InfiniteCommits – an integration between InfiniteGraph and GitHub

by William Cheung
Toronto, ~~September 30~~ October 17, 2011

Why I did it?

Aside from the obvious reason of evaluating InfiniteGraph as a graph db solution we could use at work, I developed InfiniteCommits for two primary reasons:

1. to see if InfiniteGraph could be used from the Play framework
2. to see if InfiniteGraph could be used to give GitHub junkies like myself useful info not available from GitHub's own web UI

Play is a rapid Java (and Scala) web development framework gaining popularity over Spring-based alternatives of Grails and Roo, so I wanted to see if InfiniteGraph could "play" nicely with the new kid on the block.

The feature that the GitHub web UI is missing for me is showing which files in a repo have had the "most action" in the last while. As a busy-bee developer I often don't have time to keep up with all the changes in my favourite projects, so knowing which files have had the most commits in the last week say would help me zero in on the pulse of the project.

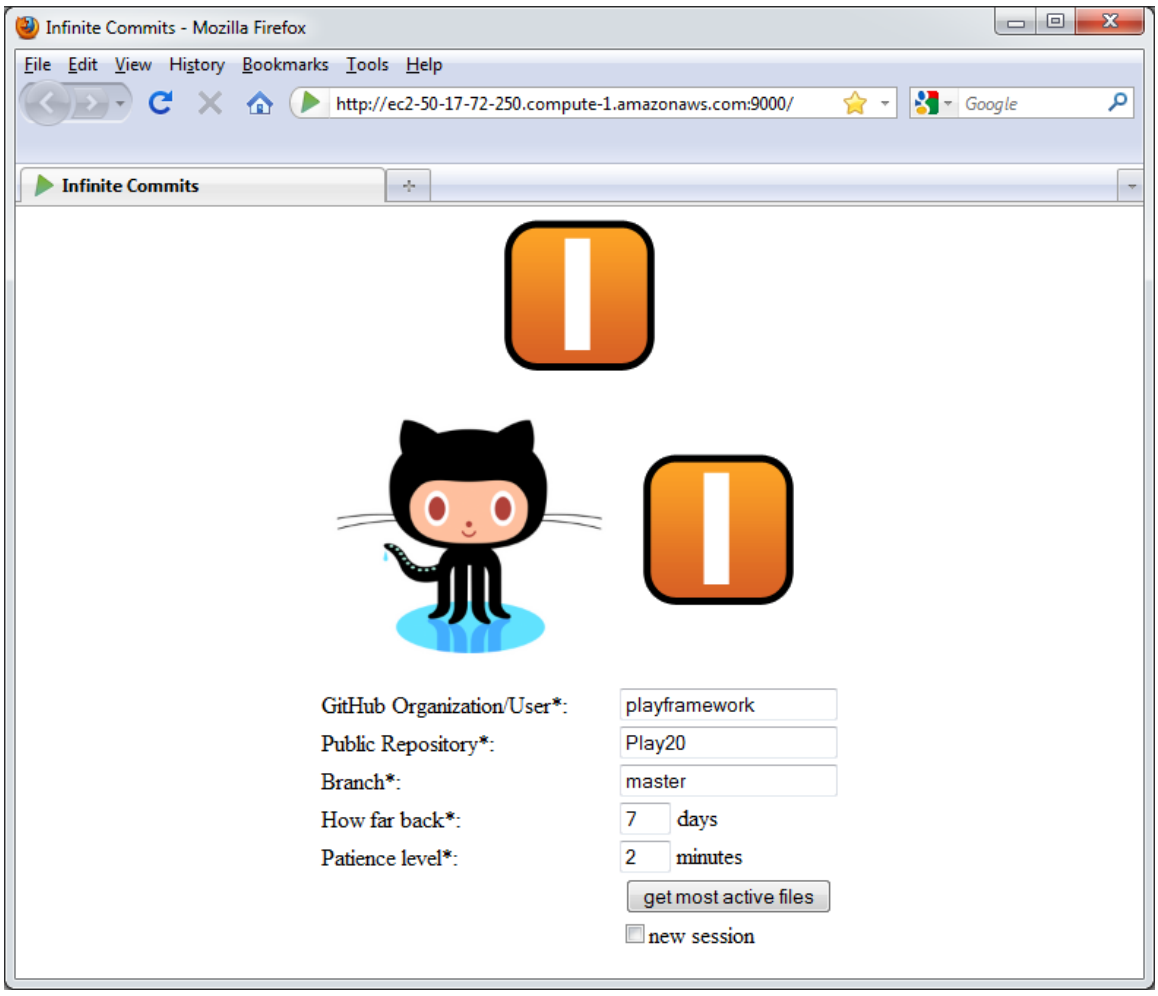
So that is what my project InfiniteCommits is about, a Play application to get a report of the most active files in a GitHub repository.

