf("%04d-%02d-%02d", $this->year,$this->month,$j);
        $int_date = intval(str_replace('-',",$tmp_date));
        $int_day = intval($this->day);

        #10일에 한번씩 [음력날짜] 계산 및 표기
        $moon_date="";
        if($j%10==0){
            $moon_date = self::get_sun2moon($int_date);
            $moon_date = substr($moon_date,4,2).'.'.substr($moon_date,-2);
        }

        # 기념일 및 휴일
        $holiday="";
        $event_title="";
        if(isset($this->memorial_arg[$int_date])){
            if(isset($this->memorial_arg[$int_date]['holiday']) && $this->memorial_arg[$int_date]['holiday']==1) $holiday = 1;
            $event_title = (isset($this->memorial_arg[$int_date]['title'])) ? $this->memorial_arg[$int_date]['title'] : "";
        }

        # 달력
        $this->days_of_month[$x][] =[
            'date'      => $tmp_date,
            'day'       => $j,
            'moon'      => $moon_date,
            'holiday'   => $holiday,

```

```

        'event_title' => $event_title,
        'this_month' => 1
    ];
    $num=date('w',mktime(0,0,0,$this->month,$j,$this->year));
    if($j==$int_day){
        $this->pre_week = $x-1;
        $this->cur_week = $x;
        $this->next_week = $x+1;
    }
    if($num== 6) $x++;
}

# 다음달
$nk=1;
$snxt=($this->lastdaydow==6) ? 0 : $this->lastdaydow+1;
for($k=$snxt;$k<7; $k++)
{
    $tmp_date=sprintf("%04d-%02d-%02d", $this->next_year,$this->next_month,$nk);
    $int_date=intval(str_replace('-',",$tmp_date));
    $this->days_of_month[$x][] = [
        'date' => $tmp_date,
        'day' => $nk,
        'moon' => ",holiday=>",
        'event_title' => "",
        'this_month' => ""
    ];
    $num=date('w',mktime(0,0,0,$this->next_month,$nk,$this->next_year));
    $nk++;
    if($num== 6) $x++;
}
}

# 이전 년월, 다음 년월 구하기
public function set_pre_next_date() : void
{
    $prev_year = $this->year-1;
    $next_year = $this->year+1;
    $month = intval($this->month);
    if($month==1){
        $this->pre_year = $prev_year;
        $this->next_year = $this->year;
        $this->pre_month = 12;
        $this->next_month = sprintf("%02d",$month+1);
    }
    else if($month==12){
        $this->pre_year = $this->year;
        $this->next_year = $next_year;
        $this->pre_month = sprintf("%02d",$month-1);
        $this->next_month = 1;
    }
    else if($month !=1 && $month !=12)
    {
        $this->pre_year = $this->year;
        $this->next_year = $this->year;
        $this->pre_month = sprintf("%02d",$month-1);
        $this->next_month = sprintf("%02d",$month+1);
    }
}

# 날짜수정 DAY
public function modifyDay(int|string $day) : void{
    $this->modify($day." day");
    self::resetTodayDate();
}

# 날짜수정 WEEK
public function modifyWeek(int|string $week) : void{
    $this->modify($week." week");
    self::resetTodayDate();
}

# 날짜수정 MONTH
public function modifyMonth(int|string $month) : void{
    $this->modify($month." month");
    self::resetTodayDate();
}

#이전주에 해당하는 마지막일을 가지고 온다
public function get_pre_week_last_date() : date{

```

```

    $args = [];
    $pre_date = "";
    if(isset($this->days_of_month[$this->cur_week])){
        $args = $this->days_of_month[$this->cur_week];
    }

    if(isset($args[0]) && isset($args[0]['date'])){
        parent::__construct($args[0]['date']);
        $this->modify('-1 day');
        $pre_date = $this->format("Y-m-d");
    }
    return $pre_date;
}

#다음주에 해당하는 첫일을 가지고 온다
public function get_next_week_first_date() : date{
    $args = [];
    $nxt_date = "";
    if(isset($this->days_of_month[$this->cur_week])){
        $args = $this->days_of_month[$this->cur_week];
    }

    if(isset($args[6]) && isset($args[6]['date'])){
        parent::__construct($args[6]['date']);
        $this->modify('+1 day');
        $nxt_date = $this->format("Y-m-d");
    }
    return $nxt_date;
}

# 해당해의 띠
public function get_zodiac_sign() : string{
    $zodiac_sign_args = ['원숭이', '닭', '개', '돼지', '쥐', '소', '범', '토끼', '용', '뱀', '말', '양'];
    $ddikey = intval($this->year % 12);
    return $zodiac_sign_args[$ddikey];
}

# 육십갑자
public function get_sexagenary_cycle() : string{
    $stengan = ['경', '신', '임', '계', '갑', '을', '병', '정', '무', '기'];
    $stenji = ['신', '유', '술', '해', '자', '축', '인', '묘', '진', '사', '오', '미'];

    $n1 = substr($this->year, -1);
    $n2 = intval($this->year % 12);

    return $stengan[$n1].$stenji[$n2];
}

#@ return
# 양력->음력
# intdate : 20101020
public function get_sun2moon($intdate) : int{
    return self::date_binary_search($this->sunargs, $this->moonargs, $intdate);
}

# 음력->양력
public function get_moon2sun($intdate) : int{
    return self::date_binary_search($this->moonargs, $this->sunargs, $intdate);
}

# 음<->양 계산메소드
public function date_binary_search(&$haystack, &$haystack2, &$needle) : mixed
{
    $high = count($haystack);
    $low = 0;

    if ($needle < $haystack[$low] || $needle > $haystack[$high - 1]) {
        return false;
    }

    while ($high - $low > 1) {
        $mid = (int)((($high + $low) / 2)); // Ensure division is part of the integer cast
        if ($haystack[$mid] < $needle) {
            $low = $mid;
        } else {
            $high = $mid;
        }
    }
    if ($high == count($haystack) || $haystack[$high] != $needle) {

```

```

        return $haystack2[$low] + ($needle - $haystack[$low]);
    } else {
        return $haystack2[$high];
    }
}

#@ return
# 프라퍼티 값 가져오기
public function __get($propertyname) : mixed{
    return $this->{$propertyname};
}
}

```

--- 파일 경로: classes/Cipher/AES256Hash.php ---

```

<?php
namespace Flex\Banana\Classes\Cipher;

use Exception;

class AES256Hash
{
    public const __version = '1.0.2';

    private string $encrypt_method = 'AES-256-CBC';

    /**
     * AES-256 암호화를 수행
     *
     * @param string $plaintext 암호화할 평문
     * @param string $secret_key 비밀 키 random_bytes(32) | hex2bin($hex_string)
     * @param string $secret_iv 초기화 벡터 (IV) random_bytes(16) | bin2hex
     * @return string 암호화된 문자열 (base64 인코딩됨)
     * @throws Exception 암호화 실패 시
     */
    public function encrypt(string $plaintext, string $secret_key, string $secret_iv): string
    {
        $key = $this->prepareKey($secret_key);
        $iv = $this->prepareIV($secret_iv);

        $encrypted = openssl_encrypt($plaintext, $this->encrypt_method, $key, 0, $iv);

        if ($encrypted === false) {
            return "";
        }

        // RAW 데이터 출력 + URL-safe Base64
        return strtr(trim(base64_encode($encrypted)), '=', '+', '-_');
    }

    /**
     * AES-256 복호화를 수행
     *
     * @param string $ciphertext 복호화할 암호문 (base64 인코딩된 상태)
     * @param string $secret_key 비밀 키
     * @param string $secret_iv 초기화 벡터 (IV)
     * @return string 복호화된 평문
     * @throws Exception 복호화 실패 시
     */
    public function decrypt(string $ciphertext, string $secret_key, string $secret_iv): string
    {
        $key = $this->prepareKey($secret_key);
        $iv = $this->prepareIV($secret_iv);

        // URL-safe -> 표준 Base64 변환
        $ciphertext = base64_decode($ciphertext);
        $ciphertext = strtr($ciphertext, '-_', '+/').str_repeat('=', strlen($ciphertext) % 4);

        $decrypted = openssl_decrypt($ciphertext, $this->encrypt_method, $key, 0, $iv);

        if ($decrypted === false) {
            return "";
        }
        return $decrypted;
    }
}

```

```

}

/**
 * 비밀 키를 준비 (32바이트로 조정)
 *
 * @param string $secret_key 원본 비밀 키
 * @return string 32바이트로 조정된 키
 */
private function prepareKey(string $secret_key): string
{
    if (strlen($secret_key) === 32) {
        return $secret_key;
    }
    if (strlen($secret_key) > 32) {
        return substr($secret_key, 0, 32);
    }
    return str_pad($secret_key, 32, "\0", STR_PAD_RIGHT);
}

/**
 * 초기화 벡터(IV)를 준비 (16바이트로 조정)
 *
 * @param string $secret_iv 원본 IV
 * @return string 16바이트로 조정된 IV
 */
private function prepareIV(string $secret_iv): string
{
    if (strlen($secret_iv) === 16) {
        return $secret_iv;
    }
    if (strlen($secret_iv) > 16) {
        return substr($secret_iv, 0, 16);
    }
    return str_pad($secret_iv, 16, "\0", STR_PAD_RIGHT);
}
}

```

--- 파일 경로: classes/Cipher/Base64UrlEncoder.php ---

```

<?php
namespace Flex\Banana\Classes\Cipher;

use Exception;

class Base64UrlEncoder
{
    public const __version = '1.0';

    protected string $data;

    public function __construct(string $data = "")
    {
        $this->data = $data;
    }

    /**
     * 데이터를 Base64Url로 인코딩
     *
     * @param string|null $data 인코딩할 데이터 (null이면 내부 데이터 사용)
     * @return string Base64Url 인코딩된 문자열
     * @throws Exception 인코딩 실패 시
     */
    public function encode(?string $data = null): string
    {
        $input = $data ?? $this->data;
        $base64 = base64_encode($input);
        if ($base64 === false) {
            throw new Exception("Base64 encoding failed", __LINE__);
        }
        return $this->urlEncode($base64);
    }

    /**
     * Base64Url 인코딩된 문자열을 디코딩

```



```

*
* @param string|null $data 디코딩할 Base64Url 문자열 (null이면 내부 데이터 사용)
* @return string 디코딩된 원본 데이터
* @throws Exception 디코딩 실패 시
*/
public function decode(?string $data = null): string
{
    $input = $data ?? $this->data;
    $base64 = $this->urlDecode($input);
    $decoded = base64_decode($base64, true);
    if ($decoded === false) {
        throw new Exception("Base64 decoding failed", __LINE__);
    }
    return $decoded;
}

/**
 * Base64 문자열을 URL 안전 형식으로 변환
 *
 * @param string $input Base64 문자열
 * @return string URL 안전 Base64 문자열
 */
protected function urlEncode(string $input): string
{
    return rtrim(strtr($input, '+/', '_-'), '=');
}

/**
 * URL 안전 Base64 문자열을 표준 Base64 형식으로 변환
 *
 * @param string $input URL 안전 Base64 문자열
 * @return string 표준 Base64 문자열
 */
protected function urlDecode(string $input): string
{
    $remainder = strlen($input) % 4;
    if ($remainder) {
        $input .= str_repeat('=', 4 - $remainder);
    }
    return strtr($input, '_-', '+/');
}

/**
 * 내부 데이터를 설정
 *
 * @param string $data 설정할 데이터
 */
public function setData(string $data): void
{
    $this->data = $data;
}

/**
 * 내부 데이터를 반환
 *
 * @return string
 */
public function getData(): string
{
    return $this->data;
}
}

```

--- 파일 경로: classes/Cipher/CipherGeneric.php ---

```

<?php
namespace Flex\Banana\Classes\Cipher;

use \ReflectionClass;
use \Exception;

class CipherGeneric
{
    public const __version = '1.0';
}

```

```

private $processor;
private static $allowedProcessors = [
    AES256Hash::class,
    HashEncoder::class,
    PasswordHash::class,
    Base64UrlEncoder::class,
    ROT13Encoder::class
];

public function __construct($processor)
{
    $this->setProcessor($processor);
}

private function setProcessor($processor): void
{
    $reflection = new ReflectionClass($processor);
    if (!in_array($reflection->getName(), self::$allowedProcessors)) {
        throw new Exception("Unsupported processor type: " . $reflection->getName());
    }

    $this->processor = $processor;
}

public function __call($name, $arguments)
{
    $reflection = new ReflectionClass($this->processor);
    if (!$reflection->hasMethod($name)) {
        throw new Exception("Method $name does not exist in " . get_class($this->processor));
    }

    $method = $reflection->getMethod($name);
    if (!$method->isPublic()) {
        throw new Exception("Method $name is not public in " . get_class($this->processor));
    }

    return $method->invokeArgs($this->processor, $arguments);
}

public static function addProcessor(string $processorClass): void
{
    if (!class_exists($processorClass)) {
        throw new Exception("Class $processorClass does not exist");
    }

    if (!in_array($processorClass, self::$allowedProcessors)) {
        self::$allowedProcessors[] = $processorClass;
    }
}

public static function getAllowProcessors(): array
{
    return self::$allowedProcessors;
}
}

```

--- 파일 경로: classes/Cipher/HashEncoder.php ---

```

<?php
namespace Flex\Banana\Classes\Cipher;

use Exception;

class HashEncoder
{
    public const __version = '1.0';
    private string $encrypt_str;

    public function __construct(string $encrypt_str)
    {
        $this->encrypt_str = $encrypt_str;
    }

    public function hash(string $algorithm = 'sha256'): string
    {
        return hash($algorithm, $this->encrypt_str);
    }
}

```

```

{
    if (!in_array($algorithm, ['sha256', 'sha512'])) {
        throw new Exception("Unsupported hash algorithm", __LINE__);
    }
    $result = hash($algorithm, $this->encrypt_str);
    if ($result === false) {
        throw new Exception("Hash generation failed", __LINE__);
    }
    return $result;
}
}

```

--- 파일 경로: classes/Cipher/PasswordHash.php ---

```

<?php
namespace Flex\Banana\Classes\Cipher;

class PasswordHash
{
    public const __version = '1.0';
    private array $options;

    /**
     * PasswordHasher 생성자
     *
     * @param int $memory_cost 메모리 사용량 (KiB, 기본값: 65536)
     * @param int $time_cost 반복 횟수 (기본값: 4)
     * @param int $threads 사용할 스레드 수 (기본값: 1)
     */
    public function __construct(int $memory_cost = 65536, int $time_cost = 4, int $threads = 1)
    {
        $this->options = [
            'memory_cost' => $memory_cost,
            'time_cost' => $time_cost,
            'threads' => $threads,
        ];
    }

    /**
     * 비밀번호를 해시.
     *
     * @param string $password 해시할 비밀번호
     * @return string 해시된 비밀번호
     */
    public function hash(string $password): string
    {
        if (defined('PASSWORD_ARGON2ID')) {
            return password_hash($password, PASSWORD_ARGON2ID, $this->options);
        } else {
            // Argon2id를 사용할 수 없는 경우 bcrypt로
            return password_hash($password, PASSWORD_DEFAULT);
        }
    }

    /**
     * 일치여부 비교
     *
     * @param string $password 확인할 비밀번호
     * @param string $hash 비교할 해시
     * @return bool 비밀번호가 일치하면 true, 그렇지 않으면 false
     */
    public function verify(string $password, string $hash): bool
    {
        return password_verify($password, $hash);
    }

    /**
     * 해시가 재해싱이 필요한지 확인
     *
     * @param string $hash 확인할 해시
     * @return bool 재해싱이 필요하면 true, 그렇지 않으면 false
     */
    public function needsRehash(string $hash): bool
    {

```

```

{
    if (defined('PASSWORD_ARGON2ID')) {
        return password_needs_rehash($hash, PASSWORD_ARGON2ID, $this->options);
    } else {
        return password_needs_rehash($hash, PASSWORD_DEFAULT);
    }
}

/**
 * 해시를 업그레이드
 *
 * @param string $password 원본 비밀번호
 * @param string $hash 현재 해시
 * @return string|null 새로운 해시 또는 업그레이드가 필요 없으면 null
 */
public function upgradeHash(string $password, string $hash): ?string
{
    if ($this->needsRehash($hash)) {
        return $this->hash($password);
    }
    return null;
}
}

```

--- 파일 경로: classes/Cipher/ROT13Encoder.php ---

```

<?php
namespace Flex\Banana\Classes\Cipher;

/**
 * 알파벳을 13글자씩 밀어서 치환
 * 대소문자를 구분
 * ex) A->N, B->O, Z->M,
 * 알파벳이 아닌 문자(숫자, 특수문자, 공백 등)는 변경하지 않고 그대로 둘
 */
class ROT13Encoder
{
    public const __version = '1.0';
    private $encrypt_str = "";

    public function __construct(string $str){
        $this->encrypt_str = $str;
    }
    public function encode() {
        return str_rot13($this->encrypt_str);
    }
    public function decode() {
        return str_rot13($this->encrypt_str);
    }
}

```

--- 파일 경로: classes/Date/DateTimez.php ---

```

<?php
namespace Flex\Banana\Classes\Date;

use \DateTime;
use \DateTimeZone;
use \DateInterval;

class DateTimez extends DateTime
{
    public const __version = '1.2';
    public DateTimeZone $dateTimeZone;
    public string $timezone;
    public array $location = [];
    public array $abbreviations = [];

    # time() || now || today || yesterday , Asia/Seoul
    public function __construct(string|int $times="now", string $timezone="")

```

```

{
    # timezone
    if(!$timezone && function_exists('date_default_timezone_get')){
        $timezone = date_default_timezone_get();
    }
    $timezone = $timezone ?? 'Asia/Seoul';
    $this->dateTimeZone = new DateTimeZone($timezone);
    $this->timezone = $this->dateTimeZone->getName();
    if(is_array($this->dateTimeZone->getLocation())){
        $this->location = $this->dateTimeZone->getLocation();
    }
    $this->filterAbbreviations(DateTimeZone::listAbbreviations());

    # datetime
    parent::__construct($this->chkTimestamp($times), $this->dateTimeZone);
}

private function filterAbbreviations(array $args) : void
{
    foreach($args as $abbr => $revisions){
        foreach($revisions as $rv){
            if($rv['timezone_id']){
                if($this->timezone == $rv['timezone_id']){
                    $this->location['abbr'] = strtoupper($abbr);
                    $this->location['offset'] = $rv['offset'];
                    $this->location['utc'] = ($rv['offset'] != 0) ? $rv['offset'] / 3600 : 0;
                    $this->location['dst'] = $rv['dst'];
                    break;
                }
            }
        }
    }
}

public function chkTimestamp(string|int $times) : string
{
    # datetime
    $_times = $times;
    if(is_int($times)){
        $_times = '@'. $times;
    }
    return $_times;
}

# modify, add, sub 기능
public function formatter(string $formatter) : DateTimeZone
{
    if(strpos($formatter, '-P') !== false){
        parent::sub(new DateInterval( str_replace('-', '', $formatter) ));
    }else if(substr($formatter, 0, 1) == 'P'){
        parent::add(new DateInterval($formatter));
    }else if(substr($formatter, 0, 2) == '+P'){
        parent::add(new DateInterval(str_replace('+', '', $formatter)));
    }else{
        parent::modify($formatter);
    }
    return $this;
}

# 날짜를 배열로 / 오늘이 무슨요일인지 / 1년중 몇째날짜인지 포함
public function parseDate(string $formatter = 'Y-m-d H:i:s') : array
{
    $result = [];
    $localtimes = localtime(parent::format('U'), true);
    $result = date_parse(parent::format($formatter));
    $result['wday'] = $localtimes['tm_wday']; # 오늘이 주의 무슨요일에 해당하는지 0~6
    $result['yday'] = $localtimes['tm_yday']; # 오늘이 1년중 몇번째 날인지 체크 100/365
    unset($result['warning_count'], $result['warnings'], $result['error_count'], $result['errors']);

    return $result;
}

# 일몰/일출 정보
public function sunInfo() : array
{
    $result = [
        'timezone' => $this->timezone,
        'country_code' => $this->location['country_code'] ?? '',
        'today' => parent::format('Y-m-d')
    ];
}

```

```

    };
    if(isset($this->location['latitude']) && $this->location['latitude']){
        $sun_info = date_sun_info(parent::format('U'), $this->location['latitude'], $this->location['longitude']);
        if(is_array($sun_info)){
            foreach ($sun_info as $fieldname => $val) {
                $result[$fieldname] = match ($val) {
                    true => 'always',
                    false => 'never',
                    default => date_create("@". $val)->setTimezone($this->dateTimeZone)->format('H:i:s')
                };
            }
        }
    }
    return $result;
}
}

```

--- 파일 경로: classes/Date/DateTimezPeriod.php ---

```

<?php
namespace Flex\Banana\Classes\Date;

use Flex\Banana\Classes\Date\DateTimez;
use \DateTimeImmutable;
use \DateInterval;
use \DatePeriod;
use \Exception;

class DateTimezPeriod
{
    public const __version = '1.2';

    # Asia/Seoul
    public string $timezone = "";
    private array $relative_pos = [
        'year','month','day','hour','minute','second'
    ];

    public function __construct(string $timezone="")
    {
        # timezone
        if(!$timezone && function_exists('date_default_timezone_get')){
            $timezone = date_default_timezone_get();
        }
        $this->timezone = $timezone ?? 'Asia/Seoul';
    }

    /**
     * 특정 날짜와 티켓 날짜사이 시간차
     * format : 시간차 포맷
     * demical : 소수점 자리
     */
    public function diff(string $start_date, string $end_date, array $formatter = ["format"=>'default','demical'=>'2']) : mixed
    {
        $s = new DateTimeImmutable($start_date);
        $e = new DateTimeImmutable($end_date);
        $interval = $s->diff($e);

        # 원하는 데이터 형
        $result = match($formatter['format']) {
            'days','day' => $interval->days,
            'seconds' => $interval->days * 86400 + $interval->h * 3600 + $interval->i * 60 + $interval->s,
            'minutes' => ($interval->days * 86400 + $interval->h * 3600 + $interval->i * 60 + $interval->s) / 60,
            'hours' => ($interval->days * 86400 + $interval->h * 3600 + $interval->i * 60 + $interval->s) / 3600,
            'minutes:seconds','i:s' => sprintf("%02d:%02d", ( ($interval->days * 86400 + $interval->h * 3600 + $interval->i * 60) / 60), $interval->s),
            'hours:minutes:seconds','h:i:s' => sprintf("%02d:%02d:%02d", ( ($interval->days * 86400 + $interval->h * 3600) / 3600), ( $interval->i * 60 / 60), $interval->s),
            'months' => (( $interval->m / 12) + ( $interval->days / 30)),
            'months:days:hours:minutes:seconds','m-d h:i:s' => sprintf("%02d-%02d %02d:%02d:%02d", $interval->m, $interval->d, $interval->h, $interval->i, $interval->s),
            'top' => $interval->format("%Y-%M-%d %H:%i:%s"),
            default => $interval->format("%Y-%M-%D %H:%I:%S")
        };

        # 소수점 자리 및 버리기
        if(isset($formatter['demical'])){

```

```

        if( in_array($formatter['format'], ['minutes','hours','months'])){
            if((int)$formatter['demical'] > 0){
                $_f = (int)"1".str_repeat("0", $formatter['demical']);
                $result = floor($result * $_f)/$_f;
            }else {
                $result = floor($result);
            }
        }
    }
}

# 시간이 큰것만 우선 순위 출력 약 시간 표시용
# 약 1분전, 약 1시간전, 약10일전
if($formatter['format'] == 'top')
{
    $relative_timef = $this->aboutTopTime ($result);
    if($relative_timef){
        $result = $relative_timef;
    }
}

return $result;
}

# 날짜와 날짜 사이 날짜
/**
 * interval : 1 // 날짜(1일, 3일)간격
 * days : 30 // 며칠(30일/개)
 */
public function period(string $start_date, int $interval, int $days) : array
{
    $result = [];

    $format = sprintf("P%dD", $interval);
    $startDateTimez = new DateTime($start_date, $this->timezone);
    $interval = new DateInterval($format);
    $period = new DatePeriod($startDateTimez, $interval, $days);
    foreach($period as $dateTimez){
        $result[] = $dateTimez->format("Y-m-d");
    }

    return $result;
}

# 최상위 순서대로만 표시 y > m > d > h > i > s
private function aboutTopTime (string $relative) : string
{
    $result = "";
    $argv = explode("-",strtr($relative,['>':'>','>':'>']));
    foreach($argv as $idx => $v){
        if($v > 0){
            $formatter = ($v > 1) ? $this->relative_pos[$idx].':s' : $this->relative_pos[$idx];
            $result = sprintf("%d %s",$v,$formatter);
            break;
        }
    }
}

return $result;
}
}

```

--- 파일 경로: classes/Db/CipherMysqlAes256Cbc.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\Cipher;
use Flex\Banana\Classes\Db\DbCipherInterface;
use Flex\Banana\Classes\Db\DbManager;
use \PDO;
use \Exception;

class CipherMysqlAes256Cbc implements DbCipherInterface
{
    public const __version = '0.1';
}

```

```

private const RANDOM_BYTES = 16; // IV 길이
public const ENCRYPTION_MODE = 'aes-256-cbc';

public function __construct(
    private string $hashkey,
    private DbManager $dbManager
){
    $this->dbManager = $dbManager;
}

public function setHashKey(string $hashkey) : self
{
    $this->hashkey = $hashkey;
    return $this;
}

public function encrypt(string $column): string
{
    return "HEX(AES_ENCRYPT('{ $column}', SHA2('{ $this->hashkey}', 512), RANDOM_BYTES('" . self::RANDOM_BYTES . "'))";
}

public function decrypt(string $column): string
{
    return "AES_DECRYPT(UNHEX($column), SHA2('{ $this->hashkey}', 512))";
}

public function set_encryption_mode(): void
{
    $pdo = $this->dbManager->pdo;

    # 서버 버전
    $mysql_version = $pdo->getAttribute(PDO::ATTR_SERVER_VERSION);
    if (version_compare($mysql_version, '5.7.0', '<')) {
        throw new Exception(sprintf("Setting(%s) is not possible for MySQL version lower than 5.7.0", CipherMysqlAes256Cbc::ENCRYPTION_MODE));
    }

    # encryption_mode 확인
    $encryption_mode_qry = "SELECT @@session.block_encryption_mode as em";
    $stmt = $pdo->query($encryption_mode_qry);
    $encryption_row = $stmt->fetch(PDO::FETCH_ASSOC);

    if (isset($encryption_row['em'])) {
        if ($encryption_row['em'] !== CipherMysqlAes256Cbc::ENCRYPTION_MODE) {
            $set_encrypt_qry = sprintf("SET @@session.block_encryption_mode = '%s'", CipherMysqlAes256Cbc::ENCRYPTION_MODE);
            $pdo->exec($set_encrypt_qry);
        }
    }
}
}
}

```

--- 파일 경로: classes/Db/CipherPgsqlAes.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\DbCipherInterface;

class CipherPgsqlAes implements DbCipherInterface
{
    public const __version = '0.1';
    public const ENCRYPTION_MODE = 'aes';
    public function __construct(
        private string $hashkey,
    ){
    }

    public function encrypt(string $column): string
    {
        return sprintf(
            "encode(encrypt(convert_to('%s', 'UTF8'), '".hash('sha256',$this->hashkey)."',".self::ENCRYPTION_MODE."'),'hex')",
            $column
        );
    }

    public function decrypt(string $column): string

```



```

    {
        return sprintf(
            "convert_from(decrypt(decode(%s, 'hex'), '''.hash('sha256',$this->hashkey).''', '''.self::ENCRYPTION_MODE.''''), 'UTF8')",
            $column
        );
    }
}

```

--- 파일 경로: classes/Db/CipherPgsqIAes256Cbc.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\DbCipherInterface;

class CipherPgsqIAes256Cbc implements DbCipherInterface
{
    public const __version = '0.1';
    public const ENCRYPTION_MODE = 'aes-cbc';
    public function __construct(
        private string $hashkey,
    ){}

    public function encrypt(string $column): string
    {
        return sprintf(
            "encode(encrypt(convert_to('%s', 'UTF8'), '''.hash('sha256',$this->hashkey).''', '''.self::ENCRYPTION_MODE.''''), 'hex')",
            $column
        );
    }

    public function decrypt(string $column): string
    {
        return sprintf(
            "convert_from(decrypt(decode(%s, 'hex'), '''.hash('sha256',$this->hashkey).''', '''.self::ENCRYPTION_MODE.''''), 'UTF8')",
            $column
        );
    }
}

```

--- 파일 경로: classes/Db/CipherPgsqIBasic.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\DbCipherInterface;

class CipherPgsqIBasic implements DbCipherInterface
{
    public const __version = '0.1';
    public function __construct(
    ){}

    public function encrypt(string $column): string
    {
        return sprintf(
            "encode(convert_to('%s', 'UTF8'), 'hex')",
            $column
        );
    }

    public function decrypt(string $column): string
    {
        return sprintf(
            "convert_from(decrypt(decode(%s, 'hex'), 'UTF8')",
            $column
        );
    }
}

```

--- 파일 경로: classes/Db/DbCipherGeneric.php ---

```
<?php
namespace Flex\Banana\Classes\Db;

use \ReflectionClass;
use \Exception;

class DbCipherGeneric
{
    public const __version = '1.0';

    private $processor;
    private static $allowedProcessors = [
        CipherMysqlAes256Cbc::class,
        CipherPgsqAes256Cbc::class,
        CipherPgsqBasic::class,
        CipherPgsqAes::class,
    ];

    public function __construct($processor)
    {
        $this->setProcessor($processor);
    }

    private function setProcessor($processor): void
    {
        $reflection = new ReflectionClass($processor);
        if (!in_array($reflection->getName(), self::$allowedProcessors)) {
            throw new Exception("Unsupported processor type: " . $reflection->getName());
        }

        $this->processor = $processor;
    }

    public function __call($name, $arguments)
    {
        $reflection = new ReflectionClass($this->processor);
        if (!$reflection->hasMethod($name)) {
            throw new Exception("Method $name does not exist in " . get_class($this->processor));
        }

        $method = $reflection->getMethod($name);
        if (!$method->isPublic()) {
            throw new Exception("Method $name is not public in " . get_class($this->processor));
        }

        return $method->invokeArgs($this->processor, $arguments);
    }

    public static function addProcessor(string $processorClass): void
    {
        if (!class_exists($processorClass)) {
            throw new Exception("Class $processorClass does not exist");
        }

        if (!in_array($processorClass, self::$allowedProcessors)) {
            self::$allowedProcessors[] = $processorClass;
        }
    }

    public static function getAllowProcessors(): array
    {
        return self::$allowedProcessors;
    }
}
```

--- 파일 경로: classes/Db/DbCipherInterface.php ---

```

<?php
namespace Flex\Banana\Classes\Db;
interface DbCipherInterface
{
    public function encrypt(string $value): string;
    public function decrypt(string $value): string;
}

```

--- 파일 경로: classes/Db/DbCouch.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Log;
use Flex\Banana\Classes\Json\JsonEncoder;
use Flex\Banana\Classes\Db\DbResultCouch;
use Flex\Banana\Classes\Db\DbInterface;
use Flex\Banana\Classes\Http\HttpRequest;
use Flex\Banana\Classes\Array\ArrayHelper;
use Flex\Banana\Classes\Date\DateTimeex;
use \Exception;
use \ArrayAccess;

class DbCouch extends QueryBuilderAbstractCouch implements DbInterface, ArrayAccess
{
    public const __version = '0.3.1';
    private const BASE_URL = "http://{host}:{port}";

    public string $baseUrl;
    private string $authHeader;
    private string $database;
    private array $params = [];
    private array $executeQueries = [];
    private string $table = "";

    public function __construct(
        WhereCouch $whereCouch
    )
    {
        parent::__construct($whereCouch);
        $this->baseUrl = "";
        $this->authHeader = "";
        $this->database = "";
    }

    # @ DbSqlInterface
    public function connect(string $host, string $dbname, string $user, string $password, int $port, string $charset, ?array $options = []): self
    {
        $this->baseUrl = $this->bindingDNS(self::BASE_URL, [
            "host" => $host,
            "port" => $port
        ]);

        $this->authHeader = "Authorization: Basic " . base64_encode("$user:$password");
        $httpRequest = new HttpRequest();
        $httpRequest->set($this->baseUrl, "", [$this->authHeader, "Content-Type: application/json"]);
        $httpRequest->get(function($response) {
            if (empty($response) || isset($response[0]['error'])) {
                throw new Exception("Failed to connect to CouchDB server");
            }
        });

        return $this->selectDB($dbname);
    }

    # @ DbSqlInterface
    public function selectDB(string $dbname): self
    {
        $httpRequest = new HttpRequest();
        $dbUrl = $this->baseUrl . "/" . $dbname;
        if ($this->database != $dbname) {
            $httpRequest->set($dbUrl, "", [$this->authHeader, "Content-Type: application/json"]);
            $httpRequest->get(function($response) use ($dbname) {
                if (empty($response) || isset($response[0]['error'])) {

