

]

XML in the real world

XML in the *real* world

真实世界中的XML

☺ *agentzh* ☺

(章亦春)

2006.10

♡ RSS is *cool*!

RSS 很酷哦！

RSS

→ Really Simple Syndication

Tired of **checking** your favorite
news and blogs sites everyday?

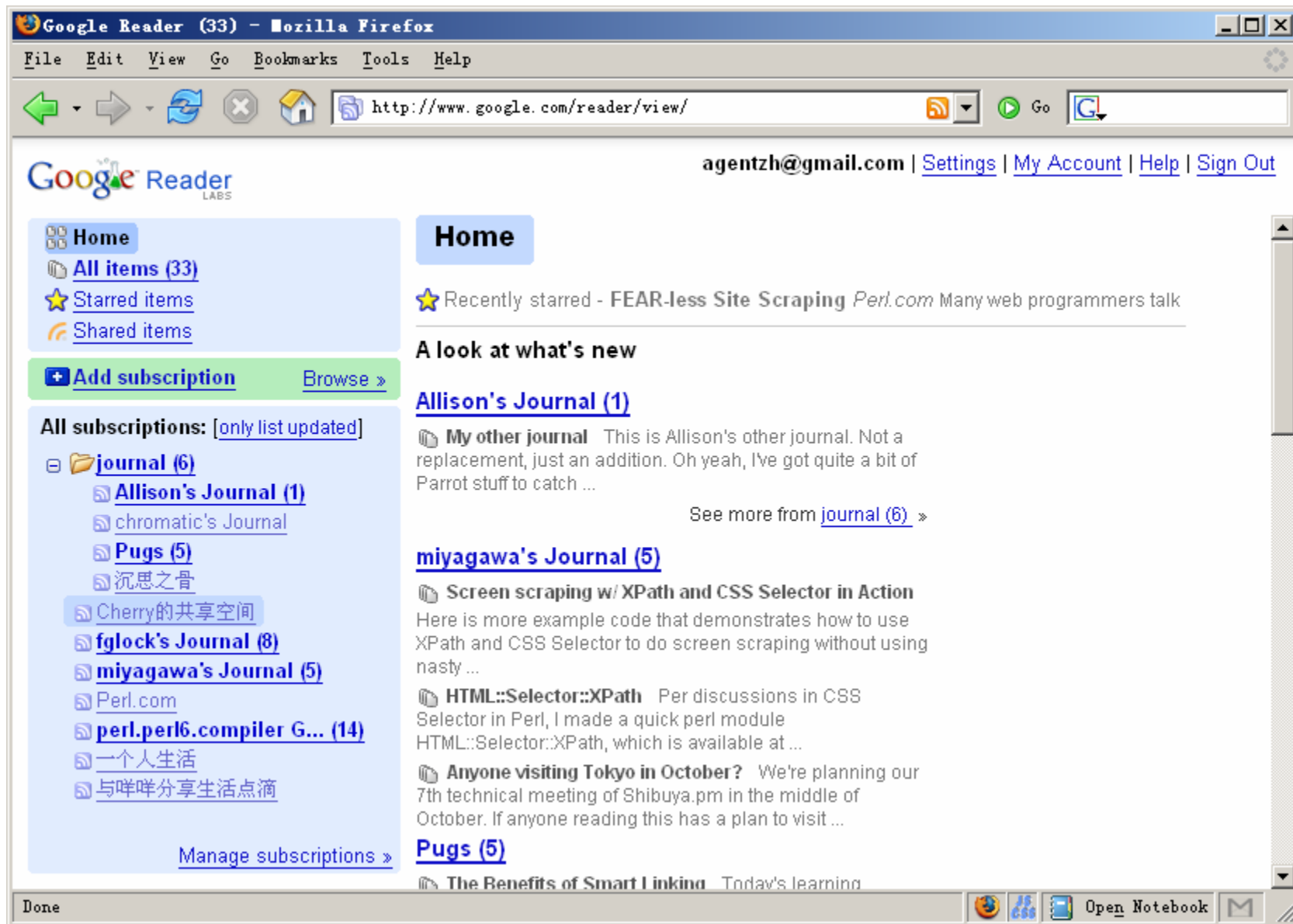
厌倦了每天去查看你最喜爱的
那些新闻和博客网站了吧？

Let me tell you
how **RSS** can *save* you.

让我来告诉你
如何让 RSS 拯救你。

Open Google Reader:
the *first* thing that I do everyday.

打开 Google Reader :
我每天做的第一件事情。



Let's read chromatic's *latest* journals
in **Google Reader**...

让我们在 Google Reader 中阅读 chromatic
最近的日记.....

