عنوان: **متغیرهای استاتیک و برنامه‌های ASP.NET**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۱۸ ۱۹:۳۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net

هر متغیر استاتیک تنها دارای یک مقدار، در یک AppDomain مشخص است (مگر اینکه با ویژگی ThreadStatic مزین شود). هر برنامه‌ی ASP.NET هم AppDomain جداگانه و منحصر به خود را دارا است. بنابراین تعریف یک متغیر استاتیک در یک برنامه‌ی ASP.NET به معنای به اشتراک گذاری آن در بین تمامی درخواست‌های رسیده به سرور است. بنابراین عموما استفاده از متغیرهای استاتیک در برنامه‌های چند کاربره ASP.NET یک اشتباه بزرگ است و در صورت استفاده از آن باید منتظر تخریب اطلاعات یا دریافت نتایج غیرمنتظره‌ای باشید (مگر اینکه واقعا می‌دانید دارید چکار می‌کنید، برای مثال کش کردن نگاشت‌های NHibernate به این صورت و استفاده از الگوی singleton یا روش‌های مشابه که باید بین تمام کاربران به یک صورت و یک شکل به اشتراک گذاشه شود و در حین اجرای برنامه تغییری در آن حاصل نمی‌شود). برای مثال اگر کاربر یک، در صفحه‌ی یک، متغیر استاتیکی را مقدار دهی کند، کاربر 2 نیز با مقدار به روز شده‌ی کاربر یک کار خواهد کرد که به طور قطع این مورد مد نظر شما نیست (چون به احتمال زیاد طراحی شما بر اساس کار کاربر در یک Session است و نه یک مقدار برای تمام سشن‌های موجود در سایت) و همچنین باید دقت داشت که امنیت سیستم نیز در این حالت زیر سؤال است (زیرا در این حالت تمامی کاربران، صرفنظر از سطوح دسترسی تعریف شده برای آن‌ها، دسترسی به اطلاعاتی خواهند داشت که نباید داشته باشند).
نکته‌ی دیگری را هم که باید در مورد ASP.NET به خاطر داشت این است که ویژگی ThreadStatic نیز در اینجا کمکی نمی‌کند؛ زیرا مطابق طراحی آن از تردها استفاده‌ی مجدد می‌گردد.به عبارت دیگر در ASP.NET الزامی ندارد که آغاز یک درخواست جدید حتما به همراه ایجاد یک ترد جدید باشد.
طول عمر این نوع متغیرها هم تا زمانی است که وب سرور یا برنامه ری استارت شوند. فقط در این حالت است که نمونه‌ی موجود تخریب شده و سپس با اجرای مجدد برنامه، بازسازی خواهند شد.
بنابراین متغیرهای استاتیک در ASP.NET همانند شیء Application عمل می‌کنند و از آن سریع‌تر هستند زیرا زمانیکه به آن‌ها ارجاع می‌شود نیازی به جستجو در یک جدول و یافتن آن‌ها نیست (برخلاف شیء Application) و همچنین در اینجا نیازی هم به عملیات تبدیل نوع داده‌ای وجود ندارد (برخلاف نوع شیء Application که به صورت Object تعریف شده است). وجود اشیاء Application در ASP.NET فقط به جهت حفظ سازگاری آن با ASP کلاسیک است و توصیه شده است در ASP.NET به دلایلی که ذکر شد، اگر و تنها اگر نیاز به اشیایی در سطح برنامه داشتید از متغیرهای استاتیک استفاده کنید. شیء Cache نیز در ASP.NET همین کاربرد را دارد با این تفاوت که می‌توان برای آن مدت زمان منقضی شدن تعریف کرد یا اینکه وب سرور بسته به حق تقدم و اهمیتی که برای آن تعریف شده است، مجاز به حذف کردن آن در زمانی است که با کمبود منابع مواجه می‌شود. همچنین باید دقت داشت که تنها مکان ذخیره سازی متغیرهای استاتیک حافظه است اما امکان دخیره سازی کش بر روی فایل سیستم تا بانک اطلاعاتی و غیره نیز مهیا است.

