# SOAP and REST Web Services - Theory


IT Learning &
Outsourcing Center

Lector: Dimitar Topuzov

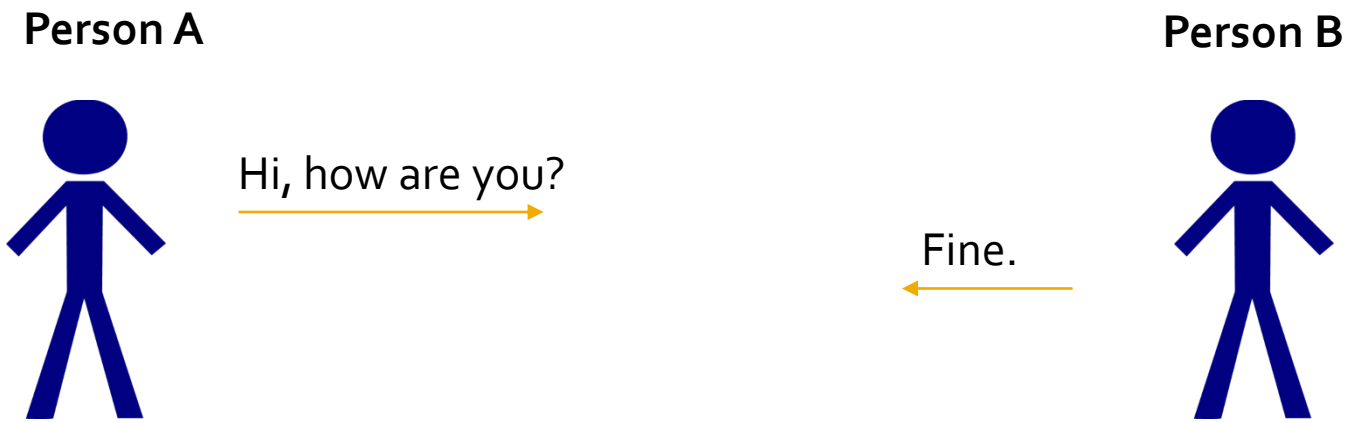
www.pragmatic.bg

E-mail: dtopuzov@gmail.com
LinkedIn: http://bg.linkedin.com/pub/dimitar-topuzov/18/470/833/en

# Content

- Different Types of Web Services
- SOAP Services – Theory
  - SOAP (Simple Object Access Protocol)
  - WSDL (Web Service Description Language)
  - UDDI (Universal Description Discovery and Integration)
- RESTful Services – Theory
  - Resource
  - Representation
  - Actions
- SOAP vs. REST

# Web Service Types




- Two important things
  - Media / Transport (Phone)
  - Message Format (English grammar)

# Web Service Types

- Based on messaging format and transport we can distinguish two types of Web Services
  - SOAP
  - REST

# SOAP Services

- Message Format
  - XML
  - SOAP strictly defines message format in SOAP protocol
- Transport
  - HTTP, FTP, UDP

# REST Services




- Message Format
  - XML, JSON, YAML, HTML, plain text
  - Message can be anything we can transfer over the network
  - Data is send as it is (not enveloped)
- Transport
  - HTTP

# What is SOAP?

- SOAP Services are based on SOAP protocol
- SOAP stands for Simple Object Access Protocol
- SOAP is
  - A communication protocol
  - A format for sending messages
  - Based on XML
  - Platform independent
  - Language independent
  - Simple and extensible
  - Will be developed as a W3C standard

# SOAP Services

- In order to access the server client should know two things
  - Location of the service (where the service is)
  - Description of the service (what this service provides and how it works)
    - Description is XML file know as WSDL
    - WSDL stands for “Web Services Description Language”

