عنوان: **چگونه نرم افزارهای تحت وب سریعتری داشته باشیم؟ قسمت اول**
نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۰۹ ۱۸:۰
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, Performance

در این سلسله مقالات قصد دارم چندین مطلب راجع به افزایش سرعت نرم افزارهای تحت وب مطرح نمایم. این مطالب هرچند بسیار مختصر می‌باشند ولی در کارایی و سرعت برنامه‌های شما در آینده تاثیر خواهند داشت.

1.کش کردن همیشه آخرین حربه می‌باشد

این مهم است که بخش‌های مختلف سایت شما در سطوح مختلف کش شوند (ASP.NET, Kernel, Server, Proxy Server, Browser ,...) ولی این موضوع باید همیشه آخرین حربه و نکته ای باشد که آن را در مورد سایت خود اعمال می‌کنید.
یعنی همیشه مطمئن شوید ابتدا تمامی نکات مربوط به افزایش کارایی در برنامه خود را رعایت کرده اید، سپس اقدام به کش داده‌ها در سطوح مختلف نمایید. توجه کنید کش کردن داده‌ها و صفحات می‌تواند مشکلات را برای شما به عنوان یک برنامه نویس یا تست کننده برنامه پنهان کند و به شما اطمینان دهد که همه چیز خوب کار می‌کند در حالی که این چنین نیست!
البته ذکر این نکته هم بی فایده نیست که کش کردن همه چیز بعضی مواقع دشمن برنامه شما محسوب می‌شود! هیچ وقت یادم نمی‌رود، در پورتال داخلی یک شرکت که در وقت استراحت به کارکنان اجازه مطالعه روزنامه‌های روز را می‌داد (به صورت آفلاین)، این نکته در بالای صفحه آورده شده بود: «لطفا برای به روز رساندن صفحات روزنامه‌ها از کلید Ctrl+F5 استفاده نمایید». این موضوع یعنی بحث کشینگ در برنامه آن پرتال در سطح فاجعه می‌باشد! حالا فرض کنید این مشکل در فرم ورود و یا مرور اطلاعات یک برنامه به وجود آید...

2.حذف View Engineهای غیر ضروری

به عنوان یک برنامه نویس ASP.NET MVC، باید اطلاع داشته باشید که CLR به صورت خودکار View Engineهای Razor و Web Forms را لود می‌کند. این موضوع به این دلیل است که اطلاعی از نحوه برنامه نویسی شما ندارد. اگر شما فقط از یکی از این دو View Engine استفاده می‌کنید،لطفا دیگری را غیر فعال کنید! فعال بودن هر دوی آنها یعنی اتلاف وقت گرانبهای CPU سرور شما برای رندر کردن تمامی صفحات شما توسط دو انجین! ایتدا viewهای شما با Web Forms Engine رندر شده سپس نتیجه به Razor Engine منتقل شده و مجدد توسط این انجین رندر می‌شود. این موضوع در سایت‌های با تعداد کاربر بالا یعنی فاجعه!
برای حل این مشکل کافی است خطوط زیر را در فایل Global.asax و در رویداد بخش Application_Start وارد نمایید:

```
ViewEngines.Engines.Clear();
ViewEngines.Engines.Add(new RazorViewEngine());
```

این دو خط یعنی خداحافظ Web Forms Engine...
قبل از استفاده از این کد، اطمینان حاصل کنید کل برنامه شما توسط Razor تهیه شده است وگرنه بنده هیچ مسئولیتی در رابطه با فریادهای کارفرمای شما متقبل نمی‌شوم!
صد البته برای حذف Razor Engine و استفاده از Web Form Engine می‌توان از کد زیر در همان موقعیت فوق استفاده کرد:

```
ViewEngines.Engines.Clear();
ViewEngines.Engines.Add(new WebFormViewEngine());
```

البته همانطور که حتما دوستان مطلع هستند امکان گسترش Engineهای فوق توسط ارث بری از کلاس BuildManagerViewEngine جهت ایجاد Engineهای دیگر همیشه محیا است. در این صورت می‌توانید تنها انجین سفارشی مورد نظر خود را لود کرده و از لود دیگر انجین‌ها پرهیز کنید.

3. استفاده از فایل‌های PDB مایکروسافت برای دیباگ و یا پروفایل کردن DLLهای دیگران

دیباگ یا پروفایل کردن برنامه ها، DLL ها، اسمبلی‌ها و منابعی از برنامه که شما آن را خود ننوشته اید (سورس آنها در دسترس شما نمی‌باشد) همیشه یکی از سخت‌ترین مراحل کار می‌باشد. جهت کمک به دیباگرها یا پروفایلرها، نیاز است فایل‌های PDB مرتبط با DLLها را در اختیار آنها قرار دهید تا به بهترین نتیجه دسترسی پیدا کنید. این فایل‌ها محتوی نام توابع، شماره خطوط برنامه و metadataهای دیگر برنامه اصلی قبل از optimize شدن توسط کامپایلر یا JIT می‌باشد. خوب حالا اگر نیاز شد این کار را در رابطه با DLLها و کلاس‌های پایه Microsoft.NET انجام دهیم چه کار کنیم؟

خیلی ساده! خود Microsoft سروری جهت این موضوع تدارک دیده که فایل‌های PDB را جهت دیباگ کردن در اختیار تیم‌های برنامه نویسی قرار می‌دهد.کافی است از منوی Tools گزینه Options را انتخاب، سپس به بخش Debugging و به بخش Symbols بروید و گزینه Microsoft Symbol Servers as your source for Symbols را انتخاب کنید. برای اطمینان از اینکه هر مرتبه که برنامه را دیباگ می‌کنید مجبور به دانلود این فایل‌ها نشوید، فراموش نکنید پوشه ای را جهت کش این فایل‌ها ایجاد و آدرس آن را در بخش Cache symbols in this directory همین صفحه وارد نمایید.

