**ASP.NET Cache**

**关于缓存的介绍**

-> 由于动态页面技术通常需要从数据源获取数据，并经过一些计算逻辑，最终变成一些HTML代码发给客户端显示。而这些计算过程显然也是有成本的，这些处理成本最直观的表现为影响服务器的响应速度，尤其是当数据的处理过程变得复杂以及访问量变大时，会变得比较明显，再此为基础上我们可以思考，如果有些数据并非时刻在发生变化，那么我们可以将这些数据的最终计算结果（包括页面输出）缓存起来，从而达到非常明显地提升程序的性能，缓存的最常见且最重要的用途就体现在这个方面，这也是为什么一说到性能优化时，一般都将缓存摆在第一位的原因

**Cache的介绍**

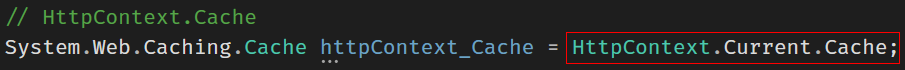
-> 在ASP.NET中，微软为我们提供了实现缓存这一特性的类【System.Web.Caching.Cache】，需要我们注意的是，该类是静态的，也就是说Web Application宿主环境在创建的过程至结束的过程当中，至始至终该类都只有一个实例

-> 【System.Web.Caching.Cache】是一个线程安全的类，这意味着我们在多线程模型的编程读取或写入Cache的时候是不需要考虑线程安全的问题的，那么Cache是如何实现线程安全的问题的呢？答案是同步锁Lock的机制。我们都知道，在开发的过程当中如果使用了同步锁的机制的话，虽然能够保证在锁住的Context中，至始至终都只有一个线程在访问其内部的资源，然而这一定程度上也带来了并发的问题，有可能再某一是个，多个线程在读写缓存的过程当中还需要等待上一个线程释放锁的资源才能进行缓存资源的获取，这显然是不合理的，微软为了尽可能减少此类的冲突，它还为【System.Web.Caching.Cache】这个类提供了一个较为强大的机制：容器化，虽然作为静态类型的【System.Web.Caching.Cache】至始至终只有一个实例，然而在这个实例被创建的过程当中，Cache的内部还会根据当前服务器CPU的个数创建多个不同的缓存容器(我们可以理解成一个缓存容器中存放的就是一个缓存的实例)，不同的缓存容器之间通过一个【HashCode】形成映射的关系，那么当一个线程(请求)在进行缓存的读写工作的时候，当前线程会给分配到一个【HashCode】，线程再根据这个【HashCode】找到所对应的缓存容器后，再对缓存进行读写操作，这样的设计可以一定成都少减少了冲突的问题，但是我们需要知道的是，不管如何设计，实际的冲突还是无法避免的

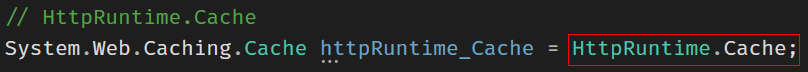
-> 使用【System.Web.Caching.Cache】不仅仅只是因为它只是一个线程安全的类型，它为程序员提供了一些非常强大的特性，诸如：**缓存依赖、缓存优先级策略，缓存过期时间、缓存移除回调**，关于这些特性会在下面的节点中进行详细的介绍

