

عنوان: ارسال PingBack در ASP.NET

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۲/۰۸ ۷:۵

آدرس: www.dotnettips.info

گروه‌ها: ASP.Net, MVC, Regular expressions, Html Agility Pack, SEO

[Pingback](#) یکی از روش‌های اطلاع رسانی به سایت‌های دیگر در مورد لینک دادن به آن‌ها در سایت خود است. برای مثال من لینکی از یکی از مطالب شما را در متن جاری خودم قرار می‌دهم. سپس به وسیله‌ی ارسال یک ping، در مورد انجام اینکار به شما اطلاع رسانی می‌کنم. حاصل آن عموماً قسمت معروف ping-backs سایت‌ها است. این مورد نیز یکی از روش‌های مؤثر SEO در گرفتن [backlink](#) است و تبلیغ محتوا.

کار کردن با پروتکل Ping-back آنچنان ساده نیست؛ از این جهت که تبادل ارتباطات آن با پروتکل [XML-RPC](#) انجام می‌شود. XML-RPC نیز توسط PHP کارها بیشتر مورد استفاده قرار می‌گیرد؛ بجای استفاده از پروتکل‌های استاندارد وب سرویس‌ها مانند Soap و امثال آن. پیاده‌سازی‌های ابتدایی Pingback نیز مرتبط است به Wordpress معروف که با PHP تهیه شده‌است. در ادامه نگاهی خواهیم داشت به جزئیات پیاده‌سازی ارسال ping-back توسط برنامه‌های ASP.NET.

یافتن آدرس وب سرویس سایت پذیرای Pingback

اولین قدم در پیاده‌سازی Pingback، یافتن آدرسی است که باید اطلاعات مورد نظر را به آن ارسال کرد. این آدرس عموماً به دو طریق ارائه می‌شود:

الف) در هدری به نام x-pingback و یا pingback

ب) در قسمتی از کدهای HTML صفحه به شکل

```
<link rel="pingback" href="pingback server">
```

برای مثال اگر به وبلاگ‌های MSDN دقت کنید، هدر x-pingback را می‌توانید در خروجی وب سرور آن‌ها مشاهده کنید:

Latest Developments in General Purpose GPU Programming with F#



dsyme 23 Apr 2014 4:06 AM

0

RATI
★★★★★

Console HTML CSS Script DOM **Net** Cookies YSlow Pixel P... Refere... Changes F

Clear Persist **All** HTML CSS JavaScript XHR Images Plugins Media Fonts

URL	Status	Domain	Size	Remote IP	Timelin
GET latest-developments-in-gp	200 OK	blogs.msdn.com	19.4 KB	0.0.0.0:80	

Headers Response HTML Cache Cookies

Response Headers [view source](#)

- Cache-Control** no-cache, no-store
- Content-Encoding** gzip
- Content-Length** 19875
- Content-Type** text/html; charset=utf-8
- Date** Sun, 27 Apr 2014 18:41:08 GMT
- Expires** -1
- P3P** CP="ALL IND DSP COR ADM CONo CUR CUSo IVAo IVDo PSA PSD TAI TELo OUR SAMo CNT COM PUR UNI", CP="DSP CUR OTPi IND OTRi ONL FIN", CP="DSP CUR OTPi IND OTRi ONL FIN"
- Pragma** no-cache
- Server** Microsoft-IIS/7.5
- Set-Cookie** msdn=L=1033; domain=.microsoft.com; expires=Tue, 27-May-2014 18:41:08 GMT; path=
- Telligent-Evolution** 5.6.50428.7875
- Vary** Accept-Encoding
- X-AspNet-Version** 2.0.50727
- X-Frame-Options** SAMEORIGIN
- X-Pingback** http://blogs.msdn.com/b/dsyme/pingback.aspx
- X-Powered-By** ASP.NET

