RESTful Web servisi (2)

Vrste zahteva nad resursima

- Idempotentne
  - Sigurne
- Nesigurne

Vrste zahteva nad resursima

- Idempotentni
  - Mogu se ponavljati više puta, a da ne izazivaju negativne bočne efekte na serveru

    GET /book
    PUT /order/x
    DELETE /order/y
  - Ukoliko server ne funkcioniše ok, zahtev se može ponavljati dok server ne proradi
  - Sigurni su oni idempotentni zahtevi koji ne menjaju stanje samog resursa na serveru

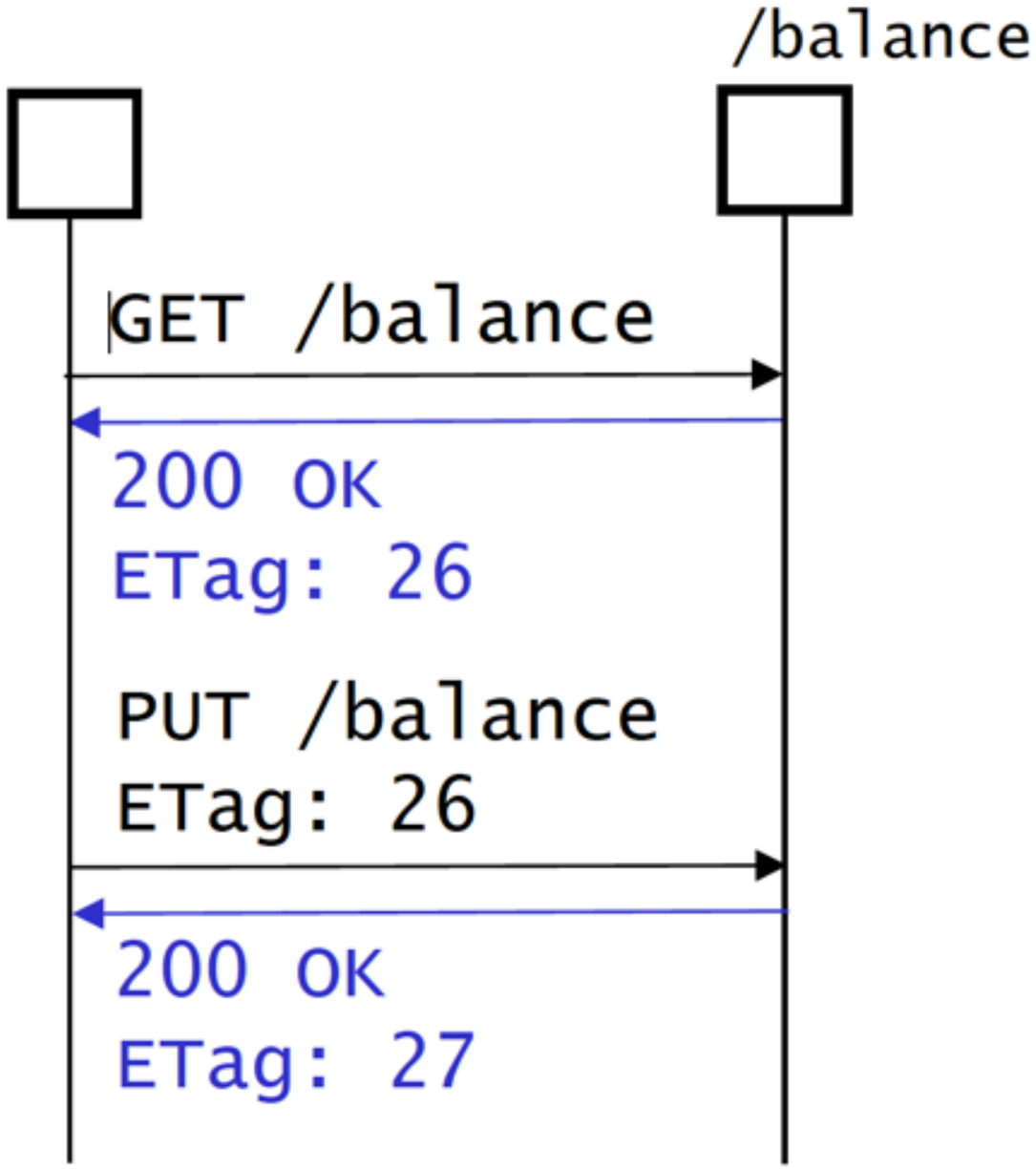
    GET /book

Vrste zahteva nad resursima

- Nesigurni zahtevi
  - ovi zahtevi modifikuju stanje na serveru i ne mogu se ponavljati, a da pri tome ne izazovu neželjene efekte
  - za ovakve zahteve neophodne su dodatne akcije u posebnim situacijama (tzv. state reconsiliation)

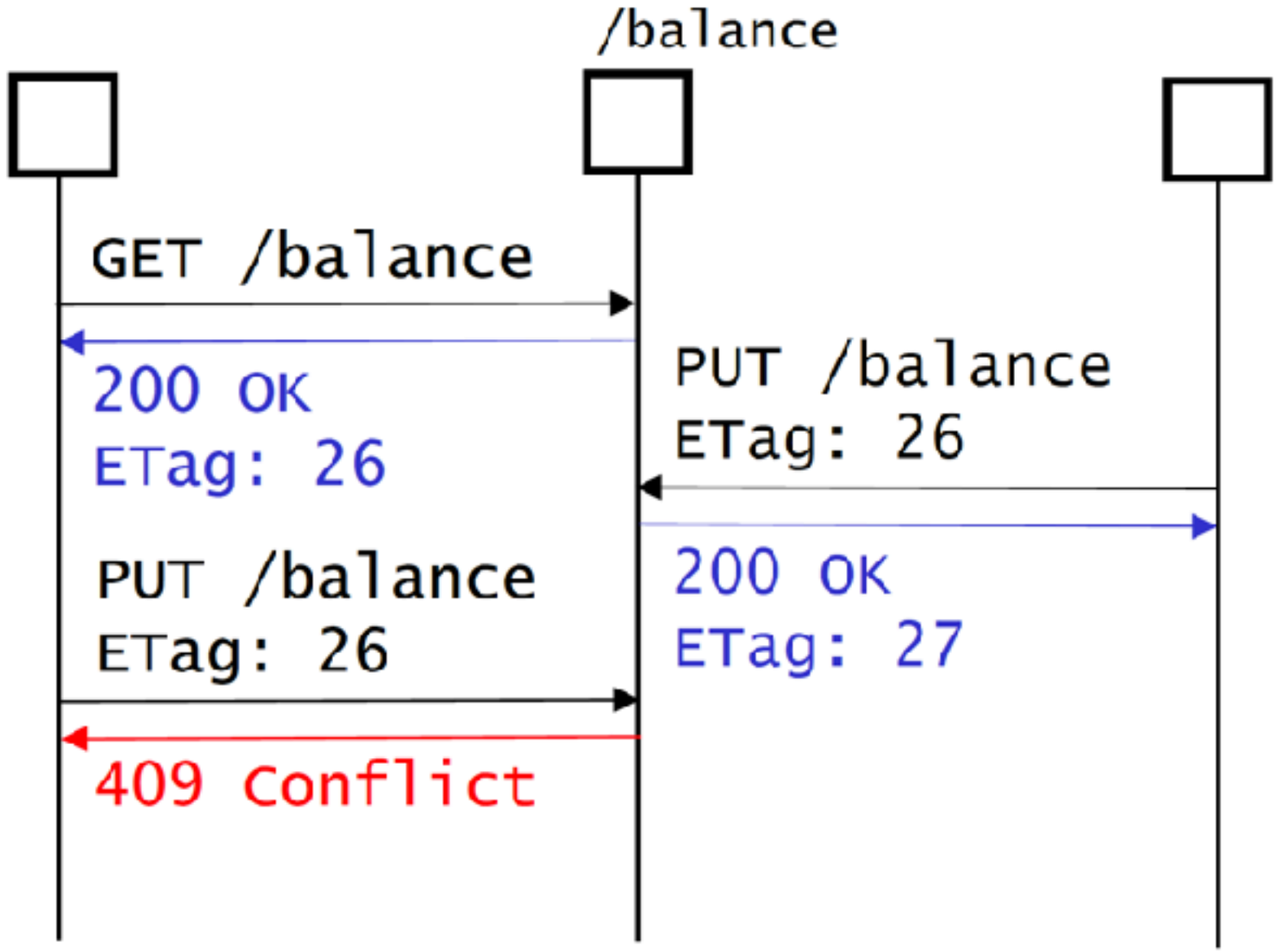
    POST /order/x/payment
  - u nekim situacijama API se može refaktorisati tako da se koriste idempotentne operacije

