4-1-2016

One-To-Many: Building a single-search interface for disparate resources

Cole Hudson
Wayne State University, fi1806@wayne.edu

Graham Hukill
Wayne State University, ej2929@wayne.edu

CHAPTER ELEVEN

ONE-TO-MANY

Building a Single-Search Interface for Disparate Resources

COLE HUDSON AND GRAHAM HUKILL

Libraries provide access to a complex array of resources, but it comes as little surprise that many struggle with this task. The ubiquity of Google in modern research has placed libraries in the position of trying to emulate or set themselves apart from the search engine, regardless of how appropriate it is to compare libraries to Google. There are numerous ways in which libraries can improve the search and discovery experience for patrons, and for many, this improvement currently comes in the form of custom-built or commercial discovery systems—which aggregate disparate library content into a single results display. But, regardless of discovery systems' potential, as Lown, Sierra, and Boyer (2013) conclude, libraries should learn and balance user expectations with the actual capabilities of library information systems. In an effort to find this balance, our work led us to a singular goal: providing a single-search interface for our complex array of library resources. What resulted was a discovery tool we named QuickSearch.

Developed in-house, QuickSearch is a single point of interaction, "bento style" search portal, displaying search results from disparate resources in discrete boxes. These resources come from numerous independent, back-end systems, including search results from the library catalog, Serials Solutions Summon Service, research

guides, journals, our institutional repository, digital collections, databases, and a site search of the entire university website. The term *bento* comes from Japanese cuisine where different parts of a meal are compartmentalized in aesthetically pleasing ways; this ultimately is the aesthetic goal of compartmentalizing search results from disparate resources on the results page.

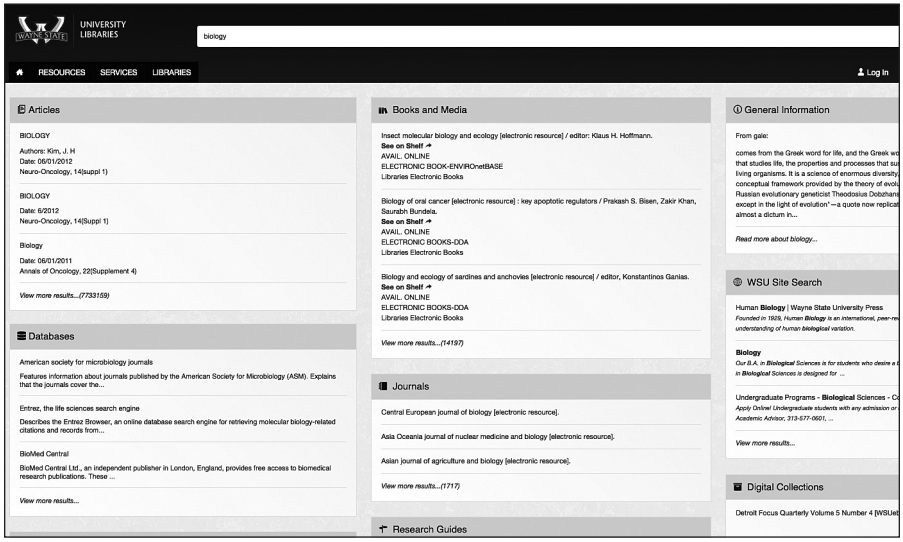
In this chapter we are not advocating for specific programming languages or technologies; instead, we aim to share our underlying design principles and philosophy in developing QuickSearch. As we see discovery moving toward constantly shifting silos of data that must be aggregated for users, we tried to imagine an architecture that accommodates this reality.

Vendors (e.g., Serial Solutions, EBSCO) offer products that aggregate library resources, but as we shall see, these products are, by nature, incomplete and incomprehensive. We are unable to rely on vendors for a single point of search, yet we are also unable to create our own single resource index. The discovery landscape is changing too quickly, and our staff is too small for a monolithic approach. Instead, we have chosen to address data silos individually and loosely couple them together under a single banner: QuickSearch—software that our small team of developers can update and maintain. This approach seems to be well received by users and has already resulted in increased access to resources.

This chapter will first outline QuickSearch's key features (see figure 11.1), discuss explored and realized approaches, and conclude with an appraisal of the impact QuickSearch has had on user behavior patterns.