# SOAP Services

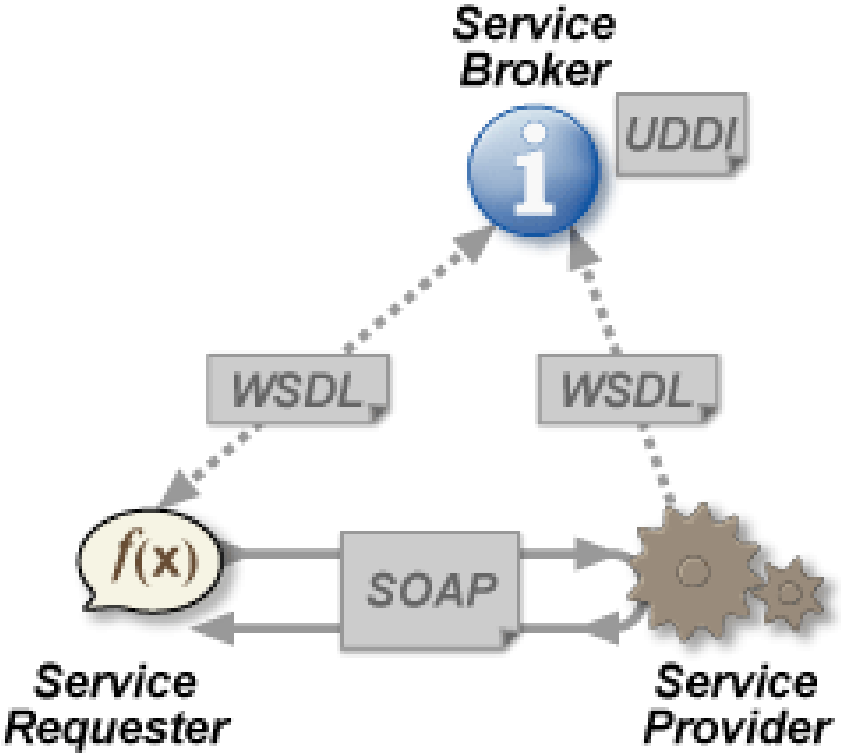
- How client locate the server
  - Server knows the client and send him WSDL
  - UDDI
    - Place where service providers register their services
    - UDDI stands for “Universal Description, Discovery and Integration”

# SOAP Services

- UDDI communication workflow
  - Client search UDDI
  - UDDI returns all services providing searched service
  - Client choses a service and get its WSDL