Problem konkurentnosti



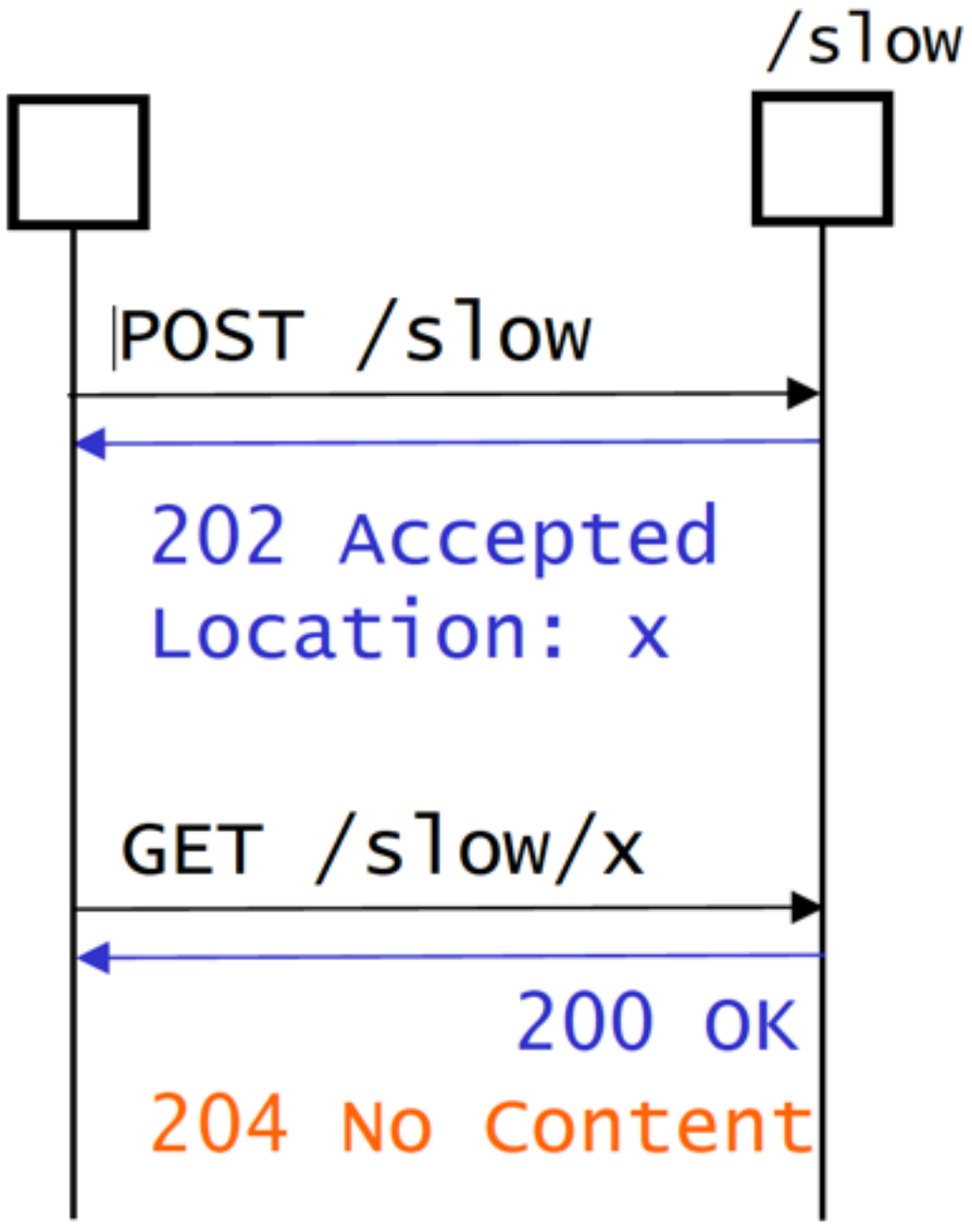
- Razlaganje APIja u niz idempotentnih zahteva pomaže da se prevaziđu problemi privremenih otkaza servisa
- Šta ako drugi korisnik konkurentno ažurira resurs?
- Da li raditi pesimističko zaključavanje resursa (eksplicitno zaključavanje) ili je moguće i neko drugo rešenje?

Problem konkurentnosti



- Status 409 može se iskoristiti da se korisniku prenese informacija da bi izvršenje zahteva dovelo do nekonzistentnosti resursa

