SearchDev

Status/RoadMap

Wishlist

- Many of the features below are not feasible until Tiki's search efficiency is improved drastically. Help wanted!
- We need to resolve the MySQL full text search issue. (**Is this still relevant?**)
- A word finds in a page title must weighs more than if it is found in the regular text.
- Search should take into account categories and category descriptions (and offer categories at the top)
- 404 pages should offer to search the name of the missing page. (useful for page renames) **assigned to: mdavey**
- also, the should be an admin option to receive an email reporting 404 errors
- All features should in search (ex.: we are missing shoutbox as of 2003-12-30)
- Advanced search page with checkboxes to choose all the fields we can search within. By general feature (wiki, articles, etc) but also certain fields of each feature (ex.: article titles, article authors, wiki description, etc)
- Advanced search should include option to search through wiki page history
- Boolean search, maybe with MySQL4searchHack ?
- Respect the users permissions when displaying search results.

Quick fix / starting point for debate - > SearchPermissionsQuickfix

- Possibility to restrict search by category.
- Possibility to restrict seach within a start/end date
- Search within files (.doc, .xls, etc)

New search functionality added by redflo

The mysql fulltext search is one thing that is definitely not DB abstractable (see DbAbstractionDev). So i decided to completely rewrite search functionality in tiki. The idea is based on the php function register_shutdown_function() and reverse key indexes. So let me explain:

1. reverse key indexes
   Imagine you want to search some text in database tables. If you do something like "select pageName from tiki_pages where data like '%some text%'" then the database server has to search the data column on all rows or the table. Assume you have a very big table, then it will slow down the database server. Therefore Mysql and others (oracle i know have) have invented things that help fulltext search in database tables. These build and maintain a reverse key index. That means, they generate a new table with all searchwords and the references to the table row. This table can have additionaly a index on the searchword and therefore there are usually < 5 disk accesses necessary to find all pages that contain a searchword.
   It is very nice, that Mysql and others implemented these things, but they are very different to handle. For Oracle for example, one has to start a extra server for that (or did they change that?). So i decided to create and maintain the reverse key index by myself. Creating is no problem, while maintaining can be a problem. One has several possibilities: Batch scripts that rebuild that index nightly, background processes that monitor changes to content tablöes and finally routines that are called in the tiki code whenever content is changed.
   I use a combination of all three. While the last is easy, but hard work (many changes in code) and batch processing requires a cgi-php, the background processing is a little tricky. See next point
2. register_shutdown_function()
   Whith this function you can register a funtion that is called when the script exits. First i write close the session, and therefore the user cannot get blocked. Then i decide, if a part of the reverse key

index has to be rebuild. I assume it is suffucient to rebuild a part only every 100 pageviews, so n=100. Then it is randomly decided what page in what section (wiki, forum, ...) shall be reindexed. Then it is reindexed. Some times i decide to reindex simply the page that has the oldest index. So the index should be relatively fresh all time.
This ensures, that also people that upgrade or decide sometime to switch on search get a good index after some time.

3. The search itself
   Is much more easy now. No more need to use the fancy mysql specific syntax - just "select * from tiki_searchindex where searchword=...".
4. Extensions
   One remaining problem is the search for parts of words. Assume you want to search for "development" and want to see "TikiDevelopment" as well as "DevelopmentTiki" and others. The "select * from tiki_searchindex where searchword like "%development %'" causes a full table scan and is very slow. For this case we have to add a additional table that saves the searchword and the found searchwords in the tiki_searchindex table. This will speedup additionally. This is made with a LRU list (LRU=last recently used). If someone does a search, then there is done a "select * from tiki_searchindex where searchword like "%development %'" and these results are chached in another table for faster access. In a list (the LRU list) the word "development" is put at the top of the list. If the list is bigger than a maximum length, the last element is purged. If the word "develoment" is already in the list, it is put at the top of the list. This ensures that only the most frequently used words are cached. I added another possibility for a word to exit from the list: if it's for a too long time on the list. This ensures that even frequently used words are "recalculated" sometimes. All parameters are in admin -> search.
5. Status
   A new module "search_new" uses the new search mechanism. Search in wiki pages works. All tested with mysql, Oracle, Postgres, Sybase and all works.

And what about relevance? This is usually the trickiest part to write, to order the results: sylvie

1. Relevance
   thanks sylvie . Relevance can be done with Mindmeld or by simply count the number of occurencies and where the searchword did occur. This can be computed at the time when a reverse key index of a page is recomputed. We also could add a rank like googles page rank, but this is probably very time consuming to compute. We'll have to investigate. Generally this part of a search engine decides more the quality of a search engine than the other parts.

New (experimental) search functionality added by mdavey

Word highlighting is now available on referals from search engines such as Google and Yahoo. When the HTTP_REFERER line includes q= or p= followed by searchwords, these searchwords will be highlighted in the page. There is a small performance hit under such circumstances as the page has to pass through an output filter.

