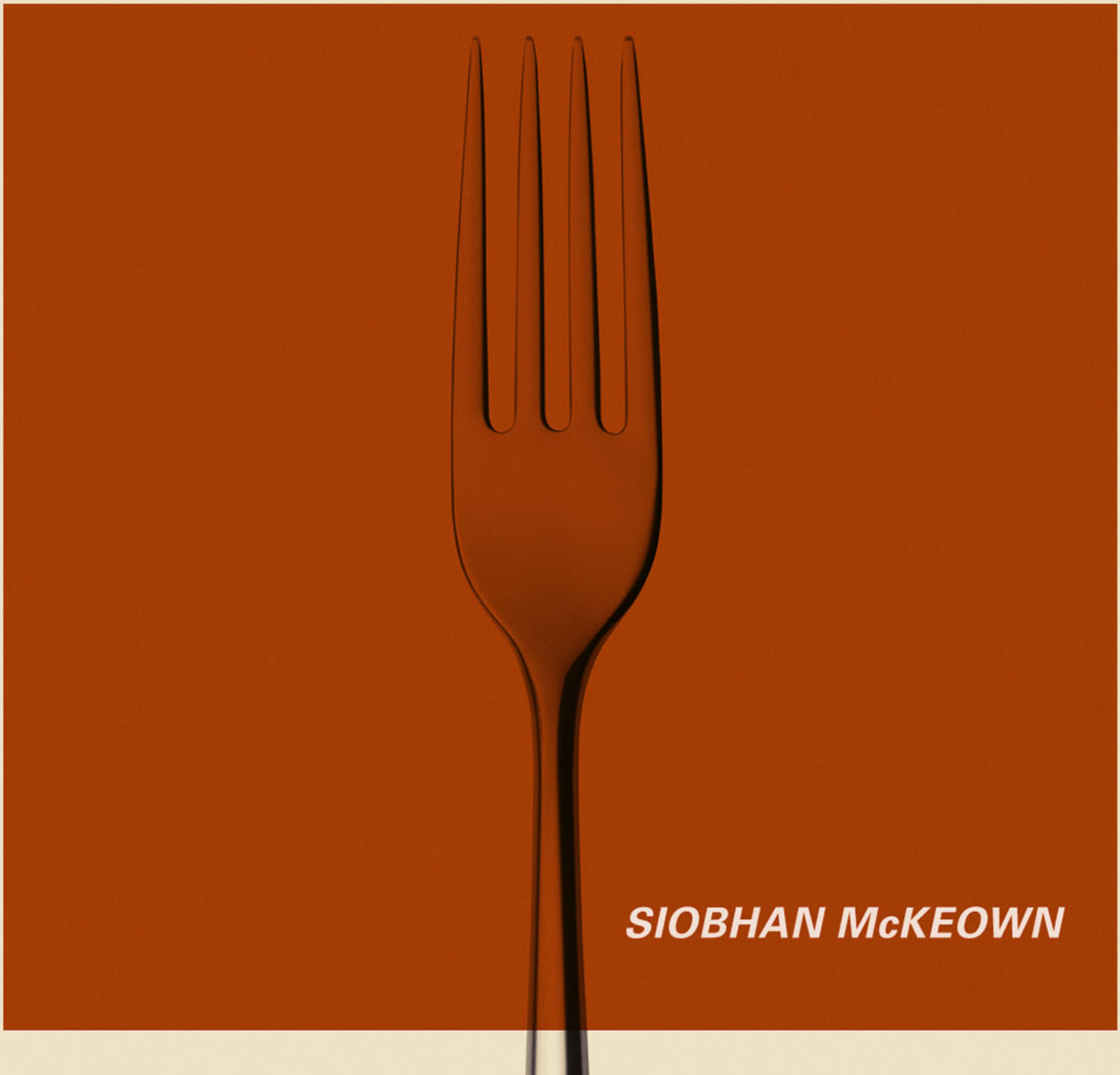
WORDPRESS 3

Freedom, Community and the Business of Open Source

Chapter Three

On Forking WordPress, Forks in General, Early WordPress and the Community



SIOBHAN McKEOWN

CHAPTER THREE

ON FORKING WORDPRESS, FORKS IN GENERAL, EARLY WORDPRESS, AND THE COMMUNITY

By January 2003, b2 was dormant. Michel, michelvaldrighi, hadn't been online for months and no one was maintaining b2. The blogging software at the core of the growing community was no longer evolving. The web, and blogging, was moving forward, but b2 had lost its driving force. The community was concerned. Michel blogged at the end of December 2002 and then didn't post again until 2004. Where was he? Was he okay?

