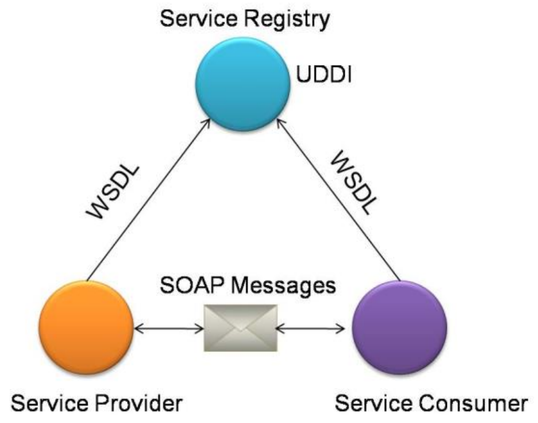
SWE 363: Web Engineering & Development

Module 8-3

Web Services



Objectives

- Learn about Web Services:
  - What,
  - Why and
  - How?

Outline

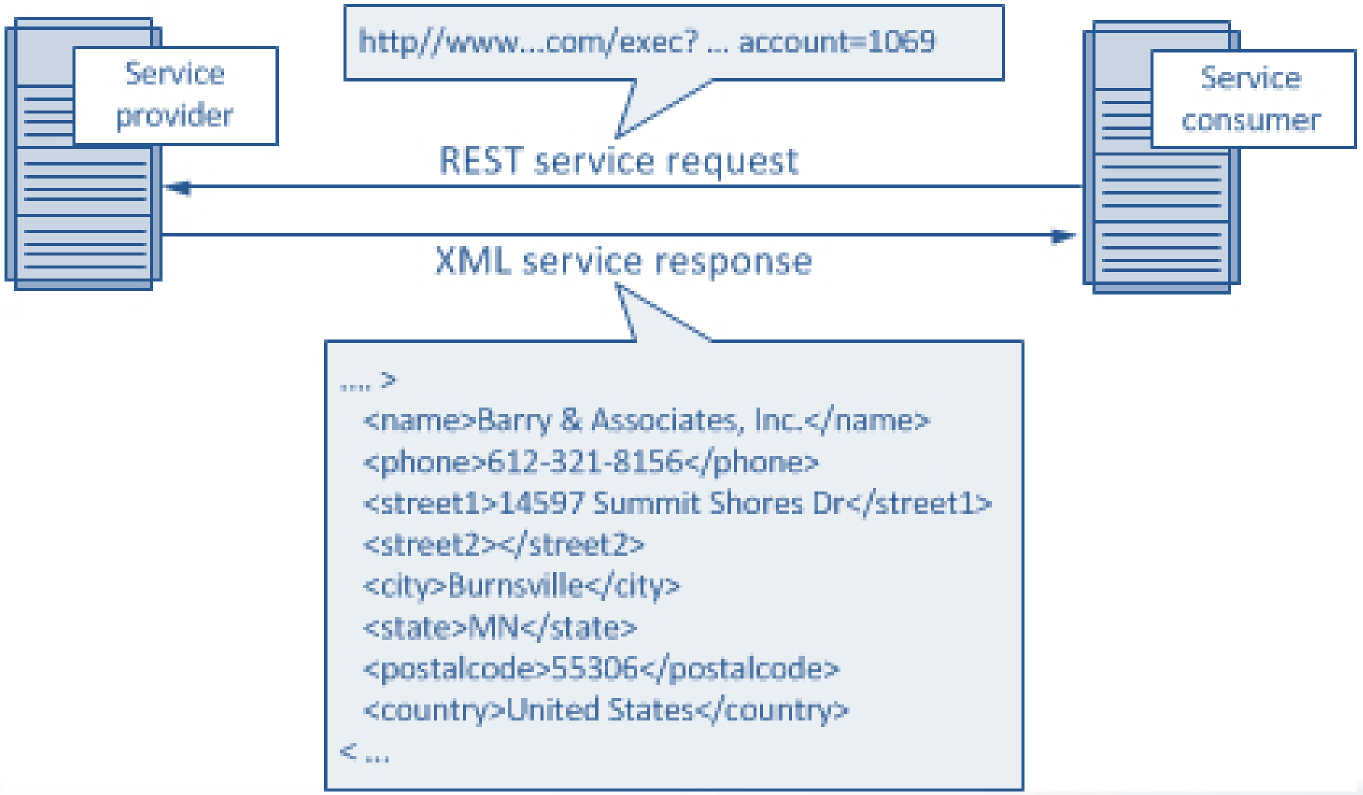
- Overview of web services
- How they work?
- Benefits of Web Services
- Examples of Web Services
- SOAP web services
- REST web services
- PHP Example

References

- "Internet & World Wide Web: How to Program 5th editions"
- "Fundamentals of Web Development" Book by Randy Connolly and Ricardo Hoar, 2015
- W3schools: https://www.w3schools.com/

