

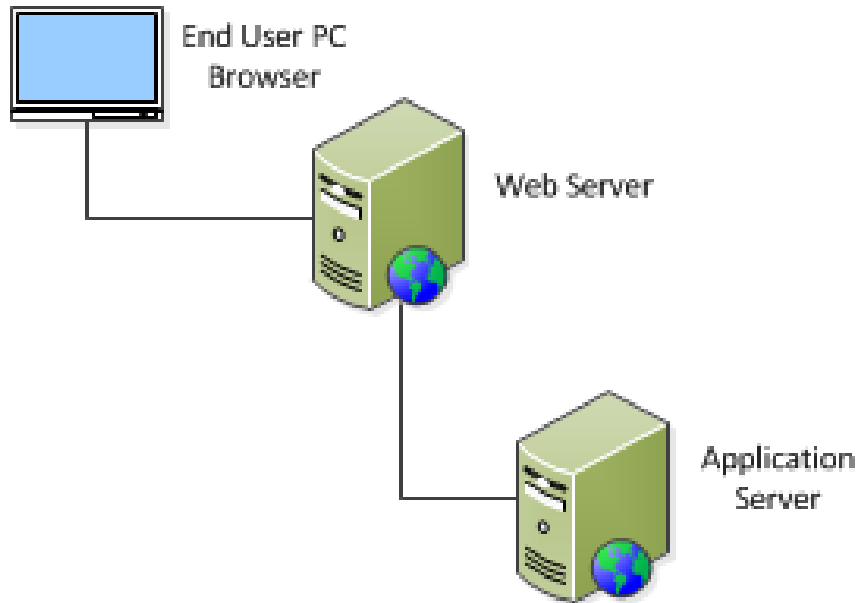
## Agenda

- Web Services Overview
- Protocols
  - http
  - xml
  - json
  - api
  - REST
  - SOAP
- **Some code examples**

