使用 FOLIO API 的经验*

盖伊·多布森（德鲁大学）
张喆昱[1] 张春景[1] 胡小菁[2]（译）（1 上海图书馆 2 华东师范大学）

摘 要 API（应用程序编程接口）提供了实现需要完成工作的能力。FOLIO 将 API 作为其构建模块之一使得其更具吸引力。当德鲁大学图书馆的管理层决定从传统的 ILS（图书馆集成系统）转换到 FOLIO LSP（图书馆服务平台）时，作者首先关注的便是 API。这些经验有助于对系统进行配置，顺利将数据从 ILS 导出，并输入到 FOLIO。通过构建 web 应用和 Perl 脚本，工作人员可以完成无法通过用户界面（UI）来开展的相关工作。

关键词 API FOLIO API Web 应用

Lessons Learned about FOLIO's APIs

Guy Dobson (Drew University)
Zhang Zheyu[1], Zhang Chunjing[1], Hu XiaoJing[2] Trans. (1 Shanghai Library; 2 East China Normal University)

Abstract APIs (Application Programming Interface) provide the ability to do what needs to be done. The fact that FOLIO includes API as one of its building blocks makes it that much more attractive. When my library's administration decided to switch from a legacy ILS(Integrated Library System) to a FOLIO LSP (Library Services Platform) the first thing that I looked at was the API. The lessons learned helped me to configure the system and massage the data from ILS output to FOLIO-friendly input. By building web applications and writing Perl scripts our staff is able to get the job done even when it is impossible to accomplish the task through the user interface (UI).

Keywords API, FOLIO API, Web Apps

1 探索 FOLIO API

通过编写 Perl 脚本使用 curl 向 FOLIO 发送 API 调用的方法来探索 FOLIO 的 API。每次 API 调用都需要一个 Okapi①（这是 FOLIO 的多租户 API 网关）令牌，所以首先需要学习如何拿到一个 Okapi 令牌。通过使用 $tenant, $username 以及 $password 向 https://[FOLIO 路径]//authn/login 发起 API 调用，能从结果中解析出 Okapi 令牌，并用它定义一个名为 $okapiToken 的变量，如下所示：

```
$post = `curl -i -s -X POST -H 'Content-type: application/json'
    -H 'X-Okapi-Tenant: $tenant' -d '{"username": "$username", "password": "$password"}' $url/authn/login`;
    @parts = split(/\n/,$post); foreach $part (@parts) {
    if ($part =~ /^x-okapi-token:/) {
    $okapiToken = "X-Okapi-Token: " . substr($part,15);
    }
    }
```

* 本文根据《国际图书馆学报》(International Journal of Librarianship, 简称 IJoL)2021 年 12 月 FOLIO 特刊发表的英文论文做全文翻译。原文出处：Dobson, G. (2021). Lessons learned about FOLIO's APIs? International Journal of Librarianship, 6(2):30–39. https://doi.org/10.23974/ijol.2021.vol6.2.206.

① https://github.com/folio-org/okapi

在提交以下 cual API 调用后：

```
$json = `curl -s -X GET -H 'Content-type: application/json' -H '$okapiToken' $url/coursereserves/terms?query=name=Fall 2021`;
```

接着用 Perl 的 JSON 模块对 $json 变量进行解码：

```
$hash = decode_json $json;
for ( @{$hash->{terms}} ) {
$termId = $_->{'id'};
}
```

并从 JSON（JavaScript 对象格式）文档中解析出需要的数据。

经验：无须在每次发起新的 API 调用时都获取新 Okapi 令牌。获取一次并保存成文件，如 okapiToken.txt，就可以重复使用，这有助于 Perl 脚本运行得更快。只有在需要给相关用户授予新权限或升级到新版本的 FOLIO 时才需要获取新令牌。

使用上述技术，可以查看许多数据库表并理解它们之间的关系。作者浏览了 API 文档[①]网站，向各种端点发送了 GET 调用，将生成的 JSON 文档与上述文档以及在 UI 中看到的内容进行比较，并做了笔记。

