How to get more from your Ex Libris Products:

Using the LORLS open source reading list management system as an example

Gary Brewerton

Middleware & Library Systems Team (IT Services)

Loughborough University

Background: journey into the unknown

- In 1999 Loughborough University's Learning & Committee decided that reading lists should be made available online to students

- Key requirements:
  - Allow academics to maintain their own online reading lists
  - Allow students to easily check Library availability of resources on reading lists

- Library Systems Team was tasked with developing the solution

Background: launch of reading list system

- We launched the Loughborough Online Reading List System (**LORLS**) in June 2000 alongside our existing LMS, although we were out to tender for a new LMS
  - To add new items to a reading list, academics could either manually fill in an online form or enter an ISBN and pull data from the LMS using a bespoke Perl CGI script
  - Provided [Check Library holdings] links against all citations to deep link into Library catalogue (regardless of holdings)

Loughborough University

Change of LMS: getting things working again

- Less than a year after implementing **LORLS** the Library replaced the prior LMS with **Aleph** - O# cr@p!

- Getting **LORLS** to work with **Aleph** was (thankfully) very straight forward

  - Supported deep linking into the catalogue using **REST** (e.g. http://aleph.lboro.ac.uk/F/?func=find-b&request=1565922840) ✓

  - Provided a **Z39.50** interface to the LMS which allowed us to standardize our method for data retrieval ✓

Change of LMS: additional improvements

- With a standard (**Z39.50**) data retrieval method we were able to create an overnight job to pre-check library holdings (no more dead ends)

- Alongside **Aleph** the Library also purchased this weird **SFX** thingy (very easy to integrate)

Example: subroutine to generate OpenURL (v0.1)

```perl
sub generateOpenUrl{
  my ($metadata)=@_;
  my $openURL='http://sfxeu10.hosted.exlibrisgroup.com/loughborough?';

  my @details;

  if($metadata->{'genre'}){push @details, "genre=".uri_escape($metadata->{'genre'})}
  if($metadata->{'issn'}){push @details, "issn=".uri_escape($metadata->{'issn'})}
  if($metadata->{'isbn'}){push @details, "isbn=".uri_escape($metadata->{'isbn'})}
  if($metadata->{'volume'}){push @details, "volume=".uri_escape($metadata->{'volume'})}
  if($metadata->{'issue'}){push @details, "issue=".uri_escape($metadata->{'issue'})}
  if($metadata->{'journalTitle'}){push @details, "title=".uri_escape($metadata->{'journalTitle'})}
  if($metadata->{'articleTitle'}){push @details, "atitle=".uri_escape($metadata->{'articleTitle'})}
  if($metadata->{'startPage'}){push @details, "spage=".uri_escape($metadata->{'startPage'})}
  if($metadata->{'endPage'}){push @details, "epage=".uri_escape($metadata->{'endPage'})}

  $openURL.=join('&',@details);

  return $openURL;
}
```

Example: invoking the OpenURL subroutine

```perl
#!/usr/bin/perl

use URI::Escape;

my $item={
  'genre'=>'article',
  'issn'=>'0305-5728',
  'issue'=>'4',
  'volume'=>'33',
  'journalTitle'=>'VINE',
  'articleTitle'=>'From local project to open source: a brief history of the Loughborough Online Reading List System (LORLS)',
  'startPage'=>'189',
  'endPage'=>'195'
};


print generateOpenUrl($item)."\n";
```

Loughborough University

Redevelopment: time to re-think reading lists

- In 2007 we started considering how to re-develop **LORLS**, development began in 2009 and we released the new version in February 2011

- Key considerations were:

  - Greater flexibility (e.g. support organisational structures other than Loughborough University)

  - Better interface design (e.g. drag and drop re-ranking)

  - Embedding of additional content in system (e.g. library holdings, book covers)

Web services: getting at the information

- Re-developing **LORLS** allowed us to take advantage of web services

  - For example: extracting holdings data from the **Aleph X-Server** in **XML** format, and embedding it in the reading list screen