Googl...Goog...learn...learn...Journ...use P...It's ...Journ...2006.10

Google Reader
LABS

agentzh@gmail.com | [Settings](#) | [My Account](#) | [Help](#) | [Sign Out](#)

Home

All items (28)

Starred items

Shared items

+ Add subscription

Browse »

All subscriptions: [only list updated]

journal (6)

Allison's Journal (1)

chromatic's Journal

Pugs (5)

沉思之骨

Cherry的共享空间 (3)

fglock's Journal

miyagawa's Journal (5)

Perl.com

perl.perl6.compiler G... (14)

一个人生活

Manage subscriptions »

chromatic's Journal

Feed settings... ▾

Expanded view

List view

All items [show only new] - [Mark all as read](#) - [Refresh](#)

★ It's Full of Sharps! »

Oct 4, 2006 (18 hours ago)

by chromatic

Ahh, so [radical simplicity is why dynamic languages on static VMs are important](#). Some days, I even miss Tcl/Tk.

★ Add star

Share

Email

Mark as read

Edit tags: journal

★ C is not Perl »

Oct 3, 2006 (2 days ago)

by chromatic

[Stevan](#) asked me to write some XS code for [Class::MOP](#), and it sounded like more fun than painting my deck or writing the non-fiction book, so I dodged segfaults for a couple of hours this afternoon.

The Perl XS API has its own kind of consistency. It's never really stuck in my brain, but it's just familiar enough that I sort of know what I want.

