IHttpHandler的妙用（1）：给图片添加水印

先给大家看几张熟悉的图片：

上面这些图片大家不熟悉，不过如果大家留心一下就会发现每张图片上都有一些有关网站的信息，只不过第一张和第二张使用的是图片，第三张就是"life.netskycn.com"文字而已，这就是所谓的图片水印技术，它一般都是在原有图片上添加一些自己的标记（一般是网站网址），这样如果别的网站直接链接使用的话，则不可避免带上了自己网站的信息，相当于给自己网站做了广告。

最近本人也做了一个asp.net网站，做一个政府网站，网站管理人员发现以前很多新闻记者不打招呼就直接使用他们的新闻信息和图片，他们很恼火，于是要求我给他们的网站添加上水印，这样如果别人要用只能通过正式渠道来向他们索取而不是从网站上把图片存下来。

一般的做法是在上传图片时直接给图片添加上水印，由于我在项目中使用了FCKeditor，在上传时不易控制，同时对方还要求他们自己用时不能有水印，于是我就使用了在图片显示时动态添加水印的办法，另外，为了提高效率，还使用了缓存技术，这样不必每次都添加水印，节省时间和提高性能。

本文中使用到的类是IHttpHandler（准确地说是一个接口），msdn对它的定义是：“定义 ASP.NET 为使用自定义 HTTP 处理程序同步处理 HTTP Web 请求而实现的协定。”HTTP处理程序是实现了System.Web.IHttpHandler接口的.NET组件，任何实现了IHttpHandler接口的类都可以用于处理输入的HTTP请求。也就是每次我们请求asp.net网站上的资源，都会由这个请求处理，这样就好控制了。

我的做法是，实现IHttpHandler接口，由实现这个接口的ImageHandler类专门处理对图片资源的请求，第一次请求某个图片时，由于缓存中没有，就读取这个图片，添加上我们指定的水印（由web.config设置指定），然后把输出到客户端，同时也把它缓存一定时间，在缓存期内就再次请求这个图片就不用添加水印了，直接把缓存中的图片输出就行了。

首先我们要编写自己的类ImageHandler实现IHttpHandler接口，代码如下：

```
using System;
using System.Web;
using System.Drawing;
using System.Drawing.Imaging;

/// <summary>
/// 说明：ImageHandler是一个图片处理类，它能实时在在图片上添加水印，避免自己网站的图片别的网站盗用
/// </summary>
public class ImageHandler:IHttpHandler
{
   public ImageHandler()
   {
      // TODO: 在此处添加构造函数逻辑
      }
```

```
#region IHttpHandler 成员
        /// <summary>
    /// 指示IHttpHandler 实例是否可再次使用
        /// </summary>
    public bool IsReusable
    {
        get { return true; }
    }
    /// <summary>
    /// 处理请求的方法
        /// </summary>
    /// <param name="context">它提供对用于为 HTTP 请求提供服务的内部服务器对象（如 Request、Response、Session 和 Server）的引用。</param>
    public void ProcessRequest(HttpContext context)
    {
        //获取请求的物理图片路径
            string imagePath = context.Request.PhysicalPath;
        Bitmap image = null;
        if (context.Cache[imagePath] == null)
        //如果当前缓存中没有指定的图片就将该图片添加水印并缓存
            {
                image = new Bitmap(imagePath);
          image = AddWaterMark(image);
          context.Cache[imagePath] = image;
        }
        else//否则就直接从混存中取出添加了水印的图片，节省时间
            {
                image = context.Cache[imagePath] as Bitmap;
        }
        image.Save(context.Response.OutputStream, ImageFormat.Jpeg);
        //将添加水印的图片输入到当前流中
        }

        #endregion

    //给图片添加水印
        private Bitmap AddWaterMark(Bitmap image)
    {
        string text = System.Configuration.ConfigurationManager.AppSettings["WaterMark"].ToString();
        int fontSize = int.Parse(System.Configuration.ConfigurationManager.AppSettings["Font-Size"].ToString());
        Font font = new Font("宋体", fontSize);

        //Brush brush = Brushes.DarkGray;
        Brush brush = Brushes.Red;
        Graphics g = Graphics.FromImage(image);
        SizeF size = g.MeasureString(text, font);
        g.DrawString(text, font, brush, image.Width - size.Width, image.Height - size.Height);
```

可以把这个类编译成一个单独的dll文件，不过我现在是演示，直接放在了App_Code文件夹下了。

然后我们编写演示文件，这是一个简单的aspx页面（纯html，没有使用任何服务器控件），代码如下：

```
    <appSettings>
    <!--添加到图片上的水印文字-->
        <add key="WaterMark" value="http://blog.csdn.net/zhoufoxcn"/>
    <!--水印文字的字体大小-->
        <add key="Font-Size" value="48"/>
 </appSettings>

    <httpHandlers>
     <!--只处理UploadImages目录下的jpg文件，别的目录下的图片不处理-->
        <add path="UploadImages/*.jpg" verb="*" type="ImageHandler"/>
```

它表明只处理网站根目录下的UploadImages文件夹中的图片，在本实例中Images文件夹也有图片，可是这个文件夹下的图片是网站logo等常规图片，不需要处理，所以这个元素的path属性值是：path="UploadImages/*.jpg"，如果你想给所有jpg图片添加水印，可以写成path="*.jpg"，在接下来的页面中我们就会看到结果如我们所愿，在images目录下的图片真的没有添加水印，而在UploadImages确实添加了水印！

运行结果：

图片位置	图片内容	图片说明
Images		网站通过图标，不需要加水印。
UploadImage		本人在绵山的照片，需要加水印。

最后说明：（一）为了演示，我故意将水印文字字体设置很大，同时还是用了醒目的红色，在实际项目中需要根据图片大小计算文字大小，文字颜色不宜过于鲜明；（二）为了演示，缓存使用过于简单，还可以进行一些更负责的设置，更大程度节省内存。（三）添加水印图片原理也很简单，使用GDI+将水印图片画在要添加水印的图片上即可；（四）即使单独在地址栏输入图片的url地址，看到的图片也一样会有水印。