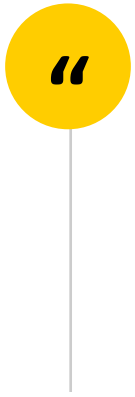
Lecture course notes
Service Oriented Computing

Neila BEN LAKHAL (PhD @ Tokyo Institute of Technology)
By Neila.benlakhal@enicarthage.rnu.tn

Course Introduction



Textbooks

- **Web services: Principles and Technology,** Michael P. Papazoglou, Pearson Education Limited, second edition, 2012.
- The teaching materials of the book "**Web Services: Concepts, Architectures and Applications**" and is thus Copyright © 2003 Gustavo Alonso, ETH Zürich and/or Copyright © 2004 Springer-Verlag Berlin Heidelberg.
- **SOA and Web Services Interface Design**: Principles, Techniques, and Standards James Bean, ISBN: 9780123748911, Morgan Kaufmann Publishers 2010 (chap1,2,3,10)

Textbooks

- **PHP Web Services, APIs for the Modern Web** By Lorna Mitchell, Published by O'Reilly Media, April 2013.
- **RESTful Web Services** By Leonard Richardson, Sam Ruby, **Published by** O'Reilly Medi, December 2008.
- **Microservices vs. Service-Oriented Architecture,** by Mark Richards Copyright © 2016 O'Reilly Media.

Class website

https://sites.google.com/view/neila/

Motivations for the course

- The course aims at developing a critical understanding of SOA and Web service technology and its possibilities today.

- The goal is for participants to be able to look at current and future developments in SOA with enough background to be able to judge how much of a contribution they are and what their true potential is.

The questions we will try to answer in this course are:

- *What is SOA?*
- *What are Web services?*
- *SOA vs Web services ?*
- *Web services types : SOAP-Web services, restful Web services, microservices ,…*
- *What can be done with Web services today?*
- *How did we get there ?*
- *And what is next…*

Learning objectives (1/3)

- We will go over the basic Web service technology available today and learn:
  - WS-* standards (XML, SOAP, WSDL, BPEL).
- We will learn creating and calling **SOAP-based** Web services :
  - Bottom Up: Service class first
  - Top Down: Service description (WSDL) first
  - Using XML Schema to represent complex datatypes

Learning objectives (2/3)

- We will learn SOA principles and automating Business Processes as Web services composition
- We will compare types of services composition:
  - Dynamic | static composition
  - Orchestration |choreography
- We will discuss advanced topics on SOAP-based Web services e.g. Reliability of message exchange, security (privacy and authenticity), transaction,…

Learning objectives (3/3)

- Implementing and calling **RESTful Web services**
- What is the difference between SOAP-based and RESTful Web services?
- What is a **Microservice**?
- What is the difference between a (big) web service and a micro-service?
- Service oriented architecture (SOA) vs. Microservice architecture(MSA)?

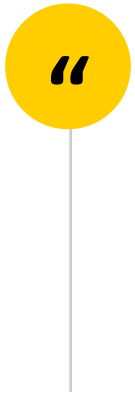
Grading

- Tests
- Project work:
  - Project content will be announced later
  - Implementing Web services (simple and composite / SOAP and RESTful)
  - Writing a report
  - Project presentation by all the team members :
    - Project presentation (≈ 10min) + questions (≈20min)

Lecture course notes
SOA and Web services

Neila BEN LAKHAL (PhD @ Tokyo Institute of Technology)
By Neila.benlakhal@enicarthage.rnu.tn

Chapter 1:
Fundamental concepts
SOA/Service definitions
Software architecture evolution
Web services vs. SOA
Web services examples



How did we get there ?

- In order to understand Web services, we need to take a step back and look at the way **the evolution the architecture of information systems(IS**) has been evolving in the last decades. Only so we will then be able to understand Web services and SOA.

How did we get there ?

The evolution of the architecture of information systems

SOA definition [computer Dictionary]

*"(**S**ervice-**O**riented **A**rchitecture) The modularization of business functions for greater flexibility and reusability. Instead of building* ***monolithic*** *applications for each department, an SOA organizes business software in a granular fashion so that common functions can be used interchangeably by different departments internally and by external business partners as well. The more granular the components (the more pieces), the more they can be reused."*

[http://computer.yourdictionary.com 2007]

SOA Definition [Gartner Group]

Service-oriented architecture (SOA) was first described by Gartner in 1996

(see SSA Research Note SPA-401-068 , 12 April 1996)

The interest in the architecture has been spurred by the emergence of a powerful industry trend: ***Web services*** *in 2001,*

"Essentially, SOA is a software architecture that starts with an interface definition and builds the entire application topology as a topology of interfaces, interface implementations and interface calls".

[Gartner Group technical spec.]

Software architecture ?

- Automating business processes crossing enterprise boundaries, requires leveraging shared resources such as Web servers, business logic components, databases etc.
- In this environment, partners must not only agree on a core set of interfaces and standards but on the **software architecture** to be used.

Software architecture ?

- Software architecture is the high-level structure of a software system.
- It is commonly specified in terms of functional components and interactions/interconnections among those components.

The architecture for a business information system

◉While constructing the architecture for a business information system, the building blocs which might be physically deployed across a set of distributed processing units (e.g. machines in a network) are :

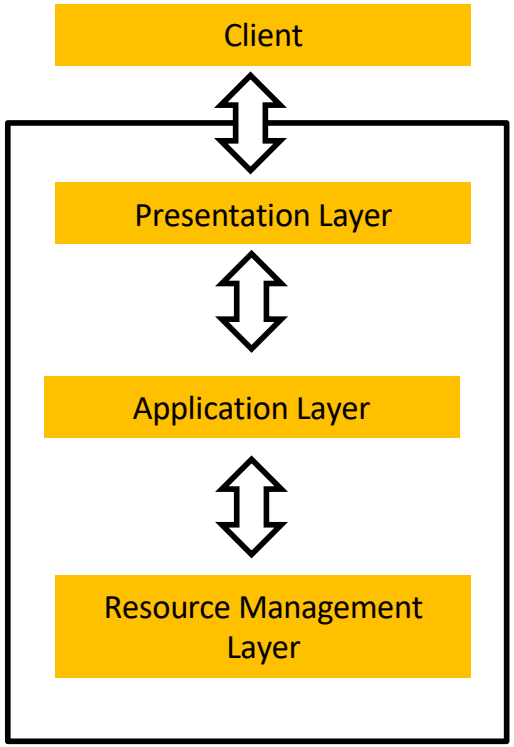
- Presentation layer (tiers)
- Application layer (tiers)
- Resource layer(tiers)

The architecture for a business information system

Presentation layer through which the user can submit operations and obtain a result.

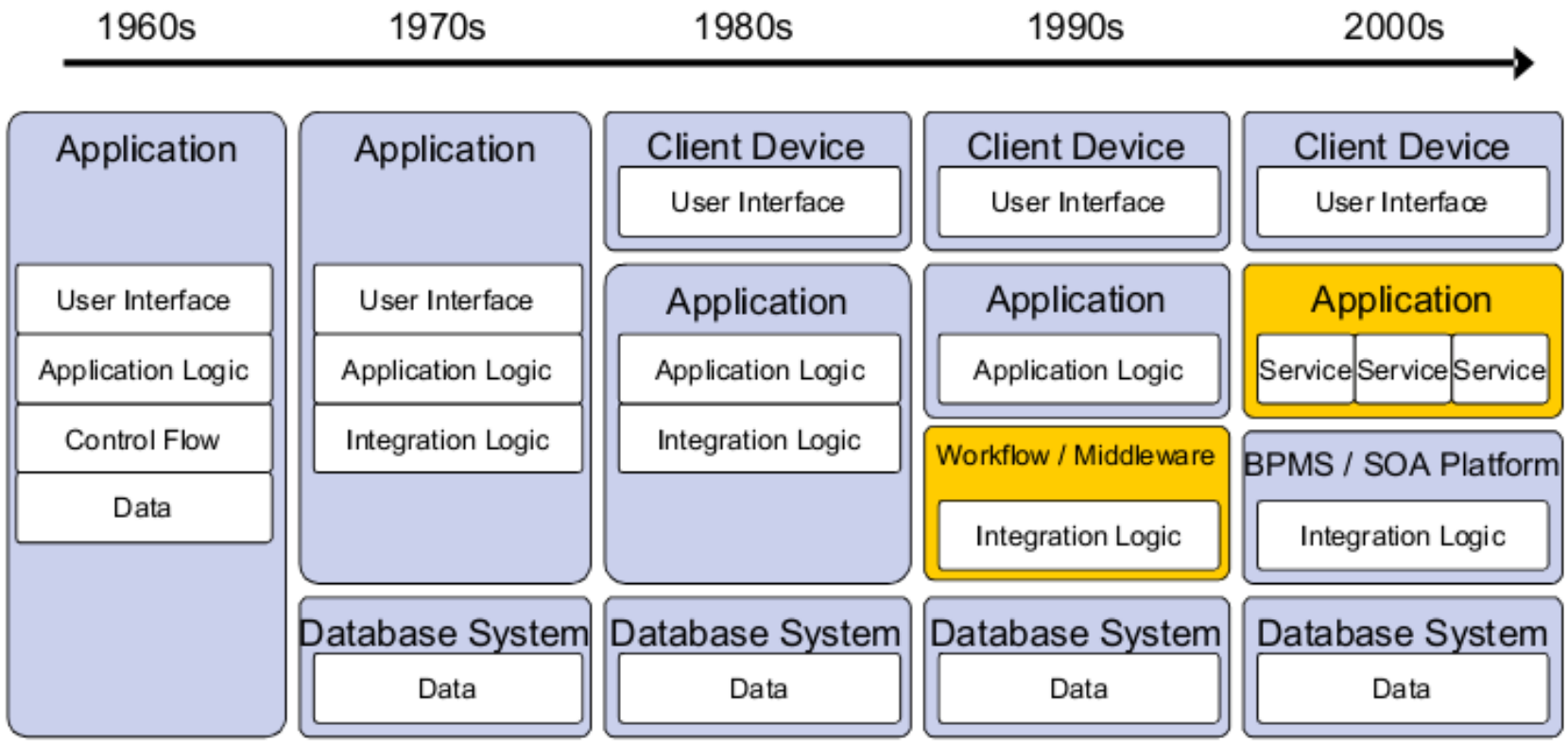
Application layer establishes what operations can be performed over the system and how they take place.