همانطور که ملاحظه می‌کنید، نیاز است Response header را آنالیز کنیم.

```
private Uri findPingbackServiceUri()
{
    var request = (HttpWebRequest)WebRequest.Create(_targetUri);
    request.UserAgent = UserAgent;
    request.Timeout = Timeout;
    request.ReadWriteTimeout = Timeout;
    request.Method = WebRequestMethods.Http.Get;
    request.AutomaticDecompression = DecompressionMethods.GZip | DecompressionMethods.Deflate;
    using (var response = request.GetResponse() as HttpWebResponse)
    {
        if (response == null) return null;

        var url = extractPingbackServiceUriFormHeaders(response);
        if (url != null)
            return url;

        if (!isResponseHtml(response))
            return null;

        using (var reader = new StreamReader(response.GetResponseStream()))
        {
            return extractPingbackServiceUriFormPage(reader.ReadToEnd());
        }
    }
}

private static Uri extractPingbackServiceUriFormHeaders(WebResponse response)
{
    var pingUrl = response.Headers.AllKeys.FirstOrDefault(header =>
        header.Equals("x-pingback", StringComparison.OrdinalIgnoreCase) ||
        header.Equals("pingback", StringComparison.OrdinalIgnoreCase));
}
```

```

        return getValidAbsoluteUri(pingUrl);
    }

    private static Uri extractPingbackServiceUriFromPage(string content)
    {
        if (string.IsNullOrEmpty(content)) return null;
        var regex = new Regex(@"(?s)<link\srel=""pingback""\shref=""(.+?)""",
        RegexOptions.IgnoreCase);
        var match = regex.Match(content);
        return (!match.Success || match.Groups.Count < 2) ? null :
        getValidAbsoluteUri(match.Groups[1].Value);
    }

    private static Uri getValidAbsoluteUri(string url)
    {
        Uri absoluteUri;
        return string.IsNullOrEmpty(url) || !Uri.TryCreate(url, UriKind.Absolute, out
        absoluteUri) ? null : absoluteUri;
    }

    private static bool isResponseHtml(WebResponse response)
    {
        var contentTypeKey = response.Headers.AllKeys.FirstOrDefault(header =>
        header.Equals("content-type",
        StringComparison.OrdinalIgnoreCase));
        return !string.IsNullOrEmpty(contentTypeKey) &&
        response.Headers[contentTypeKey].StartsWith("text/html",
        StringComparison.OrdinalIgnoreCase);
    }
}

```

نحوه‌ی استخراج آدرس سرویس Pingback یک سایت را در کدهای فوق ملاحظه می‌کنید.

targetUri، آدرسی است از یک سایت دیگر که در سایت ما درج شده‌است. زمانیکه این صفحه را درخواست می‌کنیم، response.Headers.AllKeys حاصل می‌تواند حاوی کلید x-pingback باشد یا خیر. اگر بلی، همینجا کار پایان می‌یابد. فقط باید مطمئن شد که این آدرس مطلق است و نه نسبی. به همین جهت در متد getValidAbsoluteUri، بررسی بر روی UriKind.Absolute انجام شده‌است.

اگر هدر فاقد کلید x-pingback باشد، قسمت ب را باید بررسی کرد. یعنی نیاز است محتوای HTML صفحه را برای یافتن link rel=pingback بررسی کنیم. همچنین باید دقت داشت که پیش از اینکار نیاز است حتما بررسی isResponseHtml صورت گیرد. برای مثال در سایت شما لینکی به یک فایل 2 گیگابایتی SQL Server درج شده‌است. در این حالت نباید ابتدا 2 گیگابایت فایل دریافت شود و سپس بررسی کنیم که آیا محتوای آن حاوی link rel=pingback است یا خیر. اگر محتوای ارسالی از نوع text/html بود، آنگاه کار دریافت محتوای لینک انجام خواهد شد.

