北哥大话Yii2缓存机制 - 缓存依赖

我们都知道，yii2的缓存是支持依赖的，就是我们设置的缓存是否失效除了过期时间还决定于它所依赖的东东是否变化。

用好依赖将大大提高我们使用缓存的效果，本节讲解yii2缓存的5种依赖方式。

还是先列出

1. yii\caching\DbDependency
2. yii\caching\ExpressionDependency
3. yii\caching\FileDependency
4. yii\caching\TagDependency
5. yii\caching\ChainedDependency

begin...

DbDependency

我们可以将SQL语句直接作为依赖条件。当数据库查询出的内容发生变化时候，缓存即会受到影响。

假设我当前要搜索出所有的分类信息并缓存起来，每3000秒更新一次或当分类有变化是后自动更新，使用DbDependency我们可以轻松实现。

$dependency = new \yii\caching\DbDependency(['sql'=>'SELECT COUNT(\*) FROM category']);

$cache->add('category',function($cache){

return Category::find()->all();

},3600,dependency);

使用缓存获取数据

$cache->get("category");

代码执行后，当超过3000秒或分类字段条数有变化，都将缓存设置为失效。该依赖的推荐为依赖里执行的sql语句所损耗的资源最好小于缓存所带来的损耗。

ExpressionDependency

指定的 PHP 表达式执行结果发生变化，则缓存变化，这里的表达式为整个PHP，而不局限于Yii。

$dependency = new \yii\caching\ExpressionDependency(['expression'=>'\Yii::$app->request->get("id")']);

$cache->add("hy","Hello Yii",3000,$dependency);

获取缓存数据

$cache->get("hy");

如上面代码，我们在3000之内使用hy都直接从缓存读取Hello Yii，但是当使用缓存的页面url ?id=xx发生变化的时候则失败，返回flase

FileDependency

将缓存和文件绑定在一起。如果文件的内容发生变化，那么缓存将会收到影响。而Yii是根据这个文件的最后修改时间来判断其内容是否变化的。

依赖文件 hello.txt，其内容为a

设置依赖

$dependency = new \yii\caching\FileDependency(['fileName'=>'@app/hello.txt']);

$cache->add("hy","Hello Yii",3000,$dependency);

获取缓存数据

$cache->get("hy");

现在当我们将hello.txt 内容改为b的时候,再次获取缓存则失败，返回flase。  
*小提示：文件路径可以使用别名哦*

TagDependency

该类依赖的意思是某个或某些缓存是否失效除了由时间决定，还有一个标签决定，举例说明

use yii\caching\TagDependency;

$cache->add("abc","Hello Yii",3000,new TagDependency(['tags' => 'num']));

此刻我们设置了一个缓存abc，它依赖于标签num

当我们在其他的代码中执行了

use yii\caching\TagDependency;

TagDependency::invalidate(Yii::$app->cache, 'num');

当我们执行TagDependency::invalidate(Yii::$app->cache, 'num');时，可以将与num标签相关的缓存设置为无效，因此当我们再次获取时，abc已经失效，即便3000秒还没有到。

ChainedDependency

一个系统中的数据是彼此相连的，缓存的依赖也可能出现多种联系，使用TagDependency可以将一组缓存标记为同一个标签，然后使用invalidate函数将这组缓存统一失效，但是这样还不足以满足千变万化的依赖关系。

因此 **ChainedDependency** 出现了，我们可以将一系列的缓存依赖使用 ChainedDependency 组合起来，这样当这个组合中的某个或全部依赖发生变化后，将直接影响到缓存。

现在我们仍然通过一个例子说明，假设我们的一个缓存同时依赖于两个依赖。

$cache = Yii::$app->cache;

$fileDependency = new \yii\caching\FileDependency(['fileName'=>'@app/hello.txt']);

$sqlDependency = new \yii\caching\DbDependency(['sql'=>'SELECT COUNT(\*) FROM category']);

$cache->set('abc', 'hello yii', 3000, new ChainedDependency([

'dependOnAll'=>true,

'dependencies' => [$fileDependency, $sqlDependency]

]));

这样，当 **$fileDependency** 和 **$sqlDependency** 中任何一个依赖发生变化后都会引起缓存的失效，这里要注意一个问题 dependOnAll(默认为true)，当为true时，只要依赖组合中的一个发生变化则缓存失效，否则必须每个依赖都变化才会引起缓存的失效。

北哥的话

缓存是我们提供程序性能的主要手段，以上五种依赖务必记住，当我们使用缓存的时候，优化缓存逻辑也是我们要考虑的，如何有效的使用依赖保证因为依赖而带来的各种延时，是每个攻城狮要研究的。