# *Application Architecture*

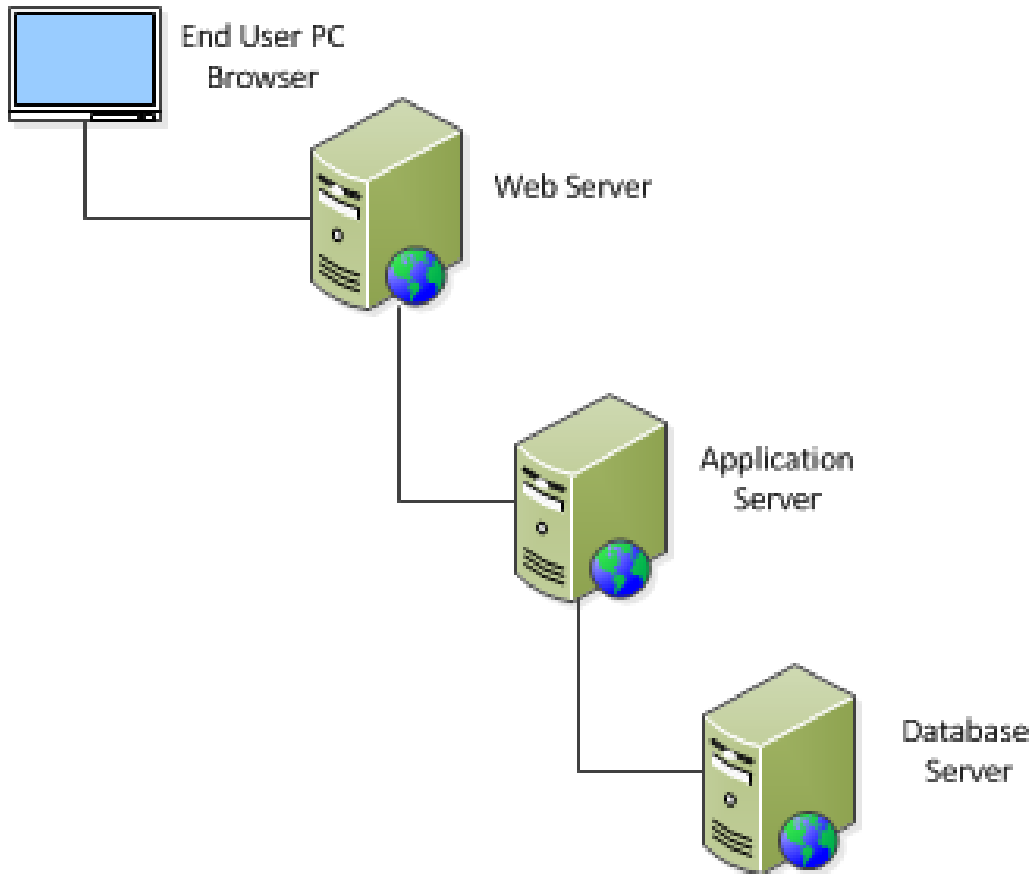
---

- **Architecture Layers**



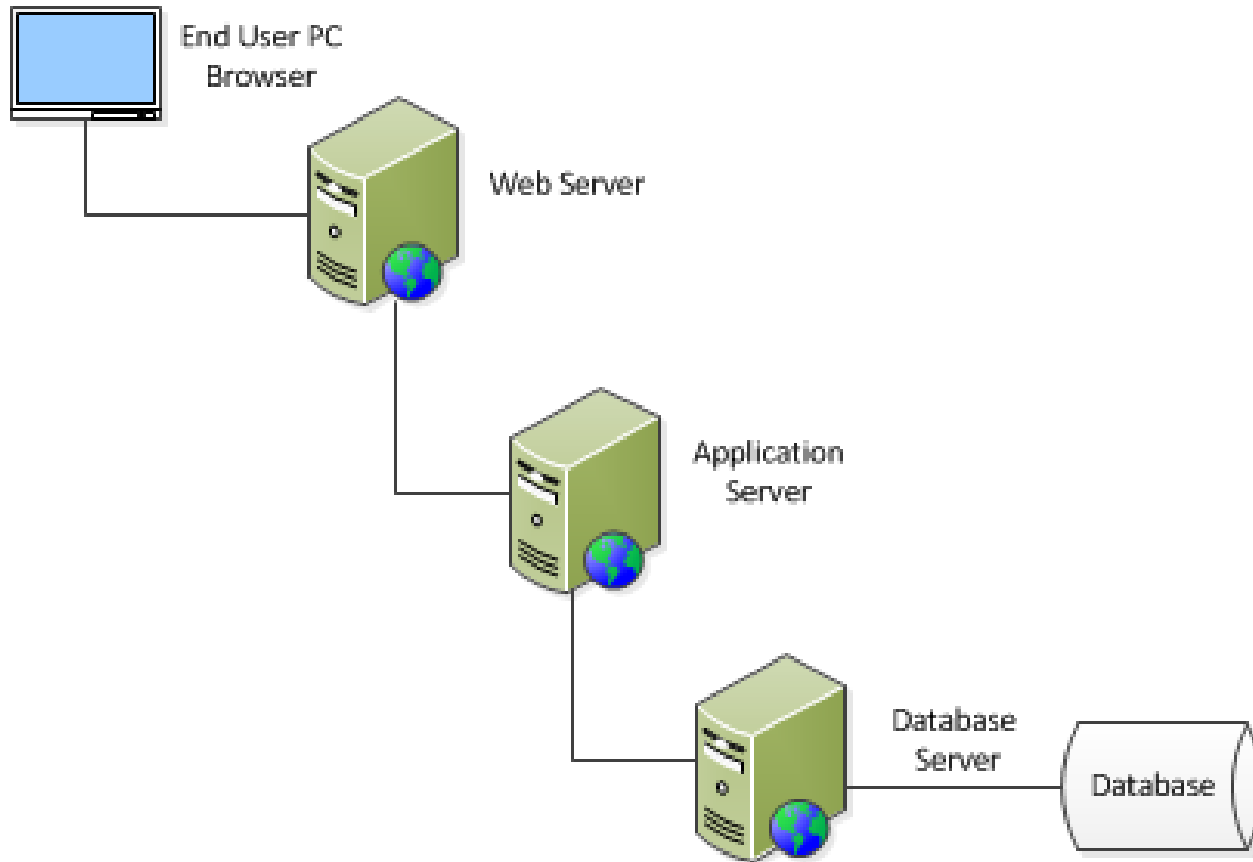
# *Application Architecture*

---

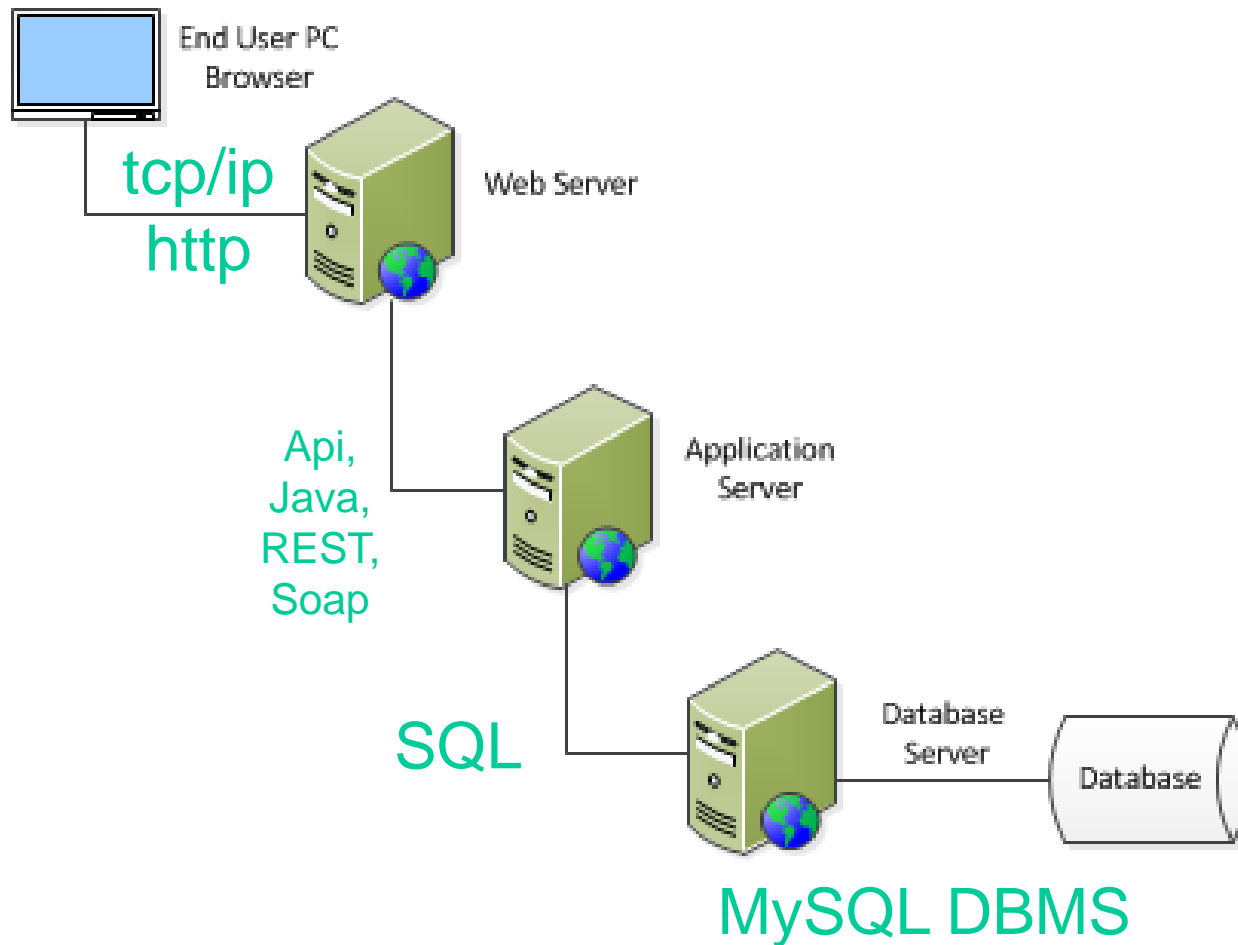