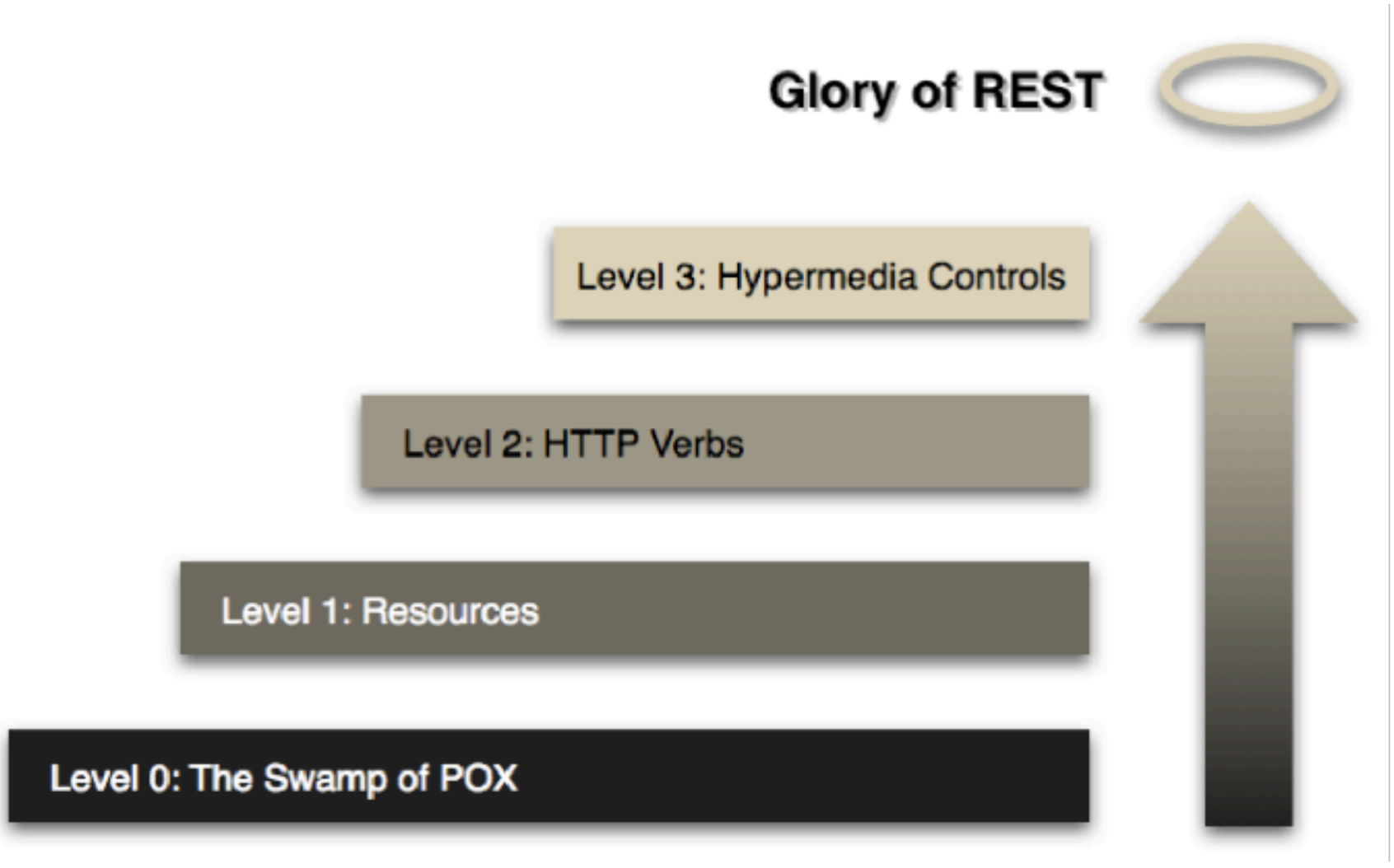
Blokirajući ili neblokirajući zahtevi?



- HTTP je u svojoj suštini sinhron, ali ipak ne mora biti blokirajući
- za zahteve za koje znamo da mogu dugo da traju može se vratiti kod 202, sa informacijom gde naknadno preuzeti rezultat
- koliko često klijent treba da “proba” čitanje rezultata
  - odgovor sa /slow/x bi mogao da sadrži i estimaciju trajanja obrade

Richardsonov model zrelosti REST servisa

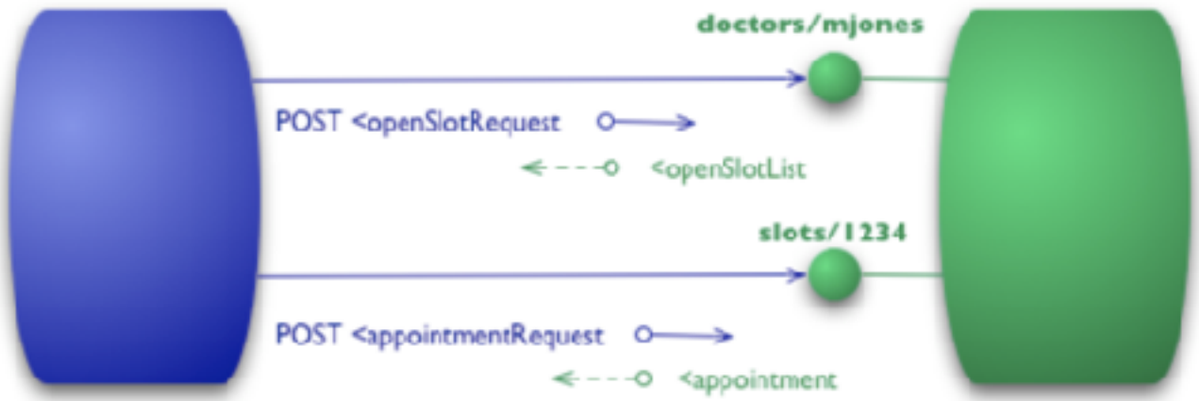
- Model koji je definisao Leonard Richardson koji promenu osnovnih principa REST-a razbija na nekoliko koraka.



Richardsonov model zrelosti REST servisa

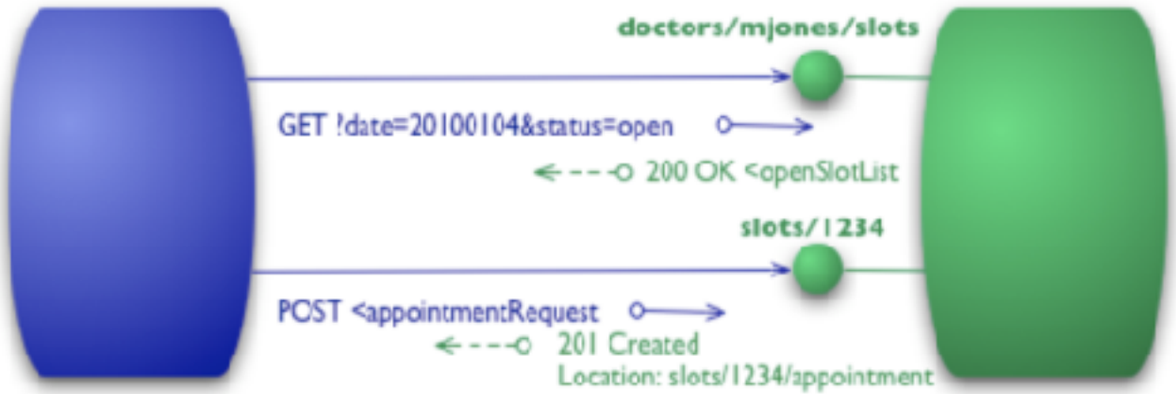
- Level 0: HTTP se koristi kao RPC Protocol (Tuneluje se POST+POX ili POST+JSON). Suštinski se koristi HTTP za udaljeni pristup ali bez korišćenja bilo kojih mehanizama weba. Obično se sve operacije obavljaju preko jedne adrese.

- Level 1: Identifikacija resursa - uvođenje više URI-ja za različite resurse (opšta adresabilnost resursa).

