Rapid Data Publishing – Infrastructure

Roy Duelfer, *Montana DEQ*

2015 Exchange Network National Meeting

Supporting the Business of Environmental Protection

September 29–October 1, 2015
Sheraton Philadelphia Society Hill Hotel
Philadelphia, Pennsylvania

http://www.exchangenetwork.net/en2015

ABSTRACT

How Montanta DEQ’s current infrastructure supports a rapid development environment.

What problem does the API attempt to solve?

The API provides 24/7 access to the Clean Water Information Center, Hazardous Waste Handlers, Petroleum Release Fund Claims and Reimbursements, and Unpermitted Releases Datasets (stage 1). This means that anyone with internet access can inspect the data at any time as long as the API is up and running.

How does the API provide access to data?

- Representational state transfer (REST) creates stateless web services, thus exposing application data over the Internet. [**1**]
- The API exposes the data in the various applications through the HTTP protocol, making it a RESTful API.

[1] (2014, Aug.) REST API Tutorial. [Online]. http://www.restapitutorial.com/lessons/httpmethods.html

Why is the REST architecture used?

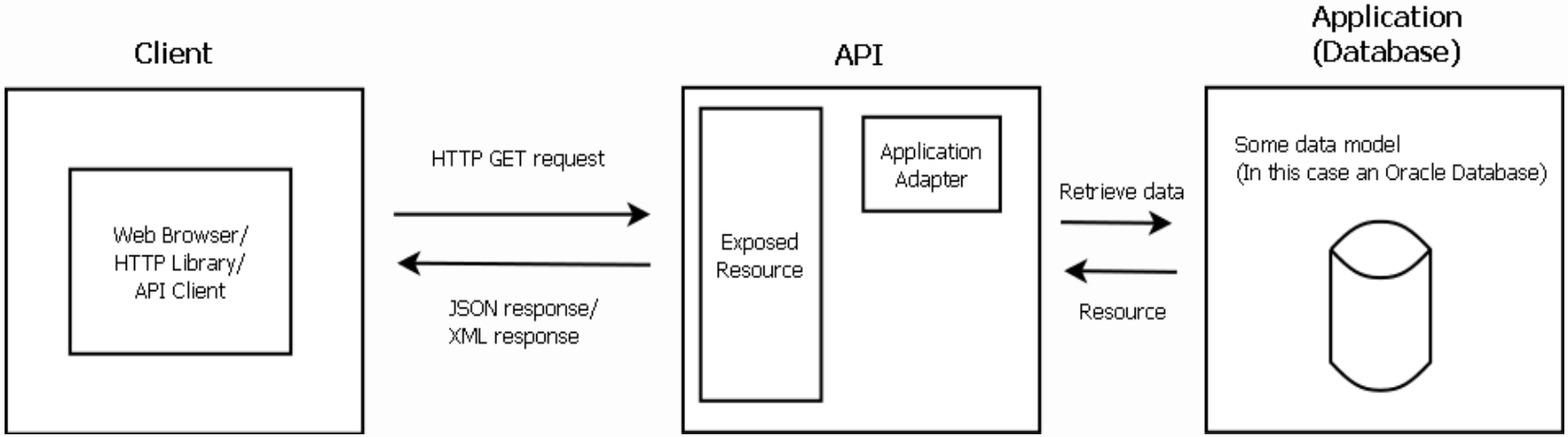
- REST is a simple API architecture that is universally accessible via HTTP methods. Moreover, because the application data is not subject to modification, REST is a lightweight implementation that appeared to be the most appropriate choice. [2]

- **Benefits of REST include:**
- Uniform interface for accessing resources [3]
- Self-describing resources when designed properly [**3**]
- Easy readability (simple noun verb relationship between resource and action) [**3**]

[2] Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California, Irvine, Doctoral dissertation 2000.
[3] Geert Jansen. (2011) RESTful API Design. [Online]. http://restful-api-design.readthedocs.org/en/latest/intro.html

REST Overview

- The three main components for a RESTful API are the client, API, and application
- The client consumes the RESTful API data through the HTTP protocol [**3**]
- The API exposes application resources over the web using HTTP methods [**3**]
- The application is some dataset, usually a database which houses the data to be exposed [**3**]



[3] Geert Jansen. (2011) RESTful API Design. [Online]. http://restful-api-design.readthedocs.org/en/latest/intro.html

Project vision

- In order to provide for a well-informed public and increase the department's transparency, the data search tools API will expose program data sets in a concise and easy-to-use format.

