<?php

namespace App\Builders;

use Phalcon\Mvc\User\Component;

class Builder extends Component

{

public function objects(array $items)

{

return kg\_array\_object($items);

}

}

<?php

namespace App\Builders;

use App\Models\Category as CategoryModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class CategoryTreeList extends Builder

{

public function handle($type)

{

$topCategories = $this->findTopCategories($type);

if ($topCategories->count() == 0) {

return [];

}

$list = [];

foreach ($topCategories as $category) {

$list[] = [

'id' => $category->id,

'name' => $category->name,

'children' => $this->handleChildren($category),

];

}

return $list;

}

protected function handleChildren(CategoryModel $category)

{

$subCategories = $this->findChildCategories($category->id);

if ($subCategories->count() == 0) {

return [];

}

$list = [];

foreach ($subCategories as $category) {

$list[] = [

'id' => $category->id,

'name' => $category->name,

];

}

return $list;

}

/\*\*

\* @param int $type

\* @return ResultsetInterface|Resultset|CategoryModel[]

\*/

protected function findTopCategories($type)

{

$query = CategoryModel::query();

$query->where('parent\_id = 0');

$query->andWhere('published = 1');

$query->andWhere('type = :type:', ['type' => $type]);

$query->orderBy('priority ASC');

return $query->execute();

}

/\*\*

\* @param int $parentId

\* @return ResultsetInterface|Resultset|CategoryModel[]

\*/

protected function findChildCategories($parentId)

{

$query = CategoryModel::query();

$query->where('published = 1');

$query->andWhere('parent\_id = :parent\_id:', ['parent\_id' => $parentId]);

$query->orderBy('priority ASC');

return $query->execute();

}

}

<?php

namespace App\Builders;

use App\Repos\Chapter as ChapterRepo;

use App\Repos\Course as CourseRepo;

use App\Repos\User as UserRepo;

class ConsultList extends Builder

{

public function handleCourses(array $consults)

{

$courses = $this->getCourses($consults);

foreach ($consults as $key => $consult) {

$consults[$key]['course'] = $courses[$consult['course\_id']] ?? new \stdClass();

}

return $consults;

}

public function handleUsers(array $consults)

{

$users = $this->getUsers($consults);

foreach ($consults as $key => $consult) {

$consults[$key]['owner'] = $users[$consult['owner\_id']] ?? new \stdClass();

$consults[$key]['replier'] = $users[$consult['replier\_id']] ?? new \stdClass();

}

return $consults;

}

public function getCourses(array $consults)

{

$ids = kg\_array\_column($consults, 'course\_id');

$courseRepo = new CourseRepo();

$courses = $courseRepo->findByIds($ids, ['id', 'title']);

$result = [];

foreach ($courses->toArray() as $course) {

$result[$course['id']] = $course;

}

return $result;

}

public function getChapters(array $consults)

{

$ids = kg\_array\_column($consults, 'chapter\_id');

$chapterRepo = new ChapterRepo();

$chapters = $chapterRepo->findByIds($ids, ['id', 'title']);

$result = [];

foreach ($chapters->toArray() as $chapter) {

$result[$chapter['id']] = $chapter;

}

return $result;

}

public function getUsers(array $consults)

{

$ownerIds = kg\_array\_column($consults, 'owner\_id');

$replierIds = kg\_array\_column($consults, 'replier\_id');

$ids = array\_merge($ownerIds, $replierIds);

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name', 'avatar']);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Models\Chapter as ChapterModel;

use App\Models\Course as CourseModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class CourseChapterList extends Builder

{

/\*\*

\* @param int $courseId

\* @return array

\*/

public function handle($courseId)

{

$list = [];

$chapters = $this->findChapters($courseId);

if ($chapters->count() == 0) {

return [];

}

foreach ($chapters as $chapter) {

$list[] = [

'id' => $chapter->id,

'title' => $chapter->title,

'model' => $chapter->model,

'children' => $this->handleChildren($chapter),

];

}

return $list;

}

/\*\*

\* @param ChapterModel $chapter

\* @return array

\*/

protected function handleChildren(ChapterModel $chapter)

{

$lessons = $this->findLessons($chapter->id);

if ($lessons->count() == 0) {

return [];

}

$list = [];

foreach ($lessons as $lesson) {

/\*\*

\* @var $attrs array

\*/

$attrs = $lesson->attrs;

if ($chapter->model == CourseModel::MODEL\_VOD) {

unset($attrs['file\_id'], $attrs['file\_status']);

}

$list[] = [

'id' => $lesson->id,

'title' => $lesson->title,

'model' => $lesson->model,

'free' => $lesson->free,

'attrs' => $attrs,

];

}

return $list;

}

/\*\*

\* @param int $courseId

\* @return ResultsetInterface|Resultset|ChapterModel[]

\*/

protected function findChapters($courseId)

{

return ChapterModel::query()

->where('course\_id = :course\_id:', ['course\_id' => $courseId])

->andWhere('parent\_id = 0 AND deleted = 0')

->orderBy('priority ASC, id ASC')

->execute();

}

/\*\*

\* @param int $chapterId

\* @return ResultsetInterface|Resultset|ChapterModel[]

\*/

protected function findLessons($chapterId)

{

return ChapterModel::query()

->where('parent\_id = :parent\_id:', ['parent\_id' => $chapterId])

->andWhere('deleted = 0')

->orderBy('priority ASC, id ASC')

->execute();

}

}

<?php

namespace App\Builders;

use App\Repos\Course as CourseRepo;

use App\Repos\User as UserRepo;

class CourseFavoriteList extends Builder

{

public function handleCourses(array $relations)

{

$courses = $this->getCourses($relations);

foreach ($relations as $key => $value) {

$relations[$key]['course'] = $courses[$value['course\_id']] ?? new \stdClass();

}

return $relations;

}

public function handleUsers(array $relations)

{

$users = $this->getUsers($relations);

foreach ($relations as $key => $value) {

$relations[$key]['user'] = $users[$value['user\_id']] ?? new \stdClass();

}

return $relations;

}

public function getCourses(array $relations)

{

$ids = kg\_array\_column($relations, 'course\_id');

$courseRepo = new CourseRepo();

$columns = [

'id', 'title', 'cover',

'market\_price', 'vip\_price',

'rating', 'model', 'level', 'attrs',

'user\_count', 'lesson\_count', 'review\_count', 'favorite\_count',

];

$courses = $courseRepo->findByIds($ids, $columns);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($courses->toArray() as $course) {

$course['cover'] = $baseUrl . $course['cover'];

$course['market\_price'] = (float)$course['market\_price'];

$course['vip\_price'] = (float)$course['vip\_price'];

$course['rating'] = (float)$course['rating'];

$course['attrs'] = json\_decode($course['attrs'], true);

$result[$course['id']] = $course;

}

return $result;

}

public function getUsers(array $relations)

{

$ids = kg\_array\_column($relations, 'user\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name', 'avatar']);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Caches\CategoryList as CategoryListCache;

use App\Models\Category as CategoryModel;

use App\Repos\User as UserRepo;

class CourseList extends Builder

{

public function handleCategories(array $courses)

{

$categories = $this->getCategories();

foreach ($courses as $key => $course) {

$courses[$key]['category'] = $categories[$course['category\_id']] ?? new \stdClass();

}

return $courses;

}

public function handleTeachers(array $courses)

{

$teachers = $this->getTeachers($courses);

foreach ($courses as $key => $course) {

$courses[$key]['teacher'] = $teachers[$course['teacher\_id']] ?? new \stdClass();

}

return $courses;

}

public function getCategories()

{

$cache = new CategoryListCache();

