50.012 Networks

Lab2: REST API

2021 Term 6

Lab 2

- Hand-out: 24 Sep
- Hand-in: 5 Oct 23:59

https://github.com/chesnutcase/networks_lab2

Special thanks to Chester for this brand new 2021 version of lab 2 based on the FastAPI framework

A brief introduction of REST

Representational state transfer

World Wide Web

- The World Wide Web (WWW) is a global hyper-linked information system, where ***resources*** are identified by Uniform Resource Identifiers (***URI***s) and accessible over the Internet
  - Tim Berners-Lee invented the Web in 1989 when he worked at CERN near Geneva, Switzerland. He developed the foundational ideas behind ***URI***, ***HTML***, and ***HTTP***
  - The graphical Mosaic ***web browser*** in 1993 developed by a team at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign (UIUC)
  - ***Web services*** are services offered by software components that communicate with each other over web

API

- An application programming interface (API) specifies how two software components should interact
- Recall that Socket is the API provided by transport layer to application protocols
- How about APIs provided by web services?

REST

- REpresentational State Transfer
  - Roy Thomas Fielding, Ph.D. thesis 2000
    https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- An architectural style, not a protocol, not a standard

> An architectural style is **a coordinated set of architectural constraints** that restricts the roles and features of architectural elements, and the allowed relationships among those elements, within any architecture that conforms to that style.
>
> -Roy Fielding

See more at "A little REST and Relaxation" by Dr. Roy. T. Fielding
https://roy.gbiv.com/talks/200804_REST_ApacheCon.pdf

REST constraints

- Client-server
- Stateless
- Caching
- Uniform interface
- Layered system
- Code-on-demand

REST

- Request (client -> server): to create / delete a specified resource, to read / update its state
  - Uniform Resource Identifier (URI)
  - The verb specified by standard HTTP methods: GET, PUT, POST, PATCH, DELETE
- Respond (server->client): representation(s) of the (current) state of the resource
  - State often represented in JSON / XML

URI, URL, and URNs

- Terminology:
  - URI = Uniform Resource Identifier
  - URL = Uniform Resource Locator
  - URN = Uniform Resource Name
- URNs and URLs are both URIs
- What is the difference?
  - URI: Uniquely identifies a resource
  - URL: identifies + provides location of resource
  - URN: identifies but not locates
    - For example, in the International Standard Book Number (ISBN) system, ISBN 0-486-27557-4 identifies a specific edition of Shakespeare's play Romeo and Juliet. The URN for that edition would be urn:isbn:0-486-27557-4.

Resources in REST

- Two types of resources
  - Collections: /rooms
    - Container, referencing other things
  - Instances: /rooms/2.506
    - Single instance
- Resources are referenced in the HTTP request line
  - GET /rooms/2.506 HTTP/1.1

Representation vs. resource

- A representation captures the current or intended state of a resource
- A resource can have multiple representations
  - Html, xml, json
  - Pdf, png
- Resources are transferred between the client and the server in some form of representation
  - Content negotiation

Question: which one to use?

- http://example.com/cities/Singapore.json
- http://example.com/cities/Singapore
- http://example.com/query_city? name=Singapore

Idempotence and safe methods

- Safe methods: not modify (non-trivial) server resources
  - Example: GET and HEAD do not change the resource on server
- Idempotent methods: may modify server resources
  - But can be executed multiple times without changing outcome
  - Example: duplicate DELETE operations have no additional effect
  - POST is *not* idempotent, multiple POSTs have multiple effects. Example: multiple rooms are created for POST /rooms
- Implications:
  - Safe methods enable caching and load distribution
  - Idempotence allows to handle lost confirmations by re-sending

PUT vs POST

- Normally, POST is used to create new resources (and get ID), PUT is used to update
- POST can be used to create an element in a collection, without explicit name
  - Server will reply with 201 message with URL of created element
- PUT can be used to update an existing resource
  - Reply will be 200