- The API is designed to do three things (more like one thing with three parts),

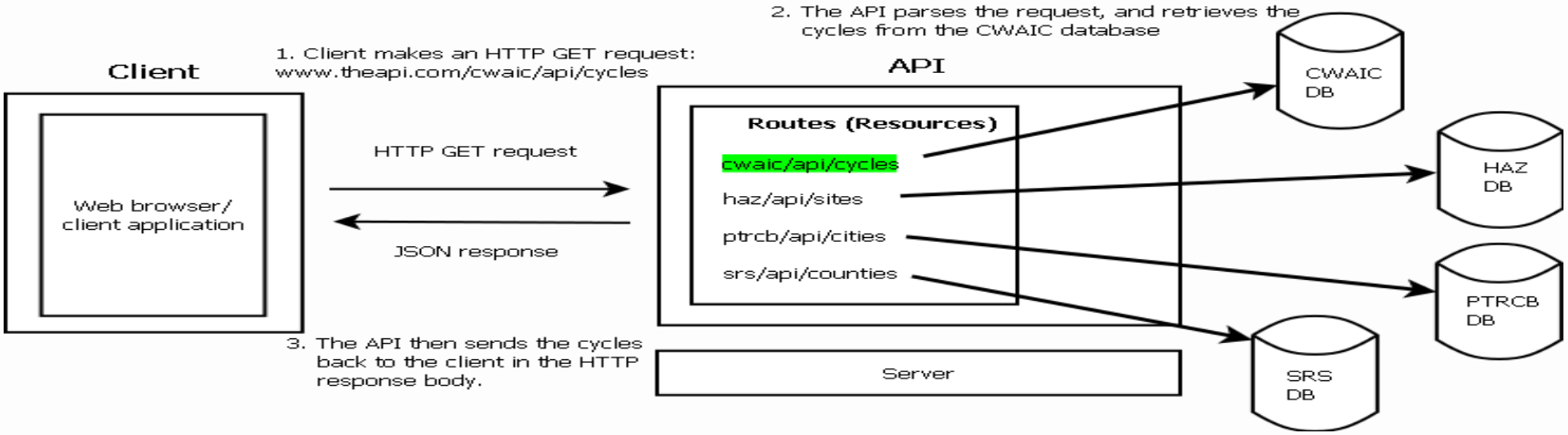
1. Respond to properly formatted user requests for data with the correct dataset as fast as possible.
2. Respond to properly formatted requests for documentation with a helpful, concise response.
3. Respond to an improper request with a helpful error message indicating to the user what resource could not be found.

Application Resources

- The API exposes resources as URL endpoints (also known as Universal Resource Identifiers or URIs).
- These endpoints are known as routes within the API code; however, the end user simply sees them as a URL representation of a resource.
- The API listens for HTTP requests, if the request matches a route it queries the database for the requested resource and returns it.

High-level Interaction

1. To retrieve a resource a client first makes an HTTP request to the API. This would be done by making an HTTP GET request for a specific resource like, www.example.com/cwaic/api/cycles.
2. Next, the API receives this request, identifies the requested resource and queries the database.
3. Finally, the API returns the results from the database to the client issuing a 200 HTTP success code. [**1**]

