

很可能会渲染10000个部分的Rails应用程序不是很多。但是1000个似乎不是一个无法达到的数字。我们来做一些算术。想象一下，你在循环中渲染了100个具有部分的对象。现在想象部分调用10个其他部分。这些数字看似合法。如果你渲染了一个包含10列的分页表，你将获得这样的设置。我们有多少渲染部分调用？已经有1000个了。我们在渲染部分功能中花多少时间？根据我的测量结果，大约200毫秒。如果我们考虑实际渲染有用内容的时间，我们将轻松超过1秒的响应时间标记，而这对任何Web应用程序来说都是不可接受的。

**乔的提问：**

**但Rails应用程序很少需要渲染10,000个部分，不是吗？**

您可能不希望使用config.log\_level =:warn完全使您的日志静音，但这样做会带给您两倍的加速。

|  |  |
| --- | --- |
| config.log\_level = :info | config.log\_level = :warn |
| 1.840 ± 0.045 | 0.830 ± 0.049 |

**表4—渲染具有不同日志级别的10000个对象的集合用时**

这仍然没有渲染集合那么快（0.1秒）。剩下的0.7秒去哪儿了？事实证明，Rails使用Observer模式实现了一个记录器。部分渲染会触发render\_partial.action\_view事件。当发生这种情况时，ActionSupport::LogSubscriber会收到通知，然后运行Logger来生成输出。这个管道代码大约需要0.2秒。剩余的步骤是模板初始化和执行上下文评估。

渲染集合没有任何开销，也不会产生过多的日志输出。这使得它明显优于在循环中渲染部分。

Rails 2.x中没有渲染集合。但是，如果您仍在使用该版本，请尝试使用模板内联插件。[[1]](#footnote-1)通过在Rails编译之前将部分代码文本插入父模板，可以实现相同的效果。

这是我在Acunote[[2]](#footnote-2)工作时写的，Acunote是使用Ruby on Rails构建的在线项目管理系统。我们在页面上渲染了数百个任务，每个任务有8-10个字段。对于每个字段，我们有一个单独的部分用于渲染，并且在Rails 2.x中没有渲染集合。

1. http://github.com/acunote/template\_inliner [↑](#footnote-ref-1)
2. http://www.acunote.com/ [↑](#footnote-ref-2)