```

```

        throw new Exception("Failed to connect to database '$dbname'");
    }
});
$this->database = $dbname;
}
return $this;
}

# @ DbSqlInterface
public function whereHelper(): WhereCouch
{
    return $this->whereCouch;
}

# @ DbSqlInterface
public function query(string $query = "", array $params = []): DbResultCouch | array
{
    if (!$query) {
        $query = JsonEncoder::toJson([$this->get()]);
    }

    try{
        $httpRequest = new HttpRequest();
        $url = ($this->table) ? $this->baseUrl."/({$this->database}/_partition/{$this->table}/_find" : $this->baseUrl."/({$this->database}/_find";
        $params = json_decode($query,true);
        foreach($params as $param){
            $httpRequest->set($url, JsonEncoder::toJson($param), [$this->authHeader,'Content-Type: application/json']);
        }

        $result = [];
        $httpRequest->post(function($response) use (&$result) {
            foreach($response as $body){
                if (empty($body) || isset($body['error'])) {
                    throw new Exception("Query failed: " . ($body['error'] ?? 'Unknown error'));
                }
                $result[] = new DbResultCouch($body);
            }
        });

        if(count($result)==1){
            $result = $result[0];
        }
        return $result;
    }catch(Exception $e){
        throw new Exception($e->getMessage());
    }
}

# @ DbSqlInterface
public function insert(): void
{
    if (empty($this->params)) {
        throw new Exception("Empty params");
    }

    if(!isset($this->params['_id']) && $this->table){
        $this->params['_id'] = $this->table.'.'.$this->generate_id();
    }

    $this->executeQueries[] = [
        "params" => $this->params
    ];

    parent::init();
    $this->params = [];
}

private function generate_id() : string {
    $now = (new DateTime("now"))->format("YmdHis");
    $microtodate = $now . substr((string)microtime(), 2, 6);
    $uniqid = substr(uniqid(rand(), true), 0, 6); // 고유한 짧은 문자열
    return $microtodate.'.'.$uniqid;
}

# @ DbSqlInterface
public function update(): void
{
    if (empty($this->params)) {

```

```

        throw new Exception("Empty parameters or selector is missing");
    }

    # where 문에서 _id 값 찾기
    if(!isset($this->params['_id']))
    {
        $selectors = (isset($this->query_params['selector'][$_id])) ? $this->query_params['selector'][$_id]: $this->query_params['selector'];
        $wheres = (new ArrayHelper($selectors))->select("_id")->value;
        if(!isset($wheres[0]) || !isset($wheres[0]['_id'])){
            throw new Exception("Empty _id is missing");
        }

        $_id = array_values($wheres[0]['_id'])[0];
        $this->params['_id'] = $_id;
    }

    # _id 값만 추출하기 및 fields 등록
    if(!isset($this->params['_id'])){
        throw new Exception("Empty _id value is missing");
    }

    # _rev 가 있는지 체크
    if(!isset($this->params['_rev'])){
        throw new Exception("Empty _rev is missing");
    }

    $this->executeQueries[] = [
        "params" => $this->params
    ];

    parent::init();
    $this->params = [];
}

# @ DbSqlInterface
public function delete(): void
{
    # where 문에서 _id 값 찾기
    if(!isset($this->params['_id'])){
        $selectors = (isset($this->query_params['selector'][$_id])) ? $this->query_params['selector'][$_id]: $this->query_params['selector'];
        $wheres = (new ArrayHelper($selectors))->select("_id")->value;
        if(!isset($wheres[0]) || !isset($wheres[0]['_id'])){
            throw new Exception("Empty _id is missing");
        }

        $_id = array_values($wheres[0]['_id'])[0];
        $this->params['_id'] = $_id;
    }

    # _id 값만 추출하기 및 fields 등록
    if(!isset($this->params['_id'])){
        throw new Exception("Empty _id value is missing");
    }

    # _rev 가 있는지 체크
    if(!isset($this->params['_rev'])){
        throw new Exception("Empty _rev is missing");
    }
    $this->params['_deleted'] = true;

    $this->executeQueries[] = [
        "params" => $this->params
    ];

    parent::init();
    $this->params = [];
}

public function createDatabase(string $dbname): void
{
    try {
        $httpRequest = new HttpRequest();
        $dbUrl = $this->baseUrl . "/" . $dbname;
        $httpRequest->set($dbUrl, "", [$this->authHeader, "Content-Type: application/json"]);
        $httpRequest->put(function($response) use ($dbname) {
            if (empty($response) || !isset($response[0]['error'])) {
                throw new Exception("Failed to create database '$dbname'");
            }
        });
    }
}

```

```

    } catch (Exception $e) {
        throw new Exception($e->getMessage());
    }
}

# @ QueryBuilderAbstractCouch
public function total(string $column_name = '_id'): int
{
    $this->set('fields', [$column_name]);
    $this->set('limit', 1);
    $query = $this->get();
    $query['execution_stats'] = true;

    $result = $this->query(JsonEncoder::toJson([$query]));

    if ($result instanceof DbResultCouch) {
        $executionStats = $result->get_execution_stats();
        if (is_array($executionStats) && isset($executionStats['total_docs'])) {
            return (int)$executionStats['total_docs'];
        }
    }

    // 실행 통계를 얻지 못한 경우, 전체 문서를 가져와서 카운트
    $this->init(); // 쿼리 파라미터 초기화
    $this->set('fields', ['_id']);
    $allDocsResult = $this->query(JsonEncoder::toJson([$this->get()]));

    return $allDocsResult->num_rows();
}

# @ QueryBuilderAbstractCouch
public function table(...$tables): self
{
    if(empty($tables[0])){
        throw new Exception("Empty table(type) is missing");
    }

    parent::init();
    $this->table = $tables[0];
    return $this;
}

# @ QueryBuilderAbstractCouch
public function select(...$columns) : self{
    if(count($columns) == 1){
        if(strpos($columns[0], ",") !== false) {
            $columns = explode(",", $columns[0]);
        }
    }
    if($columns[0] !== '*'){
        if(!in_array('._rev', $columns)){
            $columns[] = '._rev';
        }
        $this->set('fields', $columns);
    }
    return $this;
}

# @ QueryBuilderAbstractCouch
public function where(...$where): self
{
    $result = null;
    if(isset($where[0]) && $where[0]){
        $result = (!isset($where[1])) ? $where[0] : $this->buildWhere($where);
    }
    if($result !== null && $result){
        $this->set('selector', $result);
    }
    return $this;
}

# @ QueryBuilderAbstractCouch
public function orderBy(...$orderby): self
{
    $sort = [];
    foreach ($orderby as $field) {
        $direction = 'asc';
        $field = strtolower($field);
        if (strpos($field, ' desc') !== false) {

```

```

        $field = str_replace(' desc', '', $field);
        $direction = 'desc';
    }
    $sort[] = [$field => $direction];
}
$this->set('sort', $sort);
return $this;
}

# @ QueryBuilderAbstractCouch
public function limit(...$limit): self
{
    if (isset($limit[1])) {
        $this->set('skip', $limit[0]);
        $this->set('limit', $limit[1]);
    } else {
        $this->set('limit', $limit[0]);
    }
    return $this;
}

# @ QueryBuilderAbstractCouch
public function useIndex(...$index): self
{
    $this->set('use_index', $index);
    return $this;
}

public function beginTransaction(): void
{
    parent::init();
    $this->params = [];
    $this->executeQueries = [];
}

public function commit(): mixed
{
    $result = null;
    $executeQueries = $this->executeQueries;

    parent::init();
    $this->params = [];
    $this->executeQueries = [];

    $httpRequest = new HttpRequest();
    try {
        $bulkDocs = ['docs' => array_map(fn($query) => $query['params'], $executeQueries)];
        // $url = ($this->table) ? $this->baseUrl."/{$this->database}/_partition/{$this->table}/_bulk_docs" : $this->baseUrl."/{$this->database}/_bulk_docs";
        $url = $this->baseUrl."/{$this->database}/_bulk_docs";
        $params = JsonEncoder::toJson($bulkDocs);

        $httpRequest->set($url, $params, [$this->authHeader, "Content-Type: application/json"]);
        $result = $httpRequest->post(function($response) {
            foreach ($response as $body) {
                if (empty($body) || isset($body['error'])) {
                    throw new Exception("Bulk operation failed: " . ($body['error'] ?? 'Unknown error'));
                }
            }
            return true;
        });
    } catch (Exception $e) {
        throw new Exception($e->getMessage());
    }

    return $result;
}

public function rollBack(): void
{
    parent::init();
    $this->params = [];
    $this->executeQueries = [];
}

public function offsetSet($offset, $value): void
{
    $this->params[$offset] = $value;
}

# @ ArrayAccess

```

```

# 사용법 : isset($obj["two"]); -> bool(true)
public function offsetExists($offset) : bool{
    return isset($this->params[$offset]);
}

# @ ArrayAccess
# 사용법 : unset($obj["two"]); -> bool(false)
public function offsetUnset($offset) : void{
    unset($this->params[$offset]);
}

# @ ArrayAccess
# 사용법 : $obj["two"]; -> string(7) "A value"
public function offsetGet($offset) : mixed{
    return isset($this->params[$offset]) ? $this->params[$offset] : null;
}

public function __call($method, $args)
{
}

public function __get(string $propertyName)
{
    if ($propertyName == 'query') {
        return $this->get();
    } else {
        return $this->{$propertyName} ?? null;
    }
}
}

```

--- 파일 경로: classes/Db/DbInterface.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

# purpose : 각종 SQL 관련 디비를 통일성있게 작성할 수 있도록 틀을 제공
interface DbInterface
{
    public function connect(string $host, string $dbname, string $user, string $password, int $port, string $charset, ?array $options=[]) : self;
    public function selectDB( string $dbname ) : self;
    public function whereHelper() : WhereCouch|WhereSql;
    public function query(string $query="", array $params = []) : DbResultSql|DbResultCouch|array;      # 쿼리
    public function insert() : void;      # 저장
    public function update() : void;      # 수정
    public function delete() : void;      # 삭제
}

```

--- 파일 경로: classes/Db/DbManager.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Log;
use \ReflectionClass;
use \Exception;
use \ArrayAccess;

class DbManager implements ArrayAccess
{
    public const __version = '1.0.1';

    private $processor;
    private static $allowedProcessors = [
        DbMySql::class,
        DbPgSql::class,
        DbCouch::class,
    ];

    public function __construct($processor)

```



```

{
    $this->setProcessor($processor);
}

private function setProcessor($processor): void
{
    $reflection = new ReflectionClass($processor);
    if (!in_array($reflection->getName(), self::$allowedProcessors)) {
        throw new Exception("Unsupported processor type: " . $reflection->getName());
    }

    $this->processor = $processor;
}

public function offsetSet($offset, $value): void
{
    $this->processor->offsetSet($offset, $value);
}

public function offsetExists($offset): bool
{
    return $this->processor->offsetExists($offset);
}

public function offsetUnset($offset): void
{
    $this->processor->offsetUnset($offset);
}

public function offsetGet($offset): mixed
{
    return $this->processor->offsetGet($offset);
}

public function __call($name, $arguments)
{
    $reflection = new ReflectionClass($this->processor);
    if ($reflection->hasMethod($name)) {
        $method = $reflection->getMethod($name);
        if ($method->isPublic()) {
            return $method->invokeArgs($this->processor, $arguments);
        }
    }

    // 프로세서의 __call 메소드 호출
    if ($reflection->hasMethod('__call')) {
        return $this->processor->__call($name, $arguments);
    }

    throw new Exception("Method $name does not exist in " . get_class($this->processor));
}

public function __get(string $propertyName)
{
    return $this->processor->__get($propertyName);
}

public function __set(string $propertyName, mixed $value)
{
    return $this->processor->__set($propertyName, $value);
}

public static function addProcessor(string $processorClass): void
{
    if (!class_exists($processorClass)) {
        throw new Exception("Class $processorClass does not exist");
    }

    if (!in_array($processorClass, self::$allowedProcessors)) {
        self::$allowedProcessors[] = $processorClass;
    }
}

public static function getAllowProcessors(): array
{
    return self::$allowedProcessors;
}
}

```

--- 파일 경로: classes/Db/DbMySql.php ---

```
<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\QueryBuilderAbstractSql;
use Flex\Banana\Classes\Db\DbResultSql;
use Flex\Banana\Classes\Db\DbInterface;
use \PDO;
use \PDOException;
use \Exception;
use \ArrayAccess;

class DbMySql extends QueryBuilderAbstractSql implements DbInterface,ArrayAccess
{
    public const __version = '0.1.3';
    private const DSN = "mysql:host={host};port={port};dbname={dbname};charset={charset}";
    public $pdo;
    private $params = [];
    private array $pdo_options = [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
    ];

    public function __construct(
        WhereSql $whereSql
    ){
        parent::__construct($whereSql);
    }

    # @ DbSqlInterface
    public function connect(string $host, string $dbname, string $user, string $password, int $port, string $charset, ?array $options=[]): self
    {
        try {
            $dsn = $this->bindingDNS(self::DSN, [
                "host" => $host,
                "dbname" => $dbname,
                "port" => $port,
                "charset" => $charset
            ]);
            $this->pdo = new PDO($dsn, $user, $password, $this->pdo_options+$options);
        } catch (PDOException $e) {
            throw new Exception($e->getMessage());
        }

        return $this->selectDB( $dbname );
    }

    # @ DbSqlInterface
    public function selectDB( string $dbname ): self
    {
        $query ="SELECT DATABASE()";
        $result = $this->pdo->query($query)->fetchColumn();
        if ($result !== $dbname) {
            throw new Exception("Connected to database '$result' instead of '$dbname'");
        }
        return $this;
    }

    # @ DbSqlInterface
    public function whereHelper(): WhereSql
    {
        return $this->whereSql;
    }

    # @ DbSqlInterface
    public function query(string $query = "", array $params = []): DbResultSql
    {
        if (!$query) {
            $query = $this->query = parent::get();
        }

        // echo "Executing query: " . $query . PHP_EOL;
```

```

// print_r($params);

try {
    $stmt = $this->pdo->prepare($query);
    $result = $stmt->execute($params ?: null);
    if (!$result) {
        throw new Exception("Execution failed: " . implode(" ", $stmt->errorInfo()));
    }

    return new DbResultSql($stmt);
} catch (PDOException $e) {
    throw new Exception("Query failed: " . $e->getMessage());
}
}

protected function quoteIdentifier($identifier): string
{
    return "'" . str_replace("'", "", $identifier) . "'";
}

# @ DbSqlInterface
public function insert() : void {
    if (empty($this->params)) {
        throw new Exception("Empty : params");
    }

    $fields = [];
    $placeholders = [];
    $boundParams = [];

    foreach ($this->params as $field => $value) {
        $fields[] = $field;

        // Check for HEX(AES_ENCRYPT and encode(encrypt_iv
        if (is_string($value) && str_contains($value, 'HEX(AES_ENCRYPT()')) {
            $placeholders[] = $value; // Directly add the expression to placeholders
        } else {
            $placeholders[] = ":%field";
            $boundParams[":%field"] = $value;
        }
    }

    $query = sprintf(
        "INSERT INTO %s (%s) VALUES (%s)",
        $this->query_params['table'],
        implode(',', $fields),
        implode(',', $placeholders)
    );

    try {
        $this->params = [];
        $this->query($query, $boundParams);
    } catch (Exception $e) {
        throw new Exception("Query failed: " . $e->getMessage());
    }
}

# @ DbSqlInterface
public function update() : void {
    if (empty($this->params) || empty($this->query_params['where'])) {
        throw new Exception("Empty parameters or WHERE clause is missing");
    }

    $setClauses = [];
    $boundParams = [];

    foreach ($this->params as $field => $value) {
        if (is_string($value) && str_contains($value, 'HEX(AES_ENCRYPT()')) {
            $setClauses[] = "$field = $value";
        } else {
            $setClauses[] = "$field = :$field";
            $boundParams[":$field"] = $value;
        }
    }

    $query = sprintf(
        "UPDATE %s SET %s %s",
        $this->query_params['table'],
        implode(',', $setClauses),

```

```

        $this->query_params['where']
    );

    try {
        $this->params = [];
        $this->query($query, $boundParams);
    } catch (Exception $e) {
        throw new Exception("Query failed: " . $e->getMessage());
    }
}

# @ DbSqlInterface
public function delete() : void {
    $query = sprintf("DELETE FROM %s %s",
        $this->query_params['table'],
        $this->query_params['where']
    );
    try {
        $this->query($query);
    } catch (Exception $e) {
        throw new Exception("Query failed: " . $e->getMessage());
    }
}

# @ QueryBuilderAbstractSql
public function tableJoin(string $join, ...$tables) : self{
    parent::init('JOIN');

    $supcase = strtoupper($join);
    $implode_join = sprintf(" %s JOIN ", $supcase);
    switch($supcase){
        case 'UNION': # 중복제거
        case 'UNION ALL': # 중복포함
            parent::setQueryTpl('UNINON');
            $implode_join = sprintf(" %s ", $supcase);
            break;
        default :
            parent::setQueryTpl('default');
    }

    $value = implode($implode_join, $tables);
    parent::set('table', $value);
    return $this;
}

# @ QueryBuilderAbstractSql
public function select(...$columns) : self{
    $value = implode(', ', $columns);
    parent::set('columns', $value);
    return $this;
}

# @ QueryBuilderAbstractSql
public function where(...$where) : self
{
    $result = parent::buildWhere($where);
    if($result){
        $value = 'WHERE '.$result;
        parent::set('where', $value);
    }
    return $this;
}

# @ QueryBuilderAbstractSql
public function orderBy(...$orderby) : self
{
    $value = 'ORDER BY '.implode(', ', $orderby);
    parent::set('orderby', $value);
    return $this;
}

# @ QueryBuilderAbstractSql
public function on(...$on) : self
{
    $result = parent::buildWhere($on);
    if($result){
        $value = 'ON '.$result;
        parent::set('on', $value);
    }
}

```

```

return $this;
}

# @ QueryBuilderAbstractSql
public function limit(...$limit): self {
    $value = 'LIMIT ' . implode(',', $limit);
    parent::set('limit', $value);
    return $this;
}

# @ QueryBuilderAbstractSql
public function distinct(string $column_name): self{
    $value = sprintf("DISTINCT %s", $column_name);
    parent::set('columns', $value);
    return $this;
}

# @ QueryBuilderAbstractSql
public function groupBy(...$columns): self{
    $value = 'GROUP BY ' . implode(' ', $columns);
    parent::set('groupby', $value);
    return $this;
}

# @ QueryBuilderAbstractSql
public function having(...$having): self{
    $result = parent::buildWhere($having);
    if($result){
        $value = 'HAVING ' . $result;
        parent::set('having', $value);
    }
    return $this;
}

# @ QueryBuilderAbstractSql
public function total(string $column_name = ''): int {
    $value = sprintf("COUNT(%)s AS total_count", $column_name);
    parent::set('columns', $value);
    $query = parent::get();

    $result = $this->query($query);
    $row = $result->fetch_assoc();
    return (int)$row['total_count'] ?? 0;
}

# @ QueryBuilderAbstractSql
public function table(...$tables): self {
    parent::init('MAIN');
    $length = count($tables);
    $value = ($length == 2) ? $tables[0] . ' ' . $tables[1] : $tables[0];
    parent::set('table', $value);
    return $this;
}

# @ QueryBuilderAbstractSql
public function tableSub(...$tables): self{
    parent::init('SUB');
    $length = count($tables);
    $value = ($length == 2) ? implode(' ', $tables) : implode(' ', $tables);
    parent::set('table', $value);
    return $this;
}

# @ ArrayAccess
# 사용법 : $obj["two"] = "A value";
public function offsetSet($offset, $value): void {
    $this->params[$offset] = $value;
}

# @ ArrayAccess
# 사용법 : isset($obj["two"]); -> bool(true)
public function offsetExists($offset): bool{
    return isset($this->params[$offset]);
}

# @ ArrayAccess
# 사용법 : unset($obj["two"]); -> bool(false)
public function offsetUnset($offset): void{
    unset($this->params[$offset]);
}

```

```

    }

    # @ ArrayAccess
    # 사용법 : $obj["two"]; -> string(7) "A value"
    public function offsetGet($offset) : mixed{
        return isset($this->params[$offset]) ? $this->params[$offset] : null;
    }

    public function __call($method, $args)
    {
        return call_user_func_array([$this->pdo, $method], $args);
    }

    public function __get(string $propertyName) {
        if(property_exists(__CLASS__,$propertyName)){
            if($propertyName == 'query'){
                return parent::get();
            }else{
                return $this->{$propertyName};
            }
        }
    }
}
}
}

```

--- 파일 경로: classes/Db/DbPgsSql.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\QueryBuilderAbstractSql;
use Flex\Banana\Classes\Db\DbResultSql;
use Flex\Banana\Classes\Db\DbInterface;
use \PDO;
use \PDOException;
use \Exception;
use \ArrayAccess;

class DbPgsSql extends QueryBuilderAbstractSql implements DbInterface,ArrayAccess
{
    public const __version = '0.1.3';
    private const DSN = "pgsql:host={host};port={port};dbname={dbname}";

    public $pdo;
    private $params = [];
    private array $pdo_options = [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
    ];

    public function __construct(
        WhereSql $whereSql
    ){
        parent::__construct($whereSql);
    }

    # @ DbSqlInterface
    public function connect(string $host, string $dbname, string $user, string $password, int $port, string $charset, ?array $options=[]) : self
    {
        try {
            $dsn = $this->bindingDNS(self::DSN, [
                "host" => $host,
                "dbname" => $dbname,
                "port" => $port,
                "charset" => $charset
            ]);
            $this->pdo = new PDO($dsn, $user, $password, $this->pdo_options+$options);
        } catch (PDOException $e) {
            throw new Exception($e->getMessage());
        }

        return $this->selectDB( $dbname );
    }

    # @ DbSqlInterface

```

```

public function selectDB( string $dbname ): self
{
    $query = "SELECT current_database()";
    $result = $this->pdo->query($query)->fetchColumn();
    if ($result !== $dbname) {
        throw new Exception("Connected to database '$result' instead of '$dbname'");
    }
    return $this;
}

# @ DbSqlInterface
public function whereHelper(): WhereSql
{
    return $this->whereSql;
}

# @ DbSqlInterface
public function query(string $query = "", array $params = []): DbResultSql
{
    if (!$query) {
        $query = $this->query = parent::get();
    }

    // echo "Executing query: " . $query . PHP_EOL;
    // print_r($params);

    try {
        $stmt = $this->pdo->prepare($query);
        $result = $stmt->execute($params ?: null);
        if (!$result) {
            throw new Exception("Execution failed: " . implode(" ", $stmt->errorInfo()));
        }

        return new DbResultSql($stmt);
    } catch (PDOException $e) {
        throw new Exception("Query failed: " . $e->getMessage());
    }
}

protected function quoteIdentifier($identifier): string
{
    return "" . str_replace("'", "''", $identifier) . "";
}

# @ DbSqlInterface
public function insert() : void {
    if (empty($this->params)) {
        throw new Exception("Empty : params");
    }

    $fields = [];
    $placeholders = [];
    $boundParams = [];

    foreach ($this->params as $field => $value) {
        $fields[] = $field;

        // Check for HEX(AES_ENCRYPT and encode(encrypt_iv
        if (is_string($value) && (str_contains($value, 'encode('))) {
            $placeholders[] = $value; // Directly add the expression to placeholders
        } else {
            $placeholders[] = ":%field";
            $boundParams[":%field"] = $value;
        }
    }

    $query = sprintf(
        "INSERT INTO %s (%s) VALUES (%s)",
        $this->query_params['table'],
        implode(',', $fields),
        implode(',', $placeholders)
    );

    try {
        $this->params = [];
        $this->query($query, $boundParams);
    } catch (Exception $e) {
        throw new Exception("Query failed: " . $e->getMessage());
    }
}

```

```

}

# @ DbSqlInterface
public function update() : void {
    if (empty($this->params) || empty($this->query_params['where'])) {
        throw new Exception("Empty parameters or WHERE clause is missing");
    }

    $setClauses = [];
    $boundParams = [];

    foreach ($this->params as $field => $value) {
        if (is_string($value) && str_contains($value, 'encode(')) {
            $setClauses[] = "$field = $value";
        } else {
            $setClauses[] = "$field = :$field";
            $boundParams[":$field"] = $value;
        }
    }

    $query = sprintf(
        "UPDATE %s SET %s %s",
        $this->query_params['table'],
        implode(',', $setClauses),
        $this->query_params['where']
    );

    try {
        $this->params = [];
        $this->query($query, $boundParams);
    } catch (Exception $e) {
        throw new Exception("Query failed: " . $e->getMessage());
    }
}

# @ DbSqlInterface
public function delete() : void {
    $query = sprintf("DELETE FROM %s %s",
        $this->query_params['table'],
        $this->query_params['where']
    );

    try {
        $this->query($query);
    } catch (Exception $e) {
        throw new Exception("Query failed: " . $e->getMessage());
    }
}

# @ QueryBuilderAbstract
public function tableJoin(string $join, ...$tables) : self{
    parent::init('JOIN');

    $supcase = strtoupper($join);
    $implode_join = sprintf(" %s JOIN ", $supcase);
    parent::setQueryTpl('default');

    $value = implode($implode_join, $tables);
    parent::set('table', $value);
    return $this;
}

# @ QueryBuilderAbstract
public function select(...$columns) : self{
    $value = implode(',', $columns);
    parent::set('columns', $value);
    return $this;
}

# @ QueryBuilderAbstract
public function where(...$where) : self
{
    $result = parent::buildWhere($where);
    if($result){
        $value = 'WHERE ' . $result;
        parent::set('where', $value);
    }
    return $this;
}

# @ QueryBuilderAbstract

```



```

public function orderBy(...$orderby) : self
{
    $value = 'ORDER BY '.implode(' ', $orderby);
    parent::set('orderby', $value);
    return $this;
}

# @ QueryBuilderAbstract
public function on(...$on) : self
{
    $result = parent::buildWhere($on);
    if($result){
        $value = 'ON '. $result;
        parent::set('on', $value);
    }
    return $this;
}

# @ QueryBuilderAbstract
public function limit(...$limit): self {
    $value = match (count($limit)) {
        1 => 'LIMIT ' . $limit[0],
        2 => 'LIMIT ' . $limit[1] . ' OFFSET ' . $limit[0],
        default => throw new Exception("Invalid number of arguments for LIMIT clause")
    };

    parent::set('limit', $value);
    return $this;
}

# @ QueryBuilderAbstract
public function distinct(string $column_name) : self{
    $value = sprintf("DISTINCT %s", $column_name);
    parent::set('columns', $value);
    return $this;
}

# @ QueryBuilderAbstract
public function groupBy(...$columns) : self{
    $value = 'GROUP BY '.implode(' ', $columns);
    parent::set('groupby', $value);
    return $this;
}

# @ QueryBuilderAbstract
public function having(...$having) : self{
    $result = parent::buildWhere($having);
    if($result){
        $value = 'HAVING ' . $result;
        parent::set('having', $value);
    }
    return $this;
}

# @ QueryBuilderAbstract
public function total(string $column_name = '') : int {
    $value = sprintf("COUNT(%) AS total_count", $column_name);
    parent::set('columns', $value);
    $query = parent::get();

    $result = $this->query($query);
    $row = $result->fetch_assoc();
    return (int)$row['total_count'] ?? 0;
}

# @ QueryBuilderAbstract
public function table(...$tables) : self {
    parent::init('MAIN');
    $length = count($tables);
    $value = ($length == 2) ? $tables[0] . ' ' . $tables[1] : $tables[0];
    parent::set('table', $value);
    return $this;
}

# @ QueryBuilderAbstract
public function tableSub(...$tables) : self{
    parent::init('SUB');
    $length = count($tables);
    $value = ($length == 2) ? implode(' ', $tables) : implode(' ', $tables);

```

```

        parent::set('table', $value);
    return $this;
}

# @ ArrayAccess
# 사용법 : $obj["two"] = "A value";
public function offsetSet($offset, $value) : void {
    $this->params[$offset] = $value;
}

# @ ArrayAccess
# 사용법 : isset($obj["two"]); -> bool(true)
public function offsetExists($offset) : bool{
    return isset($this->params[$offset]);
}

# @ ArrayAccess
# 사용법 : unset($obj["two"]); -> bool(false)
public function offsetUnset($offset) : void{
    unset($this->params[$offset]);
}

# @ ArrayAccess
# 사용법 : $obj["two"]; -> string(7) "A value"
public function offsetGet($offset) : mixed{
    return isset($this->params[$offset]) ? $this->params[$offset] : null;
}

public function __call($method, $args)
{
    return call_user_func_array([$this->pdo, $method], $args);
}

public function __get(string $propertyName) {
    if(property_exists(__CLASS__, $propertyName)){
        if($propertyName == 'query'){
            return parent::get();
        }else{
            return $this->{$propertyName};
        }
    }
}
}
}

```

--- 파일 경로: classes/Db/DbResultCouch.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

class DbResultCouch {
    private $result;
    private $docs;
    private $currentIndex;
    private $numRows;

    public function __construct(string|array $result) {
        if(!is_array($result)){
            try {
                $result = json_decode($result, true);
            }catch (\JsonException $e) {
            }
        }
        $this->result = $result;

        $this->docs = $this->result['body']['docs'] ?? [];
        $this->currentIndex = 0;
        $this->numRows = count($this->docs);
    }

    public function fetch_assoc() {
        if ($this->currentIndex < $this->numRows) {
            return $this->docs[$this->currentIndex++];
        }
        return false;
    }
}

```

```

    }

    public function fetch_array() {
        if ($this->currentIndex < $this->numRows) {
            $doc = $this->docs[$this->currentIndex++];
            return array_merge($doc, array_values($doc));
        }
        return false;
    }

    public function fetch_row() {
        if ($this->currentIndex < $this->numRows) {
            return array_values($this->docs[$this->currentIndex++]);
        }
        return false;
    }

    public function fetch_object() {
        if ($this->currentIndex < $this->numRows) {
            return (object)$this->docs[$this->currentIndex++];
        }
        return false;
    }

    public function num_rows() {
        return $this->numRows;
    }

    public function fetch_all() {
        return array_map(function($doc) {
            return array_merge($doc, array_values($doc));
        }, $this->docs);
    }

    public function fetch_column($column = 0) {
        if ($this->currentIndex < $this->numRows) {
            $doc = $this->docs[$this->currentIndex++];
            $values = array_values($doc);
            return isset($values[$column]) ? $values[$column] : null;
        }
        return false;
    }

    // CouchDB 특화 메서드
    public function get_bookmark() {
        return $this->result['bookmark'] ?? null;
    }

    public function get_warning() {
        return $this->result['warning'] ?? null;
    }

    public function get_execution_stats() {
        return $this->result['execution_stats'] ?? null;
    }
}

```

--- 파일 경로: classes/Db/DbResultSql.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use PDO;
use PDOStatement;

class DbResultSql {
    private $statement;
    private $resultSet;
    private $currentRow;
    private $numRows;

    public function __construct(PDOStatement $statement) {
        $this->statement = $statement;
        $this->resultSet = null;
    }
}

```

