Delivering Targeted Library Resources into a Blackboard Framework

Richard Cox
ERIT, University Libraries
University of North Carolina at Greensboro
PO Box 26170
Greensboro NC 27402-6170

Abstract

This chapter describes the planning, development, and implementation of an application that pushes library content into the Blackboard course management system. Developed by the Electronic Resources and Information Technology Department (ERIT) at the University Libraries, University of North Carolina at Greensboro (UNCG), this tool is built around a number of technologies and methodologies including SOAP Web services, ASP.NET, Java, AJAX, and Adobe Flex. Through this application, the library provides up-to-date, customized links to databases and e-journals at the course level, thereby establishing a presence at a point of need. This application may be the first to integrate library content dynamically into Blackboard at this level and scale.

Introduction

Despite the breadth of quality resources made available online by academic libraries, many researchers spend relatively little time on library Web sites. Users look for simple interfaces that return relevant results immediately. They often do not want to learn multiple or complex library or learning systems, nor do they want to deal with several sign-ons across campus.

Like many development projects, the Blackboard initiative at the UNCG University Libraries was born out of the need to solve a specific problem. In 2003, as a member of ERIT, I first began looking at our Reference Department's Subject and Course Guides. This collection is a sprawling set of static Web pages that now totals a whopping 15.4 MB and encompasses 1,469 files and 166 folders. The Web pages contain the types of resources one would expect to support classroom instruction within UNCG's subject areas– relevant journals, databases, contact information, and so on. When one combines the generic data formats and categories, the repetitive look and feel, and the inherent challenges involved in maintaining over 1,000 high-traffic Web pages, it was apparent to me that the Subject Guides needed to be generated through a database application. This move would reduce the maintenance load to a much more manageable number - probably less than ten pages - and add a high level of searchability and portability to the information.

An ad hoc working group was created to revamp the Subject and Course Guide areas of the University Libraries' Web site. The group combined the technical skills of ERIT with the subject expertise of the Reference Department. ERIT put together a SQL Server 2000 database that contained a great deal of information related to our vendor database subscriptions, including breakdowns by subject area and subject-based RSS Feeds. This "metabase" has been informally dubbed VDBS (Vendor Database Services) and to this day powers the University Libraries' online database listings by both title and subject (http://library.uncg.edu/dbs/). Unfortunately, when the Reference Department reviewed the progress of the project, disagreements led to its demise. Some librarians in the department wanted the more standardized, easily maintained design, while others wanted complete control over every element on their pages despite any

maintenance overhead. ERIT did not want to develop a product for a client that could not agree on what it wanted. It was not feasible to build a database application to recreate the pages as they currently existed while at the same time including infinite customization options.

As a lead developer and project manager for ERIT, I found it difficult to allow such a worthwhile project to fade away. Moreover, as an applications programmer, I believe in the interrelatedness of projects, and that no application should stand alone. While pondering the issues relating to our subject pages and the orphaned VDBS application, it became apparent to me that the central point of contention was the user interface. If this issue could be eliminated, progress might be made. A personal interest in Web services and their potential uses led to what would become the Library Resources project: the delivery of robust, targeted library resources into Blackboard at the course level.

The strategy behind the Library Resources project reflects principles of Library 2.0. We are building a component-based application that pulls together resources from multiple locations and in different formats. As a lightweight approach to applications development, Library Resources is open to ongoing modification. As a result of its XML-based format, its content can be repurposed for more than one type of device. This is an approach that embodies the Library 2.0 concepts of perpetual beta and the long tail. The Library Resources project is most closely tied to Library 2.0 in its user-centricity and tight integration with UNCG's e-learning environment, Blackboard. By delivery library content to Blackboard, we are able to meet users' needs when, where, and how they need it and in a way that not only leaves behind the library's walls, but the

library's Web space as well.