On his blog, Matt wrote a post called "The Blogging Software Dilemma," in which he first mentioned forking b2; the post that led to the creation of WordPress. At the time, he was concerned

about his website's forward compatibility. Jeffrey Zeldman's *Forward Compatibility: Designing and Building with Standards* had influenced his thinking. Zeldman's book covered how to create standards-compliant websites that would work across browsers and devices. Matt had already converted most of his site to XHTML 1.1, but none of this mattered with b2 dormant. Michel's disappearance, however, didn't have to mean the end of the software. Matt wrote:

> *"b2/cafelog is GPL, which means that I could use the existing codebase to create a fork, integrating all the cool stuff that Michel would be working on right now if only he was around. The work would never be lost, as if I fell of the face of the planet a year from now, whatever code I made would be free to the world, and if someone else wanted to pick it up they could. I've decided that this (sic) the course of action I'd like to go in, now all I need is a name. What should it do? Well, it would be nice to have the flexibility of Movable Type, the parsing of Textpattern, the hackability of b2, and the ease of setup of Blogger. Someday, right?"*

The next day, from Stockport, UK, Mike Little, mikelittle, responded:

> *"If you're serious about forking b2 I would be interested in contributing. I'm sure there are one or two others in the community who would be too. Perhaps a post to the b2 forum, suggesting a fork would be a good starting point."*

Forking is the act of bifurcating a free software project, taking it in a new direction. The word fork derives from computing language in the 1960s. In Unix-like operating systems, the `fork()` system call causes a process to split into two by copying itself, resulting in parent and child processes. By the mid-1990s, fork was being used to describe a split in an open source project. It was often frowned upon in the open source community. Eric Raymond, an open source leader and author of *The Cathedral and the Bazaar*, writes about the taboos around forks in his essay *Homesteading the Noosphere*. "The open source culture has an elaborate but largely un-admitted set of ownership customs," he writes. "These customs regulate who can modify software, the circumstances under which

it can be modified, and (especially) who has the right to redistribute the modified versions back to the community." Raymond outlines some of the taboos around forking:

- Social pressure often prevents project forking. Usually forking only happens out of "dire necessity, with much public self justification, and requires a renaming."
- Except in cases such as porting trivial fixes, distributing changes without cooperation from the original developer is frowned upon.
- Removing a person's name, history, credits, or maintainer list is not done without the person's consent.

In general, forking was considered "a Bad Thing," to be avoided at all costs to prevent splits in the development community and arguments over successor legitimacy.

A 2012 study of 220 software forks and their outcomes took a less dim view. To be a fork, the study outlines, the project should

have four characteristics. It should have a new name, be a branch of the original software, have a parallel infrastructure (such as website, version control system, and mailing lists), and have a new development community that is separate from the original.

The study demonstrated how, despite being taboo, forks were often successful. The authors of the article argue that "when forking occurs the sustainability of the software is ensured." In the case of a fork, for example, when development on the original project ceases, it is the fork that goes on to be successful and sustains the software.

It was under these circumstances — the discontinuation of the original project — that Matt blogged about forking b2. As someone who had used the software for more than a year, he wanted it to power his blog. After all, he had tried other platforms, each of which had their own problems.

He had tried Movable Type before moving to b2. He hadn't been happy with the platform — it didn't have pingbacks, which was functionality he was fond of, and comments were in pop-ups as opposed to being inline. Also Movable Type was written in Perl with a DBD database, and Matt was more comfortable with PHP and MySQL. When it came to making alterations, it was easier for him to do so in b2. Dean Allen's Textpattern, was another option. At that time it wasn't GPL so Matt didn't feel comfortable backing it. (Textpattern was later released with a GPL license in June 2004.) Blogger involved using FTP which put another barrier between writing and publishing. With Michel gone and Matt's PHP skills growing, forking b2 was the obvious solution.

