مقدمه:

بررسی عملکرد، پیدا نمودن مشکلات و نقص های برنامه و رفع اشکالات همواره یکی از چالش های پیش رو برای توسعه دهندگان نرم افزار بوده است. ابزار های مختلفی جهت پیدا نمودن مشکلات، دیباگ نمودن، جمع آوری خطا ها و لاگ نمودن اطلاعات و ... در برنامه های تحت وب و ویندوز ارائه شده اند که هر کدام از آن ها بخشی از این مشکلات را پوشش داده اند. خوشبختانه مدتی است که برنامه نویسان وب سایت معظم StackOverFlow.com ابزار بسیار کوچک و قدرتمندی به نام MiniProfiler را برای پروفایل نمودن این وب سایت نوشته و همچنین به شکل متن باز به اجتماع برنامه نویسان ارائه کرده اند.

این ابزار به ساده ترین شکل ممکن با پروژه های ASP.NET MVC و همچنین Ruby تلفیق شده و شروع به کار می کند. فرض کنید که پروژه هنگام لود با تاخیر زیادی همراه است و شما به دنبال این موضوع هستید که این تاخیر در کدام اکشن به وجود می آید و کدام فرمان باعث این موضوع شده است. در این مقاله به بررسی این مسئله می پردازیم.

در این مطلب قصد معرفی ابتدایی این ابزار بی نظیر را داریم و جهت اطلاع از تمامی امکانات ارائه شده و جزئیات بیشتر می توانید به وب سایت MiniProfiler مراجعه کنید.

آغاز:

نصب و استفاده از این کتابخانه را به شکل گام به گام از ابتدا انجام خواهیم داد.

گام اول:

ابتدا قصد داریم این ابزار کوچک را به پروژه خود اضافه کنیم. برای انجام این کار روش های مختلفی وجود دارد که ساده ترین آن این است که پنجره Package Manager console را در برنامه Visual Studio باز نموده و فرمان **Install-Package MiniProfiler**را صادر کنیم.

```
PM> Install-Package MiniProfiler
```

پس از اندک زمانی، هسته مرکزی ابزار MiniProfiler به پروژه شما اضافه خواهد شد. این هسته شامل کتابخانه MiniProfiler.dll می باشد.

گام دوم:

به تگ Handler در فایل وب کانفیگ مراجعه نموده و خط زیر را اضافه کنید.

```
<system.webServer>

  <handlers>
    <add name="MiniProfiler" path="mini-profiler-
          resources/*" verb="*" type="System.Web.Routing.UrlRoutingModule" resourceType="Unspecified"
          preCondition="integratedMode" />
  </handlers>
```

```
</system.webServer>
```

گام سوم :

اکنون قصد داریم این ابزار را هنگامی که برنامه را **فقط در حالت لوکال** اجرا شده است، اجرا کنیم. برای انجام این کار به فایل Global.asaxمراجعه نموده و خطوط زیر را اضافه کنید.

```
protected void Application_BeginRequest()
{
   if (Request.IsLocal)
   {
      StackExchange.Profiling.MiniProfiler.Start();
   }
}

protected void Application_EndRequest()
{
   StackExchange.Profiling.MiniProfiler.Stop();
}
```

عملکرد قطعه کد بالا کاملا مشخص می باشد.

گام چهارم:

خوب، اکنون قرار است مشخص کنیم که پنجره نمایش اطلاعات ابزار MiniProfiler چگونه و در کدام View نمایش داده شود. اگر قصد دارید این پنجره در تمامی View ها نمایش داده شود، بهترین کار این است که به View ای که نقش MasterPage دارد، (یا همون View ای که سایر View ها از آن به ارث رفته اند) مراجعه نموده و قبل از بسته شدن تگ body خط کد زیر را وارد کنید.

```
@StackExchange.Profiling.MiniProfiler.RenderIncludes()
</body>
```

کار تمام است و اکنون زمان آزمایش برنامه فرا رسیده است.

