**ASP.NET与ASP.NET MVC中Cache的总结**

作者：[MyRhino](http://www.it165.net/pro/html/201408/20215.html)  发布日期：2014-08-19 20:36:49

[**我来说两句(0)**](http://www.it165.net/pro/html/201408/20215.html#comment_iframe)[http://www.it165.net/statics/css/4.0/images/tousu.jpg](http://www.it165.net/index.php?m=formguide&c=index&a=show&formid=27)

**0**

**Tag标签：**[ASP](http://s.it165.net/?t=13&q=ASP" \t "_blank)  [NET与ASP](http://s.it165.net/?t=13&q=NET%D3%EBASP)  [NET](http://s.it165.net/?t=13&q=NET)    [MVC中Cache的总结](http://s.it165.net/?t=13&q=MVC%D6%D0Cache%B5%C4%D7%DC%BD%E1)

* 前两周出去面试面试，为了充分准备一下，就没有更新博客，更新博客之前就工作的事情扯两句吧，博客园上某招聘的链接上面工作工资确实写的很高，但是实际面试的时候跟你想的完全不一样，举一个很简单的例子就是招人的范畴是1-3年，待遇5k-10k，一年工作经验的给5k，两年的给7K,三年是9K，如果你认为人家写的也没错啊，给到的那个钱是指达到指定工作年限而且比较优秀的价码，招聘还是很有水分的，很多情况下回导致自己白跑一趟，如果还有面试的哥们可以提前问下公司的工资水平。每次写文章都啰嗦，还是开始正题吧：

**Cache概述**

Cache有多种翻译，可以是高速缓冲存储器，也可以是法国的服装品牌，本文只是简单的谈谈就是[ASP](http://www.it165.net/pro/webasp/" \t "_blank).NET 中Cache，做过Web应用程序的都知道，如果网站访问量比较大，系统应用程序可以将那些频繁访问的数据，以及那些需要大量处理时间来创建的数据存储在内存中，从而提高性能。 举一个很简单的例子，如果在博客园发布文章到首页，发布之后是不能即时看到自己的文章的,大概一分钟之后才能刷出数据，这就是常用的页面缓存.缓存分为两种，一般都是页面缓存以及应用程序缓存。

**页面缓存**

页面输出缓存是一类针服务器应用程序的典型缓存。 利用输出缓存，可以存储呈现的 HTML。 提供存储的 HTML 以响应相同页的后续请求。 可使用输出缓存来缓存整个网页或仅缓存 [ASP](http://www.it165.net/pro/webasp/).NET 控件的输出。页输出缓存在内存中存储处理后的 ASP.NET 页的内容。 这可让 ASP.NET 向客户端发送页响应，而不必再次经过页处理生命周期。 页输出缓存对于那些不经常更改，但需要大量处理才能创建的页特别有用。 例如，如果创建大通信量的网页来显示不需要频繁更新的数据，页输出缓存则可以极大地提高该页的性能。 可以分别为每个页配置页缓存，也可以在 Web.config 文件中创建缓存配置文件。利用缓存配置文件，只定义一次缓存设置就可以在多个页中使用这些设置。

首先看一个常用的，用来控制页面或者用户控制输出的持续输出30s时间，WebForm只需要在aspx页面写个OutPutCache：

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.<%@ OutputCache Duration='30' VaryByParam='None' %>

　asp.net  mvc页面生命周期跟WebForm那一套不一样，这个时候就需要在Action动手改改:

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.[OutputCache(Duration = 30,VaryByParam='none')]

2.public ActionResult CacheTest()

3.{

4.return View();

5.}

页面缓存的实现和处理方式有很多种，先从OutPutCache中的各种特性的，每种特性，对应着不同的实现方式：

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

01.<%@ OutputCache Duration='100'(必填)

02.Location='Any | Client | Downstream | Server | None | ServerAndClient '

03.Shared='True | False'

04.VaryByControl='控件名称'

05.VaryByCustom='Test'

06.VaryByHeader='headers'

