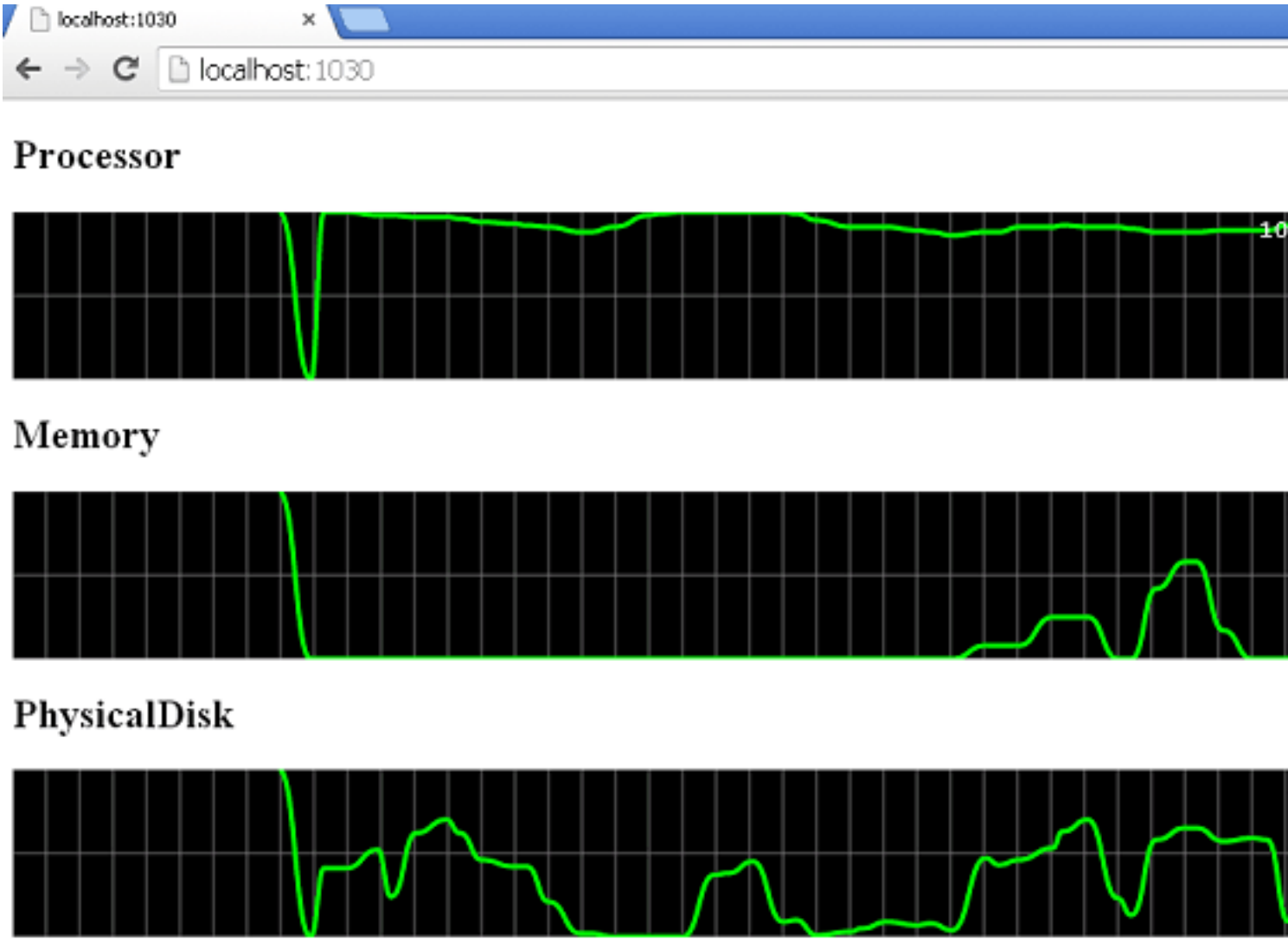
عنوان: **مثال - نمایش بلادرنگ میزان مصرف CPU و حافظه سرور بر روی کلیه کلاینت‌های متصل توسط SignalR**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۰۲ ۱۰:۳۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, jQuery, SignalR

یکی از کاربردهای جالب SignalR می‌تواند به روز رسانی مداوم صفحه نمایش کاربران، توسط اطلاعات ارسالی از طرف سرور باشد. در ادامه قصد داریم به عنوان منبع داده، آمار کارآیی سرور را به کلاینت‌ها ارسال کنیم و سپس به تصویری همانند شکل ذیل برسیم:



در اینجا از Smoothie Charts برای ترسیم نمودارهای بلادرنگ سازگار با Canvas مخصوص HTML 5 استفاده شده است.