این امکان در Visual Studio 2010, 2012 در دسترس می‌باشد.

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۵/۱۰ ۱۷:۱۴

اثر View Engineهای اضافی رو با Glimpse بهتر میشه دید.

عنوان: **چگونه نرم افزارهای تحت وب سریعتری داشته باشیم؟ قسمت دوم**
نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۱۰ ۱۸:۵
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, Performance

قسمت اول

4. فشرده سازی HTTP را فعال کنید

اطمینان حاصل کنید که HTTP Compression در تمامی بخش‌های اصلی برنامه شما فعال است. حداقل کاری که می‌توانید در این رابطه بکنید این است که خروجی HTML که توسط برنامه شما تولید می‌شود را فشرده سازی کنید. جهت فعال سازی فشرده سازی در برنامه خود بهتر است در اولویت اول از ماژول ویژه ای که جهت این کار در IIS در نظر گرفته شده استفاده کنید. این ماژول تمامی کارها را به صورت خودکار برای شما انجام می‌دهد. اگر دسترسی به IIS جهت فعال سازی آن را ندارید، می‌توانید از ماژول‌های ASP.NET که جهت این کار تهیه شده استفاده کنید. می‌توانید کمی جستجو کنید و یا خودتان یکی تهیه کنید!

5.تنظیم CacheControlMaxAge

مقدار CacheControlMaxAge را در فایل web.config را طوری تنظیم کنید تا هیچ کاربری هیچ فایل static را دیگر درخواست نکند. مثلا می‌توانید این مقدار را بر روی چند ماه تنظیم کنید و البته فراموش نکنید این مقدار را در صفحات پویای خود بازنویسی (override) کنید تا مشکلی در رابطه با کش شدن فرم‌های اصلی برنامه (همانطور که در نکته اول بخش اول ذکر شد) پدید نیاید. البته کش کردن فایل‌های استاتیک برنامه بار مالی نیز برای شما و کاربرانتان خواهد داشت. دیگر هزینه پهنای باند اضافی جهت دانلود این فایل‌ها در هر درخواست برای شما (در سمت سرور) و کاربرانتان (در سمت کاربر) پرداخت نخواهد شد!

6. استفاده از OutputCache

اگر از MVC استفاده می‌کنید، فراموش نکنید که از OutputCache در کنترل‌های MVC استفاده نمایید. اگر سرور شما بتواند اطلاعات را از رم خود بازیابی کند بهتر از آن است که آن را مجدد از دیتابیس واکشی نماید و عملیاتی نیز بر روی آن انجام دهد. البته مدیریت حافظه .NET به صورت خودکار کمبود حافظه را مدیریت کرده و از نشت حافظه جلوگیری خواهد کرد. برای توضیحات بیشتر در این رابطه می‌توانید از این مقاله کمک بگیرید.

7. بهره برداری از ORM Profiler

ORM Profiler ها تمامی فعالیت‌های ORM تحت نظر گرفته، دستورات T-SQL ارسالی به بانک اطلاعاتی را واکشی کرده و برای شما نمایش می‌دهند. تعدادی از آنها نیز این دستورات را آنالیز کرده پیشنهاداتی در رابطه با بهبود کارایی به شما ارائه می‌دهند. برای مثال به جای اینکه شما 2000 رکورد را یکی یکی از بانک بازیابی کنید، می‌توانید آن را به صورت یک query به بانک ارسال کنید. این موضوع به سادگی توسط ORM Profilerها قابل بررسی است. نمونه ای از این نرم افزارها را می‌توانید در این سایت یا این سایت پیدا کنید. البته در صورتی که نمی‌خواهید از نرم افزارهای جانبی استفاده کنید، می‌توانید از ابزارهای توکار بانک‌های اطلاعاتی مانند SQL Profiler نیز استفاده کنید (راهنمایی).

عنوان: **چگونه نرم افزارهای تحت وب سریعتری داشته باشیم؟ قسمت سوم**
نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۱۱ ۱۲:۵
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, Performance

قسمت دوم

8.ORM Lazy Load

در هنگام استفاده از ORMها دقت کنید کجا از Lazy Load استفاده می‌کنید. Lazy Load باعث می‌شود وقتی شما اطلاعات مرتبط را از بانک اطلاعات واکشی می‌کنید، این واکشی اطلاعات در چند query از بانک انجام شود. درعوض عدم استفاده از Lazy Load باعث می‌شود تمامی اطلاعات مورد نیاز شما در یک query از بانک اطلاعاتی دریافت شود. این موضوع یعنی سربار کمتر در شبکه، در بانک اطلاعاتی، در منابع حافظه و منابع پر ارزش cpu در سرورها. البته استفاده از include در حالت فعال بودن یا نبودن lazy هم داستان مجزایی دارد که اگر عمری باقی باشد راجع به آن مقاله ای خواهم نوشت.
به این نمونه دقت کنید:

```
List<Customer> customers = context.Customers.ToList();
foreach (Customer cust in context.Customers){
   Console.WriteLine("Customer {0}, Account {1}", cust.Person.LastName.Trim() + ", " +
cust.Person.FirstName, cust.AccountNumber);
}
```

همچین کدی (در صورت فعال بودن Lazy Load در ORM) در صورتی که جدول Customers دارای 1000 رکورد باشد، باعث می‌شود برنامه 1001 دستور sql تولید و در بانک اجرا گردد.
برای اطلاع بیشتر می‌توانید به این مقاله مراجعه نمایید.