# *Application Architecture*

---



# *Application Architecture*

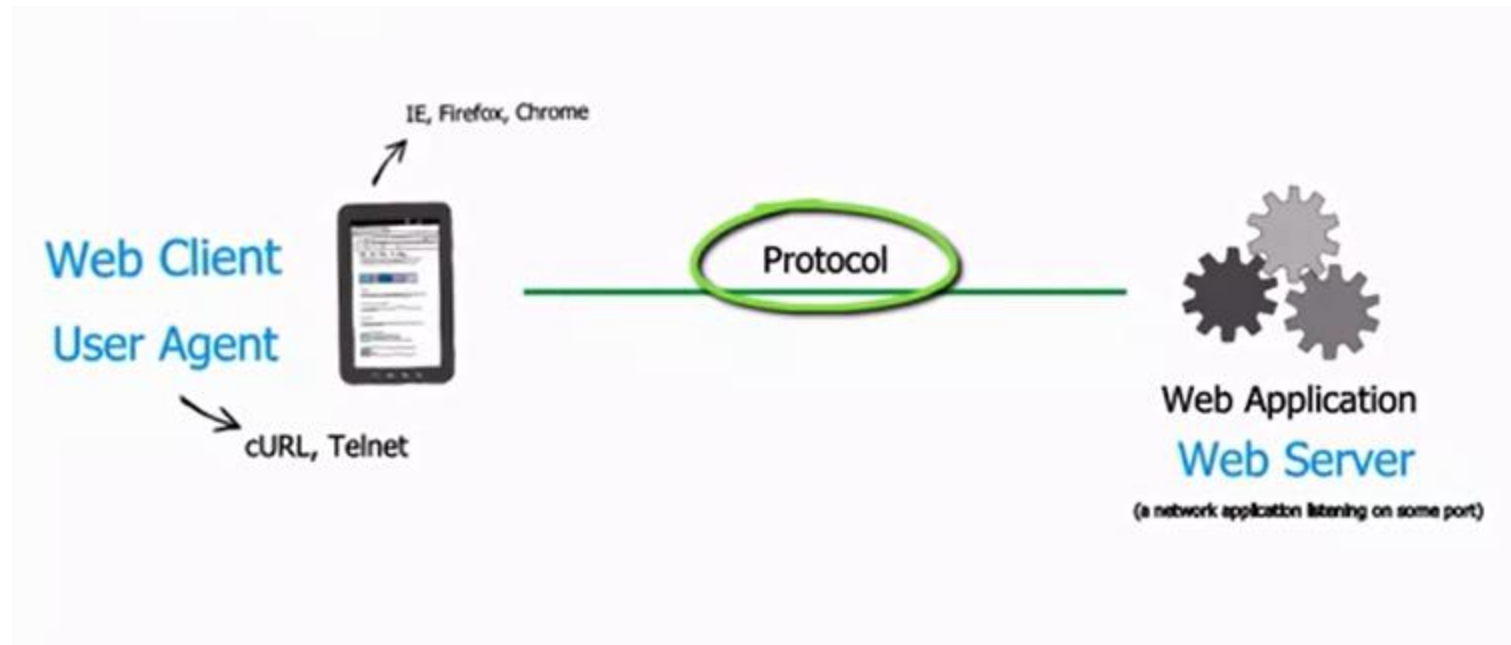


- How do we pass messages and requests from one layer to another?

PROTOCOLS !

