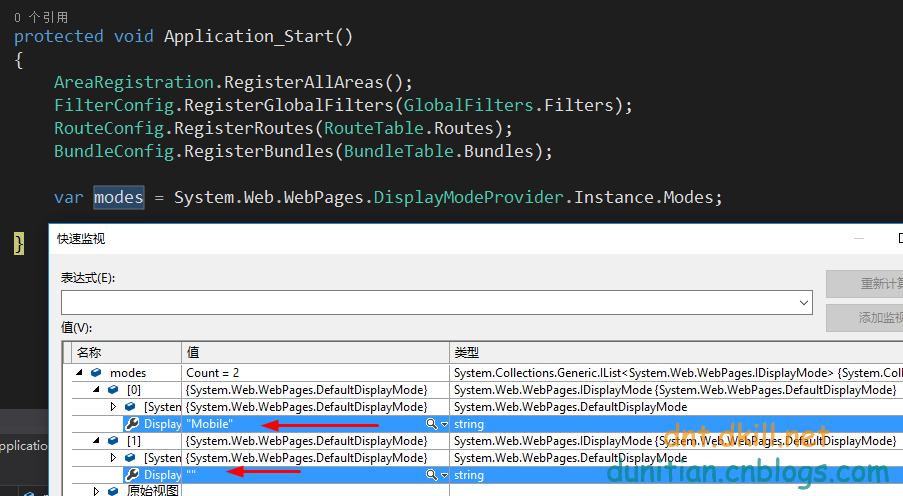
网站就必须用响应式布局吗？MVC视图展现模式之移动布局：[http://www.cnblogs.com/dunitian/p/5213787.html](http://www.cnblogs.com/dunitian/p/5213787.html" \t "_blank)

demo:<http://pan.baidu.com/s/1bnTUaKJ>

有人会疑问，为什么他能识别.mobile的后缀却不能识别例如：.mac .dnt 等等后缀呢？这些又是放在哪里的呢？

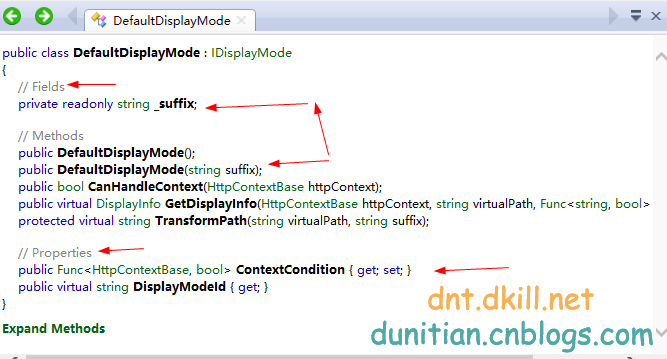
mobile 这个后缀其实是存放在：**DisplayModeProvider.Instance.Modes** 里面的，我们监视一下，发现里面就一个mobile，还有一个是默认的



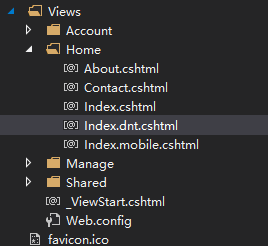
可以猜想，运行的时候是从上往下匹配的，“”的是通用匹配，那么我们加入一个自定义的后缀看看==>(**可以思考一下，为什么用 insert 不用 add**)

|  |  |
| --- | --- |
| 1  2  3  4  5 | //添加一个自定义后缀  DisplayModeProvider.Instance.Modes.Insert(0, new DefaultDisplayMode("dnt")  {      ContextCondition = (Context) => Context.Request.UserAgent.Contains("dnt")  }); |

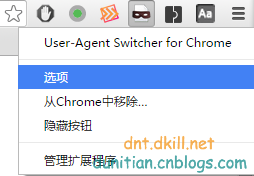
可能你不是很理解 DefaultDisplayMode，看看反编译吧----构造函数为suffix赋值（后缀）

