API

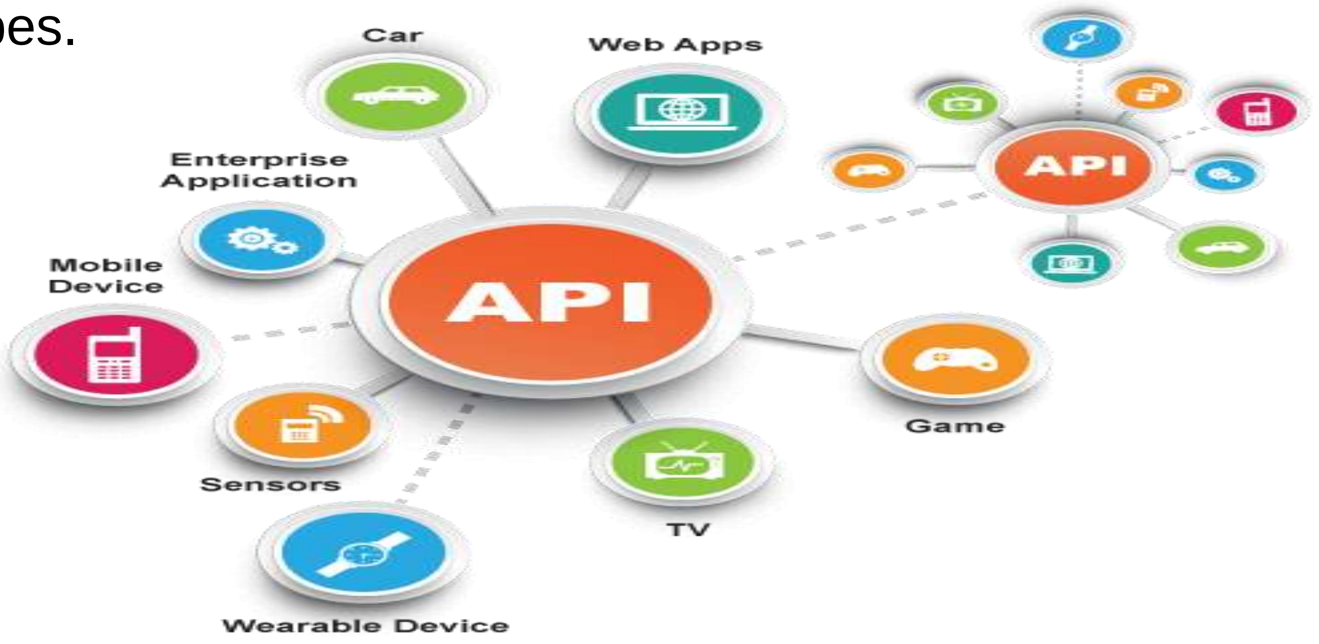
APPLICATION PROGRAMMING INTERFACE

Prepared By:

A.Rajamanikam

WHAT IS AN API?

In computer programming, an ***application programming interface (API)*** is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types.



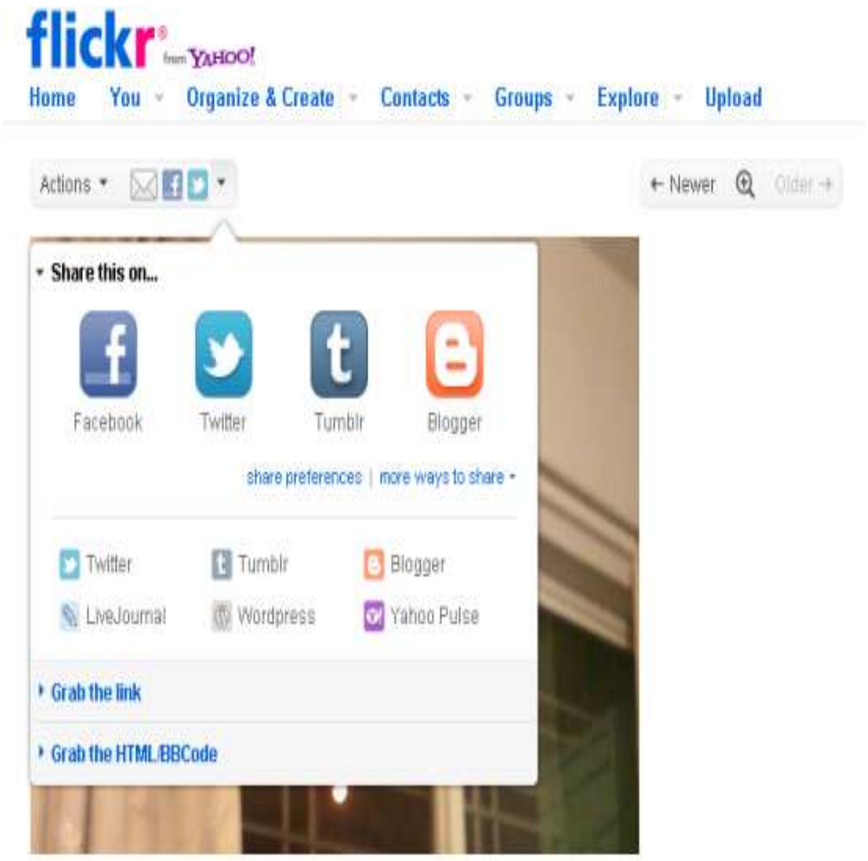
WEB APIs

- ***Web APIs*** are the defined interfaces through which interactions happen between an enterprise and applications that use its assets. *Usually in Extensible Markup Language* ***(XML)*** *or JavaScript Object Notation (**JSON**) format.*

- ***API*** is typically defined as a set of *Hypertext Transfer Protocol* **(*HTTP*)** request messages.

USABILITY OF WEB APIs

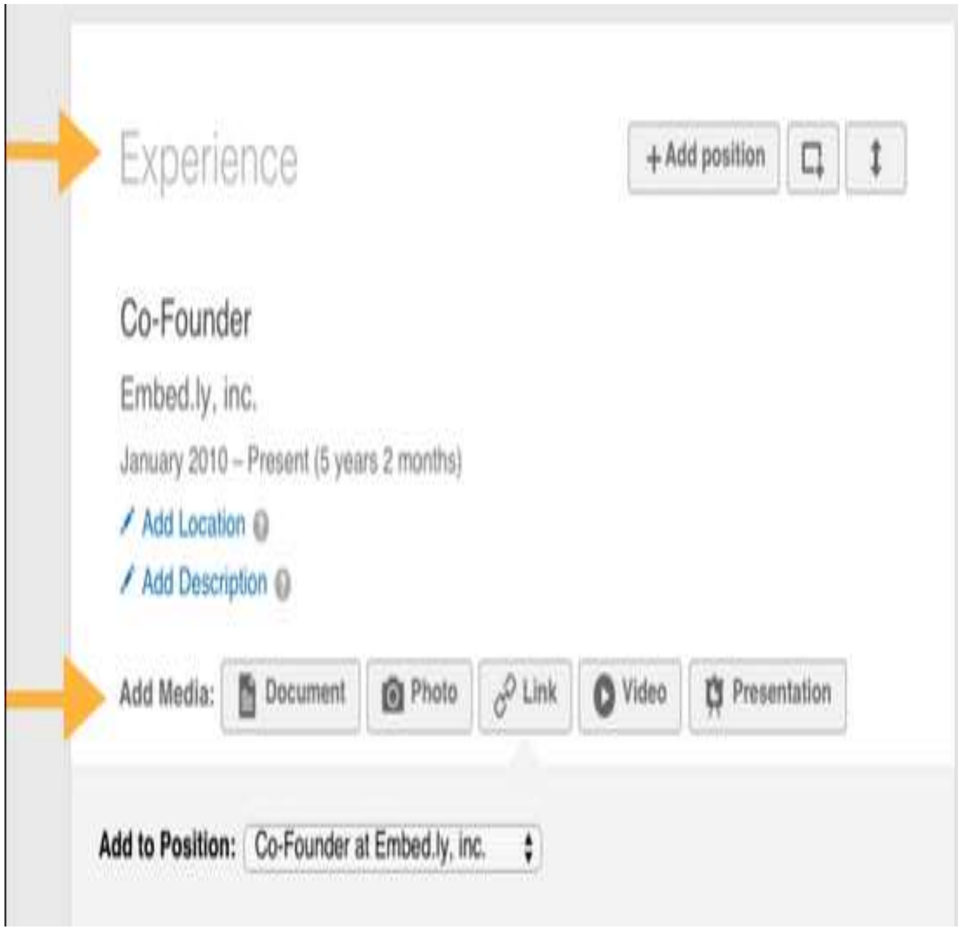
Photos can be shared from sites like Flickr and Photobucket **to social network sites** like Facebook and MySpace.