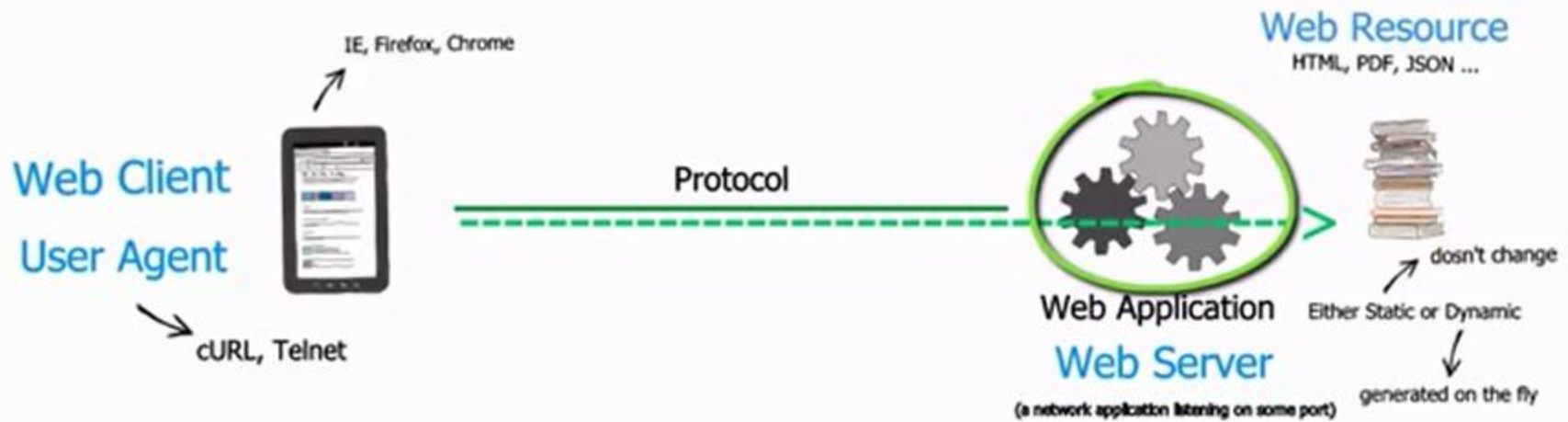
- Internet Protocols
- What happens when you type a URL into a browser and press <ENTER>?
- What happens when you click on a hyperlink in a web page?

# Protocols

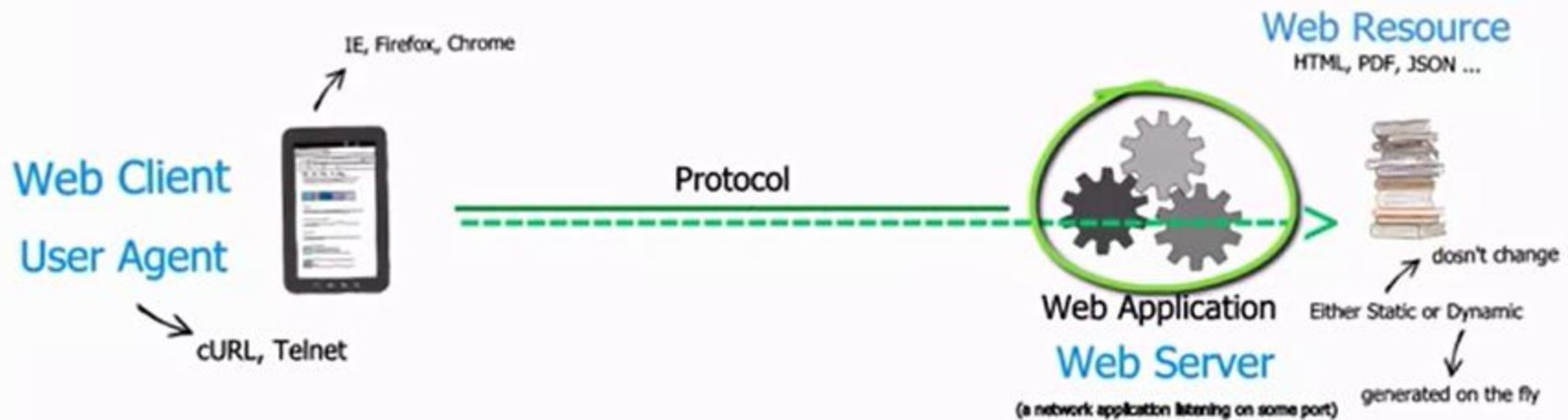




# Protocols



# Protocols



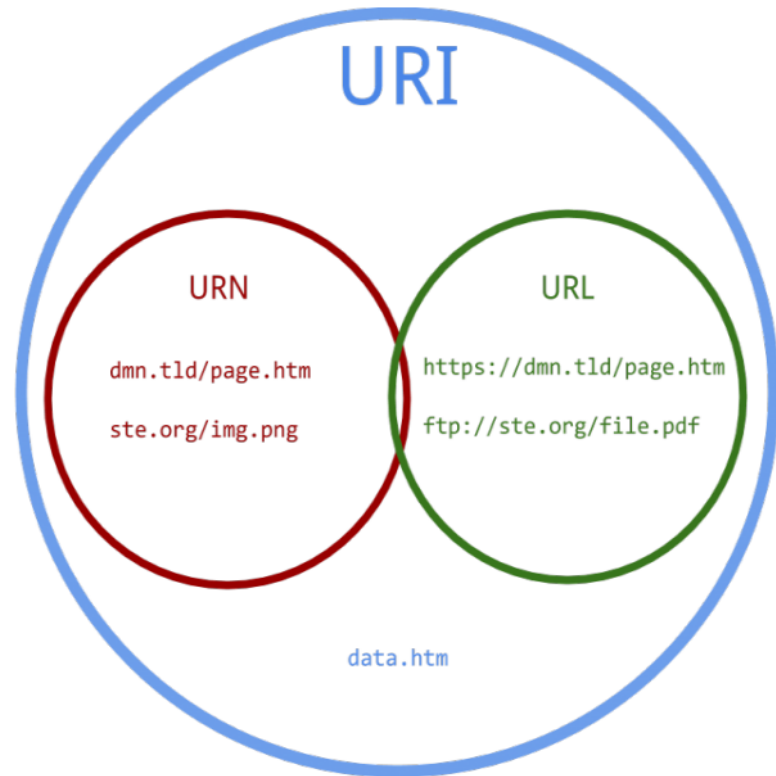
Each web resource is identified by a URI

- The URI
  - <http://www.colorado.edu>
  - URI, URL, URN ?

