PowerSchool API

A Real-World Example

Valley Christian Schools

Presented at PSUGCAL, October 16, 2015

LIBRO

- LIBRO – Spanish for “book”
- Lightly Integrated Bookstore Resources Organizer

- Web-based application.
- Replaces a custom Point-of-Sale, DOS-based application; unsupported.
- Written in PHP/MySQL (w/ jQuery) on your typical LAMP stack with SSL Cert installed.

PowerSchool API - PowerQueries

- API - Application Programming Interface, allows one program to access functions and resources on another
- LIBRO uses the API to access information in PowerSchool
- PowerQuery – a defined SQL statement (SELECT) made available to the API
- LIBRO needs data across multiple related tables
  - With the traditional API, a combination of several API calls
  - With PowerQueries, it’s one API call

Steps to Utilizing the API

- Decide what data you need
- Write and test your SQL statement
  - using SQL developer
- Create the PowerQuery as a Plug-In
- Install the Plug-In
- Test the PowerQuery using a REST client
  - such as Advanced Rest Client, or node.js
- Incorporate the PowerQuery into the application

SQL Statement: Multiple Tables

```
SELECT
    SUBSTR(cc.termid,3,2)||'-'||SUBSTR('0'||cc.expression,INSTR(cc.expression,'(')-1 ) as term_exp,
    sec.course_number,
    cor.course_name,
    cc.sectionid,
    users.last_name,
    users.first_name

FROM cc
INNER JOIN students stu ON stu.id=cc.studentid
INNER JOIN sections sec ON sec.id=cc.sectionid
INNER JOIN courses cor ON cor.course_number=sec.course_number
INNER JOIN schoolstaff ss ON ss.id=sec.teacher
INNER JOIN users ON users.dcid=ss.users_dcid

WHERE
    stu.student_number= :student and ROUND( cc.termid / 100 )= :year - 1990

ORDER BY
    SUBSTR('0'||cc.expression,INSTR(cc.expression,'(')-1,2),cc.termid
```

SQL Statement: Single Table

```
1   SELECT
2       student_number,
3       first_name,
4       last_name,
5       grade_level
6   FROM students
7   WHERE
8       grade_level>8 AND grade_level<13
9       AND enroll_status = 0
10      AND student_number > :lastnumber
11  ORDER BY student_number
12
```

Create the Plug-In Files

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin xmlns="http://plugin.powerschool.pearson.com"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://plugin.powerschool.pearson.com plugin.xsd"
        name="LIBRO Integration"
        version="0.5"
        description="LIBRO - Integration PlugIn">
    <oauth></oauth>
    <access_request>
        <field table="STUDENTS" field="ID" access="ViewOnly" />
        <field table="STUDENTS" field="STUDENT_NUMBER" access="ViewOnly" />
        <field table="STUDENTS" field="FIRST_NAME" access="ViewOnly" />
        <field table="STUDENTS" field="LAST_NAME" access="ViewOnly" />
        <field table="STUDENTS" field="GRADE_LEVEL" access="ViewOnly" />
        <field table="STUDENTS" field="ENROLL_STATUS" access="ViewOnly" />
    </access_request>
    <publisher name="Valley Christian Schools">
        <contact email="support@vcschools.org"/>
    </publisher>
</plugin>
```