9.استفاده از MiniProfiler

سعی کنید از MiniProfiler در تمامی پروژه‌ها استفاده کنید. البته وقتی نرم افزار را در اختیار مصرف کننده قرار می‌دهید، آن را غیر فعال کنید. می‌توانید از متغیرهای compiler برای مجزا کردن buildهای متفاوت در برنامه خود استفاده کنید:

```
#if DEBUG then
// فعال سازی MiniProfiler
#endif
```

ایده دیگری هم وجود دارد. شما می‌توانید MiniProfiler را برای کاربر Admin یا کاربر Debugger فعال و برای بقیه غیر فعال کنید.
در باب MiniProfiler مسائل زیادی وجود دارد که چند نمونه از آن در همین سایت در این مقاله و این مقاله در دسترس است.
البته می‌توانید از ابزارهای دیگری مانند Glimpse که در این زمینه وجود دارد نیز استفاده کنید. لب کلام این نکته استفاده از profiler برای نرم افزار خود می‌باشد.

10. Data Paging در بانک اطلاعاتی

هنگامیکه از کامپوننت‌های شرکت‌های دیگر (Third party) استفاده می‌کنید، اطمینان حاصل کنید که صفحه بندی اطلاعات در بانک اطلاعاتی انجام می‌شود. برای نمونه کاپوننت گرید شرکت Telerik چند نوع صفحه بندی را پشتیبانی می‌کند. صفحه بندی سمت کاربر (توسط JavaScript)، صفحه بندی سمت سرور توسط کامپوننت و صفحه بندی مجازی. صفحه بندی سمت کاربر یعنی تمامی اطلاعات از سرور به کاربر فرستاده شده و در سمت کاربر عمل صفحه بندی انجام می‌شود. این یعنی واکشی تمامی اطلاعات از بانک و در مورد نرم افزارهای پرکاربر با حجم اطلاعات زیاد یعنی فاجعه. صفحه بندی سمت سرور ASP.NET هم یعنی واکشی اطلاعات از سرور بانک به سرور برنامه و سپس صفحه بندی توسط برنامه. این موضوع هم ممکن است مشکلات زیادی را ایجاد نماید چون باید حداقل تمامی رکوردها از اولین رکورد تا آخرین رکورد صفحه جاری از بانک واکشی شود که این عمل علاوه بر ایجاد سربار شبکه، سربار IO در بانک اطلاعاتی و سربار cpu در سرور ASP.NET ایجاد می‌کند. استفاده از صفحه بندی مجازی، شما را قادر می‌کند بتوانیم اطلاعات را در بانک صفحه بندی کرده و فقط صفحه مورد نظر خود را از بانک واکشی کنیم.

این حالت مجازی در اکثر componentها که توسط شرکت‌های مختلف ایجاد شده وجود دارد ولی ممکن است نام‌های متفاوتی داشته باشد. برای این موضوع باید به راهنمای component خریداری شده مراجعه کنید و یا به فروم‌ها و... مراجعه نمایید.

11. بررسی تعداد کوئری‌های صادر شده در یک صفحه و تعداد رکوردهای بازگشت داده شده توسط آن‌ها

این به این معنا نیست که برای هر query یک context مجزا ایجاد کنید، منظور این است که به بهانه اینکه اطلاعات مختلفی از جداول مختلف مورد نیاز است، query خود را آن قدر پیچیده یا گسترده ننویسیم که یا process آن در بانک زمان و سربار زیادی ایجاد کند و یا حجم اطلاعات بلا استفاده ای را از بانک به سرور برنامه لود نماید. به جای این موضوع می‌توانید در یک یا چند context دستورات مجزای واکشی اطلاعات صادر کنید تا تنها اطلاعات مورد نیاز خود را واکشی نمایید. البته این موضوع باعث نشود که تعداد queryها مثلا به 1000 عدد برسد! یعنی باید فیمابین queryهای پیچیده و queryهای ساده ولی با تعداد یکی را که مناسبتر با پروژه است انتخاب کنید که این موضوع با تجربه و تست حاصل می‌شود.

نظرات خوانندگان

نویسنده: جواد
تاریخ: ۱۳۹۲/۰۵/۱۲ ۳:۱۵

"استفاده از صفحه بندی مجازی، شما را قادر می‌کند بتوانیم اطلاعات را در بانک صفحه بندی کرده و فقط صفحه مورد نظر خود را از بانک واکشی کنیم. این حالت مجازی در اکثر componentها که توسط شرکت‌های مختلف ایجاد شده وجود دارد "

میشه این رو بیشتر توضیح بدید که منظورتون چیه .یا باید تمامی اطلاعات رو بفرستیم بعد صفحه بندی کنه یا اینکه به ازای هر صفحه یک کوئری به بانک بفرسته و اطلاعات رو نشون بده . حالا این صفحه بندی مجازی کجا کاربرد داره .

