REST VS "BIG" WEB SERVICES

SOAP

- Communication protocol with a header and an envelope
- The header contains metadata (QoS mainly)
- The envelope contains the data
- It is basically an RPC+XML mechanism

WSDL

- Web Service Definition Language
- Strictly define the service interface (method signature)
- Not mandatory for usage with SOAP
- But it’s very handy
- Decouples ws from underlying tech
- Clients can be generated automatically
- WSDL can be generated from code

```
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="uri:weblogscom">
<types>
  <s:schema targetNamespace="uri:weblogscom">
    <s:complexType name="pingResult">
      <s:sequence>
        <s:element maxOccurs="1" minOccurs="1" name="flerror" type="s:boolean"/>
        <s:element maxOccurs="1" minOccurs="1" name="message" type="s:string"/>
      </s:sequence>
    </s:complexType>
  </s:schema>
</types>
<message name="pingRequest">
  <part name="weblogname" type="s:string"/>
  <part name="weblogurl" type="s:string"/>
</message>
<message name="pingResponse">
  <part name="result" type="tns:pingResult"/>
</message>
<portType name="pingPort">
  <operation name="ping">
    <input message="tns:pingRequest"/>
    <output message="tns:pingResponse"/>
  </operation>
</portType>
<binding name="pingSoap" type="tns:pingPort">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="ping">
    <soap:operation soapAction="/weblogUpdates" style="rpc"/>
    <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="uri:weblogscom" use="encoded"/>
    </input>
    <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="uri:weblogscom" use="encoded"/>
    </output>
  </operation>    </binding>
<service name="weblogscom">
  <document>For a complete description of this service, go to the following
URL: http://www.soapware.org/weblogsCom </document>
  <port binding="tns:pingSoap" name="pingPort">
    <soap:address location="http://rpc.weblogs.com:80/"/>
  </port>
</service>
```

WSDL

- The same information conveyed with Java code

```
public class PingResponse {
  boolean error;
  String message;
}

public PingResponse ping(String weblogName, String weblogurl)
```

WSDL

- Every single change in the service must be reflected in its WSDL
- And clients must be recompiled against the new WSDL

UDDI

- Four-level hierarchy of federated, distributed yellow pages
- It was killed by complexity and by the its times
- Microsoft and IBM shut their own UDDIs in 2006
- Sharing economy was yet to come

Transport protocol

REST

- HTTP

WS-*

- Any protocol:
- TCP, SMTP, JMS etc.
- Synch and Asynch
- No assumptions on protocol
- QoS must be in the SOAP message

Payload format

REST

- Content negotiation (not necessarily successful)
  - JSON, XML, YAML
  - Any MIME type

WS-*

- SOAP

Service identification

REST

- URI

WS-*

- WS-Addressing

Service description

REST

- WADL (almost abandonware)
- Swagger is getting momentum
- Still no standard —> no client code generation

WS-*

- WSDL

Security

REST

- HTTPS:
  - Authentication
  - Authorization

WS-*

- WS-Security in SOAP header:
  - Authentication
  - Authorization
  - Message signing (non repudiation)
  - Identity federation (SAML)

Transactions (different providers)

REST

- None

WS-*

- WS-AtomicTransaction adopts 2PC

Reliable messaging

REST

- HTTP —> best effort

WS-*

- WS-ReliableMessaging defines XML elements for:
  - AtMostOnce, AtLeastOnce, ExactlyOnce and InOrder message delivery
  - Message sequence tracking
  - Retry logic

Service discovery

REST

- Google:
  - decentralized
  - not controllable

WS-*

- UDDI:
  - controllable at a very high cost
  - services should be kept healthy
  - failure for public
  - worked for industrial internal use

Summary

- The WS-* stack is by far the most complete
- It comes at a (big) price
- There are complex cases where this price must be paid

Bibliography

- RESTful Web Services vs. “Big” Web Services: Making the Right Architectural Decision -- Cesare Pautasso, Olaf Zimmermann, Frank Leymann

- RESTful web services — Leonard Richardson & Sam Ruby