URL = locator  
(where/how to find it)

URN = name  
(what is its name)

URI = either one



- **HTTP** – a request/response protocol
  - It is STATELESS
  - The client submits a request, HTTP responds with the requested resource and a return code
    - Resources may be static or dynamic
    - Resources may redirect, include other resources, etc.
- **HTTP Methods**
  - GET       Retrieves the URI
  - POST       Submits a resource to the URI  
              Like submitting a FORM to be processed by a script
  - PUT        Stores a resource under the URI
  - DELETE   Deletes the URI

- **Common HTTP return codes**

- 200 : OK
- 302 : Redirect
- 400 : Bad Request
- 401 : Unauthorized
- 403 : Forbidden
- 404 : Not Found
- 500 : Server Error

- **Passing data to/from the web server**
- XML          Extensible Markup Language
- JSON          Java Script Object Notation

## XML Extensible Markup Language

- “Tag” based, like HTML
- Tags are user-defined
- XML is human readable AND machine readable
- Tags describe the data (XML tags do NOT display the data like HTML tags do)

```
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price> </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price> </book>
  <book category="web">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <author>Per Bothner</author>
    <author>Kurt Cagle</author>
    <author>James Linn</author>
    <author>Vaidyanathan Nagarajan</author>
    <year>2003</year>
    <price>49.99</price> </book>
  <book category="web" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price> </book>
</bookstore>
```



## Retrieving the data from an XML document

```
txt = xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
```

`xmlDoc` - the XML DOM object created by the parser.

`getElementsByTagName("title")[0]` - get the first <title> element

`childNodes[0]` - the first child of the <title> element (the text node)

`nodeValue` - the value of the node (the text itself)

### Example:

[https://www.w3schools.com/xml/dom\\_intro.asp](https://www.w3schools.com/xml/dom_intro.asp)

- Represents data in key:value pair format.
- Many folks think JSON is easier to use than XML
- More compact than XML
- Like XML, JSON is easy for both humans & computers to understand

## JSON:

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

## 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 is Like XML Because

- Both JSON and XML are "self describing" (human readable)
- Both JSON and XML are hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest

## JSON is Unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

For AJAX applications, JSON is faster and easier than XML:

## Using XML

- Fetch an XML document
- Use the XML DOM to loop through the document
- Extract values and store in variables

## Using JSON

- Fetch a JSON string
- `JSON.Parse` the JSON string

OK. So what is AJAX?

**AJAX** stands for Asynchronous JavaScript and XML.

**AJAX** is a technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

**Ajax** uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display. (JavaScript runs on the CLIENT)

Example AJAX versus CGI (Common Gateway Interface)

[https://www.tutorialspoint.com/ajax/ajax\\_examples.htm](https://www.tutorialspoint.com/ajax/ajax_examples.htm)