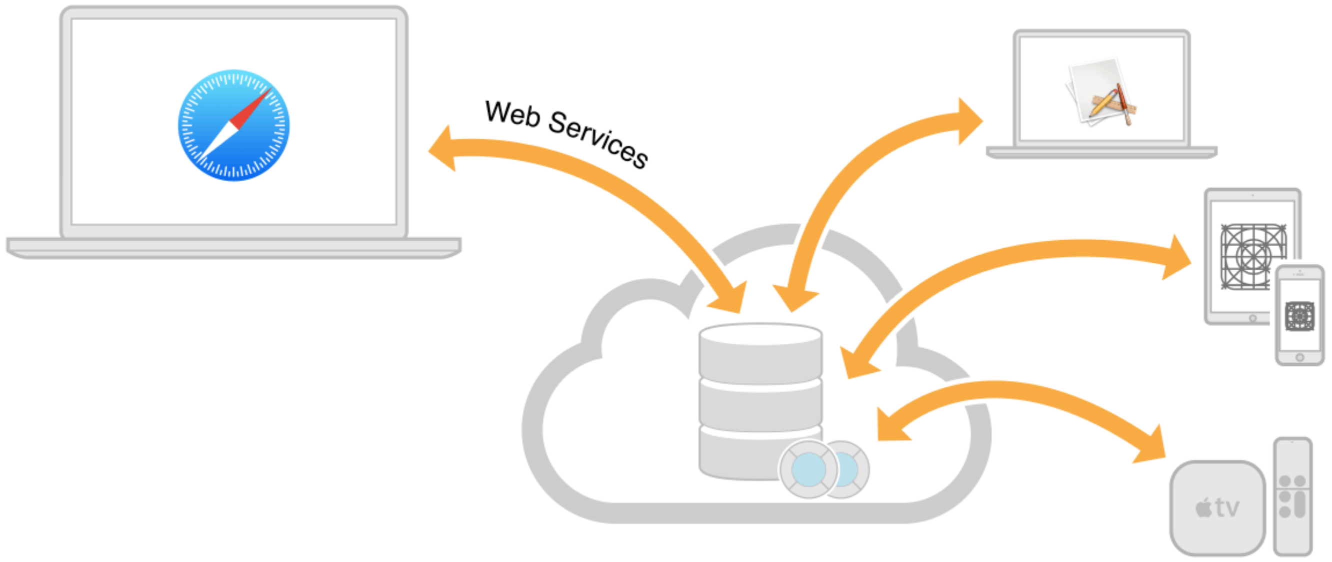
Introduction

- A service is a piece of software with a *platform-independent interface* that can be dynamically located and invoked.



```
.... >
  <name>Barry & Associates, Inc.</name>
  <phone>612-321-8156</phone>
  <street1>14597 Summit Shores Dr</street1>
  <street2></street2>
  <city>Burnsville</city>
  <state>MN</state>
  <postalcode>55306</postalcode>
  <country>United States</country>
< ...
```

Introduction..



- A web service is a software system designed to support inter-operable machine-to-machine interaction over a network. [W3C definition]

- A web service is any service that:
  - Is available over the Internet or private (intranet) networks
  - Is not tied to any operating system or programming language (Cross-platform and Platform-independent)