经验：在 UI 设置程序中找不到贡献者的名称类型（与 MARC（机读目录记录）100、110 和 111 等字段相关联的标签）。必须通过 API 对它们进行添加、编辑或删除。

使用 EBSCO 构建的测试环境时，已经预先配置了参考（示例）数据；在加载书目或读者数据之前需配置的表已经设置好了。甚至预先配置了示例机构哥本哈根大学的读者和图书的测试数据。一些参考数据可能在各个机构之间是相同的，因为它们只是描述 MARC 记录中各个字段的标签。那些包含 RDA（资源描述与检索）标签的表也是如此。像是馆藏地这样的数据则需要由每个图书馆来定制。

仔细观察参考数据发现，交替题名的类型列表并不像一部分图书馆馆员和工作人员所希望的那样完整。它包括“统一题名”（即 130、240 或 730）和为 246 的第二指示符所列出的选项，但它不包括与其他题名字段（210、222、242、243、247 或 740）相关的选项。出于这个原因，决定不使用参考数据，并且使用 Perl 脚本将这些标签加载到测试环境中，这样所要做的就是更改脚本的 URL（统一资源定位器）和租户，以便 EBSCO 在设置时将相同的标签加载到生产环境中。

2 配置数据库表

借助美国国会图书馆的 MARC 21 书目数据格式[②]网站，能够找到 FOLIO 的几个表格的内容，比如上面提到的交替题名类型和贡献者名称类型。其他包括：

- 贡献者类型
  - 一共 278 个—参见 MARC 关系人代码表
- 格式
  - 即 RDA 载体类型[③]
- 标识符
  - MARC 的 01X–09X
- 发行方式
  - 见 MARC 头标第 7 位
- 实例注释类型
  - 5XX
  - 我们决定在标签中加入 MARC 字段，例如论文附注（502）
- 资源类型
  - 即 RDA 内容类型[④]
  - MARC 的 336 字段
  - 通过 API 在 instanceTypes 表中找到
  - 创建实例的必备字段
- URL 关系
  - 856 的第二指示符

经验：FOLIO 实例不是一条 MARC 记录。它是实例表中的一行，可以通过收集包含题名、作者等和各种外键的 JSON 文档来创建和 / 或编辑它。外键包括：来自描述实例内容的其他表中的 UUID（例如交替题名类型、贡献者类型、标识符等。）

① https://dev.folio.org/reference/api/
② https://www.loc.gov/marc/bibliographic/
③ https://www.loc.gov/standards/valuelist/rdacarrier.html
④ https://www.loc.gov/standards/valuelist/rdacontent.html

与 MARC 或 RDA 无关联的其他字段标签包括 ...

- 实例状态类型
  - 实例不需要状态，但一旦分配了一个就不能移除——只能将其更改为其他状态。参考数据包括以下预置状态（和代码）：
    - 已编目（cat）
    - 未编目（uncat）
    - 批量加载（batch）
    - 临时（temp）
    - 其他（other）
    - 尚未分配（none）
  - …但我们决定改为加载以下内容…
    - 已剔除（即 Withdrawn）
    - 原始编目（notCopyCat）
    - 需要更多信息（thereRquestions）
    - 可供流通（noProblem）
    - 丛编错误（oops）
    - 等待发票（stillWaiting）
- 馆藏地
  - FOLIO 的四层位置结构相比 ILS 能够实现更细粒度的解决方案。我们充分利用了这一点。
- 资料类型
  - 我们还利用这个机会，通过消除过时的内容并使其他与 RDA 载体类型或 ISBD 资料标识一致的方法，来整合过多的单件类型列表。
- 内容的性质
  - 本质上与字段的作用类似。我们决定使用它们来帮助识别那些可能没有馆藏记录或单件，因此没有馆藏地或资料类型的线上标题（品种）：
    - 政府文件
    - 参考资料
    - 学位论文
    - 视频
- 服务点
  - 又称流通台。这些与用户和馆藏地（需要一个）有关。