$items = $cache->get(CategoryModel::TYPE\_COURSE);

if (empty($items)) {

return [];

}

$result = [];

foreach ($items as $item) {

$result[$item['id']] = [

'id' => $item['id'],

'name' => $item['name'],

];

}

return $result;

}

public function getTeachers($courses)

{

$ids = kg\_array\_column($courses, 'teacher\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name', 'avatar']);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\Course as CourseRepo;

use App\Repos\Topic as TopicRepo;

class CourseTopicList extends Builder

{

public function handleCourses($relations)

{

$courses = $this->getCourses($relations);

foreach ($relations as $key => $value) {

$relations[$key]['course'] = $courses[$value['course\_id']] ?? new \stdClass();

}

return $relations;

}

public function handleTopics($relations)

{

$topics = $this->getTopics($relations);

foreach ($relations as $key => $value) {

$relations[$key]['topic'] = $topics[$value['topic\_id']] ?? new \stdClass();

}

return $relations;

}

public function getCourses($relations)

{

$ids = kg\_array\_column($relations, 'course\_id');

$courseRepo = new CourseRepo();

$columns = [

'id', 'title', 'cover',

'market\_price', 'vip\_price',

'rating', 'model', 'level', 'attrs',

'user\_count', 'lesson\_count',

];

$courses = $courseRepo->findByIds($ids, $columns);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($courses->toArray() as $course) {

$course['cover'] = $baseUrl . $course['cover'];

$course['attrs'] = json\_decode($course['attrs'], true);

$result[$course['id']] = $course;

}

return $result;

}

public function getTopics($relations)

{

$ids = kg\_array\_column($relations, 'topic\_id');

$topicRepo = new TopicRepo();

$topics = $topicRepo->findByIds($ids, ['id', 'title', 'summary']);

$result = [];

foreach ($topics->toArray() as $topic) {

$result[$topic['id']] = $topic;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\Course as CourseRepo;

use App\Repos\User as UserRepo;

class CourseUserList extends Builder

{

public function handleCourses($relations)

{

$courses = $this->getCourses($relations);

foreach ($relations as $key => $value) {

$relations[$key]['course'] = $courses[$value['course\_id']] ?? new \stdClass();

}

return $relations;

}

public function handleUsers($relations)

{

$users = $this->getUsers($relations);

foreach ($relations as $key => $value) {

$relations[$key]['user'] = $users[$value['user\_id']] ?? new \stdClass();

}

return $relations;

}

public function getCourses($relations)

{

$ids = kg\_array\_column($relations, 'course\_id');

$courseRepo = new CourseRepo();

$columns = [

'id', 'title', 'cover',

'market\_price', 'vip\_price',

'rating', 'model', 'level', 'attrs',

'user\_count', 'lesson\_count', 'review\_count', 'favorite\_count',

];

$courses = $courseRepo->findByIds($ids, $columns);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($courses->toArray() as $course) {

$course['cover'] = $baseUrl . $course['cover'];

$course['attrs'] = json\_decode($course['attrs'], true);

$result[$course['id']] = $course;

}

return $result;

}

public function getUsers($relations)

{

$ids = kg\_array\_column($relations, 'user\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name', 'avatar']);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\Chapter as ChapterRepo;

use App\Repos\Course as CourseRepo;

use App\Repos\User as UserRepo;

class DanmuList extends Builder

{

public function handleCourses(array $danmus)

{

$courses = $this->getCourses($danmus);

foreach ($danmus as $key => $danmu) {

$danmus[$key]['course'] = $courses[$danmu['course\_id']] ?? new \stdClass();

}

return $danmus;

}

public function handleChapters(array $danmus)

{

$chapters = $this->getChapters($danmus);

foreach ($danmus as $key => $danmu) {

$danmus[$key]['chapter'] = $chapters[$danmu['chapter\_id']] ?? new \stdClass();

}

return $danmus;

}

public function handleUsers(array $danmus)

{

$users = $this->getUsers($danmus);

foreach ($danmus as $key => $danmu) {

$danmus[$key]['owner'] = $users[$danmu['owner\_id']] ?? new \stdClass();

}

return $danmus;

}

public function getCourses(array $danmus)

{

$ids = kg\_array\_column($danmus, 'course\_id');

$courseRepo = new CourseRepo();

$courses = $courseRepo->findByIds($ids, ['id', 'title']);

$result = [];

foreach ($courses->toArray() as $course) {

$result[$course['id']] = $course;

}

return $result;

}

public function getChapters(array $danmus)

{

$ids = kg\_array\_column($danmus, 'chapter\_id');

$chapterRepo = new ChapterRepo();

$chapters = $chapterRepo->findByIds($ids, ['id', 'title']);

$result = [];

foreach ($chapters->toArray() as $chapter) {

$result[$chapter['id']] = $chapter;

}

return $result;

}

public function getUsers(array $danmus)

{

$ids = kg\_array\_column($danmus, 'owner\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name', 'avatar']);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Caches\CategoryList as CategoryListCache;

use App\Models\Category as CategoryModel;

class HelpList extends Builder

{

public function handleCategories(array $helps)

{

$categories = $this->getCategories();

foreach ($helps as $key => $help) {

$helps[$key]['category'] = $categories[$help['category\_id']] ?? new \stdClass();

}

return $helps;

}

public function getCategories()

{

$cache = new CategoryListCache();

$items = $cache->get(CategoryModel::TYPE\_HELP);

if (empty($items)) {

return [];

}

$result = [];

foreach ($items as $item) {

$result[$item['id']] = [

'id' => $item['id'],

'name' => $item['name'],

];

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\User as UserRepo;

class ImFriendUserList extends Builder

{

public function handleFriends(array $relations)

{

$users = $this->getFriends($relations);

foreach ($relations as $key => $value) {

$relations[$key]['friend'] = $users[$value['friend\_id']] ?? new \stdClass();

}

return $relations;

}

public function getFriends(array $relations)

{

$ids = kg\_array\_column($relations, 'friend\_id');

$userRepo = new UserRepo();

$columns = [

'id', 'name', 'avatar', 'title', 'about', 'vip',

'gender', 'area', 'active\_time',

];

$users = $userRepo->findByIds($ids, $columns);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\Course as CourseRepo;

use App\Repos\User as UserRepo;

class ImGroupList extends Builder

{

public function handleCourses(array $groups)

{

$courses = $this->getCourses($groups);

foreach ($groups as $key => $group) {

$groups[$key]['course'] = $courses[$group['course\_id']] ?? new \stdClass();

}

return $groups;

}

public function handleUsers(array $groups)

{

$users = $this->getUsers($groups);

foreach ($groups as $key => $group) {

$groups[$key]['owner'] = $users[$group['owner\_id']] ?? new \stdClass();

}

return $groups;

}

public function getCourses(array $groups)

{

$ids = kg\_array\_column($groups, 'course\_id');

$courseRepo = new CourseRepo();

$courses = $courseRepo->findByIds($ids, ['id', 'title']);

$result = [];

foreach ($courses->toArray() as $course) {

$result[$course['id']] = $course;

}

return $result;

}

public function getUsers(array $groups)

{

$ids = kg\_array\_column($groups, 'owner\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name', 'avatar']);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\ImGroup as ImGroupRepo;

use App\Repos\User as UserRepo;

class ImGroupUserList extends Builder

{

public function handleGroups(array $relations)

{

$groups = $this->getGroups($relations);

foreach ($relations as $key => $value) {

$relations[$key]['group'] = $groups[$value['group\_id']] ?? new \stdClass();

}

return $relations;

}

public function handleUsers(array $relations)

{

$users = $this->getUsers($relations);

foreach ($relations as $key => $value) {

$relations[$key]['user'] = $users[$value['user\_id']] ?? new \stdClass();

