NEW BREED LIBRARIAN

V O L U M E 1 N U M B E R 6 ::: DECEMBER 2001 :::

FEATURE – Robin Paynter

A Reference Librarian's take on Library of Congress Classification

Robin Paynter is a Reference Librarian for the University of Oregon. She collects materials and teaches in the subject areas of Southeast Asian, Pacific Islands, and Judaic Studies. She is an LC Classification maven and continually impresses her colleagues with the in-depth, intricate, LC minutia she consistently pulls out of her astounding memory-bank.

PEOPLE

Michael Whang is a Library and Information Science graduate student at the University of Hawaii at Manoa. He is the founder and galvanizer of the Support Group for Information Architecture at his school.

ASK SUSU

Susu, our sometimes irreverent advice columnist, answers your questions about work, school, the job hunt, and librarianship in general. In this issue, Susu advises on the future of our profession.

TECH TALK

In this issue, Colleen shows you how to build a web-based form using PHP, one of the newest web programming languages. She firmly believes that PHP can make almost anyone with a moderate knowledge of HTML at least a novice web programmer.

LETTERS

Say What? If you have something to say, we want to hear it. **Send us** your letters and we'll post them here. If you're responding to an article or interview, place the headline in the subject of your email.

NEW BREED LIBRARIAN

F E A T U R E   A R T I C L E   ::: DECEMBER 2001 :::

A Reference Librarian's take on Library of Congress Classification

ROBIN PAYNTER

Robin Paynter is a Reference Librarian for the University of Oregon. She collects materials and teaches in the subject areas of Southeast Asian, Pacific Islands, and Judaic Studies. She is an LC Classification maven and continually impresses her colleagues with the in-depth, intricate, LC minutia she consistently pulls out of her astounding memory-bank. Even more, she loves this stuff!

Ideally, an organizational scheme would be timeless and logical. As we all know, the real world is far more complex and ceaselessly changing — here, the LCCS reflects both the time period of its creation and the work of succeeding generations of catalogers. Understanding these effects has made the system more comprehensible, albeit still frustrating when someone thinks something should have clearly been classified in one area ends up in another. For the sake of brevity, these general comments will be largely limited to only a couple of call number ranges: D and P. You can see a brief outline of the LCCS at **http://lcweb.loc.gov/catdir/cpso/lcco/lcco.html** for more detail.

During the time the LCCS was being composed, the world was a vastly different place — colonial powers ruled the majority of the world, imposed their languages on far flung nations, and the United States was "expanding," becoming a colonial power in its own right. D, E, F are the predominant classes for history (D: History: General and Old World, E: History: America, F: History: America). Some of the premises the originators endowed the system with have proven wise — for instance, giving a lot of room to the history of the United States (almost all of E, and a large part of F) because US libraries naturally would have available to them and collect extensively in the subject. D, on the other hand, is a relic of that colonial time period: the emphasis on European history (DA-DR history of individual countries of Europe, for instance Switzerland has all of DR) whereas the history of Asia, which in LCCS spans the present day Middle East to Japan, was given DS. Africa fared no better, the entire continent's history is classed in DT. The assumptions about the volume of publication on the non-European world have proven false over time (that was originally given as a reason for the paucity of space given to the subjects), however the scheme has not been able to adapt to these changed circumstances.

Another case in point: Hawaii. Around the time the LCCS was created, Hawaii's monarchy was overthrown by the US government, and later it became a state (1959). To this day, it is not included in the F section devoted to the history of the other 49 states — rather it remains classed in DU (History of Australia/Oceania). Looking at the broader scale, one finds that the order in which the history of world's regions are represented in D is paralleled in other classifications (and thus can be helpful to know when you are in the P section for instance): Western Europe first, Eastern Europe, Asia, Africa, Oceania.

The P section is THE section for language dictionaries and literary arts and hence very popular with students — I would recommend the Ps as one of the first areas to tackle! As with D, E, F the Ps likewise reflect the 'Western' colonial mentality prevalent during the time in which the LCCS was created: the language dictionaries start in PA ([Western] Classical languages, i.e. Greek and Latin) continue through Western European languages, Eastern European, Asian, Australian/Oceania, African, and Indigenous American. Fairly straightforward, with a little twist — literature written in non-European languages it was decided should follow the language dictionaries. So for instance, Chinese language dictionaries and literature are both in PL. Whereas for European languages (English, Romance languages, Germanic/Scandinavian) separate call numbers later in the Ps were developed just for their literature. Perhaps not a ruinous solution, given the likelihood that US libraries would have and indeed do have largely European language collections. Some interesting issues present themselves even so: where do you class an African writer who is writing in English? If you are Okot p'Bitek the answer is in both PL (African language and literature) and PR (British and Commonwealth literature); authors are thus classified according to which language they write in and/or where they live (or sometimes both), which can substantially effect shelf browsing for both staff and patrons.