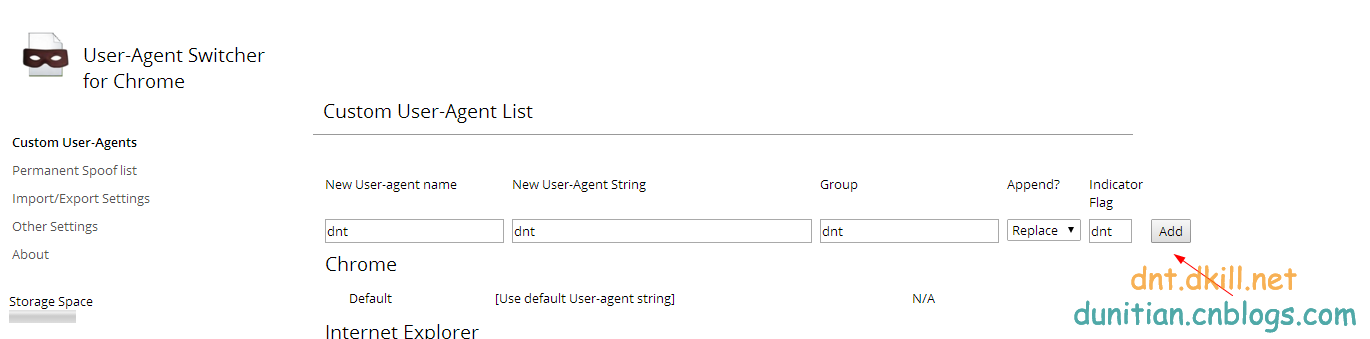


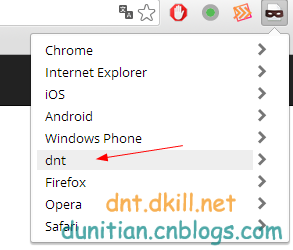
添加一个自定义的后缀视图

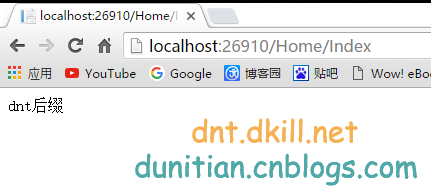


打开谷歌浏览器，设置一下User-Agent Switcher的浏览模式









附录：

DisplayModeProvider：

|  |
| --- |
| // Generated by .NET Reflector from F:\Work\Net\Mobile\packages\Microsoft.AspNet.WebPages.3.2.3\lib\net45\System.Web.WebPages.dll  namespace System.Web.WebPages  {  using System;  using System.Collections;  using System.Collections.Generic;  using System.Diagnostics;  using System.Runtime.CompilerServices;  using System.Web;    public sealed class DisplayModeProvider  {  private static readonly object \_displayModeKey = new object();  private readonly List<IDisplayMode> \_displayModes;  private static readonly DisplayModeProvider \_instance = new DisplayModeProvider();  public static readonly string DefaultDisplayModeId = string.Empty;  public static readonly string MobileDisplayModeId = "Mobile";    internal DisplayModeProvider()  {  List<IDisplayMode> list = new List<IDisplayMode>();  DefaultDisplayMode item = new DefaultDisplayMode(MobileDisplayModeId) {  ContextCondition = context => context.GetOverriddenBrowser().IsMobileDevice  };  list.Add(item);  list.Add(new DefaultDisplayMode());  this.\_displayModes = list;  }    private int FindFirstAvailableDisplayMode(IDisplayMode currentDisplayMode, bool requireConsistentDisplayMode)  {  if (!requireConsistentDisplayMode || (currentDisplayMode == null))  {  return 0;  }  int index = this.\_displayModes.IndexOf(currentDisplayMode);  if (index < 0)  {  return this.\_displayModes.Count;  }  return index;  }    public IEnumerable<IDisplayMode> GetAvailableDisplayModesForContext(HttpContextBase httpContext, IDisplayMode currentDisplayMode)  {  return this.GetAvailableDisplayModesForContext(httpContext, currentDisplayMode, this.RequireConsistentDisplayMode);  }    internal IEnumerable<IDisplayMode> GetAvailableDisplayModesForContext(HttpContextBase httpContext, IDisplayMode currentDisplayMode, bool requireConsistentDisplayMode)  {  int iteratorVariable0 = this.FindFirstAvailableDisplayMode(currentDisplayMode, requireConsistentDisplayMode);  for (int i = iteratorVariable0; i < this.\_displayModes.Count; i++)  {  IDisplayMode iteratorVariable2 = this.\_displayModes[i];  if (iteratorVariable2.CanHandleContext(httpContext))  {  yield return iteratorVariable2;  }  }  }    public DisplayInfo GetDisplayInfoForVirtualPath(string virtualPath, HttpContextBase httpContext, Func<string, bool> virtualPathExists, IDisplayMode currentDisplayMode)  {  return this.GetDisplayInfoForVirtualPath(virtualPath, httpContext, virtualPathExists, currentDisplayMode, this.RequireConsistentDisplayMode);  }    internal DisplayInfo GetDisplayInfoForVirtualPath(string virtualPath, HttpContextBase httpContext, Func<string, bool> virtualPathExists, IDisplayMode currentDisplayMode, bool requireConsistentDisplayMode)  {  for (int i = this.FindFirstAvailableDisplayMode(currentDisplayMode, requireConsistentDisplayMode); i < this.\_displayModes.Count; i++)  {  IDisplayMode mode = this.