

Building LaTeX Lab

Bobby Soares

May 26, 2010

Contents

1	Project Goals	2
2	First Draft	3
2.1	Implementation	3
2.2	Challenges	4
2.3	Problems	5
3	Second Draft	5
3.1	Problems	6
4	Third Draft (final)	6
4.1	Adding Collaboration	7

This document describes the initial and final drafts of LaTeX Lab. It was initially written in Google Docs, with Google Drawings, and subsequently translated into LaTeX and compiled in LaTeX Lab.

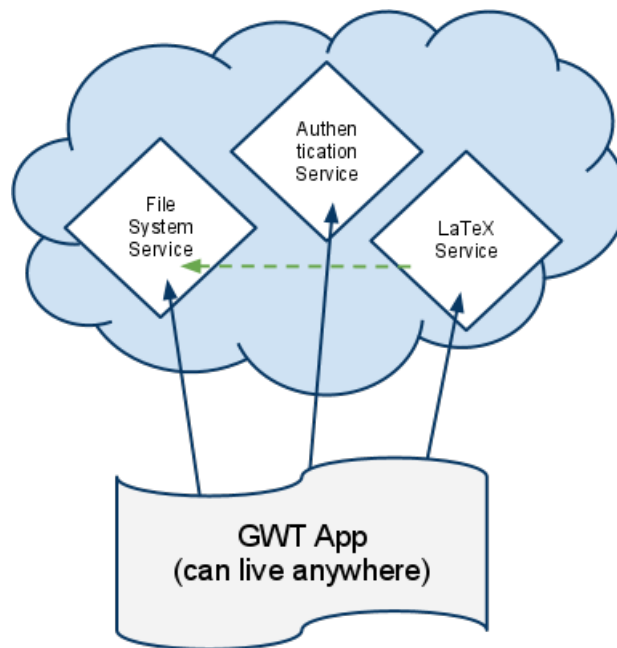
*For more information on the LaTeX Lab project visit:
<http://code.google.com/p/latex-lab>*

1 Project Goals

The following goals motivated the LaTeX Lab project:

- Create Open Source, web based editor.
- Use a single Eclipse code project (GWT + AppEngine in Java).
- Consume services already available on the web - promotes reuse and standardization.

2 First Draft



The first draft, a serverless app.

In this draft the LaTeX Editor is built with GWT and interacts with a collection of web services - possibly all third party - directly from the browser. The application does not have a dedicated server-side presence and could be served from the AppEngine or live in the user's desktop.

2.1 Implementation

- *File System* - Google Docs with GData API
- *Authentication* - GData AuthSub
- *LaTeX Compiler* - No 3rd party service available. Build as generic a LaTeX service as possible, allow 3rd parties to host a LaTeX service with minimum effort.

2.2 Challenges

- *How to expose LaTeX as a service?*

Build a minimal chroot containing a stripped down TeX Live instance. Dynamically provision a jail per user. Expose as a web service - in LaTeX Lab, as an XML based PHP service - see CLSI project.

- *How to compile projects with multiple resources? (e.g. one LaTeX document, 1 style file, 4 images)*

Have GWT app send current document contents, along with references to required project resources, to the LaTeX service. The LaTeX service should be able to fetch and process all resources, using a resource cache for performance.

- *LaTeX compiler will need to live in a separate domain (it can be plugged into LaTeX Lab as a third party service). How to get around cross domain issues?*

Use hidden form to POST compile request to LaTeX service. LaTeX service writes a response .js file containing the result. LaTeX Lab checks for response. Times out after X seconds and informs the user.

- *How to interact with Google Docs directly from the browser?*

There is a JS client library for GData. Create GWT bindings for this library and use within app - see gwt-gdata project.

