Introduction

Web Services

Boniface Kabaso
Guest Lecturer
THE HAGUE UNIVERSITY OF APPLIED SCIENCES
Delft, Netherlands
13 December 2011

Agenda

- Distributed Computing History
- SOAP Based Web Services
- REST based Web Services

A look at 30 years of Software Development

- In the early 1980's, software was not yet distributed as in Distributed Systems,
- It was only "copied around"

- Programmers wanted to install software once on a server, and then call it remotely over a network from many clients
- And Remote Procedure Call (RPC) was born

A quest began in the 1980's

- Several major efforts were made to do RPC over the next 25 years
- All suffered from at least one of the usual wrong assumptions of distributed computing:
  - The network is secure
  - The network is homogeneous
  - The network is fast enough
- HTTP partially fulfilled the need in the early 1990's
  - Programmers could make HTTP GET requests in those days, but the language support for it was not great until recent years

Major client-server distributed computer paradigms

- 1980's:
- RPC using C/C++
- EDI with ASN.1
- Microsoft DCOM

- 1990's:
- CORBA (for Unix/Linux only)/RMI
- HTTP

By the late 1990's...

- still no way to distribute an application across multiple computers that was:
  - *standards-based*
  - *platform-independent*

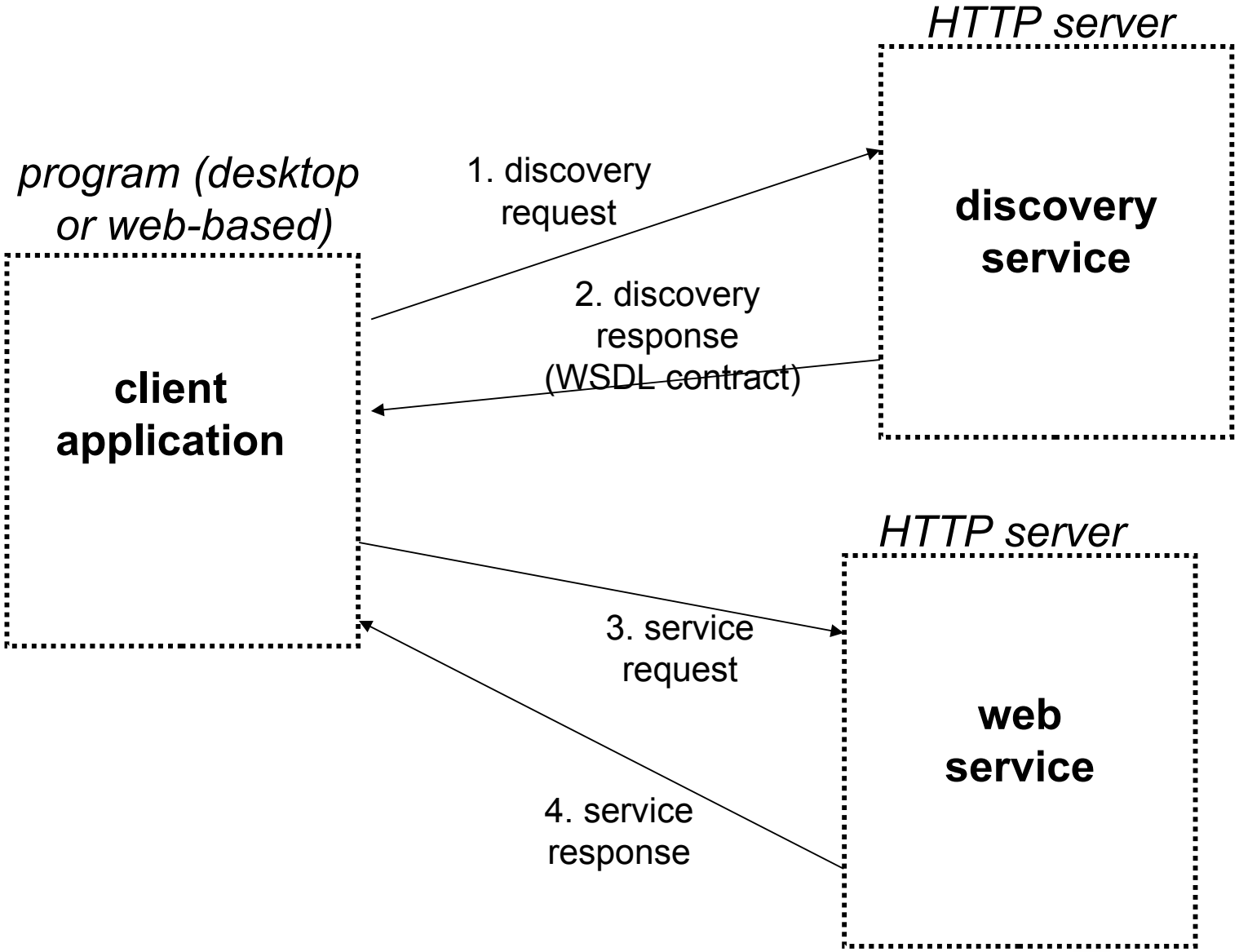
1998, birth of XML

- XML was standardized by the W3C in 1998
  - soon all the compilers started supporting XML
- Driven in part by Microsoft, but supported by many other companies, work began on a series of standards for XML-based RPC's
  - based on the idea of an existing open-source initiative called "XML RPC"
- A new idea was taking shape due to XML being standardized:
  - "web services" and Service Oriented Architecture (SOA)

SOA promises and vision for "web services"

- They would be standards-based, platform-independent, and immune to firewalls
  - some kind of XML would be the wire format
- Each service's contract would be expressed in a formal manner and registered in a catalog
  - programming languages could "parse" this contract and utilize it at runtime.
  - there would be a "factory" call that returned a reference to the currently preferred implementation of a given service contract
  - software architects would "compose" designs by shopping among available services
- Network and machine speed and capacity would increase to make the overhead of XML tolerable
  - massive software reuse would be achieved