-> 在ASP.NET中我们可以通过以下三种方式去访问【System.Web.Caching.Cache】的实例

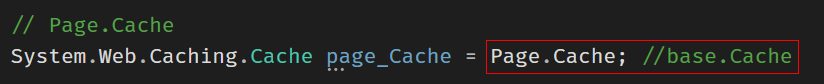
1. HttpContext.Cache



1. HttpRuntime.Cache

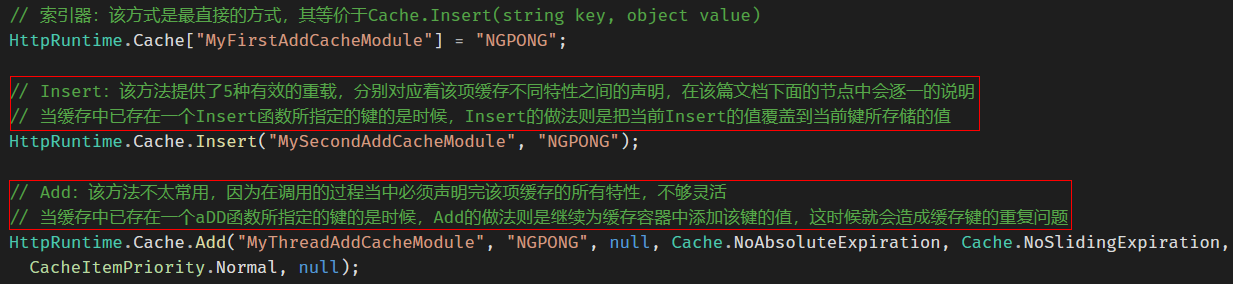


1. Page.Cache



**Cache的使用方式**

1. 添加和移除缓存的方式
2. 添加指定项的缓存：在这里我们只需要记忆【Insert】函数即可，其提供了5种不同签名的重载以实现该项缓存间的不 同特性，较为灵活



1. 手动移除指定项的缓存：如果缓存容器中并不存在所移除指定键的缓存的时候也不会抛异常



1. 缓存的过期时间策略
2. ASP.NET中的Cache类支持两种不同的过期策略，绝对过期时间和滑动过期时间

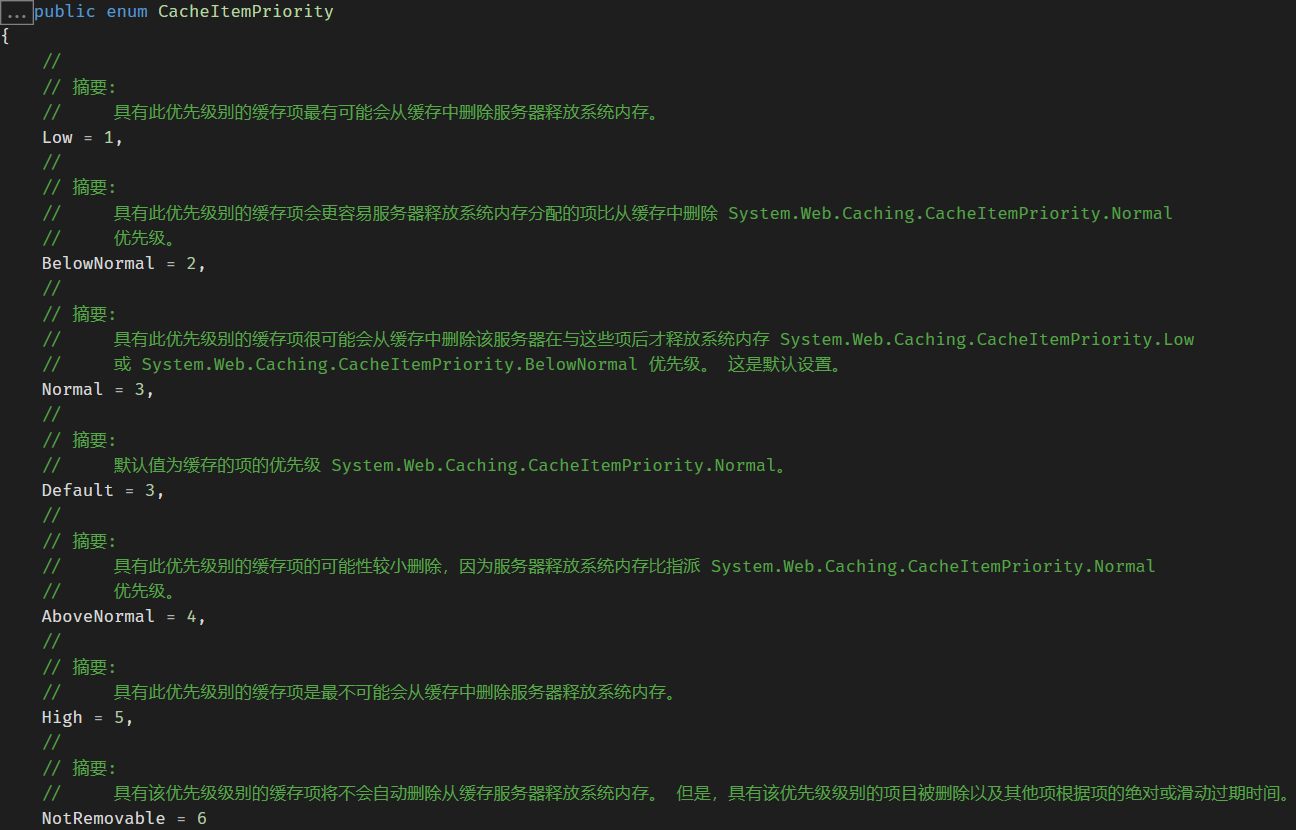
-> 绝对过期时间 absoluteExpiration：当到达一个具体的时间后该项缓存的内容立即移除