FIGURE 11.1

Screenshot of QuickSearch displaying results for "biology" search



POTENTIAL APPROACHES AND WHY THEY DIDN'T WORK

To meet our goal of providing a comprehensive, intuitive search tool across all library resources, from a single interface, we first explored a range of different possible approaches. This section will discuss these, noting why they were not a good fit for us, in an effort to contextualize the development of QuickSearch and explain how it addressed problems with these other approaches. (See table 11.1.)

TABLE 11.1

Overview of explored approaches

Approach	Problem	QuickSearch's Solution
Web-Scale Discovery Services	Do not contain all library resources	WSDS results are valuable, incorporated as "Articles" box
Single Index of Library Resources	Difficult to rank, difficult to maintain	Currently avoids ranking
Monolithic Software Approach	Brittle, hard to update, too much maintenance	Simply designed, low barrier to construction, easy to update, resilient to catastrophic failure

WEB-SCALE DISCOVERY SERVICES

Providing a single point of interaction for search is increasingly a focus of vendors, as they realize how valuable this would be to libraries and their users. Companies such as EBSCO, Ex Libris, OCLC, and ProQuest all have products that index multiple resource types, including databases that they operate and provide access to, abstracts from other vendors' databases, and even MARC records for books and media from a library's catalog. These platforms are often referred to as "web-scale discovery services" (WSDS). The number of distinct items in these platforms is staggering—our own ProQuest "Summon" platform returns 632,286,219 results for a blanket search. Yet still, some library resources are not included, making them effectively invisible to our users.

Web-scale discovery services include resources from a variety of sources. They are well suited for searching individual articles because they include articles from a wide variety of journals and databases, in addition to MARC records they pull from library catalogs and other resources. But in spite of the increasingly wide net that web-scale discovery services cast over a library's resources, vast swaths are often missed: institutional repositories, digital collections, databases, research guides, information on the university or library website, and so on. These omissions can happen for a variety of reasons, three of which we will explore further below.

One reason resources might be absent from a WSDS is that they don't fit neatly into the model the service is using for indexing and searching resources. As advanced and sophisticated as they get, web-scale discovery services still rely on quasi-traditional approaches to modeling library resources: titles, authors, dates, publishers, and so forth. For example, they are ill prepared to accommodate a university's website search results where complex, algorithmic ranking of pages is useful, if not essential. Library research guides are another example of resources that do not fit neatly into that model. Web-scale discovery services are successful only insofar as the resources we seek can be described and searched in a meaningful way through their interface.

Second, it is possible the resources are inaccessible to the WSDS, or too small and unique to be indexed by a large, vendor-created WSDS. For example, at Wayne State University we have a growing institutional repository (DigitalCommons@WayneState) and online digital collections, neither of which are indexed by our Summon platform. These are valuable library resources, but are missed completely.

Thirdly, even within resource types that a WSDS is able to index and search, there might be gaps when resources are not included due to "reluctant vendor participation/partnerships or to choices of resource inclusion made by libraries" (Ellero 2013). In effect, the coverage of a WSDS is a moving target, which does not always support comprehensive, repeatable searching.

The point here is not to provide an in-depth analysis of web-scale discovery services, which other articles have expertly done (e.g., Ellero 2013), but instead to show that despite their vast reach, there are library resources they currently cannot, and perhaps will not, index for search.

SINGLE INDEX: APPLES AND ORANGES

Another approach to providing a single-search interface for library patrons is to put metadata records from disparate sources into a single index, and provide a search interface from that single point. Cosmetically and conceptually it is quite similar to Google, in that all resources would come from a central index, interleaved into a single list of results. This approach looks elegant on paper, but is fraught with its own problems. A particularly thorny one is that

> library data just doesn't have the same characteristics as Web data. It just doesn't. Though . . . lots of folks working on Web Scale Discovery have made big strides, nobody has "solved" relevance ranking for full library discovery the way Google solved it for the Web. (Thomale 2015)

This quote from Jason Thomale during his talk at Code4Lib Annual in 2015 (also see chapter 10, "The Bento Box Design Pattern," by Thomale, Philipps, and Hicks) very nicely sums up the problem of creating a single index, comprised of multiple resource types, to search and provide results: the resource types that are included are simply too different, and ranking them as search results is an effort in futility. Search algorithms that power a single database or resource type are complex at best, but ranking multiple resource types increases that complexity exponentially.