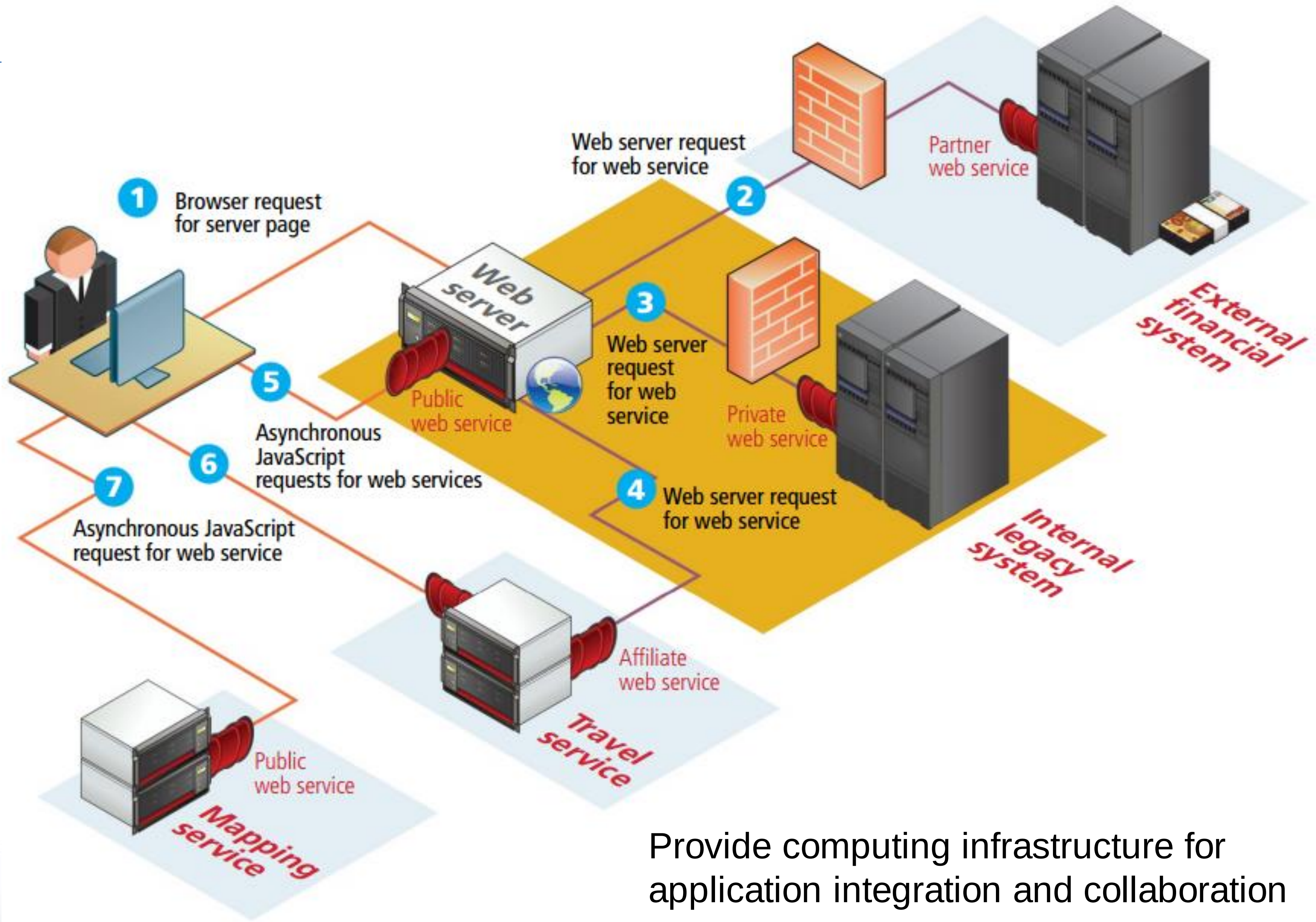
Overview of web services

- Web services are a relatively standardized mechanism by which one software application can connect to and communicate with another software application using web protocols.
  - Web services make use of the HTTP protocol so that they can be used by any computer with Internet connectivity.

- Web services typically use XML or JSON to encode data within HTTP transmissions so that almost any platform should be able to encode or retrieve the data contained within a web service.

- Web Service
  - Accessed by Programs
  - Response will be in XML or JSON
  - (API) that runs on the server
  - Used to communicate between heterogeneous systems and platforms

Overview of web services..


1 Browser request for server page
Web server request for web service
2
Partner web service
Web server
3
Web server request for web service
Public web service
Private web service
External financial system
5
Asynchronous JavaScript requests for web services
6
7
Asynchronous JavaScript request for web service
4 Web server request for web service
Internal legacy system
Affiliate web service
Travel service
Public web service
Mapping service

Provide computing infrastructure for application integration and collaboration

Benefits of using Web Services

- Web services use common and universally supported standards (HTTP and XML/JSON), they are supported on a wide variety of platforms.

- Web services allow various applications to talk to each other and share data and services regardless of their platforms
  - Platform independence opens up the opportunity for heterogeneous systems to access Web services

- Web services can be used to implement a service-oriented architecture (SOA).
  - SOA architecture aims to achieve very loose coupling among interacting software services.
  - SOA provides a very palatable potential solution to application integration issues.

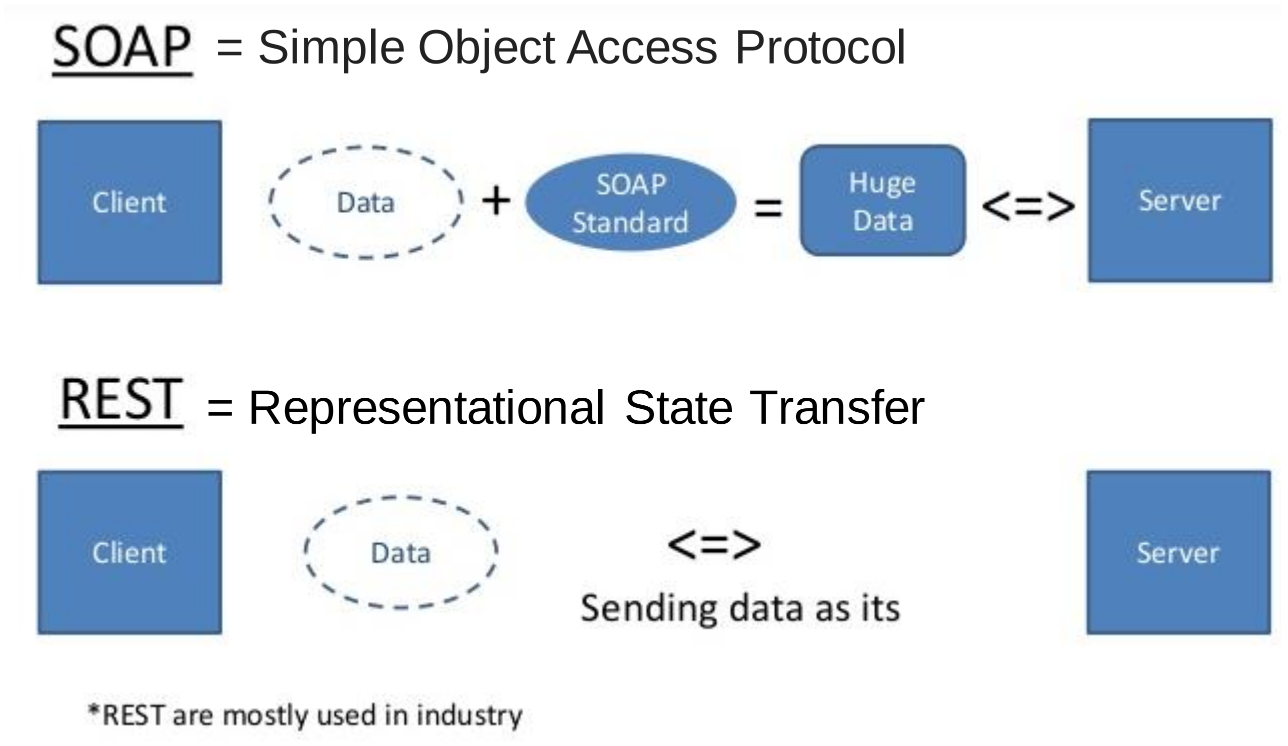
- Web services allows exposing the functionality of existing code over the network

