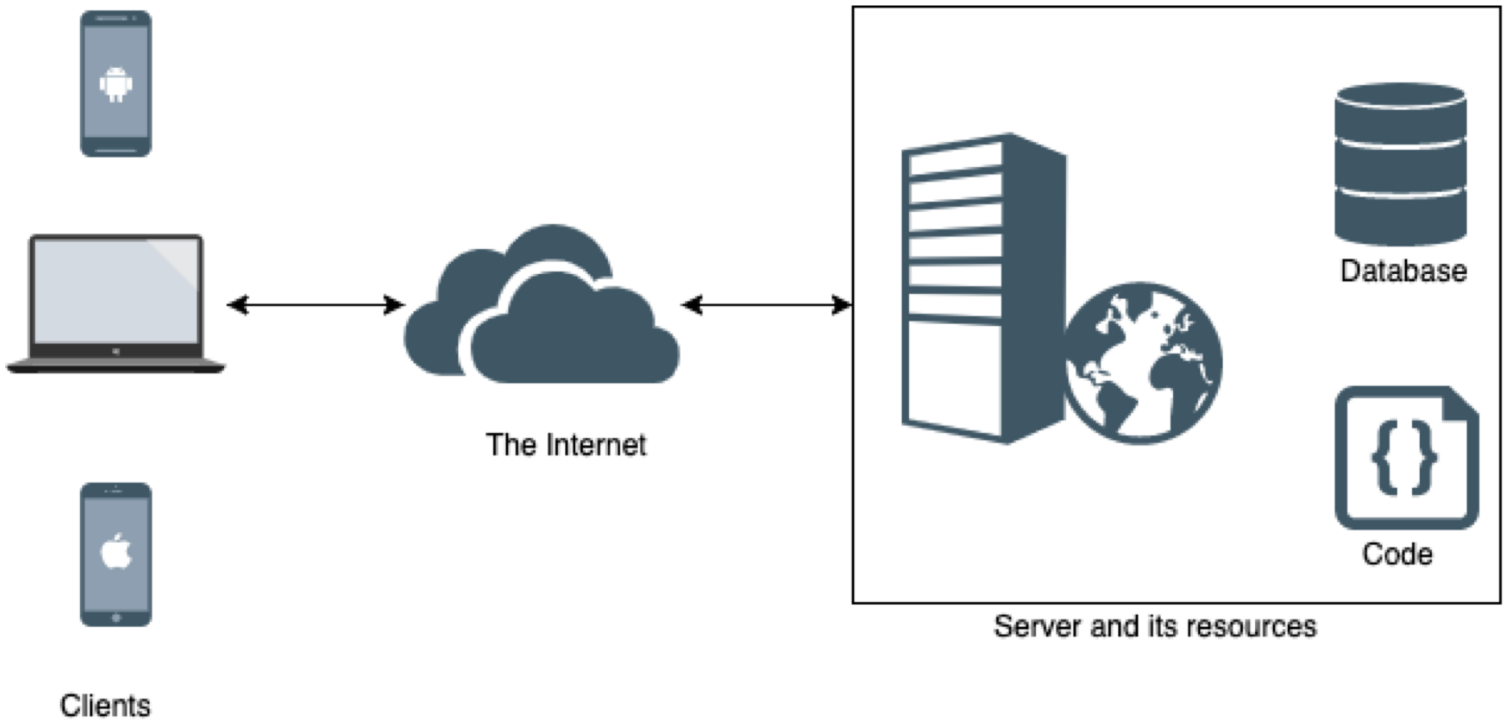
# WEB SERVICES AND APIS

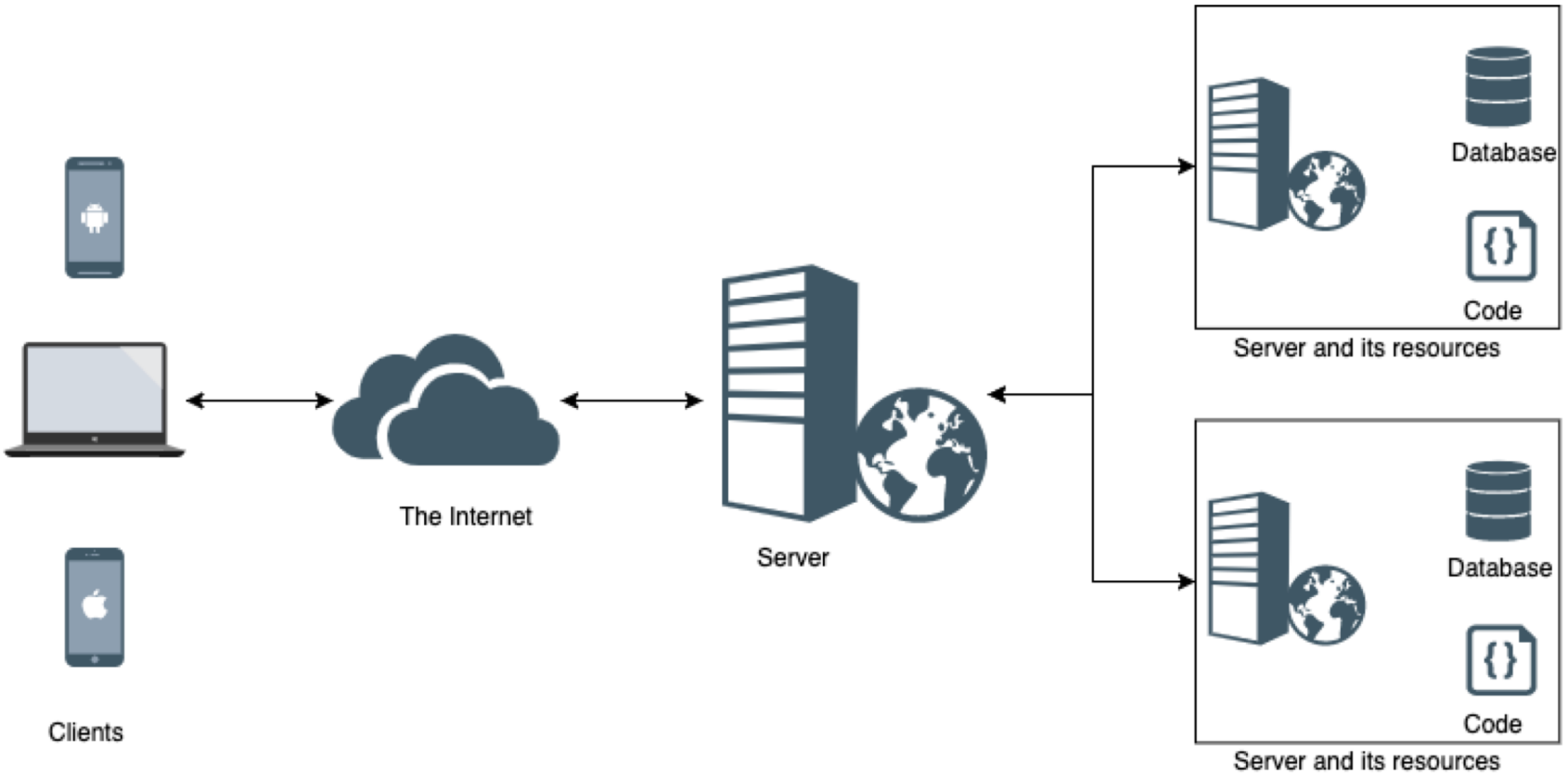
# WEB SERVICES – INTRODUCTION

- So far in our knowledge of the web architecture, a client requests resources from the server, the server sends back the files to be displayed (or some data to a client code that will display it).



# WEB SERVICES – INTRODUCTION

- Consider this situation:



# WEB SERVICES – INTRODUCTION

- **Services** are provided through a web server.
  - Web servers can respond with not just documents.
  - Access to **software components:**
    - **Data (e.g. Get Tweets related to a certain hashtag)**
    - **Functionality (e.g. Post a tweet via a 'share' button)**

# WEB SERVICES – INTRODUCTION

- A web service is a more general concept. It is an integration technique.
- A web service is a web application that requires an interaction with another application (remotely callable methods).
- A **Web service** is a collection of one or more related methods that can be called by remote systems by using standard protocols on the Web
- A web application can require several web services.
  - Build HTTP requests (POST, GET) dynamically in code.

# WEB SERVICES – ADVANTAGES

- Benefits of using web services:
  - It is platform and language independent.