Another interesting area of the Ps is the PNs (Literature, general), where in addition to general works on poetry, short stories, novels, and the theater — works on radio, television, and films are also classed. I assume that due to early radio often broadcasting plays among other things it was thought of as an extension of the theater; and likewise when early television broadcast dramas or plays it followed suit. I do not, however, know why the section begins with theater/drama proceeds to radio, television, film, and then back to theater again, most interesting.

The Zs are another matter and perhaps another article unto themselves. I have heard it said that they were developed when the Library of Congress Reference collection was a closed stacks system, so there was no need for shelf-browsing capability by patrons and that it was the Reference librarians who wanted bibliographies, national bibliographies, and indexes (the major research tools of the day) in an easily accessible area. This may be so, but over different generations of catalogers adhering to different schools of thought on classification multiplied by the changing nature of research and research materials available, the Zs have become somewhat of a black hole. I encourage you to go out and find the treasure trove of materials that have been classed in the Zs — particularly noteworthy are the literary arts and historical materials.

And now, a few of the more quirky tidbits: BM - Judaism, BS - The Bible, HV - Sign Language (classes with people with hearing disabilities as a social group rather than in the Ps as a language), and my favorite so far: 1970's throwback CB 156 - Terrestrial Evidence of Interplanetary Voyages. Of course, it is easy in hindsight to criticize the original underpinnings of the system or the classification of a particular piece; ultimately, it is our responsibility to understand the historical issues underlying why things are as they are so that we can work the system to benefit of our patrons, and so that we can work with technical services staff to improve classification of materials. As we have seen all too vividly with the Internet — having a subject classification scheme in place is better than no system at all.

Michael Whang is a Library and Information Science graduate student at the University of Hawaii at Manoa. He is the founder and galvanizer of the **Support Group for Information Architecture** at his school. Here, he tells us about the Group and of the efforts to incorporate it into the curriculum.

The Support Group for Information Architecture is the first student-based support group at the University of Hawaii dedicated to fostering growth, education, research, and practical experience in the field of information architecture — the art and science of organizing, scaling, structuring, and indexing Web sites and complex information systems. The Group was formed in March of 2001 with the hopes of aggregating students from varied disciplines across academia who were interested in developing web sites that focused on content and the end user. SGIA's purpose was to build community around a common thread — information architecture — and create a Petri dish for collaboration and idea exchange.

The organization is for students and by students who want to gain more experience in a collaborative and team-based environment, building better Web sites putting the user at the heart of the design process. The group consists of mainly LIS students, some with technical backgrounds, but mostly those students who are just breaking ground in learning basic HTML and publishing info on the Web.

The idea was inspired by me, a graduate student at the University of Hawaii School of Library and Information Science. I organized the SGIA because I wanted to aggregate others who felt passionate about web development and about building better, user-centered web sites. I wanted to build an online and physical community within the University that would allow this kind of free-flow idea exchange. It takes a lot of time and perseverance on everyone's part including students and faculty.

Selling IA and getting students involved at UH has been a hard sell. First, one of the difficulties I experienced here on campus was explaining what IA is to students and how their unique skill sets could contribute to the cause. The skill level related to web development has been very minimal among LIS students. So the technical language barrier is something that is somewhat omnipresent in our program.

Second, our program is trying to build a physical bridge to the computer science department. Computer science and LIS students are two disparate groups here on campus, each containing a culture very different from the other. Trying to create a collaborative environment between them has been difficult.

I know for a fact that one UH LIS student is working with a LIS professor this semester to build a working curriculum around IA. From what I've heard, the computer science department is planning to host its first tentative IA undergraduate course sometime in the fall of 2002.

Finally, the UH LIS program has currently four full-time instructors, each carrying a long-list of courses to teach and to prepare for. So, it's been hard for faculty to introduce some of these new, exciting ideas such as IA into their existing course load. That being said, I think it's important for students to take the initiative and conduct their own research, initiate their own workshops, and do what SGIA has tried to do. We design our own futures.

