Paweł Rajba
pawel@cs.uni.wroc.pl
http://pawel.ii.uni.wroc.pl/

RESTful Services

Agenda

- Introduction
- Drivers for REST
- Properties of REST
- Common wrong assumptions
- REST constraints
- Uniform Interface
- RPC vs. REST
- Maturity Model
- Client
- Is always REST the best?

Introduction

- Issues with SOAP web services
  - Tight coupling between client and server
  - When changing server API, all clients need to be rebuild and tested
  - No support for mobile clients
  - Many complicated specifications very often supported only partly by libraries
- Next level of web services: REST

Drivers for REST

- Interoperability
  - Network based API vs. library based API
  - Many services in the web
- Devices
  - phones, tablets, laptops, cars, fridges benefits from different services
- Scalability
  - architecture should allow to benefit from infrastructure scalability, e.g. benefit from cloud like Azure or Amazon

Properties of REST

- Heterogeny
- Scalability
- Evolvability (of clients)
- Visibility
- Reliability
- Efficiency
- Performance
- Manageability

REST is…

- REST is not
  - RPC (with SOAP and WSDL)
  - HTTP (not every HTTP API is RESTful)
  - URIs (cool URLs don't make REST)
- REST is an architectural style
  - Representational State Transfer (state machine)
  - Roy Fielding and year 2000

Common wrong assumptions

- Network liability
  - always there are problems, even in big cloud providers
- Latency
  - there is always one, in distributed systems even bigger
- Bandwidth
  - there is no infinite one, sometimes a cost is associated
- Security
  - network is not secure
- Network topology
  - is changing, client and servers are moving around, especially in cloud environment, e.g. IP addresses, DNS names, relative paths and URLS (e.g. in scaling)

Common wrong assumptions

- Administration
  - usually there is no one administrator, system needs to be created in a way of easy to administer and low learning curve
- Transport cost
  - is not zero, someone needs to pay for it (even it is hidden)
- Heterogeneous network
  - nodes in the Web are different
- Complexity
  - it could a problem that not everyone understands how to use a service (understands semantic behind the methods, data model)

REST constraints

- Client-Servers separation of concerns
  - Drivers
    - Complexity – only clients know about servers
    - Heterogeneous network
  - Benefits
    - Portability (of clients)
    - Scalability
    - Evolvability (of clients)

REST constraints

- Stateless
  - Doesn't mean there is no session – it is on client
  - Drivers
    - Network reliability
    - Network topology
    - Complexity
    - Administration
  - Benefits
    - Visibility
    - Reliability
      - If server fail, client can just ask again or react appropriately
      - There is no need to repeat whole transaction
    - Scalability
      - much easier to add more servers and implement load balancers

REST constraints

- Cacheable
  - Answers from server must be marked as cacheable or not
  - Drivers
    - Latency
    - Bandwidth
    - Transport cost
  - Benefits
    - Efficiency
    - Scalability
    - Performance

REST constraints

- Layered system
  - Client can't assume direct connection with server
  - Drivers
    - Network topology
      - We don't care about topology, maybe cache components are involved
    - Complexity
      - Complexity can be hidden
    - Security
      - Additional components can be added to increase security in trust boundary
  - Benefits
    - Scale
    - Manageability

REST constraints

- Code on Demand
  - Allows to download implemented features
  - Optional constraint
  - Pros
    - You got ready stuff
    - Client is simplified
      - E.g. federation using OpenID Connect (with form to AuthN)
  - Cons
    - Visibility
    - Security

REST constraints

- Uniform Interface
  - Constraints:
    - Identification of resources
    - Manipulation through representations
    - Self-descriptive messages (e.g. by adding media types information)
    - HATEOAS (hypermedia as the engine of application state)
  - Drivers
    - Network reliability
    - Network topology
    - Administration
    - Heterogenous network
    - Complexity
  - Benefits
    - Visibility
    - Evolvability

REST constraints

- Main constituents
  - Resource
    - Resource identifier
    - Resource metadata
  - Representation
    - Representation Metadata
  - Control Data
  - Hypermedia (HATEOAS)
    - Hypermedia control

Uniform Interface

- Resource
  - A concept (expressed by a noun, no verbs)
  - Mapped to entities
  - Should be stable during the time
  - Examples of resources
    - Customers
    - Customer orders
    - Statuses of an customer's order
    - Courses in which a student is enrolled
    - Blog post's comments
- Read more
  - http://mark-kirby.co.uk/2013/creating-a-true-rest-api/
  - http://www.restapitutorial.com/lessons/restfulresourcenaming.html
  - https://www.thoughtworks.com/insights/blog/rest-api-design-resource-modeling

Uniform Interface

- Resource identifier
  - Allows to access resources
  - Uniquely identify resources
  - Examples
    - http://ex.com/customers
    - http://ex.com/customers/<id>/orders
    - http://ex.com/customers/<cid>/orders/<oid>/status

Uniform Interface

- Resource metadata
  - In HTTP information attached with HTTP headers
  - Identifier is in the **Location** header
  - **ETag** state in time, helpful in
    - caching or
    - optimistic concurrency scenarios

Uniform Interface

- Representation
  - Resource state at point in time
  - Resource may have a few representations
  - Representation Metadata describes representation
    - Helps client and server with understanding how to process data (structure, meaning)
  - Content negotiation is a process of selecting the representation
    - Accept (e.g. **application/json**, but also **image/png**)
    - Accept-Encoding
    - Accept-Language
    - Accept-Charset

Uniform Interface

- Control Data
  - Change the default behaviour of a client or server
    - … or intermediate components (e.g. proxies)
  - Expressed in
    - status codes (very important!)
    - headers
      - if-none-match
      - cache-control
      - …

Uniform Interface

- Common interpretation of HTTP methods
  - GET – retrieve
  - POST – create
  - PUT – full entity update
  - PATCH – delta update
  - DELETE – delete

Uniform Interface

- HTTP Status Codes
  - Very important to use it
  - Several most common ones
    - 200 Ok
    - 201 Created
    - 204 No content
    - 400 Bad request
    - 401 Unauthorized
    - 404 Not found
  - More
    - http://www.restapitutorial.com/httpstatuscodes.html

Uniform Interface

- Hypermedia
  - Drives decoupling between client and server
    - client doesn't need to know about the whole structure
  - Client should get only entry point and navigate through resolved associated entries (news feed)
  - HATEOAS
    - Hypermedia As The Engine Of Application State
  - Parameters
    - rel – type of relationship
    - href – URL which uniquely identifies a resource
  - Media types decribe type of content
    - Register page: https://www.iana.org/form/media-types
    - Interesting media type: HAL (http://stateless.co/hal_specification.html)
    - WRML, interesting article: https://www.programmableweb.com/news/rest-api-design-put-type-content-type/2011/11/18

Uniform Interface

- Hypermedia control
  - Based on HTML
    - `<a href="http://localhost/orders/delayed" rel="delayedOrders">Delayed Orders</a>`
    - `<form class="newOrder" action="orders" method="post">`
      `…`
      `</form>`
  - Based on XML
    - ```
      <?xml version="1.0"?>
      <account>
        <account_number>12345</account_number>
        <balance currency="usd">100.00</balance>
        <link rel="deposit" href="https://somebank.org/account/12345/deposit" />
        <link rel="withdraw" href="https://somebank.org/account/12345/withdraw" />
        <link rel="transfer" href="https://somebank.org/account/12345/transfer" />
        <link rel="close" href="https://somebank.org/account/12345/close" />
      </account>
      ```

https://en.wikipedia.org/wiki/HATEOAS
      ```

Uniform Interface

- Types of hypermedia

Line type	Example
Embedded	HTML <img>
Outbound	HTML <a>
Templated	HTML form with method GET
Idempotent	HTML form with method PUT
Non-idempotent	HTML form with method POST

Uniform Interface

- Description
  - We still need to understand resources, representations, media types, etc.
  - There are description standards, but should be used differently than WSDL
  - Some proposals
    - Swagger
      - http://swagger.io/
    - WADL
      - http://www.w3.org/Submission/wadl/
    - XRD
      - http://docs.oasis-open.org/xri/xrd/v1.0/xrd-1.0.html

RPC vs. REST comparison

RPC	REST
Contract is made by service and operations getUser(), addUser(), getAddress()	Contract is Uniform Interface (UI)
Actions meaning come from specification	Actions meaning come from all constituents (UI, state transitions, etc.)
Errors meaning are part of specification	Errors meaning come from UI
Limited cache support	Caching is built into the REST design
Client and server own the URL namespace	Only server owns the URL namespace
Input and outputs come from runtime types	Input and outputs come from media type specification
Can be tranported by any protocol HTTP, MSMQ, TCP, etc.	Tightly connect with UI related protocol (usually HTTP)

Richardson Maturity Model

- Level 0: POX (plain old XML)
- Level 1: Resources
- Level 2: HTTP Verbs
- Level 3: Hypermedia

- Only Level 3 means that a RESTful service

Client

- Client shouldn't assume any URL structure
  - Totally different approach than in RPC SOAP WS
- The whole interactions should be based on hypermedia
  - The only known URL is an entry point
  - Similarity to web browser

Is always REST the best?

- SOAP+WS-*
  - More capabilities based on WS-*
  - More control, better in more complex interactions
  - Many tools
  - Cheaper in one-time
- REST
  - A lot of advantages discussed so far
  - Many cloud providers offer REST API (Amazon, Yahoo, …)
  - Less capabilities, less tools
  - Cheaper in run-time

DEMO

- GMail API
  - Main pages
    - Strona główna
      - https://developers.google.com/gmail/api/
    - Wprowadzenie
      - https://developers.google.com/gmail/api/guides/
    - Dokumentacja
      - https://developers.google.com/gmail/api/v1/reference/
  - Steps
    - List of messages
      - https://developers.google.com/gmail/api/v1/reference/users/messages/list#try-it
    - Selected message
      - https://developers.google.com/gmail/api/v1/reference/users/messages/get#try-it
    - List of lables
      - https://developers.google.com/gmail/api/v1/reference/users/messages/list#try-it
    - Create a new label
      - https://developers.google.com/gmail/api/v1/reference/users/labels/create#try-it
  - What's wrong?

DEMO

- Google Tasks
  - But this time we use interesting tool:
    - https://apigee.com/console/google-tasks

DEMO

- Examples of other services
  - GitHub
    - https://developer.github.com/v3/
  - PayPal
    - https://developer.paypal.com/docs/api/
  - Twilio
    - https://www.twilio.com/docs/api/rest
  - Confluence
    - https://docs.atlassian.com/confluence/REST/latest/
  - Foxy
    - https://api.foxycart.com/docs

DEMO

- Creating HTTP Api
  - BooksApi
    - Test with PostMan
  - Status code in WebApi
    - Custom status code: return StatusCode(404);
    - 200: return Ok()
    - 201: return Created()
    - 204: return NoContent()
    - 400: return BadRequest()
    - 401: return Unauthorized()
    - 404: return NotFound()
- Read more
  - https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api

Next steps

- Investigate
  - Swagger
    - http://swagger.io/
    - http://petstore.swagger.io/
    - https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger
  - OData
    - http://www.odata.org/

References

- RESTful services

  http://en.wikipedia.org/wiki/Representational_state_transfer
  https://spring.io/understanding/HATEOAS
  https://en.wikipedia.org/wiki/HATEOAS
  http://www.restapitutorial.com/lessons/whatisrest.html
  http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven
  https://www.thoughtworks.com/insights/blog/rest-api-design-resource-modeling
  http://restcookbook.com/
  http://whatisrest.com/
  http://www.infoq.com/articles/roy-fielding-on-versioning
  http://msdn.microsoft.com/en-us/magazine/dd315413.aspx
  http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
  https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
  http://blog.steveklabnik.com/posts/2011-07-03-nobody-understands-rest-or-http

- WADL

  https://wadl.java.net/
  http://www.w3.org/Submission/wadl/
  http://en.wikipedia.org/wiki/Web_Application_Description_Language
  http://stackoverflow.com/questions/2215646/difference-between-wsdl-2-0-wadl-xrd