By blogging about forking b2 and asking the question openly, he was inviting a community on board. From the other side of the world, Mike Little wanted to get involved. For Matt and Mike, forking was a way to continue to use the software, and develop it for their own needs. On April 1st, 2003, Matt created a new branch

of b2 on SourceForge, and, with the name coined by his friend Christine Tremoulet, called it WordPress.

WordPress' first improvements focused on HTML semantics and web standards. After initializing the repository in CVS, the version control system then in use, and uploading the files, Matt made basic semantic changes to the `index.php` file. He fixed whitespace issues, and converted `<div>` tags into heading tags. Using the correct tags to generate proper headings reinforced the content's semantic meaning on the webpage WordPress generated. Matt also enhanced `index.php`'s default structure, aiming for XHTML strict compliance. Matt and Mike were both committed to standards compliance. Matt often wrote about the importance of standards compliance (in 2004 he would become a member of the web standards project) while Mike wrote enthusiastically about his employer's first standards-compliant website. This iterative approach, along with strict standards compliance, would define WordPress' development over the coming ten years.

Three weeks after Matt set up the repository, Mike made his first commit, repopulating files that were missing from the branch. The first functionality Matt added to WordPress was WP-Texturize, a tool he created to make text more typographically correct. Mike's first feature was excerpt functionality which allowed users to display handcrafted post summaries in RSS feeds and in other places.

Over the coming months, Mike and Matt made over 100 commits to the WordPress repository. Notable commits included WordPress' branding, Mike's b2links hack, which remained in WordPress until it was no longer turned on by default in WordPress 3.5 (released in 2012), major changes to the administration panel, and installation process improvements. Creating a simple installation process was something that both the developers felt strongly about. Both had experience of Movable Type, and like many others hadn't found the process straightforward. It was important for WordPress to create as low a barrier to entry as possible. Anyone should be able to get on the web and publish their content. Matt replaced the `b2install.php` file with a new `wp-install.php` file. The

aim was to keep configuration to a minimum. In its first version, the user had to create a MySQL database, add the configuration details to `b2config.php`, transfer the files to their server using FTP, and then run the script.The "famous five-minute install" was refined over time as the developers tried to make it as easy as possible to get WordPress installed.

Working with b2 had its own challenges. Michel admits that he was learning how to write PHP when he wrote b2. The result: some inefficient code and techniques counter-intuitive to an experienced coder. Michel's code reflected a developer learning PHP. Rather than taking a modular approach to solve a logic problem, the code grew organically, written as Michel thought about it. Code wasn't so much a tool to solve a problem, but a tool to get the newest feature on the screen. This created multiple code interdependencies which could cause problems for developers new to the project. A line of code would change, and break something that appeared unrelated. It wasn't all bad, however. Michel's inexperience also meant that the code had a level of simplicity that made it easy for

other developers to understand. Whereas a more experienced developer might write complex code, Michel often took the simplest route to solve a problem. “In a way it was beautiful because it was so simple,” says developer Alex King, alexkingorg, “It wasn’t elegant but it was very straightforward and very accessible. For someone who didn’t have a lot of development experience coming in — like me — it was very comfortable understanding what was going on.”

During the initial development period, Matt and Mike had commit access to the repository; both could make changes to WordPress whenever they felt like it. They used their own experience as bloggers, as well as their b2 forum activity, to guide them. “As bloggers, we had similar desires to those other people had,” says Mike. “I seem to remember still sticking around the b2 forums and looking at what people were asking about and what people wanted while it was still in b2 and getting inspiration and ideas from that.”

However, much of the early work simply involved refactoring code and making tweaks. With Matt's changes to the admin screens, the first version of WordPress looked very different to b2, but under-the-hood the changes were minor.

Early on, developer Dougal Campbell questioned whether b2 should be rewritten, or whether the code base should simply be built upon. On the support forum, he asked how far the developers intended to go with the rewrite: were they planning to rewrite the whole codebase from scratch, or would something recognizably b2 still remain? Matt said they planned to structure the code more logically as they went along. Object oriented code was the long-term goal to facilitate code readability and usability.