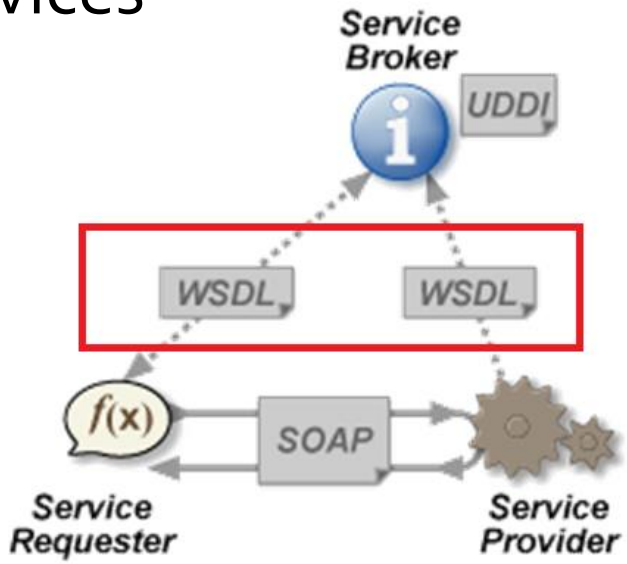
# SOAP Services

- SOAP Services communication in picture

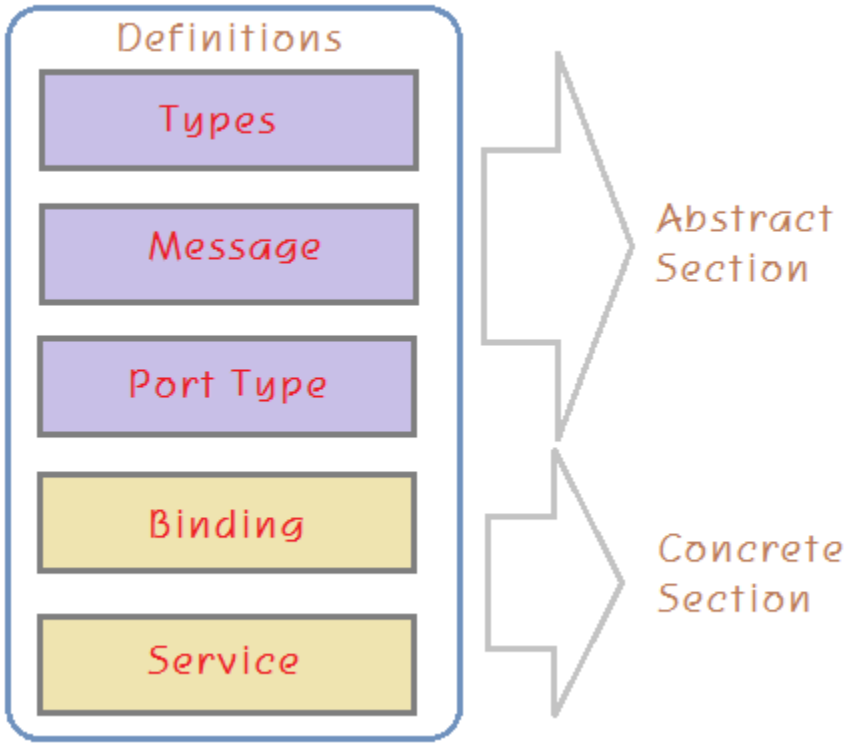


# What is WSDL?

- WSDL is an XML-based language
- WSDL document is actually XML document
- WSDL is used to describe Web services
- WSDL is also used to locate Web services
- WSDL is W3C recommendation



# WSDL Structure




- Abstract Section
  - What messages service use
  - What operations service can perform
- Concrete Section
  - Where service is located
  - How we can access the service

# WSDL Abstract Section



- Types
  - Defines the data type definitions for messages that will be exchanged by the web service.
- Message
  - Defines the set of actual messages that will be exchanged.
- PortType
  - Defines the operations provided/available and involved messages.
  - Operation refers to the messages involved in the transaction.

# WSDL Concrete Section



- Bindings
  - Defines transport protocol
  - Defines the message format for operations defined by the portType.
- Service
  - Defines the endpoint where the web service will be exposed

# WSDL Example

- Examples
  - Example 1
  - Example 2

# SOAP Messages

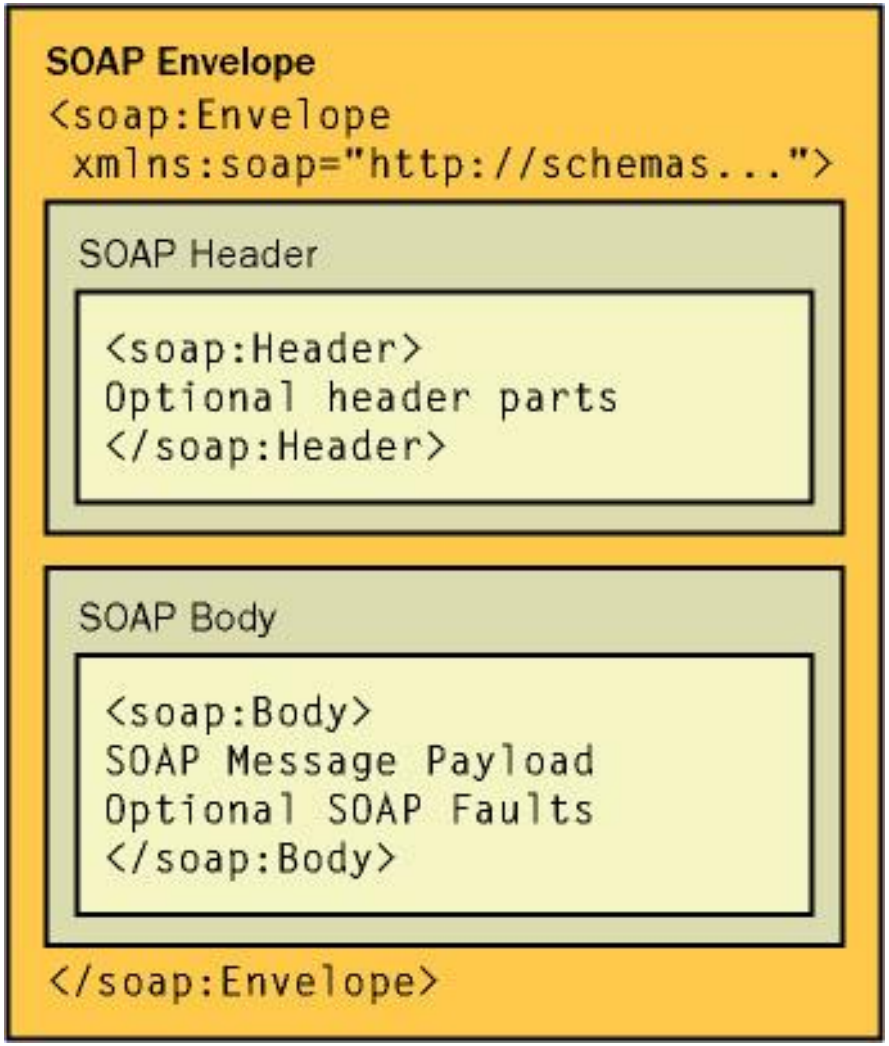
- Client and server communicate with SOAP messages