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۵/۱۲ ۱۵:۱۷

هیچ کامپوننتی وجود خارجی نداره که قسمت مدیریت سمت بانک اطلاعاتی رو هم خودش به تنهایی انجام بده. همین کنترل‌های پیش فرض ASP.NET رو هم اگر ازشون درست استفاده کنیم، مشکلات کارآیی ندارند. مثلا: (نکته مهمش Skip.Take.ToList استفاده شده هست)

واکشی اطلاعات به صورت chunk chunk (تکه تکه) و نمایش در ListView

نویسنده: مرتضی
تاریخ: ۱۳۹۲/۰۵/۱۲ ۲۱:۲۴

چرا وجود نداره!
از گرید Kendo استفاده کنید اگر paging رو فعال کنید خودش مدیریت میکنه

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۵/۱۲ ۲۲:۲۷

بله. مدیریت می‌کنه، نه به تنهایی. اینجا هم باید اطلاعات Skip و Take رو بهش بدی تا صفحه بندی کم هزینه‌ای رو اعمال کنه . خود GridView در وب فرم‌ها هم paging داره. مشکلش اینه که در حالت پیش فرض کل اطلاعات رو از سرور واکشی می‌کنه و بعد یک صفحه رو نمایش می‌ده. بحث اینه که گرید اگر قراره یک صفحه رو نمایش بده که 20 ردیف داره، بتونه فقط 20 رکورد رو واکشی کنه و نه کل اطلاعات رو و این نیاز به کوئری‌های خاصی در سمت سرور داره. یک نمونه‌اش رو در واکشی اطلاعات به صورت تکه تکه لینک دادم.

نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۱۲ ۲۲:۴۶

1. اکثر کنترل‌های ASP.NET WebForm قابلیت bind به DataSet را دارد. اگر از آنها در این مدل استفاده نمایید کار صفحه بندی به عهده کنترل+DataSet می‌افتد.
2. صفحه بندی مجازی یعنی شما به کنترل می‌گویید کل اطلاعات شما مثلا 51 صفحه است، الان صفحه 4 را نمایش می‌دهی این هم اطلاعات صفحه 4. بعد وقتی کاربر بر روی گزینه صفحه بعد کلیک کرد، به کنترل می‌گویید کل اطلاعات 51 صفحه است، الان صفحه 5 را نمایش می‌دهی و این هم اطلاعات آن.
برای مثال می‌توانی به این مثال در DataTables مراجعه کنید

نویسنده: مرتضی

تاریخ: ۱۳۹۲/۰۵/۱۳ ۰:۱۳

گفتم که خودش مدیریت می‌کنه **یعنی اینکه اطلاعات Take , Skip رو نمی‌خواد بهش بدی و خودش اینکار رو انجام میده** - من دارم ازش تو پروژم استفاده میکنم - **کل اطلاعات رو واکشی نمی‌کنه و همون 20رکورد فرضا صفحه 3 رو واکشی میکنه**

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۵/۱۳ ۰:۳۱

می‌تونی یک مثال با پروفایل SQL نهایی آن ارائه بدی. مطابق بررسی که کردم و حداقل دو تا لینکی که دادم در مورد این کتابخانه، موارد پردازش Take و Skip سمت سرور اون خودکار نیست.

نویسنده: مرتضی
تاریخ: ۱۳۹۲/۰۵/۱۳ ۴:۴۹

کد Razor - کد VB - خروجی SQL

```
@(Html.Kendo.Grid(Of Models.vProject).Name("ProjectsGrid") _
        .Columns(Sub(column)
                     With column
                         .Bound(Function(p) p.ProjectId).Hidden()
                         .Bound(Function(p) p.Supervisor).Title("ناظر")
                         .Bound(Function(p) p.MapNumber).Title("شماره نقشه")
                         .Bound(Function(p) p.MapCode).Title("کد نقشه")
                         .Bound(Function(p) p.NewStructureArea).Title("متراژ")
                         .Bound(Function(p) p.NumberOfFloors).Title("تعداد طبقات")
                         .Bound(Function(p) p.InsuranceName).Title("بیمه")
                     End With
                End Sub).Pageable(Sub(p)
                                      p.Enabled(True)
                                      p.Info(True)
                                      p.PageSizes(True)
                                      p.Messages(Sub(m)
                                                     m.Display("{0} - {1} رکورد {2}از ")
                                                     m.Empty("رکوردی برای نمایش وجود ندارد")
                                                     m.Of("از")
                                                     m.Page("صفحه")
                                                     m.ItemsPerPage("رکورد در هر صفحه")
                                                     m.Refresh("بروزرسانی")
                                                 End Sub)
                                  End Sub).Selectable(Sub(s)
s.Enabled(True).Mode(GridSelectionMode.Single).Type(GridSelectionType.Row)) _
                                              .DataSource(Sub(ds)
ds.Ajax.ServerOperation(True).Read("GetProjects", "Home") _
                                                                  .Model(Sub(m) m.Id(Function(modelId)
modelId.ProjectId)))
    )
```

```
<HttpPost>
        Function GetProjects(<DataSourceRequest> request As DataSourceRequest) As JsonResult
            Return Json(db.vProjects.ToDataSourceResult(request))
        End Function
```

```
SELECT TOP (5) [Extent1].[ProjectId]        AS [ProjectId],
               [Extent1].[Supervisor]       AS [Supervisor],
               [Extent1].[MapCode]          AS [MapCode],
               [Extent1].[MapNumber]        AS [MapNumber],
               [Extent1].[EmployerName]     AS [EmployerName],
               [Extent1].[InsuranceName]    AS [InsuranceName],
               [Extent1].[NewStructureArea] AS [NewStructureArea],
               [Extent1].[NumberOfFloors]   AS [NumberOfFloors]
FROM   (SELECT [Extent1].[ProjectId]                                    AS [ProjectId],
               [Extent1].[Supervisor]                                   AS [Supervisor],
               [Extent1].[MapCode]                                      AS [MapCode],
               [Extent1].[MapNumber]                                    AS [MapNumber],
               [Extent1].[EmployerName]                                 AS [EmployerName],
               [Extent1].[InsuranceName]                                AS [InsuranceName],
               [Extent1].[NewStructureArea]                             AS [NewStructureArea],
               [Extent1].[NumberOfFloors]                               AS [NumberOfFloors],
               row_number() OVER (ORDER BY [Extent1].[ProjectId] ASC) AS [row_number]
        FROM   (SELECT [vProject].[ProjectId]        AS [ProjectId],
                       [vProject].[Supervisor]       AS [Supervisor],
```

```
                        [vProject].[MapCode]          AS [MapCode],
                        [vProject].[MapNumber]        AS [MapNumber],
                        [vProject].[EmployerName]     AS [EmployerName],
                        [vProject].[InsuranceName]    AS [InsuranceName],
                        [vProject].[NewStructureArea] AS [NewStructureArea],
                        [vProject].[NumberOfFloors]   AS [NumberOfFloors]
                FROM    [dbo].[vProject] AS [vProject]) AS [Extent1]) AS [Extent1]
WHERE  [Extent1].[row_number] > 5
ORDER  BY [Extent1].[ProjectId] ASC
```