Share photos from Flickr

USABILITY OF WEB APIs [2]

Content can be embedded, e.g. embedding a presentation from SlideShare on a LinkedIn profile.



Experience

+Add position

Co-Founder

Embed.ly, inc.

January 2010 – Present (5 years 2 months)

Add Location

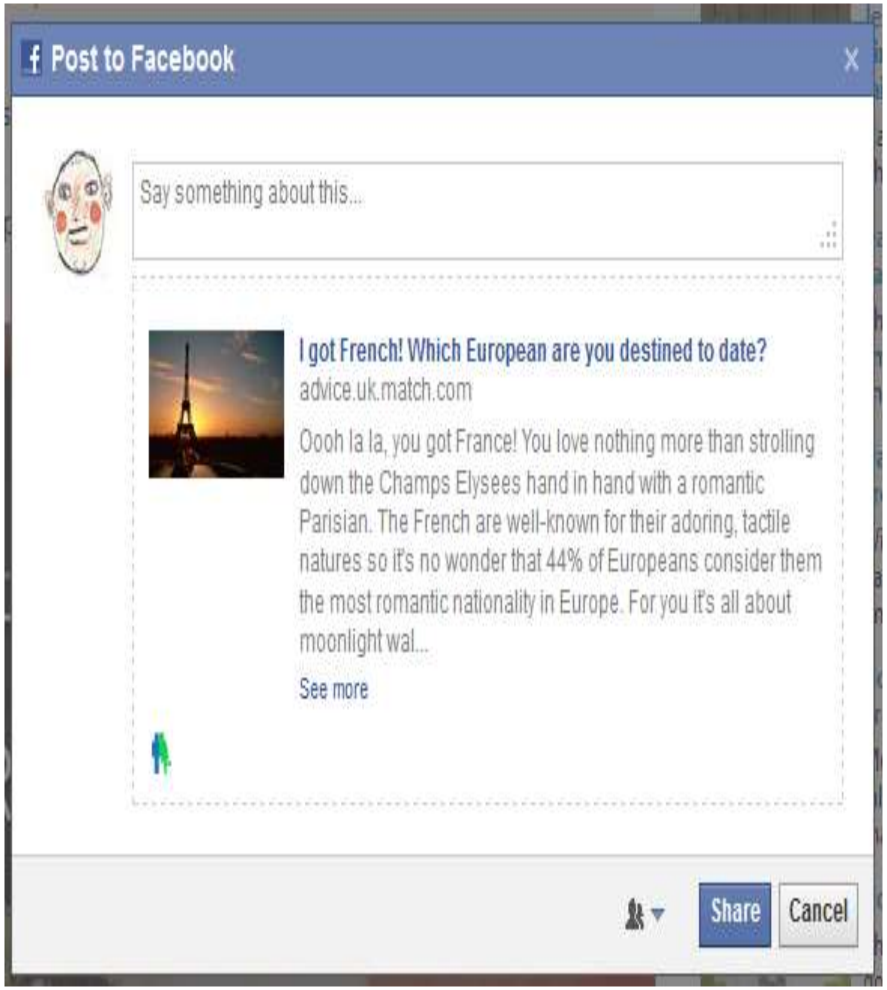
Add Description

Add Media: Document Photo Link Video Presentation

Add to Position: Co-Founder at Embed.ly, inc.

USABILITY OF WEB APIs [3]

Content can be dynamically posted. Sharing live comments made on Twitter with a Facebook account, for example, is enabled by their APIs. Etc.


Post to Facebook

Say something about this...

I got French! Which European are you destined to date?
advice.uk.match.com
Oooh la la, you got France! You love nothing more than strolling down the Champs Elysees hand in hand with a romantic Parisian. The French are well-known for their adoring, tactile natures so it's no wonder that 44% of Europeans consider them the most romantic nationality in Europe. For you it's all about moonlight wal...
See more

Share Cancel

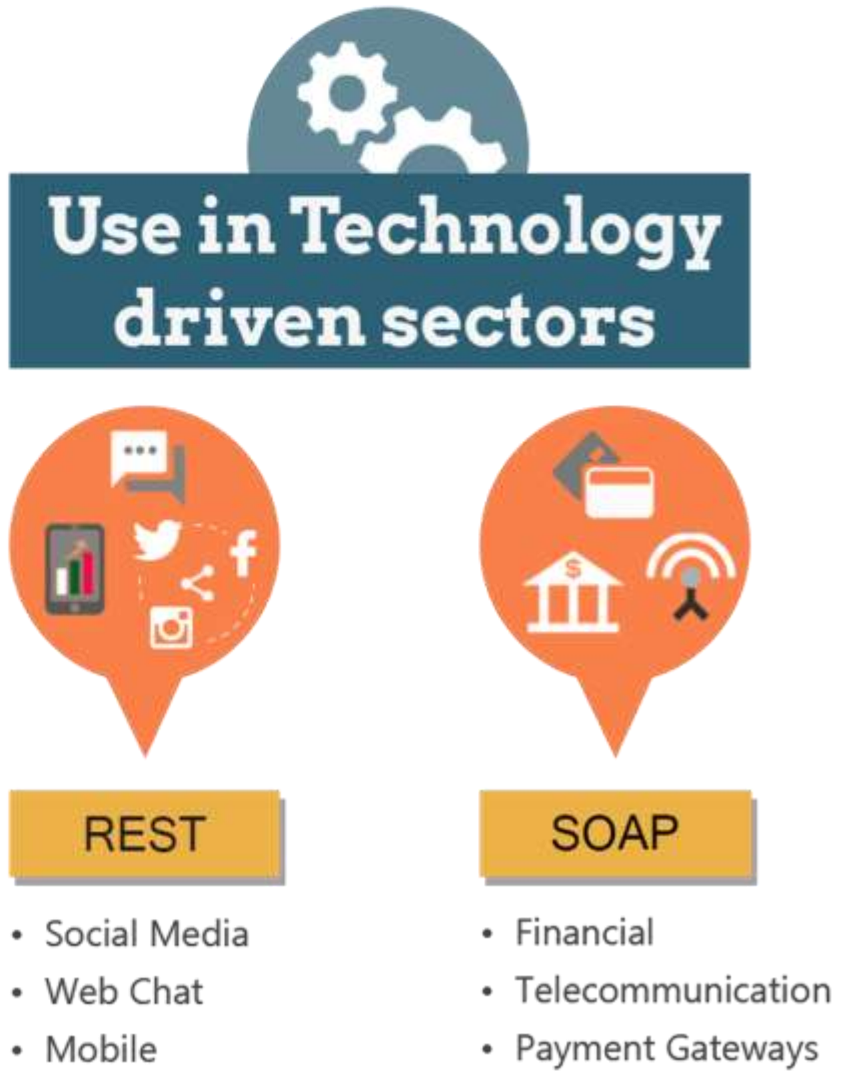
WEB SERVICES

The two approaches for interfacing to the web with web services, namely:

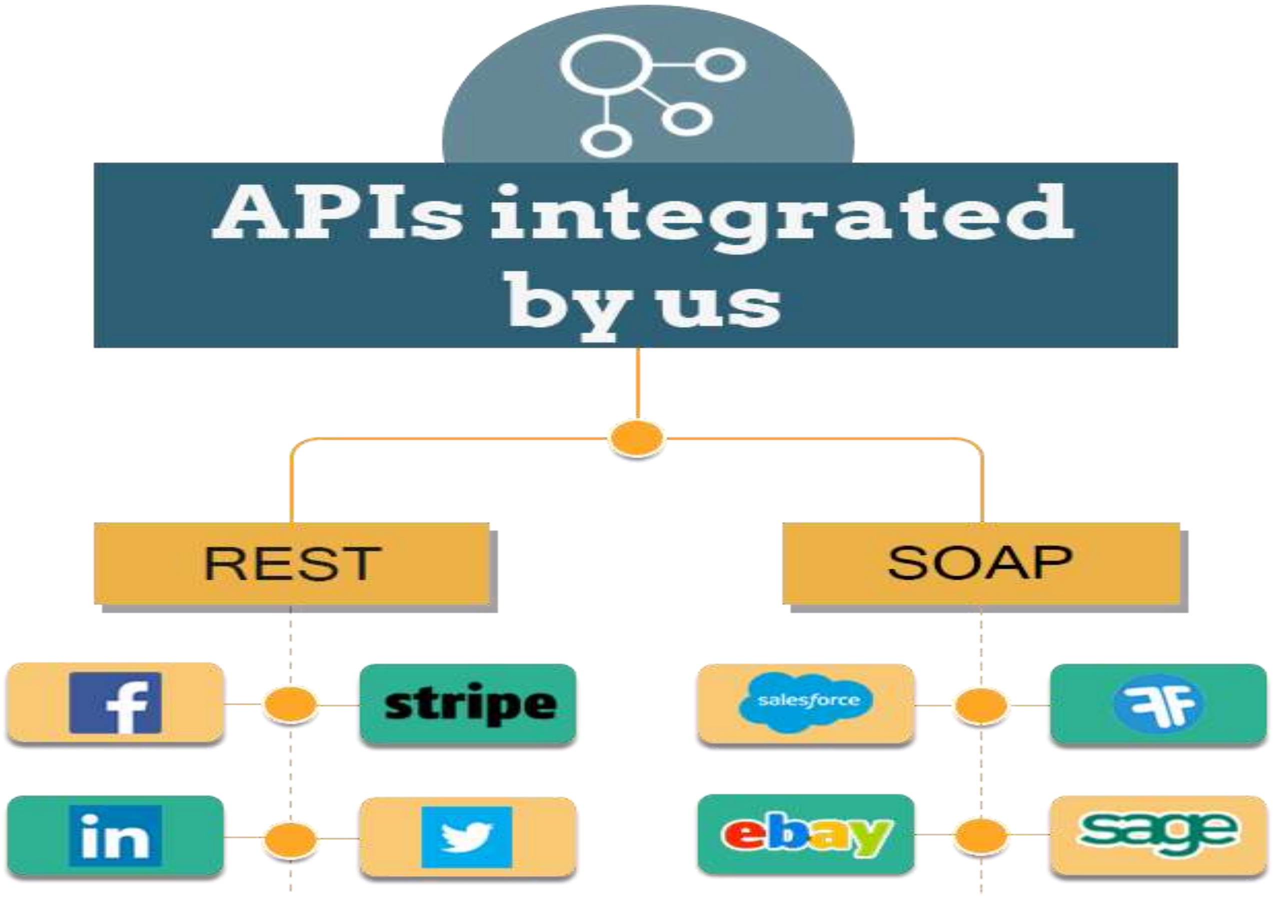
- **SOAP (Simple Object Access Protocol)**
- **REST (Representational State Transfer)**

SOAP VS REST

SOAP works on a standard set of rules based on ***XML*** (eg. ***HTTP***) otherwise **REST** supports many format (***JSON***, ***XML,****etc*) and doesn't employ any additional messaging layer.

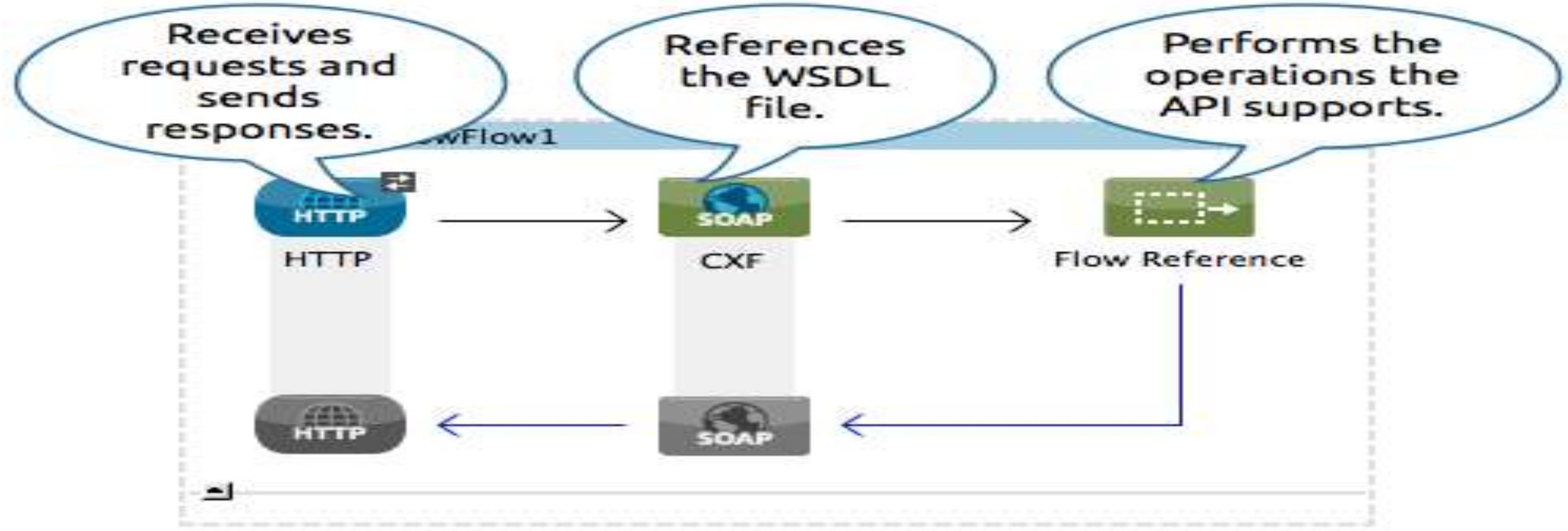


SOAP VS REST (USERS)

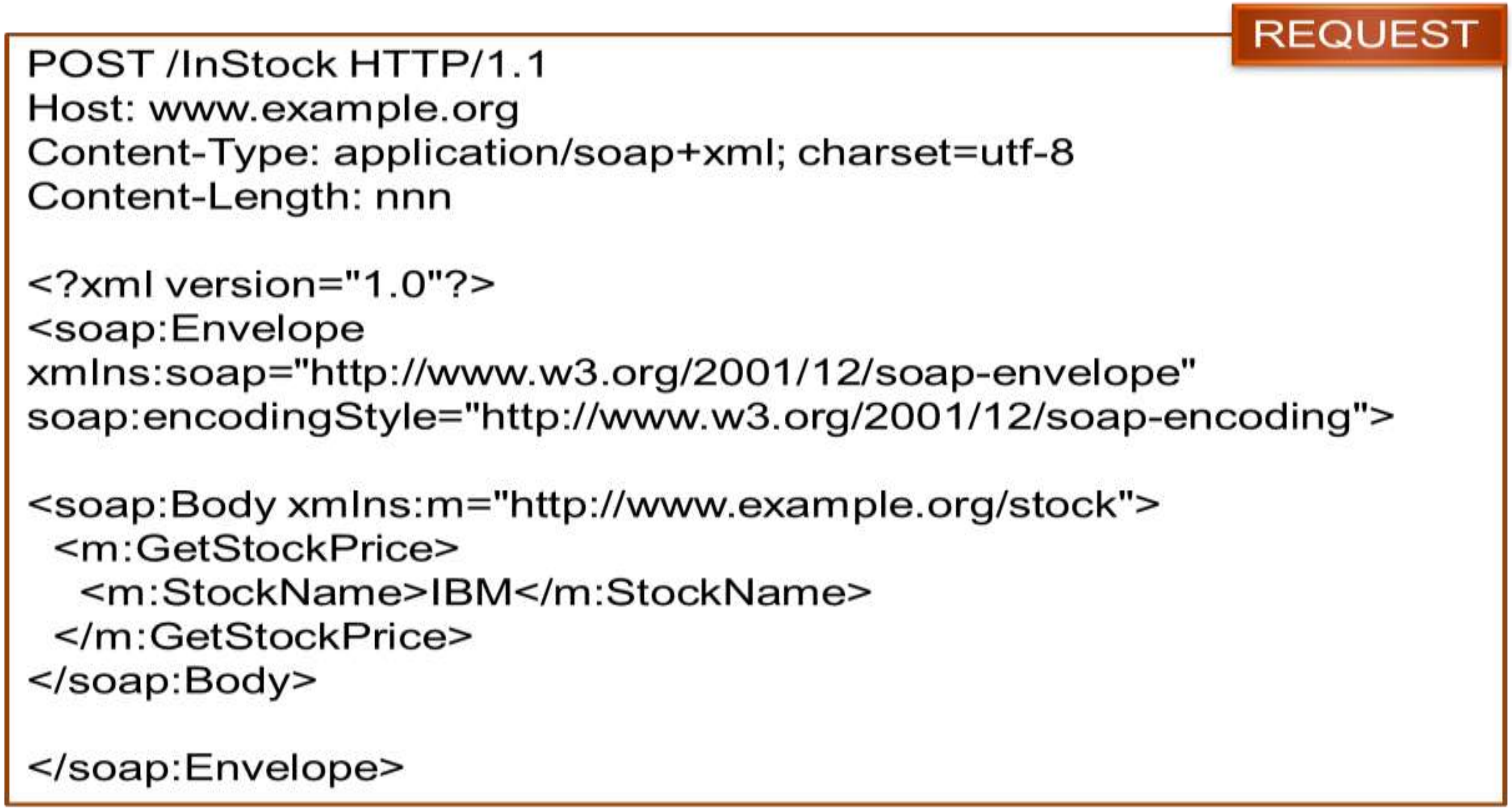


WHAT IS SOAP?