# SOAP Messages

- Plain SOAP Message Diagram

# SOAP Messages

- SOAP Message contains following main elements:
  - Envelope (mandatory)
  - Header (optional)
  - Body (mandatory)
- Both SOAP Requests and Responses use Envelope



# SOAP Message

```
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <soap:Header>
        [Here's where extraneous information, like password data, resides]
    </soap:Header>
    <soap:Body>
        [Here's where the actual message content resides]
        <soap:Fault>
            [Here are instructions to the server about how to handle errors]
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

# SOAP Envelope

- SOAP Envelope encapsulates the entire message
- SOAP Envelope is the root element of a SOAP message
- This element defines the XML document as a SOAP message.
- SOAP Envelope contains two child elements, an optional <Header> and a mandatory <Body>

# SOAP Header

- SOAP Header is optional
- Must be the first child element of the Envelope
- Header elements can occur multiple times
- Header includes information that might be needed by the receiver but isn't strictly part of the message content, like
  - login and password information
  - digital signatures
  - maximum time the SOAP request may take to process
  - state

# SOAP Header Attributes

- Attributes appear in Header elements
- Determine how a recipient processes a message
- The SOAP 1.1 specification defines two attributes that can appear in SOAP Header Element:
  - actor
  - mustUnderstand
- The SOAP 1.2 specification defines three attributes:
  - role (a new name for actor)
  - mustUnderstand
  - relay

# Role/Actor Attribute

- Header blocks (elements) can be targeted at nodes acting in specific roles
- If a header block is targeted for nodes acting in the "ultimateReceiver" role, then only nodes acting as ultimate receivers must process that header block. All other nodes should leave it unprocessed.

# mustUnderstand Attribute



- Indicate whether a header entry is mandatory or optional
- Has two valid values
  - True
    - Means that any node (computer) processing the SOAP message must understand the given header block
    - If intermediate node does not understand the header block (element) containing the mustUnderstand attribute, it must return a SOAP fault.
  - False
    - Means that node might not understand given header block

# Relay Attribute

- Determines if a header block is allowed to be relayed if not processed
- Has two valid values
  - True
    - Header element can be forwarded even if not processed
  - False (Default)
    - Header element should be removed if the message is forwarded

# SOAP Body

- The SOAP Body element is mandatory
- The SOAP Body element contains the actual SOAP message
- Sample request:

```
<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>
```

- Sample response:

```
<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>
```

# SOAP Fault

- SOAP Fault element is optional
- Must appear as a child element of the Body element
- Fault element can only appear once in a SOAP
- Holds errors and status information for a SOAP message

# SOAP Fault

- SOAP Fault element has the following sub elements

**<faultcode>**

A code for identifying the fault

**<faultstring>**

A human readable explanation of the fault

**<faultactor>**

Information about who caused the fault to happen

**<detail>**

Holds application specific error information related to the Body element

# SOAP Faultcode Values



- SOAP Fault code values

**<VersionMismatch>**

Found an invalid namespace for the SOAP Envelope element

**<MustUnderstand>**

Child element of the Header element, with the mustUnderstand attribute set to "1", was not understood

**<Client>**

The message was incorrectly formed or contained incorrect information

**<Server>**

There was a problem with the server so the message could not proceed

# SOAP Example




- Examples
  - Example Request
  - Example Response

# What is REST?

- REST stands for “Representational State Transfer”
- Definition
  - Representational State Transfer (REST) is a software architecture style consisting of guidelines and best practices for creating scalable web services.
- REST is not a standard!

# REST Concepts

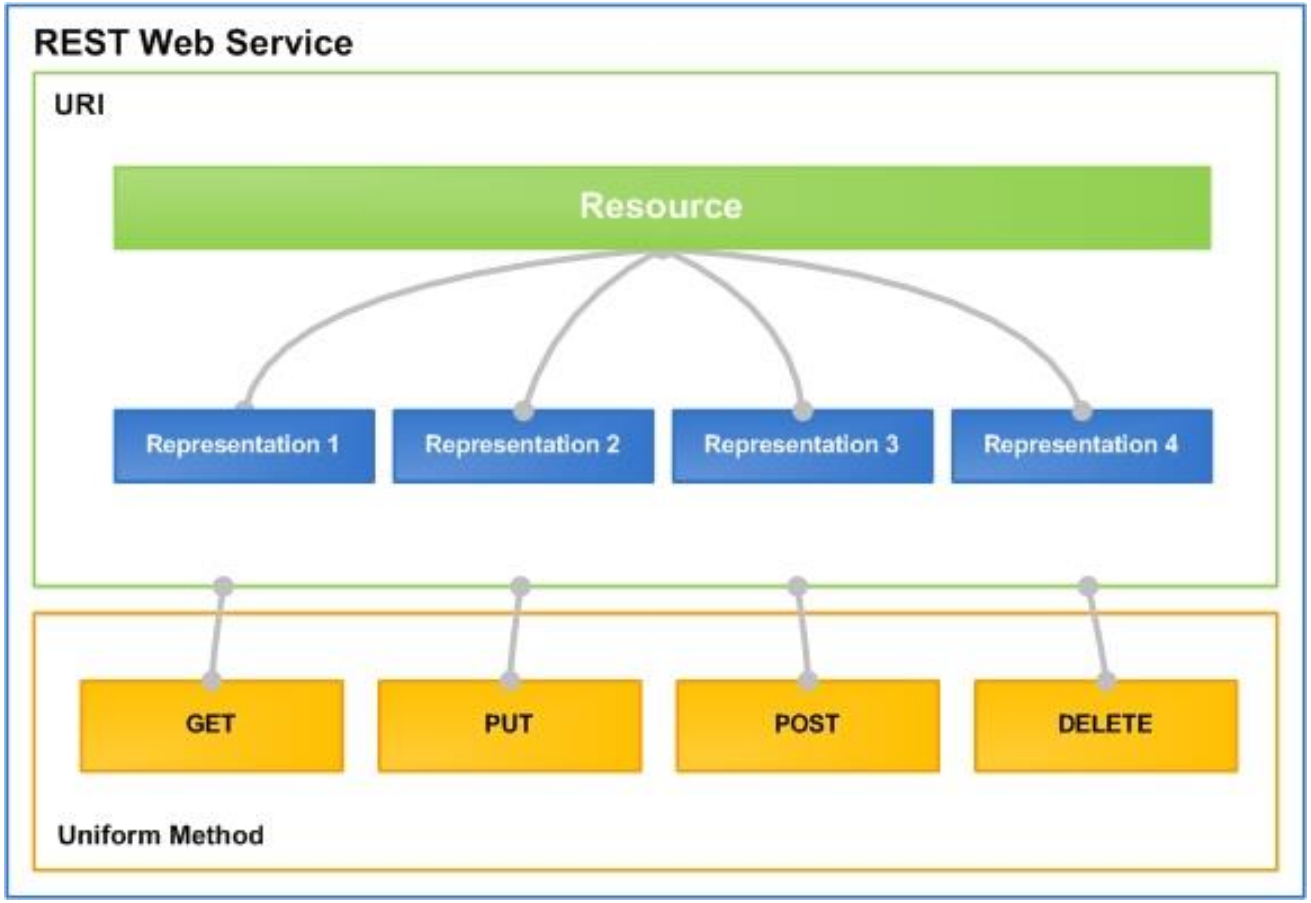


- A RESTful system should be
  - Client-server
  - Stateless
    - Each request should be independent of others
  - Cacheable
    - Clients are able to cache responses
    - Responses must therefore, implicitly or explicitly, define themselves as cacheable, or not
  - Uniformly accessible
    - Each resource must have a unique address and a valid point of access

# A RESTful System Main Actors



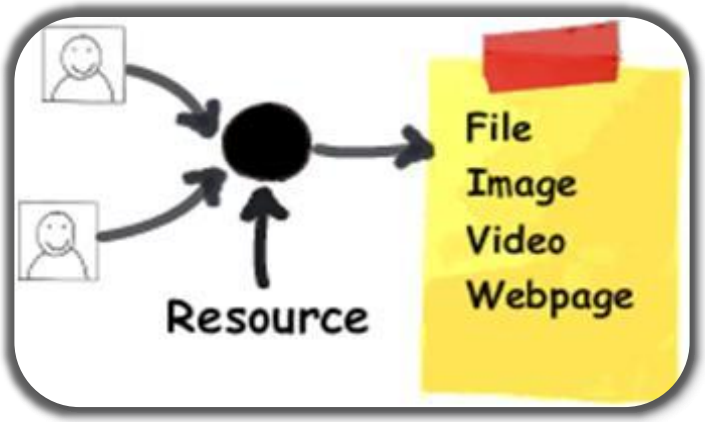
- Resources, Representations and Actions



# Resources

- A resources is "everything" the service can provide
- State and functions of a remote application are also considered as resources
- A resource must have a unique address over the Web

- Example of resources:
  - Title of a movie from IMDb
  - YouTube video
  - Images from Flicker
  - Order info from eBay



# Representations

- The representations of resources is what is sent back and forth clients and servers
- We never send or receive resources, only their representations

# Representation Formats

- Different clients are able to consume different representations of the same resource
- A representation can take various forms, but its resource has to be available through the same URI
- The format of the representations is determined by the content-type
  - Content type is a reusable collection of settings that you want to apply to a certain category of content

