Addon registration

Release 0.1

October 20, 2015

Contents

The goal of this service is to provide a scalable API to do addon files registration.

The initial brainstorm is there, and a PRD document is also available.

CHAPTER 1

More doc

1.1 Design document

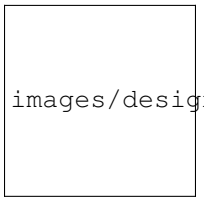
So we need to do a service able to handle a lot of simultaneous calls.

The file registration should interact closely with addons.mozilla.org since they are providing the information we need about user accounts.

It seem to make sense to have a way of communication between zamboni and addonreg. It seems that having an HTTP API is the way to go here. It should be async and put the jobs in a queue to avoid overwhelming the server. Since the writes are less important than the reads, that seems the way to go.

The idea is to have a completely separated service handling the API, and to let AMO handle the authentication.

The routing will be done at a higher level by a reverse proxy or directly at the client level, by using a different domain to access to service.



Here, we can see that the writes are going trough the AMO API and then to the addon registration API, whereas the reads are going directly to the addon registration API.

1.1.1 Software architecture

The API is done in python, using the cornice framework. It is done so that the actual storage and retrieval of the data can be interchanged with other implementations at any time. This is done to avoid having too much work in case we find out a better way to make things scale.

The plan is to start a basic implementation with everything in memory and then iterate quickly to something doing raw SQL calls to a database.

1.1.2 Caching

XXX

We have a lot of addons that will do calls to the API with the same data (assuming a lot of people install the same version of the same addon).

That would be silly to do a check on the database each time we have a request, and even more silly to have the client do the same call all the time with the same response.

We should use caching mechanisms whenever possible.

Invalidation

"Caching is easy, invalidation is hard".

One possibility would be to have all the information provided as static files. This way, it is really easy to make it scale without issues.

Then, we could queue all the changes to the files (the writes) in memory and do the appropriate changes to the files whenever that's needed.

This way, with very little effort, we get caching for free (caching mechanisms use the last modification date of files for caching) and scaling this is like scaling static files.

1.2 Open questions

These are a number of questions that had been asked / answered while doing the preliminary research on the subject. They can be useful for newcomers.

1.2.1 Why don't we register the known "bad" addons?

The original brainstorming document proposes to remove the addon-ids from the database once they have been found the be bad. Rather than doing that, it could make sense to register the bad addons as well, so we know for sure which one aren't OK for us.