This methodology is attentive to needs pointed out within the OCLC 2003 Environmental Scan, which notes that users are looking for disaggregation, self-service, seamlessness, and satisfaction (http://www.oclc.org/reports/escan/). The Library Resources project, with a native user interface within Blackboard (or, really, any Course Management System), provides a single login system, relevant resources where the users already are, and content customized to the needs of users at the time and place of need. As a result, each of the needs identified by OCLC is addressed through the use of Library 2.0 concepts.

The Library Resources project is a Web 2.0 application on both front and back end. The two browser clients, detailed below, are based upon two Rich Internet Application (RIA) techniques. One is an Adobe Flex-based solution, while the other is based upon Ajax and JSON utilizing CSS and XHTML for markup. On the back end of the application, the University Libraries is delivering data via the open-source JSON.NET and, most importantly, XML-based Web services Application Programming Interfaces (APIs).

What Are Web Services?

The World Wide Web Consortium (W3C) defines Web Services as providing

> A standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions thanks to the use of XML (World Wide Web Consortium 2007).

To put it another way, Web Services are a platform independent method by which differing systems may seamlessly communicate with and transfer data between one another. The normal mode of communication is via the SOAP (Service Oriented Architecture Protocol) XML standard (http://www.w3.org/2000/xp/Group/), using WSDL (Web Services Description Language) (http://www.w3.org/2002/ws/desc/). SOAP and WSDL are both XML-based languages. SOAP is the primary protocol over which Web service requests and responses are transmitted between the client and server, while WSDL is a description of all of the actions the Web service is capable of performing, as well as how the clients and servers should interact. WSDL is generally used for automatically generating code (based upon the WSDL definitions) and for Web service configuration.

Amazon.com provides a well-known example of a Web service. The site contains a section entitled "Where's My Stuff?," where customers can track their recent United States Postal Service (USPS) or United Parcel Service (UPS) orders. After logging in to Amazon.com, they can select a link for the appropriate shipment, and voilá -- their tracking information appears *on the Amazon.com Web site*. This occurs without redirecting customers to the USPS or UPS sites. How is this accomplished? Web Services. When customers select the "track" button, the Amazon.com Web server connects to the appropriate Web service, sending along the tracking number and using a format specified within the USPS or UPS WSDL. The Web service then searches the provider's own databases and returns a response, also conforming to the WSDL definition. The Amazon.com Web application formats the returned data and presents the results to the user. **[Insert Figure Cox-1 approximately here.] [Figure 1: Amazon.com Track Your Order Web Service]** Thus timely tracking information is made available to Amazon.com

customers directly from USPS (United States Postal Service 2005) or UPS (United Parcel Service) without leaving the Amazon.com storefront. This portability across different systems and servers from a central data source makes Web Services an ideal model for delivering resources within Blackboard or other applications outside the library network and Web space.

The Initiative

The Library Resources project at UNCG is an application that pushes library content into the campus Blackboard course management system, thereby serving users at a point of need. This project was built by ERIT under the direction of the author, who was also the primary application developer.

The Back End and Administration

The back end and administration for the Library Resources project consists of several distinct content components brought together into a single administrative interface. Microsoft's Active Directory was utilized in order to provide both security and librarian-related information within the project, while also benefiting from data originating within three other, stand-alone ERIT or UNCG applications: Banner, Journal Finder, and VDBS. Pieces of each program are pulled together into a new administrative interface called the Library Resources Content Administration Tool which lies behind and drives the Library Resources project. It cannot be stressed enough how much more complex an undertaking this project would have been if not for the ability to leverage preexisting tools and applications in constructing the Library Resources Content Administration Tool.

Windows Server 2003 Active Directory

Microsoft's Active Directory is a Windows Server-based implementation of the LDAP (Lightweight Directory Access Protocol) standard. Among its many uses, Active Directory organizes network objects such as users and computers into a centralized data store, and provides centralized authentication services for the Windows environment. The user storage and authentication pieces of Active Directory are used to pull librarian subject and departmental liaison information into the Library Resources database, and its authentication procedures monitor access to and rights within the administration tool. We use this information to build a link from a Blackboard course to the e-mail (and potentially Instant Messaging) account of the relevant subject expert in the Reference Department.

