Building Reusable APIs

Stephanie Rabbani – Seiden Group (www.seidengroup.com)
What we’ll cover

- The what, why and how of APIs!
- Best practices
- Some code references
- Some code examples
What is an API?

- What do you think of when you think of the term API?
What is an API?

- Application program **interface**
  - Business capabilities that are exposed for others to use
- Internal via program calls (or procedure calls)
- External via HTTP APIs

- My history with APIs
  - Large project a few years ago where the client wanted to use the same business logic in RPG and the web.
  - More recently I’ve been working on HTTP APIs integrating with 3rd party web services
  - We are just starting one to integrate with salesforce
Why APIs?

- Encapsulate functionality
- Allows for reusability
- Allow other groups, customers to integrate with your system
- Bi-product is modularizing and streamlining your code and processes

- Taking pieces of your monolithic application and breaking it down to a set of APIs will pay dividends down the road
- A good way to start your refactoring process
Where do I start?

- Understand who will be using the APIs & how they will be using it
- Remember, the I Stands for interface
- Start laying out what information the “user” will require, and what information your API will need in order to furnish that information (parameters and return value)
Example (a restaurant)

- A waiter will require one set of information
  - List of specials, place an order
- A cook will require a different set
  - List of ingredients in a dish, list of pending orders, update order as complete
- Both may want to list the menu
- And then we start designing the program interface to allow this data to be accessed
Consuming APIs in RPG

- There are many different APIs that you take advantage of!
- IBM APIs (API finder):
  ➤ See Scott Klement’s session Tuesday at 2:00pm
- SQL services are APIs:
  https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_73/rzajq/rzajaservicessys.htm
- Consuming web APIs in RPG
  ➤ You can use SQL services to consume XML APIS (SYSTOOLS.HTTPGETBLOB):
    https://www.ibm.com/developerworks/ibmi/library/i-incorporating-web-service/ (Birgitta Hauser’s session today at 3:30)
  ➤ Scott Klement’s HTTPAPI: http://www.scottklement.com/httpapi/
  ➤ Others?
Providing APIs in RPG

  - Scott Klement’s session today at 11:00am
- Other vendor tools that create RPG CGI programs (BCD’s WebSmart, etc)
- Others?
REST APIs

- REST APIs are HTTP URLs that use HTTP “the way it should be used”
- In the past, we have typically only used the GET and POST protocols
- REST uses:
  - POST to create
  - DELETE to delete
  - PUT to update
  - GET to list or retrieve
- REST APIs are stateless – each request needs all of the information, does not depend on previous requests
- Communicate via XML (SOAP) or JSON
- HTTP APIs are language independent – RPG can consume a PHP API just as well as a Node API
RPC vs. REST – Restaurant example

- Old way (RPC)
  - GET /menu-items-list
  - POST /menu-item?task=new
  - GET /menu-item?id=444
  - POST /menu-item?task=delete&id=444
  - GET /ingredients-list?item=123

- New way (REST):
  - GET /menu-items/
  - POST /orders/
  - GET /orders/444/
  - DELETE /orders/444/
  - GET /menu-items/123/ingredients/
Designing your end points

- Separate your API into logical resources. These should be **nouns** that make sense to your user
- Do not show file extension (.php, etc)
- Should be all lower case
- Spaces should be replaced with hyphens or underscores (choose one, and stick to it)
REST API Do’s

1. Always use SSL and use authentication when appropriate.
2. You can make use of HTTP caching.
3. Look for SDKs and code examples.
4. Use pretty printing for your JSON.
REST API don’ts

1. Have terse/cryptic error messages
2. Use HTTP rules incorrectly
3. Use verbs in your end points
Good API Documentation tools

- Swagger (PHP): https://packagist.org/packages/zircote/swagger-php
- APIGEN (PHP): https://github.com/ApiGen/ApiGen
- I/O docs (written in Node): https://github.com/mashery/iodocs
- APIDOC (written in Node): http://apidocjs.com/
- Sphinx (for Python APIs): http://www.sphinx-doc.org/en/1.5.1/tutorial.html
Documentation Do’s

1. Keep a change log
2. Keep a road map
3. Release notes
4. Have a forum
5. Create a quickstart guide
6. Provide multiple SDKs
7. Status page?
Documentation Don’ts

- Is your documentation...
  - Out of date?
  - Incomplete?
  - Inaccurate?
  - Unloved?
  - Hard to read?
  - Hard to find things?
Versioning

- It's a good idea to version your API
- It helps you iterate faster and prevents invalid requests from hitting updated endpoints
- Can use as the first part of your API url:
  - /api/v1/orders/
- Some say to use it as part of your header:
  - Accept: application/vnd.company.myapp.customer-v3+json
API Security

1. OAUTH2 (more complex, more secure)
   - Request is made using user name and password
   - Access token is returned
   - Following requests send the access token to authenticate

1. HTTP Basic (simpler)
   - Basic authentication sends a Base64-encoded string that contains a user name and password for the client
   - Set up at the Apache level for authorization
Testing your API

- Postman (demo)
What is JSON?