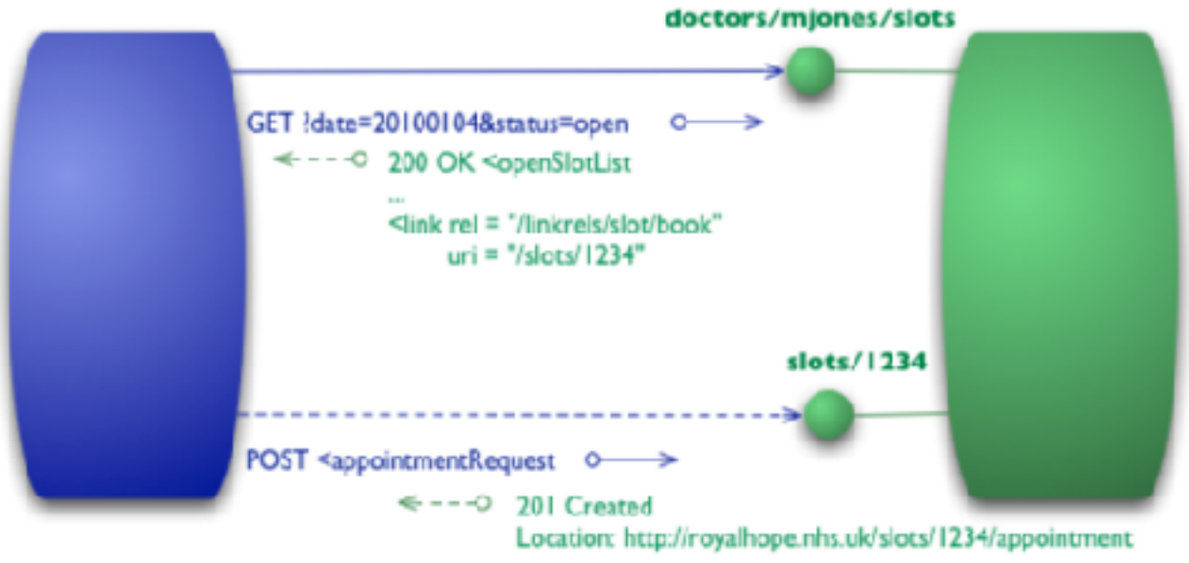


Richardsonov model zrelosti REST servisa

- Level 2: Uniformno korišćenje HTTP metoda (*Contract Standardization*). Na ovom nivou uvodi se konzistenta upotreba HTTP metoda - pojednostavljuje se razumevanje API-ja i bez previše pojašnjenja očekuje se određeno ponašanje. Obratiti pažnju i na povratne statusne kodove.



- Level 3: Hypermedia Controls / HATEOAS (Hypertext As The Engine Of Application State)
  Na ovom nivou i same osobine servisa i mogućnosti koje nudi klijentu mogu da se odrede pristupom servisu



Level 0, *antipattern (antišablon)* za REST

HTTP samo za tunelovanje zahteva

- Tunelovanje kroz 1 HTTP metod

  GET /api?method=addCustomer&name=Wilde
  GET /api?method=deleteCustomer&id=42
  GET /api?method=getCustomerName&id=42
  GET /api?method=findCustomers&name=Wilde*

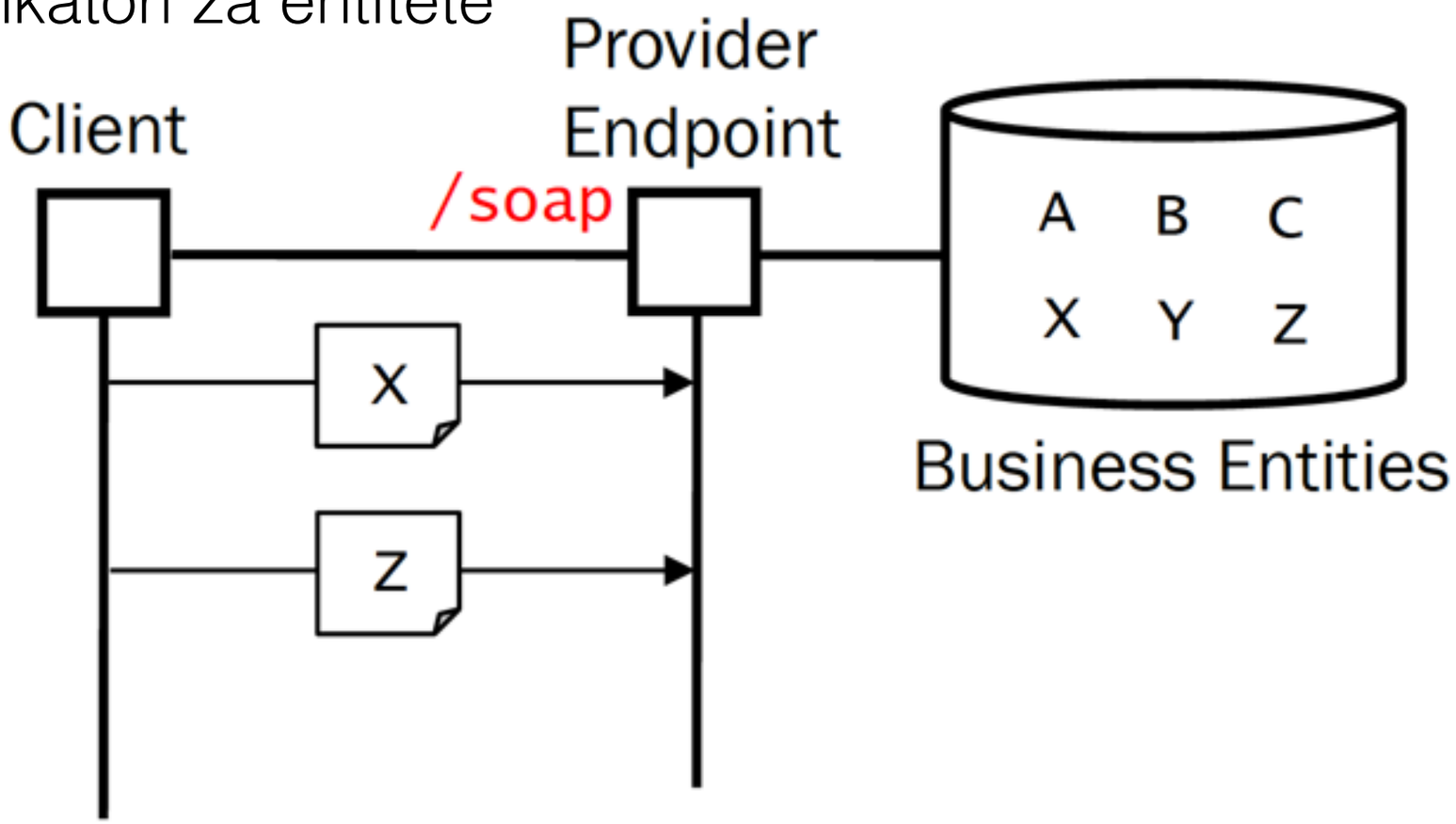
- Sve se radi preko GET zahteva
  - Prednost: lako se testira iz browsera
  - Mana: GET bi trebalo sa se koristi samo za čitanje
  - Ograničenje: max. količina podataka koju je moguće ovako slati je oko 4KB

Antišablon za REST
HTTP samo za tunelovanje zahteva

- Tunelovanje kroz 1 HTTP metod
- Sve se radi preko POST zahteva
- Prednost: moguće je preneti veće količine podataka, pa i uploadovati fajlove
- Mana: POST zahtev se ne smatra idempotentnim niti sigurnim (pa se stoga ne kešira, a i koristi se za potencijalno “nesigurne operacije”)

