REST Vs. SOAP

Week 10

REST Vs SOAP

- Simple web service as an example: querying a phonebook application for the details of a given user

- Using Web Services and SOAP, the request would look something like this:

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
          <soap:body pb="http://www.acme.com/phonebook">
                    <pb:GetUserDetails>
                              <pb:UserID>12345</pb:UserID>
                    </pb:GetUserDetails>
          </soap:Body>
</soap:Envelope>
```

REST Vs SOAP

- Simple web service as an example: querying a phonebook application for the details of a given user

- And with REST? The query will probably look like this:
  http://www.example.com/phonebook/UserDetails/12345

- GET /phonebook/UserDetails/12345 HTTP/1.1
  Host: www.example.com
  Accept: application/xml

- Complex query:
  http://www.example.com/phonebook/UserDetails?firstName=John&lastName=Doe

Why REST?

Web

- Roy Fielding and his doctoral thesis, “Architectural Styles and the Design of Network-based Software Architectures.”
- Why is the Web so prevalent and ubiquitous?
- What makes the Web scale?
- How can I apply the architecture of the Web to my own applications?

- The set of the architectural principles given by Roy Fielding to answer these questions - REpresentational State Transfer (REST)

REST - set of principles

- *Addressable resources*

  -Resource oriented, and each resource must be addressable via a URI

  -The format of a URI is standardized as follows:

  scheme://host:port/path?queryString#fragment

- *A uniform, constrained interface*

  -Uses a small set of well-defined methods to manipulate your resources.

  -The idea behind it is that you stick to the finite set of operations of the application protocol you're distributing your services upon.