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۵/۱۳ ۸:۲۰

کد سمت سرور db.vProjects.ToDataSourceResult چطور تهیه شده؟ در موردش اینجا بحث شده . شما یک IQueryable باید در اختیارش قرار بدی (که از لحاظ لایه بندی کار مشکل داره) تا بر اساس اطلاعات شماره صفحه و غیره‌ای که از کلاینت می‌رسه خودش مباحث Take و Skip رو پیاده سازی کنه. در حقیقت این کتابخانه فقط یک متد الحاقی اضافه‌تر برای اینکار جهت مدیریت مباحث سمت سرور داره.

نویسنده: مرتضی
تاریخ: ۱۳۹۲/۰۵/۱۳ ۱۳:۴۵

درسته محسن خان- متد الحاقی ToDataSourceResult در خواست رو می‌گیره و....
بحث سر این بود که هیچ کامپوننتی وجود خارجی نداره که قسمت مدیریت سمت بانک اطلاعاتی رو هم خودش به تنهایی انجام بده
آره شیء DataSourceRequest شامل PageSize , PageNumber , ... میشه
ولی دیگه ما کاری بهش نداریم و خودش صفحه بندی رو انجام می‌ده حالا به طریقی
نمیشود گفت اگر ما از IQueryable استفاده کردیم **حتما لایه بندی ما مشکل داره**

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۵/۱۳ ۱۴:۱۷

داره. بهش می‌گن leaky abstraction .

نویسنده: مرتضی
تاریخ: ۱۳۹۲/۰۵/۱۳ ۱۴:۲۴

درسته بهش میگن leaky abstraction اما نمیگن 100% ایراد

عنوان: **چگونه نرم افزارهای تحت وب سریعتری داشته باشیم؟ قسمت چهارم**
نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۱۲ ۱۵:۵
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, Performance

قسمت سوم

12.استفاده از validation سمت کاربر

برای جلوگیری از ارسال و دریافت‌های متناوب اطلاعات به سرور، از validation سمت کاربر استفاده نمایید. فرم‌های html 5 قابلیت‌های چک کردن نوع ورودی‌ها را به صورت خودکار دارد ولی ازاتکای به آن پرهیز کنید چون ممکن است یا کاربران برنامه شما از مرورگری استفاده کنند که از html5 پشتیبانی نکند و یا پشتیبانی کاملی از آن نداشته باشند. برای حل این مشکل می‌توانید از کتابخانه هایی مانند JQuery و ابزارهایی مانند JQuery Validation استفاده کنید. البته در MVC استفاده وسیعی از JQuery Validation شده که می‌تواند مورد استفاده قرار گیرد.

فراموش نکنید می‌توانید از ابزارهایی مانند Regex برای چک کردن سختی کلمات عبور و... نیز در JavaScript بهره برداری نمایید. البته دقت کنید که حتما پیامی مرتبط با خطای به وقوع پیوسته در اختیار کاربر قراردهید تا بتواند آن را بر طرف کند در غیر این صورت بنده مسئولیتی راجع به از دست دادن کاربرانتان و یا عصبانیت کارفرما بر عهده نمی‌گیرم!

13.استفاده از validation سمت سرور

حتما به خود می‌گویید نویسنده دچار چندگانگی شخصیت شده است! ولی چنین نیست. این مطلب بیشتر از اینکه در رابطه با ایجاد سرعت بیشتر باشد مربوط به امنیت است. چون validation سمت کاربر به سادگی قابل دور زدن می‌باشد. اگر شما تنها validation را سمت کاربر انجام دهید و سمت سرور از آن چشم پوشی کنید، به سرعت تمام برنامه شما هک می‌شود. لطفا دقت کنید که **امنیت را فدای هیچ چیز نکنید.** این یک نکته کلیدی است. البته سوای اینکه این یک نکته امنیتی است، validation سمت سرور باعث می‌شود شما بخشی از درخواست‌ها را قبل از انجام process زیاد از گردونه خارج کنید و از ارسال اطلاعات اضافی به بانک و ایجاد سربار اضافی جلوگیری کنید.

14.چک کردن scriptهای مورد استفاده سمت کاربر

استفاده از master pageها بسیار سرعت کار را زیاد می‌کنند. بیشتر دوستان scriptهای سمت کاربر خود را در master page قرار می‌دهند تا در تمامی صفحات لود شوند. این موضوع از طرفی سرعت برنامه نویسی را زیاد می‌کند ولی از طرف دیگر به دلیل اینکه باعث می‌شود فایل‌های script در تمامی صفحات بارگذاری شوند، باعث هدر رفت منابع شبکه شما (و کاربرانتان)، ایجاد سربار حافظه و cpu در سمت کاربر و در نتیجه سرعت پایین‌تر برنامه شما خواهد شد. سخت گیری در این موضوع می‌تواند این باشد که حتی شما function اضافی هم در سمت کاربر نداشته باشید.

