Web Services

CS 442: Mobile App Development

Michael Lee <lee@iit.edu>

Definitions

What is a web service?

(What is any software service?)

A software is a piece of execuatble program that can be compliled and gives ouput
Software service is something tht is a collection of such methosd/softwares/API that provides the service of doing something instead of the user.

API - even a smal piece of local library/package is an API

a software service comprises:

- a documented set of API routines
- well defined procedure call and data exchange mechanisms

A way of communication between the user and the software service provider

a web service is a software service invoked “over the web”

we contact the remotely available service through web like the web browser

i.e., web-based *remote procedure call* (RPC)

“over the web” = conducted via **HTTP**

HTTP ≡ hypertext transfer protocol,
lingua franca of web clients/servers

HTTP is a **stateless** protocol

i.e., it does not inherently track the state or progress of a client/server conversation

e.g., Client: Hi, I’m Michael

Server: Hi Michael

Client: My password is @#$^!*

Server: Who are you again?

Client: I’m Michael, with password @#$^!*

Server: You’re logged in, with token %$#@

Client: Update resource 1234 for me

Server: You’re not logged in!

“Stateless” doesn’t mean that HTTP requests cannot change server state!

It just means that each request needs to supply **all** relevant information.

e.g., C: Hi, I'm Michael, logging in with password @#$^!*

S: You're logged in, with token %$#@

C: Use token %$#@ to access resource 1234

S: Here's resource 1234 { … }

C: Use token %$#@ to update resource 1234 with new data XYZ

S: Resource 1234 has been updated

only nine HTTP methods (aka “verbs”);

only 4 typically used in web service APIs:

- GET
- POST
- PUT
- DELETE

like an API with a fixed number of functions

… how to build rich services with this?

Representational State Transfer, aka “REST”

- set of principles and constraints for designing web services atop HTTP

Concept 1: Resources

- URLs represent **resources**
  - e.g., http://foo.org/users,
    http://bar.org/cart/1234/items
    http://baz.org/author/john/articles

Concept 2: HTTP methods = actions

- GET: Read (a resource)
- POST: Create (a resource)
- PUT: Update (an existing resource)
- DELETE: Delete (an existing resource)

POST/GET/PUT/DELETE = “CRUD”

Concept 3: Statelessness

- Each request must contain all information required to process it
- Server tracks no context!
- Resource state is communicated as necessary / requested

but the token provided here has all the previous data shared by user, so just send token and the new data/request

http://blog.com/posts/2023/10/

- GET to retrieve list of posts
- POST to create new post

http://blog.com/posts/2023/10/some-post

- GET to retrieve post body
- PUT to update post
- DELETE to delete post

Issues:

- How do we name resources (aka endpoints)?

Naming /path of where these resources are located - associating methods/actions with route/resources -->Route

  - “Routing” conventions & definitions

- How do we track of conversations?

  - Session-handling

eg - you keep adding multiple items to cart and checkout atlast ...we dont repeat info again...
..the saved items are stored in session

- How do we know when resources change?

  - Polling for updates

application keeps checking for updates constantly to see if there is a change, there is no push notification in http.

We cannot ask the server to notify us when there is a change-because then we swtich the client/server role itself

Modern alternatives to REST:

- GraphQL: more granular resource identification
- WebSockets: full-duplex RTC
- gRPC: high-performance RPC mechanism