Tiki for Web agencies

Why your web agency should adopt Tiki as a major web platform

Tiki.org



Tiki Wiki CMS Groupware

- A software to make all kinds of websites
- Free/Libre/Open Source
- Community-managed

The general idea

- Like Wikipedia, but for code instead of content
- The general direction of computing -> everything is going to the browser (Firefox, Chrome, Safari, Explorer, Opera, and mobile browsers) -> The client.
- The Server is a collection of software installed on a server and designed to generate HTML/JavaScript for these browsers.
- Tiki is the Free/Libre/Open Source Web Application with the most built-in features. It runs on top of other software (Linux, MySQL, PHP that the user doesn't see)
- **T**ightly **I**ntegrated **K**nowledge **I**nfrastructure

How?

- Stakeholders collaborate
  - Consultancies
  - IT departments
  - Volunteers
- Commercial eco-system based on services, not code.
  - No free/community code vs proprietary code situation
- Half the code is including code from other projects: "Standing on the shoulders of giants"

Specs & stats

- Tech: PHP / MySQL / Zend Framework / Smarty / jQuery
- License: LGPL
- 500 contributors with commit access
- Downloaded over 1 million times
- 10 years old
- A commit every 2 hours
- 1M Line of Code (incl 3rd party libraries)
- 40 translations

Use cases/features

- **Knowledge base**: Wiki, FAQs, File gallery, Photo Album, Tags, Search, Kaltura video management

integration, etc.
- **Collaboration/Project Management**: Wiki, Forums, Tasks, Permissions, Timeline, Proposals/Votes, Blog, Categories, Watch, etc.
- **Publishing**: News articles, Blog, RSS, Newsletter, Maps, Themes, Banners, WYSIWYG, SEO, etc.
- **Commerce**: Shopping Cart, Payment, Membership, Credits, Accounting, etc.
- **Social networking/Community**: Friends, Maps, Inter-user messages, Surveys, Polls, Chat, Share link, Comments, Calendar, etc.
- **CRM / Help Desk**: User database, membership, Bug & issue trackers, polls, etc.
- **Office suite**: Wiki, Spreadsheet, Slideshow, Drawings, Database (trackers) and Reports, etc.
- **Personal Information Management (PIM)**: Calendar, Webmail, Address book, Notepad, Time Sheet, etc.
- **E-learning**: Quizzes, Slideshow, BigBlueButton webinar integration, etc.
- **Framework**: App builder (forms & reports), Workflow, Profiles, Advanced wiki syntax, etc.

There is overlap between the features needed for each use case.

The Tiki Model ("Software made the wiki way")

- Wiki community
  - Do-ocracy
- Wiki way participation to the code
  - 500+ accounts with write access to the complete code base
- Scheduled releases
  - Every 6 months, with LTS every 18 months (supported for 3.5 years)
- All-in-one codebase
  - Inherent synchronized releases
- Lots of features, but no duplication
  - Do you have two wiki pages for the same thing? No. So, why would it be so for features?
- Dogfood
  - *.tiki.org sites are upgraded before a .0 is released

http://tiki.org/Model

All-in-one design

“In later discussions Torvalds explained the reasons for its choice: **a fully modular architecture, like the one adopted for HURD, would have posed problems to a degree of complexity that it could have compromised the accomplishment of the project.** To avoid such risks and keep the degree of complexity of the project as low as possible, Torvalds decided to design a monolith and he actually wrote all the architectural specs himself, avoiding all the problems related to collective projects (e.g. division of labor, coordination, communication). On the other hand, the HURD micro–kernel, a project in direct competition with the Linux kernel, has paid for the choice of pursuing a fully modular approach from the beginning in terms of the continuous delays that have plagued its development. Nowadays, it is still under active development and still lacks the stability and performance assured by the Linux kernel.”

Source: Modular Design and the Development of Complex Artifacts: Lessons from Free/Open Source Software http://oss2005.case.unibz.it/Papers/25.pdf
http://dev.tiki.org/Modularity

Benefits of integrated model

- Tons of features, without duplication, excellent code re-use and code review, more collaboration, tight integration, easy upgrades, excellent interaction between features, etc.
- Permits huge changes between versions because we don’t have to worry about breaking 3rd party extensions.