}

return $relations;

}

public function getUsers(array $relations)

{

$ids = kg\_array\_column($relations, 'user\_id');

$userRepo = new UserRepo();

$columns = ['id', 'name', 'avatar', 'title', 'about', 'vip', 'gender', 'area'];

$users = $userRepo->findByIds($ids, $columns);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$result[$user['id']] = $user;

}

return $result;

}

public function getGroups(array $relations)

{

$ids = kg\_array\_column($relations, 'group\_id');

$groupRepo = new ImGroupRepo();

$columns = ['id', 'type', 'name', 'avatar', 'about', 'owner\_id', 'user\_count', 'msg\_count'];

$groups = $groupRepo->findByIds($ids, $columns);

$users = $this->getGroupOwners($groups->toArray());

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($groups->toArray() as $group) {

$group['avatar'] = $baseUrl . $group['avatar'];

$group['owner'] = $users[$group['owner\_id']] ?? new \stdClass();

unset($group['owner\_id']);

$result[$group['id']] = $group;

}

return $result;

}

protected function getGroupOwners(array $groups)

{

$ids = kg\_array\_column($groups, 'owner\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name']);

$result = [];

if ($users->count() > 0) {

foreach ($users->toArray() as $user) {

$result[$user['id']] = $user;

}

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\User as UserRepo;

class ImMessageList extends Builder

{

public function handleSenders(array $messages)

{

$users = $this->getSenders($messages);

foreach ($messages as $key => $message) {

$messages[$key]['sender'] = $users[$message['sender\_id']] ?? new \stdClass();

}

return $messages;

}

public function getSenders(array $messages)

{

$ids = kg\_array\_column($messages, 'sender\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name', 'avatar']);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\Chapter as ChapterRepo;

use App\Repos\Course as CourseRepo;

use App\Repos\User as UserRepo;

class LearningList extends Builder

{

public function handleCourses($relations)

{

$courses = $this->getCourses($relations);

foreach ($relations as $key => $value) {

$relations[$key]['course'] = $courses[$value['course\_id']] ?? new \stdClass();

}

return $relations;

}

public function handleChapters($relations)

{

$chapters = $this->getChapters($relations);

foreach ($relations as $key => $value) {

$relations[$key]['chapter'] = $chapters[$value['chapter\_id']] ?? new \stdClass();

}

return $relations;

}

public function handleUsers($relations)

{

$users = $this->getUsers($relations);

foreach ($relations as $key => $value) {

$relations[$key]['user'] = $users[$value['user\_id']] ?? new \stdClass();

}

return $relations;

}

protected function getCourses($relations)

{

$ids = kg\_array\_column($relations, 'course\_id');

$courseRepo = new CourseRepo();

$courses = $courseRepo->findByIds($ids, ['id', 'title']);

$result = [];

foreach ($courses->toArray() as $course) {

$result[$course['id']] = $course;

}

return $result;

}

protected function getChapters($relations)

{

$ids = kg\_array\_column($relations, 'chapter\_id');

$chapterRepo = new ChapterRepo();

$chapters = $chapterRepo->findByIds($ids, ['id', 'title']);

$result = [];

foreach ($chapters->toArray() as $chapter) {

$result[$chapter['id']] = $chapter;

}

return $result;

}

protected function getUsers($relations)

{

$ids = kg\_array\_column($relations, 'user\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name']);

$result = [];

foreach ($users->toArray() as $user) {

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\Chapter as ChapterRepo;

use App\Repos\Course as CourseRepo;

use App\Repos\User as UserRepo;

class LiveList extends Builder

{

public function handleCourses(array $lives)

{

$courses = $this->getCourses($lives);

foreach ($lives as $key => $live) {

$lives[$key]['course'] = $courses[$live['course\_id']] ?? new \stdClass();

}

return $lives;

}

public function handleChapters(array $lives)

{

$chapters = $this->getChapters($lives);

foreach ($lives as $key => $live) {

$lives[$key]['chapter'] = $chapters[$live['chapter\_id']] ?? new \stdClass();

}

return $lives;

}

public function getCourses(array $lives)

{

$courseIds = kg\_array\_column($lives, 'course\_id');

$courseRepo = new CourseRepo();

$courses = $courseRepo->findByIds($courseIds, ['id', 'title', 'cover', 'teacher\_id']);

$teacherIds = kg\_array\_column($courses->toArray(), 'teacher\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($teacherIds, ['id', 'name', 'title', 'avatar', 'about']);

$baseUrl = kg\_cos\_url();

$teachers = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$teachers[$user['id']] = $user;

}

$result = [];

foreach ($courses->toArray() as $course) {

$course['cover'] = $baseUrl . $course['cover'];

$course['teacher'] = $teachers[$course['teacher\_id']] ?? new \stdClass();

$result[$course['id']] = [

'id' => $course['id'],

'title' => $course['title'],

'cover' => $course['cover'],

'teacher' => $course['teacher'],

];

}

return $result;

}

public function getChapters(array $lives)

{

$ids = kg\_array\_column($lives, 'chapter\_id');

$chapterRepo = new ChapterRepo();

$chapters = $chapterRepo->findByIds($ids, ['id', 'title']);

$result = [];

foreach ($chapters->toArray() as $chapter) {

$result[$chapter['id']] = $chapter;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Models\Nav as NavModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class NavTreeList extends Builder

{

public function handle($position = 'top')

{

$topNavs = $this->findTopNavs($position);

if ($topNavs->count() == 0) {

return [];

}

$list = [];

foreach ($topNavs as $nav) {

$list[] = [

'id' => $nav->id,

'name' => $nav->name,

'target' => $nav->target,

'url' => $nav->url,

'children' => $this->handleChildren($nav),

];

}

return $list;

}

protected function handleChildren(NavModel $nav)

{

$subNavs = $this->findSubNavs($nav->id);

if ($subNavs->count() == 0) {

return [];

}

$list = [];

foreach ($subNavs as $nav) {

$list[] = [

'id' => $nav->id,

'name' => $nav->name,

'target' => $nav->target,

'url' => $nav->url,

];

}

return $list;

}

/\*\*

\* @param int $navId

\* @return ResultsetInterface|Resultset|NavModel[]

\*/

protected function findSubNavs($navId)

{

return NavModel::query()

->where('parent\_id = :parent\_id:', ['parent\_id' => $navId])

->andWhere('published = 1')

->orderBy('priority ASC')

->execute();

}

/\*\*

\* @param string $position

\* @return ResultsetInterface|Resultset|NavModel[]

\*/

protected function findTopNavs($position)

{

return NavModel::query()

->where('position = :position:', ['position' => $position])

->andWhere('level = 1 AND published = 1')

->orderBy('priority ASC')

->execute();

}

}

<?php

namespace App\Builders;

use App\Models\Order as OrderModel;

use App\Repos\User as UserRepo;

class OrderList extends Builder

{

protected $imgBaseUrl;

public function \_\_construct()

{

$this->imgBaseUrl = kg\_cos\_url();

}

/\*\*

\* @param array $orders

\* @return array

\*/

public function handleUsers(array $orders)

{

$users = $this->getUsers($orders);

foreach ($orders as $key => $order) {

$orders[$key]['owner'] = $users[$order['owner\_id']] ?? new \stdClass();

}

return $orders;

}

/\*\*

\* @param array $orders

\* @return array

\*/

public function handleItems(array $orders)

{

foreach ($orders as $key => $order) {

$itemInfo = $this->handleItem($order);

$orders[$key]['item\_info'] = $itemInfo;

}

return $orders;

}

/\*\*

\* @param array $order

\* @return array|mixed

\*/

public function handleItem(array $order)