With the goal of a single-search box interface, a bento-style approach sidesteps some problems with this approach, namely it does not purport to rank variant resource types. Instead, it provides these resources on equal footing, leveraging the ranking algorithms internal to each resource's individual search interface.

MONOLITHIC SOFTWARE APPLICATION: LESS CODE, MORE UNIFORMITY

A third approach is to design a system in which resources share common pathways and code. Each "box" on the page has to ask its associated database for data in a particular fashion. Each database returns this data in a particular form. Given five to ten boxes, the complexity of code needed to handle these differences can grow quite dramatically. The way to prevent this would be to pass everything through a common processing pipe that would handle requesting and returning data to the "boxes" on the page. This approach would allow for less overall code and more uniform features between the boxes. If every box uses the same processing pipe, potentially when an update to one box is applied, it would apply to all. This approach, however, is not without its pitfalls.

The main issue with this kind of approach can be best expressed in its need for dedicated staff to manage the system. In a 2001 IBM white paper, Paul Horn states that the greatest obstacle facing the IT industry is complexity. "In fact, the growing complexity of the I/T infrastructure threatens to undermine the very benefits information technology aims to provide." This problem is one of scale, because soon, he warns, there would not be enough IT professionals to handle the rise in complex, interconnected systems (Horn 2001). Dystopic vision aside, Horn does touch upon a key issue that applies to any organization regardless of size—the increased human cost of ever more complex systems. Horn's premonitions are relevant to our work with QuickSearch. Maintaining a more complex and better-integrated system would be cost-prohibitive in terms of dedicated staff time. We have no full-time front or back-end web developers.

The main development team consists of librarians and system administrators, for which application development is only a part of their job duties. If we followed a monolithic development approach, one in which we wrote DRY (Don't Repeat Yourself) code, this would naturally lead to something such as a common processing pipe for all resource types, but the costs of maintaining this approach would increase.

Another issue to contend with is that the resources which power QuickSearch undergo changes and upgrades at different rates either because of (1) internal decisions to change how we interact with a resource, or (2) changes to resources outside of our control (such as Serial Solutions' Summon service or Springshare's LibGuides API). When the resources share a common processing pipe, any changes we make to one resource would have to not disrupt the functionality of other resources. This, in turn, increases the amount of testing needed to ensure that all the resources still retrieve and display results correctly. With enough people and very established application development workflows, interconnected systems can be run smoothly. With our small team, this was a major roadblock to adopting this overall approach.

QUICKSEARCH: OUR APPROACH, OUR SOLUTION

Given our goal of a single-search interface across all library resources, we developed QuickSearch with the shortcomings of other approaches firmly in mind, focusing on a low-barrier, manageable approach we hoped would hit all of our requirements.

The design principle is straightforward: QuickSearch is a representative specimen of the bento box-style search interface, where results from different sources are returned to visually discrete boxes on the results page. One page, one search box, results from as many library resource types as possible.