  - Can connect disparate platforms and systems.
  - Facilitate reuse.

# WEB SERVICES – APIS

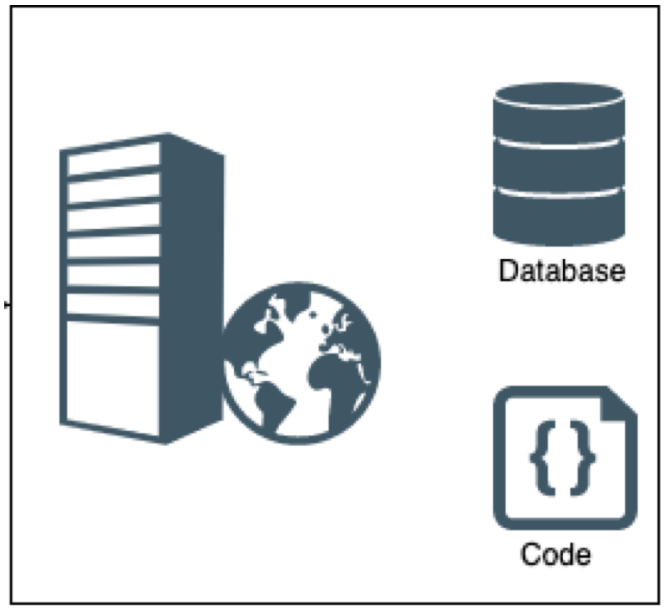
- **How to contact and request a web service?**
  - **Using the web service's Application Programming Interface (API)**
  - **An API shows the documentation of the service (URL, Method, Parameters, and returned type)**
  - **An API is "The point of interaction or communication between a computer and any other entity [2]"**

# WEB SERVICES – APIS

'Function calls' in the web paradigm:

`getStudent(String id)` → `getStudent.php?id=…`

Request: `getStudent.php?id=123456`




Server and its resources

Response:

```
<student>
    <name>Afnan AlSubaihin</name>
    <email>aalsubaihin@ksu.edu.sa<email>
</student>
```

# WEB SERVICES – EXAMPLES

Example: Saudi National Addresses API

https://api.address.gov.sa/

## للمطورين

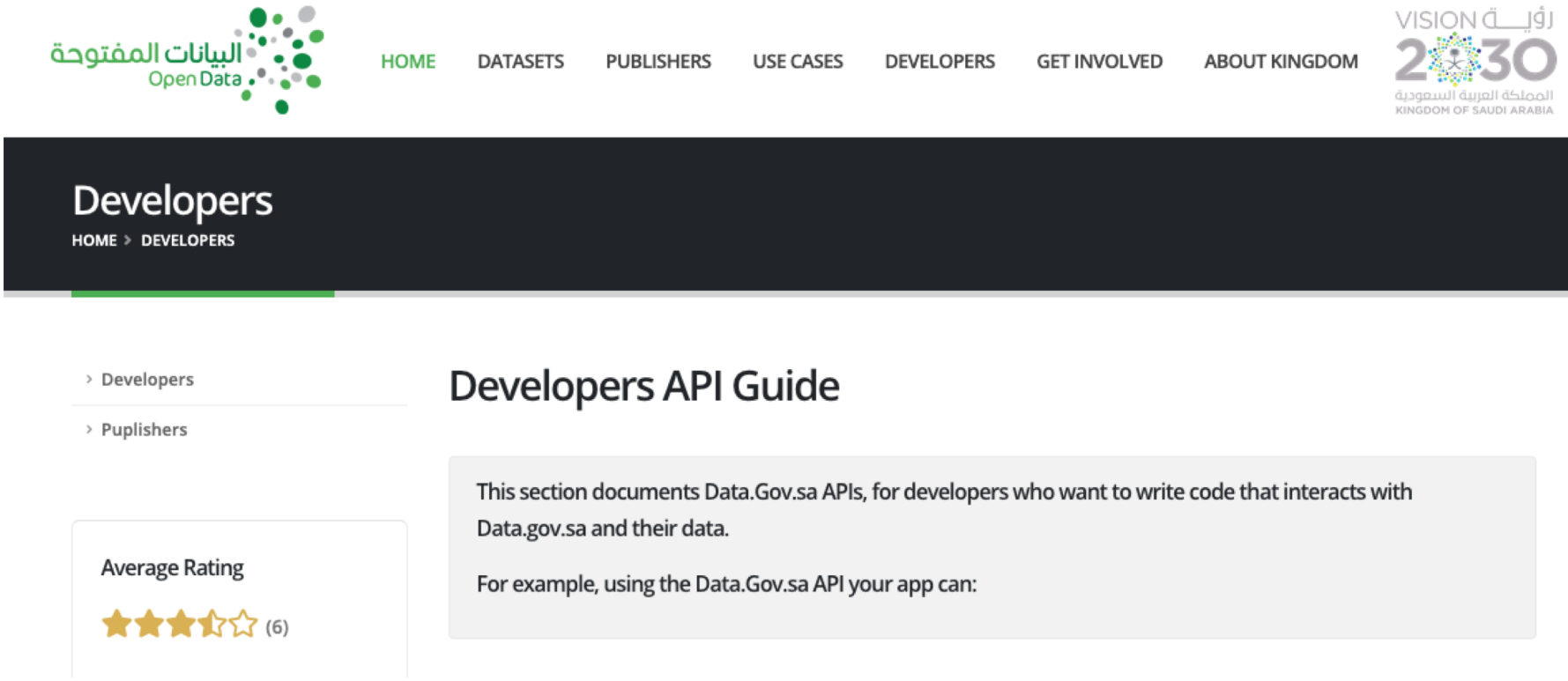
تم تطوير باقات ومنتجات مخصصة للمطورين بهدف اتاحة وتوفير آلية للحصول على قواعد بيانات العنوان الوطني. ويستطيع المطورون من ربط تطبيقاتهم بالمعلومات الجغرافية والمكانية في المملكة مثل، قائمة المدن، والأحياء، التحقق من صحة العناوين المدخلة من قبل مستخدمين تطبيقاتهم وغيرها. ويستطيع المطورون من التسجيل الإلكتروني ومجاناً للبدء في استخدام الـ *API*

