FAS Google Calendar App
Developer's Manual

I began developing the FAS Google Calendar App with another intern during the Summer 2016 co-op semester. This is a record of what we have so far so that future co-ops/interns/whoever can more effectively develop the app. Disclaimer - neither I nor my partner have very much experience programming, so the code may be a little ugly.

Motivation for the app

The point of the app is to organize and display information from Google Calendar so that project managers and office schedulers can do their job more effectively. The company has a Google domain (fas-atlanta.com), and each employee has a company Google Calendar. Once the app is being used by the office, the use of the company Google Calendar will be strictly enforced. As I write this, almost no one actually fills out their calendar, so the app is functional but basically useless. One of the key features is that it generates extra data about employees, such as which of their events are out of office (OOO), whether or not they are searching for work, and whether or not they have gaps during the workday with no events recorded. This is based on the assumption that there will be some policy about what tags to put in the summary of an event to signify what type it is. The two most important are OOO and searching for work.

Anyone at FAS should be able to download and install the application from our fileserver. To do this use ClickOnce publishing, under Build – Publish in Visual Studio.

General Structure

The first thing we got the app to do was simply print out everyone's information to the console. Then they requested that it write everything to Excel. This was easy enough, but then we realized that just copying everything from Google into a spreadsheet isn't really that useful. Now there are three main things the app does:

- Writes an Excel spreadsheet with every employee's event, with multiple sheets for different categories.
- Creates a chart with a color-coded time line for each employee
- Creates a quick lookup window in which the user can type in a name and instantly bring up all the information about that person's events, as well as see how many hours they are missing, OOO, etc.

The last two are probably the most useful. The spreadsheet portion is essentially finished, so development going forward should be focused on the chart/lookup window, and possibly adding new features.

Main Form

The main form is where users will choose settings and then press the button that gets the data, and then processes and displays it. Currently the user can choose:

- The directory where the backup folder is created. This folder will contain the spreadsheet and a PNG image of the time line chart.
- The start date
- The end date

The app will get all data from Google between the start and end date selected by the user. Once the data is processed, it is first written to the Excel spreadsheet, then made into the chart, and finally connected with the lookup window.

Google

We use the Google Calendar API to get data from Google. It is important to note that this API was really designed to add events to someone's calendar, not to retrieve extensive information about events that already exist. We are sort of using it backwards... Getting the information is relatively simple, and is mostly just of a bunch of code copy-pasted from Google's API documentation. Is all happens in one method that makes one request for each employee and creates a dictionary mapping from employees to a list of their events. This is the slowest part of the application, usually taking about 10-15 seconds. We are not sure if there is a way to speed this up, but it is worth looking into. Another possibility is creating an app that employees use to interface with the company calendar. Inserting events in code with the API allows you to attach extra information in a dictionary that you otherwise couldn't create. This might be helpful for keeping track of things like job numbers, vacation, etc. In other words, no one would have to actually interact with the calendar itself. We would just use it as a database to store events entered through the app.

Another important issue is the actual connection to Google. Right now the code that actually authorizes and makes a connection is some mystical incantation that we don't understand. Most of it was just copied and pasted from Google's example code. What really needs to happen is we need a service account. Please get one set up and have Jim Pursley give it the right security permissions. You can give the service account access to our domain.

Excel

This is some of the worst code in the app. We don't anticipate users of the app paying much attention to the spreadsheet, but it may come in useful if they accidentally close the app and don't want to reconnect to Google. The spreadsheet has a sheet for all data, OOO events, and searching for work events, as well as a spreadsheet documenting employees who have no events listed during work hours. Again, this part of the app is mostly done, and is not anticipated to be used heavily, so don't waste too much development time here. Please do, however, test this code a lot. Make sure it follows the proper conventions for COM Interop.

Events vs. TimePeriods

The app currently generates two main mappings: from `Employees` to `Events` and from `Employees` to `TimePeriods`. Both mappings are kept in a `Dictionary`. The distinction between them is that the list of `Events` for a given `Employee` corresponds directly to the events they have in Google Calendar, while the list of `TimePeriods` represents the continuous time span from the start date to the end date, broken up into sections according to whether the employee is out of the office, searching for work, etc. This is the data structure that is used for the chart. One problem with the chart is that if someone has contiguous events of the same type, the algorithm that creates the `TimePeriods` makes each event separate. This is conceptually wrong since the `TimePeriod` is meant to represent a continuous time span of the same type of activity. The chart is meant to show a general view of each employee's schedule, so if someone has three one hour meetings back to back, it should display one three hour `TimePeriod`. This will involve some fiddling with the `TimePeriod` algorithm, which is found in the `EventUtils` class.

Extensions

Glenn McConnell approached us about the app since he is working on a similar scheduling project that will hopefully replace the whiteboard in Jackie's office. It keeps track of who is in and out of the office. Ideally there would be some connection between these two applications, at least in terms of how they connect to Google. He would like to be able to use the Google Calendar API in VB.NET and store data about employees in a SQL database. Unfortunately this idea came together at the very end of the semester, so we didn't have time to develop it. Please get in touch with Glenn about this. It would be great to develop some sort of integrated work calendar database system.

Testing

Lastly, this code needs to be tested thoroughly. We didn't do much testing, so we don't have a good idea of how stable the app is. Make sure it can be reliably published, installed, and run on all the computers in the office, and that you can't break it by doing the wrong thing. Most likely there will need to be more error handling code to account for something the user could do that we haven't though of.