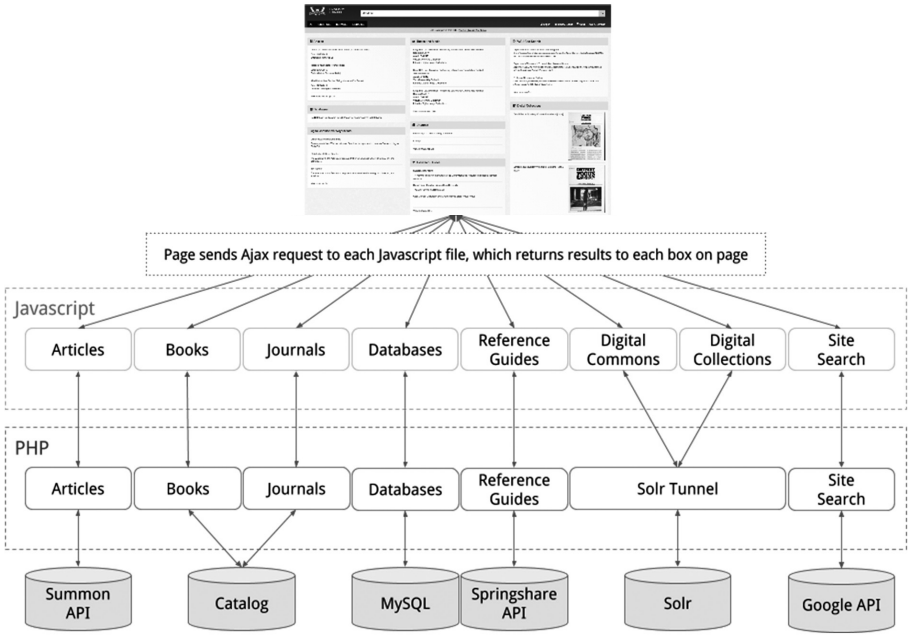
DESIGN PRINCIPLE: EVERY BOX FOR ITSELF

As opposed to the monolithic approach discussed above, we opted for an architecture with a much lower barrier to entry. Figure 11.2 is an overview of the QuickSearch architecture.

Our strategy going into development—when the number of resources types (boxes) was still fluctuating—was to have each "box" on the search results page have its own "pipe" back to the original source that provided results relevant to the current search. The advantages to this approach have been numerous.

FIGURE 11.2

Architecture of QuickSearch



LOW BARRIER TO CREATION

Even during the process of identifying and configuring data sources that would populate the search results, we could start building "pipes" for known resource types and could begin prototyping the page. In a monolithic-based approach, it might have been cumbersome to wait on unknowns like this, such that we could fold them into more terse and purposeful code.

EASY TO COMPREHEND

Just follow the arrows! Most librarians who work with electronic resources will be familiar with the databases at the bottom of the figure (some are vendors, some are databases with indexed resources). Except for a couple of unique cases where different "boxes" share "pipes," there is a near 1:1:1 ratio from JavaScript-populated "box," to PHP "tunnel," to database on the back end.

WELL SUITED FOR A SMALL TEAM

The distributed nature of the design—highly modular and autonomous routes for each "box"—made it easy to develop with a small number of people working on the front and back ends. In many instances, we split up the work by resource type, where each person would code the "pipe" for a given resource from beginning to end. They would be responsible for the actual database queries, moving up through PHP, JavaScript, and finally into displaying results. In a larger, more monolithic system, this kind of workflow would be nearly impossible. The only constraint we had was that box results must be reliable and consistent with the resource's native results set; the mechanics of each "pipe" were determined by the nature of the resource database.

NUTS AND BOLTS OVERVIEW OF QUICKSEARCH

Aside from the traditional markup and styling in HTML and CSS, the application consists of JavaScript- and PHP-mediated interactions with HTTP-accessible data end points. JavaScript renders resource data onto a user's page, while PHP communicates with the data resource. Each resource has its own JavaScript and PHP scripts which handle querying and response separately from the other available QuickSearch resources.

When a patron first uses the QuickSearch tool, their query term(s) are captured via JavaScript. AJAX (Asynchronous JavaScript and XML) calls, each connected to a separate box on the page, then pass along the data for processing by the server. Server-side scripts written in PHP take care of formatting and sending query terms to their corresponding database. When the database responds back with the patron's requested resources, the PHP script processes and sends the data back up the pipeline. Finally, the data is passed back to the QuickSearch web page where it is inserted into the database's corresponding box. Due to the built-in asynchronicity of the AJAX calls, which handle the patron's input and the database's corresponding output, each set of data is returned only when it is ready and the resource's failure and speed (or lack thereof) has no impact upon the other resource boxes or their ability to load data from their own data sources.

IMPACT AND ENGAGEMENT

QuickSearch was launched November 6, 2013, as an additional search option among a cluster of search interfaces. Though it provided an umbrella search that

covered resources from other search interfaces, it was not until August 2014, when we launched a redesign of the website, that we did away with the search interface cluster approach and featured QuickSearch as the primary search interface from the library home page.

Since that time, inroads for search and discovery have changed dramatically. What used to be unique and isolated search interfaces have become a single point of interaction. The result has been largely positive. Users have demonstrated active and sustained engagement on the search page; resources that were previously unsearchable and inaccessible from the library website are now discoverable; and we received a healthy dose of "No News is Good News" feedback from users.