与用户相关的字段包括地址类型和读者组。这些都是最直接的设置。

本列表并非详尽完整，只是说明性的。事实上，还没有实现 FOLIO 的所有应用，所以在环境中还有许多表仍然是空的。一旦上述表加载了决定使用的标签，就可以开始从 ILS 加载数据了，但目前数据还没为 FOLIO 准备好。

3 数据的处理和迁移

关于哪些字段属于哪些表，在 FOLIO 和传统的 ILS 之间有几个重要的区别：

（1）在 FOLIO 中馆藏地与馆藏记录相关，而在旧系统中，他们是与单件相关的。

（2）卷号和作为 ILS 中索书号一部分的其他详细信息，在 FOLIO 中属于单件记录，而非在馆藏记录中和索书号在一起。

此外，从 999 字段中来自 ILS 的单件信息并不像所希望的那样完整，例如缺少工作人员的说明等。除了 MARCs w/999s 之外，还必须从 ILS 中提取多批次的数据，并对所有信息进行处理，这样方便顺利把数据导入 FOLIO。第一步是把 ILS 的输出放到这些 MySQL 表中：

mysql> describe marcRecords;

Field	Type
marc	mediumtext
instanceId	varchar(36)
shadowFlag	varchar(1)
catalogedDate	varchar(8)
marcHoldings	varchar(1)
tcn	int(11)

mysql> describe itemData;

Field	Type
analyticPasition	int(11)
callNumShadowFlag	varchar(1)
circNote	mediumtext
publicNote	mediumtext
staffNote	mediumtext
itemCat1	varchar(10)

itemCat2	varchar(10)
itemShadowFlag	varchar(1)
tcn	int(11)
barcode	varchar(16)

mysql> describe marcHoldingsRecords;

Field	Type
tcn	int(11)
m852c	varchar(10)
m863	mediumtext
m866	mediumtext
m867	mediumtext
m868	mediumtext
id	int(11)

Perl 脚本可以完成以下功能：

（1）选择一条 instanceId 字段为 NULL 的 MARC 记录

（2）将 MARC 数据解析为 FOLIO 友好的 JSON 文档

（3）将 JSON 文档 curl（POST）到 /instance-storage/instances

a. 从生成的 JSON 文档中解析 instanceId（经验：/inventory/instances 也会创建一个实例，但它不会返回一个有用的 JSON 文档来解析出十分必要的 instanceId）

b. 将 instanceId 添加到 marcRecords MySQL 表的相关行中

（4）用 999 字段和 itemData 表中的数据建立一个 itemInfo 数组

（5）如果有 MARC 馆藏记录（marcRecords.marcHolidngs=“T”），则建立一个 marcHoldings 数组

（6）对于每个独特的索书号，解析出 itemInfo 数组中的数据，包括 ..

a. 索书号，

b. 馆藏地（以往与单件相关），

c. 以及任何 MARC 馆藏说明（以往是在单独的记录中）

…一起保存在包含 instanceId 的 holdingsRecord JSON 文档中

（7）将 JSON 文档 curl（POST）到 /holdings-storage/holdings

a. 从生成的 JSON 文档中解析出 holdingsRecordId

（8）对于每个有索书号的馆藏，解析出 itemInfo 数组中的其余数据，包括…

a. 条形码，

b. 资料类型，

c. 卷号（以往是索书号的一部分）

…一起保存在包含 holdingsRecordID 的单件 JSON 文档中

（9）将 JSON 文档 curl（POST）到 /item-storage/items

（10）重复以上所有操作，直到 marcRecords 表中没有取值为 NULL 的 instanceIds Id 为止

并未使用 UI 的数据导入应用，因为很难让它发挥作用。作者已经编写了 Perl 脚本来创建实例、馆藏记录和单件，因此对如何通过 API 来实现这一目标有了相当清晰的认识。