Previous item

Next item

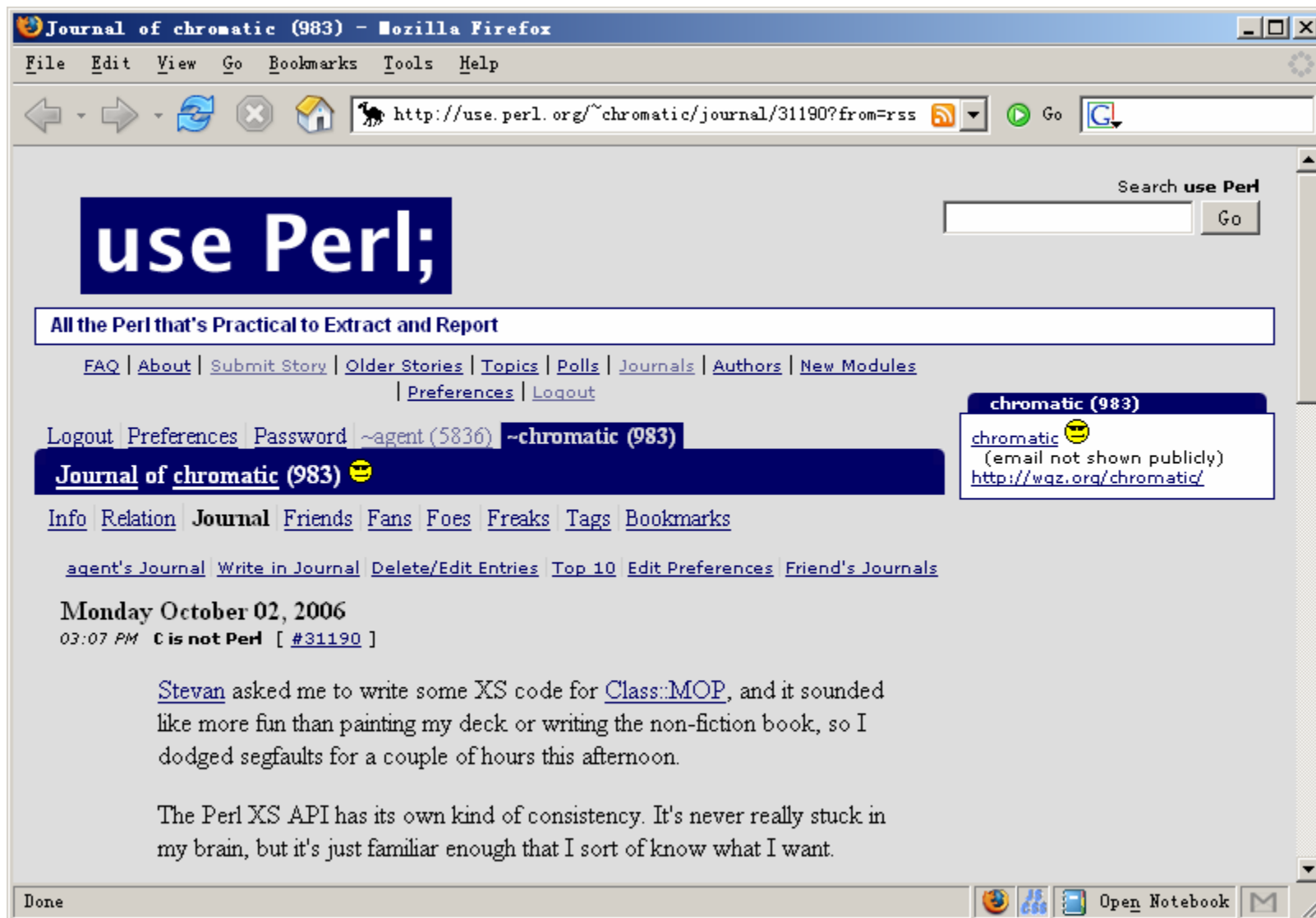
more than 20 items

Done

Open Notebook

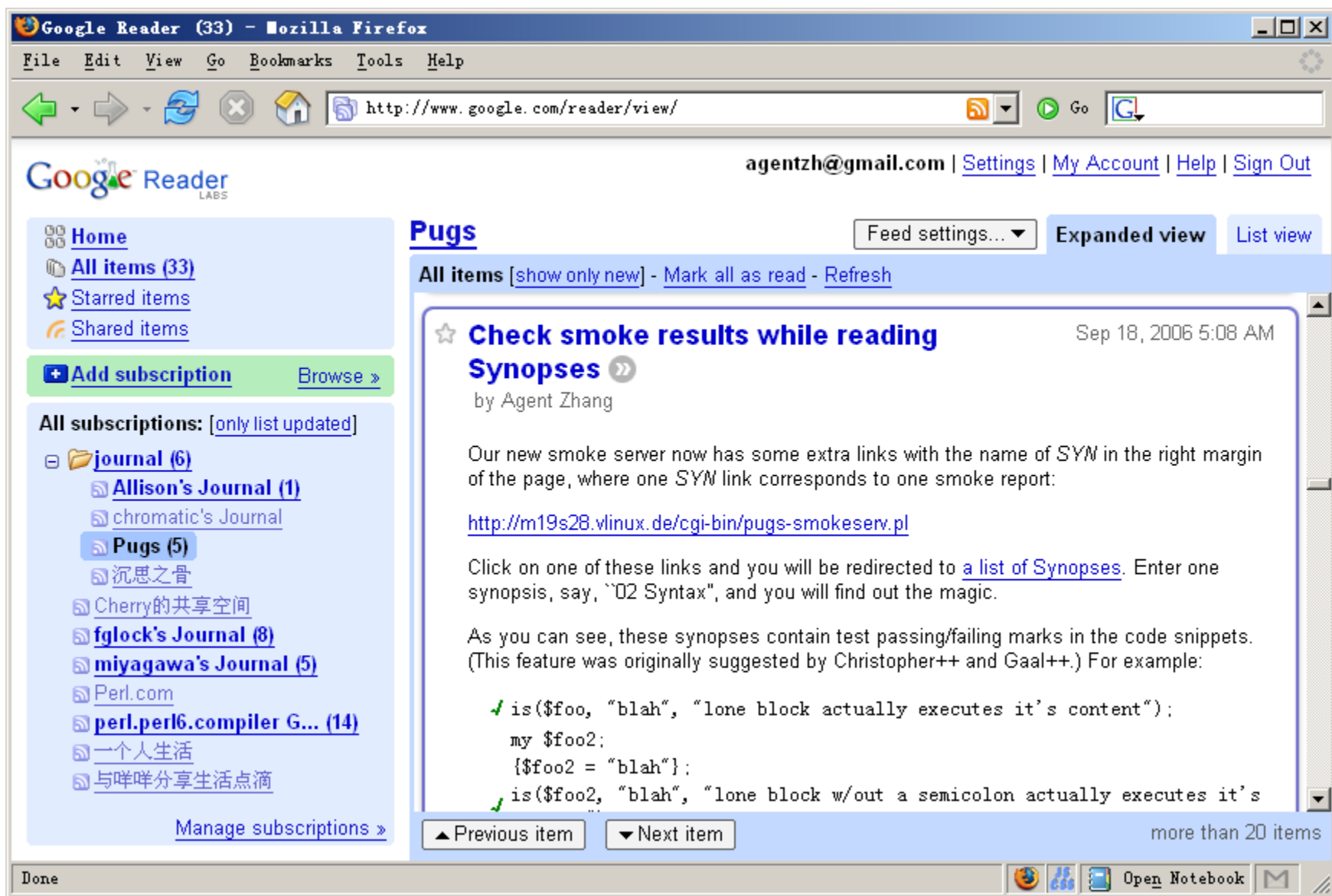
Let's take a look at the *original* journal item
on the use.perl.org site...

让我们来看看
use.perl.org 站点上的原始日记.....



