Comprehensive Guide to a Professional Blog Site: A WordPress Example

by Michael K. Bergman

September 2005

About Mike

*Michael K. Bergman is a co-founder, chief technology officer, and chairman of BrightPlanet Corporation. He is also the lead project coordinator for the **DiDia** (document intelligence, document information automation) open source project. Mike's software management experience includes products in Internet search tools, data warehousing, bioinformatics, accounting, finance, management, planning, electronic mapping and technical areas. He is the author of an award-winning Internet search tutorial and the definitive deep Web white paper, among other publications. He is acknowledged as an expert in search, information theory, and document content. Mike's current musings may be found on his Web blog site, http://mkbergman.com.*

SUMMARY

Gone beyond Blogger? Want to really be aggressive in functionality and scope of content for your personal, professional or corporate blog? If so, this *Comprehensive Guide to a Professsional Blog Site* may be useful to you.

This *Guide* is the result of 350 hrs of learning and experimentation to test the boundaries of blog functionality, scope and capabilities. I myself began this process as a total newbie about six months ago – which likely shows in gaps and naïveté – but I have been aggressive in documenting as I have gone. The learning from my professional blog journey, still ongoing, is reflected in these pages.

This *Guide* addresses about 100 individual "how to" blogging topics and lessons, all geared to the content-focused and not occasional blogger. More than 140 citations, 80 of which are unique, are provided to other experts with guidance for all of us. The *Guide* itself occupies 80 pages. It is all free.

Is it all good stuff? Of course not! But there is hopefully more than one pony under the pile for those needing to join the "1% club" of purposeful, content-oriented, professional bloggers. In this *Guide* you will find discussion of these useful topics:

- How to choose blogging software and add-on tools
- Taking control of the blogging process by hosting your own site
- Getting your blog to display and perform right
- Effective techniques for converting existing documents to your blog site HTML
- Being efficient in posting, organizing and work-flowing to allow your diarist activities to flow naturally and productively
- Keeping the blog site pump primed with fresh and relevant content.

I created this *Guide* as a discipline in learning how to be a diarist or journalist, akin to the heyday era of "persons of letters" prior to the telegraph. In part, I undertook this discipline to rekindle those daily journal skills of the past. But, for the most part, I undertook the effort because I believe a fundamentally new means and mechanism for adaptive advantage is being created with social computing, of which blogging is a part.

This document is not meant for a single sitting; it is a reference. I hope my learning from capturing the new-to-blogging experience benefits you as well.

Michael K. Bergman

Iowa City, IA
September 2005

Table of Contents

List of Tables

List of Figures

I. INTRODUCTION

This guide is the result of a journal kept over four months documenting my own experience in creating an "industrial strength" blog.

In late April 2005, I came across a reference from Technorati that there were more than 10 million blogs currently in existence, with growth doubling every five months. A later reference from BlogPulse suggested that the number of blogs was closer to 11 million![1]

The phenomenal growth rate of this medium convinced me to investigate setting up my own blog and understanding better what this new medium was all about. This guide captures my learning (with thanks and citations to many!) in that process. While heavily oriented to the WordPress blog software, it hopefully offers useful guidance to other blog environments.

I began in April 2005 in earnest the process of learning and putting in place the systems that would make me an effective blogger. Within a short period of time I realized the lessons deserved keeping in a journal, and then I realized that at the conclusion of creating this journal that the entries could be "re-factored" into a complete guide. This *Guide* is the result of that process.

Because of work demands and other delays, the actual site was not released until mid-July 2005. This *Guide* was not completed until September 2005.

Throughout I refer to my own blog site, mkbergman.com, which is also designated as **AI3** for *Adaptive Information*, *Adaptive Innovation* and *Adaptive Infrastructure*.

In setting up my own blog, I had objectives in mind that you may not share. For example, I wanted to:

- Understand the blogging and self-publishing phenomenon
- Get my hands dirty with respect to existing tools and infrastructure
- Actually put in place a procedure that will allow me to continue to contribute in an efficient way
- Be aggressive about capabilities and understand "gaps" for bloggers (esp. the "top 1%") in moving forward, and
- Test whether major software pieces supporting a comprehensive blog play "nicely" together or not.

These objectives certainly led to a comprehensive investigation of the mechanisms and software for blogging. Here is what I learned. Enjoy!

[1] By the time of publication of this *Guide* on Sept. 19, 2005, Technorati was now claiming 17.3 million blogs tracked.

II. OTHER BLOGGING GUIDES

I remember the early days (1994 or thereabouts for me, at least) of Internet search and search engines. In fact, according to a History of Search Engines, when Lycos first went public in 1994, its index had something like only 54,000 total documents. In those days, when someone posted a "how to" guide it became prominent and likely was pretty intelligent, too.

I contrast that now to a recent task I set for myself: Find the best "how to" guides for setting up a purposeful, content-driven blog. In other words, find the best analogs to this current guide I am preparing.

There is a shortage of free "how to" guides to blogging that emphasize software, mechanics and infrastructure.

I can first report that the number of such items has increased greatly from the days of the early Internet — we're now dealing in thousands or tens or thousands of "how to" postings, not simply a handful. Second, I can report most all is dross and self-serving. I guess we all know what the astrological symbol looks like representing the Age of Aquarius; I wonder what the symbols for the Age of Spam or the Age of Hype should look like?

These issues are emblematic of some broader problems. Namely, when everything is online, when everyone is a publisher, when many can figure out how to jimmy the system, how can one efficiently find and distill the best? Hmmm. This is not so easy ….

I pride myself on being a "1%"-er in terms of search and information discovery. Indeed, for many years I can proudly say my search tutorial[2] was often deemed the best on the Web, or at least very good. Sure, I continue to learn techniques and elsewhere on the mkbergman.com blog I frequently sum them up or re-cap them. But my experience in trying to find the definitive "how to" guides for professional, content-oriented blogs makes me feel pretty humble.

I've tried many of my tricks to find definitive blogging guides (with Yahoo search counts shown for each):

- Obviously, the best is phrase searching, with "bloggers guide" (9,300), "blogger's guide" (7,320), "blogging guide" (19,400), "how to" blogging (14,400,000), etc,. and many variants showing some promise, but ultimately the yield is poor with too many results
- I have learned that it is often useful to restrict such searches by file type (esp. PDF) to cull out the most serious postings (yes, you miss some great stuff, but the premise is that taking the effort to create a PDF also signals seriousness of the content). To take our lowest count from the example

[2]M.K. Bergman, *Search Tutorial: A Guide to Effective Searching of the Internet*, published by BrightPlanet Corporation, July 2000 (updated December 2004), 68 pp. See http://www.brightplanet.com/technology/SearchGuide.asp

above, adding the restriction of PDFs for "bloggers guide" numbers about two results. Somehow, I went from too many to too few, since my intent was to assemble a vetted list of 5-15 or so "best" guides

- Even when applying these same techniques to the big results of the "how to" blogging query above limited to PDFs, the results set was still 9,340 hits. Though, again, some useful links were found, the yields were low
- Another technique is to seek review or compilation sites, using those terms. There are ranking qualifiers ("top" "best" "100", etc.) that can be used for this purpose
- A further technique is to search on qualifying terms that often point to guides or tutorials. Example terms are guide, tutorial, beginners, basics, 101, etc., and their plural variants
- Additional narrowing of results can come from using terms such as professional, business or corporate, since my interest was in more-or-less these types of blogs, and
- Still another technique is to string together a few terms-of-art (read, nouns), that specifically and collectively pertain to the topic at hand. In the case of our blog topic, such terms could include blog, WordPress, blogger, trackback, ping, RSS, post, "blog roll," etc., etc.

It is becoming increasing difficult to filter out the excellent from the dross when using search engines.

Despite these tricks, and some others not enumerated, I was singularly unable to find a "high yield" set of search results pointing me to what I wanted. Granted, each search, esp. the refined ones, surfaced some possible gems or OK results, but the general yield was poor.

Why is this?

Well, certainly, one reason is the growth of blogging. Having more blogs online means much larger numbers of postings and more difficulty in finding the gems. Another reason is that the sheer growth and popularity of blogs attracts those interested in making money. For example, Build a Better Blog System is a pretty comprehensive guide but its insights come at a price ($49). Other sites use "how to" come ons for drawing visitors into their marketing, advertising or PR services. Finally, still another reason is that some of the better guides are specific to a specific hosted service such as Blogger or specific blogging software such as WordPress or Movable Type.

Nonetheless, these investigations did produce a few additional guide references:

- The Executive Blogger's Guide to Building a Nest of Blogs, Wikis & RSS, an entertaining guide from Ogilvy PR Worldwide
- Time to Check: Are You Using the Right Blogging Tool? covers blog software comparison chart, plus pretty good glossary
- How To Blog 101, a pretty comprehensive "How To" blog site

- Beginners' Guide to Corporate Blogging is fairly short and elementary
- Beginner's Guide to Business Blogging, which is a 41-slide presentation
- Google's Blogger Guide, useful for those wanting a quick blog set-up
- A good practical guide that emphasizes thought and planning in launching a blog is by Stephen Downes, How to be Heard. He offers good advice on discovering ***why*** you are blogging, design and marketing and awareness conditions, etc. Stephen is the veteran of the introduction of a few blogs, and his experience shows in this short and easily read guide.

Nonetheless, the review of these guides continued to suggest to me the worth in pulling together a comprehensive guide such as this one.

III. APPROACH AND DEVELOPMENT

I write a lot of material, generally business and technical stuff for clients. My approach for years has been to sketch out an outline, do research, refine the outline, and bomb away.

This document and the entire blogging process in general have led me to try a different approach. Namely, I develop small "chunk-size" topics that can be posted as standalones, then when done, I re-combine (or "re-factor") into a more comprehensive treatment.

I have to admit I'm still getting used to this new approach. I am convinced it is a more efficient way to combine feeding a blog and standard writing, and I'm intrigued with its closer resemblance to the journal writing of centuries past, but it is a different approach with which I am still trying to get comfortable.

Re-factoring of My Preparing to Blog Journal Diary

Blogging may cause a shift in writing styles to "document as you go" and the journal/diary style of centuries past.

This guide is the combination of more than 30 separate posts on the mkbergman.com blog site, produced over a period of about four months. These topics, and their actual dates of posting in 2005 are:

- The Purposeful Blogger
- First Test Drive – *April 28*
- WordPress – *April 29*
- Local Hosting – *May 2*
- Install Difficulties – *May 6*
- Design and Hacking CSS – *May 6*
- No Local Images – *May 7*
- Posts and Comments – *May 8*
- Advanced Functionality – *May 9*
- Site Transfer – *May 17*
- Begin Content – *May 18*
- Release Checklist – *May 20*
- Editor Comparisons – *May 27*
- Xinha Integration – *June 1*
- External Credits and Thanks – *June 13*
- Permalink Problems – *June 15*
- Word Docs to HTML I – *June 16*
- Word Docs to HTML II – *June 27*

This Guide is a combination of more than 30 posts over a four-month period.

- Site Project Management – *June 17*
- Not Playing Nice in the Sandbox – *June 19*
- Use of Styles and Stylesheets – *June 20*
- Some Best Practices – *June 22*
- Cross-browser Compatibility – *June 24*
- File Organization and Naming – *June 25*
- XHTML Validation – *June 26*
- Screen Resolution – *July 5*
- Trackback and Ping Testing – *July 12*
- Better Comment Quicktags – *July 14*
- Time and Effort Estimates – *July 15*
- Actual Site Release – *July 18*
- Keeping the Blog Alive
- PDF Compendium – *September 19.*

Of course, in combining these into a single guide, redundant information was removed and new "glue" was added. Nonetheless, each of the original posts may be found under the Chronological Listing section of the **AI3** blog.

My Caveats

People, of course, blog for different reasons and with different frequencies. Some post for more social reasons, others track interesting developments in their domains of interest, still others aspire to provide original writings or comprehensive reference locations. For myself, I have more interest in research and analysis and longer postings.

As a result, my **AI3** site on Adaptive Information perhaps has longer content, which will require over time more organization and advanced search functionality. I furthermore wanted to have complete control over the site, so I installed my own software and did not use one of the many excellent hosting services. Finally, while a prolific writer for decades, I'm also a newbie to blogging.

This emphasis on an "industrial strength" blog for more professional or business users using the WordPress software may not apply to your own circumstance or needs. In such cases, this full *Guide* may be of limited interest for you and close perusal of the table of contents is recommended.

IV. PREPARING TO BLOG AND BASIC DECISIONS

Your most important blogging decision is whether to use a service or install and configure your own software.

As a way of getting my first taste of what a blog is, how difficult it is to set up, and how one maintains it, I first set up a test site on Blogger. The ease of setting up the site was remarkable! Truly, within five minutes of sign-up, I was editing and working with a blog site. It is clear why blogging has grown so fast, with reportedly 5 million of the sites alone being hosted by Blogger.

However, because my interests go more to scale and bleeding edge functionality, I decided to forego using a hosted site (though it appears to be the right answer for most users). Instead, I investigated and chose my own software package(s).

Hosted or Own Site?

In most instances, having a third-party host your blog is likely the best choice. Circumstances where you should consider hosting yourself, however, include the following:

- Your content, presentation or functional needs go beyond the "normal." A quick peruse of this guide should indicate whether you fit in this category
- You anticipate a very long life for your site, and have concerns that the hosting service may go out of business or you may want to change hosting software. Exports and conversions from one site to another or in changing software are notoriously difficult
- You will be introducing complicated functionality or content that you desire to be able to post and test in an "offline," private mode. Once you set up a local hosted system, it is easy to create a mirrored version on which you can do considerable offline testing and refinement
- You will be posting complicated HTML (tables, formatting) that require the use of an online editor, or
- You're a glutton for punishment.

However, even if you decide to host your own site, I recommend you begin with a quick set-up on Blogger first.

Picking Blog Software

Blog software packages more generally fall under the heading of content management systems (CMS) software. The first criterion I had for choosing an implementation package is that it must be open source, since I am not sure what my longstanding commitment will be to maintaining a blog site and I did not want to make a big financial outlay.

According to a Gilbane report and information on opensourceCMS, there are on the order of 80 open source CMS packages today, out of perhaps a total of 500 CMS packages including those for sale.

There are perhaps 500 different CMS packages (many useful or specific to blogging), about 80 of which are free, open source.

I looked at online demos for a number of the packages. (The opensourceCMS has nice online demos for about 40 systems.) The buzz around WordPress, however, was particularly strong. I was very much impressed with a rolling set of WordPress templates from a recent competition on Alex King's blog site. I highly recommend you use the template browser on this site to exemplify the presentation control available. I also found that the ease of adding plug-ins and the number available for WordPress would meet my near-term functional needs.

Though virtually all blog packages support varied templates and layouts, not all support the creation of standard site pages (such as copyright, about you, etc.) independent of the standard blog posting "loop" (see below). This aspect was also important to me (all links in my standard **AI3** navigation panel are static pages), another plus for WordPress.

Though I could have done a more detailed comparison, which is my normal style, my approach in this instance was to act more like a "standard" new blogger and only do as much investigation as necessary to make a defensible choice.

Local Hosting

Unfortunately, in order to test design and content for my blog-in-waiting, I needed to set up my local machine as a Web server and, therefore, obtain all of the necessary supporting software. Because my selection of WordPress as blog software was open source, I decided all other software components needed to be open source as well.

Installing your own localhost blog software requires many other packages to also be installed, configured, and made to work together.

Many open source packages go by the acronym LAMP (Linux-Apache-MySQL-PHP/Perl/Python). WordPress falls into this category, but it also has a WAMP configuration (Windows instead of Linux), which is my local machine's operating system. As a result, I downloaded the WordPress 1.5 WAMP package and prepared to do battle.

Though my employees have set up these environments for years, I had never done so. I consider myself only marginally competent in such matters, and therefore dangerous when around software and standing water simultaneously. Again, since my intent was to see how a "standard" individual would go about setting up the whole blog environment, I purposely chose not to ask any of my colleagues for help. If I couldn't find the guidance and references on the Web, I just wouldn't be able to proceed and do it.

After some searching, my first breakthrough in finding an installation guide came from Reza Baharin. Reza's Tutorial: Running a Personal Website on Apache HTTP Server on WinXP is specifically tailored to WordPress. Even with this guide, setting up a locally hosted system is not easy. Section VI deals in its entirety with this subject.

Name and Design

After the hosting decision, your next most important decisions relate to site design, scope and "look."

The next set of decisions regards the overall design, style and template for your Web site. Some of the preliminary decisions I made for my own site were:

- I will call the site 'Adaptive Information' [later updated to **AI3**]
- I will use a three-column presentation framework
- I will use my favorite Claremont (CA) HS Wolfpack color scheme (gray, maroon) as it has evolved over time (for me) with black, charcoal, bluebird blue and white.

I also decided to check on whether mkbergman.com was available as a domain name. I'm embarrassed to say that I had already reserved the name some years earlier, though obviously I had forgotten about it. However, in most cases, your name will not be available. Choosing a good Web domain name in this crowded era is very difficult. Make full use of whois services such as from register.com or betterwhois.com to test and try ideas.

Do You Need or Want a WYSIWYG Editor?

A fairly critical decision is whether you need a WYSIWYG HTML editor or not. For myself, I anticipated long posts with complicated formatting and tables, plus embedded images and cross-linkages. These requirements strongly indicated the need for a linked what-you-see-is-what-you-get editor.

Another important decision for me was a WYSIWYG editor for anticipated complicated HTML for some of my posts.

