

Web 2.0

- Web 2.0
 - Discover what is behind the hype and how to develop Web 2.0 applications.
- By Eric van der Vlist (vdv@dyomedea.com)
- For ATHENS 2006
- November 2006

Goal

“what is behind the hype and how to develop Web 2.0 applications”... Sounds like a huge agenda!



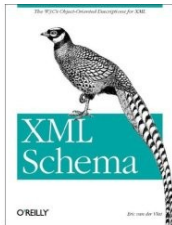
- What's behind the hype
 - As an introduction we'll see what is called Web 2.0.
- How to develop
 - We'll walk through a simple but real Web 2.0 sample application

About the author

And why should I trust you to give this talk?



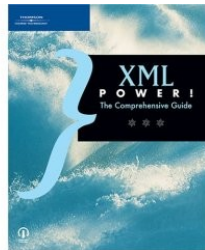
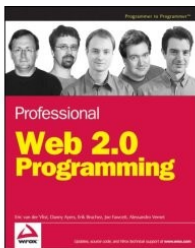
- I am a well known XML & Web expert and trainer.



I am the author (or co-author) of:



- XML Schema (O'Reilly, 2002)
- RELAX NG (O'Reilly, 2003)
- Schematron (O'Reilly shortcut, 2006)
- XML Power (Thompson, 2006)



What is Web 2.0?

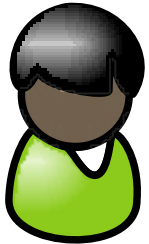
I see so many definition of Web 2.0 that I wonder if Web 2.0 isn't only a buzzword for anything new and cool on the Web!



- Frequent Web 2.0 definitions include:
 - A read/write (aka “social”) Web.
 - A ubiquitous and self sufficient Web (“the Web as a platform”).
 - Fluid Web applications with a bunch of XML and Javascript (“Ajax”).

Which Web 2.0?

Yeah, that's exactly what I meant! Which definition is the right one?



- They are interdependent
 - “Web as a platform” and “Ajax” are needed to make the Web writable again

A writable Web

Why should the Web be writable?



- Two main motivations
 - Social (some people want to share, see Wikipedia)
 - Business (Web owners have understood that it was more cost effective when users create the content)

The Web 2.0 business model

What's the business model for that?



- The dominant business model for Web 2.0 is advertising. See Google:
 - Google's business isn't search services but advertising.
 - One of the media on which Google sells adverts are their own search engine pages

Data lock-in

That's fragile! They loose their media as soon as people move!



- Web 2.0 sites are tempted to lock their users' data to keep them.
 - Data lock-in can be seen as the next era after hardware lock-in and software lock-in.

The Web is the platform

What's the link with the Web as a platform?



- hardware lock-in == the hardware is the platform
- software lock-in == the software is the platform
- data lock-in == the Web is the platform

Writable again

You said that Web 2.0 was writable again?
Isn't that a new concept?



- The very first Web (Web 0.1?) was writable:
 - Invented by scientist to share documents
 - Using a simple document format
 - Used by a community ready to hand edit and publish documents

A read only Web

What went wrong? Why did the Web become read only?



- The Web became read only for two reasons:
 - The technologies became more complex
 - Its community moved beyond tech savvy users.

Making the Web writable again

And how does Web 2.0 make the Web writable again?



- Web 2.0 is made writable again by adding technologies:
 - Ajax (Asynchronous JavaScript and XML) and the other technologies used by Web 2.0 make it easier to edit content.

The Web 2.0 technology stack

What are the technologies used by Web 2.0 applications?



- The common Web 2.0 technology stack is:
 - HTTP (and REST)
 - (X)HTML
 - CSS
 - JavaScript

The web as it was meant to be

You must be kidding! What's new in these technologies?



- Web 2.0 is about making user's experience better with using old technologies.
- That's why:
 - Some say that it is the Web as it was meant to be.

Is there a Web 2.0?