\_displayModes[i];  if (mode.CanHandleContext(httpContext))  {  DisplayInfo info = mode.GetDisplayInfo(httpContext, virtualPath, virtualPathExists);  if (info != null)  {  return info;  }  }  }  return null;  }    internal static IDisplayMode GetDisplayMode(HttpContextBase context)  {  if (context == null)  {  return null;  }  return (context.Items[\_displayModeKey] as IDisplayMode);  }    internal static void SetDisplayMode(HttpContextBase context, IDisplayMode displayMode)  {  if (context != null)  {  context.Items[\_displayModeKey] = displayMode;  }  }    public static DisplayModeProvider Instance  {  get  {  return \_instance;  }  }    public IList<IDisplayMode> Modes  {  get  {  return this.\_displayModes;  }  }    public bool RequireConsistentDisplayMode { get; set; }    [CompilerGenerated]  private sealed class <GetAvailableDisplayModesForContext>d\_\_4 : IEnumerable<IDisplayMode>, IEnumerable, IEnumerator<IDisplayMode>, IEnumerator, IDisposable  {  private int <>1\_\_state;  private IDisplayMode <>2\_\_current;  public IDisplayMode <>3\_\_currentDisplayMode;  public HttpContextBase <>3\_\_httpContext;  public bool <>3\_\_requireConsistentDisplayMode;  public DisplayModeProvider <>4\_\_this;  private int <>l\_\_initialThreadId;  public int <first>5\_\_5;  public int <i>5\_\_6;  public IDisplayMode <mode>5\_\_7;  public IDisplayMode currentDisplayMode;  public HttpContextBase httpContext;  public bool requireConsistentDisplayMode;    [DebuggerHidden]  public <GetAvailableDisplayModesForContext>d\_\_4(int <>1\_\_state)  {  this.<>1\_\_state = <>1\_\_state;  this.<>l\_\_initialThreadId = Environment.CurrentManagedThreadId;  }    private bool MoveNext()  {  switch (this.<>1\_\_state)  {  case 0:  this.<>1\_\_state = -1;  this.<first>5\_\_5 = this.<>4\_\_this.FindFirstAvailableDisplayMode(this.currentDisplayMode, this.requireConsistentDisplayMode);  this.<i>5\_\_6 = this.<first>5\_\_5;  goto Label\_00A5;    case 1:  this.<>1\_\_state = -1;  break;    default:  goto Label\_00BD;  }  Label\_0097:  this.<i>5\_\_6++;  Label\_00A5:  if (this.<i>5\_\_6 < this.<>4\_\_this.\_displayModes.Count)  {  this.<mode>5\_\_7 = this.<>4\_\_this.\_displayModes[this.<i>5\_\_6];  if (this.<mode>5\_\_7.CanHandleContext(this.httpContext))  {  this.<>2\_\_current = this.<mode>5\_\_7;  this.<>1\_\_state = 1;  return true;  }  goto Label\_0097;  }  Label\_00BD:  return false;  }    [DebuggerHidden]  IEnumerator<IDisplayMode> IEnumerable<IDisplayMode>.GetEnumerator()  {  DisplayModeProvider.<GetAvailableDisplayModesForContext>d\_\_4 d\_\_;  if ((Environment.CurrentManagedThreadId == this.<>l\_\_initialThreadId) && (this.<>1\_\_state == -2))  {  this.<>1\_\_state = 0;  d\_\_ = this;  }  else  {  d\_\_ = new DisplayModeProvider.<GetAvailableDisplayModesForContext>d\_\_4(0) {  <>4\_\_this = this.<>4\_\_this  };  }  d\_\_.httpContext = this.<>3\_\_httpContext;  d\_\_.currentDisplayMode = this.<>3\_\_currentDisplayMode;  d\_\_.requireConsistentDisplayMode = this.<>3\_\_requireConsistentDisplayMode;  return d\_\_;  }    [DebuggerHidden]  IEnumerator IEnumerable.GetEnumerator()  {  return this.System.Collections.Generic.IEnumerable<System.Web.WebPages.IDisplayMode>.GetEnumerator();  }    [DebuggerHidden]  void IEnumerator.Reset()  {  throw new NotSupportedException();  }    void IDisposable.Dispose()  {  }    IDisplayMode IEnumerator<IDisplayMode>.Current  {  [DebuggerHidden]  get  {  return this.<>2\_\_current;  }  }    object IEnumerator.Current  {  [DebuggerHidden]  get  {  return this.<>2\_\_current;  }  }  }  }  } |