Banner Student Information System

The University Libraries maintains an interface into the University's Banner system (http://banner.uncg.edu/). Banner is a comprehensive computer information system that contains information on courses, students, faculty, staff, and alumni. It is through this interface that ERIT pulls new course information (course, number, section, and abbreviation) each semester for inclusion in the Library Resources application.

Journal Finder and VDBS

Journal Finder is an OpenURL resolver and knowledge base developed locally by ERIT (http://journalfinder.uncg.edu/uncg/). Journal Finder includes a subject area-browsable interface based upon the areas of study at UNCG. Journal listings by subject within Journal Finder are accessed by the Library Resources administrative tool.

VDBS, as mentioned above, is a SQL Server 2000 database that contains vendor database subscriptions, including the capacity to sort by subject area. As with Journal Finder, database information is acquired by the Library Resources administrative tool from VDBS broken down by subject. Using our extant Journal Finder and VDBS databases, we can push into Blackboard links to the most relevant e-journals and databases for each individual class.

Library Resources Content Administration Tool

Of course, someone needs to decide which library resources are most critical for a given class or area of study. The assignment of particular resources to specific classes is accomplished through the Library Resources Content Administration Tool. Access to and permissions within this tool are overseen by Active Directory. The administration tool then completes several tasks automatically, storing the results within a new SQL Server database:

- Associate librarians, also provided from Active Directory, with their liaison subject/course responsibility area, creating the Library Liaisons listings
- Attaches default liaisons to courses pulled from Banner within their subject/course responsibility area
- Generates contact information from Active Directory for each Subject Liaison

Once logged into the system, a subject liaison may change liaison assignments on a course-by-course basis and maneuver between semesters. The core functionality of the tool lies in the ability of the liaison to associate specific courses with electronic journals culled from Journal Finder and/or databases pulled from VDBS, as well as a limited range of additional resources.

These resources can be assigned in three distinct ways:

1. An entire subject area (all English courses)
2. An entire course level (all English composition [101] courses)
3. A specific section of a course (English 101, section 01).

[Insert Figure Cox-2 approximately here.] [Figure 2: Content Administration Tool, sample course listing] As new resource associations are created within the administrative tool, they are dynamically picked up by the Library Resources Web Service and populated into Blackboard. As a result, we are customizing content for each class or entire groups of classes, all in real-time in order to meet the immediate needs of both instructors and students.

Taking It to the Masses: Powering the Web Service

The UNCG University Libraries works within a Microsoft development environment, but a similar architecture could be built based upon any number of frameworks or languages, including J2EE (Java), PHP, Perl, Python, or a number of other options. The Library Resources Web Service is built in ASP.NET 2.0, and looks for three distinct parameters in any SOAP request: academic departmental abbreviation, course number, and section number. Once the .NET application has checked the validity of the request, it will query the SQL Server 2000 database fronted by the Content Administration Tool and return the relevant records (journals, databases, and liaison contact information) to the client via a SOAP response.

Blackboard and Adobe Flex

In coordination with UNCG's Blackboard Application Administrator, who manages all programs built to interface with Blackboard in addition to the course management system itself, we were able to create a Web services client as a Blackboard Building Block. A Building Block is an application built to work within the Blackboard framework and extend its functionality via openly available APIs. We attempted to do this by having Java, Blackboard's programming language of choice, speak directly to ASP.NET over the campus network. Due to network configuration issues, this proved unwieldy. As a result, we had to turn to a Web browser/client-based approach. We chose Adobe Flex because it is the client-side development platform the Blackboard Application Administrator is most familiar with. Adobe Flex is a Flash-based application framework used to create rich client-based, cross-platform Web applications (http://www.adobe.com/products/flex/).