كما يتيح استخدام قواعد بيانات العنوان الوطني في بناء التطبيقات بغض النظر عن أنظمة التشغيل المختلفة، بالاضافة الى امكانية توظيفها لبناء التطبيقات البرمجية للأجهزة الذكية او لأجهزة الحاسوب

سجل الآن

# WEB SERVICES – EXAMPLES

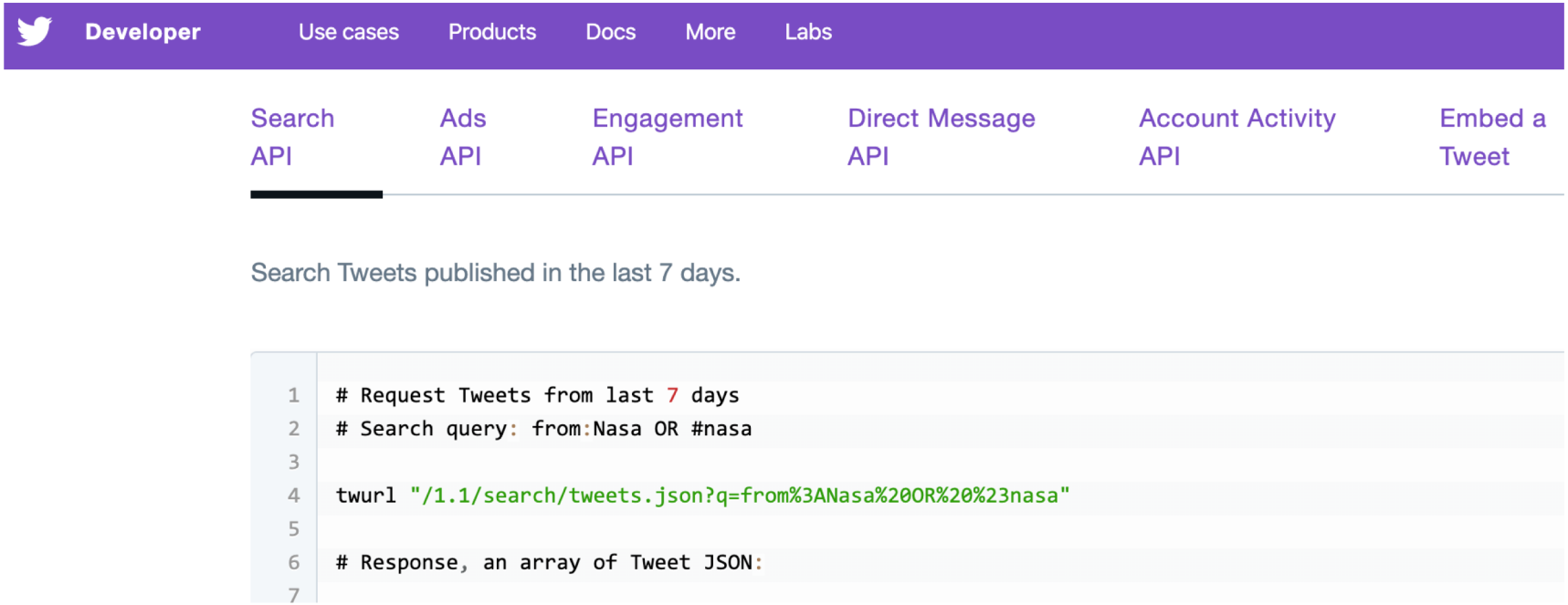
Example: Saudi Government Open Data API
https://data.gov.sa/en/developers



# WEB SERVICES – EXAMPLES

Example: Twitter API

https://developer.twitter.com/en.html



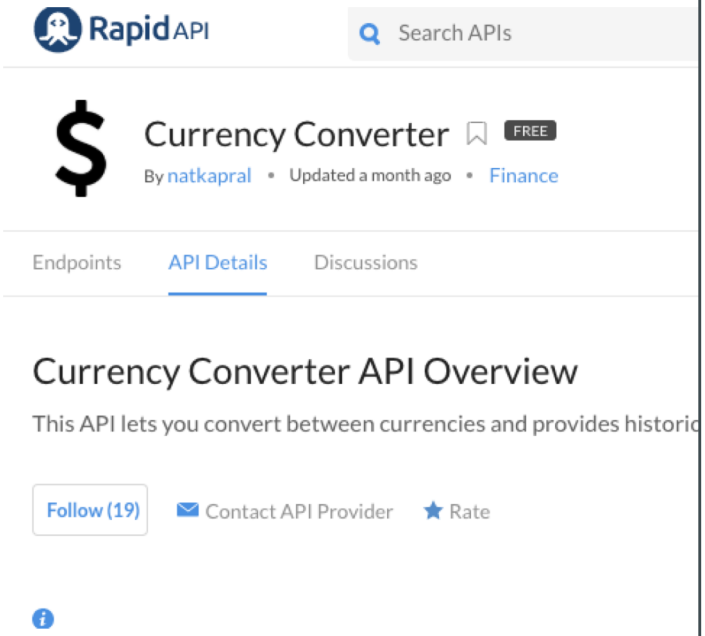

Using the Twitter API, you can build a web application that can also show latest tweets, for example.

