playgen:

A playlist generator for streaming services like LiquidSoap

presented by

Mark Jeghers

playgen overview

- A background data service that generates automatically randomized playlists
- I've been using it with liquidsoap for several years now.
- playgen is written in JavaScript and uses NodeJS and Express technologies
- https://github.com/jeghers/playgen

Full set of features…

- Completely randomized song selection unattended 24/7
- Based on NodeJS/Express technologies
- Highly RESTful API
- Can be accessed programmatically with HTTP requests (including with 'curl')
- Supports multiple playlists (e.g. for multiple program streams)
- Randomizes song playback with advanced features
  - Optimize randomization logic to insure maximum "fairness"
  - No song ever repeated until complete song list is gone through
  - Configurable deferral of duplicate song titles
  - Configurable deferral of duplicate song artists

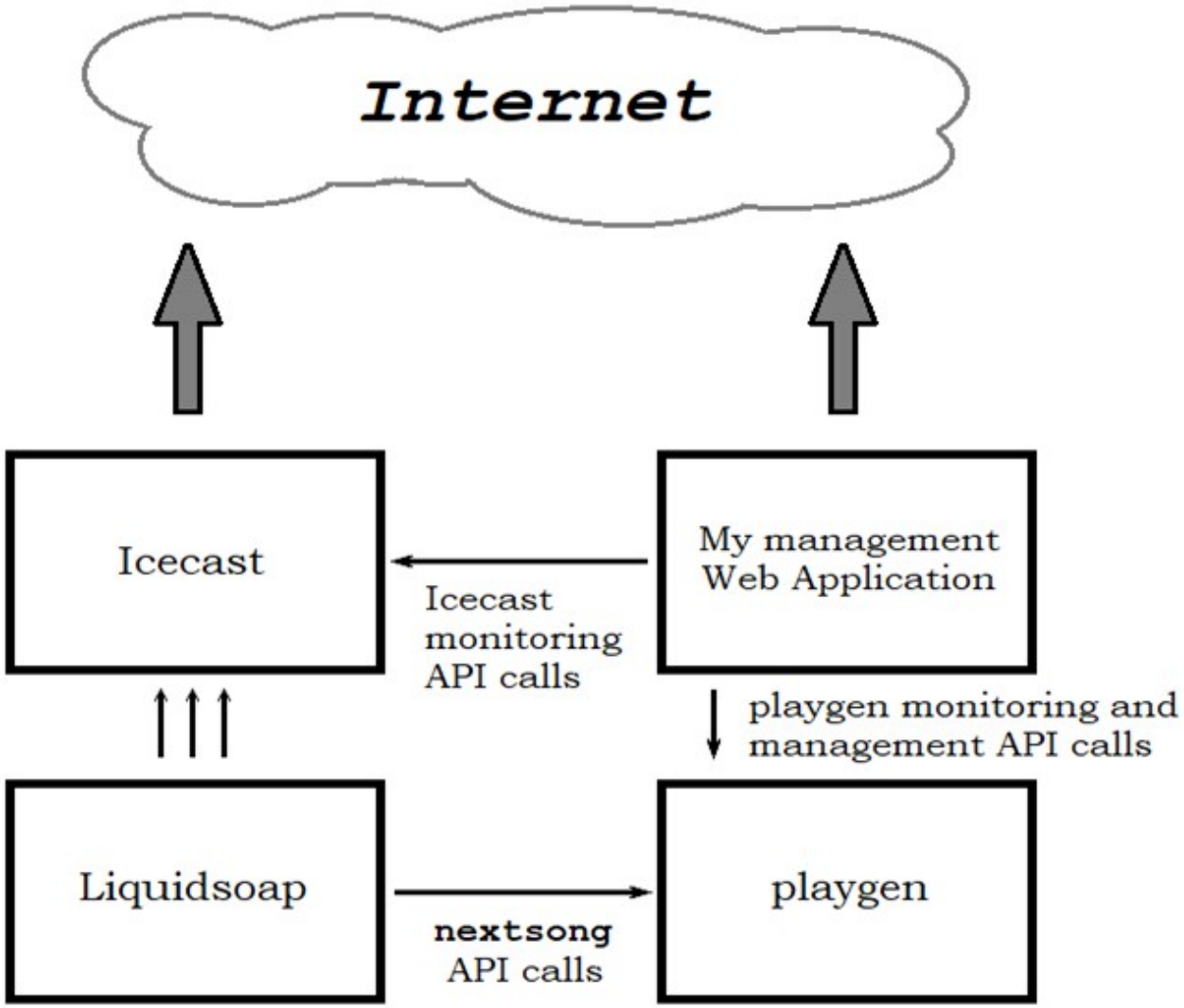
Full set of features…

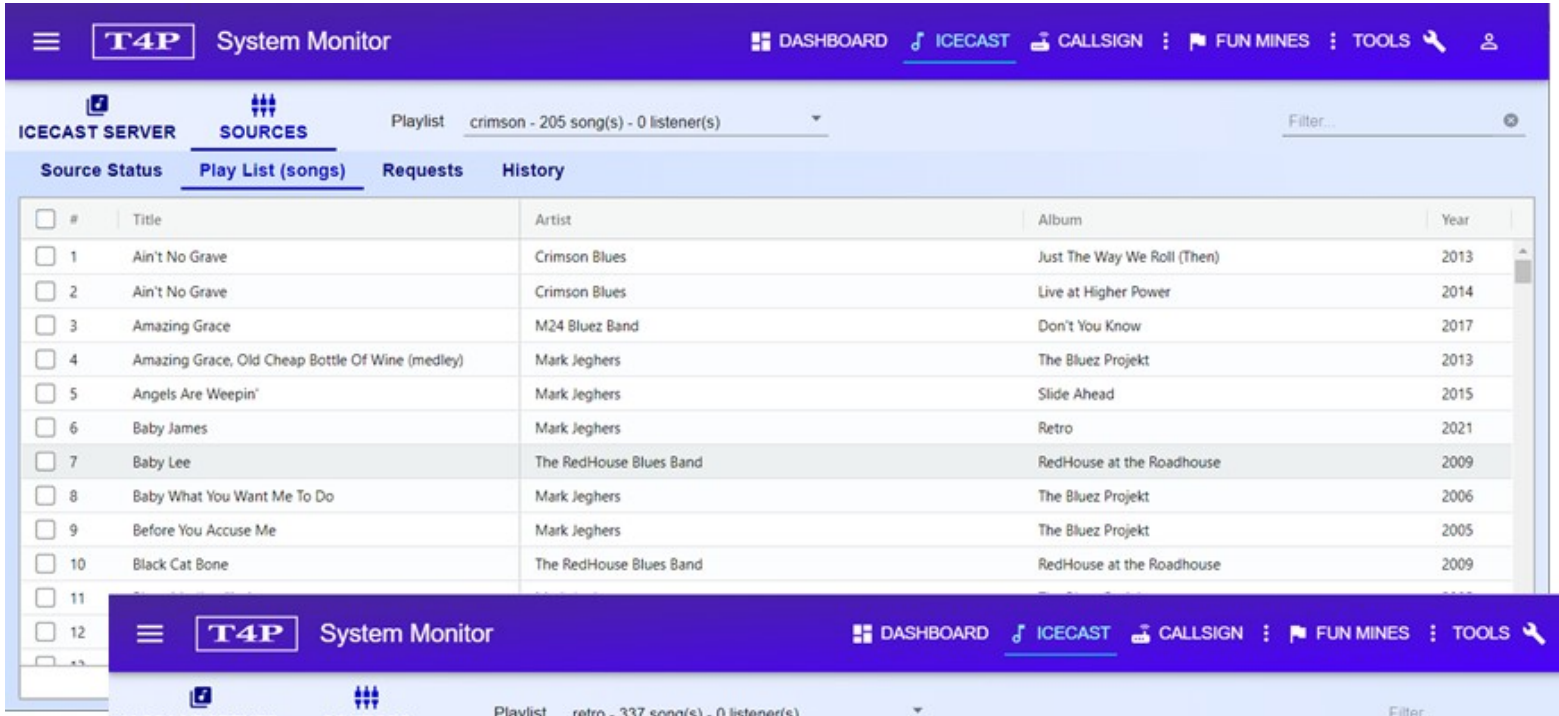
- Automatic songlist generation for a directory full of song files
- A new request feature allows a queue of specified song(s) to be unconditionally played next after the current song completes
- Uses MySQL to store playlist options (the songs themselves are listed in text files)
- New plugin architecture allows customized logic for logging and song details extraction
- I've not implemented HTTPS because I assume this will be used behind a firewall anyway. Feel free to make such an enhancement and submit a PR.