{

$itemInfo = [];

switch ($order['item\_type']) {

case OrderModel::ITEM\_COURSE:

$itemInfo = $this->handleCourseInfo($order['item\_info']);

break;

case OrderModel::ITEM\_PACKAGE:

$itemInfo = $this->handlePackageInfo($order['item\_info']);

break;

case OrderModel::ITEM\_VIP:

$itemInfo = $this->handleVipInfo($order['item\_info']);

break;

}

return $itemInfo;

}

/\*\*

\* @param string $itemInfo

\* @return mixed

\*/

protected function handleCourseInfo($itemInfo)

{

if (!empty($itemInfo) && is\_string($itemInfo)) {

$itemInfo = json\_decode($itemInfo, true);

$itemInfo['course']['cover'] = $this->imgBaseUrl . $itemInfo['course']['cover'];

}

return $itemInfo;

}

/\*\*

\* @param string $itemInfo

\* @return mixed

\*/

protected function handlePackageInfo($itemInfo)

{

if (!empty($itemInfo) && is\_string($itemInfo)) {

$itemInfo = json\_decode($itemInfo, true);

foreach ($itemInfo['courses'] as $key => $course) {

$itemInfo['courses'][$key]['cover'] = $this->imgBaseUrl . $course['cover'];

}

}

return $itemInfo;

}

/\*\*

\* @param string $itemInfo

\* @return mixed

\*/

protected function handleRewardInfo($itemInfo)

{

if (!empty($itemInfo) && is\_string($itemInfo)) {

$itemInfo = json\_decode($itemInfo, true);

$itemInfo['course']['cover'] = $this->imgBaseUrl . $itemInfo['course']['cover'];

}

return $itemInfo;

}

/\*\*

\* @param string $itemInfo

\* @return mixed

\*/

protected function handleVipInfo($itemInfo)

{

if (!empty($itemInfo) && is\_string($itemInfo)) {

$itemInfo = json\_decode($itemInfo, true);

}

return $itemInfo;

}

/\*\*

\* @param array $orders

\* @return array

\*/

protected function getUsers(array $orders)

{

$ids = kg\_array\_column($orders, 'owner\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name']);

$result = [];

foreach ($users->toArray() as $user) {

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\Order as OrderRepo;

use App\Repos\User as UserRepo;

class RefundList extends Builder

{

public function handleOrders(array $trades)

{

$orders = $this->getOrders($trades);

foreach ($trades as $key => $trade) {

$trades[$key]['order'] = $orders[$trade['order\_id']] ?? new \stdClass();

}

return $trades;

}

public function handleUsers(array $refunds)

{

$users = $this->getUsers($refunds);

foreach ($refunds as $key => $refund) {

$refunds[$key]['owner'] = $users[$refund['owner\_id']] ?? new \stdClass();

}

return $refunds;

}

public function getOrders(array $trades)

{

$ids = kg\_array\_column($trades, 'order\_id');

$orderRepo = new OrderRepo();

$orders = $orderRepo->findByIds($ids, ['id', 'sn', 'subject', 'amount']);

$result = [];

foreach ($orders->toArray() as $order) {

$result[$order['id']] = $order;

}

return $result;

}

public function getUsers(array $refunds)

{

$ids = kg\_array\_column($refunds, 'owner\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name']);

$result = [];

foreach ($users->toArray() as $user) {

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\Upload as UploadRepo;

class ResourceList extends Builder

{

public function handleUploads($relations)

{

$uploads = $this->getUploads($relations);

foreach ($relations as $key => $value) {

$relations[$key]['upload'] = $uploads[$value['upload\_id']] ?? new \stdClass();

}

return $relations;

}

public function getUploads($relations)

{

$ids = kg\_array\_column($relations, 'upload\_id');

$uploadRepo = new UploadRepo();

$columns = ['id', 'name', 'path', 'mime', 'md5', 'size'];

$uploads = $uploadRepo->findByIds($ids, $columns);

$result = [];

foreach ($uploads->toArray() as $upload) {

$result[$upload['id']] = $upload;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\Course as CourseRepo;

use App\Repos\User as UserRepo;

class ReviewList extends Builder

{

public function handleCourses(array $reviews)

{

$courses = $this->getCourses($reviews);

foreach ($reviews as $key => $review) {

$reviews[$key]['course'] = $courses[$review['course\_id']] ?? new \stdClass();

}

return $reviews;

}

public function handleUsers(array $reviews)

{

$users = $this->getUsers($reviews);

foreach ($reviews as $key => $review) {

$reviews[$key]['owner'] = $users[$review['owner\_id']] ?? new \stdClass();

}

return $reviews;

}

public function getCourses(array $reviews)

{

$ids = kg\_array\_column($reviews, 'course\_id');

$courseRepo = new CourseRepo();

$courses = $courseRepo->findByIds($ids, ['id', 'title']);

$result = [];

foreach ($courses->toArray() as $course) {

$result[$course['id']] = $course;

}

return $result;

}

public function getUsers(array $reviews)

{

$ids = kg\_array\_column($reviews, 'owner\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name', 'avatar']);

$baseUrl = kg\_cos\_url();

$result = [];

foreach ($users->toArray() as $user) {

$user['avatar'] = $baseUrl . $user['avatar'];

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Repos\Order as OrderRepo;

use App\Repos\User as UserRepo;

class TradeList extends Builder

{

public function handleOrders(array $trades)

{

$orders = $this->getOrders($trades);

foreach ($trades as $key => $trade) {

$trades[$key]['order'] = $orders[$trade['order\_id']] ?? new \stdClass();

}

return $trades;

}

public function handleUsers($trades)

{

$users = $this->getUsers($trades);

foreach ($trades as $key => $trade) {

$trades[$key]['owner'] = $users[$trade['owner\_id']] ?? new \stdClass();

}

return $trades;

}

public function getOrders($trades)

{

$ids = kg\_array\_column($trades, 'order\_id');

$orderRepo = new OrderRepo();

$orders = $orderRepo->findByIds($ids, ['id', 'sn', 'subject', 'amount']);

$result = [];

foreach ($orders->toArray() as $order) {

$result[$order['id']] = $order;

}

return $result;

}

public function getUsers($trades)

{

$ids = kg\_array\_column($trades, 'owner\_id');

$userRepo = new UserRepo();

$users = $userRepo->findByIds($ids, ['id', 'name']);

$result = [];

foreach ($users->toArray() as $user) {

$result[$user['id']] = $user;

}

return $result;

}

}

<?php

namespace App\Builders;

use App\Models\User as UserModel;

use App\Repos\Role as RoleRepo;

class UserList extends Builder

{

public function handleUsers(array $users)

{

$baseUrl = kg\_cos\_url();

foreach ($users as $key => $user) {

$users[$key]['avatar'] = $baseUrl . $user['avatar'];

}

return $users;

}

public function handleAdminRoles(array $users)

{

$roles = $this->getAdminRoles($users);

foreach ($users as $key => $user) {

$users[$key]['admin\_role'] = $roles[$user['admin\_role']] ?? ['id' => 0, 'name' => 'N/A'];

}

return $users;

}

public function handleEduRoles(array $users)

{

$roles = $this->getEduRoles();

foreach ($users as $key => $user) {

$users[$key]['edu\_role'] = $roles[$user['edu\_role']] ?? ['id' => 0, 'name' => 'N/A'];

}

return $users;

}

protected function getAdminRoles(array $users)

{

$ids = kg\_array\_column($users, 'admin\_role');

$roleRepo = new RoleRepo();

$roles = $roleRepo->findByIds($ids, ['id', 'name']);