```

        $this->currentRow = 0;
        $this->numRows = $statement->rowCount();
    }

    public function fetch_assoc() {
        return $this->statement->fetch(PDO::FETCH_ASSOC);
    }

    public function fetch_array($resultType = PDO::FETCH_BOTH) {
        return $this->statement->fetch($resultType);
    }

    public function fetch_row() {
        return $this->statement->fetch(PDO::FETCH_NUM);
    }

    public function fetch_object() {
        return $this->statement->fetch(PDO::FETCH_OBJ);
    }

    public function num_rows() {
        return $this->numRows;
    }

    public function fetch_all($resultType = PDO::FETCH_ASSOC) {
        if ($this->resultSet === null) {
            $this->resultSet = $this->statement->fetchAll($resultType);
        }
        return $this->resultSet;
    }

    public function fetch_column($column = 0) {
        return $this->statement->fetchColumn($column);
    }
}

```

--- 파일 경로: classes/Db/QueryBuilderAbstractCouch.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\WhereCouch;
use Flex\Banana\Classes\Log;
abstract class QueryBuilderAbstractCouch
{
    public const __version = '0.0.2';
    protected array $query_params;

    protected const _QUERY_INIT_PARAMS_ = [
        'selector' => ["_id" => ["$gt" => null]],
        'fields'   => [],
        'sort'     => [],
        'limit'    => null,
        'skip'     => null,
        'use_index' => null
    ];

    public function __construct(
        protected WhereCouch $whereCouch
    )
    {
        $this->init();
    }

    abstract public function table(...$tables) : mixed;
    abstract public function select(...$columns) : mixed;
    abstract public function where(...$where) : mixed;
    abstract public function orderBy(...$orderby) : mixed;
    abstract public function limit(...$limit) : mixed;
    abstract public function total(string $column_name) : int;
    abstract public function useIndex(...$index): self;

    public function init(): void
    {

```

```

        $this->query_params = self::_QUERY_INIT_PARAMS_;
    }

    public function set(string $key, $value): void
    {
        if($key == 'selector'){
            $this->query_params[$key] = new stdClass();
        }
        $this->query_params[$key] = $value;
    }

    public function get(): array
    {
        $query = [];
        foreach ($this->query_params as $key => $value) {
            if ($value !== null) {
                if($key == 'sort'){
                    if(!empty($this->query_params[$key])){
                        $query[$key] = $value;
                    }
                }else $query[$key] = $value;
            }
        }

        return $query;
    }

    public function bindingDNS (string $tpl, array $dsn_options) : string
    {
        preg_match_all("/\{(\{+\}(.*?))\}/", $tpl, $matches);
        $patterns = $matches[0];
        $columns = $matches[2];

        # binding
        foreach($patterns as $idx => $text){
            $column_name = $columns[$idx];
            $render_args[$text] = (trim($dsn_options[$column_name])) ? $dsn_options[$column_name] : "";
        }
        return trim(strtr($tpl, $render_args));
    }

    protected function buildWhere(array $conditions): array
    {
        $this->whereCouch->__construct();
        $this->whereCouch->begin('and');
        if (is_array($conditions) && count($conditions) >= 2) {
            if (count($conditions) == 2) {
                $this->whereCouch->case($conditions[0], '=', $conditions[1]);
            } elseif (count($conditions) == 3) {
                $this->whereCouch->case($conditions[0], $conditions[1], $conditions[2]);
            }
        }
        $this->whereCouch->end();
        return $this->whereCouch->__get('where');
    }
}

```

--- 파일 경로: classes/Db/QueryBuilderAbstractSql.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\WhereSql;

# purpose : 각종 SQL 관련 디비를 통일성있게 작성할 수 있도록 틀을 제공
abstract class QueryBuilderAbstractSql
{
    public const __version = '1.5.3';
    private string $query_mode;
    protected array $query_params;
    private array $sub_query_params;
    private string $query_tpl = "";
    private array $tpl = [
        'union' => '{table}{where}{groupby}{having}{orderby}{limit}',

```

```

'default' => 'SELECT {columns}FROM {table}{on}{where}{groupby}{having}{orderby}{limit}'
];
protected string $query = "";
const _QUERY_INIT_PARAMS_ = ['columns'=>"", 'table'=>'', 'where'=>'', 'orderby'=>'', 'on'=>'', 'limit'=>'', 'groupby'=>'', 'having'=>''];

abstract public function table(...$tables) : mixed;
abstract public function tableJoin(string $type,...$tables) : mixed;
abstract public function tableSub(...$tables) : mixed;
abstract public function select(...$columns) : mixed;
abstract public function where(...$where) : mixed;
abstract public function orderBy(...$orderby) : mixed;
abstract public function on(...$on) : mixed;
abstract public function limit(...$limit) : mixed;
abstract public function distinct(string $column_name) : mixed;
abstract public function groupBy(...$columns) : mixed;
abstract public function having(...$columns) : mixed;
abstract public function total(string $column_name) : int;

public function __construct(
    protected WhereSql $whereSql
)
{
    $this->init();
}

public function init(string $type = 'main') : void
{
    $this->query_mode = strtoupper($type);

    if($this->query_mode == 'JOIN'){
        $this->sub_query_params = [];
        $this->query_params = [];
        $this->query_params = self::_QUERY_INIT_PARAMS_;
    }
    else if($this->query_mode == 'SUB'){
        $this->query_tpl = $this->tpl['default'];
        $this->sub_query_params = [];
        $this->sub_query_params = self::_QUERY_INIT_PARAMS_;
    }
    else {
        $this->sub_query_params = [];
        $this->query_params = [];
        $this->query_params = self::_QUERY_INIT_PARAMS_;
        $this->query_tpl = $this->tpl['default'];
    }
}

public function setQueryTpl (string $tpl_mode){
    $supcase = strtoupper($tpl_mode);
    if($supcase == 'UNINON') $this->query_tpl = $this->tpl['union'];
    else $this->query_tpl = $this->tpl['default'];
}

public function set(string $style, string $value) : void {
    if($this->query_mode == 'SUB') $this->sub_query_params[$style] = $value;
    else $this->query_params[$style] = $value;
}

public function get() : string
{
    preg_match_all("/{+}(.*?)}/{+}/", $this->query_tpl, $matches);
    $patterns = $matches[0];
    $columns = $matches[2];

    $render_args = [];
    $query_params = [];
    $query_params = ($this->query_mode == 'SUB') ? $this->sub_query_params : $this->query_params;

    # binding
    foreach($patterns as $idx=>$text){
        $column_name = $columns[$idx];
        $render_args[$text] = (trim($query_params[$column_name])) ? $query_params[$column_name] : '.';
    }
    $this->query = trim(strtr($this->query_tpl, $render_args));

    # reset
    if($this->query_mode == 'SUB' || $this->query_mode == 'JOIN') {
        $this->query_mode = 'MAIN';
    }
}
return $this->query;

```

```

}

public function bindingDNS (string $tpl, array $dsdn_options) : string
{
    preg_match_all("/{+}(.*?)()/", $tpl, $matches);
    $patterns = $matches[0];
    $columns = $matches[2];

    # binding
    foreach($patterns as $idx => $text){
        $column_name = $columns[$idx];
        $render_args[$text] = (trim($dsdn_options[$column_name])) ? $dsdn_options[$column_name] : "";
    }
    return trim(strtr($tpl, $render_args));
}

public function buildWhere(...$w) : string
{
    $result = "";
    $length = (isset($w[0])) ? count($w[0]) : 0;
    if($length > 0)
    {
        $wa = $w[0];
        if(isset($wa[0]) && $wa[0])
        {
            $result = $wa[0];
            if($length > 1)
            {
                $this->whereSql->__construct();
                # 배열
                if(is_array($wa[0]))
                {
                    $this->whereSql->begin('AND');
                    foreach($wa as $idx => $argv)
                    {
                        $argv_length = count($argv);
                        if($argv_length == 2){
                            $this->whereSql->case($argv[0], '=', $argv[1]);
                        }else if($argv_length == 3){
                            $this->whereSql->case($argv[0], $argv[1], $argv[2]);
                        }
                    }
                    $this->whereSql->end();
                }else{ # string
                    if($length == 2){
                        $this->whereSql->begin('AND')->case($wa[0], '=', $wa[1])->end();
                    }else if($length == 3){
                        $this->whereSql->begin('AND')->case($wa[0], $wa[1], $wa[2])->end();
                    }
                }
            }
            $result = $this->whereSql->__get('where');
        }
    }
    return $result;
}
}

```

--- 파일 경로: classes/Db/WhereCouch.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\WhereInterface;
use Flex\Banana\Classes\Log;

# 데이터베이스 QUERY구문에 사용되는 WHERE문 만드는데 도움을 주는 클래스
class WhereCouch implements WhereInterface
{
    public const __version = '0.1.0';
    private array $where_group = [];
    private string $current_group = "";
    private string $current_coord = "";
    private array $where_groups_data = [];
}

```

```
private string $coord = 'AND'; # 전체 그룹을 마지막으로 묶을 coord
```

```
# void
# @fields : name+category+area 복수필드
# @coord : [AND | OR]
public function __construct(string $coord = 'AND')
{
    $this->coord = $coord;
    $this->init();
}
```

```
# where 그룹룩기 시작
public function begin(string $coord) : WhereCouch
{
    // Log::d(__CLASS__, __METHOD__, $coord);
    $groupname = strtr(microtime(), [' '=>', '0.'=>'w']);
    if(!isset($this->where_group[$groupname])){
        $this->where_group[$groupname] = [];
    }

    # end 자동닫기
    if($this->current_group){
        $this->end();
    }

    # 현재그룹 시작
    $this->current_group = $groupname;
    $this->current_coord = $coord;
    return $this;
}
```

```
# where 그룹룩기 종료
public function end() : WhereCouch
{
    // Log::d(__METHOD__, $this->where_group);
    // 현재 그룹에 조건이 있으면
    if (count($this->where_group[$this->current_group])) {
        // 조건을 배열로 저장
        $group_conditions = ['$' . strtolower($this->current_coord) => []];
        foreach ($this->where_group[$this->current_group] as $condition) {
            $group_conditions['$' . strtolower($this->current_coord)][] = $condition;
        }
        // 조건을 and, or로 구분하여 추가
        $this->where_groups_data[] = $group_conditions;
    }

    // 현재 그룹과 coord 초기화
    $this->current_group = "";
    $this->current_coord = "";

    return $this;
}
```

```
# void
# 구문어를 만든다.
# @where_str : name='홍길동'
# @condition : [=,!=,<,>,<=,>=,IN,LIKE-R=dd%,LIKE-L=%dd,LIKE=%dd%]
# @value : NULL | VALUE | % | Array
public function case(string $field_name, string $condition ,mixed $value, bool $is_qutawrap=true, bool $join_detection=true) : WhereCouch
{
    // Log::d(__METHOD__);
    $is_append = false;
    if($value == "0") $is_append = true;
    else if($value && $value != " "){
        $is_append = true;
    }

    # where 문을 그룹별로 묶기
    if($is_append)
    {
        $in_value = [];
        if (is_array($value)){ // array
            $in_value = $value;
        } else if (strpos($value, ",") !=false){
            $in_value = explode(",", $value);
        } else{
            $in_value[] = $value;
        }
        $condition = strtolower($condition);
    }
}
```



```

if($condition == 'like' || $condition == 'like-r' || $condition == 'like-l'){
    foreach($in_value as $word)
    {
        // $_word = preg_replace("/[#&!+%-@=\\|:;,\\.\'\"^~_\\/\\/?\\*\\$#<>()\\[\\]\\{\\}/i",'',$word);
        $_word = preg_replace("/[#&!+%-@=\\|:;,\\.\'\"^~_\\/\\/?\\*\\$#<>()\\[\\]\\{\\}/i",'',$word);

        $this->where_group[$this->current_group][[$field_name]] = [
            'regex' => $this->buildRegexForLike($condition, $_word)
        ];
    }
}
else if($value == 'null'){
    $this->where_group[$this->current_group][[$field_name]] = null;
}
else{
    // 다른 조건 처리
    if (count($in_value) === 1) {
        // 단일 값일 경우
        $this->where_group[$this->current_group][[$field_name]] = [
            $this->mapConditionToOperator($condition) => $in_value[0] // 단일 값 사용
        ];
    } else {
        // 다수의 값일 경우 (IN 조건)
        $this->where_group[$this->current_group][[$field_name]] = [
            $this->mapConditionToOperator($condition) => $in_value
        ];
    }
}
}
return $this;
}

# 상속한 부모 프라퍼티 값 포함한 가져오기
public function __get($propertyName)
{
    // Log::d(__METHOD__, $propertyName);
    if ($propertyName == 'where')
    {
        // 아직 종료되지 않은 그룹이 있으면 종료
        if ($this->current_group) {
            $this->end();
        }

        // Log::d('where_groups_data', $this->where_groups_data);

        // 그룹 데이터를 배열로 리턴
        $result = [];
        if(count($this->where_groups_data) > 1){
            $_coord = '$' . strtolower($this->coord);
            $result = [$_coord => $this->where_groups_data];
        }else{
            $result = (isset($this->where_groups_data[0])) ? $this->where_groups_data[0] : $this->where_groups_data;
        }
        $this->init();
        return $result;
    }else{
        return $this->{$propertyName};
    }
}

private function mapConditionToOperator(string $condition): string
{
    return match ($condition) {
        '=' => 'Seq',
        '!=' => 'Sne',
        '<' => 'Sit',
        '<=' => 'Site',
        '>' => 'Sgt',
        '>=' => 'Sgte',
        'in' => 'Sin',
        'not in' => 'Snin',
        default => 'Seq'
    };
}

private function buildRegexForLike(string $condition, string $value): string
{
    $value = preg_quote($value, '/');
    return match ($condition) {

```

```

        'like' => ".*$value.*",
        'like-r' => "$value.*",
        'like-l' => ".*$value",
        default => $value,
    };
}

# 초기화
private function init() : void {
    $this->current_group = "";
    $this->current_coord = "";
    $this->where_group = [];
    $this->where_groups_data = [];
}

public function fetch() : array
{
    if ($this->current_group) {
        $this->end();
    }
    $result = $this->where_group;
    $this->init();
    return $result;
}

public function __destruct(){
    $this->current_group = "";
    $this->current_coord = "";
    $this->where_group = [];
    $this->where_groups_data = [];
}
}

```

--- 파일 경로: classes/Db/WhereHelper.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\WhereInterface;
use \ReflectionClass;
use \Exception;

class WhereHelper
{
    public const __version = '1.0.1';

    private $processor;
    private static $allowedProcessors = [
        WhereSql::class,
        WhereCouch::class,
    ];

    public function __construct($processor)
    {
        $this->setProcessor($processor);
    }

    private function setProcessor($processor): void
    {
        $reflection = new ReflectionClass($processor);
        if (!in_array($reflection->getName(), self::$allowedProcessors)) {
            throw new Exception("Unsupported processor type: " . $reflection->getName());
        }

        $this->processor = $processor;
    }

    public function case(string $field_name, string $condition, mixed $value, bool $is_qutawrap = true, bool $join_detection = true): self{
        $this->processor->case($field_name, $condition, $value, $is_qutawrap, $join_detection);
        return $this;
    }

    public function begin(string $coord): self{
        $this->processor->begin($coord);
    }
}

```

```

        return $this;
    }
    public function end(): self{
        $this->processor->end();
        return $this;
    }
    public function fetch(): array{
        return $this->processor->fetch();
    }

    public function __call($name, $arguments)
    {
        $reflection = new ReflectionClass($this->processor);
        if ($reflection->hasMethod($name)) {
            $method = $reflection->getMethod($name);
            if ($method->isPublic()) {
                return $method->invokeArgs($this->processor, $arguments);
            }
        }

        // 프로세서의 __call 메소드 호출
        if ($reflection->hasMethod('__call')) {
            return $this->processor->__call($name, $arguments);
        }

        throw new Exception("Method $name does not exist in " . get_class($this->processor));
    }

    public function __get($propertyName) : mixed
    {
        return $this->processor->__get($propertyName);
    }

    public function __destruct()
    {
        $this->processor->__destruct();
    }

    public static function addProcessor(string $processorClass): void
    {
        if (!class_exists($processorClass)) {
            throw new Exception("Class $processorClass does not exist");
        }

        if (!in_array($processorClass, self::$allowedProcessors)) {
            self::$allowedProcessors[] = $processorClass;
        }
    }

    public static function getAllowProcessors(): array
    {
        return self::$allowedProcessors;
    }
}

```

--- 파일 경로: classes/Db/WhereInterface.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

interface WhereInterface {
    public function __construct(string $coord = 'AND');
    public function case(string $field_name, string $condition, mixed $value, bool $is_quotawrap = true, bool $join_detection = true): self;
    public function begin(string $coord): self;
    public function end(): self;
    public function fetch(): array;
    public function __get($propertyName);
}

```

--- 파일 경로: classes/Db/WhereSql.php ---

```

<?php
namespace Flex\Banana\Classes\Db;

use Flex\Banana\Classes\Db\WhereInterface;
use Flex\Banana\Classes\Log;
# 데이터베이스 QUERY구문에 사용되는 WHERE문 만드는데 도움을 주는 클래스
class WhereSql implements WhereInterface
{
    public const __version = '2.0';
    private string $where = "";
    private array $where_group = [];
    private string $current_group = "";
    private string $current_coord = "";
    private array $where_groups_data = [];
    private string $scoord = 'AND'; # 전체 그룹을 마지막으로 묶을 coord

    # void
    # @fields : name+category+area 복수필드
    # @coord : [AND | OR]
    public function __construct(string $scoord = 'AND')
    {
        $this->where = "";
        $this->scoord = $scoord;
        $this->init();
    }

    # void
    # 구문어를 만든다.
    # @where_str : name+category+area 복수필드
    # @condition : [=,!=,<,>,<=,>=,IN,LIKE-R=dd%,LIKE-L=%dd,LIKE=%dd%]
    # @value : NULL | VALUE | % | Array
    public function case(string $field_name, string $condition ,mixed $value, bool $is_qutawrap=true, bool $join_detection=true) : WhereSql
    {
        $is_append = false;
        if($value == "0") $is_append = true;
        else if($value && $value != ""){
            $is_append = true;
        }

        # where 문을 그룹별로 묶기
        if($is_append)
        {
            $in_value = [];
            if (is_array($value)){ // array
                $in_value = $value;
            } else if (strpos($value, ",") !== false){
                $in_value = explode(',', $value);
            } else{
                $in_value[] = $value;
            }

            $_upper_condition = strtoupper($condition);
            if($_upper_condition == 'LIKE' || $_upper_condition == 'LIKE-R' || $_upper_condition == 'LIKE-L'){
                foreach($in_value as $n => $word)
                {
                    // $_word = preg_replace("/[#&|+|--%@=\\|\\:;\\.\\\"'^`~_\\/!\\?\\*#$<>()\\[\\]\\{\\}/i",'',$word);
                    $_word = preg_replace("/[#&|+|%@=\\|\\:;\\.\\\"'^`~_\\/!\\?\\*#$<>()\\[\\]\\{\\}/i",'',$word);

                    // append
                    $this->where_group[$this->current_group][] = match($_upper_condition) {
                        'LIKE' => sprintf("%s LIKE '%s%s'", $field_name, $_word),
                        'LIKE-R' => sprintf("%s LIKE '%s%s'", $field_name, $_word),
                        'LIKE-L' => sprintf("%s LIKE '%s%s'", $field_name, $_word),
                    };
                }
            }
            else if($_upper_condition == 'IN' || $_upper_condition == 'NOT IN'){
                if(strpos($in_value[0],'.') !== false){
                    $in_value_str = implode ( ",", $in_value );
                }else{
                    $in_value_str = ($is_qutawrap) ? "" . implode ( ",", $in_value ) . "" : implode ( ",", $in_value );
                }

                // append
                $this->where_group[$this->current_group][] = sprintf("%s %s (%s)", $field_name, $_upper_condition, $in_value_str);
            }
            else if($_upper_condition == 'JSON_CONTAINS'){
                $in_value_str = json_encode($in_value, JSON_UNESCAPED_UNICODE);
            }
        }
    }
}

```

```

        // append
        $this->where_group[$this->current_group][] = sprintf("JSON_CONTAINS(%s, '%s')", $field_name, $in_value_str);
    }
    else if($value == 'NULL'){
        // append
        $this->where_group[$this->current_group][] = sprintf("%s %s %s", $field_name, $condition, $value);
    }
    else{
        // set "a.name 형태인지| 체크"
        $__value__ = ($is_quotewrap) ? sprintf("%s", $in_value[0]) : $in_value[0];
        $d_value = sprintf("%s %s %s", $field_name, $condition, $__value__);
        if($join_detection)
        {
            $pattern = "/^([a-zA-Z0-9]|_)(\\.)?([a-zA-Z0-9]|_)/i";
            if(preg_match($pattern, $in_value[0])){
                $d_value = sprintf("%s %s %s", $field_name, $condition, $in_value[0]);
            }
        }

        $this->where_group[$this->current_group][] = $d_value;
    }
}
return $this;
}

# 상속한 부모 프라퍼티 값 포함한 가져오기
public function __get($propertyName){
    if($propertyName == 'where'){
        #아직 종료되지 않은 begin end가 있는지 체크
        if($this->current_group){
            $this->end();
        }
        $this->where = (count($this->where_groups_data) ? "(" . implode ( " " ) $this->coord) ("", $this->where_groups_data ) . ")" : "";
        $this->init();
    }
}

return $this->{$propertyName};
}

# 초기화
private function init() : void {
    $this->current_group = "";
    $this->current_coord = "";
    $this->where_group = [];
    $this->where_groups_data = [];
}

public function fetch() : array
{
    $result = $this->where_group;
    $this->init();
    return $result;
}

# where 그룹룩기 시작
public function begin(string $coord) : WhereSql
{
    $groupname = strtr(microtime(),[' '=>','0.'=>'w']);
    $this->where_group[$groupname] = [];

    # end 자동닫기
    if($this->current_group){
        $this->end();
    }

    # 현재그룹 시작
    $this->current_group = $groupname;
    $this->current_coord = $coord;
    return $this;
}

# where 그룹룩기 종료
public function end() : WhereSql{
    if(count($this->where_group[$this->current_group])){
        $wher_str = implode(sprintf(" %s ", $this->current_coord), $this->where_group[$this->current_group]);
        $this->where_groups_data[] = $wher_str;
    }
}

# 현재그룹 시작

```

```

        $this->current_group = "";
        $this->current_coord = "";
    return $this;
}

    public function __destruct(){
        $this->where = "";
        $this->current_group = "";
        $this->current_coord = "";
        $this->where_group = [];
        $this->where_groups_data = [];
    }
}

```

--- 파일 경로: classes/Dir/DirInfo.php ---

```

<?php
namespace Flex\Banana\Classes\Dir;

# purpose : 디렉토리 관련
class DirInfo
{
    public const __version = '1.1.0';
    public string $directory;
    const permission = 0707;

    public function __construct(string $dir)
    {
        $this->directory = $dir;
    }

    # 복수 폴더 만들기
    public function makesDir() : void
    {
        if(strpos($this->directory, '/') !== false)
        {
            $dir_args = explode('/', $this->directory);
            $current_dir = _ROOT_PATH_;
            if(is_array($dir_args)){
                foreach($dir_args as $folder){
                    $current_dir = $current_dir.'/'.$folder;
                    if(!isDir($current_dir)){
                        if(!mkdir($current_dir,self::permission)) throw new \Exception('e_filenotfound');
                        if(!chmod($current_dir,self::permission)) throw new \Exception('e_filenotfound');
                    }
                }
            }
        }
    }

    # 폴더 만들기
    public function makeDirectory(string $dir) : bool
    {
        $result = true;
        $directory = $this->directory.'/'.$dir;
        # compile_dirname 폴더 이전 경로 생성
        if(!isDir($directory)){
            if(!mkdir($directory,self::permission)) $result= false;
            if(!chmod($directory,self::permission)) $result= false;
            #if(!@chown($chkdirpath,getmyuid())) $result= false; break;
        }
        return $result;
    }

    # 디렉토리인지 확인
    protected function isDir(string $dir) : bool{
        if(!is_dir($dir)) return false;
        return true;
    }
}

```

--- 파일 경로: classes/Dir/DirObject.php ---

```
<?php
namespace Flex\Banana\Classes\Dir;

use Flex\Banana\Classes\Dir\DirInfo;

# 디렉토리 목록 및 디렉토리에 해당하는 파일 가져오기
class DirObject extends DirInfo
{
    public const __version = '1.2';

    public function __construct(string $dir){
        parent::__construct($dir);
    }

    #@ return array
    # 특정폴더안에 있는 모든 파일 및 폴더명을 넘겨받는다.
    # nothing = array("", "gif", "html") 포함 시키고 쉽지 않은 폴더 제외 및 파일명 제외"
    public function findFiles(string $pattern="*", Array $nothing=array()) : Array
    {
        $result = array();
        $files= glob($this->directory.DIRECTORY_SEPARATOR.$pattern);
        if(is_array($files))
        {
            foreach($files as $filename){
                if (is_file($filename)){
                    $short_filename = basename($filename);
                    $count= strrpos($short_filename, '.');
                    $file_extension= strtolower(substr($short_filename, $count+1));
                    if(!in_array($file_extension, $nothing)) $result[] = $short_filename;
                }
            }
        }
        return $result;
    }

    #@ return array
    # 특정폴더안에 있는 모든 폴더명을 넘겨받는다.
    # nothing = array("디렉토리명") 포함 시키고 쉽지 않은 폴더 제외"
    public function findFolders(Array $nothing=array()) : Array
    {
        $result = array();
        $dirs= glob($this->directory.DIRECTORY_SEPARATOR.'*', GLOB_ONLYDIR);
        if(is_array($dirs))
        {
            foreach($dirs as $dirname){
                if ($this->isDir($dirname)){
                    $short_dirname = basename($dirname);
                    if(!in_array($short_dirname, $nothing)) $result[] = $short_dirname;
                }
            }
        }
        return $result;
    }
}
```

--- 파일 경로: classes/Enum/EnumValueStorage.php ---

```
<?php
namespace Flex\Banana\Classes\Enum;

class EnumValueStorage
{
    private static array $values = [];

    public static function setValue(string $enumClass, string $key, $value): void
    {
        self::$values[$enumClass][$key] = $value;
    }

    public static function getValue(string $enumClass, string $key)
    {
    }
```

```

        return self::$values[$enumClass][$key] ?? null;
    }

    public static function getValues(string $enumClass): array
    {
        return self::$values[$enumClass] ?? [];
    }

    public static function reset(string $enumClass): void
    {
        self::$values[$enumClass] = [];
    }
}

```

--- 파일 경로: classes/File/Download.php ---

```

<?php
namespace Flex\Banana\Classes\File;

use Flex\Banana\Classes\File\FileSize;

# parnet :
# purpose : 파일다운로드
final class Download extends FileSize
{
    public const __version = '1.1';

    # 다운로드 허용 확장자
    private array $allowed_filetypes = ['pdf','xls','xlsx','doc','docx','zip','hwp','ppt','pptx','jpg','jpeg','png','gif'];
    public string $file_extension = "";
    private string $title = "";
    private array $headers = [
        'Content-type' => 'application/octet-stream',
        'Cache-control' => 'private',
        "Content-Transfer-Encoding" => "binary",
        "Pragma" => "no-cache"
    ];

    final public function __construct(string $filenamez){
        parent::__construct($filenamez);
        $this->getExtName();
    }

    # 파일 확장자 추출
    private function getExtName() : void{
        $tmpfile = basename($this->filename);
        $count = strrpos($tmpfile,'.');
        $this->file_extension = strtolower(substr($tmpfile, $count+1));
    }

    # 다운로드 허용 파일 확장자 등록
    public function setFileTypes(array $allowed_filetypes = []) : Download
    {
        if(count($allowed_filetypes)){
            $this->allowed_filetypes = $allowed_filetypes;
        }
        return $this;
    }

    public function getContents () : string
    {
        # 다운로드 허용 파일인지 체크
        if(!in_array($this->file_extension,$this->allowed_filetypes)){
            throw new \Exception( 'e_extension_not_allowed' );
        }

        return file_get_contents($this->filename);
    }

    public function __get(string $propertyName) : mixed
    {
        $result = [];
        if(property_exists(__CLASS__,$propertyName)){
            if($propertyName == 'headers'

```



```

        || $propertyName == 'allowed_filetypes'
        || $propertyName == 'title'){
            $result = $this->{$propertyName};
        }
    }
    return $result;
}

# header 값 추가 및 변경
public function __set(string $propertyName, mixed $propertyValue) : void
{
    if(property_exists(__CLASS__,$propertyName)){
        if($propertyName == 'headers'){
            if(is_array($propertyValue)){
                $this->headers = array_merge($this->headers, $propertyValue);
            }
        }
    }
}

# 다운로드파일명 설정
public function setFileName (string $title) : Download {
    $this->title = $title;
    $this->headers['Content-Disposition'] = sprintf('attachment;filename="%s"', $title);
    return $this;
}

public function download() : void
{
    # file contents
    $file_contents = $this->getContents ();

    # header
    $headers = [];
    foreach($this->headers as $hkey => $hval)
    {
        # header content
        $headerstring = sprintf("%s:%s", $hkey, $hval);

        # append
        $headers[] = $headerstring;
    }

    # output
    foreach($this->headers as $_header){
        header($_header);
    }

    # 출력
    echo $file_contents;
}
}

```

--- 파일 경로: classes/File/FileRemove.php ---

```

<?php
namespace Flex\Banana\Classes\File;

use Flex\Banana\Classes\Dir\DirObject;

# purpose : 파일삭제
final class FileRemove extends DirObject
{
    public const __version = '1.1';

    public array $list = [];

    final function __construct(string $dir) {
        parent::__construct($dir);
    }

    # 디렉토리내 파일 찾기
    public function find (string $pattern, array $nothing=['html','md','php']) : FileRemove
    {

```

```

        # 디렉토리인지 체크
        if($this->isDir($this->directory)){
            $this->list = $this->findFiles($pattern,$nothing);
        }

        return $this;
    }

    # 파일삭제
    public function remove() : void
    {
        if(count($this->list))
        {
            foreach($this->list as $filename){
                unlink($this->directory.'/'.$filename) or throw new \Exception('e_file_deletion_failed');
            }
        }
    }

    public function __get(string $propertyName){
        $result = [];
        if(property_exists($this,$propertyName)){
            $result = $this->{$propertyName};
        }
        return $result;
    }
}

```

--- 파일 경로: classes/File/FileSize.php ---

```

<?php
namespace Flex\Banana\Classes\File;

use \Exception;

# 파일 용량을 알아보기 쉽도록 변환
class FileSize
{
    public const __version = '1.0';
    protected string $filename;
    protected $filesize_bytes = 0;
    private array $convert_type = array('B', 'Kb', 'MB', 'GB', 'TB', 'PB');

    #@ void
    # 파일전체 경로
    public function __construct(string $filename=""){
        if($filename){
            if(!file_exists($filename)) throw new Exception( 'e_filenotfound' );
            if(!file_exists($filename)) throw new Exception( 'e_filenotfound' );

            $this->filename = $filename;
            $this->filesize_bytes = filesize($this->filename);
        }
    }

    # 파일사이즈 등록
    public function setBytes(int $bytes) : FileSize{
        if(!empty($bytes))
            $this->filesize_bytes = $bytes;

        return $this;
    }

    #@ 바이트 단위로
    private function bytes() : int{
        return $this->filesize_bytes;
    }

    #@ 문자 단위로
    private function size() : string{
        $result = "0";
        if(!empty($this->filesize_bytes)){
            $e = floor(log($this->filesize_bytes)/log(1024));
            $result = sprintf("%0.2f %s", ($this->filesize_bytes/pow(1024, floor($e))), $this->convert_type[$e]);
        }
    }
}

```

```

    }
    return $result;
}

public function __call(string $method, array $params = []) : mixed {
    $result = "";
    if(!method_exists($this, $method)){
        return throw new Exception( 'e_not_found_method' );
    }

    $result = match($method){
        'bytes' => $this->bytes(),
        'size' => $this->size()
    };

    return $result;
}
}