A S K **S U S U** ::: DECEMBER 2001 :::

Susu, our sometimes irreverent advice columnist, answers your questions about work, school, the job hunt, and librarianship in general. In this issue, Susu speculates on the future of librarianship.

Dear Susu, I am a library student. The topic I am writing my final paper on is, "Is Librarianship A Dying or Growing Profession?" Any opinions? To date, I have interviewed a new librarian, a very seasoned librarian, and lots of research. I'm a firm believer in "straight from the horse's mouth."

Dear Firm Believer, Any profession is a dying profession if it doesn't continue to build and evolve. Given today's ever escalating changes in information technology, this holds especially true for us. Librarians must be willing to morph and adapt if we hope to stick around and play vital roles in our communities, schools, and businesses. To maintain methods and practices from a decade ago, without any recognition of a changing environment, is to sink deep into a well of stagnation - one that reeks of staleness and ennui. How effective is that?

I've always believed that Library School equips students with conceptual foundations. We learn to select, classify, organize, evaluate, filter, and intermediate between user needs and available resources. We inculcate the importance of intellectual freedom and access. With this schema in place, we're prepared to go out in the real world and begin really learning by materializing these conceptions. If you're not learning in your work, you're not doing enough.Day-in day-out of same-same eventually places you in last place, far behind the folks who are making a difference because they get what's going on and care enough to do something about it. Case in piont: do we have an equivalent to AskJeeves and About.com? What are we doing to demonstrate our skills and unique value at large?

Brunella Longo — librarian, teacher, and entrepreneur — recently wrote an excellent piece entitled "**How a Librarian Can Live Nine Lives in a Knowledge-Based Economy**." In it, she shares valuable wisdom she's acquired throughout perpetual career-adaptation:

- Continuous learning and a constant passion for research have been my keys for the growth and development of my professional identity.
- I believe it is very important to have a plan for attaining new skills based on continuous learning, on motivation, on the vision of the whole.
- In sum, I believe that we should be aware of the dynamics with which human activities change because of technological innovations, and at the same time be aware of the dynamics with which we, at a personal level, accept, anticipate, and resist changes.

The key here is to have our radar on, to know our users, to be willing to adapt, keep up, stay current, and remain passionate. We need to keep our users in mind when we're redesigning our library sites: does the interface make sense to the user or is it perceptible only to us? When we teach, do we make it real or put our classes to sleep? Do we care enough about our communities to incorporate useful and pertinent programs that require additional work or stay with the tried, true, and tired? We need to keep asking ourselves these evaluative questions and then purposefully doing something about what we find.

Selecting, classifying, organizing, and evaluating are important skills, and they're ours. But with the argosy of information available and cozy corporate book chains equipped with coffee nooks, why should people bother using us if they can get their needs met conveniently elsewhere?

Don't get me wrong: I believe library as place is not going to vanish. But whether or not librarianship *thrives* depends on librarians themselves. If we're committed to life-long learning, engaged in our work, reflect and adapt to changes and attitudes infiltrating society and culture, are passionate, and most importantly, keep our users needs and wishes above the convenience of methods we're comfortable with, we'll define our boundaries, recognize our potential, and the profession will last a long, long time.

T E C H   T A L K

::: DECEMBER 2001 :::

In this issue, Colleen shows you how to build a web-based form using PHP, one of the newest web programming languages. She firmly believes that PHP can make almost anyone with a moderate knowledge of HTML at least a novice web programmer.

You're not a programmer, but you're reasonably comfortable with HTML. You want to be able to do some simple programming on the web (such as forms) without waiting for your project to move to the top of your IT folks' project list.

Take heart - there's a new programming language on the block, and it's working its way into the repertoires of programmers and non-programmers alike. It's called PHP, it's free, and it is completely compatible with HTML. But enough with the chit-chat - let's get down to business.

Web-based forms typically have two components - an HTML file that solicits information from a user, and a script that processes that information in some way. For example, a comment form typically solicits feedback from a user, then sends it to someone via email. For this column I've created a simple comment form in PHP and HTML, which I'll walk you through step by step, but first, three things you need to know about PHP:

1. PHP file names typically have a ".php" extension - check with your IT folks to see what they require.
2. PHP statements are included inside angle brackets, using the following structure: <?php *some PHP statement(s)* ?>. Unlike HTML, you can include multiple PHP statements inside the angle brackets.
3. In UNIX or LINUX environments, PHP files must be executable. If you don't know how to do this, talk to your IT folks.