Examples of Web Services

- YouTube API Code Samples
  - https://developers.google.com/youtube/code_samples
- Flickr
  - https://www.flickr.com/services/api/
- Google maps
  - https://developers.google.com/maps/
- Yahoo Weather API
  - https://developer.yahoo.com/weather/?guccounter=1

REST and Web Services

- ❑ Web services are further classified into SOAP and REST.
- ❑ In the present day scenario most services prefer REST over SOAP.


SOAP = Simple Object Access Protocol
Client
Data
+
SOAP Standard
=
Huge Data
<=>
Server
REST = Representational State Transfer
Client
Data
<=>
Sending data as its
Server
*REST are mostly used in industry

SOAP services

- For integrating web-based applications, the "web service" describes a standardized way using the XML, SOAP, WSDL and UDDI open standards over an Internet Protocol backbone.
  - XML is the data format used to contain the data and provide metadata around it,
  - SOAP is used to transfer the data,
  - WSDL is used for describing the services available and
  - UDDI lists what services are available.

- While SOAP and WSDL are **complex XML** schemas,
  - this standard is well supported in the .NET and Java environments (perhaps a little less so with PHP).

XML = eXtensible Markup Language
SOAP = Simple Object Access Protocol
WSDL = Web Services Description Language
UDDI = Universal Description, Discovery, and Integration

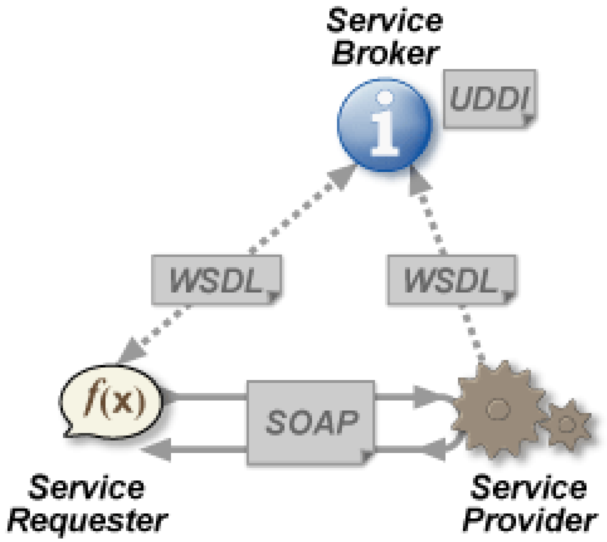
SOAP services

- The *service provider* sends a WSDL file to UDDI which defines: which software system should be contacted for which type of data