The Library Resources/Adobe Flex Building Block is passed to the standard three course parameters by Java. Flex then initiates a SOAP request to the University Libraries Web service, which returns a list of e-journals, vendor databases, and contact liaison information. Flex then takes the returned XML data and formats it for the user as Adobe Flash.

The User Experience

When students log into UNCG's Blackboard environment, they see a list of courses in which they are enrolled. After selecting one, they go into that course and see a button labeled "Tools," which in turn leads to a "My Library Resources" option. **[Insert Figure Cox-3 approximately here.] [Figure 3: "My Library Resources" link]** Selecting this link opens a new page within

the primary frame. This contains the results of the Web services data request, laid out in a familiar table format.

[Insert Figure Cox-4 approximately here.] [Figure 4: Adobe Flex client displaying results for REL-231-01]

Once students have selected a journal or database from the "My Library Resources" table within Blackboard, they are redirected through a library Web server which transparently verifies that their request is coming from within Blackboard, and therefore pre-authenticated as affiliated with the University. As a result, students do not need to be diverted through the University Libraries' authentication mechanism before being passed on to their chosen resource. This is an important step in the process, as it eliminates the necessity for multiple sign-ons. This can be very frustrating to users.

The Good and the Bad

Technology

The client portion of the service application relies heavily upon Flash Player version 9. This is a small, widely installed browser plugin with which many students are familiar. As of March 2007, the version 9 player had was installed across 84% of Internet-enabled devices in the United States and Canada (Adobe Systems Incorporated 2007a). In addition, Flash 9 is the installed player across the UNCG campus computer labs. For an overwhelming majority of users, therefore, access to library resources within Blackboard is a seamless process. The UNCG Division of Continual Learning utilizes a great deal of interactive Flash applications in its online

courses, and as a result should be able to readily utilize the available code base in the same Flex tool as employed in Blackboard.

There were a number of technological challenges faced in implementing this application. As we debugged and overcame one hurdle, another tricky area was uncovered. The Blackboard Application Administrator and I believed that we were having difficulty getting ASP.NET and Java to talk to one another because we were not very familiar with each other's programming language of choice. This turned out not to be the case, as the two platforms were having no communication issues. Rather, we discovered that the UNCG network architecture was placing limitations on port 80 communications which prevented server-to-server communication. This, our greatest hurdle, was dealt with by the adopting a browser client-based approach. Such an approach brought its own set of concerns, and we had to circumvent cross site scripting (XSS) security vulnerabilities (CGISecurity.com 2002) through the implementation of an Adobe Flash cross domain policy file (Adobe Systems Incorporated 2007b).

Unfortunately, our solution is not as portable as we would like, as it requires knowledge of both Flash/Flex and Java in order to modify the interface for a customized implementation and to access the Web service. The high degree of required programming skills could reduce the usefulness of the tool if we wanted to utilize it in an environment outside of Blackboard, for example embedded within departmental Web pages. Another client, with less of a technological overhead, was required if we wanted to utilize our application in other environments.. This will be addressed below.

Usability

From the user's perspective, the approach is streamlined. The interaction required from the students using the Flex-based approach is relatively minimal. They select a link from a list of targeted resources, and are connected to their information without the requirements of additional authentication. Should they encounter difficulties or questions, the contact information for their subject liaison is provided on the same page as the data sources.

Because of the need for a default implementation that would place no burden on our teaching faculty, the Flex client is buried fairly deeply (three levels to be exact) within a course. Without directions - be it from teaching faculty or as a result of library instruction efforts - or a goodly amount of simply playing around with Blackboard course navigation, many students are not aware of the project's existence. In addition, there is a lack of customization options with this approach. The presentation layer is standard across all courses, and does not reflect the individualized approach that course builders in Blackboard may take in designing a distinct presentation for their class. As a next step, we need a way of implementing a Web service client on the browser that addresses these concerns as well as other issues including the limited portability and high degree of required technical expertise in developing with Flash/Flex and Java. Enter JSON and AJAX.