Wao! Even Sir Tim Berners-Lee says that there is no such thing as Web 2.0! What do **you** think?



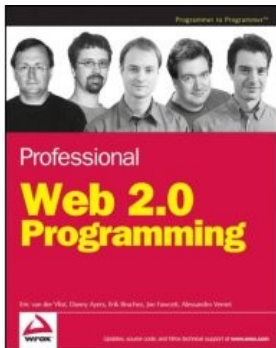
- There is no denying that:
 - The Web is changing
 - A new category of Web applications make the Web writable, use the Web as a platform and good old Web technologies differently.
- “Web 2.0” is the name for these applications

Introducing BuzzWatch

Enough theory, what is this sample application of yours?



- Sample application created for Chapter 1 of “Professional Web 2.0 Programming” (Wrox, ISBN 0470087889).



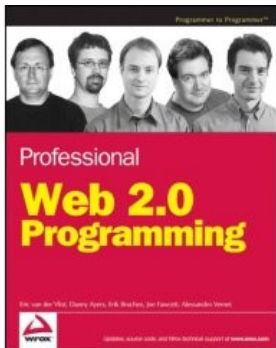
- A “mashup” that aggregates a number of informations from Yahoo! Finance and delicious ([Demo!](#)).

BuzzWatch's landscape

Kewl! What is BuzzWatch based on?



- Server side: PHP5 (with SQLite and Cache_Lite), libxml and libxslt (versions 3 and 4).
- Client side: XHTML, CSS, JavaScript, Yahoo! User Interface, JKL.ParseXML and Sarisa (versions 3 and 4).



BuzzWatch's four versions

“versions 3 and 4”, what are these versions?



- If time permit, you will see 4 versions:
 - v1: naive Web 2.0 implementation
 - v2: more web friendly
 - v3: less code duplication
 - v4: with cool URIs

Behind the scene

Let's be naive, then! How does that work?



- To understand how it works, we'll look at the web server's log and analyze the TCP/IP network exchanges.
- Try it:
 - Load [v1's entry page](#)
 - Look at the server log [[01-log.txt](#)]

First HTTP exchange

Waaa! No user action yet (except the initial load) and already a bunch of traffic! Where does that

come from?



- All this traffic is initiated from the first HTTP exchange (first line in the log).

See:

- The HTTP request [[02-http.txt](#)]
- The HTTP answer [[03-http.txt](#)]

JavaScript in action

Hold on, that doesn't explain the last access:
“12:37:00 200 GET /buzzwatch/watch.php
(application/javascript)”



- Right, this is the first Ajax exchange...
 - To see the effect of the scripts in this page, disable JavaScript and reload [v1's entry page](#).
- JavaScript takes control through [\[04.js\]](#) in [script.js](#).

JavaScript makes a huge difference in this page!
Can you shown for instance, how to create a
menu bar?



- This is typical YUI programming:
 - The menu bar is defined in [[05-html.txt](#)].
 - An rendered through [[06.js](#)] in [menuBar.js](#).

Ajax

Isn't it high time to show us this first Ajax exchange? Show us how the “dummy” menu item is implemented



- The Ajax exchange is triggered by [07.js] in the initialization of controller.js with this definition of loadList() : [08.js]
 - this is the first Ajax call
 - it uses a YUI wrapper around the

Ajax callback

What's that callback?



- Callbacks are needed because we're using asynchronous interactions with the server.
 - This one is defined by [\[09.js\]](#) in [controller.js](#).

HTTP request

So, that's the call to “YAHOO.util.Connect.asyncRequest” that triggers the last HTTP exchange that we saw in the



- The last line (12:37:00 200 GET /buzzwatch/watch.php (application/xml)) was triggered by this call
 - It represents this HTTP request: [[10-http.txt](#)].

Server side

And what does this request trigger on the server.