$result = [];

foreach ($roles->toArray() as $role) {

$result[$role['id']] = $role;

}

return $result;

}

protected function getEduRoles()

{

return [

UserModel::EDU\_ROLE\_STUDENT => [

'id' => UserModel::EDU\_ROLE\_STUDENT,

'name' => '学员',

],

UserModel::EDU\_ROLE\_TEACHER => [

'id' => UserModel::EDU\_ROLE\_TEACHER,

'name' => '讲师',

],

];

}

}

<?php

namespace App\Caches;

use App\Repos\App as AppRepo;

class App extends Cache

{

protected $lifetime = 365 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "app:{$id}";

}

public function getContent($id = null)

{

$appRepo = new AppRepo();

$result = $appRepo->findByAppKey($id);

return $result ?: null;

}

}

<?php

namespace App\Caches;

use Phalcon\Cache\Backend\Redis as RedisCache;

use Phalcon\Mvc\User\Component;

abstract class Cache extends Component

{

/\*\*

\* @var RedisCache

\*/

protected $cache;

public function \_\_construct()

{

$this->cache = $this->getDI()->getShared('cache');

}

/\*\*

\* 获取缓存内容

\*

\* @param mixed $id

\* @return mixed

\*/

public function get($id = null)

{

$key = $this->getKey($id);

if (!$this->cache->exists($key)) {

$content = $this->getContent($id);

$lifetime = $this->getLifetime();

$this->cache->save($key, $content, $lifetime);

} else {

$content = $this->cache->get($key);

}

return $content;

}

/\*\*

\* 删除缓存内容

\*

\* @param mixed $id

\*/

public function delete($id = null)

{

$key = $this->getKey($id);

$this->cache->delete($key);

}

/\*\*

\* 重建缓存内容

\*

\* @param mixed $id

\*/

public function rebuild($id = null)

{

$this->delete($id);

$this->get($id);

}

/\*\*

\* 获取缓存有效期

\*

\* @return int

\*/

abstract public function getLifetime();

/\*\*

\* 获取键值

\*

\* @param mixed $id

\* @return string

\*/

abstract public function getKey($id = null);

/\*\*

\* 获取原始内容

\*

\* @param mixed $id

\* @return mixed

\*/

abstract public function getContent($id = null);

}

<?php

namespace App\Caches;

use App\Repos\Category as CategoryRepo;

class Category extends Cache

{

protected $lifetime = 365 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "category:{$id}";

}

public function getContent($id = null)

{

$categoryRepo = new CategoryRepo();

$category = $categoryRepo->findById($id);

return $category ?: null;

}

}

<?php

namespace App\Caches;

use App\Models\Category as CategoryModel;

use Phalcon\Mvc\Model\Resultset;

class CategoryList extends Cache

{

protected $lifetime = 365 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($type = null)

{

return "category\_list:{$type}";

}

/\*\*

\* @param null $type

\* @return array

\*/

public function getContent($type = null)

{

/\*\*

\* @var Resultset $categories

\*/

$categories = CategoryModel::query()

->columns(['id', 'parent\_id', 'name', 'priority', 'level', 'path'])

->where('type = :type:', ['type' => $type])

->andWhere('published = 1')

->execute();

if ($categories->count() == 0) {

return [];

}

return $categories->toArray();

}

}

<?php

namespace App\Caches;

use App\Builders\CategoryTreeList as CategoryTreeListBuilder;

class CategoryTreeList extends Cache

{

protected $lifetime = 365 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($type = null)

{

return "category\_tree\_list:{$type}";

}

public function getContent($type = null)

{

$builder = new CategoryTreeListBuilder();

$list = $builder->handle($type);

return $list ?: [];

}

}

<?php

namespace App\Caches;

use App\Repos\Chapter as ChapterRepo;

class Chapter extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "chapter:{$id}";

}

public function getContent($id = null)

{

$chapterRepo = new ChapterRepo();

$chapter = $chapterRepo->findById($id);

return $chapter ?: null;

}

}

<?php

namespace App\Caches;

use App\Library\Cache\Backend\Redis as RedisCache;

use Phalcon\Mvc\User\Component;

abstract class Counter extends Component

{

/\*\*

\* @var RedisCache

\*/

protected $cache;

/\*\*

\* @var \Redis

\*/

protected $redis;

public function \_\_construct()

{

$this->cache = $this->getDI()->getShared('cache');

$this->redis = $this->cache->getRedis();

}

/\*\*

\* 获取缓存内容

\*

\* @param mixed $id

\* @return array

\*/

public function get($id = null)

{

$key = $this->getKey($id);

$content = $this->redis->hGetAll($key);

if (!$this->cache->exists($key)) {

$content = $this->getContent($id);

$lifetime = $this->getLifetime();

$this->redis->hMSet($key, $content);

$this->redis->expire($key, $lifetime);

}

return $content;

}

/\*\*

\* 删除缓存内容

\*

\* @param mixed $id

\*/

public function delete($id = null)

{

$key = $this->getKey($id);

$this->cache->delete($key);

}

/\*\*

\* 重建缓存内容

\*

\* @param mixed $id

\*/

public function rebuild($id = null)

{

$this->delete($id);

$this->get($id);

}

public function hGet($id, $hashKey)

{

$key = $this->getKey($id);

if (!$this->redis->exists($key)) {

$this->get($id);

}

return $this->redis->hGet($key, $hashKey);

}

public function hDel($id, $hashKey)

{

$key = $this->getKey($id);

return $this->redis->hDel($key, $hashKey);

}

public function hIncrBy($id, $hashKey, $value = 1)

{

$key = $this->getKey($id);

if (!$this->redis->exists($key)) {

$this->get($id);

}

$this->redis->hIncrBy($key, $hashKey, $value);

}

public function hDecrBy($id, $hashKey, $value = 1)

{

$key = $this->getKey($id);

if (!$this->redis->exists($key)) {

$this->get($id);

}

$this->redis->hIncrBy($key, $hashKey, 0 - $value);

}

/\*\*

\* 获取缓存有效期

\*

\* @return int

\*/

abstract public function getLifetime();

/\*\*

\* 获取键值

\*

\* @param mixed $id

\* @return string

\*/

abstract public function getKey($id = null);

/\*\*

\* 获取原始内容

\*

\* @param mixed $id

\* @return mixed

\*/

abstract public function getContent($id = null);

}

<?php

namespace App\Caches;

use App\Repos\Course as CourseRepo;

class Course extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "course:{$id}";

}

public function getContent($id = null)

{

$courseRepo = new CourseRepo();

$course = $courseRepo->findById($id);

return $course ?: null;

}

}

<?php

namespace App\Caches;

use App\Models\Category as CategoryModel;

use App\Repos\Course as CourseRepo;

class CourseCategoryList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "course\_category\_list:{$id}";

}

public function getContent($id = null)

{

$courseRepo = new CourseRepo();

$categories = $courseRepo->findCategories($id);

if ($categories->count() == 0) {

return [];

}

return $this->handleContent($categories);

}

/\*\*

\* @param CategoryModel[] $categories

\* @return array

\*/

public function handleContent($categories)

{

$result = [];

foreach ($categories as $category) {

$result[] = [

'id' => $category->id,

'name' => $category->name,

];

}

return $result;

}

}

<?php

namespace App\Caches;

use App\Builders\CourseChapterList as CourseChapterListBuilder;

class CourseChapterList extends Cache

{

protected $lifetime = 7 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "course\_chapter\_list:{$id}";

}

public function getContent($id = null)

{

$builder = new CourseChapterListBuilder();

$list = $builder->handle($id);

return $list ?: [];