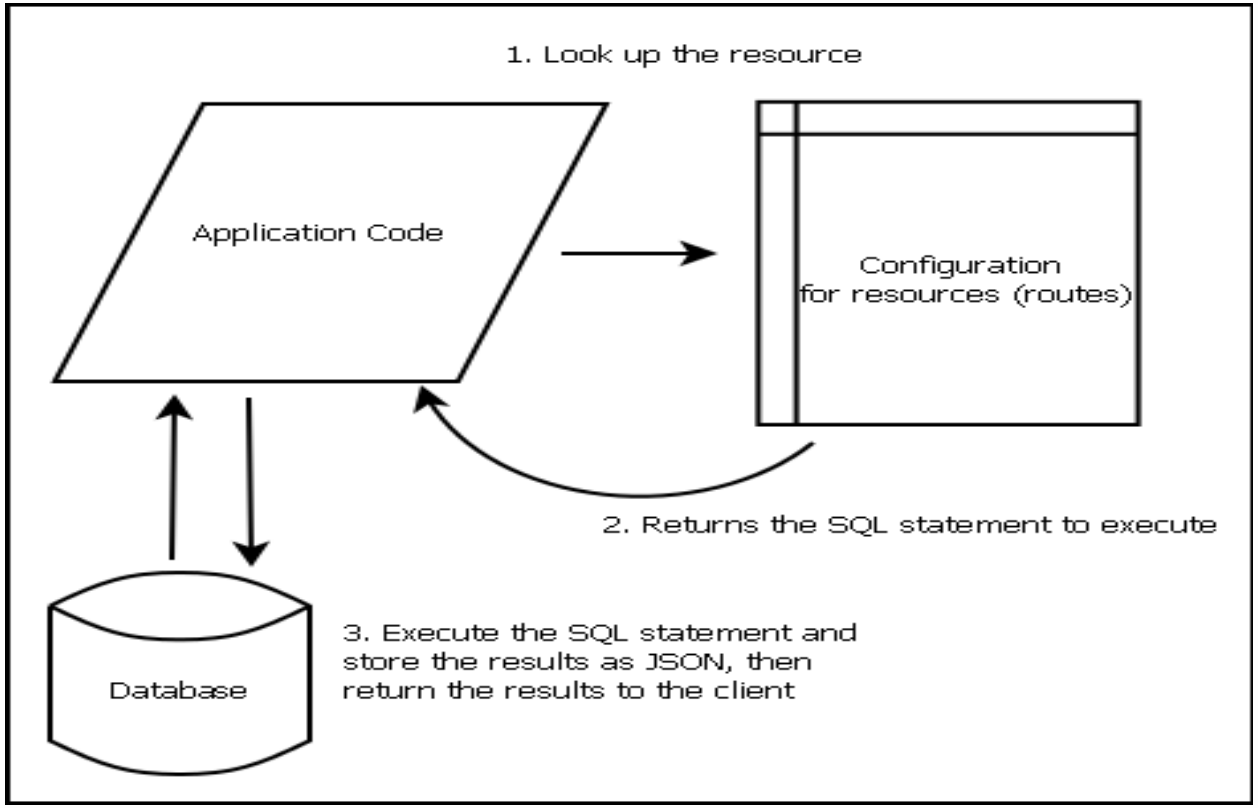


[1] (2014, Aug.) REST API Tutorial. [Online]. http://www.restapitutorial.com/lessons/httpmethods.html

API Route Lookup

1. Using dictionary objects, the API code performs a lookup populated from a configuration database.
2. A request from a client comes in, the request: */cwaic/api/cycles*
3. The API parses the request and sets the **app variable = cwaic**, and the **resource variable to cycles**.
4. The API then looks up the app in the configuration to see if it exists, if it does not it returns saying the application could not be found.
5. However, if the application does exist, the API then searches the configuration for the resource, when it finds a match it searches within the sub object for the specific route (the whole resource requested).
6. In this example: */cwaic/api/cycles*, when the API matches the route, it extracts the SQL query from the configuration and carries on with running the query and returning the result.

- The benefit to this "levelled-lookup" scheme, is the search running in O(log n) or O(1) time, which is extremely efficient for a data structure lookup. [**8**]