How you use it?

InfiniteCommits is deployed here:
http://ec2-50-17-72-250.compute-1.amazonaws.com:9000/ [1]

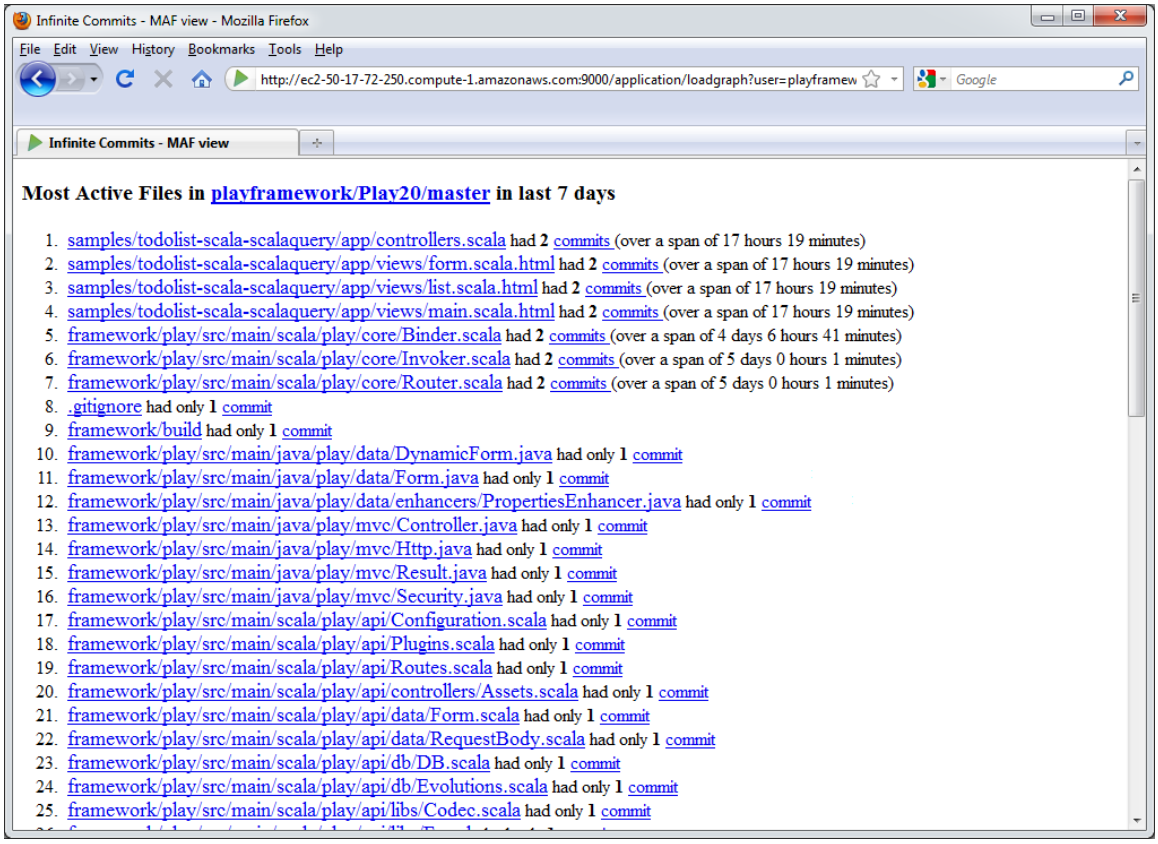
The UI is as simple as can be. First you have a form to fill in info for your search:[2]

[1] This is an AWS micro instance. See the topic I started in InfiniteGraph's Google Group for more details on this deployment scenario: https://groups.google.com/forum/#!topic/infinitegraph/2jFrHTi5bMI
[2] For a video demo of a local deployment see http://screencast.com/t/GYI7dz9Jh9B where I show everything running on my Windows 7 notebook.