- JavaScript Object Notation
- Name : value pairs, separated by commas
- Nest arrays using square brackets, objects using curly brackets

```json
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021"
    },
    "phoneNumber": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```
Working with JSON

- JSONLint (demo)
Understanding/Using HTTP Response codes

- 200 – OK
- 400 – Bad request
  - With accompanying error message
- 401 or 403 – Not authorized
- 404 – Not found
- 405 – Method not allowed (tried to POST to a GET only end point)
- 429 – Too many requests
- 500 – Internal server error
- 503 – Service unavailable
Examples of good APIs

- Twilio (add messaging, voice & video to your app): https://www.twilio.com/docs
- Stripe (credit card services): https://stripe.com/docs/api
- Constant contact (email marketing): https://developer.constantcontact.com/docs/developer-guides/api-documentation-index.html
- Send Grid (email marketing): https://sendgrid.com/docs/API_Reference/index.html
Consuming APIs in PHP – 1/3

- File_get_contents()

```php
//no header authorization data needed
$response = file_get_contents('https://www.metaweather.com/api/location/2428344/');
$response = json_decode($response);
```
Consuming APIs in PHP – 2/3

- Curl

```php
$curl = curl_init();
curl_setopt($curl, CURLOPT_POST, 1);
//optional authentication
curl_setopt($curl, CURLOPT_HTTPHEADER, array('Authorization: Bearer sk_test_WaxdrZSYwJgTsUucuXvJlFxF2'));
curl_setopt($curl, CURLOPT_URL, 'https://api.stripe.com/v1/customers');
curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
$result = curl_exec($curl);
curl_close($curl);
```
Consuming APIs in PHP – 3/3

- Guzzle Library (http://guzzle.readthedocs.io/en/latest/)
- Install via composer: `php composer.phar require guzzlehttp/guzzle:~6.0`
- Then include composer's autoloader: `require 'vendor/autoload.php';`

```php
$client = new GuzzleHttp\Client();
$headers = ['Authorization' => 'Bearer sk_test_WaxdrZSYwJgTsUucuXvJlFx2'];
$res = $client->get('https://api.stripe.com/v1/customers', $headers);
echo $res->getStatusCode(); // 200
echo $res->getHeader('content-type'); // 'application/json; charset=utf8'
echo $res->getBody();
```
Providing APIs in PHP – 1/3

**Basic:**

```php
$method = $_SERVER['REQUEST_METHOD'];
$path = $_SERVER['REQUEST_URI'];
$paths = explode($path);
$resource = array_shift($path);
if ($resource == 'orders') {
    switch($method) {
        case 'PUT':
            $this->create_order($name);
            break;
        case 'DELETE':
            $this->delete_order($name);
            break;
        case 'GET':
            $this->display_order($name);
            break;
        default:
            header('HTTP/1.1 405 Method Not Allowed');
            header('Allow: GET, PUT, DELETE');
            break;
    }
} else {
    // We only handle resources under 'orders'
    header('HTTP/1.1 404 Not Found');
}
```
Providing APIs in PHP – 2/3

- Laravel Framework: [https://laravel.com/docs/5.1/controllers#restful-resource-controllers](https://laravel.com/docs/5.1/controllers#restful-resource-controllers)

  ```php
  php artisan make:controller OrdersController
  Route::resource('orders', 'OrdersController');
  ```

- ZF2/ZF3 framework:
  - Module: [https://github.com/zfcampus/zf-rest](https://github.com/zfcampus/zf-rest)
  - Example: [https://github.com/stevenalexander/zf2-restful-api](https://github.com/stevenalexander/zf2-restful-api)

- Zend expressive example:
  - Example: [https://github.com/ezimuel/zend-expressive-api](https://github.com/ezimuel/zend-expressive-api)
Providing APIs in PHP – 3/3

- Apigility - “Apigility is an API Builder, designed to simplify creating and maintaining useful, easy to consume, and well structured APIs. Regardless of your experience in API building, with Apigility you can build APIs that enable mobile apps, developer communities, and any other consumer controlled access to your applications.”

- It generates PHP code that can run on your IBM i

http://sobo.red/Installing-and-Using-Apigility-on-IBM-i/
http://sobo.red/Apigility-with-OAuth2-on-IBM-i/
http://sobo.red/Apigility-Entity-Mapper-Tips/
Parsing the response in PHP

```php
$array = json_decode($response);
```
import requests

headers = {'Authorization': 'Bearer sk_test_WaxdrZSYwJgTsUucuXvJ1Fx2'}

resp = requests.get('https://api.stripe.com/v1/customers', headers=headers)
Parsing the response in Python

- Json library (https://docs.python.org/3.4/library/json.html)

```python
import json
if(resp.ok):
    jData = json.loads(resp.content)
```
Providing APIs in Python (1/2)

- Flask:
  - $ pip install flask
  - $ pip install flask-restful

```python
from flask import Flask, request
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

class Orders(Resource):
    def post(self, order_id):
        # api creation code here
    def delete(self, order_id):
        # api deletion code here

api.add_resource(Orders, '/orders/<string:order_id>')

if __name__ == '__main__':
    app.run()
```
Providing APIs in Python (2/2)

- **Bottle:**
  - `$ pip install bottle`

```python
from bottle import route, run, static_file, request

@route('/orders/<order-id>', method='POST')
def create_order(order-id=""):
    if "" != order-id:
        return create_function_call(order-id)
    else:
        return { "success" : False, "error" : "orders called without an id" }

@route('/orders/<order-id>', method='DELETE')
delete_order(order-id=""):
    if "" != order-id:
        return delete_function_call(order-id)
    else:
        return { "success" : False, "error" : "orders called without an id" }
```

run(host='localhost', port=8080, debug=True)
How will you use APIs?

1. Sharing data between
   - 2+ languages
   - 2+ servers

2. Accessing remote data/services

3. Updating portions of your page

4. As part of a refactoring effort
Thank you!

Please provide feedback in http://sched.co/9CEK

Contact Info:
Stephanie Rabbani
steph@seidengroup.com
250-721-0817
@jordiwes