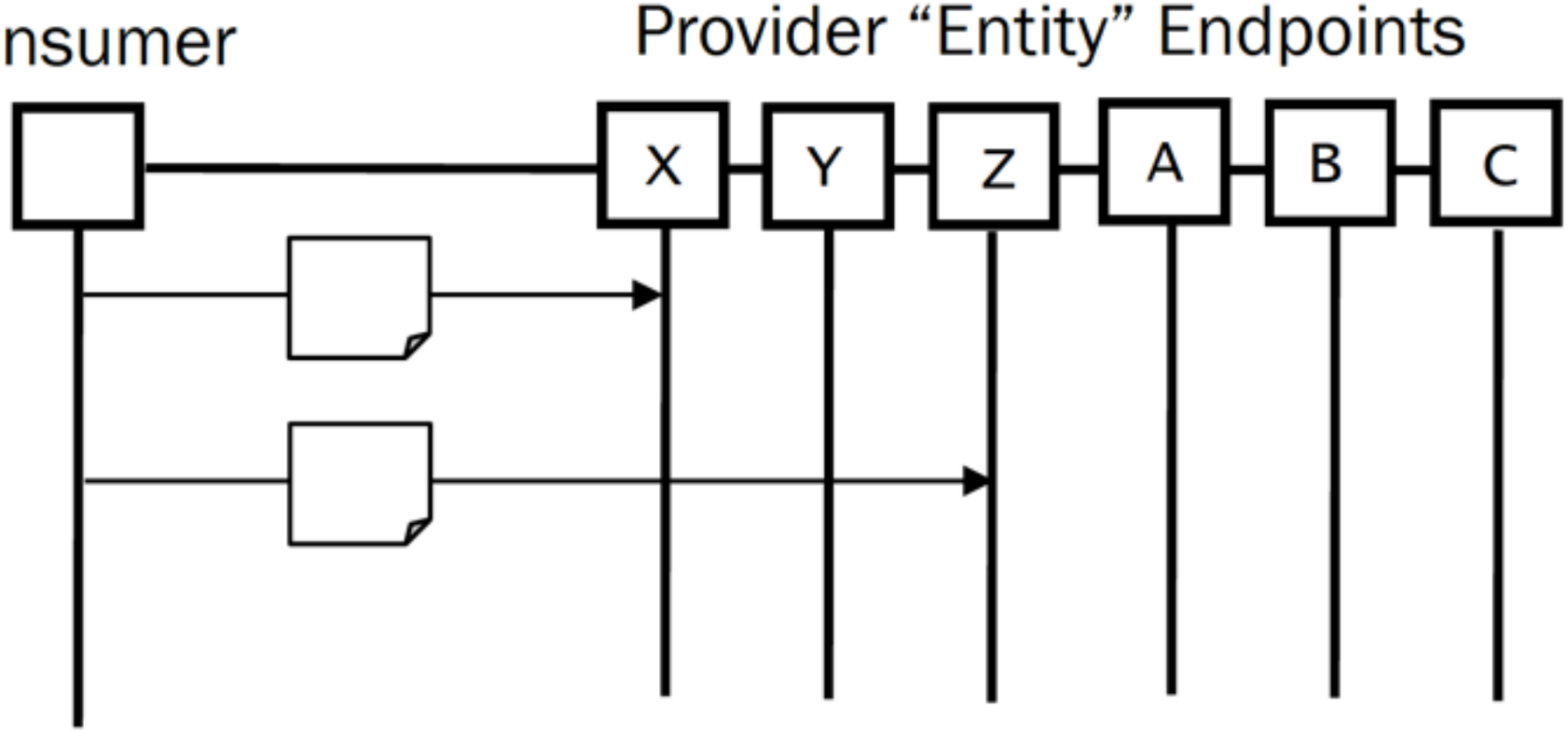
Antišablon za REST
HTTP samo za tunelovanje zahteva

- Tunelovanje kroz 1 endpoint
- Servis sa samo jednim *endpoint-om* nije dovoljno granularan u smislu pristupa pojedinim entitetima-resursima. Klijent tada mora da radi sa najmanje dva identifikatora, jedan identifikator za sam servis, identifikatori za entitete



Opšta adresabilnost

- Rešenje za prethodni problem - svaki resurs postaje dostupan preko svog individulanog *endpoint-a*



Antišablon - Cookies

- Da li su Cookies RESTful?
  - Zavisi od načina primene - REST se zasniva na *stateless* komunikaciji

1. “kolačiči” mogu i sami biti “zatovreni”/samoopisujući - tako da sadrže sve informacije koje su neophodne za njihovu interpretaciju pri svakom zahtevu/odgovoru
2. “kolačići” sadrže samo reference na stanje aplikacije (koje se samo ne održava kao resurs)
   - prenose samo tzv. ključ sesije
   - Prednost: manje podataka se prenosi
   - Mane: Zahtevi više nisu “zatvoreni” pošto nose informaciju o nekom kontekstu koji bi server trebao da pamti.
     Takođe neophodno je imati mehanizam koji će počistiti nekorišćene reference (istekle sesije)

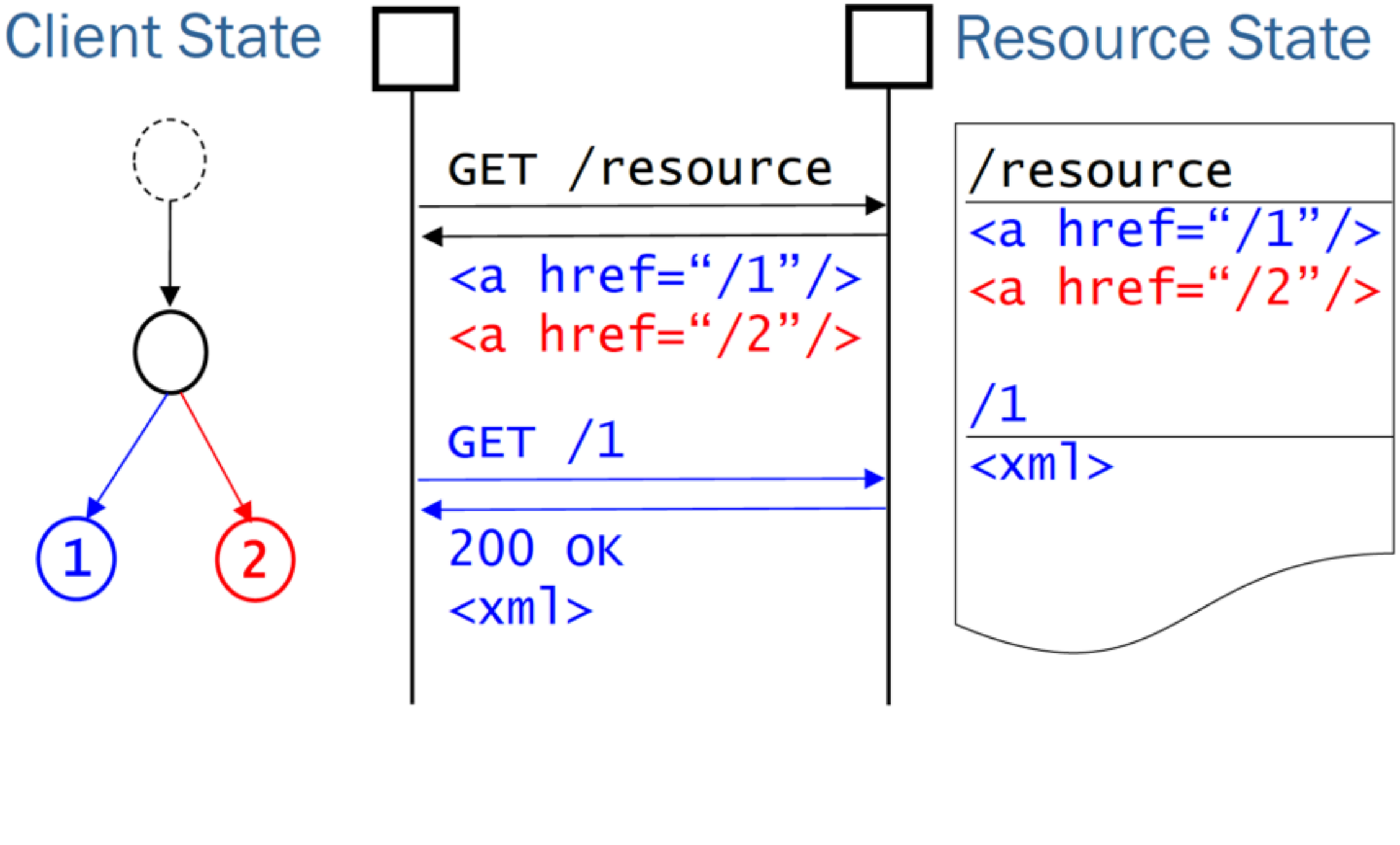
Stateless vs. Stateful?
Stanje aplikacije