```

--- 파일 경로: classes/File/Storage.php ---

```

<?php
namespace Flex\Banana\Classes\File;

use \SplFileObject;

# 파일을 이용한 스토리지 데이터 관리
class Storage extends SplFileObject
{
    public const __version = '1.0.3';
    protected $file_name = "";
    private $open_mode;

    public function __construct ( string $file_name, string $mode ){
        $this->file_name = $file_name;
        $this->open_mode = $mode;
        parent::__construct($this->file_name, $this->open_mode);
        if (parent::isFile()) {
            $this->file_name = parent::getRealPath();
        }
    }

    #@ 파일쓰기
    public function write(string $context) : int|bool{
        $written=0;
        if(parent::isWritable()){
            $written = parent::fwrite($context);
        }
        return $written;
    }

    #@ 파일 읽기
    public function read() : array{
        $contents = [];
        if(parent::isFile() && parent::isReadable())
        {
            while (!parent::eof()) {
                $contents[] =parent::fgets();
            }
        }
        return $contents;
    }

    #@ 파일쓰기
    public function put(string $context) : int|bool{
        $written=0;
        if(parent::isWritable()){
            if(function_exists('file_put_contents')){
                $written = file_put_contents($this->file_name, $context);
            }
        }
        return $written;
    }

    #@ 파일 읽기

```

```

public function get() : string|false{
    $contents = "";
    if(parent::isFile() && parent::isReadable())
    {
        if(function_exists('file_get_contents')){
            $contents = file_get_contents($this->file_name);
        }
    }
    return $contents;
}

#@ 쓰기 CSV
/*
 * $list = array (
    array('aaa', 'bbb', 'ccc', 'dddd'),
    array('123', '456', '789')
);
*/
public function write_csv(array $args) : void {
    if(is_array($args))
    {
        if(PHP_VERSION_ID >= 50400){
            foreach ($args as $fields) {
                parent::fputcsv($fields);
            }
        }else{
            $fp = fopen($this->file_name, $this->open_mode);
            if(is_resource($fp)){
                foreach ($args as $datav) {
                    fputcsv($fp, $datav);
                }
            }
            fclose($fp);
        }
    }
}

#@ 읽기 CSV
public function read_csv() : array{
    $args = [];
    while (!parent::eof()) {
        $args[] = array_filter(parent::fgetcsv());
    }
    return array_filter($args);
}
}

```

--- 파일 경로: classes/File/Upload.php ---

```

<?php
namespace Flex\Banana\Classes\File;

use Flex\Banana\Classes\Dir\DirInfo;
use Flex\Banana\Classes\Image\ImageExif;
use Flex\Banana\Classes\Cipher\CipherGeneric;
use Flex\Banana\Classes\Cipher\HashEncoder;
use Flex\Banana\Classes\Log;
use \Exception;

class Upload extends DirInfo
{
    public const __version = '2.2.3';
    public string $file_extension = "";
    public string $mimeType;
    public $process;
    public string $savefilename = "";

    private array $error_msg = [
        UPLOAD_ERR_INI_SIZE => 'e_upload_max_filesize',
        UPLOAD_ERR_FORM_SIZE => 'e_upload_max_filesize',
        UPLOAD_ERR_PARTIAL => 'e_partially_uploaded',
        UPLOAD_ERR_NO_FILE => 'e_no_was_uploaded',
        UPLOAD_ERR_NO_TMP_DIR => 'e_miss_temp_folder',
        UPLOAD_ERR_CANT_WRITE => 'e_failed_write_disk',
    ];
}

```

```

        UPLOAD_ERR_EXTENSION => 'e_upload_stopped',
    ];

    # 1
    public function __construct(string $directory)
    {
        parent::__construct($directory);
    }

    # 2 첨부파일
    public function process(string $process_id, array $_files) : Upload
    {
        # 값이 정상적인지 체크
        if (!isset($_files[$process_id])) {
            self::exceptionsErrorHandler(UPLOAD_ERR_NO_FILE);
        }

        $this->process = $_files[$process_id];
        $filename = method_exists($this->process, 'getClientFilename') ? $this->process->getClientFilename() : $this->process['name'];
        $mimeType = method_exists($this->process, 'getClientMediaType') ? $this->process->getClientMediaType() : $this->process['type'];
        $size = method_exists($this->process, 'getSize') ? $this->process->getSize() : $this->process['size'];
        $error = method_exists($this->process, 'getError') ? $this->process->getError() : $this->process['error'];

        Log::d('-----< Upload >-----');
        Log::d('filename', $filename);
        Log::d('mimeType', $mimeType);
        Log::d('size', $size);
        Log::d('error', $error);
        Log::d('-----');

        # 기초에러
        if ($error !== UPLOAD_ERR_OK) {
            self::exceptionsErrorHandler($error);
        }

        return $this;
    }

    # 3 업로드 허용된 파일 인치 체크
    public function filterExtension(array $allowe_extension=['jpg','jpeg','png','gif']) : Upload
    {
        $this->getExtName();
        if (!in_array($this->file_extension, $allowe_extension)) {
            self::exceptionsErrorHandler(UPLOAD_ERR_EXTENSION);
        }
        return $this;
    }

    # 4 파일크기 체크 8(M),12(M),100(M)
    public function filterSize(int $size) : Upload
    {
        $maxsize = (int)(1024 * 1024 * $size);
        $filesize = $this->process->getSize() ?? $this->process['size'];
        if ($filesize >= $maxsize) {
            self::exceptionsErrorHandler(UPLOAD_ERR_INI_SIZE);
        }
        return $this;
    }

    # 5 업로드할 디렉토리 체크 및 만들기
    public function makeDirs() : Upload
    {
        try {
            parent::makeDir();
        } catch (Exception $e) {
            throw new Exception($e->getMessage());
        }
        return $this;
    }

    # 6 업로드 파일 복사하기
    public function save(): Upload
    {
        #저장할파일명
        $tempfilename = str_replace(['.', ':'], ['_', ':'], microtime());
        $this->savefilename = sprintf("%s.%s", (new CipherGeneric(new HashEncoder($tempfilename)))->hash(), $this->file_extension);
        $fullname = sprintf("%s/%s", $this->directory, $this->savefilename);
        # 파일 저장
    }

```

```

try {
    if ($this->process->getStream() != null) {
        # BufferedBody 객체에서 내용을 가져옴
        $bodyContent = (string)$this->process->getStream();
        if (file_put_contents($fullname, $bodyContent) === false) {
            throw new Exception('Failed to write file to disk.');
```

```

        }
    } else {
        # 업로드된 파일인지 체크
        if (!$this->is_upload_files()) {
            self::exceptionsErrorHandler(UPLOAD_ERR_NO_FILE);
        }
    }

```

```

    # 파일 이동 저장
    if (!move_uploaded_file($this->process['tmp_name'], $fullname)) {
        self::exceptionsErrorHandler(UPLOAD_ERR_CANT_WRITE);
    }
}

```

```

} catch (Exception $e) {
    Log::e(__LINE__, $e->getMessage());
    self::exceptionsErrorHandler(UPLOAD_ERR_CANT_WRITE);
}

```

```

return $this;
}

```

```

# 7 orientation
public function filterOrientation() : Upload
{

```

```

    # jpeg, jpg 인자 체크
    if (preg_match('/(jpeg|jpg)/', $this->file_extension)) {
        $fullname = sprintf("%s/%s", $this->directory, $this->savefilename);
        $ifdo = (new ImageExif($fullname))->getIfdo();
        if (isset($ifdo['Orientation']) && !empty($ifdo['Orientation'])) {
            $im = imagecreatefromjpeg($fullname);
            switch ($ifdo['Orientation']) {
                case 8:
                    $rotate = imagerotate($im, 90, 0);
                    imagejpeg($rotate, $fullname);
                    break;
                case 3:
                    $rotate = imagerotate($im, 180, 0);
                    imagejpeg($rotate, $fullname);
                    break;
                case 6:
                    $rotate = imagerotate($im, -90, 0);
                    imagejpeg($rotate, $fullname);
                    break;
            }
        }
    }
}

```

```

return $this;
}

```

```

# end fetch
public function fetch() : array
{

```

```

    return [
        'filesize' => $this->process->getSize() ?? $this->process['size'],
        'mimeType' => $this->mimeType,
        'filename' => self::cleansEtcWords(),
        'sfilename' => $this->savefilename
    ];
}

```

```

# 파일 확장자 추출
private function getExtName() : void
{

```

```

    $filename = $this->process->getClientFilename() ?? $this->process['name'];
    $count = strrpos($filename, '.');
    $this->file_extension = strtolower(substr($filename, $count + 1));
    $this->mimeType = (preg_match('/(gif|jpeg|jpg|png)/', $this->file_extension)) ? 'image/' . $this->file_extension : 'application/' . $this->file_extension;
}

```

```

# 업로드된 파일인지 체크

```

```

private function is_upload_files(): bool {
    if (!isset($this->process['tmp_name']) || !is_uploaded_file($this->process['tmp_name'])) {
        return false;
    }
}

```

```

    }
    return true;
}

# 첨부 실패일명 특수문자 제거
private function cleansEtcWords() : string {
    $filename = preg_replace("/[ #&+!-%@=\|\\:;'\\""'^~\|\\!?!$#<>()\[\]{}]/", '_', $this->process->getClientFilename() ?? $this->process['name']);
    $filename = preg_replace('/s+/', '_', $filename); // 연속된 공백을 하나의 문자로 변경
    return $filename;
}

# 예러 발생시킴
private function exceptionsErrorHandler(int $error_no) : void {
    if (array_key_exists($error_no, $this->error_msg)) {
        throw new Exception($this->error_msg[$error_no]);
    }
}
}
}

```

--- 파일 경로: classes/Ftp/Ftp.php ---

```

<?php
namespace Flex\Banana\Classes\Ftp;

use Flex\Banana\Classes\Ftp\FtpObject;

final class Ftp extends FtpObject
{
    public const __version = '1.1';
    private $ascii_type = [
        'txt', 'htm', 'html', 'phtml', 'php', 'php3', 'php4',
        'inc', 'ini', 'asp', 'aspx', 'jsp', 'css', 'js'
    ];

    public function __construct(string $host, string $user, string $passwd, int $port, bool $is_ssl=false, int $time=60){
        parent::__construct($host, $port, $is_ssl, $time);
        $this->ftp_login($user, $passwd);
    }

    # 파일 내용 읽어 오기
    public function open_file_read(string $tmpfile, string $remote_file) : string
    {
        if(!$this->ftp_get($tmpfile, $remote_file, $this->chk_open_mode($remote_file)))
            return false;

        $fp=fopen($tmpfile, 'r');
        $contents = fread($fp, filesize($tmpfile));
        fclose($fp);

        return $contents;
    }

    public function open_file_write(string $tmpfile, string $remote_file, string $contents) : bool
    {
        if(!$this->isExists($tmpfile)){
            return false;
        }

        if(empty($contents))
            return false;

        $fp = fopen($tmpfile, 'w');
        fwrite($fp, $contents);
        fclose($fp);
        if(!$this->ftp_put($remote_file, $tmpfile, $this->chk_open_mode($remote_file)))
            return false;

        @unlink($tmpfile);
        return true;
    }

    # 파일 삭제
    public function delete_file(string $dir, string $del_filename) : void{
        $files = $this->ftp_nlist($dir);
    }
}

```

```

        if(is_array($files)){
            foreach ($files as $file){
                $realname = basename($file);
                if($del_filename == $realname){
                    $this->ftp_delete($file);
                    break;
                }
            }
        }
    }

    # 로컬 파일인지 체크
    private function isExists(string $filename) : bool{
        if(!file_exists($filename)) return false;
        return true;
    }

    #@ return int
    private function chk_open_mode(string $filename) : int
    {
        $extention = strtolower($this->getExtension($filename));

        if(!in_array($extention, $this->ascii_type)) return FTP_ASCII;
        else return FTP_BINARY;
    }

    # 파일 확장자 추출
    private function getExtension(string $filename) : string{
        $tmpfile = basename($filename);
        $count= strrpos($tmpfile, '.');
        $extention= strtolower(substr($tmpfile, $count+1));
        return $extention;
    }
}

```

--- 파일 경로: classes/Ftp/FtpObject.php ---

```

<?php
namespace Flex\Banana\Classes\Ftp;

class FtpObject
{
    public const __version = '1.1';
    public $conn;

    public function __construct(string $ftp_url, int $port, bool $is_ssl, int $time){
        if($is_ssl){
            if(false === ($this->conn = @ftp_ssl_connect($ftp_url, $port, $time))){
                throw new \Exception("ftp ssl connect fail!!!!");
            }
        } else {
            if (false === ($this->conn = @ftp_connect($ftp_url, $port, $time))) {
                throw new \Exception("ftp connect fail!!!!");
            }
        }
    }

    public function __call(string $func,array $params){
        if(strpos($func,'ftp_') !== false && function_exists($func)){
            array_unshift($params,$this->conn);
            return call_user_func_array($func,$params);
        }
    }

    public function __destruct(){
        ftp_close($this->conn);
    }
}

```

--- 파일 경로: classes/Html/XssChars.php ---



```

<?php
namespace Flex\Banana\Classes\Html;

# purpose : xss 방지 및
class XssChars
{
    public const __version = '1.9.1';
    private string $description;
    private array $allow_tags = [];

    public function __construct(string $description){
        $this->description = $description;
    }

    #@ void
    # 허용 태그 설정
    public function setAllowTags(string $value) : void{
        if(is_array($value)) $this->allow_tags = array_merge($this->allow_tags,$value);
        else $this->allow_tags[] = $value;
    }

    # strip_tags
    public function cleanTags() : string{
        return strip_tags(htmlspecialchars_decode($this->description),implode(", ", $this->allow_tags));
    }

    #@ return String
    # Xss 태그 처리
    public function cleanXssTags() : string
    {
        $xss_tags = array(
            '@<script[^\>]*?>.*?</script>@si',
            '@<style[^\>]*?>.*?</style>@siU',
            '@<iframe[^\>]*?>.*?</iframe>@si',
            '@<meta[^\>]*?>.*?>@si',
            '@<form[^\>]*?>.*?>@si',
            '@|script[^\>]*?>.*?|/script>@si', // [\xC0][\xBC]script>[code][\xC0][\xBC]/script>
            '/:.*?expression\(.*?\)/si',
            '/:.*?binding:(.*?)url\(.*?\)/si',
            '/javascript:[^\\"']*\/si',
            '/vbscript:[^\\"']*\/si',
            '/livescript:[^\\"']*\/si',
            '@<![\s\S]*?--[ \t\r\n]*>@// multi-line comments including CDATA
        );

        $event_tags = array(
            'dynsrc','datasrc','frameset','ilayer','layer','applet',
            'onabort','onactivate','onafterprint','onsubmit','onunload',
            'onafterupdate','onbeforeactivate','onbeforecopy','onbeforecut',
            'onbeforedeactivate','onbeforeeditfocus','onbeforepaste','onbeforeprint',
            'onbeforeunload','onbeforeupdate','onblur','onbounce','oncellchange',
            'onchange','onclick','oncontextmenu','oncontrolselect','oncopy','oncut',
            'ondataavaible','ondatachanged','ondatacomplete','ondblclick',
            'ondeactivate','ondrag','ondragdrop','ondragend','ondragenter',
            'ondragleave','ondragover','ondragstart','ondrop','onerror','onerrorupdate',
            'onfilterupdate','onfinish','onfocus','onfocusin','onfocusout','onhelp',
            'onkeydown','onkeypress','onkeyup','onlayoutcomplete','onload','onlosecapture',
            'onmousedown','onmouseenter','onmouseleave','onmousemove','onmouseout',
            'onmouseover','onmouseup','onmousewheel','onmove','onmoveend','onmovestart',
            'onpaste','onpropertychange','onreadystatechange','onreset','onresize',
            'onresizeend','onresizestart','onrowexit','onrowsdelete','onrowsinserted',
            'onscroll','onselect','onselectionchange','onselectstart','onstart','onstop'
        );

        // 허용 태그 확인
        if(is_array($this->allow_tags)){
            $this->allow_tags = explode(', ',strtr(implode(', ', $this->allow_tags),['<=>','>=>']));
            $tmp_eventtag= str_replace($this->allow_tags,implode('|', $event_tags));
            $event_tags = explode('|', $tmp_eventtag);
        }

        return preg_replace($xss_tags, "", str_ireplace($event_tags, '_badtags', $this->description));
    }

    # 자동 링크 걸기
    public function setAutoLink() : string
    {
        $homepage_pattern = "(/([^\\""]|=))(mms|market|http|https|HTTP|ftp|FTP|telnet|TELNET):\\:\\V( / [^\\""]|<\"")+/";
    }
}

```

```

$this->description = preg_replace($homepage_pattern, "\\1<a href='\\2://\\3' target='_blank'>\\2://\\3</a>'", '$this->description');

// 메일 치환
$email_pattern = "/([ \\n]+)([a-z0-9\\_\\-\\.]+)@([a-z0-9\\_\\-\\.]+)/";
return preg_replace($email_pattern, "\\1<a href='mailto:\\2@\\3>\\2@\\3'</a>", " ".$this->description);
}

# url 링크에 http가 있는지 확인후 붙여서 리턴해 주기
public function setHttpUrl() : string
{
    if($this->description)
        $this->description = trim($this->description);

    if (strpos($this->description, 'http') === false) {
        $this->description = 'http://'.$this->description;
    }
    return $this->description;
}

# code html highlight
public function getXHtmlHighlight() : string
{
    $str = highlight_string($this->description, true);
    $str = preg_replace('#<font color="([\\^\\"]*)">([\\^\\"]*)</font>#', '<span style="color: \\1">\\2</span>', $str);
    return preg_replace('#<font color="([\\^\\"]*)">([\\^\\"]*)</font>#U', '<span style="color: \\1">\\2</span>', $str);
}

# 여러형태의 모양
public function getContext(string $mode='XSS') : string
{
    $this->description = stripslashes($this->description);
    switch(strtoupper($mode)){
        case 'TEXT':
            $this->description = strtr($this->description, ["&nbsp;"=>' ']);
            $this->description = strtr($this->description, ["\\r\\n"=>"\\n"]);
            $this->description = $this->setAutoLink();
            $this->allow_tags = ['<a>'];
            $this->description = $this->cleanTags();
            break;
        case 'XSS':
            $this->description = strtr($this->description, ["\\r\\n"=>"\\n"]);
            $this->description = strtr($this->description, ["\\n"=>"<br>"]);
            $this->description = strtr($this->description, ["<br/>"=>"<br>"]);
            $this->description = $this->setAutoLink();
            $this->description = $this->cleanXssTags();
            break;
        case 'HTML':
            $this->description = strtr($this->description, ["\\r\\n"=>"\\n"]);
            $this->description = strtr($this->description, ["\\n"=>"<br>"]);
            $this->description = $this->setAutoLink();
            $this->description = htmlspecialchars($this->description);
            break;
        case 'XHTML' :
            $this->description = $this->getXHtmlHighlight();
            $this->description = $this->setAutoLink();
            break;
    }
    return $this->description;
}

public function __call(string $query, array $args=[]) : mixed
{
    $_query = strtolower($query);

    # 배열을 dictionary Object
    if($_query == 'gettext'){
        return $this->getContext('TEXT');
    }else if($_query == 'getxss'){
        return $this->getContext('XSS');
    }else if($_query == 'gethtml'){
        return $this->getContext('HTML');
    }else if($_query == 'getxhtml'){
        return $this->getContext('XHTML');
    }else {
        return null;
    }
}
}

```

--- 파일 경로: classes/Http/HttpRequest.php ---

```
<?php
namespace Flex\Banana\Classes\Http;
use Flex\Banana\Classes\Log;

class HttpRequest {
    public const __version = '1.3.0';
    private $urls = [];
    private $mch;

    public function __construct(array $argv = []) {
        if (!is_array($argv)) {
            throw new \Exception(__CLASS__.' :: '.__LINE__.' is not array');
        }
        $this->urls = $argv;
        $this->mch = curl_multi_init();
    }

    public function set(string $url, string $params, array $headers = []): HttpRequest {
        if (trim($url)) {
            $this->urls[] = [
                "url" => $url,
                "params" => $params,
                "headers" => $headers
            ];
        }
        return $this;
    }

    public function get(callable $callback = null) {
        $response = $this->execute('GET');
        if ($callback !== null && is_callable($callback)) {
            $callback($response);
        }
        return $response;
    }

    public function post(callable $callback = null) {
        $response = $this->execute('POST');
        if ($callback !== null && is_callable($callback)) {
            $callback($response);
        }
        return $response;
    }

    public function put(callable $callback = null) {
        $response = $this->execute('PUT');
        if ($callback !== null && is_callable($callback)) {
            $callback($response);
        }
        return $response;
    }

    public function delete(callable $callback = null) {
        $response = $this->execute('DELETE');
        if ($callback !== null && is_callable($callback)) {
            $callback($response);
        }
        return $response;
    }

    public function patch(callable $callback = null) {
        $response = $this->execute('PATCH');
        if ($callback !== null && is_callable($callback)) {
            $callback($response);
        }
        return $response;
    }

    private function execute(string $method)
    {
        $response = [];
    }
}
```

```

foreach ($this->urls as $idx => $url)
{
    $ch[$idx] = curl_init($url['url']);

    $headers = $url['headers'] ?? [];
    $params = $url['params'];

    curl_setopt($ch[$idx], CURLOPT_CUSTOMREQUEST, $method);
    curl_setopt($ch[$idx], CURLOPT_SSL_VERIFYHOST, false);
    curl_setopt($ch[$idx], CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($ch[$idx], CURLOPT_RETURNTRANSFER, true);

    $contentType = $this->getContentType($headers);

    if ($method !== 'GET') {
        $postFields = $this->preparePostFields($params, $contentType);
        curl_setopt($ch[$idx], CURLOPT_POSTFIELDS, $postFields);
    } else if ($params) {
        $url['url'] .= (strpos($url['url'], '?') === false ? '?' : '&') . $params;
        curl_setopt($ch[$idx], CURLOPT_URL, $url['url']);
    }

    if (!$this->hasContentTypeHeader($headers) && $contentType) {
        $headers[] = "Content-Type: $contentType";
    }

    curl_setopt($ch[$idx], CURLOPT_HTTPHEADER, $headers);
    curl_multi_add_handle($this->mch, $ch[$idx]);
}

do {
    curl_multi_exec($this->mch, $running);
    curl_multi_select($this->mch);
} while ($running > 0);

foreach (array_keys($ch) as $index) {
    $statusCode = curl_getinfo($ch[$index], CURLINFO_HTTP_CODE);
    $body = curl_multi_getcontent($ch[$index]);

    $contentTypeHeader = curl_getinfo($ch[$index], CURLINFO_CONTENT_TYPE);
    $isJsonResponse = strpos($contentTypeHeader, 'application/json') !== false;

    // 이미 배열인지 확인
    if (is_array($body)) {
        $decodedBody = $body;
    } else if (is_string($body) && !empty($body)) {
        if ($isJsonResponse) {
            // JSON 디코딩 시도
            $decodedBody = $body; // 기본값으로 원본 설정
            if (version_compare(PHP_VERSION, '7.3.0', '>=')) {
                try {
                    $tempDecoded = json_decode($body, true, 512, JSON_THROW_ON_ERROR);
                    if (is_array($tempDecoded)) {
                        $decodedBody = $tempDecoded;
                    }
                } catch (\JsonException $e) {
                    Log::e($index, 'JSON decode error', $e->getMessage());
                    throw new \Exception(json_encode(['message' => $e->getMessage(), 'body' => $body]));
                }
            }
        } else {
            $tempDecoded = json_decode($body, true);
            if (json_last_error() === JSON_ERROR_NONE && is_array($tempDecoded)) {
                $decodedBody = $tempDecoded;
            } else {
                Log::e($index, 'JSON decode error', json_last_error_msg());
                throw new \Exception(json_encode(['message' => $e->getMessage(), 'body' => json_last_error_msg()]));
            }
        }
    } else {
        $decodedBody = $body;
    }
} else {
    $decodedBody = $body;
}

$response[$index] = [
    'code' => $statusCode,
    'body' => $decodedBody,
    'url' => curl_getinfo($ch[$index], CURLINFO_EFFECTIVE_URL)
]

```

```

    ];
    curl_multi_remove_handle($this->mch, $ch[$index]);
}

$this->urls = [];
return $response;
}

private function getContentType($headers): ?string {
    foreach ($headers as $header) {
        if (stripos($header, 'Content-Type:') === 0) {
            list($contentType) = explode(':', $header, 2);
            return trim($contentType);
        }
    }
    return null;
}

private function hasContentTypeHeader($headers): bool {
    foreach ($headers as $header) {
        if (stripos($header, 'Content-Type:') === 0) {
            return true;
        }
    }
    return false;
}

private function preparePostFields($params, $contentType) {
    switch ($contentType) {
        case 'application/json':
            return $params; // JSON string as is
        case 'application/x-www-form-urlencoded':
            return $params; // URL encoded string as is
        case 'multipart/form-data':
            parse_str($params, $parsedParams);
            $postFields = [];
            foreach ($parsedParams as $key => $value) {
                if (is_string($value) && strpos($value, '@') === 0 && file_exists(substr($value, 1))) {
                    $filePath = substr($value, 1);
                    $finfo = finfo_open(FILEINFO_MIME_TYPE);
                    $mimeType = finfo_file($finfo, $filePath);
                    finfo_close($finfo);
                    $fileName = basename($filePath);
                    $postFields[$key] = new \CURLFile($filePath, $mimeType, $fileName);
                } else {
                    $postFields[$key] = $value;
                }
            }
            return $postFields;
        default:
            return $params;
    }
}

public function __destruct() {
    curl_multi_close($this->mch);
}
}

```

--- 파일 경로: classes/Http/HttpResponse.php ---

```

<?php
namespace Flex\Banana\Classes\Http;

final class HttpResponse {
    public const __version = '0.5';

    public function __construct(
        private int $code,
        private array $headers,
        private mixed $message
    ){}

    public function __invoke(): mixed

```

```

{
    // HEADERS
    foreach ($this->headers as $header_key => $header_value) {
        header(sprintf("%s:%s", $header_key, $header_value));
    }

    // HTTP 상태 코드 설정
    http_response_code($this->code);

    return $this->message;
}

public function __toString(): string
{
    return (string)$this();
}
}

```

--- 파일 경로: classes/Http/HttpUrlFilter.php ---

```

<?php
namespace Flex\Banana\Classes\Http;

class HttpUrlFilter
{
    public const __version = '0.5';
    public function __construct(
        private string $url
    )
    {}

    # http|https 가 있는지 확인 후 glue 붙이기
    public function httpPrefix(string $glue='http') : HttpUrlFilter
    {
        if (!preg_match("~^(?:[ht]tps?://~i", $this->url)) {
            $this->url = $glue . '://' . $this->url;
        }
        return $this;
    }

    public function wwwPrefix() : HttpUrlFilter
    {
        // http(s)://가 있지만 www가 없는 경우
        if (preg_match("/^https?:\/\/(?:!www\.)i", $this->url)) {
            $this->url = preg_replace("/^https?:\/\/i", "$0www.", $this->url);
        }
        // https(s)가 없어도 http(s)://만 있고 www가 없는 경우
        else if (preg_match("/^:\/(?:!www\.)i", $this->url)) {
            $this->url = preg_replace("/^:\/i", "$0www.", $this->url);
        }
        else if (!preg_match("/^www\./i", $this->url)) {
            $this->url = "www." . $this->url;
        }
    }

    return $this;
}

# 가져오기
public function __get(string $propertyName) : mixed{
    return $this->url;
}
}

```

--- 파일 경로: classes/Image/ImageExif.php ---

```

<?php
namespace Flex\Banana\Classes\Image;

# purpose : 카메라 촬영 정보
class ImageExif

```

```

{
    public const __version = '0.9';
    private $exifargs = [];

    # computed : 넓이, 높이, 조리개, 촬영거리, CCD
    # ifdo : 카메라 정보
    # exif : 노출모드, 조리개값, 플래시사용여부, 화이트밸런스, 줌, ISO감도, 초점거리, 측광모드, 오리지널촬영시간
    # makenote : 펌웨어버전, 사용렌즈
    private $setkey_args = [
        'file' => ['FileName', 'FileSize', 'FileDateTime', 'MimeType'],
        'computed' => ['Width', 'Height', 'ApertureFNumber', 'FocusDistance', 'CCDWidth'],
        'ifdo' => ['Make', 'Model', 'Software', 'Orientation'],
        'exif' => ['ExposureTime', 'FNumber', 'Flash', 'WhiteBalance', 'DigitalZoomRatio', 'ISOSpeedRatings', 'FocalLength', 'MeteringMode', 'DateTimeOriginal'],
        'makenote' => ['FirmwareVersion', 'UndefinedTag:0x0095']
    ];

    # 사진 전체 경로
    public function __construct(string $picture){
        # 로컬 파일인지 체크
        if(!file_exists($picture))
            throw new \Exception(__CLASS__.' :: '.__LINE__.' ' .strval($picture).' not found');

        # 함수 enable 체크
        if(function_exists('exif_read_data')){
            $this->exifargs = @exif_read_data($picture, 0, true);
            if($this->exifargs === false)
                throw new \Exception(__CLASS__.' :: '.__LINE__.' exif_read_data functions are not available');
        }
    }

    # FILE
    public function getFile() : array
    {
        $result = [];
        if(isset($this->exifargs['FILE'])){
            $result = $this->exifargs['FILE'];
        }
        return $result;
    }

    # COMPUTED
    public function getComputed() : array
    {
        $result = [];
        if(isset($this->exifargs['COMPUTED'])){
            $args =& $this->exifargs['COMPUTED'];
            foreach($args as $k => $v){
                switch($k){
                    case 'FocusDistance':
                        $result[$k] = $v;
                        if(strpos($v, '/') !== false){
                            $tmpdistance = explode('/', $v);
                            $result[$k] = ($tmpdistance[0]/$tmpdistance[1]).'mm';
                        }
                        break;
                    case 'CCDWidth':
                        $result[$k] = (!empty($v)) ? substr($v, 0, 5).' mm' : '';
                        break;
                    default :
                        $result[$k] = $v;
                }
            }
        }
        return $result;
    }

    # IFDO
    public function getIfdo() : array
    {
        $result = [];
        if(isset($this->exifargs['IFD0'])){
            $args =& $this->exifargs['IFD0'];
            foreach($args as $k => $v){
                switch($k){
                    case 'Make':
                        $result[$k] = str_replace('CORPORATION', '', $v);
                        break;
                    default:
                        $result[$k] = $v;
                }
            }
        }
    }
}

```

```

    }
  }
}
return $result;
}

# EXIF
public function getExif() : array
{
    $result = [];
    if(isset($this->exifargs['EXIF'])){
        $args =& $this->exifargs['EXIF'];
        foreach($args as $k => $v){
            switch($k){
                case 'Flash': $result[$k] = ($v==1) ? 'ON' : 'OFF'; break;
                case 'ExposureTime':
                    $result[$k] = $v;
                    if(strpos($v, '/') !== false){
                        $tmpexpo = explode('/', $v);
                        $result[$k] = ($tmpexpo[0]/$tmpexpo[0]).'/'.(($tmpexpo[1]/$tmpexpo[0]).'s';
                    }
                    break;
                case 'FocalLength':
                    $result[$k] = $v;
                    if(strpos($v, '/') !== false){
                        $tmpfocal = explode('/', $v);
                        $result[$k] = ($tmpfocal[0]/$tmpfocal[1]).'mm';
                    }
                    break;
                case 'MakerNote':
                    break;
                default: $result[$k] = $v;
            }
        }
    }
}
return $result;
}

# GPS
public function getGPS() : array
{
    $result = [];
    if(isset($this->exifargs['GPS'])){
        $result = $this->exifargs['GPS'];
    }
}
return $result;
}

# MAKENOTE
public function getMakenote() : array
{
    $result = [];
    if(isset($this->exifargs['MAKENOTE'])){
        $args =& $this->exifargs['MAKENOTE'];
        foreach($this->setkey_args['makenote'] as $k => $v){
            $result[$k] = $v;
        }
    }
}
return $result;
}

# 한번에 추출하기
public function fetch() : array
{
    $args = [];
    if(count($this->exifargs)>0){
        foreach($this->setkey_args as $k => $v){
            $methodName = 'get'.ucwords($k);
            $args += call_user_func_array(array(&$this, $methodName), array());
        }
    }
}
return $args;
}
}