Simple Object Access Protocol (SOAP) is a protocol for exchange of structured information on a decentralized and distributed platform using **XML (eXtensible Markup Language)**.

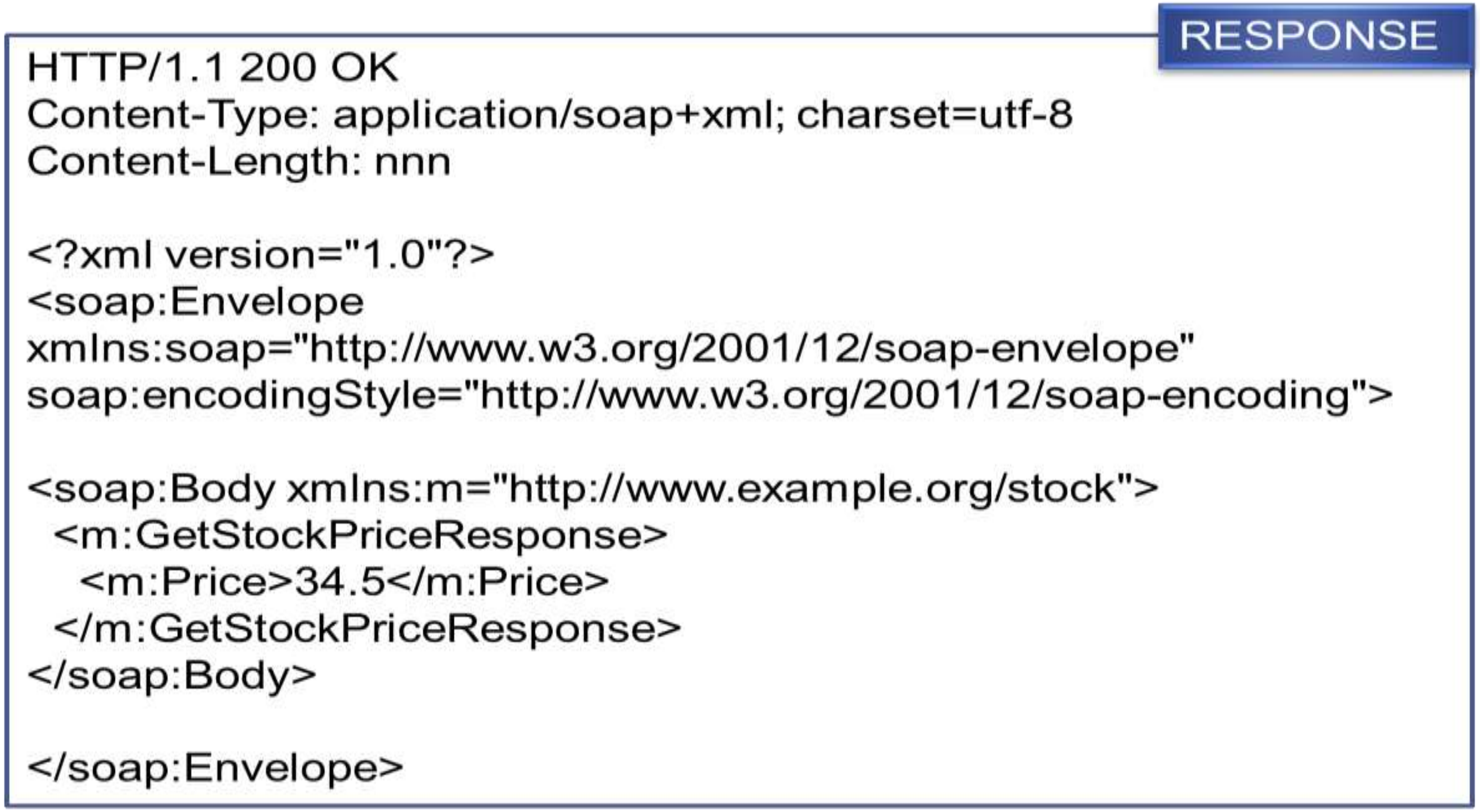

Receives requests and sends responses.
References the WSDL file.
Performs the operations the API supports.
...wFlow1
HTTP
HTTP
SOAP
CXF
Flow Reference
HTTP
SOAP