- Samo ime REST-a sugeriše da se posmatra stanje resursa u distribuiranom sistemu, ali komunikacija je stateless.
- Stanje na klijentu:
  - Klijent se kreće po raspoloživim resursima tako što prati linkove, njegovo stanje je određeno posećenim linkovima
  - Server može da utiče na tranzicije stanja koje su klijentu raspoložive tako što mu u odgovoru na GET zahteve može slati hiperlinkove do resursa koje treba pratiti

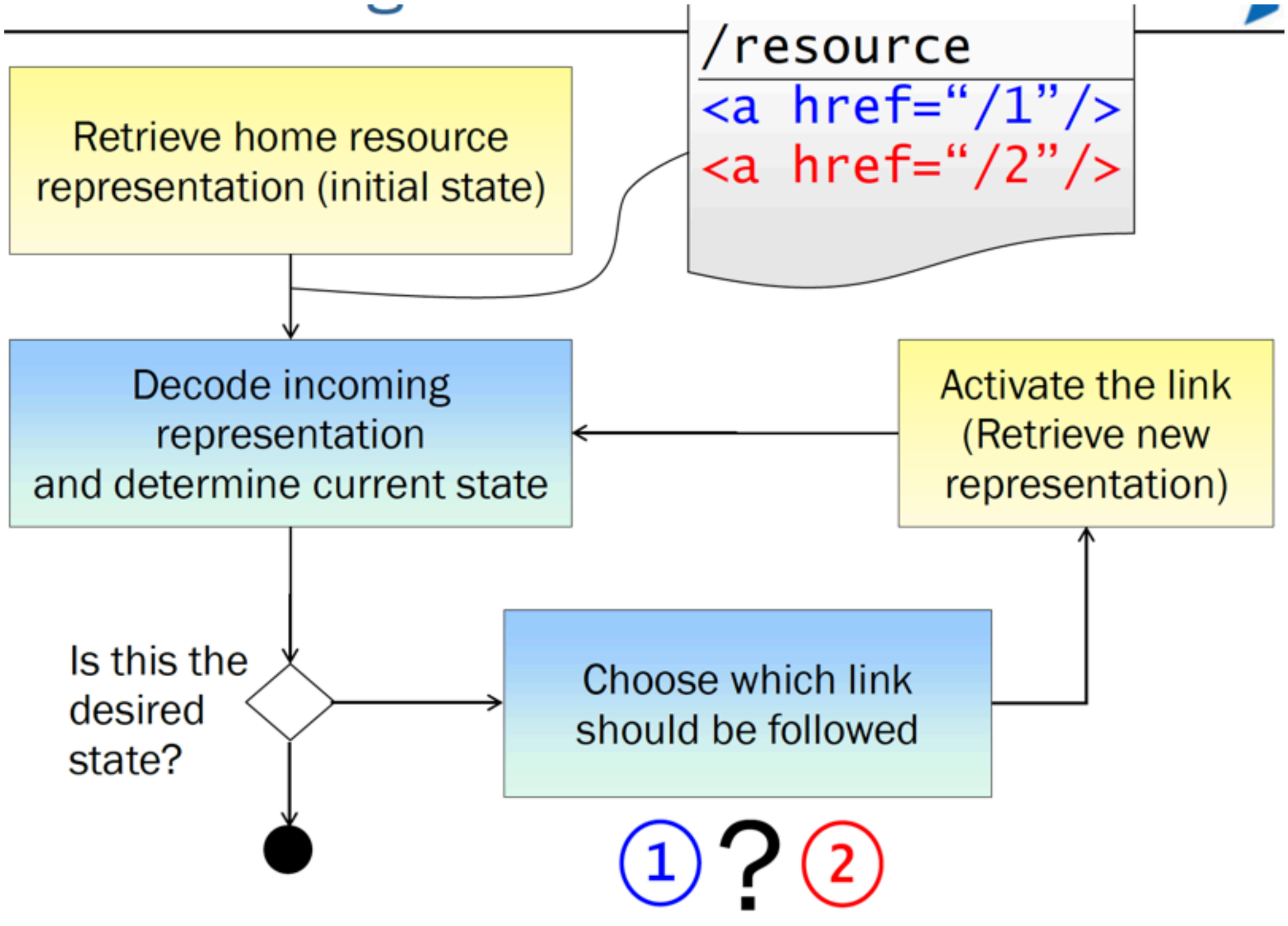
Stanje aplikacije

- Stanje na serveru:
  - Stanje resursa definiše perzistetno trenutno stanje aplikacije
  - Klijenti mogu pristupiti resursima koristeći različite reprezentacije
  - Klijent manipuliše stanjem resursa na serveru preko uniformisanog interfejsa koji obavlja CRUD operacije