}

}

<?php

namespace App\Caches;

use App\Models\Package as PackageModel;

use App\Repos\Course as CourseRepo;

class CoursePackageList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "course\_package\_list:{$id}";

}

public function getContent($id = null)

{

$courseRepo = new CourseRepo();

$packages = $courseRepo->findPackages($id);

if ($packages->count() == 0) {

return [];

}

return $this->handleContent($packages);

}

/\*\*

\* @param PackageModel[] $packages

\* @return array

\*/

protected function handleContent($packages)

{

$result = [];

foreach ($packages as $package) {

$result[] = [

'id' => $package->id,

'title' => $package->title,

'course\_count' => $package->course\_count,

'market\_price' => $package->market\_price,

'vip\_price' => $package->vip\_price,

];

}

return $result;

}

}

<?php

namespace App\Caches;

use App\Models\Course as CourseModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class CourseRecommendedList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "course\_recommended\_list:{$id}";

}

public function getContent($id = null)

{

$courses = $this->findCourses(5);

if ($courses->count() == 0) {

return [];

}

return $this->handleContent($courses);

}

/\*\*

\* @param CourseModel[] $courses

\* @return array

\*/

public function handleContent($courses)

{

$result = [];

foreach ($courses as $course) {

$result[] = [

'id' => $course->id,

'title' => $course->title,

'cover' => $course->cover,

'market\_price' => $course->market\_price,

'vip\_price' => $course->vip\_price,

'model' => $course->model,

'level' => $course->level,

'user\_count' => $course->user\_count,

'lesson\_count' => $course->lesson\_count,

];

}

return $result;

}

/\*\*

\* @param int $limit

\* @return ResultsetInterface|Resultset|CourseModel[]

\*/

public function findCourses($limit = 5)

{

return CourseModel::query()

->where('published = 1 AND market\_price > 0')

->orderBy('RAND()')

->limit($limit)

->execute();

}

}

<?php

namespace App\Caches;

use App\Models\Course as CourseModel;

use App\Repos\Course as CourseRepo;

class CourseRelatedList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "course\_related\_list:{$id}";

}

public function getContent($id = null)

{

$courseRepo = new CourseRepo();

$courses = $courseRepo->findRelatedCourses($id);

if ($courses->count() == 0) {

return [];

}

return $this->handleContent($courses);

}

/\*\*

\* @param CourseModel[] $courses

\* @return array

\*/

public function handleContent($courses)

{

$result = [];

foreach ($courses as $course) {

$result[] = [

'id' => $course->id,

'title' => $course->title,

'cover' => $course->cover,

'market\_price' => $course->market\_price,

'vip\_price' => $course->vip\_price,

'model' => $course->model,

'level' => $course->level,

'user\_count' => $course->user\_count,

'lesson\_count' => $course->lesson\_count,

];

}

return $result;

}

}

<?php

namespace App\Caches;

use App\Models\User as UserModel;

use App\Repos\Course as CourseRepo;

class CourseTeacherList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "course\_teacher\_list:{$id}";

}

public function getContent($id = null)

{

$courseRepo = new CourseRepo();

$users = $courseRepo->findTeachers($id);

if ($users->count() == 0) {

return [];

}

return $this->handleContent($users);

}

/\*\*

\* @param UserModel[] $users

\* @return array

\*/

public function handleContent($users)

{

$result = [];

foreach ($users as $user) {

$result[] = [

'id' => $user->id,

'name' => $user->name,

'avatar' => $user->avatar,

'title' => $user->title,

'about' => $user->about,

];

}

return $result;

}

}

<?php

namespace App\Caches;

use App\Models\Topic as TopicModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class CourseTopicList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "course\_topic\_list:{$id}";

}

public function getContent($id = null)

{

$topics = $this->findTopics(5);

if ($topics->count() == 0) {

return [];

}

return $this->handleContent($topics);

}

/\*\*

\* @param TopicModel[] $topics

\* @return array

\*/

public function handleContent($topics)

{

$result = [];

foreach ($topics as $topic) {

$result[] = [

'id' => $topic->id,

'title' => $topic->title,

];

}

return $result;

}

/\*\*

\* @param int $limit

\* @return ResultsetInterface|Resultset|TopicModel[]

\*/

public function findTopics($limit = 5)

{

return TopicModel::query()

->where('published = 1')

->orderBy('RAND()')

->limit($limit)

->execute();

}

}

<?php

namespace App\Caches;

use App\Repos\Help as HelpRepo;

class Help extends Cache

{

protected $lifetime = 7 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "help:{$id}";

}

public function getContent($id = null)

{

$helpRepo = new HelpRepo();

$help = $helpRepo->findById($id);

return $help ?: null;

}

}

<?php

namespace App\Caches;

use App\Models\Category as CategoryModel;

use App\Models\Help as HelpModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class HelpList extends Cache

{

protected $lifetime = 365 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'help\_list';

}

public function getContent($id = null)

{

$categories = $this->findCategories();

if ($categories->count() == 0) {

return [];

}

$result = [];

foreach ($categories as $category) {

$item = [];

$item['category'] = [

'id' => $category->id,

'name' => $category->name,

];

$item['helps'] = [];

$helps = $this->findHelps($category->id);

if ($helps->count() > 0) {

foreach ($helps as $help) {

$item['helps'][] = [

'id' => $help->id,

'title' => $help->title,

];

}

}

$result[] = $item;

}

return $result;

}

/\*\*

\* @return ResultsetInterface|Resultset|CategoryModel[]

\*/

protected function findCategories()

{

return CategoryModel::query()

->where('type = :type:', ['type' => CategoryModel::TYPE\_HELP])

->andWhere('level = 1 AND published = 1')

->orderBy('priority ASC')

->execute();

}

/\*\*

\* @param int $categoryId

\* @return ResultsetInterface|Resultset|CategoryModel[]

\*/

protected function findHelps($categoryId)

{

return HelpModel::query()

->where('category\_id = :category\_id:', ['category\_id' => $categoryId])

->andWhere('published = 1')

->orderBy('priority ASC')

->execute();

}

}

<?php

namespace App\Caches;

use App\Models\ImMessage;

use App\Models\ImMessage as ImMessageModel;

use App\Models\User as UserModel;

use App\Repos\ImGroup as ImGroupRepo;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class ImActiveGroupList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'im\_active\_group\_list';

}

public function getContent($id = null)

{

$groups = $this->findGroups();

if (empty($groups)) {

return [];

}

$result = [];

foreach ($groups as $group) {

$result[] = [

'id' => $group->id,

'type' => $group->type,

'name' => $group->name,

'avatar' => $group->avatar,

'about' => $group->about,

'user\_count' => $group->user\_count,

'msg\_count' => $group->msg\_count,

];

}

return $result;

}

/\*\*

\* @param int $days

\* @param int $limit

\* @return ResultsetInterface|Resultset|UserModel[]

\*/

protected function findGroups($days = 7, $limit = 12)

{

$result = [];

$startTime = strtotime("-{$days} days");

$endTime = time();

$rows = ImMessageModel::query()

->columns(['receiver\_id', 'total\_count' => 'count(receiver\_id)'])

->groupBy('receiver\_id')

->orderBy('total\_count DESC')

->where('receiver\_type = :type:', ['type' => ImMessageModel::TYPE\_GROUP])

->betweenWhere('create\_time', $startTime, $endTime)

->limit($limit)

->execute();

if ($rows->count() > 0) {

$ids = kg\_array\_column($rows->toArray(), 'receiver\_id');

$groupRepo = new ImGroupRepo();

$result = $groupRepo->findByIds($ids);

}

return $result;

}

}

<?php

namespace App\Caches;

use App\Models\ImMessage as ImMessageModel;

use App\Models\User as UserModel;

use App\Repos\User as UserRepo;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class ImActiveUserList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'im\_active\_user\_list';