SOAP SIMPLE REQUEST



```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?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 xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

SOAP SIMPLE RESPONSE

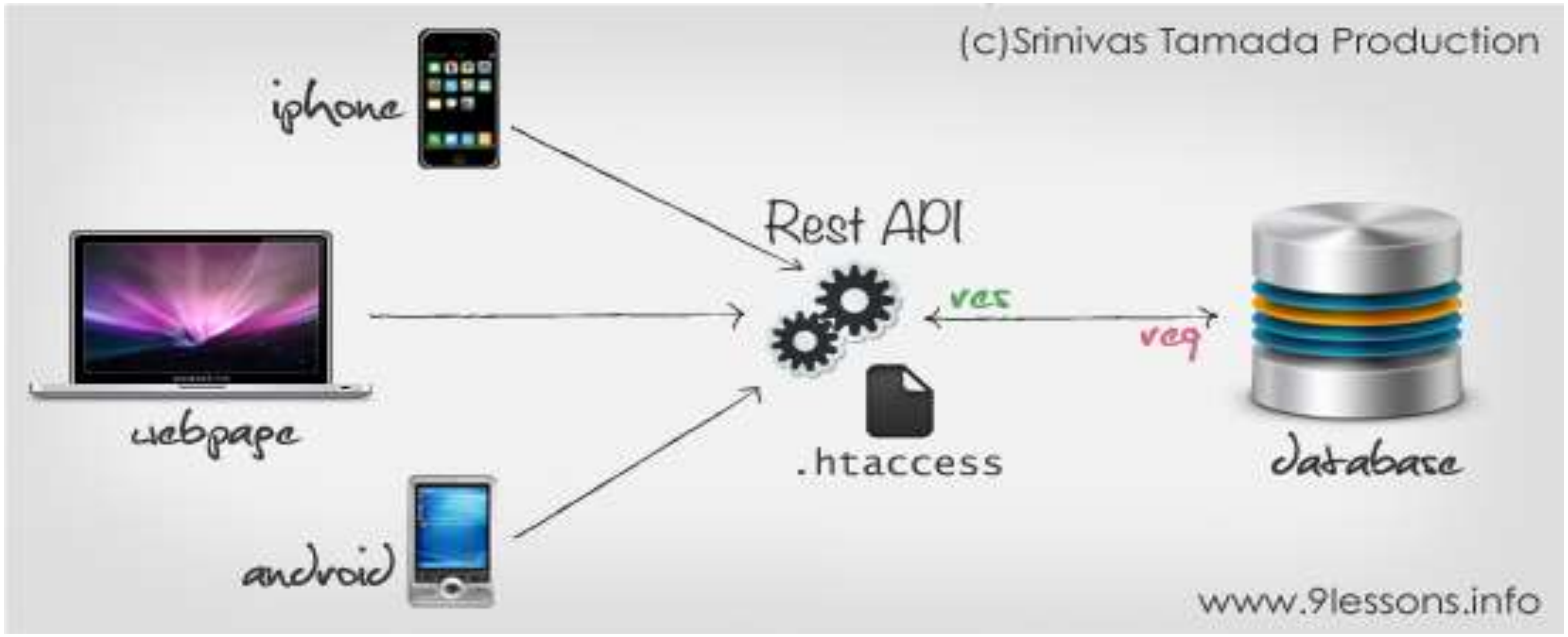


RESPONSE

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?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 xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

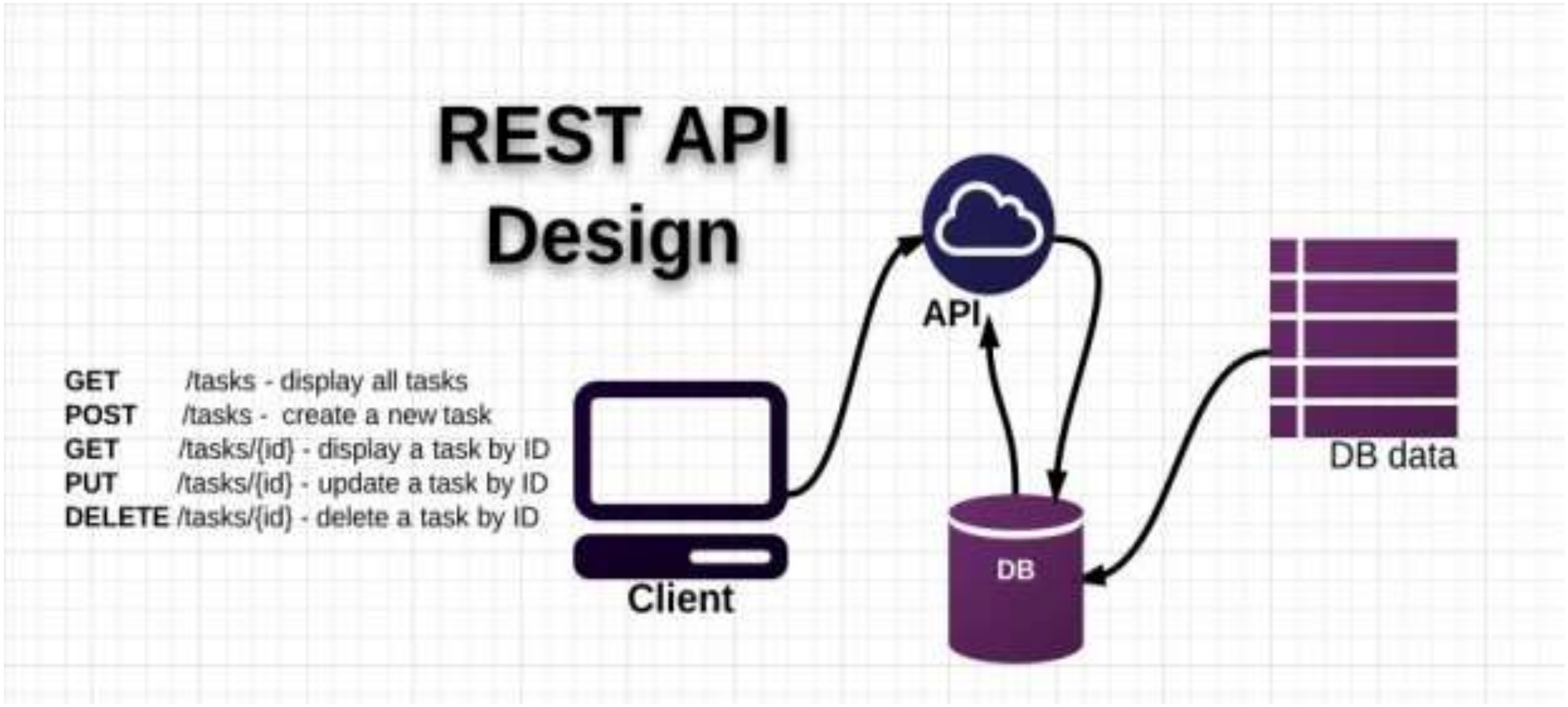
WHAT IS REST?

Representational State Transfer or REST basically means that each unique URL is a representation of some object and supports format like **JSON**, **XML etc.**



HOW REST WORKS?

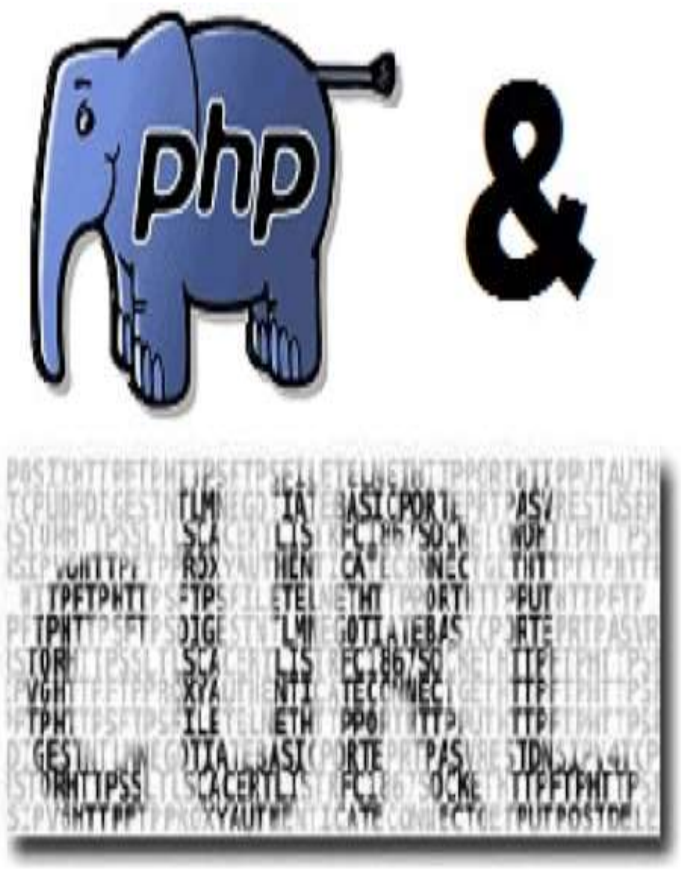
Using REST, you can get the contents of that object using an **HTTP** using some methods like **GET**, **POST, PUT, DELETE or POST** to modify the object.



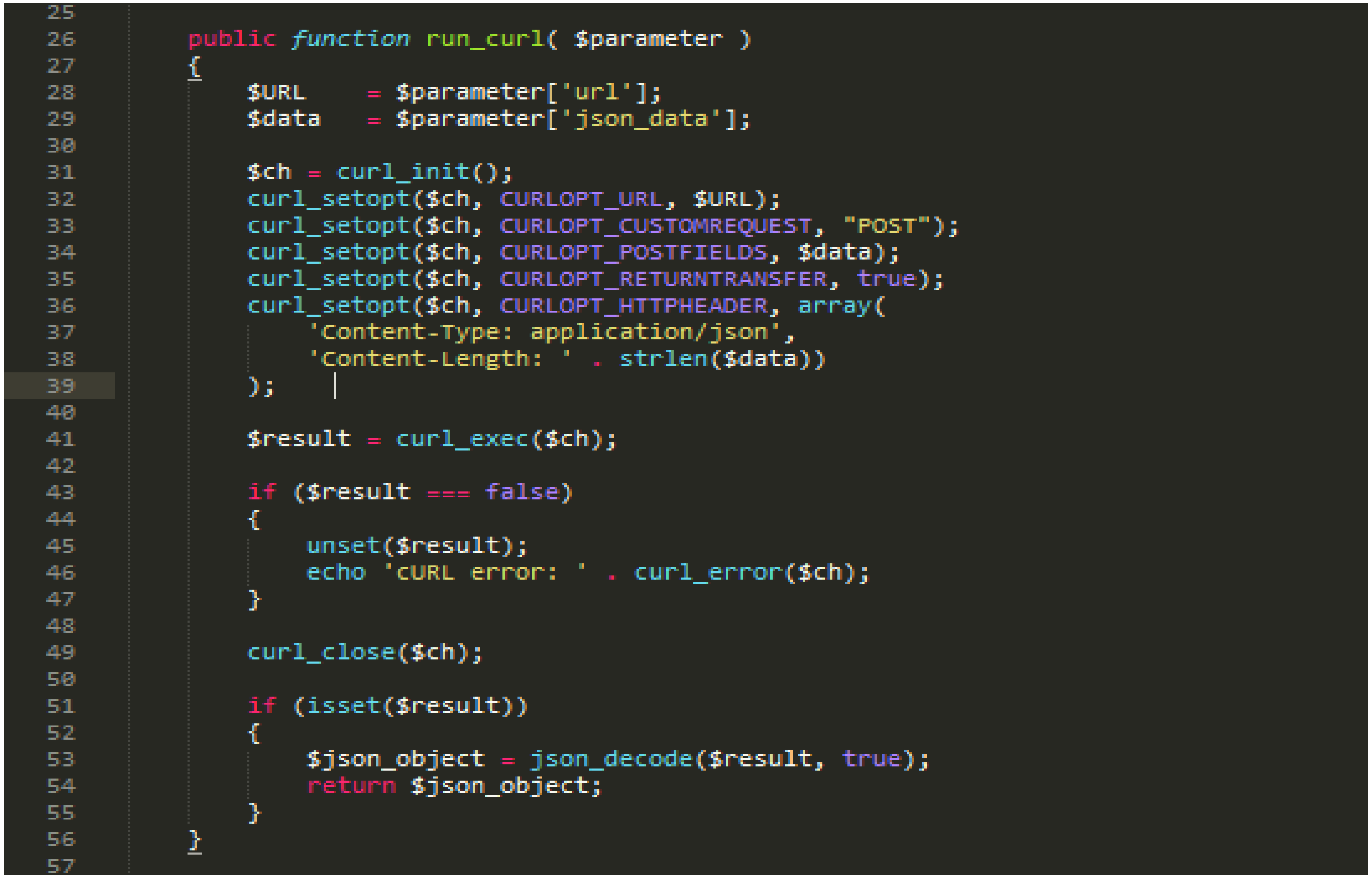
```
GET     /tasks - display all tasks
POST    /tasks -  create a new task
GET     /tasks/{id} - display a task by ID
PUT     /tasks/{id} - update a task by ID
DELETE /tasks/{id} - delete a task by ID
```

REST API WITH cURL

- **cURL** allows you to connect and communicate to many different types of servers with many different types of protocols.
- **cURL** currently supports the **http, https, ftp etc**.



HTTP POST THROUGH CURL