Quite of few of the topics in this guide deal with how to select and install such tools. See especially Appendix A.

Posts and Comments

Besides overall layout design and general color scheme, you should spend some time dealing with the presentation and formats of posts and comments, since they will be the bulk of the action on your site over time.

As with other topics related to design, it is very helpful to look at as many blog examples as you can and study how different authors and publishers have made their design decisions. Use of blog listing sites such as BlogPulse can help you find both popular blogs plus those in your specific domain areas. (They are also useful in setting up monitoring lists for tracking topics of interest.)

Advanced Functionality

In looking at other sites and thinking about my own, I knew that over time I wanted to move to such complicated functionality as integration with Wikis or integration with BrightPlanet's dynamic placement engine for automatic categorization of content.

Another important decision area is advanced functionality, often provided through so-called "plug-ins." (Also an important factor in host software evaluation.)

However, in the more immediate term, I also saw a need for utilities to do such things as list most popular posts, create a chronological listing of posts, provide alternative display options, and the like. These capabilities are often expressed as "plug-ins" to a baseline CMS or blogging package. Thus, one of the important aspects in chosing a proper hosting packagee is to investigate the variety of plug-ins available. Also, if you have your own site hosted locally, you can download and install some of these packages to see how they will actually work in your production setting.

Obviously, each blog package has its own plug-in listings. Some very useful ones for WordPress can be found at WordPress's own site, the non-affiliated WordPress Plugin DB, and the non-affiliated WordPress Plugin Directory. (Similar ones are available for other major blogging packages.) Check out listings such as these to get an appreciation for how basic blogging packages can be extended through these plug-ins. Also see Section VII below.

Word Docs to HTML

A last consideration that I personally had was the ease or process for converting existing MS Word documentation into possible blog postings. Since I do write and communicate so much with Word, a graceful path for getting double duty from this content was important to me.

Because of this importance, I actually spent considerable time developing techniques and procedures for doing so. If Word conversion is important to you, see Appendix B for step-by-step guidance using what I believe to be the current state-of-the-art tools and approaches.

V. BLOG BASICS 101

OK, so the first steps of thinking through the design and purpose of your blog site have been completed. It is also useful to understand some of the basic terminology and operation of blog software.

Your first exposure to blogging – as with anything else – introduces new terminology and concepts. This section helps you learn about the basics.

The Loop

The "loop" is used by blogging software to display individual posts and comments that might be submitted by others in response to those posts. The "loop" is thus the key engine within blogging software, geared solely to posts and response comments. Obviously, the loop generally displays posts in order of their posting chronologically (though even that can sometimes be controlled by other bases, depending on the software).

Within the loop there are tags and other format characteristics that tell the blog software what, how and with which styles to display these posts and comments. For example, most all blog software packages give you the choice about how many total posts or duration of days for posts will show on your current Web site. Typical content control would include whether or not titles, body, dates, respondent names, etc., etc., are included or not in your standard loop template.

Some blog software packages only allow content to be displayed within the loop; others provide separate static page content (such is the case with WordPress and most major packages; see above). I recommend software that mixes both loop and static pages.

Generally, the loop statements are found within the main software file within your blog package (such as index.php for WordPress), because the loop is so central to the dynamic operation of the software.

The Stylesheet

One of the beauties of modern CMS software is that presentation is separated from the content. Content is organized and presented dynamically by the CMS software; layout, color scheme and presentation is guided by the cascading style sheet (CSS). To see how this presentation layer can be easily and widely modified, see the style browser on Alex King's blog site. The entire Section VIII covers style sheets and display of your blog.

Plug-ins

Another beauty of modern CMS software is that the basic shell framework of the CMS and its associated stylesheets can be extended with additional "plug-in" functionality. Sometimes this functionality is provided by the original developer of the CMS package; most often, however, by outside third parties.

Open APIs (application programming interfaces), basic CMS package popularity, and a roster of useful existing plug-ins are positives you should look for in picking a package. Most of Section VII below deals with plug-ins or other additional functionality.

The Dashboard or Management Center

Most CMS or blog systems provide a password-protected administration section where new posts are entered or edited, comments reviewed and moderated, and other general settings. In WordPress, this center is called the dashboard and the example from my blog is shown below:

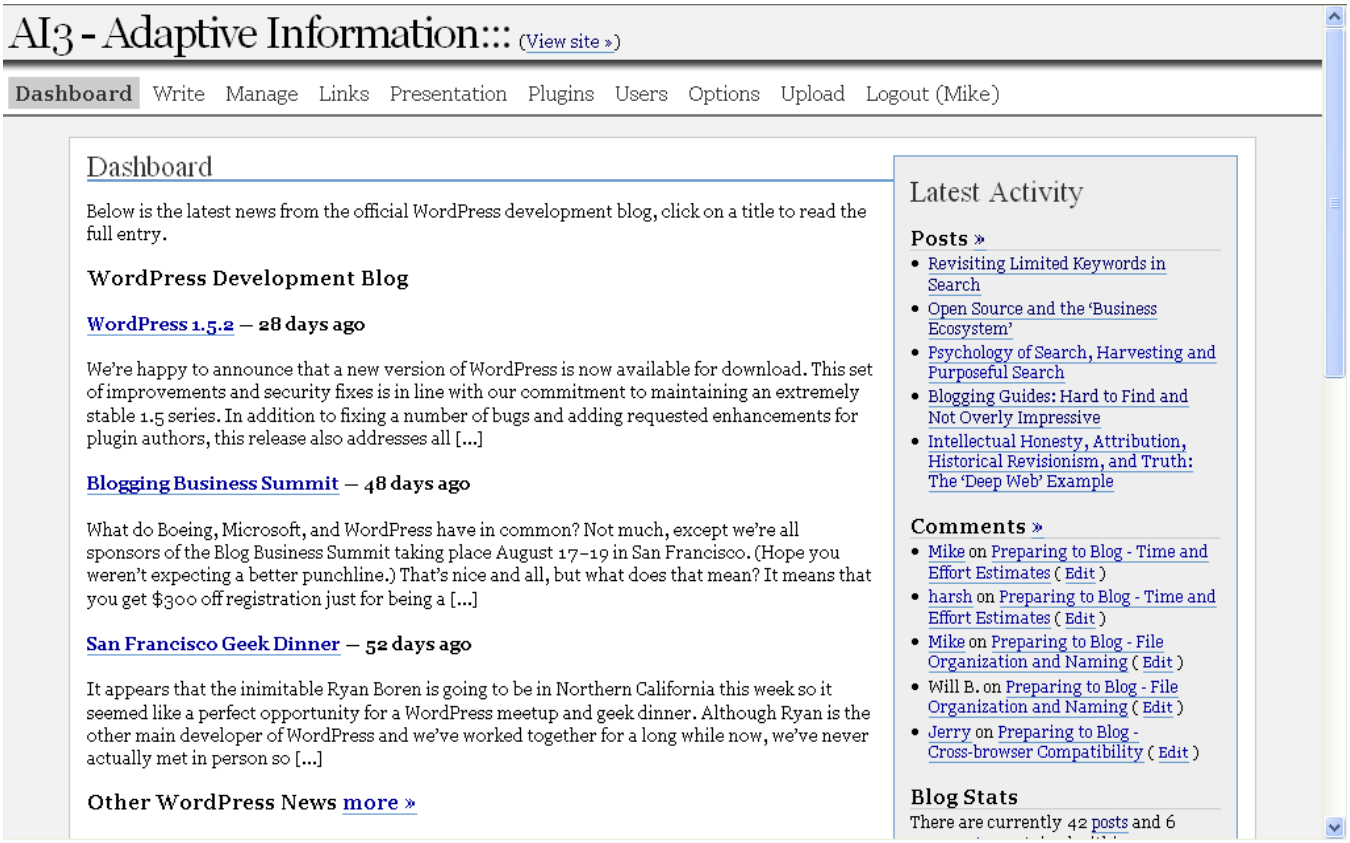


Figure 1. Example WordPress Dashboard

Pinging

Pinging is a powerful way to alert posting and blog searching services that you have posted a new entry. For example, there is a facility within the WordPress administration center for setting a ping list under Options-Writing-Update Services (see the dashboard menu above); the standard ping provided with the default installation is Pingomatic. Elliott Back has provided a useful starting list

of ping sites to include with WordPress. In reviewing his list, I looked at and decided to include these entries, excluding dead links and most foreign sites:

"Pinging" is an essential means for telling the external world that you have updated content on your site – give this area attention!

```
www.a2b.cc/setloc/bp.a2b
api.feedster.com/ping
api.moreover.com/ping
api.my.yahoo.com/rss/ping
www.blogdigger.com/RPC2
blogmatcher.com/u.php
www.blogshares.com/rpc.php
www.blogsnow.com/ping
www.blogstreet.com/xrbin/xmlrpc.cgi
coreblog.org/ping/
www.mod-pubsub.org/kn_apps/blogchatter/ping.php
www.newsisfree.com/xmlrpctest.php
ping.blo.gs/
ping.feedburner.com
ping.syndic8.com/xmlrpc.php
ping.weblogalot.com/rpc.php
www.popdex.com/addsite.php
rpc.blogrolling.com/pinger/
rpc.pingomatic.com/
rpc.technorati.com/rpc/ping
rpc.weblogs.com/RPC2
www.snipsnap.org/RPC2
topicexchange.com/RPC2
xping.pubsub.com/ping/
```

IMPORTANT NOTE: In WordPress 1.5 there have been many reported problems of slow updates of finally published posts, that worsens with longer posts, but is not seen during draft saves. Many causes have been speculated for this, but the current consensus on the WordPress support site is that the cause is due to many and slow-responding ping sites.

Until this problem is fixed, WordPress users may want to consider limiting their ping sites to a single meta-ping source such as Ping-o-matic or Pingoat.

Trackback

Trackback is a mechanism for a third-party to post on its own site a detailed response to one of your posts. (Or for you to do the same for a posting on another site.) When done, the third party site pings your site, provides an excerpt that displays in the standard comment field, and can then be viewed by URL reference from your comments list. Thus, the third party respondent need not post on both sites and more detailed responses can be provided on each author's respective site. Trackbacks can also be used for content aggregation purposes by topic.

There are many trackback overviews – which sometimes seem hard to understand because of its poor name – available on the Web. A couple that are useful include

the newbie guide from Moveable Type, the original developer of the trackback function and protocol, and a different short version..

Should Trackback be Used?

Initial difficulties in getting trackback configured for my site led me to question whether the function was even desirable. In fact, within the past year, there has been an explosion of spamming against trackback facilities. 'Trackback is Dead' is one of the more provocative discussions arguing that the time for trackback is past; Matthew Mullenweg, a founding developer of WordPress, has spent time looking at how spamming can be overcome.

Because WordPress 1.5 has what appears to be a pretty effective trackback moderating facility – the same as what is used for comments – and because there appear to be some trackback spam filters and other plug-in utilities, I decided to implement the feature for now, see if it is used, and if spam does occur deal with that problem at that time.

Permalinks

For example, rather than use the `?page_id=num` and `?p=num` internal references for postings, WordPress provides a permalink feature that converts these ID references into URL strings that help in search engine indexing. Here is the permalink structure I first tried for my site:

```
/index/ai3/%year%/%postname%/
```

This produces a URL that contains a truncation of the post name title, plus other relevant information. That is well and good and the search engines would love me, but turning this feature on caused: 1) images were lost due to reference changes; and 2) Jonathan Foucher's 'Popular Posts' plug-in ceased displaying. Resolving these issues is discussed in Section VII.

Blogrolls (or –rolling)

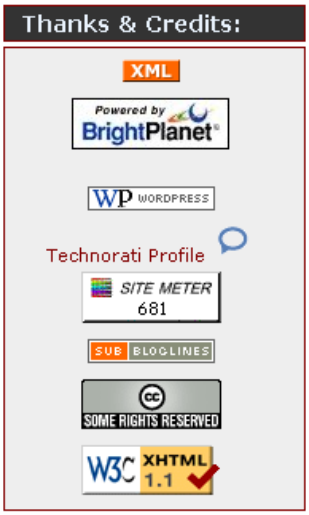
Blogrolls are simply listings of other blog sites that you frequently read or contain coverage of similar interests. I have not yet implementing this feature (though I am tracking them external to what appears on **AI3**). I also find the blogrolls on new interesting sites I discover to be a source for still further discovery. (Information discovery is discussed more in Section XII.)

External Credits and Thanks

Another step is to provide links to external parties that deserve credit in one manner or another. I call this section Thanks & Credits on the **AI3** site and it is created by modifying the various index pages for the site (requires a minor understanding of PHP (in this case) and HTML. I was very impressed with how

easy it was to add these systems; their Web sites and on-site instructions with accompanying HTML and Javascript (if used) were excellent. I set up links for:

- XML (RSS feed)
- BrightPlanet
- WordPress
- Technorati (contemplating dropping)
- Blogstreet (later dropped)
- Site Meter (a useful, recommended free service for site statistics)
- Creative Commons
- W3C XHTML Validation (added later).



Other sites list different credits and may use simple links instead of images.

VI. CREATING A LOCAL TESTBED

A local testbed for a blog site is useful either to work out the kinks before going live and/or because you intend to host the system anyway. Both of those conditions applied to **AI3**; if neither apply to you, you should skip this section.

Local Install Difficulties

Unfortunately, there is a scarcity of localhost installation guides, they often don't address your own specifics, and they are hard to find!

My initial installation guide came from Reza Baharin's Tutorial: Running a personal website on Apache HTTP server on WinXP, which is specifically tailored to WordPress. This guide is excellent and without it I'm sure I would still be pulling my hair out. While I did indeed have some installation difficulties, in truth most were self-inflicted and arose from being too impatient, inattentive or careless. But, these frustrations were real and I try to summarize them so that others may not make my same mistakes:

- My first mistake was not to use the vanilla directory structure and naming suggested in the guide. Rather, I installed everything in my standard, rather complicated, nested existing file system. That poor choice continued to give me fits as my paths were too long and deep and my naming conventions ('1 - MainA', '2 - MainB', etc.) were also difficult when I needed to go directly into DOS
- My second mistake was to not read the entire tutorial carefully and in advance. Because I was impatient I plunged into the first pages and found I was following directions at the top of the tutorial for restores, yet I had not even done the initial install. I ended up doing some tweaks and settings that I had no business worrying about
- My third mistake was not properly understanding network domain, server name, and computer name for the Apache (and later MySQL) installation. I kept getting confused between use of 'localhost' and actual names
- My fourth mistake was not knowing how to do relative addressing of directories in the configuration files, especially for PHP, which I had a trying time to get to load. Because of these difficulties, I did not know how to properly set my URLs in Mozilla to load the PHP test (phpinfo.php). Hint: make sure your URL address begins with localhost. In the guide's **Test PHP** section, refer to the URL address screenshot shown for **Test Perl** to get this reference correct
- And, finally, my last mistake was really not understanding the MySQL settings for 'host', 'user' and 'password'. The only solution I found was to delete the MySQL installation and re-install with `host=localhost, user=root and password="" [blank]`. If you run into similar problems, carefully note your error messages and use them as a basis for a Google search (see this one, for example).

However, despite all of this, late one night I had a Web server running on my local machine with a local copy of WordPress.

However, now that a local version was installed, I was finally able to see other problems and issues. The remainder of this section deals with some of those issues.

No Local Images

I spent an incredibly frustrating amount of time trying to figure out why I could not get images to display in WordPress. I ultimately realized that the internal link references needed to be like the local URL addresses in the browser. Localhost needs to be explicitly spelled out, and also in the reference position for where the root Web server directory resides. Thus, for my local installation, this becomes:

```
<img alt="This is My Photo" src="http://localhost/wordpress/wp-
content/themes/myTheme/images/myPhoto.jpg" />
```

This change to the relative addressing now displays images properly.

Site Transfer

Kevin Klawonn, BrightPlanet's sys admin, got the basic blog package I had set up on my local machine transferred and installed on commercial servers. His report on this transfer follows:

> *Like most sys admins out there, my plate is always full. So when I was asked to install WordPress I was hoping that it would be a simple installation taking less than 30 minutes to review the installation procedure, install the software, and then make the necessary configuration changes. Upon visiting the WordPress website, I saw the "5 Minute Installation" page and started laughing because my karma isn't good enough that I should actually be able to have WordPress installed in 5 minutes.*
>
> *I perused the requirements:*
>
> - *PHP v4.1 or greater. I had installed PHP before and this is a simple install*
> - *MySQL v3.23.23 or greater. I have MySQL installed on another server so this requirement was already met*
> - *Apache module mod_rewrite support. This was going to be the most difficult part of the installation.*
>
> *The server identified to host WordPress already has IIS running on it and therefore did not have mod_rewrite support built in. Instead of finding a 3rd party option for this requirement, I decided to install Apache 2.0 on the same*

server. To accomplish this, I needed to force IIS to bind only to the IP Addresses which it actually uses. By default IIS binds to all IP Addresses of the server whether they are used or not. A Microsoft support topic I had used before explains how to accomplish this task; in my previous experience it had no side effects so I felt comfortable doing the same thing in this situation.

Once done configuring IIS, I added another IP address to the server, opened port 80 in the firewall to the new IP Address and installed Apache 2.0 configuring it to use the new IP Address. I then installed PHP, and configured Apache to use PHP.

The final step before installing WordPress was to setup a database on MySQL. Although MySQL resided on a different server, this did not matter to the configuration of WordPress. I created a new database and user for WordPress to use.

