Mike Hahn – hahnbytes@gmail.com

Lyvathon

Lyvathon is a platform that hosts and monetizes web apps, and includes 2 brand-new open source languages: Lyvathon (similar to Python), and LVM (similar to HTML). Eventually Lyvathon-based web apps will be hosted at Lyvware.com. Lyvathon is implemented using Python, Flask, and MySQL. Future versions of Lyvathon include a search engine which indexes LVM body text.

Language Features

Lyvathon is a statically typed Python work-alike (with some features of Java), but has Lisp-like syntax. All program code in memory is stored in 12-byte nodes, and arrays are implemented using binary trees or linked lists. Lyvathon Markup Language (LVM) includes tags enclosed in brace brackets. The tag name (and any optional arguments) is terminated by a vertical bar (|), in case there is any body text prior to the closing brace bracket. Some tags include multiple blocks of body text separated with vertical bars (the left vertical bar of each block is preceded by optional arguments). Lyvathon is used for both client-side and server-side code (client-side code is converted to JavaScript at compile time, and LVM is converted to HTML at run time).

Business Model

All end-users are either subscribers or non-subscribers. Subscribers pay $20/year for the privilege of accessing features and apps unavailable to non-subscribers. Some end-users are also employees (team members). Companies which employ team members pay data usage fees and are users of team-based apps. The data usage fee is proportional to the weighted sum of 3 counters maintained for each team member: no. of megapixels (in images downloaded), no. of Lyvathon/LVM tokens, and no. of new Lyvathon 12-byte nodes created. All counters are reset to zero at the end of each month.

Lyvathon users who are owners of web apps receive 50 percent of the data usage fees. Owners of consumer apps (as opposed to team-based apps) receive 50 percent of the data usage fees utilized by subscribers when accessing those consumer apps. Owners of team-based apps receive 50 percent of the data usage fees utilized by employees (team members) when accessing those team-based apps. All app owners pay app hosting fees of $10 per year per public app (or just $2 per year for the 6th and each subsequent public app). Every team-based app owner pays an additional amount equal to that owner's share of the data usage fees utilized by non-subscribers when accessing consumer apps. All app owners who choose to make use of Google AdSense ads in their apps receive 100 percent of that revenue.

About Me

I am Mike Hahn, the founder of Lyvware.com. On August 9, 2014 I began working on AppaTeach (a tutoring website). I started developing Lyvathon on January 4, 2015, after dabbling in its precursors (sporadically) since the mid 90s. On January 24, 2015 I resumed working on AppaTeach, and resumed working on Lyvathon on May 9, 2015. I started designing Lyvware on August 18, 2015, and implementing it 12 days later. I was previously employed at Brooklyn Computer Systems as a Delphi Programmer and a Technical Writer (I worked there between 1996 and 2013). Just prior to starting Lyvathon I quit my job as a volunteer tutor at Fred Victor on Tuesday afternoons, where for 5 years I taught math, computers, and literacy. I'm now a volunteer computer tutor at West Neighbourhood House. My hobbies are reading the news at cbc.ca and going for walks in my neighbourhood. About twice a year I get together with my sister who lives in Victoria. She comes here or I go out there usually in the summer. At those times I'm much more active, but most of the year I tend to lie on the couch a lot, and not be very active. I do, however, visit my brother once a month or so and listen to or visit my disabled friend (she has schizophrenia and talks to me on the phone).

Data Usage Fees

Let $s_1[i]$ = megapixel count for subscribers of i-th app

Let $s_2[i]$ = token count for subscribers of i-th app

Let $s_3[i]$ = node count for subscribers of i-th app

Let $n_1[i]$ = megapixel count for non-subscribers of i-th app

Let $n_2[i]$ = token count for non-subscribers of i-th app

Let $n_3[i]$ = node count for non-subscribers of i-th app

Let $e_1[i,j]$ = megapixel count for employees of i-th team app, j-th team

Let $e_2[i,j]$ = token count for employees of i-th team app, j-th team

Let $e_3[i,j]$ = node count for employees of i-th team app, j-th team

Let a_1 = megapixel weight factor

Let a_2 = token weight factor

Let a_3 = node weight factor

Let s[i] = data usage fee per month, for subscribers of i-th app

Let n[i] = data usage fee per month, for non-subscribers of i-th app

Let e[i,j] = data usage fee per month, for subscribers of i-th app, j-th team

Then $s[i] = a_1s_1[i] + a_2s_2[i] + a_3s_3[i]$

And $n[i] = a_1n_1[i] + a_2n_2[i] + a_3n_3[i]$

And $e[i,j] = a_1e_1[i,j] + a_2e_2[i,j] + a_3e_3[i,j]$

Let k = no. of teams

Let m = no. of consumer apps

Let n = no. of team apps

Let C = total costs per month

Let E = team-based costs per month

Then E = SUM(j=1,k: SUM(i=1,n: e[i,j]))