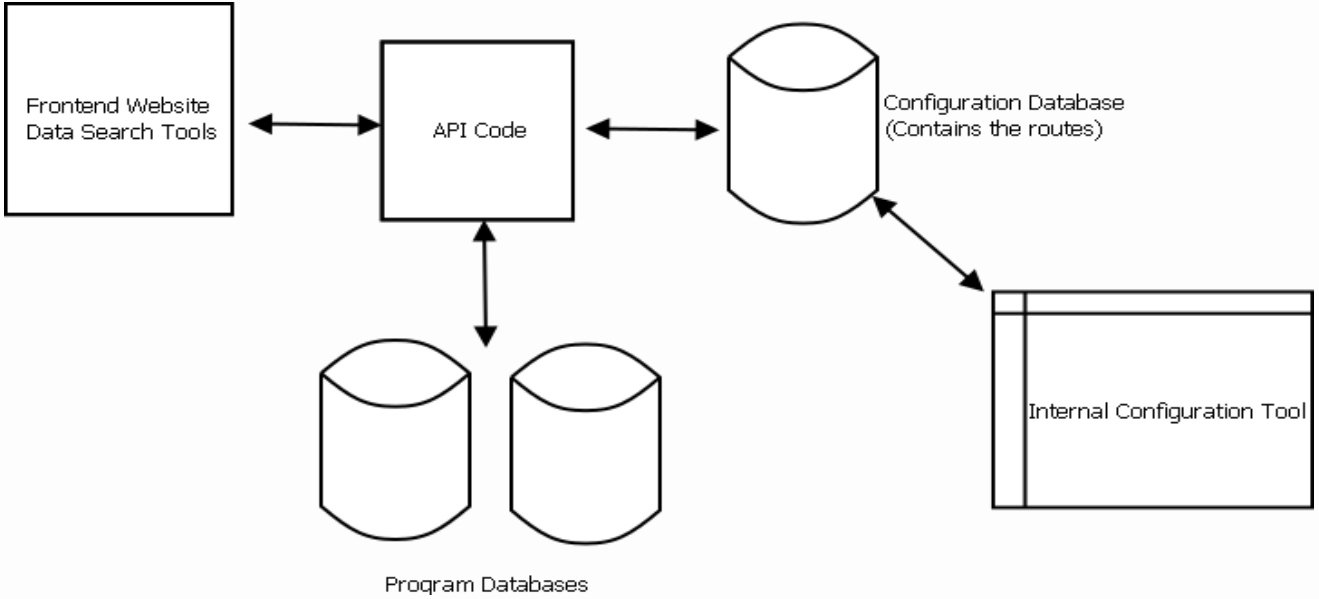
[8] JavaScript Guides Advanced. [Online]. http://javascript-reference.info/javascript-implementation-of-hashtable.htm

Overall Structure

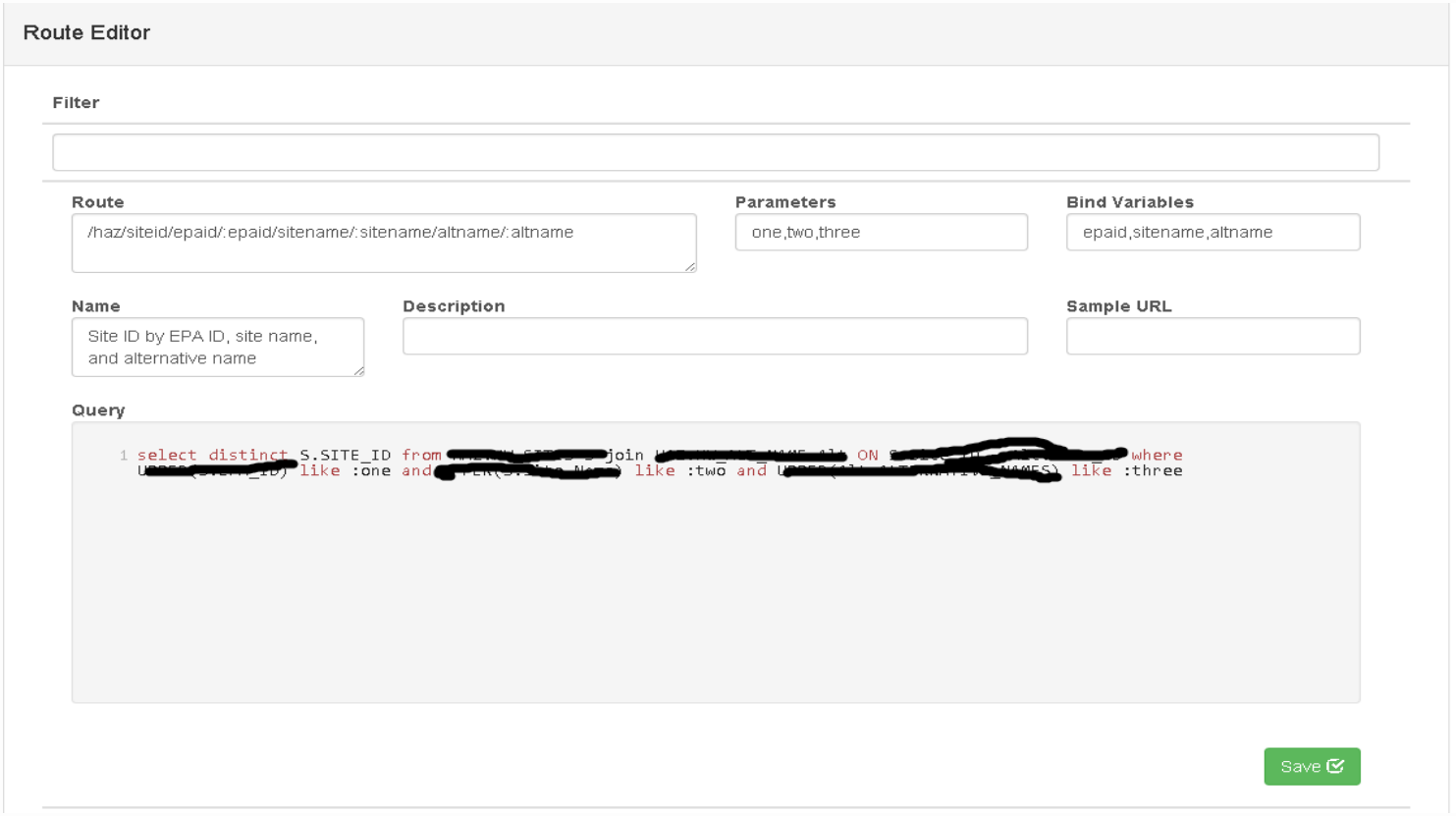
This structure allows us to add new programs as soon as the SQL is written, allowing for very fast development turnaround times.