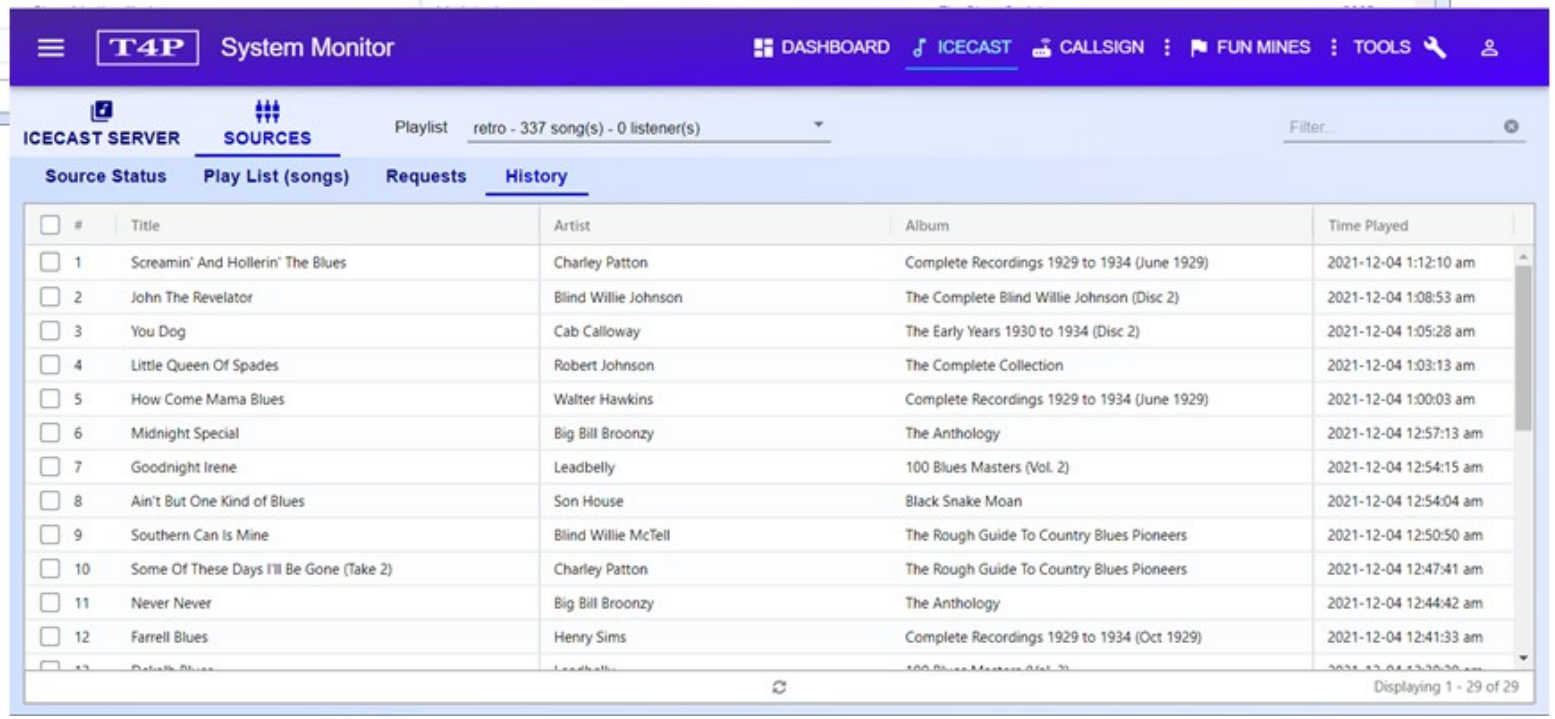
Full set of features…

- New support for Windows 10 service (beta)
- In addition to random song selection, playgen can provide song list and status information to other webapps.
- My web portal shows me the full status of my Liquidsoap/Icecast system
  - including displays of all the playlists, song requests and histories
  - playgen provides much of this data.