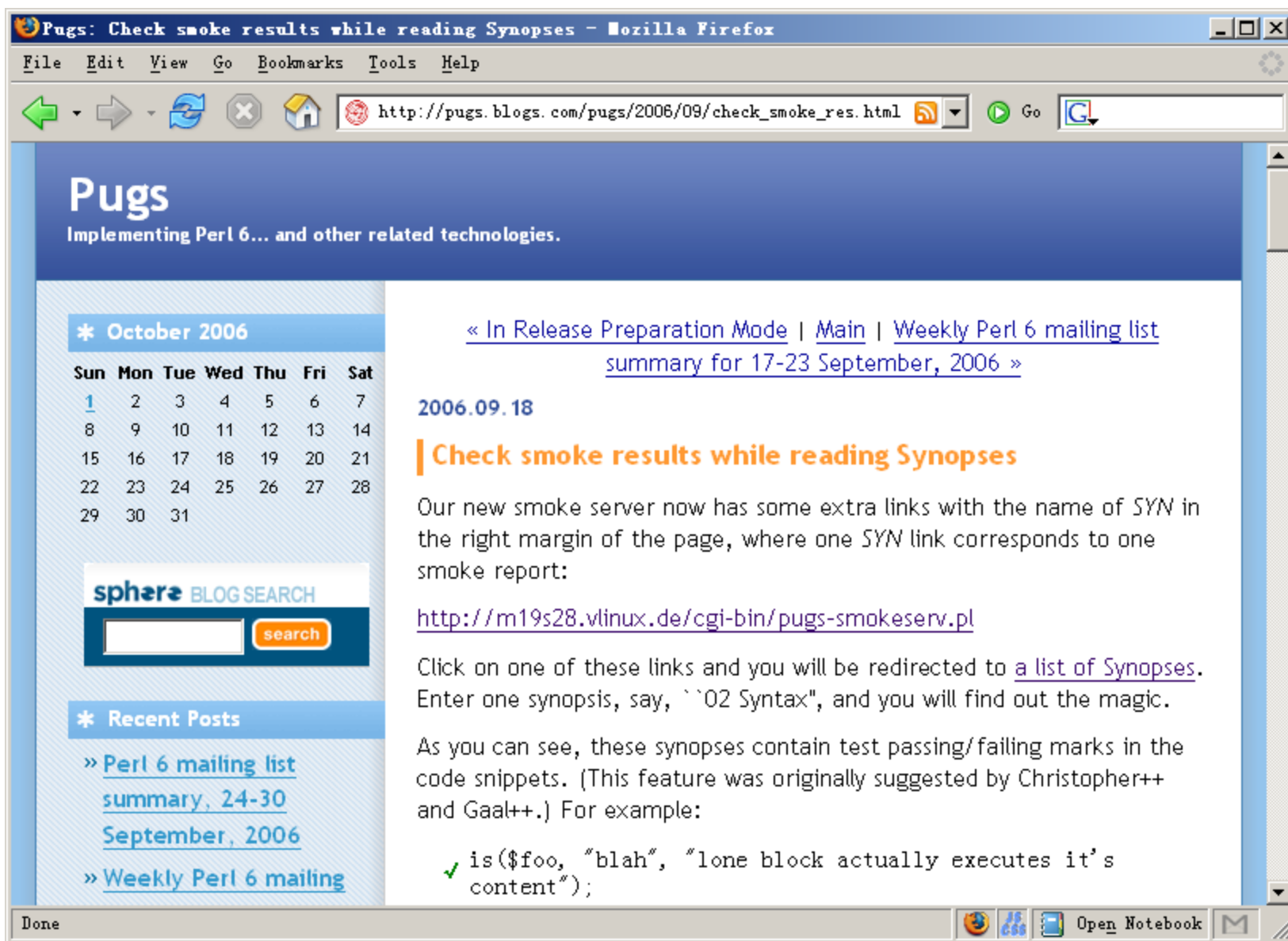
Now let's turn to the *latest* Pugs blog posts
in Google Reader...

现在让我们在 Google Reader
中转向新的 Pugs 博客文章.....



The *original* post I published
onto the pugs.blogs.com site...

我最初发布在
use.perl.org 站点上的帖子.....



The XML magic
behind the curtain...

幕后的 XML 魔法.....

Settings[« Back to Google Reader](#)**Subscriptions** [Tags](#) [Goodies](#) [Import/Export](#) [Preferences](#)Select: [All 9 subscriptions](#), [None](#), [Unassigned](#)

More actions... ▼

Unsubscribe

Filter by name, label, or URL

☐ Allison's Journal http://use.perl.org/~Allison/journal/rss	Rename		Change folders... ▼ journal
☐ Cherry的共享空间 http://cherrychuxinyun.spaces.msn.com/feed.rss	Rename		Add to a folder... ▼
☐ chromatic's Journal http://use.perl.org/~chromatic/journal/rss	Rename		Change folders... ▼ journal
☐ fglock's Journal http://use.perl.org/~fglock/journal/rss	Rename		Add to a folder... ▼
☐ miyagawa's Journal http://use.perl.org/~miyagawa/journal/rss	Rename		Add to a folder... ▼
☐ Perl.com http://www.oreillynet.com/pub/feed/16	Rename		Add to a folder... ▼
☐ Pugs http://pugs.blogs.com/pugs/index.rdf	Rename		Change folders... ▼ journal

RSS feed for *chromatic*'s journals...