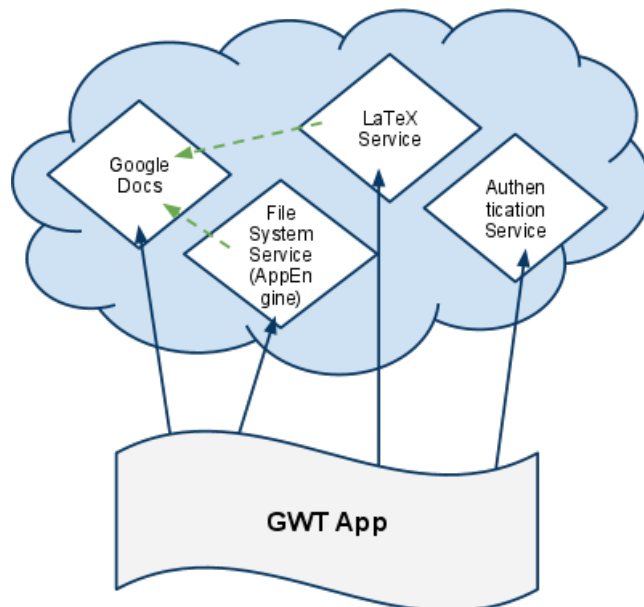
- *How to emulate Google Docs UI?*

Use as many native GWT elements as possible. Leave all style definitions in external .css file. Use gwt-dnd for drag-and-drop and gwt-incubator to fill in the gaps.

2.3 Problems

- JS client library for GData does not support reading or writing document contents - though it can be used to create, delete, rename and list documents. This means that a GWT friendly service will need to sit in front of GData.

3 Second Draft



The second draft, with an AppEngine dependency.

In this draft the client app still interacts directly with Google Docs via the GData JS client library, but only for creating, deleting, renaming and listing documents.

Queries to retrieve or update document contents go through a new, dedicated GWT RPC service - this service introduces additional costs and adds a server-side dependency, on the AppEngine.

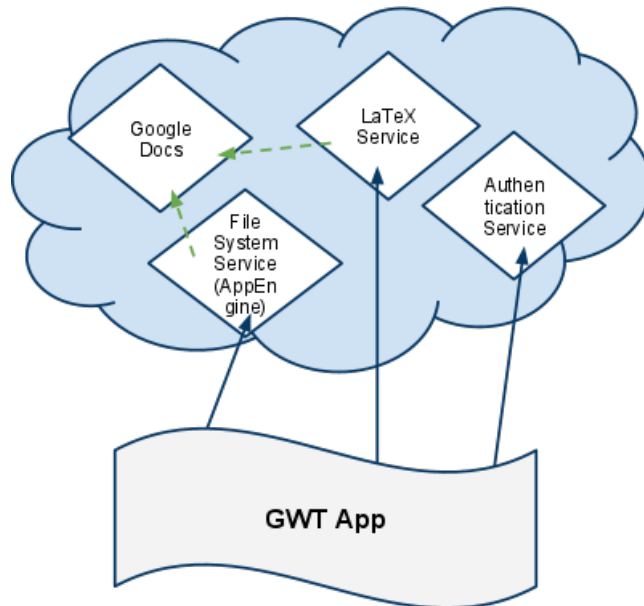
3.1 Problems

Allowing the LaTeX service to retrieve resources in Google Docs requires sharing the authentication token. This would share all of a user's documents with what may be a 3rd party service.

This risk can be eliminated through the use of secure AuthSub, which imposes a requirement that all Google Docs queries be signed with a certificate. The 3rd party service will not be able to sign unsolicited queries because it does not have the certificate.

Since Secure AuthSub is not supported by the GData JavaScript client library, it can no longer be used.

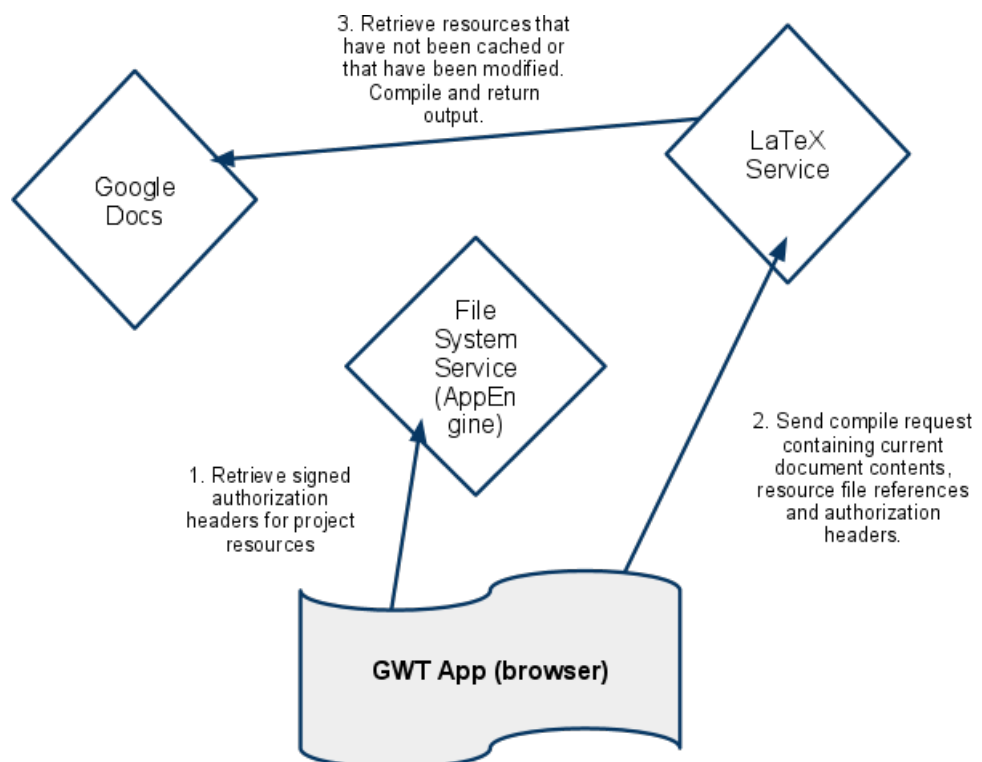
4 Third Draft (final)