Part of my Liquidsoap/Icecast "BluesWire" streaming solution…




T4P
System Monitor
DASHBOARD
ICECAST
CALLSIGN
FUN MINES
TOOLS
ICECAST SERVER
SOURCES
Playlist
crimson - 205 song(s) - 0 listener(s)
Filter...
Source Status
Play List (songs)
Requests
History
Title Artist Album Year
1 Ain't No Grave Crimson Blues Just The Way We Roll (Then) 2013
2 Ain't No Grave Crimson Blues Live at Higher Power 2014
3 Amazing Grace M24 Bluez Band Don't You Know 2017
4 Amazing Grace, Old Cheap Bottle Of Wine (medley) Mark Jeghers The Bluez Projekt 2013
5 Angels Are Weepin' Mark Jeghers Slide Ahead 2015
6 Baby James Mark Jeghers Retro 2021
7 Baby Lee The RedHouse Blues Band RedHouse at the Roadhouse 2009
8 Baby What You Want Me To Do Mark Jeghers The Bluez Projekt 2006
9 Before You Accuse Me Mark Jeghers The Bluez Projekt 2005
10 Black Cat Bone The RedHouse Blues Band RedHouse at the Roadhouse 2009
11
12


T4P
System Monitor
DASHBOARD
ICECAST
CALLSIGN
FUN MINES
TOOLS
ICECAST SERVER
SOURCES
Playlist
retro - 337 song(s) - 0 listener(s)
Filter...
Source Status
Play List (songs)
Requests
History
Title Artist Album Time Played
1 Screamin' And Hollerin' The Blues Charley Patton Complete Recordings 1929 to 1934 (June 1929) 2021-12-04 1:12:10 am
2 John The Revelator Blind Willie Johnson The Complete Blind Willie Johnson (Disc 2) 2021-12-04 1:08:53 am
3 You Dog Cab Calloway The Early Years 1930 to 1934 (Disc 2) 2021-12-04 1:05:28 am
4 Little Queen Of Spades Robert Johnson The Complete Collection 2021-12-04 1:03:13 am
5 How Come Mama Blues Walter Hawkins Complete Recordings 1929 to 1934 (June 1929) 2021-12-04 1:00:03 am
6 Midnight Special Big Bill Broonzy The Anthology 2021-12-04 12:57:13 am
7 Goodnight Irene Leadbelly 100 Blues Masters (Vol. 2) 2021-12-04 12:54:15 am
8 Ain't But One Kind of Blues Son House Black Snake Moan 2021-12-04 12:54:04 am
9 Southern Can Is Mine Blind Willie McTell The Rough Guide To Country Blues Pioneers 2021-12-04 12:50:50 am
10 Some Of These Days I'll Be Gone (Take 2) Charley Patton The Rough Guide To Country Blues Pioneers 2021-12-04 12:47:41 am
11 Never Never Big Bill Broonzy The Anthology 2021-12-04 12:44:42 am
12 Farrell Blues Henry Sims Complete Recordings 1929 to 1934 (Oct 1929) 2021-12-04 12:41:33 am
Displaying 1 - 29 of 29

Prerequisites

- NodeJS (at least version 12 recommended)
- MySQL (preferably running on localhost)
- Runs 24/7 on a CentOS 7 server
- Should run well on other reasonable Linux flavors
- Routinely tested on Windows 10

How the random selection works

A simple selection of a random song from a list has several problems:

- Same song could get selected again while other songs are never selected at all.
- If there are many songs of the same title (different performances of the same song), the same song could occur again soon after it was already heard.
- More than once song from the same artist could be selected close to one another.
- There is no guarantee that all songs in the list get an equally fair chance to be selected.

playlist songs

song 1
song 2
song 3
song 4
song 5
song 6
song 7
song 8
song 9
song 10
song 11
song 12
song 13
song 14
song 15
song 16
song 17
song 18
song 19
song 20
song 21
song 22
song 23
song 24

Purely random selections...

song 11
song 6
song 20
song 6
song 18
song 3
song 11
song 6
song 17
song 1
song 8
song 7
song 6
song 21
song 9
song 6
song 7
song 5

Oops!