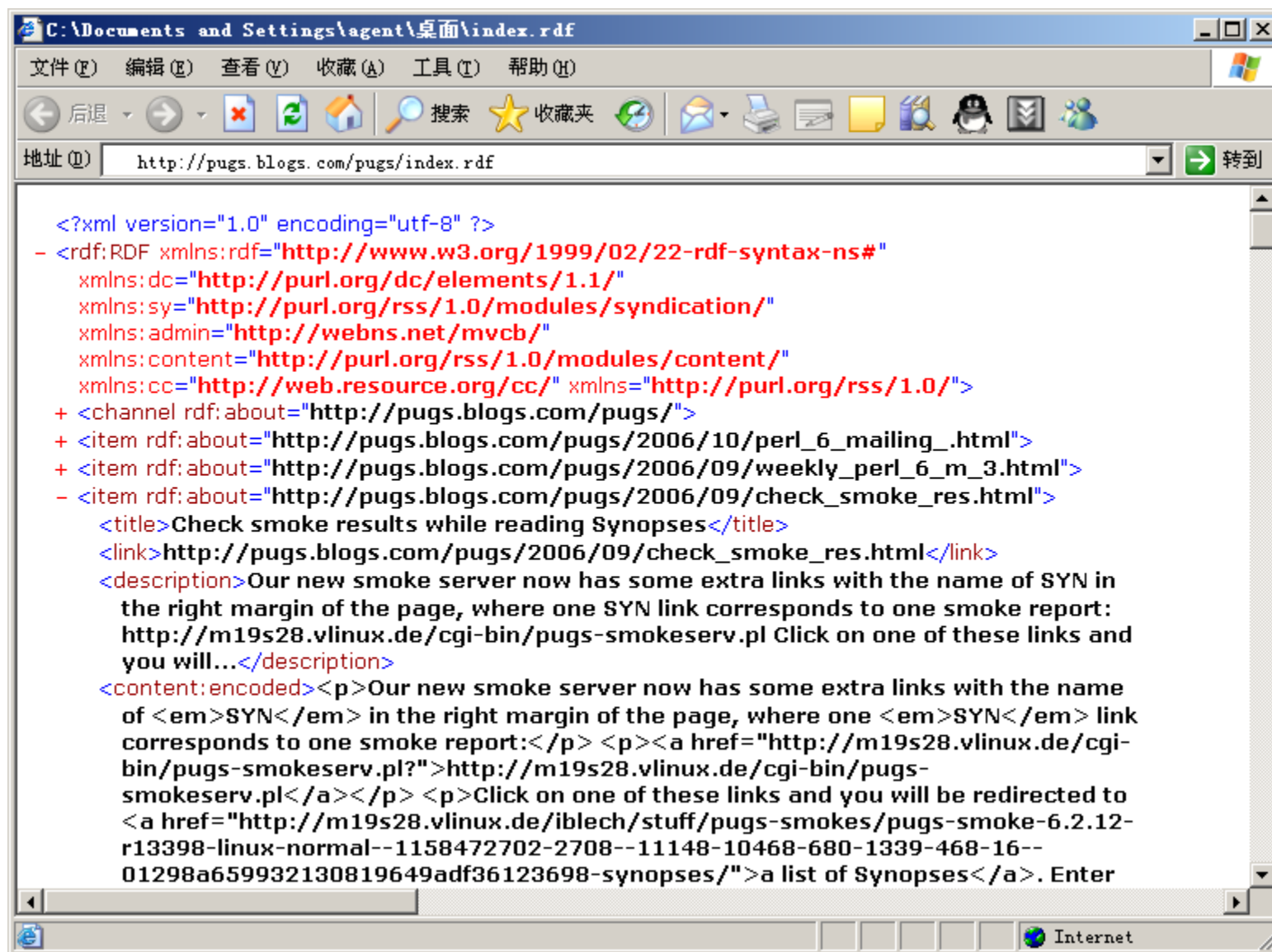
chromatic 的日记的
RSS 反馈.....