Example: Aleph X-Server call and results

- http://aleph.lboro.ac.uk/X?op=circ-status&sys_no=001212234&library=lbo01&user_name=<USERNAME>&user_password=<PASSWORD>

```
<circ-status>
  <item-data>
    <z30-description/>
    <loan-status>Long loan</loan-status>
    <due-date>On Shelf</due-date>
    <due-hour>22:00</due-hour>
    <sub-library>Library</sub-library>
    <collection/>
    <location>006.76/LEA</location>
    <pages/>
    <no-requests/>
    <location-2/>
    <barcode>0403866928</barcode>
    <opac-note/>
  </item-data>
  <item-data>
    <z30-description/>
    <loan-status>Long loan</loan-status>
    <due-date>25/10/13</due-date>
```
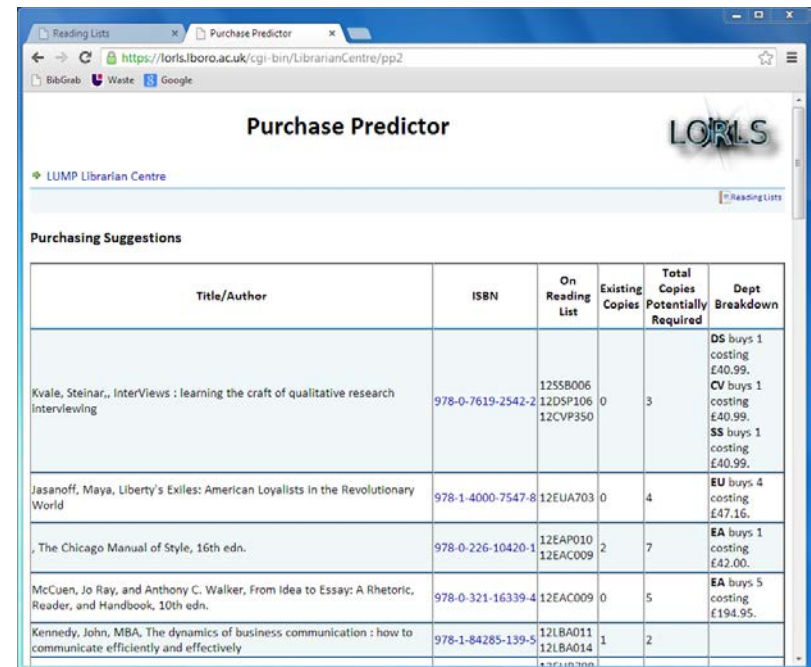
```
    <due-hour>22:00</due-hour>
    <sub-library>Library</sub-library>
    <collection/>
    <location>006.76/LEA</location>
    <pages/>
    <no-requests/>
    <location-2/>
    <barcode>0403866936</barcode>
    <opac-note/>
  </item-data>
  <session-id>
    8D7DJ2B34RG697D…2QIS8R21EI6FKR2
  </session-id>
</circ-status>
```

Purchasing prediction: Big-ish data

- ## Step-by-step guide to how it works

  1. Identify items to check (e.g. recently edited items) from **LORLS**
  2. Validate ISBNs
  3. Estimate initial copies required from item importance and student numbers in **LORLS**
  4. Drop books which have sufficient copies in stock according to **Aleph**
  5. Find costs (from **Aleph**, Amazon, GoogleBooks, etc.)
  6. Discard books without available prices
  7. Get recent loan information from **Aleph**
  8. Drop underused items
  9. Identify latest edition of books
  10. Round robin purchasing algorithm to keep purchasing suggestions within **Aleph** budget
  11. Output results (onscreen or email)

Loughborough University

bX: adding article recommendations

- When the Library purchased **Primo** they also purchased **bX**
  - Used with **Primo** and **SFX**
  - Added option to **LORLS** dashboard to see suggested articles for inclusion and allow editors to append to a reading list with a single click