07.VaryByParam='parametername'

08.VaryByContentEncoding='encodings'

09.CacheProfile='cache profile name | '''

10.NoStore='true | false'

11.SqlDependency='database/table name pair | CommandNotification'

12.ProviderName='Provider Name'

13.%>

OutPutCache上面的设置很简单，Duration是必填的，页或用户控件进行缓存的时间（秒），Location是指定缓存的位置，由枚举OutputCacheLocation来确定，包含在用户控件（.ascx 文件）中的 @ OutputCache 指令不支持此特性。

VaryByControl一个分号分隔的字符串列表，用于更改用户控件的输出缓存。 这些字符串代表用户控件中声明的 ASP.NET 服务器控件的 ID 属性值，在 ASP.NET 页和用户控件上使用 @ OutputCache 指令时，需要此特性或 VaryByParam 特性，有的时候我们缓存整个页面肯定不是太现实的，很多情况下只是缓存一部分页面，新建一个用户控件：

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.1 <%@ Control Language='C#' AutoEventWireup='true' CodeBehind='Test.ascx.cs' Inherits='MyWebForm.Test' %>

2.2 <%@ OutputCache Duration='5' VaryByParam='None' %>

3.3 <% string t = DateTime.Now.ToString();

4.4     Response.Write(t); %>

 在aspx页面中使用用户控件:

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.1 <%@ Register Src='~/Test.ascx' TagPrefix='uc1' TagName='Test' %>

2.2 <%@ OutputCache Duration='10' VaryByParam='None' %>

3.3 <uc1:Test runat='server' ID='Test1' />

这里面涉及到一个问题，就是页面设置的时间和控件设置的时间，页面是10s，控件是5s，控件与页面都是10s刷一次，以页面时间为主；如果页面是10s，控件是20s，那么页面10s刷一次，控件20s刷一次，简单点理解就是控件以时间长的为主；

asp.net mvc  中ascx所有的功能就是View页面，跟局部视图调用一样，新建一个Test.ascx,通过控制器去访问:

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.1 <%@ Control Language='C#' Inherits='System.Web.Mvc.ViewUserControl' %>

2.2 <% string t = DateTime.Now.ToString(); Response.Write(t); %>

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.1        [OutputCache(Duration=5,VaryByParam='none')]

2.2         public ActionResult Test()

3.3         {

4.4             Response.Cache.SetOmitVaryStar(true);

5.5             return View();

6.6         }

注意:控制器多加了Response.Cache.SetOmitVaryStar(true),这个Bug还比较隐蔽，详情可参考:

http://www.cnblogs.com/dudu/archive/2012/08/27/asp\_net\_mvc\_outputcache.html

VaryByCustom表示自定义输出缓存要求的任意文本。 如果特性的赋值为 browser，缓存将随[浏览器](http://www.it165.net/edu/ewl/" \t "_blank)名称和主要版本信息的不同而异。 如果输入自定义字符串，则必须在应用程序的 Global.asax 文件中重写  GetVaryByCustomString方法, 根据用户的主机名或者[浏览器](http://www.it165.net/edu/ewl/" \t "_blank)起版本给出缓存，不存在的话就需要新建缓存，如果存在话就不需要新建缓存。

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

01.public override string GetVaryByCustomString(HttpContext context,

02.string arg)

03.{

04.if (arg == 'Test')

05.{

06.return '浏览器版本=' +

07.context.Request.Browser.MinorVersion.ToString();

08.//return '用户主机=' +

09.//    context.Request.UserHostName;

10.}

11.return base.GetVaryByCustomString(context, arg);

12.}

VaryByHeader

分号分隔的 HTTP 标头列表，用于使输出缓存发生变化。 将该特性设为多标头时，对于每个指定标头组合，输出缓存都包含一个不同版本的请求文档。设置 VaryByHeader 特性将启用在所有 HTTP 1.1 版缓存中缓存项，而不仅仅在 ASP.NET 缓存中进行缓存。 用户控件中的 @ OutputCache指令不支持此特性。