}

public function getContent($id = null)

{

$users = $this->findUsers($id);

if (empty($users)) {

return [];

}

$result = [];

foreach ($users as $user) {

$result[] = [

'id' => $user->id,

'name' => $user->name,

'avatar' => $user->avatar,

'title' => $user->title,

'about' => $user->about,

'vip' => $user->vip,

];

}

return $result;

}

/\*\*

\* @param int $days

\* @param int $limit

\* @return ResultsetInterface|Resultset|UserModel[]

\*/

protected function findUsers($days = 7, $limit = 12)

{

$result = [];

$startTime = strtotime("-{$days} days");

$endTime = time();

$rows = ImMessageModel::query()

->columns(['sender\_id', 'total\_count' => 'count(sender\_id)'])

->groupBy('sender\_id')

->orderBy('total\_count DESC')

->betweenWhere('create\_time', $startTime, $endTime)

->limit($limit)

->execute();

if ($rows->count() > 0) {

$ids = kg\_array\_column($rows->toArray(), 'sender\_id');

$userRepo = new UserRepo();

$result = $userRepo->findByIds($ids);

}

return $result;

}

}

<?php

namespace App\Caches;

use App\Repos\ImGroup as ImGroupRepo;

class ImGroup extends Cache

{

protected $lifetime = 365 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "im\_group:{$id}";

}

public function getContent($id = null)

{

$groupRepo = new ImGroupRepo();

$group = $groupRepo->findById($id);

return $group ?: null;

}

}

<?php

namespace App\Caches;

use App\Models\ImMessage as ImMessageModel;

use App\Models\User as UserModel;

use App\Repos\User as UserRepo;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class ImGroupActiveUserList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return "im\_group\_active\_user\_list:{$id}";

}

public function getContent($id = null)

{

$users = $this->findUsers($id);

if (empty($users)) {

return [];

}

$result = [];

foreach ($users as $user) {

$result[] = [

'id' => $user->id,

'name' => $user->name,

'avatar' => $user->avatar,

'title' => $user->title,

'about' => $user->about,

'vip' => $user->vip,

];

}

return $result;

}

/\*\*

\* @param int $groupId

\* @param int $days

\* @param int $limit

\* @return ResultsetInterface|Resultset|UserModel[]

\*/

protected function findUsers($groupId, $days = 7, $limit = 5)

{

$result = [];

$startTime = strtotime("-{$days} days");

$endTime = time();

$rows = ImMessageModel::query()

->columns(['sender\_id', 'total\_count' => 'count(sender\_id)'])

->groupBy('sender\_id')

->orderBy('total\_count DESC')

->where('receiver\_id = :group\_id:', ['group\_id' => $groupId])

->andWhere('receiver\_type = :type:', ['type' => ImMessageModel::TYPE\_GROUP])

->betweenWhere('create\_time', $startTime, $endTime)

->limit($limit)

->execute();

if ($rows->count() > 0) {

$ids = kg\_array\_column($rows->toArray(), 'sender\_id');

$userRepo = new UserRepo();

$result = $userRepo->findByIds($ids);

}

return $result;

}

}

<?php

namespace App\Caches;

use App\Models\ImGroup as ImGroupModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class ImNewGroupList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'im\_new\_group\_list';

}

public function getContent($id = null)

{

$limit = 12;

$groups = $this->findGroups($limit);

if ($groups->count() == 0) {

return [];

}

return $this->handleContent($groups);

}

/\*\*

\* @param ImGroupModel[] $groups

\* @return array

\*/

protected function handleContent($groups)

{

$result = [];

foreach ($groups as $group) {

$result[] = [

'id' => $group->id,

'type' => $group->type,

'name' => $group->name,

'avatar' => $group->avatar,

'about' => $group->about,

'user\_count' => $group->user\_count,

'msg\_count' => $group->msg\_count,

];

}

return $result;

}

/\*\*

\* @param int $limit

\* @return ResultsetInterface|Resultset|ImGroupModel[]

\*/

public function findGroups($limit = 12)

{

return ImGroupModel::query()

->where('published = 1')

->orderBy('id DESC')

->limit($limit)

->execute();

}

}

<?php

namespace App\Caches;

use App\Models\User as UserModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

class ImNewUserList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'im\_new\_user\_list';

}

public function getContent($id = null)

{

$limit = 12;

$users = $this->findUsers($limit);

if ($users->count() == 0) {

return [];

}

return $this->handleContent($users);

}

/\*\*

\* @param UserModel[] $users

\* @return array

\*/

protected function handleContent($users)

{

$result = [];

foreach ($users as $user) {

$result[] = [

'id' => $user->id,

'name' => $user->name,

'avatar' => $user->avatar,

'title' => $user->title,

'about' => $user->about,

'vip' => $user->vip,

];

}

return $result;

}

/\*\*

\* @param int $limit

\* @return ResultsetInterface|Resultset|UserModel[]

\*/

public function findUsers($limit = 12)

{

return UserModel::query()

->where('deleted = 0')

->orderBy('id DESC')

->limit($limit)

->execute();

}

}

<?php

namespace App\Caches;

use App\Models\Category as CategoryModel;

use App\Models\Course as CourseModel;

use App\Services\Category as CategoryService;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

/\*\*

\* 免费课程

\*/

class IndexFreeCourseList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'index\_free\_course\_list';

}

public function getContent($id = null)

{

$categoryLimit = 5;

$courseLimit = 8;

$categories = $this->findCategories($categoryLimit);

if ($categories->count() == 0) {

return [];

}

$result = [];

foreach ($categories as $category) {

$item = [];

$item['category'] = [

'id' => $category->id,

'name' => $category->name,

];

$item['courses'] = [];

$courses = $this->findCategoryCourses($category->id, $courseLimit);

if ($courses->count() == 0) {

continue;

}

$categoryCourses = [];

foreach ($courses as $course) {

$categoryCourses[] = [

'id' => $course->id,

'title' => $course->title,

'cover' => $course->cover,

'market\_price' => $course->market\_price,

'vip\_price' => $course->vip\_price,

'model' => $course->model,

'level' => $course->level,

'user\_count' => $course->user\_count,

'lesson\_count' => $course->lesson\_count,

];

}

$item['courses'] = $categoryCourses;

$result[] = $item;

}

return $result;

}

/\*\*

\* @param int $limit

\* @return ResultsetInterface|Resultset|CategoryModel[]

\*/

protected function findCategories($limit = 5)

{

return CategoryModel::query()

->where('type = :type:', ['type' => CategoryModel::TYPE\_COURSE])

->andWhere('level = 1 AND published = 1')

->orderBy('priority ASC')

->limit($limit)

->execute();

}

/\*\*

\* @param int $categoryId

\* @param int $limit

\* @return ResultsetInterface|Resultset|CourseModel[]

\*/

protected function findCategoryCourses($categoryId, $limit = 8)

{

$categoryService = new CategoryService();

$categoryIds = $categoryService->getChildCategoryIds($categoryId);

return CourseModel::query()

->inWhere('category\_id', $categoryIds)

->andWhere('published = 1')

->andWhere('market\_price = 0')

->orderBy('score DESC')

->limit($limit)

->execute();

}

}

<?php

namespace App\Caches;

use App\Models\ChapterLive as ChapterLiveModel;

use App\Repos\Chapter as ChapterRepo;

use App\Repos\Course as CourseRepo;

use Phalcon\Mvc\Model\Resultset;

/\*\*

\* 直播课程

\*/

class IndexLiveList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'index\_live\_list';

}

public function getContent($id = null)