Bringing JSON into the Mix

JavaScript Object Notation (JSON) is, like SOAP, a data interchange format. While SOAP is an XML-based format, JSON is a subset of JavaScript. Essentially, it is a JavaScript object containing an array of data and, as a result, is lightweight, nimble, easy to read, and portable

(http://www.json.org/).

ERIT has completed work on the beta version of an AJAX Web services client for our Blackboard tool using JSON as the data exchange format instead of SOAP. While not a programming language, AJAX (Asynchronous JavaScript and XML) is a programming technique that allows for small sections of a Web page to be reloaded either at a specified time by the application or through interaction with the user. This allows for greater interaction between the user and the Web site or application, as well as enhanced usability, speed, and functionality (Garrett 2005). Generally, AJAX Internet applications are created through the use of:

- XHTML and Cascading Style Sheets (CSS) for markup and styling
- JavaScript and the Document Object Model, especially the XMLHttpRequest Object, for client-side programming (http://www.w3.org/TR/XMLHttpRequest/)
- A data exchange format, generally either XML or JSON

AJAX is in many ways the backbone of Web 2.0 (thus Library 2.0), as it is the programming method many Web 2.0 applications rely upon. Examples include Flickr (http://www.flickr.com/), Gmail (http://www.gmail.com/), and Microsoft Virtual Earth (http://maps.live.com/).

AJAX offers a number of benefits. One is that clients become AJAX applications instead of Flash/Flex objects. This removes the requirements for the Flash plugin. As another benefit, an AJAX approach lowers the technology barrier by placing the pre-existing JavaScript functions

and objects used to call the Web service and convert the resulting array from JSON to XHTML into remote files. The piece of simple JavaScript shown below is all that need be included on the client Web page.

```
<script type="text/javascript">
function buildResource() {
    var url = "http://library.uncg.edu/cgpub/json.aspx";
    url += "?dept=lis";
    url += "&crs=688e";
    url += "&sec=01";
    url += "&output=json";
    url += "&callback=jsonResults";
    getJson(url);
    }
addLoadEvent(buildResource);
</script>
```

Web or Blackboard course developers need only edit the three lines shown in bold, substituting the information specific to their courses. The bulk of the programming work is accomplished through calls to remotely linked-in JavaScript files that the developer need never access. Changes to the Web services request parameters are written in plain text rather than embedded inside a Flex object. In addition, AJAX utilizes standards-based languages such as valid Document Object Model (http://www.w3.org/DOM/), CSS, and XHTML, which are similarly understood across multiple clients without the need for a plugin. As a result, the AJAX client module can be embedded anywhere within an online course that the developer or instructor chooses. Another benefit of utilizing Web standards instead of Flex is that the structure is based

on meaningful HTML elements, e.g., headers, a data table, and a definition item list, and the presentation manipulable through the use of an additional, locally maintained and created CSS file. Developers can change the presentation of the displayed data to fit the needs of their course using the same tools as those used in the construction of the course itself.

It is important to note that while the above AJAX approach requires the developer to manually add a piece of code to delineate the different courses, the standard Flex implementation does not have any extra code for the developer to add. All course designation changes occur from within the Blackboard application.

How the AJAX Client Works

Blackboard course developers input the three standard course parameters into the JavaScript as shown in the example above. When students visit the site and the page begins loading, the JavaScript initiates an Http request to a proxy service residing within the University Libraries Web environment. The proxy issues a call to the Web service based upon the provided course parameters. The SOAP response is then converted into a JSON object array which is returned to the requesting Web page for processing. JavaScript then takes the received array, transforms the data into XHTML, and embeds it dynamically into the page. Applying a default or localized style sheet then completes the transformation for the users.