Stanje aplikacije

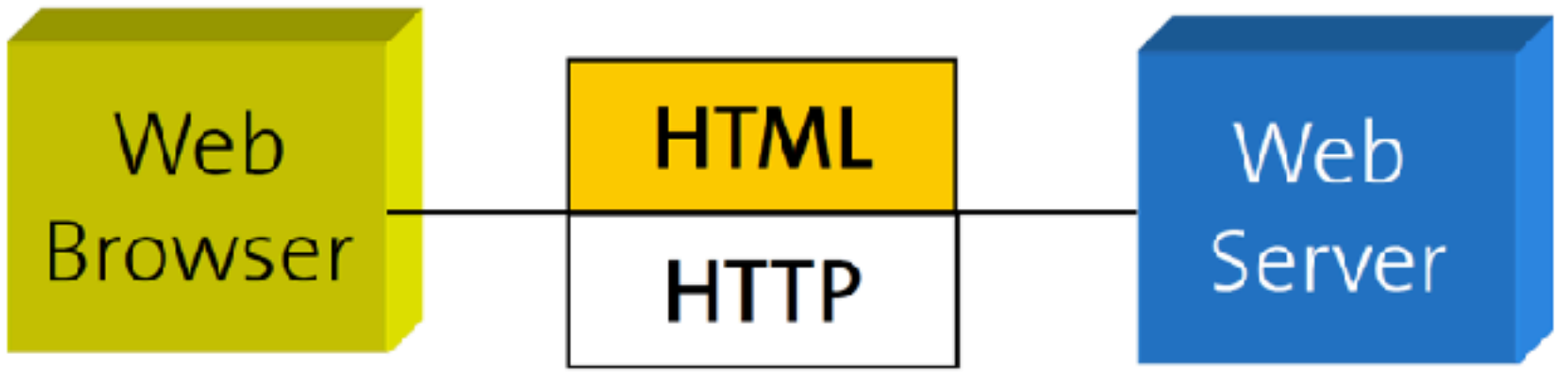


Algoritam rada klijenta

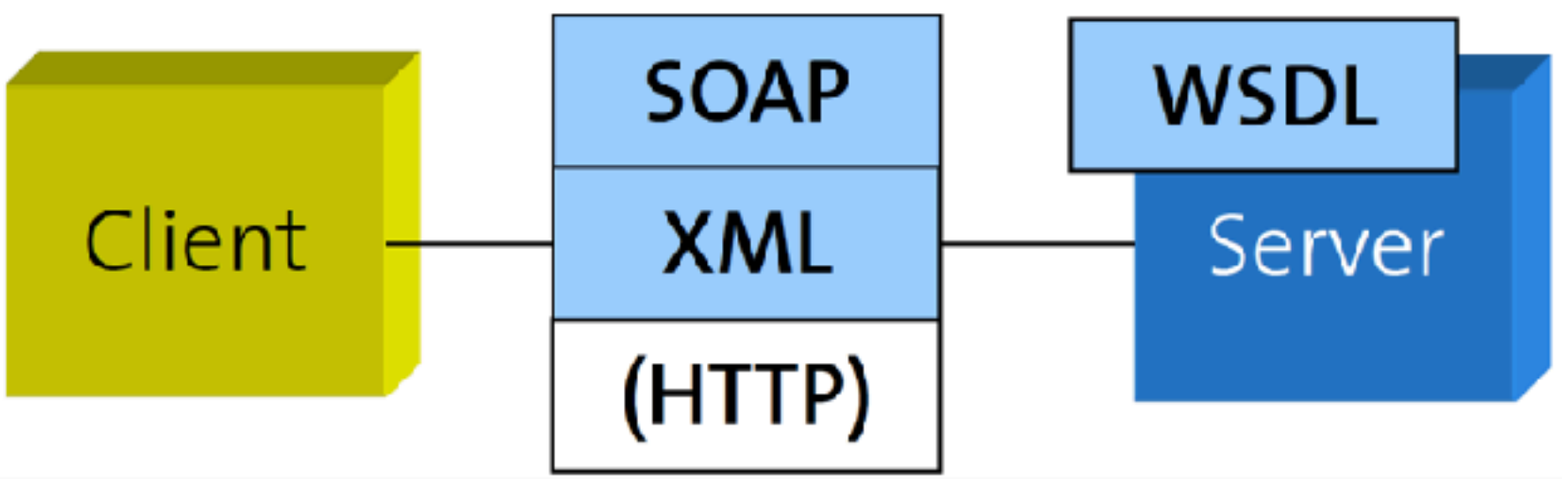


*REST vs. WS-**

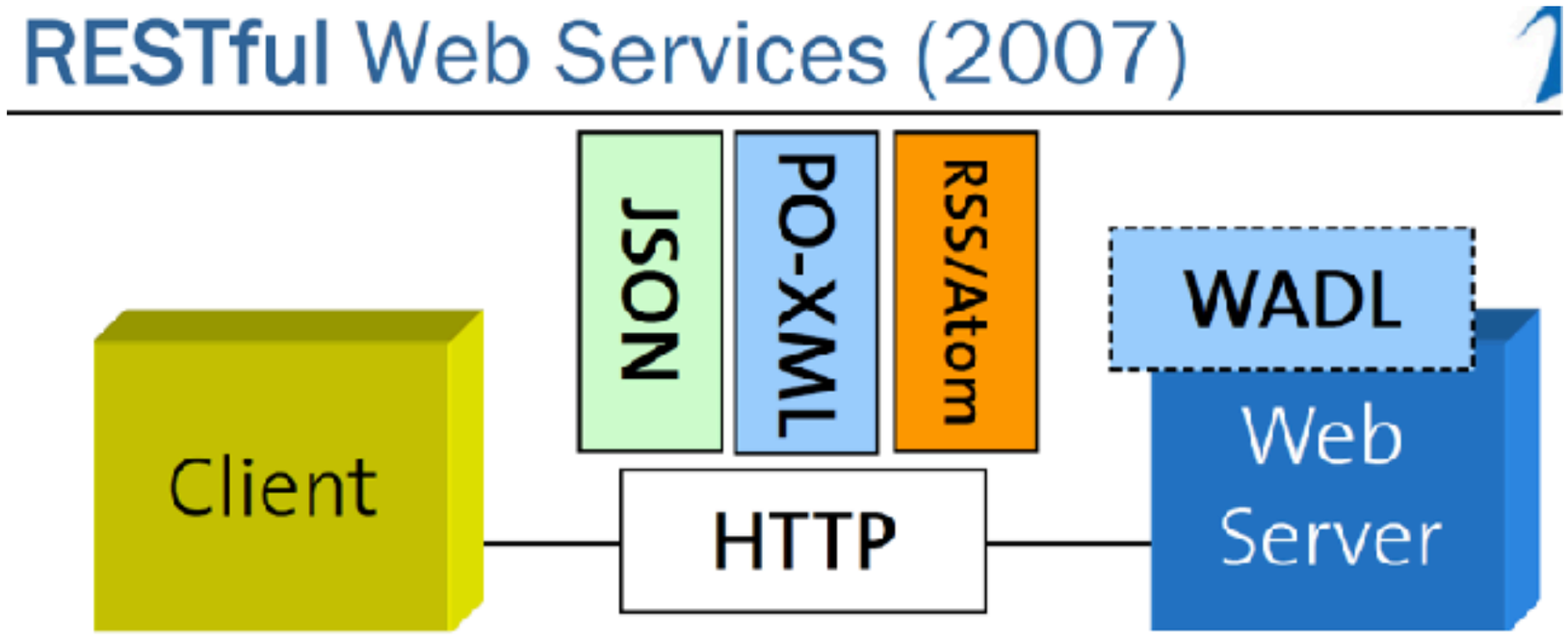
Web Sites (1992)