Using the internal configuration tool, routes are created in the configuration database and are immediately present to the API and applications.

Moreover, the internal tool also contains the metadata for the route documentation.



Internal Configuration Tool



API Documentation

Documentation

- In addition to keeping the configuration separate from code logic, another major benefit to this structure is easy documentation.

- By removing sensitive fields such as the query, it is possible to send a JSON route configuration response to the client as documentation for the API itself.

Documentation Example

- If a client was to request the CWAIC documentation, he could simply make the request: www.example.com/cwaic/docs

- The response is a filtered version of the configuration for the specific application. This allows the end user to see all of the routes available to him with an easy to read title that describes the route and resource.

Response:


```
{
  "cycles": {
    "/cycles": {
      "route": "/cycles",
      "title": "Cycles"
    }
  },
  "auids": {
    "/auids/cycle/:cycle": {
      "route": "/auids/cycle/:cycle",
      "title": "AUID's by cycle"
    },
    "/auids": {
      "route": "/auids",
      "title": "AUID's"
    }
  }
}
```

Works Cited

[1] (2014, Aug.) REST API Tutorial. [Online]. http://www.restapitutorial.com/lessons/httpmethods.html

[2] Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California, Irvine, Doctoral dissertation 2000.

[3] Geert Jansen. (2011) RESTful API Design. [Online]. http://restful-api-design.readthedocs.org/en/latest/intro.html

[4] Joyent. (2014) Node.js. [Online]. http://nodejs.org/

[5] Tomasz Janczuk. (2011, August) iisnode wiki. [Online]. https://github.com/tjanczuk/iisnode/wiki/iisnode-wiki

[6] Merrick Christensen. (2012, October) _iammerrick. [Online]. http://merrickchristensen.com/articles/javascript-dependency-injection.html

[7] Jakob Jenkov. tutorials.jenkov. [Online]. http://tutorials.jenkov.com/dependency-injection/dependency-injection-benefits.html

[8] JavaScript Guides Advanced. [Online]. http://javascript-reference.info/javascript-implementation-of-hashtable.htm

[9] TJ Holowaychuk. (2014) visionmedia. [Online]. http://visionmedia.github.io/mocha/

[10] Bocoup. Grunt.js. [Online]. http://gruntjs.com/

[11] M Shiek Uduman Ali. (2011, September) codeproject. [Online]. http://www.codeproject.com/Articles/255568/WCAT-Simple-Performance-Test-Tool-for-your-NET-web

[12] express - request. express. [Online]. http://expressjs.com/4x/api.html#req.params

[13] express - response. express. [Online]. http://expressjs.com/4x/api.html#res.status

[14] Don Nguyen, Jump Start Node.js, 1st ed.: SitePoint, 2012.