Meanwhile, in France, Francois Planque forked b2 to create b2evolution. Like the WordPress developers, the lack of b2 development frustrated Francois and he wanted to continue to develop b2 for his own needs. Over in Cork, on the west coast of Ireland, Donncha Ó Caoimh, donncha, was working on his own multi-user

project, b2++. Donncha discovered b2 while searching for a platform to create a blog network for his Linux user group. He found b2 small, basic, and easy to modify. With b2 as his starting point, he made major modifications to create blogs.linux.ie. b2++'s templating system used Smarty, which separated code and presentation, making it easier for users on the network to change their site's design. Donncha didn't consider b2++ as a fork of b2. "A fork gives the impression that it was competing — it wasn't competing because most of the things it did was add multi-user aspects to the project." b2 was a platform aimed at individual bloggers, but everything that Donncha did in b2++ created a better multi-user environment.

Matt considered using Smarty for templating in WordPress. In April 2003, he wrote about Smarty and SmartTemplate on the WordPress.org development blog. Matt dismissed SmartTemplate, saying, "the syntax is so painful I wouldn't want to subject it on anyone." Smarty, on the other hand, "becomes more and more complicated with each release." With a commitment to keeping things

simple, neither seemed like the ideal option for WordPress. After researching templating systems, Matt settled on Smarty and Donncha's first commit to WordPress, in November 2003, was an initial Smarty Templating system import. It wouldn't be until WordPress 1.5 that a theme system would be released. In the end, the developers decided to create their own templating system. Smarty was too complicated and the developers wanted to keep templating as simple as WordPress itself.

With b2 spawning multiple forks, the successor to b2 remained unclear. But on May 23rd 2003, Michel, still jobless, announced that once WordPress was launched, it would become the official branch of b2. On May 27th 2003, the first version of WordPress, WordPress 0.7, was released. Users who switched from b2 to WordPress would find some new features, most notably the new, simplified administration panel and the WordPress Links Manager, which allowed users to create a blogroll. With the focus on the user experience the developers tried to remove any barrier between writing content and publishing it on the web. At the time, the website WordPress pro-

duced was standards compliant, brought up to date with XHTML 1.1 — ensuring a semantic, forward-compatible website that would work in any browser.

WordPress' launch sparked immediate project growth. On May 29th, 2003 Matt emailed Donncha to ask if he would consider merging b2++ with WordPress. Donncha agreed, raising the number of official WordPress developers to three. Matt also reached out to Francois Planque to join the project and rewrite his b2evolution improvements for WordPress. Francois considered it but felt that "it was too much work for too little benefit."

After WordPress 0.7 launched, Matt outlined his thoughts on the future of WordPress on WordPress.org. Even then, he maintained that "one thing that will never change is our commitment to web standards and an unmatched user experience," — principles that guide WordPress development to the present day. These principles reflect the fact that the software's developers were also its users. WordPress grew out of a need to publish easily to an elegant,

semantic website. And who better to drive development forward, than the users of that software?

WordPress 0.71 was released only a month after 0.7. As with the first release, many new features reflected the needs of the developers. For instance, OPML import was added to enable importing to the link manager. At that point, Matt hadn't yet imported his 100 blogrolling.com links into WordPress. The new OPML importer let him do that.

Alex King, a Denver-based UX/UI designer and front-end developer, wrote about upgrading from b2 0.6 to either WordPress 0.7 or b2++. Installing WordPress didn't offer any significant speed improvements, while b2++ gave him a site that he couldn't log into. In the end, he decided to wait until the next version of WordPress to upgrade. Matt responded, noting that there would be significant improvements to database speed later, announced in WordPress 0.71's release as a 300% performance boost: "We're not kidding," the announcement post reads, "this release will perform about three times

(or more) faster than previous releases of WordPress and b2." It wasn't, however, fast enough to convince Alex to upgrade.

Additionally, WordPress 0.71 included improvements to the Links Manager, administration screens, and the upgrade process. New post and comment statuses were added. Bug and security fixes were carried out.