پیشنیازها

پیشنیازهای این مطلب با مطلب « مثال - نمایش درصد پیشرفت عملیات توسط SignalR » یکی است. برای مثال، نحوه دریافت وابستگی‌ها، تنظیمات فایل global.asax و افزودن اسکریپت‌ها، تفاوتی با مثال قبلی ندارند.

تهیه منبع داده اطلاعات نمایشی

```
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;

namespace SignalR04.Common
{
    public class Counter
    {
        public string Name { set; get; }
        public float Value { set; get; }
    }

    public class PerformanceCounterProvider
    {
        private readonly List<PerformanceCounter> _counters = new List<PerformanceCounter>();

        public PerformanceCounterProvider()
        {
            _counters.Add(new PerformanceCounter("Processor", "% Processor Time", "_Total", readOnly:
true));
            _counters.Add(new PerformanceCounter("Memory", "Pages/sec", readOnly: true));
            _counters.Add(new PerformanceCounter("PhysicalDisk", "% Disk Time", "_Total", readOnly:
true));
        }

        public IList<Counter> GetResults()
        {
            return _counters.Select(c => new Counter
                                    {
                                        Name = c.CategoryName,
                                        Value = c.NextValue()
                                    }).ToList();
        }
    }
}
```

کلاس PerformanceCounterProvider، سه مؤلفه کارآیی سرور را بررسی کرده و هربار توسط متد GetResults، آن‌ها را بازگشت می‌دهد. از این منبع داده، در هاب برنامه استفاده خواهیم کرد.

تهیه هاب ارسال داده‌ها به کلاینت‌ها

```
using System.Threading;
using Microsoft.AspNet.SignalR;
using ThreadTimer = System.Threading.Timer;

namespace SignalR04.Common
{
    public class PerformanceCounterHub : Hub
    {
        private ThreadTimer _threadTimer; //keep it alive
        private readonly PerformanceCounterProvider _perfService = new PerformanceCounterProvider();

        public PerformanceCounterHub()
        {
            _threadTimer = new ThreadTimer(timerCallback, null, Timeout.Infinite, 1000);
            _threadTimer.Change(dueTime: 1000, period: 2000);
        }

        private void timerCallback(object state)
        {
            var results = _perfService.GetResults();
            this.Clients.All.newCounters(results);
        }
    }
}
```

در این هاب، یک thread timer ایجاد شده است که هر دو ثانیه یکبار، اطلاعات را از PerformanceCounterProvider دریافت و سپس با فراخوانی this.Clients.All.newCounters، آن‌ها را به کلیه کلاینت‌های متصل ارسال می‌کند.
این هاب به صورت خودکار با اولین بار وهله سازی، پس از فراخوانی متد connection.hub.start در سمت کلاینت، شروع به کار

می‌کند.

کدهای سمت کلاینت نمایش نمودارها

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="Scripts/jquery-1.6.4.min.js" type="text/javascript"></script>
    <script src="Scripts/jquery.signalR-1.1.3.min.js" type="text/javascript"></script>
    <script type="text/javascript" src='<%= ResolveClientUrl("~/signalr/hubs") %>'></script>
    <script src="Scripts/smoothie.js" type="text/javascript"></script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <div>
            <h2>Processor</h2>
            <canvas id="Processor" width="800" height="100"></canvas>
        </div>
        <div>
            <h2>Memory</h2>
            <canvas id="Memory" width="800" height="100"></canvas>
        </div>
        <div>
            <h2>PhysicalDisk</h2>
            <canvas id="PhysicalDisk" width="800" height="100"></canvas>
        </div>
    </div>
    </form>
    <script type="text/javascript">
        var ChartEntry = function (name) {
            var self = this;
            self.name = name;
            self.chart = new SmoothieChart({ millisPerPixel: 50, labels: { fontSize: 15} });
            self.timeSeries = new TimeSeries();
            self.chart.addTimeSeries(self.timeSeries, { lineWidth: 3, strokeStyle: "#00ff00" });
        };

        ChartEntry.prototype = {
            addValue: function (value) {
                var self = this;
                self.timeSeries.append(new Date().getTime(), value);
            },

            start: function () {
                var self = this;
                self.canvas = document.getElementById(self.name);
                self.chart.streamTo(self.canvas);
            }
        };

        $(function () {
            $.connection.hub.logging = true;
            var performanceCounterHub = $.connection.performanceCounterHub;
            var charts = [];
            performanceCounterHub.client.newCounters = function (updatedCounters) {
                $.each(updatedCounters, function (index, updateCounter) {
                    var entry;
                    $.each(charts, function (idx, chart) {
                        if (chart.name == updateCounter.Name) {
                            entry = chart;
                            return;
                        }
                    });

                    if (!entry) {
                        entry = new ChartEntry(updateCounter.Name);
                        charts.push(entry);
                        entry.start();
                    }
                    entry.addValue(updateCounter.Value);
                });
            };
            $.connection.hub.start();
        });
    </script>
</body>
```