In BRANCH-1-9, edit tiki-setup.php and change

```
    $feature_referer_highlight = 'n';
```

to

```
    $feature_referer_highlight = 'y';
```

around line 234.

New search page layout potions added by mdavey

Since 12th February 2005 in BRANCH-1-9, Tiki PHP files can have rudimentary control over the layout of the search results templates (templates/tiki-searchresults.tpl and templates/tiki-searchindex.tpl). By using $smarty->assign('var', 'value') as follows:

var	**value**	**default**	**description**
searchStyle	button¦menu	button	display sections to search as buttons or a pull-down selection
searchNoResults	true¦false	false	omit H1 title and search results from page
searchOrentation	horiz¦vert¦auto	auto	Use a horizontal or vertical layout, or determine layout based on the other vars (currently ignored - auto is always used)

See the bottom of tiki-searchresults.php and tiki-searchindex.php source files for exact syntax.

TikiTeam

|mdavey - some minor usability improvements

Trackers

Bugs

- {SF(aid=>788425)}{SF} **priority 7**, medium security flaw
- {SF(aid=>783605)}{SF}
- Fields not searched
  - page description
  - articles body using search module (works in the list articles page) *Chealer9 20031025*
  - blog posts title
- Files found have real filename as link title instead of Tiki name.
- FAQ : Searching for FAQs displays multiple results (as many as there are questions/answers for the found FAQ, try to test "Linux" on this site) *Chealer9*
- If by example tiki_p_view is not assigned to anonymous but it is on specific pages, search result (only when searching in **Wiki pages**) is "Permission denied you cannot view this section". This is a minor but possibly confusing bug. *Chealer9*

RFEs

- {SF(aid=>813756,tag=>rfe)}{SF}
- {SF(aid=>665440,tag=>rfe)}{SF}
- {SF(aid=>721265,tag=>rfe)}{SF}
- {SF(aid=>791710,tag=>rfe)}{SF}
- {SF(aid=>802268,tag=>rfe)}{SF} (priority 5)
- {SF(aid=>814666,tag=>rfe)}{SF} (priority 2)
- {SF(aid=>739646,tag=>rfe)}{SF} (priority 5)
- {SF(aid=>792892,tag=>rfe)}{SF} Maybe later...
- a wiki page with a name "MultilingualDev" must be found by a query "multilingual" even if the full search feature is on (idem a query "rss" must find the pages named RSSFeedDev and RssFeedDev.)
- full-text search on the wiki page titles and descriptions only
  - The Search Wiki Page Name module is nice, but it only finds pages if multiple words are entered in the exact order as the page names with no missing words in between

Advanced search wishlist

It's obvious that we could use advanced search features, multiple filtering options. This will be hard to do

with DB abstraction. Let's stil list some trackers, you can add your ideas.

- {SF(aid=>813345,tag=>rfe)}{SF}

Competition and standards

CVS Doc section

Discussion/participation

Some ideas from the devel list

http://php.warpedweb.net/sitesearch/
mwexler: Marc on the dev list mentioned mindmeld as a potential partner. While I like the idea, it does not solve the search problem. Mindmeld requires that the world be structured into thoughts and questions, and that's a bit more structured than tiki. It would be a great module to add to the left nav, however. Perhaps something like Wordindex at sourceforge would be more useful. 2003/09/12
Marc Laporte: interesting. It can index PDFs and other file types, it skips a list of "bad" words, etc But it uses Perl...

Are the comments part of the container or not? If I look for a word that occurs only in a page comment, do the search give back the page? Idem does a found word in a comment change the relevance of the page?
Marc Laporte thinks yes

PHP Search Engine (SEARCpHp)

http://www.hansanderson.com/php/search/
broubrou: I suggest using SEARCHpHp instead of writing the whole search thing from scratch. It's an already working solution and it's all php-based. From what I understand of the problem, SEARCpHp does pretty much everything you need.

- Index is completely portable, copy it to any machine using PHP and SEARCpHp, Unix or other
- You can index from file system, databases, the web, whatever. If you can access it in PHP, you can index it
- It's Free, and always will be, under the same license as PHP itself

phpMySearch

http://phpmysearch.web4.hm/index.php
mwexler: I stumbled on this one while seeing what people thought of SEARCHpHp. Support is commercial, but code is open source. Neither of these had overwhelming crowd support, but either is probably better than starting from scratch.

mnoGoSearch

http://www.mnogosearch.ru/
mwexler: Another one I stumbled across. Commcercial for Windows, but free for Perl and PHP versions. Same comments as above.

Deno: of the three above mentioned solutions, the mnoGoSearch is the only one I saw used in a "real life" situation. It's used by mysql.com, and it works quite well. Interesting part is that there isn't really a need to fully integrate mnoGoSearch into tiki - all that needs to be done is a good documentation page explaining how to fully disable the tiki built-in search, and how to get the most out of mnoGoSearch+tiki combination.

mdavey's notes

[+]