Over the year, Dougal Campbell, dougal and Alex King joined the team. Dougal had started blogging on September 11th 2001. He started a blog after terrorist attacks destroyed the World Trade Centers. "I don't know if it's possible," he wrote on that day, "to express the soul-wrenching horror that I feel over the events that have occurred this morning."

At the time, Dougal wasn't a b2 user, though he had investigated using b2 for his blog. In November 2002, he posted about writing his own blogging software, which he called dBlog. Dougal wanted a number of key features in in his blogging software, in-

cluding security, connectivity to other blogging platforms, and content separated from presentation Finally, Dougal wanted configurability — dBlog would have a rich API enabling developers to extend it.

When Matt emailed Dougal to join WordPress development, Dougal was too busy with work to help out, and didn't want to commit only to let the other developers down. Matt suggested he contribute ideas, which led to writing code, and in September 2003 — two years after starting his own blog — Dougal made his first commit to WordPress. As with the other developers, his first commits were small, iterative steps. He added definition lists to the `wpautop()` function that changes double line breaks into HTML paragraph tags. Many of his early contributions to WordPress were RSS enhancements. At the time, people would check their RSS feed over and over again and it would regenerate just like a pageview. This could increase the load on the server, slowing the site down. Web discussions proposed that a HTTP 304 response could be the fix: the server could determine if there was a cached

version at the local end, and if nothing had changed, it would display the cached version. Dougal used this fix to improve WordPress' RSS capabilities.

At the same time, Alex King started making commits. Alex started blogging in October 2002, using b2. With a background in User Interface and User Experience, Alex had only minimal experience with PHP. Using b2 let him develop his skills with a supportive community around him. He particularly recalls Mike Little helping him to refine and improve his code. When b2 ceased development, Alex kept watch on the forks. Despite finding WordPress too slow at first, he maintained communication with Matt and in July 2003, Alex announced that he would help Matt launch a hacks section on WordPress.org.

In August 2003, Alex officially became a WordPress contributing developer. His first project was checking in a cursor-aware quicktag code. This enabled users to highlight a word in the text editor and use a hotkey to surround the text with HTML tags. In the

end, the hacks section on WordPress.org was never built, as hacks were superseded by the plugin system.

In the early days, WordPress developed organically. The core developers identified new features and bug fixes, and committed code when they wished. Most developers focused on the aspects of web development that they already had an interest or a background in. Matt focused on semantics and usability — it was important to him to remove any barrier between writing and publishing. Mike improved his b2 links plugin. He also introduced `wp-config-sample.php`. At the time, all b2 and WordPress configuration information was stored in b2config.php. This meant that an upgrading user had to store the file and information safely. If they overwrote it, their configuration information would be lost, and they'd end up on the support forums looking for help. Including `wp-config-sample.php` meant that there was no `wp-config.php` file bundled with WordPress — the user renamed the file `wp-config.php`, protecting it from being overwritten. This configuration file protection was something Mike had done for pre-

vious clients, and while he recalls now that it seems like an obvious thing to do, it solved a problem that users had encountered repeatedly. Dougal worked on the XML-RPC layer, which at that time supported Blogger's API. XML-RPC is a remote transfer protocol, which allows for remote calls via HTTP. This means that the user can post to their blog or website using a client. The Blogger API didn't cover all of the features that WordPress had. The Movable Type API and MetaWeblog API had additional features that built upon the Blogger API. Dougal added the new features so that the XML-RPC layer covered the entire feature-set of WordPress.

In late 2003, major changes to the file structure of WordPress involved replacing "b2" files with "wp-" in what Alex called The Great Renaming. Tidying up b2's files had been on Michel's agenda in 2001 and had made some improvements already, but they lacked consistency. When Matt finally took on the task, it caused initial problems, especially for hack writers who had used b2 filenames, but the upheaval was necessary to organize the file structure for long-term stability. WordPress' file structure morphed from b2

to the familiar WordPress file structure used today, with many files consolidated into the `wp-includes` and `wp-admin` folders.