```
Accept: application/json, text/javascript, */*
Content-Type: application/json; charset=utf-8
Accept-Encoding: gzip, deflate
```

# XML Format

- XML is markup-language for encoding documents in machine-readable form
  - Text-based format
  - Consists of tags, attributes and content
  - Provide data and meta-data in the same time

```
<?xml version="1.0"?>
<library>
  <book><title>HTML 5</title><author>Bay Ivan</author></book>
  <book><title>WPF 4</title><author>Microsoft</author></book>
  <book><title>WCF 4</title><author>Kaka Mara</author></book>
  <book><title>UML 2.0</title><author>Bay Ali</author></book>
</library>
```

# JSON Format



- JSON (JavaScript Object Notation)
  - Standard for representing simple data structures and associative arrays
  - Lightweight text-based open standard
  - Derived from the JavaScript language

```
{
  "firstName": "John", "lastName": "Smith", "age": 25,
  "phoneNumber": [{ "type": "home", "number": "212 555-1234"},
    { "type": "fax", "number": "646 555-4567" }]
},
{
"firstName": "Bay", "lastName": "Ivan", "age": 79
}
```

# JSON vs. XML



**XML**

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

**JSON**

```
{"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]}
```

# Actions

- Actions are used to operate on resources
- For example they can be used for
  - Getting info about a movie
  - Adding photo to Flicker
  - Deleting a post from Facebook
  - Updating Facebook status

# HTTP Based Actions




- Under HTTP, actions are standard HTTP request
  - GET – retrieve a resource
  - POST – create a resource
  - PUT – update a resource
  - DELETE – delete a resource

# HTTP Status Codes



- REST services use HTTP status codes to return information about the response

| | |
|---|---|
| 1xx | informational message |
| 2xx | success message |
| 3xx | redirects the client to another URL |
| 4xx | client-side error |
| 5xx | server-side error |

# SOAP vs REST

- REST
  - Exposes RESOURCES which represent DATA
  - Use HTTP Verbs (GET/POST/PUT/DELETE)
  - Supports multiple data formats
- SOAP
  - Exposes OPERATIONS which represent LOGIC
  - Use HTTP POST
  - Supports only XML (and attachments)

# SOAP vs REST



**Consider "Martin Lawrence" as your data**

## SOAP

Client

+

=

<- ->

Server

Data + SOAP Standards

Your data sent to SERVER would become huge as Big mama

## REST

Client

<- ->

Rest is like sending the DATA as such

Server

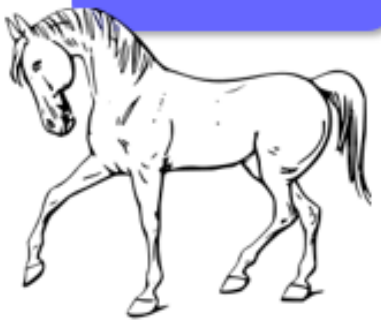
# SOAP vs REST



REST

Rides directly on HTTP. Plain and simple. In reality, this is all you need to send data from point A to point B and get the required response. Catch: Until something that represents a service contract is put in to place, it's kinda "anything goes".

HTTP



SOAP

The coach is your SOAP envelope: it wraps your data. Main strength is the presence of a contract: the WSDL. Gives you the "comfort" of easily generating artifacts. Catch: look at the complexity and added weight.

# Who Use REST?




# Who Prefer SOAP?

- Big old companies
- Mission critical software systems

# Additional Resources




- SOAP Specification
  - http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
- REST Articles
  - http://en.wikipedia.org/wiki/Representational_state_transfer
  - http://www.ibm.com/developerworks/library/ws-restful/

# Questions