30.模型的预加载

学习要点：

1.预加载

本节课我们来开始学习数据库模型的预加载功能。

一. 预加载

1. 预加载，就是解决关联查询中产生的 N+1 次查询带来的资源消耗
2. 我们要获取所有书籍的作者(或拥有者)，普通查询方案如下：

```
//获取所有书籍列表
$books = Book::all();

//遍历每一本书
foreach ($books as $book) {
    //每一本书的关联用户的姓名
    DebugBar::info($book->user->username);
}
```

PS：通过调试器 Debugbar 中 SQL 语句的分析，发现包含十多条 SQL 语句；
PS：原因是关联查询时，每遍历一次就会执行一遍 SQL 语句，导致性能欠佳；
PS：所谓 N+1 条，就是起初获取全部数据的 1 条和，遍历的 N 条；

3. 使用 with()关键字，进行预载入设置，提前将 SQL 整合；

```
//with 关键字预载入
$books = Book::with('user')->get();

foreach ($books as $book) {
    DebugBar::info($book->user->username);
}
```

PS：此时的 SQL 执行数目为：1+1 条；也支持数组多个关联 with['book','prifile']；
PS：预加载也可以设置显示的列；

```
//预载入设置指定的列
$books = Book::with('user:id,username')->get();
```

4. 如果每次都必须使用预载入进行关联查询，可以在模型中定义；

```
protected $with = ['user'];
```

PS：此时就可以像最初那样写代码，而不需要使用 with()方法了；

5. 为了演示方便，暂时取消模型$with，再看下预载入结合筛选；

```
$books = Book::with(['user' => function ($query) {
    $query->where('id', 19);
}])->get();
```

PS：预载入筛选不可以使用 limit、take 方法；

6. 有时，可能会产生逻辑判断是否查询数据，但预加载会提前关联执行；
7. 这样，会导致资源性能的浪费，这时，可以采用延迟预载入；

```
$books = Book::all();

if (true) {
    $books = $books->load('user');   //load(['user' => function () {}])
    foreach ($books as $book) {
        DebugBar::info($book->user->username);
    }
}
```

8. 使用 loadCount()方法，可以实现延迟关联统计；

```
$users = User::all();
if (true) {
    return $users->loadCount('book');
}
```