In parallel to WordPress developments, Donncha worked on WPMU (WordPress MU). While the initial intent was to merge the b2++ codebase with the WordPress codebase, they ended up remaining separate. WPMU had its own version control system and, eventually, its own trac. Donncha recalls that WordPress and WPMU targeted different audiences. “Most people just have one blog,” says Donncha, “they don’t have half-a-dozen blogs running on one server so multiple sites wouldn’t have been a requirement for most people.” While this situation changed as it became easier and cheaper for people to host their blogs, in 2003 it didn’t make sense to have multi-user functionality available to every WordPress user. Instead, Donncha worked on WPMU alongside WordPress, and merged the changes from WordPress into WPMU. When a new version of WordPress was released, Donncha had to merge each file individually into WPMU. He used Vimdiff to load the two files side by side. He could review changes and push them from one file to

another. It wasn't always easy. "I had to keep track of the changes that were made in case they broke anything. So at the back of my mind I'd be thinking 'did that change I made five files back, will that affect this change?'" As WordPress got bigger and bigger the merges became more difficult to manage.

As the software developed, so did the community around it. Matt launched WordPress.org, in April, 2003. It included a development blog, some schematic documentation, and support forums. The original WordPress homepage told the world that "WordPress is a semantic personal publishing platform with a focus on aesthetics, web standards, and usability." The site gave the WordPress community a home.

Just as Matt and Mike had started out answering questions on the b2 forums, new WordPress users got involved, answering forum questions. This allowed developers to spend more time writing code. A community of non-developers grew in parallel to the development community. People tried WordPress, liked it, and wanted to get involved. The project's user-centric nature meant there was work for people from a variety of backgrounds: anyone could answer support forum questions, or get involved with IRC discussions. Craig Hartel, nuclearmoose, was one early community member who signed up at WordPress.org in November 2003. He was interested in blogging and had a little bit of programming experience. "I didn't have any specific skills," he says "but there was no better way than jumping right in. I decided I was going to find some way to get involved." He got to know people by asking thoughtful questions and letting people know that he was interested in helping. Craig hung out on the IRC channel until he "realized that getting involved was a matter of just doing something."

Along with a few others, Craig pushed for WordPress documentation. As the community grew, more people asked questions on the support forums. Documentation was needed to support the users. In December 2003 Matt, at the behest of WordPress community members, launched the WordPress Wiki. Originally, the wiki was designed to complement the official documentation. The landing page told visitors that it was "designed for us to be able to work together on projects." While little work was done on the official docs, the wiki grew, perhaps because the wiki felt like a much more informal, freeform way to create documents. The official support documents, such as a guide to template tags, hacking, and using WordPress, were much more formal. In a post on WordPress.org, Cena Mayo, cena, who had taken on the role of reporting on the WordPress.org blog, outlined some of the issues:

> *"Part of the problem is the rapidly changing face of WordPress itself. The CVS is currently at version 1.2-alpha, with almost daily updates. 1.2, which will be the next official release, is much different from the widely used 1.0.1/1.02 series, and even more different from the still-used .72."*

With file structures changing, new functionality appearing, new template tags, and new database tables, writing formal documentation must have seemed like a pointless task when the writer knew that things would change quickly. Over time, the wiki grew, with more and more people contributing to it. By July 2004, it was the main documentation for WordPress, and it needed a name. In WordPress' IRC chat room, monkinetic suggested "Codex" and in an email to the mailing list, Matt adopted it, saying it was "short, sweet, and we can totally own that word on Google."

Design was another area of growth. Before WordPress' theme system, users had to apply a new CSS stylesheet to change the look of their blog. To enable users to quickly change their blog's design, Alex King wrote a CSS Style Switcher hack. This came with three CSS stylesheets. To grow the number of stylesheets available, Alex ran a WordPress CSS Style competition. Prizes, donated by members of the community, were offered for the top three stylesheets; $70, $35, and $10. The competition created a considerable buzz around WordPress. In total, Alex received 38 submissions, increas-

ing the number of stylesheets available for WordPress from seven to 45. He was also able to create a WordPress analogue to the popular CSS Zen Garden — a style browser for looking through all of the different stylesheets. The competition was a success, and Alex ran it again in 2005, this time receiving more than a hundred submissions.

