## **什么是缓存？**

缓存是一种将经常使用的数据/信息存储在内存中的技术,这样,下次需要相同的数据/信息时,可以直接从内存检索,而不是再从应用程序中生成。

## **ASP.NET 中的缓存**

ASP提供如下几种不同类型的缓存：

* ****输出缓存****：输出缓存可以存储最后显现的网页的副本，或者是发送到客户机的部分页面。下次客户机请求该页面时，这个页面的缓存副本就会被发送给客户机，而不是重新生成这个页面，这样一来就节省了时间。
* ****数据缓存****：数据缓存是指从数据源缓存数据。只要缓存没被替换，那么再请求该数据时就会从缓存中获取。当缓存被替换的时候，会从数据源中获取新数据，缓存也会被再次充满。
* ****对象缓存****：对象缓存是缓存页面的对象，比如数据绑定控件等。缓存的数据放在服务器的内存。
* ****类缓存****：网页或 Web 服务是第一次运行时在组装编译成页类。然后组装会在服务器缓存。当下次请求该页面或者服务,就会使用缓存的装配。当改变源代码时,CLR 重新编译程序集。
* ****配置缓存****：应用程序配置信息存储在一个配置文件。配置缓存存储在服务器内存配置信息。

## **输出缓存**

呈现一个页面可能涉及一些复杂的过程,如,数据库访问，呈现复杂的控件等。输出缓存允许通过在内存中缓存数据，而绕过往返服务器。甚至可以缓存整个页面。

OutputCache 指令负责输出缓存。它启用输出缓存,并对其行为提供一定程度的控制。

OutputCache 指令的语法：

<**%@** OutputCache Duration="15" VaryByParam="None" %>

把这个指令放在页面指令下。这告诉环境需要缓存页面,持续 15 秒。以下页面加载事件处理程序将帮助确认页面是否已被缓存完毕。

**protected** **void** **Page\_Load**(**object** sender, EventArgs e){

Thread.Sleep(10000);

Response.Write("This page was generated and cache at:" +

DateTime.Now.ToString());

}

当帮助控制输出缓存的行为 OutputCache 指令有以下特性:

| ****属性**** | ****值**** | ****描述**** |
| --- | --- | --- |
| DiskCacheable | true/false | 描述输出是否可以写入带有缓存的磁盘。 |
| NoStore | true/false | 描述 "no store" 缓存头部是否被发送。 |
| CacheProfile | 字符串名 | 存储在 web.config 中的缓存配置文件名字。 |
| VaryByParam | None \* 参数名 | GET 请求中使用分号分隔的字符串值或者是 POST 请求中的变量值。 |
| VaryByHeader | \* 头文件名 | 可能是由客户端提交的用分号分隔的指定头的字符串。 |
| VaryByCustom | 浏览器 自定义字符串 | 通知 ASP.NET 通过浏览器名字版本或者客户端字符串改变输出缓存。 |
| Location | 任何 客户端 下载流 服务器 None | 任何:页面可能缓存在任何位置 客户端:缓存内容包含在浏览器中 下载流:缓存内容保存在下载流和服务器中 服务器:缓存仅保存在服务器之中 None:不允许缓存。 |
| Duration | 数字 | 被缓存页面或者操作的秒数。 |

## **数据缓存**

数据缓存的主要方面是数据源控件缓存。我们已经讨论了数据源控件代表一个数据源中的数据,如数据库或 XML 文件。这些控件从抽象类 DataSourceControl 中派生，并有以下继承属性以实现缓存:

* 缓存期 — 为缓存数据的数据源计时。
* 缓存期满策略 — 定义了当数据在缓存中过期时，缓存的行为。
* 缓存值依赖 — 定义了一个控件值，这个控件可以在数据期满时自动将其移出缓存。
* 启用缓存 — 可以确认是否缓存了数据。

<**asp:SqlDataSource** ID = "SqlDataSource1" runat = "server"

ConnectionString = "<%$ ConnectionStrings: ASPDotNetStepByStepConnectionString %>"

ProviderName = "<%$ ConnectionStrings: ASPDotNetStepByStepConnectionString.ProviderName %>"

SelectCommand = "SELECT \* FROM [DotNetReferences]"

EnableCaching = "true" CacheDuration = "60"> </**asp:SqlDataSource**>

## **对象缓存**

仅仅需要给这些项目分配一个值名，它们就可以被添加到缓存中，就像下面展示的这样：

**Cache**["key"] = item;

为了在缓存中插入对象， ASP 提供了 Insert() 方法。这种方法有四种重载版本。我们来看一下：

| ****重载**** | ****描述**** |
| --- | --- |
| Cache.Insert((key, value); | 以键值对的方式插入缓存，优先权和生命周期为默认 。 |
| Cache.Insert(key, value, dependencies); | 以键值对的方式插入缓存，优先权和生命周期为默认，和链接到其他文件或内容的缓存依赖，这样缓存修改就不再还有限的了。 |
| Cache.Insert(key, value, dependencies, absoluteExpiration, slidingExpiration); | 指出上述配置的有效期。 |
| Cache.Insert(key, value, dependencies, absoluteExpiration, slidingExpiration, priority, onRemoveCallback); | 与配置一起也允许设置缓存内容的优先权并委派，指出一种方法来调用当一个对象移除时。 |