地址 @) http://use.perl.org/~chromatic/journal/rss 转到

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns="http://purl.org/rss/1.0/" xmlns:slash="http://purl.org/rss/1.0/modules/slash/"
  xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:syn="http://purl.org/rss/1.0/modules/syndication/"
  xmlns:admin="http://webns.net/mvcb/">
+ <channel rdf:about="http://use.perl.org/~chromatic/journal/">
+ <image rdf:about="http://use.perl.org/images/topics/useperl.gif">
- <item rdf:about="http://use.perl.org/~chromatic/journal/31213?from=rss">
  <title>It's Full of Sharps!</title>
  <link>http://use.perl.org/~chromatic/journal/31213?from=rss</link>
  <description><p>Ahh, so <a
    href="http://www.learningpython.com/2006/10/02/ironpython-hello-world-
    tutorial/">radical simplicity is why dynamic languages on static VMs are
    important</a>. Some days, I even miss Tcl/Tk.</p></description>
  <dc:creator>chromatic</dc:creator>
  <dc:date>2006-10-04T07:45:54+00:00</dc:date>
  <dc:subject>journal</dc:subject>
</item>
- <item rdf:about="http://use.perl.org/~chromatic/journal/31190?from=rss">
  <title>C is not Perl</title>
  <link>http://use.perl.org/~chromatic/journal/31190?from=rss</link>
  <description><p><a href="http://use.perl.org/~stevan">Stevan</a> asked me to
    write some XS code for <a href="http://search.cpan.org/perldoc?
    Class::MOP">Class::MOP</a>, and it sounded like more fun than painting my deck
    or writing the non-fiction book, so I dodged segfaults for a couple of hours this
    afternoon.</p><p>The Perl XS API has its own kind of consistency. It's never
```

RSS feed for
our *Pugs* blog site...

Pugs 博客站点的
RSS 反馈.....



♡ **AJAX**, our *good* friends!

AJAX，我们的好朋友！

AJAX

→ Asynchronous JavaScript and XML

☺ Let's open *Cherry*'s Qzone blogs...

让我们打开 Cherry 的 Qzone 博客.....



☺ *Click* one of the articles and **enter** it...

点击其中的一篇文章进入.....



What happened *behind the curtain*
when we're performing these **actions**?

在我们执行这些动作的时候，
幕后都发生了哪些事情？

Here is the underlying **HTTP traffic**
between the *Qzone site* and my *IE browser*
recorded by **HTTP::Proxy...**

这里有 HTTP::Proxy 模块记录下的
Qzone 站点与我的 IE 浏览器之间
的底层 HTTP 通信.....

[16:04:56] GET http://u13.qzone.qq.com/cgi-bin/cgi_client_entry.cgi?uin=11854905

[16:05:40] GET <http://u13.qzone.qq.com/proxy.html>

...

[16:09:37] GET http://b1.qzone.qq.com/cgi-bin/blog/blog_signature.cgi?uin=11854905

[16:10:00] GET [http://b1.qzone.qq.com/cgi-bin/blog/blog_get_category.cgi?
uin=11854905](http://b1.qzone.qq.com/cgi-bin/blog/blog_get_category.cgi?uin=11854905)

[16:10:00] GET <http://imgcache.qq.com/qzone/proxy.vbs>

[16:10:02] GET [http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?
uin=11854905&blogid=39&flag=0](http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=39&flag=0)

[16:10:04] GET [http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?
uin=11854905&blogid=39&archive=-2](http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=39&archive=-2)

...

Most of the **HTTP requests** were initiated
by the **JavaScript** code
running in your *web browser*.

这些 HTTP 请求中的大部分是由
运行在你的网络浏览器中的
JavaScript 代码发起的。

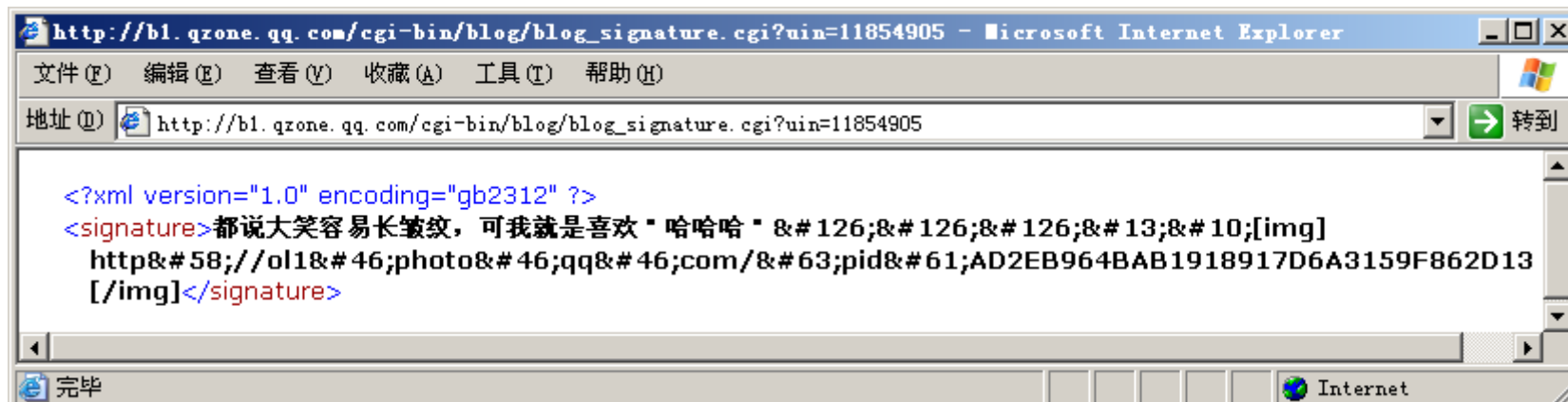
Let's *check* some of
the **HTTP requests** by hand...

让我们来手工查看一下
其中的几个 HTTP 请求.....

XML data for Cherry's *signature*

Cherry 的个性签名所对应的 XML 数据

➔ [http://b1.qzone.qq.com/cgi-bin/blog/
blog_signature.cgi?uin=11854905](http://b1.qzone.qq.com/cgi-bin/blog/blog_signature.cgi?uin=11854905)



XML data for Cherry's article *category* list

Cherry 的文章类别列表所对应的 XML 数据

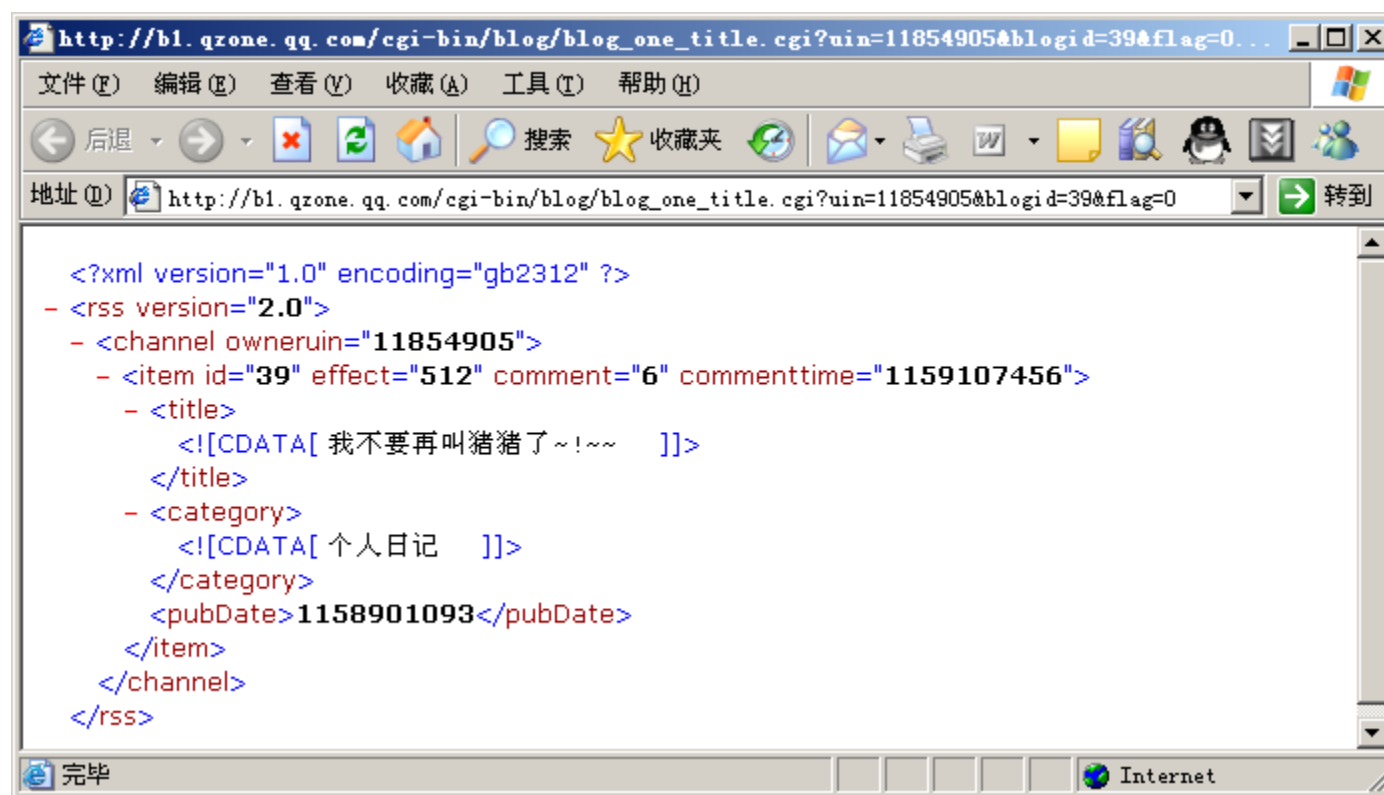
➔ [http://b1.qzone.qq.com/cgi-bin/blog/
blog_get_category.cgi?uin=11854905](http://b1.qzone.qq.com/cgi-bin/blog/blog_get_category.cgi?uin=11854905)



XML data for the *title* of
Cherry's 40th post (with ID 39)

Cherry 的第 40 篇帖子(标识为 39) 的标题
所对应的 XML 数据

➔ [http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?
uin=11854905&blogid=39&flag=0](http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=39&flag=0)



XML data for the *body and comments* of
Cherry's 40th post (with ID 39)

Cherry 的第 40 篇帖子(标识为 39) 的正文及评论
所对应的 XML 数据

➔ [http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?
uin=11854905&blogid=39&archive=-2](http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=39&archive=-2)



Our **web browser** renders these **XML data** files using **HTML templates** sent by the Qzone server, and generates the final **HTML source**.