And C = SUM(i=1,m: s[i] + n[i]) + E

Let p = public app fee = $10/year

Let r = subscriber rate = $20/year

Let q = no. of subscribers

Then qr/12 = SUM(i=1,m: s[i])

Let t[j] = revenue from j-th team

Then SUM(j=1,n: t[j]) = SUM(i=1,m: n[i]) + E

And t[j] = SUM(i=1,n: e[i,j])(1 + (SUM(i=1,m: n[i]) / E))

Let u[i] = cost of i-th consumer application, per month

Let v[i] = cost of i-th team app, per month

Then u[i] = (0.5)s[i]

And v[i] = (0.5)SUM(i=1,n: SUM(j=1,k: e[i,j]))

Database Tables

Users

- usrid
- empid
- flags:
    - active
    - subscriber
    - employee
    - developer
- username
- firstname
- lastname
- password
- email
- altemail
- phone
- question1,2
- answer1,2
- customq
- balance
- joindate
- expirydate
- leavedate
- wordcount
- pixelcount
- nodecount

Emps

- empid
- usrid
- phone
- cell
- fax
- email
- web
- addr1
- addr2
- city
- state
- country
- postalcode

Orgs

- orgid
- temid
- name
- tradename

Teams

- temid
- orgid
- parid
- nextid
- childid
- empid
- supid
- name

Projects

- prjid
- temid
- appid
- name
- title
- descr
- isapp

Uorg

- usrid
- orgid
- startdate
- active

Uteam

- usrid
- temid
- title
- rank
- startdate
- active

Uproj

- usrid
- temid
- prjid
- title
- rank
- startdate
- active

Contact Info

Mike Hahn
Founder, Lyvware.com
515-2495 Dundas St. West
Toronto, ON M6P 1X4

Country: Canada
Phone: 416-533-4417
Email: hahnbytes (AT) gmail (DOT) com
Blog: lyvathon.blogspot.ca

Lyvaic

Lyvaic is the first sample Lyvathon project: an image-collection manager. Image files are stored in folders, which may contain other folders recursively, and are displayed in mosaic form (like a grid). Clicking on an image enlarges it to full size. Slides containing multiple images can be created and grouped into slide shows. Images can have captions and folders can have descriptions (folders about a person include that person's name). Images in slides have a zoom factor (zoom in or out). Images consist of either files on the server or links to external sites.

Public and Private

Every user has zero or more followers. Every user has a static tree of slides, and each folder in the tree can be public, followers only, or a specific follower.

Lyvaic Modes

1. **Mosaic** - grid of images/folders belonging to a folder
2. **Slide-Grid** - grid of slides, 1 to 3 images side-by-side, terminated by a vertical black line on a white background
3. **Mono** - single full-size image, possibly filling available screen space
4. **Slide** - group of 1 to 3 images, side-by-side, filling available screen space

Commands

Z - Undo
X - Cut
C - Copy
V - Paste
D - Delete
T - Toggle Slide/Mono or Slide-Grid/Mosaic
N - Insert new image/folder
E - Edit properties:
    Folder - folder properties
    File,
    Mono - images (path, filename properties)
K - Show/hide kaption
S - Search images
Q - Quit (logout)
Shift+Z - Redo
Shift+C - Select/deselect multiple
Shift+K - Show/hide all kaptions
Minus (-) - Zoom out
Plus (+) - Zoom in
Equals (=) - Zoom in
Zero (0) - Cancel zoom
F1 - Rotate: help page, hide, show menu (default)

Arrow - Select in grid
Left Arrow - Previous mono/slide
Right Arrow - Next mono/slide
Up Arrow - Previous image in slide
Down Arrow - Next image in slide
Click or Enter:
    Folder - expand folder
    File - go to Mono
    Slide-Grid - go to Slide
    Mono - go to Mosaic/Slide-Grid
    Slide - go to Mono
Esc - go to parent folder