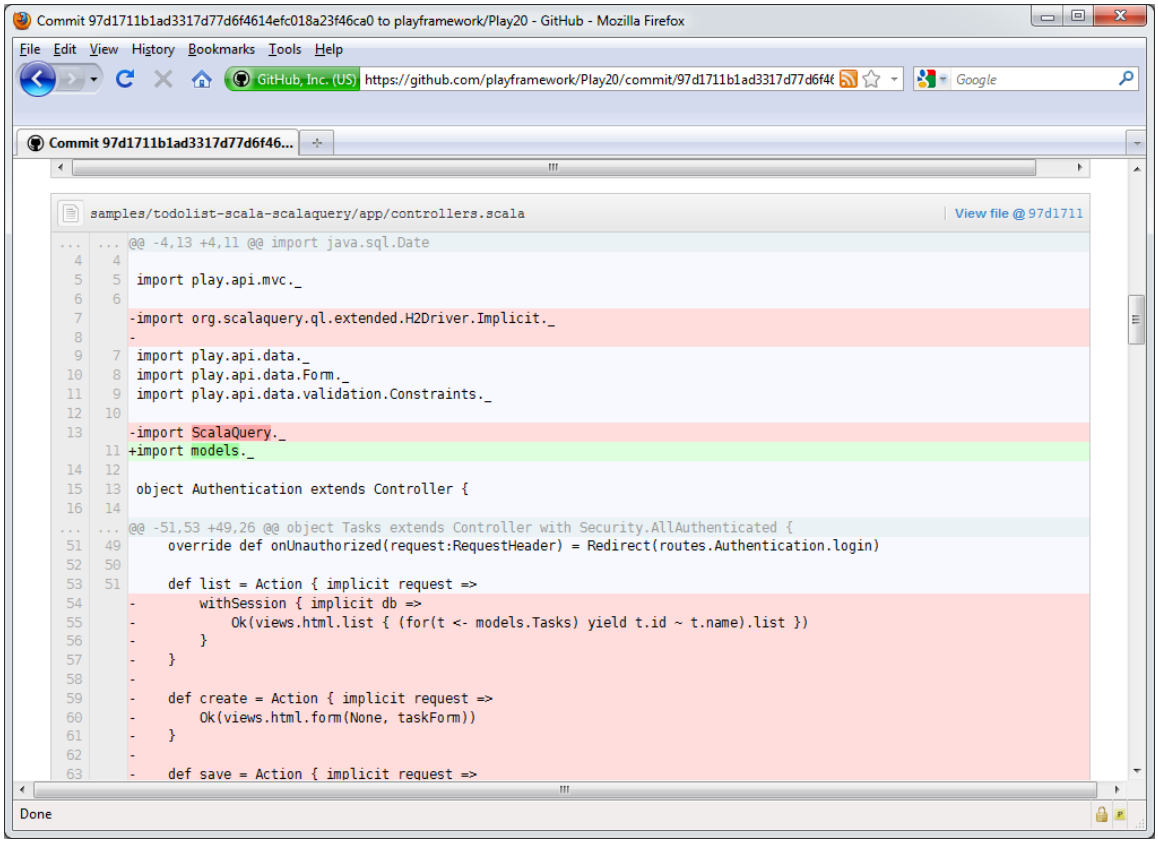
How far back lets you specify how far back in the commit history of this repo you want to go starting from now. *Patience level* lets you control how long you are willing to wait before getting your results. The *new session* checkbox lets you use a new search session instead of your last session for this repo (sessions are cached in InfiniteGraph).

Click the *get most active files* button and you get your results:



The results will be a list of files from the repo, with the most active files (most commits in the time span) at the top. Note that in the event of a tie (multiple files having the same number of commits), then those items are ordered by commit speed (highest rate of commit activity).

For the first item returned in the example here (controllers.scala) you can click on its *commits* link to get to the commits for this file. Then you can choose which commit you're interested in, for example the most recent one, and get the GitHub web view specifically for this file, as below:



This flow lets me go directly to the file changes I'm interested in seeing, saving me time tediously going through the GitHub web UI to pick through each recent commit.

How I did it?

The code[3] is organized into two projects, *infinite-commits* (the "business service" layer) and *infinite-play* (the web app).

