Awash in a sea of connections: making Evergreen talk

Galen Charlton, Equinox Software
28 April 2011

This and that

- A little planning, a little technical

- Linking with online services

- Ye olde legacy protocols

- Pulling data in

- Pulling data out

Planning

- Priorities

- Testing

... and technical

- OpenSRF: the Swiss army chainsaw of talking to Evergreen

- ... this is not about OpenSRF

- ... in fact, pretend that OpenSRF doesn't exist – alas, most systems are uncouth and don't speak it

Linking with online services

- Book jackets

- "Enhanced content"

- Whither an ISBN/OCLC #/UPC goes, linky goodness follows

Ye olde legacy protocols

- SIP2

- Z39.50

- SRU/W

- unAPI

Pulling data in

- Bibliographic data

- Patron data

Pushing data out

- Discovery interfaces

- A data feed is just another name for a report

Specific examples

- Book jackets

- SIP2

Book jackets

- Out of the box
  - OpenLibrary, Syndetics, ContentCafe, Amazon
- Adding another

Book jackets out of the — opensrf.xml
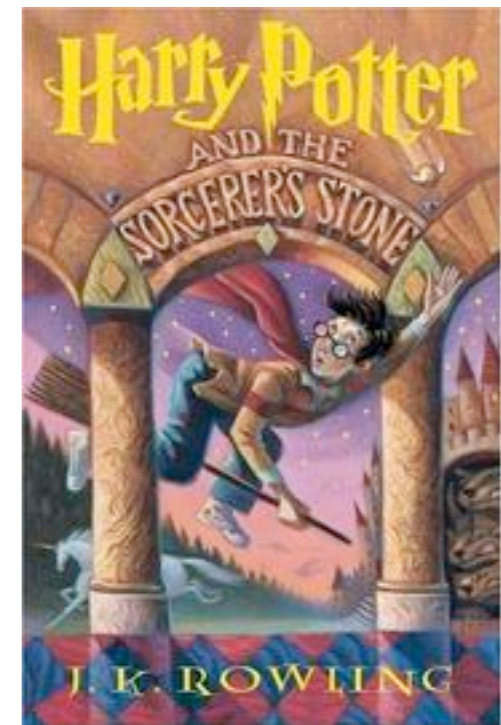
```
<added_content>

    <!-- load the OpenLibrary added content module -->

    <module>OpenILS::WWW::AddedContent::OpenLibrary</module>

    <!--

    Max number of seconds to wait for an added content request

    to return data.  Data not returned within the timeout is
    considered a failure

    -->

    <timeout>1</timeout>
```

JAAS

http://<evergreen>/opac/extras/ac/jacket/medium/9780590353403

- From ISBN (or soon, thanks to Jeff Godin, bib ID) to cover image

- "JAAS" gives us caching, and will let us do fancier tricks

JAAS is not just jackets

- Let's look at that URL again:

  http://<evergreen>/opac/extras/ac/jacket/ medium/**9780590353403** is really

  http://<evergreen>/opac/extras/ac/type/ format/identifier

Added content handlers

```
Open-ILS/src/perlmods/lib/OpenILS/WWW/AddedContent/OpenLibrary.pm

sub jacket_small {

    my( $self, $key ) = @_;

    return $self->send_img(

        $self->fetch_cover_response('-S.jpg', $key));

}

...

sub ebooks_html {

    my( $self, $key ) = @_;

    my $book_data_json = $self->fetch_data_response($key)->content();

    $logger->debug("$key: " . $book_data_json);
```

Adding added content handlers

- Figure out how to fetch the content – usually some kind of web service

- OpenILS::WWW::AddedContent::Foo with following methods

  - new($config)

  - *type_format*($key)

New jacket source

- Should supply jacket_small(), jacket_medium(), and jacket_large()

Another approach

- Instead of using the enhanced content handler, supply a <div> and use JavaScript provided by the content source

