要使用它，请将插件添加到您的Rails应用程序中，并将内联：true添加到呈现语句：

<% @objects.each do |object| %>

<%= render partial: 'object' , locals: { object: object }, inline: true %>

<% end %>

Rails从未看到呈现部分调用，因此我们得到了相同的两个数量级性能改善的结果。

Rails 2.x Rails 2.x with template inliner

0.003 ± 0.0001 0.335 ± 0.006

表5-内联呈现10，000个部分的时间

用禁用GC测量，以秒为单位。结果与分词数成线性关系。

避免Iterator不安全的帮手和函数

所有呈现帮助程序都是我所说的迭代器不安全的。它们需要时间和记忆，所以在循环中使用时要小心，尤其是link\_to, url\_for,和img\_tag。

出于两个原因，我没有比小心更好的建议了。首先，您不能避免使用这些助手(特别是在较新的Rails中)。其次，很难对它们进行基准测试。帮手的性能取决于太多的因素，使任何综合基准无用。例如，当路由的复杂性增加时link\_to和url\_for会变慢。当您添加更多的资产时，img\_tag的性能更差。在一个应用程序中，在循环中呈现1000个URL是安全的，而在另一个应用程序中则不是。所以…请注意。

Takeaways

为了更快地使用Rails应用程序，我们在第2章“修复常见性能问题”第13页中学到了同样的技术：

* 优化ActiveRecord占用的内存，通过积极的预加载、选择性属性获取和批量数据处理。
* 将视图中的显式迭代器替换为呈现集合，这将减少时间和内存。
* 让您的数据库服务器执行数据操作。

这一章和前两章包含了我可以给您的所有建议，以使您的Ruby和Rails代码更快。继续，应用它，并收获你的绩效利益。