Finally, I was ready to install the actual WordPress software. I uploaded the zip file to the webserver and unzipped it. I edited the wp-config-sample.php file per the "5 Minute Installation" instructions. Once the database information was configured, I opened my web browser and ran the `http://{website}/wp-admin/install.php` *script. And in just that quick the blog website was setup.*

Of course, not all of us have a sys admin at hand and therefore much of the foregoing discussion may sound like so much gobbledygook. At least it does to me.

I figured out how to install an instance on my local machine. Transitioning that to a commercial release, however, can involve many issues. Hosting your own blog site is not for the fainthearted or the inexperienced.

VII. ADDING AND CONFIGURING TOOLS

A brilliant design aspect of modern content management systems (CMS) is their ability to provide an open framework for plug-in tools and other extended functionality. Most CMS packages come with an application programming interface (API) that allows third parties to write functionality that will easily integrate and interoperate with the base CMS system.

"Add-ins" are often an essential consideration to extend the vanilla functionality of your blog software.

However, as with most code and open source in general, there is a wide variety in the quality of these tools. Some don't integrate well or require frustrating efforts to overcome glitches. Some fail or don't operate properly because of version mismatches between the add-in and the base system. Some integrate well but just don't have adequate or useful functionality.

But some does work well. And in finding and integrating the right tools, you can greatly enhance the functionality of your blog and the efficiency by which you create and maintain it.

Plug-ins

A plug-in is an external application that can easily be integrated into a base CMS system (or general software) and interoperate within it as if it were original embedded functionality. Though an external capability, it does not act like a call to an external program. Generally, plug-ins use base calls and syntax, and can piggyback on existing base functionality.

Most often plug-ins are written by committed users of the base product and are offered as open source with no support. As volunteer contributions, versions are often allowed to grow stale and documentation is most often totally lacking except for some minor comments in the header of the plug-in software code.

The better CMS packages, such as WordPress, not only offer published APIs for supporting functional integration, but also provide registration, activation and management facilities for plug-ins (through the dashboard in the WordPress case). Examples of this are shown below.

Because of the importance of plug-ins, all popular CMS packages have their own roster of plug-ins. As noted before, useful ones for WordPress can be found at WordPress's own site, the non-affiliated WordPress Plugin DB, and the non-affiliated WordPress Plugin Directory.

Example Code

Here is example header code for one of the **AI3** plug-ins, this one from Jonathan Foucher for creating a most popular topics listing:

```
<?php
/*
Plugin Name: Most Popular posts
Version: 0.12
Plugin URI: http://jfoucher.info
Description: Displays a list of the most viewed
plugins.
Author: Jonathan Foucher
Author URI: http://jfoucher.info/2004/10/04/teaser-
style-wordpress-plugin/
This plugin supports Automatic Update :
http://wiki.wordpress.org/AutomaticUpdate
Update: http://jfoucher.info/plugin-update.php?p=40

Copyright (c) 2004
Released under the GPL license
http://www.gnu.org/licenses/gpl.txt
    This program is distributed in the hope that it
    will be useful, but WITHOUT ANY WARRANTY; without
    even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General
    Public License along with this program; if not,
    write to the Free Software Foundation, Inc., 59
    Temple Place, Suite 330, Boston, MA  02111-1307 USA
*/

/************************************
** TO TAKE ADVANTAGE OF UPDATE **
** DETECTION, PLEASE CONSIDER **
** MAKING THE FOLLOWING ADDITION **
** TO YOUR WORDPRESS INSTALL **
** **
************************************

  OPEN FOR EDITING : wp-admin/plugins.php
  FIND
          if ( preg_match("|Version:(.*)|i",
            $plugin_data, $version) )
            $version = $version1?;
          else
            $version ='';
    ADD THE FOLLOWING CODE AFTERWARDS
etc.
etc.
```

The AI3 Plug-ins

Note that the header file above follows a set format for specifying plug-in name, author, version number, etc. Thus, when a new plug-in file is copied to a specified WordPress installation subdirectory, the program will register it, and allow it to be managed via the dashboard, as the **AI3** example shows below:

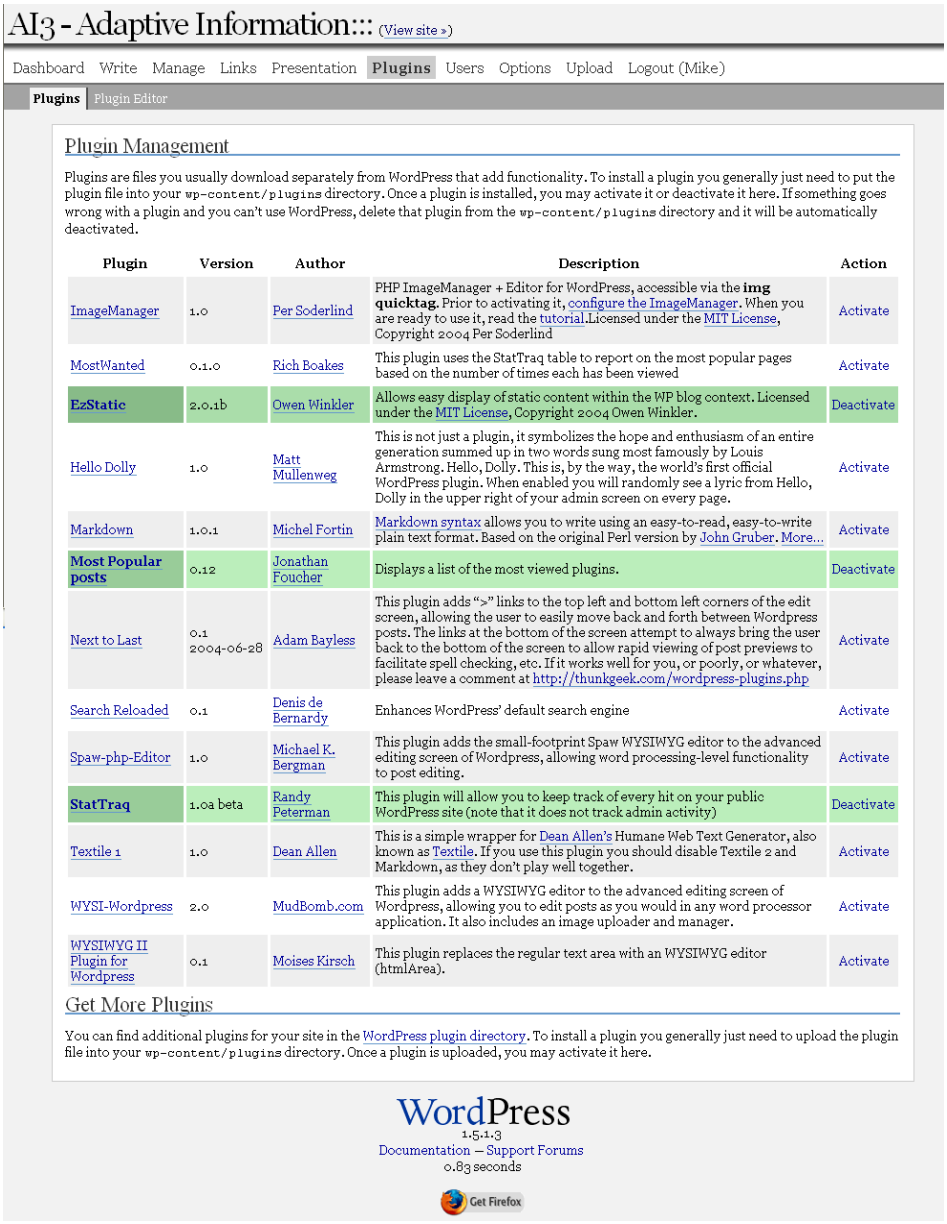


Figure 2. Example AI3 Plug-ins

Note that all of the active plug-ins for **AI3** display in this case in green. Others have been installed and tried but are not active. Active plug-ins for **AI3** are:

StatTraq

This is a useful statistics gathering plug-in. StatTraq was developed by Randy Peterman. It is the basis for a number of other plug-ins, including Most Popular (next). It is also a bit more difficult to install and configure than most plug-ins. (Some discussion below addresses some of the conflicts that can arise when plug-ins are installed.)

Most Popular

The Most Popular plug-in is the one featured from Jonathan Foucher. It requires the StatTraq plug-in. Number of items to display and what to display is controlled through a parameter setting in the embedded PHP, but the image to the right shows how this plug-in finally appears on the **AI3** site.

Try many plug-ins, but keep your active ones to a minimum to prevent conflicts and upgrade hassles as versions change.

EzStatic

The EzStatic plug-in lets you embed static HTML pages from your root directory into a WordPress template simply by specifying `?static=<filename>` in the URL. It also enables an "Execute PHP" checkbox setting when publishing posts or pages. Its use on **AI3** is not obvious to the outside world. (BTW, Owen Winkler's RedAlt site where this plug-in is found also has another 20 or so useful plug-ins available.)

Obviously, these **AI3** plug-ins are merely a sampling of the dozens available on the listing sites noted previously.

WYSIWYG Editor

A central possible functional add-in is a WYSIWYG ("what you see is what you get") HTML editor. Because of its importance, the entire Appendix A is devoted to this topic.

Better Comment Quicktags

Most blogs provide some text explanation above the comment field for what HTML tags the commenter may provide in her response. However, I'd seen some other sites that had some buttons that automated some of this process. I set out to add this capability to my site with two objectives in mind: 1) make it easy for the non-HTML user to format a post comment; and 2) keep options narrowed to what I thought was an appropriate subset of HTML format tags.

The WordPress Quicktags Facility

The internal WordPress post and page editor comes with what it calls "quicktags." These tags, and their explanations, are:

str
"Strong" - creates a tag that gives strong emphasis (read - bold) to your text
em
"Emphasis" - creates a tag that gives *emphasis* (read - *italics*) to your text
link
Link - creates a hyperlink to a web address which you supply in the pop-up box that is activated by the quicktag. If you select text before you click on the link tag, that text will be used as the link text (the clickable stuff) that will be displayed in your post
b-quote
Blockquote - creates a set of blockquote tags that indent text on both the left and right margins. An example follows

> This text is within a blockquote tag

del
Delete - , text that has a strikethrough line through it
ins
Insert - inserted text, text that has been inserted; marked with an underline
img
Image - this works in much the same way as the link tag, you enter the URL of an image into a pop-up box and the image will be inserted into your post
ul
Unordered List - this adds the opening tag to create an unordered list (bulleted) in your post
ol
Ordered List - this adds the opening tag to create an ordered list (numbered) in your post
li
List Item - this adds a single list item to either the unordered or ordered lists. This tag **requires** either the ordered or unordered list tags to precede it.
code
Code - `text that is formatted` in a monospaced font to differentiate between code clips and regular text
more
More - this tag adds a `<!–more–>` tag to your post, which puts in a "more…" link and puts the rest of your post on another page
page

Page - this tag adds a <!–nextpage–> tag to your post, which continues your post on a second page

Dict.

Dictionary lookup - looks up the word you enter into the pop-up box at dictionary.com and opens the definition page in a new window of your browser

Close Tags

Close Tags - this closes any tags (str, em, link, b-quote, del, ins, ul, ol, li, code) that must be closed in order to stop the formatting from continuing down the page. Each tag also turns into a close tag sign (ul becomes /ul, etc.) to allow you to close that individual tag as well.

You can customize the HTML options available to your site users who comment on postings.

Alex King's Expanded Javascript Quicktags

As an addition to either the existing administrator quicktags or for addition in formatting online comments, Alex King expanded this listing and provided it in a LGPL-based Javascript download. This Javascript Quicktags is up to version 1.2 and is the basis for what I incorporated on my site for formatting online comments:

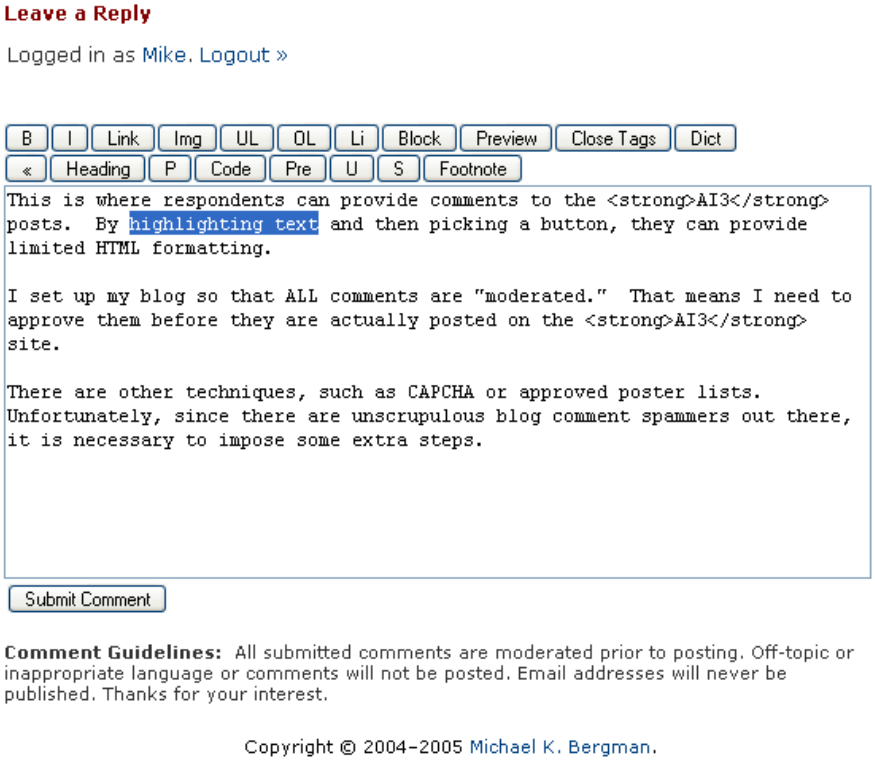


Figure 3. Example AI3 Comment Field w/ Formatting.

While I liked this version, it had some limitations in meeting my objectives:

1. Too many buttons were provided; I wanted a simpler subset of format tools

2. None of the buttons had tooltips, so that it was hard to understand what some of them did
3. Some of the labels needed clarification
4. I wanted a 'Preview' option that would show respondents what their final comments would look like once posted.

The result was that I modified Alex's starting code, the results of which are displayed above. If you would like to install the same version I have on this site – click here to download the zipped Javascript code file and LGPL license (14 KB).

You can download my own source code for how to modify comment 'quicktags.'

In later updates I plan to add a preview capability so that the commenter can see her formatted comments before submitting the post. Longer term, I want to install a limited format WYSIWYG editor, similar to what I am using for my own posts.

Configuring and Testing Trackback

My attempts to grapple with trackback had some difficulties, compounded by my fundamental misunderstandings (which some newbies certainly share). First, the term is not descriptive, and it took me quite a bit of time to understand exactly what trackback was designed to accomplish. Second, most blogs that have a trackback function show it as a link prior to the comments section. I first had difficulty writing my comments code such that the trackback link URL displayed properly (for some reason, I needed to pass arguments in single quotes, not double quotes). Then, when that problem was fixed, clicking on the link produced a 403 error from my hosting server. This latter problem suggested that I did not have trackback properly configured, when, actually, it was my misunderstanding of how the function worked that was the problem.

The breakthrough in understanding came from external sites that offer ping and trackback testing. The one I used was from Red Alt . Definitely use these utilities! I also testing pinging with Pingomation.

The Red Alt instructions showed that my site was indeed sending and receiving pings properly. In fact, the use of the Trackback link appeared solely to be a means to get the reference URL to display for the third party to properly link to it. That caused me to re-think the use of a link for trackbacking in the first place.

Displaying Trackback and Permalinks within Comments Section

To prevent others from having the same confusions I had, I decided to make two changes to how most sites handle trackbacks. First, I decided to eliminate the active links for both trackbacks and permalinks, instead replacing them with more descriptive text fields with URLs that can be copied directly off the browser page. For example, for one sample **AI3** post, the references are:

The URI link reference to this post is: **http://mkbergman.com/?p=103**

The URI to trackback this post is: **http://mkbergman.com/wp-context/trackback/?p=103**

Second, I also decided to make a clear distinction between direct *comments* and *trackback*s in my post comments field. I did this by testing for the type of comment; if a trackback, it is shown as a different type. The PHP code for implementing this is as follows, placed into the comments.php file where the comment display loop is shown:

```
<?php if ($comment->comment_type == 'trackback') : ?>
   provided a trackback on
<?php else : // the comment is a true comment ?>
   commented on
<?php endif; ?>
```

When activitated, a regular comment will show as `XXX commented on date;` if a trackback, it will show as `XXX provided a trackback on date`.

I have found the use of the term 'trackbacks' confusing and the use of a link to get its URL misleading.

With these changes, I now had pinging working, trackback working, and clear distinctions in my comments fields as to true comments or trackbacks.

Permalinks and Issues

Actual implementation of permalinks also posed some difficulties, mostly in relation to the installed plug-ins. As the discussion below indicates, it may sometimes be necessary to make decisions as to which desired functionality can be chosen because plug-ins may be in conflict.

Lost Images

The first problem in turning on permalinks was that all of my site images no longer could be found. According to the last entries in the WordPress support file, I needed to add these lines of code to my main site index file:

```
<?php $basehref =
"http://".$_SERVER['SERVER_NAME'].($_SERVER['SCRIPT_NAME']); ?>
<base href="<?php echo"$basehref"; ?>">
```

As well, I needed to preface my internal image references by '/index.php/'. These changes again allowed my images to be properly displayed, but I guess I'm not sure why all of the pieces worked. With this first problem fixed, I could now address the second.