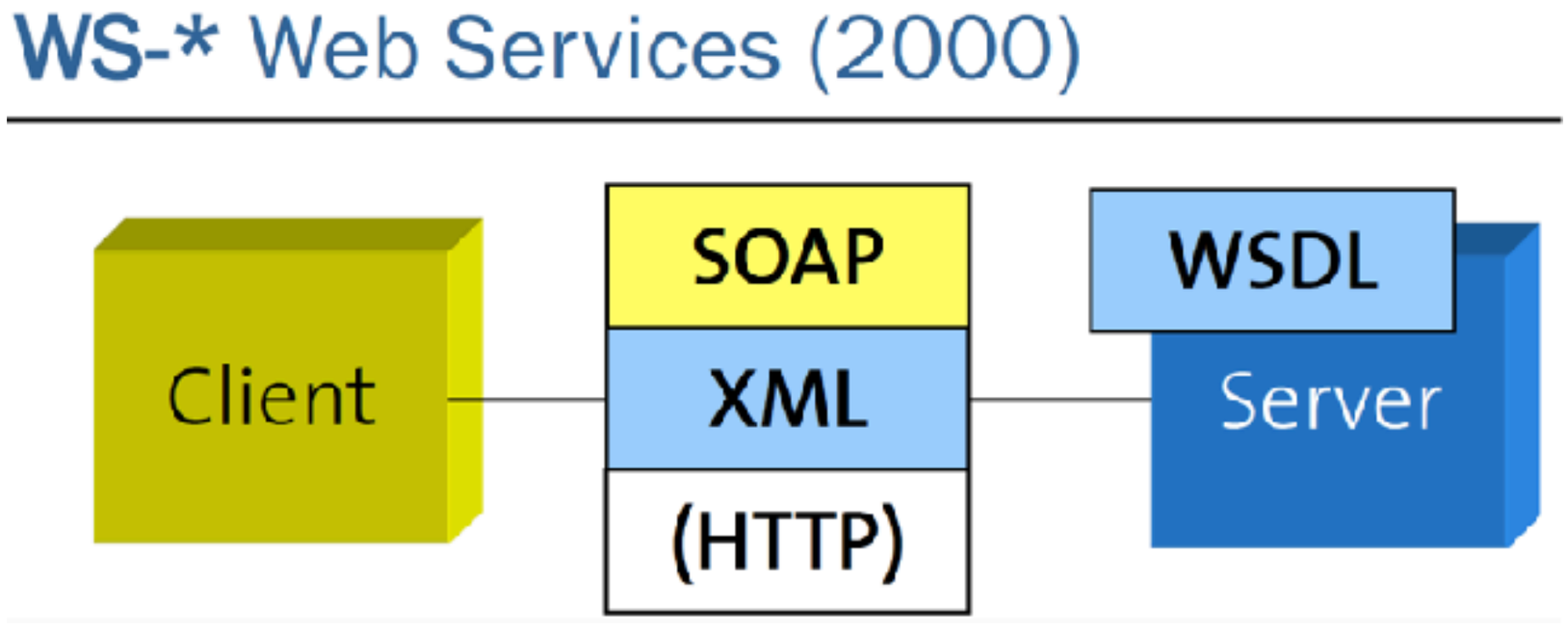


WS-* Web Services (2000)

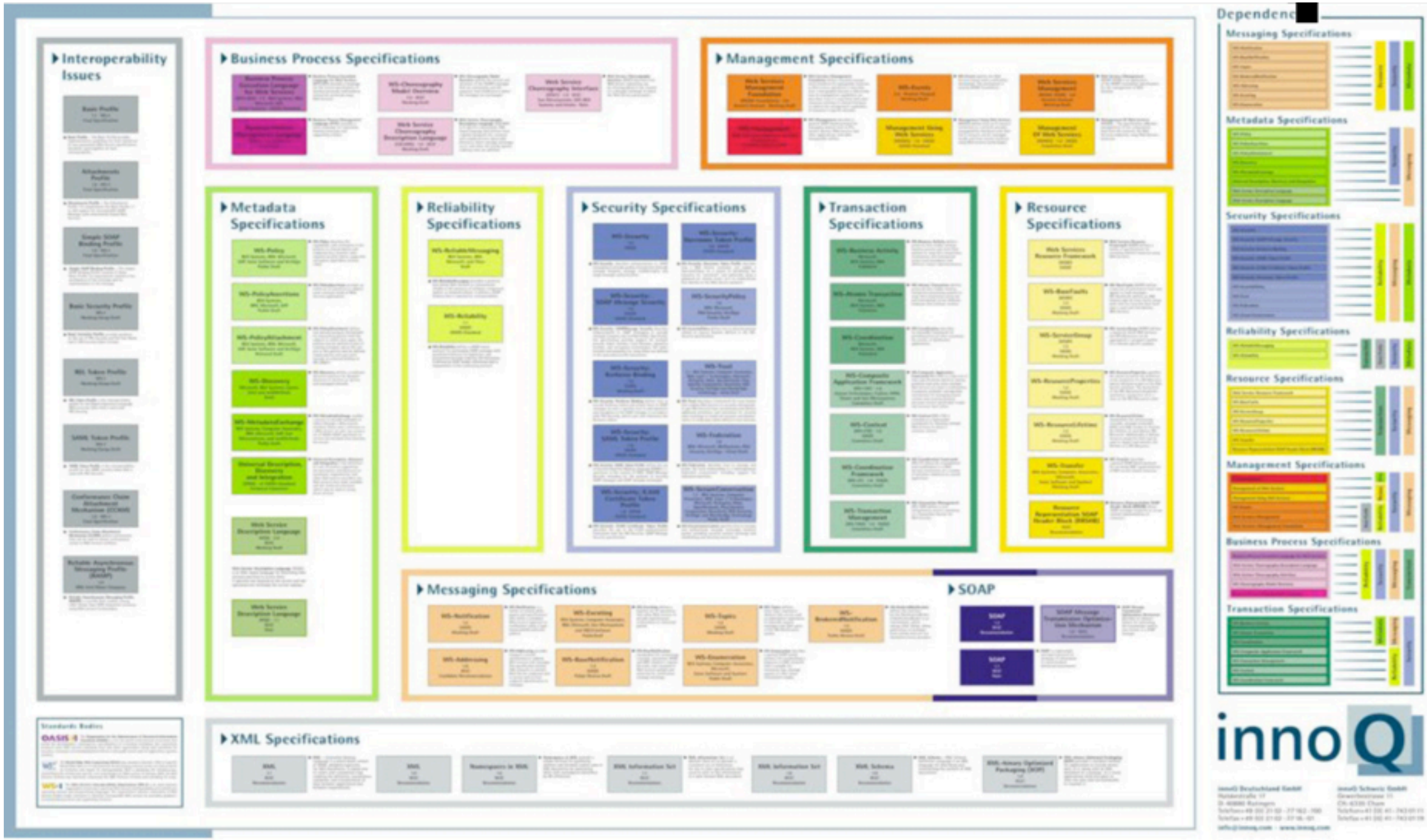


*REST vs. WS-**


RESTful Web Services (2007)
Client
JSON
PO-XML
RSS/Atom
HTTP
WADL
Web
Server


WS-* Web Services (2000)
Client
SOAP
XML
(HTTP)
WSDL
Server

WS- standardni stack tehnologija*



REST standardni stack tehnologija