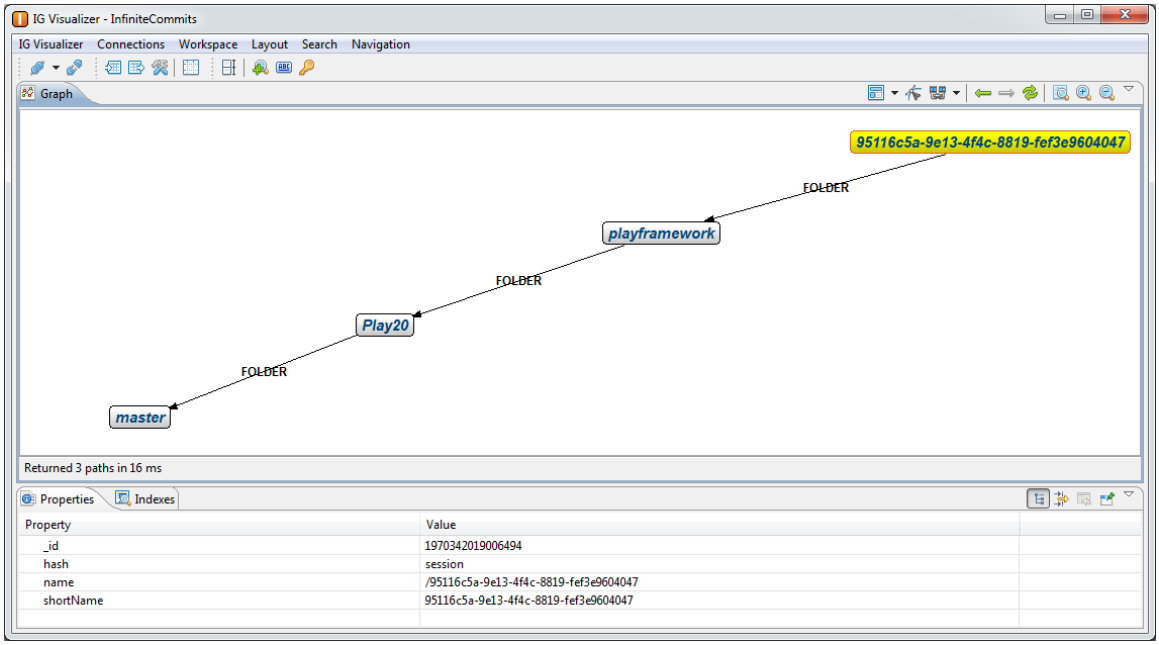
The business service layer has a **GraphManager** class responsible for creating and loading the graph. Loading the graph means loading the commits from GitHub for a given repo into InfiniteGraph as a session for later searching. The searching (to get the most active files in a repo) is done by the **GraphSeacher** class, within this method:

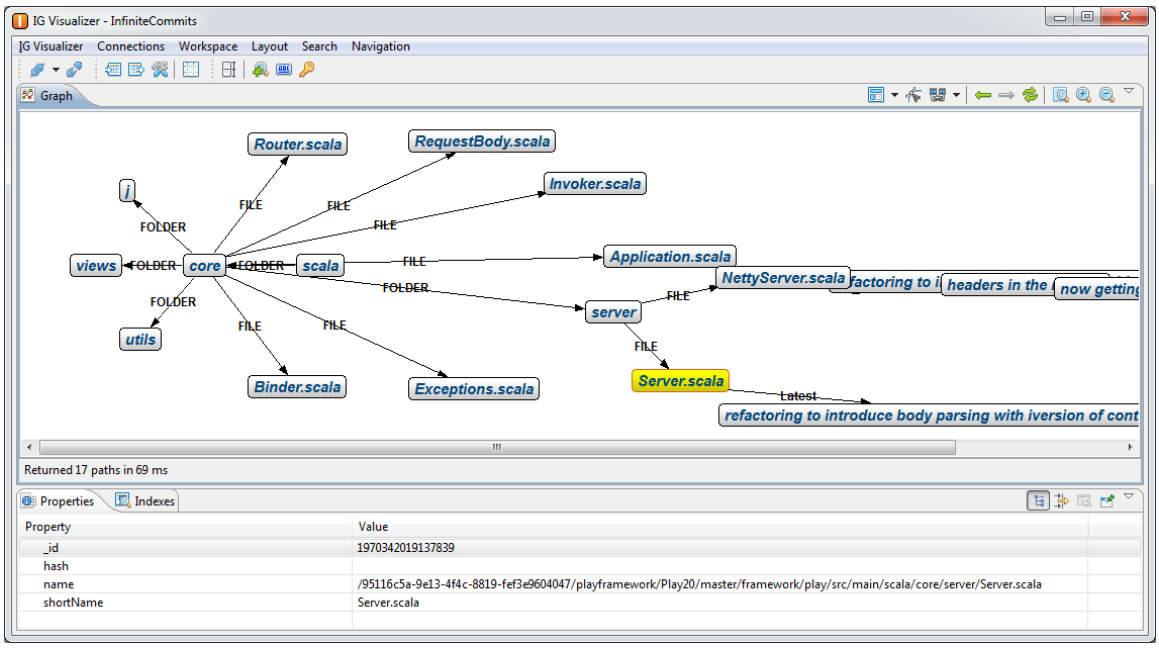
```
    public SortedSet<FileActivity> findMostActiveFiles() {
        Guide guide = Guide.SIMPLE_DEPTH_FIRST;
        AgeQualifier pathQualifier = new AgeQualifier(spanInDays);
        Qualifier resultQualifier = new NotFolderQualifier();

        MostActiveFilesListBuilder resultHandler = new
MostActiveFilesListBuilder(pathQualifier);

        Vertex startVertex =
graphDB.getNamedVertex(startingSearchVertexName);
```