سؤال: آیا تعریف SqlConnection به صورت استاتیک جزو مواردی است که "مگر واقعا می‌دانید دارید چکار می‌کنید؟" ؟
پاسخ: خیر. در اینجا هم واقعا این شخص نمی‌داند که دارد چکار می‌کند! یعنی در مورد سازوکار درونی ADO.NET اطلاعاتی ندارد. باز کردن یک کانکشن در ADO.NET به معنای مراجعه به استخر (pool) کانکشن‌ها و بازکردن یکی از آن‌ها و در مقابل، بستن یک کانکشن هم به معنای علامتگذاری یک کانکشن به صورت غیرفعال است و آماده سازی آن برای استفاده در درخواست بعدی. به معنای دیگر این عملیات سربار آنچنانی ندارد که بخواهید آن‌را استاتیک تعریف کنید.
همچنین مورد دیگری را هم که این برنامه نویس نمی‌داند این است که متغیرهای استاتیک thread safe نیستند. به عبارتی حین استفاده از آن‌ها در یک برنامه‌ی چندکاربره‌ی ASP.NET حتما باید مکانیزم‌های قفل‌گذاری بر روی این نوع متغیرها و اشیاء اعمال شود (که این هم خود یک سربار اضافی است در مقیاس چند 10 یا چند 100 کاربر همزمان). این مشکلات همزمانی به چه معنا است؟
فرض کنید کاربر یک، شیء استاتیک SqlConnection ایی را باز کرده است و با آن مشغول کوئری گرفتن است. کاربر 2 نیز همزمان شروع به استفاده از این کانکشن باز در حال استفاده می‌کند (SqlConnection استاتیک یعنی استفاده‌ی تمام کاربران فقط و فقط از یک کانکشن باز شده)، نتیجه این خواهد بود که برای مثال پیغام خطایی را دریافت می‌کند مانند: فیلد مورد نظر در جدول موجود نیست! چرا؟ چون روی شیء استاتیک SqlConnection تعریف شده قفل گذاری صورت نگرفته است و در حین استفاده از آن هر کاربری در سایت نیز همان را استفاده خواهد کرد یا از آن بدتر ممکن است یک کاربر زودتر از کاربر دیگری آن‌را ببندد!

کاربر سوم در وسط کار با پیغام غیرمعتبر بودن کانکشن مواجه می‌شود، یا اینکه به صورت پیش فرض یک datareader را بیشتر نمی‌توان بر روی یک کانکشن باز شده اعمال کرد. کاربر 4 مشغول خواندن اطلاعات است، کاربر 5 ، پیغام غیرمعتبر بودن کوئری را دریافت می‌کند.

نظرات خوانندگان

نویسنده: Farhad
تاریخ: ۱۳۸۹/۰۹/۱۸ ۲۱:۰۶:۱۷

مطلب بسیار جالبی بود.

نویسنده: Mehran Zand
تاریخ: ۱۳۸۹/۰۹/۱۸ ۲۲:۱۸:۵۷

در لایه بیزینس تعریف مند ها استاتیک مشکلی ایجاد میکند؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۱۸ ۲۳:۲۹:۰۰

لطفا جهت تکمیل بحث به این مطلب مراجعه کنید: (+)

نویسنده: A
تاریخ: ۱۳۸۹/۰۹/۱۹ ۰۰:۵۲:۵۴

خیلی ممنون.

من برای معماری دسترسی به داده‌ام از روشی استفاده کرده بودم که لازم داشتم متن داده‌ام برای هر ریسمان یکتا باشد.
این کار را با استفاده از متدهای

Thread.AllocateDataSlot

Thread.GetData

Thread.SetData

انجام داده بودم.

اکنون استفاده از یک متغییر Static دارای ویژگی ThreadStatic را تست کردم و به خوبی جواب گرفتم.

فقط یک نکته کوچک وجود دارد. من یک متغییر bool نیز دارم که چون ValueType است در اولین بار که مورد دسترسی قرار می‌گیرد مقدار Defaultاش که همان false است را خواهد داشت. درحالی که باید از دسترسی برای اولین بار به این متغییر آگاهی پیدا کنم. البته یک روش ساده که فعلا به ذهنم رسید استفاده از bool? است.

باز هم ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۱۹ ۰۱:۳۳:۰۲

ببینید محور اصلی بحث این تاپیک ASP.NET است و اگر از ویژگی ThreadStatic استفاده کرده‌اید، اشتباه است چون بحث استفاده مجدد از یک ترد موجود در ThreadPool در اینجا مطرح است (در بالا ذکر کردم).
در ASP.NET اگر می‌خواهید اطلاعاتی را صرفا برای استفاده در طول عمر یک درخواست ذخیره کرده و به اشتراک بگذارید از HttpContext.Current.Items استفاده کنید که برای این منظور طراحی شده و استاندارد است : (+)

نویسنده: A
تاریخ: ۱۳۸۹/۰۹/۱۹ ۰۲:۰۷:۵۲

بله کاملا متوجه این موضوع بودم. در اینجا محیطی که برای آن طراحی می‌کنم نیز ASP.NET است. اما من حتماً اسرار دارم که به

ازاء هر Thread یک شی DataContext در حافظه داشته باشم و نه به ازاء هر درخواست. حال اگر ASP.NET از Thread استفاده مجدد می‌کند چه بهتر، از Dispose و New اضافی خودداری می‌شود. و البته علاوه بر این در نهایت باعث خواهد شد که طراحی این لایه از لایه Presentation بیشتر جدا شود و حتی برای Win نیز کاربرد داشته باشد.

البته! در ASP.NET اگر امکان داشت که دو Client همزمان از یک Thread دو خروجی متفاوت بگیرند با مشکل مواجه می‌شدم؛ که فکر می‌کنم چنین چیزی ممکن نیست.

در کل تنها مشکلی که داشتم این بود که وقتی فکر این اگوی طراحی به ذهنم خطور کرد از ThreadStaticAttribute اطلاع نداشتم و با آن متدها که گفتم کار کردم. که توسط شما از این موضوع مطلع شدم.

باز هم ممنون برای نکات و ارجاء.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۱۹ ۰۸:۴۷:۱۱

در ASP.NET این استفاده‌ی مجدد از یک Thread منحصر به یک سشن نیست. همچنین از یک ترد مشخص الزاما برای درخواست بعدی استفاده نمی‌شود.
به علاوه DataContext نیز thread safe نیست و مباحثی را که در بالا ذکر شد در نظر داشته باشید. ممکن است وسط کار توسط یک کاربر دیگر استفاده شود.
+ این کار کردن با یک مرورگر و load یک کاربر ... روش صحیحی برای آزمودن نیست.
اگر خیلی علاقمند به انجام اینکار هستید باید از روش یک DataContext به ازای هر درخواست (per request) استفاده کنید، یعنی همان روش استفاده از Current.Items ذکر شده. اصطلاحا به این الگو، الگوی UnitOfWork گفته می‌شود. یک پیاده سازی خوب در این مورد اینجا هست: (+) و (+)

نویسنده: A
تاریخ: ۱۳۸۹/۰۹/۱۹ ۱۱:۲۹:۱۹

آیا این امکان وجود دارد که یک متغییر که دارای ویژگی ThreadStatic است توسط دو Thread همزمان مورد استفاده قرار بگیرد؟ جواب Microsoft خیر است. پس اینچنین متغییری نیاز به ThreadSafe بودن و حتی lock (فارق از محیط اجرا) ندارد. مگر اینکه راه دیگری برای اجرای چند کد به صورت موازی در یک AppDomain علاوه بر Thread وجود داشته باشد. به عبارت دیگر من Thread را Atomic ترین شی موازی‌سازی می‌دانم. اشتباه می‌کنم؟

البته اینکه یک Thread لزوماً برای درخواستی بعدی استفاده نشود نیز مشکلی ایجاد نمی‌کند.

لینک‌ها را مطالعه کردم. ولی بازهم فکر می‌کنم مشکلی در استفاده از این روش نیست! با توجه به جمله‌ای که در بالا گفتم نمی‌دانم چه مشکلی می‌تواند از نظر Concurrency اتفاق بیفتد؟

در مورد تست، من هم با شما موافقم و تست کردن فقط با یک کاربر در یک مرورگر را مطمئناً تست کاملی نمی‌دانم. درواقع فقط با بررسی نقاط بحرانی و حالت‌های بحرانی در ذهنم موارد Concurrent را چک کرده‌ام. که البته حتماً سعی می‌کنم با استفاده از یک نرم‌افزار خوب یک محیط چند کاربره را شبیه‌سازی کنم.

راستی یک نکته خیلی مهم. در این روش هر شیی که DataContext را برای اولین بار به وجود آورده باید بعد از اتمام کارش خودش آنرا Dispose کند. تمامی عملیات کار با داده در یک بلاک using صورت می‌گیرد اما شیی که در using استفاده شده مانند یک شی Repository است که مدیریت DataContext را برعهده دارد اما در داخل تفاوت‌هایی برای بالابردن کارایی دارد.

البته اسرار من بر این روش به جای استفاده از HttpContext این است که: لایه دسترسی به داده به صورت BlackBox کاملا از لایه Presentaion جدا باشد. درواقع بدون پیاده‌سازی یک Interface یا چیزی شبیه آن بتوان از یک لایه داده «همزمان» توسط

چند لایه دیگر (در یک پروژه) استفاده شود و البته برنامه‌نویس به دردسر نیفتد.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۱۹ ۱۱:۵۲:۴۱

- Atomic Operation داریم ولی شیء Atomic ؟
- بحث استفاده مجدد از یک ترد در ASP.NET به این معنا است:
ترد یک در سشن کاربر یک، یک DataContext استاتیک را ایجاد کرده. حتی آن‌را با ویژگی ThreadStatic هم مزین کرده است. اکنون به نظر در پایان درخواست کارش به پایان رسیده است. در این لحظه ASP.NET این ترد یک را در اختیار سشن کاربر 2 قرار می‌دهد. این DataContext استاتیک شما که با توجه به ویژگی ThreadStatic بودن آن در این ترد زنده است و جهت Tracking بسیاری از موجودیت‌ها از آن استفاده شده، اطلاعات خود را در اختیار کاربر 2 قرار داده است. این DataContext نه لزوما می‌تواند معتبر باشد (شاید dispose شده) و یا شاید حاوی اطلاعات حساس و غیرضروری. هر دو مورد در یک برنامه چند کاربره مشکل ساز است.
- زمانیکه از ORM استفاده می‌کنید، لایه دسترسی به داده همان ORM است و از دید لایه‌های دیگر مخفی است. شما یک لایه دیگر به نام BLL برای جداسازی اعمال انجام شده توسط آن از لایه نمایش بایدایجاد کنید.

پ.ن.
- اسرار با اصرار کمی متفاوت است.

نویسنده: A
تاریخ: ۱۳۸۹/۰۹/۱۹ ۱۳:۴۶:۵۷

منظورم از Atomic این است که بیشتر از این نمی‌توانیم یک موجودیت را به اجزاء ریزتر تقسیم کنیم. مثل یک نقش(Role)اتمیک.