Resource layer deals with the organization (storage, indexing and retrieval) of the data necessary to support the application layer.

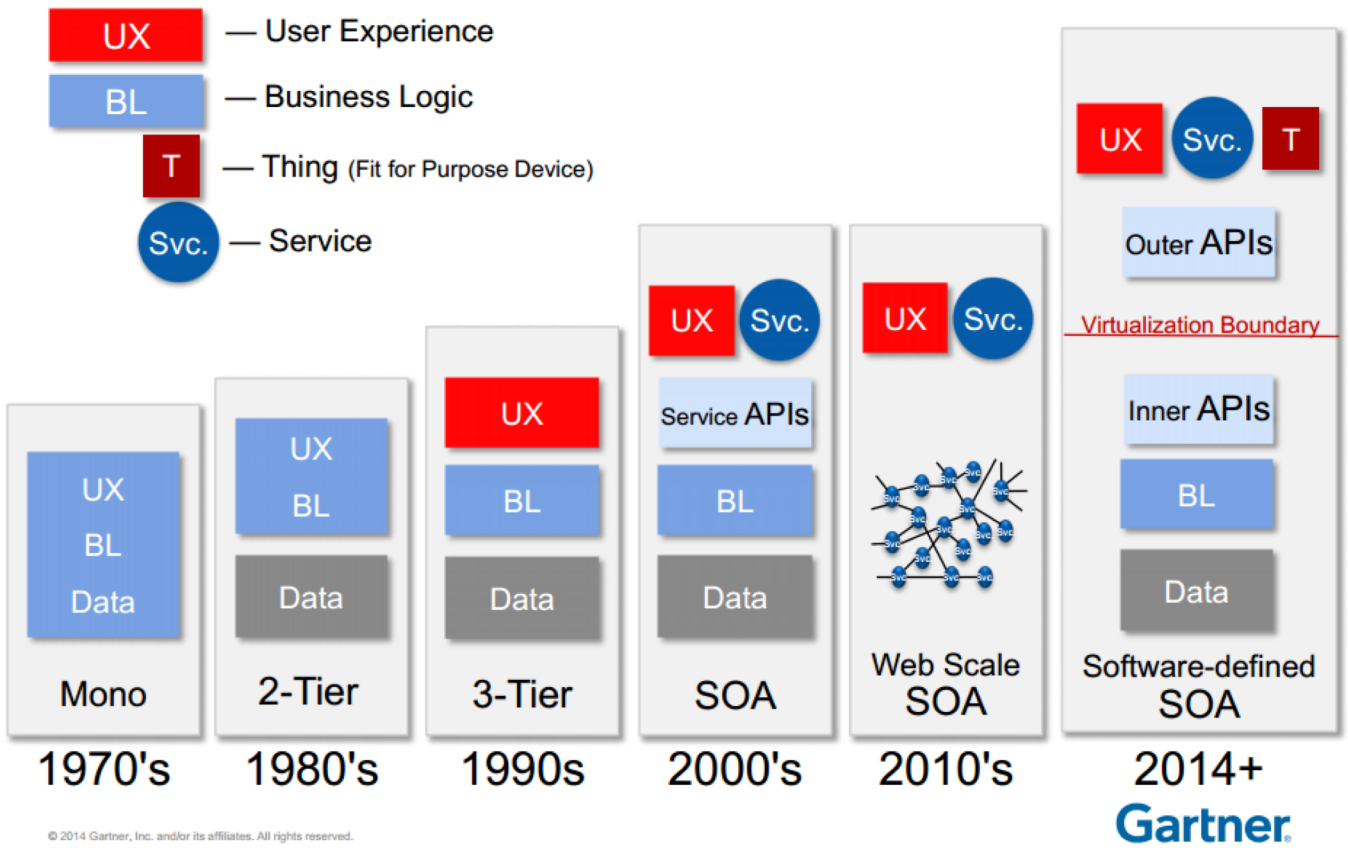


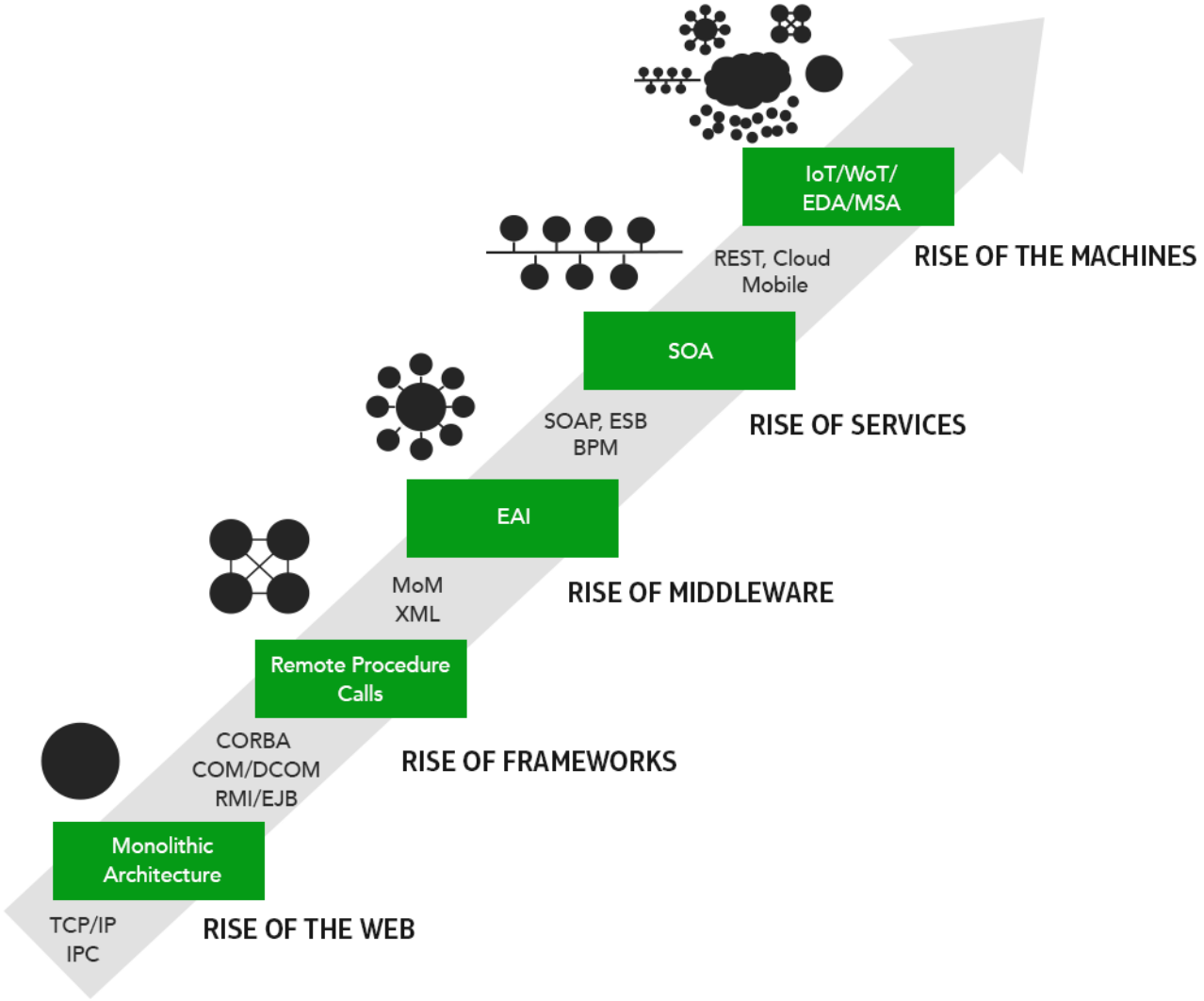
Types of architecture

- Those building blocs can be assembled in different ways:
  - 1-tier architecture
  - 2-tier architecture
  - 3-tier architecture | N-tiers architecture
  - SOA architecture
  - Advanced SOA architecture (micro-services).



Software architecture evolution

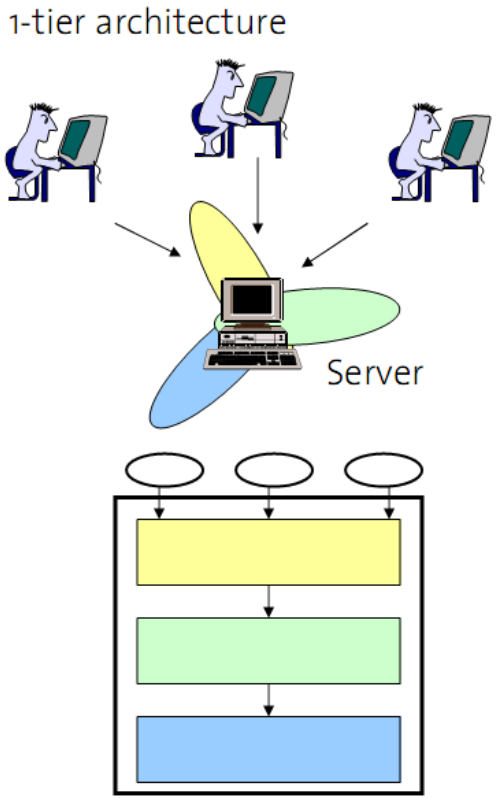

UX — User Experience
BL — Business Logic
T — Thing (Fit for Purpose Device)
Svc. — Service
UX
BL
Data
Mono
1970's
UX
BL
Data
2-Tier
1980's
UX
BL
Data
3-Tier
1990s
UX
Svc.
Service APIs
BL
Data
SOA
2000's
UX
Svc.
Web Scale
SOA
2010's
UX
Svc.
T
Outer APIs
Virtualization Boundary
Inner APIs
BL
Data
Software-defined
SOA
2014+
Gartner
© 2014 Gartner, Inc. and/or its affiliates. All rights reserved.