Lyvaic Database

Users

- usrid
- ctgid

Folders

- ctgid
- usrid
- parid
- nextid
- childid
- filid
- sldid
- imcid
- name
- descr
- public
- flwid

Files

- filid
- ctgid
- usrid
- nextid
- imgid

Images

- imgid
- usrid
- imcid
- width
- height
- isurl
- path
- filename
- caption

Followers

- id
- srcid
- dstid

Clicks

- id
- imgid
- year
- month
- clkcount

Slides

- sldid
- usrid
- ctgid
- nextid
- previd
- celid

Cells

- celid
- sldid
- nextid

Imghdrs

- imhid
- imgid
- zoom
- left
- top

Imgclasses

- imcid
- usrid
- parid
- nextid
- childid
- clsname

Imgfields

- imfid
- imcid
- fldno
- fldname

Imgvalues

- id
- imfid
- imgid
- isfolder
- value

Note: all database tables (except followers) need a usrid field.

LVM Format

Heading	`==, ===, ...`
Bold/Italics/both	`'', ''', ''''`
Numbered List	`#, ##, ...`
Bulleted List	`*, **, ...`
Container Tag	`{ ... | ... }`
Table/Grid/Tag	`{ ... }`
Open Row	`{row`
	`{row fld=val`
Close Row	`}`
Open Column	`||`
	`|fld=val|`
	`|f1=v1;f2=v2;...|`
Vert. Grid Line	`\\|`
Horiz. Grid Line	underscore (_)
Grid Intersection	plus (+)
Escape Char.	backslash (\)
Lyvathon Code	`{% ... %}`
Lyvathon Expr.	`{{ ... }}`

Tags:

- table, row, grid, dtab, meta
- super, sub, text, pre, br, hr, img, a, ch
- input, radio, checkbox
- styles, include, h1..h5, b, i, u, ol, ul

Fields:

- width=50/0.5 (pixels/ratio)
- pad=50/0.5
- x, y = 50/0.5
- height = n (pixels)
- topb=1 (pixels)
- bottomb, leftb, rightb, midb = 1
- color=FF00FF (rgb)
- fcolor=00FF00 (text)
- bcolor=000000 (borders)
- colspan, rowspan = n
- just="L/C/R"
- b, i, u (bold, italics, underline)
- same (same as previous)
- vis (visible)
- coldefs
- rows, cols = n (grid size)
- id="mynode", id="mytag"

Implementation Steps

1. Lyvaic:
    1. Finish user authentication
    2. Implement image collection manager
    3. Post help-wanted notice on SourceForge
    4. Implement Lyvathon (console mode)
    5. Combine Lyvathon with Flask
    6. Convert image collection manager from Python to Lyvathon
2. Lyvathon:
    1. Flask-based SDK written in Python using Linux
    2. Assembler: LVA >> LVC
    3. LYvathon Runtime Environment (LYRE): run LVC code
    4. Compiler: LV >> [LVA] >> LVC
    5. Support try stmts.
    6. Converter: LV >> JS (client-side version)
    7. Lyvathon software is open source
3. LyvAIDE:
    1. LyvAthon Integrated Development Environment
    2. Written in Python using Linux/Windows
    3. LVM Rendering >> display UI
    4. LVM/LV Editor
    5. GUI Editor: display UI >> LVM
    6. Syntax Highlighting
    7. Unlimited Undo/Redo
    8. Debugger
    9. Monetization: sell web hosting for sites written in Lyvathon and LVM
    10. LyvAIDE software is open source

Join Our Team

The open source Lyvathon project needs volunteer Python programmers to contribute to the project. Here is a list of tasks which are all implemented using Python (except Step 3). The first task is reserved for Mike, the Founder of Lyvathon.org.

1. Lyvathon Compiler
2. Converter: LVM >> HTML
3. Write LVM full specs
4. LVM Code Editor
5. LVM WYSIWYG Editor
6. LyvAIDE: fancier Lyvathon Code Editor
7. Convert image collection manager: Python >> Lyvathon
8. Monetization
9. Search Engine: indexes LVM body text
10. Apache/MySQL Integration

If you wish to contribute to this project, please contact me. If you have any preferences as to which of the above tasks you wish to work on, please indicate up to 3 choices ranked from most preferred to least preferred. Thank you for your interest in the Lyvathon project.