# WEB SERVICES – EXAMPLES

- **Code example:** We would like to be able to **convert** price amounts from Saudi Riyals to British pounds.
- Writing code that will keep up to date with currency rate changes might be too **challenging**.
- **Solution**: Use an API (get the information when needed from a dedicated service)

# WEB SERVICES – INTRODUCTION

- API Documentation:




**Currency Conversion**

Currency Conversion Lookup converts one currency to another.

Endpoint:

```
GET: /currency/convert?&amp;from={from}&amp;to={to}&amp;amount={amount}&amp;format={format}
```

Parameters

- from: the code of the base currency.The default base currency is EUR.
- to: the code of the currency to convert to. If the value is not provided then the list of all supported currencies will be returned. You can provide several currencies using comma (to=GBP,EUR,AUD)
- amount: amount that needs to be converted
- format: json or xml

Example response:

```
{
    "status":"success",
    "updated_date":"2018-12-27",
    "base_currency_code":"EUR",
    "amount":10,"
    "base_currency_name":"Euro",
    rates":{
        "GBP":{
            "currency_name":"Pound Sterling",
            "rate":"0.9007",
            "rate_for_amount":"9.0070"
        }
    }
}
```

## index.html

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
        $("#riyal").blur(function(){
                var r = $("#riyal").val();
                $.get( "convert.php",
                        {riyal: r} ,
                        function(data){$("#pound").val(data.rates.GBP.rate_for_amount);},
                        "json");});
        });
</script>
<title>Convert Saudi Riyals to British Pounds</title>
</head>
<body>

<form>
<input type="text" required name="riyal" id="riyal"/>  ﷼ <br/>

is <br/>

<input type="text" required name="pound" id="pound"/> £

</form>
<body>
</html>
```

## convert.php

```php
<?php
//Get data from request:
$base_amount = $_GET['riyal'];
//prepare 2nd request parameters:
$from = "SAR";
$to = "GBP";
$format = "json";


//call URL:
$curl = curl_init();

curl_setopt_array($curl, array(
	CURLOPT_URL => "https://currency-converter5.p.rapidapi.com/currency/convert?format=".$format."&from=".$from."&to=".$to."&amount=".$base_amount,
	CURLOPT_RETURNTRANSFER => true,
	CURLOPT_FOLLOWLOCATION => true,
	CURLOPT_ENCODING => "",
	CURLOPT_MAXREDIRS => 10,
	CURLOPT_TIMEOUT => 30,
	CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
	CURLOPT_CUSTOMREQUEST => "GET",
	CURLOPT_HTTPHEADER => array(
		"x-rapidapi-host: currency-converter5.p.rapidapi.com",
		"x-rapidapi-key: 65ff423612msh81cd689a790e809p1be8a4jsn9345b94ecf93"
	),
));

$response = curl_exec($curl);
$err = curl_error($curl);

curl_close($curl);