While WordPress.org was a gathering place, it wasn't the only home for the WordPress community. WordPress users wrote about WordPress on their own blogs, sharing tips and tricks. In late 2003, a new kind of blog sprang up — the WordPress community blog. Early examples include WordLog, by Carthik Sharma, and BloggingPro. The very first, though, was WeblogToolsCollection, which was founded by developer Mark Ghosh, laughinglizard.

Mark became interested in blogging tools while studying for his Masters in computer science. By the time he came across b2, Michel had abandoned the project and forum support had dwin-

dled. He found he was spending weeks waiting for responses to support requests. Instead, Michel pointed him to WordPress.

In WordPress, Mark quickly found a home, and soon after signing up he set up his own blog. “I looked for Weblog Tools Collection as a Google search and I saved it, and I would always get very interesting results.” He thought to himself, “if it’s just a good, cool, set of search terms, I’m sure other people are searching for it too, I’ll make a blog called Weblog Tools Collection built on WordPress.”

When it was launched, WeblogToolsCollection (WLTC) covered different weblog tools, but slowly it focused more on WordPress. “I suddenly got all of this attention for not knowing a lot and not really doing a lot,” says Mark, “and that really pleased me.” The appreciation he got from the WordPress community for running WLTC spurred him to help out more with the forums, write his own plugins, and get more involved. Posts on WLTC about platforms like Movable Type quickly tailed off and almost all of the

posts are on WordPress, or on migrating from other platforms to WordPress.

At its peak, the blog received 12,000 to 15,000 unique hits per day but Mark was never fully able to capitalize on the traffic. Running a niche community blog takes a lot of work and doesn't result in a huge monetary payoff. "Most of the people who came to WLTC wanted news about plugins, or they wanted to know how to do X, Y, or Z. They were trying to find this information and the quality of audience was kind of low." People who visited WLTC were looking for ways to fix their blog or website, learn tricks, or find out about the latest free themes and plugins. They weren't necessarily the types of people that advertisers would value. However, WLTC would have a major role to play in the development of the WordPress community, providing a home for discussion and debate away from WordPress.org.

In late 2003, another key developer joined the community. Ryan Boren, ryan, was a developer at Cisco Systems. He had con-

tributed to open source projects like Gnome and the Linux kernel and was a big advocate of open source software: “I've never run Windows in my life,” he says. Ryan had been blogging for a number of years, and wrote his first blog post in September 2000. His earliest blog posts were on Blogger then Greymatter until he decided that he “wanted something new and a little nicer.” He liked WordPress' markup and CSS so he made the switch, writing a Greymatter importer to move his content to WordPress.

Almost straight away, Ryan contributed to WordPress. He submitted patches to Matt, until “Matt gave me commit [access] because he couldn't keep up.” He had little experience with PHP, but became one of WordPress' most prolific contributors, and remains a cornerstone of WordPress development to this day. Much of his work on the project has had a big influence on the WordPress ecosystem: one of these contributions was WordPress' plugin system.

Prior to the plugin system, developers created hacks to extend WordPress, a practice carried over from b2. A “hack” was a bunch of code with instructions on where to insert the code into the b2 core files. In b2, administration was carried out in a separate PHP file. The b2 user would open up a text file, `b2menutop.txt`, and add the name of the PHP file that they wished to appear in the menu. When the code ran, the new menu item would appear after the default menu items. To add a hack to the administration screens, the user needed to put the PHP file into the admin directory and add a reference to it in the text file. If the hack output was intended to appear on the website, the user needed to edit b2’s `index.php` file to put it in the right place. This meant that when the user updated b2, they would have to save the text file and the index file to ensure that their changes weren’t overwritten, and integrate their changes into the new files.

The new plugin system used hooks to enable developers to extend WordPress without having to mess with core files. Hooks are pieces of code placed throughout the codebase that developers can

hook into to run their own code at specific points. There are two types of hooks: filters and actions. Filters were already available in b2 for developers to create hacks which changed content. Actions, which were first added to WordPress 1.2, let developers run code when events, such as posting, were carried out.