{

/\*\*

\* 限制输出多少天数（一维限额）

\*/

$dayLimit = 3;

/\*\*

\* 限制每天维度下的输出数（二维限额）

\*/

$perDayLimit = 10;

$beginTime = strtotime('today');

$endTime = strtotime("+30 days");

/\*\*

\* @var Resultset|ChapterLiveModel[] $lives

\*/

$lives = ChapterLiveModel::query()

->betweenWhere('start\_time', $beginTime, $endTime)

->orderBy('start\_time ASC')

->execute();

if ($lives->count() == 0) {

return [];

}

$result = [];

$chapterIds = kg\_array\_column($lives->toArray(), 'chapter\_id');

$chapterRepo = new ChapterRepo();

$chapters = $chapterRepo->findByIds($chapterIds);

$chapterMapping = [];

foreach ($chapters as $chapter) {

$chapterMapping[$chapter->id] = $chapter;

}

$courseIds = kg\_array\_column($lives->toArray(), 'course\_id');

$courseRepo = new CourseRepo();

$courses = $courseRepo->findByIds($courseIds);

$courseMapping = [];

foreach ($courses as $course) {

$courseMapping[$course->id] = $course;

}

foreach ($lives as $live) {

if (count($result) >= $dayLimit) {

break;

}

$day = date('y-m-d', $live->start\_time);

if (isset($result[$day]) && count($result[$day]) >= $perDayLimit) {

continue;

}

$chapter = $chapterMapping[$live->chapter\_id];

$course = $courseMapping[$chapter->course\_id];

$chapterInfo = [

'id' => $chapter->id,

'title' => $chapter->title,

'start\_time' => $live->start\_time,

'end\_time' => $live->end\_time,

];

$courseInfo = [

'id' => $course->id,

'title' => $course->title,

'cover' => $course->cover,

'market\_price' => $course->market\_price,

'vip\_price' => $course->vip\_price,

'model' => $course->model,

'level' => $course->level,

'user\_count' => $course->user\_count,

'lesson\_count' => $course->lesson\_count,

];

$result[$day][] = [

'course' => $courseInfo,

'chapter' => $chapterInfo,

];

}

return $result;

}

}

<?php

namespace App\Caches;

use App\Models\Category as CategoryModel;

use App\Models\Course as CourseModel;

use App\Services\Category as CategoryService;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

/\*\*

\* 新上课程

\*/

class IndexNewCourseList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'index\_new\_course\_list';

}

public function getContent($id = null)

{

$categoryLimit = 5;

$courseLimit = 8;

$categories = $this->findCategories($categoryLimit);

if ($categories->count() == 0) {

return [];

}

$result = [];

foreach ($categories as $category) {

$item = [];

$item['category'] = [

'id' => $category->id,

'name' => $category->name,

];

$item['courses'] = [];

$courses = $this->findCategoryCourses($category->id, $courseLimit);

if ($courses->count() == 0) {

continue;

}

$categoryCourses = [];

foreach ($courses as $course) {

$categoryCourses[] = [

'id' => $course->id,

'title' => $course->title,

'cover' => $course->cover,

'market\_price' => $course->market\_price,

'vip\_price' => $course->vip\_price,

'model' => $course->model,

'level' => $course->level,

'user\_count' => $course->user\_count,

'lesson\_count' => $course->lesson\_count,

];

}

$item['courses'] = $categoryCourses;

$result[] = $item;

}

return $result;

}

/\*\*

\* @param int $limit

\* @return ResultsetInterface|Resultset|CategoryModel[]

\*/

protected function findCategories($limit = 5)

{

return CategoryModel::query()

->where('type = :type:', ['type' => CategoryModel::TYPE\_COURSE])

->andWhere('level = 1 AND published = 1')

->orderBy('priority ASC')

->limit($limit)

->execute();

}

/\*\*

\* @param int $categoryId

\* @param int $limit

\* @return ResultsetInterface|Resultset|CourseModel[]

\*/

protected function findCategoryCourses($categoryId, $limit = 8)

{

$categoryService = new CategoryService();

$categoryIds = $categoryService->getChildCategoryIds($categoryId);

return CourseModel::query()

->inWhere('category\_id', $categoryIds)

->andWhere('published = 1')

->orderBy('id DESC')

->limit($limit)

->execute();

}

}

<?php

namespace App\Caches;

use App\Models\Course as CourseModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

/\*\*

\* 简版免费课程

\*/

class IndexSimpleFreeCourseList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'index\_simple\_free\_course\_list';

}

public function getContent($id = null)

{

$limit = 8;

$courses = $this->findCourses($limit);

if ($courses->count() == 0) {

return [];

}

$result = [];

foreach ($courses as $course) {

$result[] = [

'id' => $course->id,

'title' => $course->title,

'cover' => $course->cover,

'market\_price' => $course->market\_price,

'vip\_price' => $course->vip\_price,

'model' => $course->model,

'level' => $course->level,

'user\_count' => $course->user\_count,

'lesson\_count' => $course->lesson\_count,

];

}

return $result;

}

/\*\*

\* @param int $limit

\* @return ResultsetInterface|Resultset|CourseModel[]

\*/

protected function findCourses($limit = 8)

{

return CourseModel::query()

->where('published = 1')

->andWhere('market\_price = 0')

->orderBy('score DESC')

->limit($limit)

->execute();

}

}

<?php

namespace App\Caches;

use App\Models\Course as CourseModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

/\*\*

\* 简版新上课程

\*/

class IndexSimpleNewCourseList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'index\_simple\_new\_course\_list';

}

public function getContent($id = null)

{

$limit = 8;

$courses = $this->findCourses($limit);

if ($courses->count() == 0) {

return [];

}

$result = [];

foreach ($courses as $course) {

$result[] = [

'id' => $course->id,

'title' => $course->title,

'cover' => $course->cover,

'market\_price' => $course->market\_price,

'vip\_price' => $course->vip\_price,

'model' => $course->model,

'level' => $course->level,

'user\_count' => $course->user\_count,

'lesson\_count' => $course->lesson\_count,

];

}

return $result;

}

/\*\*

\* @param int $limit

\* @return ResultsetInterface|Resultset|CourseModel[]

\*/

protected function findCourses($limit = 8)

{

return CourseModel::query()

->where('published = 1')

->orderBy('id DESC')

->limit($limit)

->execute();

}

}

<?php

namespace App\Caches;

use App\Models\Course as CourseModel;

use Phalcon\Mvc\Model\Resultset;

use Phalcon\Mvc\Model\ResultsetInterface;

/\*\*

\* 简版会员课程

\*/

class IndexSimpleVipCourseList extends Cache

{

protected $lifetime = 1 \* 86400;

public function getLifetime()

{

return $this->lifetime;

}

public function getKey($id = null)

{

return 'index\_simple\_vip\_course\_list';

}

public function getContent($id = null)

{

$limit = 8;

$courses = $this->findCourses($limit);

if ($courses->count() == 0) {

return [];

}

$result = [];

foreach ($courses as $course) {

$result[] = [

'id' => $course->id,

'title' => $course->title,

'cover' => $course->cover,

'market\_price' => $course->market\_price,

'vip\_price' => $course->vip\_price,

'model' => $course->model,

'level' => $course->level,

'user\_count' => $course->user\_count,

'lesson\_count' => $course->lesson\_count,

];

}

return $result;

}

/\*\*

\* @param int $limit

\* @return ResultsetInterface|Resultset|CourseModel[]

\*/

protected function findCourses($limit = 8)

{

return CourseModel::query()

->where('published = 1')

->andWhere('vip\_price >= 0')

->orderBy('score DESC')

->limit($limit)

->execute();

}

}