或者，你也可以选择更简短的表示方法：

<%= render @objects %>

这会为集合的每个成员插入一个部分，自动计算部分名称并传递局部变量。[[1]](#footnote-1)这样操作能够快20倍。

|  |  |
| --- | --- |
| Rails 3.x | Rails 4.x |
| 0.066 ± 0.001 | 0.100 ± 0.005 |

**表3—渲染10000个对象的集合用时**

渲染集合的速度快的原因是它只是初始化模板一次。然后它重用相同的模板来渲染集合中的所有对象。在循环中渲染10000个部分将不得不重复初始化10000次。

初始化模板需要做多少工作？我已经在Rails 4中分析了10000个部分的渲染来说明这一点。我们来看看摘要。

|  |  |
| --- | --- |
| 操作 | 占总时长百分比 |
| 日志记录 | 45% |
| 查找和读取模板（从磁盘或缓存） | 21% |
| 设置执行上下文（局部变量等） | 9% |
| 模板类实例化 | 5% |
| 渲染 | 5% |
| 其它 | 15% |

我想我们都会对此感到惊喜。实际渲染只需要5％的时间。难怪如果我们跳过初始化，我们将获得两个数量级的加速—正如我们的测量结果一样。

让我们看看为什么日志记录占用了45％的时间。事实证明，在生产模式下使用默认的config.log\_level =:info，Rails会产生过多的输出。

INFO --: Started GET "/test" for 127.0.0.1 at 2014-08-13 10:21:40 -0500

INFO --: Processing by TestController#index as HTML

INFO --: Rendered test/\_object.html.erb (0.1ms)

*«9998 more object.html.erb partial rendering notifications»*

INFO --: Rendered test/\_object.html.erb (0.0ms)

INFO --: Rendered test/\_dummy.html.erb (1904.0ms)

INFO --: Rendered test/index.html.erb within layouts/application (1945.4ms)

INFO --: Completed 200 OK in 1952ms (Views: 1948.6ms | ActiveRecord: 0.0ms)

1. http://guides.rubyonrails.org/layouts\_and\_rendering.html#rendering-collections [↑](#footnote-ref-1)