

# **SEARCHMONKEY APPLICATION**

**How to make a CUSTOM DATA SERVICE for use with an  
Infobar (using XPather)**

# OVERVIEW

SearchMonkey apps can display content scraped from almost any page on the internet.

Using a programming language called [XSLT](#), this tutorial will teach you how to create a “custom data service” which will feed content to the SearchMonkey apps you will learn to create in the next tutorial.

Don't worry, this sounds more technically complex than it actually is! **You don't need to know XSLT code** to get your Custom Data Service up and running, thanks to the good people who made the wonderful XPather Firefox Add-on.

# Custom Data Service – To Start:

1. Download **XPather**, the Firefox Add-on

<https://addons.mozilla.org/en-US/firefox/addon/1192>

This extension will help you to quickly and easily grab the XSLT code from a webpage and paste that code into your Custom Data Service

2. Go to <http://developer.yahoo.com/searchmonkey/> and Click on “Build an App”.

3. SCROLL DOWN to the *Custom Data Services* section and click on “Create a new data service”.

# Building your Custom Data Service

## Step 1: Basic Info

The screenshot shows the 'Basic Info' step of the 'Download.com data service' setup. The page has a yellow header with the Yahoo! logo and 'APPLICATION PLATFORM SearchMonkey'. A search bar is present. Below the header, the breadcrumb 'Application Dashboard / Download.com data service' is shown. A 'File a bug' button is in the top right. The 'Basic Info' tab is selected, with other tabs for 'URLs', 'Data Extraction', and 'Confirmation'. A green heading reads 'Step 1: Specify your custom data service's name, type, and description.' A note explains that custom data services are slower than the Yahoo! Index. The 'Name' field contains 'Download.com data service' and is marked as required. The 'Type' section shows 'Page' selected with a radio button, with a description '(extracts information from a web page)'. The 'Web Service' option is also available. The 'Description' field contains 'Custom data service for download.com.'. At the bottom, the 'Terms of Service' section has a checked checkbox for 'I have read and agree to the SearchMonkey Terms of Service', which is also marked as required. 'Next Step' and 'Cancel' buttons are at the bottom.

Yahoo! | My Yahoo! | Mail | More ▼ | Make Y! Your Home Page | Welcome, Chris ▼ | Sign Out | Help

YAHOO! APPLICATION PLATFORM SearchMonkey

Search WEB SEARCH

Application Dashboard / Download.com data service [File a bug](#)

Basic Info | URLs | Data Extraction | Confirmation

**Step 1: Specify your custom data service's name, type, and description.**

NOTE: Custom data services are intrinsically slower than data residing in the Yahoo! Index. If a custom data service is particularly slow, presentation applications that use it might automatically revert to infobar style, in order to provide users with faster access to basic search results.

Name:  (\* Required)

Type: ☒ Page (extracts information from a web page) ☐ Web Service (calls a web service and transforms the results)

Description:

Terms of Service

☒ I have read and agree to the [SearchMonkey Terms of Service](#) (\* Required)

[Next Step](#) [Cancel](#)

1. Give your Data Service a nice name.
2. Select “Page” for Type.
3. Give your Data Service a nice description.
4. Agree to the Terms of Service. Thanks for being agreeable.
5. Click NEXT and you should be magically transported to the URLs page.

# URLs - Trigger URL Pattern Overview

The screenshot shows the Yahoo! Application Platform interface. At the top, there's a navigation bar with links like 'Yahoo!', 'My Yahoo!', 'Mail', 'More', and a 'Welcome, Chris' message. Below this is the 'APPLICATION PLATFORM' header with a 'Search' button. The main content area is titled 'Application Dashboard / Download.com data service'. There are tabs for 'Basic Info', 'URLs', 'Data Extraction', and 'Confirmation'. The 'URLs' tab is selected, and the sub-header is 'Step 2: Specify your trigger and test URLs.' The instructions state: 'Create the Trigger URL. Specify a URL pattern to match against Yahoo! Search results and trigger your application. You may use the wildcard '\*' to match any string at the beginning or end of your pattern. For example, \*.wikipedia.org/wiki/\* causes your application to trigger on all Wikipedia pages, while en.wikipedia.org/wiki/\* only triggers your application on English Wikipedia pages. You may separate multiple wildcards by a ', ' so \*.wikipedia.org/wiki/\*,\*.wikimedia.org/\* will work on both Wikipedia and Wikimedia.' Below this, the 'Trigger URL Pattern:' field is highlighted with a red box and contains the text '\*download.com/\*'. At the bottom, there's a section 'Add Test URLs' with a list of 10 URLs generated by the 'AUTOFIND URLS' button. The buttons at the bottom are 'Next Step', 'Back', and 'Cancel'.