برخی ناظران پروژه به این موضوعات دقت زیادی می‌کنند. در پروژه ای که به عنوان ناظر بودم مجری همین کار را انجام داده بود و به دلیل نیاز مبرم کارفرما به سرعت برنامه، این بخش از نظر اینجانب مردود اعلام شده و مجری مجبور به نوشتن دوباره کدهای آن گردید.

عنوان: **چگونه نرم افزارهای تحت وب سریعتری داشته باشیم؟ قسمت پنجم**
نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۱۳ ۲۳:۵
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, Performance

قسمت چهارم

15.استفاده از using

اگر از objectهایی استفاده می‌کنید که interface مربوط به IDisposable را پیاده سازی کرده اند، حتما از عبارت using استفاده کنید. استفاده از دستور using باعث می‌شود زمانی که دیگر نیازی به object شما نباشد، به صورت خودکار از حافظه حذف شود و در روال جمع آوری زباله (GC) قرار گیرد. این عمل باعث حداقل رسیدن احتمال نشت حافظه در نرم افزار شما می‌شود. برای مثال:

```
using System;
using System.Text;

class Program
{
    static void Main()
    {
// Use using statement with class that implements Dispose.
using (SystemResource resource = new SystemResource())
{
    Console.WriteLine(1);
}
Console.WriteLine(2);
    }
}

class SystemResource : IDisposable
{
    public void Dispose()
    {
// The implementation of this method not described here.
// ... For now, just report the call.
Console.WriteLine(0);
    }
}
```

برای اطلاعات بیشتر می‌توانید از این مقاله استفاده کنید.

16.اطلاعات ارسالی توسط شبکه را به حداقل برسانید

حجم اطلاعات ارسالی به شبکه را به حداقل برسانید. ارسال اطلاعات در شبکه به معنی گذر اطلاعات شما از 7 لایه مختلف شبکه در رایانه شما، گذر از media شبکه، گذر مجدد از 7 لایه شبکه در رایانه مقصد می‌باشد. به این معنی که هرچه اطلاعات بیشتری در شبکه ارسال کنید، سربار بیشتری متوجه سیستم شما خواهد بود. برای رفع این مشکل از فشرده سازهای css و javascript استفاده کنید. این فشرده سازها فواصل خالی، دستورات اضافی و... را از کد شما حذف و حجم آن را به حداقل می‌رسانند.
کم کردن تعداد درخواست‌ها و در نتیجه کم کردن تعداد فایل‌های ارسالی از سرور به کاربر نیز حربه ای در این زمینه می‌باشد.
برای مقایسه فشرده سازها به صورت آنلاین و استفاده از بهترین آنها (متناسب کد شما) می‌توانید از این سایت استفاده کنید.
امکانات توکاری هم وجود دارد که در زمان اجرای برنامه css و javascript شما را فشرده و تلفیق می‌کند ولی با توجه به اینکه برای سرور در هر مرتبه فراخوانی سربار دارد (حتی در صورت کش کردن) اکیدا توصیه می‌شود از فشرده سازها قبل از اجرای برنامه (Pre-Compressed) به جای فشرده سازهای زمان اجرا (Run-time Compression)استفاده کنید.

نظرات خوانندگان

نویسنده: حسین حقیان
تاریخ: ۱۳۹۲/۰۵/۱۶ ۱۲:۳۸

با سلام و تشکر از مجموعه مطالب مرتبط که ارائه کردید
میشه برای این قسمت مثالی رو ذکر بفرمایید
اکیدا توصیه می‌شود از فشرده سازها قبل از اجرای برنامه (Pre-Compressed) به جای فشرده سازهای زمان اجرا (Run-time Compression)استفاده کنید.

با تشکر

نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۱۶ ۱۹:۵۱

فشرده سازی قبل از اجرای برنامه (Pre-Compression) یعنی که شما قبل از اینکه برنامه خود را در محیط اصلی نصب و اجرا کنید، فایل‌های اسکریپت آن را فشرده کنید. یعنی کاربران فایل اسکریپت فشرده شده را درخواست و دانلود می‌کنند و عملیات اضافی در سمت سرور انجام نمی‌شود. به عنوان مثال شما از فایل JQuery.min.js به جای jquery.js استفاده کنید. یعنی استفاده از نسخه فشرده شده اسکریپت ها.
فشرده سازی زمان اجرا (Run-time Compression) یعنی فشرده سازی اسکریپت‌های مورد نیاز کاربر توسط خود برنامه وب (به صورت خودکار و یا توسط یک ماژول اضافی). این عمل باعث می‌شود که در هر بار درخواست هر کاربر برای یک فایل، برنامه آن را مجدد فشرده سازی کند (و یا از cache استفاده کند). این عمل به معنی استفاده بیشتر از منابع پر ارزش سرور شما می‌باشد. به عنوان مثال شما بخواهید در هر مرحله درخواست هر کاربر jquery.js را فشرده کنید!

از مقایسه دو حالت بالا مشخص است وقتی شما فقط یکبار اسکریپت‌های خود را فشرده می‌کنید بسیار از حالتی که در هر مرتبه از درخواست کاربران آن را فشرده کنید بهتر است و کمتر منابع سرور را هدر می‌دهد.

