Web Services

Alexander Nelson

November 8th, 2017

University of Arkansas - Department of Computer Science and Computer Engineering

Web Services

What is a web service?

Set of functions that can be accessed over HTTP protocols

Provide response through a serialized data protocol (e.g. XML, JSON)

Why use Web Services?

- Allow transmission of data from or to a remote server
- Perform proprietary data transformations
- Transmit data between programs

Web Service Design

What is the information flow of a Webservice?

REST Webservices

REST – Representational State Transfer

Developed in tandem with HTTP 1.1 and formalized in 2000

Distill data transfer to a core set of principles

REST Architecture

Core Architectural Properties:

- Performance – Maximize network efficiency
- Scalability – Support large number of components and interactions
- Simplicity and Uniformity in interfaces
- Visibility of communication between components
- Portability of program code and data
- Reliability of components to failures

REST Architecture

Six guiding constraints:

- Client-Server Architecture/Model
- Statelessness – No client context is stored on the server between requests
- Cacheability – Clients can cache responses
- Layered System – Client doesn't know if it is connected directly to the end server
- Code on demand – Temporarily extend functionality of client by transferring executable code
- Uniform Interface

REST Methods

REST Web services use the available HTTP methods
Most often, REST services use four common methods:

- GET
- PUT
- POST
- DELETE

Behavior depends on whether the URL specifies an element or a collection

REST URLs

REST Services use URLs to specify data and method locations
Example:
http://example.com/api/path/?parameter=1&otherparameter=2

The URL will specify:

- HTTP (or HTTPS) as the transfer protocol
- The server domain location
- The path on the server to the particular function
- Optional set of parameters following a ? and separated by &s

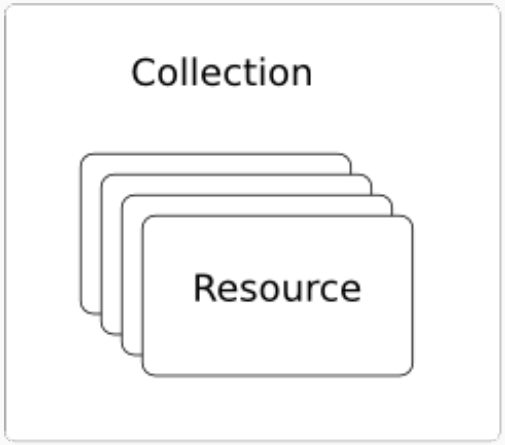
Collection vs. Element

Element (or Resource) – Object with a type

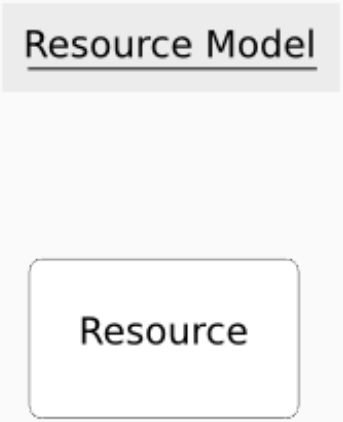
May have:

- Associated Data
- Relationships to other resources
- Set of methods that operate on it

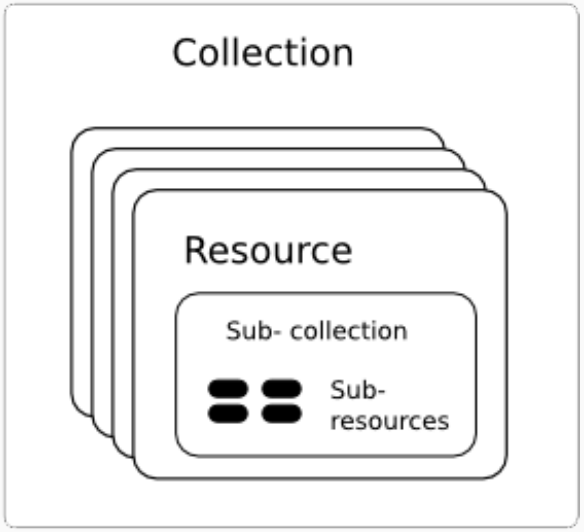
Collection – Set of elements



A Collection with Resources



A Singleton Resource



Sub-collections and Sub-resources

GET Method

If the URL points to a single element

Example:

GET http://del.icio.us/api/ahnelson/bookmarks/a211528

Returns information about a specific bookmark

If the URL points to a collection:

Example:

GET http://del.icio.us/api/ahnelson/bookmarks

Returns information about all the bookmarks associated with the user ahnelson

POST Method

POST – Creates a new object

URL specifies the collection to which the object should be added

Specify fields using optional parameters

Often returns the URL of the created object

PUT Method

PUT – Can be used to create or update existing record*(s)
URL specifies the resource to be modified

Can prevent creation using PUT by returning a 404 error if resource doesn't exist
Specify fields to modify using optional parameters
Typically returns the URL of the modified object

DELETE Method

DELETE – Remove record(s)

URL specifies the element or the collection to be deleted

Often returns an HTTP response of 204 (No Content) if successful

Responses

The Responses include an HTTP Response code and optional information in serialized language
Response Codes:

- 200 – OK
- 201 – Created
- 202 – Accepted
- 401 – Unauthorized
- 403 – Forbidden
- 404 – Not Found

And others

Example

Let's look at an example using test data from the web:
https://jsonplaceholder.typicode.com/

Securing a REST Web Service

Authentication – Verify that a user of the service has permission
Authorization – Verify that the connection is allowed

Perform these actions in order to verify that the user is allowed to use the service and has a valid connection

Authentication – Basic

Simplest form of authentication
Send encoded Username & Password Secret as additional HTTP header

Encoded $=/=$ Encrypted! Use SSL (HTTPS) when doing basic authentication
Password Secret should not be the same as the actual password!
Use hashing of passwords at server to prevent holding onto passwords

Authentication – Hashed

The server and the client should be the only ones that know the actual password secret

Combine parts of the request with the secret, and perform a hash function

The server will be able to perform the same hash and determine if there is a match

Authentication – OAuth

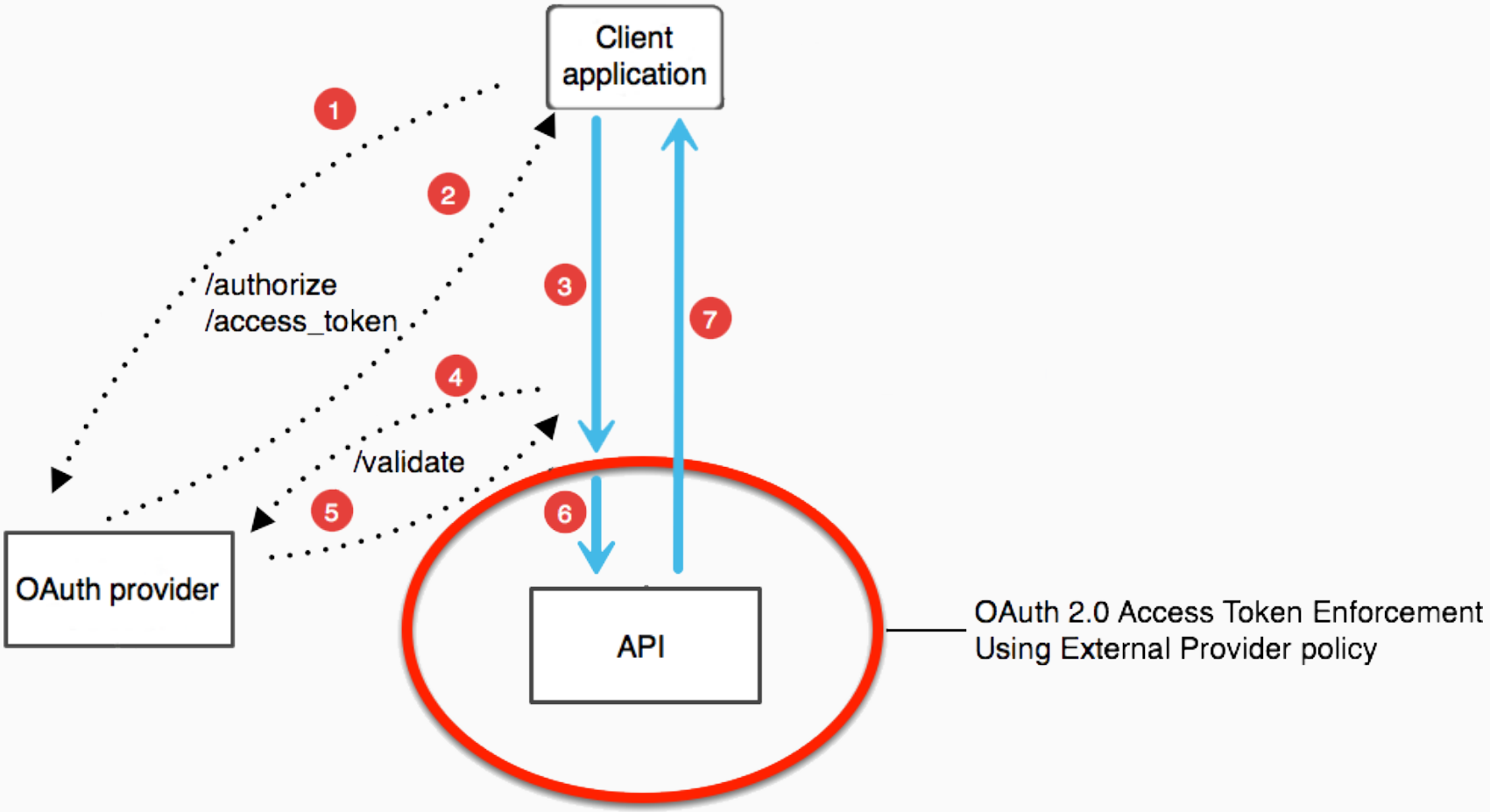
OAuth – Token based security

An authentication provider determines if the user is authorized and provides a token

The token is used in place of the password secret

Tokens expire after a certain amount of time, require user to authenticate again

Authentication – OAuth



OAuth Authentication Pattern