https://www.appcentrica.com/the-rise-of-microservices/

1-tier Architecture

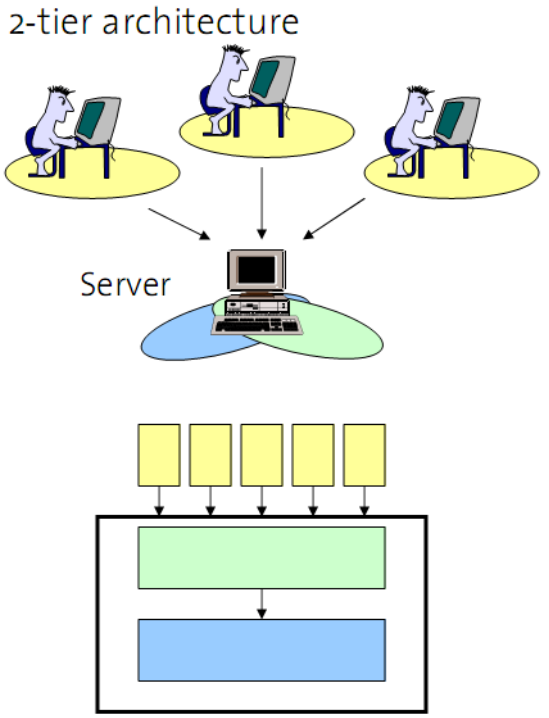
- The presentation layer, application logic and resource manager are built as a ***monolithic*** entity.
- Users/programs access the system through display terminals but what is displayed and how it appears is controlled by the server. (These are the *"**dumb" terminals***).
- This was the typical architecture of ***mainframe*** applications, offering several advantages:
  - *All is centralized; managing and controlling resources is easier,*
  - *The design can be highly optimized by blurring the separation between layers.*



2-tier Architecture

◉As computers became more powerful, the presentation layer moves to the client. This has several advantages:

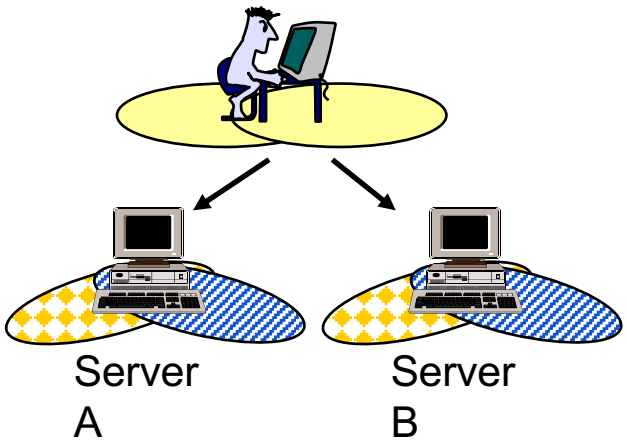
- Clients are ***independent of each other***: one could have several presentation layers depending on what each client wants to do.
- One can take advantage of the computing power at the client machine to have more sophisticated presentation layers. This also saves computer resources at the server machine.
- The resource manager only sees one client: the application logic. This greatly helps with performance since there are no connections/sessions to maintain.



What if a client wants to access 2 or more systems ?

- There is very little that can be done to solve this problems if staying within the 2-tier model. It can be solved by adding a level of indirection:

middleware



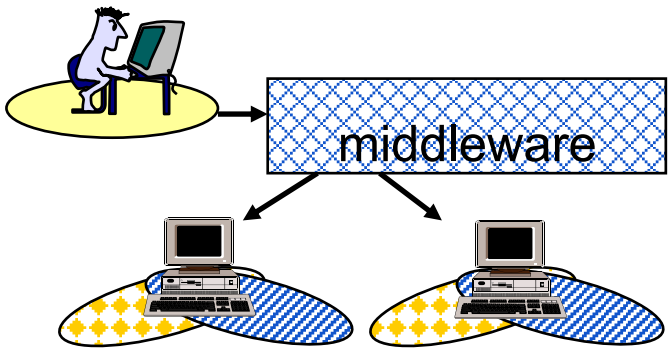
- Clients end up wanting to access two or more systems. With a 2-tier architecture, this creates several problems:
  - The underlying systems don't know about each other; there is no common business logic. If it is necessary, it needs to be implemented at the client.
  - The underlying systems are probably different. The complexity of dealing with two heterogeneous systems needs to be addressed by the client.
  - The client becomes responsible for knowing where things are, how to get to them, and how to ensure consistency!
  - This is tremendously inefficient from all points of view (very fat clients are not a solution).

Middleware ?

Middleware is the software “glue” that helps programs and databases (which may be on different computers) work together. Its most basic function is to enable communication between different pieces of software. [gartner Group]

Middleware facilitates and manages the interaction between applications across heterogeneous computing platforms.

It is the architectural solution to the problem of integrating a collection of servers and applications under a common service interface.

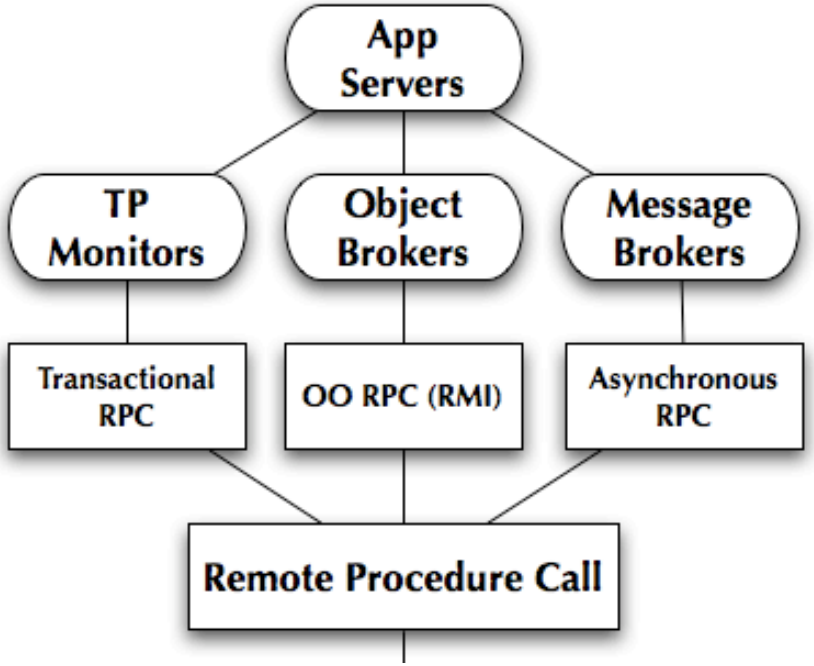


Types de middleware

Object-Oriented Middleware (OOM)
– Java RMI
– CORBA

Message-Oriented Middleware (MOM)
– Java Message Service
– IBM MQSeries
– Web Services

Event-Based Middleware
– Cambridge Event Architecture
– Hermes

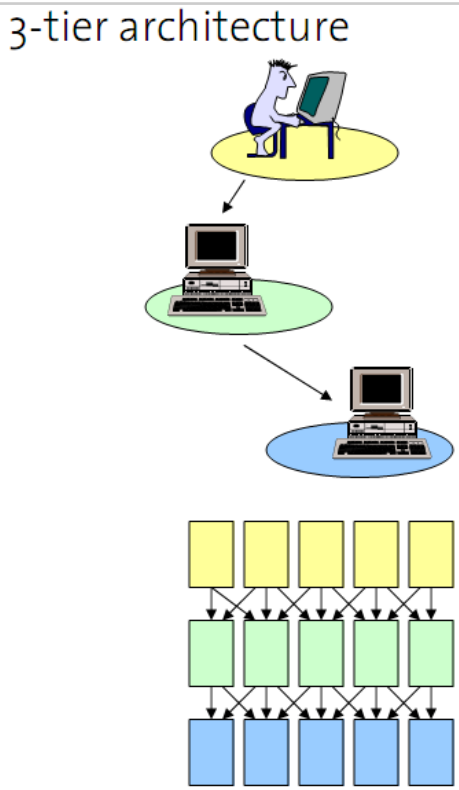


For more details :
http://www.cl.cam.ac.uk/teaching/1011/CDSysII/12-middleware.pdf

Limitations of Middleware

- ***Non-standard interfaces***. Traditional middleware systems and tools suffer from lack of standardization: they are not compatible. Thus, it is very expensive to build integrated distributed systems across different middleware platforms.
- ***Lack of trust***. How to trust the clients? Building integrated systems spawning across different trust domains can be difficult.
- ***Middleware systems are (logically) centralized***. Thus, there is no place for them in B2B Integration scenarios as they should be distributed across all partners. Point to Point integration does not scale.
- ***Slow interactions*** across organizational boundaries and should be handled asynchronously.

3-tier Architecture



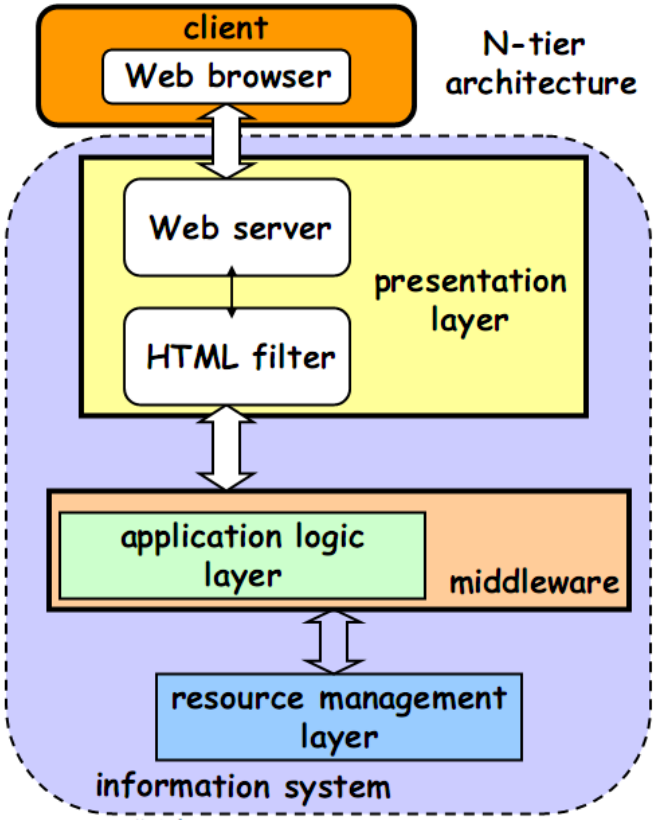
- In a 3 tier system, the three layers are ***fully separated***.
- The layers are also typically ***distributed.***
- A middleware-based system is a 3 tier architecture. This is a bit oversimplified but conceptually correct since the underlying systems can be treated as black boxes. In fact, 3 tier makes only sense in the context of middleware systems (otherwise the client has the same problems as in a 2 tier system).