*** some songs were selected much too often**

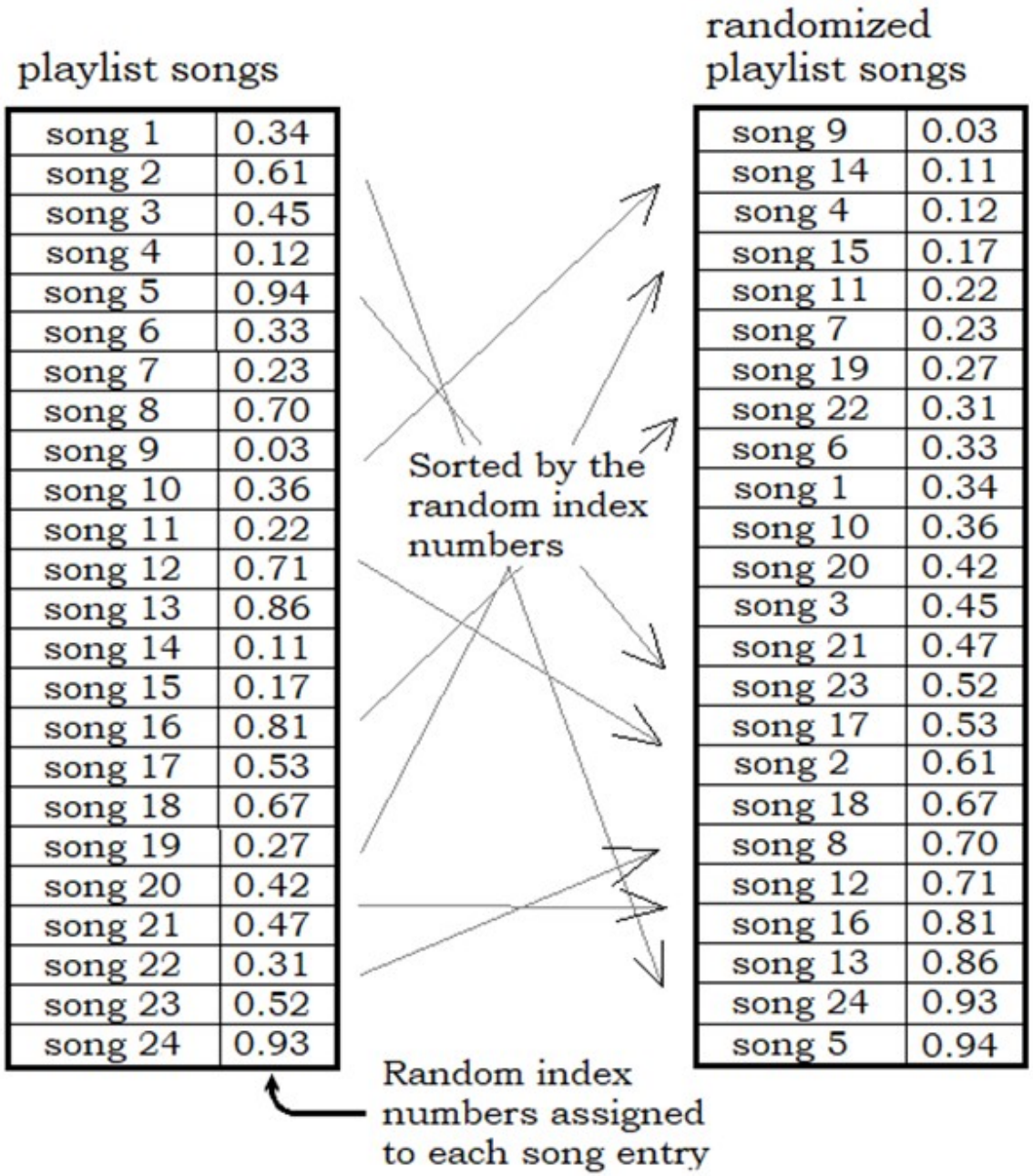
*** some songs were not selected at all**

*** I'll bet some songs were duplicate titles too!**

How the random selection works

Playgen uses an algorithm for random song selection that overcomes all these problems