ارسال Ping به آدرس سرویس Pingback

اکنون که آدرس سرویس pingback یک سایت را یافته‌ایم، کافی است ping ایی را به آن ارسال کنیم:

```

public void Send()
{
    var pingUrl = findPingbackServiceUri();
    if (pingUrl == null)
        throw new NotSupportedException(string.Format("{0} doesn't support pingback.",
        _targetUri.Host));

    sendPing(pingUrl);
}

private void sendPing(Uri pingUrl)
{
    var request = (HttpWebRequest)WebRequest.Create(pingUrl);
    request.UserAgent = UserAgent;
    request.Timeout = Timeout;
    request.ReadWriteTimeout = Timeout;
    request.Method = WebRequestMethods.Http.Post;
    request.ContentType = "text/xml";
    request.ProtocolVersion = HttpVersion.Version11;
    makeXmlRpcRequest(request);
    using (var response = (HttpWebResponse)request.GetResponse())
    {
        response.Close();
    }
}

```

```

    }
}

private void makeXmlRpcRequest(WebRequest request)
{
    var stream = request.GetRequestStream();
    using (var writer = new XmlTextWriter(stream, Encoding.ASCII))
    {
        writer.WriteStartDocument(true);
        writer.WriteStartElement("methodCall");
        writer.WriteElementString("methodName", "pingback.ping");
        writer.WriteStartElement("params");

        writer.WriteStartElement("param");
        writer.WriteStartElement("value");
        writer.WriteElementString("string", Uri.EscapeUriString(_sourceUri.ToString()));
        writer.WriteEndElement();
        writer.WriteEndElement();

        writer.WriteStartElement("param");
        writer.WriteStartElement("value");
        writer.WriteElementString("string", Uri.EscapeUriString(_targetUri.ToString()));
        writer.WriteEndElement();
        writer.WriteEndElement();

        writer.WriteEndElement();
        writer.WriteEndElement();
    }
}

```

اینبار `HttpWebRequest` تشکیل شده از نوع `post` است و نه `get`. همچنین مقداری را که باید ارسال کنیم نیاز است مطابق پروتکل XML-RPC باشد. برای کار با XML-RPC در دات نت یا می‌توان از کتابخانه‌ی [Cook Computing's XML-RPC.Net](http://cook-computing.com/XML-RPC.Net) استفاده کرد و یا مطابق کدهای فوق، دستورات آنرا توسط یک `XmlTextWriter` کنار هم قرار داد و نهایتاً در درخواست `Post` ارسالی درج کرد. در اینجا `sourceUri` آدرس صفحه‌ای در سایت ما است که `targetUri` ایی (آدرسی از سایت دیگر) در آن درج شده‌است. در یک `pinback`، صرفاً این دو آدرس به سرویس دریافت کننده‌ی `pingback` ارسال می‌شوند. سپس سایت دریافت کننده‌ی `ping`، ابتدا `sourceUri` را دریافت می‌کند تا عنوان آنرا استخراج کند و همچنین بررسی می‌کند که آیا `targetUri`، در آن درج شده‌است یا خیر (آیا spam است یا خیر؟) تا اینجا اگر این مراحل را کنار هم قرار دهیم به کلاس `Pingback` ذیل خواهیم رسید:

[Pingback.cs](#)

نحوه‌ی استفاده از کلاس `Pingback` تهیه شده

کار ارسال `Pingback` عموماً به این نحو است: هر زمانیکه مطلبی یا یکی از نظرات آن، ثبت یا ویرایش می‌شوند، نیاز است `Pingback`‌های آن ارسال شوند. بنابراین تنها کاری که باید انجام شود، استخراج لینک‌های خارجی یک صفحه و سپس فراخوانی متد `Send` کلاس فوق است.

یافتن لینک‌های یک محتوا را نیز می‌توان مانند متد `extractPingbackServiceUriFormPage` فوق، توسط یک `Regex` انجام داد و یا حتی با استفاده از کتابخانه‌ی معروف [HTML Agility Pack](#) :

```

var doc = new HtmlWeb().Load(url);
var linkTags = doc.DocumentNode.Descendants("link");
var linkedPages = doc.DocumentNode.Descendants("a")
    .Select(a => a.GetAttributeValue("href", null))
    .Where(u => !String.IsNullOrEmpty(u));

```