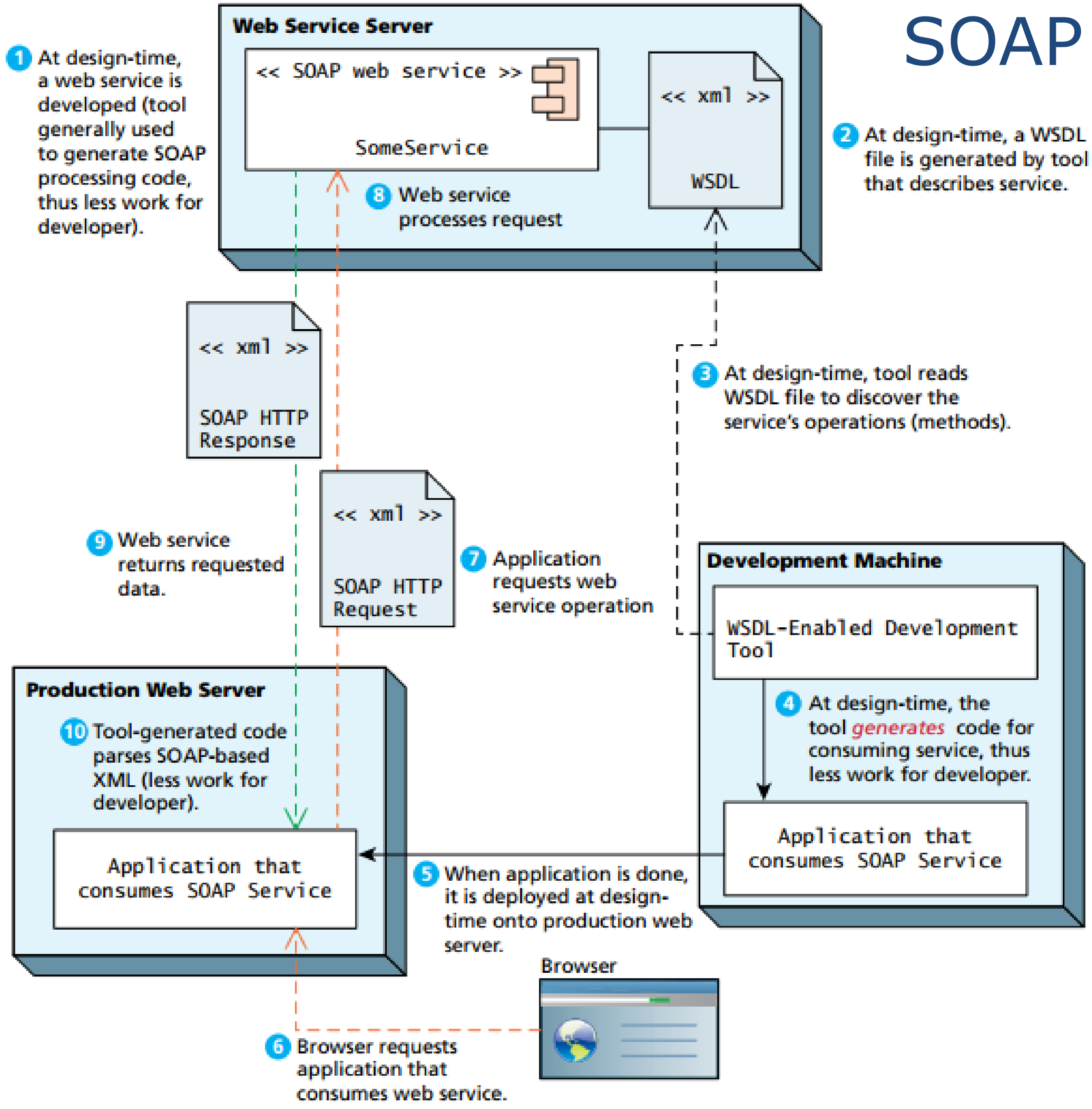
- The *service requester* contacts UDDI to find out who is the provider for the data it needs, and then it contacts the *service provider* using the SOAP protocol.

- The *service provider* validates the service request and sends structured data in an XML file, using the SOAP protocol.

- This XML file would be validated again by the service requester using an XSD file.



SOAP services


Web Service Server
<< SOAP web service >>
SomeService
<< xml >>
WSDL
1 At design-time, a web service is developed (tool generally used to generate SOAP processing code, thus less work for developer).
2 At design-time, a WSDL file is generated by tool that describes service.
8 Web service processes request
<< xml >>
SOAP HTTP Response
3 At design-time, tool reads WSDL file to discover the service's operations (methods).
<< xml >>
SOAP HTTP Request
9 Web service returns requested data.
7 Application requests web service operation
Development Machine
WSDL-Enabled Development Tool
4 At design-time, the tool generates code for consuming service, thus less work for developer.
Application that consumes SOAP Service
Production Web Server
10 Tool-generated code parses SOAP-based XML (less work for developer).
Application that consumes SOAP Service
5 When application is done, it is deployed at design-time onto production web server.
Browser
6 Browser requests application that consumes web service.

Web services 1.0 and 2.0

Web service 1.0

- ❑ "SOAP Web Services"
- ❑ Using the SOAP Protocol
- ❑ Uses XML

Web service 2.0

- ❑ "RESTful Web Services"
- ❑ Using the REST Protocol
- ❑ Uses Standard HTTP methods(GET, PUT, POST , DELETE)
- ❑ Uses JSON or XML