Ideal web service scenario



Web service standards were ready by about 2000

1. For automatic discovery
   - ***Universal Description, Discovery and Integration (UDDI)***

1. For the contract
   - ***Web Service Definition Language (WSDL)***
   - an XML-based schema for web service contracts: how to call a service, but not how it's implemented

1. For message *serialization*
   - ***SOAP***
   - client/server protocol for exchanging messages over computer networks, with HTTP/HTTPS (POST) as the transport; it passes parameters in XML and return values in XML, and has a standardized error mechanism

Web services gradually became viable

- 2004: Web Services Interoperability Organization (WS-I) issued a SOAP test tool suite
- http://www.ws-i.org
- 2005: Microsoft made calling a web service simple with C# 2.0 and .NET 2
- Programmer providse a web service URL and calls it (via a local proxy object) in a try-catch block; the proxy hides all of the SOAP messaging and error handling
- 2006-8: Ruby and many other languages provided proxy-automated web service calling
- In 2009, Java provided proxy-automated web service calling
- Of course, web services could always be done without an *automated* proxy, but in practice, this was too much trouble and few did that

The Rebellion

- Many companies, including Yahoo! and Google, initially provided SOAP-based API's
  - Windows programmers used these a lot due to the automation included in .NET 2.0
- Then, a massive anti-SOAP campaign arose
  - Led by Roy Fielding (then employed by SUN)
    - REpresentational State Transfer (REST)
      - Really HTTP GET calls with incoming parameters in the URL
    - Claim: RPC calls not needed, even "harmful", too inefficient, and no bookmark for the data object that is the topic of the call
    - During the fray, Amazon, Yahoo! and Google cancelled all their SOAP API's and switched back to API's using HTTP GET
  - The new movement was referred to as "RESTafarians", due to their anti-SOAP rants which had the passion of a cult, and which were possibly motivated by a competitive spirit with Microsoft (which had strongly supported SOAP)
    - They touted early interoperability problems as a show-stopper

Introduction to Web Services Protocols

Communication and standards

- Efficient (or indeed any) communication is dependent on a shared vocabulary and grammar.
- Because web services deals with inter-organisation communication these must be universal standards.