- This request executes [11.php] in **watch.php**.
 - This is a “GET” with no query string and listAll () [12.php] is executed.
 - Which, in turn, calls displayOne() [13.php]
- See how we use PHP5's simpleXML

HTTP answer

What's the result when these PHP instructions are executed?



- **This XML document** is returned by the HTTP response [**14-http.txt**].

Callback

And then, the JavaScript callback is called on the client...



- When everything goes well, `handleListSuccess()` [15.js] is called in `controller.js`.
 - `getMaxAge()` [16.js] uses HTTP cache headers to avoid duplicating this info in XML
 - `xotree.parseDOM()` converts XML into a JavaScript structure.

XML Data Binding

Hmmm... From PHP SimpleXML to JavaScript xotree, we're using XML to transfer data structures!



- 3 categories of interoperable tools to hide XML and focus on structures:
 - Integrated in the language (JS/E4X, C#, ...)
 - Dynamic for dynamic languages (PHP, JS, Python, Ruby, ...)
 - Static for other languages (Java, C#, ...)

Eric van der Vlist (vvd@dyxmedia.com) -- Web 2.0

Next exchanges

We've seen what's happening when you first load the page. What if we click on one of these watches in the menu?



- The server's log shows another burst of traffic [[17-http.txt](#)].
- These exchanges follow the same Ajax pattern but there are some interesting points to note.

SQL injection

OK, let's see the first one...



- The HTTP request to `watch.php` [18-[http.txt](#)] now includes a query string.
- The action in `watch.php` is handled by `readOne()` [19-[phps](#)]
 - What if, instead of “`name=goog`”, the query was “

Eric van der Vlist (vov@[biomedea.com](#)) `name=goog';%20delete%20from%20watches;select` -- Web 2.0 --

Caching

Frightening! We've not seen PHP Cache_Lite in action yet... What is it used for?



- Accesses to external services (Yahoo! Finance, delicious, ...) need to be cached for a number of reasons:
 - The terms of use of these services often requires caching.
 - You can transform what you cache.
 - Browsers wouldn't allow your scripts to access them directly.

Same Origin Policy

Hold on, why would a browser refuse to access external services from my scripts? Isn't it what



- Would be a huge security breach (scripts could retrieve and send sensitive information).
 - Scripts can only access resources from the same domain “Same Origin Policy”.
- Service providers wanting to allow browsers to access their services directly

Caching in PHP

OK, so we need to cache... How do we do that?



- This is done in [cache.inc](#) which relies on the Cache_Lite PHP Pear module.
 - [delicious.php](#) uses that straightaway
 - [yahoo_quotes.php](#) converts CVS into XML
 - [yahoo_chart.php](#) shows that images can be cached too.

Saving documents (client side)

So far, we've only seen how data can be retrieved from the server but you said Web 2.0 is about making the web writable again... How do you save



- Documents are saved using a HTTP POST method
 - Logged as [[20-log.txt](#)].
 - initiated by [[21.js](#)] in [controller.js](#) (note how the JKL.ParseXML library is used).
 - and sending this request: [[22-http.txt](#)].

Saving documents (server side)

And server side?



- This is done in the last function [[23.php](#)] that we've not seen in [watch.php](#).
 - The XML is validated using a [RELAX NG schema \(compact syntax\)](#).
 - Values are escaped (again) as a defense against SQL injection.

BuzzWatch v1 is not a good Web citizen

BuzzWatch is cool, but which URL can I send to share the Microsoft watch? And how will Google index these watches?



- V1 is a naive implementation which is:
 - Not accessible (requires JavaScript)
 - Not conform to the Web architecture (its resources have no URIs).
- This is why we need BuzzWatch v2!

BuzzWatch v2

Yeah, I agree, but I wouldn't want to lose the fluid user experience... How can you fix that in a Web 2.0 application?



- That's simple... We will:
 - give its own URI to each watch, such as <http://web2.0thebook.com/.../?name=ibm>
 - Populate these pages so that they have content that doesn't depend on JavaScript
 - Reload pages when users switch watches
 - Keep the other Ajax based features

index.html becomes index.php

Sounds sensible, and in practice?