```php

    public function run_curl( $parameter )
    {
        $URL     = $parameter['url'];
        $data    = $parameter['json_data'];

        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $URL);
        curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
        curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_HTTPHEADER, array(
            'Content-Type: application/json',
            'Content-Length: ' . strlen($data))
        );

        $result = curl_exec($ch);

        if ($result === false)
        {
            unset($result);
            echo 'cURL error: ' . curl_error($ch);
        }

        curl_close($ch);

        if (isset($result))
        {
            $json_object = json_decode($result, true);
            return $json_object;
        }
    }

```

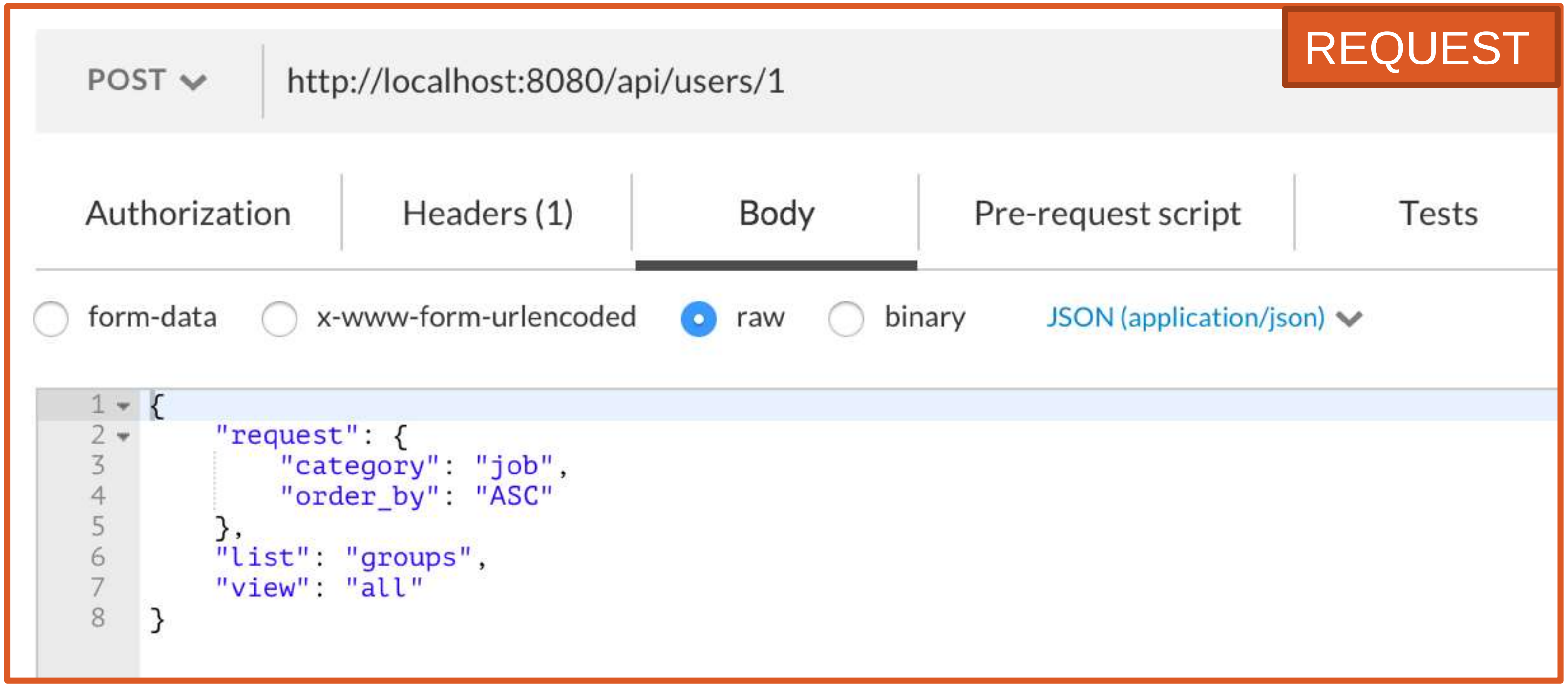
HTTP POST THROUGH CURL [2]

cURL functions:

- **curl_init —** Initialize a cURL session
- **curl_setopt** — Set an option for a cURL transfer
    - Ch — A cURL handle returned by curl_init().
    - Option — The *CURLOPT_XXX* option to set.
        - **CURLOPT_URL**
        - **CURLOPT_CUSTOMREQUEST**
        - **CURLOPT_POSTFIELDS**
        - **CURLOPT_RETURNTRANSFER**
        - **CURLOPT_HTTPHEADER**
    - Value — The value to be set on option.
- **curl_exec** — Perform a cURL session
- **curl_error** — Return a string containing the last error for the current session
- **curl_close —** Close a cURL session

REST cURL SIMPLE REQUEST

- URL: http://localhost:8080/api/users/1
- Method: **POST**
- Content-Type: **application/json**

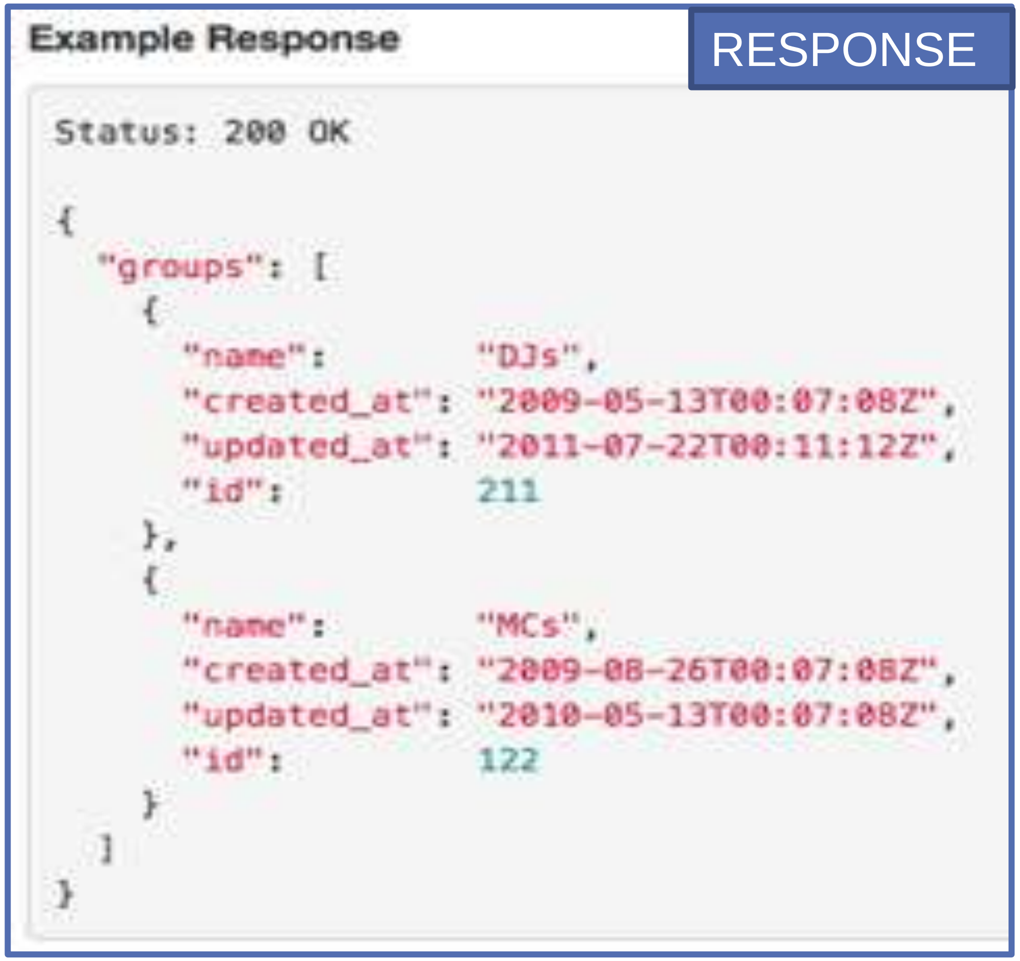


POST http://localhost:8080/api/users/1

Body: [✓ raw] JSON (application/json)