INCREASED ACCESS TO RESOURCES

A driving force for the creation of QuickSearch was to provide discoverability to resources that were previously unsearchable from the library website. Nevertheless, many of the resources now searched by QuickSearch were, in fact, previously searchable from the library website via a tabbed search interface on the library home page: Articles ("Everything"), Books and Media ("Catalog"), Databases ("Article Databases"), Journals ("Online Journals"), and even Site Search.

Though all of these resource types were searchable, tab usage was inconsistent, and search was certainly not possible across all resource types at once. The effect hurt serendipitous discovery across resource types, and required a duplication of effort on the user's part, constantly repeating searches in different interfaces, navigating the challenge and peculiarities of each.

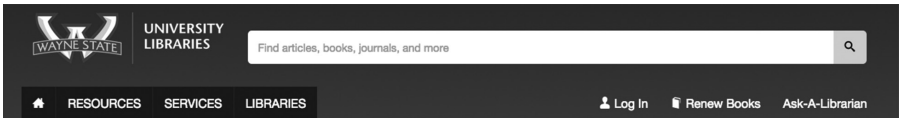
With the launch of QuickSearch, a new tab was created allowing users to search across these resources with one search (see figure 11.3).

FIGURE 11.3

Pre-redesign website tabbed search interfaces with QuickSearch included

FIGURE 11.4

Current website without tabbed search interfaces, featuring QuickSearch as the sole search box in header



Even with the addition of QuickSearch, tab usage was still inconsistent and confusing, and patrons still had the option to search specific resource types via their isolated search interface. With the redesign in August 2014, we moved away from the tabbed search cluster, and for the first time, started pushing all library search and discovery traffic from our new standard header (see figure 11.4) through QuickSearch.

As user searching began routing through QuickSearch only, many library resources that were previously hidden from search and discovery on the main library website were, for the first time, exposed to all searches, for all users. This includes our Institutional Repository (DC@WSU), our Digital Collections (WSUDOR), Library Research Guides, and a more sophisticated Site Search of the entire wayne.edu domain.

This exposure has resulted in increased use and awareness of these resources. Since QuickSearch launched in August 2014, through March 2015, it has directed users to resources previously unsearchable from the library website:

- 800 + visits to digital collections
- 3,800 + visits to library research guides
- 3,500 + visits to DC@WSU

Those visits to DC@WSU, as an example, are particularly encouraging. DC@WSU is a Software-as-a-Service (SaaS) from Bepress, with a website and search interface entirely removed from the library infrastructure. Usage of this resource was limited to the traffic we could direct there, often just for known items such as electronic theses or dissertations. These near 4,000 visits not only demonstrate increased usage of DC@WSU's valuable, Wayne State scholarship, but have produced increased awareness around DC@WSU and its mission as the university's institutional repository. Increased campus awareness of DC@WSU is a major, somewhat unforeseen benefit of simply showing up on the QuickSearch results page. The same can be said about Digital Collections, Research Guides, and other parts of the university website that Site Search scours.

USER ENGAGEMENT BY THE NUMBERS

Just shy of 300,000 visits (where a "visit" is defined as a new user visiting the page, or a returning user after thirty minutes or more) at the time of this writing, QuickSearch represents a substantial portion of all traffic to the library website, which had approximately 1,600,000 visits in this same time period. But even more interesting than simple visits to the page is user behavior once there.

To collect statistics on QuickSearch, we use an open-source website analytics platform, Piwik (http://piwik.org/). We tracked visits to the page, searches on the page, and clicks to actual resources. While "outlinks" (in Piwik nomenclature) may contain clicks to the header menu items, we believe the vast majority of those clicks are to QuickSearch search results. We have recently started quantifying only QuickSearch clicks; this will allow us greater granularity in future analysis. From this tracking, a couple interesting insights emerge.

First, use of QuickSearch was minimal until it was featured as the primary library search interface; even though it was offered as a tab on the search interface cluster, and at one point even the default one, it was not widely used. Traffic increased sharply around the beginning of the fall semester after the redesign. For our team, this slow rollout was a good time to identify and fix bugs, while traffic was slow. Having other librarians test the system by performing searches and noting irregularities was instrumental during that time.

Second, and perhaps the most interesting insight to emerge from the numbers, is the behavior of users on the page. Reviewing the analytics have identified a rough 1:2:4 ratio of visits: searches: resource use.

