**跟蓝狐学MVC教程--MiniProfiler.EF6监控调试MVC5和EF6的性能**

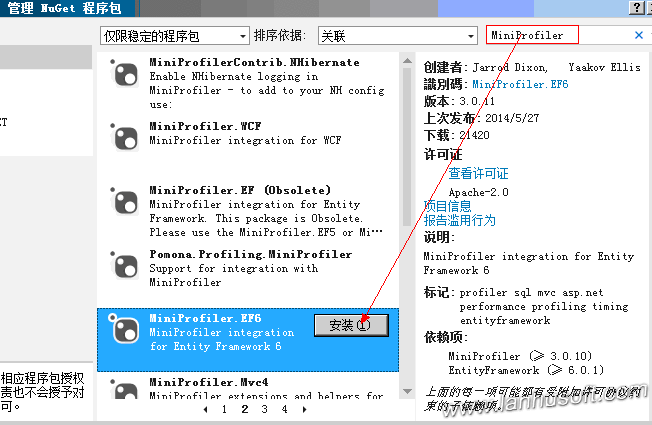
Posted By : 蓝狐

Updated On : 2015-05-19 17:12

以前开发Webform的时候可以开启trace来跟踪页面事件，这对于诊断程序的性能是有很大的帮助的，起到事半功倍的作用，今天我就来谈用[mvc](http://www.lanhusoft.com/Article/169.html)开发项目的调试和性能监控。EF框架自动给我生成sql语句，当我们的程序遇到性能问题的时候我们可以用MiniProfiler.EF来监控调试MVC和EF的性能，查看生成的sql语句、运行了哪些sql，以及所花的时间。MiniProfiler.EF，一个轻量级开源的mvc性能调试、监控组件MiniProfiler专门为EF定制的版本。下面通过一个具体一例子的说明怎么在我们的项目中用MiniProfiler.EF6监控调试MVC和EF的性能。下面的项目是基于我上面的一篇文章的，[MVC5与EF6 Code First 第一个入门完整实例教程](http://www.lanhusoft.com/Article/123.html)

**1、安装MiniProfiler.EF6**

在[nuget](http://www.lanhusoft.com/Article/185.html)搜索框中输入MiniProfiler,将出现下面结果：



点击安装将把MiniProfiler.EF6相关的dll加到项目中。

**2、添加MiniProfiler.EF相关代码到项目里面**

**1、在Global.asax加入MiniProfiler相关的监控代码**

修改之后完整内容为：

1. using System.Web.Mvc;
2. using System.Web.Optimization;
3. using System.Web.Routing;
4. using StackExchange.Profiling;
5. using StackExchange.Profiling.EntityFramework6;
6. namespace MiniProfilerDemo
7. {
8. public class MvcApplication : System.Web.HttpApplication
9. {
10. protected void Application\_Start()
11. {
12. AreaRegistration.RegisterAllAreas();
13. FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
14. RouteConfig.RegisterRoutes(RouteTable.Routes);
15. BundleConfig.RegisterBundles(BundleTable.Bundles);
16. MiniProfilerEF6.Initialize();
17. }
18. protected void Application\_BeginRequest()
19. {
20. MiniProfiler.Start();
21. }
22. protected void Application\_EndRequest()
23. {
24. MiniProfiler.Stop();
25. }
26. }
27. }

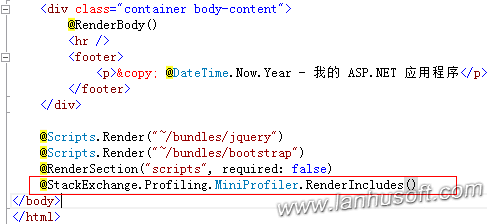
其中是在Application\_Start加入了MiniProfilerEF6.Initialize()和添加了Application\_BeginRequest、Application\_BeginRequest两个Application的事件函数，这个的作用分别是初始化MiniProfilerEF6和开始、结束MiniProfiler监控。

**2、修改\_Layout.cshtml视图文件**

在Views\Shared\\_Layout.cshtml文件的body前面加上一段代码，让监控展示在页面上。

1. @StackExchange.Profiling.MiniProfiler.RenderIncludes()