```



--- 파일 경로: classes/Image/ImageGDS.php ---

```
<?php
namespace Flex\Banana\Classes\Image;

# purpose : 이미지 효과주기
class ImageGDS
{
    public const __version = '1.3';
    public $filename;

    public $im;
    private $quality = 100;
    private $bgcolor = 0x7ffffff;
    private $fontsrc, $fontangle=0, $color = [0.0,0], $fontsize = 20, $x=5, $y=5;

    # 시작
    public function __construct(string|null $filename=null){
        if($filename && !file_exists($filename)) {
            throw new \Exception(__METHOD__.' '.$filename,__LINE__);
        }

        if(!is_null($filename)){
            $this->filename = $filename;
        }
    }

    # void 퀄리티 설정
    public function setCompressionQuality(int $quality) : void
    {
        $this->quality = $quality;
    }

    # 칼라 채우기
    public function setFilledrectangle(mixed $image,int $x1, int $y1, int $x2, int $y2, string $color) : mixed
    {
        if(false === ($im = imagefilledrectangle($image,$x1,$y1,$x2,$y2,$color))) return false;
        return $im;
    }

    # 칼라 채우기 RGB
    public function setColorallocate(mixed $image, int $r, int $g, int $b) : mixed
    {
        if(0 > ($im = imagecolorallocate($image,$r,$g,$b))) return false;
        return $im;
    }

    # alpha
    public function setAlphaablending(mixed $image,bool $boolean=false) : void
    {
        imagealphablending($image, $boolean);
    }

    # alpha
    public function setSavealpha(mixed $image,bool $boolean=false) : void
    {
        imagesavealpha($image, $boolean);
    }

    public function setFttext(mixed $image, int $fontcolor, string $text){
        imagefttext($image,$this->fontsize,$this->fontangle,$this->x,$this->y,$fontcolor,$this->fontsrc,$text);
    }

    # 폰트 파일 경로 지정
    public function setFont(string $fontsrc) : void { $this->fontsrc = $fontsrc; }

    # 칼라 지정
    public function setFontColor(array $color) : void { $this->color = $color; }

    # 폰트 사이즈
    public function setFontSize(string $pixel): void { $this->fontsize = $pixel; }

    # 배경칼라
    public function setBgColor(string $bgcolor) : void { $this->bgcolor = $bgcolor; }

    # 폰트 앵글
    public function setFontAngle(int $angle) : void { $this->fontangle = $angle; }
    # x:y 축
```

```

public function setXY(int $x, int $y) : void { $this->x = $x; $this->y = $y; }

# 텍스트 이미지 만들기
public function writeTextImage(int $width, int $height, string $text) : void{
    $this->im = $this->createTrueImage($width,$height);
    $this->setAlphaBlending($this->im);
    $this->setFilledrectangle($this->im,0,0,$width,$height,$this->bgcolor);

    $fontcolor = $this->setColorallocate($this->im,$this->color[0],$this->color[1],$this->color[2]);
    $this->setFttext($this->im,$fontcolor,$text);
    $this->setSavealpha($this->im,true);
}

public function setAntialias(mixed $image,bool $boolean=false): void {
    imageantialias($image,$boolean);
}

public function setTTFTText(mixed $image,float $size,int $x, int $y,int $color,string $text){
    imagettftext($image,$size,$this->fontangle,$x,$y,$color,$this->fontsrc,$text);
}

# 그림자 입체 텍스트 쓰기
public function writeShadowText(int $width, int $height,string $text,array $bgRGB=[255,255,255], array $mdRGB=[128,128,128], array $frontRGB=[0,0,0]) :void
{
    $this->im = $this->createTrueImage($width,$height);

    $bg = $this->setColorallocate($this->im,$bgRGB[0],$bgRGB[1],$bgRGB[2]);
    $middle = $this->setColorallocate($this->im, $mdRGB[0],$mdRGB[1],$mdRGB[2]);
    $front = $this->setColorallocate($this->im, $frontRGB[0],$frontRGB[1],$frontRGB[2]);
    $this->setFilledrectangle($this->im,0,0,$width-1,$height-1,$bg);

    // Add some shadow to the text
    $this->setTTFTText($this->im,$this->fontsize,$this->x,$this->y,$middle,$text);

    // Add the text
    $this->setTTFTText($this->im,$this->fontsize, ($this->x - 1), ($this->y - 1),$front,$text);
}

# 이미지 위에 텍스트 쓰기
public function combineImageText(int $width, int $height, string $text, string|null $filename=null) : void{
    $this->im = $this->createTrueImage($width,$height);
    $this->setAntialias($this->im,true);
    $fontcolor = $this->setColorallocate($this->im,$this->color[0],$this->color[1],$this->color[2]);

    $filename = ($filename) ? $filename : $this->filename;
    if(!$filename) throw new \Exception(__CLASS__.'.'.__METHOD__.'.'.__LINE__);

    $image = $this->readImage($filename);
    $this->copy($this->im,$image,0,0,0,0,$width,$height);
    $this->setTTFTText($this->im,$this->fontsize,$this->x,$this->y,$fontcolor,$text);
}

# margin_x : 가로 여백, margin_y : 세로 여백
# RB : 오른쪽 아래 기준, LB : 왼쪽 아래 기준, LT : 왼쪽 위 기준, RT : 오른쪽 위 기준
public function filterWatermarks(string $marksfilename,int $margin_x=10,int $margin_y=10, string $position='RB'): void
{
    if(!file_exists($marksfilename))
        throw new \Exception(__CLASS__.'.'.__METHOD__.'.'.$marksfilename);

    $this->im = $this->readImage($this->filename);
    $this->setAntialias($this->im,true);
    $image = $this->readImage($marksfilename);

    $width = imagesx($image);
    $height = imagesy($image);

    # switch
    $im_x = $margin_x;
    $im_y = $margin_y;
    switch ($position){
        case 'RB' :
            $im_x = imagesx($this->im) - $width - $margin_x;
            $im_y = imagesy($this->im) - $height - $margin_y;
            break;
        case 'LB' :
            $im_x = $margin_x;
            $im_y = imagesy($this->im) - $height - $margin_y;
            break;
        case 'LT' :

```

```

        $im_x = $margin_x;
        $im_y = $margin_y;
        break;
    case 'RT' :
        $im_x = imagesx($this->im) - $width - $margin_x;
        $im_y = $margin_y;
        break;
    }

    $this->copy($this->im,$image,$im_x,$im_y,0,0,$width,$height);
}

# void 이미지 자르기 int width,height,x,y
public function cropImage(int $width,int $height, int $x, int $y) : void{
    $this->im = $this->createTrueImage($width,$height);
    $image = $this->readImage($this->filename);
    if($this->copy($this->im,$image,0,0,$x,$y,$width,$height) === false)
        throw new \Exception(__METHOD__,__LINE__);
}

# void 이미지 자르기 (center) int width,height
public function cropThumbnailImage(int $width, int $height) : void
{
    $imgsize = $this->getImageSize($this->filename);

    # 조정
    $im_x = 0;
    $im_y = 0;
    $image_x = 0;
    $image_y = 0;

    $wm = $imgsize->width/$width;
    $hm = $imgsize->height/$height;
    $h_height = $height/2;
    $w_width = $width/2;

    if($imgsize->width > $imgsize->height){
        $width = $imgsize->width / $hm;
        $half_width = $width / 2;
        $im_x = -($half_width - $w_width);
    }else if(($imgsize->width < $imgsize->height) || ($imgsize->width == $imgsize->height)){
        $height = $imgsize->height / $wm;
        $half_height = $height / 2;
        $im_y = $half_height - $h_height;
    }

    $this->im = $this->createTrueImage($width,$height);
    $image = $this->readImage($this->filename);
    if($this->copyResampled($this->im,$image,$im_x,$im_y,$image_x,$image_y,$width,$height,$imgsize->width,$imgsize->height) === false)
        throw new \Exception(__METHOD__,__LINE__);
}

# 썸네일 이미지 만들기 int width, height
public function thumbnailImage(int $width, int $height) : void
{
    $imgsize = $this->getImageSize($this->filename);

    # 썸네일 사진 사이즈 설정
    if($imgsize->width>$imgsize->height){
        $height= ceil(($imgsize->height*$width)/$imgsize->width);
    }
    else if($imgsize->width<$imgsize->height || $imgsize->width == $imgsize->height){
        $width= ceil(($imgsize->width*$height)/$imgsize->height);
    }

    $this->im = $this->createTrueImage($width,$height);
    $image = $this->readImage($this->filename);
    if($this->copyResampled($this->im, $image, 0,0,0,0,$width,$height,$imgsize->width,$imgsize->height) ===false)
        throw new \Exception(__METHOD__,__LINE__);
}

# imagecopy
public function copy(mixed $im, mixed $image,int $im_x, int $im_y, int $image_x, int $image_y, int $width, int $height) : bool
{
    if(imagecopy($im,$image,$im_x,$im_y,$image_x,$image_y,$width,$height) === false) return false;
    return true;
}

# imagermerge

```

```

public function copyMerge(mixed $im, mixed $image, int $im_x, int $im_y, int $image_x, int $image_y, int $width, int $height, $pct) : bool
{
    if(!imagecopymerge($im, $image, $im_x, $im_y, $image_x, $image_y, $width, $height, $pct)) return false;
    return true;
}

# imagecopyresampled
public function copyResampled(mixed $im, mixed $image, $im_x, $im_y, $image_x, $image_y, $width, $height, $oriwidth, $oriheight):bool
{
    if(imagecopyresampled($im, $image, $im_x, $im_y, $image_x, $image_y, $width, $height, $oriwidth, $oriheight) === false) return false;
    return true;
}

# void : createtruecolor
public function createTrueImage(int $width, int $height){
    return $im = imagecreatetruecolor($width, $height);
}

# void
public function readImage(string $filename){
    $count = strrpos($filename, '.');
    $extention = strtolower(substr($filename, $count+1));
    try{
        switch($extention){
            case 'gif': $image = imagecreatefromgif($filename); break;
            case 'png': $image = @imagecreatefrompng($filename); break;
            case 'jpeg':
            case 'jpg': $image = imagecreatefromjpeg($filename); break;
        }
    }catch(\Exception $e){
        throw new \Exception($e->getMessage());
    }
    return $image;
}

# string filename
public function write(string $filename) : void
{
    $count = strrpos($filename, '.');
    $extention = strtolower(substr($filename, $count+1));
    try{
        switch($extention){
            case 'gif': imagegif($this->im, $filename); break;
            case 'png': imagepng($this->im, $filename, ($this->quality/10)-1); break;
            case 'jpeg':
            case 'jpg': imagejpeg($this->im, $filename, $this->quality); break;
        }
    }catch(\Exception $e){
        throw new \Exception($e->getMessage());
    }
}

# @ void : GD 버전
public function getVersion() : float{
    if(function_exists('gd_info')){
        $info = gd_info();
        return floatval(preg_replace('/bundled \((.*) compatible\)/', '\\1', $info['GD Version']));
    }
    return 0.0;
}

# image data base64 이미지 소스 읽기
public function readImageFromBase64(string $base64)
{
    $data = explode(',', $base64);
    $data = base64_decode($data[1]);
    $image = imagecreatefromstring($data);
    if (!$image) {
        throw new \Exception("Invalid base64 image data");
    }
    return $image;
}

# image data base64 이미지 쓰기
public function writeImageToBase64(): string
{
    ob_start();
    imagepng($this->im);
    $image_data = ob_get_contents();
}

```

```

        ob_end_clean();
        return 'data:image/png;base64,' . base64_encode($image_data);
    }

    # image data base64 이미지 크기 변경하기
    public function resizeBase64Image(string $base64, int $width, int $height): string
    {
        $this->im = $this->readImageFromBase64($base64);
        $imgsize = imagesx($this->im);
        $imgheight = imagesy($this->im);

        if ($imgsize > $imgheight) {
            $height = ceil(($imgheight * $width) / $imgsize);
        } else if ($imgsize < $imgheight || $imgsize == $imgheight) {
            $width = ceil(($imgsize * $height) / $imgheight);
        }

        $resized = $this->createTrueImage($width, $height);
        if ($this->copyResampled($resized, $this->im, 0, 0, 0, 0, $width, $height, $imgsize, $imgheight) === false)
            throw new \Exception(__METHOD__, __LINE__);

        $this->im = $resized;
        return $this->writeImageToBase64();
    }

    # 이미지 사이즈
    public function getImageSize(string $filename) : \stdClass {
        $size = getimagesize($filename);
        if ($size === false) {
            throw new \Exception("Cannot get image size of $filename");
        }
        $imageSize = new \stdClass();
        $imageSize->width = $size[0];
        $imageSize->height = $size[1];
        $imageSize->mime = $size['mime'];
        return $imageSize;
    }

    # @ void
    public function destroy() : void {
        if ($this->im) {
            imagedestroy($this->im);
        }
    }

    public function __destruct(){
        $this->destroy();
    }
}

```

--- 파일 경로: classes/Image/ImageViewer.php ---

```

<?php
namespace Flex\Banana\Classes\Image;

use Flex\Banana\Classes\Image\ImageGDS;
use Exception;

# 이미지 뷰어
final class ImageViewer extends ImageGDS
{
    public const __version = '2.0.1';

    # 이미지 경로
    public string $file_extension = "";
    public string $mimeType;
    public string $basename;
    public string $directory;
    public int $compression;
    public array $viewsizes = [];

    final public function __construct(string $filename){
        $this->filename = $filename;
        $this->getExtName();
    }
}

```

```

}

# 파일 확장자 추출
private function getExtName() : void{
    $this->basename = basename($this->filename);
    $count = strrpos($this->basename, '.');
    $this->file_extension = strtolower(substr($this->basename, $count+1));
    $this->directory = str_replace('/', $this->basename, $this->filename);
}

public function setFilter(int $compression, string $size, array $allowe_extension=['jpg','jpeg','png','gif']) : ImageViewer
{
    # mimeType
    $file_type = 'application';
    if(in_array($this->file_extension, $allowe_extension)){
        $file_type = 'image';
    }else throw new Exception('e_extension_not_allowed');

    $this->mimeType = $file_type.'.'.$this->file_extension;
    $this->compression = $compression;
    $this->viewsize = (strpos($size, 'x') !== false) ? explode('x', $size) : [];

    return $this;
}

/**
 * @ filename : 파일명
 * @ compression : 압축률
 * @ size : 이미지 사이즈
 * @ file_extension : ['jpg','jpeg','png'] 허용파일 확장자
 */
public function getContents () : string
{
    $imagecontents = "";
    if(strpos($this->mimeType, 'image/') !== false)
    {
        $fullname = "";
        $thumb_filename = 'thumb'. $this->compression.implode('x', $this->viewsize).'_'. $this->basename;
        if(file_exists($this->directory.'/' . $thumb_filename)){
            $fullname = $this->directory.'/' . $thumb_filename;
        }else{
            try{
                // $image_size = @getimagesize($fullname);
                parent::__construct( $this->filename );
                parent::setCompressionQuality($this->compression);

                # resize
                if(isset($this->viewsize[0])){
                    parent::thumbnailImage($this->viewsize[0], $this->viewsize[1]);
                    parent::write($this->directory.'/' . $thumb_filename);
                    $fullname = $this->directory.'/' . $thumb_filename;
                }

                # 압축
                parent::write($this->directory.'/' . $thumb_filename);
                $fullname = $this->directory.'/' . $thumb_filename;
            }catch(Exception $e){
                throw new Exception($e->getMessage());
            }
        }

        # base64 image data
        $imagecontents = file_get_contents($fullname);
    }

    return $imagecontents;
}

public function fetch() : array {
    $result = [
        'filename' => $this->basename,
        'mimeType' => $this->mimeType,
        'extension' => $this->file_extension,
        'contents' => $this->getContents()
    ];

    return $result;
}
}

```

--- 파일 경로: classes/Json/JsonDecoder.php ---

```
<?php
namespace Flex\Banana\Classes\Json;

final class JsonDecoder
{
    public const __version = '0.1.0';

    # 문자열 또는 중첩 문자열을 json 배열로 변환
    final public static function toArray( string $value, bool $assoc = true, int $depth = 512, bool $throwExceptionOnError = false ) : array
    {
        do {
            $decoded = json_decode($value, $assoc, $depth);
            if (json_last_error() !== JSON_ERROR_NONE) {
                if ($throwExceptionOnError) {
                    throw new \InvalidArgumentException('Invalid JSON: ' . json_last_error_msg());
                }
                return $value;
            }
            $value = $decoded;
        } while (is_string($value));

        return $value;
    }
}
```

--- 파일 경로: classes/Json/JsonEncoder.php ---

```
<?php
namespace Flex\Banana\Classes\Json;

final class JsonEncoder
{
    public const __version = '0.1.1';

    # 배열을 json string utf8
    final public static function toJson(array $data, array $except_numeric_keys = [], int $options=JSON_UNESCAPED_UNICODE) : string
    {
        $data = JsonEncoder::applyNumericExceptions($data, $except_numeric_keys);
        return json_encode($data, $options);
    }

    # 특정키를 제외한 numeric으로 변형하기
    private static function applyNumericExceptions(array $data, array $exceptNumericKeys) : array
    {
        foreach ($data as $key => &$value) {
            if (is_array($value)) {
                $value = JsonEncoder::applyNumericExceptions($value, $exceptNumericKeys);
            } elseif (is_numeric($value) && in_array($key, $exceptNumericKeys)) {
                $value = (string) $value;
            } elseif (is_numeric($value)) {
                $value = $value + 0; // 숫자 형 변환
            }
        }
        return $data;
    }
}
```

--- 파일 경로: classes/Log.php ---

```
<?php
```

```

namespace Flex\Banana\Classes;

final class Log
{
    public const __version = '1.2.2';
    const MESSAGE_FILE = 3; # 사용자 지정 파일에 저장
    const MESSAGE_ECHO = 2; # 화면에만 출력
    const MESSAGE_SYSTEM = 0; # syslog 시스템 로그파일에 저장

    public static $message_type = 3;
    public static $logfile = 'log.txt';

    public static $debugs = ['d','v','i','w','e'];
    public static $options = [
        'datetime' => true, # 날짜 시간 출력
        'debug_type' => true, # 디버그 타입 출력
        'newline' => true # 한줄내리기 출력
    ];

    # init
    public static function init(int $message_type = -1, string $logfile = null){
        Log::$message_type = ($message_type > -1) ? $message_type : Log::MESSAGE_ECHO;
        Log::$logfile = $logfile ?? 'log.txt';
    }

    # 출력 옵션 설정
    public static function options (array $options=[], bool $datetime=true, bool $debug_type=true, bool $newline=true) : void
    {
        $_options = [];
        if(is_array($options) && count($options)){
            $_options = $options;
        }else{
            $_options = [
                'datetime' => $datetime, 'debug_type' => $debug_type, 'newline' => $newline
            ];
        }

        Log::$options = array_merge(Log::$options, $_options);
    }

    # 출력하고자 하는 옵션 선택
    public static function setDebugs(string|array $m1, ...$mores): void
    {
        $debug_modes = [];
        $debug_modes[] = $m1;
        if(is_array($mores)){
            foreach($mores as $debug_type){
                $debug_modes[] = $debug_type;
            }
        }

        Log::$debugs = $debug_modes;
    }

    # debug
    public static function d (mixed $message, ... $message2) : void
    {
        if(in_array('d', Log::$debugs)){
            $output = Log::filterMessage($message).' | '.implode(' | ',array_map([Log::class, 'filterMessage'],$message2));
            Log::print_('D', $output);
        }
    }

    # success
    public static function v (mixed $message, ... $message2) : void
    {
        if(in_array('v', Log::$debugs)){
            $output = Log::filterMessage($message).' | '.implode(' | ',array_map([Log::class, 'filterMessage'],$message2));
            Log::print_('V', $output);
        }
    }

    # info
    public static function i (mixed $message, ... $message2) : void
    {
        if(in_array('i', Log::$debugs)){
            $output = Log::filterMessage($message).' | '.implode(' | ',array_map([Log::class, 'filterMessage'],$message2));
            Log::print_('I', $output);
        }
    }
}

```



```

}

# warning
public static function w (mixed $message, ... $message2) : void
{
    if(in_array('w', Log::$debugs)){
        $output = Log::filterMessage($message).' | '.implode(' | ',array_map([Log::class, 'filterMessage'],$message2));
        Log::print('W', $output);
    }
}

# error
public static function e (mixed $message, ... $message2) : void
{
    if(in_array('e', Log::$debugs)){
        $output = Log::filterMessage($message).' | '.implode(' | ',array_map([Log::class, 'filterMessage'],$message2));
        Log::print('E', $output);
    }
}

private static function filterMessage ( mixed $message) : mixed
{
    $result = $message;
    $typeof = gettype($message);
    if($typeof == 'array' || $typeof == 'object'){
        $result = print_r($message,true);
    }
    return $result;
}

# print
private static function print_ (string $debug_type, string $message) : void
{
    $logfile = (Log::$message_type == Log::MESSAGE_FILE ) ? Log::$logfile : null;
    $out_datetime = (Log::$options['datetime']) ? date('Y-m-d H:i:s').'' : "";
    $out_debug_type = (Log::$options['debug_type']) ? '>> '.$debug_type.' : ' : "";
    $out_newline = (Log::$options['newline']) ? PHP_EOL : "";

    if(Log::$message_type == Log::MESSAGE_ECHO){
        echo sprintf("%s%s%s%s", $out_datetime, $out_debug_type, addslashes($message), $out_newline);
    }else{
        error_log (
            sprintf("%s%s%s%s", $out_datetime, $out_debug_type, addslashes($message), $out_newline),
            Log::$message_type,
            $logfile
        );
    }
}
}

```

--- 파일 경로: classes/Mail/MailSend.php ---

```

<?php
namespace Flex\Banana\Classes\Mail;

class MailSend
{
    public const __version = '0.1';
    private $to = ['email'=>'', 'name'=>];
    private $from = ['email'=>'', 'name'=>];
    private $headers_args = [];
    private $message = "";

    private $headers = "";
    private $charset = 'utf-8';
    private $encoding = '8bit';
    private $boundary;

    public function __construct(){
        $this->boundary='_Part_'. md5(rand() . microtime());
        $this->headers = 'MIME-Version: 1.0' . "\r\n";

    }

    public function setTo($name,$email){

```

```

        $this->to['email'] = $email;
        $this->to['name'] = $name;
    }

    public function setFrom($email, $name){
        $this->from['email'] = $email;
        $this->from['name'] = $name;
    }

    public function setHeader($key, $value){
        if(!isset($this->headers[$key])){
            $this->headers_args[$key] = $value;
        }
    }

    public function setTextHtml($content){
        $this->headers = 'Content-Type: text/html; charset='. strtoupper($this->charset).'; format=flowed' ."\r\n";
        $this->message = $this->encodeMessage($content) . "\r\n";
    }

    public function setTextPlain($content){
        $this->headers = 'Content-Type: text/plain; charset='. strtoupper($this->charset) ."\r\n";
        $this->message = $this->encodeMessage($content) . "\r\n";
    }

    private function encodeMessage($message){
        switch($this->encoding){
            case 'base64':
                $message = chunk_split(base64_encode($this->setCharet($message)));
                $this->headers = 'Content-Transfer-Encoding: '.$this->encoding."n";
                break;
            default : $message = $this->setCharet($message); break;
        }
        return $message;
    }

    # 문자 출력 값이 utf-8인지 체크 후 변환하기
    public function setCharet($msg){
        if($this->charset=='euc-kr') return $this->isEuckrChg($msg);
        return $this->isUTF8Chg($msg);
    }

    #@ return String
    # utf-8 문자인지 체크 /--
    public function isUTF8Chg($msg)
    {
        if(iconv("utf-8","utf-8",$msg)==$msg) return $msg;
        else return iconv('euc-kr','utf-8',$msg);
    }

    #@ return String
    # euc-kr 문자인지 체크 /--
    public function isEuckrChg($msg)
    {
        if(iconv("euc-kr","euc-kr",$msg)==$msg) return $msg;
        else return iconv('utf-8','euc-kr',$msg);
    }

    public function send($subject)
    {
        # to
        $to = '='.strtoupper($this->charset).'?B?'.base64_encode($this->setCharet($this->to['name'])).'?=' <'. $this->to['email']. '>'. "\n";
        //$to = base64_encode($this->setCharet($this->to['name'])).' <'. $this->to['email']. '>';
        //$this->headers .= 'To: '.$to. "\n";

        # from
        $this->headers .= 'From: ='?'.strtoupper($this->charset).'?B?'.base64_encode($this->setCharet($this->from['name'])).'?=' <'. $this->from['email']. '>'. "\n";
        //$this->headers .= 'Reply-To: ='?'.strtoupper($this->charset).'?B?'.base64_encode($this->setCharet($this->from['name'])).'?=' <'. $this->from['email']. '>'. "\n";

        # subject
        $subject= '='.strtoupper($this->charset).'?B?'.base64_encode($this->setCharet($subject)).'?=';

        #send
        if(mail($to,$subject,$this->message,$this->headers)){ return true; }else{
            throw new \ErrorException('mail send error');
        }
    }
}

```

--- 파일 경로: classes/Model.php ---

```
<?php
namespace Flex\Banana\Classes;

class Model implements \ArrayAccess {
    public const __version = '2.0';
    private $args = [];

    public function __construct(?array $args = []) {
        if (is_array($args) && count($args)) {
            $this->args = $args;
        }
    }

    public function fetch(): array {
        return $this->args;
    }

    public function &__get(string $propertyName) {
        if (!array_key_exists($propertyName, $this->args)) {
            $this->args[$propertyName] = [];
        }
        return $this->args[$propertyName];
    }

    public function __set(string $propertyName, $value) {
        $this->args[$propertyName] = $value;
    }

    public function __isset(string $name): bool {
        return isset($this->args[$name]);
    }

    public function __unset(string $name): void {
        unset($this->args[$name]);
    }

    public function offsetSet(mixed $offset, mixed $value): void {
        if (is_null($offset)) {
            $this->args[] = $value;
        } else {
            $this->args[$offset] = $value;
        }
    }

    public function offsetExists(mixed $offset): bool {
        return isset($this->args[$offset]);
    }

    public function offsetUnset(mixed $offset): void {
        unset($this->args[$offset]);
    }

    public function &offsetGet(mixed $offset): mixed {
        if (!isset($this->args[$offset])) {
            $this->args[$offset] = [];
        }
        return $this->args[$offset];
    }

    public function __destruct() {
        unset($this->args);
    }
}
```

--- 파일 경로: classes/Paging/Relation.php ---

```
<?php
```

```

namespace Flex\Banana\Classes\Paging;

class Relation
{
    public const __version = '2.0';
    private $page = 1;      # 현재 페이지
    private $totalPage = 0;  # 총페이지
    private $qLimitStart = 0; # query LIMIT [0,[]
    private $qLimitEnd = 0;  # query LIMIT [],[0]
    private $totalBlock = 0;
    private $blockCount = 0;
    private $blockLimit = 10;
    private $blockStartPage = 0; # 블록 시작페이지
    private $blockEndPage = 0;   # 블록 끝페이지
    private $pageLimit = 0;
    private $totalRecord = 0;    # 총레코드 수

    private $relation = ['first'=> 0,'pre'=> 0,'next'=> 0,'last'=> 0];
    private $relation_current = [];

    /**
     * 필요한 기본값 등록
     * @param $totalRecord : 총 레코드 갯수
     * @param $page : 요청 페이지
     */
    public function __construct(int $totalRecord, int $page){
        $this->totalRecord = $totalRecord;
        $this->page = (empty($page)) ? $page : 1;
    }

    # 2 한페이지에 출력할 레코드 갯수
    public function query(int $pagecount=10, int $blockLimit=10) : Relation
    {
        $this->blockLimit = $blockLimit;
        $this->totalPage = @ceil($this->totalRecord/$pagecount);

        if($this->totalRecord ==0){
            $this->qLimitStart =0;
            $this->qLimitEnd =0;
        }else{
            $this->qLimitStart = $pagecount * ($this->page-1);
            $this->qLimitEnd = $pagecount;
        }

        $this->totalBlock = ceil($this->totalPage/$this->blockLimit);
        $this->blockCount = ceil($this->page/$this->blockLimit); // 현재속해 있는 block count
        $this->blockStartPage = ($this->blockCount-1) * $this->blockLimit;
        $this->blockEndPage = $this->blockCount*$this->blockLimit;

        if($this->totalBlock <=$this->blockCount) {
            $this->blockEndPage = $this->totalPage;
        }

        $this->pageLimit = $pagecount;
        return $this;
    }

    # @ 3
    # 출력
    public function build() : Relation
    {
        $this->rewindPage();
        $this->prevPage();
        $this->currentPage();
        $this->nextPage();
        $this->lastPage();
        return $this;
    }

    # @void 현재페이지 출력
    private function currentPage() : void
    {
        $s_page = $this->blockStartPage + 1;
        for($i = $s_page; $i <= $this->blockEndPage; $i++)
        {
            $this->relation_current[] = $i;
        }
    }

    # 이전페이지

```

```

private function prevPage() : void{
    if($this->page > 1 && $this->page <= $this->totalPage){
        $this->relation['pre'] = $this->page - 1;
    }
}

#다음페이지
private function nextPage() : void{
    if($this->page > 0 && $this->page < $this->totalPage){
        $this->relation['next'] = $this->page + 1;
    }
}

#처음페이지
private function rewindPage() : void{
    if($this->page > 1 && $this->page <= $this->totalPage){
        $this->relation['first'] = 1;
    }
}

#마지막페이지
private function lastPage() : void{
    if($this->page > 0 && $this->page <= ($this->totalPage-1)){
        $this->relation['last'] = $this->totalPage;
    }else{
        $this->relation['last'] = 0;
    }
}

# 프라퍼티 값 포함한 가져오기
public function __get($propertyName){
    if(property_exists(__CLASS__,$propertyName)){
        $result = $this->{$propertyName};
        if($propertyName == 'totalPage'){
            if($result==0){
                $result = 1;
            }
        }
    }
    return $result;
}

# 프라퍼티 값 변경하기
public function __set($propertyName, $value){
    if(property_exists(__CLASS__,$propertyName)){
        return $this->{$propertyName} = $value;
    }
}

#@ array
#first : 0
#pre : 0
#next : 3
#last : 5
#chanel : [1,2,3]
#페이지 채널 배열 출력
public function paging() : array
{
    $result = array_merge($this->relation,array('chanel'=>$this->relation_current));
    return $result;
}
}

```

--- 파일 경로: classes/R.php ---

```

<?php
namespace Flex\Banana\Classes;

use ArrayObject;
use Exception;
use JsonException;

final class R
{

```

```

public const __version = '2.3.2';
public static string $language = ""; // 국가코드

# resource 값
public static array $sysmsg = [];
public static array $strings = [];
public static array $arrays = [];
public static array $tables = [];
public static array $numbers = [];

private static array $cache = [];

# 배열값 추가 등록
public static function init(string $lang = ""): void
{
    self::$language = trim($lang);
}

# 특정 리소스 키에 해당하는 값 리턴
protected static function get(string $query, string $fieldname): mixed
{
    $cacheKey = "{$query}_{$fieldname}_". self::$language;
    if (isset(self::$cache[$cacheKey])) {
        return self::$cache[$cacheKey];
    }

    $target = self::getTarget($query);
    $result = $target[self::$language][$fieldname] ?? null;

    self::$cache[$cacheKey] = $result;
    return $result;
}

# 특정 리소스에 전체 값 바꾸기
public static function set(string $query, array $data): void
{
    $target = &self::getTarget($query);
    $target[self::$language] = $data;
    self::clearCache($query);
}

protected static function fetch(string $query): array
{
    $cacheKey = "{$query}_fetch_". self::$language;
    if (isset(self::$cache[$cacheKey])) {
        return self::$cache[$cacheKey];
    }

    $target = self::getTarget($query);
    $result = $target[self::$language] ?? [];

    self::$cache[$cacheKey] = $result;
    return $result;
}

# 특정리소스의 키에 해당하는 값들을 배열로 돌려받기
protected static function selectR(array $params): array
{
    $cacheKey = md5(serialize($params) . self::$language);
    if (isset(self::$cache[$cacheKey])) {
        return self::$cache[$cacheKey];
    }

    $argv = [];
    foreach ($params as $query => $fieldname) {
        $columns = str_contains($fieldname, ",") ? explode(",", $fieldname) : [$fieldname];
        foreach ($columns as $columnname) {
            $argv[$columnname] = self::get($query, trim($columnname));
        }
    }

    self::$cache[$cacheKey] = $argv;
    return $argv;
}

public static function __callStatic(string $query, array $args = []): mixed
{
    return match(true) {
        strtolower($query) === 'dic' && !empty($args) => (object)$args[0],
    }
}