我们的网络浏览器根据 Qzone 服务器传过来的 HTML 模板对这些 XML 数据进行渲染, 生成最终的 HTML 源码。

XML data + HTML templates = final HTML source

XML 数据 + HTML 模板 = 最终的 HTML 源码

The *whole* process happens
in our **web browser**.

整个过程都发生在我们的浏览器内部。

But *where* are
the **HTML templates**?

但是 HTML 模板究竟在哪里呢？

Let's check the *raw* HTML source
sent from the Qzone *server*

让我们来看看 Qzone 服务器
传过来的原始 HTML 源码



...

<!-- 日志/-->

<div id="tpl_blog_b" class="mode_table" style="display:none">...

<table cellSpacing="0" cellpadding="0" width="100%" class= ...

[%repeat_0 match="/rss/channel/item" repeat_num="10"%]

<tr><td class="index_blog_btd">

[<a href="#" onclick="openCategory('\'**[%=@type%]**\');return false"

...title="点击进入分类">**[%=@category%]**]

<a href="#" title="**[%=@title%]** -- 发表于 **[%=@pubTimeString%]**"

onClick="openBlog('\'**[%=@archive%]**\',\'**[%=@id%]**\');return false">

... </td>

<td class="info">评论(**[%=@comment%]**)</td>

[%_repeat_0%]

</table>

</div>

...

You see,
it's a *client*-side HTML *template*!



你看，
这是一个客户端的 HTML 模板！

It's the JavaScript code that *grabs* the XML data from the web and fills it into the HTML templates automatically, resulting in the final appearance we see in the browser.

所以是 JavaScript 代码自动从网上获取 XML 数据并将之填入到 HTML 模板中，最终得到我们在浏览器中看到的效果。

Then **why can't** we do XML data
grabbing *ourselves*?

那么为什么我们就不可自己去
攫取 XML 数据呢？

For example, we can obtain the **data**
for **all** of Cherry's articles
by simply changing the **URL**!

比如，我们可以通过简单地修改网址
得到的有文章的数据！

http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=39&flag=0

http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=39&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=38&flag=0

http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=39&flag=0

http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=38&flag=0

http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=37&flag=0

http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=39&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=38&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=37&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=36&flag=0

http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=39&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=38&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=37&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=36&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=35&flag=0

http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=39&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=38&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=37&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=36&flag=0
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=35&flag=0
...
http://b1.qzone.qq.com/cgi-bin/blog/blog_one_title.cgi?uin=11854905&blogid=0&flag=0

http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=39&archive=-2

http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=39&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=38&archive=-2

http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=39&archive=-2

http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=38&archive=-2

http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=37&archive=-2

http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=39&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=38&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=37&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=36&archive=-2

http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=39&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=38&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=37&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=36&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=35&archive=-2

http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=39&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=38&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=37&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=36&archive=-2
http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=35&archive=-2

...

http://b1.qzone.qq.com/cgi-bin/blog/blog_commentlist.cgi?uin=11854905&blogid=0&archive=-2

It means that...

now we can *directly* access Qzone's **database**,
completely *bypassing* its cumbersome HTML interface!

这意味着.....

现在我们可以直接访问 Qzone 的数据库
完全绕过它那笨重的 HTML 界面！

Let's write a tiny **Perl**
script to do *all* these tricks
for us!

让我们来编写一个小小的 Perl
脚本来为我们实现所有这些把戏。