Conflicts with 'Most Popular Links'

As noted, I use the most popular' plug-in, which broke when I introduced a permalink without an ID field. In researching this problem, I came across a posting by Jonathan Foucher (the plug-in author) noting the issue was indeed when no post id is included in the permalink structure. Because the post id is not

added to the StatTraq table in the 'article_id' field, it causes the 'page views' StatTraq report to show only 'Mixed' page views, not the actual posts viewed by visitors.

I tried Jonathan's suggested resolution by making these line (25 and 26) changes in the stattraq.php file:

```
if (($p != '')){
$p = intval($p);
```

With this replacement:

```
if (($post->ID != '')){
$p = intval($post->ID);
```

Unfortunately, that did not fix my problem.

Since I could not get images and popular links to work simultaneously, I decided to pass, and therefore gave up on using permalinks for the present. I suspect that later updates to StatTraq and WordPress will better address these problems (for example, Randy has announced a pending update for StatTraq). While I like the fact these tools are extensible and many discuss successful hacks, it does concern me to hack code unnecessarily that might make installing later upgrades and bug fixes even more problematic.

So, in thinking about the fact that **AI3** is likely to be very content-filled anyway (besides readibility, one of the cited advantages of permalinks is better rankings from search engines like Google), I decided to put off resolving the permalink issue until another day after spending too many hours on a dead-end.

VIII. GETTING YOUR BLOG TO DISPLAY RIGHT

This section deals with all topics related to presentation and display of your blog. Though the major emphasis is on styles and cascading style sheets (CSS), screen resolutions and cross-browser topics are also discussed.

Most CMS packages use dynamic HTML and XML-compliant XHTML, which have changes from earlier conventional HTML (4.01) standards. Creating clean HTML requires a good understanding of styles and the use of cascading style sheets (CSS).

Understanding styles and style sheets will enable you to control the "look" of your blog effectively.

For example, in older HTML, line breaks were standard with the `<br>`. However, in new code, `<br>` is not handled gracefully and often is replaced with `<br />`. Changes occur in other tags where closing brackets, such as for `<a href="mysource.html">Link here</a>`, is replaced by `<a href="mysource.html" />Link here</a>`.

With respect to the `<br>` issue, it is probably good design to use the paragraph `<p>` tag. (Here, too, standards have changed, wherein paragraphs now are best formatted with a close tag `</p>`, whereas past practice had the close tag optional.)

The real advantage of the paragraph element is that spacing between paragraphs can be precisely controlled in the style definition to achieve the presentation look you prefer.

In any event, all of these considerations suggest a need to better understand how this entire CSS stuff works. Unfortunately, for me, I really didn't understand CSS well when I began to develop my blog (some may surely claim that I still do not have that good an understanding!) What little knowledge I had came from a bunch of posted style sheets that largely acted only to confuse me further.

Realizing I finally needed to bite this bullet, I set off to discover whatever useful guides and guidance I could find online. This section presents the results of those investigations.

Styles

Styles can be added to a Web site in one of three ways:

1. Invoke an external style sheet (*.css) file, which is the best method;
2. Embed and then invoke the entire style definition within a given Web page; and/or
3. Embed style declarations within in-line HTML syntax (e.g., `style="declaration"`).

For many reasons (see below), the first method is by far preferable, especially for a dynamic application such as WordPress.

Therefore, assuming the method used is external, we can now look at the style sheet's building blocks. A style sheet (*.css) file is a plain text file that contains a series of syntax statements, or rules, to instruct how the HTML elements to which they refer should be displayed. Each statement has two parts; a *selector* and a *declaration*. The declaration is enclosed within curly brackets:

```
selector declaration
p { color: blue }
```

Use external style sheets in almost all cases over embedded styles in your blog.

Each declaration has two sub-parts: a *property* (color:), and a *value* (blue). Depending on the number of attributes that can be declared for each specification type, there may be multiple declarations, each separated by a semi-colon, then concluded for that specification when all declarations are completed with a closing curly bracket.

The naming of the selector relates the rule to a particular element of the webpage (in this example, the paragraph tag and the declaration states how that element should be displayed - or blue color in this instance). The selector-declaration has these syntax rules:

- Every listed property must be followed by a colon
- Multiple declarations are allowed for a single selector, each has to be finished with a semi-colon
- You can layout the style sheet to make it easier to read; as with HTML; extra spaces don't matter, and comments can be introduced (see format below)
- There must be a space between each selector and its declaration (because of this, a selector name can not have any spaces in it)
- There must be a space between the property and its value, after its separating colon
- Some selector names are “reserved” because they relate directly to HTML elements
- There is “hierarchy” to some selectors within the CSS; earlier selectors may govern or be inherited by later selectors. Inheritance driven by the CSS occurs within the block invoked in the Web page, unless subsequently overridden within that block, based on the pecking order attributes in the CSS, and
- There is a sequential replacement of equal selectors by those encountered later in the CSS file.

Learn the syntax of style naming to be most effective in using styles.

I prefer the use of the "class" style designation.

There are three ways in which selectors can be named. The first is `selectorname`, with no prefix. The syntax allows this for any named selector to use this format, but this way is also treated specially for "reserved" HTML or style elements. As a result, I advise limiting this naming convention to reserved elements only. The second way is `#selectorname`, where the pound sign defines what is called an "id." And, the third way is `.selectorname`, where the period defines what is called a "class."

Some experts say a Class should be used to select recurring elements, while an ID should be used to select a unique element. Truthfully, for me, I have found these distinctions less than helpful and I have found no examples of where the distinction is useful.

The ID is an earlier form with no differentiating functionality from a class, less functionality than the current class, with a use profile that could lead to possible confusion with an internal anchor link in a Web page. Moreover, classes can be pseudoclassed, whereas IDs can not.

As a result, when I need to class something, I use the class syntax and do not use ID selectors. I'm probably missing some nuances, but this helps keep things simpler for me.

General References and Guides

There are some very useful references on styles.

Everyone should begin with a basic introduction to CSS. I found a very excellent one from the Lowtech Ltd organization in Sheffield, UK, available as a short, 7 pp PDF copy (55 KB). Another useful intro guide that is slightly longer at 19 pp. is from Patrick Griffin at htmldog. While that site contains the document in parts, you can also obtain a PDF version from Stanford University in its complete form.

Finally, the complete bible reference on all things CSS is Cascading Style Sheets, Level 2, available in PDF from the W3C organization. This document is about 338 pages and a 1.6 MB download. While it is not useful as a starting learning guide or tutorial, if you are going to be serious about CSS this reference is indispensable.

In-line v. External Style Sheets

I find it generally cleaner to maintain all styles in an external style sheet (style.css). Embedded styles (in-line) work fine but require each file to be edited if changes are necessary. It is also often difficult to find the style code. And, where there are browser differences, as there are, it is easier to handle exceptions for cross-browser compatibility in an external style sheet(s). By keeping everything in an external style sheet, everything is central and changes are easy.

To further make styles easier to find, I also make extensive use of comments, separate my style sheet into sections, and provide a table of contents of sorts in the header. I also try to use very logical names for each style, such that they are easily remembered and logically apply to the styling task at hand.

Here is the header for the **AI3** site's style.css, also showing comments and nested brackets:

A well organized cascading style sheet helps ongoing control and maintenance of your styles.

```
/*
Theme Name: Adaptive Information
Theme URI: http://mkbergman.com/theme/
Description: Wolfpack theme from MK Bergman's 'Adaptive Information' (AI3)
Version: 1.10 Released 20 June 2005
Author: Michael K. Bergman
Author URI: http://mkbergman.com/

This CSS has been developed by the author to (hopefully) conform with the standard
WordPress style naming conventions.  The general organization of this stylesheet is:

STRUCTURE-WIDE STYLES
   General
   Typographic/Font Styles
   LinkStyles
HEADER STYLES
   H1
   H2
   H3
   H4
   H5
   H6
   Other
   LinkStyles
NAVIGATION (reserved)
PANEL STYLES
   HeaderPanel
   MainPanel
   Story Panel
   LeftPanel
   RightPanel
   Other
OTHER STYLES
   No Border
   Calendar
MULTIPLE SPECIFICATIONS (shared)
UNUSED STYLES (held for reserve)

Except for the standard link styles, shown at top, alternative ones are shown at
the bottom of the appropriate parent style.

/* <<<<<<<<<<<<<<<<<<<<<<<<< STRUCTURE-WIDE STYLES >>>>>>>>>>>>>>>>>>>>>>>>> */
/*------------------------------ General ------------------------------------*/

body {
   color: #000000;
   background: #FFFFFF;
   margin: 10px;
   padding: 0px;
   font: 12px verdana, arial, helvetica, sans-serif;
   }
```

Figure 4. Header for the AI3 style.css

DIV v. SPAN

When you want to assign styles to longer blocks of text within your Web pages or templates use either the `div` or the `span` tags. There are considerations to the use of each:

- The `span` open and close tags are useful when a block of text within a paragraph should be assigned its own style. The insertion of `span` does not introduce a carriage return
- The `div` open and close tags can embrace whole subsections or the entire page within the `body` tags. The `div` tag, when used, causes a carriage return after the close tag. The `div` tag is very useful for sites such as this one where php calls or execution loops can be nested within the `div` element. I use `div` tags aggressively.

The DIV tag is very useful if you must provide different styles to large sections of text.

I personally rarely use `span`, preferring to have actual named styles with open and close brackets if I need to make style changes within a paragraph. I therefore use `div` almost exclusively, and because of the way it works, I most often set a class style definition within `div` as well.

Reserved Selectors

When defining selectors, there are standard HTML elements that should not be confused with custom ones. Generally, you can define a reserved element to have any applicable attribute for that element in your style sheet. You do NOT preface a reserved selector with either the id (#) or class (.) conventions. A useful guide for reserved HTML elements is the HTML 4.01 Quick List from W3Schools. Here are some of the reserved elements that should be considered to avoid in your custom selector naming, and given great attention in your baseline CSS definitions:

This is the 'Body' reserved element.

```
<body>
Visible text goes here
</body>
```

There are up to six 'Heading' elements:

```
<h1>Largest Heading (by convention, but it is not
absolutely required)</h1>

<h2> . . . </h2>
<h3> . . . </h3>
<h4> . . . </h4>
<h5> . . . </h5>
```

```
<h6>Smallest Heading (by convention, but it is not
absolutely required)</h6>
```

Here are other important 'Text' and 'Link' elements :

Naming styles is important – be careful to avoid "reserved" style names.

```
<p>This is a paragraph</p>
<hr> (horizontal rule)
<pre>This text is preformatted</pre>
<code>This is some computer code</code> <a
href="http://www.w3schools.com/"><img src="URL"
alt="Alternate Text"></a>
```

There are many different 'List' elements, starting first with unordered (bullet) lists:

```
<ul>
<li>First bulletitem</li>
<li>Next bullet</li>
</ul>
```

... and ordered (numeric or lettered) lists:

```
<ol>
<li>First item</li>
<li>Next item</li>
</ol>
```

... and less frequently used definition lists:

```
<dl>
<dt>First term</dt>
<dd>Definition</dd>
<dt>Next term</dt>
<dd>Definition</dd>
</dl>
```

There are many elements reserved related to 'Tables':

```
<table border="1">
<tr>
<th>someheader</th>
</tr>
<tr>
<td>sometext</td>
</tr>
</table>
```

Also, 'Frames' (though it is unlikely you would ever use this in a CSS):

```
<frameset cols="25%,75%">
  <frame src="page1.htm">
  <frame src="page2.htm">
</frameset>
```

Though not frequent, it is also possible to set standard appearance conditions for 'Forms' (mostly for the input types and textarea):

```
<form
action="http://www.somewhere.com/somepage.asp"
method="post/get"> <input type="text"
name="lastname" value="Nixon" size="30"
maxlength="50">
<input type="password">
<input type="checkbox" checked>
<input type="radio" checked>
<input type="submit">
<input type="reset">
<input type="hidden">

<select>
<option>Apples
<option selected>Bananas
<option>Cherries
</select>
<textarea name="Comment" rows="60"
cols="20"></textarea>
```

Finally, there are a few 'Other Elements':

```
<blockquote>
Text quoted from some source.
</blockquote>

<address>
Address 1<br>
Address 2<br>
City<br>
</address>
```

Margins and Related

Multiple specifications per style can be made clearer by separating out individual specifications.

In his style sheets, Jerry Tardif explicitly enters the four margin types (or whatever subsets of margins are applicable), rather than use the four sequence convention, a practice I've now adopted. For example, it is possible to label a margin such as:

```
Margin: 10px,10px,10px,10px
```

But what does the ordering of these things mean? Actually, by convention, this shorthand ordering of presentation is T,R,B,L (top, right, bottom, left). However, since I can never remember TRBL (isn't that TeRriBLe!), I agree with Jerry's recommendation to be explicit:

```
margin-top: 10px
margin-right: 10px
margin-bottom: 10px
margin-left: 10px
```

It takes more lines of code, but that is a minor price for clarity. This approach applies to other descriptors such as padding, etc. Generally, however, only the relevant TRBL aspect need be specified.

Custom Elements

If you avoid the reserved names above, you may create and name as many custom style elements as you wish. As the discussion above noted, I tend to only use class definitions, but that is likely because I'm not smart enough to know the difference and keeping things simpler in my styles sheets is important.

Use online W3C services to "validate" your CSS. Not only does it lead to better Web code, but you'll learn a lot along the way.

Use logical names that are *descriptive* with *generality* and *inheritance* in your custom elements. The reason to be descriptive is obvious. By general, I mean named "classes" such as 'Panels' or 'BlockText' or 'Inventory'. By inheritance, I mean nested names from the general such as 'LeftPanel' and 'RightPanel' related to 'Panel'. While tempting, try to avoid names that have too much style specificity such as 'RedText'. If you later decide red is not for you and you prefer green, you will have confusing style references in your HTML.

Validate Your CSS

Finally, when all CSS changes are updated and finalized, validate your CSS syntax. The W3 organization offers an easy online CSS validator. You may need to make final editing changes based on the validation findings.

Cross-browser Compatibility

As I got near to releasing my site, I realized my site was not displaying well in Internet Explorer, and only slightly better in Opera. I typically do all of my development and Internet use with Mozilla (1.7.6 at present). This section shows what was wrong and what I've learned.

Examples of the Challenge

Depending on how the style sheet is written, you can have major problems with cross-browser display. For example, here is the main screen shot for **AI3** as represented in Mozilla (Firefox displays similarly) (~45% reduced size) prior to my giving this issue attention:

Figure 5. Mozilla Display Before Cross-Browser Fixes

Note the display is not too bad, since I develop using Mozilla.

However, when I brought the same page up with Internet Explorer, suddenly everything was screwed up!!@$^&* Here is the same size screen shot of the same page in Internet Explorer:

Oh, boy! Watch out! Cross-browser differences can make your site look terrible and are a real sink hole of effort to get right.

Figure 6. Internet Explorer Display Before Cross-Browser Fixes

Note that fonts are of greatly different sizes and the layout registry is messed up with overlaps, etc. How do we fix this stuff?

Some Cross-browser Analysis

Cascading style sheets (CSS) and browser support are of relative late vintage. Today, most modern browsers (Netscape past v. 4; IE past v. 5-6; Mozilla past v. 1.6; Firefox; Opera past v. 7; Safari past v. 1) have generally good conformance to standards, but there are some differences. Discrepancies with IE tend to be the greatest, though Microsoft has made strides in better compatibility. For browser comparisons, see further here.

Discrepancies tend to deal mostly with font sizing and treatment, tables and display, and various degrees or not for forgiving older HTML designs. To overcome these issues, users and experts tend to take one of three approaches (or combinations thereof) (a general, useful intro on style differences and browsers is provided by Code Style):

1. Very careful attention to cross-browser CSS syntax;
2. Browser detections with swap outs of replacement styles in other CSS with the browser-specific syntax (for arguments supporting this approach using @import, see Stylegala or Avoiding Hacks. A more definitive treatment of different style sheets for swapping is provided by the Site Wizard);
3. CSS "hacks" or filters that use browser tricks or quirks to set branching and choice conditions for different browsers. The definitive expert in this field and the discovery and developer of many of these techniques is Tantek Celik, now at Technorati;
4. And, avoidance of browser-incompatible CSS.

Since the latter option eliminates too many options, I will not discuss it further. In general, because browsers are constantly upgrading, experts recommend NOT making style adjustments within actual Web pages. Use of external style sheets enable global changes and quicker, more consolidated updates and changes.

Differences in font rendering are a key source of browser display differences.

I am by no means a CSS expert, and, like previous topics, there is a dearth of central information on these topics anyway. However, I did discover that the biggest incompatibility I had for my own site were for font treatment between IE and the Gecko (Mozilla/Firefox) browsers (tables and padding were a big problem prior to IE v. 6 as well).