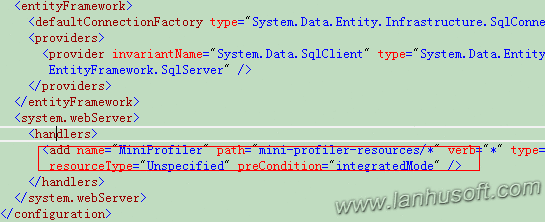
如下图：



**3、在Web.config加入代码**

1. <system.webServer>
2. <handlers>
3. <add name="MiniProfiler" path="mini-profiler-resources/\*" verb="\*" type="System.Web.Routing.UrlRoutingModule" resourceType="Unspecified" preCondition="integratedMode" />
4. </handlers>
5. </system.webServer>

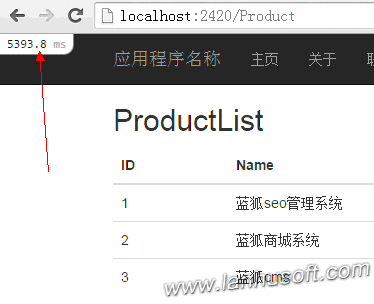
为了要在页面上显示MVC和EF的调试跟踪时间必须要加入上面的代码。如下图：



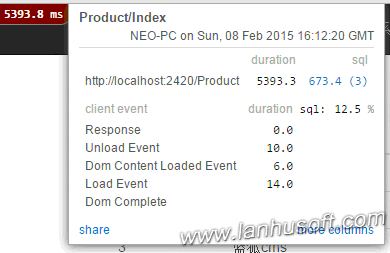
在system.webServer配置结点下的handlers结点，加入了一个名为MiniProfiler的handler。

**3、查看运行结果**

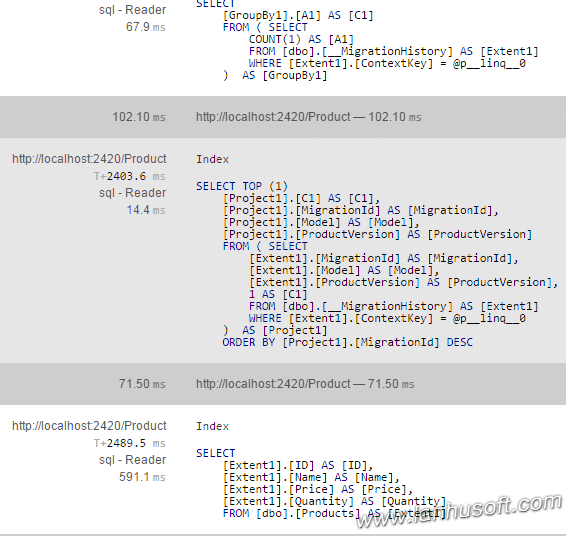
运行程序，查看页面如下图：



可以看到左角多了一个数字的区块，表示这个页面所花的毫秒数，点击上面的数字就可以弹出详细的时间跟踪信息，如下图：



可以看到这个页面运行了三个sql语句，sql所花时间为您673.4毫秒，占部时间为为12.5%。还可以点击这个sql的时间，将显示运行了哪些sql，如下图：

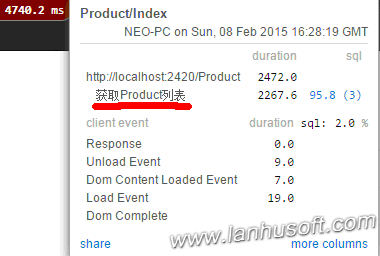


**4、细微监控方法内部的时间**

现在我们为ProductController加上监控，监控获取从数据库中获取Product记录所花的时间。我们把ProductController修改为：

1. using MiniProfilerDemo.DAL;
2. using System.[linq](http://www.lanhusoft.com/Article/163.html);
3. using System.Web.Mvc;
4. using StackExchange.Profiling;
5. using System.Collections.Generic;
6. using MiniProfilerDemo.Models;
7. namespace MiniProfilerDemo.Controllers
8. {
9. public class ProductController : Controller
10. {
11. public ActionResult Index()
12. {
13. using (EFDbContext db = new EFDbContext())
14. {
15. var profiler = MiniProfiler.Current;
16. List<Product> m;
17. using (profiler.Step("获取Product列表"))
18. {
19. m = db.Products.ToList();
20. }
21. return View(m);
22. }
23. }
24. }
25. }

重新生成项目，再次运行查看页面，如下图：



可以看到上面多了我们刚才手动加的“获取Product列表”条记录。这样可以细微监控方法内部的时间，方便、快速地帮我们找出我们的程序的瓶颈所在。