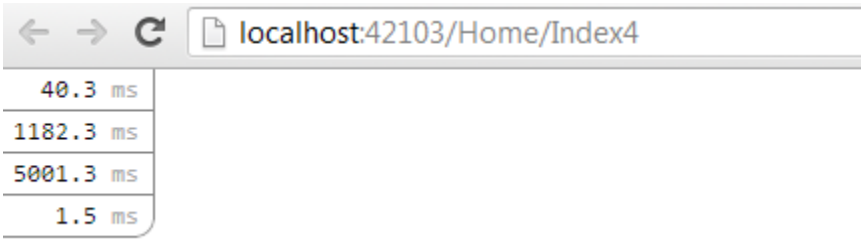
یک کنترلر به نام HomeController به شکل زیر می نویسیم.

```
public class HomeController : Controller
{
   public ActionResult Index()
   {
      return RedirectToAction("Index2");
```

```
    }
    public ActionResult Index2()
    {
        return RedirectToAction("Index3");
    }
    public ActionResult Index3()
    {
        System.Threading.Thread.Sleep(5000);

        return RedirectToAction("Index4");
    }
    public ActionResult Index4()
    {
        return View();
    }
}
```

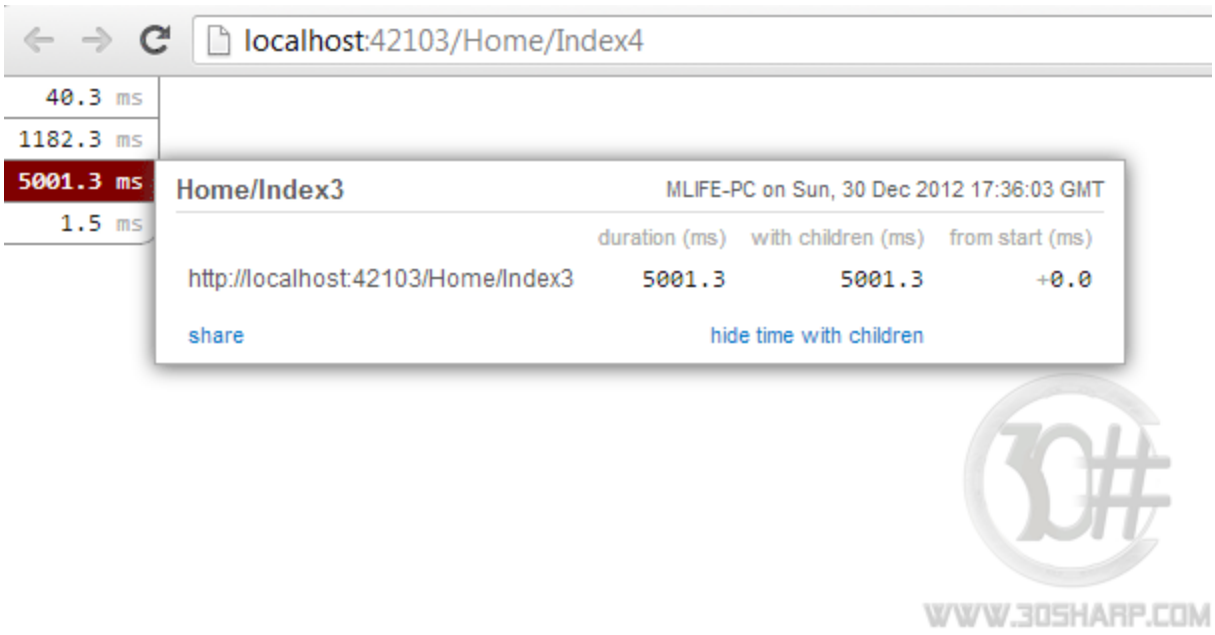
قطعه کد بالا عملکرد ساده ای دارد. هر کدام از اکشن ها، اکشن بعدی را فراخوانی می کنند و در نهایت اکشن چهارم یعنی Index4یک View خالی را رندر می کند. نکته جالب در قطعه کد بالا این است که در اکشن سوم یعنی Index3 ما عمدا یک تاخیر ۵ ثانیه این ایجاد نموده ایم.

اکنون برنامه را اجرا کنید. احتمالا با شکلی مشابه زیر مواجه شده اید.