Yahoo! | My Yahoo! | Mail | More ▼ Make Y! Your Home Page Welcome, Chris ▼ Sign Out | Help

**YAHOO! APPLICATION PLATFORM** SearchMonkey

Application Dashboard / Download.com data service [File a bug](#)

Basic Info **URLs** Data Extraction Confirmation

**Step 2: Specify your trigger and test URLs.**

Create the Trigger URL

Specify a URL pattern to match against Yahoo! Search results and trigger your application. You may use the wildcard '\*' to match any string at the beginning or end of your pattern. For example, \*.wikipedia.org/wiki/\* causes your application to trigger on all Wikipedia pages, while en.wikipedia.org/wiki/\* only triggers your application on English Wikipedia pages. You may separate multiple wildcards by a ', ' so \*.wikipedia.org/wiki/\*,\*.wikimedia.org/\* will work on both Wikipedia and Wikimedia.

**Trigger URL Pattern:**

\*download.com/\*

**Add Test URLs**

You may specify up to ten test URLs. The results of your application running on a given test URL appear in the **Preview Pane** at the bottom of the screen. Use the **Preview Pane** to cycle through your test URLs and verify that your application is working properly. If you click **Autofind URLs**, SearchMonkey automatically selects ten test URLs for you.

**AUTOFIND URLS**

1. [http://music.download.com/canseidesersexy/3600-8592\\_32-1002020](http://music.download.com/canseidesersexy/3600-8592_32-1002020)
2. [http://music.download.com/thewhitestripes/3600-8691\\_32-101066148](http://music.download.com/thewhitestripes/3600-8691_32-101066148)
3. [http://www.download.com/Avast-Home-Edition/3000-2239\\_4-1001922](http://www.download.com/Avast-Home-Edition/3000-2239_4-1001922)
4. [http://www.download.com/FLV-Player/3000-2139\\_4-10467081.html](http://www.download.com/FLV-Player/3000-2139_4-10467081.html)
5. [http://www.download.com/Free-iPod-Video-Converter/3000-2194\\_4-10](http://www.download.com/Free-iPod-Video-Converter/3000-2194_4-10)
6. [http://www.download.com/Palm-OS/2001-2008\\_4-0.html](http://www.download.com/Palm-OS/2001-2008_4-0.html)
7. [http://www.download.com/Antivirus/3150-2239\\_4-0.html](http://www.download.com/Antivirus/3150-2239_4-0.html)
8. [http://www.download.com/PrimoPDF/3000-10743\\_4-10575974.html?i](http://www.download.com/PrimoPDF/3000-10743_4-10575974.html?i)
9. [http://www.download.com/3150-2641\\_4-0.html](http://www.download.com/3150-2641_4-0.html)
10. <http://www.download.com/Trellian/3000-2150-10047473.html>

**Next Step** **Back** **Cancel**

## A note on making a Trigger URL Pattern:

Trigger URL patterns are simply the root url of any page, minus the beginning portion.

For example, if you wanted to extract information from the Pitchforkmedia.com Album Review pages, you would go to:  
<http://www.pitchforkmedia.com/article/record>

From this URL you would remove <http://www> and the page name, so that you are left with the root:  
<http://www.pitchforkmedia.com/article/record> review/\*

If, on the other hand, you wanted to pull content from Pitchfork's feature stories, your "Trigger URL Pattern" would look more like this: [pitchforkmedia.com/article/feature](http://www.pitchforkmedia.com/article/feature)/\*. Because Pitchfork keeps features in that folder, as you can see from this feature:  
<http://www.pitchforkmedia.com/article/feature>

# Building your Custom Data Service

## Step 2: URLs

Application Dashboard / download.com demo data service [File a bug](#)

Basic Info **URLs** Data Extraction Confirmation

**Step 2: Specify your trigger and test URLs.**

Create the Trigger URL

Specify a URL pattern to match against Yahoo! Search results and trigger your application. You may use the wildcard '\*' to match any string at the beginning or end of your pattern. For example, \*.wikipedia.org/wiki/\* causes your application to trigger on all Wikipedia pages, while en.wikipedia.org/wiki/\* only triggers your application on English Wikipedia pages. You may separate multiple wildcards by a '|', so \*.wikipedia.org/wiki/\*|\*.wikimedia.org/\* will work on both Wikipedia and Wikimedia.

Trigger URL Pattern:

---

Add Test URLs

You may specify up to ten test URLs. The results of your application running on a given test URL appear in the **Preview Pane** at the bottom of the screen. Use the **Preview Pane** to cycle through your test URLs and verify that your application is working properly. If you click **Autofind URLs**, SearchMonkey automatically selects ten test URLs for you.

**AUTOFIND URLs**

1.
2.
3.
4.