if ($err) {
	echo "cURL Error #:" . $err;
} else {
	echo $response;
}

?>
```

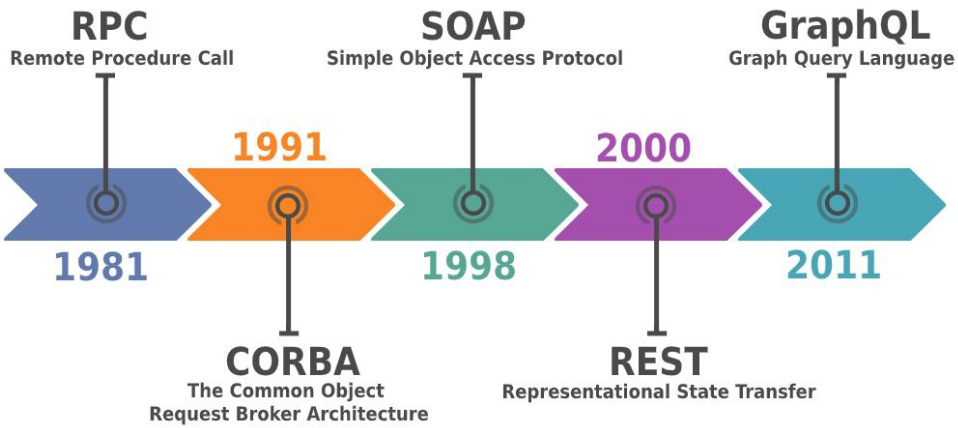
# WEB SERVICES – EXAMPLES

- **In PHP: cURL is the library that builds and dispatches HTTP requests**
- **Live demo :** http://afnan.ws/api_example/

# WEB SERVICES - BACKGROUND

- It is not a new idea
- It is closely related to the earlier concept of Remote Procedure Call (RPC).
  - RPC is used for communication in distributed systems.
  - 2 widely used RPC technologies:
    - DCOM (Distributed Component Object Model)
      - Only Microsoft software systems.
    - CORBA (Common Object Request Broker Architecture)
      - Cross-platform technology.
      - A lot of manual integration work.
  - Both are too complex, and not compatible.
    - → Do not support the goal of Web Services.

# PUBLISHING WEB SERVICES




- There are currently **three** common ways of designing/publishing web services:
  - (1) Using the **SOAP** protocol (built on top of the HTTP protocol with extra 'envelope' data).
  - (2) Using the **REST** architecture (built directly on top of the HTTP protocol)
  - (3) Newcomer -> **GraphQL** also built directly on top of HTTP protocol. Similar to REST, but with a single endpoint, querying and stronger data typing.

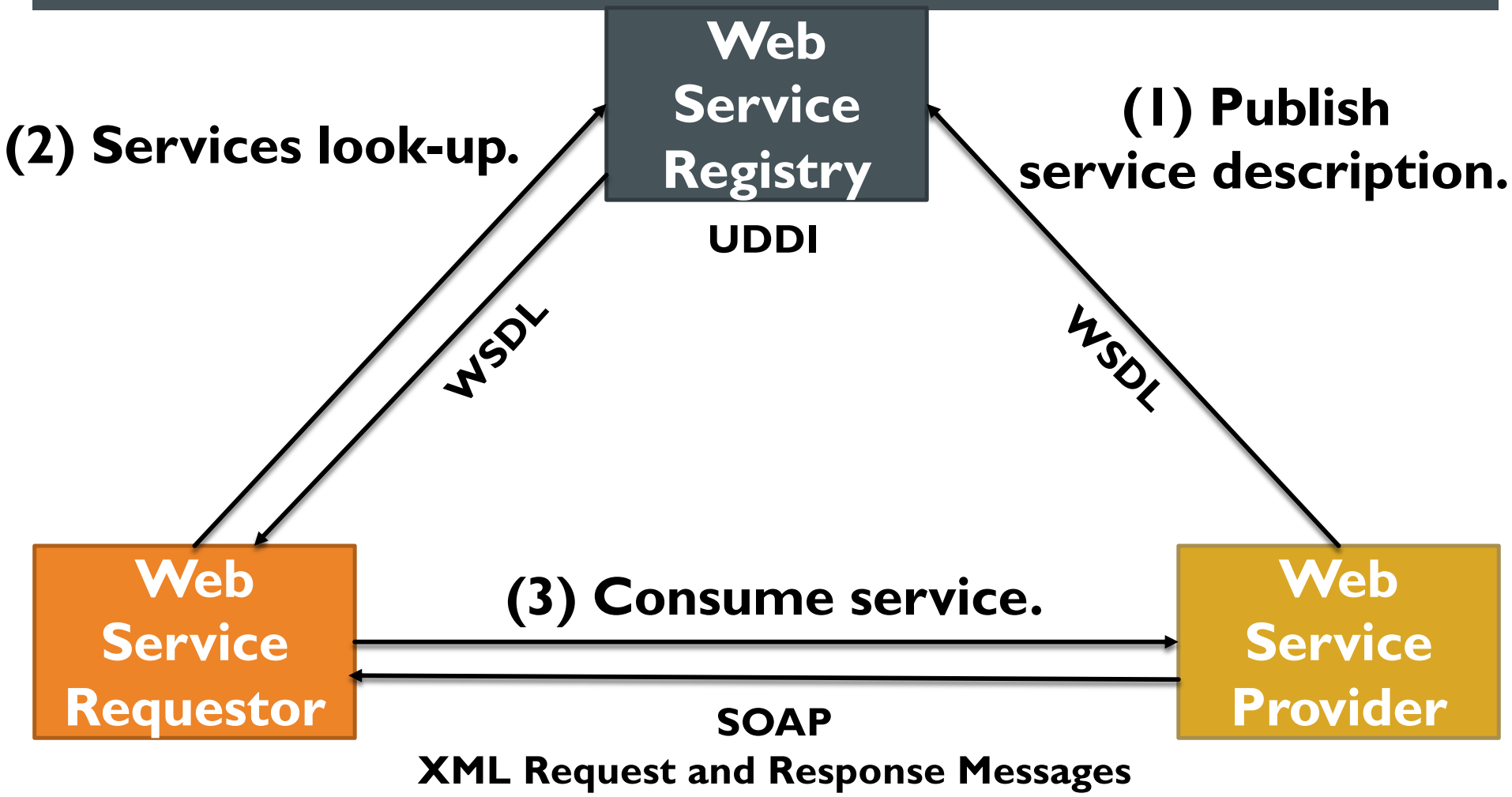
# SOAP

## WEB SERVICES

# STANDARD PROTOCOLS

- A typical approach for Web services uses:
  - SOAP (**S**imple **O**bject **A**ccess **P**rotocol)
  - WSDL (**W**eb **S**ervices **D**escription **L**anguage)
  - UDDI (**U**niversal **D**escription, **D**iscovery, and **I**ntegration)
- **Consists of three main roles:**
  - (1) Web Service Providers
  - (2) Web Service Registry
  - (3) Web Service Requesters (Consumers)

# WEB SERVICES ARCHITECTURE



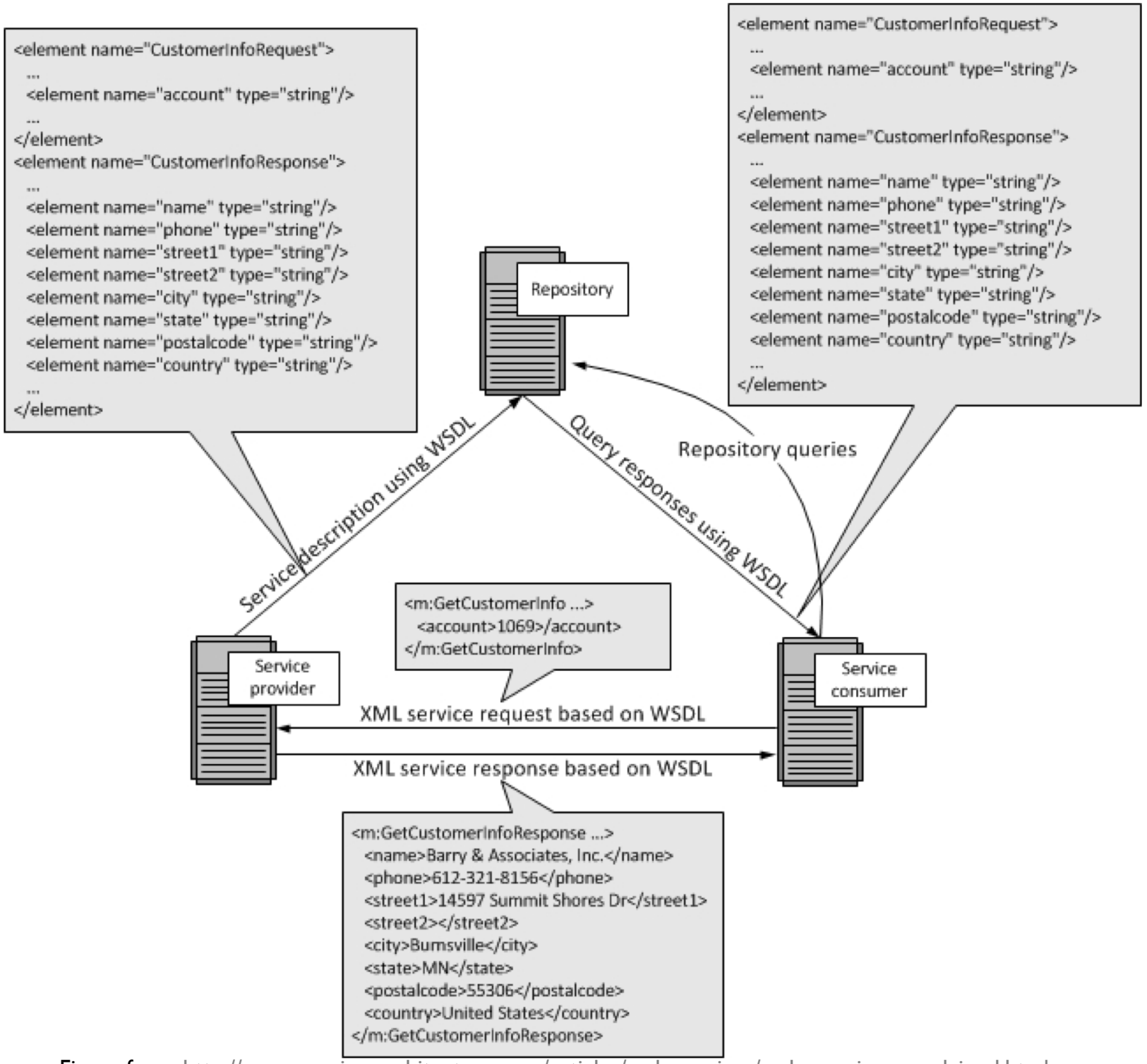



Figure from: http://www.service-architecture.com/articles/web-services/web_services_explained.html

# SOAP WEB SERVICES EXAMPLE

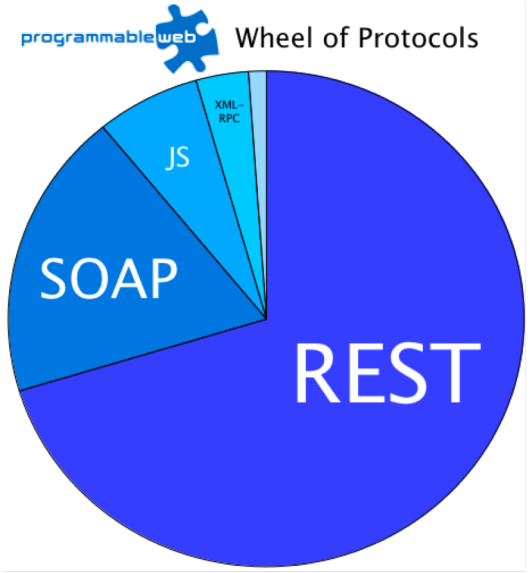

http://www.livedepartureboards.co.uk/ldbws/

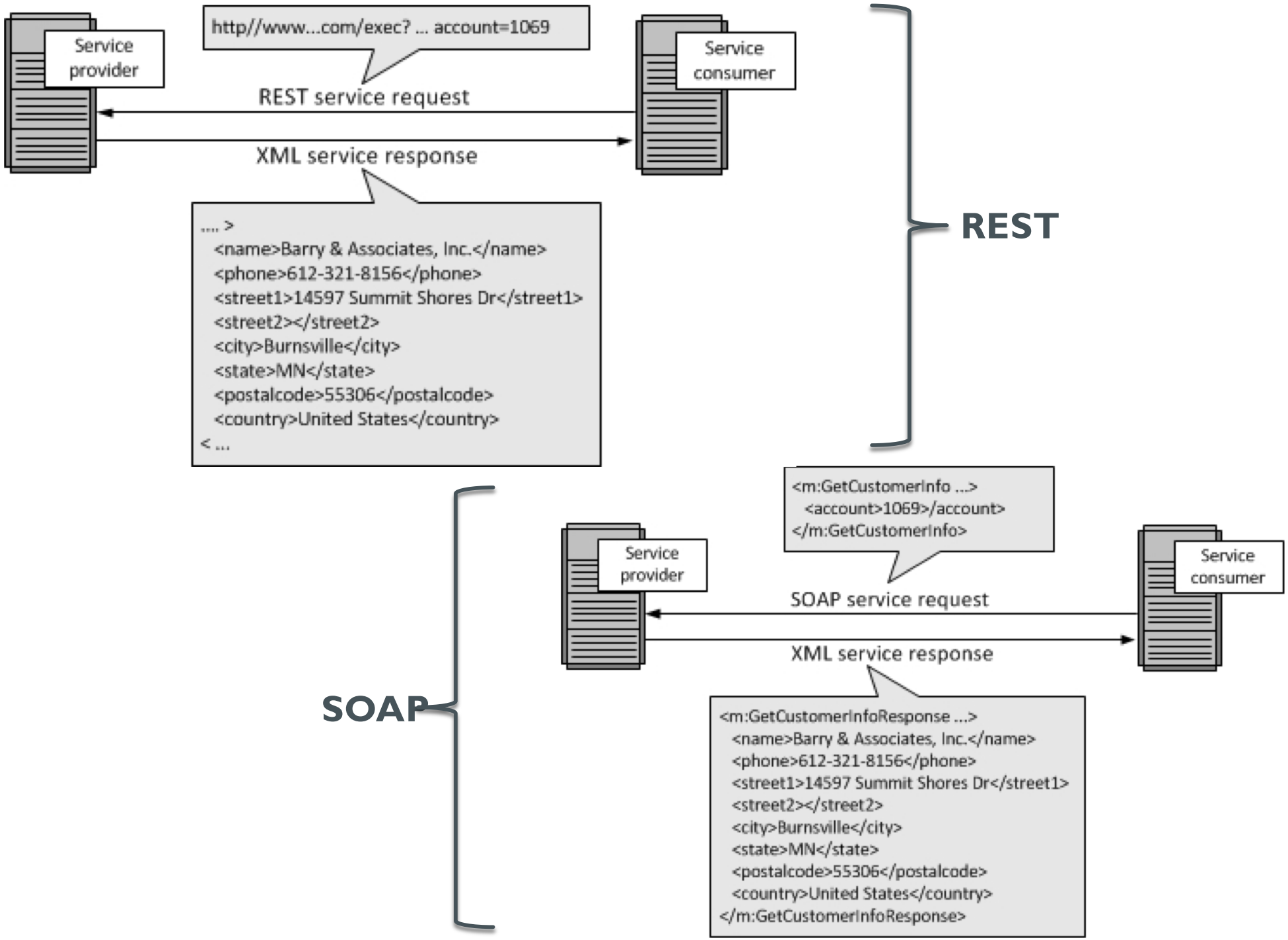
# RESTFUL

## WEB SERVICES

# ANOTHER STANDARD APPROACH FOR WEB SERVICES

- Other than SOAP,
  - Can use the **REST** concept using HTTP+XML.
  - It is the most common approach






Figures from: http://www.service-architecture.com/articles/web-services/web_services_explained.html

# WHAT IS REST?

- **What is REST?**
  - **Representational State Transfer** "REST" was coined by Roy Fielding in his Ph.D. dissertation [2] to describe a design pattern for implementing networked systems.