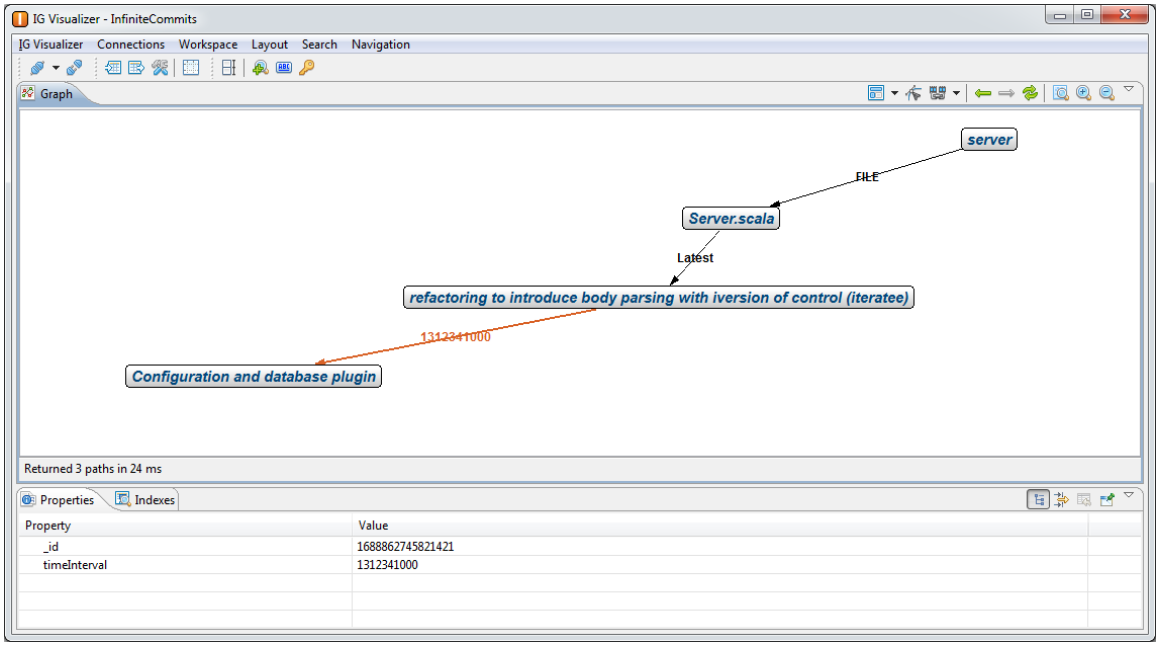
[3] The code for my project is open-sourced at: https://github.com/williamcheung/infinite-commits