1. For the tutorial, we will be extracting data from <http://www.download.com/>
  - Enter **\*.download.com/\*** as the *Trigger URL Pattern*.
  3. Click **Autofind URLs**
- NOTE:** If **Autofind URLs** isn't working for you, you can just enter five or six URLs by hand. For the sake of this tutorial, just make sure you have <http://www.download.com/Trillian/3000-2> (Download.com's Trillian page) in the first slot so that your screen shots match mine.
4. Click NEXT and you should be magically transported to the Data Extraction page...

# Building your Custom Data Service

## Step 3: Data Extraction

Yahoo! My Yahoo! Mail More ▼ Make Y! Your Home Page Welcome, Chris Sign Out Help

**YAHOO! APPLICATION PLATFORM** SearchMonkey

Application Dashboard / download.com demo data service [File a bug](#)

Basic Info **URLs** **Data Extraction** Confirmation

**Step 3: Define your page extraction rules.**

Specify [XSLT](#) code for extracting information from the page and representing that information as [DataRSS](#). Your [XSLT](#) will operate on a cleaned-up version of the page's HTML. SearchMonkey sets a single attribute, `{CURRURL}` set to the URL of the page you are extracting data from. Avoid using namespaces in your XPATH expressions, as SearchMonkey strips these out.

For assistance with writing XPATH expressions, try the [XPather Firefox extension](#). Once you install XPather, you can select text on the page, right-click, and select "show in XPather" to generate an XPath expression that matches the selected text.

See an [Example XSLT](#)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
<adjunctcontainer>
<adjunct id="smid:{smid}" version="1.0">

<!-- here is a simple example of data extraction -->
<!-- modify this or replace it with your own code -->

<item rel="rel:Photo" resource="//div[@class='picture']/img/@src">
  <meta property="media:width"><xsl:value-of
select="//div[@class='picture']/img/@width"/></meta>
  <meta property="media:height"><xsl:value-of
select="//div[@class='picture']/img/@height"/></meta>
  <item rel="rel:Thumbnail" resource="//div[@class='thumb']/img/@src">
    <meta property="media:width"><xsl:value-of
select="//div[@class='thumb']/img/@width"/></meta>
    <meta property="media:height"><xsl:value-of
select="//div[@class='thumb']/img/@height"/></meta>
  </item>
</item>

<!--
  many other kinds of <item> resources and <meta> properties
  are available, consult the documentation and examples at
  http://developer.yahoo.com/searchmonkey/smguide/datarss.html
-->

</adjunct>
</adjunctcontainer>
</xsl:template>
```

1. On the Data Extraction page, delete the text in the middle starting with "`<item`" and ending with `</item>` so your page looks like this:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
<adjunctcontainer>
<adjunct id="smid:{smid}" version="1.0">

<!--
  many other kinds of <item> resources and <meta> properties
  are available, consult the documentation and examples at
  http://developer.yahoo.com/searchmonkey/smguide/datarss.html
-->

</adjunct>
</adjunctcontainer>
</xsl:template>
</xsl:stylesheet>
```

NOTE: Clicking "Next Step" will take a while, as we load all your test URLs.

# Building your Custom Data Service

## Step 3: Data Extraction - Extracting a **Picture** with XPather

- In a different browser window, open the page you want to extract data from. For the sake of this tutorial, please open:  
<http://www.download.com/Trillian/3000-2150-1004>

Below is a screenshot with the elements that you will be extracting circled, numbered, and highlighted in a lovely yellow. (1. LINK, 2. PICTURE, 3. TEXT)

### Trillian 3.1.10.0

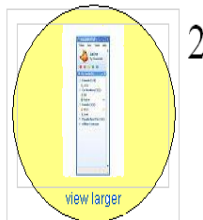


License: Free  
Editor's Rating: ★★★★★  
Average User Rating: ★★★★★ (out of 1434 votes) [Rate](#)  
Downloads: 36,072,060  
Requirements: Windows 95/98/Me/2000/XP/2003 Server/Vista  
Limitations: No limitations  
Date Added: May 19, 2008