OK, let's get started. I've created a simple web form that does the following:

1. Displays a web form with three fields: name, email address, and comments box.
2. Once the user fills in the box and clicks the submit button, emails the input to someone, then displays a thank you note on the screen.

Below is the HTML and PHP code (let's name it *form.php*) that will accomplish this:

```
1: <?php
2:    if ($FormSubmit) {
3:       process_form(); # $FormSubmit is not blank
4:    } else {
      display_form(); # $FormSubmit is blank
   } #end of if ($FormSubmit)

   function display_form() { # Display form for user input
      global $PHP_SELF;
?>
<html>
<head>
   <title>Comment Form</title>
</head>

<body>
<h1>We Want To Hear From You</h1>

<p>Please fill in the form below to tell us what you think; you will
receive a response within 24 hours.</p>

<form action="<?php echo $PHP_SELF; ?>" method="post">
<p><input type="text" name="Name" size="30"> Your Name<br />
<input type="text" name="Email" size="30"> Your Email Address</p>

<textarea name="Comments" cols="65" rows="3"></textarea><br />
<input type="hidden" name="FormSubmit" value="Yes">
<input type="submit" value="Submit Your Feedback">
</form>
</body>
</html>
<?php
   } #end of function display_form()

   function process_form() { # Process form after user clicks on the submit button
      global $Name;
      global $Email;
      global $Comments;
      global $FormSubmit;

#Email comments to someone
      $to = "noone@somewhere.com";
      $subject = "Feedback Form";
      $message = "Name: $Name\nEmail: $Email\nComments: $Comments";
#This line is optional, but useful for replying to the sender
      $header = "From: $Name <$Email>\r\nReply-To: $Email\r\n";
      mail ($to, $subject, $message, $header);

#Provide web response to user
?>
<html>
<head>
   <title>Thank You!</title>
</head>

<body>
<h1>Thank You!</h1>

<p>Thank you for your feedback - here's what you wrote:</p>
<hr />
<p><strong>Name: </strong><?php echo "$Name"; ?><br />
<strong>Email: </strong><?php echo "$Email"; ?><br />
<strong>Comments: </strong><?php echo "$Comments"; ?></p>
<hr />
<p>If you included an email address, you should receive a reply within
24 hours. Thanks again for your feedback.</p>
</body>
</html>
<?php
   } #end of function process_form()
?>
```

The first thing you'll probably notice is that HTML looks like HTML, and PHP doesn't. To help you visualize these differences, I've used color-coding: red for HTML, black for PHP, blue showing you where the PHP code starts and ends, and yellow for comments (information that is useful for anyone reading the program, but not part of the program itself).

Most programs are written in chunks, starting with the easiest, smallest task and building from there (just like web publishing). Let's take a look at this script, piece by piece.

Lines 11-31: this section should be easily recognizable as an HTML document; note the HTML element at the beginning and end. If you look closely at the code, you'll see that it creates a simple **web form**.

There are two special features of this web form:

1. **Line 22** uses a bit of PHP code to define the action in the FORM element. $PHP_SELF is a special PHP variable (placeholder for data) that refers to the current document. When the user clicks on the form's submit button, the server looks to this same file for instructions on what to do next.
2. **Line 27** contains a hidden variable ($FormSubmit) that is used to determine whether the form has been filled out or not.

Lines 51-68 should also be easily recognized as an HTML document, this time a response thanking the user for the feedback and letting the user know what was sent. Note the use of PHP in **lines 60 and 62**; the variable names correspond to the ones used in the HTML form (see lines 23, 24 and 26). Variables in PHP are indicated by using a dollar (or string, $) sign in front of the variable name. Note also the use of angle brackets to format the user's email address.

Lines 8, 33: these lines of code form a wrapper, called a function, around a piece of a program. Note the name of the function - display_form() - which is descriptive enough to give you an idea of what the function does, in this case display the web form for user input. A comment in plain English (notice that it begins with a #) provides additional information to the person reading this program. Functions are a very important part of programming in PHP - they allow you to group statements (commands) together to accomplish a purpose, then execute that group of statements in another part of the program. This allows you to create extremely efficient programs - experienced programmers always look for ways to accomplish the task with the least amount of code.