```
C:\ VC 2003 命令提示符

D:\projects\UniSimu\Perl\GetQzone>update 279005114
Property ID: 1207E6068A02A427F58EE6BDBB6937EF6864CEC513203144
  info: last blog's id is 6.
  info: 0 message(s) found in message board.
Blog 6
Blog 5
Blog 4
Blog 3
Blog 2
Blog 1
Blog 0
  info: 7 article(s) found in blogs.
279005114.yml generated.
279005114.html generated.

D:\projects\UniSimu\Perl\GetQzone>
```

One **sample output** of the program
for *Cherry*'s Blogs

➔ <http://perlcabal.org/agent/cherry.html>

该程序针对 Cherry 的博客的一次典型输出





This program uses `LWP::UserAgent` to *get* the XML data directly from the web and uses `XML::Simple` to *parse* it.

该程序使用 LWP::UserAgent 模块
直接从网上获取 XML 数据，并利用
XML::Simple 解析之。

No need for Audrey's **Template::Extract**
to *extract* data from the HTML source.
That is the *power* of **AJAX** and **XML**!

不再需要唐凤的 Template::Extract 模块来
从 HTML 源码中提取数据。
这就是 AJAX 和 XML 的威力！

Get the **slides** today!



<http://agentzh.org/misc/slides/xmlapp.pdf>

Thank you!