```

```

        $query === 'fetch' && isset($args[0]) && is_string($args[0]) => self::fetch($args[0]),
        $query === 'select' && !empty($args) => self::selectR($args[0]),
        isset($args[0]) && is_string($args[0]) => self::get($query, $args[0]),
        isset($args[0]) && is_array($args[0]) => self::mergeData($query, $args[0]),
        default => null
    };
}

private static function &getTarget(string $query): array
{
    if (in_array($query, ['sysmsg', 'strings', 'numbers', 'arrays', 'tables'])) {
        return self::$$query;
    }
}

# 배열값 추가 금지
private static function mergeData(string $query, array $args): void
{
    $target = &self::getTarget($query);
    $target[self::$language] = ($target[self::$language] ?? []) + $args;
    self::clearCache($query);
}

# 데이터 로딩된 상태인지 체크
private static function is(string $query): bool
{
    $target = self::getTarget($query);
    return isset($target[self::$language]);
}

public static function parser(string $filename, string $query): void
{
    if (!$query) {
        throw new Exception(__CLASS__ . ' :: ' . __LINE__ . ' ' . $query . ' is null!');
    }

    if (!self::is($query)) {
        $real_filename = self::findLanguageFile($filename);
        $storage_data = file_get_contents($real_filename);
        if ($storage_data) {
            $data = self::filterJSON($storage_data, true);
            if (!is_array($data)) {
                throw new Exception(__CLASS__ . ' :: ' . __LINE__ . ' ' . $real_filename . ' / ' . $data);
            }
            self::mergeData($query, $data);
        }
    }
}

public static function filterJSON(string $json, bool $assoc = false, int $depth = 512, int $options = 0): mixed
{
    $json = preg_replace(['/\/.*$/m', '/\/.*?\/s/'], '', $json);
    $json = preg_replace('/s+/', '', $json);
    $json = preg_replace('/([{}])(s*)?([^\s]+?)s*/', '$1"$3"', $json);

    try {
        return json_decode($json, $assoc, $depth, JSON_THROW_ON_ERROR | $options);
    } catch (JsonException $e) {
        return $e->getMessage();
    }
}

public static function findLanguageFile(string $filename): string
{
    $path_parts = pathinfo($filename);
    $nation_filename = sprintf('%s/%s.%s.%s',
        $path_parts['dirname'],
        $path_parts['filename'],
        self::$language,
        $path_parts['extension']
    );
    return file_exists($nation_filename) ? $nation_filename : $filename;
}

private static function clearCache(string $query): void
{
    foreach (self::$cache as $key => $value) {
        if (strpos($key, $query) === 0) {
            unset(self::$cache[$key]);
        }
    }
}

```

```

    }
}

public function __destruct()
{
    foreach (['sysmsg', 'strings', 'numbers', 'arrays', 'tables', 'r', 'cache'] as $property) {
        unset(self::$$property);
    }
}
}

```

--- 파일 경로: classes/Random/Random.php ---

```

<?php
namespace Flex\Banana\Classes\Random;

# purpose : 랜덤문자 만들기
class Random {
    public const __version = '0.7';
    protected array $specialChars = ['!', '@', '#', '$', '%', '&', '_'];
    protected array $characters = [];

    public function __construct(array $characters = []){
        if(is_array($characters) && count($characters)){
            $this->characters = $characters;
        }else{
            if(!count($this->characters)){
                $this->characters = array_merge(range('A', 'Z'), range(0, 9));
            }
        }
    }

    # 숫자로 정해진 범위의 숫자로 난수를 만들어 낸다
    # min : 시작범위, max : 끝범위
    # 리턴 길이
    public function _number(int $min=0,int $max=9, int $length=1) : int
    {
        $result = "";
        // 첫 번째 숫자는 0이 아닌 숫자로 시작
        $result = (string)random_int(max(1, $min), $max);

        for($i=1; $i<$length; $i++){
            $result .= random_int($min,$max);
        }

        return (int) $result;
    }

    # 특수문자 랜덤
    public function _specialChars(int $length = 1): string {
        $result = "";
        $charCount = count($this->specialChars);

        for ($i = 0; $i < $length; $i++) {
            $result .= $this->specialChars[random_int(0, $charCount - 1)];
        }
        return $result;
    }

    # 배열중에서 갯수 만큼 추출해 내기
    public function _string(int $length=1, bool $includeSpecialChars = false) : string
    {
        $result = "";

        # 특수문자 포함 여부
        if($includeSpecialChars){
            $specialCharsCount = count($this->specialChars);
            $this->characters = array_merge($this->characters, $this->_specialChars( $specialCharsCount ));
        }

        # 갯수만큼 추출
        $array_keys = ($length==1) ? [array_rand($this->characters,$length)] : array_rand($this->characters,$length);
        # 결과
    }
}

```



```

        $cnt = count($array_keys);
        for($i=0; $i<$cnt; $i++){
            $result .= $this->characters[$array_keys[$i]];
        }
        return $result;
    }
}

```

--- 파일 경로: classes/Request/FormValidation.php ---

```

<?php
namespace Flex\Banana\Classes\Request;

use Flex\Banana\Classes\R;
use Flex\Banana\Classes\Request\Validation;

# 폼체크
class FormValidation extends Validation
{
    public const __version = '2.2';
    protected bool $required = false;
    protected $shouldSkipAllValidations = false;
    protected $conditions = [];

    public function __construct(
        protected string $fieldName,
        protected string $title,
        protected mixed $value
    ){
        parent::__construct($value);
    }

    # 조건문 생성기
    public function when($condition): FormValidation
    {
        if (is_callable($condition)) {
            $result = $condition();
        } elseif (is_bool($condition)) {
            $result = $condition;
        } else {
            $result = false;
        }

        if (!$result) {
            $this->shouldSkipAllValidations = true;
        }

        return $this;
    }

    # 조건문 이후의 항목 통과 체크용 메소드
    protected function shouldSkipValidation(): bool
    {
        return $this->shouldSkipAllValidations;
    }

    # 필수 옵션
    public function null () : FormValidation
    {
        if ($this->shouldSkipValidation()) {
            return $this;
        }

        $this->required = true;
        if(parent::isNull()) {
            $this->error_report($this->fieldName, 'e_null', sprintf("%s %s", $this->title, R::sysmsg('e_null')));
        }
        return $this;
    }

    # 길이
    public function length (int $min, int $max) : FormValidation
    {
        if ($this->shouldSkipValidation()) {

```

```

        return $this;
    }

    if($this->str && !parent::isStringLength([$min, $max])){
        $err_msg = sprintf( R::sysmsg('e_string_length'), $min, $max );
        $this->error_report($this->fieldName, 'e_string_length', sprintf("%s %s", $this->title, $err_msg));
    }
    return $this;
}

# 특수 문자 있으면 reject
public function disliking (array $arguments=[]) : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str){
        # 허용된 특수문자를 제거 한다.
        if(is_array($arguments)){
            foreach($arguments as $etcstr){
                $this->str = str_replace($etcstr, "", $this->str);
            }
        }

        if(!parent::isEtcString()){
            $etc_msg = (count($arguments) ? '['.implode(', ', $arguments).']' : "");
            $err_msg = sprintf(R::sysmsg('e_etc_string'), $etc_msg);
            $this->error_report($this->fieldName, 'e_etc_string', sprintf("%s %s", $this->title, $err_msg));
        }
    }
    return $this;
}

# 특수 문자 없으면 에러 (최소 1개이상 입력)
public function liking (array $arguments=[]) : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && parent::isEtcString()){
        $this->error_report($this->fieldName, 'e_chk_etc_string', sprintf("%s %s", $this->title, R::sysmsg('e_chk_etc_string')));
    }
    return $this;
}

# 공백체크
public function space () : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !parent::isSpace()){
        $this->error_report($this->fieldName, 'e_spaces', sprintf("%s %s", $this->title, R::sysmsg('e_spaces')));
    }
    return $this;
}

# enum
public function enum (array $arguments=[]) : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str){
        if(array_search($this->str, $arguments) === false)
            $this->error_report($this->fieldName, 'e_enum', sprintf("%s %s", $this->title, R::sysmsg('e_enum')));
    }
    return $this;
}

# 영문또는 숫자 만
public function alnum () : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

```

```

    }

    if($this->str && !ctype_alnum($this->str)){
        $this->error_report($this->fieldName, 'e_ctype_alnum', sprintf("%s %s", $this->title, R::sysmsg('e_ctype_alnum')));
    }
    return $this;
}

# 연속반복문자 체크
public function repeat(int $max) : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !parent::isSameRepeatString($max)){
        $err_msg = sprintf(R::sysmsg('e_same_repeat_string'), $max);
        $this->error_report($this->fieldName, 'e_same_repeat_string', sprintf("%s %s", $this->title, $err_msg));
    }
    return $this;
}

# 숫자인지 체크
public function number() : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !parent::isNumber()){
        $this->error_report($this->fieldName, 'e_number', sprintf("%s %s", $this->title, R::sysmsg('e_number')));
    }
    return $this;
}

# 영어만 체크
public function alphabet () : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !parent::isAlphabet()){
        $this->error_report($this->fieldName, 'e_alphabet', sprintf("%s %s", $this->title, R::sysmsg('e_alphabet')));
    }
    return $this;
}

# 알파벳인지 대문자 인지 체크
public function upal () : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !parent::isUpAlphabet()){
        $this->error_report($this->fieldName, 'e_up_alphabet', sprintf("%s %s", $this->title, R::sysmsg('e_up_alphabet')));
    }
    return $this;
}

# 알파벳인지 소문자 인지 체크
public function lowal () : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !parent::isLowAlphabet()){
        $this->error_report($this->fieldName, 'e_low_alphabet', sprintf("%s %s", $this->title, R::sysmsg('e_low_alphabet')));
    }
    return $this;
}

# 첫글자가 알파벳인지 체크
public function firstal () : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

```

```

    }

    if($this->str && !parent::isFirstAlphabet()){
        $this->error_report($this->fieldName, 'e_first_alphabet', sprintf("%s %s", $this->title,R::sysmsg('e_first_alphabet')));
    }
    return $this;
}

# json 타입의 데이터인지 체크
public function jsonf() :FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !parent::isJSON()){
        $this->error_report($this->fieldName, 'e_json', sprintf("%s %s", $this->title,R::sysmsg('e_json')));
    }
    return $this;
}

# 날짜데이터인지 체크
public function datef() :FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !parent::chkDate()){
        $this->error_report($this->fieldName, 'e_date', sprintf("%s %s", $this->title,R::sysmsg('e_date')));
    }
    return $this;
}

# 시간 데이터인지 체크
public function timef() :FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !parent::chkTime()){
        $this->error_report($this->fieldName, 'e_time', sprintf("%s %s", $this->title,R::sysmsg('e_time')));
    }
    return $this;
}

# 시작날짜와 종료날짜 이 올바른지 체크
public function dateperiod (string $end_date) : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str){
        $this->str = $this->str.'.'.$end_date;
        if(!parent::chkDatePeriod()){
            $this->error_report($this->fieldName, 'e_date_period',sprintf("%s %s", $this->title,R::sysmsg('e_date_period')));
        }
    }
    return $this;
}

# 두 문자가 일치하는지 체크
public function equal (mixed $value) : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str){
        $this->str = $this->str.'.'.$value;
        if(!parent::equals()){
            $this->error_report($this->fieldName, 'e_equals', sprintf("%s %s", $this->title,R::sysmsg('e_equals')));
        }
    }
    return $this;
}

# 이메일 데이터인지 체크

```

```

public function email () : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !filter_var($this->str, FILTER_VALIDATE_EMAIL)){
        $this->error_report($this->fieldName, 'e_formality', sprintf("%s %s", $this->title,R::sysmsg('e_formality')));
    }
    return $this;
}

# http:: url 데이터인지 체크
public function url () : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !filter_var($this->str, FILTER_VALIDATE_URL)){
        $this->error_report($this->fieldName, 'e_link_url', sprintf("%s %s", $this->title,R::sysmsg('e_link_url')));
    }
    return $this;
}

# 소수형 데이터 인지 체크
public function floatf () : FormValidation
{
    if ($this->shouldSkipValidation()) {
        return $this;
    }

    if($this->str && !is_float(floatval($this->str))){
        $this->error_report($this->fieldName, 'e_float', sprintf("%s %s", $this->title,R::sysmsg('e_float')));
    }
    return $this;
}

private function error_report(string $field, string $msg_code, string $msg)
{
    throw new \Exception(strval(
        json_encode(
            ['result'=>'false', 'fieldname'=>$field, 'msg_code'=>$msg_code, 'msg'=>$msg],
            JSON_UNESCAPED_UNICODE
        )
    ));
}
}

```

--- 파일 경로: classes/Request/Request.php ---

```

<?php
namespace Flex\Banana\Classes\Request;

class Request
{
    public const __version = '1.0.4';
    private array $params = [];
    private array $headers = [];
    public string $ip;
    public string $uri_path;
    public string $method;
    public string $port;

    public function __construct()
    {
        $this->ip = $this->getClientIp();
        $this->uri_path = $_SERVER['REQUEST_URI'] ?? "";
        $this->method = $_SERVER['REQUEST_METHOD'] ?? "";
        $this->port = $_SERVER['SERVER_PORT'] ?? "";
    }

    public function post(bool $is_trim = true): self
    {

```

```

    if (!empty($_POST)) {
        $this->trimParams($_POST, $is_trim);
    } else {
        $this->getInputContents($is_trim);
    }
    return $this;
}

public function input(bool $is_trim = true): self
{
    $this->getInputContents($is_trim);
    return $this;
}

public function patch(bool $is_trim = true): self
{
    $this->getInputContents($is_trim);
    return $this;
}

public function get(bool $is_trim = true): self
{
    $this->trimParams($_GET, $is_trim);
    return $this;
}

public function delete(bool $is_trim = true): self
{
    $this->trimParams($_GET, $is_trim);
    return $this;
}

public function trimParams(array $arg, bool $is_trim): void
{
    foreach ($arg as $k => $v) {
        if ($is_trim && !is_array($v)) {
            $v = trim($v);
        }
        $this->params[$k] = $v;
    }
}

private function getInputContents(bool $is_trim): void
{
    $post_data = file_get_contents('php://input');
    if ($post_data) {
        $post_json = json_decode($post_data, true);
        if (json_last_error() === JSON_ERROR_NONE) {
            $this->trimParams($post_json, $is_trim);
        } else {
            parse_str($post_data, $post_variables);
            if (!empty($post_variables)) {
                $this->trimParams($post_variables, $is_trim);
            }
        }
    }
}

public function getHeaders(): array
{
    if (empty($this->headers)) {
        if (function_exists('getallheaders')) {
            $this->headers = getallheaders();
        } elseif (function_exists('apache_request_headers')) {
            $this->headers = apache_request_headers();
        } else {
            foreach ($_SERVER as $name => $value) {
                if (str_starts_with($name, 'HTTP_')) {
                    $header_name = str_replace('_', '-', ucwords(strtolower(str_replace('_', ' ', substr($name, 5)))));
                    $this->headers[$header_name] = $value;
                }
            }
        }
    }
    return $this->headers;
}

public function getHeaderLine(string $name): string
{

```

```

        return $this->headers[$name] ?? "";
    }

    public function fetch(): array
    {
        return $this->params;
    }

    private static function getClientIp(): string
    {
        return $_SERVER['HTTP_CLIENT_IP'] ??
            $_SERVER['HTTP_X_FORWARDED_FOR'] ??
            $_SERVER['HTTP_X_FORWARDED'] ??
            $_SERVER['HTTP_FORWARDED_FOR'] ??
            $_SERVER['HTTP_FORWARDED'] ??
            $_SERVER['REMOTE_ADDR'] ?? "";
    }

    public function __get(string $propertyName): mixed
    {
        return $this->params[$propertyName] ?? null;
    }

    public function __set(string $key, mixed $value): void
    {
        $this->params[$key] = $value;
    }

    public function __isset(string $name): bool
    {
        return isset($this->params[$name]);
    }
}

```

### --- 파일 경로: classes/Request/Validation.php ---

```

<?php
namespace Flex\Banana\Classes\Request;

# purpose : 문자를 체크(Ascii 문자 코드를 활용하여) 한다 / preg,ereg 정규식 보다 훨 빠름
class Validation
{
    public const __version = '2.0';
    public string $str;
    public int $len = 0;

    public function __construct($s){
        $this->str = trim($s);
        if(!$this->isNullOrEmpty()){
            $this->len = strlen($s);
        }
    }

    # null 값인지 체크한다 [ 널값이면 : true / 아니면 : false ]
    public function isNullOrEmpty() : bool{
        $result = false;

        if(is_null($this->str) || $this->str==""){
            $result = true;
        }
        return $result;
    }

    # 문자와 문자사이 공백이 있는지 체크 [ 공백 있으면 : false / 없으면 : true ]
    public function isSpace(): bool{
        $str_split = count(preg_split("/ /", $this->str)); //split("[:space:]]+", $this->str);
        if($str_split > 1){
            return false;
        }
        return true;
    }

    # 연속적으로 똑같은 문자는 입력할 수 없다 [ 반복문자 max 이상이면 : false / 아니면 : true ]
    # ex : 010-111-1111,010-222-1111 형태제한

```

```

# max = 3; // 반복문자 3개 "초과" 입력제한
public function isSameRepeatString(int $max=3): bool{
    $result = true;
    $sameCount = 0;
    $preAsciiNumber = 0;
    for($i=0; $i < $this->len; $i++){
        if(isset($this->str[$i])){
            $asciiNumber = Ord($this->str[$i]);
            if( ($preAsciiNumber == $asciiNumber) && ($preAsciiNumber>0) ) $sameCount += 1;
            else $preAsciiNumber = $asciiNumber;

            if($sameCount==$max){
                $result = false;
                break;
            }
        }
    }
    return $result;
}

# 숫자인지 체크 [ 숫자면 : true / 아니면 : false ]
# Ascii table = 48 ~ 57
public function isNumber(): bool{
    $result = true;
    for($i=0; $i < $this->len; $i++){
        if(isset($this->str[$i])){
            $asciiNumber = Ord($this->str[$i]);
            if($asciiNumber<47 || $asciiNumber>57){
                $result = false;
                break;
            }
        }
    }
    return $result;
}

# 영문인지 체크 [ 영문이면 : true / 아니면 : false ]
# Ascii table = 대문자[75~90], 소문자[97~122]
public function isAlphabet(): bool{
    $result = true;
    for($i=0; $i < $this->len; $i++){
        if(isset($this->str[$i])){
            $asciiNumber = Ord($this->str[$i]);
            if(($asciiNumber>64 && $asciiNumber<91) || ($asciiNumber>96 && $asciiNumber<123)){
                $result = false;
            }
        }
    }
    return $result;
}

# 영문이 대문자 인지체크 [ 대문자이면 : true / 아니면 : false ]
# Ascii table = 대문자[75~90]
public function isUpAlphabet(): bool{
    $result = true;
    for($i=0; $i < $this->len; $i++){
        if(isset($this->str[$i])){
            $asciiNumber = Ord($this->str[$i]);
            if($asciiNumber<65 || $asciiNumber>90){
                $result = false;
                break;
            }
        }
    }
    return $result;
}

# 영문이 소문자 인지체크 [ 소문자면 : true / 아니면 : false ]
# Ascii table = 소문자[97~122]
public function isLowAlphabet(): bool{
    $result = true;
    for($i=0; $i < $this->len; $i++){
        if(isset($this->str[$i])){
            $asciiNumber = Ord($this->str[$i]);
            if($asciiNumber<97 || $asciiNumber>122){
                $result = false;
                break;
            }
        }
    }
    return $result;
}

```



```

return $result;
}

# 한글인지 체크한다 [ 한글이면 : true / 아니면 : false ]
# Ascii table = 128 >
public function isKorean(): bool{
    $result = false;
    for($i=0; $i < $this->len; $i++){
        if(isset($this->str[$i])){
            $asciiNumber = Ord($this->str[$i]);
            if($asciiNumber>128){
                $result = true;
                break;
            }
        }
    }
    return $result;
}

# 특수문자 입력여부 체크 [ 특수문자 찾으면 : false / 못찾으면 : true ]
# space 공백은 자동 제외
public function isEtcString(): bool
{
    $result = true;
    for($i=0; $i < $this->len; $i++){
        if(isset($this->str[$i])){
            $asciiNumber = Ord($this->str[$i]);
            if( ($asciiNumber<48) && ($asciiNumber != 46 && $asciiNumber != 32) ){ $result = false; break; }
            else if($asciiNumber>57 && $asciiNumber<65){ $result = false; break; }
            else if($asciiNumber>90 && $asciiNumber<97){ $result = false; break; }
            else if($asciiNumber>122 && $asciiNumber<128){ $result = false; break; }
        }
    }
    return $result;
}

# 첫번째 문자가 영문인지 체크한다[ 찾으면 : true / 못찾으면 : false ]
public function isFirstAlphabet(): bool{
    $result = true;
    $asciiNumber = Ord($this->str[0]);
    if(($asciiNumber>64 && $asciiNumber<91) || ($asciiNumber>96 && $asciiNumber<123)){
        else{ $result = false; }
    }
    return $result;
}

# 문자길이 체크 한글/영문/숫자/특수문자/공백 전부포함
# min : 최소길이 / max : 최대길이 utf-8
public function isStringLength(array $arguments): bool
{
    $strCount = 0;
    $min = $arguments[0];
    $max = $arguments[1];
    $str = $this->str;
    for($i=0;$i<$this->len;$i++)
    {
        if(isset($str[$i])){
            $asciiNumber = Ord($str[$i]);
            if($asciiNumber<=127 && $asciiNumber>=0){ $strCount++; }
            else if($asciiNumber<=223 && $asciiNumber>=194){ $strCount++; $i+1; }
            else if($asciiNumber<=239 && $asciiNumber>=224){ $strCount++; $i+2; }
            else if($asciiNumber<=244 && $asciiNumber>=240){ $strCount++; $i+3; }
        }
    }

    if($strCount<$min) return false;
    else if($strCount>$max) return false;
    else return true;
}

# 날짜가 정확한 날짜인지 체크
# 날짜 데이터 타입 (2012-01-12)
public function chkDate(): bool{
    if(strpos($this->str,'-')==false){
        return false;
    }
    $ymd_args = explode('-', $this->str);
    if(is_array($ymd_args)){
        foreach($ymd_args as $v){
            if(!is_numeric($v)){

```

```

        echo $v.PHP_EOL;
        return false;
    }
}
if(!checkdate($ymd_args[1],$ymd_args[2],$ymd_args[0])){
    return false;
}
}
return true;
}
# 시간이 정확한 시간에 속하는지 형태인지 체크
# 시간 데이터 타입(13:59:59)
public function chkTime(): bool{
    if(strpos($this->str,':') === false){
        return false;
    }
    $pattern1 = '/^(0?\d|1\d|2[0-3]):[0-5]\d:[0-5]\d$/'; // 12:30:20
    $pattern2 = '/^(0?\d|1\d|2[0-3]):[0-5]\d$/'; // 12:30
    if ( (preg_match($pattern1, $this->str)) || (preg_match($pattern2, $this->str)) ){
        return true;
    }
    return false;
}

# 두 날짜(2012-01-12 ~ 2012-01-13)가 정확한 기간인지 체크
# 뒤에 날짜가 앞에 날짜보다 작으면 안됨
# 두 날짜 데이터 타입(2012-01-12/2012-01-11)
public function chkDatePeriod_(): bool{
    $date = explode('/', $this->str);
    $s = explode('-', $date[0]);
    $e = explode('-', $date[1]);

    $sres= mktime(0,0,0,$s[1],$s[2],$s[0]);
    $eres= mktime(0,0,0,$e[1],$e[2],$e[0]);
    if($sres>$eres) return false;
    return true;
}

# JSON String 값인지 체크
public function isJSON(): bool{
    json_decode($this->str);
    return (json_last_error() == JSON_ERROR_NONE);
}

# 두 문자나 값이 서로 같은지 비교
public function equals(): bool{
    $result = true;
    $str = explode(' ', $this->str);

    if($str[0] != $str[1] ) $result = false;
    return $result;
}
}

```

--- 파일 경로: classes/Route/DbRoute.php ---

```

<?php
namespace Flex\Banana\Classes\Route;

use Flex\Banana\Classes\Json\JsonDecoder;
use Flex\Banana\Classes\Log;
use Flex\Banana\Classes\Db\DbManager;
use Flex\Banana\Adapters\DbAdapter;
use Flex\Banana\Interfaces\RouteInterface;

class DbRoute extends DbAdapter implements RouteInterface
{
    private string $table;
    public function __construct(DbManager $db, string $table) {
        # parent
        parent::__construct( $db );
        $this->table = $table;
    }
    #@ RouteInterface

```

```

        public function getRoutes(): array
        {
            $routes = [];
            if($result = $this->db->table($this->table)->select("url,types,flow")->query()){
                while ($row = $result->fetch_assoc()){
                    $routes[$row['url']] = [
                        'method' => strtoupper($row['types']),
                        'tasks' => JsonDecoder::toArray($row['flow'])
                    ];
                }
            }
            return $routes;
        }
    }
}

```

--- 파일 경로: classes/Route/JsonRoute.php ---

```

<?php
namespace Flex\Banana\Classes\Route;

use Flex\Banana\Classes\Json\JsonDecoder;
use Flex\Banana\Classes\Log;
use Flex\Banana\Interfaces\RouteInterface;

class JsonRoute implements RouteInterface
{
    public function __construct(
        private string $dir,
        private string $filename
    ) {}

    /** @ RouteInterface
     * {@$this->dir}/res/routes/index.json
     * public function getRoutes(): array
     */
    {
        $indexPath = sprintf("%s/%s", $this->dir, $this->filename);
        if (!file_exists($indexPath)) return [];
        $taskFiles = JsonDecoder::toArray(file_get_contents($indexPath));
        $routes = [];
        foreach ($taskFiles as $file) {
            $path = sprintf("%s/%s.json", $this->dir, $file);
            $routes = array_merge($routes, JsonDecoder::toArray(file_get_contents($path)));
        }
        return $routes;
    }
}

```

--- 파일 경로: classes/Route/RouteLoader.php ---

```

<?php
namespace Flex\Banana\Classes\Route;

use Flex\Banana\Classes\Log;
use Flex\Banana\Interfaces\RouteInterface;

final class RouteLoader
{
    private array $sources = [];

    public function addSource(RouteInterface $source): self {
        $this->sources[] = $source;
        return $this;
    }

    public function routes(): array {
        $merged = [];
        foreach ($this->sources as $source) {
            $merged = array_merge($merged, $source->getRoutes());
        }
    }
}

```

```

    return $this->expandVersionedRoutes($merged);
}

/**
 * [/t1.v1]/path 형식 확장
 */
private function expandVersionedRoutes(array $rawRoutes): array
{
    $expanded = [];

    foreach ($rawRoutes as $routePattern => $config)
    {
        if (preg_match('#^[(.*?)](/.+)$#', $routePattern, $matches))
        {
            $prefixes = explode(',', $matches[1]);
            $path = $matches[2];

            foreach ($prefixes as $prefix)
            {
                $prefix = rtrim($prefix, '/');
                $fullPath = $prefix . $path;
                $expanded[$fullPath] = $config;
            }
        } else {
            $normalizedPath = '/' . ltrim($routePattern, '/');
            $expanded[$normalizedPath] = $config;
        }
    }

    return $expanded;
}
}

```

--- 파일 경로: classes/Sharding/ConsistentHashing.php ---

```

<?php
namespace Flex\Banana\Classes\Sharding;

use Flex\Banana\Interfaces\ShardingStrategyInterface;

final class ConsistentHashing implements ShardingStrategyInterface
{
    private array $ring = []; // 해시 링
    private array $weights = []; // 서버별 가중치 기록
    private int $replicaUnit; // 기본 replica 단위
    private bool $sorted = false;

    public function __construct(int $replicaUnit = 3) {
        $this->replicaUnit = $replicaUnit;
    }

    public function addServer(string $group, string $server, int $weight = 1): void {
        $this->weights[$server] = $weight;
        $replicas = $this->replicaUnit * $weight;

        for ($i = 0; $i < $replicas; $i++) {
            $hash = $this->hash($server . ':' . $i);
            $this->ring[$hash] = $server;
        }
        $this->sorted = false;
    }

    public function removeServer(string $group, string $server): void {
        $weight = $this->weights[$server] ?? 1;
        $replicas = $this->replicaUnit * $weight;

        for ($i = 0; $i < $replicas; $i++) {
            $hash = $this->hash($server . ':' . $i);
            unset($this->ring[$hash]);
        }

        unset($this->weights[$server]);
        $this->sorted = false;
    }
}

```

```

public function getServer(string $group, string $key): ?string {
    if (empty($this->ring)) {
        throw new \RuntimeException("No servers registered.");
    }

    if (!$this->sorted) {
        ksort($this->ring);
        $this->sorted = true;
    }

    $hash = $this->hash($key);
    foreach ($this->ring as $ringHash => $server) {
        if ($hash <= $ringHash) {
            return $server;
        }
    }
    return reset($this->ring); // 링 끝을 넘으면 처음부터

private function hash(string $key): int {
    return hexdec(substr(md5($key), 0, 8)); // 32비트 해시
}
}

```

--- 파일 경로: classes/Sharding/JumpHashing.php ---

```

<?php
namespace Flex\Banana\Classes\Sharding;

use Flex\Banana\Interfaces\ShardingStrategyInterface;

final class JumpHashing implements ShardingStrategyInterface
{
    private array $servers = [];

    public function addServer(string $group, string $server, int $weight = 1): void {
        if (!in_array($server, $this->servers, true)) {
            $this->servers[] = $server;
        }
    }

    public function removeServer(string $group, string $server): void {
        $this->servers = array_values(array_filter(
            $this->servers,
            fn($s) => $s !== $server
        ));
    }

    public function getServer(string $group, string $key): ?string {
        $numServers = count($this->servers);
        if ($numServers === 0) {
            throw new \RuntimeException("No servers registered.");
        }

        // 'bcmath' 체크
        if (!extension_loaded('bcmath')) {
            throw new \RuntimeException("The Bcmath extension is required for this JumpHashing implementation to work correctly. Please install and enable the 'bcmath' extension.");
        }

        $hash = $this->hash64_for_bcmath($key);
        $b = -1;
        $j = 0;

        // Bcmath 연산을 위한 상수 (문자열)
        $multiplier = '2862933555777941757';
        $increment = '1';
        $modulus = '18446744073709551616'; // 2^64 (uint64 래핑 용)
        $shift_divisor = '8589934592'; // 2^33
        $jump_dividend = '2147483648'; // 2^31

        while ($j < $numServers) {
            $b = $j;
            $hash = bcmod(bcadd(bcmul($hash, $multiplier), $increment), $modulus);
            $randomizer = bcadd(bcdiv($hash, $shift_divisor, 0), '1');
        }
    }
}

```

```

        $dividend = bcmul((string)$b + 1, $jump_dividend);
        $j = (int)bcddiv($dividend, $randomizer, 0);
    }

    return $this->servers[$b];
}

private function hash64_for_bcmath(string $key): string {
    $unsigned_crc = sprintf("%u", crc32($key));
    return bcadd($unsigned_crc, '4294967296');
}
}

```

--- 파일 경로: classes/Sharding/ShardManager.php ---

```

<?php
namespace Flex\Banana\Classes\Sharding;

use Flex\Banana\Interfaces\ShardingStrategyInterface;

final class ShardManager {
    private static array $strategies = [];

    public static function use(string $group, ShardingStrategyInterface $strategy): void {
        self::$strategies[$group] = $strategy;
    }

    public static function addServer(string $group, string $server, int $weight = 1): void {
        self::$strategy($group)->addServer($group, $server, $weight);
    }

    public static function removeServer(string $group, string $server): void {
        self::$strategy($group)->removeServer($group, $server);
    }

    public static function getServer(string $group, string $key): ?string {
        return self::$strategy($group)->getServer($group, $key);
    }

    private static function strategy(string $group): ShardingStrategyInterface {
        if (!isset(self::$strategies[$group])) {
            throw new \RuntimeException("No strategy defined for group: {$group}");
        }
        return self::$strategies[$group];
    }
}