## **Early Web (CGI) 1989**

- hypertext / hyperlinks
- page by page

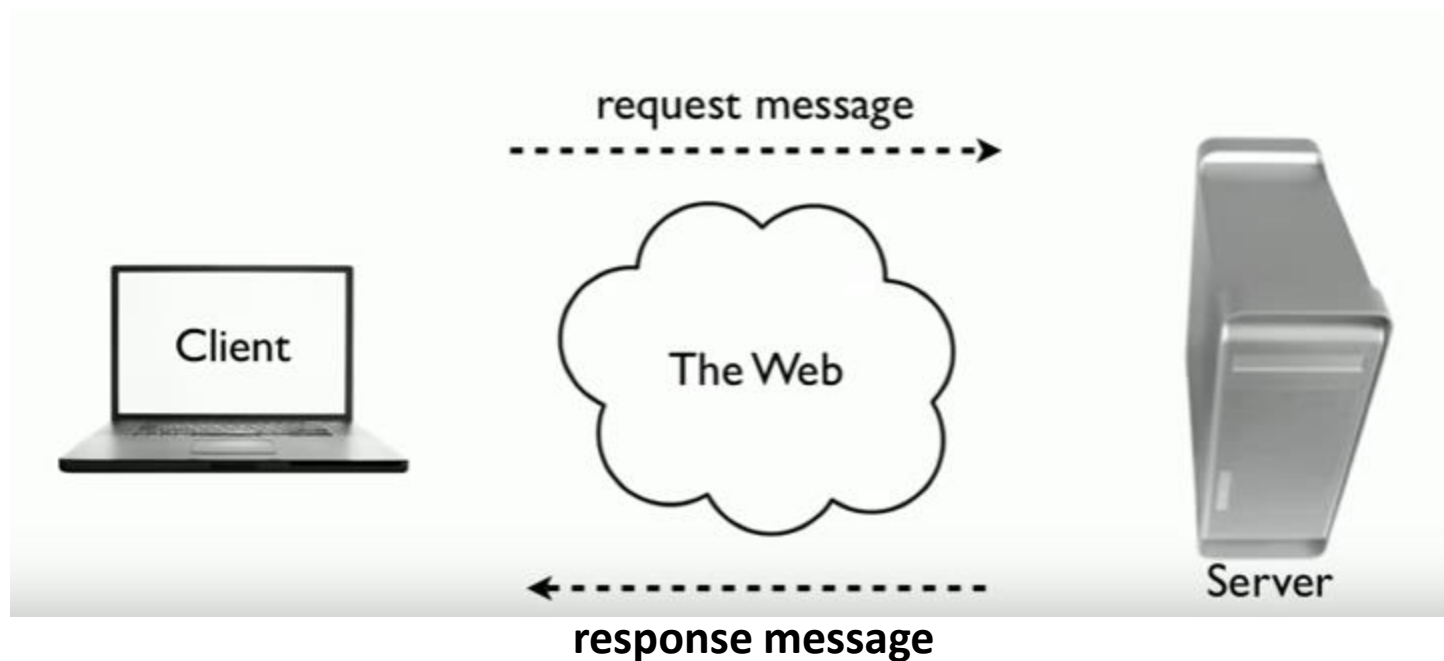
## **Web 2.0 (AJAX) 2004**

- web page stays in place
- parts of the web page are updated

**How are web 2.0 requests handled between client and server?**

**Web Services !**

**A framework for a conversation between computers over the web**





**If you want to use a web service, you must use an API (application programming interface)**

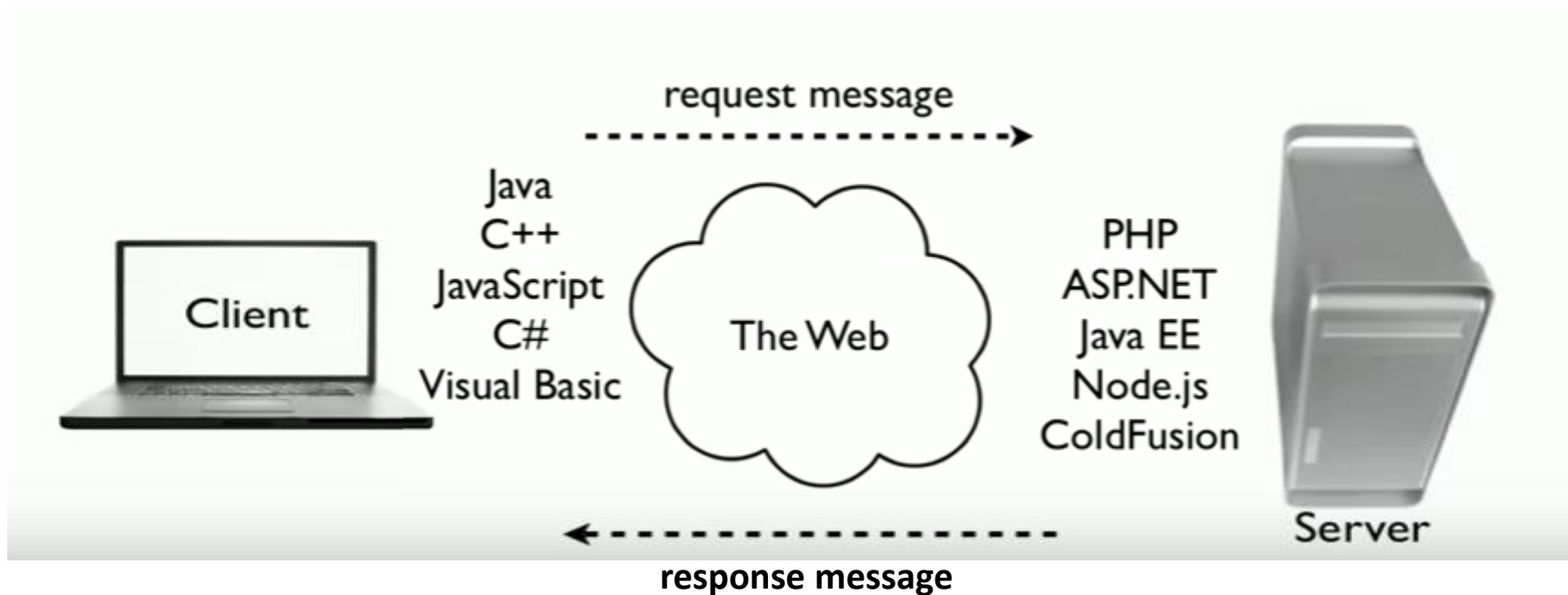
**Defines everything you need to know to talk to a web service:**

- 1. Message format: SOAP, XML, JSON, etc.**
- 2. Request syntax: URI, Parameters & Data types**
- 3. Actions on the server: named methods, HTTP verbs**
- 4. Security: authentication (username & password)**
- 5. Response format: SOAP, XML, JSON, etc.**

The web service hides its complexity behind the API

# Web Services

The web service hides its complexity behind the API



**RE**presentative

**S**tate

**T**ransfer

**REST**  
is an *architectural style*

**Modern Architectural Style:**



**Colonial Architectural Style:**



**The “architectural style” is an abstract concept  
it defines the characteristics and features you would  
find in a house built according to that style**

**It is NOT the same as the house itself.**

**REST is an abstract concept that defines the  
characteristics and features you would find in a web  
service request built according to the REST style**

- Everything in REST is considered as a resource.
  - Every resource is identified by an URI.
  - Uses uniform interfaces. Resources are handled using POST, GET, PUT, DELETE operations
  - Stateless. Every request is an independent request. Each request from client to server must contain all the information necessary to understand the request.

## Web Services

- RESTFulweb services are based on HTTP methods
- a RESTFulweb service typically defines the base URI for the services, the supported MIME-types (XML, text, JSON, user-defined, ...) and the set of operations (POST, GET, PUT, DELETE) which are supported.