在解析了从大学 Banner 数据库中传输过来的数据后，可以用一个类似且不那么复杂的过程将用户数据 POST 到 /users。开始这样做时，使用的是 Fameflower 版本，但平台报错，因为没有在 JSON 文档中包括 userID。但其他表在 POST 时提供了一个 UUID，这是一个意外之喜。在学习了如何制作一个 UUID 并将该功能添加到 postUsers.pl 脚本后，测试环境升级了。在 Goldenrod 中，不再需要 craft_uuid 子程序，但这是一个非常有趣的经验。

当前收费是通过在这个 MySQL 表中存储来自 ILS 的数据来迁移的。

mysql> describe chargeData;

Field	Type
circRule	varchar(10)
dateCharged	varchar(12)
dateDue	varchar(12)
itemBarcode	varchar(14)
itemType	varchar(10)
userBarcode	varchar(14)
userProfile	varchar(10)
loanId	varchar(36)
id	int(11)

然后，就可以通过 POST 到 /circulation/

check-out-by-barcode 来创建借阅。对于有滚动到期日的借阅，有必要从生成的 JSON 文件中解析出 loanId，并跟进 /circulation/loans/$loanId/change-due-date 来修改到期日。然后该 loanId 可以被添加到 MySQL 表中，并重复这个过程，直到 chargeData 表中没有一个 loanId 是 NULL 为止。

4 用 Perl 构建 web 应用和命令行解决方案

流通台首先提出的要求之一是能够扫描条形码，并能一目了然地看到单件的作者、题名、出版者信息、馆藏地、索书号、状态和以往的借阅数据，包括借阅日期、到期日期、归还日期、状态以及先前借阅者的姓名和条码号。这些信息在 UI 中是有的，但并未全部显示在一个屏幕上。解决方案是构建一个简单的 HTML 表单，该表单将条码和密码传递给一个 Perl 脚本，然后：

（1）GET 所有的单件数据，包括 itemId 和 holdingsRecordId

/item-storage/items?query=barcode="$barcode"

（2）GET 包含 instanceId 的所有馆藏数据

/holdings-storage/holdings?query=id="$holdingsRecordId"

（3）GET 所有实例数据

/instance-storage/instances?query=id="$instanceId"

（4）GET 包含任何 userId 的所有借阅数据

/circulation/loans?query=itemId="$itemId"

（5）GET 所有用户数据

/users?query=id="$userId"

查找单件的目的不同，借阅数据可能就不相关，因此它被放入一个 JavaScript 变量中，以便在必要时可以按需显示。

经验：请注意，上面的步骤 2 和步骤 3 中，API 调用包含一个查询，同样的信息可以像这样检索：

/holdings-storage/holdings/$holdingsRecordId

/instance-storage/instances/ $instanceId

这些 API 调用返回的 JSON 不包括开头相关表的名称（“holdingsRecords”和“instances”），也不包括末尾返回的记录总数（在这种情况下‘"total records" : 1’）。这些细节通常很有用。这也使得编写解析 JSON 代码的工作更加一致，不容易引起混乱。

考虑到没有使用 FOLIO 的数据导入应用，因此需要一个 web 应用来向新的 LSP 加载题名。作者和编目员一起设计了一个称之为“仪表盘”（The Dashboard）的东西，它可以单个或批量加载 MARC 记录，并提供后续链接，带到 UI 或 OPAC 中，以便核查。这背后的脚本与上述的用于从 ILS 加载 MARC 的脚本基本相同。主要的区别是，这个过程并不试图创建馆藏记录或单件，因为只需按照生成的链接进入 UI 即可创建。它还允许您点击 OPAC 中的链接或输入实例的 HRID 来处理系统中已有的题名。之后，您可以下载 MARC 记录，并在 MarcEdit 中对其进行处理，接着将其覆盖到 MySQL 中并在 FOLIO 中更新实例。

经验：没必要将 MARC 记录存放在 FOLIO 中。现在已经有了一个 MySQL 数据库，来保存支持 OPAC 的 MARC 记录，因此就用它来保存这些记录。另一个重要的 web 应用是“教学参考书”。在页面上整合信息的过程如下：