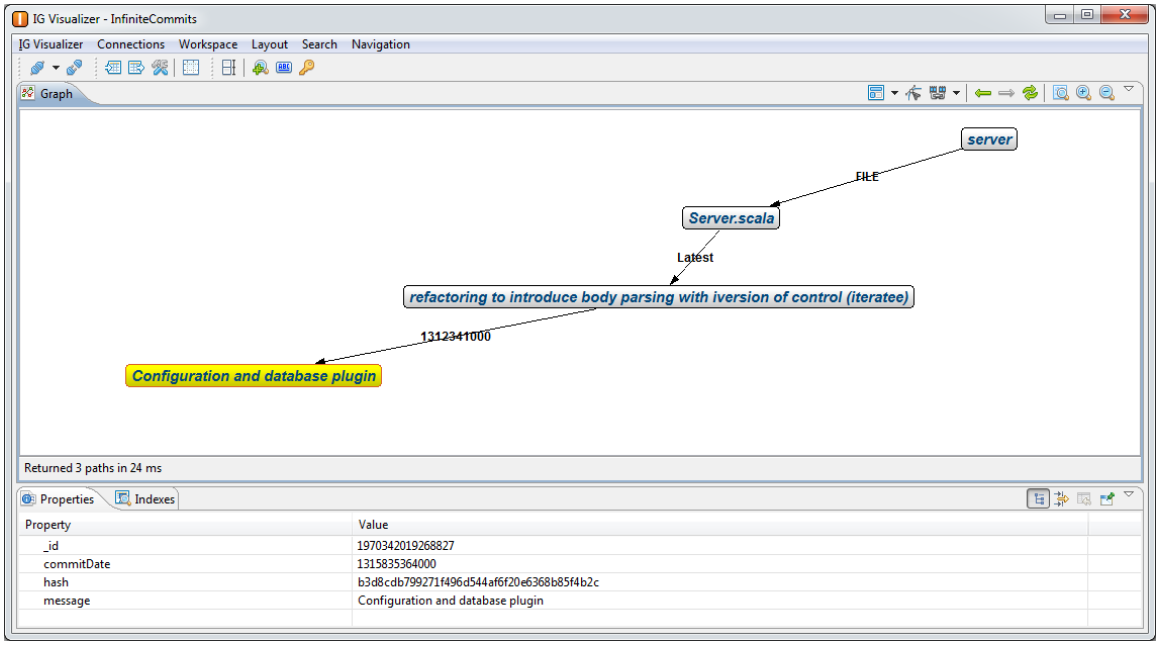
```
        Navigator nav = startVertex.navigate(guide, pathQualifier,
resultQualifier, resultHandler);
        nav.start();
        nav.stop();
        return resultHandler.getMostActiveFiles();
    }
```

The domain model for the graph consists of obvious and natural entities *Folder*, *File*, and *CommitEntry* used as vertices, with edges *Latest*, *Previous*, and *Child*, as shown in these example visualizations of a search session:









The loading of commits from GitHub is encapsulated in a **CommitLoader** class which makes calls to GitHub using an open source GitHub Java SDK[4]. Because the underlying GitHub REST API itself limits an IP address to at most 60 requests per minute, there is a singleton **ApiRateLimiter** class I implemented to avoid blowing the limit.

[4] https://github.com/nabeelmukhtar/github-java-sdk – released under the Apache 2.0 license

To decouple the loading of commits into InfiniteGraph from the **CommitLoader**, there's a *LoadListener* interface provided. **CommitLoader** sends notifications to the *LoadListener* interface which is implemented by a class which the **GraphManager** instantiates during loading a graph, **CommitLoadListener**.

The *web app* is built using the Java version of the Play framework. There are only two views: *index.html* for the form and *loadGraph.html* for the results, both using Play's Groovy template language. The **Application** class in the infinite-play project is the controller which builds the models for both views. **Application** maps your input to calls on the business service layer (**GraphManager** and **GraphSearcher**[5]). There is also a Bootstrap job which creates the graph db if need be, using the InfiniteGraph properties file located in the Play app's standard *conf* directory.

Note that in order to use InfiniteGraph 2.0 in Play, you *must* currently add this call after you open the graph db:

```
com.objy.db.app.Connection.current().useContextClassLoader(true);
```

See the topic I started in the InfiniteGraph Google Group for more details on why:
https://groups.google.com/forum/#!topic/infinitegraph/DOL-hEYvtH8

Conclusion

InfiniteCommits serves as an example of how InfiniteGraph can be used to build a web app on the increasingly popular Play framework where the domain (commit histories of GitHub repository trees) maps naturally to a graph structure, allowing us the ability to leverage InfiniteGraph's powerful, customizable graph navigation capability to easily distill essential insights such as areas of highest activity from a stream of infinite commit noise.

Less Noise, More Stuff

[5] You may have noticed that my business service classes reside in a *com.firstclassmandarin* package. For the curious, this refers to my wife's registered domain. She runs a Mandarin school here in Toronto, dragging me in to help :(