REST - set of principles

- *Representation-oriented*

  -Interaction with services using representations of that service.

  -Different platforms, different formats - browsers -> HTML, JavaScript -> JSON and a Java application -> XML?

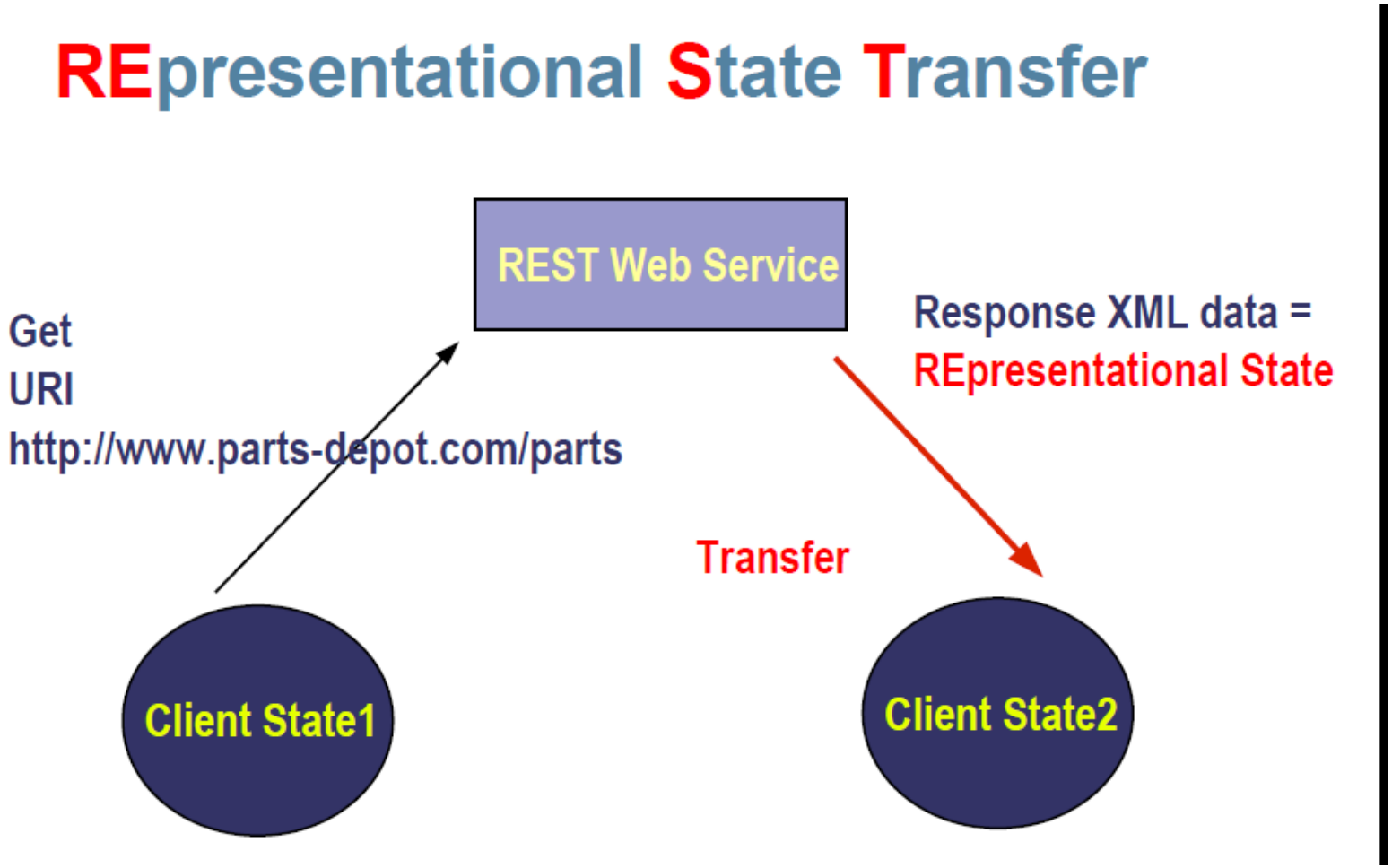
- *Communicate statelessly*

  -Stateless applications are easier to scale.

- *Hypermedia As The Engine Of Application State (HATEOAS)*

  -Let your data formats drive state transitions in your applications.

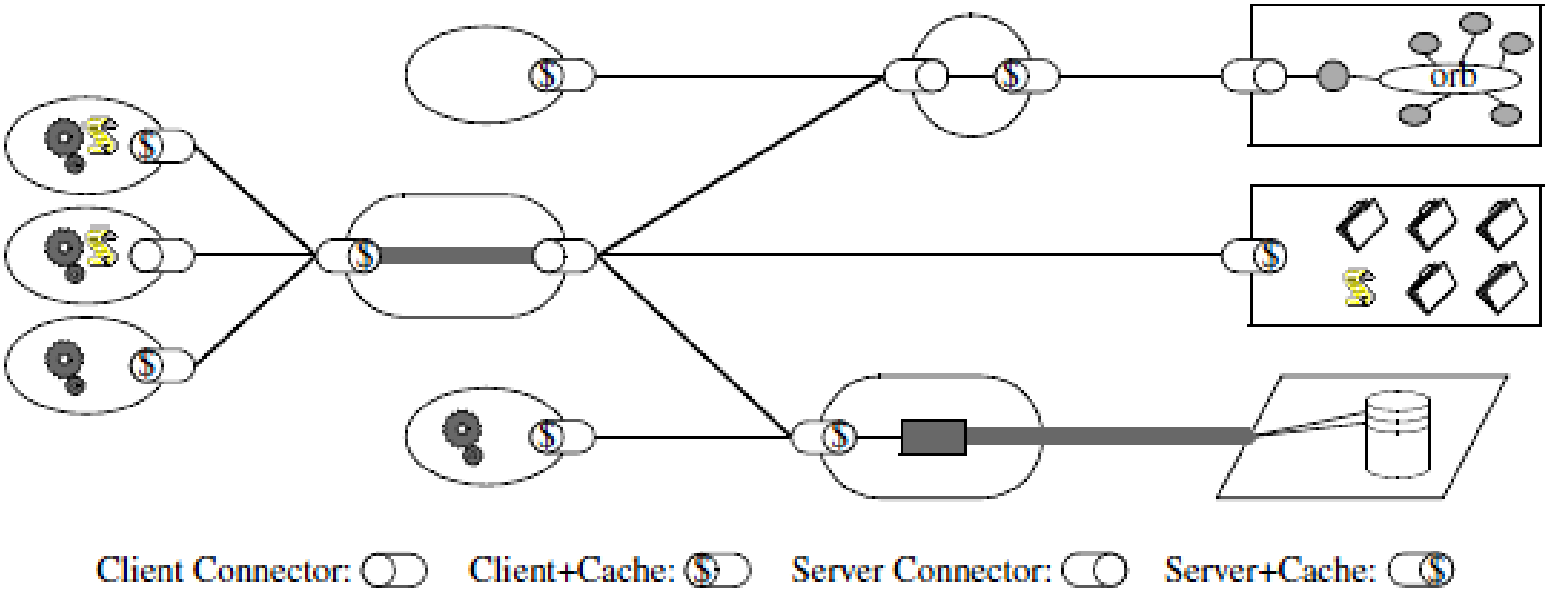
What is REST?


REpresentational State Transfer
REST Web Service
Get
URI
http://www.parts-depot.com/parts
Response XML data =
REpresentational State
Transfer
Client State1
Client State2

HTTP Request/Response As REST

http://www.javapassion.com/webservices/RESTPrimer.pdf

RESTful Architectural Style



Set of Constraints

- Client-Server: Separation of concerns
- Client-Stateless-Server: Visibility, Reliability, Scalability
- Caching : improves efficiency, scalability and user perceived performance, reduces average latency
- Uniform Interface: simplify overall system architecture and improved visibility of interactions
- Layered System: Simplifying components, Shared caching, Improved Scalability, Load balancing
- Code-On-Demand: Simplifies clients, Improves extensibility

WADL

WADL elements

- Application
- Grammar
- Include
- Resources
- Resource
- Resource Type
- Method
- Request
- Response
- Representation
- Param
- Link
- Doc

Why We need Web API

- If your app - your business's data model - has an API, then suddenly your Web API is opened up to native apps, iPhone apps, Windows 8 apps, whatever, apps. It's Web Services.
- You can use XML or JSON or something else with Your API. JSON is nice for mobile apps with slow connections.