-> 滑动过期时间 slidingExpiration：在具体的时间内如果没有出现访问该项缓存的内容行为，则缓存项立即移除

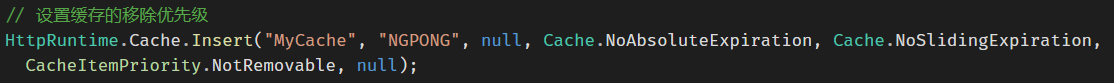
1. 需要注意的是我们在调用【Insert】函数添加一个缓存项的时候，绝对过期时间和滑动过期时间的设置是同时出现在方 法的签名内的，然而我们却不能同时指定这两种过期时间为一个有效的值，否则该项缓存则视为没有指定过期时间策略， 由垃圾回收器回收，当我们只想使用一种时间策略而忽略另一种时间策略的时候，我们就需要为所忽略的时间策略的 参数值赋值为Cache类的分别定义忽略两种不同时间策略的静态属性NoAbsoluteExpiration和NoSlidingExpiration



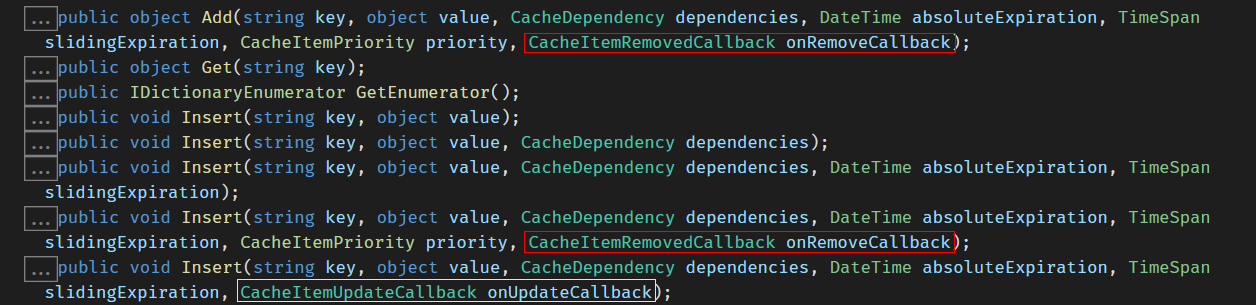
1. 缓存的移除优先级
2. 当缓存占用内存过多，且内存在不够用的情况下，Cache类会根据我们所定义不同项的缓存的移除优先级去把一些优先 级比较低的缓存项移除掉，以来缓解服务器内存上的压力
3. 关于缓存的优先级.NET平台通过一个枚举来定义



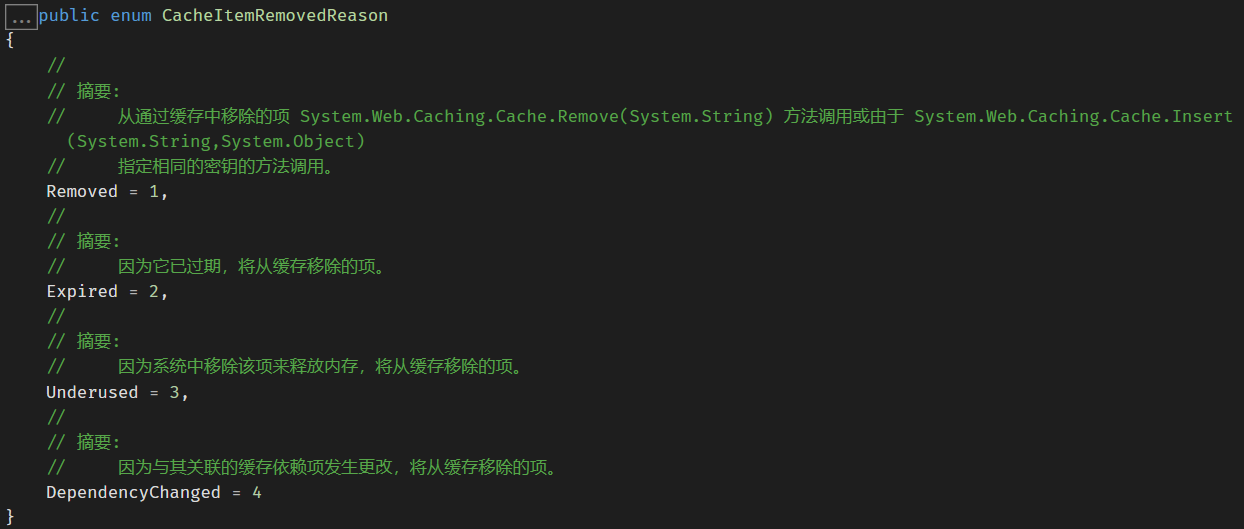
1. 为缓存项添加一个移除优先级，需要注意的是，当我们没有为该项缓存指定移除优先级，则默认为【Normal】

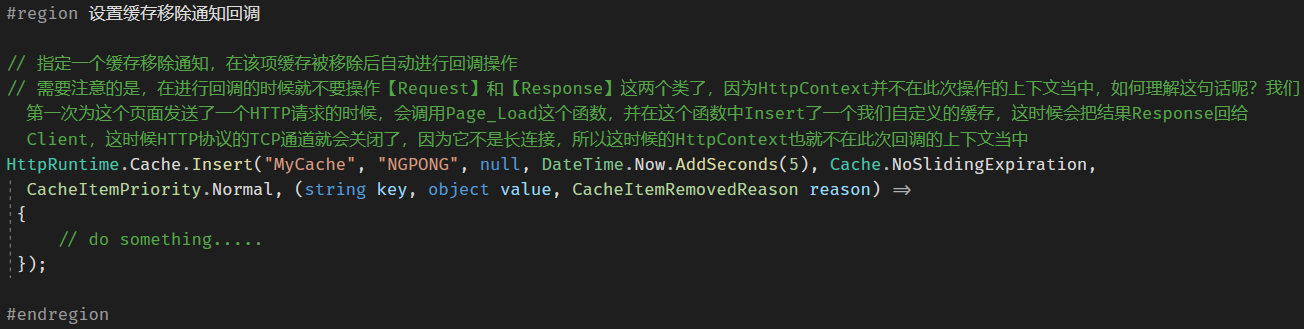


1. 缓存移除通知回调
2. 当一个缓存由于一些原因(过期、手动移除、系统缓存回收)被移除的时候，我们可以指定一个回调函数的委托来进行缓 存移除通知的回调，需要注意的是，关于回调的委托在不同的【Insert】函数重载的版本里面共对应着两个，它们两个的 功能是一模一样的，页是在该项缓存被移除的时候调用该委托所指向的函数进行移除回调通知的操作，不同的是，这两个 委托的签名是不同的



1. 为指定项的缓存指定一个移除回调通知，在这里，需要注意该委托的签名【CacheItemRemovedReason】，它是一个 枚举，它代表着不同情况下缓存移除指示，我们可以在回调的实现中通过这个枚举不同的值去展开不同的实现



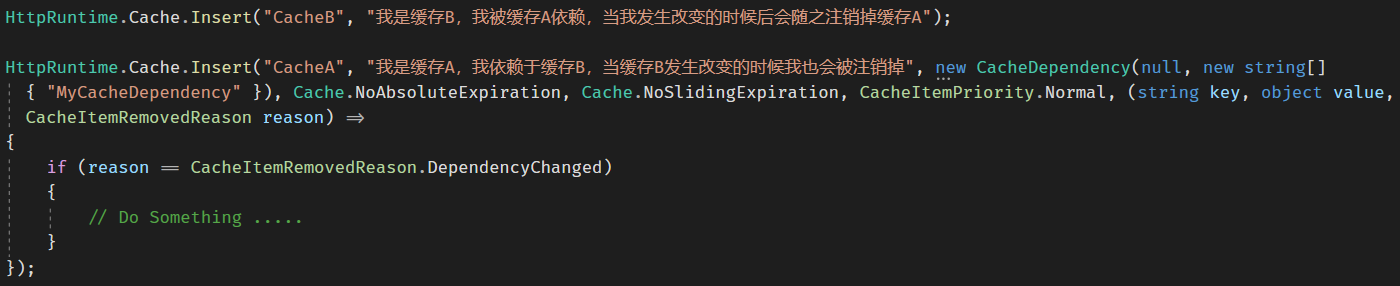


1. 缓存的依赖
2. 依赖于其他缓存

-> 当我们为一个缓存A设置了依赖于缓存B的时候，这时候缓存A与缓存B之间的关系则为：缓存A依赖于缓存B，缓 存B被缓存A所依赖，那么当被依赖的一方(缓存B)发生更新或移除的操作的时候，缓存A也会被注销掉

-> 需要注意的是，我们如果想要实现正常的缓存A依赖于缓存B，那么在声明缓存A的缓存项与其所对应的依赖关系的 时候，服务器的内存中必须要已存在被依赖一方(缓存B)的缓存项内容，否则的话在声明缓存A的过程当中，缓存A会直 接判定为所依赖的一方已发生改变，缓存A直接注销掉

-> 如何声明两个缓存之间的依赖关系

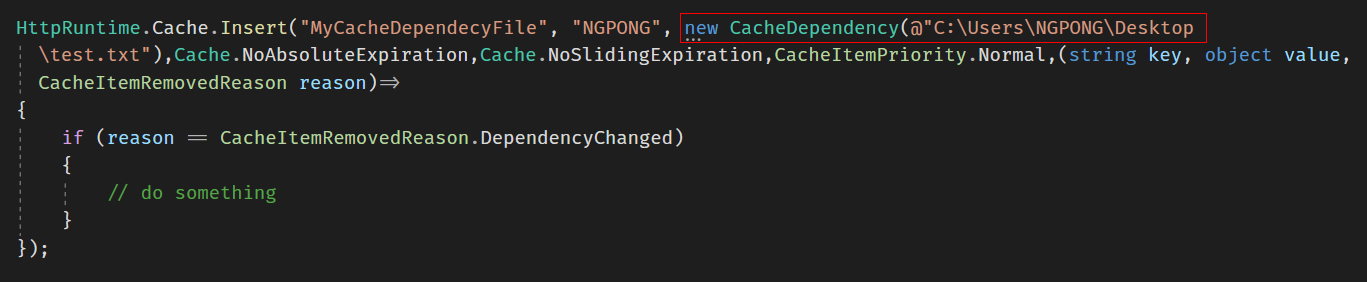


1. 依赖于文件

-> 当我们为一个缓存项设置了文件依赖后，当被依赖的文件发生了改变(修改/删除)，那么相对应缓存项也会被移除

-> 需要注意的是，我们为缓存项所设置的依赖文件是真实存在于服务器中的，否则该缓存依赖关系则失效

-> 如何实现缓存项依赖于文件



1. 依赖于数据库中的指定表

-> 我们可以指定一个缓存项依赖于数据库中的一张表，当被依赖的表的内容发生改变的时候，相应的缓存项也会被移除

-> 指定缓存依赖于数据库的底层实现原理，ASP.NET是通过为指定表中加入一个触发器，每当修改/删除/新增表的数据的时候，都会通过触发器去往一张ASP.NET自己为我们所创建的表里面修改一条数据，该条数据就是对应着我们所依赖的数据表还有该表所修改的次数，然后ASP.NET会当前Web Application的Application Domain中开一个定时器轮询检查这张表的数据，当发生改变后则立刻移除制定项的缓存数据

-> 关于如何创建数据库缓存依赖项的内容在网上搜索文章即可，该文档不做过多讲解

1. 页面缓存
2. 我们可以指定一个aspx页的输出内容缓存至内存里面，当我们指定了aspx的输出缓存，那么在指定时间间隔之内再 次访问该aspx页，然后Http请求走到ASP.NET管道事件【ResolveRequestCache】的时候，会自动地把上一次缓存该 aspx的输出内容直接返回交回到Client手上，也就是不会进入到后续的ASP.NET管道事件当中
3. 关于如何实现页面缓存，只需要按照下图使用一段文本模板指令即可，其中需要注意的是两个属性的值需要指定
4. 【Duration】：当前aspx页缓存的绝对过期时间，单位是Second
5. 【VaryByParam】，这个属性需要重点的介绍，这个属性的意思是指定当前aspx页的缓存是否按照指定参数进行保存，它的值可以为none：不按照参数进行保存，当前页面的缓存只存储一份、指定参数(多个参数用冒号隔开)：当前页面缓存会根据当前页面所请求带有的指定参数去保存多分、\*：当前页面缓存内容会根据所有参数值的不同去保存多份，那么这个属性具体是什么意思呢，举个例子，比如说我们现在为一个 test.aspx 指定了需要页面缓存，当我们请求这个页面的是时候会把这个页面的输出内容缓存下来，当我们下一次请求的时候会把缓存中所保存的数据直接返回给Client，那么当我们指定这个属性(VaryByParam)为none的时候，那么关于这个页面的不同参数格式的请求都只会保存一份，比如说我们有可能请求该页面的 test.aspx?Id=1 或者请求该页面的 test.aspx?Id=2，这两个页面所展示的结果都是不同的，但是【VaryByParam】属性值为none的时候，永远都会只保存一份的内容，可能是Id=1也可能是Id=2，那么当缓存只保存了Id=1的输出内容的时候，用户请求 Id=2 的时候就看不到2的内容只看到1的内容了，因为ASP.NET关于这个页面的缓存只是保存了一份，也就是Id=2，那么当我们指定【VaryByParam】属性的值为Id的时候，就会根据Id的不同去保存两份的缓存内容，这时候就不会出现错误信息的呈现了

截图20190630204213

**关于Cache类的参考blog**

-> [https://www.cnblogs.com/fish-li/archive/2011/12/27/2304063.html#\_label5https://www.cnblogs.com/fish-li/archive/2011/12/27/2304063.html#\_label5](https://www.cnblogs.com/fish-li/archive/2011/12/27/2304063.html" \l "_label5)

-> <https://www.cnblogs.com/kissdodog/archive/2013/05/07/3064895.html>