- Each song in the playlist is assigned a random number index.
- The list is sorted according to those numbers, resulting in a uniquely random sequence.
- By iterating through the list according to the random indices, every song is now selected with "equal fairness".
- Over time, every song will eventually get selected


playlist songs
song 1 0.34
song 2 0.61
song 3 0.45
song 4 0.12
song 5 0.94
song 6 0.33
song 7 0.23
song 8 0.70
song 9 0.03
song 10 0.36
song 11 0.22
song 12 0.71
song 13 0.86
song 14 0.11
song 15 0.17
song 16 0.81
song 17 0.53
song 18 0.67
song 19 0.27
song 20 0.42
song 21 0.47
song 22 0.31
song 23 0.52
song 24 0.93
Sorted by the random index numbers
randomized playlist songs
song 9 0.03
song 14 0.11
song 4 0.12
song 15 0.17
song 11 0.22
song 7 0.23
song 19 0.27
song 22 0.31
song 6 0.33
song 1 0.34
song 10 0.36
song 20 0.42
song 3 0.45
song 21 0.47
song 23 0.52
song 17 0.53
song 2 0.61
song 18 0.67
song 8 0.70
song 12 0.71
song 16 0.81
song 13 0.86
song 24 0.93
song 5 0.94
Random index numbers assigned to each song entry

How the random selection works

- The playlist can be configured to have a "redundant title threshold". A song with duplicate title will be moved to a later point in the list (postponed).
- The playlist can be configured to have a "redundant artist threshold", allowing songs of the same artist to be postponed as well.
- The random selection is bypassed if song requests are added to the request queue.
  - Those songs go to the “front of the line”.
  - After the request queue is depleted, then the random selection will resume.

How to deploy playgen

- Install a reasonably new version of NodeJS, at least version 12 or greater
- **git clone** the software into a fresh new directory
- Run **npm install** to download dependancy modules
- Edit **config/default.js** to configure your MySQL database
- Create the '**playlists**' table in the MySQL database

How to deploy playgen

- Populate the 'playlists' table with your desired playlist(s)
  - The "Playlist Configuration" section in documentation will guide you.
- Use “npm run start” to run the playgen service
- The playlist generator can be accessed with API calls like these:

GET http://myhost:3000/api/playlists
GET http://myhost:3000/api/playlists/myplaylist
GET http://myhost:3000/api/playlists/myplaylist/nextsong
GET http://myhost:3000/api/playlists/myplaylist/currentsong

Playlist configuration

In the 'playlists' table, each row describes a playlist.

- **name**: the id for the playlist
- **filePath**: text file naming all the songs in the playlist, or a directory full of song files
- **description**: a human-readable description of the playlist
- **redundantTitleThreshold**: how soon another song with an identical name is allowed
- **redundantArtistThreshold**: how soon another song with an identical artist is allowed
- **partialTitleDelimiters**: a collection of characters that will be shorten duplicate title comparisons
- **songDetailsPluginName**: optional customization of song details extraction

Playlist configuration

partialTitleDelimiters allow similar titles to be considered duplicate:

- Example: **`partialTitleDelimiters = "(,"`**
- These titles would all be treated as identical
  - "Sing Along“
  - "Sing Along (live)“
  - "Sing Along, Sing With Me (medley)“
- They match because their first delimited sections are identical

Playlist filenames

- Details about the songs are embedded into the songs filenames
- Well-defined fields that denote title, artist, etc
- By having this metadata in the filenames, the song files themselves can be treated as totally content-agnostic, avoiding expensive parsing
- Example: **`title-artist-album-label-year.mp3`**

Playlist filenames

For example: an MP3 file contains the song "Pearline" by "Son House", from an album "The Original Delta Blues" released by "Columbia Legacy" in 1965:

- title: Pearline
- artist: Son House
- album: The Original Delta Blues
- label: Columbia Legacy
- year: 1965

- The filename should be:

`Pearline-Son House-The Original Delta Blues-Columbia Legacy-1965.mp3`

Playlist filenames

What if your files don’t match this naming convention?

- You can make an automated shell script program to take song files and convert them to this naming convention. That will save you a lot of bother over time.
- Using the new plugin architecture, a provided “mp3Tags” plugin can be used to parse MP3 tags for song details (but slower startup)

Playlist filenames

My own automatic conversion script uses…

- A data file maps all my MP3s into their desired standard names, and my shell script uses this file…

```
/var/www/t4p.com/public_html/blues/cakewalk:/usr2/Blues:Ac
oustic Medley:If I Leave This World Tomorrow, One Way
Ticket (medley)-Crimson Blues-Live at Bernal-T4P Music-
2012

/var/www/t4p.com/public_html/blues/cakewalk:/usr2/Blues:Sa
die:Sadie-Mark Jeghers-The Bluez Projekt-T4P Music-2013
```

```
Field 1 - the source directory
Field 2 - the destination to copy the file to
Field 3 - the file to copy over
Field 4 - the standardized name to rename it to
```