Font Treatment

Text for the screen is sized with CSS using either pixels, ems, percentages, size numbers or keywords. Though a pixel appears precise, IE does not handle re-sizing well with it. em is a precise measure that has fewer issues. Different browsers use different sizing conventions for keywords (such as `small`, `medium` and `large`). This example shows that both keywords and size numbers are not handled equivalently by "modern" browsers:

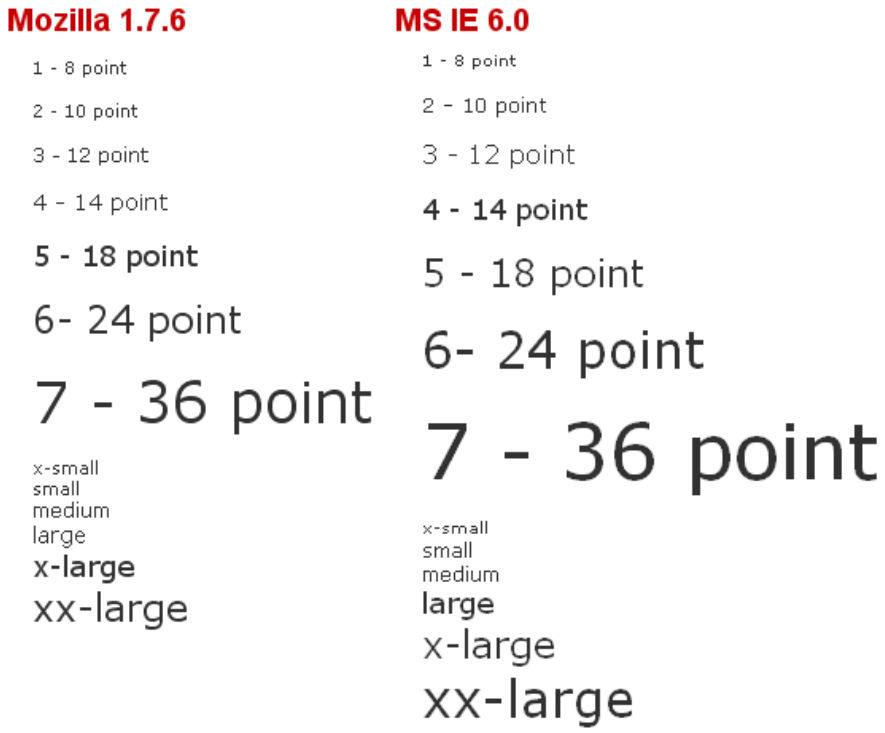


Figure 7. Differences in Font Treatment Between Mozilla and IE

Try setting base font settings to pixels in your CSS, with percent variants thereafter.

The main problem shown in the screen captures at the top of this post is a mismatch in font sizes between Mozilla and IE; most of the layout problems cascade from these font size differences. My investigations suggest there are two main reasons for this:

- First, many recommend specifying the font-size attribute as a percent value because it maintains the differences users set within their own browsers. Percent values are OK in elements that are inherited, but if used for the base font definition in a style sheet, (`p` and `body`, for example), this exacerbates the inherent differences in the default font sizes used by browsers. IE 6, for example, uses a different default font size than Mozilla 1.7 (see above). If you instead set `p` and `body` as, say, 12 px, that equalizes the starting points and later percent differences from that baseline will be adequately reflected.
- Second, there appears to be greater discrepancy in how `id` and `class` are handled. To be safe, I have moved to embed style or heading differences where fonts are involved from the normal baseline by enclosing them within `div` tags.

Finally, for a nice tutorial on fonts and CSS, I encourage you to see the 2003 one by Owen Briggs at the Noodle Incident.

Screen Resolution

As time to release the site drew near, I was presenting the draft **AI3** site to some colleagues for commentary. I was using a projection system that forced my laptop screen display to 800 x 600, unlike my normal desktop setting of 1024 x 768. Unfortunately, registry of key elements was totally screwed up with this lower resolution. I realized I needed to also keep in mind the 20% of users or so that use this lower resolution. (Note: up until a few years ago, we designed our products for the 640 x 480 resolution, now totally abandoned. We now standardly design to 1024 x 768, but again some users with older machines and browsers or who need larger fonts keep the 800 x 600 setting.)

What Others Have Found

Like the previous cross-browser issues, there are differences between browsers (specifically IE) that need to be accounted for in screen resolution-compatible design. However, the major issue relates to the use (or, more accurately, the improper use) of the so-called "box" model in cascading style sheet and modern HTML design.

Here are some sites that reference the "box" model and positioning that I found helpful:

- Brainjar has a good multi-part tutorial on CSS positioning, and
- These general considerations are also related to the design of "liquid" Web pages, as a helpful The Man in Blue discussion shows.

Oh boy! Screen resolutions are another potential gotcha!

The guidance from these sites suggested that the key aspects necessary for modifying my site for different screen resolutions were to: 1) make the display elements "relative" in positioning, and 2) set display orders (`z-index`) properly for wrapping and overlaps. (The `z-index` sets the display order of an element with negative values allowed, with a higher number always in front of another element with lower stack order. The `z-index` can also be set to 'auto', in which case it has the same display order as the parent element. `Z-index` only works on elements that have been positioned, e.g., '`position: absolute`'.) However, as noted next, there were undocumented changes also necessary to get the proper display.

Specific Changes to the AI3 Site

I made all of the changes to my class definitions to '`position: relative`' and '`z-index`' to reflect the display order and foreground/background preferences suggested by these references. But I was still not getting the display desired. After experimentation, I found that I needed to wrap all display elements that occurred below the masthead in a parent class that would provide a "relative"

positioning to the masthead, after which the left and right panels could be absolutely positioned below it. This I called the BoxParent class:

```
.BoxParent {
    position: relative;
   }
```

In addition, my index.php files also called in left or right panel includes at the bottom of the file. For some reason for registry order to still work properly, I also needed to move these up to the top of the file prior to the posting loop that occurs in the main panel. While I did not search exhaustively, I did not find any reference to why this change was necessary nor why it worked. Oh, well, another mystery for this newbie.

These changes provided proper registry in Mozilla/Firefox and IE for different screen resolutions. However, for reasons unknown, and perhaps related to missing end tags in the multiple php files that are included and invoked during the dynamic display, I found I was seeing both indent and paragraph centering problems in IE that did not show in the other browsers.

Use of the "box" model and relative positioning helps quite a bit in different screen displays.

My solution was to put extra calls in my source files either for ending a center (`/center`) or specifically setting margins in-line rather than coming from the external CSS. These are surely hacks and probably can be chalked up to the standard cross-browser gremlins. Nonetheless, they worked for now, and I was able to move on.

(**NOTE:** Later HTML clean-up related to XHTML compliance made many of these hacks moot. If you see similar problems, you may want to go through the XHTML compliance steps below before invoking other browser-specific hacks.)

One of the compounding difficulties of the **AI3** design is the use of a GIF in a table within the masthead that forces width in other columns to be narrower at lower resolutions. This effect causes the overall masthead depth to be quite large at lower resolutions, which is not aesthetically pleasing even though the efforts above fixed the registry and layout problems.

The next step is for me to include some Javascript to detect browser screen resolution. At lower resolutions, I would switch out a smaller version of the masthead GIF and lower the scaling factors for the **AI3** and other fonts in other cells. When implemented, I will invoke these four changes:

1. Use the Javascript to detect screen resolution, and then at lower resolution
2. Replace out the masthead image
3. Change the 900% scaled fonts to 600%
4. Change out the 150% scaled fonts to 115%.

XHTML Validation

Though certainly not required, in keeping with my interest in embracing the full scope of blog standards and techniques I decided to get W3C (World Wide Web Consortium) certification for the validation of my site XHTML. I had done so earlier from my cascading style sheets (see above) had verified through the W3C validator that the WordPress site, itself written in WordPress, had valid XHTML. If any errors occurred on my own site, these indicators suggested that they were only being introduced by me.

Why XHTML Validation?

The W3C validator checks for valid XHTML 1.1 (other versions check for version 2), which means the site identifies itself as "XHTML 1.1" and that W3C successfully performed a formal validation using an SGML or XML parser (depending on the markup language used). Passing this test shows readers that you have taken the care to create an interoperable Web page, you may therefore display the W3C XHTML validation icon on any page that validates.

Like CSS, XHTML validation using online tools from W3C produces better code and helps instruct in modern HTML coding.

XHTML validation suggests that your site and its code meets current standards and will likely render and display properly in modern browsers. By giving attention to these factors early, effort in later post-cleanup are greatly reduced. (Though it is the case that dynamic sites that are template driven are easier to cleanup than static sites where mistakes are repeated on every HTML page.) I was also interested in the validation because of my assigned scope to get familiar with the entire blog and current environment.

What Initial Testing Showed

My first validation test on the **AI3** entry page showed 284 errors. After checking further to remove duplicated errors resulting from the main display posting-and-comments loop, I found there to be about 142 errors in the main page template, which included the cascading includes of header, masthead, left- and right-hand panels, and main display area, which displays the results of the WordPress loop.

Here is where I would first turn my attention, then followed by the loop errors. In general, here are the types of repeated errors I had made:

- A lack of full close tags (`__/`) v. (`/item`) for elements such as link and image tags and line breaks
- Sloppy definitions of color and other parameter definitions that were not included in quotes (e.g., `color="#820000"` v. `color=#820000`). All of these were subsequently quoted; this is an important area to be attentive for future entries
- Place `alt` tags on all image references and elements

- Avoid the older italic (`<i>`) and bold (`<b>`) tags, using instead `<em>` (emphasis) and `<strong>`, respectively
- Replace the old style line break (`<br>`) with the new style closed break (`<br />`), and avoid breaks where open and close paragraphs can be used instead (`<p> paragraph body </p>`)
- If using open and close paragraphs for spacing, make sure a space is included between the tags (`<p> </p>`)
- Take care when using `<div>` tags and make sure they are even and balanced with equal number of closes (`</div>`). This problem in particular caused display differences within MSIE and some temporary hacks that could later be removed when the `div` tags were balanced.

Efforts to Cleanup Errors

My efforts to identify and cleanup these errors took about six hours. After a short period spent understanding the cryptic messages from the W3C validator, the errors began to fit into patterns and corrections occurred more quickly. Also, here were some of the lessons learned:

Here are some hard-earned lessons that may aid your own display efforts.

- Except for very targeted and limited use of in-line styles, use external CSS where possible
- On the validator, set to verbose comments to get more error descriptions
- Re-submit validations frequently as various parts of the code are cleaned up. This reinforces lessons learned and provides gratifying feedback
- Be careful about the use of in-line v. block elements, and make sure block elements embrace in-line ones. For example, bold is an in-line element. It should be used internal to a block element tag such as paragraph. The following is correct syntax: `<p><strong>This is a bolded paragraph.</strong></p>`. Reversing the order of `<p>` and `<strong>` is incorrect syntax
- Inspect the WordPress (or other blog software source) code for internally named classes and give them particular care in the CSS, adding if necessary and using where possible, and
- Avoid all use of the older `<center> text body </center>` style.

Resulting Testing and Validation of the Site

When the cleanup was completed, most pages within my site validated, especially the static ones and postings based on the WordPress loop. I had some particular problems with the PHP calls for the trackback mechanism when the reference was within link tags (`<a href="item" /> <?php trackback stuff>Trackback<a />`). Strangely enough, by single quoting the reference, I was able to clear up this nasty error.

I have found a couple of places where my posts will not validate. These point to errors within the WordPress PHP code and when encountered I have chosen not to make the cleanup. This is based on my desire to not alter or hack basic WordPress code that might change in future versions and cause integration problems with my prior hacks.

Use of Valid XHTML Icon

Taking this effort and cleaning up the code enables you to display the valid XHTML 1.1 icon:

If you use XHTML 2, there is a different icon and more stringent online validation tests.

IX. SITE AND SOFTWARE UPGRADES

The strengths noted earlier in flexibility and extensibility via third-party functionality using CMS framework shells such as WordPress can create weaknesses, as well. Multiple contributing parts means multiple version upgrrades and schedules. Keeping these moving parts coordinated is an issue in its own right.

Major Upgrades

Of course, the key driver for software upgrades is the CMS base hosting system – WordPress in the case of **AI3**. The initial installation of **AI3** used version 1.5.1 of WordPress; by the time of this document, the version had moved to 1.5.2 with 1.6 reportedly in the wings.

A similar pattern exists for all add-ins, including the Xinha WYSIWYG editor (Appendix A) and all plug-ins.

Obviously, with multiple greyhounds racing around the track at different speeds, it is difficult to keep all software aspects in sync, not a trivial problem. It is endemic to an open source approach with multiple tools, as is the case for **AI3**. There are some management techniques to lessen the impact and the pain.

But, first, let's look at what typically needs to happen in version upgrades, using WordPress as the example.

Upgrades have a cascading effect depending on how many third-party tools or plug-ins you use – most don't keep pace with base blog software changes.

Version 1.5.1.3 Update

Below is a list of the updated files, following the same steps as indicated in the David Russell blog. Because he was going from v1.5.1.2 to v1.5.1.3 he only had to upgrade 6 files. For us to go from v1.5.1 to 1.5.1.3 we had to upgrade 16 files:

```
/wp-admin/quicktags.js
/wp-content/themes/default/header.php
/wp-includes/functions-post.php
/wp-includes/functions.php
/wp-includes/pluggable-functions.php
/wp-includes/template-functions-category.php
/wp-includes/template-functions-general.php
/wp-includes/template-functions-links.php
/wp-includes/template-functions-post.php
/wp-includes/version.php
/wp-includes/wp-db.php
/readme.html
/wp-blog-header.php
/wp-login.php
```

```
/xmlrpc.php
```

For reasons noted below regarding unanticipated consequences of hacks, we chose to only update these selected files.

Use of the Local Testbed

An unanticipated advantage I have found to having a local testbed is that I can post, play and refine changes in both look-and-feel and functionality without exposing myself to going "live."

This approach actually requires that you maintain a private, local version of your blog site as well as the one that is publicly available.

I do so myself. Fortunately, the local true "testbed" need not be kept up to date with the actual site ***content***. Rather, synchronization is important in terms of versions (having a local backup site also works well for testing commitments to version upgrades), functionality (including add-ins and plug-ins), and style and layout changes.

A local "testbed" is also useful for testing major functional changes, as I am presently doing with the test integratin of Wikis, industrial-strength search, and large-scale content categorization. These latter pieces represent close to quantum leaps in current state-of-the-art for blog software and merit much offline testing.

Be EXTREMELY careful about hacking base blog software code – it can pose great difficulties when upgrades are inevitable.

Cautions About Hacking

As you incorporate more add-ins and plug-ins to your basic CMS software, it is tempting to modify the underlying CMS software code (be it PHP, Perl, Java, or whatever) to get each new addition to work properly and as advertised. Such *ad hoc* modifications are generally known by the term 'hacks'.

This incremental make-it-work approach appears well and good, but, of course, when a new upgrade of the base software comes out, it has no knowledge of what you have done locally to the base code. If you upgrade without thinking, the new code will overwrite your hacks and potentially leave you in the same dysfunctional state that required the hacks in the first place.

Hacked code can, of course, also be compared using various utilities to identify differences, which can then be changed in the new version code. This is not only laborious, but most often these CMS packages consist of tens or hundreds of files. Keeping track or even analyzing differences within this complexity is silly, and if actually attempted, foolish.

The best strategy, therefore, is to adopt the rule: No Hacks!

This advice may not apply if you are a developer and futzing with code is part of your joy and existence. But if you are using this software for purposeful content-publishing ends, then resist changes! You will only incur heartache and complexity to achieve some minor functional purpose, all at the cost of breaking your blog whenever an upgrade occurs.

X. CREATING ACTUAL BLOG CONTENT

You will note that this topic of creating actual content comes quite late. That is because most all successful blog authors recommend that you plan and organize before starting to write and post. As in all matters, "Ready! Aim! Fire!" works better than spraying bullets indiscriminantly.

Only after I had worked out the kinks in local hosting the blogging software, had completed design and table of content decisions, had identified and added plug-in functionality, and had settled on some organization and work flow issues, did I actually turn my attention to writing.

Standard Site Content

I first tackled the standard site content, not time sensitive, and the "pages" vs. "posts" on the AI3 site. I'd done some earlier work pulling together bio and mission material; I spent most of a day completing those items and some of the other linkage glue.

The process of creating actual content began to surface some weaknesses in my initial planning. I was very disappointed, for example, with how difficult it was to compose in a free-flowing manner without having to worry about formatting and HTML. The Xinha WYSIWYG editor I had chosen is not available to both posts and pages, and copy and pasting between environments does really screwy things with respect to word wraps, picking up (or not) embedded HTML (if it doesn't, it requires pretty painstaking editing). In fact, this whole disappointing experience led me to work out the process for migrating content to the blog (see Appendix B).

Preparation for site release ultimately requires draft posting of content to surface new problems and issues.

Content in Anticipation of Site Release

After about two months of preparatory effort, I finally released my blog site. I kept a diary chronicle of releasing the **AI3** site called "Preparing to Blog" that is the basis of this Guide.

After preparing the content for my standard site boilerplate, I used this diary approach to establish a rhythm and work flow for eventual "live" blogging. I don't know if this approach is useful to others; however, it certainly was to me. Despite being an inveterate writer, I was actually nervous about posting my site "live" and simply getting in the habit of writing daily or frequently using my adopted blog posting procedures helped.

The actual site release was a bit more laborious than I first would have thought, though most of the effort arose from my recording and changing dates than due to

actual release posting demands. Nonetheless, these were the efforts I needed to complete for the final unveiling:

- Update the entire site to WordPress release 1.5.1.3, reportedly necessary for security updates and because of some trackback implementation issues (see later)
- Complete all open postings; there were about four of these carried over from earlier posts. I need to continue to remind myself to post and "document as you go" rather than these difficult reachback efforts
- I needed to change the date stamps on all "Preparing ..." posts and put the standard author's note at the end of each. Though not strictly demanding, these efforts actually took about 3-4 hours because of the delays in post updates noted earlier
- The site domain needed to be changed from the temporary IP that had been used to mkbergman.com, which led to the next set of problems
- Switchover of URLs and internal links needed to be checked and confirmed
- I had to add the actual ping sites used in the **AI3** dashboard options (see listing above), and
- All efforts charts and total time estimates also needed to be updated (not necessary except to support my own documentary demands).

