Web Services
Endpoints

Software Maintenance

REST - Representational State Transfer

REST Architecture

- Well-designed web application consisting of a network of web pages (a virtual state machine)

Roy Fielding Ph.D. dissertation, 2000

REST Architectural

REST style has formal and informal “constraints:

- Client-server
- Uniform interface
- Layered system
- Cacheable
- Stateless
- Code on demand (optional)

https://medium.com/@sagar.mane006/understanding-rest-representational-state-transfer-85256b9424aa

Uniform Interface

Fielding identified four constraints for the uniform interface:

1. Identification of resources – each resource has a unique URI
2. Manipulation of resources through representations – JSON, HTML
3. Self-descriptive messages –send desired state in request, get current state in response
4. Hypermedia as the engine of application state (HATEOAS) – resources include links to related resources

http://shop.oreilly.com/product/0636920021575.do

URIs

Format of Uniform Resource Identifiers (URIs) :

scheme “://” authority “/” path [“? Query] [“#” fragment]

Example:

http://api.example.restapi.org/france/paris/louvre
/leonardo-da-vince/mona-lisa

URIs Rules

Rules for URIs:

- Forward slash (/) separator is used in path to indicate a hierarchical relationship
- Don't use a trailing forward slash
- Use hyphens (-) to include readability in paths (authority portion is not case sensitive, paths are case sensitive)
- Don't use underscores (_)
- Prefer lowercase letters in paths
- Don't use file extensions in URIs

Resource Modeling

Each forward slash separated path segment corresponds to a unique resource within the model's hierarchy

Example:

http://api.soccer.restapi.org/leagues/seattle/teams/trebuchet

Indicates that each of these URIs also identify an addressable resource:

http://api.soccer.restapi.org/leagues/seattle/teams

http://api.soccer.restapi.org/leagues/seattle

http://api.soccer.restapi.org/leagues

http://api.soccer.restapi.org

Resource Archetypes

Align each resource with only one of the following archetypes:

1. Document

   http://api.soccer.restapi.org/leagues/seattle

2. Collection

   http://api.soccer.restapi.org/leagues

3. Store - for a client-managed repository, clients can put resources in, get them back out and decide when to delete them

   PUT /users/1234/favorites/alonso

4. Controller – executable functions

   POST /alerts/246554/resend

Rules for Archetypes

Rules for archetypes:

- Use singular nouns for document names
- Use plural nouns for collection names
- Use a plural noun for store names

  http://api.music.restapi.org/artists/mikemassedocom/playlists

- Use a verb or verb phrase for controller names

  http://api.college.restapi.org/students/morgan/register

  http://api.build.restapi.org/qa/nightly/run-test-suite

Rules for Archetypes

More rules for archetypes:

- CRUD function names should not be used in URIs, instead use

  DELETE /users/1234

REST Architectural Steps

Steps

- Define the domain and data.
- Organize the data into groups.
- Create URI to resource mapping.
- Define the representations to the client (XML, HTML, CSS, …).
- Link data across resources (connectedness or hypermedia).
- Create use cases to map events/usage.
- Plan for things going wrong.