Customizing with plugin architecture

- Playgen now allows customized Javascript code to be added as “plugins”
- Allows customization for logging and song details extraction
- Sample plugins provided:
  - Logging to console, local file, or rsyslog server
  - Song details extracted from filename or MP3 tags
  - Boilerplate samples provided for developers

Using playgen with liquidsoap

- To get a new song selection, we call this playgen API…
  - **`/api/v1/playlists/retro/nextsong?format=text`**
- Using the Linux “curl” command to call the API
- By default, the API returns JSON data
- The option “**`format=text`**” returns the next song file as plain text
  - This is easier for Liquidsoap to consume

Using playgen with liquidsoap

- Sample .liq script

```
playlistJingles = audio_to_stereo(playlist("~liquidsoap/playlists/jingles-playlist.txt"))

def retro_request_function() =
  result = get_process_output
    ("curl 'http://localhost:3000/api/v1/playlists/retro/nextsong?format=text'")
  # in V2 use process.read instead
  # in Windows, you might have to remove the single-quotes from the above command
  log("Next song "^result)
  request.create(result)
end

# ... more similar request functions here for other sources ...

# Create the sources
plr = request.dynamic(retro_request_function) # <-- this is where we call playgen
# Play a station ID jingle after every fourth song
plr = rotate(weights=[1,4], [playlistJingles, plr])

# ... more sources created here ...

output.icecast(
        %mp3(bitrate=48, id3v2=true),
        host="localhost", port=8000, password="some-password",
        mount="/retro", genre="Old Pre-War Blues",
        description="Retro Pre-war blues all the time",
        url="http://www.yourdomain.com:8000/retro", mksafe(plr))

# ... more sources connected to icecast here ...
```

The REST API for playgen

- Given: a host named **somehost** and port **3000**

GET http://somehost:3000/api/v1/playlists

Returns a list of all playlists

Sample response body


```
{
  "status": "OK",
  "result": [
    {
      "name": "johnson",
      "filePath": "/usr/local/nodeapps/playgen/playlists/johnson-playlist.txt",
      "description": "Songs by Robert Johnson",
      "redundantTitleThreshold": 0,
      "partialTitleDelimiters": "",
      "redundantArtistThreshold": 0,
      "songCount": 29,
      "uri": "http://192.168.0.248:3000/api/v1/playlists/johnson"
    },
    ... etc etc etc ...
    "count": 6
}
```

The REST API for playgen

- More API examples
- **POST** **http://myhost:3000/api/v1/playlists**
  - Creates a new playlist
- **PUT** **http://myhost:3000/api/v1/playlists/{playlist}**
  - Updates a playlist (except for song details)
- **DELETE** **http://myhost:3000/api/v1/playlists/{playlist}**
  - Deletes a playlist (song files are not removed)

The REST API for playgen

- Songs within a playlist
- **GET http://somehost:3000/api/v1/playlists/{playlist}/songs**
- Returns a list of all the songs in the specified playlist
- **GET http://somehost:3000/api/v1/playlists/{playlist}/songs/{songIndex}**
- Returns a specified song in the specified playlist

The REST API for playgen

- Song requests within a playlist
- **POST** **http://somehost:3000/api/v1/playlists/{playlist}/requests**
  - Creates a new song request for the specified playlist
  - *Sample request body*

```
{
   "songIndex": 22
}
```

- **GET** **http://somehost:3000/api/v1/playlists/{playlist}/requests**
- Returns a list of all the song requests for the specified playlist

The REST API for playgen

- Song selection (most important!)
- **GET http://somehost:3000/api/v1/playlists/{playlist}/nextsong[?format=text]**
  - Returns the next song selection for the specified playlist.
  - Either random selection, or, if the request queue is not empty, the first song in the request queue.
- **GET http://somehost:3000/api/v1/playlists/{playlist}/currentsong**
  - Returns the most recent song selection
  - Does not cause any new song selection to occur