Underlying standards

- The basic standards for web services are:
- XML (Extensible Markup Language)
- SOAP (simple object access protocol)
- WSDL (web services description language)
- UDDI (universal description, discovery and integration)

The state of standards

- XML 1.0 fairly stable, although Schema are in the process of replacing DTDs (currently Schema 1.1 being worked on).
- SOAP 1.2
- WSDL 2.0 (coming out, 1.2 current)
- UDDI version 3 (Aug 2003)
- BPEL 1.1 (Business Process Execution Language)
- choreography description language (web services work flows)
  started January 2003.

Standards are still volatile and in the process of development.

Web Services Architecture

- Web Services involve three major roles
  - Service Provider
  - Service Registry
  - Service Consumer
- Three major operations surround web services
  - Publishing – making a service available
  - Finding – locating web services
  - Binding – using web services

Making a service available (1)

In order for someone to use your service they have to know about it.

- To allow users to discover a service it is published to a registry (UDDI).
- To allow users to interact with a service you must publish a description of it's interface (methods & arguments).
- This is done using WSDL.

Making a service available (2)

- Once you have published a description of your service you must have a host set up to serve it.
- A web server is often used to deliver services (although custom application – application communication is also possible).
- This is functionality which has to be added to the web server. In the case of the apache web server a ‘container’ application (Tomcat) can be used to make the application (servlet) available to apache (deploying).

The old transfer protocols are still there.

- Like the grid architecture web services is layered on top of existing, mature transfer protocols.
- HTTP, SMTP are still used over TCP/IP to pass the messages.
- Web services, like grids, can be seen as a functionality enhancement to the existing technologies.

XML

- All Web Services documents are written in XML
- XML Schema are used to define the elements used in Web Services communication

SOAP

- Actually used to communicate with the Web Service
- Both the request and the response are SOAP messages
- The body of the message (whose grammar is defined by the WSDL) is contained within a SOAP “envelope”
- “Binds” the client to the web service

WSDL

- Describes the Web Service and defines the functions that are exposed in the Web Service
- Defines the XML grammar to be used in the messages
  - Uses the W3C Schema language

UDDI

- UDDI is used to register and look up services with a central registry
- Service Providers can publish information about their business and the services that they offer
- Service consumers can look up services that are available by
  - Business
  - Service category
  - Specific service

XML

What is XML

- XML stands for extensible markup language
- It is a hierarchical data description language
- It is a sub set of SGML a general document markup language designed for the American millitary.
- It is defined by w3c.

Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
xmlns="document" >
<xs:element name = "DOCUMENT">
        <xs:element name="CUSTOMER"> </xs:element>
</xs:element>
</xs:schema>
```

Simple schema saved as order.xsd