- Examples: LibraryThing, Syndetics Plus, ChiliFresh

Example: ChiliFresh

```
eg_vhost.conf:

    # Specify a ChiliFresh account to integrate their services with the OPAC

    #SetEnv OILS_CHILIFRESH_ACCOUNT

    #SetEnv OILS_CHILIFRESH_PROFILE

    #SetEnv OILS_CHILIFRESH_URL http://chilifresh.com/on-site/js/evergreen.js
```

Example: ChiliFresh

```
Open-ILS/web/opac/skin/default/xml/rdetail/rdetail_extras.xml:
    <!-- ChiliFresh setup -->
    <script language='javascript' type='text/javascript'>
        var chilifresh = '<!--#echo var="OILS_CHILIFRESH_ACCOUNT"-->';
        if (chilifresh == '(none)') { chilifresh = false; }
    </script>
    <!--if expr="${OILS_CHILIFRESH_ACCOUNT} && ${OILS_CHILIFRESH_ACCOUNT}!='(none)'"-->
        <input type="hidden" id="chilifresh_account" name="chilifresh_account"
            value="<!--#echo var='OILS_CHILIFRESH_ACCOUNT'-->" />
        <input type="hidden" id="chilifresh_profile" name="chilifresh_profile"
            value="<!--#echo var='OILS_CHILIFRESH_PROFILE'-->" />
        <input type="hidden" id="chilifresh_version" name="chilifresh_version"
            value="onsite_v1" />
        <input type="hidden" id="chilifresh_type" name="chilifresh_type" value="search" />
        <script language="javascript" type="text/javascript"
            src="<!--#echo var='OILS_CHILIFRESH_URL'-->"></script>
    <!--endif-->
```

Example: ChiliFresh

```
Open-ILS/web/opac/skin/default/js/rdetail.js:
    // ChiliFresh
    if (chilifresh && chilifresh != '(none)' && currentISBN) {
        $('chilifreshReviewLink').setAttribute('id','isbn_'+currentISBN);
        $('chilifreshReviewResult').setAttribute('id','chili_review_'+currentISBN);
        unHideMe($('rdetail_reviews_link'));
        unHideMe($('rdetail_chilifresh_reviews'));
        try {
            chili_init();
        } catch(E) {
            console.log(E + '\n');
            hideMe($('rdetail_reviews_link'));
            hideMe($('rdetail_chilifresh_reviews'));
        }
    }
```

Data quality is key

- Records must have accurate identifiers

- Of course, quality of enhanced content is key

  - OpenLibrary allows for sharing and improving its cover images and metadata

SIP2

- Self-checkout, patron authentication (e.g., ezProxy), PC management

- Old-school standard: fast, not verbose

- But also: can easily be set up insecurely

SIP2 setup

- How many devices and services?

- Raw vs. Telnet (but either way, secure it!)

oils_sip.xml

```xml
<listeners>

  <service
    port="8023/tcp"
    transport="telnet"
    protocol="SIP/1.00"
    timeout="60" />

  <service
    port="127.0.0.1:6001/tcp"
    transport="RAW"
    protocol="SIP/2.00"
    timeout="60" />
</listeners>
```

oils_sip.xml

```xml
<institutions>

        <institution id="gapines" implementation="OpenILS::SIP">

                <!-- This defines what actions we want to allow
                        remote clients (self-check machines) to perform -->
                <policy
                        checkin="true"
                        checkout="true"
                        renewal="true"
                        status_update="false"
                        offline="false"
                        timeout="600"
                        retries="3"/>
```

oils_sip.xml

```xml
<accounts>
        <login id="scclient" password="clientpwd" institution="gapines"/>
</accounts>
```

SIP2 flow

- SIP request to OpenNCIP

- OpenNCIP parses request, then passes off to Evergreen's OpenNCIP driver module

- The driver makes the appropriate OpenSRF requests, then it's back up the stack

General points

- You don't have to teach OpenSRF just to get data out of Evergreen ... you can build services on top of OpenSRF

- The better your data, the better you can make connections

Thanks!

Galen Charlton

VP for Data Services, Equinox

gmc@esilibrary.com