عنوان: **چگونه نرم افزارهای تحت وب سریعتری داشته باشیم؟ قسمت ششم**
نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۱۵ ۱۱:۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Performance

قسمت پنجم

17. پرهیز از استفاده نسخه debug

وقتی به ASP.NET مراجعه می‌کنید، توجه فرمایید که از چه نوع build برای محصول نهایی استفاده می‌کنید. وقتی از نسخه debug برنامه استفاده می‌کنید، بهبود دهنده‌های سطح کامپایلر عمل نکرده و کدشما در حالت بهینه اجرا نخواهد شد (کد شما همانگونه که هست اجرا می‌شود!).

برای مثال هنگامی که از نسخه release استفاده می‌کنید، کامپایلر c# به صورت خودکار از StringBuilderها به جای تلفیق عادی رشته ها، از آرایه‌ها به جای لیست ها، از دستور switch/case به جای دستورات if/then/else، تلفیق شروط با یکدیگر و... استفاده کرده و کد شما را در حالت بهینه‌تری اجرا می‌کند. عدم استفاده از این نسخه شما را از این مزایا محروم می‌سازد و نرم افزار شما به کندی اجرا خواهد شد. البته ناگفته نماند این موضوع فقط باید برای محصول نهایی استفاده شود و جهت دیباگ کردن برنامه همچنان باید از نسخه debug استفاده نمایید.

توجه نمایید می‌توانید با استفاده از متغیرهای کامپایلر در کد خود بخشی از کد را مختص build خاصی از برنامه کنید. مثلا اگر برنامه در حال debug کامپایل شد، MiniProfiler را فعال کن در غیر این صورت غیر فعال باشد.

```
#if DEBUG
    //فعال کردن MiniProfiler
#endif
```

18.تنظیم دقیق لاگ‌های سیستم در محیط اجرا

وقتی محصول نهایی را آماده می‌کنید، فراموش نکنید که سطح لاگ گیری را در سطح مطلوبی قرار دهید تا بتوانید در صورت نیاز برنامه را اشکال زدایی کنید. البته زیاده روی در این مورد نیز می‌تواند مشکل زا باشد.

اکثر برنامه نویسان هنگامی که محصول نهایی را برای مشتری آماده می‌کنند، لاگ را غیر فعال می‌کنند تا کاربر سرعت بیشتری را تجربه کند. این سیاست غلط شما را از امکانات بی نظیر لاگ کردن (مانند وقایع نگاری امنیتی، رفع سریع مشکلات و...) محروم می‌سازد. بنابر این حتما سیستم لاگ خود را در زمان تولید محصول اصلی (و نصب بر روی سرور اصلی) در حالت متعادلی تنظیم نمایید. کمی تست و تجربه شما را در این امر یاری می‌کند.

19.مشخص کردن اندازه عکس

مشخص کردن اندازه عکس در تک img به صورت css یا attribute باعث می‌شود که همان اولین بار که صفحه رندر می‌شود، اندازه مورد نیاز عکس به آن اختصاص یابد تا در صورت دانلود سریعا جایگزین آن گردد. عدم مشخص کردن سایز عکس (طول و عرض) باعث رندر شدن مجدد تمامی المان‌های صفحه بعد از دانلود هر عکس از سرور می‌شود و منابع با ارزش cpu کاربر شما را به سادگی از بین می‌برد.

```
<img src="smiley.gif" alt="Smiley face" height="42" width="42">
```

نظرات خوانندگان

نویسنده: مرتضی
تاریخ: ۱۳۹۲/۰۷/۰۷ ۱۶:۵۴

سلام. ممنون از مطلب خوبتون. بنده وقتی سایتم رو با ویژوال استادیو باز میکنم فقط مود دیباگ داره و مود release رو نداره. فقط برای برنامه‌های ویندوز فرم و wpf مود release داره. پس با این اوصاف بنده چطوری سایتم رو تویه مود release پاپلیش کنم؟ فقط کد زیر رو تویه وب کانفیگ قرار میدم

```
<compilation debug="false" targetFramework="4.0">
```

آیا قطعه کد بالا همون کار release رو انجام میده. با تشکر

نویسنده: م منفرد
تاریخ: ۱۳۹۲/۱۰/۱۴ ۰:۱۰

در بخش publish از منوی debug می‌توانید نوع خروجی را مشخص کنید که برنامه با debug پابلیش شود یا realease

عنوان: **چگونه نرم افزارهای تحت وب سریعتری داشته باشیم؟ قسمت هفتم**
نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۱۶ ۲۰:۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Performance

قسمت ششم

20.اسکریپت در پایین صفحه

لینک‌های مربوطه به javascriptهای خود را تا جای ممکن در پایین صفحه قرار دهید. وقتی parser مرورگر به فایل‌های javascript می‌رسد، تمامی فعالیت‌ها را متوقف کرده و سعی در دانلود و سپس اجرای آن دارد. برخلاف اینکه مرورگرها امکان دانلود چند فایل را به صورت همزمان از سرور دارند، هنگامی که به اسکریپت‌ها می‌رسند، تنها یک فایل را دانلود می‌کنند. یعنی اجرای برنامه و دانلودهای مرتبط با صفحه شما متوقف شده و پس از دانلود و اجرای اسکریپت اجرای آنها ادامه پیدا می‌کند. این مسئله وقتی نمود بیشتری پیدا می‌کند که شما فایل هغای اسکریپت با حجم و تعداد بالا در برنامه خود استفاده می‌کنید.
برای فرار از این مشکل می‌توانید تگهای مربوط به اسکریپت را در آخر صفحات خود بگذارید. فقط دقت کنید اگر نیاز است که قبل از نمایش صفحه تغییری ذر DOM ایجاد کنید، باید حتما اسکریپت‌های مربوطه را بالای صفحه قرار دهید.
روش دیگر دانلود فایل‌های اسکریپت به وسیله AJAX است که انشاء الله در آینده مقاله ای در این رابطه خواهم نوشت.