- Less code for the community to maintain: http://en.wikipedia.org/wiki/Technical_debt
- No abandoned module, because a new shiny one just came out
- No dependency hell (needing to re-install 3rd party modules at every upgrade, hoping they all still work!)
- No hunting around to find the best 3rd party module

Challenges of the integrated model

- Huge code base to maintain
- Admin panels
  - hundreds of features
  - with a total of over 1000 settings/options! (we had to add a search!)
- Learning curve: 1000+ pages of documentation
- What should be sensible defaults?

More about complexity and files

Tiki 7.1 contains 11348 files and it's the FLOSS Web application with the most built-in features. About half the code in Tiki is maintained by the Tiki community and the other half is re-using code from external libraries like Smarty, Zend Framework, jQuery, etc.

So say we maintain about 6000 files. Sounds like quite a bit, but let's put this into perspective:

- Joomla! has 9463 "extensions"
- Drupal has 15948 "modules"
- WordPress has 19382 "plugins"

Tiki covers the vast majority of features that these 3 systems offer via the thousands of extensions. So just about any project you could do with Joomla!, WordPress or Drupal, you could also do it with Tiki. Yet, they have more extensions to maintain than we have files! (and since they can't possibly maintain them all, it leads to dead-end extensions and disappointed end-users).

See: http://tiki.org/Coping+with+Complexity
http://en.wikipedia.org/wiki/Technical_debt

profiles.tiki.org

Instead of having thousands of extensions, we collaborate on the code base, and can make very specific apps thanks to profiles.

- Value of a free source project
  - Beyond code: the community and the experience
  - Combining the different features, producing new benefits never imagined by the authors
  - How to share this knowledge?
- Profiles to configure your Tiki
  - On wiki pages (collaborative, version history, etc.)
  - Can be used not just at install, but **at any time** and can be **combined**
  - Not just for **settings**, but also for **data**
  - Since there is no extra code, can be designed for very specific, long tail uses

We can have hundreds of profiles for an out-of-the-box experience, with a single code-base!

How much is it "worth"?

- $20 million according to the Basic COCOMO model
  - https://www.ohloh.net/p/tikiwiki/estimated_cost

    - "Beyond just development time, COCOMO is meant to include the design, specification drafting, reviewing and management overhead that goes along with producing quality software."

- Yet, the Tiki Software Community Association (TSCA) has no employees
- Tiki doesn't depend on any funding from any company, foundation, government or anyone. It thrives thanks to the community.



Power users

- http://tiki.org/FLOSS+Web+Application+with+the+most+built-in+features
- Profiles: can use, share and build recipes to cater to diverse use cases

Devs

- Easy commit access
- Commit to trunk, stable within 6 months
- Collaborative community
- See http://tiki.org/Framework
- Profiles: can build very specific applications, without custom code

SysAdmin

- Designed for shared hosting (PHP/MySQL)
- LTS versions
    - Tiki9LTS supported until November 2015
    - Can go from LTS to LTS or even skip one LTS (3.5 year support)
- Easy upgrades
- See benefits for http://tiki.org/Hosting+Company

Mass deployments

- Tiki Remote Instance Manager -> TRIM
- In other web apps, if you have 300 projects, you can have up to 300 code bases, because of various extensions (or you have unhappy users because you didn't install their favourite extension). If you have 300 Tiki projects, you have 1 shared code base
    - You can restrict or force features or preferences thanks to the System Configuration tool
- Profiles: can deploy a configuration to hundreds of instances

Next 10 years

- SaaS
- End User Programming
    - Javascript, Smarty and wiki syntax
    - GUI tools
- Become excellent at all the use cases
- Evolve with browsers (HTML5, realtime, offline, etc.)
- Tiki Suite: http://suite.tiki.org

An example of app building: CartoGraf

- Would have been much more expensive to do in-house, even using FLOSS components
- Other Frameworks would have lead to a lot of custom code, or a FLOSS 3rd party extension which would have been hard to maintain
    - Now, it's a one-click profile and a theme

http://profiles.tiki.org/Cartograf