The User Experience

The user experience with the AJAX client will vary depending on the design of the online course component. Generally, users log into Blackboard and select a course from the list of those in which they are enrolled. Assuming that they have chosen a course with a uniquely designed

online component, access to the Library Resources service can be achieved in any number of ways. Usually, the service is available on the home page of the course or through a link within the course navigation to a page containing the client. Such a page might also link to other library materials within Blackboard, for example electronic reserves. Nothing prevents librarians from including other materials on the same page as the Library Resources service client.

When users select the link, a page opens and a default "Loader" image will briefly appear. This image informs the user that there is activity happening, while behind the scenes JavaScript begins communication with the University Libraries Web service and formatting data for the screen. **[Insert Figure Cox-5 approximately here.] [Figure 5: AJAX client loader image]** Once the AJAX client has received the response data set and formatted it, a JavaScript function removes the loader along with any other placeholder content, and then writes the Library Resources information to the screen, laid out in table format. **[Insert Figure Cox-6 approximately here.] [Figure 6: AJAX client displaying results for BLS-340-41D]** In several test cases, the loading occurs fast enough so that the loader image is never seen by users. This result will vary based upon a number of variables including the size of the page, amount of data being transferred, bandwidth availability, and the speed of the user's Internet connection.

Once users have selected a journal or database, the process is very much like that of the Flex client, in that they are verified as affiliated users within Blackboard, and redirected as necessary. Like the Flex Building Block, the necessity for multiple sign-ins is eliminated.

Should JavaScript be disabled or the connections to the Web service fail, the application writes a

set of default links to resources hosted on the University Libraries' Web server. **[Insert Figure Cox-7 approximately here.] [Figure 7: Default, non-JavaScript content]** As a result, users are always provided with resources, though the dynamic results are tailored to their immediate needs.

Flex vs. AJAX

In comparing the different Web services client implementations, the JavaScript-based AJAX/JSON client is a much more flexible, faster solution than the Adobe Flex version. The AJAX client allows for a greater degree of customization, has a slightly smaller footprint and faster load speed, and can be implemented easily both within and outside of Blackboard (or any other Course Management System) at any point within a course hierarchy. The client can be broken into remote components hosted by the University Libraries for ease of managing functionality or adding new functions. In addition, it requires only a minimal amount of code work, limited to the editing of three variables. Developing a Flex client that would operate outside of Blackboard would require in-depth knowledge of Adobe Flex/Flash.

On the other hand, the Adobe Flex client has the benefit of being always available within Blackboard as a Building Block, whether or not there is a customized course component available. This would theoretically give a larger presence for the University Libraries, even if not in an optimal location. The Flex object has the added benefit of being the preferred format for UNCG's Distance Education unit. Coordination with this unit is a key component of extending our reach beyond the library walls to the widest possible student audience. The Flex client also precludes the requirement for the developer to maintain any code, even the three lines required for the AJAX solution.

Where We Are Today

In late 2006, ERIT began building the database that would drive the Web service push into Blackboard. The initiative launched in beta in January of 2007 with the push of the Adobe Flex client to our production Blackboard server. During the Spring 2007 semester, we pushed content into thirty courses within the UNCG Blackboard framework. Based on anecdotal feedback derived from library instruction courses, the initial response has been positive. During the summer, we conducted targeted training on the use of the Content Administration Tool to our subject liaison librarians who did not participate in its beta development. We also began teaching first year students to locate and use the Library Resources service. Our first courses utilizing the AJAX/JSON client went live in beta over the first summer course session. ERIT expects a full release of the application to our subject liaisons and interested instructors in time for the Fall 2007 semester.

Future Plans

We expect to implement the default Adobe Flex/Java implementation of the tool campus-wide before the end of the Fall 2007 semester. As the Flex client goes fully live, the beta testing of the AJAX/JSON Web services client will continue until January, 2008, when it will move out of beta and be marketed to various course builders across the University as the preferred method with which to consume University Libraries Web services.