```
</html>
```

- در ابتدا سه canvas بر روی صفحه قرار گرفته‌اند که معرف سه PerformanceCounter دریافتی از سرور هستند.
- id هر canavs به Name اطلاعات دریافتی از سرور تنظیم شده است تا نمودارها در جای صحیحی ترسیم شوند.
- سپس نحوه کپسوله سازی SmoothieChart را مشاهده می‌کنید؛ چطور می‌توان از آن یک شیء جاوا اسکریپتی ایجاد کرد و چطور اطلاعات را به آن اضافه نمود.
- نهایتا کار هاب را آغاز می‌کنیم. Callback ایی به نام performanceCounterHub.client.newCounters دقیقا متصل است به فراخوانی this.Clients.All.newCounters سمت سرور. در اینجا updatedCounters دریافتی، یک آرایه جاوا اسکریپتی است که هر عضو آن دارای Name و Value است. بر این اساس، تنها کافی است این مقادیر را که هر دو ثانیه یکبار به روز می‌شوند، به SmoothieChart برای ترسیم ارسال کنیم.

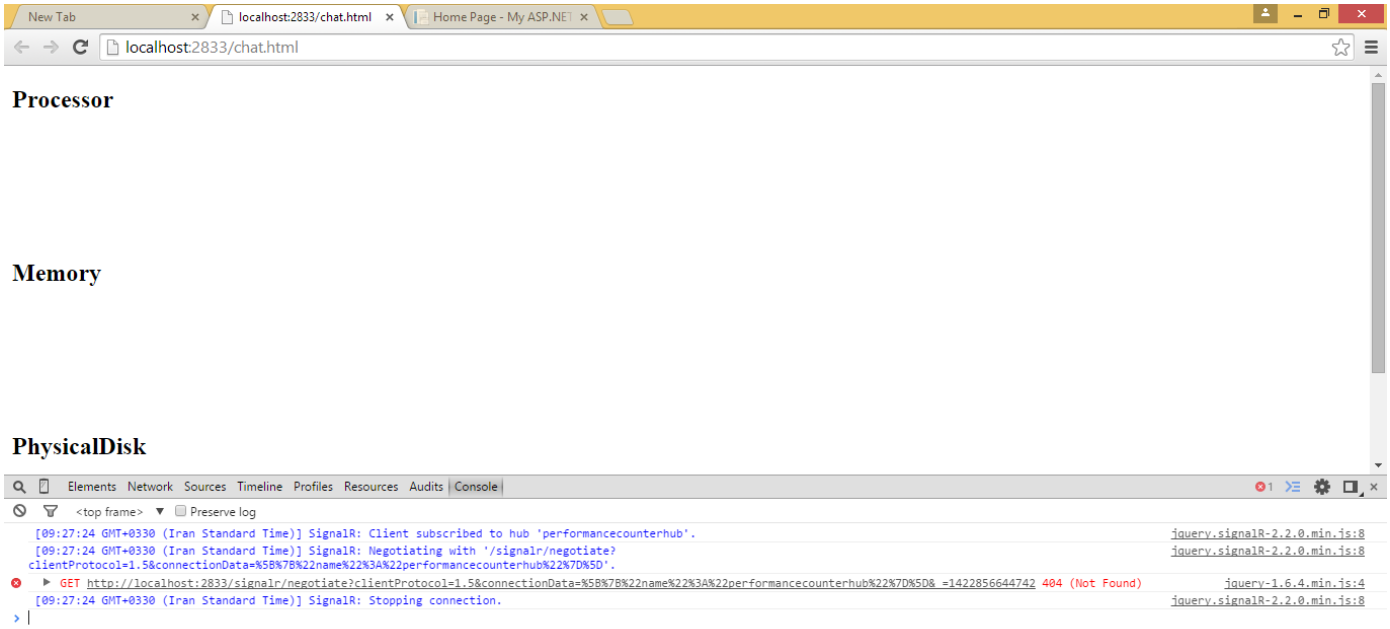
کدهای کامل این مثال را از اینجا نیز می‌توانید دریافت کنید:

SignalR04.zip

نظرات خوانندگان

نویسنده: عباس معینی
تاریخ: ۱۳۹۳/۱۱/۱۳ ۹:۴۹

با سلام؛ من مراحل بالا رو بطور کامل انجام دادم (از ورژن SignalR 2 استفاده کردم)
البته مثال‌های دیگه ای انجام دادم و جواب گرفتم ولی اینجا با این خطا روبرو شدم. ممنون میشم راهنمایی کنید



نویسنده: عباس معینی
تاریخ: ۱۳۹۳/۱۱/۱۳ ۱۰:۲۳

با سلام؛ مشکل حل شد.
راه حل: اضافه کردن این خط کد به صفحه‌ی کلاینت :

```
$.connection.hub.url = 'http://localhost:2663/signalr'; //چون در یک پروژه دیگر قرار داریم
```