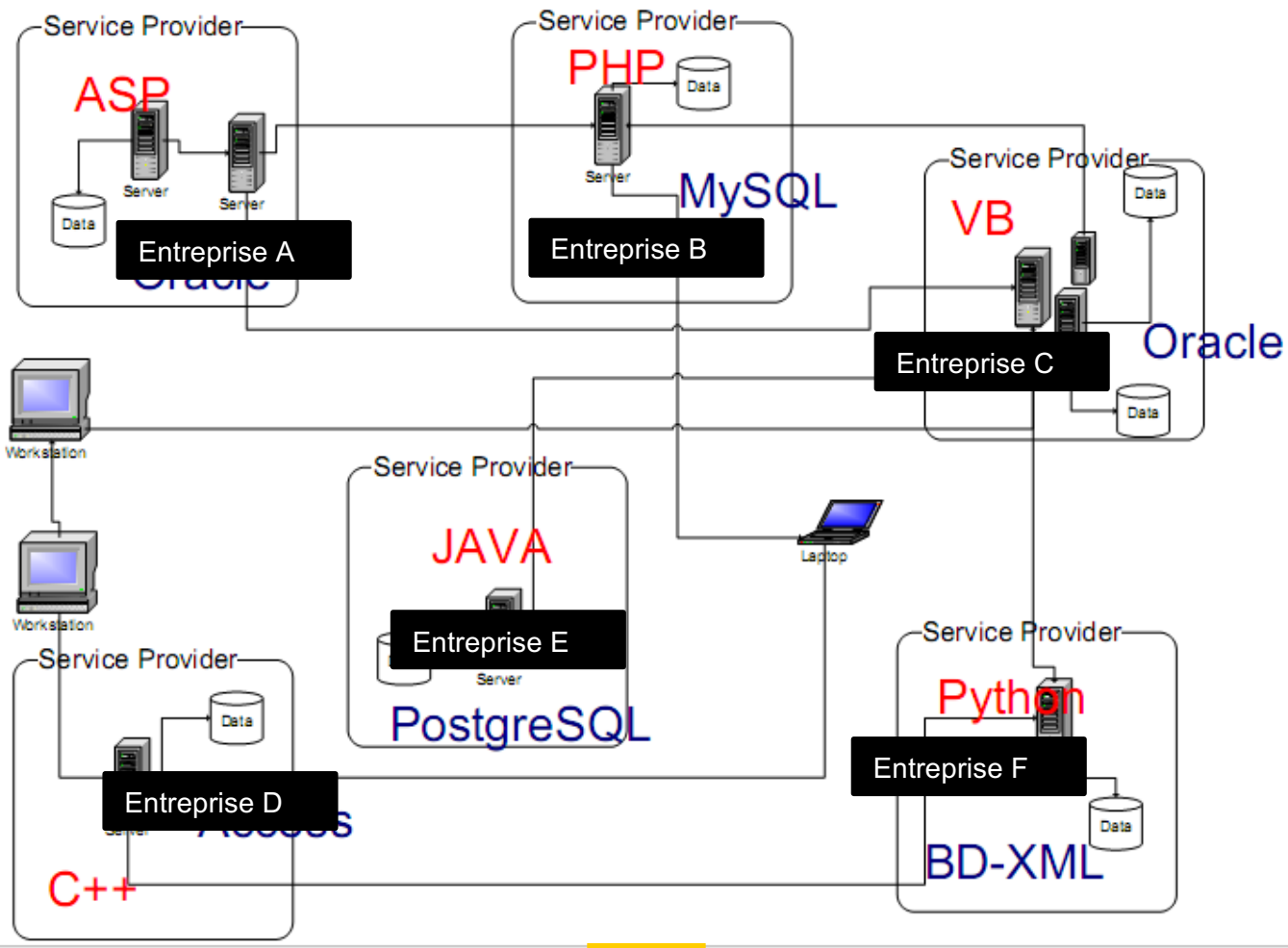
N-tier Architecture

- N-tier architectures result from connecting several 3-tier systems to each other and/or by adding an additional layer to allow clients to access the system through a "**Web server**"

- The Web layer was initially external to the system (a true additional layer); today, it is incorporated into a presentation layer that resides on the server side

- The addition of the Web layer led to the notion of "***application servers***",





How to interact beyond entreprise boundaries ?

Why do we need SOA?

◉Most companies today have a **large**, **heterogeneous** IT infrastructure that:

- **Keep changing**
- **Needs to evolve to adopt new technology**
- **Needs to be connected of that of commercial partners**
- **Needs to support an increasing amount of purposes and goals**

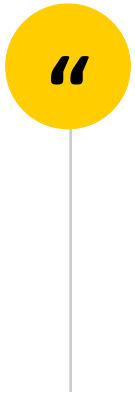
◉This was the field of *Enterprise Application Integration* using systems like CORBA. However, solutions until now suffered from:

- **Tightly integrated systems**
- **Vendor lock-in**
- **Technology lock-in (e.g., CORBA)**
- **Lack of flexibility and limitations when new technology arises**

Service ?

- *“Services represent self-contained business functionalities that can be part of one or more processes, and on the other hand, can be implemented by **any** technology on **any** platform.”*

« Un Service est un composant logiciel distribué, exposant des fonctionnalités à forte valeur ajoutée d’un domaine métier ».



[http://www3.opengroup.org/]

A service:

- *Is a logical representation of a repeatable business activity that has a specified outcome (for example, check customer credit; provide weather data, consolidate drilling reports)*
- *is self-contained,*
- *may be composed of other services,*
- *is a **black box** to consumers of the service,*

Service ?

- *Services are the key building blocks in an SOA.*
- *Services provide access to well described functionality in an easy-to-use and*
- *transparent fashion for service consumers.*

- *They are composed of three parts:* **Interface**, **Contract** *and* **Implementation**

Service contract?

Contracts *specify the conditions under which services can be consumed and what consumers can expect from service providers.*

This can include :

- *Guarantees on response times and availability,*
- *applied security measures,*
- *cost, and so on.*

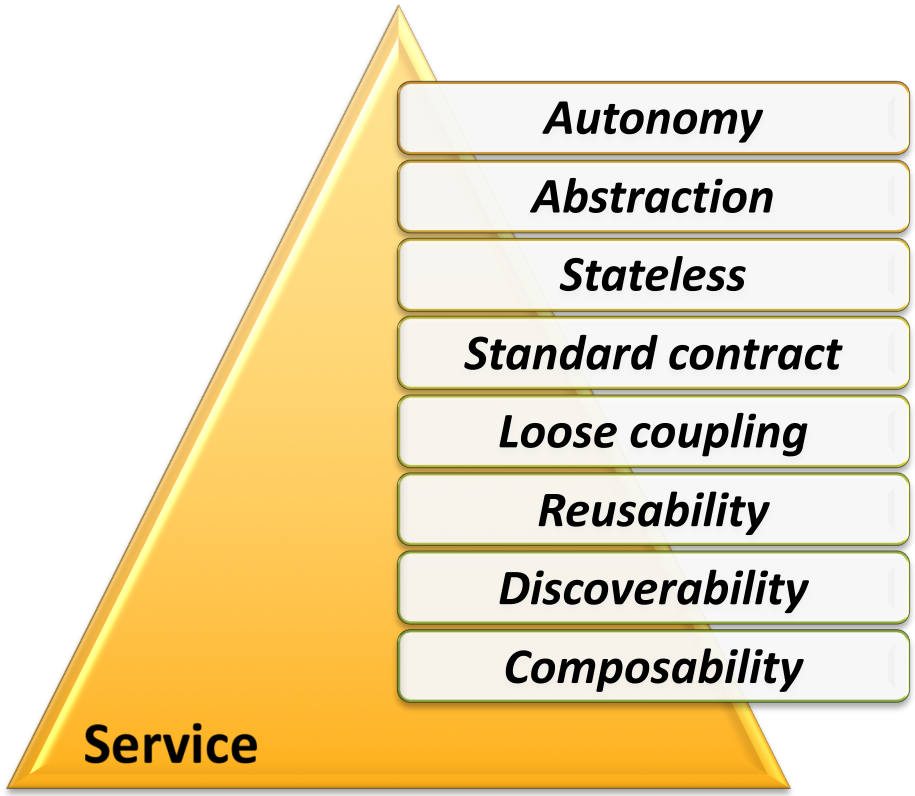
Service implementation?

- **Implementation** of a service is the actual realization of the business logic of the service.
- There are three ways to realize services:
  - *Use existing software*
  - *Build the logic or implementation for the service you need*
  - *Combine existing services into a new service*

Service interface?

- **Interface** is what is available to the service consumers.
- There are two types of interface:
  - Proprietary interfaces, specific to a programming language or system
  - Standard interfaces, based on web services, there are two types of web service:
    - ***SOAP-based services***
    - ***RESTful services***

Service properties



Les 8 aspects qui caractérisent un service

1. **Contrat standardisé** : L'ensemble des services d'un même système technique sont exposés au travers de contrats respectant les mêmes règles de standardisation.
2. **Couplage lâche** : Un service doit être au minimum lié à la structure d'un autre et doit imposer le minimum de contraintes à un service l'utilisant également.
3. **Abstraction** : Le contrat d'un service ne doit contenir que les informations essentielles à son invocation. Seules ces informations doivent être publiées.
4. **Réutilisabilité** : Un service exprime une logique et peut ainsi être positionné comme une ressource réutilisable.