```
<?xml version="1.0"?>
<DOCUMENT xmlns="document"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
Xsi:schemaLocation="order.xsd">
<DOCUMENT>
        <CUSTOMER>sam smith</CUSTOMER>
        <CUSTOMER>sam smith</CUSTOMER>
</DOCUMENT>
```

XML document derived from schema.

SOAP

Request Response Web Services

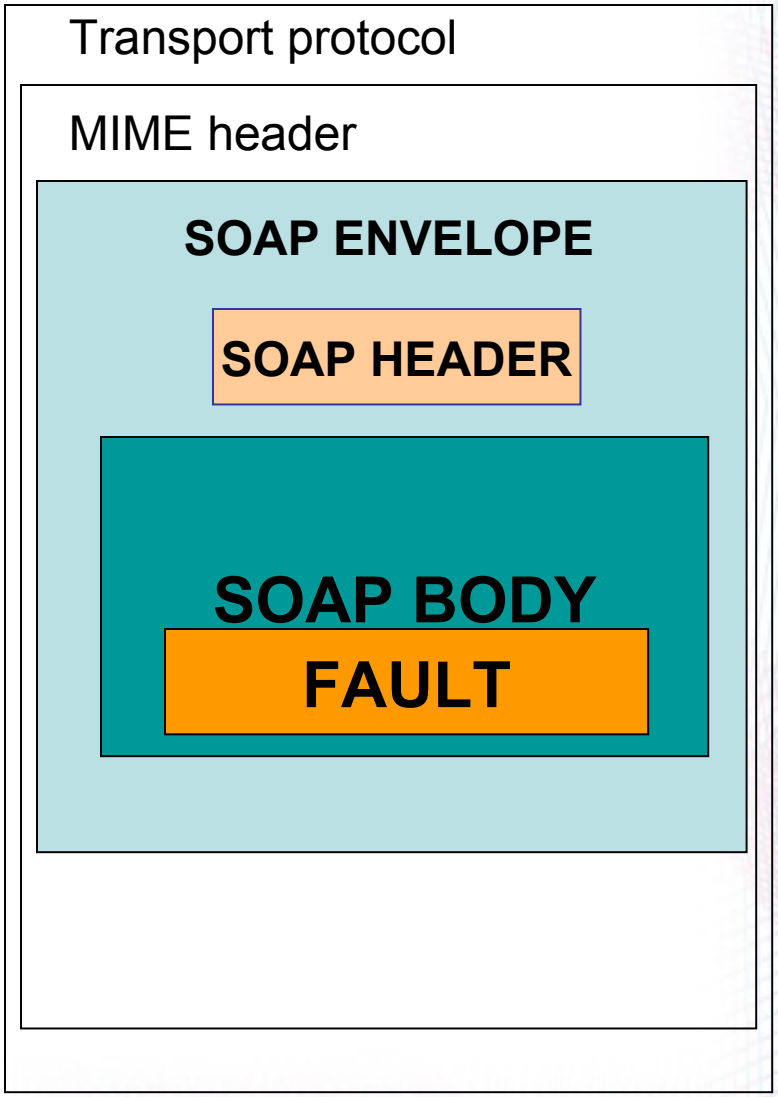
- Currently the most common implementation of Web Services
- Work in a very simple ‘request – response’ paradigm
- For Example:
  - A Weather Service– simple request for weather in an area, simple response with the weather report
  - An Airline special offers service – travel agents would simply make requests for latest offers and would receive the offers as a response

SOAP messages

- SOAP provides a standard ‘envelope’ within which a message can be delivered.
- SOAP is mechanism (protocol) for transferring information (messages) between applications which may be widely distributed.
- SOAP says nothing about the content of the message – the sender and the receiver must understand the message for themselves.
- SOAP is part of a communication stack.

SOAP Structure(1)

- Each SOAP message will have:
  - An `Envelope`
  - A `Header` (optional)
  - A `Body`
  - The `Body` may contain a `Fault` element



SOAP Structure(2)

- The envelope wraps the entire soap document
- The header contains allows additional information to be passed as well as the body of the document – e.g. authentication
- The body element contains the core of the SOAP document – this will contain either the RPC call or the XML message itself
- The fault information will contain any exception information

Anatomy of a SOAP message

```
<?xml version='1.0' encoding='UTF-8'?>


<SOAP-ENV:Envelope xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3c.org/1999/XMLSchema">
        <SOAP-ENV:Header>

        </SOAP-ENV:Header


        <SOAP_ENV:Body>


        </SOAP-ENV:Body>


</SOAP-ENV:Envelope>
```

SOAP protocol binding

```
SOAPAction = "urn:soaphttpclient-action-uri"
Host = localhost
Content-Type = text/xml; charset=utf-8
Content-Length = 701
```

```
<SOAP-ENV:Envelope xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3c.org/1999/XMLSchema">




</SOAP-ENV:Envelope>
```

SOAP RPC

- SOAP RPC messages contain XML that represents a method call or method response
- The SOAP XML will be converted into a method call on the server and the response will be encoded into SOAP XML to be returned to the client

SOAP Faults

- SOAP errors are handled using a specialised envelope known as a Fault Envelope
- A SOAP Fault is a special element which must appear as an immediate child of the body element
- <faultcode> and <faultstring> are required.

A SOAP fault

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
   xmlns:xsd="http://www.w3c.org/1999/XMLSchema">
   <SOAP_ENV:Body>
```