The system transformed WordPress development for both the core developers and the wider community. It meant that developers no longer had to try to persuade the core team to implement their pet feature, and the core team could focus entirely on what made sense for users. Ryan says that the plugin system enabled the core developers to implement the 80/20 rule: “Is this useful to 80% of our users? If not, try it in a plugin.” Unlike hacks, which involved making edits to core files, plugins could be dropped into a directory in a user's WordPress install. Non-technical users were able to extend their blogs without having to mess around with PHP.

The first plugin was the Hello Dolly plugin. The plugin, which is still bundled with WordPress, randomly displays a lyric from the

Louis Armstrong song *Hello, Dolly* in the top right of the admin. It was intended as a guide to making plugins. The "first [plugin] that actually does something useful" was the blogtimes plugin, created by Matt, which generated bar graph image showing when posts were created over a certain period of time.

Ryan's other major early contribution was around internationalization. From its very beginning, the WordPress community was marked by its international nature. The original developers were from the USA, the United Kingdom, Ireland, and France, and a forum thread from January 2004 shows how international the growing community was. Community members came Hong Kong, Wales, New Zealand, Japan, and Brazil. With people from all over the world using WordPress, platform translations soon followed. The Japanese WordPress site was set up in December 2003, only six months after WordPress launched. As WordPress wasn't yet set up for localization, Otsukare, a community member from Japan, wrote a multilingual hack that enabled users to create multilingual versions of WordPress. Translators used the WordPress wiki to host

the translations and the multilingual hack pulled them into the remote WordPress admin.

When it came to internationalizing WordPress, however, they decided to take a different approach: `gettext()`. This method involved marking up the translatable strings with the `gettext()` function so that a `.pot` file could be created with all of the strings for translation. Translators could translate the strings and generate `.po` and `.mo` files for localized versions of WordPress. To internationalize WordPress, someone had to wrap all of the translatable strings with the `gettext()` function and put them in a format that provided a full string to the translator that retained context. This job fell to Ryan. He went through the code, one line at a time, found everything that could be translated, and marked it up. This meant that when WordPress 1.2 was released, it not only contained the plugin API but was fully internationalized.

On May 19th 2004, Matt wrote about the first full localization that he'd seen -- a localization in Hindi by Pankaj Narula, panjak.

This was before WordPress 1.2 was released with the first official `.pot` file (the file that contains a list of all the translatable strings). Following the release of WordPress 1.2, there was an explosion of WordPress translations, for example French and Norwegian.

By May 2004, a small, enthusiastic community had grown around WordPress. It was composed of developers who built the software, bloggers, volunteers who answered support forum queries and wrote documentation, and a growing number of international users who started to translate WordPress. The release of WordPress 1.2 in May 2004 brought major improvements in flexibility. The new plugin system meant that developers could create their own functionality that could plug in to WordPress without having to create hacks, and WordPress was internationalized, making it considerably easier for people to translate WordPress. And yet while WordPress use steadily increased, it was a catalyst from outside the community propelled that growth exponentially.

In early 2004, the most popular self-hosted blogging platform was Movable Type. The blogcensus.net service recorded Movable Type of holding around 70% of the market share for self-hosted blog platforms in February 2004. It was used all over the world, by everyone from individual bloggers to big media outlets.

On May 13th 2004, Six Apart, the company behind Movable Type, announced that it would change its licensing. Movable Type 3.0, the newest version, came with licensing restrictions which meant that users had to not only pay for the software but pay for each additional install of the software that they created. With Movable Type users frustrated and upset by Six Apart's actions, WordPress community members stepped up to help Movable Type users migrate to WordPress. Downloads of WordPress of from Sourceforge more than doubled, increasing from 8,670 in April 2004 to 19,400 in May.

The Movable Type licensing change threw into relief who held the power in the relationship between developer and user. At any

time, Six Apart could increase prices, change its licensing, and change the rules for its users. The license protected the developers. WordPress, on the other hand, had a license that protected its users, and it was to this user-focused, user-driven community, that Movable Type users flocked. It was the first of many times that WordPress' license, the GPL, would ignite the community, and it positive effects saw WordPress go from a small fork of b2 to a major competitor as a standalone blogging platform.