Other Minor Tweaks

As the site finally went "live" both on my own laptop and other public users items that had been hidden while in localhost testing began to emerge. The first was an image display problem in Internet Explorer:

If you're like me, you'll be pretty nervous about actually going "live" with your site; some techniques can help.

```
img {
     border-style: none;
     padding: 4px;
     position: relative;
    }
```

Somehow, by adding the new last line above to my standard style sheet, images began appearing properly in IE (I think this fix came from some serious Googling but I forget to note the source).

There were actually numerous of these. As my colleague Jerry Tardif commented:

> *Unfortunately, what you're finding are the little "pain in the ass" problems that are so idiosyncratic with browsers. No doubt you remember how I've regaled you over the years about this being the worse part of GUI design and debugging (I can be relentless bitching about this kind of problem -- it frustrates me so). These*

> *things are not documented anywhere and I have literally wasted hours searching everywhere for a report of a similar problem and its resolution to no avail. Then we try twenty different things that kind of make sense and one of them works. You feel great that you've prevailed, but feel lousy inside that the solution was more serendipity then logic and you know you're going to get bitten in the ass by something similar again. Somehow, there's little solace in that.*

So, I suppose the last admonition as you create content and go "live" is: Expect things to break and to spend time without guidance figuring out indecipherable ways to fix it.

XI. ORGANIZATION, CHECKLISTS, TIME ESTIMATES AND BEST PRACTICES

There can be many frustrations in releasing and then working with a blog. I personally have lost many hours of work due to quirks and screw-ups. Some of the culprit reasons for the **AI3** site may be:

- The delays with server-side posting and editing begin the process; it is not natural to "suspend" normal editing actions while the latency of the network and the server conspire. This, in turn, is complicated by
- This whole "small fry" CMS perspective that has all data being hosted by MySQL. I truly don't know where the bottlenecks occur, but the delays in posting and updates can be HORRIBLE[3]
- There quite conceivably are editor issues associated with these embedded frameworks. Despite everything I've said about tools and testing and the like, most of these pieces "feel" untested, less than "commercial," and incomplete
- There is also an open source aspect. Granted, anyone can put an open source project out there, and many are impressive, including the ones adopted for **AI3**. But they often feel unready and incomplete.

Effective and efficient use of blogging requires changes to work flow and business processes, besides simpy adopting new tools.

It is perhaps not a surprise that productivity benefits from information technology appeared non-existent until just a few short years ago, and then apparently had a major effect on the accelerated growth that did occur. Not only do tools need to mature, but our work processes and organization need to as well.

This section thus deals with some observations about how to organize for efficient blog use and updates, what kind of time it takes to get a blog set up and active, and other best practices.

Site Project Management and Organization

A useful device I have discovered is to create an internal "page" (explicitly excluded from site posting – most of the CMS packages have ways to do this) that is a posting spot for site to-dos or pending topics. Using a unposted page in this manner provides an internal task listing.

Many of the previous 'Prepare to Blog' categories on the **AI3** site dealt with various public aspects of the site. What I realized, however, is that the site itself becomes a sort of filing cabinet or other kind of background management

[3] It appears one culprit for slow performance of final "published" posts may be due to too many ping sites that are slow performing. See further the note on p. 13.

capability. Thus, I created and populated a set of four internal management pages:

- Common Inserts
- Common Styles
- Pending Topics
- Pending Task Lists.

You can not find links to these on the **AI3** site; they are only accessible via my dashboard. They provide a central focal point for project management and task tracking, aspects not seen by the general public.

File Organization and Naming

Many of my posts have images, PDFs or spreadsheet downloads. Thus, another organizational issue is how to name and file these linked references to posts. While there are innumerable ways to handle these questions, here is the approach I have undertaken and refined, with some rationale for each.

Clever file naming conventions can help you keep things organized and sort and find them.

File Naming

I have three objectives for my naming conventions:

1. Enable files or images to be sorted in a logical order
2. Enable files or images to be clustered within their parental post
3. Provide some logical component in the name to identify the content of the file or image.

To achieve these objectives, I construct a four-part name:

`datestampsequence_logicname.extension`

The *datestamp* is provided in YYMMDD format. This order is used because it enables proper sorting in file managers or Open and Save dialogs. The *sequence* is simply an alphabetical sequence to account for potentially multiple posts within a given day. Most will obviously have the *a* sequence; rarely will there be more than a *b* to *d* in the sequence. The *logicname* is the content designator and is prefaced by an underscore for readability. (If there are multiple words in the *logicname*, I also initial cap with no spaces for readability and to save space.) For example, longer posts may have multiple images embedded in them; the *logicname* simply allows quick choices among these multiples. Lastly, the *extension* simply conforms to the file type.

Thus, a logo GIF included in my second post of June 25, 2005, could have an HTML reference in the post somewhat like (without angle brackets):

```
img src="./wp-content/themes/ai3/images/050625b_MyLogo.gif"
```

Note the first part of the path is a contextual reference to the subdirectory location on the server.

A sorted directory listing may also look somewhat as follows, with all related items properly sequenced and clustered:

```
050625a_PriceChart.gif
050625b_BarbLogo.gif
050625b_JoeLogo.gif
050625b_MyLogo.gif
050627a_Revenues.jpg
```

Using these techniques provides uniformity of referencing within my posts and a quick, known path for getting to and identifying every file and image available.

File Organization

Under my WordPress theme directory (**AI3**), I have set up separate subdirectories for files and images. Under each of those subdirectories, I have set up a number of parallel subdirectories:

- PaperName – wherein the 'papername' is a logical name related to the post and is set up for long, complicated posts
- Posts2005 (etc.) – all files or images for other posts are placed under this subdirectory. I chose to use the year designator knowing that the volume of my posts and associated files and images will require at least that level of granularity. Should too many files be placed in these directories, I may move to a quarterly designator in future years as well
- All images are placed under the images branch; all files (PDFs and XLSs, etc.) are placed under the files branch.

Checklists

When in doubt, create checklists!

Though I'm impressed with how quickly a blog can initially be set up and released, my approach has been to adopt a game plan like Patton invading Sicily prior to releasing my site. There's a whole culture, language and set of technologies with which I'm not yet familiar. Once a site goes "live" it's hard to hide stupidity. When in doubt, create checklists!

Check out these lists – we think you'll find them helpful.

Some of the major items and specific tasks, none in particular order, that must be done and learned to getting a blog running on a routine, easily maintained basis are:

- Posting and comments formatting, utilities
- Completion of Blog Web page content

- Figuring out navigation and external link references (the general blogosphere connectivity)
- Longer-term functionality and its integration
- Efficient (?) update and maintenance procedures.

Here, then, are some of the specific areas needing attention as I looked to final release conditions for my **AI3** site:

General Editing

1. Generalize the CSS style sheet, with more objectification and logical class names
2. Create more consistent, minor style sheet classes and see if they can be referenced through a WYSIWYG editor (below)
3. Learn how to have more WYSIWYG editing
4. Learn how to enter standard HTML formatting into new post entries (incl. using bullets and numbered lists; when this displays OK that has been accomplished)
5. Distinguish between list types
6. Add a standard WYSIWYG editor
7. Figure out procedures for moving existing and developing Word content for posting to the site.

Posts and Comments

1. Decide about use of email address on site and for author posting comments --> do others have concerns about email capture for spam and does that suggest some captcha capability?
2. Figure out how to split long posts – such as some of the main articles and white papers I intend to post – into multi-part segments
3. Figure out if there is a slick way to use classes or styles for HTML insertions
4. Add mechanisms for emailing a post, printing a post or downloading a PDF (where that makes sense)
5. Add a standard Next - Back set of links at the bottom of every post for easier internal navigation between posts
6. Work out the kinks in the entire Comments to posts area, specifically whether minor WYSIWYG editing should be provided and how to preview comments before posting.

I keep my own lists online as a "hidden" blog page – they are always at my fingertips.

Site Pages

Major templating of a few page types has already been done. However, text and templates for "standard" internal pages are needed for:

1. 404 error
2. Copyright

3. About me
4. About me (detailed and bibliography)
5. About site (mission)
6. BrightPlanet page
7. DiDia page
8. Masthead picture reference
9. Chronology listing (work out formatting, plug-in and PHP)

Navigation and External Links

1. Add external links, GIFs where necessary, and do the registrations as appropriate for:
   RSS (XML)
   BrightPlanet
   Blogstreet
   Site Meter
   Technorati
   WordPress
2. The selected Wiki
3. Creative Commons stuff
4. Limit some pages from appearing on site navigation
5. Reserve, but don't yet implement, the entire advanced feature anticipated for detailed categorization and placement
6. Replace the current footer section; the queries stuff seems silly
7. Separate out the calendar and archive sections

Wiki Integration

Find a Wiki tool that:

- "Plays nice" with the WordPress environment
- Can use the same style sheet
- Can use the same editor
- Has posted content that is seamless with the site for such things as site searching, categorization, etc.

Other General

1. Research and better understand some of the spam and privacy protection concerns, especially in comments, emails, robots interactions and others
2. Figure out the email return thing where the title line requires the sender to do clever replacements
3. Figure out the Permalink think
4. Figure out the Trackback thing
5. Figure out the Blogroll thing and decide if I want to use that feature or not
6. Get a better search function and box
7. Figure out procedures, icons, etc., for allowing PDF download capabilities

8. Figure out how to archive and get formatted posts and pages from the MySQL database.
9. Add shortcut icon (favicon.ico) for appearing in the browser address bar.

Final Release

1. Clean up all entries
2. Add date stamps/updated dates to all main pages
3. Fix all broker links
4. Finalize all styles, sizing, positions
5. Complete and update this start-up checklist
6. Write up a document best practices for starting and then running a blog.

Stuff to Defer Until After Initial Release

[Note: these items were added just prior to initial release.]

- Defer improved search function
- Defer Wiki integration
- Defer adding advanced categorization.

Time Estimates

I spent over 350 documented hours creating the techniques and learnings in this guide.

As my efforts proceeded in getting this blog set up, I began to realize I was devoting substantially more time and effort to the activity than I had originally anticipated. It was roughly at this time of realization that I began tracking time and effort. I spent about 350 hours(!!) getting my site ready to go, but I know this is in no way typical.

In fact, with services like Blogger, you can be up and running in 5 minutes *and for free* and posting comments immediately. For reasons noted in my 'Prepare to Blog' diary, this *de minimus* effort may not be advisable. On the other hand, my own needs and demands should not be indicative either. In any event, I present below the breakdown of my time and effort tracking and discuss what may be more "typical" expectations.

Unusual Demands for AI3

As I've stated elsewhere, there are some unique and unusual circumstances I have placed on my set-up and investigations leading to **AI3**. I have wanted, for example, to:

- Understand the blogging and self-publishing phenomenon
- Get my hands dirty with respect to existing tools and infrastructure
- Actually put in place a procedure that will allow me to continue to contribute in an efficient way

- Be aggressive about capabilities and understand "gaps" for bloggers (esp. the "top 1%" in moving forward
- Learn and test tools and techniques to discover gaps and friction points suitable for commercial attention
- Push the edge of the envelop on performance, scale and functionality so as to approach industrial-strength blog sites, perhaps suitable for enterprise use; and
- In general, thoroughly immerse myself into this new culture and technology.

I think I've been successful in these aims, but as noted before, incurred time and effort is not typical. As I present the numbers below, I will try to be specific about what may be applicable. Please understand the reference and viewpoint I present is for a serious blog content site, perhaps only applicable to 10% or fewer of bloggers.

The "typical" serious blogger with local hosting can probably get a site organized and ready for "live" release in about 100 hours – half of which for research and techniques.

Time and Effort Breakdowns

The table below presents the results of my time and effort tracking. The table shows that about 350 hours have been spent getting **AI3** ready over a time from decision to do it until commercial release of about three months. These times and efforts are well removed from a 5-min Blogger site!

The table lists about 30 subtasks (generally documented as individual posts on **AI3**) broken into the six major activity areas of Research, Set-up, Adding (or integrating) Tools, Composing Posts, or Posting clean posts with review and formatting.

Please note the red entries, since these are deemed to be specific to my unusual demands for **AI3** and are therefore ***not typical*** of what a serious blogger without these aims might experience. The unusual entries are either entire tasks associated with investigating tools and techniques or the efforts spent in composing and posting the 'Preparing to Blog' diary.

		Hours by Major Area						
Blog Link	Date	Research	Set-up	Add Tools	Techniques	Composition	Posting	Total
First Post - Decided to Blog	4/27/05	1.0				0.2	0.1	1.3
First Blog Test Drive	4/28/05		1.0	0.5		0.2	0.1	1.8
WordPress	4/29/05	4.0	2.0			0.6	0.3	6.9
Local Hosting	5/2/05	5.0				1.0	0.5	6.5
Install Difficulties and Then Success!	5/5/05	2.0	14.0			1.2	0.6	17.8
Design and Hacking CSS	5/6/05	5.0	2.0			0.2	0.1	7.3
No Local Images	5/7/05	3.0	0.5		2.0	0.4	0.2	6.1
Posts/Comments Behavior	5/8/05	1.0	1.5			0.1		2.6
Advanced Functionality	5/9/05	5.0				0.2	0.1	5.3
Site Transfer	5/17/05		6.0		1.0	0.1		7.1

		Hours by Major Area						
Blog Link	Date	Research	Set-up	Add Tools	Techniques	Composition	Posting	Total
Begin Content	5/18/05					0.5	0.1	0.6
Release Checklist	5/20/05	2.0			2.0	4.0	2.0	10.0
Editor Comparisons	5/20/05	6.0	2.0		1.0	6.0	3.0	18.0
Xinha Integration	5/31/05	1.0	1.0	6.0	1.0	0.4	0.2	9.6
External Credits and Thanks	6/13/05	3.0	1.0			0.8	0.4	5.2
Permalink Problems	6/15/05	3.0	0.8		3.0	1.6	0.8	9.2
Standard Site Content	6/15/05		1.0		3.0	16.0	8.0	28.0
Word Docs to HTML	6/16/05	4.0	2.0	3.0	6.0	8.0	4.0	27.0
Site Project Management	6/17/05	1.0			1.5	3.0	0.3	5.8
Not Playing Nice in the Sandbox	6/19/05	2.0			6.0	2.0	1.0	11.0
Use of Styles and Style Sheets	6/20/05	3.0			6.0	4.0	1.5	14.5
Clean Up Posts [not posted]	6/22/05					2.0	8.0	10.0
Some Best Practices	6/22/05	1.0			4.0	6.0	3.0	14.0
Large Document Transfer	6/24/05	1.0		1.0	2.0	1.0	8.0	13.0
Cross-browser Compatibility	6/24/05	2.0			3.0	4.0	2.0	11.0
File Organization and Naming	6/25/05				1.0	1.0	0.5	2.5
The Purposeful Blogger	6/25/05					2.0	1.0	3.0
Time Estimates	6/26/05	2.5				0.8	0.4	3.7
Word Docs to HTML II	6/26/05	1.0			2.0	3.0	1.0	7.0
W3C XHTML Validation	6/26/05	4.0	0.1		3.0	1.5	0.5	9.1
Screen Resolution Fix	7/5/05	4.0	0.1		1.5	1.5	0.5	7.6
Trackback and Ping Setup/Testing	7/12/05	3.0			1.5	1.0	0.3	5.8
Better Quicktags for Comments	7/14/05	3.0	1.0	1.0	0.5	2.0	0.5	8.0
Formal Site Release!	7/18/05					3.0	1.0	4.0
Prepare to Blog Summary and PDF	9/12/05	6.0	14.0		3.0	22.0	7.0	52.0
Summary - 'Typical' Tasks								
Total		37.0	10.6	8.5	33.0	16.0	8.0	113.1
% of Total		32.7%	9.4%	7.5%	29.2%	14.1%	7.1%	
Summary - All AI3 Tasks (incl. red)								
Total		78.5	50.0	11.5	54.0	101.3	57.0	352.3
% of Total		22.3%	14.2%	3.3%	15.3%	28.8%	16.2%	
Summary - Non-'Typical' AI3 Tasks								
Total		41.5	39.4	3.0	21.0	85.3	49.0	239.2
% of Total		17.4%	16.5%	1.3%	8.8%	35.7%	20.5%	

Table 1. Time Estimates by Task to Set-up a Professioinal Blog

Observations and Guidance for the Serious Blogger

A serious blogger should be able to get a fairly comprehensive and well-designed site up and running in less than 100 hours, less if some of the lessons and guidance from this ***Guide*** are followed, and further less if standard site content (mission, about me, etc.) is shorter than what I provided on the **AI3** site. Moreover, unlike the three months it took to get **AI3** released, much quicker

turnarounds could be easily accomplished. The longer times for **AI3** were exacerbated by the three-times effort associated with the site's unusual demands, business travel and demands, and one family vacation!

These other guidelines can help you plan time commitments for ongoing posting and care and feeding of your blog site.

Some other observations that may guide planning for serious blogging from these numbers are (with the obvious caveats that different styles and skills may significantly alter these points):

- As a rule of thumb, consider that research and reading in advance of a given post takes about two times longer than actually writing up the results
- Besides normal composition time, consider adding another 50% of time to make sure the formatting is correct and the posting will display properly. In other words, preparing a "content-rich" document for your blog may require 150% of the time it formerly took you
- Set-up time, checklists, site management techniques, naming and filing conventions, etc., are well worth getting worked out in advance to reduce ongoing maintenance and relieve you to post and respond, and
- Continue to record and maintain best practices as you encounter them.