  - It is not a new programming language or protocol: it's only a specification of a design architecture.
  - Web services following a REST architecture are called RESTful.
  - In order for a web service to be RESTful, it follows certain constraints.

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

# REST WEB SERVICES

- RESTful web service does away with the service description layer, and needs no separate protocol for encoding message requests and responses.
- It simply uses HTTP URLs for requesting a resource/object (and for encoding input parameters).
- The serialized representation of this object, usually an XML or JSON stream, is then returned to the requestor as a normal HTTP response.
- REST appears to have almost completely displaced SOAP services.

# CONSTRAINTS ON NETWORK APPLICATIONS TO BE RESTFUL

# CONSTRAINTS ON NETWORK APPLICATIONS TO BE RESTFUL

- **The constraints of RESTful architecture are:**
  - 1) Client-Server System
  - 2) Statelessness
  - 3) Caching
  - 4) Uniformly Accessible

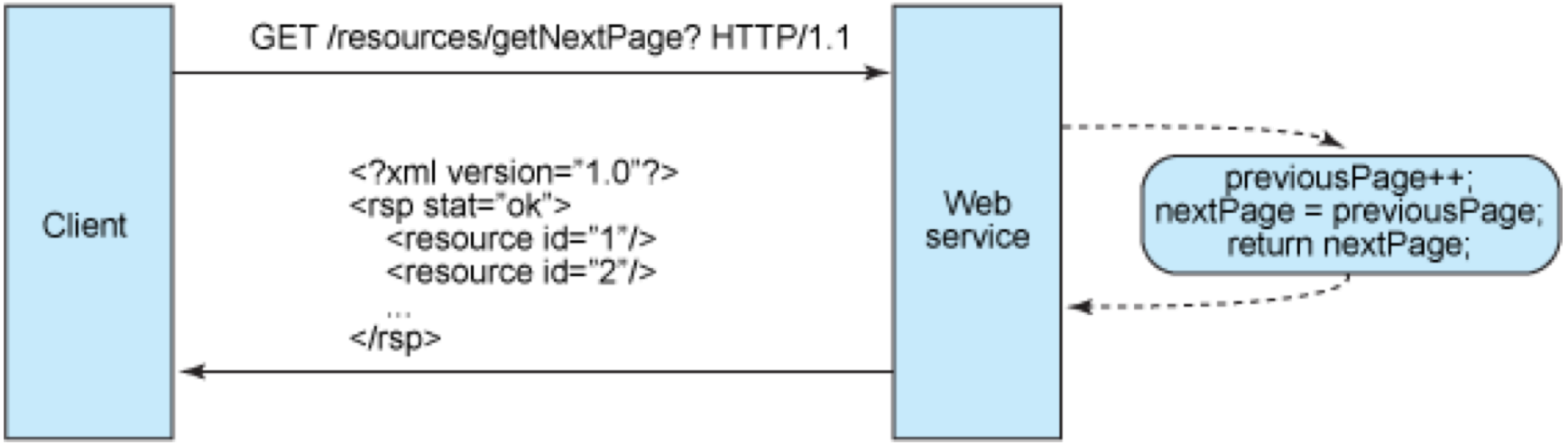
# I- CLIENT-SERVER SYSTEM

- **1) Client-Server System:**
  - Separate user interfaces from data storage.
  - Advantage: It supports scalability.

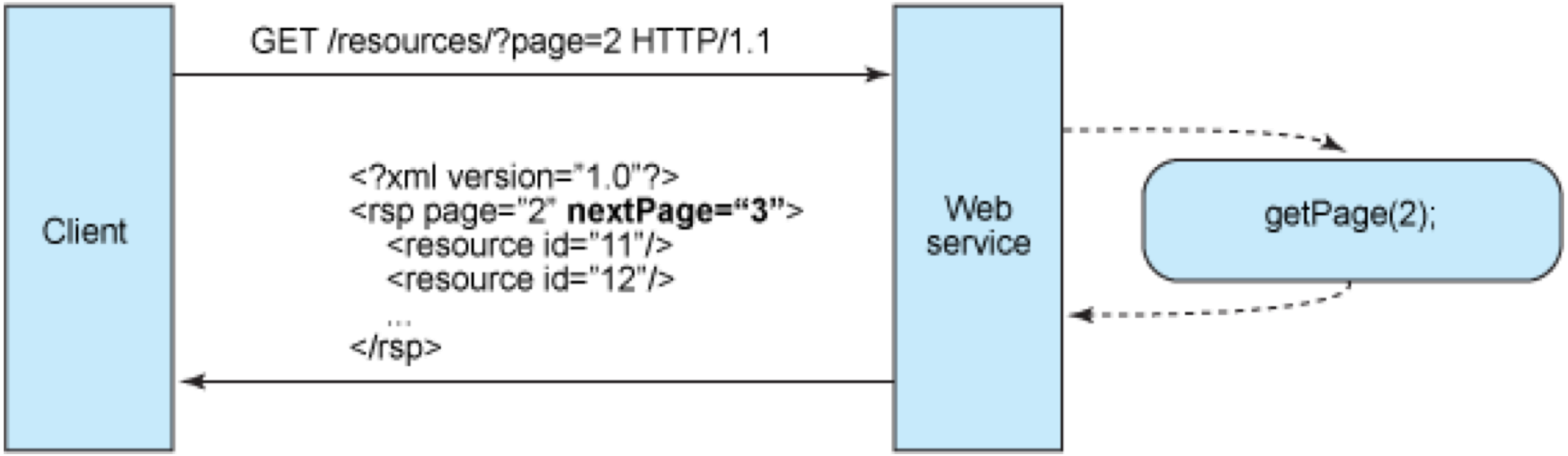
# 2- STATELESSNESS

- **2) Statelessness:**
  - Every HTTP request should happen in complete isolation.
  - It means that when the client makes an HTTP request all the information required to process that request must be present.
  - Advantages: It improves reliability and scalability.

1- Session




2- Stateless communication




1- http://www.ibm.com/developerworks/webservices/library/ws-restful/figure1.gif

2- http://www.ibm.com/developerworks/webservices/library/ws-restful/figure2.gif

# 3- CACHING SYSTEM

- **3) Caching System:**
  - The resources should be cacheable whenever possible with an expiration date and time (less load on the server).

# 4- UNIFORMLY ACCESSIBLE

- **Uniformly Accessible:**
  - General verbs that simplify and decouple the communication between the client and the server.

# UNIFORMLY ACCESSIBLE (CONT.)

- For example : The client interacts via the defined HTTP methods: GET, POST, PUT, DELETE. These methods define the things that can be done to a resource.

| HTTP method | CRUD Action |
|---|---|
| GET | Retrieve resource |
| POST | Create resource |
| PUT | Update resource |
| DELETE | Delete resource |