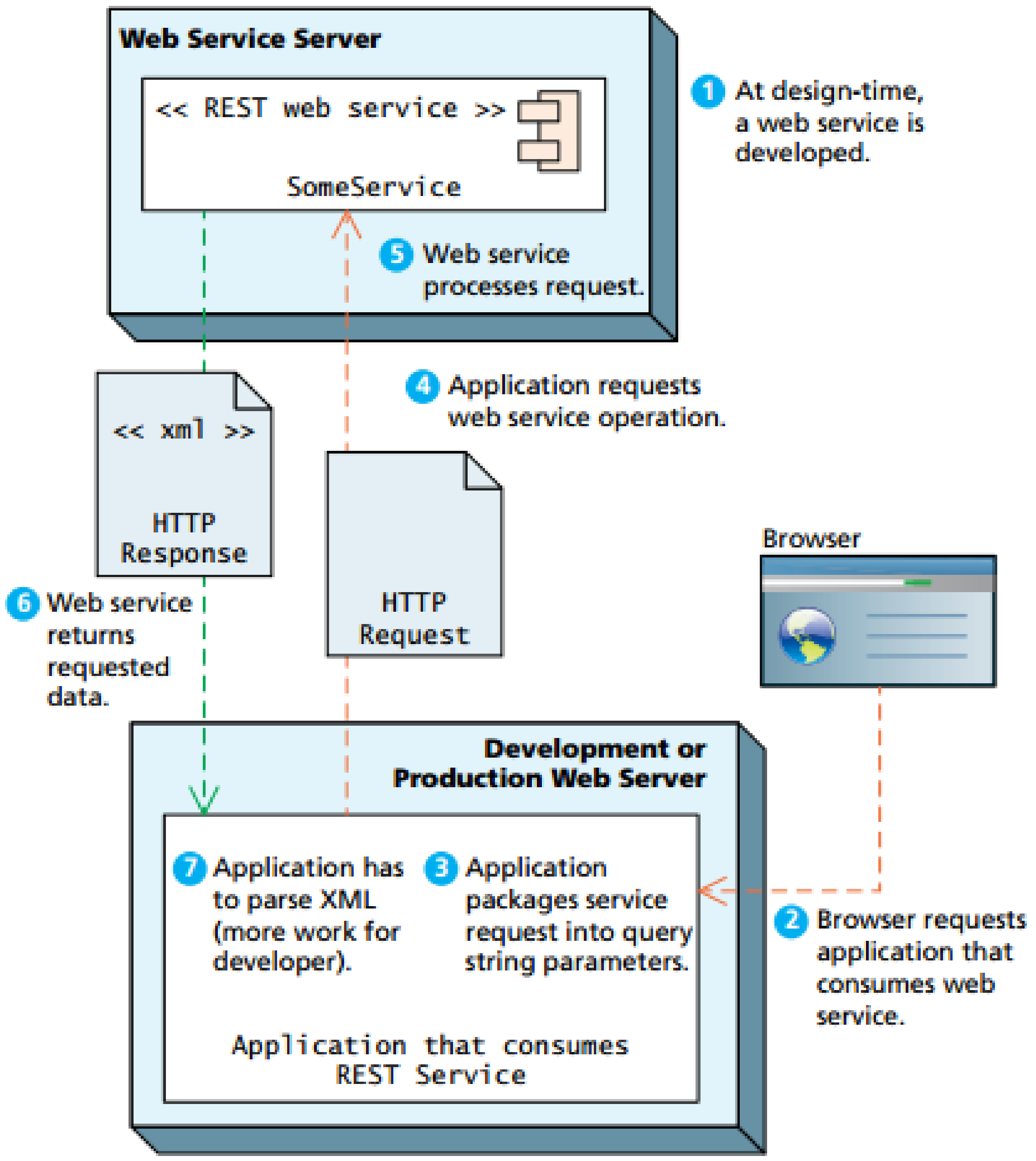
REST appears to have almost completely displaced SOAP services.