<SOAP-ENV:Fault>
<faultcode>SOAP-ENV:Server</faultcode>
<faultstring>Test fault</faultstring>
<faultactor>/soap/servlet/rpcrouter</faultactor>
<detail>

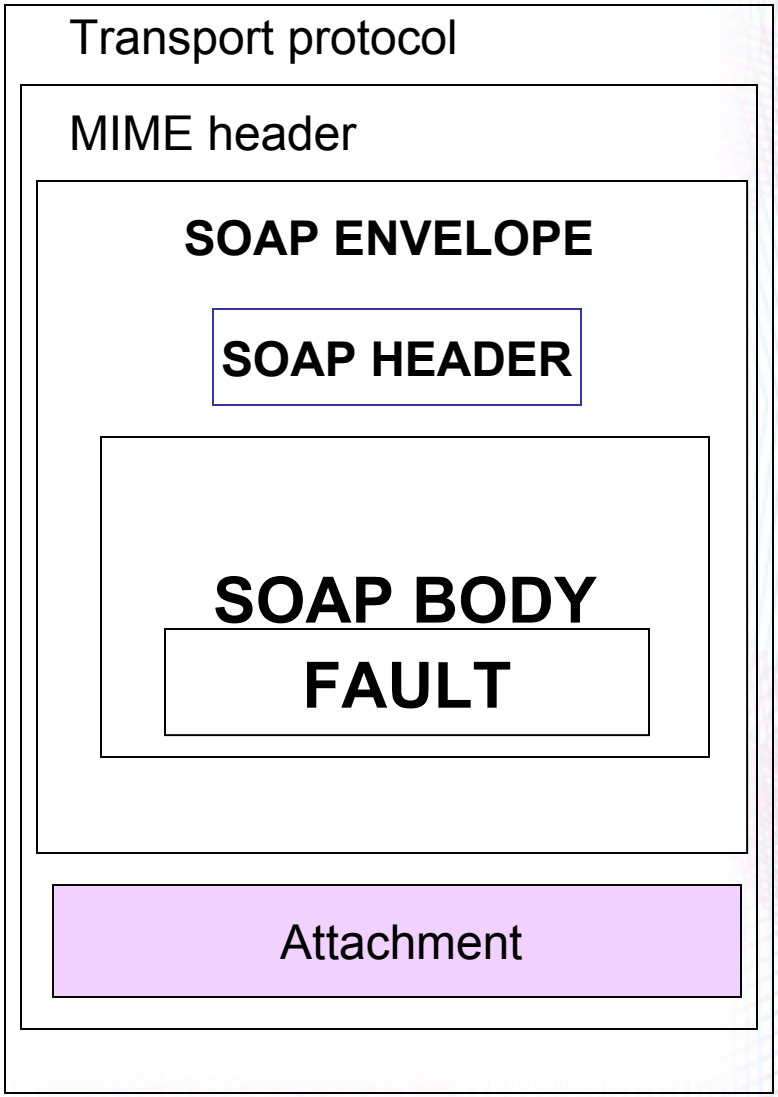
..

