**10个技巧优化PHP程序Laravel 5框架**

这里是简单的列表：

配置信息缓存 artisan config:cache

路由缓存 artisan route:cache

类映射加载优化 artisan optimize

自动加载优化 composer dumpautoload

使用 Memcached 来存储会话 config/session.php

使用专业缓存驱动器 config/cache.php

数据库请求优化

为数据集书写缓存逻辑

使用即时编译器（JIT），如：HHVM、OpCache

前端资源合并 Elixir

1. 配置信息缓存

使用以下 Artisan 自带命令，把 config 文件夹里所有配置信息合并到一个文件里，减少运行时文件的载入数量：

**php artisan config:cache**

上面命令会生成文件 bootstrap/cache/config.php，可以使用以下命令来取消配置信息缓存：

**php artisan config:clear**

此命令做的事情就是把 bootstrap/cache/config.php 文件删除。

注意：配置信息缓存不会随着更新而自动重载，所以，开发时候建议关闭配置信息缓存，一般在生产环境中使用，可以配合 Envoy 任务运行器 一起使用。

2. 路由缓存

路由缓存可以有效的提高路由器的注册效率，在大型应用程序中效果越加明显，可以使用以下命令：

**php artisan route:cache**

以上命令会生成 bootstrap/cache/routes.php 文件，需要注意的是，路由缓存不支持路由匿名函数编写逻辑。

可以使用下面命令清除路由缓存：

**php artisan route:clear**

此命令做的事情就是把 bootstrap/cache/routes.php 文件删除。

注意：路由缓存不会随着更新而自动重载，所以，开发时候建议关闭路由缓存，一般在生产环境中使用，可以配合 Envoy 任务运行器 一起使用。

3. 类映射加载优化

optimize 命令把常用加载的类合并到一个文件里，通过减少文件的加载，来提高运行效率：

**php artisan optimize --force**

会生成 bootstrap/cache/compiled.php 和 bootstrap/cache/services.json 两个文件。

你可以可以通过修改 config/compile.php 文件来添加要合并的类。

在 production 环境中，参数 --force 不需要指定，文件就会自动生成。

要清除类映射加载优化，请运行以下命令：

**php artisan clear-compiled**

此命令会删除上面 optimize 生成的两个文件。

注意：此命令要运行在 php artisan config:cache 后，因为 optimize 命令是根据配置信息（如：config/app.php 文件的 providers 数组）来生成文件的。

4. 自动加载优化

此命令不止针对于 Laravel 程序，适用于所有使用 composer 来构建的程序。此命令会把 PSR-0 和 PSR-4 转换为一个类映射表，来提高类的加载速度。

**composer dumpautoload -o**

注意：php artisan optimize --force 命令里已经做了这个操作。

5. 使用 Memcached 来存储会话

每一个 Laravel 的请求，都会产生会话，修改会话的存储方式能有效提高程序效率，会话的配置信息是 config/session.php，建议修改为 Memcached 或者 Redis 等专业的缓存软件：

**'driver' => 'memcached',**

6. 使用专业缓存驱动器

「缓存」是提高应用程序运行效率的法宝之一，默认缓存驱动是 file 文件缓存，建议切换到专业的缓存系统，如 Redis 或者 Memcached，不建议使用数据库缓存。

**'default' => 'redis',**

7. 数据库请求优化

数据关联模型读取时使用 延迟预加载 和 预加载 ；

使用 Laravel Debugbar 或者 Clockwork 留意每一个页面的总数据库请求数量；

这里的篇幅只写到与 Laravel 相关的，其他关于数据优化的内容，请自行查阅其他资料。

8. 为数据集书写缓存逻辑

合理的使用 Laravel 提供的缓存层操作，把从数据库里面拿出来的数据集合进行缓存，减少数据库的压力，运行在内存上的专业缓存软件对数据的读取也远远快于数据库。

**$posts = Cache::remember('index.posts', $minutes = 30, function()**

**{**

**return Post::with('comments', 'tags', 'author', 'seo')->whereHidden(0)->get();**

**});**

remember 甚至连数据关联模型也都一并缓存了，多么方便呀。

9. 使用即时编译器

HHVM 和 OpCache 都能轻轻松松的让你的应用程序在不用做任何修改的情况下，直接提高 50% 或者更高的性能，PHPhub 之前做个一个实验，具体请见：使用 OpCache 提升 PHP 5.5+ 程序性能。

10. 前端资源合并

作为优化的标准，一个页面只应该加载一个 CSS 和 一个 JS 文件，并且文件要能方便走 CDN，需要文件名随着修改而变化