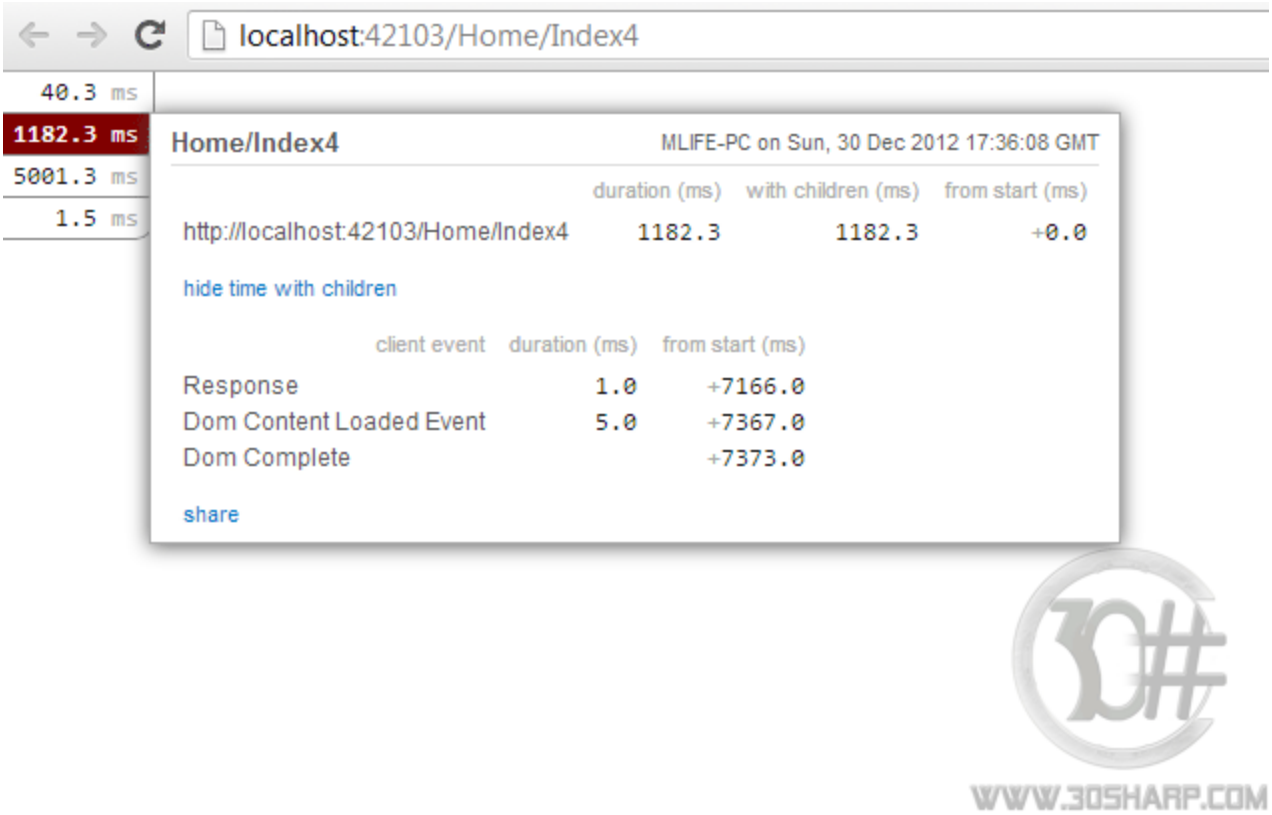
در شکل بالا مشاهده می کنید که ابزار MiniProfiler یک پنجره در بالا و سمت چپ صفحه وب ایجاد نموده است که دارای ٤ سطر می باشد و روی هر سطر یک عدد نوشته شده است. این سطر ها اکشن های اجرا شده بوده و عدد نوشته روی هرکدام، مدت زمانی است که در این اکشن صرف شده است.

روی سطر سوم که عدد ۵۰۰۱,۳ نوشته شده است کلیک کنید تا جزئیات بیشتر نمایان گردد. روی هر کدام از سطر ها کلیک نموده و جزئیات نمایش داده شده را ملاحظه کنید .