# REST ARCHITECTURAL ELEMENTS

# 1-RESOURCES AND RESOURCE IDENTIFIERS

- Everything that a service provides is a resource (customers, pictures, invoices, etc.).
- Every Resource will have at least one URI (a unique id).
- If a piece of information does not have a URI, it is not a resource and it is not on the Web.
- Two resources cannot have the same URI

# 2- REPRESENTATIONS

- A representation is a sequence of bytes, plus representation metadata to describe those bytes.
- Other commonly used but less precise names for a representation include: document, file, and HTTP message entity.
- A resource representation in a response (e.g. XML or JSON) can be different than the internal representation in server.

# RESTFUL WS IS DEFINED BY:

- The base URI for the web service, such as http://example.com/resources/
- The Internet media type of the data supported by the web service. This is often JSON or XML but can be any other valid Internet media type.
- The set of operations supported by the web service using HTTP methods (e.g., GET, PUT, POST, or DELETE).

# REST ADVANTAGES

- Much like Web Services, a REST service is:
  - Platform-independent (you don't care if the server is Unix, the client is a Mac, or anything else),
  - Language-independent (PHP can talk to Java, etc.),
  - Standards-based (runs on top of HTTP).

# WEB SERVICE CLIENT

- Steps to call a web service in your application:
  - –Open up a TCP/IP socket connection.
  - –Then send something, i.e. HTTP request.
  - –The web service sends back something.
  - –Finally, close the socket connection.
- To implement this, you need HTTP Libraries in your programming language (e.g. in Java: *java.net.HttpURLConnection or* Jakarta Commons HTTP Client or Jersey client*)*.

# WEB SERVICE PROVIDER

RESTful web service design principles can be summarized in the following:

- Requirements gathering.
- Resource identification.
- Resource representation definition.
- URI definition.

# AN EXAMPLE WEB SERVICE

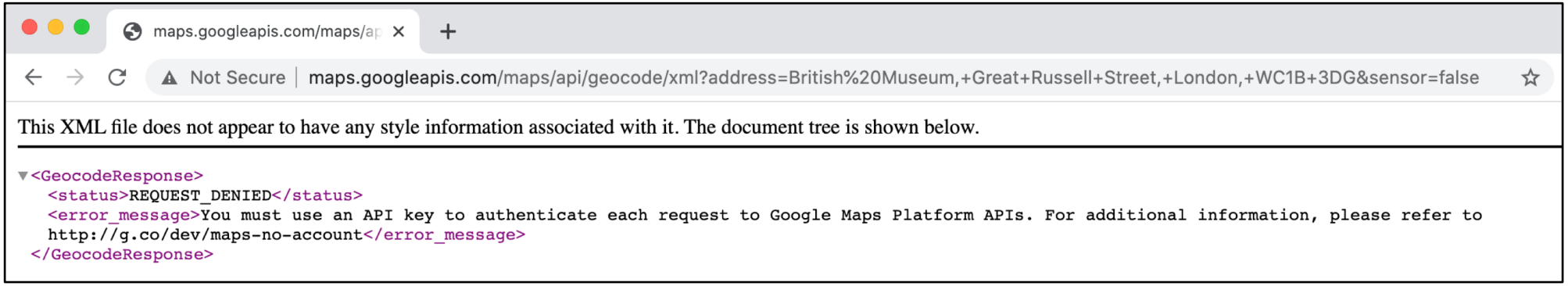
- Consider the Google Geocoding API.
- The Google Geocoding API provides a way to perform geocoding operations via an HTTP GET request, and thus is an especially useful example of a RESTful web service.
- **Geocoding** typically refers to the process of turning a real-world address into geographic coordinates, which are usually latitude and longitude values
- The request will take the following form:

  http://maps.googleapis.com/maps/api/geocode/xml?***address***