CDN.21

CDN یا Content Delivery Network سرورهای توزیع شده ای در سطح دنیا هستند که یک نسخه از برنامه شما برای اجرا بر روی آن قرار دارد. هنگامی که کاربر می‌خواهد به سایت شما دسترسی پیدا کند به صورت خودکار به نزدیکترین سرور منتقل می‌شود تا بتواند سرعت بیشتری را تجربه کند. علاوه بر این CDN باعث بالانس شدن بار ترافیک شبکه شما شده خط حملات D.O.S و D.D.O.S را به حداقل می‌رساند.

زمانیکه شما یک سیستم CDN را فعال میکنید تاثیر آن بصورت زیر خواهد بود:
۱- شبکه توزیع محتوا یا همان CDN تمامی سرورهای شبکه جهانی اینترنت را پوشش میدهد. بنابراین زمانیکه شما این سیستم را برای سایت خود فعال میکنید اطلاعات شما بر روی تمامی این سرورها کپی و ذخیره میشود و زمانیکه یک بازدیدکننده به سایت یا وبلاگ شما وارد میشود محتوای سایت شامل تصاویر و متون را از نزدیک‌ترین سرور نزدیک به خود دریافت میکند و مستقیما به هاست یا سرور شما متصل نمیشود. این کار موجب بهبودی چشمگیر در عملکرد سایت شما خواهد شد.
۲- CDN تمام اطلاعات ثابت شما مانند تصاویر، کدهای CSS و javascript، mp3، pdf و فایلهای ویدئویی شما را پشتیبانی میکند و تنها اطلاعاتی که قابل تغییر و بروزرسانی هستند مانند متون و کدهای HTML از سرور اصلی شما فراخوان میشوند. با این کار مصرف پهنای باند هاست شما کاهش یافته و هزینه ای که سالانه برای آن میپردازید کاهش چشمگیری خواهد داشت.
۳- تفاوت سرعت و عملکرد برای خودتان یا افرادی که در نزدیکی سرور اصلی شما هستند تفاوت زیادی نخواهد داشت ولی برای کسانی که ار نقاط مختلف جهان به سایت شما وارد میشوند این افزایش سرعت ناشی از CDN کاملا محسوس خواهد بود، با توجه به اینکه سایتهای ایرانی معمولا سرور و هاست خود را از خارج و کشورهایی مانند آلمان و آمریکا تهیه میکنند و عموم بازدیدکنندگان

از داخل کشور هستند استفاده از CDN میتواند بسیار موثر باشد. برای تعیین تاثیر CDN بر سرعت سایت میتوانید عملکرد خود را با ابزارهایی مانند Pingdom و GTmetrix بعد و قبل از فعال سازی CDN بررسی و مقایسه کنید.
ابزارها، تکنیک‌ها و روش‌های متفاوتی برای راه اندازی CDN وجود دارد که نیازمند مقاله ای مجزا می‌باشد.

عنوان: **چگونه نرم افزارهای تحت وب سریعتری داشته باشیم؟ قسمت هشتم**
نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۲۸ ۲۱:۵
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, Performance

قسمت هفتم

22. استفاده از CSS Sprites

ایده اصلی این تکنیک به این صورت است که تمامی عکس‌های کوچک (دراینجا همه 100 عکس) در قالب یک تصویر بزرگ قرار خواهد گرفت و با استفاده از CSS مختصات هر عکس کوچک را در تصویر بزرگ پیدا کرده و نمایش می‌دهیم. یکی شدن 100 عکس کوچک به یک عکس بزرگ، تاثیر زیادی در پایین آمدن حجم عکس جدید خواهد داشت و مرورگر شما به جای درخواست 100 عکس از سرور، تنها یکی دانلود می‌کند و از این به بعد از کش مرورگر برای بازیابی آن استفاده می‌کند. این موضوع به معنی ترافیک کمتر شبکه و آزاد شدن منابع پر ارزش حافظه، cpu و پهنای باند در سمت سرور و کاربران. برای اطلاع بیشتر از این تکنیک می‌توانید به این مقاله مراجعه نمایید.

23. استفاده مطلوب از AJAX

شما می‌توانید برای لود کردن بخش‌های مخفی در صفحه خود از AJAX کمک بگیرید. به جای دانلود کردن تمامی بخش‌های صفحه در مرورگر کاربر، بخش هایی که در دید کاربر قرار ندارد را به صورت AJAX بارگیری کنید. نمونه ای از این تکنیک را در این صفحه مشاهده نموده و البته از کد آن استفاده نمایید.

24. حذف HTTP modules های اضافی

HTTP modules هایی را که در برنامه خود استفاده نمی‌کنید را حذف کنید. این کار یعنی سربار مدیریتی کمتر در ماژول ASP.NET سرور شما. برای اجرای این مورد می‌توانید از کدی مشابه این کد در web.config خود استفاده کنید:

```
<httpModules>
    <remove name="OutputCache"/>
    <remove name="Session"/>
    <remove name="WindowsAuthentication"/>
    <remove name="FormsAuthentication"/>
    <remove name="PassportAuthentication"/>
    <remove name="RoleManager"/>
    <remove name="UrlAuthorization"/>
    <remove name="FileAuthorization"/>
    <remove name="AnonymousIdentification"/>
    <remove name="Profile"/>
    <remove name="ErrorHandlerModule"/>
    <remove name="ServiceModel"/>
</httpModules>
```

البته حواستان به این موضوع باشد ماژول‌های مورد استفاده در برنامه خود را حذف نکنید که در این صورت ممکن است این آخرین پروژه شما با صاحب کارتان باشد!