می بینید همانطور که انتظار داشتیم نام اکشن مربوطه Index3 بوده و مدت زمان تاخیر ۵ ثانیه ای نیز برای آن ثبت گردیده است.

با توجه به اینکه از میان اکشن های نوشته شده، فقط Index4 یک View را رندر کرده است، اگر روی آن کلیک کنید، جزئیات بیشتری را ملاحظه می کنید و هر چه پروژه شما بزرگتر و پیچیده تر باشد، جزئیات جالب تری مشاهده می گردد!



اکنون قصد داریم کدهای داخل یک اکشن را به قسمت های کوچک تر تقسیم نموده و مدت زمان هر قسمت را به طور جداگانه بررسی کنیم.

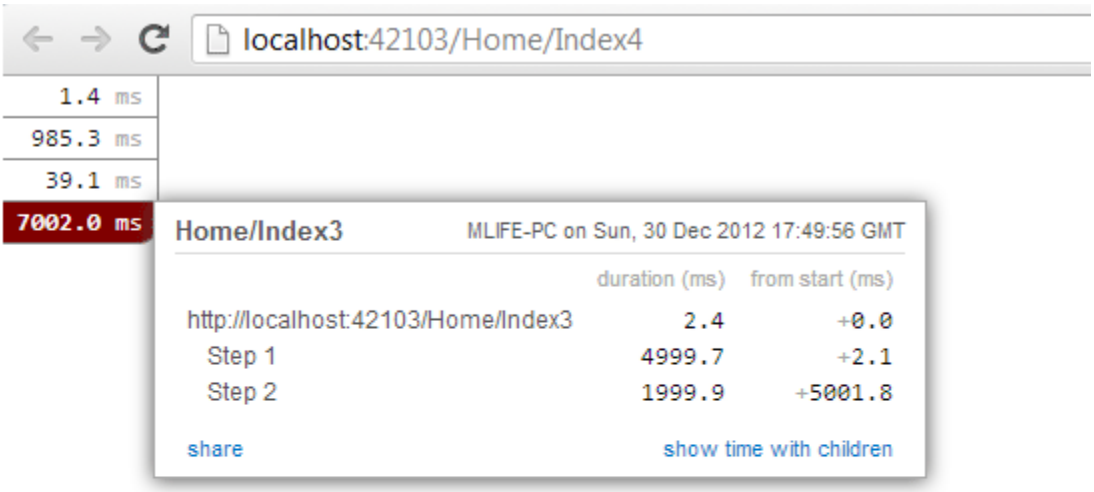
برای شبیه سازی این کار اکشن Index3 را به شکل زیر تغییر دهید.

```
public ActionResult Index3()
{

   var profiler = MiniProfiler.Current;

   using (profiler.Step("Step 1"))
   {
      System.Threading.Thread.Sleep(5000);
   }

   using (profiler.Step("Step 2"))
   {
      System.Threading.Thread.Sleep(2000);
   }
```

```
    return RedirectToAction("Index4");
}
```

در قطعه کد بالا ما از MiniProfiler.Current جهت بازیابی نمونه جاری از کلاس MiniProfiler استفاده کرده ایم. سپس داخل بلاک using قطعه کد ها را مجزا نموده ایم. اولین بلاک را Step 1 نامگذاری کرده و تاخیر ۵ ثانیه ای برایش در نظر گرفته ایم و بلاک دوم را Step 2 و میزان ۲ ثانیه تاخیر.

برنامه را اجرا نمایید.



ملاحظه می کنید این بار اکشن Index3 مدت زمانی حدود ۷ ثانیه را صرف نموده است و در جزئیات نیز می توانید مدت زمان صرف شده در هر Step را مشاهده کنید.

ابزار MiniProfiler امکانات جالب دیگری نیز جهت پروفایل نمودن دستورات اجرا شده بر روی پایگاه داده توسط تکنولوژی های LINQ to SQL و Entity Framework و ... را نیز دارد و حتی می تواند تعداد دستورات تکراری اجرا شده بر روی پایگاه داده را بررسی نموده و هشدار هایی را نمایش دهد. برای کسب اطلاعات بیشتر به وب سایت رسمی MiniProfiler مراجعه کنید.