## **Characteristics of a request/response following the REST protocol**

### **Resources follow the rules**

**URI** (identifies the resource being requested)

**Uniform Interface Methods** (GET, PUT, POST, etc.)

**Uniform Interface Representation** (XML, JSON, HTML)

### **Protocols offer features**

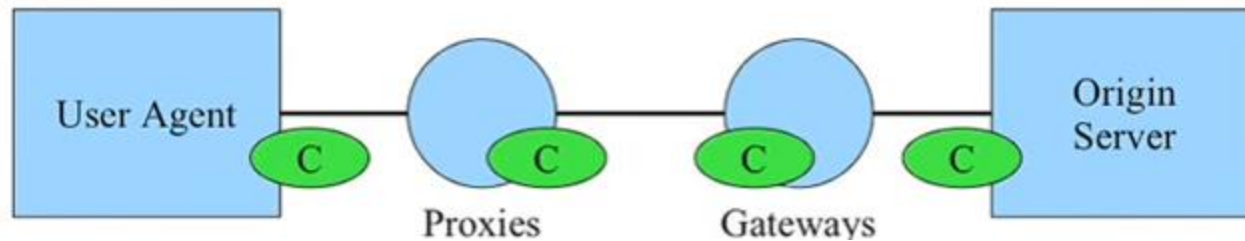
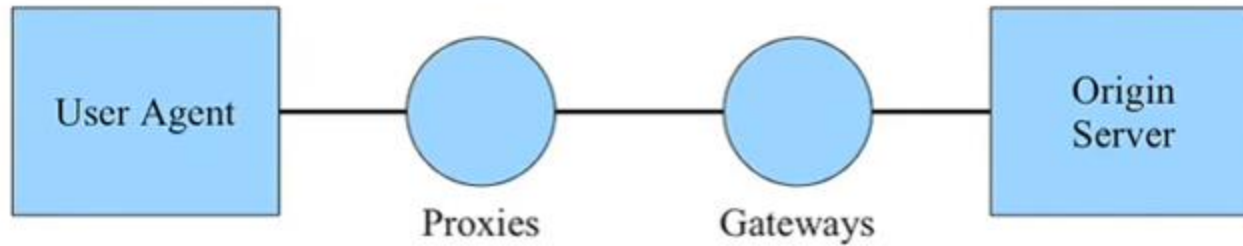
**Client-Server** (like HTTP)

**Stateless** (each request is independent)

**Layered** (may pass through intermediaries)

**Cacheable** (intermediaries may cache for performance)

# Web Services





## **Advantages of a request/response following the REST protocol**

- **Efficiency**  
(through caching & compression)
- **Scalability**  
(gateways distribute traffic, caching, statelessness allows different intermediaries)
- **User Perceived Performance**  
(code on demand, client validation, caching)
- **Simplicity**

**S**imple  
**O**bject  
**A**ccess  
**P**rotocol

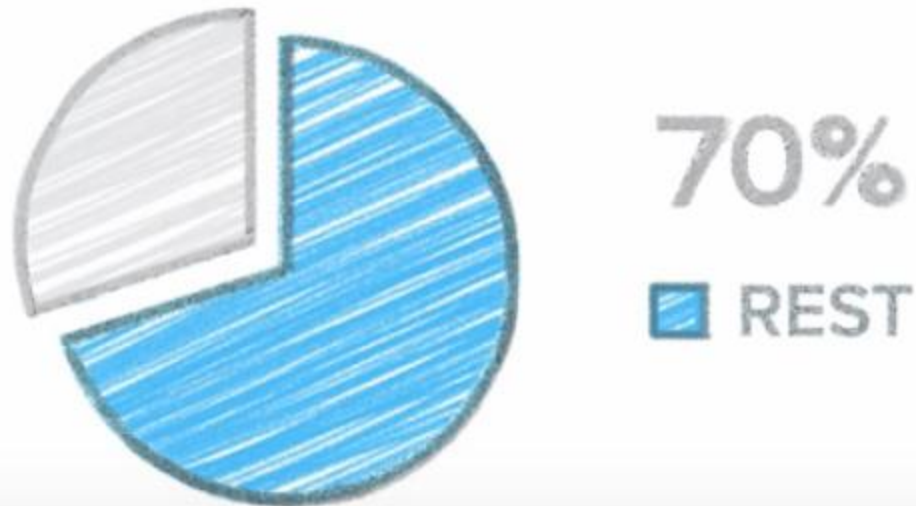
## SOAP -- Simple Object Access Protocol

REST	SOAP
Representational State Transfer	Simple Object Access Protocol
Architecture Style	An actual protocol
Uses simple HTTP	Uses SOAP envelope, then HTTP (or FTP, or other) to transfer the data
Uses many different data formats like JSON, XML, YAML*	Supports only XML format
Performance & Scalability & Caching	Slower performance. Scalability is limited and complex. Caching is not possible.
Widely and frequently used	Used where REST is not possible

\*YAML: YAML Ain't Markup Language

What It Is: YAML is a human friendly data serialization standard for all programming languages.

## public APIs



- WSDL (Web Service Description Language) is an XML document that defines contract between client and service and is static by its nature.
- SOAP builds an XML based protocol on top of HTTP or some other protocol according to the rules described in the WSDL for that Web Service.