Piwik's web logs serve as a cross-check. We can look at anonymized individual user visits to see the activity in search terms and links out. While these do not perfectly model the 1–2-4 ratio we see in the graphs, they serve as sketches of common user behavior that leads to the patterns we observe among users overall.

QuickSearch demonstrates a high level of user engagement. By contrast, even a website such as our LibGuides platform—with plenty of dynamic content—has a much closer 1:1:1 ratio of visits, activity, and outlinks.

We believe we can assert that users are engaged with the website, performing and refining queries, and perhaps most importantly, are being exposed to a large and diverse array of library resources. While we cannot make inferences about the relevance of search results for our users at this time, we are happy with this level of engagement. Understanding user engagement with QuickSearch in a more comprehensive way is a next step as we continue to develop and refine this tool.

"NO NEWS IS GOOD NEWS"

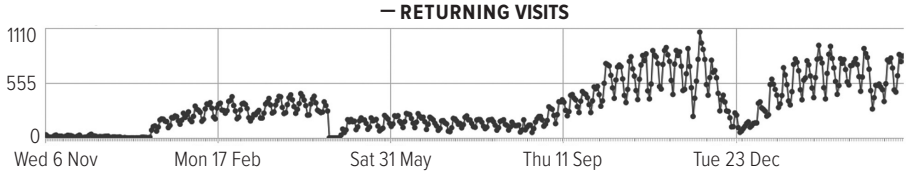
In thinking about the impact QuickSearch has had on discovery for our patrons, we spoke with our user experience librarian who has been operating a survey since the launch of QuickSearch and the website redesign. The survey was provided to users as a banner across the top of QuickSearch. While we initially lamented only 45–50 survey responses regarding QuickSearch, this librarian astutely pointed out that considering more than 300,000 QuickSearch searches, only 45–50 responses, and the somewhat unfortunate reality that most feedback is negative when it does come back, one might chalk up a small response rate to "No news is good news," and assume it quietly became a successful and functional search interface for users. If things were not working well, we most likely would have heard about it.

Further user data supports this as well. Figure 11.5 represents returning visitors to QuickSearch. Over the course of a semester, the number of returning visitors increases. While QuickSearch is our primary search interface now, other more targeted search interfaces do still exist, even featured prominently as "QuickLinks" on the front page, yet users continue to return to QuickSearch to perform their searches.

Given these markers, and some anecdotal testing and feedback from users, we feel confident that QuickSearch has integrated itself as the primary search interface for our users, with a high degree of use and engagement, providing discoverability and access to a wider array of library resources than ever before, from one location.

FIGURE 11.5

Returning users to QuickSearch (visits triggered by user searches)



CONCLUSION

In building QuickSearch, we are confident that we met our goal of providing a service that unites our disparate resources. It is sustainable for our development team, and we see the continued and sustained use by our patrons as positive. However, regardless of the tool's current success, we realize that user expectations about resources and their presentation will evolve. Even now, we are evaluating radical changes to QuickSearch's interface. Regardless of how it evolves, we are confident the design principles that underlie QuickSearch—modular components, ease of development for a small team, and flexibility towards resource types—will remain constant as we move our discovery efforts forward.

ACKNOWLEDGMENTS

Building QuickSearch was a collaborative effort within the library. Development would not have been possible without the feedback and input of our librarians and staff. Particular credit and thanks to Rachael Clark, Joseph Gajda, Niranjan Jadhav, Axa Mei Liauw, Joshua Neds-Fox, Elliot Polak, Vinay Potluri, and Negib Sherif.

References

Ellero, Nadine P. 2013. "Integration or Disintegration: Where Is Discovery Headed?" *Journal of Library Metadata* 13, no. 4: 311–29.

Horn, Paul. 2001. *Autonomic Computing: IBM's Perspective on the State of Information Technology*. Armonk, NY: IBM.

Lown, Cory, Tito Sierra, and Josh Boyer. 2013. "How Users Search the Library from a Single Search Box." *College and Research Libraries* 74, no. 3: 227–41.

Thomale, Jason. 2015. "You Gotta Keep 'em Separated: The Case for Bento Box Discovery Interfaces." Lecture presented at the Code4Lib conference, Portland, Oregon, on February 11, 2015.