Finally, ongoing requirements and care-and-feeding will remain demands. If one assumes roughly three "good" posts per week to keep a blog active, the numbers above suggest a weekly effort of about 20-25 hours per week or about 1-2 hrs per day, exclusive of responding to user comments. This may suggest lowering expectations to only a couple of quality postings per week.

Best Practices

Now that I've got some experience under my belt and a few scars, I guess I'll be bold enough to suggest some best practices as one begins blogging:

- Always try to enter new posts in WYSIWYG mode (in my case using Xinha). Understand the editor thoroughly; none are perfect, all have quirks, and being practiced with your chosen one will eliminate bigger problems down the road
- Because of these nuances document, document, and save, save (for example, with today's version of Xinha when adding new content after saving a draft causes the system to sometimes lose line focus or reference, sending the cursor to never-never land in the file; once understood, this one quirk can be overcome by editing within text and not at the end of a line end)
- When creating posts, keep two instances of the browser open. Instance 1) is for completing the text entry on the post; and, instance 2) is for referencing and getting URLs and other external references
- Understand CSS. Somehow, I suspect this is huge moving forward. Too much "mingling" of content and style may constrain (big time!) future changes. For example, use the paragraph designator <p> rather than breaks

 for spacing and presentation (see above). In fact, the entire idea of XHTML and CSS is to remove presentation from content

- Moving external content is difficult. Every cascade of steps introduces whatever quirks each tool has. More tools equals more quirks equals more headaches. Starting from the MS Word environment has its own set of issues, which likely can not be ignored given the ubiquity of these apps
- When content is imported, take the time to get it clean with the content-style distinctions noted above. If you don't do it now, it will be a major pain-in-the-ass conversion later
- Work on new posts as drafts, and perhaps keep some backlogged. When the post is deemed ripe, publish it as a final post
- Prior to final posting, you may need to actually review the page code. Because of editor nuances, prior to posting, review the XHTML code and do some slight re-formatting. I like to split paragraphs with an extra line break. This increases readability
- Prior to final posting, review and take seriously your actual HTML code. If you are using a WYSIWYG editor, this requires a toggle to source. Remember: Your life will change; you need to be able to convert and understand what you have already been posting, and
- Prior to final posting, occasionally validate the code you are producing in draft form using the W3C Validator. You can get the URL reference by "viewing" the draft post while in the system administrator utility of WordPress and noting the URL.

Following a few best practices can make your blogging life easier and more productive.

I know I have set a rather high set of thresholds for my own site in comparison to what I suspect most bloggers want or experience. On the other hand, I have had the opportunity to do a lot of experimentation and tools testing and integration while totally "offline." Some of this experience may be of benefit.

XII. KEEPING THE BLOG ALIVE

It is not instantaneous nor a Eureka moment to move from idea generation, brainstorming, white boarding, or whatever to something that is actually viewable by a broader public and understandable on your blog. In other words, simply because we have the facility to do a quick post, should we?

I strongly believe ideas that get posted need time, thought and attention before being posted. Even then, perhaps many ideas that are subject to the harsh anvil of scrutiny and time may not prove to have posting potential. My guess, actually the case to date, is that committing to a meaningful blog site means literally tens of hours every week or so spent on deciding, researching, preparing, analyzing, writing, and (often, frustratingly given the status of tools capabilities) refining the appearance of an eventual post.

Depending on your style, you may need to commit 20 hrs a week to keep your blog fresh and alive.

Quick, from the hip, ideas can be generated and written fast, but may not have the legs to get uncorked. Because once released a blog becomes part of the permanent electronic midden, it is important to keep in mind "no wine before its time" about posting. And don't post what you may later regret or find embarrassing!

Time Commitments Relating to Style

Different blogging styles will have immense differences on the time it takes to keep up and maintain a blog site, and differences between types of posts also have great variability.

Providing a link and a one or two sentence commentary as a separate post can require as little as a half hour per post, most related to reading the original reference. However, if you track tens or hundreds of Web sites, blogs and news sites, you could easily be spending many hours per day simply reading and monitoring potential grist. This short posting style can perhaps result in an average of one (sometimes two) posts per day yet still not be excessive in total time spent per week.

(Some of the absolutely most popular blog sites are full-time affairs with multiple, short posts per day.)

If you tend toward preparing longer posts – as I do – with some references and much research, the time per post can be more on the order of 4 to 8 hour each (see time statistics in previous Section). With this style, a willingness to commit to 20-25 hours per week devoted to your blog may result in only three or four quality posts per week.

Nonetheless, it should be clear that any but the most casual blog will entail a substantial commitment by you, the author.

An RSS aggregator like Bloglines can be an effective way to prime the writing pump on your desired topics.

Use of RSS Aggregators

There are many "how to" posts out there about tips for adding content to blogs. Common themes are post often, keep it short, keep it interesting, provide new insights and content, and the like. Another grouping of "how to" posts provides a different set of tips around how use of RSS aggregators or other techniques can keep the idea pool fresh for new blog topics.

These guides are useful and I read my share as I began to get serious about posting my own blog. For example, I have particularly found Bloglines helpful in setting up daily monitoring of some of my other favorite blogs and some common monitoring queries. As you discover useful reference sites, you can add them to your own monitoring list. As you focus on specific topics, you can also monitor specific queries. My own listing of monitored sites is shown below:

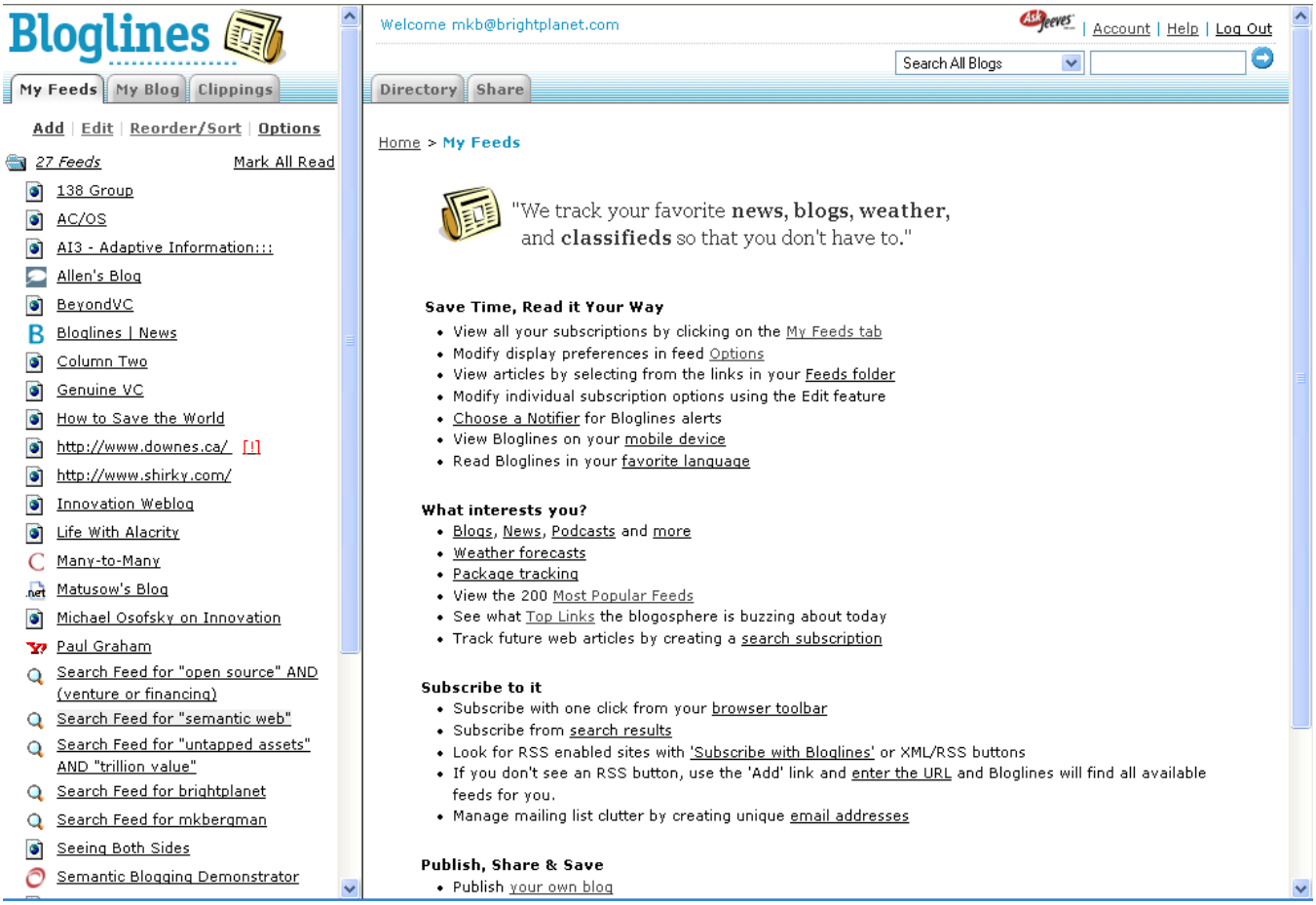


Figure 8. Blogline Monitoring Sample for AI3

This listing is updated for me about every hour or so, with only new posts being listed. Of the 27 sites and few queries above, it takes me no more than a few minutes daily to keep up with favorite topics. These references result in stimulating a new post by me every few days or so.

There are other services besides Bloglines that provide the same aggregation and update service based on your own interest profile. As the aggregator/news reader directory at CMS Review indicates, there are both Web hosted and desktop client readers available. Most of the hosted Web services offer both free and premium versions, and some of the client readers are free or open source.

Whether you choose to use Bloglines or some other aggregation service, I highly recommended this approach for priming your blog writing pump.

"Hidden Page" Topic Listings

As I noted earlier, I maintain a few static pages defined for my **AI3** blog that I specifically restrict from displaying on my site. (In the case of WordPress, this can be set via an exclude statement on the appropriate panel template PHP file):

```
<?php
   wp_list_pages('exclude=36,45,63
   &title_li=&sort_column=menu_order');
?>
```

The excluded pages are still viewable through the administrator's dashboard, where I access and update them.

This approach gives me a central place to record all ideas, remove completed earlier ideas, keep them organized by major thread, and so forth. This simple technique eliminates lost ideas on scattered yellow pads. It is one I have found easy and effective for maintaining posting topic ideas.

Following a few simple guidelines can fuel your blog content with less effort.

Posting Approaches

The creation of posts is the fuel that keeps a blog site going. As I have drafted and prepared this site for release, I've come to appreciate a few guidelines:

1. When an idea of a post occurs, try to draft it completely. Simply beginning a post as a placeholder draft intending to later come back and add the content means efforts are being obligated for the future and the freshness of the insight and ideas goes stale. It is obviously hard to research and complete drafts when first contemplated; after all, we all have regular jobs and demands on our time. But, if possible, complete drafting and delay as little as possible

2. On the other hand, there are options to leave a post-in-progress as a draft. This is useful for reasons of timing, spreading out posts, or keeping in draft absent a missiing reference, event or development
3. However, as noted above, that complete draft should be kept in the draft mode; after some time and reflection, publish it or don't be afraid to deep six it
4. Another advantage of complete drafting is the challenges of editing and re-posting long drafts in the editor and within the blog CMS. These inefficiencies are discussed elsewhere
5. Try to post on a regular basis. That pace obviously differs by individual. Steady paces win the race: "Inch by inch, life's a cinch; yard by yard, life is hard"
6. Post in smaller chunks; it makes the points above easier
7. Be willing to look back on some earlier related posts and re-factor into a more complete treatment. In fact, this Guide itself, is a re-factoring from my original 'Prepare to Blog' series on **AI3**.

Blogs offer a new medium for "chunk-size" postings leading to later "re-factored" comprehensive white papers or guides.

Re-factoring Into More Comprehensive Treatments

One of the areas that intrigues me most is "documenting as I go" with an eye towards possibly a more complete treatment in the future. In software coding, re-organizing and optimizing existing code is termed "re-factoring." Here, I mean it to be combining related individual blog posts into an integrated, comprehensive treatment. Such packaging conforms to my personal likes for paper documents or PDF guides and reports that I can read on the road or before the TV.

I have a few to more ongoing ideas regarding eventual complete treatments of ultimate white papers or guide books. Via the category options within blogging software, it is easy to keep related individual posts clustered.

Should the amount or quality or individual posts in a given topic area warrant – or simply if the muse strikes – it is relatively straightforward for me to combine those individual posts into a larger Word document, strip out redunancies, add missing connecting glue, and then publish as a PDF file.[4]

I have also set up icon GIFs and standard file directory locations and naming conventions for posting such completed reports on the **AI3** blog. For an example of how this appears for this specific ***Guide***, see http://www.mkbergman.com/?p=93.

[4] Of course, it is a little more complicated than that. In fact, I have worked out procedures, sort of similar to Appendix B in reverse, for taking individual HTML postings and then cut-and-paste and convert them into a Word document. I may separately post this technique at a latter time.

Getting Readers and a Community

The seduction of becoming a blogger or auteur is self-evident. However, moving forward without thought and foresight could create heartaches for sites that eventually move into the big time. I now have a great appreciation for those who simply took the blog plunge, learned by doing while exposed in such a public way, and had to fix and correct on the fly.

Readership and community is great, but keep true to thyself in the why, what and when you write your posts.

Within a community of millions of bloggers, there are perhaps nearly as many motivations and types of blogging styles. We know that about one-third of blog sites offer ads, so possibly a partial motiviation for those sites is income; popularity and eyeballs are important. Besides income, other motivations can include a desire for influence or the catharsis of shouting at the rooftops or tilting at windmills.

Common to almost all motivations is a desire, for some an obsession, to gaining greater readership and popularity. There are thousands of guides out there for gaining or increasing readership; it is not my purpose here to recommend or critique them.

Some of the themes in marketing a blog site or making it popular are to link aggressively, to post frequently (generally recommended to be shorter and more frequent), to write on interesting subjects, to write interestingly with a ‘tude and distinct voice, and so on. I’m sure all of this advice is worthwhile.

For me, personally, I certainly would prefer a larger rather than smaller readership, but I am not obsessed (nor, even greatly concerned) with it. My belief is to write on topics of interest to me, to research and analyze, to write as long or short as the topic dictates and as often or not as the muse or ideas strike.

My belief is also to minimize the overhead and unnecessary effort associated solely with blogging. To keep the ideas fresh and flowing, keeping things simple, systematized, and natural is important. Much of what I have attempted to learn and understand – with its resultant techniques – have been geared to me pushing the infrastructure and demands of the blogging process itself into the background.

Through this manner I believe the community interested in my ideas and views will find me, as I am slowly finding them in researching my own interests. And, in the process, to document things like this ***Guide*** that may be helpful to others.

As much as anything, I am fascinated with social computing and the blogging phenomenon that is a part of it. So far, I am very much enjoying being a part of this fundamental change in the dynamics and level of participation in communications and ideas dissemination. May you enjoy this phenomenon as well!

APPENDIX A – WYSIWYG EDITOR

Early in the development and testing of my blog I realized that a lack of WYSIWYG editing was severely limiting my effectiveness. I practically live within Word in my normal work life, and have become totally dependent on efficient word processing tools.

I thus set out to investigate and then settle upon an editor for WordPress. This post presents the results of my investigations. BTW, the editor I chose to use is Xinha, which is not necessarily the best, but became the one I was most comfortable with and confident that continued development will occur.

Starting Points

Among many, here are some useful references to other editor listings and comparisons, some with specific reference to WordPress-compatible ones:

- The original HTMLArea project now retains a list of dozens of editors
- The Content Management System review site, which is useful in general for CMS reviews, also has links to about 20 editors, 14 of which are open source (but not including Xinha)
- The Primate Brow Flash blog has a pretty nice write-up on editors, with specific attention to FCKEditor
- The TWiki site has a pretty good listing of editors and reviews.

There are also editor and plug-in references on the WordPress site, but I found them to be disjointed and less useful than the ones mentioned above.

Design and Use Objectives

Prior to any downloads or evaluations, I wrote out some of my objectives and criteria for eventually selecting an editor:

- Basically, ignored it if did not have strong cross-browser support
- Preferred open source with active support
- Also, wanted to be able to use with Wiki
- Cut-and-pasted as source, rather than HTML (also method of transfer)
- Rendering of non-supported HTML
- Also, wanted to be able to insert styles, consistent with my site CSS stylesheet
- Also, very much like the idea of being able to cut-and-paste from existing Word documents, with extraneous MS-generated HTML (it's *ugly!*) then removed (will try this with the one HTMLarea example)

Some Editor Comparisons

The table below compares some of the major editors that I investigated:

Editor	Notes
Spaw	• like small footprint • seems to be fresh, with support "legs" • like configurability • not current WordPress plug-in
FCKEditor	• acronym leads to bad thoughts • most full featured of all investigated • code looks clean, complete; documentation is another matter • installation appeared very difficult • very strong multi-language, multi-environment support • not current WordPress plug-in
HTMLArea	• many versions of this, but older open source appear largely not to be supported • the zhuo.org site is the most modern, but updates have been nearly one year ago • very much like the idea of copying from Word and removing extraneous codes • See also http://www.zhuo.org/htmlarea/ for an example with an image editor and manager; Zhuo appears to have taken the lead in continuing to support the HTMLArea open source project
WYSI-WordPress	• did not test due to install questions and issues
Cross-browser, Rich Text Editor	• learned of late; did not fully investigate • demo screen looks interesting, with most features desired • looks very clean
TinyMCE	• small footprint has multiple language packs • like many of these other editors, inserts manual line breaks that can wreck havoc on edited code • based on Javascript
WYSIWYGII	• Another based on HTMLArea • Concerned about number of bug reports • Non-English site; concerned about the quality of documentation and support • As a result, did not test; used Xinha instead as the HTMLArea branch
Kupu	• looks quite intriguing • written in Javascript • uses CSS in preference to HTML; could be conversion issues • extensible, but also with limited initial functionality • not yet a WordPress plug-in

Table 2. Comparison of Possible WordPress WYSIWYG Editors

The Choice of Xinha

Xinha is a free WYSIWYG editor replacement for `<textarea>` fields. Use of Xinha is granted by the terms of the htmlArea License (based on BSD license).