- v1's index.html becomes v2's [index.php](#)
 - Its body [[24.phps](#)] uses v1's index.html as a template
 - populateQuotes() [[25.phps](#)]. Note the similarity with its JS equivalent [[26.js](#)].
- Also required: PHP refactoring and JavaScript adaptations to reload pages.

Code redundancy

Waaa... You mean that you have to code the same treatments in PHP and JavaScript? That's a maintenance nightmare!



- With its good web citizenship principles, BuzzWatch v2 performs the same treatments server side and client side.
- 2 solutions to fix that:
 - Use a common language for clients and servers
 - Avoid the problem by shipping (X)HTML fragments.

AHAH

Ajax without XML? Is that still XML?



- Alternative formats to XML include:
 - JSON (less attractive if you use data binding tools)
 - (X)HTML (Asynchronous HTML and HTTP)
- Using (X)HTML makes it more difficult to reuse the services exposed by BuzzWatch.

Common languages

A common language between clients and servers, sounds like a plan, but is there such a beast?



- Some options:
 - JavaScript: lack of traction server side
 - Other scripting languages: need to be installed on the clients
 - Java applets: heavyweight
 - Why not XSLT?

Client side XSLT

Client side XSLT? Are we ready yet?



- Well supported through `<?xml-stylesheet?>` in IE 6+, Firefox, Opera and Safari.
- Supported through scripts in IE 6+, Firefox and Opera.
 - Sarissa provides a wrapper class to hide differences between these browsers

Client side XSLT

Client side XSLT? Are we ready yet?



- Well supported through `<?xml-stylesheet?>` in IE 6+, Firefox, Opera and Safari.
- Supported through scripts in IE 6+, Firefox and Opera.
 - Sarissa provides a wrapper class to hide differences between these browsers

v3: XSLT in action

And how would you put XSLT in action?



- BuzzWatch v3 uses a common XSLT transformation:
 - Server side to create the pages.
 - Client side to convert the XML received by Ajax requests into (X)HTML

Server side XSLT

Let's see that server side first...



- [index.php](#) is rewritten to
 - Create [[27.php](#)] a XML document[[29-xml.txt](#)] with all the information needed to build a page using a common function [[28.php](#)].
 - Transform this page [[30.php](#)] into XHTML using PHP 5's libxslt library.

Client side XSLT

Easy... and client side?



- The XSLT transformation is loaded (using Ajax) and parsed [[31.js](#)] once in [controller.js](#)
- And used in the Ajax callback functions, that can be made generic [[32.js](#)].

The XSLT transformation

And the XSLT transformation?



- The **XSLT transformation** keeps the same “templating” approach than we had in index.php v2.
 - For example, this template [[33-xslt.txt](#)] replaces an `img/@src` attribute with the right cached Yahoo! chart URI.
- More on this approach at

<http://www.xml.com/pub/a/2000/07/26/xslt/xsltstyle>.

Not there yet

This v3 seems right to me, why would we need a v4?



- The URIs manipulated by BuzzWatch v3 are like:
 - <http://web2.0thebook.com/buzzwatch/v3/?name=goog>
 - <http://web2.0thebook.com/buzzwatch/v3/delicious.php?tag=google>
 - http://web2.0thebook.com/buzzwatch/v3/yahoo_chart.php?tag=ms
- And that doesn't look right!

Future-Proofing Your URIs

I see this kind of URIs all day long! What's wrong with them?



- Cool URI don't change...
 - These ones expose the technology that powers the web site (.php)
 - They also expose the repartition of features between PHP scripts
- Also, they don't show the links between different resources for a same name or tag

Better URIs

How should they look like, then?