VaryByParam

分号分隔的字符串列表，用于使输出缓存发生变化。 默认情况下，这些字符串对应于使用 GET 方法特性发送的查询字符串值，或者使用 POST 方法发送的参数。 将该特性设置为多个参数时，对于每个指定参数组合，输出缓存都包含一个不同版本的请求文档。 可能的值包括 none、星号 (\*) 以及任何有效的查询字符串或 POST 参数名称。

 页面通过URL链接访问，要缓存页面，百分之九十的情况的下页面是会有参数传递的，如果你设置为none，那么你用get或者post请求的时候是不会缓存的，设为\*是最简单的什么都不用管，如果你只想查name的时候缓存一下，那么你直接设置一下就行.

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

01.[OutputCache(Duration = 30,VaryByParam='name')]

02.public ActionResult CacheTest(string name)

03.{

04.if (name=='1')

05.{

06.ViewData['Test'] = '测试';

07.}

08.return View();

09.}

CacheProfile

配置文件中设置缓存。 这是可选特性，默认值为空字符串 ('')。包含在用户控件（.ascx 文件）中的 @ OutputCache 指令不支持此特性。 在页中指定此属性时，属性值必须与 outputCacheSettings 节下面的 outputCacheProfiles 元素中的一个可用项的名称匹配。 如果此名称与配置文件项不匹配，将引发异常。

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.[OutputCache(CacheProfile = 'myProfile',VaryByParam='none')]

2.public ActionResult Test()

3.{

4.Response.Cache.SetOmitVaryStar(true);

5.return View();

6.}

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

01.<system.web>

02.<caching>

03.<outputCacheSettings>

04.<outputCacheProfiles>

05.<add  name='myProfile'  duration='10'/>

06.</outputCacheProfiles>

07.</outputCacheSettings>

08.</caching>

09.</system.web>

 Shared一个布尔值，确定用户控件输出是否可以由多个页共享。 默认值为 false，包含在 ASP.NET 页（.aspx 文件）中的 @ OutputCache 指令不支持此特性。

NoStore一个布尔值，它决定了是否阻止敏感信息的二级存储。在用户控件(ascx)中使用的时候需要注意加上Response.Cache.SetNoStore();

SqlDependency标识一组数据库/表名称对的字符串值，页或控件的输出缓存依赖于这些名称对。

 ProviderName  一个字符串值，标识要使用的自定义输出缓存提供程序。

VaryByContentEncodings 以分号分隔的字符串列表，用于更改输出缓存。

**应用程序缓存**

应用程序缓存提供了一种编程方式，可通过键/值对将任意数据存储在内存中。 使用应用程序缓存与使用应用程序状态类似。 但是，与应用程序状态不同的是，应用程序缓存中的数据是易失的， 即数据并不是在整个应用程序生命周期中都存储在内存中。 使用应用程序缓存的优点是由 ASP.NET 管理缓存，它会在项过期、无效、或内存不足时移除缓存中的项。 还可以配置应用程序缓存，以便在移除项时通知应用程序。使用应用程序缓存的模式是，确定在访问某一项时该项是否存在于缓存中，如果存在，则使用。 如果该项不存在，则可以重新创建该项，然后将其放回缓存中。 这一模式可确保缓存中始终有最新的数据。

通过键和值直接设置项向缓存赋值

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.List<Person> list = new List<Person>(){new Person(){Name='Small',Age=24},

2.new Person(){Name='Fly',Age=24},

3.new Person(){Name='Elephant',Age=24}};

4.Cache['Test'] = list;

Insert的方式

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.Cache.Insert('Fly', '缓存2');

添加依赖项，可以是其他已存在的缓存，也可以是已存在的文件：

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.Cache.Insert('Fly', '缓存2');

2.Cache.Insert('Elephant', '缓存3',new CacheDependency(null, new string[]{'Fly'}));//key为Elephant, value为缓存3，依赖于key为Fly的缓存