# AN EXAMPLE WEB SERVICE

- An example geocode request would look like the following:

  http://maps.googleapis.com/maps/api/geocode/xml?address=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG&sensor=false



# IDENTIFYING AND AUTHENTICATING SERVICE REQUESTS

- Most web services are not open. Instead they typically employ one of the following techniques:
  - **Identity**. Each web service request must identify who is making the request.
  - **Authentication**. Each web service request must provide additional evidence that they are who they say they are.

# IDENTITY EXAMPLES

- Web services that make use of an API key typically require the user (i.e., the developer) to register online with the service for an API key. This API key is then added to the GET request as a query string parameter.

- For instance, to request to the Microsoft Bing Maps web service will look like the following :

  http://dev.virtualearth.net/REST/v1/Locations?o=xml&query=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG,+UK&**key=[*BING API KEY HERE*]**

# AUTHENTICATION

- Some web services are providing private/proprietary information or are involving financial transactions.
- In this case, these services not only may require an API key, but they also require some type of user name and password in order to perform an authorization.
- Many of the most well-known web services instead make use of the OAuth standard.

# CHALLENGES OF USING WEB APIS

- Data Integration.
  - Combining data from different sources with different formats.
- API Stability.
  - Same behavior and data for services.
  - Continuous support for services.

# SUMMARY

- Web service is a software system designed to support interoperable machine-to-machine interaction over a network.
- WS will help more technologies of distributed architecture to collaborate.
- W3C has defined standards for SOAP web services, contrary to REST web services.
- Web Services is not a new hype but a trend to follow.
- The future will be Web *of* Services.

# REFERENCES


- "Programming the World Wide Web" by Robert W. Sebesta. (Sections 7.11 and 12.6)
- “Fundamentals of Web Development” by Randy Connolly and Ricardo Hoar (Sections 19.4, 19.5, 19.6, and 19.7)
- http://docs.oracle.com/cd/E19776-01/820-4867/ggnyk/index.html
- http://en.wikipedia.org/wiki/Web_service
- http://gdp.globus.org/gt4-tutorial/multiplehtml/ch01s02.html
- http://download.oracle.com/docs/cd/E19776-01/820-4867/index.html
- http://www.vogella.de/articles/REST/article.html