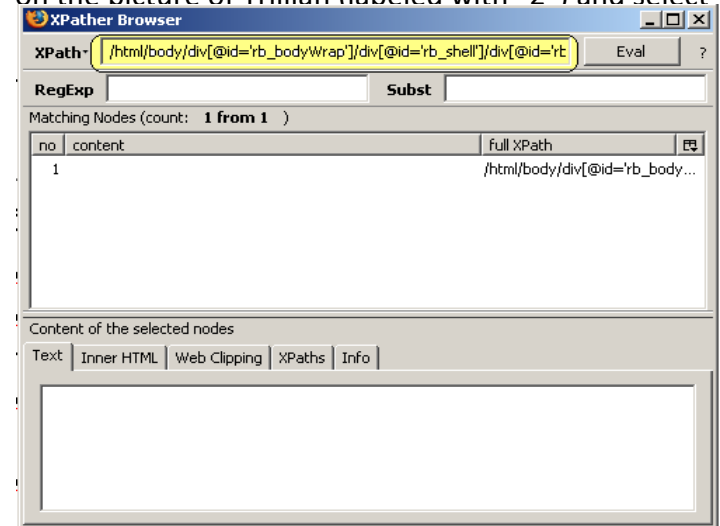
### Publisher's description of Trillian

From Cerulean Studios: 3

Trillian is a fully featured, stand-alone, skinnable chat client that supports AIM, ICQ, MSN, Yahoo Messenger, and IRC. It provides capabilities not possible with original network clients, while supporting standard features such as audio chat, file transfers, group chats, chat rooms, buddy icons, multiple simultaneous connections to the same network, server-side contact importing, typing notification, direct connection (AIM), proxy support, encrypted messaging (AIM/ICQ), SMS support, tabbed chatting, system-tray notifications and alerts, and privacy settings.



- We are going to extract the picture first, so right click on the picture of Trillian (labeled with "2") and select



- Copy the "XPath" at the top (highlighted in a lovely yellow)
- Paste that XPath into the Data Extraction page in the Dashboard so that you end up with this...



# Building your Custom Data Service

## Step 3: Data Extraction - Extracting a **Picture** with XPath

Application Dashboard / download.com demo data service [File a bug](#)

Basic Info

URLs

**Data Extraction**

Confirmation

### Step 3: Define your page extraction rules.

Specify [XSLT](#) code for extracting information from the page and representing that information as [DataRSS](#). Your [XSLT](#) will operate on a cleaned-up version of the page's HTML. SearchMonkey sets a single attribute, {[\\$CURRURL](#)} set to the URL of the page you are extracting data from. Avoid using namespaces in your [XPath](#) expressions, as SearchMonkey strips these out.

For assistance with writing [XPath](#) expressions, try the [XPather Firefox extension](#). Once you install XPather, you can select text on the page, right-click, and select "show in XPather" to generate an [XPath](#) expression that matches the selected text.

See an [Example XSLT](#)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
<adjunctcontainer>
<adjunct id="smid:{$smid}" version="1.0">

/html/body/div[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/div/a[1]/img

<!--
many other kinds of <item> resources and <meta> properties
are available, consult the documentation and examples at
http://developer.yahoo.com/searchmonkey/smguide/datarss.html
-->

</adjunct>
</adjunctcontainer>

</xsl:template>
</xsl:stylesheet>
```

NOTE: Clicking "Next Step" will take a while, as we load all your test URLs.

**good job! now, you have to tweak the code a bit...**

# Building your Custom Data Service

## Step 3: Data Extraction - Tweaking the picture's XPath

Right now, your picture code looks like this



/  
**html/body/div[@id='rb\_bodyWrap']/div[@id='rb\_shell']/div[@id='rb\_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/div/a[1]/img**



You need it to look like this

**<meta rel="rel:photo"  
resource="{//\*[@id='rb\_bodyWrap']/div[@id='rb\_shell']/div[@id='rb\_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/div/a[1]/img/@src}" />**

But how??? Well...

# Building your Custom Data Service

## Step 3: Data Extraction - Tweaking the picture's XPath

- First, trim off everything at the beginning of your line of code until you get to the first **div**. In this case, you're going to trim off **/html/body/** from the beginning. So you're left with:

```
div[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/div/a[1]/img
```

- Next, add **/@src** to the end of the line. NOTE: If you were extracting a link, you'd add **/@href**. And for text, add nothing.

```
div[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/div/a[1]/img/@src
```

- You're extracting a picture, so replace that **div** near the beginning of the line with an asterisk (\*). NOTE: For text and links, leave it as **div**.

```
*[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/div/a[1]/img/@src" />
```

# Building your Custom Data Service

## Step 3: Data Extraction - Tweaking the picture's XPath

4. Next, enclose your line of code with **resource="{//** at the beginning, and **}" />** at the end (Remember to put a space between the end quote and back slash). Now, your code should look like this:

```
resource="{//*[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/div/a[1]/img/@src}" />
```

5. Then, give it a name tag at the beginning of the line. The name is for your use only to describe what you have extracted. I'm going to call this data resource "photo" by adding **<meta rel="rel:photo"** and then a space. So your finished product should look like this:

```
<meta rel="rel:photo"
resource="{//*[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/div/a[1]/img/@src}" />
```

**NOTE:** If you were extracting a link, you'd add **<item rel** at the beginning.

# Building your Custom Data Service

## Step 3: Data Extraction - Tweaking the picture's XPath

6. Click Save & Refresh, and now the lower window panel should look just like this:



YOU DID IT! I'm proud of you. Next: Extracting Text...

\* If it doesn't look like the above example, see our troubleshooting page.

# Building your Custom Data Service

## Step 3: Data Extraction - Extracting **Text** with XPather

- Go back to the Trillian download.com page:  
<http://www.download.com/Trillian>
- Highlight the text on that page that you want to extract.  
For now, you're going to highlight the Publisher's Description of Trillian. Like in this screenshot →
- Right click on the highlighted text and select "Show in XPather" from the drop-down menu.

### Trillian 3.1.10.0



**Download Now** (8.64MB)

Tested spyware free

License:	Free
Editor's Rating:	★★★★★
Average User Rating:	★★★★☆ (out of 1436 votes) <a href="#">Rate it!</a>
Downloads:	36,079,421
Requirements:	Windows 95/98/Me/2000/XP/2003 Server/Vista
Limitations:	No limitations
Date Added:	May 19, 2008

### Publisher's description of Trillian

From **Cerulean Studios**:

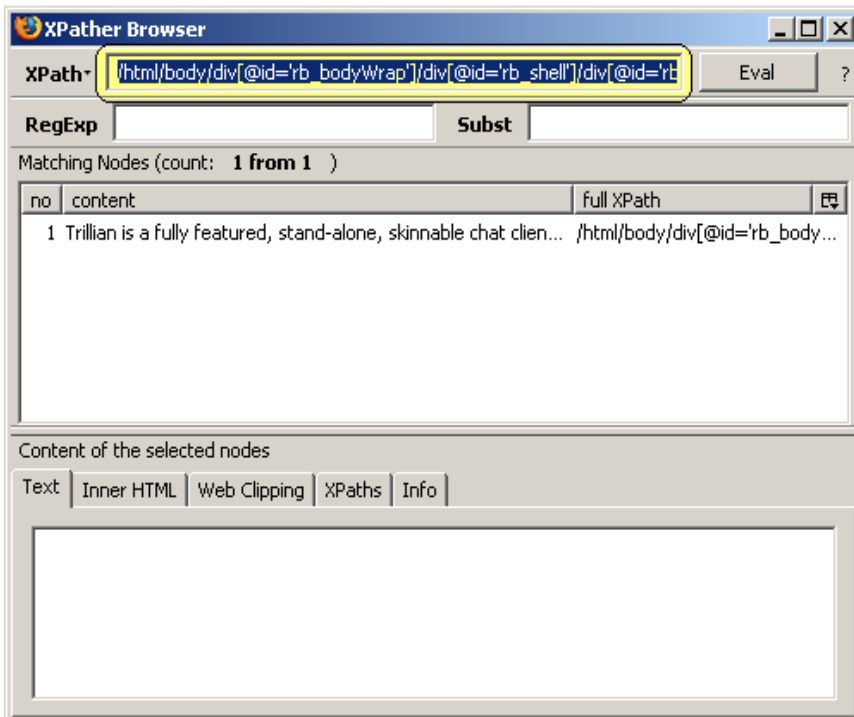
Trillian is a fully featured, stand-alone, skinnable chat client that supports AIM, ICQ, MSN, Yahoo Messenger, and IRC. It provides capabilities not possible with original network clients, while supporting standard features such as audio chat, file transfers, group chats, chat rooms, buddy icons, multiple simultaneous connections to the same network, server-side contact importing, typing notification, direct connection (AIM), proxy support, encrypted messaging (AIM/ICQ), SMS support, tabbed chatting, system-tray notifications and alerts, and privacy settings.



# Building your Custom Data Service

## Step 3: Data Extraction - Extracting **Text** with XPath

Your Xpath Browser should look like this (see below):



4. Select and COPY the text in the "XPath" box (highlighted in a lovely yellow above).

5. Paste that text into the Search Monkey application dashboard, a line or two below your "rel:photo" code, so that your dashboard looks like this (see below):



NOTE: Clicking "Next Step" will take a while, as we load all your test URLs.

# Building your Custom Data Service

## Step 3: Data Extraction – Tweaking the **Text's** XPath

Freshly copied from XPather, the text XPath looks like this:

```
/html/body/div[@id='rb_bodyWrap']/div[@id='rb_shell']  
/div[@id='rb_content']/div[@id='contentMain']/div[@id=  
'contentBody']/div[@id='publisherDesc']/p[1]
```

But, in order for the Search Monkey tool to make sense of it, you need it to look like this:

```
<meta rel="rel:text"  
resource="{//div[@id='rb_bodyWrap']/div[@id='rb_shell']  
/div[@id='rb_content']/div[@id='contentMain']/div[@id=  
'contentBody']/div[@id='publisherDesc']/p[1]}" />
```



# Building your Custom Data Service

## Step 3: Data Extraction - Tweaking the **Text's** XPath

1. First trim off everything at the beginning of your line of code until you get to the first **div**. In this case, you're going to trim off `/html/body/` from the beginning. So you're left with:

```
div[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/p[1]
```

- Then, enclose your line of code with `resource="{//` at the beginning, and `}"/>` at the end (remember to include a space between the end quote and the back slash). Now, your code should look like this:

```
resource="{//div[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/p[1]}" />
```

**NOTE:** We are extracting text, so we're going to leave that **div** near the beginning of the line. If we were extracting a picture, we would replace that **div** with an asterisk `*`.

3. Then, give it a name. I'm going to call this data resource "text" by adding `<meta rel="rel:text"` and a **SPACE** to the beginning of the line so that it looks like this:

```
<meta rel="rel:text"
resource="{//div[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='publisherDesc']/p[1]}" />
```

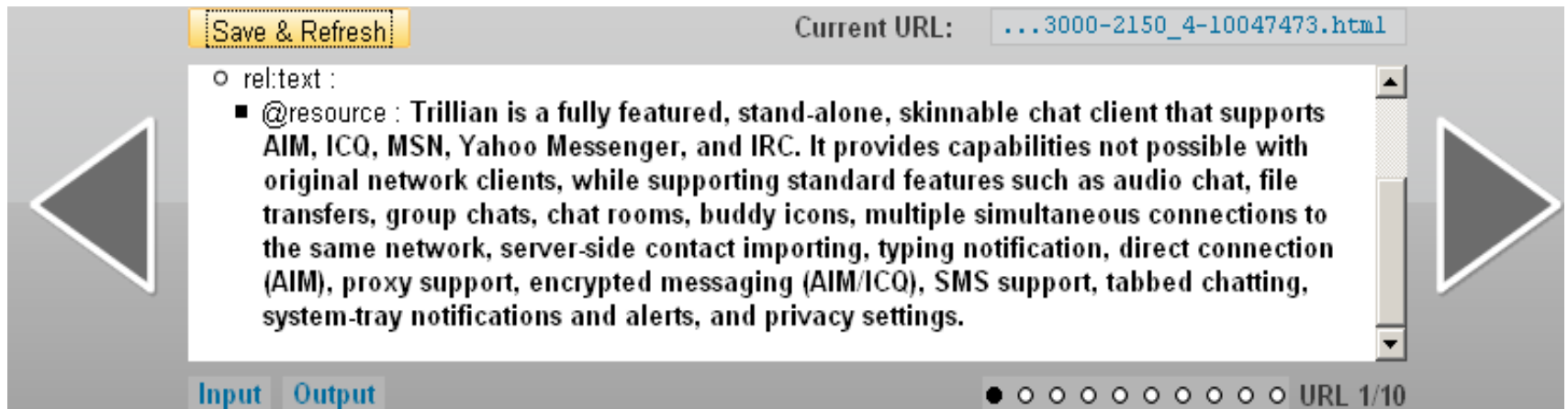
# Building your Custom Data Service

## Step 3: Data Extraction - Tweaking the **Text's** XPath

4. Your finished product should look like this:

```
<meta rel="rel:text" resource="{//div[@id='rb_bodyWrap']/  
  
div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody'  
]/div[@id='publisherDesc']/p[1]}" />
```

5. Now click Save & Refresh, and now the lower window panel should look just like this:



**YAY! Next: Extracting Links...**

# Building your Custom Data Service

## Step 3: Data Extraction – Extracting a **Link** with XPather

- Go back to your open download.com page:  
<http://www.download.com/Trillian/>
- 3. Right Click on the link you want to extract.  
For now, grab the big "Download Now" link at the top (highlighted here in a lovely yellow) →
- 6. Select "Show in XPather" from the menu that appears.  
Now, your XPather Browser should look like this (next page)...

### Trillian 3.1.10.0



**Download Now** (8.64MB)

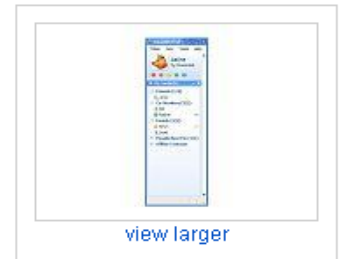
Tested spyware free

License:	Free
Editor's Rating:	★★★★★
Average User Rating:	★★★★☆ (out of 1436 votes) <a href="#">Rate it!</a>
Downloads:	36,086,881
Requirements:	Windows 95/98/Me/2000/XP/2003 Server/Vista
Limitations:	No limitations
Date Added:	May 19, 2008

### Publisher's description of Trillian

#### From Cerulean Studios:

Trillian is a fully featured, stand-alone, skinnable chat client that supports AIM, ICQ, MSN, Yahoo Messenger, and IRC. It provides capabilities not possible with original network clients, while supporting standard features such as audio chat, file transfers, group chats, chat rooms, buddy icons, multiple simultaneous connections to the same network, server-side contact importing, typing notification, direct connection (AIM), proxy support, encrypted messaging (AIM/ICQ), SMS support, tabbed chatting, system-tray notifications and alerts, and privacy settings.



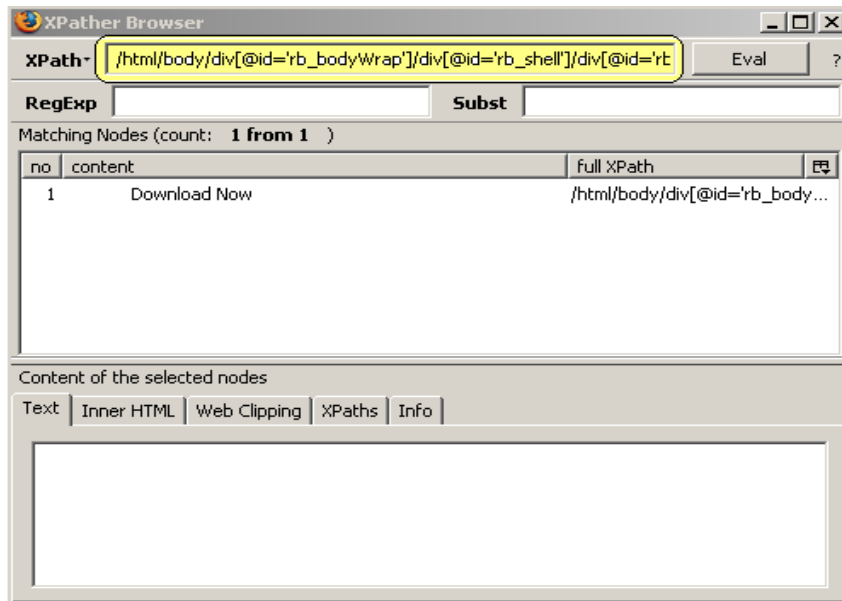
[view larger](#)

Version 3.1.10.0 is a maintenance release and resolves a security bug.

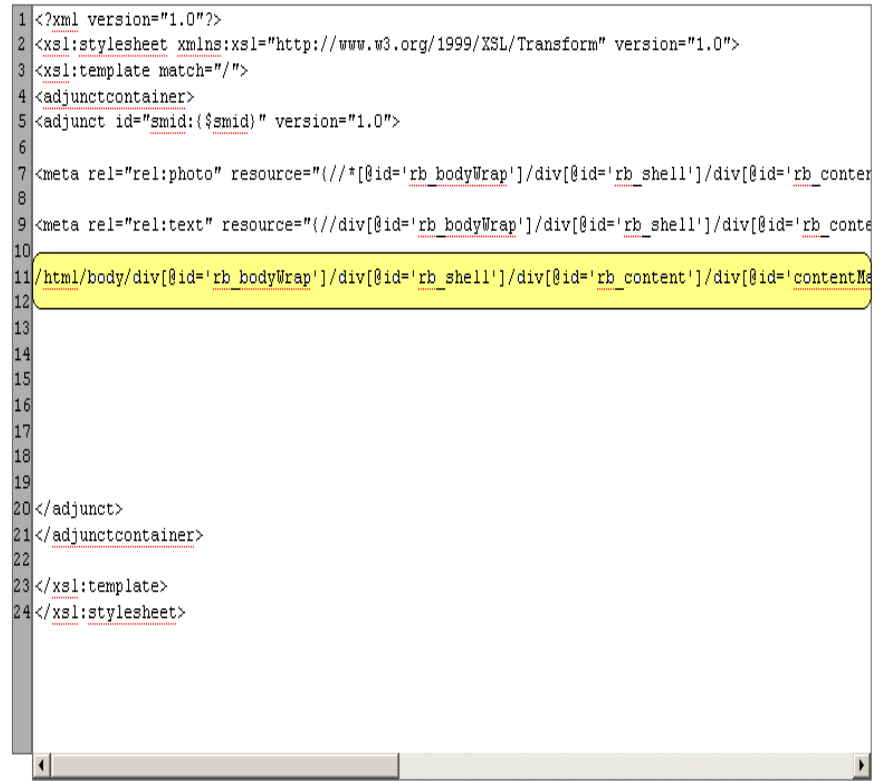
#### See more CNET content tagged:

[Trillian](#), [AOL Instant Messenger](#), [Cerulean Studios](#), [mIRC](#), [ICQ](#)

## Step 3: Data Extraction - Extracting a Link with XPath



- Copy the text from the XPath box (highlighted above in a LOVELY yellow).
- Paste that below the picture and text code in your Data Extraction page in the Search Monkey Application Dashboard so that it looks like this



NOTE: Clicking "Next Step" will take a while, as we load all your test URLs.

Done and done! Now you're ready for the next part...Tweaking the XPath

# Building your Custom Data Service

## Step 3: Data Extraction – Tweaking the **Link's** XPath

So, XPather gave you this:

```
/html/body/div[@id='rb_bodyWrap']/div[@id='rb_shell']  
/div[@id='rb_content']/div[@id='contentMain']/div[@id=  
'contentBody']/div[@id='dlNow']/a[2]
```

But you really want this:

```
<item rel="rel:link"  
resource="{//div[@id='rb_bodyWrap']/div[@id='rb_shell'  
]/div[@id='rb_content']/div[@id='contentMain']/div[@id=  
'contentBody']/div[@id='dlNow']/a[2]/@href}" />
```

What're ya gonna do?

# Building your Custom Data Service

## Step 3: Data Extraction – Tweaking the **Link's** XPath

- Trim off the first part of your code until you get to the first *div*.  
So, in this case, you're going to trim off `/html/body/` from the beginning of the line so that you're left with:

```
div[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='dlNow']/a[2]
```

- Enclose your code with `resource="{//` at the beginning, and `/@href}" />` at the end.

```
resource="{//div[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='dlNow']/a[2] /@href}" />
```

NOTE: Because this is a link, you're adding an **@href** to this code. If this was a picture, you would add an **@src**, and you were grabbing text, you would add nothing.

- Give your data resource a name by adding `<item rel="rel:link"` and a *SPACE* to the beginning of the line:

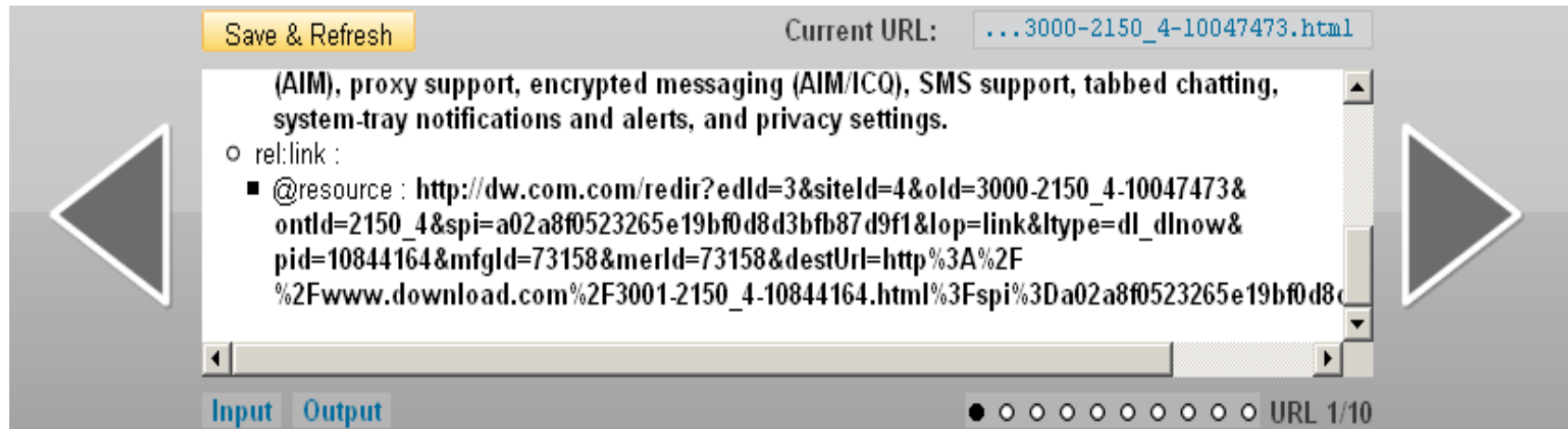
```
<item rel="rel:link" resource="{//div[@id='rb_bodyWrap']/div[@id='rb_shell']/div[@id='rb_content']/div[@id='contentMain']/div[@id='contentBody']/div[@id='dlNow']/a[2]/@href}" />
```

NOTE: Because this is a link, you're tagging it with **<item rel** as opposed to **<meta rel** which you would use if this was a picture or text.

# Building your Custom Data Service

## Step 3: Data Extraction – Tweaking the Link's XPath

**4. At last, click Save & Refresh, and now that lower panel should look something like this:**



You're done! Go ahead and click on "Next Step" at the bottom of the dashboard.

# Building your Custom Data Service

## Step 4: **Confirmation**

The hardest part is over. Now that you have extracted data you can arrange it into a SearchMonkey App by **Creating a Presentation Application**

To get started, click on the "Create a new Presentation Application" link from the confirmation page.

Note: You can also create the Presentation App at a later date by going to the [Developer Tool](#) Home page, clicking on "Create an App" and choosing "Create a new Application" under Presentation Applications.