```

--- 파일 경로: classes/Strings/StringTools.php ---

```

<?php
namespace Flex\Banana\Classes\Strings;

class StringTools
{
    public const __version = '0.2.2';
    public function __construct(private string $data=""){}

    # convert 10진 to string
    public function ascii2String () : StringTools
    {
        if(trim($this->data))
        {
            $cdata = "";
            $len = strlen($this->data);
            for($i=0; $i<=$len ; $i+=2){
                $charno = substr($this->data,$i,2);
                if($charno>=33 && $charno <= 99){ $cdata .= chr($charno); }
                else {
                    $charno = substr($this->data,$i,3);
                    if($charno =100 && $charno <= 127){ $cdata .= chr($charno); $i++;}
                }
            }
        }
    }
}

```

```

    }
}

$this->data = $cdata;
}

return $this;
}

# convert 16진 to 10진
public function hex2Ascii () : StringTools
{
    if(trim($this->data))
    {
        $cdata = "";
        $len = strlen($this->data)-1;
        for ($i=0; $i < $len; $i+=2){
            $v = base_convert($this->data[$i].$this->data[$i+1], 16, 10);
            if($v != '0'){
                $cdata .= $v;
            }
        }
        $this->data = $cdata;
    }

    return $this;
}

# convert string to ASCII
public function string2Ascii() : StringTools
{
    if(trim($this->data))
    {
        $cdata = "";
        $len = strlen($this->data);
        for($i = 0; $i < $len; $i++) {
            $cdata .= ord($this->data[$i]);
        }
        $this->data = $cdata;
    }
    return $this;
}

# convert ASCII to 16진 (hex)
public function ascii2Hex() : StringTools
{
    if(trim($this->data))
    {
        $cdata = "";
        $len = strlen($this->data);
        for($i = 0; $i < $len; $i += 3) {
            $ascii = substr($this->data, $i, 3);
            $cdata .= sprintf("%02x", $ascii);
        }
        $this->data = $cdata;
    }
    return $this;
}

public function __get(string $propertyName) : mixed{
    $result = [];
    if(property_exists($this,$propertyName)){
        $result = $this->{$propertyName};
    }
    return $result;
}
}

```

--- 파일 경로: classes/TaskFlow.php ---

```

<?php
namespace Flex\Banana\Classes;
use Flex\Banana\Classes\Model;

```

```

final class TaskFlow extends Model
{
    public const __version = '1.2.2';
    private mixed $active = null;
    private array $adapters = [];

    private $errorCallback = null;

    public function __construct(?array $args = []) {
        parent::__construct($args);
    }

    public function do(mixed $instance): mixed
    {
        if ($instance instanceof \Closure) {
            try {
                $instance($this);
                return $this;
            } catch (\Throwable $e) {
                Log::e($e->getMessage() . "\n" . $e->getTraceAsString());
                if (is_callable($this->errorCallback)) {
                    call_user_func($this->errorCallback, $e);
                }
                throw $e;
            }
        }

        $this->active = $instance;
        return $this; // 변경: 클로저가 아닌 경우 $instance($this) 제거
    }

    public function adapter(string $name){
        return $this->adapters[$name] ?? null;
    }

    public function registerAdapter(object $adapter): static
    {
        if (is_object($adapter)) {
            $fullClass = get_class($adapter);
            $className = substr(strrchr($fullClass, '\\'), 1);
            $this->adapters[$className] = $adapter;
        } else {
            throw new \InvalidArgumentException("Invalid arguments for registerAdapter()");
        }

        return $this;
    }

    public function onError(callable $callback): static
    {
        $this->errorCallback = $callback;
        return $this;
    }

    public function __destruct() {
        parent::__destruct();
        unset($this->active);
        unset($this->errorCallback);
    }
}

```

--- 파일 경로: classes/Text/TextKeyword.php ---

```

<?php
namespace Flex\Banana\Classes\Text;

# purpose : 문자를 변경하거나 더하거나 등 가공하는 역할을 한다.
class TextKeyword
{
    public const __version = '1.1';
    const CHARSET = 'utf-8';
    private $value;
    private array $allow_tags = [];
    # 키워드 중 지우고 싶은 글자 및 단어

```



```

protected array $filter_words = [];

# 키워드 중 끝 -1 글자 지우기
protected array $filter_end_words = [];

public function __construct(string $keyword, array $allow_tags=[]){
    if($keyword && $keyword !="){
        # 문자 특수 문자
        if(is_array($allow_tags) && count($allow_tags)){
            $this->allow_tags = $allow_tags;
        }
        $this->value = $keyword;
        $this->cleanWord();
    }
}

# 특수문자 제거 및 단어별 배열로 리턴
private function cleanWord() : TextKeyword
{
    # 한글 영어 숫자만 추출
    $pattern = '/[\\xE4-\\xED][\\x80-\\xBF]{2}[a-zA-Z0-9]';
    if(count($this->allow_tags)){
        # 허용된 특수 문자
        $pattern .= '|';
        foreach($this->allow_tags as $etcstr){
            $pattern .= '\\'.$etcstr;
        }
        $pattern .= '|';
    }
    $pattern .= ')+/';

    # 클린
    $keywords = $this->value;
    preg_match_all($pattern, $keywords, $match);
    $keywords = $match[0];
    if(is_array($keywords)){
        foreach($keywords as $n => $w)
        {
            # 영어만 추출
            preg_match_all('(?:<eng>[A-Za-z]+)|su", $w, $out);
            $eng = (isset($out['eng'][0])) ? $out['eng'][0] : "";
            if($eng){
                // echo $eng;
                # 발견된 단어에서 영어만 삭제
                $keywords[$n] = strtr($w, [$eng=>""]);
                # 발견된 영단어 추가
                $keywords[] = $eng;
            }
        }
    }

    $this->value = array_filter(array_unique($keywords));
    return $this;
}

# 분리된 단어중에서 필터 단어로 등록된 단어 지운 후 리턴
/**
 * filter_words : 특정 문자를 특정문자로 변환
 * filter_end_words : 마지막 끝 글자를 특정문자로 변환
 */
public function filterCleanWord(array $filter_words=[], array $filter_end_words=[]): TextKeyword
{
    # init
    if(is_array($filter_words) && count($filter_words)){
        $this->filter_words = $filter_words;
    }

    if(is_array($filter_end_words) && count($filter_end_words)){
        $this->filter_end_words = $filter_end_words;
    }

    # 필터
    $argv = (is_array($this->value)) ? $this->value : [$this->value];
    $data = array();
    foreach($argv as $w)
    {
        $s = strtr($w, $this->filter_words);
        if($s && $s !=")
        {

```

```

        $es = mb_substr($s, -1, NULL, TextKeyword::CHARSET);
        if (in_array($es, $this->filter_end_words)) {
            $slen = mb_strlen($s, TextKeyword::CHARSET) - 1;
            $data[] = mb_substr($w, 0, $slen, TextKeyword::CHARSET);
        }else{
            $data[] = $s;
        }
    }
}
if(count($data)>0){
    $this->value = array_unique($data);
}
return $this;
}

public function __get(string $propertyName){
    $result = [];
    if(property_exists($this,$propertyName)){
        $result = $this->{$propertyName};
    }
    return $result;
}
}

```

### --- 파일 경로: classes/Text/TextUtil.php ---

```

<?php
namespace Flex\Banana\Classes\Text;

# purpose : 문자를 변경하거나 더하거나 등 가공하는 역할을 한다.
class TextUtil
{
    public const __version = '1.1';
    private string $value;

    public function __construct(string $s){
        $this->value = $s;
    }

    # 기존문자에 문자 덧붙이기
    public function append(string|int $s) : TextUtil{
        $this->value .= $s;
        return $this;
    }

    # 기본 문자 앞에 덧붙이기
    public function prepend(string|int $s) : TextUtil {
        $this->value = $s.$this->value;
        return $this;
    }

    # 문자를 지정된길이부터 특정 문자로 변경하기
    # 010-4023-7046 => 010-****-7046
    # startNumber : 시작위치(index), endNumber : 길이만큼, chgString : 변형될 문자
    public function replace(int $startNumber, int $endNumber, string $chgString) : TextUtil
    {
        $result = "";
        $s = [];
        $sLength = strlen($this->value);
        $str =&$this->value;
        $cnt=0;
        $endNumber2 = ($startNumber-1)+$endNumber;
        for($i=0; $i<$sLength; $i++){
            if((Ord($str[$i])<=127)&&(Ord($str[$i])>=0)){ $result .= ($cnt>=$startNumber && $cnt<=$endNumber2) ? $chgString : $str[$i]; $cnt++;}
            else if((Ord($str[$i])<=223)&&(Ord($str[$i])>=194)){ $result .= ($cnt>=$startNumber && $cnt<=$endNumber2) ? $chgString : $str[$i].$str[$i+1]; $i+1; $cnt++;}
            else if((Ord($str[$i])<=239)&&(Ord($str[$i])>=224)){ $result .= ($cnt>=$startNumber && $cnt<=$endNumber2) ? $chgString : $str[$i].$str[$i+1].$str[$i+2]; $i+2; $cnt++;}
            else if((Ord($str[$i])<=244)&&(Ord($str[$i])>=240)){ $result .= ($cnt>=$startNumber && $cnt<=$endNumber2) ? $chgString : $str[$i].$str[$i+1].$str[$i+2].$str[$i+3]; $i+3; $cnt++;}
        }

        $this->value = $result;
        return $this;
    }

    # 문자 자르기

```

```

# length : 문자길이
public function cut(int $length, bool $is_apeend_cutstr = true, string $strip_tags = '<font><strong><b><strike>') : TextUtil
{
    $result = "";
    $str =&$this->value;

    # 예외태그 허용
    if(trim($strip_tags)) {
        $str = strip_tags($str, $strip_tags);
    }

    # 허용된 태그와 문자 분리
    preg_match_all("/<[^>]+>(.*)</[^>]+>|U", $str, $match);
    $match_str = (count($match[1])) ? implode(' ', $match[1]) : $str;

    # 태그를 제외한 문자 길이만 체크
    $len = strlen($match_str);
    if($len > $length)
    {
        for($i=0;$i<$length;$i++){
            if((Ord($match_str[$i])<=127)&&(Ord($match_str[$i])>=0)){ $result .= $match_str[$i];}
            else if((Ord($match_str[$i])<=223)&&(Ord($match_str[$i])>=194)){ $result .= $match_str[$i].$match_str[$i+1];$i+1;}
            else if((Ord($match_str[$i])<=239)&&(Ord($match_str[$i])>=224)){ $result .= $match_str[$i].$match_str[$i+1].$match_str[$i+2];$i+2;}
            else if((Ord($match_str[$i])<=244)&&(Ord($match_str[$i])>=240)){ $result .= $match_str[$i].$match_str[$i+1].$match_str[$i+2].$match_str[$i+3];$i+3;}
        }

        # 최종 길이에서 허용 태그 적용
        if(count($match[1])){
            foreach($match[1] as $idx => $match_val){
                $result = str_replace($match_val, $match[0][$idx], $result);
            }
        }

        # 끝에 줄임문자 붙이기
        if($is_apeend_cutstr) {
            $result.='...';
        }
    }

    if($result) $this->value = $result;
    return $this;
}

#숫자를 특정문자 타입의 형태로 출력
public function numberf(string $str='-') : TextUtil{
    $result = $this->value;
    $patterns = [
        4 => ['/(d{1,1})(d{1,3})/', '1'.'$str.'2],
        5 => ['/(d{1,2})(d{1,3})/', '1'.'$str.'2],
        6 => ['/(d{1,3})(d{1,3})/', '1'.'$str.'2],
        7 => ['/(d{1,3})(d{1,4})/', '1'.'$str.'2],
        8 => ['/(d{1,4})(d{1,4})/', '1'.'$str.'2],
        10 => ['/(d{1,3})(d{1,3})(d{1,4})/', '1'.'$str.'2'.'$str.'3],
        11 => ['/(d{1,3})(d{1,4})(d{1,4})/', '1'.'$str.'2'.'$str.'3],
        12 => ['/(d{1,4})(d{1,4})(d{1,4})/', '1'.'$str.'2'.'$str.'3],
        13 => ['/(d{1,1})(d{1,4})(d{1,4})(d{1,4})/', '1'.'$str.'2'.'$str.'3'.'$str.'4],
        14 => ['/(d{1,2})(d{1,4})(d{1,4})(d{1,4})/', '1'.'$str.'2'.'$str.'3'.'$str.'4],
        15 => ['/(d{1,3})(d{1,4})(d{1,4})(d{1,4})/', '1'.'$str.'2'.'$str.'3'.'$str.'4],
        16 => ['/(d{1,4})(d{1,4})(d{1,4})(d{1,4})/', '1'.'$str.'2'.'$str.'3'.'$str.'4]
    ];

    $length = strlen($result);
    if(isset($patterns[$length])){
        $this->value = preg_replace($patterns[$length][0], $patterns[$length][1],$this->value);
    }
    return $this;
}

public function __get(string $propertyName){
    $result = [];
    if(property_exists($this,$propertyName)){
        $result = $this->{$propertyName};
    }
    return $result;
}
}

```

--- 파일 경로: classes/Uuid/UuidGenerator.php ---

```
<?php
namespace Flex\Banana\Classes\Uuid;

class UuidGenerator
{
    public const __version = '1.3.1';
    public function __construct(){
    }

    #@ String
    public function v3(string $uuid, string $keyname) : mixed
    {
        if(!$this->is_valid($uuid)) return false;

        $nhex = str_replace(array('-', '{', '}', ' ', $uuid);
        $nstr = "";

        for($i = 0; $i < strlen($nhex); $i+=2) {
            $nstr .= chr(hexdec($nhex[$i].$nhex[$i+1]));
        }

        $hash = md5($nstr . $keyname);

        return sprintf("%08s-%04s-%04x-%04x-%12s",
            substr($hash, 0, 8),
            substr($hash, 8, 4),
            (hexdec(substr($hash, 12, 4)) & 0x0fff) | 0x3000,
            (hexdec(substr($hash, 16, 4)) & 0x3fff) | 0x8000,
            substr($hash, 20, 12)
        );
    }

    #@ String
    # GenerateUUID V4
    # prekey 0| timestamp, ymdhis, microtime(date('YmdHis') . substr((string)microtime(), 2, 6)) 를 사용할 경우 asc, desc 정렬이 가능함
    public function v4(?string $prekey=null)
    {
        $uuid = sprintf("%04x%04x-%04x-%04x-%04x%04x%04x",
            mt_rand(0, 0xffff), mt_rand(0, 0xffff),
            mt_rand(0, 0xffff),
            mt_rand(0, 0x0fff) | 0x4000,
            mt_rand(0, 0x3fff) | 0x8000,
            mt_rand(0, 0xffff), mt_rand(0, 0xffff), mt_rand(0, 0xffff)
        );

        return ($prekey !== null && $prekey) ? $prekey.'-'.$uuid : $uuid;
    }

    #@ String
    public function v5(string $uuid, string $keyname) : mixed
    {
        if(!$this->is_valid($uuid)) return false;

        $nhex = str_replace(array('-', '{', '}', ' ', $uuid);
        $nstr = "";

        for($i = 0; $i < strlen($nhex); $i+=2) {
            $nstr .= chr(hexdec($nhex[$i].$nhex[$i+1]));
        }

        $hash = sha1($nstr . $keyname);

        return sprintf("%08s-%04s-%04x-%04x-%12s",
            substr($hash, 0, 8),
            substr($hash, 8, 4),
            (hexdec(substr($hash, 12, 4)) & 0x0fff) | 0x5000,
            (hexdec(substr($hash, 16, 4)) & 0x3fff) | 0x8000,
            substr($hash, 20, 12)
        );
    }

    # 시간순 정렬 가능 DESC, ASC
    public function v7(string|int $prekey=null): string
    {
        $timestamp = ($prekey) ? time() + $prekey : floor(microtime(true) * 1000);
        # Random bits (74 bits)
```

```

$randA = random_bytes(5);
$randB = random_bytes(5);

# imESTAMP hex (12 chars)
$time_hex = str_pad(dechex($timestamp), 12, '0', STR_PAD_LEFT);

# variant bits
return sprintf('%s-%s-%s-%s-%s',
    $time_hex,
    bin2hex(substr($randA, 0, 2)),
    '7' . bin2hex(substr($randA, 2, 1)),
    '8' . bin2hex(substr($randA, 3, 1)),
    bin2hex($randB)
);
}

#@ String
public function is_valid(string $uuid) : mixed {
    return preg_match('/^\{?[0-9a-f]{8}\-[0-9a-f]{4}\-[0-9a-f]{4}\-[0-9a-f]{4}\-[0-9a-f]{12}\}?$$/i', $uuid) === 1;
}
}

```

--- 파일 경로: composer.json ---

```

{
    "name": "apmsoft/flexphp-banana",
    "description": "PHP",
    "type": "library",
    "license": "MIT",
    "homepage": "http://flexphp.fancyupsoft.com",
    "authors": [
        {
            "name": "Kim JongKwan",
            "email": "apmsoft@gmail.com",
            "homepage": "http://flexphp.fancyupsoft.com",
            "role": "Developer"
        }
    ],
    "require": {
        "php": ">=8.1"
    },
    "autoload": {
        "psr-4": {
            "Flex\\Banana\\Classes\\": "classes",
            "Flex\\Banana\\Adapters\\": "adapters",
            "Flex\\Banana\\Interfaces\\": "interfaces",
            "Flex\\Banana\\Traits\\": "traits",
            "Flex\\Banana\\Utils\\": "utils",
            "Flex\\Banana\\Task\\": "task"
        }
    }
}

```

--- 파일 경로: interfaces/BaseAdapterInterface.php ---

```

<?php
namespace Flex\\Banana\\Interfaces;

interface BaseAdapterInterface
{
    public function getVersion(): string;
}

```

--- 파일 경로: interfaces/DeleteInterface.php ---

```

<?php
namespace Flex\Banana\Interfaces;

interface DeleteInterface{
    public function doDelete(?array $params=[]) : ?string;
}

```

--- 파일 경로: interfaces/DoInterface.php ---

```

<?php
namespace Flex\Banana\Interfaces;

interface DoInterface{
    public function do(?array $params=[]) : ?string;
}

```

--- 파일 경로: interfaces/EditInterface.php ---

```

<?php
namespace Flex\Banana\Interfaces;

interface EditInterface{
    public function doEdit(?array $params=[]) : ?string;
}

```

--- 파일 경로: interfaces/EditUpdateInterface.php ---

```

<?php
namespace Flex\Banana\Interfaces;

use Flex\Banana\Interfaces\EditInterface;
use Flex\Banana\Interfaces\UpdateInterface;

interface EditUpdateInterface extends EditInterface, UpdateInterface {}

```

--- 파일 경로: interfaces/EnumInterface.php ---

```

<?php
namespace Flex\Banana\Interfaces;

interface EnumInterface
{
    public function filter(mixed $data = null, ...$params): mixed;
    public function format(mixed $data = null, ...$params): mixed;
    public function validate(mixed $data = null, ...$params): void;
}

```

--- 파일 경로: interfaces/EnumValueInterface.php ---

```

<?php
namespace Flex\Banana\Interfaces;

interface EnumValueInterface
{
    public static function byName(string $name, string $case = 'UPPER'): ?object;
    public function setValue(string $key, $value): void;
}

```

```
        public function getValue(string $key);
    }
```

--- 파일 경로: interfaces/ImageCompressorInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

use Flex\Banana\Classes\Image\ImageGDS;

interface ImageCompressorInterface
{
    public function getImageGDS(): ImageGDS;
}
```

--- 파일 경로: interfaces/InsertInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

interface InsertInterface{
    public function doInsert(?array $params=[]) : ?string;
}
```

--- 파일 경로: interfaces/ListInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

interface ListInterface{
    public function doList(?array $params=[]) : ?string;
}
```

--- 파일 경로: interfaces/PostInsertInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

use Flex\Banana\Interfaces\PostInterface;
use Flex\Banana\Interfaces\InsertInterface;

interface PostInsertInterface extends PostInterface, InsertInterface{
}
```

--- 파일 경로: interfaces/PostInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

interface PostInterface{
    public function doPost(?array $params=[]) : ?string;
}
```

--- 파일 경로: interfaces/ReplInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

interface ReplInterface{
    public function doRepl(?array $params=[]) : ?string;
}
```

--- 파일 경로: interfaces/ReplyInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

interface ReplyInterface{
    public function doReply(?array $params=[]) : ?string;
}
```

--- 파일 경로: interfaces/ReplyReplInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

use Flex\Banana\Interfaces\ReplyInterface;
use Flex\Banana\Interfaces\ReplInterface;

interface ReplyReplInterface extends ReplyInterface, ReplInterface{}
```

--- 파일 경로: interfaces/RouteInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

interface RouteInterface {
    public function getRoutes(): array;
}
```

--- 파일 경로: interfaces/ShardingStrategyInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

interface ShardingStrategyInterface {
    public function addServer(string $group, string $server, int $weight = 1): void;
    public function removeServer(string $group, string $server): void;
    public function getServer(string $group, string $key): ?string;
}
```

--- 파일 경로: interfaces/UpdateInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

interface UpdateInterface{
    public function doUpdate(?array $params=[]) : ?string;
}
```



```
}
```

--- 파일 경로: interfaces/ViewInterface.php ---

```
<?php
namespace Flex\Banana\Interfaces;

interface ViewInterface{
    public function doView(?array $params=[]) : ?string;
}
```

--- 파일 경로: res/sysmsg.json ---

```
{
    "e_null": "을(를) 입력 하세요",
    "e_spaces": "공백없이 입력 하세요",
    "e_enum": "에 해당하는 값을 찾을 수 없습니다.",
    "e_ctype_alnum": "영문 또는 숫자만 입력하세요",
    "e_same_repeat_string": "연속된 문자를 %s자 이상 입력할 수 없습니다.",
    "e_number": "숫자만 입력하세요",
    "e_korean": "한글을 입력할 수 없습니다",
    "e_string_length": "길이는 %d~%d자를 입력하세요",
    "e_etc_string": "허용된 특수문자%s 외에는 입력할 수 없습니다",
    "e_chk_etc_string": "특수문자를 최소 1개 이상 입력하세요",
    "e_alphabet": "영어(alphabet)을 입력 하세요",
    "e_date": "날짜를 정확하게 입력 하세요",
    "e_time": "시간을 정확하게 입력 하세요",
    "e_date_period": "날짜 기간을 정확하게 입력 하세요",
    "e_equals": "일치하지 않습니다.",
    "e_up_alphabet": "대문자로 입력 하세요",
    "e_low_alphabet": "소문자로 입력 하세요",
    "e_first_alphabet": "첫 글자는 영문으로만 입력 하세요",
    "e_json": "데이터를 JSON 형태로 입력 하세요",
    "e_float": "숫자와 소수형 숫자만 입력하세요",
    "e_link_url": "URL 주소 정확하게 입력 하세요"
}
```

--- 파일 경로: res/sysmsg\_en.json ---

```
{
    "e_null": "Please enter a value",
    "e_spaces": "Please enter without spaces",
    "e_enum": "Cannot find a value corresponding to this",
    "e_ctype_alnum": "Please enter only letters or numbers",
    "e_same_repeat_string": "You cannot enter more than %s consecutive characters",
    "e_number": "Please enter numbers only",
    "e_korean": "Korean characters cannot be entered",
    "e_string_length": "Please enter a length between %d and %d characters",
    "e_etc_string": "You cannot enter special characters other than the allowed ones%s",
    "e_chk_etc_string": "Please enter at least one special character",
    "e_alphabet": "Please enter English alphabet",
    "e_date": "Please enter the date correctly",
    "e_time": "Please enter the time correctly",
    "e_date_period": "Please enter the date period correctly",
    "e_equals": "Does not match",
    "e_up_alphabet": "Please enter in uppercase",
    "e_low_alphabet": "Please enter in lowercase",
    "e_first_alphabet": "The first character must be an English letter",
    "e_json": "Please enter the data in JSON format",
    "e_float": "Please enter only numbers and decimal numbers",
    "e_link_url": "Please enter the URL address correctly"
}
```

--- 파일 경로: res/sysmsg\_jp.json ---

```
{
  "e_null": "値を入力してください",
  "e_spaces": "スペースなしで入力してください",
  "e_enum": "該当する値が見つかりません",
  "e_ctype_alnum": "文字または数字のみを入力してください",
  "e_same_repeat_string": "%s文字以上の連続した文字は入力できません",
  "e_number": "数字のみを入力してください",
  "e_korean": "韓国語の文字は入力できません",
  "e_string_length": "%dから%d文字の長さで入力してください",
  "e_etc_string": "許可された特殊文字%s以外は入力できません",
  "e_chk_etc_string": "少なくとも1つの特殊文字を入力してください",
  "e_alphabet": "英語のアルファベットを入力してください",
  "e_date": "日付を正しく入力してください",
  "e_time": "時間を正しく入力してください",
  "e_date_period": "日付期間を正しく入力してください",
  "e_equals": "一致しません",
  "e_up_alphabet": "大文字で入力してください",
  "e_low_alphabet": "小文字で入力してください",
  "e_first_alphabet": "最初の文字は英語の文字でなければなりません",
  "e_json": "データをJSON形式で入力してください",
  "e_float": "数字と小数点の数字のみを入力してください",
  "e_link_url": "URLアドレスを正しく入力してください"
}
```

--- 파일 경로: res/sysmsg\_ko.json ---

```
{
  "e_null": "을(를) 입력 하세요",
  "e_spaces": "공백없이 입력 하세요",
  "e_enum": "에 해당하는 값을 찾을 수 없습니다.",
  "e_ctype_alnum": "영문 또는 숫자만 입력하세요",
  "e_same_repeat_string": "연속된 문자를 %s자 이상 입력할 수 없습니다.",
  "e_number": "숫자만 입력하세요",
  "e_korean": "한글을 입력할 수 없습니다",
  "e_string_length": "길이는 %d~%d자를 입력하세요",
  "e_etc_string": "허용된 특수문자%s 외에는 입력할 수 없습니다",
  "e_chk_etc_string": "특수문자를 최소 1개 이상 입력하세요",
  "e_alphabet": "영어(alphabet)을 입력 하세요",
  "e_date": "날짜를 정확하게 입력 하세요",
  "e_time": "시간을 정확하게 입력 하세요",
  "e_date_period": "날짜 기간을 정확하게 입력 하세요",
  "e_equals": "일치하지 않습니다.",
  "e_up_alphabet": "대문자로 입력 하세요",
  "e_low_alphabet": "소문자로 입력 하세요",
  "e_first_alphabet": "첫 글자는 영문으로만 입력 하세요",
  "e_json": "데이터를 JSON 형태로 입력 하세요",
  "e_float": "숫자와 소수형 숫자만 입력하세요",
  "e_link_url": "URL 주소 정확하게 입력 하세요"
}
```

--- 파일 경로: res/sysmsg\_zh.json ---

```
{
  "e_null": "请输入一个值",
  "e_spaces": "请输入时不要包含空格",
  "e_enum": "找不到与此对应的值",
  "e_ctype_alnum": "请只输入字母或数字",
  "e_same_repeat_string": "不能输入超过%s个连续的相同字符",
  "e_number": "请只输入数字",
  "e_korean": "不能输入韩文字符",
  "e_string_length": "请输入长度在%d到%d个字符之间",
  "e_etc_string": "不能输入除允许的特殊字符%s以外的特殊字符",
  "e_chk_etc_string": "请至少输入一个特殊字符",
  "e_alphabet": "请输入英文字母",
  "e_date": "请正确输入日期",
  "e_time": "请正确输入时间",
}
```

```

        "e_date_period": "请正确输入日期期间",
        "e_equals": "不匹配",
        "e_up_alphabet": "请输入大写字母",
        "e_low_alphabet": "请输入小写字母",
        "e_first_alphabet": "第一个字符必须是英文字母",
        "e_json": "请以JSON格式输入数据",
        "e_float": "请只输入数字和小数",
        "e_link_url": "请正确输入URL地址"
    }
}

```

### --- 파일 경로: task/EnumImportBasicTask.php ---

```

<?php
namespace Flex\Banana\Task;

use Flex\Banana\Classes\Log;

class EnumImportBasicTask
{
    public const __version = '0.2.0';
    private array $enumClassNames;

    public function __construct(array $enumClassNames)
    {
        $this->enumClassNames = $enumClassNames;
    }

    public function execute(): array
    {
        $result = [];
        // Log::d('*** enumClassNames ***', $this->enumClassNames);

        foreach ($this->enumClassNames as $enum) {
            Log::d('>> raw enum input', $enum);

            if (!is_string($enum)) {
                Log::w("Skipped non-string enum: " . json_encode($enum));
                continue;
            }

            if (empty($enum)) {
                Log::w("Skipped empty enum after unescape.");
                continue;
            }

            if (!class_exists($enum)) {
                Log::e("Enum class not found: " . $enum);
                continue;
            }

            $ref = new \ReflectionClass($enum);
            if ($ref->isEnum()) {
                $cases = $enum::cases();
                $short = $ref->getShortName();
                $result[$short] = $cases[0] ?? null;
            }
        }

        // Log::d('*** enumClassNames :: result ***', $result);

        return $result;
    }
}

```

### --- 파일 경로: task/ExceptionBasicTask.php ---

```

<?php
namespace Flex\Banana\Task;

use Flex\Banana\Classes\Json\JsonEncoder;

```

```

use Flex\Banana\Classes\Log;

class ExceptionBasicTask
{
    public const __version = '0.1.0';

    public function __construct(
    ){}

    public function execute(string|array $message = '예외가 발생했습니다.'): void
    {
        if (is_array($message)) {
            $message = JsonEncoder::toJson($message);
        }
        Log::e("[ExceptionBasicTask]", $message);
        throw new \Exception($message);
    }
}

```

--- 파일 경로: task/HttpRequestTask.php ---

```

<?php
namespace Flex\Banana\Task;

use Flex\Banana\Classes\Http\HttpRequest;
use Flex\Banana\Classes\Log;

class HttpRequestTask
{
    public const __version = '0.1.0';

    public function __construct(
    ) {}

    public function execute(string $method, array $set): mixed
    {
        try{
            $request = new HttpRequest();

            if (empty($set['url'])) {
                throw new \Exception("HttpRequestTask::execute - empty or invalid URL");
            }

            if(!is_string($set['params'])){
                throw new \Exception("HttpRequestTask::execute - 'params' must be string, got " . gettype($set['params']));
            }

            $headers = is_array($set['headers'] ?? null) ? $set['headers'] : [];
            $request->set($set['url'], $set['params'] ?? "", $headers);
            $responses = match(strtoupper($method)){
                "POST" => $request->post(),
                "GET"  => $request->get(),
                "PUT"  => $request->put(),
                "PATCH" => $request->patch(),
                "DELETE" => $request->delete(),
                default => throw new \Exception("HttpRequestTask::execute - Unsupported method: {$method}")
            };

            return $responses;
        }catch(\Exception $e){
            Log::e("HttpRequestTask::execute error", [
                'message' => $e->getMessage(),
                'set' => $set,
                'method' => $method
            ]);
            throw new \Exception("HttpRequestTask::execute failed: " . $e->getMessage());
        }
    }
}

```

--- 파일 경로: task/PagingRelationBasicTask.php ---

```
<?php
namespace Flex\Banana\Task;

use Flex\Banana\Classes\Paging\Relation;
use Flex\Banana\Classes\Log;

class PagingRelationBasicTask
{
    public const __version = '0.2.2';

    public function __construct(
        private int $total_record,
        private int $page
    ){}

    public function execute(int $page_count=10, int $block_limit=5) : array
    {
        $paging = new Relation( $this->total_record ?? 0, $this->page ?? 1 );
        $relation = $paging->query( $page_count, $block_limit )->build()->paging();

        return [
            "page" => $paging->page,
            "totalPage" => $paging->totalPage,
            "qLimitStart" => $paging->qLimitStart,
            "qLimitEnd" => $paging->qLimitEnd,
            "totalRecord" => $paging->totalRecord,
            "blockStartPage" => $paging->blockStartPage,
            "blockEndPage" => $paging->blockEndPage,
            "relation" => $relation
        ];
    }
}
```

--- 파일 경로: task/QueryDeleteBasicTask.php ---

