# REST vs Web Services

TIE-23600 Palvelupohjaiset järjestelmät

Tämän kalvosetin lähteenä on käytetty julkaisua
Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. 2008. Restful web services vs. "big"' web services: making the right architectural decision. http://ramb.ethz.ch/CDstore/www2008/www2008.org/papers/pdf/p805-pautassoA.pdf

# Web Services (WS-*)

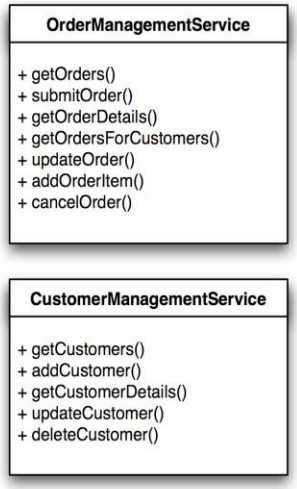
- Web Services = ”WS-*” = a set of XML-based standards for implementing SOA
  - SOAP as message format
  - WSDL as service contract definition language
  - Other WS-* standards for other things

# REST

- Architectural style defined by six constraints
  - Client-server
  - Stateless
  - Cacheable
  - Layered system
  - Uniform interface
  - Code-on-demand (optional)
- HTTP as the uniform interface
  - URI
  - GET/POST/PUT/DELETE
  - MIME representations
  - Hyperlinks between resources

[From: http://www.infoq.com/articles/rest-introduction]

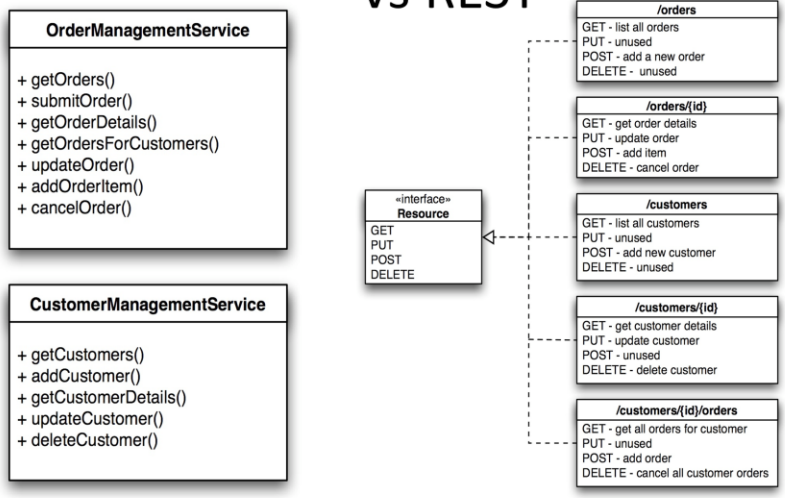
# WS-*




The interfaces are specific to the task.
The interfaces define the services' application protocol.

[From: http://www.infoq.com/articles/rest-introduction]

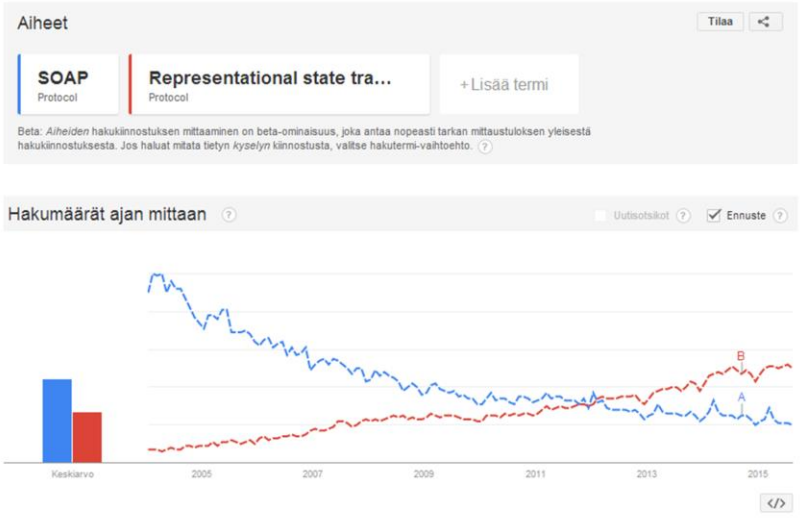
# vs REST




Web service: a client needs to be coded against this particular interface

RESTful service: a client uses URI hierarchy and HTTP to access the resources

# Popularity (SOAP vs REST)



# WS-* strengths

- Protocol transparency and independency
  - SOAP messages can be transported over a variety of protocols, not just HTTP
- Machine-processable service contracts
  - WSDL
- Synchronous and asynchronous interaction patterns
- Tool support
  - Hide the complexity of underlying formats

# WS-* weaknesses

- Verbosity
  - XML processing overhead
- Wide range of standards
  - Varying tool support
  - Mitigated by WS-I guidelines
- ”Complexity”
  - Dependency on tools

# REST strengths

- Leverage well-supported standards
  - HTTP, URI, MIME
- Lightweight
  - No heavy dependendence on tools
- Scalability
  - Statelessness, cacheable, …
- Multiple message formats

# REST weaknesses

- Confusion about what is ”RESTful”
  - Any HTTP API is not RESTful
- Reliance on HTTP
  - Methods other than GET and POST not universally supported
  - GET input data limits
    - Very long URIs don’t work everywhere
    - How to encode complex data in the URI?

# HTTP

**In WS-***

- HTTP is one possible *transport* protocol
  - on top which SOAP messages can be transported
- Only POST
- URI identifies a messaging endpoint
  - Which may contain multiple operations

**In RESTful services**

- HTTP is the *application* protocol
- GET, POST, PUT, DELETE
- URI identifies a resource

# WS-* design decisions

1. Data modeling
   - XML Schema data types
2. Message exchange patterns
   - Synchronous or asynchronous
3. Service operations enumeration
   - Define the set of operations exposed by the service
   - How to group the operations into services?

# REST design decisions

1. Resource identification
2. URI design
3. Resource interaction semantics
   - Which of the HTTP verbs are applicable to a given resource?
4. Resource relationships
   - HATEOAS
5. Data presentation
   - JSON, XML, …

# Technology comparison

| | WS-* | REST |
|---|---|---|
| **Transport protocol** | (many) | HTTP |
| **Message format** | SOAP (XML) | (many) |
| **Service identification** | URI, WS-Addressing | URI |
| **Service description** | WSDL, XML Schema | Textual documentation, WADL |
| **Security** | HTTPS, WS-Security | HTTPS |
| **Service composition** | BPEL, BPMN | (mashups) |
| **Implementation technology** | (many) | (many) |