ccmixter.org Vision and Teams

March 6, 2017

Admiral Bob

Overview

Ccmixter has succeeded at becoming a top destination for musicians who want to make music in community with purpose, and for content creators who need resources with which to produce multimedia projects. This document outlines the technical challenges we have to meet in order for that to continue and expand, and the kind of help we need from the community if we're going to make it happen.

Goals

1. To find volunteers who can help with both our short term technical needs and our long term technology aspirations.
2. To organize them into teams specializing in these two areas.
3. By doing this, ensure ccmixter.org progresses at a stable and steady rate.

Problem Statement

Ccmixter.org is a volunteer run organization; but since it is a web site and application, it requires:

- web administration to keep running on a daily basis
- short term code patching and adjustment to keep up with API and language changes (such as changes in PHP, jQuery, Facebook's Graph API, Apache, and MySQL
- Longer term dev team help to carry out our vision:
    - a secure read/write API that uses API keys for access.
    - an architecture that puts more of the work onto the client with the server hosting only REST APIs and UI resources to facilitate that.
    - A consistent, adaptive, and pleasant UI up and down the site.

Buzzwords and Nerd Talk Used in this Document

Devops: this refers to developers, Technical Admins, Help Deskers, and documentation writers working together as part of the same team and project.

API: Application programming interface - the open programming features we add to our site so developers at ccmixter and elsewhere can build on our work.

REST: *Representational state transfer* - a way of designing APIs so that even the simplest possible interactions with the API can do useful things. Ccmixter's existing API is already pretty REST-like, but is read-only.

LAMP: Linux/Apache/MySQL/PHP is the name of a combination of technologies that are frequently used together, and which are called a "stack", a grouping dependent on the other.

MEAN: another rival stack, MongoDB/Express/AngularJS/NodeJS, purely designed around modern REST paradigms.

Dev Teams

To ensure that this can be achieved, ccmixter.org wants to stabilize its future with two volunteer devOps teams.

Current Site Maintenance Team

This team is a devOps team focusing on the “ops” side, but requires dev skill as well. This team would be responsible for:

- Advising on site and server performance. Looking for optimizations that can improve performance and trim bandwidth usage where practical.
- Keeping up to date on used platforms (Facebook Graph API, gingerly attempting to see if the site can be moved onto latest jQuery and Ember, PHP/Apache and MySQL.)
- Making small usability changes and enhancements, where these can be done quickly and/or without too much pain.
- Helping implementers of the current ccmixter.org API make best use of API calls.

Skills Required for the Team

Not everyone has to have all these skills, but they should all be represented on team.

- Linux sysadmin skills via SSH
- PHP and Apache
- Some Javascript and HTML would be useful.
- Help desk skills

Next Version Team

The “next version” team would also be devOps, but focused on development. The key vision of this team is to move to a modern REST model where posting to the server is done in a predictable, API tokened, JSON oriented way, with results rendered to the UI in a reliable JSON fashion. This team would be responsible for:

- **Platform and technology review:** is our current LAMP (Linux/Apache/MySQL/PHP stack going to do the job? Or are there any elements of the MEAN stack (Mongo/Express/Angular/Node) that may work better for us? This requires the team to look at REST capabilities for PHP such as the Slim framework, and whether this is good enough for a modern application.
- **Next Version Design and Planning:** there is a beta.ccmixter.org - the team needs to evaluate it, see if it is the right track for a future version, and decide to either complete the work, or start fresh. The team will then need to set that plan, and establish milestones and timelines to carry it out.
- **Playground Server/Staging:** the team will need to set up an alternate ccmixter.org that can serve as the development test platform to start, and a staging ground as work gets close to completion.

- **Implementation Planning:** working with the current maintenance team to deploy and launch the new version.

Skills Required for the Team

Not everyone has to have all these skills, but they should all be represented on team.

- Full stack skills in Apache/PHP and MEAN (or comparable frameworks.)
- Visual Design skills
- Wireframing and sketchy gui
- People with a lot of patience and creativity.
- Strong opinions about what music collaboration should look like
- Willing to read "ccmixter: A Memoir."

Next Version Principles

The next version of ccMixter.org, to remain maintainable in future, needs a strong modern separation of tiers using the modern REST/JSON paradigm.

RESTFul

Ccmixter.org absolutely needs to preserve and magnify its RESTful heritage. The API needs to become READ/WRITE and future versions of the API (or at least additions to it) need to be route/path based, highly normalized around REST conventions, such as the use of verbs (GET/PUT/POST/DELETE) for operations, JSON Web Tokens for preservation of authentication state, and an API key for access, so that apps that use ccmixter can be throttled to protect the site against DOS attacks or excessive traffic.

SEO Optimized

The current version of CCMixter is very attractive to search engines, and any move to a client heavy paradigm must ensure that discoverability of ccmixter.org resources via search remains excellent.

Usable

Any new version of the site should preserve what people love about ccmixter.org or improve it. The site should be good enough that people don't look for shortcuts back to the original site.

Well Architected

The site should be planned so that a new developer coming on board can make sense of what has been done, with enough modularity that you don’t have to search code contents to figure out how to do changes.