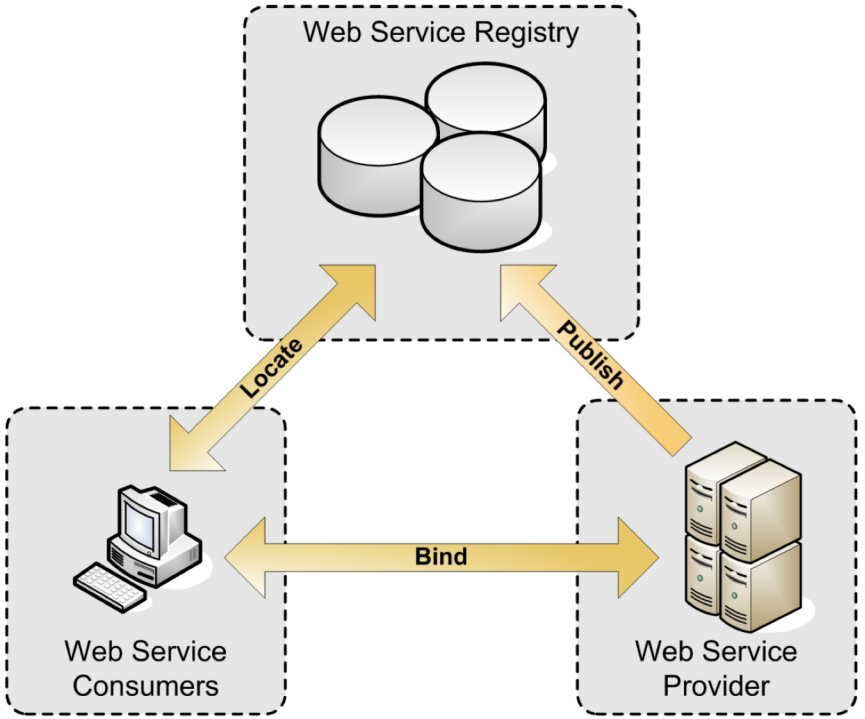
Les 8 aspects qui caractérisent un service

5. **Autonomie** : Un service doit exercer un contrôle fort sur son environnement d'exécution sous-jacent. Plus ce contrôle est fort, plus l'exécution d'un service est prédictible.
6. **Stateless (sans état)** : Un service doit minimiser la consommation de ressources en déléguant la gestion des informations d'état quand cela est nécessaire.
7. **Découvrabilité** : Un service est complété par un ensemble de méta-données de communication au travers desquelles il peut être découvert et interprété de façon effective.
8. **Composabilité** : Un service doit être conçu de façon à participer à des compositions de services.

◉Ces 8 aspects sont issus du livre « SOA Principles of Service Design » de Thomas Erl.

Roles of interaction in the SOA

- The main building blocks of an SOA are three-fold and they are determined on the basis of three primary ***roles***. These are :
- The **service provider**,
- The **service registry**, and
- The **service requestor**



Service Provider

- From a **business perspective** the Web services provider is the organization that owns the Web service and implements the business logic that underlies the service.
- From an **architectural perspective** this is the platform that hosts and controls access to the service.

Service Requester

- The next major role in the Web services architecture is that of the Web services requestor (or client).
- From a ***business perspective*** this is the enterprise that requires certain functions to be satisfied.
- From an ***architectural perspective***, this is the application that is looking for, and subsequently invoking, the service.
- The Web services requestor searches the service registry for the desired Web services.

Service Registry

- The last important role that can be distinguished in the Web services architecture is that of the Web services registry, which is a searchable directory where service descriptions can be published and searched.
- Service requestors find service descriptions in the registry and obtain binding information for services.
- This information is sufficient for the service requestor to contact, or bind to, the service provider and thus make use of the services it provides.

Invocation de service

Il y a 3 types d'invocation de services:

- ***Invocation synchrone*** : Dans ce cas, (service requester) invoque le service, passe la main au (service provider) et attend le résultat de l'invocation .
- ***Invocation Asynchrone:*** Dans ce cas, (service requester) invoque le service, continue sa tache SANS attendre, (entre temps, (service provider) prend en charge l'invocation du service, qui à la fin du service, informe le (service requester) des résultats.
- ***Invocation via un ESB (Entreprise Service Bus)*** : dans ce cas, tout les messages entre (service provider) et (service Requester) transite par le ESB. Et les 2 types d'invocation synchrone ou asynchrone peuvent se faire dans ce cas la MAIS toujours via le ESB qui joue le rôle d'un médiateur,

Service Repository vs. Service Registry

◉***Service discovery*** in a service **repository**:

- Le processus de rechercher des services remplissant une certaine fonctionnalité par un ensemble de mots clé par exemple.

◉***Service lookup*** in a service **registry** :

- Le processus de trouver le point d'appel (end point) d'un service en se basant sur un nom de service, le nom de son interface, ses paramètres etc.

ESB : couche de médiation

- C’est le point d’entrée vers un service => invocation indirecte du service au travers du bus
- Infrastructure qui optimise les échanges entre consommateurs et fournisseurs de services. Il peut prendre en charge :
  - *Routage, transformation des données, transactions, sécurité, qualité de service,…*
- Le but d’un ESB est de permettre de communiquer de manière simple et standardisée entre des applications hétérogènes.

AOS : d'un concept à sa mise en application

Comment les **web** services permettent de concrétiser réellement l'AOS ?

SOA/Web services

- The real momentum for SOA was created by **Web Services**, which reached a broader public in 2001,
- Although Web Services do not necessarily translate to SOA, and not all SOA is based on Web Services, the relationship between the two technology directions is important and they are mutually influential: Web Services momentum will bring SOA to mainstream users, and the best-practice architecture of SOA will help make Web Services initiatives successful. [Gartner]

Web Services Definitions

*« Web Services are software applications that can be discovered, described, and accessed based on **XML** and standard Web protocols over **intranets**, **extranets**, and the **Internet**. »*

*Together, these standards are denoted as **WS-*** and are maintained by standardization committees W3C and OASIS.*

Examples of web service standards are : WS-Security, Web Service Description Language(WSDL), WS-RM (Reliable Messaging), and Simple Object Access Protocol(SOAP).”

Web Services definition

- **SOAP-based services** have ***interfaces*** that describe the available operations, the inputs and outputs of these operations, and the technology binding through which the operations are made available.
- These interfaces are documented using WSDL and XSD documents.
- A WSDL defines the ***operations*** of a service and can define **input** and ***output*** messages internally or refer to external XSD documents for this.

Examples of Web services

- A Web service can be:
  - ***A self-contained business task***, e.g. a funds withdrawal or funds deposit service;
  - ***A full-fledged business process***, e.g. as the automated purchasing of office supplies;
  - ***An application***, e.g. a life insurance application;
  - **A service-enabled resource**, e.g. access to a particular back-end database containing patient medical records.

- Web services can vary in function from ***simple requests*** to ***complete business applications,***

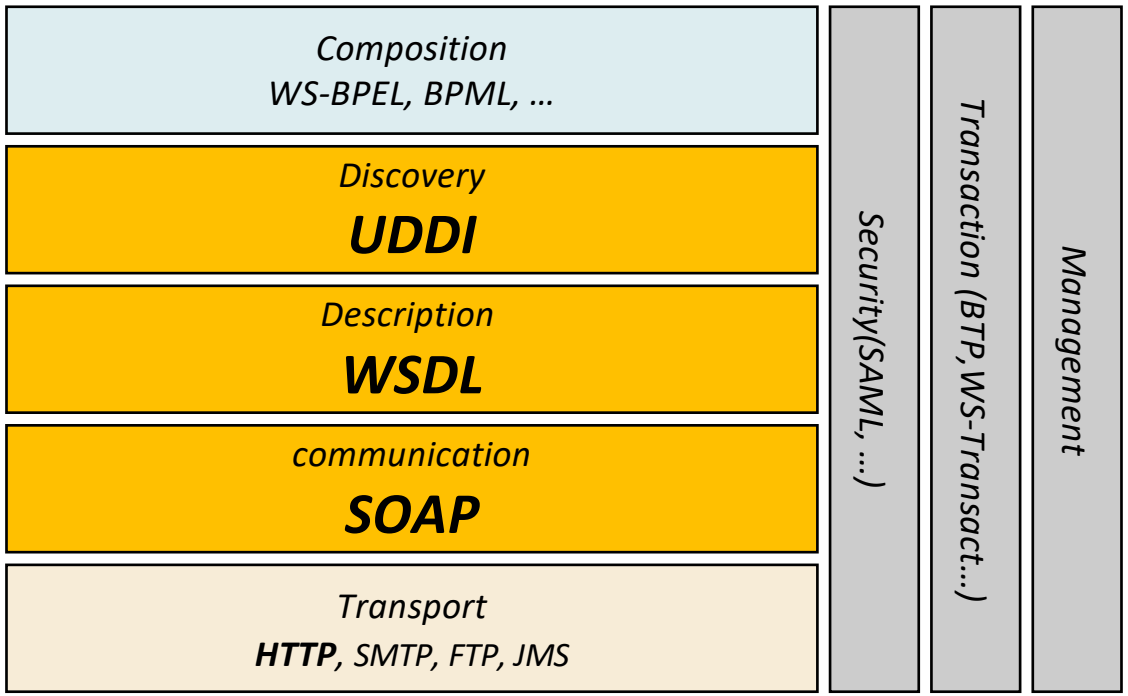
WS technologies

◉A Web service is a software component can be accessed by another application (such as a client, a server or another Web service) through the use of generally available, ubiquitous protocols and transports, such as HTTP.