```
{
    "request": {
        "category": "job",
        "order_by": "ASC"
    },
    "list": "groups",
    "view": "all"
}
```

REST cURL SIMPLE RESPONSE

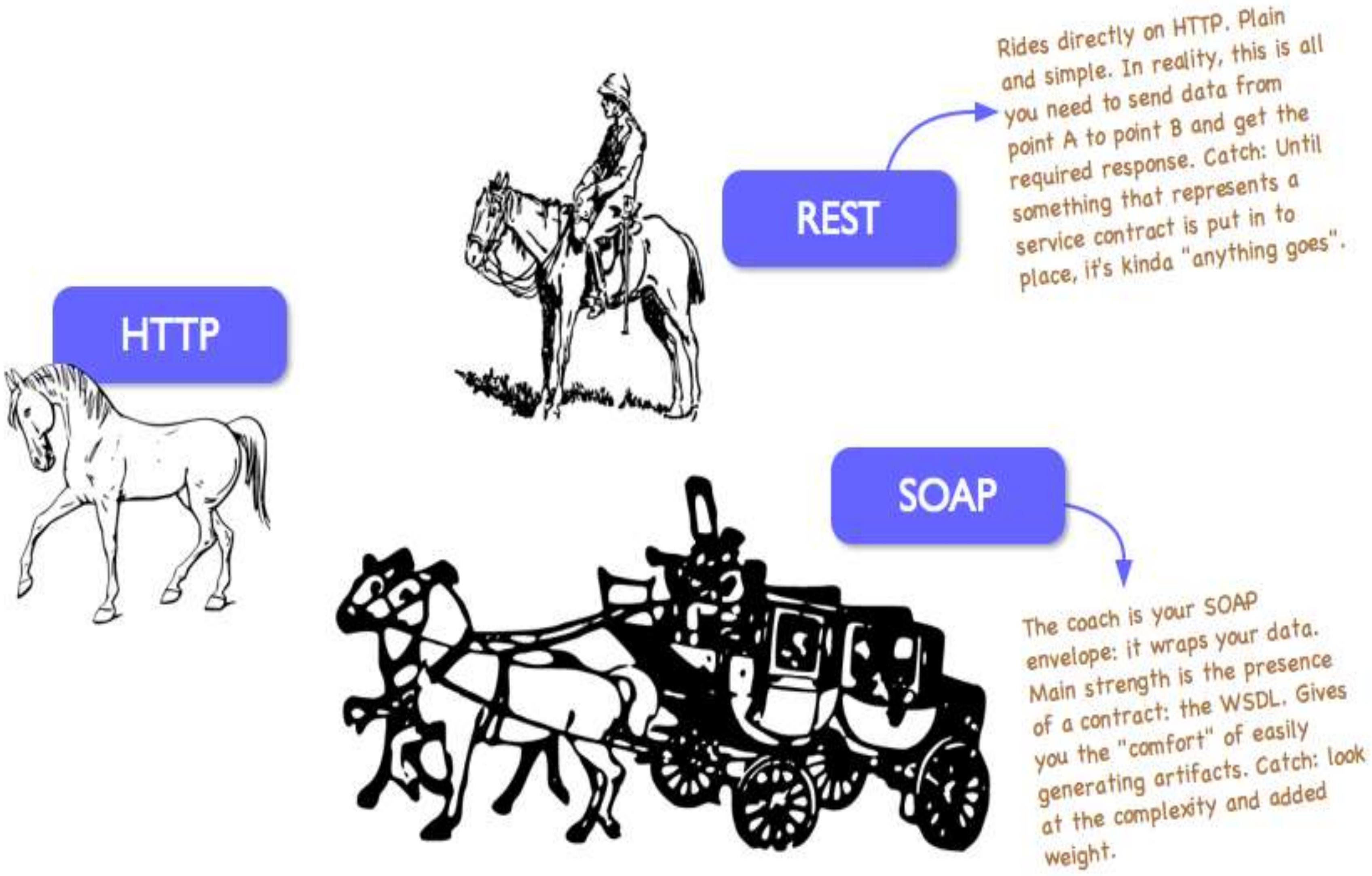


Example Response

RESPONSE