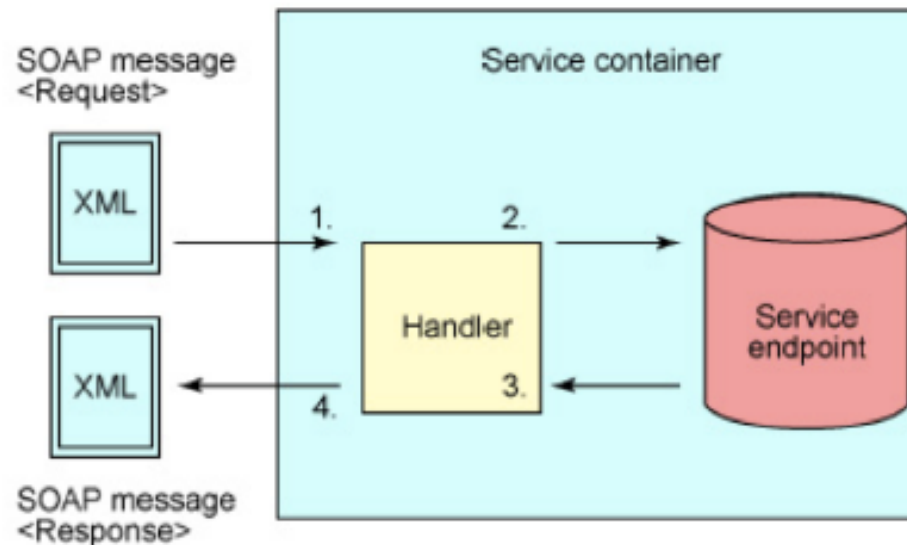
## SOAP

- ◆ A SOAP message is an XML document containing the following elements:
  - An **Envelope** element that identifies the XML document as a SOAP message
  - A **Header** element that contains header information
  - A **Body** element that contains call and response information
  - A **Fault** element containing errors and status information

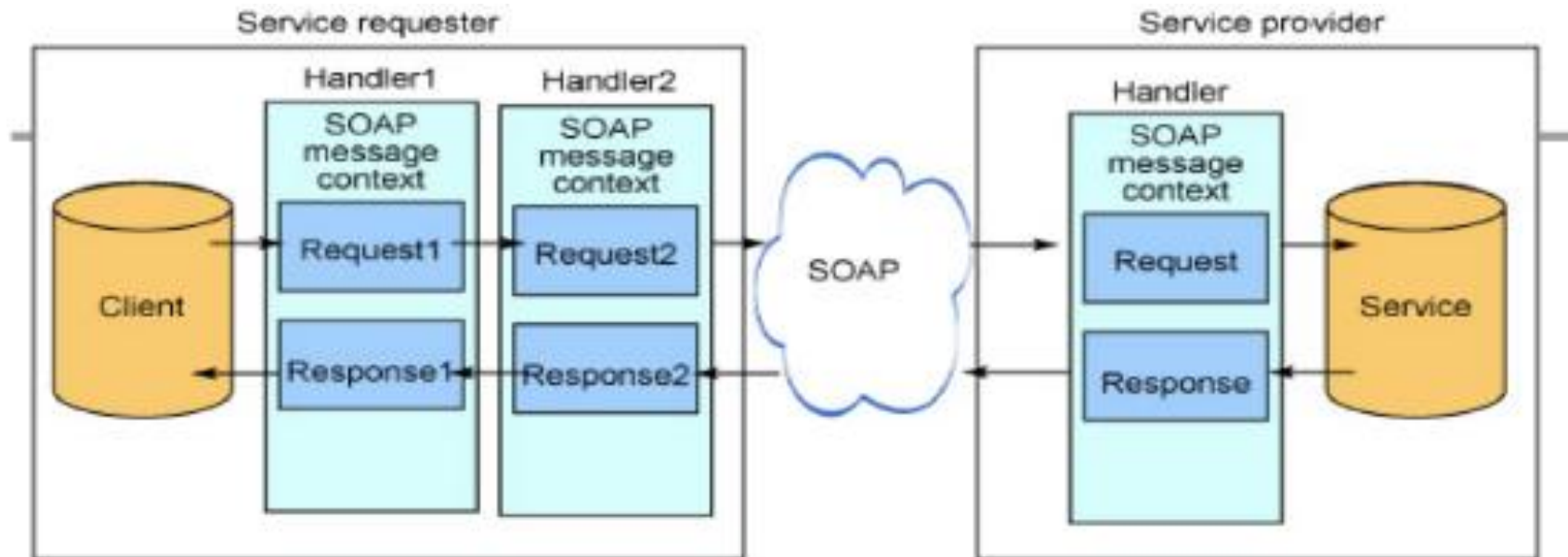


## SOAP Handlers

- ◆ **Handlers** are pluggable classes that can be associated with a Web service or Web service client to provide pre-processing or post-processing of XML messages.
  - Ex: logging XML traffic through a Web service
  - Ex: measure performance by accessing the SOAP header to insert initial and finish times between two checkpoints



# Web Services





[http://www.w3schools.com/graphics/google\\_maps\\_basic.asp](http://www.w3schools.com/graphics/google_maps_basic.asp)

Demo of an API for using a web service

<http://apigee.com/console/>

A list of published web services and APIs

<https://www.youtube.com/watch?v=7YcW25PHnAA>

- This video shows a clear example of how we can use REST framework API for using a web service

<https://www.youtube.com/watch?v=RTjd1nwvlj4>

- This video shows a clear example of how we can code php to do a RESTful php program to send a form to a web service and process its response.  
(He goes really fast. You should pause often to let it sink in.)