As a result of the University Libraries' successful work with UNCG's Blackboard Application Administrator in creating the Library Resources project, Web services and AJAX are being considered as the primary means through which Blackboard will accept interaction with outside

units within the UNCG community. In addition, there are early discussions regarding turning over the "Library" page in Blackboard, a static tab that links to our home page, to ERIT for development and administration within the Blackboard framework. This would allow for the potential development of expanded integration of targeted resources across the platform.

From here, a number of options are being considered for building upon the initiative and using the same administration tool to drive other services. For example, additional resources could be pushed into Blackboard using the same application. These might include on-screen contextual Instant Messaging, course-level federated searching, e-Reserve listings, and links to more extensive resources from within the library. JavaScript-based clients similar to the implementation in Blackboard could bring the Web service to departmental Web sites or be utilized to expand the reach to distance education students through other means.

The tools built for the Blackboard initiative also could be brought to bear within the library Web space, providing deeper integration with new AJAX and ASP.NET-driven course and subject guide dashboards. These might potentially feature built-in, on-page IM widgets, course and subject-related news via RSS feeds, and customizations based upon the selection of a specific course or subject.

Conclusion

Through the need to solve a basic technical concern, UNCG University Libraries has been able to leverage technologies such as SOAP and applications available on campus in a new way for academic libraries. By providing library data only, and employing user-adaptable clients, we have been able to tackle concerns about the presentation of library resources by moving them

beyond the library space into the user space where they can be customized to meet the needs of their given context. Through use of the Library Resources project, the University Libraries has begun integrating deeply into the user space, presenting the right resources at the point of need.

By bringing this project to its current stage, ERIT has been able to leverage various 2.0 technologies and concepts in an academic library environment. Web services, though an older technology, have only recently gained wider acceptance outside of large enterprises, especially when used in conjunction with a flexible Rich Application Interface (RAI) such as those achieved through the use of newer programming techniques such as AJAX and Adobe Flex. Through the portability of Web services, we can drive multiple interfaces across multiple platforms and servers from one resource - all updated simultaneously from a centralized, locally maintained database. The extensibility of this programming technique is limited only to the imagination of the implementation team, the bandwidth available, the immediate needs of the user, and a little common sense.

Additionally, through this tool the UNCG University Libraries has been able to address user needs as described in the OCLC 2003 Environmental Scan. Most users do not want to be confronted with the full range of resources; they only want the ones targeted to their current needs. This is achieved by the Library Resources project by pushing the resources deeply into the user's academic space at the course level. The tool allows users ready access to course-related library information while also providing a means to contact a librarian should the need arise.

Finally, we tried to make the client tools easy to use, with a minimum of links required for

accessing the Library Resources Building Block. We were able to achieve this ease of use both within the Blackboard hierarchy and within the process needed to get to the resources.

The Library Resources project addresses the primary, user-centric mission of the library. It also looks toward the future as we seek new ways to propagate our services beyond the library walls.

References

Adobe Systems Incorporated. 2007a. Adobe Flash Player version penetration. http://www.adobe.com/products/player_census/flashplayer/version_penetration.html.

_____. 2007b. TechNote: External data not accessible outside a Macromedia Flash movie's domain. http://www.adobe.com/cfusion/knowledgebase/index.cfm?id=tn_14213.

CGISecurity.com. 2002. The cross site scripting (XSS) FAQ. http://www.cgisecurity.com/articles/xss-faq.shtml.

Garrett, Jesse James. 2005. Ajax: A new approach to Web applications. http://www.adaptivepath.com/publications/essays/archives/000385.php.

United Parcel Service. Integrate tracking tools. http://www.ups.com/content/us/en/tracking/tools/.

United States Postal Service. 2005. Delivery information APIs.

http://www.usps.com/webtools/delivery.htm.

World Wide Web Consortium. 2007. Web services activity statement.

http://www.w3.org/2002/ws/Activity.