◉Joint efforts between IBM and Microsoft, with the support of other vendors such as Ariba and Iona Technologies, have produced agreement on a basic set of ***XML-based standards*** for Web service interface **definition**, **discovery** and **remote calling**. They include:

- **Simple Object Access Protocol (SOAP),** which enables an application to call a Web service
- **Web Services Description Language (WSDL)** for describing Web service interfaces
- **Universal Description, Discovery and Integration (UDDI)** as the means for users to publish and locate available Web services, their characteristics and interfaces.[Gartner]

Diagramme des protocoles



Transport	HTTP, IIOP, SMTP, JMS		
Messaging	XML, **SOAP**		WS-Addressing
Description	XML Schema, **WSDL**		WS-Policy, SSDL
Discovery	**UDDI**		WS-MetadataExchange
Choreography	WSCL	WSCI	**WS-Coordination**
Business Processes	**WS-BPEL**	BPML	WSCDL
Stateful Resources	**WS-Resource Framework**		
Transactions	WS-CAF		**WS-Transactions** WS-Business Activities
Reliable Messaging	**WS-Reliability**		**WS-ReliableMessaging**
Security	**WS-Security** **SAML, XACML**		WS-Trust, WS-Privacy **WS-SecureConversation**
Event Notification	WS-Notification		WS-Eventing
Management	**WSDM**		**WS-Management**

WS Standards and specifications

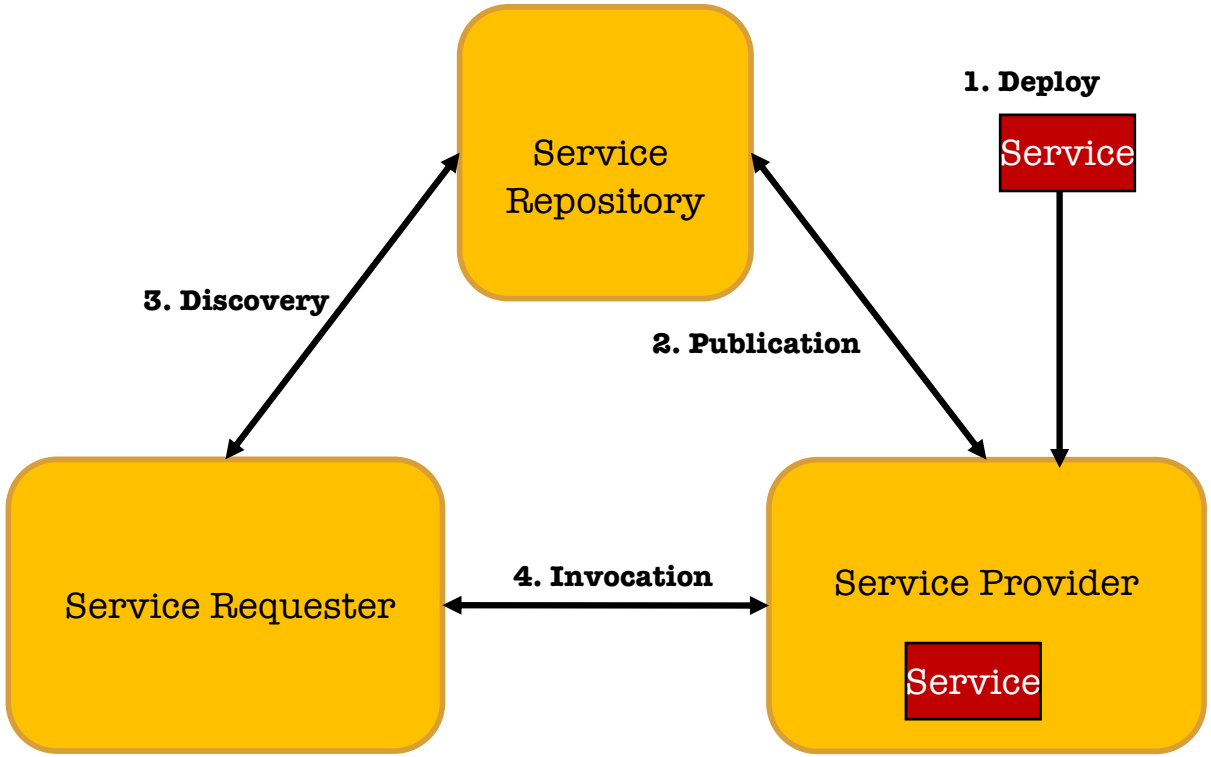
Web services Aim

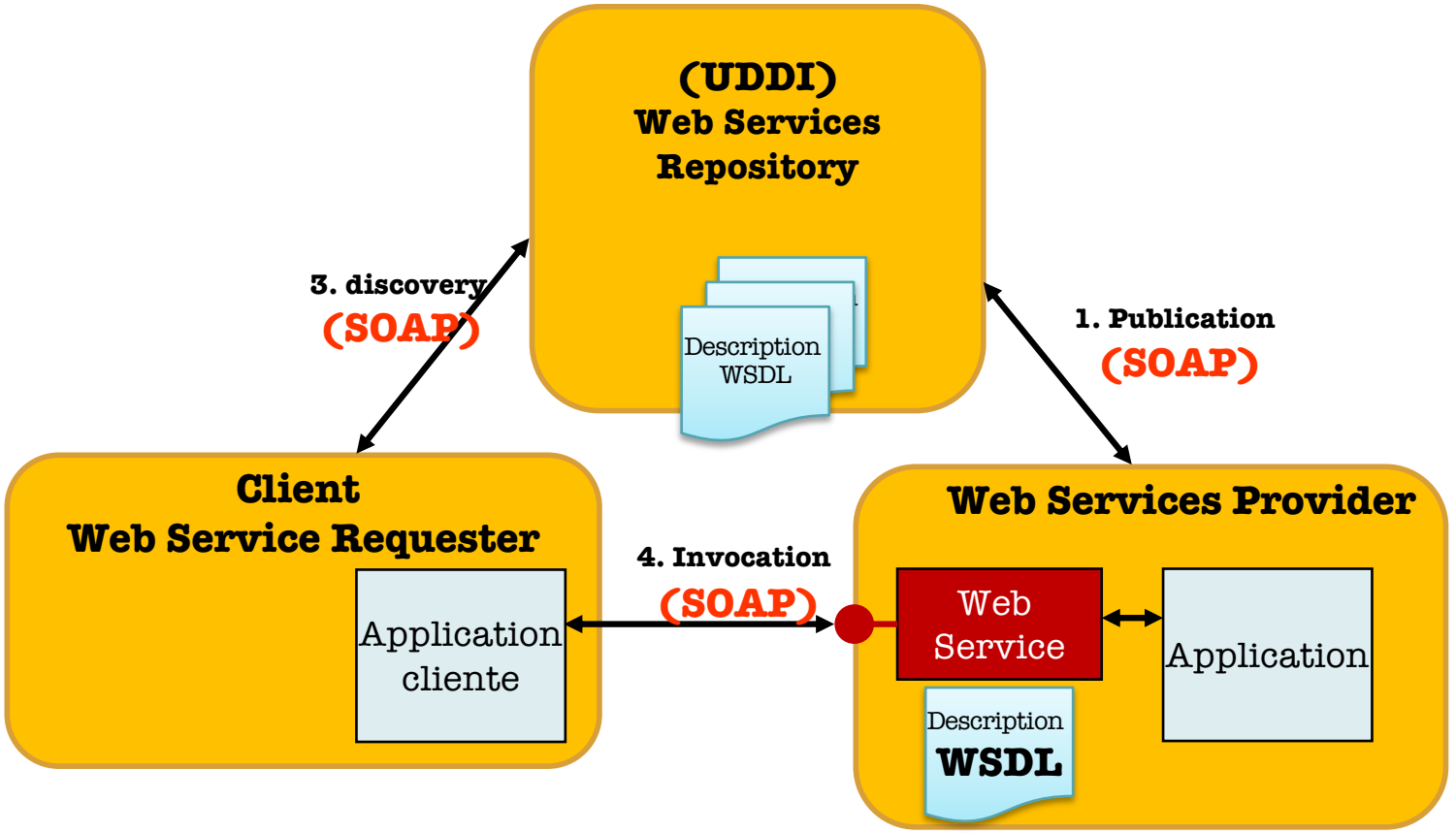
- The promise of **web services** lies is the interoperability between **services** that are running on different platforms and implemented using different programming languages.

Interoperability ?

“The ability of computer systems or software to exchange and make use of information”

Web services architecture




(UDDI)
Web Services
Repository
Description
WSDL
3. discovery
(SOAP)
1. Publication
(SOAP)
Client
Web Service Requester
Application
cliente
4. Invocation
(SOAP)
Web Services Provider
Web
Service
Application
Description
WSDL

62

Le scenario complet

Étape 1 : définition, description du service

On doit décrire d'un point de vue informatique ce que fait le service, la solution qu'il propose, ...

La définition est faite en WSDL au sein du fournisseur de services (i.e. le serveur d'applications)

Étape 2 : publication du service

Une fois le service définit et décrit en termes de mise en œuvre, il peut être déclaré dans un annuaire, on parle alors de publication du service afin de le rendre accessible aux clients.

Le scenario complet

Étape 3 : recherche du service

Le client se connecte, sur un annuaire pour effectuer une recherche de service, ou cherche via des API des web services.

Étape 4 : enregistrement au service web

Une fois le service trouvé par le client, ce dernier doit s'enregistrer auprès du fournisseur associé au service. Cet enregistrement indique au fournisseur l'intention du client d'utiliser le service suivant les conditions décrites dans la publication.

Le scenario complet

Étape 5 : mise en œuvre du service

Le client peut invoquer le service suivant les conditions inscrites au sein de l'annuaire lors de la publication du service web (étape 2)

Étape 6 : composition

C'est la possibilité de combiner plusieurs services. En fait, un service peut devenir le client d'un autre service.