The final draft, with all Google Docs queries relayed through the AppEngine.

In this draft all Google Docs queries are routed through a dedicated AppEngine service. Secure AuthSub is used.

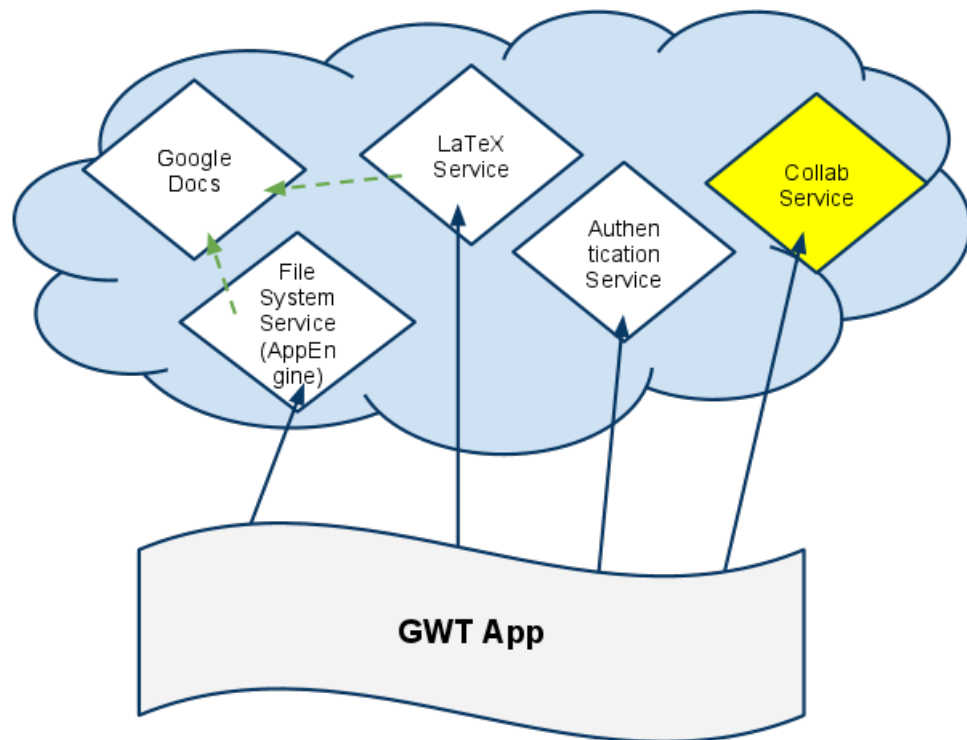
When compiling, the client obtains signed authorization headers for the documents to be compiled and includes them in the compile request. The LaTeX service will be able to access only those resources.



The compilation process.

4.1 Adding Collaboration

Collaboration can be added as an additional dedicated service, manipulated directly from the browser, though there will be cross-domain issues to deal with:



Collaboration as a plugin service.

Some additional features would also fit the service model and could exist as dedicated services:

- Cloud clipboard service.
- Spellchecking service.

By adding reusable services we significantly decrease the level of effort and upfront costs inherent in building web based applications.

Open Questions:

- Can a Collaboration service live separately from the File System service.
- Can the AppEngine dependency be removed, yielding a pure GWT, serverless app?

- How to register a custom app, such as LaTeX Lab, in Google Docs, as the default editor for a given file type?