```
<queries>
    <query name="org.vcschools.libro.school.hs_students" coreTable="students" flattened="false">
        <description>Active High School Students</description>
        <args>
            <arg name="lastnumber" type="primitive" required="true" />
        </args>
        <columns>
            <column column="student_number">StudentID</column>
            <column column="first_name">FirstName</column>
            <column column="last_name">LastName</column>
            <column column="grade_level">GradeLevel</column>
        </columns>
        <sql>
            <![CDATA[
            SELECT
                student_number,
                first_name,
                last_name,
                grade_level
            FROM students
            WHERE
                grade_level>8 AND grade_level<13
                AND enroll_status = 0
                AND student_number > :lastnumber
            ORDER BY student_number
            ]]>
        </sql>
    </query>
</queries>
```

Install the Plug-In

- Package the XML files into a ZIP file.
  - Root contains the “plugin.xml” file
  - Subfolder named “queries_root” contains the XML file that defines the PowerQueries (libro.queries.xml)
- In PowerSchool (System->System Settings->PlugIn Management Configuration), install the plug-in.
- After it installs, check the box to enable it.
- Open the PlugIn and get info needed:
  - the Client ID
  - the Client Secret.

Testing your PowerQuery

- Use a REST client
  - Postman
  - Advanced Rest Client
  - node.js – command line Javascript processor
- Get the PlugIn Client ID and Secret
  - Created when the PlugIn is enabled
- Create a base64encoded string with the ID and Secret
  - “<ClientID>:<ClientSecret>”
  - Use encoder in your development environment or www.base64encoder.org

Request OAuth Token

URL:

https://<ps_server>/oauth/access_token

Request Type:

POST

Headers:

Authorization Basic NzE2 … hiOA==

Content:

grant_type=client_credentials

Access the PowerQuery

URL:

https://<ps_server>/ws/schema/query/org.psugcal.ps8.school.hs_students

Request Type:

POST

Headers:

Authorization	Bearer 420035ea-6a1f-461c-ae33-83f270b8f352
Accept	application/json

Content:

```
{ "lastnumber": 0 }
```

Results


```
{
"name":"students",
"record":[
    {
        "name":"students",
        "tables":{
            "students":{
                "gradelevel":"12",
                "lastname":"Anderson",
                "firstname":"Trevor",
                "studentid":"1"
            }
        }
    },
    {
        "name":"students",
        "tables":{
            "students":{
                "gradelevel":"12",
                "lastname":"Adair",
                "firstname":"Brandonn",
                "studentid":"3"
            }
        }
    },

    ....

],
"@extensions":"c_studentlocator,s_stu_crdc_x,activities"
}
```

Using the PowerQuery

- Where can PowerQueries be used?
  - Directly in custom pages as data objects for dynamic elements (AngularJS Controllers)
  - Automated exports using command line tools (node.js)
  - Resources for data visualization tools
  - Data Export Manager in PowerSchool
    - In 9.1 this can be scheduled!
  - Custom applications (like LIBRO)

- Development Environments
  - PHP/MySQL
  - ASP / Visual Studio
  - xCode (iOS Apps)

```php
    public function get_hs_students($last_student_number = 0) {
            // uses a Powerschool Named Query (org.vcschools.libro.school.hs_students)

            $resource = $this->ps_url."/ws/schema/query/org.vcschools.libro.school.hs_students";
            $payload = '{ "lastnumber" : '. $last_student_number .' }';

            // set options for POST call
            $opts = array('http' =>
              array(
                'method'  => 'POST',
                'header'  => "Content-Type: application/json\r\n".
                  "Authorization: Bearer $this->access_token\r\n",
                'content' => $payload
              )
            );

            // call the server's oauth gateway
            $result = file_get_contents($resource,false,stream_context_create($opts));

            // get the JSON data
            $jsondata = json_decode($result, true);

            // collapse the array a bit if there is data
            $hs_students = array();
            if (isset($jsondata['record']))
                foreach($jsondata['record'] as $item) $hs_students[]=$item['tables']['students'];

            //echo '<pre>' . print_r($hs_students, true) . '</pre>';
            //exit;

            return $hs_students;
    }
```

Hands On Time

- Create base64 encoded string
- Getting the Bearer token
- Get student info (API v.1.0)
- Get student info with extensions
- INSERT a new student record
- UPDATE a student’s name
- Call a PowerQuery

Additional Resources

- PowerSchool Developer
  - http://support.powerschool.com/developer
- Base64 Encoder
  - http://www.base64encode.org
  - Do not use this for plugins on your production server.
- Rest Clients
  - node.js – http://nodejs.org (command-line)
  - Advanced Rest Client – http://