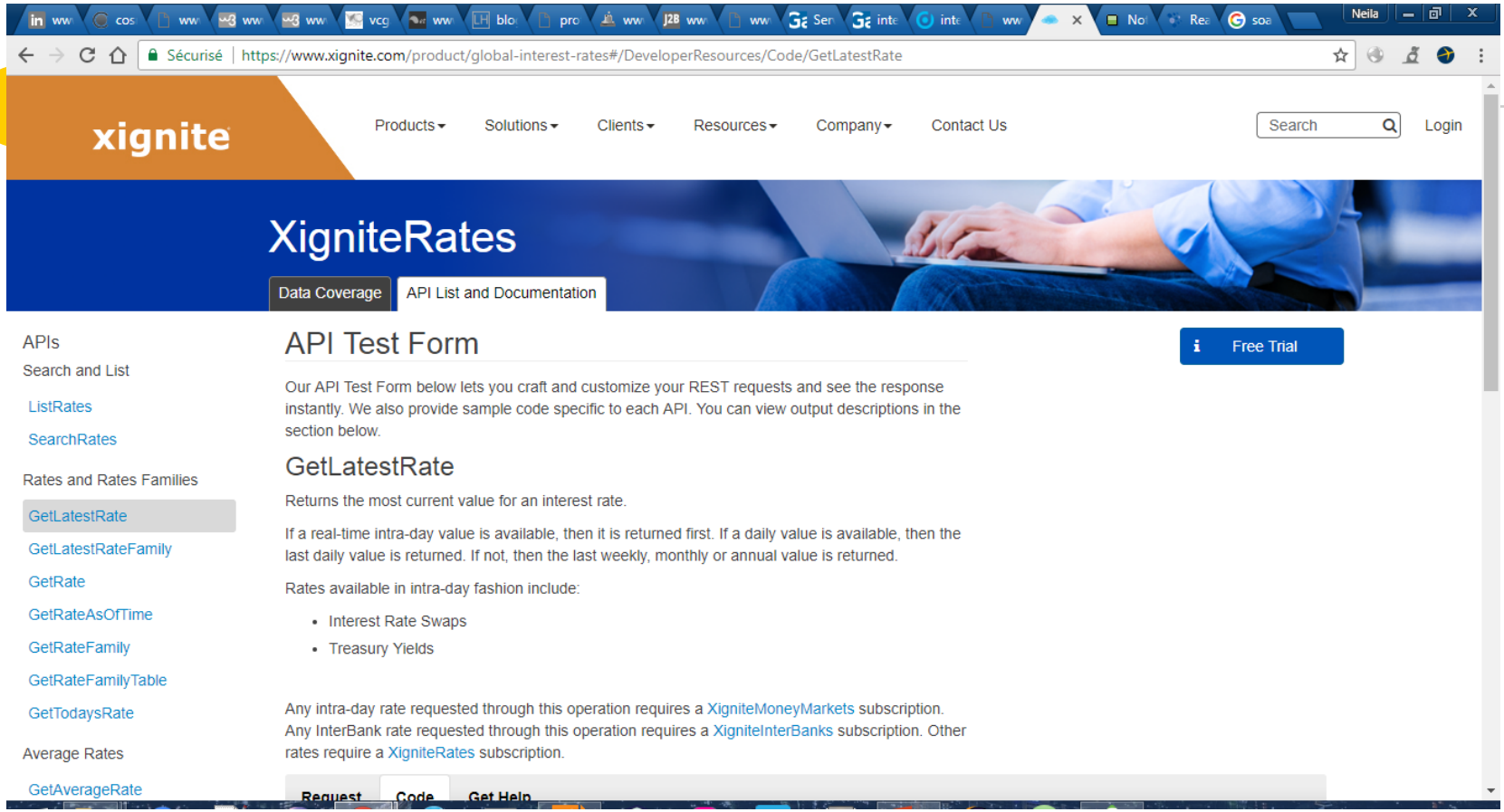
Exemples de Web Service

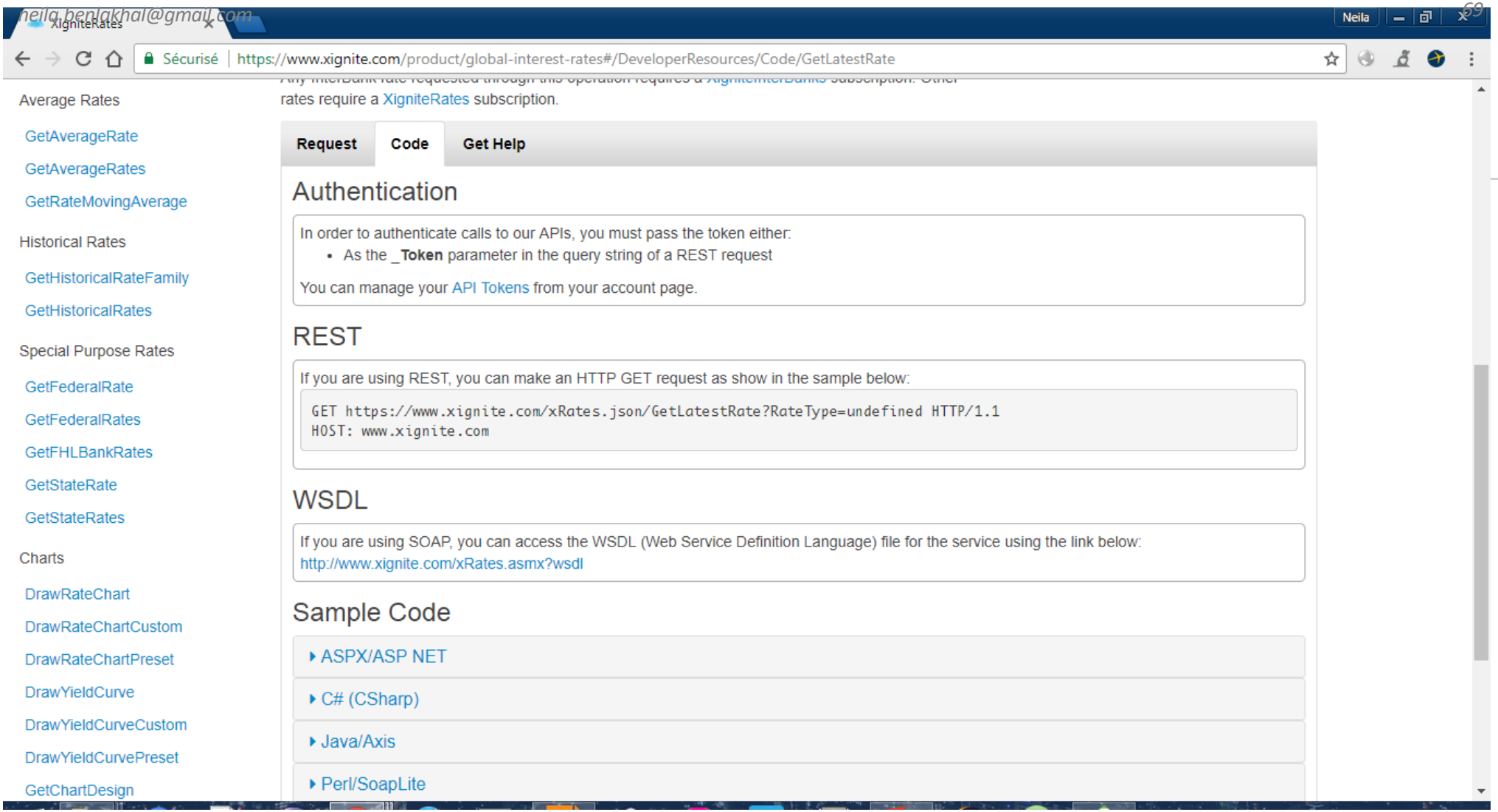
- Un service Web peut par exemple :
  - Récupérer un cours de bourse
  - Faire une demande automatiquement mise à jour d'un prix ;
  - Accéder à un calendrier universel faisant les conversions entre calendriers internationaux et connaissant, pour chaque pays, les dates des jours fériés ;
  - Traduire un passage
  - Valider un numéro international de code postal...
  - …

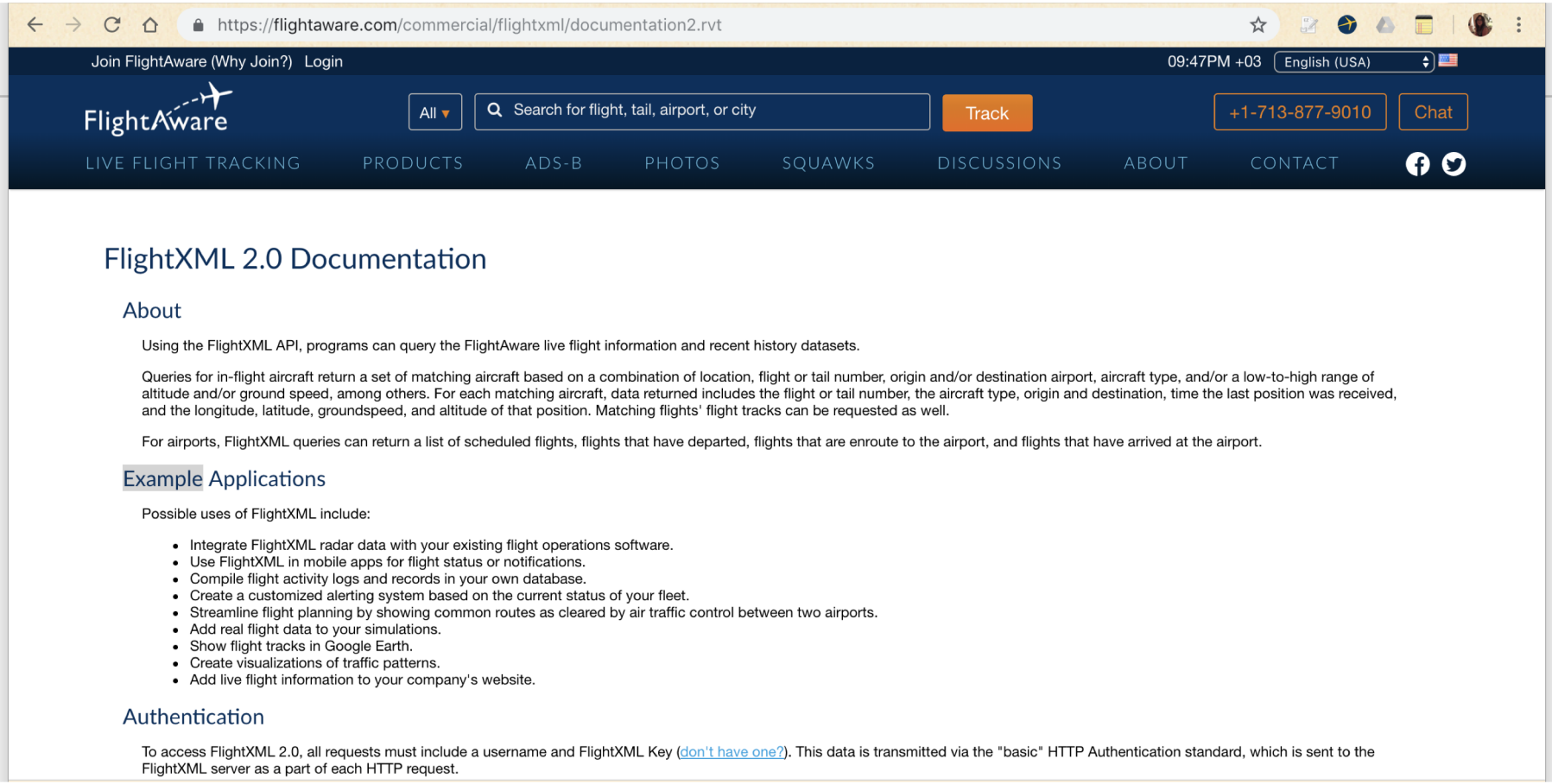
Where to find Web services ?