https://nordicapis.com/rest-vs-soap-nordic-apis-infographic-comparison/

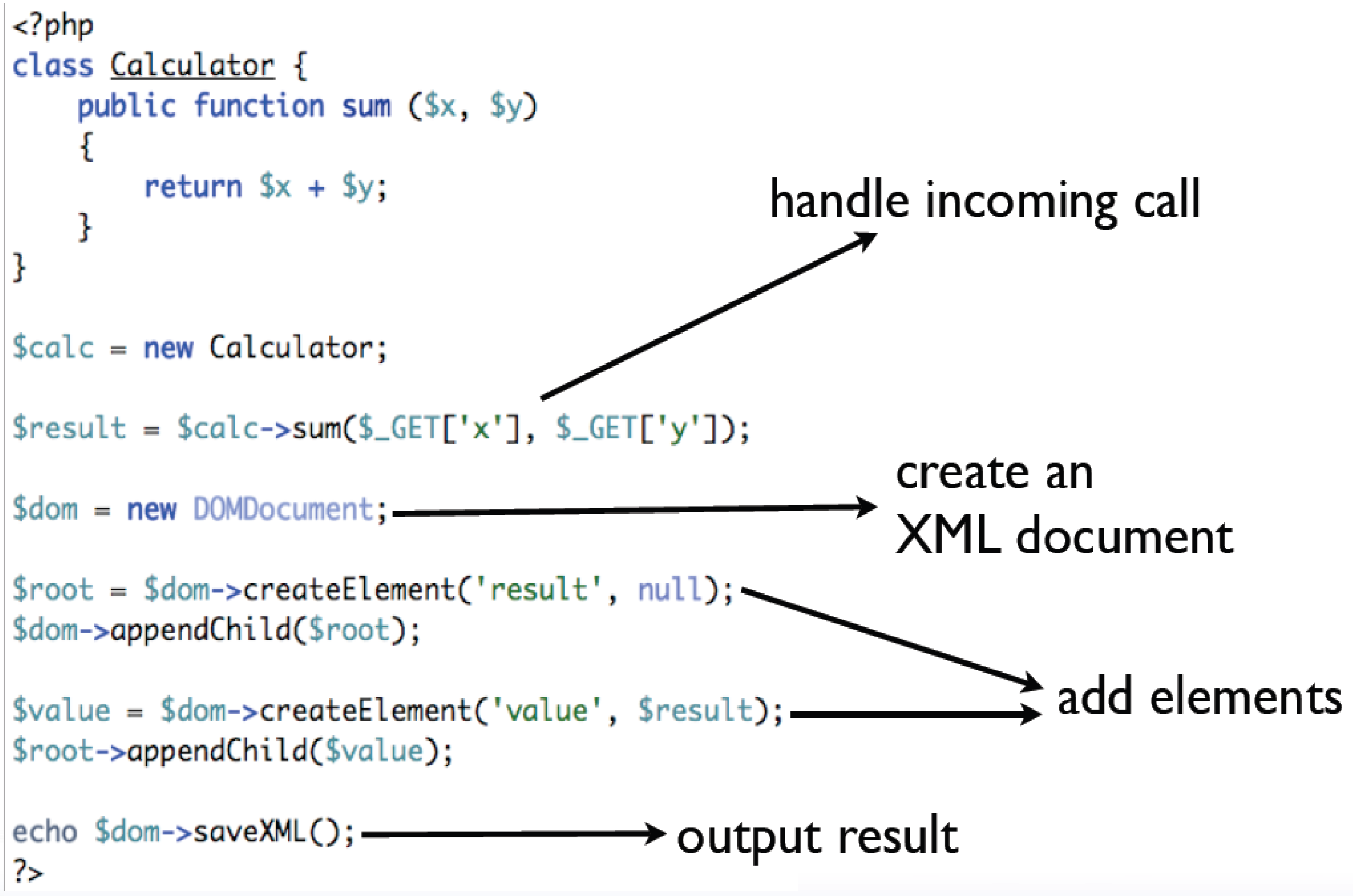
REST web services

- It simply uses HTTP URLs for requesting a resource/object (and for encoding input parameters).
- The serialized representation of the object, usually an XML or JSON stream, is then returned to the requestor as a normal HTTP response.
- No special tools are needed to deploy/test a REST-based service
- With the broad interest in the asynchronous consumption of server data at the browser using AJAX, it is easier to consume in JavaScript than SOAP.
  - if an object is serialized via JSON, it can be turned into a complex JavaScript object in one simple line of JavaScript.
  - However, for REST web services use XML as the data format, manual XML parsing and processing is required

