Publications and Research   LaGuardia Community College

2014

Automate the Internet With "If This Then That" (IFTTT)

Steven Ovadia
CUNY La Guardia Community College

INTERNET CONNECTION

Automate the Internet With "If This Then That" (IFTTT)

STEVEN OVADIA

LaGuardia Community College, Long Island City, New York

It is difficult to explain If This Then That (IFTTT), a Web service that helps connect other Web services to each other. On its own, IFTTT (https://ifttt.com) does not do anything. Instead, it serves as the glue between other Web-based services using a concept called recipes, where two services are combined to accomplish a task. For instance, it can be used to send files uploaded into Dropbox into Evernote, the popular note-taking software, automatically. It can also work with mobile hardware, using mobile device functions, like location tracking, to trigger events like e-mails. The point of IFTTT is to automate tasks that lend themselves to automation, without requiring any formal programming skills.

Before exploring the specifics of the service and how it can help academics, it's best to briefly discuss its technical underpinnings. Many of IFTTT's recipes take advantage of Web-based application programming interfaces (APIs) (Vander Wal 2012). APIs are "a set of methods to access data in otherwise closed systems. APIs give programmers and developers the tools necessary to build software and services with data and services from external sources" (Michel 2013, xi). APIs are what allow data from services like Instagram, Twitter, and Tumblr to be used in clients and websites. If the concept sounds familiar, it has previously been referred to as a "mashup." Sonvilla-Weiss's (2010) explanation of mashups should also seem familiar:

> Comparably, in Mashup Cultures the code that makes possible information and knowledge exchange must be maintained liberally as a public good. Important steps in this direction are APIs that allow Web communities to create an open architecture for sharing content between communities and applications. (10)

Address correspondence to Steven Ovadia, LaGuardia Community College, Library Media Resources Center, 31-10 Thomson Avenue, Long Island City, NY 11101. E-mail: sovadia@lagcc.cuny.edu

The concept of APIs and IFTTT might also seem familiar to those who remember Yahoo! Pipes (http://pipes.yahoo.com), another service that made it relatively simple for nonprogrammers to connect Web-based services. Pipes is still around, but as of this writing, the project blog (http://pipes.yqlblog.net) has not been updated since 2012.

While IFTTT has been written about within the library literature, it has mostly been as a productivity tool. Ekart (2012) provided an overview of the service highlighting many of its useful features. Prince (2012) identified it as essential cloud-based application.

As mentioned earlier, IFTTT uses a recipe concept to describe actions. Users can search existing recipes, where everything is preconfigured. The user then needs to give permission for the services to allow IFTTT access to the personal data associated with the accounts. For some users, this will be a nonstarter. The IFTTT privacy policy (https://ifttt.com/privacy) says the company uses industry standards to protect private information. Users concerned about privacy might feel concerned about giving IFTTT access to an important service like e-mail or Dropbox, as opposed to something less personal, like a user's location for weather services or an RSS reader.

Existing recipes are keyword searchable. Users can type in a concept and see which recipes match and which services the recipes use. Users can also search by service, so if someone is looking to see what can be done with something like Evernote, she can browse all of the recipes involving Evernote.

As an example, a recent column discussed ResearchGate, the academic social network (Ovadia 2014). A search for ResearchGate reveals a recipe that takes notifications from the social network and adds it as a new row to a Google Drive spreadsheet. This might be useful for someone looking to track and quantify her activity within ResearchGate, without manually entering data into a spreadsheet. To use that particular recipe (which can be seen at, https://ifttt.com/recipes/148985-track-ResearchGate-notifications), the user must activate her Gmail channel and her Google Drive channel, giving IFTTT access to those services. Recipes can also be tweaked to a user's individual preferences. For instance, the recipe just described takes any message sent to a Gmail account from the default ResearchGate e-mail address and adds a row to a Google Drive spreadsheet, with one column for the date the e-mail was received, one column for the subject of the e-mail, a column for the ResearchGate attachment, and a column for the URL of the attachment. The spreadsheet will be named ResearchGate - Notifications and will be in a folder called IFTTT. While this information is not exceptionally easy to parse from the recipe, a few minutes of reviewing the fields, and perhaps the results of the recipe, will allow most users to get up to speed. IFTTT also includes the available fields, or ingredients, in the recipe, so users can see which other fields are available. For instance, using the preceding recipe, a user might see that the body of the

ResearchGate e-mail is an available ingredient and might decide to capture that in the spreadsheet. This can easily be done using IFTTT's drop-down menus. Users of this modified recipe will have the text of every ResearchGate e-mail in a single spreadsheet. The spreadsheet could be used to document or track activity on ResearchGate, including ResearchGate score increases and new followers. But rather than manually transcribing the data, IFTTT is handling the bulk of the work.