صحبت شما درست است ولی در این مورد این مشکلاتی که گفتید ایجاد نمی‌شود. عرض کردم که قبل از اینکه Thread بخواهد به کاربر 2 سپرده شود DataContext نابود شده است. معماری من کمی متفاوت است. فکر میکنم اگر بخواهیم بحث کنیم باید بر سر معماری بحث کنیم.
پاسخ اینکه آیا این معماری درست کار میکند یا نه در این جواب است که آیا امکان دارد دو Thread همزمان به یک شی که دارای ویژگی ThreadStatic است دسترسی پیدا کنند؟ اگر جواب خیر است پس همه چیز مرتب است (التبه کمی توضیح دارد).

و همچنین بله صحبت شما در مورد BLL درست است اما گاهی برای پروژه‌های کوچک بهتر است لایه‌ها با یک شبه BLL که داخل DAL قرار دارد صحبت کنند. البته فکر می‌کنم نمی‌توانم به درستی بیان کنم.

اما در کل شما به یک نکته خیلی خوب اشاره کردید و آن هم امنیت است. چک نکردم اما فکر می‌کنم در صورتی که «برنامه‌نویس خراب کاری کند» ممکن است این سیستم امنیت خوبی نداشته باشد. سعی میکنم برای این موضوع فکری کنم. اگر این مورد جدی باشد و اشتباهات برنامه‌نویس بتواند موجب مشکل شود حتماً از این روش صرف نظر کرده و از همان HttpContext.Current استفاده خواهم کرد.

دیکته من هرگز خوب نبوده و نیست. منظورم از اسرار، رازها نبوده منظورم پافشاری بوده (اصرار) :)

ممنون.

نویسنده: Nima
تاریخ: ۱۳۸۹/۰۹/۱۹ ۱۸:۱۴:۴۲

بسیار عالی، استفاده کردم و دقیقا همچین اشتباهی رو داشتم. میرم که بیشتر تحقیق کنم
ممنون

نویسنده: میثم نوایی
تاریخ: ۱۳۸۹/۱۰/۱۴ ۱۷:۱۵:۴۱

من یکامپوننت چت آنلاین نوشتم وچون کاربران هر چهارثانیه یکبار لیست پیغام های دریافتی خود را چک میکند برای اینکه سرعت برنامه بالا باشه مجبور شدم کلیه پیغام هارو در یک لیست استاتیک قرار بدم.آیا کسی پیشنهاد بهتری نسبت به لیست استاتیک برای اینکار سراغ داره؟
بازم سپاسگزارم.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۰/۱۴ ۱۷:۳۱:۴۴

این مطلب جاری به معنای نفی استفاده از متغیرهای استاتیک نبود. اگر بد است چرا اصلا در زبان قرار داده شده؟ بنابراین با دید صرفا منفی به این قضیه نگاه نکنید.
در کار شما آیا این لیست برای تمام کاربران یکسان است؟ آیا سطح دسترسی در کار نیست؟ آیا همه موارد مشابهی را مشاهده می‌کنند؟ اگر بله مشکلی ندارد، فقط در نظر داشته باشید که متغیرهای استاتیک thread safe نیستند. برای این موارد کلاس Cache قرار گرفته در فضای نام System.Web.Caching ، مطابق مستندات آن Thread safe است و read/write آن در یک محیط چند کاربره مشکل زا نیست: (+)

ضمنا در مورد طراحی سیستم چت خوب در ASP.NET در مورد COMET تحقیق کنید. این روش سربار کمی دارد چون سرور به کلاینت پیغام ارسال می‌کند نه اینکه کلاینت متناوبا سرور را چک کند که آیا پیغام جدیدی هست یا نه: (+)

نویسنده: میثم نوایی
تاریخ: ۱۳۸۹/۱۰/۲۶ ۰۹:۵۸:۲۵

سپاسگزارم.راهنمایی های شما مخصوصا معرفی این کامپوننت آخری خیلی بدرد بخور بود.احتمالا بهترین پیشنهاد برای پیاده سازی سیستم چت استفاده از همین کامپوننت می باشد.