Line 9: this is a declaration that indicates that the $PHP_SELF variable will be used in the function. The use of the "global" keyword indicates that this variable is available throughout the program; if "global" were not included, this variable would be local only to this particular function or a particular statement.

Lines 35, 70: another function wrapper, this time for the process_form() function.

Lines 36-39: these variable declarations provide placeholders for the data passed from the web form; declaring them here actually reads the data in from the form. Note that the names correspond exactly to the names used in the form (lines 23-24, 26-27), with the dollar sign prefix indicating a PHP variable. Names in PHP are case sensitive - $Comments and $comments are two separate variables - so it's important to be careful. Consistency is an important part of programming style; most programmers adopt naming conventions for variables and functions. In my case, I capitalize variable names, and in the case of compound names like $FormSubmit I capitalize each individual part of the name; for functions I use lower case and underscores, but I could just as easily use something like displayForm() instead of display_form(). The parentheses in function names serve a special purpose, as you'll see in the next section, just like the dollar sign in front of a variable name. Many programmers also adopt styles for spacing, indenting, and comments. The most important thing is to be consistent.

Lines 42-47: these 5 lines create the body and headers of an email message, then send it. The email recipient ($to) used here is a dummy one - you would obviously change this to a real address before you implemented the form. **Line 44** creates the body of the email message. The '\n' creates a line break, and begins the text following it on a new line (it's often referred to as the newline character). **Line 46**, which is optional, provides some additional email headers, in this case the From: and Reply-to: fields, which allows the recipient to use their email software's Reply feature. The '\r' before the newline character in this line is required for UNIX mail servers, so be sure to include it. Finally, **line 47** sends the email message. The **mail()** function is a built-in PHP function.

There's just one final step before we can run our program, and that's to actually execute the functions we've created. This is where that hidden variable ($FormSubmit) from our web form comes into play.

Lines 2-6 are the keys to this little script. They introduce a condition into our program that essentially reads, "If the $FormSubmit field is not blank (in other words, it exists), then go ahead and process the form; otherwise (else), just display the form." The first time the program is run, it should skip the first part of the conditional statement and display the form, because it doesn't find the $FormSubmit variable. But once the form has been displayed, that hidden field fills in a value for $FormSubmit, and the next time the program is run (when the user clicks on the form's submit button), the program processes the form as instructed.

And that's it - all that's left is to try it out, but since our server doesn't currently support PHP, you'll have to put it on your own local server to do that (make sure you change the email address to your own so that you see how it works).

As you can see, it's not complicated when you break it down. Of course, if this were a form you were planning to implement, you'd probably want to add some additional features, like the ability to make corrections to the data before it's submitted, add some security to the form by validating a user's email address before the information is sent by email, and check to make sure that required fields have been filled out so that incomplete information isn't sent on to the recipient.

What we've seen here is just the tip of the iceberg. One of PHP's greatest strengths is its ability to integrate HTML and SQL databases into web-based database applications - that's when you really start to cook. But that's a whole series of articles that maybe one day I'll write...

Related Links:
PHP (the official home of PHP on the web)
PHPBuilder (articles, forums, code library, mail archive)
Introduction to PHP (C|Net)
Code for this script

L E T T E R S ::: DECEMBER 2001 :::

15 JAN 02 :: **"The Other MLS in the Northwest"**

I love your website, but notice that your 1/11/02 issue [news item] called the forthcoming UW MLS program the only MLS in the Northwest. Don't forget Emporia State University's cohort model MLS program. Although ostensibly a distance program, Emporia differs from other programs, such as Syracuse, because teachers fly to Portland and give "weekend intensive" classes - one weekend per cedit hour. Since we regularly meet face-to-face with a variety of instructors (and each other), our program is not exactly a "distance" program.

Thanks,
Rachel Mendez
Oregon III Cohort, Emporia State University
OLAweb Editor
Reference, Belmont Library, Multnomah County Library

NBL responds: We've been seeing quite a bit of traffic about this announcement. As the **press release** states, UW *is* the only library school in the northwest, and thus the only one in the northwest able to offer an online program. We realize that several states in the northwest, like Oregon, have imported programs like the Emporia one, and that several folks in the northwest have chosen to obtain their MLS at a distance from many different institutions outside the northwest.