If this process sounds similar to programming, that's because it is, in a way. Greenwalt (2012) recommended IFTTT as a gateway into the programming process. IFTTT manages variables, showing users what's available, and allowing the user to manipulate and control the variables. In the preceding example, it's taking part of an e-mail message and placing it in a spreadsheet. Once users are used to parsing existing IFTTT recipes, other options and ideas that can automate work become apparent. For instance, the ResearchGate recipe funnels e-mails based upon a specific ResearchGate e-mail address. But the recipe could be tweaked to funnel e-mail from all kinds of e-mail addresses. For instance, someone who regularly receives Machine-Readable Cataloging (MARC) records from a vendor might configure a recipe to place the links to the MARC records in a Google Drive spreadsheet. Then the records would always be available, and e-mail would not need to be searched to find the records when it's time to upload the MARC.

Like any tool, there are limitations. IFTTT can send e-mails to any e-mail account but cannot access the content of any e-mail account. The preceding ResearchGate recipe specifically uses Gmail accounts but the recipe would not work for someone using Outlook.com or their work e-mail (unless the institution uses Gmail as its e-mail service). The recipe also filters based upon the sending e-mail address. If that e-mail address were to change, the recipe would no longer work. This means users need to periodically check on their recipes and make sure everything is continuing to work.

IFTTT uses other services besides e-mail and Google Drive. Bessette (2013) wrote about using IFTTT to archive tweets she had favorited. Bessette used Twitter's favoriting option as a form of bookmarking, using it to capture links, journals, and ideas found on Twitter that she wanted to return to later. Once a user has a sense of how IFTTT works, it is simply a matter of finding the right recipe. In Bessette's example, she knew she wanted a record of tweets she had favorited. IFTTT has quite a few recipes that solve that problem. Some recipes will save favorited tweets into Evernote or OneNote. Others will e-mail the favorited link somewhere. It is really just a matter of thinking what needs to be captured and where it needs to be put.

In terms of academic work, IFTTT also works with RSS feeds, as do many subscription databases, as discussed previously (Ovadia 2012). This could also be the start of a useful workflow. Users could use IFTTT to direct RSS feeds from a journal or search into something other than an RSS reader.

This could be anything from Evernote to an e-mail message to a Google Drive document. IFTTT allows users to get a bit more precise, though. Rather than working with the raw RSS, users can also work with a specific RSS reader, like Feedly (www.feedly.com) or NewsBlur (www.newsblur.com). Using a specific reader, rather than plain RSS, IFTTT can leverage functionality within the reader, so instead of sending all of an RSS feed into something like Evernote, a user can instead just send saved stories from NewsBlur, so rather than Evernote becoming filled with everything from an RSS feed, it is instead filled with work that has already been filtered to some extent, without having to manually be moved or documented.

IFTTT is a high-concept service, in that it allows users to combine other services together. Using it takes a certain willingness to poke around and deconstruct existing recipes. This is especially true for academics, as there do not seem to be many existing recipes focusing on academic research and processes. However, once familiar with the services available within IFTTT, potential workflows can arise. This is especially true if a user is already using a Web-based service that integrates with IFTTT.

The beauty of IFTTT is that it can automate certain tasks. Rather than manually entering data into two places, IFTTT can often handle the duplication. IFTTT is also very useful for capturing information. Researchers often find useful material in different channels—anywhere from Twitter to reddit to RSS feeds. IFTTT is a way to make sure materials discovered in those disparate channels all wind up in the same place, or in the correct different places.

REFERENCES

Bessette, Lee Skallerup. 2013. How to archive your favorite tweets. *ProfHacker (blog)*. *The Chronicle of Higher Education*, September 10. http://chronicle.com/blogs/profhacker/how-to-archive-your-favorite-tweets/52201

Ekart, Donna F. 2012. Tech tips for every librarian. *Computers in Libraries 32*(4): 36–37.

Greenwalt, R. Toby. 2012. 2012: The year code broke. *Public Libraries 52*(4): 23–50.

Michel, Jason Paul. 2013. *Web service APIs and libraries*. Chicago, IL: American Library Association.

Ovadia, Steven. 2012. Staying informed with Really Simple Syndication (RSS). *Behavioral and Social Sciences Librarian 31*(3–4): 179–83.

Ovadia, Steven. 2014. ResearchGate and Academia.edu: Academic social networks. *Behavioral and Social Sciences Librarian 33*(3): 165–9.

Prince, J. Dale. 2012. Ten essential cloud-based apps for medical librarians. *Journal of Electronic Resources in Medical Libraries 9*(2): 155–66.

Sonvilla-Weiss, Stefan. 2010. Introduction: Mashups, remix practices and the recombination of existing digital content. In *Mashup cultures*, ed. Stefan Sonvilla-Weiss, 8–23. New York, NY: SpringerWienNewYork.

Vander Wal, Thomas. 2012. Pull & push with IFTTT. *KM World 21*(6): 1–21.