</detail>
</SOAP-ENV:Fault>

```
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Attachment

- Large quantities or binary data may not fit well into a XML SOAP message.
- In which case it can be sent ‘out of band’ by attaching it to a SOAP message
- *Analogy : email attachments.*



Attaching a file to a SOAP message

- To add a file to a SOAP message a tag is added within the body of the message.

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
      xmlns:xsd="http://www.w3c.org/1999/XMLSchema">
      <SOAP_ENV:Body>

            <attachment href="{URL}"/>

      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The balanced view

- XML-RPC and SOAP web services *are* prone to inefficiency
  - due to the verboseness of XML as compared with more concise wire formats
- XML-RPC and SOAP web services *are* capable of being automated ("easy" to use) and platform independent
  - This has true value at times, especially for quick prototyping
- Add-ons to the web services standards have now made it possible to send binary files in the data (multimedia), to authenticate users, and for transaction processing (multiple actions with roll-back if any fail)

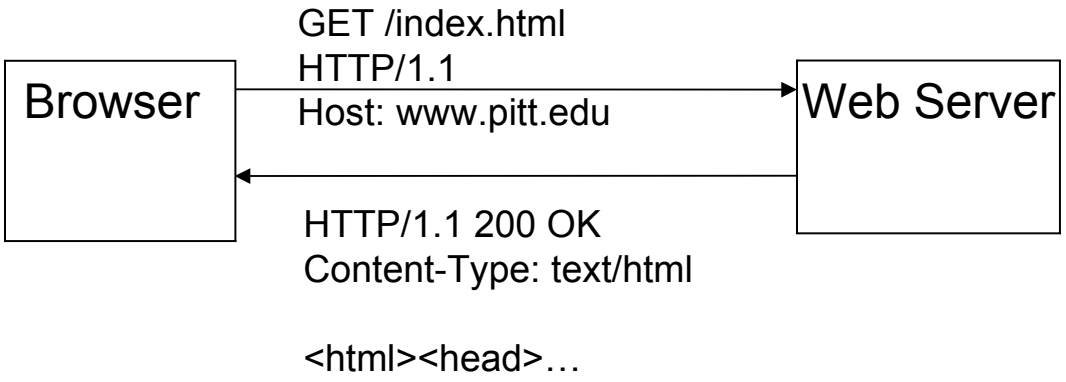
SOAP strengths and weaknesses

- Good: Platform independent (well supported by most programming languages)
  - And easy because of WSDL automation by proxy (both client and service easy to write)
- Good: SOAP protocol provides standardized error handling
- Good: Human-readable data is self-describing
- Good: Immune to firewalling (text over HTTP)
- Bad: Not designed for speed
  - will not scale to enterprise speeds for really large object passing, because of its verbosity (larger messages take longer create, transmit and parse)
- Bad: if automation fails, you have to look at SOAP's dirty details

Representational State Transfer (REST): Representing Information

Hypertext Transfer Protocol (HTTP)

- A communications protocol
- Allows retrieving inter-linked text documents (hypertext)
  - World Wide Web.
- HTTP Verbs
  - HEAD
  - **GET**
  - **POST**
  - PUT
  - DELETE
  - TRACE
  - OPTIONS
  - CONNECT



Representational State Transfer (REST)

- A style of software architecture for distributed hypermedia systems such as the World Wide Web.
- Introduced in the doctoral dissertation of Roy Fielding