Xinha was originally based on work by Mihai Bazon which is copyrighted 2003-2004 by dynarch.com and copyrighted 2002-2003 by interactivetools.com, inc. Xinha stands for "Xinha

is not HTMLArea," referring to Xinha's fork from the earlier open source HTMLArea editor. The HTMLArea editor received much attention, had many plug-in, and had advanced to the full version 3.0 when its standard developers ceased supporting it.

A great demo of Xinha and its various plug-in is available. This site will allow you to test out the editor without going through the pain of installing it.

As the screen shot below shows, there is significant functionality within Xinha and a robust set of plug-ins. The screen shot includes all available plug-ins.

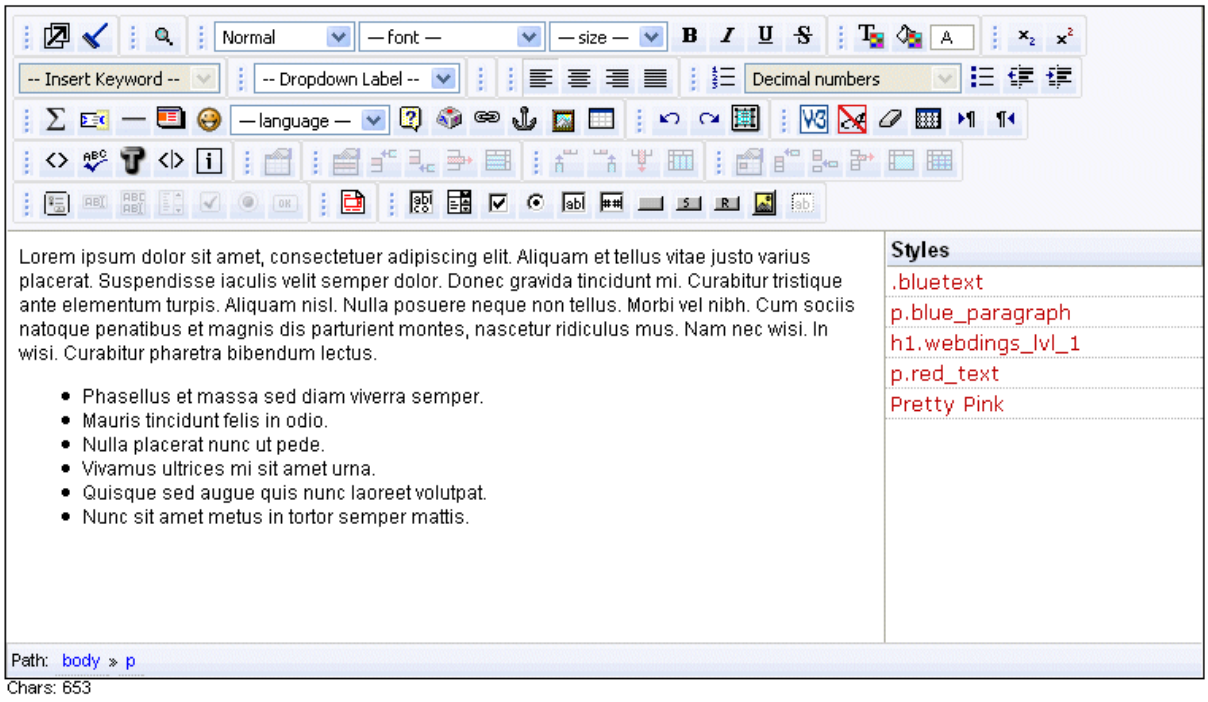


Figure 9. Example Xinha Editor Interface

As you can see, there is quite a bit of format, style, image and table support, plus other important functionality like search-and-replace, etc. The other aspect of the system that is helpful is the ability to toggle to source.

The combination of these features, the earlier legacy from HTMLArea, and the appearance that James Sleeman intends on actively supporting this project caused me to select it for **AI3**'s use.

Gaps and Wishes

In daily use I have come to have both a love and hate relationship with Xinha. On the positive side, it does most of what I need it to do and has not crashed too frequently. On the negative side, it does have quirky behavior and some of the other problems noted below. My specific gaps and wishes are for:

- Better documentation

- Search and replace when in code view
- Suppress forced line breaks when importing code
- More stable behavior when moving from full to partial screen editing
- Easier integration and standard plug-in packaging for WordPress.

Xinha Integration

After deciding that Xinha would be the WYSIWYG editor of choice (at least for the time being), I asked Kevin Klawonn, BrightPlanet's most capable sys admin, to tackle the integration task. Here is his report on his experience:

> *Incorporating Xinha into WordPress has been more than a frustrating experience. I was hoping that because of the apparent widespread use of WordPress and Xinha, the integration would be seamless. At a bare minimum, I was hoping that someone else had achieved and documented exactly what I was trying to accomplish. After looking into how WordPress handles plug-ins, my goal was to configure Xinha so that it would be similar to the existing plug-ins. I wanted to be able to drop the Xinha folder into the plug-ins folder, click the activate button in the admin console, and have Xinha "magically" appear in the editing boxes of WordPress. However, this was not the case.*
>
> *What I found was that for Xinha to work (sort of), I needed to do just as the instructions said and do some hardwiring into the existing pages of the blog. Once I had the new editor working on one page, I went back to looking into making the integration simpler. After scouring several sites for information and spending quite some time on the task of making the implementation of Xinha simple, I am back to the idea that I just need to make Xinha work. Now that I am back on the track of just getting it to work, I am still confronted with some problems.*
>
> *In the Newbie Guide for Xinha, it says that the textareas that are to be converted to Xinha editors need to have their names listed in the my_config.js file. The two text areas, the ones on the edit-page-form.php and the edit-form-advanced.php pages, both have the id of "contents". So in the my_config.js file I entered "contents" in the appropriate list. I checked the two pages, and only the editor on the edit-form-advanced.php is using Xinha. Why would one work and not the other?*
>
> *I read every entry I could on the Xinha website. The only thing that came remotely close to helping was a debunked entry saying to change the:*
>
> ```
> "window.onload = xinha_init;" line at the bottom of the
> my_config.js file to read "window.onload = xinha_init();"
> ```
>
> *Immediately following that entry on the website was another saying to NOT do that because it completely changes the meaning of that line and can have unforeseen consequences. I tried it anyway. Both editors appeared correctly.*

Unfortunately, if a person refreshed the webpage, the Xinha editors were replaced with the regular text areas. So adding the "()" to that line did not work correctly anyway.

Another problem that I have encountered so far with the Xinha editor is that when I do get it to work on a page, it only appears correctly when I use Firefox v1.0.4. If I use Opera v8.0 Xinha does not appear at all. If I use IE v6 I only get a partial view of Xinha. In my research I have found no one else that has these problems. So, once I finally get Xinha working on all the appropriate pages, I then need to find out why it is not working for certain browsers.

Our objective in this continuing effort is to get Xinha integrated as a true plug-in, allow it to be switched on of off with the existing minimal editor, ensure it works across browsers, and other things we have not yet discovered.

Nonetheless, Kevin has done a great job in areas I can't even fathom. It is clear that some blog tasks should not be tackled by mere mortals.

APPENDIX B – WORD DOCS TO HTML

There are *major*, *major* problems and issues with moving large, existing Word documents into a blogging framework. This appendix documents the steps and process by which I have discovered to make this process somewhat less painful.

My first test involved a beast of an analysis piece I had done on the $3 trillion value of U.S. enterprise document assets. (It was eventually posted here.) In its long form, it is 42 pages long, with many tables, a few figures and more than 100 citations and links. It is over 1 MB (1070 KB) in its original form.

Most of us have created Web pages directly from a Word document, and I tried this first. I did the conversion with the filter option, then pasted the result directly into the editor, and attempted to update the site. The transfer seemed to take forever and then the server hung. My suspicion was that the Word HTML code was too complex.

After much trial-and-error, I have found a process for this transfer that has many steps associated with it, works and is quite clean.

When to Convert

This conversion procedure should ONLY be used for longer, complicated documents that have already been created in Word. (It applies to Excel spreadsheets, as well, whether straight from Excel or embedded in Word.) There are thus four ways I use to create content for **AI3**:

- Large, existing Word documents (most often produced for other purposes), for which I use the process documented here. The reference document used in this example is `DocName.doc`
- Short, existing Word documents, which I save as text and then final format within WordPress using the Xinha WYSIWYG editor
- Large, new postings (most with complicated things like tables, etc.), which I compose offline using the Nvu WYSIWYG HTML editor (see below), and then import under the Prepare to Post sections below, and
- Short, new postings, which I compose directly within the WordPress administration center using Xinha.

Thus, assuming we are in the first or third categories, here are the revised steps for Word doc conversions and postings.

First Conversion Attempts

Before packing it in and splitting the original doc into multiple pieces so that my site could choke it down, I decided to do a bit of investigation on alternative clean up utilities and approaches. One good review I found was by Laurie Rowell on 'Clean HTML from Word: Can it be Done?'. I recommend the four-parter.

Laurie's review suggested some improvements could be made from the MS Web page with filtering option using third-party tools, but they did not appear enough to enable me to proceed without splitting my files. Nonetheless, after following that advice, mostly using the MS Web page with filtering, I again attempted a transfer with results as before.

After this initial cleaning, I used both Word and Composer (the Mozilla HTML editor) to do some search-and-replace removal of further HTML tags. Using pattern replacements, esp. with Word, it is possible to also replace line breaks and tab characters, so long as the base file does not have an expected Word extension such as .doc, .rtf or .html (for convention, I always use .txt). This was going well in reducing the sizes of the files (the best I was able to achieve on the 1 MB file was about 450 KB), but the process was laborious even with global search-and-replace. Furthermore, WordPress was not choking down the smaller files. And unbeknownst at this stage, other issues were being introduced into the files that would make later steps even more difficult.

Splitting Files

I then split the document into six parts, the largest being about 250 kB. As before, they were cut-and-pasted into the editor and then posted. I again got server errors and time-outs. With the assistance of Kevin Klawonn of BrightPlanet, he was able to determine that the Apache server was timing out after 30 sec. With a minor parameter change, we were able to get all files uploaded.

However, while the system was now choking down the files, they looked terrible! Line breaks were totally messed up and being able to edit them within the Xinha editor was close to hopeless. Clearly, and unfortunately, the code was still not clean enough.

Problems with Composer

A natural assumption was that these open source editors were "buggy" and unable to handle more commercial strength requirements. As a natural response, I turned to Composer, a standard HTML utility in my Mozilla browser.

I had not used Composer much before, but found much to like. It has nice toggles between HTML source and WYSIWYG. It offers menu options for most "standard" activities I would undertake with a Web page. In short, I liked working with it and thought it might become an offline (at least from my blog) standard for doing HTML WYSIWG editing of large imports. I actually started becoming familiar with the app and its controls and features.

However, upon actual incorporation of the results, I found a nasty truth. Composer introduces forced line breaks at about margin 70. As a basis to incorporate into other apps -- all of which need to work nicely together -- this was fatal. So much for Composer. I was sad .....

Problems with Xinha

I have observed line breaks being introduced by Xinha, but have not been able to reproduce the actual steps. In general, the system seems to be OK about not introducing spurious breaks, including when editing moves from full to smaller screen.

Final Recommended Conversion Process

These early attempts led to further tests and refinements. The steps presented below are the recommended conversion process.

Create Original Word Document

1. Create the Word doc as normal
2. When completely finalized, create as a "special" HTML-ready version. That is, reduce unnecessary styles, move footnotes to endnotes, remove repeated page headers and footers, or eliminate anything else extraneous that may make sense in a multi-page Word or printed document, but not as a Web page
3. Save As the document using the Web Page, Filtered option. Give the document a logical name similar to the original, but include HTML in the name to distinguish it (for example, `DocNameHTML.html`). This is also the version you may need to return to should you have to go back to square one in this conversion process.

Cleanup Word HTML

You are now ready to stage the document for posting. The first phase is to cleanup the ugly HTML created by Word.

1. Create an account with Textism for the Word HTML Cleaner. The service charge is minimal (~$6 for 24 hrs or ~$25 per year) and it does a fantastic job of stripping down to essential HTML and formatting the code for readability
2. Submit the file to the Textism service
3. Cut-and-paste the resulting clean version to your local drive; give the new document a logical name similar to the original, but include Clean or some other standard designator to distinguish it (for example, `DocNameClean.html`).

Edit HTML into Final Form

Assuming you want to make final formatting changes for what appears on your site, such as for example final table formatting and the like, you now need to edit the document into its final presentation look. You will need to use a WYSIWYG HTML editor or composer. In my previous post on this subject, I used Mozilla' Composer. Most recently, I have been trying out the Nevus ("N-view") editor, which is a new branch from Composer by Daniel Glazman for Linspire. (Though there are a couple of frustrations with the product such as using `body text` rather than `paragraph` as the default style and that insidious problem of inserting line breaks shared with Composer, it is an advancement from Composer that looks to be heading in a great

direction.) (The Nvu user guide is also much better than that for Composer.) Thus the edit steps are:

1. Use the editor to make all final HTML and formatting changes. Because of steps to follow that are a pain to repeat, remain at this step until the document looks exactly as you want it with final edits. If, per chance, you look at your document at this stage in WordPress, you may see funky line breaks and some other minor problems (addressed below). Don't worry about them now, they will be cleaned up in the last steps
2. Besides formatting, you may want to look for and remove items such as excess anchor tags (Word cross-referencing can introduce more than desired), caption and other style-specific entries, etc.
3. You will also need to provide image references relative to your site. Click on each image and in the image dialog provide a link path similar to (for my site) `./wp-content/themes/ai3/images/more/moreless/imageXXX.gif`, and make sure you have set a checkbox if it is there to URL is relative to page location. Of course, using this approach means you have already placed the referenced images in the path location so indicated
4. When editing is complete, save the document using a logical name similar to the original, but include Format or some other standard designator to distinguish it (for example, `DocNameFormat.html`)
5. And, because of the next step, you may need a text version. Thus, after saving the HTML version, save another version using the same name but with the .txt extension (for example, `DocNameFormat.txt`)

Prepare to Post

We are now at the final steps to remove some of the problems created because of the quirks in previous tools. If you use a different set of tools or perhaps emphasize different elements (such as forms) in your HTML, you may have slightly different steps that what I outline below. Also, I use MS Word as the final cleanup editor because 1) we began with Word doc problems; and 2) Word has the global search-and-replace (S & R) capabilities and tab and paragraph recognition abilities needed for these steps. Any other text editor that has these capabilities may be substituted. (Note. we needed to name our file as a `.txt` because Word infers formatting based on extension; other text editors can handle html extensions without this problem.). Thus the final preparation steps are:

1. Open Word and make sure you turn on the show paragraph (¶) option; it acts to display paragraph, tab and spaces, useful for determining S & R patterns
2. S & R first step for line breaks: Replace `</p>^p` → `</p>@` (the `</p>` is the actual HTML paragraph end or break, the `^p` is the line break symbol, the `@` sign is used temporarily as a pattern for later replacement)
3. S & R second step for line breaks: Replace `^p` → `[single space]` (don't enter the square brackets, simply the single space. This step removes the insidious inserted line breaks and re-instates proper spacing)

4. S & R third step for line breaks: Replace `</p>@` → `</p>^p^p` (this now re-instates the line break at the correct paragraph end and adds an extra break for code readability)
5. S & R for bullet replacements may need to find HTML patterns created in previous steps that introduce funky characters. In my documents, I often see this symbol (§) as a replacement for the Word bullet, which is shown by the HTML of `&sect;`. Since the pattern I see is `&sect;[single space]`, I replace that with null. Depending on the bullet type you use in Word, you may see different patterns
6. S & R other changes as necessary
7. Then, save the document under a logical name similar to the original, but add Post or some other standard designator (for example, `DocNamePost.txt`).

Post Final

You are now ready to post to WordPress. To do so:

1. Remove the first and last line from your `DocNamePost.txt` file. These are HTML header and body statements. At the top of the file, remove everything up to and including `<body>`; at the bottom of the file remove everything after and including `</body>`
2. Select all and copy
3. Finally, within the WordPress Write section, paste into the text area. If you are using a WYSIWYG editor, make sure you are using code view (< >) when you paste
4. Publish or save as a draft.

Voilà, you are done.

Note that the various versions created during this process enable you to return to any part of the process and make revisions from there. However, should you need to make major changes to the content *after* you have posted it to WordPress, you may be better off deleting the entire post and then re-creating a new one prior to re-posting. This saves update time for some reason.