```
Status: 200 OK

{
  "groups": [
    {
      "name":       "DJs",
      "created_at": "2009-05-13T00:07:08Z",
      "updated_at": "2011-07-22T00:11:12Z",
      "id":         211
    },
    {
      "name":       "MCs",
      "created_at": "2009-08-26T00:07:08Z",
      "updated_at": "2010-05-13T00:07:08Z",
      "id":         122
    }
  ]
}
```

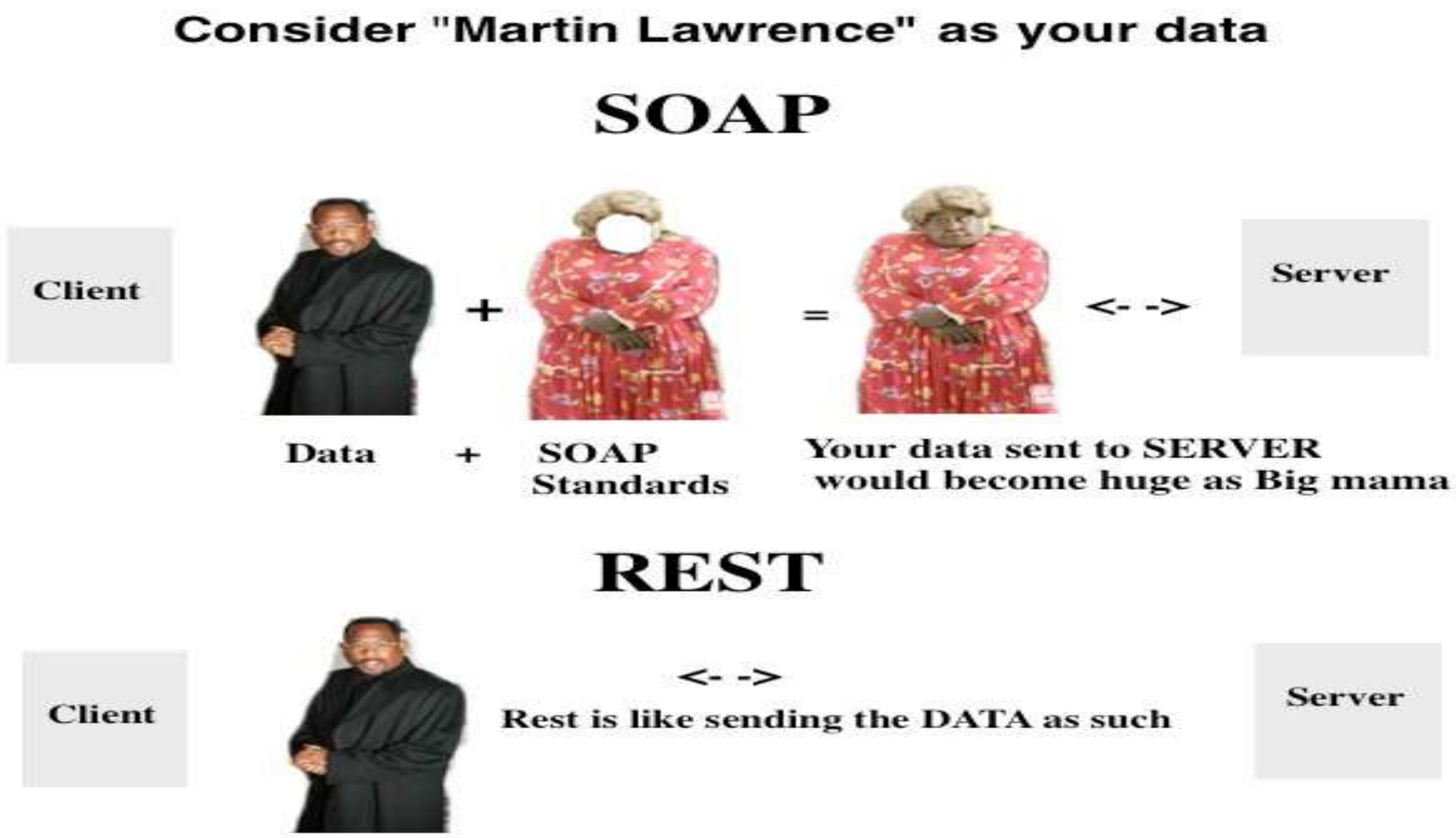
SOAP VS REST (2)



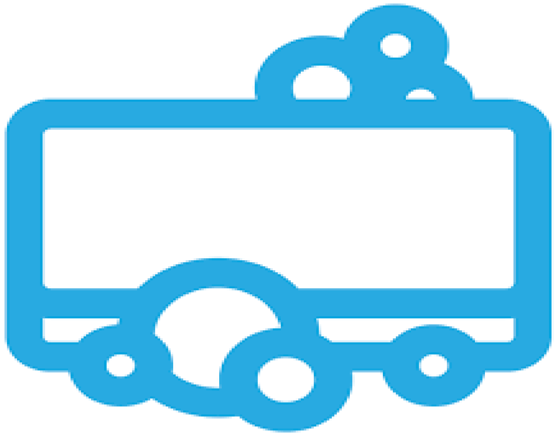
Rides directly on HTTP. Plain and simple. In reality, this is all you need to send data from point A to point B and get the required response. Catch: Until something that represents a service contract is put in to place, it's kinda "anything goes".

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.

SOAP VS REST (3)


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

WHY USING SOAP?



The main advantages of **SOAP** web services are:

- **Easy to consume** – sometimes.
- **Rigid** – type checking, adheres to a contract (provider and consumer) have to agree on the exchange format.
- **Development tools** – using tools
- ***Can use almost any transport to send the request*** — SMTP (Simple Mail Transfer Protocol), JMS (Java Messaging Service).
- ***Asynchronous processing and invocation***—guaranteed level of reliability and security then ensure this type of operation.
- ***Stateful operations***—provide support to contextual information and conversational state management. (Security, Transactions, Coordination, etc).

WHY USING REST?

The main advantages of **REST** web services are:

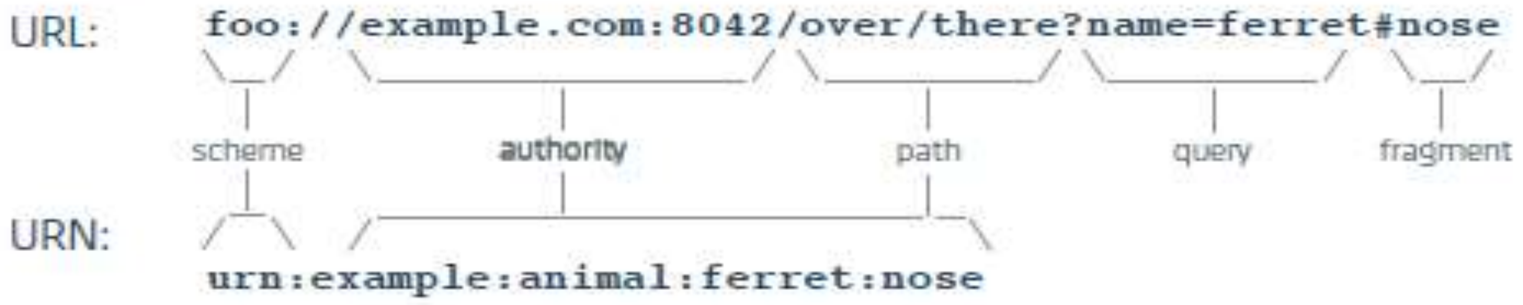
1. **Lightweight** – The requests and responses can be short.
2. **Human Readable Results** – Flexible & Simple**,** URIs for Identification
3. **Easy to build** – No toolkits required
4. **Totally stateless operations** – Stateless CRUD (Create, Read, Update, and Delete) operations.
5. **Caching situations** – Information can be cached because of the totally stateless operation.

URIs FOR IDENTIFICATION

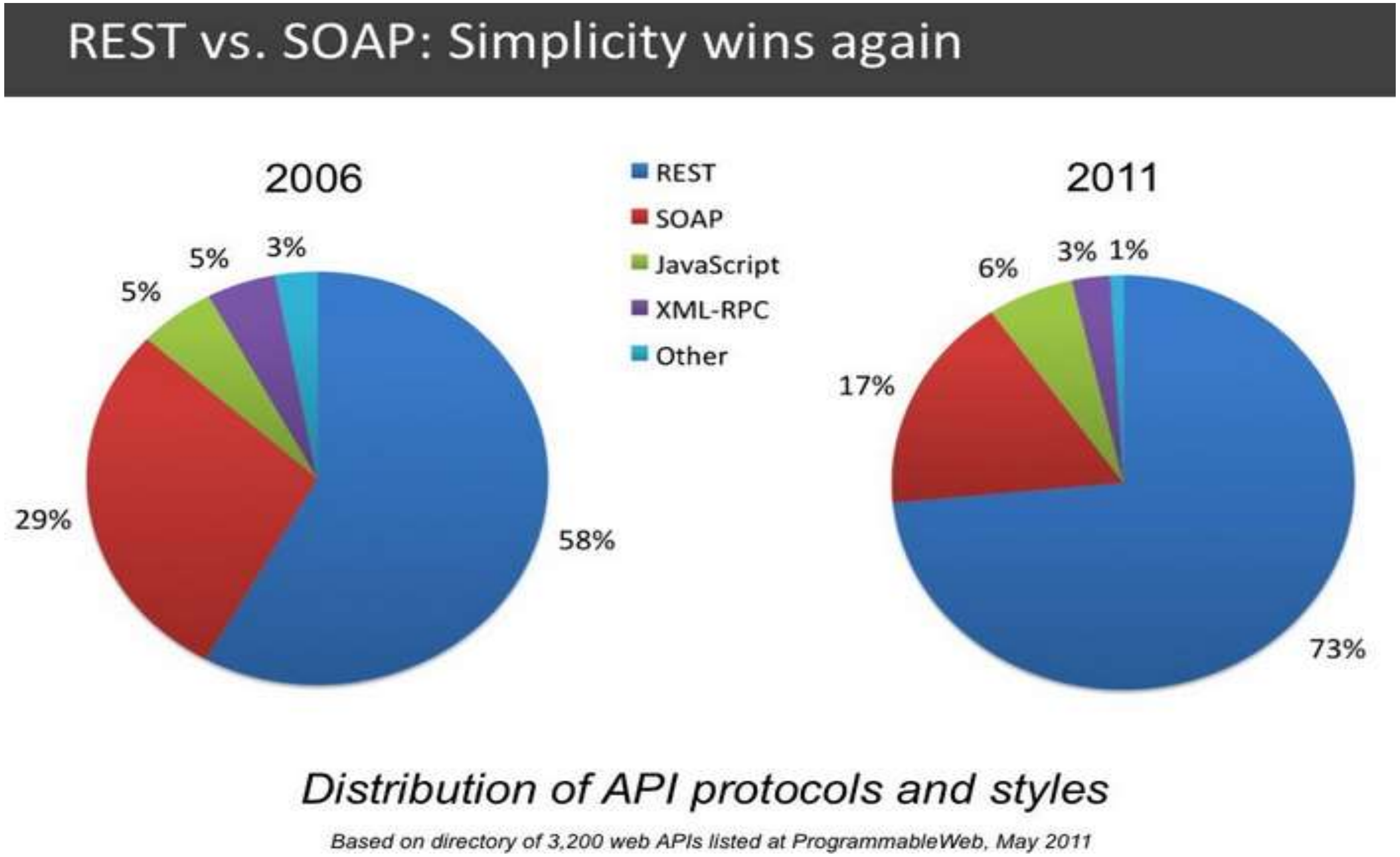
The components of a **URI** (Uniform Resource Identifier) include:

- **Scheme Name**—Identifies the protocol (e.g., FTP:, HTTP:, HTTPS:, IRC:)
- **Hierarchical Part**—Intended to hold information hierarchical in nature.
- **Query**—contains additional identification information that is non-hierarchical in nature and often separated by a question mark ("?")
- **Fragment**—provides direction to a secondary resource within the primary one identified by the Authority and Path and separated from the rest by a hash ("#")

The structure of URIs


URL:
foo://example.com:8042/over/there?name=ferret#nose
scheme
authority
path
query
fragment
URN:
urn:example:animal:ferret:nose

SOAP AND REST COMPARISON



CONCLUSION

- ***SOAP*** requests use ***POST*** and require a complex XML while ***REST*** doesn't.
- ***SOAP*** reads cannot be cached on other hand ***REST*** could do by a proxy server.
- ***REST*** allows different data formats where ***SOAP*** only allows ***XML***.
- ***REST*** better performance and scalability.

Continue operations? ***SOAP*** it. Stateless operations? ***REST*** it.

THANK YOU