  - One of the principal authors of the HTTP specification.
- A collection of network architecture principles which outline how resources are defined and addressed

REST and HTTP

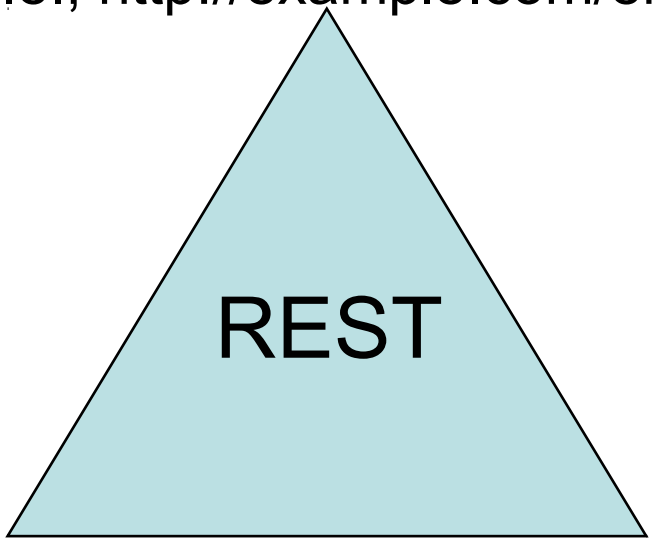
- The motivation for REST was to capture the characteristics of the Web which made the Web successful.
  - URI Addressable resources
  - HTTP Protocol
  - Make a Request – Receive Response – Display Response
- Exploits the use of the HTTP protocol beyond HTTP POST and HTTP GET
  - HTTP PUT, HTTP DELETE

REST - not a Standard

- REST is not a standard
  - JSR 311: JAX-RS: The Java™ API for RESTful Web Services
- But it uses several standards:
  - HTTP
  - URL
  - XML/HTML/GIF/JPEG/etc (Resource Representations)
  - text/xml, text/html, image/gif, image/jpeg, etc (Resource Types, MIME Types)

Main Concepts

Nouns (Resources)
unconstrained
i.e., http://example.com/employees/12345



Verbs
constrained
i.e., GET

Representations
constrained
i.e., XML

Resources

- The key abstraction of information in REST is a resource.
- A resource is a conceptual mapping to a set of entities
  - Any information that can be named can be a resource: a document or image, a temporal service (e.g. "today's weather in Los Angeles"), a collection of other resources, a non-virtual object (e.g. a person), and so on
- Represented with a global identifier (URI in HTTP)

Naming Resources

- REST uses URI to identify resources
  - http://localhost/books/
  - http://localhost/books/ISBN-0011
  - http://localhost/books/ISBN-0011/authors
  - http://localhost/classes
  - http://localhost/classes/cs2650
  - http://localhost/classes/cs2650/students
- As you traverse the path from more generic to more specific, you are navigating the data

Verbs

- Represent the actions to be performed on resources

- HTTP GET
- HTTP POST
- HTTP PUT
- HTTP DELETE

HTTP GET

- How clients ask for the information they seek.
- Issuing a GET request transfers the data from the server to the client in some representation
- GET http://localhost/books
  - Retrieve all books
- GET http://localhost/books/ISBN-0011021
  - Retrieve book identified with ISBN-0011021
- GET http://localhost/books/ISBN-0011021/authors
  - Retrieve authors for book identified with ISBN-0011021

HTTP PUT, HTTP POST

- HTTP POST creates a resource