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.Cache.Insert('Elephant', '缓存3',new CacheDependency(Server.MapPath('路径')));

插入多个缓存依赖

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.CacheDependency dep1 = new CacheDependency(null, new string[] { 'Fly' });

2.CacheDependency dep2 = new CacheDependency(Server.MapPath('路径'));

3.AggregateCacheDependency cacheList = new AggregateCacheDependency();

4.cacheList.Add(dep1);

5.cacheList.Add(dep2);

6.Cache.Insert('Elephant', '缓存3',cacheList);

绝对过期和滑动过期这个两个名字比较有意，绝对过期枚举英文是NoSlidingExpiration(不滑动)，滑动过期NoAbsoluteExpiration(不绝对)，这里面还设置了CacheItemPriority，这枚举大家自己看下就行，还有一个自动删除，缓存到期后执行删除方法:

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

01.//绝对过期时间的时间为30s，过完30s之后就没有缓存

02.Cache.Insert('Tiger', '老虎',null, DateTime.Now.AddSeconds(30), Cache.NoSlidingExpiration,CacheItemPriority.High,Show)；

03.//滑动过期时间的时间为1分钟，只要一直刷新一直有，如果刷新间隔超过一分钟的缓存就没啦

04.Cache.Insert('Dragon', '龙', null, Cache.NoAbsoluteExpiration,new TimeSpan(0, 0, 60));

05.public void Show(string key, object value, CacheItemRemovedReason call)

06.{

07.if (key=='Pig')

08.{

09.Cache.Remove('Pig');

10.}

11.}

Add的形式(该方法没有重载)

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.Cache.Add('Cow', '牛', null, Cache.NoAbsoluteExpiration, Cache.NoSlidingExpiration, CacheItemPriority.Default, null);

**可扩展性缓存**

ASP.NET 1到3.5中的输出缓存有一个限制，就是缓存存储本身不是可扩展的，ASP.NET 4.0之后是支持缓存扩展的。 通过缓存的扩展，可以保存HTML页面或者内存中的数据， 这些存储选项包括本地或远程磁盘、云存储和分布式缓存引擎。ASP.NET 中的输出缓存提供程序扩展性，可以为网站设计更好的输出缓存策略。 例如，可以创建这样一个输出缓存提供程序，该程序在内存中缓存站点流量排名靠前的页面，而在磁盘上缓存流量较低的页面，也可以对所呈现页面的各种变化因素组合进行缓存，但应使用分布式缓存以减少前端 Web 服务器的内存消耗。实现可扩展，需要创建一个继承自 OutputCacheProvider 的类：

加载中...

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

01.public class MyCacheProvide : OutputCacheProvider

02.{

03.public override object Add(string key, object entry, DateTime utcExpiry)

04.{

05.throw new NotImplementedException();

06.}

07.

08.public override object Get(string key)

09.{

10.throw new NotImplementedException();

11.}

12.

13.public override void Remove(string key)

14.{

15.throw new NotImplementedException();

16.}

17.

18.public override void Set(string key, object entry, DateTime utcExpiry)

19.{

20.throw new NotImplementedException();

21.}

22.}

然后配置文件中进行相应配置：

[view source](http://www.it165.net/pro/html/201408/20215.html#viewSource)[print](http://www.it165.net/pro/html/201408/20215.html#printSource)[?](http://www.it165.net/pro/html/201408/20215.html#about)

1.<caching>

2.<outputCache defaultProvider='AspNetInternalProvider'>

3.<providers>

4.<add name='DiskCache' type='MyWeb.Controllers.MyCacheProvide, MyWeb' />

5.</providers>

6.</outputCache>

7.</caching>

默认情况下，所有 HTTP 响应、所呈现的页面和控件（其中 defaultProvider 特性设置为 AspNetInternalProvider）所示的内存输出缓存。通过为 defaultProvider 指定不同的提供程序名称，可以更改用于 Web 应用程序的默认输出缓存提供程序。默认值为“AspNetInternalProvider', 这是 ASP.NET 提供的内存缓存。