```
<?php
namespace Flex\Banana\Task;

use Flex\Banana\Classes\Db\DbManager;
use Flex\Banana\Classes\Log;

class QueryDeleteBasicTask
{
    public const __version = '0.2.0';

    public function __construct(
        private DbManager $db,
        private string $table
    ){}

    private function _where(string|array $where): string|null {
        if (empty($where)) return null;

        if (is_string($where)) {
            return $where;
        }

        if (is_array($where)) {
            return $this->db->buildWhere($where);
        }

        return null;
    }

    public function execute(string | array $where) : void
    {
        $_where = $this->_where($where['where'] ?? "");
        if($_where)
        {
            try{
                $data = $this->db->table($this->table)->where($_where)->query()->fetch_assoc();
            }
        }
    }
}
```

```
<?php
namespace Flex\Banana\Task;

use Flex\Banana\Utils\Requested;
use Flex\Banana\Classes\Db\DbManager;
use Flex\Banana\Traits\FidTrait;
use Flex\Banana\Classes\Log;

class QueryInsertBasicTask
{
    public const __version = '0.3.0';
    use FidTrait;

    public function __construct(
        private DbManager $db,
        private string $table,
        private array $enums
    ) {}

    #@ Fid
    public function getTable(): string
    {
        return $this->table;
    }

    #@ Fid
    public function getFidColumnName(): string
    {
        return "fid";
    }

    public function execute(array $requested): void
    {
        try {
            $this->db->beginTransaction();

            foreach ($this->enums as $item) {
                if (is_array($item) || count($item) === 0) {
                    continue;
                }

                $enum = $item[0];
                $options = array_slice($item, 1);

                // 필요한 경우 클래스 문자열을 ENUM 인스턴스로 변환
                if (is_string($enum) && enum_exists($enum)) {
                    $cases = $enum::cases();
                    $enum = $cases[0] ?? null;
                }

                if (!$enum instanceof \BackedEnum) {
                    continue;
                }

                $columnName = $enum->value;
                if ($columnName == $this->getFidColumnName()) {
                    $this->db[$columnName] = $this->createParentFid();
                } else {
                    $this->db[$columnName] = $enum->filter($requested[$columnName] ?? "", ...$options);
                }
            }
        } catch (\Exception $e) {
            $this->db->rollBack();
            Log::error($e->getMessage());
        }
    }
}
```

```

    }
}

$this->db->table($this->table)->insert();
$this->db->commit();
} catch (\Exception $e) {
    throw new \Exception($e->getMessage());
}
}
}

```

--- 파일 경로: task/QueryReplyBasicTask.php ---

```

<?php
namespace Flex\Banana\Task;

use Flex\Banana\Utils\Requested;
use Flex\Banana\Classes\Db\DbManager;
use Flex\Banana\Traits\FidTrait;
use Flex\Banana\Classes\Log;

class QueryReplyBasicTask
{
    public const __version = '0.1.0';
    use FidTrait;

    public function __construct(
        private DbManager $db,
        private string $table,
        private array $enums
    ) {}

    /** @ Fid */
    public function getTable(): string
    {
        return $this->table;
    }

    /** @ Fid */
    public function getFidColumnName(): string
    {
        return "fid";
    }

    private function _where(string|array $where): string|null {
        if (empty($where)) return null;

        if (is_string($where)) {
            return $where;
        }

        if (is_array($where)) {
            return $this->db->buildWhere($where);
        }

        return null;
    }

    public function execute(string|array $where, array $requested): void
    {
        # 부모글이 있는지 체크하기
        $_where = $this->_where($where['where'] ?? "");
        if($_where)
        {
            $data = $this->db->table($this->table)->where($_where)->query()->fetch_assoc();
            if($data){
                try {
                    $this->db->beginTransaction();

                    foreach ($this->enums as $item) {
                        if (!is_array($item) || count($item) === 0) {
                            continue;
                        }
                        $enum = $item[0];

```

$$\left\{ \begin{array}{l} \\ \end{array} \right\}$$

--- 파일 경로: task/QuerySelectBasicTask.php ---

```
?php
namespace Flex\Banana\Task;

use Flex\Banana\Classes\Db\DbManager;
use Flex\Banana\Traits\FidTrait;
use Flex\Banana\Classes\Log;

class QuerySelectBasicTask
{
    public const __version = '0.2.1';

    use FidTrait;

    public function __construct(
        private DbManager $db,
        private string $table,
        private array $enums
    ){
        # query start
        $this->db->table($this->table)->select(
            $this->findSelectColumns()
        );
    }

    #@ Fid
    public function getTable(): string {
        return $this->table;
    }

    #@ Fid
    public function getFidColumnName(): string {
        return "fid";
    }

    #@ Select Columns String 목록 만들기
    private function findSelectColumns(): string
    {
        $columns = [];
        foreach ($this->enums as $item)
        {
            if (!is_array($item) || count($item) === 0) {
                continue;
            }
        }
    }
}
```



```

    }

    $enum = $item[0];

    // 필요한 경우 클래스 문자열을 ENUM 인스턴스로 변환
    if (is_string($enum) && enum_exists($enum)) {
        $enum = $enum::cases()[0];
    }

    if (!(($enum instanceof \BackedEnum)) {
        continue;
    }

    $columns[] = $enum->value;
}

return implode(",", $columns);
}

private function matches(string $query, mixed $qitem) : void
{
    match($query) {
        "where" => $this->_where($qitem ?? ""),
        "orderBy" => $this->_orderBy($qitem ?? ""),
        "limit" => $this->_limit($qitem ?? ""),
        default => throw new \Exception("Not Found {$query}")
    };
}

private function _where(string | array $where) : void {
    if (!empty($where))
    {
        if(is_string($where)){
            $this->db->where($where);
        }
        else if(is_array($where)){
            $this->db->where(...$where);
        }
    }
}

private function _orderBy(string | array $orderby) : void {
    if (!empty($orderby))
    {
        if(is_string($orderby)){
            $this->db->orderBy($orderby);
        }
        else if(is_array($orderby)){
            $this->db->orderBy(...$orderby);
        }
    }
}

private function _limit(int|array $limit): void {
    if (is_int($limit)) {
        $this->db->limit($limit);
    } elseif (is_array($limit)) {
        $this->db->limit(...$limit);
    }
}

/**
 * "params": [{
 *     "where": ". _id=1" | ["_id","asfdsd"] | ["age", ">=", "100"] | [{"_id", "1"}, ["age", ">=", "100]],
 *     "orderBy": "regidate DESC" | ["regidate DESC"] | ["regi_date DESC", "fid ASC"],
 *     "limit": 10 | [0,10]
 * }]
 */
public function execute(array $queries) : array
{
    try {
        $queryString = "";
        foreach($queries as $query => $qitem){
            $this->matches($query, $qitem);
        }

        $queryString = $this->db->query;
        Log::d('queryString', $queryString);
        $result = $this->db->query( $queryString );
    }
}

```

```

        $data = [];
        while ($row = $result->fetch_assoc())
        {
            $formattedRow = [];
            foreach ($this->enums as $item)
            {
                if (!is_array($item) || count($item) === 0) {
                    continue;
                }

                $enum = $item[0];
                $options = array_slice($item, 1);

                // 필요한 경우 클래스 문자열을 ENUM 인스턴스로 변환
                if (is_string($enum) && enum_exists($enum)) {
                    $enum = $enum::cases()[0];
                }

                if (!$enum instanceof \BackedEnum) {
                    continue;
                }

                $columnName = $enum->value;
                if ($columnName == $this->getFidColumnName()) {
                    $formattedRow[$columnName] = $enum->format($row[$columnName] ?? "", ...$options);
                    $formattedRow["depth"] = $this->getDepthCount($row[$columnName]);
                } else {
                    $formattedRow[$columnName] = $enum->format($row[$columnName] ?? "", ...$options);
                }
            }
            $data[] = $formattedRow;
        }

        return $data;
    } catch (\Exception $e) {
        throw new \Exception($e->getMessage());
    }
}
}

```

--- 파일 경로: task/QueryUpdateBasicTask.php ---

```

<?php
namespace Flex\BananaTask;

use Flex\BananaUtils\Requested;
use Flex\Banana\Classes\Db\DbManager;

class QueryUpdateBasicTask
{
    public const __version = '0.3.0';

    public function __construct(
        private DbManager $db,
        private string $table,
        private array $enums
    ){}

    private function _where(string | array $where) : void {
        if (!empty($where))
        {
            if (is_string($where)){
                $this->db->where($where);
            }
            else if (is_array($where)){
                $this->db->where(...$where);
            }
        }
    }

    public function execute(string | array $where, array $requested) : void
    {
        try {
            $this->db->beginTransaction();

```

```

foreach ($this->enums as $item)
{
    if (!is_array($item) || count($item) === 0) {
        continue;
    }

    $enum = $item[0];
    $options = array_slice($item, 1);

    // 필요한 경우 클래스 문자열을 ENUM 인스턴스로 변환
    if (is_string($enum) && enum_exists($enum)) {
        $enum = $enum::cases()[0];
    }

    if (!$enum instanceof \BackedEnum) {
        continue;
    }

    $columnName = $enum->value;
    $this->db[$columnName] = $enum->filter($requested[$columnName] ?? " ", ...$options);
}
$this->db->table($this->table)->where($this->_where($where))->update();
$this->db->commit();
} catch (\Exception $e) {
    throw new \Exception($e->getMessage());
}
}
}

```

--- 파일 경로: task/QueryWhereCaseBasicTask.php ---

```

<?php
namespace Flex\Banana\Task;

use Flex\Banana\Classes\Db\WhereHelper;
use Flex\Banana\Classes\Log;

class QueryWhereCaseBasicTask
{
    public const __version = '0.2.0';

    private mixed $where = "";

    public function __construct(
        private WhereHelper $dbWhere
    ){

        # execute('and', "name","=", "나당");
        # execute('and', ["name","=", "나다"]);
        # execute('and', ["name","=", "나다"], ["age",">=", 19]);
        public function execute(string $coord, ...$params) : void
        {
            $conditions = [];

            # 단일 조건: "name", "=", "나당"
            if (count($params) === 3 && !is_array($params[0])) {
                $conditions[] = [$params[0], $params[1], $params[2]];
            } else {
                foreach ($params as $index => $param) {
                    if (!is_array($param)) {
                        throw new \InvalidArgumentException(
                            "Parameter at index {$index} must be an array of 3 elements."
                        );
                    }
                    if (count($param) !== 3) {
                        throw new \InvalidArgumentException(
                            "Each condition array must contain exactly 3 elements: field, operator, value. Problem at index {$index}."
                        );
                    }
                    $conditions[] = $param;
                }
            }
        }

        # case
    }
}

```

```

        $coordCase = strtoupper($coord);
        $this->dbWhere->begin($coordCase);

        foreach($conditions as $casewh){
            list($fieldname, $condition, $value) = $casewh;
            Log::d($fieldname,$condition,$value);
            $this->dbWhere->case($fieldname, $condition, $value);
        }

        $this->dbWhere->end();
    }

    public function __get($propertyName) : mixed
    {
        Log::d(__CLASS__, 'propertyName', $propertyName);
        if ($propertyName === 'where') {
            $this->where = $this->dbWhere->__get('where');
            return $this->where;
        }
        return null;
    }
}

```

--- 파일 경로: task/RequestedFetchBasicTask.php ---

```

<?php
namespace Flex\Banana\Task;

use Flex\Banana\Utils\Requested;

class RequestedFetchBasicTask
{
    public const __version = '0.1.0';

    public function __construct(
        private Requested $requested
    ){

    }

    private function postFetch() : array
    {
        return $this->requested->post()->fetch();
    }

    private function getFetch() : array
    {
        return $this->requested->get()->fetch();
    }

    public function execute(string $method) : array
    {
        $methodCase = strtoupper($method);
        return match($methodCase) {
            "POST" => $this->postFetch(),
            "GET" => $this->getFetch(),
            default => throw new \InvalidArgumentException("Invalid method: {$method}"),
        };
    }
}

```

--- 파일 경로: task/SortByFidBasicTask.php ---

```

<?php
namespace Flex\Banana\Task;

use Flex\Banana\Utils\Requested;
use Flex\Banana\Classes\Db\DbManager;
use Flex\Banana\Traits\FidTrait;
use Flex\Banana\Classes\Log;

class SortByFidBasicTask

```

```

{
    public const __version = '0.1.0';
    use FidTrait;

    public function __construct(
        private DbManager $db,
        private string $table
    ) {}

    #@ Fid
    public function getTable(): string
    {
        return $this->table;
    }

    #@ Fid
    public function getFidColumnName(): string
    {
        return "fid";
    }

    public function execute(string $mode, string $fid, string $columnName = '_id'): void
    {
        Log::d($mode, $fid, $columnName);
        if($mode == 'down'){
            # 화살표 위
            $data = $this->getSortDown($fid);
            $this->changeSortFid ($data['cur_fids'], $data['ano_fids'], $columnName);
        }else if($mode == 'up'){
            # 화살표 아래
            $data = $this->getSortUp($fid);
            $this->changeSortFid ($data['cur_fids'], $data['ano_fids'], $columnName);
        }
    }
}

```

--- 파일 경로: task/TotalRecordBasicTask.php ---

```

<?php
namespace Flex\Banana\Task;

use Flex\Banana\Classes\Db\DbManager;
use Flex\Banana\Classes\Log;

class TotalRecordBasicTask
{
    public const __version = '0.1.0';

    public function __construct(
        private DbManager $db
    ){}

    public function execute(string $table, string $where) : int
    {
        return $this->db->table( $table )->where($where)->total();
    }
}

```

--- 파일 경로: task/ValidationBasicTask.php ---

```

<?php
namespace Flex\Banana\Task;

use Flex\Banana\Classes\Log;

class ValidationBasicTask
{
    public const __version = '0.2.1';

    public function __construct(

```

```

        private array $enums
    ){
        // Log::d("*** ValidationBasicTask enums", $this->enums);
    }

    public function execute(array $requested) : void
    {
        try {
            foreach ($this->enums as $item) {
                if (is_array($item) || count($item) === 0) {
                    continue;
                }

                $enum = $item[0];
                $options = array_slice($item, 1);

                // 필요한 경우 클래스 문자열을 ENUM 인스턴스로 변환
                if (is_string($enum) && enum_exists($enum)) {
                    $enum = $enum::cases()[0];
                }

                if (!$enum instanceof \BackedEnum) {
                    continue;
                }

                $key = $enum->value;
                $enum->validate($requested[$key] ?? "", ...$options);
            }
        } catch (\Exception $e) {
            throw new \Exception($e->getMessage());
        }
    }
}

```

--- 파일 경로: traits/DelimitedStringTrait.php ---

```

<?php
namespace Flex\Banana\Traits;

trait DelimitedStringTrait
{
    public function parseDelimited ( string $separator, string $value, array $default=[] ) : ?array
    {
        $result = ($value ?
            ((strpos($value, $separator) !== false) ? explode($separator, $value) : [$value]) :
            $default;
        return $result;
    }
}

```

--- 파일 경로: traits/EntryArrayTrait.php ---

```

<?php
namespace Flex\Banana\Traits;

trait EntryArrayTrait
{
    public static function names(): array
    {
        return array_column(self::cases(), 'name');
    }

    public static function values(): array
    {
        return array_column(self::cases(), 'value');
    }

    public static function array(): array
    {
        if (count(self::names()) == count(self::values())) {

```

```

        return array_combine(self::names(), self::values());
    } else {
        return [];
    }
}

public static function byName(string $name, string $case = 'UPPER'): ?object
{
    $NAME = ('UPPER' == strtoupper($case)) ? strtoupper($name) :
        (('LOWER' == strtoupper($case)) ? strtolower($name) : $name);

    foreach (self::cases() as $case) {
        if (strtoupper($case->name) === $NAME) {
            return (object)[
                'name' => $case->name,
                'value' => $case->value
            ];
        }
    }

    return null;
}

public static function __callStatic(string $name, array $args = []): string
{
    return (self::byName($name, $args[0] ?? 'UPPER'))->value;
}
}

```

--- 파일 경로: traits/EnumInstanceTrait.php ---

```

<?php
namespace Flex\Banana\Traits;

use Flex\Banana\Classes\Enum\EnumValueStorage;

trait EnumInstanceTrait
{
    public static function create(): self
    {
        return self::cases()[0];
    }

    public function setValue(string $key, $value): void
    {
        EnumValueStorage::setValue(static::class, $key, $value);
    }

    public function getValue(string $key)
    {
        return EnumValueStorage::getValue(static::class, $key);
    }

    public static function resetValues(): void
    {
        EnumValueStorage::reset(static::class);
    }

    public function getInstanceValues(): array
    {
        return EnumValueStorage::getValues(static::class);
    }
}

```

--- 파일 경로: traits/FidTrait.php ---

```

<?php
namespace Flex\Banana\Traits;

use Flex\Banana\Classes\Log;

```

```

# 스트링 다단 처리
# @ FidProviderInterface : required
# 이 클래스를 사용하려면 반드시 이 이클래스 사용하는 클래스에서 구현하세요
trait FidTrait
{
    # FidProviderInterface : required
    abstract protected function getTable(): string;
    abstract protected function getFidColumnName(): string;

    # reple 하부메뉴 및 답글에 사용
    # 다단 fid > "9999999997.01" AND fid < "9999999997.0199";
    public function createChildFid(string $fid) : string
    {
        # 해당 fid 중 가장 큰 값 찾기
        // [$this->getFidColumnName(), '>', $fid], [$this->getFidColumnName(), '<', $fid.'99']
        $where = sprintf("%s > '%s' AND %s < '%s'", $this->getFidColumnName(), $fid, $this->getFidColumnName(), $fid.'99');
        $fid_max = $this->db->table($this->getTable())
            ->select(sprintf("max(%s)", $this->getFidColumnName()))
            ->where($where)
            ->query()->fetch_row();

        # depth
        $depth = (isset($fid_max[0])) ? ((int)substr($fid_max[0], -2) + 1) : 1;

        # result
        return sprintf("%s%02d", $fid, $depth);
    }

    # Insert 에 fid 키 만들기
    public function createParentFid() : string
    {
        # fid min 값 가져오기
        $fid_row = $this->db->table( $this->getTable() )
            ->select(sprintf("min(%s)", $this->getFidColumnName()))->query()->fetch_row();

        # make fid
        $_fid = (isset($fid_row[0])) ? explode('.', $fid_row[0])[0] - 1 : '9999999999';
        return sprintf("%s.", $_fid);
    }

    # 현재 데프 길이
    public function getDepthCount(string $fid) : int
    {
        # fid
        $fids = (strpos($fid, ".") !== false) ? explode('.', $fid)[1] : "";
        return (int) (strlen($fids) / 2);
    }

    # depth 값이 만큼 가계 뽑아주기
    public function getFidGenealogy(string $fid) : array
    {
        # result
        $result = [];

        # depth
        $depth = $this->getDepthCount($fid);

        # category argv
        $root_pos = strpos($fid, ".");
        if($root_pos !== false){
            $start_pos = $root_pos + 1;
            for($i=0; $i<=$depth; $i++){
                $end_pos = $start_pos + ($i*2);
                $result[] = substr($fid, 0, $end_pos);
            }
        }

        return $result;
    }

    # list query
    # FID depth 정렬
    # mysql : fid+0 asc | pgsql : fid asc
    public function orderBy(?string $columnName=null, ?string $asc = 'ASC') : string
    {
        return sprintf("%s %s", $columnName ?? $this->getFidColumnName(), $asc);
    }

    # 화살표 위

```



```

public function getSortUp (string $fid) : array
{
    # current fid array
    $cur_fids = [];
    $cur_rlt = $this->db->table($this->getTable())
        ->where( $this->getFidColumnName(), 'LIKE-R', $fid)
        ->orderBy( $this->orderBy() )
        ->query();
    while($cur_row = $cur_rlt->fetch_assoc()){
        $cur_fids[] = $cur_row;
    }

    # < fid array
    $pre_fids = [];
    $depth = $this->getDepthCount($fid);
    $fids = explode('.', $fid);

    $pre_fid = "";
    if($depth < 1){
        $pre_fid = ($fids[0]-1).".";
    }else {
        $end_2str = substr($fid,-2);
        if($end_2str != '01'){
            $end_fid = ((int)$end_2str - 1);
            $pre_fid = sprintf("%s%02d", substr($fid,0,-2), $end_fid);
        }
    }

    if($pre_fid)
    {
        // Log::d('pre_fid ->', $pre_fid);
        $pre_row = $this->db->table($this->getTable())
            ->select($this->getFidColumnName())
            ->where(sprintf("%s >= '%s'", $this->getFidColumnName(), $pre_fid))
            ->limit(1)
            ->orderBy( $this->orderBy() )
            ->query()->fetch_assoc();
        if(isset($pre_row[$this->getFidColumnName()]))
        {
            # pre_fid
            $pre_fid = $pre_row[$this->getFidColumnName()];
            $pre_depth = $this->getDepthCount($pre_fid);
            // Log::d('pre_fid', $pre_fid, 'pre_depth', $pre_depth);

            if($depth == $pre_depth)
            {
                # query
                $pre_rlt = $this->db->table($this->getTable())
                    ->select("")
                    ->where($this->getFidColumnName(), 'LIKE-R', $pre_fid)
                    ->orderBy( $this->orderBy() )
                    ->query();
                while($pre_row = $pre_rlt->fetch_assoc()){
                    $pre_fids[] = $pre_row;
                }
                // Log::d('pre_fids', $pre_fids);
            }
        }
    }

    return [
        'cur_fids' => $cur_fids,
        'ano_fids' => $pre_fids
    ];
}

```

# 화살표 다운

```

public function getSortDown (string $fid) : array
{
    # current fid array
    $cur_fids = [];
    $cur_rlt = $this->db->table($this->getTable())
        ->where($this->getFidColumnName(), 'LIKE-R', $fid)
        ->orderBy( $this->orderBy() )
        ->query();
    while($cur_row = $cur_rlt->fetch_assoc()){
        $cur_fids[] = $cur_row;
    }
    Log::d('cur_fids', $cur_fids);
}

```

```

# > fid array
$next_fids = [];
$depth = $this->getDepthCount($fid);
$fids = explode('.', $fid);
$next_fid = ($depth < 1) ? ($fids[0]+1).".": sprintf("%s%02d", substr($fid,0,-2), ((int)substr($fid,-2) + 1));
Log::d('next_fid ->', $next_fid);
$next_row = $this->db->table($this->getTable())
->select($this->getFidColumnName())
->where(sprintf("%s >= '%s'", $this->getFidColumnName(), $next_fid))
->limit(1)
->orderBy( $this->orderBy() )
->query()->fetch_assoc();
if(isset($next_row[$this->getFidColumnName()]))
{
    # next_fid
    $next_fid = $next_row[$this->getFidColumnName()];
    $next_depth = $this->getDepthCount($next_fid);
    Log::d('next_fid', $next_fid, 'next_depth', $next_depth);

    if($depth == $next_depth)
    {
        # query
        $next_rlt = $this->db->table($this->getTable())
        ->select("")
        ->where($this->getFidColumnName(), 'LIKE-R', $next_fid)
        ->orderBy( $this->orderBy() )
        ->query();
        while($next_row = $next_rlt->fetch_assoc()){
            $next_fids[] = $next_row;
        }
        Log::d('next_fids', $next_fids);
    }
}

return [
    'cur_fids' => $cur_fids,
    'ano_fids' => $next_fids
];
}

```

# 데이터베이스의 fid 값 변경 하기

public function changeSortFid (array \$cur\_fids, array \$ano\_fids, string \$using\_where\_key='\_id') : array

```

{
    $result = [];
    # fid change
    if(count($cur_fids) && count($ano_fids))
    {
        # change cur -> ano
        $ano_root_fid = $ano_fids[0][$this->getFidColumnName()];
        $ano_depth = $this->getDepthCount($ano_root_fid);
        Log::d('ano', $ano_root_fid, $ano_depth);
        $ano_parent_fid = ($ano_depth<1) ? (explode('.', $ano_root_fid))[0].".": $ano_root_fid;

        # db update
        foreach($cur_fids as $cur_fid)
        {
            $this_fid = $cur_fid[$this->getFidColumnName()];
            $cur_depth = $this->getDepthCount($this_fid);
            $cur_update_fid = ($ano_depth == $cur_depth) ? $ano_parent_fid: sprintf("%s%s", $ano_parent_fid, substr($this_fid, ($cur_depth-$ano_depth)*-2));
            Log::d('cur -> ano', $this_fid, '->', $cur_update_fid);
            $result[] = sprintf("cur => ano : %s -> %s", $this_fid, $cur_update_fid);
            try{
                $where = sprintf("%s='%s'", $using_where_key, $cur_fid[$using_where_key]);
                $this->db->beginTransaction();
                $this->db[$this->getFidColumnName()] = $cur_update_fid;
                $this->db->table($this->getTable())->where($where)->update();
                $this->db->commit();
            }catch(\Exception $e){
                $this->db->rollBack();
                Log::e($e->getMessage());
            }
        }

        # change ano -> cur
        $cur_root_fid = $cur_fids[0][$this->getFidColumnName()];
        $cur_depth = $this->getDepthCount($cur_root_fid);
        Log::d('cur', $cur_root_fid, $cur_depth);
        $cur_parent_fid = ($cur_depth<1) ? (explode('.', $cur_root_fid))[0].".": $cur_root_fid;
        # db update
    }
}

```

```

        foreach($ano_fids as $ano_fid)
        {
            $this_fid = $ano_fid[$this->getFidColumnName()];
            $ano_depth = $this->getDepthCount($this_fid);
            $ano_update_fid = ($cur_depth == $ano_depth) ? $cur_parent_fid: sprintf("%s%s", $cur_parent_fid, substr($this_fid, ($ano_depth-$cur_depth)*-2));
            Log::d('ano -> cur', $this_fid, '->', $ano_update_fid);
            $result[] = sprintf("ano => cur : %s -> %s", $this_fid, $ano_update_fid);
            try{
                $this->db->beginTransaction();
                $this->db[$this->getFidColumnName()] = $ano_update_fid;
                $this->db->table($this->getTable())->where(sprintf("%s='%s'", $using_where_key, $ano_fid[$using_where_key]))->update();
                $this->db->commit();
            }catch(Exception $e){
                $this->db->rollBack();
                Log::e($e->getMessage());
            }
        }
    }
    return $result;
}
}

```

--- 파일 경로: traits/ImageCompoessorEditjsTrait.php ---

```

<?php
namespace Flex\Banana\Traits;

# javascript editjs 이미지 내용 찾아 압축하기
# @ FidProviderInterface : required
# @ ImageCompressorBase64Trait : required
trait ImageCompoessorEditjsTrait
{
    public function compressDescriptionBase64Image(array $descriptions, int $width, int $height) : array
    {
        if(is_array($descriptions) && isset($descriptions['blocks'])){
            foreach($descriptions['blocks'] as $idx => $content)
            {
                if($content['type'] == 'image')
                {
                    $base64_image = $content['data']['url'];
                    $resize_base64_image = $this->resizeBase64Image($base64_image, $width, $height);
                    $descriptions['blocks'][$idx]['data']['url'] = $resize_base64_image;
                }
            }
        }

        return $descriptions;
    }

    # @ ImageCompressorBase64Trait : required
    abstract protected function resizeBase64Image(string $base64_image, int $width, int $height) : string;
}

```

--- 파일 경로: traits/ImageCompressorBase64Trait.php ---

```

<?php
namespace Flex\Banana\Traits;

use Flex\Banana\Classes\Log;

# Base64 Image 리사이즈
# @ ImageCompressorProviderInterface : required
# 이 클래스를 사용하려면 반드시 이 이클래스 사용하는 클래스에서 구현하세요
trait ImageCompressorBase64Trait
{
    public function resizeBase64Image(string $base64_image, int $width, int $height) : string
    {
        $result = "";
        try{

```

```

        # ImageGDS class
        $imageGDS = $this->getImageGDS();
        $result = $imageGDS->resizeBase64Image($base64_image, $width, $height);
    } catch (\Exception $e) {
        Log::e($e->getMessage());
    }

    return $result;
}

#@ ImageCompressorProviderInterface : required
abstract protected function getImageGDS();
}

```

--- 파일 경로: traits/NullableValidationTrait.php ---

```

<?php
namespace Flex\Banana\Traits;

use Flex\Banana\Classes\Request\FormValidation as Validation;

trait NullableValidationTrait
{
    /**
     * 공통 optional 유효성 검사 메서드
     *
     * @param string $column_name 컬럼 이름 (예: DB 컬럼명)
     * @param string $column_title 컬럼 제목 (예: 사용자 노출용 이름)
     * @param mixed $data 검증할 데이터
     * @param mixed ...$params 추가 파라미터 ('optional' 또는 '?' 등)
     *
     * @return Validation
     */
    public function checkNullOrOptional(string $column_name, string $column_title, mixed $data, string $optional): Validation
    {
        $validation = new Validation($column_name, $column_title, $data);

        if (!in_array($optional ?? null, ['optional', '?', true])) {
            $validation->null();
        }

        return $validation;
    }
}

```

--- 파일 경로: traits/PasswordHashTrait.php ---

```

<?php
namespace Flex\Banana\Traits;

use Flex\Banana\Classes\Cipher\CipherGeneric;
use Flex\Banana\Classes\Cipher\PasswordHash;

trait PasswordHashTrait
{
    public function hashPassword(string $value): string
    {
        $passwordCipher = new CipherGeneric(new PasswordHash());
        return $passwordCipher->hash($value);
    }
}

```

--- 파일 경로: traits/TimeZoneTrait.php ---

```

<?php

```

```

namespace Flex\Banana\Traits;

use Flex\Banana\Classes\Array\ArrayHelper;
use Flex\Banana\Classes\Date\DateTime;
use \DateTimeZone;

# 날짜 관련 데이터베이스 저장 및 뷰
trait TimeZoneTrait
{
    public function nowInTZ(string $utcgmtime): string
    {
        return (new DateTime("now", $utcgmtime))->format("Y-m-d H:i:s P");
    }

    public function toTZFormat(string $datetimeptz, string $utcgmtime, string $convert_utcgmtime, array $timezone_formats) : ?string
    {
        if(!$datetimeptz){
            return null;
        }

        $dateTimeZ = (new DateTime($datetimeptz, $utcgmtime));
        $dateTimeZ->setTimezone(new DateTimeZone($convert_utcgmtime));
        return $dateTimeZ->format(
            ((new ArrayHelper( $timezone_formats ))->find("timezone",$convert_utcgmtime)->value)['format']
        );
    }
}

```

--- 파일 경로: traits/UniqueldTrait.php ---

```

<?php
namespace Flex\Banana\Traits;

use Flex\Banana\Classes\Uuid\UuidGenerator;

# 각종 토큰 및 id 생성
trait UniqueldTrait
{
    public function genId(): string
    {
        $uuid = (new UuidGenerator())->v7();
        return $uuid;
    }
}

```

--- 파일 경로: utils/Requested.php ---

```

<?php
namespace Flex\Banana\Utils;

use Psr\Http\Message\ServerRequestInterface;

# reactphp ServerRequestInterface 용 확장 클래스
class Requested
{
    public const __version = '1.1.0';
    private array $params = [];

    public function __construct(
        private ServerRequestInterface $request
    ){
        $this->params = [];
    }

    public function post() : Requested
    {
        if ($this->request->getHeaderLine('Content-Type') == 'application/json') {
            $this->params = array_merge($this->params,json_decode($this->request->getBody()->getContents(),true) ?? []);
        }else {
            $this->params = array_merge($this->params,$this->request->getParsedBody());
        }
    }
}

```

```

    }
    return $this;
}

public function get() : Requested
{
    $this->params = $this->request->getQueryParams();
    return $this;
}

public function __call($name, $arguments)
{
    if (str_starts_with($name, 'with')) {
        $newRequest = $this->request->{$name}(...$arguments);
        $clone = clone $this;
        $clone->request = $newRequest;
        return $clone;
    }
    return call_user_func_array([$this->request, $name], $arguments);
}

public function fetch() : array{
    return $this->params;
}

public function __get($propertyName) : mixed
{
    if(isset($this->params[$propertyName])){
        return $this->params[$propertyName];
    }
    return null;
}

public function __set($key, $value){
    $this->params[$key] = $value;
}

public function __isset($name) {
    return isset($this->params[$name]);
}
}

```