DefaultDisplayMode：

|  |
| --- |
| // Generated by .NET Reflector from F:\Work\Net\Mobile\packages\Microsoft.AspNet.WebPages.3.2.3\lib\net45\System.Web.WebPages.dll  namespace System.Web.WebPages  {  using System;  using System.IO;  using System.Runtime.CompilerServices;  using System.Web;  public class DefaultDisplayMode : IDisplayMode  {  private readonly string \_suffix;  public DefaultDisplayMode()  : this(DisplayModeProvider.DefaultDisplayModeId)  {  }  public DefaultDisplayMode(string suffix)  {  this.\_suffix = suffix ?? string.Empty;  }  public bool CanHandleContext(HttpContextBase httpContext)  {  if (this.ContextCondition != null)  {  return this.ContextCondition(httpContext);  }  return true;  }  public virtual DisplayInfo GetDisplayInfo(HttpContextBase httpContext, string virtualPath, Func<string, bool> virtualPathExists)  {  string arg = this.TransformPath(virtualPath, this.\_suffix);  if ((arg != null) && virtualPathExists(arg))  {  return new DisplayInfo(arg, this);  }  return null;  }  protected virtual string TransformPath(string virtualPath, string suffix)  {  if (string.IsNullOrEmpty(suffix))  {  return virtualPath;  }  string extension = Path.GetExtension(virtualPath);  return Path.ChangeExtension(virtualPath, suffix + extension);  }  public Func<HttpContextBase, bool> ContextCondition { get; set; }  public virtual string DisplayModeId  {  get  {  return this.\_suffix;  }  }  }  } |

参考：<https://msdn.microsoft.com/en-us/magazine/hh975347.aspx>

<http://www.asp.net/mvc/overview/older-versions/aspnet-mvc-4-mobile-features>

<http://stackoverflow.com/questions/9354188/asp-net-mvc-4-mobile-display-modes-stop-working>