（1）GET 当前学期的 termId

/coursereserves/terms?query= name=Fall 2021

（2）GET courseListingIds 和 Prof. Names

/coursereserves/courselistings?query=termId=$termId

（3）GET 每个 courseListingId 的课程号和名称

/coursereserves/courses?query=courseListingId=$courseListin gId

（4）将上述内容 push 进一个数组

push(@courses,"$courseNumber|$profName|$courseName|\n")

（5）GET 每一个 courseListingId 教学参考书的题名

/coursereserves/reserves?query=courseListingId=$courseListi ngId

（6）将每个题名 push 进一个数组

push(@reserves,"$sortTitle|$title|$callNumber|$courseNumber |$location|\n")

（7）对每个数组进行排序并解析，以显示教学参考书的每个标题，像这样：

● LIBR 101 图书馆学导论
教授：Dewey Decimal
○ 图书馆学 : 战略忽视的艺术
总馆教学参考书 025 D519l

并非所有的解决方案都能通过 web 应用来获得最好的服务。有些服务只会产生一次和 / 或不产生交互，通过一个简单的 Perl 脚本，无须 web 页面就可以完成。比如将单件费用作为备注添加到单件记录中。将来自 ILS 的所有数据都加载到 FOLIO 后，会发现单件记录中并未包括单件的费用（数据在 999s 中）。单件记录中也并没有包括这个数据点的字段。我们创建了一个名为 Price 的新单件注释类型。然后写了一个脚本在 999s 中进行筛选，从 /item-storage/items 中获取每个单件的 JSON 文档，使用正则表达式将文档拆开重组，以便将一个新的单件注释添加到现有数组中，或是生成一个新的单件注释数组，然后将新 JSON 文档 PUT 到 /item-storage/items/ 覆盖旧的。

另一个例子是延长当前用户的到期日期。这是每学期都要为学生做的事情，每年为教职员工做一次。这个过程使用 /groups?query=group="*" 来获取所有用户组及其 groupIds，然后使用 /users?query=patronGroup=="$groupId" 来获取所有需要的用户。之后对于每个用户，Banner 系统会进行检查，如果该用户是当前用户，则用正则表达式修改到期日期。

$data=~s/expirationDate":"\d\d\d\d-\d\d-\d\d/expirationDate":"2022- 02-14/

然后 `curl -s -X PUT -H 'Content -type: application/json' -H '$okapiToken'-d '$data' $url/users/$userId` 完成。

5 结论和展望

我们基于 Fameflower 版本开始与 FOLIO 的合作，与 EBSCO 员工的第一次对话就是关于文档的。更确切地说，是关于在哪里可以找到有关使用 UI 的文档。回答是，目前还没有任何文档。通过使用 API 深入研究数据库表并拆分 JSON 输出，学到了配置系统以及加载书目和读者数据所需的经验。现在，虽然我们急切地等待升级到 Kiwi ，但这依旧是一个非常不同的环境。这个 LSP 有更多的应用程序，项目维基 [①] 也包含更多的信息。我们计划很快就会开始使用采访和 ERM（电子资源管理）应用程序。无论是否需要使用 API 来设置这些应用程序，肯定会像第一次开始使用 Fameflower 时那样，从相关数据表开始。随着扩大对 FOLIO 的使用以及 FOLIO 的持续发展，我们所获得的经验将十分有益。

① https://wiki.folio.org/

盖伊·多布森 新泽西州麦迪逊市德鲁大学的技术服务主管和系统馆员。高中和大学时，他在当地公共图书馆工作，之后在哥伦比亚大学图书馆服务学院获得了理学硕士学位。除了公共图书馆外，他还在学术、医疗、音乐和学校图书馆工作过。他曾在全国会议上发表过多次演讲，介绍如何构建系统、为读者和工作人员增加或改善图书馆网站及目录的功能。E-mail: gdobson@drew.edu

张喆昱 女，上海图书馆（上海科学技术情报研究所），馆员。 上海 200031

张春景 女，上海图书馆（上海科学技术情报研究所），研究员。 上海 200031

胡小菁 女，华东师范大学图书馆，研究馆员。 上海 200062