- Financial web services http://www.xignite.com/
- API directory: https://www.programmableweb.com/
- http://www.cdyne.com/
- http://www.fraudlabs.com/
- Weather web services (https://graphical.weather.gov/xml/SOAP_server/ndfdXMLserver.php?wsdl)
- Flight Trackers(uk.flightaware.com/flightxml/)
- Amazon Web services, Google Maps API Web Services, etc.

https://www.xignite.com/product/global-interest-rates#/DeveloperResources/Code/GetLatestRate




XigniteRates
Neila
Sécurisé | https://www.xignite.com/product/global-interest-rates#/DeveloperResources/Code/GetLatestRate
Average Rates
GetAverageRate
GetAverageRates
GetRateMovingAverage
Historical Rates
GetHistoricalRateFamily
GetHistoricalRates
Special Purpose Rates
GetFederalRate
GetFederalRates
GetFHLBankRates
GetStateRate
GetStateRates
Charts
DrawRateChart
DrawRateChartCustom
DrawRateChartPreset
DrawYieldCurve
DrawYieldCurveCustom
DrawYieldCurvePreset
GetChartDesign
rates require a XigniteRates subscription.
Request
Code
Get Help
Authentication
In order to authenticate calls to our APIs, you must pass the token either:
As the _Token parameter in the query string of a REST request
You can manage your API Tokens from your account page.
REST
If you are using REST, you can make an HTTP GET request as show in the sample below:
GET https://www.xignite.com/xRates.json/GetLatestRate?RateType=undefined HTTP/1.1
HOST: www.xignite.com
WSDL
If you are using SOAP, you can access the WSDL (Web Service Definition Language) file for the service using the link below:
http://www.xignite.com/xRates.asmx?wsdl
Sample Code
ASPX/ASP NET
C# (CSharp)
Java/Axis
Perl/SoapLite


https://flightaware.com/commercial/flightxml/documentation2.rvt
Join FlightAware (Why Join?) Login
09:47PM +03
English (USA)
FlightAware
All
Search for flight, tail, airport, or city
Track
+1-713-877-9010
Chat
LIVE FLIGHT TRACKING
PRODUCTS
ADS-B
PHOTOS
SQUAWKS
DISCUSSIONS
ABOUT
CONTACT
FlightXML 2.0 Documentation
About
Using the FlightXML API, programs can query the FlightAware live flight information and recent history datasets.
Queries for in-flight aircraft return a set of matching aircraft based on a combination of location, flight or tail number, origin and/or destination airport, aircraft type, and/or a low-to-high range of altitude and/or ground speed, among others. For each matching aircraft, data returned includes the flight or tail number, the aircraft type, origin and destination, time the last position was received, and the longitude, latitude, groundspeed, and altitude of that position. Matching flights' flight tracks can be requested as well.
For airports, FlightXML queries can return a list of scheduled flights, flights that have departed, flights that are enroute to the airport, and flights that have arrived at the airport.
Example Applications
Possible uses of FlightXML include:
Integrate FlightXML radar data with your existing flight operations software.
Use FlightXML in mobile apps for flight status or notifications.
Compile flight activity logs and records in your own database.
Create a customized alerting system based on the current status of your fleet.
Streamline flight planning by showing common routes as cleared by air traffic control between two airports.
Add real flight data to your simulations.
Show flight tracks in Google Earth.
Create visualizations of traffic patterns.
Add live flight information to your company's website.
Authentication
To access FlightXML 2.0, all requests must include a username and FlightXML Key (don't have one?). This data is transmitted via the "basic" HTTP Authentication standard, which is sent to the FlightXML server as a part of each HTTP request.

https://flightaware.com/commercial/flightxml/documentation2.rvt

Examples (SOAP / WSDL)

Microsoft .NET

Show example (Visual Studio C#)...

Show example (CLI C#)...

PHP

Hide example (SoapClient)...

The built-in SoapClient provided in PHP5 supports modern Document/Literal SOAP services, like FlightXML2.

Requirements

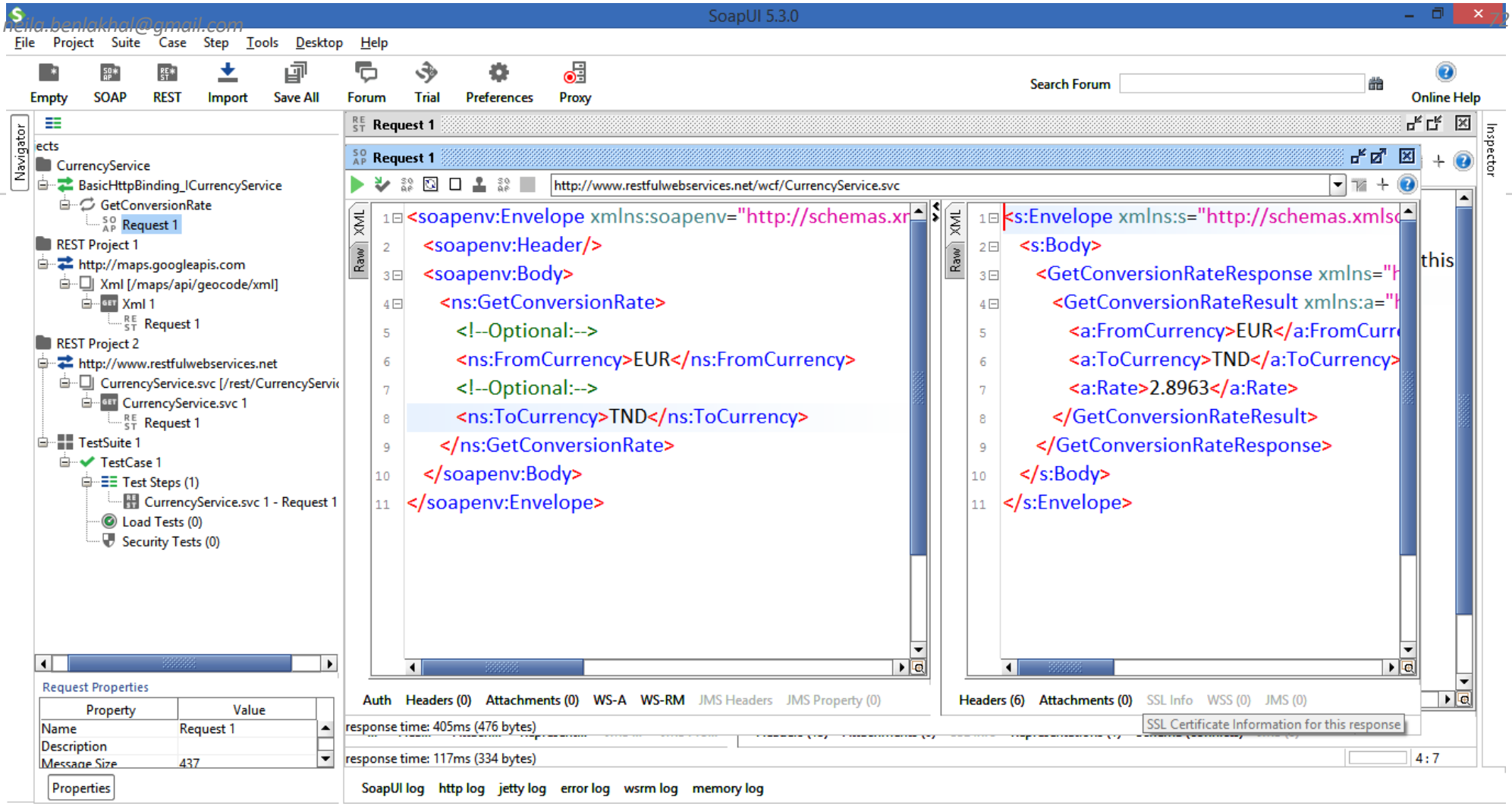
- PHP5 or above
- SoapClient
  - if building PHP from source, it must have been compiled with: ./configure --enable-soap
  - on Ubuntu or Debian based systems: sudo aptitude install php-soap
  - on RHEL/Fedora based systems: yum install php-soap

Save the following file as testsoap.php, but substitute your actual username and API key:

```
<?php

$options = array(
                 'trace' => true,
                 'exceptions' => 0,
                 'login' => 'sampleUser',
                 'password' => 'abc123abc123abc123abc123abc123abc123abc123',
                );
$client = new SoapClient('http://flightxml.flightaware.com/soap/FlightXML2/wsdl', $options);

// get the weather.
$params = array("airport" => "KAUS");
$result = $client->Metar($params);
print_r($result);
```


SoapUI 5.3.0
File Project Suite Case Step Tools Desktop Help
Empty SOAP REST Import Save All Forum Trial Preferences Proxy
Search Forum
Online Help
Navigator
CurrencyService
BasicHttpBinding_ICurrencyService
GetConversionRate
Request 1
REST Project 1
http://maps.googleapis.com
Xml [/maps/api/geocode/xml]
Xml 1
Request 1
REST Project 2
http://www.restfulwebservices.net
CurrencyService.svc [/rest/CurrencyServic
CurrencyService.svc 1
Request 1
TestSuite 1
TestCase 1
Test Steps (1)
CurrencyService.svc 1 - Request 1
Load Tests (0)
Security Tests (0)
Request Properties
Property
Value
Name
Request 1
Description
Message Size
437
Properties
Request 1
Request 1
http://www.restfulwebservices.net/wcf/CurrencyService.svc
XML
Raw
<soapenv:Envelope xmlns:soapenv="http://schemas.xr
<soapenv:Header/>
<soapenv:Body>
<ns:GetConversionRate>
<!--Optional:-->
<ns:FromCurrency>EUR</ns:FromCurrency>
<!--Optional:-->
<ns:ToCurrency>TND</ns:ToCurrency>
</ns:GetConversionRate>
</soapenv:Body>
</soapenv:Envelope>
<s:Envelope xmlns:s="http://schemas.xmlso
<s:Body>
<GetConversionRateResponse xmlns="h
<GetConversionRateResult xmlns:a="h
<a:FromCurrency>EUR</a:FromCurr
<a:ToCurrency>TND</a:ToCurrency>
<a:Rate>2.8963</a:Rate>
</GetConversionRateResult>
</GetConversionRateResponse>
</s:Body>
</s:Envelope>
Auth Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0)
Headers (6) Attachments (0) SSL Info WSS (0) JMS (0)
SSL Certificate Information for this response
response time: 405ms (476 bytes)
response time: 117ms (334 bytes)
4 : 7
SoapUI log http log jetty log error log wsrm log memory log
Inspector