TOETAG
Tagged Object Environment

Bill Ward
and
Holly Ward

http://toetag.sf.net

What is Toetag?

- Toetag is a platform for building applications that organize content using tags.
  - Provides storage, tagging, and retrieval functionality.
  - Provides a generalized entity/ relationship mechanism for building any kind of application.
  - Highly flexible & extensible.
- Current project status: in design phase, based on earlier work. Sourceforge project "toetag" registered – code coming soon!

Possible Applications

- Bookmark site (del.icio.us): store URL as content and let users tag them
- Flickr: store images
- Wiki: store text with versioning
- Personal document organizer: find your files using tags instead of a maze of twisty little subdirectories, all alike
- Corporate environment: group documents

Genesis of Toetag

- Usenet newsgroup rec.humor.funny has been putting “keywords” on jokes since the 1980's
- Fall 2000: first version of Hatrack Moose: organized by “categories”
  - Site was never launched due to copyright issues

Hatrack Moose 2005

- Spring 2005: we discovered del.icio.us and Flickr – tagging!
- Realized that Hatrack Moose's “categories” were similar to tags
- Summer 2005: Dusted off Hatrack Moose code and updated to a more modern look
- Toetag – based on generalizing the Hatrack Moose design

How does it work?

- Toetag has two kinds of entities
  - Objects: content (defined by application), users, tags, and app-specific types
  - Relationships (tags)
    - Link two objects together
    - Optionally specfy a type of relationship
    - Allows tagging + much more

Tags as Objects?!!??!

- When you tag something it creates an object entry for the tag (if there isn't one already)
- Why do this?
  - You can tag tags
  - You can use other types of objects as tags

Tagging Tags

- Group together related tags
  - Tags like “blogging,” “technorati,” “blogs,” and “blogosphere” can all be tagged “blog” for easy retrieval
  - Tags like “toread” on del.icio.us could be tagged “usefultag”

Other types of objects as tags

- Reduces need for metadata about objects
  - Tag a document with a user (e.g., “user=hollyward” automatically instead of having a user id field
- Represent other kinds of relationships
  - Tag one user with another (Friends)
  - Tag a document with another document (e.g., A document may be tagged with another document that summarizes it)

Format of a Tag

- The full syntax of a tag is:
  - *objecttype operator name . reltype*
- *Where:*
  - *objecttype* – type of object used as tag (default “tag”)
  - *operator* – usually = or != (default “=”)
  - *name* – name of tag
  - *reltype* – type of relationship (default “tagged”)

Object Types

- Out of the box the only object types are “user” and “tag”.
- Others can be added for specific applications
- Examples:
  - Bookmark management site: “url”
  - Photo-sharing tool: “photo”
  - Document app: “document”
  - Etc.

More Object Types

- Object types can also be used to implement special features of applications
  - Groupware application: “project” used to associate some documents with a set of users
  - Access permissions: “access” to tag the users with a special tag “access=superuser”, “access=guest”, etc. which lets them do certain things in the application

Relationship Types

- Describe type of relationship
  - Author of a document could be tagged "user=hollyward.authored". Another user reviews the document and it is tagged "user=billward.reviewed"
- Default tag type is "tagged"
  - A folksonomy tag "foo" is short for "tag=foo.tagged"

Relationship Types & Searching

- A search without giving a relationship type will return all tags regardless of their relationship type
- Example: Assume document X has tag “user=billward.authored” and document Y has tag “user=billward.reviewed”
  - Search for “user=billward” will return both
  - Search for “user=billward.authored” will return only document X

Tags and Anti-tags

- To search for content that *doesn't* have a certain tag use ! or != instead of = in the tag.
- Examples:
  - Search for tags “tag=tagging tag!=grafitti” (or just “tagging !grafitti” for short) to find folksonomy without spray paint
  - Find documents on lizards not written by Holly with “lizards user!=hollyward”

Magic Object Types

- Some tag object types don't correspond to actual objects
  - Dates: date=2005-10-27.created
  - Versions (for content under revision control)
  - Others??
- In these cases the operators > < >= and <= are also allowed:
  - date>27-Oct-2005-18:00:00.updated

Possible Applications

- Web-based apps
  - Social apps like del.icio.us, Flickr, new version of Hatrack Moose?
  - Bulletin boards, Craigslist, blogs, wikis
- Personal or group/corporate document organizer
  - Use tagging instead of a tree of subdirectories
  - Use tagging to record comments about documents
- Other ideas???

Virtual filesystem driver

- We could make a special filesystem that does automatic interaction with Toetag server
- Examples:
  - Directory “japan/lizards” or “lizards/japan” would include all documents tagged “japan” and “lizards”
  - Drag files to another “directory” to add a tag to them
- Accessible using Samba or FTP

Versioning

- To implement a wiki you would need to record versions
  - Toetag could even act as a full change control system
  - RCS backend would track changes from one version to the next
  - Apply tags to specific versions or ranges of versions

Implementation

- Perl, MySQL, SOAP::Lite
- Run as TOAP server (Tagged Object Access Protocol) and talk to it using standard SOAP tools
- Or link the Perl module into your application (mod_perl under Apache if Web-based)

Conclusion

- Toetag will enable rapid development of all kinds of tagging applications
- The possibilities are endless, limited only by our creativity & imagiation

- Join the development effort!
- Visit http://toetag.sf.net