REST web services

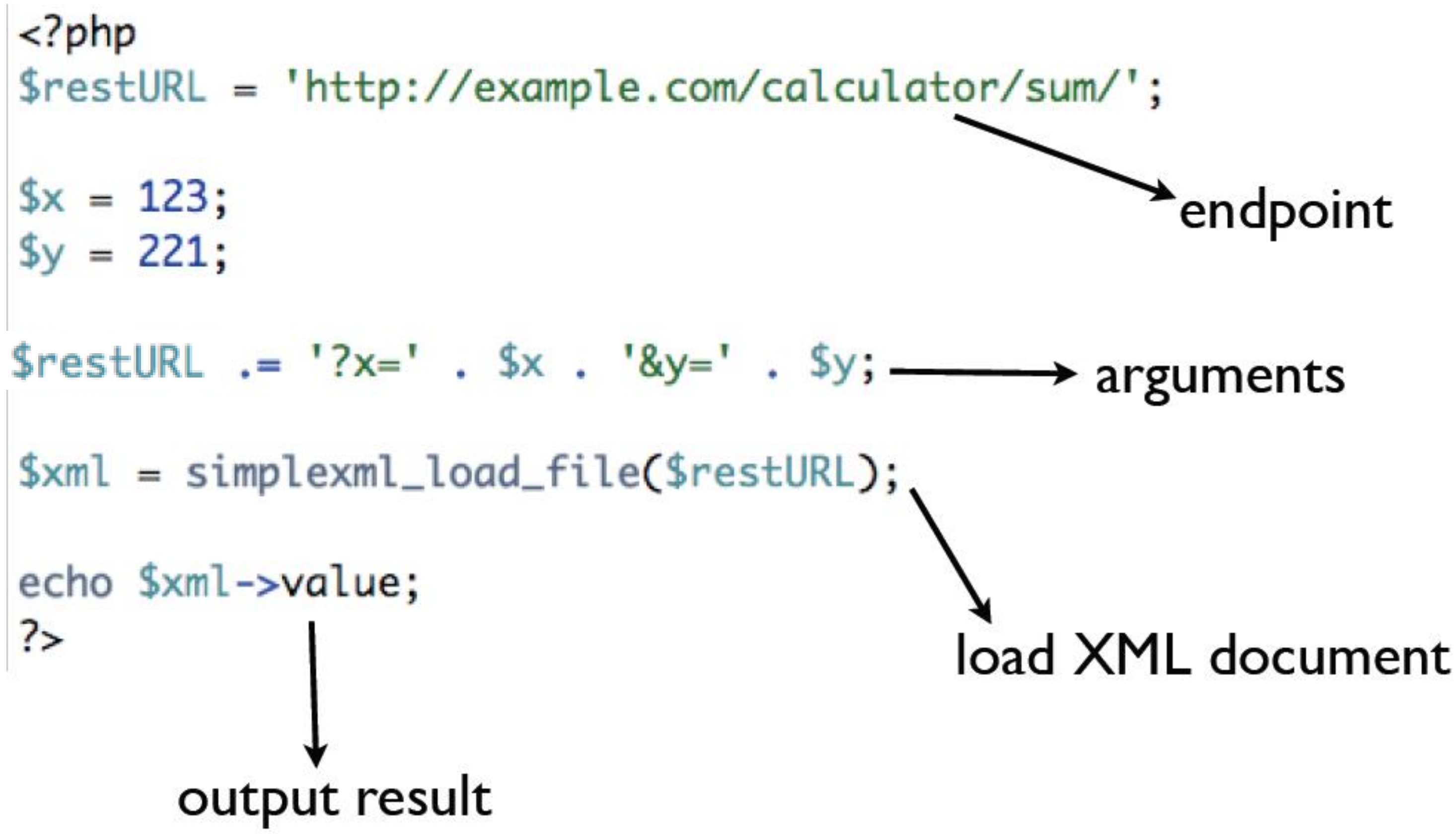

Web Service Server
<< REST web service >>
SomeService
1 At design-time, a web service is developed.
5 Web service processes request.
4 Application requests web service operation.
<< xml >>
HTTP Response
HTTP Request
Browser
6 Web service returns requested data.
Development or Production Web Server
7 Application has to parse XML (more work for developer).
3 Application packages service request into query string parameters.
Application that consumes REST Service
2 Browser requests application that consumes web service.

Simple PHP REST: Provider



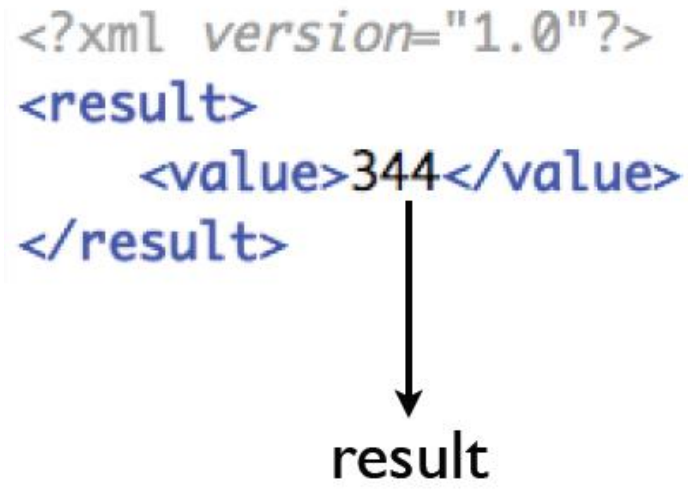
```php
<?php
class Calculator {
    public function sum ($x, $y)
    {
        return $x + $y;
    }
}

$calc = new Calculator;

$result = $calc->sum($_GET['x'], $_GET['y']);

$dom = new DOMDocument;

$root = $dom->createElement('result', null);
$dom->appendChild($root);

$value = $dom->createElement('value', $result);
$root->appendChild($value);

echo $dom->saveXML();
?>
```

Simple PHP REST: Consumer



```php
<?php
$restURL = 'http://example.com/calculator/sum/';

$x = 123;
$y = 221;

$restURL .= '?x=' . $x . '&y=' . $y;

$xml = simplexml_load_file($restURL);

echo $xml->value;
?>
```

Simple PHP REST: Output



```
<?xml version="1.0"?>
<result>
    <value>344</value>
</result>
```

Creating/Consuming a RESTful Web Service in PHP: Example

- Creating a RESTful Web Service in PHP
  - https://www.youtube.com/watch?v=5eWC-lf1FxM&t=599s

- Consuming a RESTful webservice in PHP
  - https://www.youtube.com/watch?v=UciopWMTdUM

- Create a Basic Web Service Using PHP, MySQL, XML, and JSON
  - https://davidwalsh.name/web-service-php-mysql-xml-json

- Working with REST
  - https://www.youtube.com/watch?v=LooL6_chvN4