- <http://web2.0thebook.com/buzzwatch/v3/?name=goog> -->
 - <http://web2.0thebook.com/buzzwatch/v4/watch/goog/>
- <http://web2.0thebook.com/buzzwatch/v3/delicious.php?tag=g>
-->
 - <http://web2.0thebook.com/...h/v4/tag/google/delicious.xml>
- http://web2.0thebook.com/buzzwatch/v3/yahoo_chart.php?tag
-->
 - <http://.../v4/company/msft/yahoo/finance/chart.png>

URL Rewriting

Waaa... and how would you do that



- URIs can be decoupled from the actual server's implementation using URL rewriting
 - Available as `mod_rewrite` in Apache
 - Available as add-ons in IIS
- On Apache, URL rewriting can be configured in `.htaccess`

Final tweaks

Is that all what's needed to make BuzzWatch v4 work?



- Relative resource URIs have been hardcoded in BuzzWatch's PHP, JavaScript, XSLT and XHTML files.
 - These files need to be updated to use the new URIs
- After that, we have a pretty sane Web 2.0 application that behave like a good Web citizen!

Developing Web 2.0 applications

Does that mean that I'll need to be fluent in (X)HTML, JavaScript, PHP and XSLT to develop my Web 2.0 applications and develop all that stuff



- Don't forget CSS, SQL and mod_rewrite that have also been used by BuzzWatch!
- This is why:
 - You need to buy our book :-)
 - Tools and frameworks are being developed

Web 2.0 frameworks

Yeah, I am sure I need one of these magic frameworks that will hide all these details...



- Their goal is to provide a “single view” approach for developing Web 2.0 applications
 - less redundancy
 - less languages to learn and use

Web 2.0 on Rails

Isn't it what Ruby on Rails is about?



- Examples of Web 2.0 frameworks include:
 - Ruby on Rails
 - Python frameworks such as Django, CherryPy, Pylons, ...
 - J2EE with JSF or Google Web Toolkit
 - Many others...

Declarative Web 2.0

They all seem to want to develop Web 2.0 applications like good all pre-Web client/server applications!



- Another approach is beginning to appear.
 - Based on a declarative approach (say what you want to get instead of describing how the result should be achieved).
 - XForms can be an alternative.

Powered by XForms

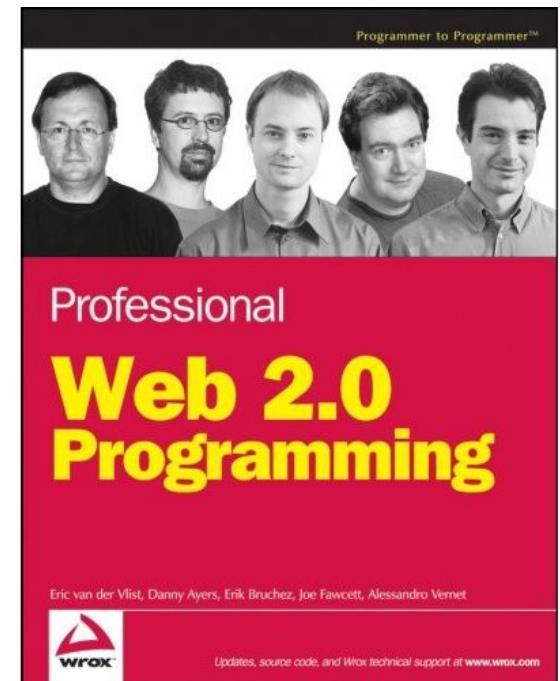
XForms? I thought that this W3C client Web forms technology was dead born!



- Client/server XForms implementation are getting mature
 - They generate Ajax/XHTML pages from XForms description
- See [Orbeon Forms](#) and [Chiba](#).

Thanks

- This tutorial heavily borrows from “Professional Web 2.0 Programming” (ISBN: 0470087889) published by WROX



- The code source can be downloaded from

<http://www.wrox.com/>

Eric van der Vlist (vdv@dyomedea.com)

([BuzzWatch](#) is presented in

--Web 2.0 --