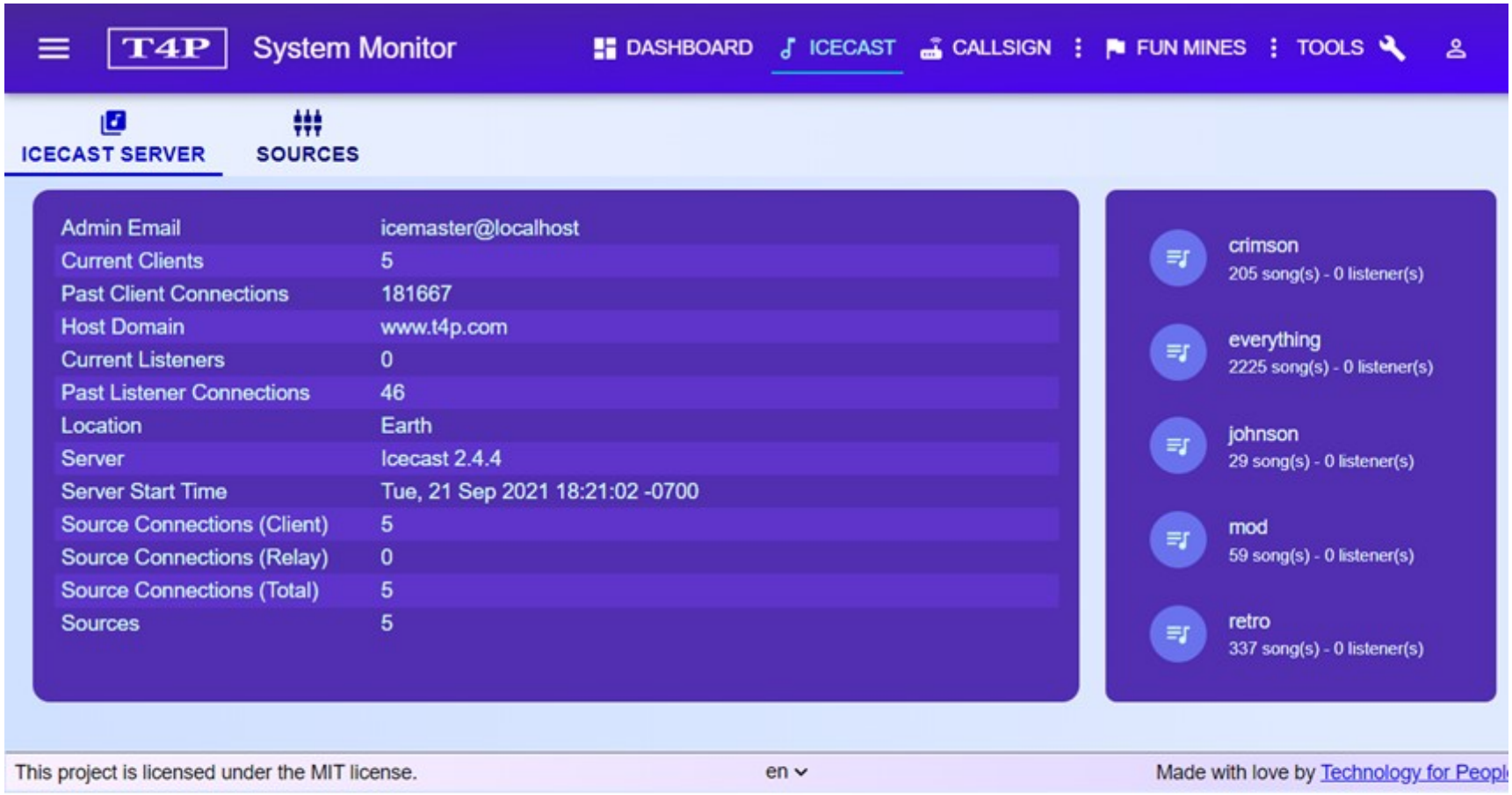
Future ideas

- More new plugins to allow the use of non-standardized filenames?
- Currently, song history is cleared upon restart
  - Perhaps persist the history on file or DB table?

Summary

- I’ve been using playgen for years
- The randomization of song selection is fine-tuned for the best human experience possible
- The RESTful API allows you to build robust web-apps for managing the playlists
- We are open to ideas to improve playgen, making it fit more use cases

Thank You!



https://github.com/jeghers/playgen