- HTTP PUT updates a resource

- POST http://localhost/books/
  - Content: {title, authors[], …}
  - Creates a new book with given properties

- PUT http://localhost/books/isbn-111
  - Content: {isbn, title, authors[], …}
  - Updates book identified by isbn-111 with submitted properties

HTTP DELETE

- Removes the resource identified by the URI
- DELETE http://localhost/books/ISBN-0011
  - Delete book identified by ISBN-0011

Representations

- How data is represented or returned to the client for presentation.
- Two main formats:
  - JavaScript Object Notation (JSON)
  - XML
- It is common to have multiple representations of the same data

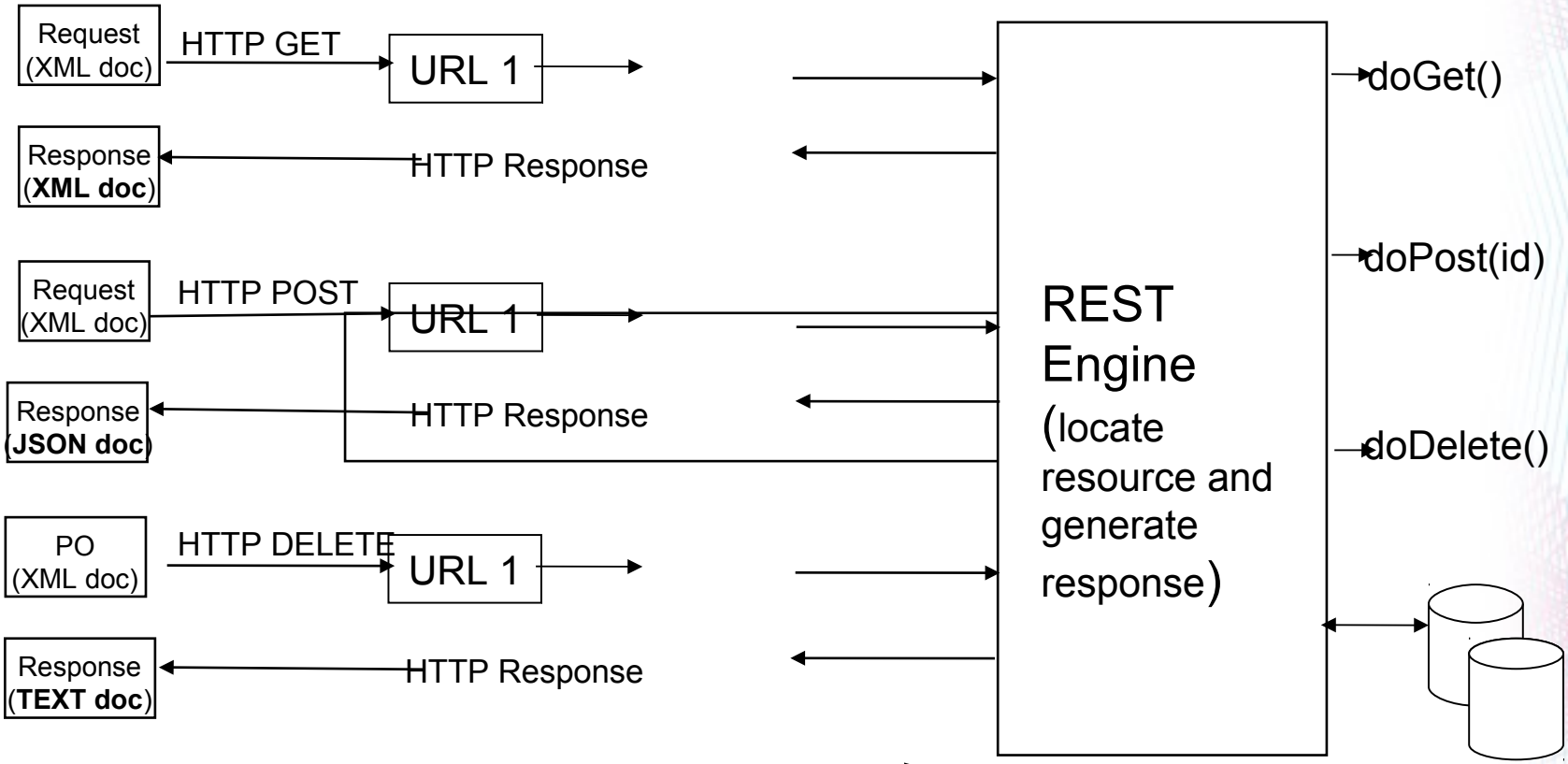
Representations

- XML
  - <COURSE>
    - <ID>CS2650</ID>
    - <NAME>Distributed Multimedia Software</NAME>
  - </COURSE>
- JSON
  - {course
    - {id: CS2650}
    - {name: Distributed Multimedia Sofware}
  - }

Why is it called "Representational State Transfer"?

The Client references a Web resource using a URL. A **representation** of the resource is returned (in this case as an HTML document).
The representation (e.g., Boeing747.html) places the client application in a **state**.
The result of the client traversing a hyperlink in Boeing747.html is another resource accessed. The new representation places the client application into yet another state. Thus, the client application changes (**transfer**s) state with each resource representation --> Representation State Transfer!

Architecture Style



Real Life Examples

- Google Maps
- Google AJAX Search API
- Yahoo Search API
- Amazon WebServices

REST and the Web

- The Web is an example of a REST system!
- All of those Web services that you have been using all these many years - book ordering services, search services, online dictionary services, etc - are REST-based Web services.
- Alas, you have been using REST, building REST services and you didn't even know it.

REST Implementations

- Restlet
  - http://www.restlet.org/
- Project Zero
  - http://www.projectzero.org
- GlassFish Jersey
  - https://jersey.dev.java.net/
- JBoss RESTeasy
  - http://www.jboss.org/resteasy/

References

- Representational State Transfer http://en.wikipedia.org/wiki/Representational_State_Transfer

- Roy Fieldings Thesis

  http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm