Université Nice SOPHIA ANTIPOLIS

Applications Réparties : Les Services Web

* D'après les cours de

Eric Garcia, Julien Henriet, Didier Donsez,
Jean-Yves Tigli, Stéphane Lavirotte

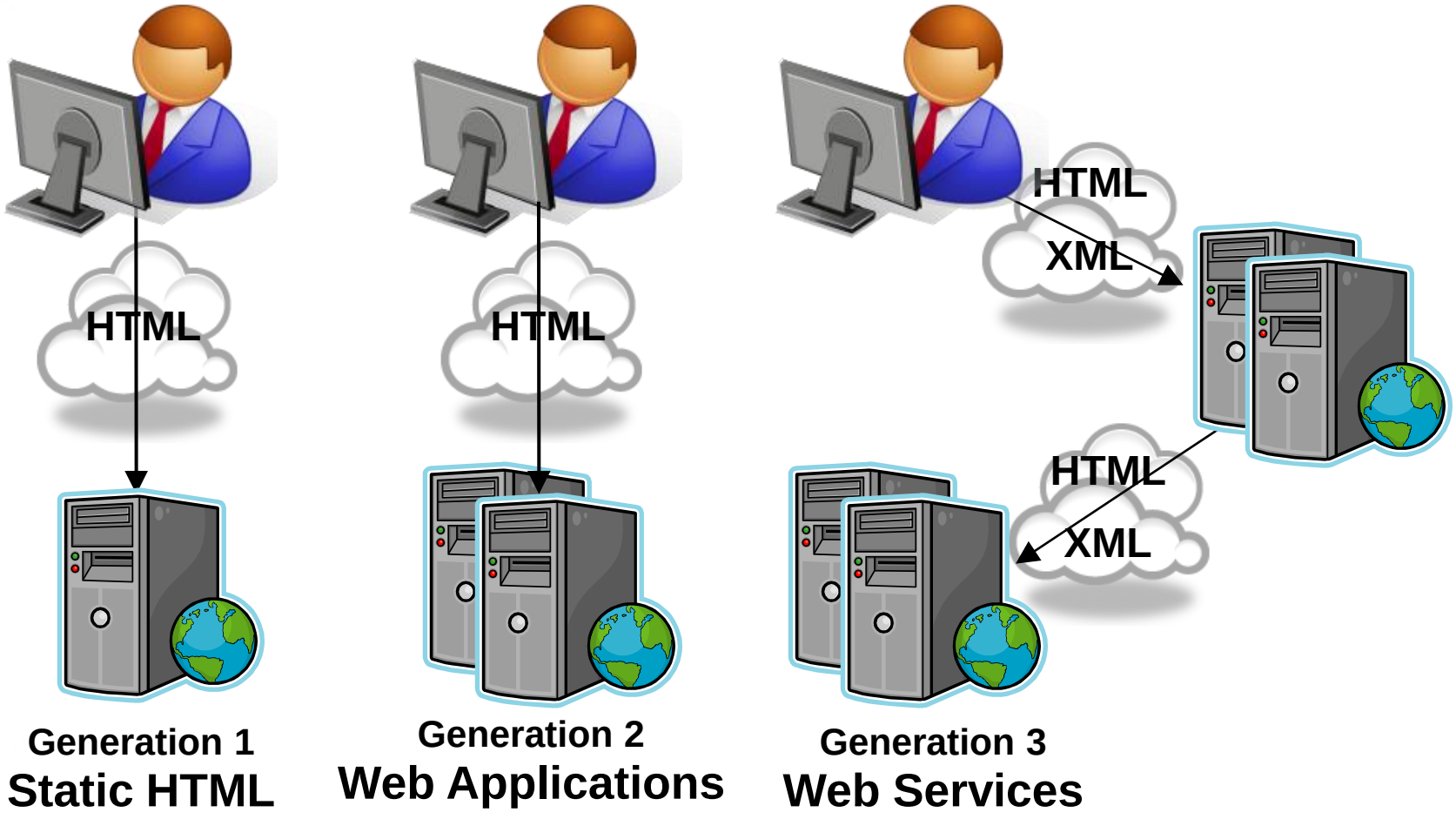
Les Services Web

- ✓ Plan du cours complet:
  - Généralités
  - Architecture
  - Zoom sur les Technologies:
    - SOAP
    - WSDL
    - UDDI
  - Limitations des Services Web « classiques »
  - Conclusion

Généralités

Les Services Web

Evolution du Web



Un Service Web, c'est quoi ?

- ✓ Définition pour le profane : Un service Web est
  - Juste une page Web destinée à être interrogée et traitée par un ordinateur
  - Plus exactement, est une page Web qui est destinée à être consommée par un programme autonome par opposition à un navigateur Web ou tout outil semblable
- ✓ Définition pour l'expert, un Service Web est :
  - Une « unité logique applicative » accessible en utilisant les protocoles standards d'Internet
  - Une «librairie» fournissant des données et des services à d'autres applications
  - Un objet métier qui peut être déployé et combiné sur Internet avec une faible dépendance vis-à-vis des technologies et des protocoles
  - Une combinaison des meilleurs aspects du développement à base de composants et du Web

Caractéristiques d'un Service Web

- ✓ Caractéristiques
  - Architecture de type Client / Serveur
    - Le serveur rend des services à un client
  - Réutilisable
    - Par plusieurs clients (simultanément ou pas)
  - Indépendamment de
    - la plate-forme (UNIX, Windows, ...)
    - du langage pour l'implémentation (VB, C#, Java, ...)
    - la plate-forme de développement sous-jacente (.NET, J2EE, Axis...)

Pour quoi faire ?

- ✓ Les Services Web permettent d'interconnecter :
  - Différentes entreprises
  - Différentes applications
  - Différents clients
  - Différents matériels
- ✓ Utilisé dans différents cadres:
  - B2B (Business To Business)
  - EAI (Enterprise Application Integration)
  - EAI (Enterprise Application Integration)
  - ...

Services Web et Interopérabilité

- ✓ **Plateform independent** : A l'instar de Corba, les Services Web gèrent l'interopérabilité au niveau du protocole d'échange

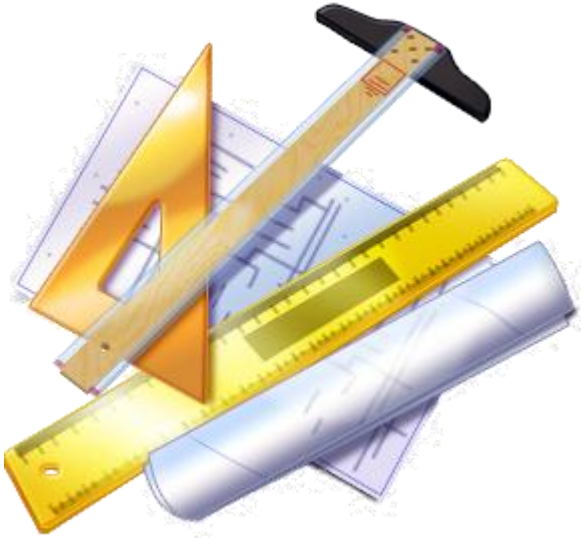
- ✓ **Plateform dependent**: d'autres approches gèrent l'interopérabilité par le portage de la plate-forme d'exécution (ex. OSGi, RMI sur Java et historiquement COM/DCOM)

Quels objectifs ?

- ✓ Remplacer les outils (RPC, DCOM, CORBA, RMI) par une approche entièrement ouverte et interopérable, basée sur la généralisation des serveurs Web avec scripts CGI.
- ✓ Faire interagir des composants hétérogènes, distants, et indépendants avec un protocole standard (SOAP).
- ✓ Passer les politiques de sécurité grâce en grande partie à une couche session basée sur HTTP (port 80).

Et plus concrètement ?

- ✓ Une nouvelle technologie des objets distribués ?
  - Invocation distante des services Web : **SOAP**
  - Description des services Web : **WSDL**
  - Enregistrement et découverte de services Web : **UDDI**
- ✓ Basés sur des standards
  - Standards du W3C : XML, SOAP, WSDL
  - Standards industriels : UDDI, ebXML
- ✓ Implémentations actuelles :
  - Microsoft .Net
  - Sun JavaONE : J2EE + Web services (WSDP = JAXP, JAX-RPC, JAXM...)
  - Apache SOAP / Axis, IBM WSTK
  - Oracle, Bea, Iona, Enhydra ...



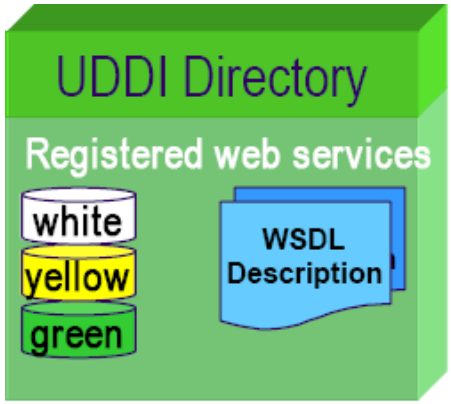
Architecture

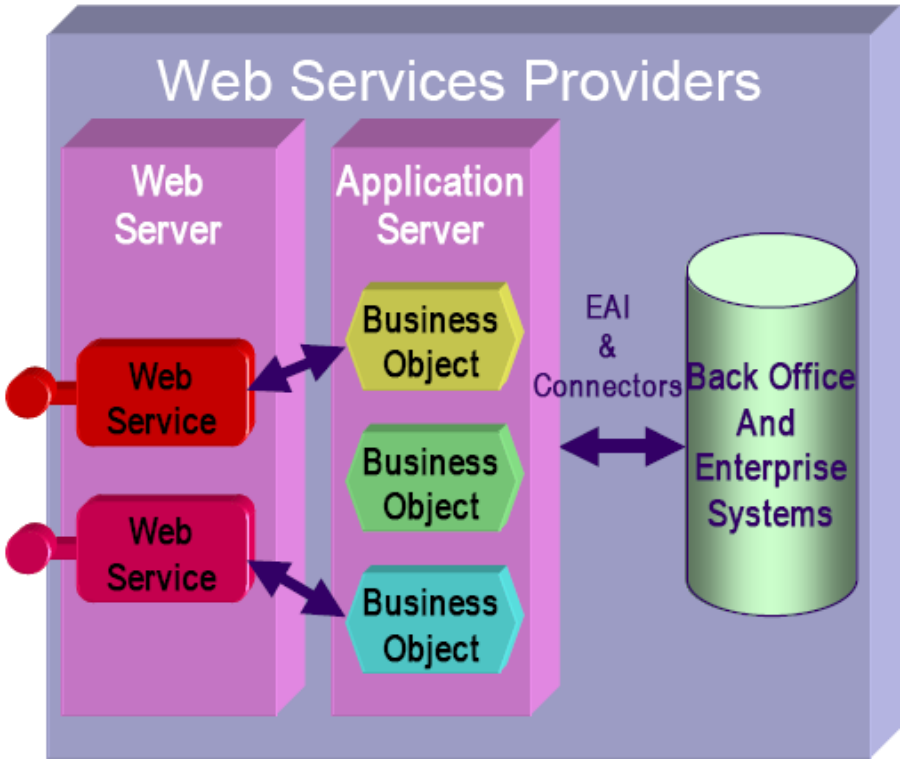
Les Services Web

Cycle de Vie

- ✓ Etape 1 : Déploiement du service Web
  - Dépendant de la plate-forme
- ✓ Etape 2 : Enregistrement du service Web
  - WSDL : description du service
  - Référentiels : DISCO (local), UDDI (global)
- ✓ Etape 3 : Découverte du service Web
- ✓ Etape 4 : Invocation du service Web par le client

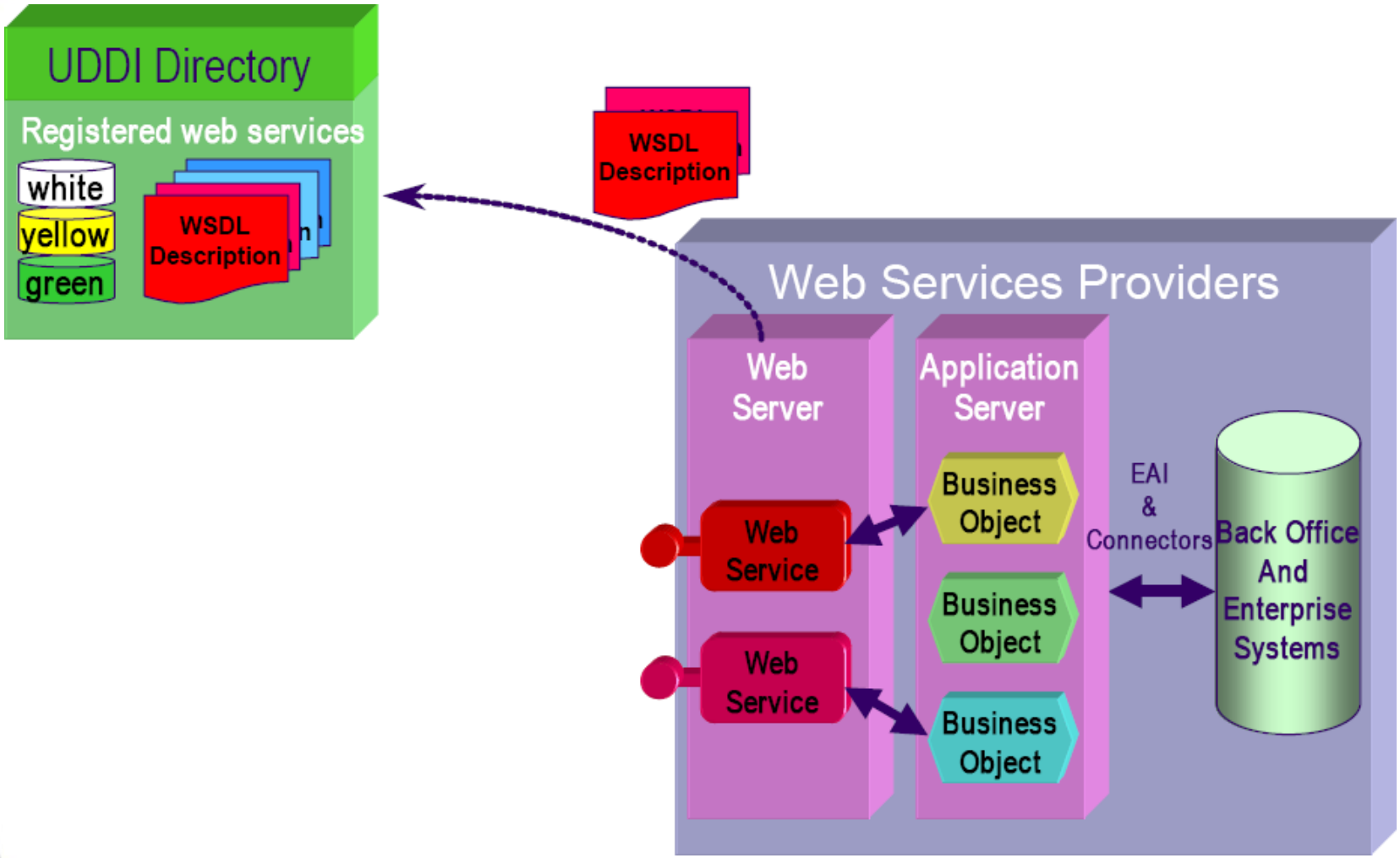
Déploiement du Service Web





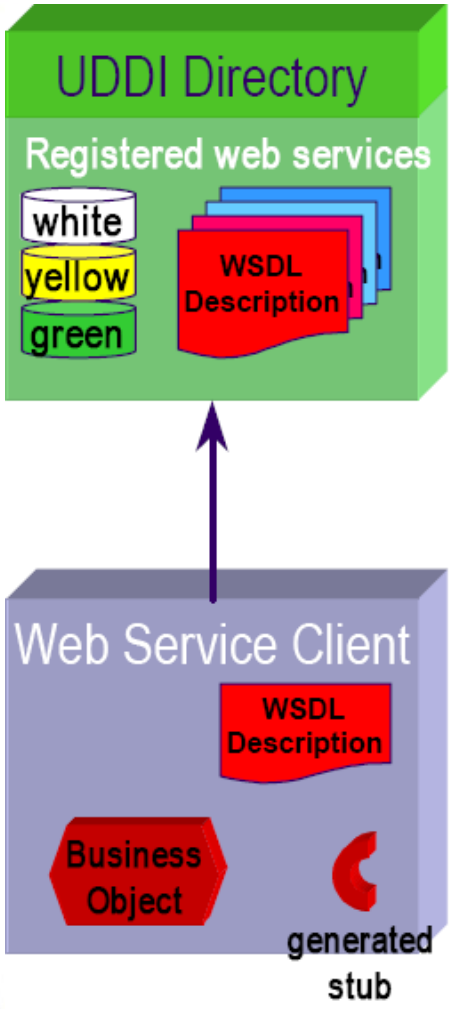
Source: Didier Donsez

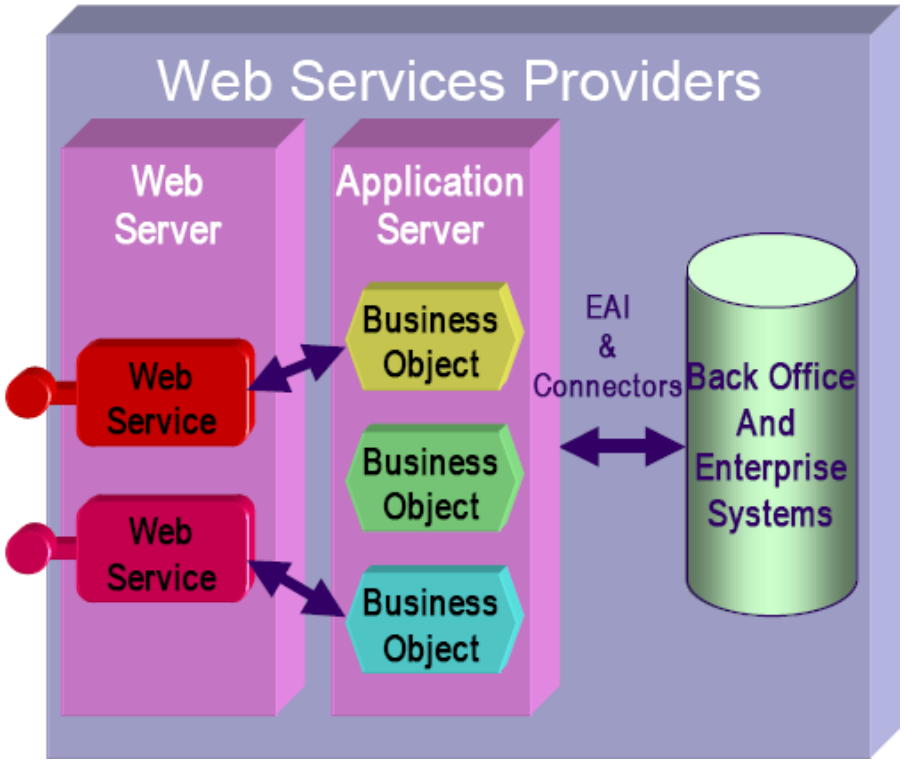
Enregistrement du Service Web



Source: Didier Donsez

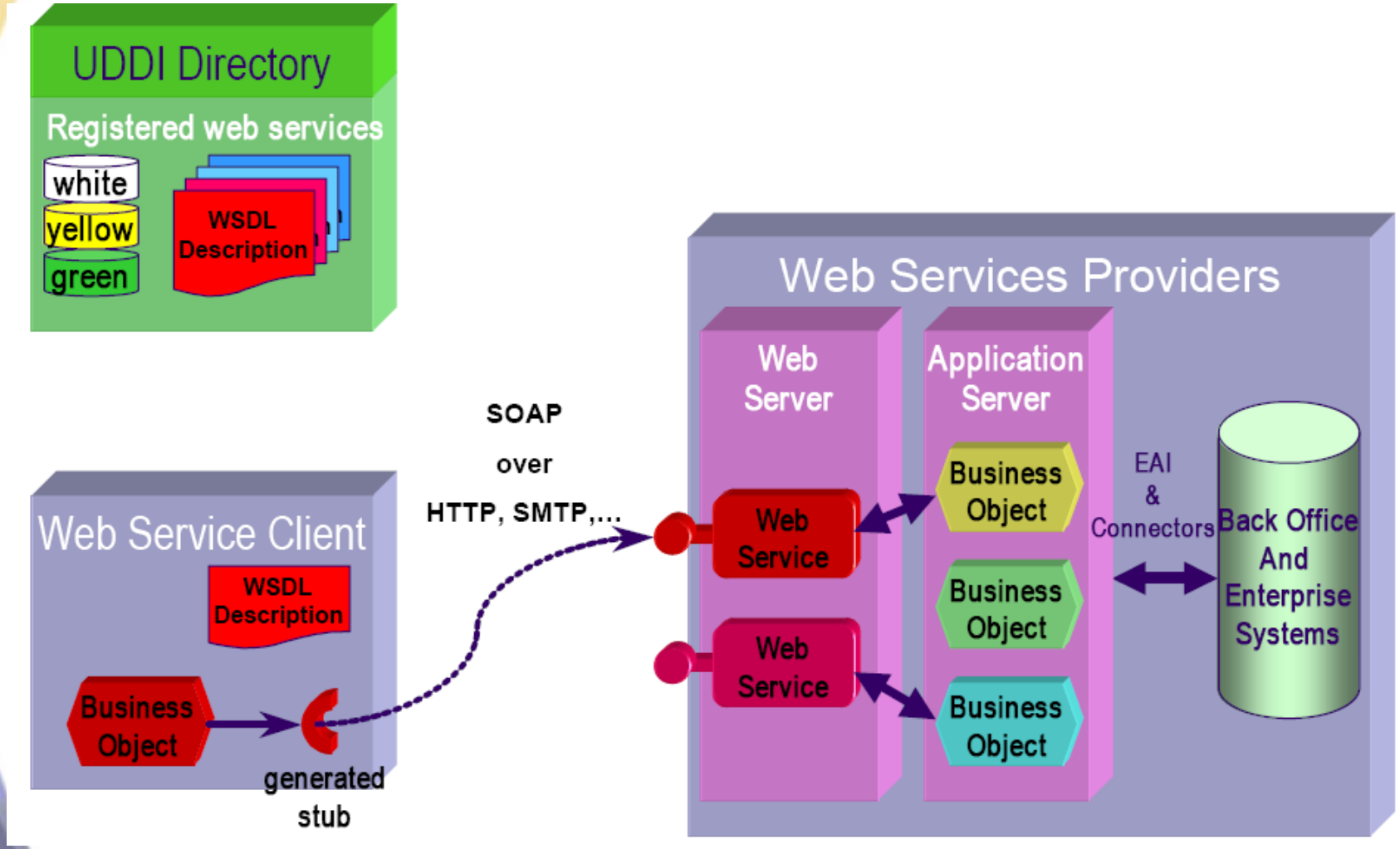
Découverte du Service Web





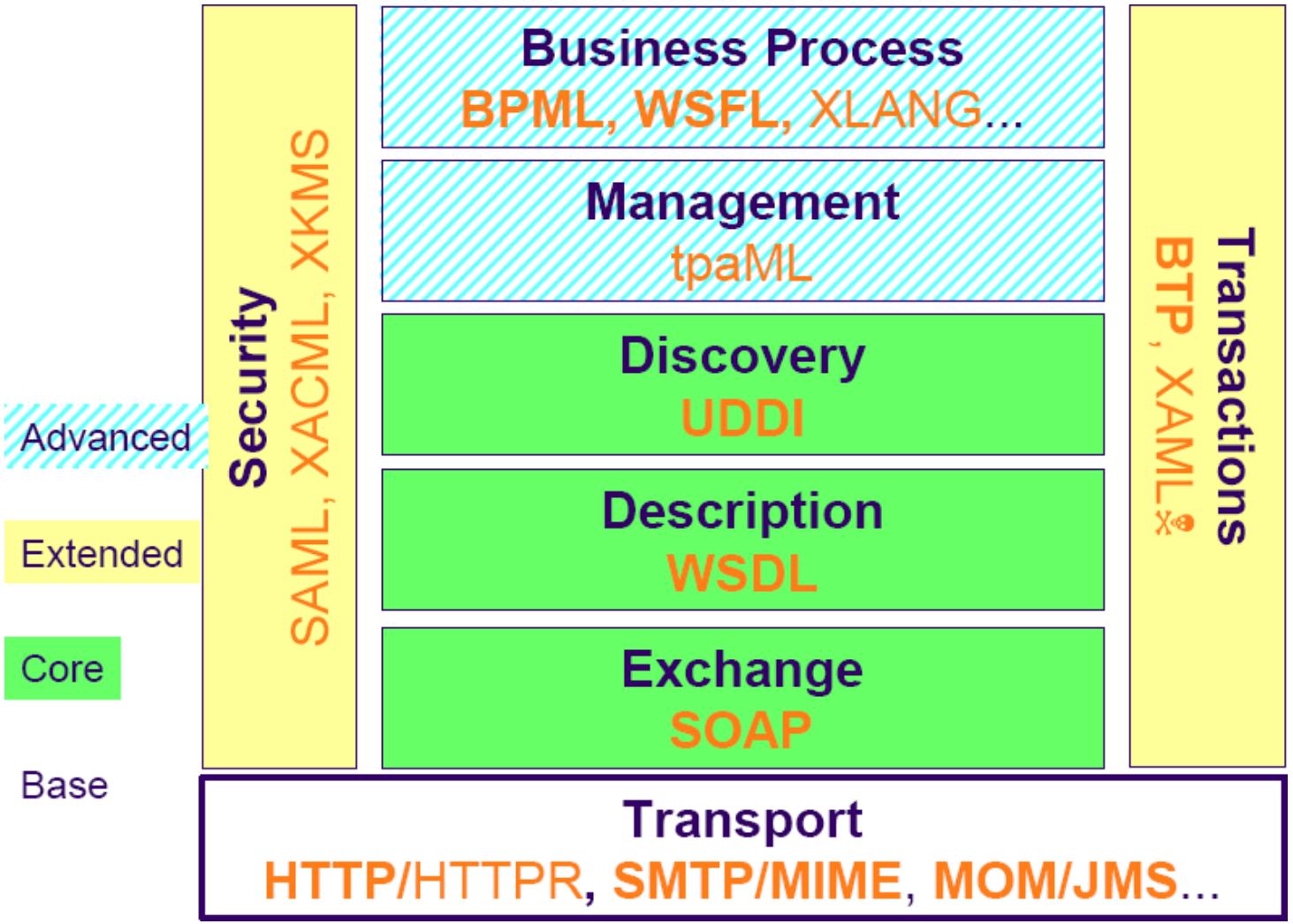
Source: Didier Donsez

Invocation du Service Web



Source: Didier Donsez

Protocoles des Services Web



Source: M. Pontacq

Simple Object Access Protocol

SOAP

Le Web et le Client/Serveur

- ✓ Proposition Web insuffisante pour faire des applications
- ✓ Autres plates-formes Client/Serveur
  - Java RMI
    - mono-langage : Java, multi-plateforme (JVM), SUN
    - Difficile pour une application industrielle (performance, sécurité, ...)
  - CORBA / IIOP
    - Norme définie par l'OMG
    - Multilangage, multi-plateforme, Multi-vendeurs
    - Installation « coûteuse » si on doit acheter un ORB
      - Mais les open-sources sont gratuits et souvent plus complets
        - www.objectweb.org
  - DCOM
    - multi-langages, plateforme Win32, Propriétaire Microsoft
    - Faible diffusion
      - Pas disponible sur MacOS, NT3.51, Win95, WinCE2
      - Coûteux sur UNIX, MVS, VMS ou NT

Le bilan...

- ✓ Approche insatisfaisante :
  - Protocoles sophistiqués
    - Coût d'installation (faite par un administrateur, consomme des ressources : machines, personnels, ...)
    - Difficile à porter sur d'autres plates-formes
  - Règles de fonctionnement strictes en environnement ouvert (le Net)
    - Environnement sécurisé (intérieur d'un intranet)
    - Incapacité à fonctionner en présence de pare-feu (utilisation impossible sur Internet)
      - Les nouvelles versions de CORBA peuvent ouvrir un port sur un pare-feu comme le port 80 d'HTTP

… et ses conséquences

- ✓ Le Web a besoin d'un nouveau protocole
  - Multi-langages, multi-plateformes
  - Respectant les formats d'échanges du Web
    - Réponses et requêtes en XML
  - Facile à implémenter sur différents protocoles de transport
    - RPC, HTTP ou autre MOM
  - Permettant de franchir les politiques de sécurité (firewalls et proxies)
  - Avec une spécification non propriétaire garantie par un organisme indépendant
    - W3C
- ✓ La réponse : SOAP (Simple Object Access Protocol)

La philosophie SOAP

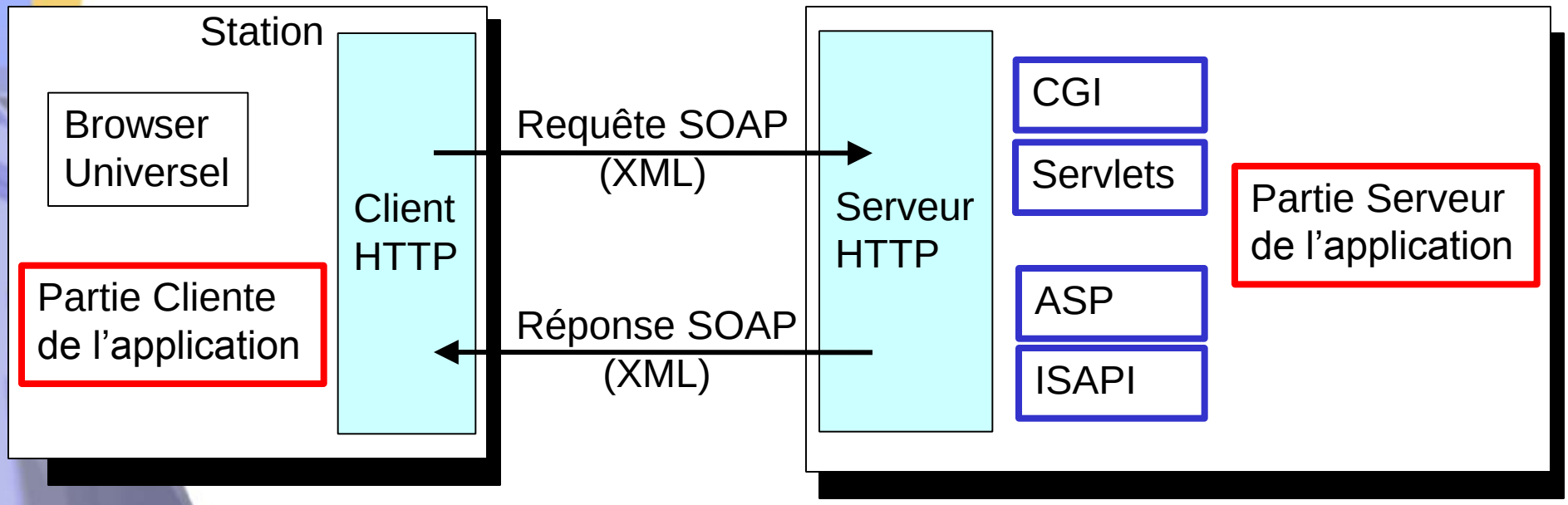
- ✓ SOAP codifie simplement une partie existante
  - Utilisation conjointe de XML et HTTP
- ✓ SOAP est un protocole minimal pour appeler les méthodes sur des serveurs, services, composants, objets
  - Ne pas imposer une API ou un Run-Time
  - Ne pas imposer l'utilisation d'un ORB (CORBA, DCOM, ...) ou d'un serveur WEB particulier (Apache, IIS, ...)
  - Ne pas imposer un modèle de programmation
  - Ne pas réinventer une nouvelle technologie
- ✓ SOAP a été construit pour pouvoir être aisément porté sur toutes les plates-formes et les technologies

Les 3 aspects d'un appel SOAP

- ✓ SOAP peut être vu comme un autre RPC Objects
  - Les requêtes contiennent des paramètres IN et INOUT
  - Les réponses contiennent des paramètres INOUT et OUT
- ✓ SOAP peut être vu comme un protocole d'échange de messages
  - La requête contient un seul message
  - La réponse contient un seul message
- ✓ SOAP peut être vu comme un format d'échange de documents
  - La requête contient un document XML
  - Le serveur retourne une version transformée
- ✓ Ces vues ne sont pas imposées par le protocole

En résumé

- ✓ SOAP = HTTP + XML



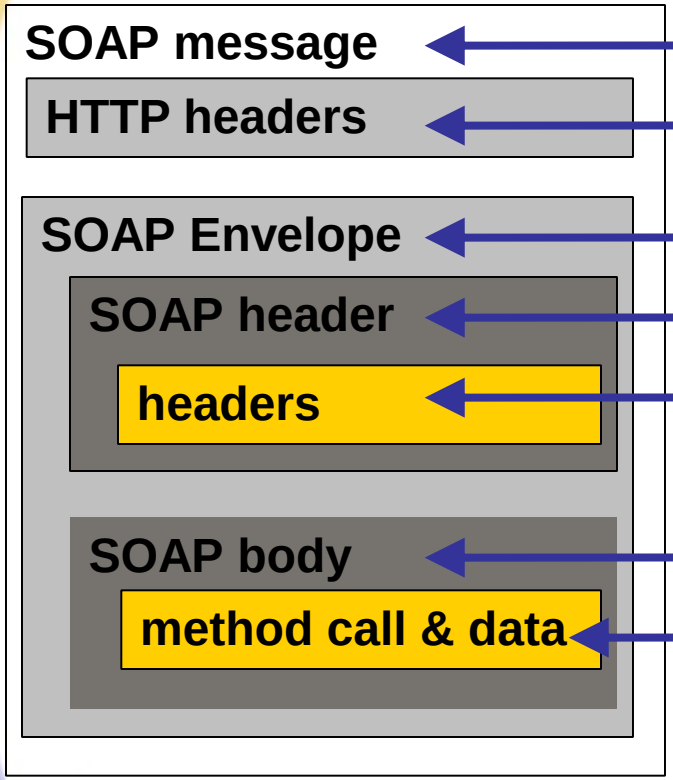
Pourquoi utiliser HTTP ?

- ✓ HTTP
  - est devenu de facto le protocole de communication d'Internet
  - est disponible sur toutes les plateformes
  - est un protocole simple, qui ne requiert que peu de support pour fonctionner correctement
  - passe de manière standard à travers les politiques de sécurité (firewalls et proxies)

Pourquoi utiliser XML ?

- ✓ Utilise du texte (peut être lu et écrit directement)
  - Attention : le texte est globalement peu lisible et vite complexe pour un humain
- ✓ Construire correctement du texte XML est simple
- ✓ XML est aujourd'hui adopté par tous les acteurs d'Internet : plateformes, éditeurs, etc.
- ✓ XML permet aujourd'hui de typer et de structurer les informations
  - L'information peut être sauvegardée n'importe où sur le Net
  - Les données fournies par de multiples sources peuvent être agrégées en une seule unité
  - Chaque partie a sa propre structure XML
  - Chaque partie peut définir des types spécifiques

La structure des messages SOAP



Message complet

Entête standard HTTP et entête SOAP HTTP

Enveloppe

Entête

Entête individuelle

Corps contenant les appels de méthodes SOAP

Appel de méthode et description en XML des données

Exemple de requête SOAP utilisant HTTP

- ✓ Demande de cotation à un serveur :

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAP-Action: "Some-URI"
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/enveloppe/"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
     <m:GetLastTradePrice xmlns:m="Some-URI">
            <symbol>DIS</symbol>
     </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Exemple de réponse SOAP utilisant HTTP

- ✓ Réponse du serveur

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
     <m:GetLastTradePrice xmlns:m="Some-URI">
           <Price>34.5</Price>
     </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Modèle de message

- ✓ SOAP permet une communication par message
  - d'un expéditeur vers un récepteur
- ✓ Structure d'un message
  - Envelop (enveloppe)
    - Elément racine
    - Namespace : SOAP-ENV http://schemas.xmlsoap.org/soap/envelope/
  - Header (entête)
    - Elément optionnel
    - Contient des entrées non applicatives (transactions, session, ...)
  - Body (corps)
    - Contient les entrées du message
      - Nom d'une procédure, valeur des paramètres, valeur de retour
    - Peut contenir les éléments « fault » (erreurs)

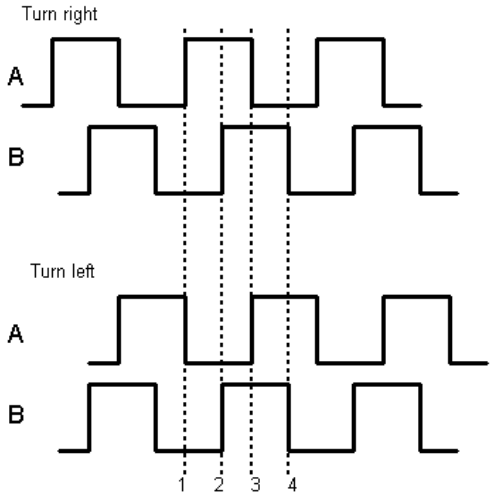
Entête d'un message: Header

- ✓ Contient des entrées non applicatives
  - Transactions, sessions, ...
- ✓ L'attribut mustUnderstand
  - Si absent ou =0 l'élément est optionnel pour l'application réceptrice
  - si =1, l'élément doit être compris par l'application réceptrice sinon le traitement du message par le récepteur doit échouer
- ✓ Exemple

```
<SOAP-ENV:Header>
    <t:Transaction xmlns:t="Some-URI" SOAP-ENV:mustUnderstand="1">
    </t:Transaction>
</SOAP-ENV:Header>
```

Corps d'un message

- ✓ Contient des entrées applicatives
- ✓ Encodage des entrées
- ✓ Namespace pour l'encodage
  - SOAP-ENC http://schemas.xmlsoap.org/soap/encoding/
  - XSD : XML Schema
- ✓ Principe des règles d'encodage
  - Les règles d'encodage définissent un système de type
    - SOAP utilise les conventions XSD
    - Les tableaux et les références sont typés de manière spécifique en utilisant XSD



Règles d'Encodage

Types Primitifs

- ✓ Types primitifs

```
<element name="age" type="int/>
<element name="price" type="float"/>
<element name="displacement" type="negativeInteger"/>
<element name="greeting" id="#id1" type="xsd:string"/>
```

- ✓ Utilisation / Instance

```
<age>45</age>
<price>15.57</price>
<displacement>-450</displacement)
<greeting>Hello</greeting>
```

Enumération

- ✓ **Enumération**

```
<element name="color">
  <simpleType base="xsd:string">
      <enumeration value="green"/>
      <enumeration value="blue"/>
  </simpleType>
</element>
```

- ✓ **Utilisation / Instance**

```
<color>blue</color>
```

Références

- ✓ Références

```
<element name="salutation" type="xsd:string"\>
<salutation href="#id1"/>
<e:book>
   <title>My life and work</title>
   <author href="#Person-1"/>
</e:book>
```

- ✓ Utilisation

```
<e:Person id="Person-1"/>
   <name>Henry Ford</name>
   <address xsi:type="m:Electronic-address"/>
         <email>mailto:henryford@hotmail.com</email>
         <web>http://www.henryford.com</web>
   </address>
</e:Person>
```

Types Complexes: Structures

- ✓ Structures

```
<element name="book">
    <complexType>
        <element name="author" type="xsd:string"/>
        <element name="title" type="xsd:string"/>
    </complexType>
</element>
```

- ✓ Utilisation / Instance

```
<e:book>
    <author>J.R.R. Tolkien</author>
    <title>A hobbit story</title>
</e:book>
```

Types Complexes: Tableaux

- ✓ Tableaux

```
<SOAP-ENC:Array id="id3" SOAP-ENC:arrayType=xsd:string[2,2]/>
  <item>r1c1</item>
  <item>r1c2</item>
  <item>r2c1</item>
  <item>r2c2</item>
<SOAP-ENC:Array>
```

- ✓ Tableaux d 'octets

```
<picture xsi:type="SOAP-ENC:base64">
  aG913IGF0
</picture>
```

- ✓ Tableaux creux

```
<SOAP-ENC:Array id="array-1" SOAP-ENC:arrayType=xsd:string[10,10]>
  <item SOAP-ENC:position="[2,2]">Third row, third col</item>
  <item SOAP-ENC:position="[7,2]">Eigth row, third col</item>
</SOAP-ENC:Array>
```

Retour d'Erreurs

- ✓ 4 éléments
  - faultcode (obligatoire)
    - Code d'erreur utilisé par le logiciel (switch(faultcode) { case ...
  - faultstring (obligatoire)
    - Explication lisible d' un humain
  - faultactor (optionel)
    - Erreur en cours de cheminement du message (firewall, proxy, MOM)
  - Detail
    - Détail de l' erreur non lié au Body du message
  - Autres
    - D' autres éléments qualifiés par un namespace peuvent être ajoutés
- ✓ Faultcode
  - 4 groupes de code d' erreur
    - Client, Server, MustUnderstand, VersionMismatch
    - Ex: Client.Authentication

Exemple de Retour d'Erreur

- ✓ Erreur sur le corps:

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf8"
Content-Length: nnnn

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/enveloppe/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Server Error</faultstring>
      <detail>
        <e:myfaultdetails xmlns:e="Some-URI">
          <message>my application didn't work</message>
          <errorcode>1001</errorcode>
        </e:mydefaultdetails>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Les Extensions de SOAP

« Contourner » les limitations

Limitations de SOAP…

- ✓ Sécurité :
  - Limité à la sécurisation de HTTP ?
- ✓ Transfert de données :
  - Données échangées en XML. Donc
    - Données multi-parties ?
    - Données binaires ?
  - Fournir ou consommer des flux de données ?
- ✓ Des extensions pour répondre à ces problématiques

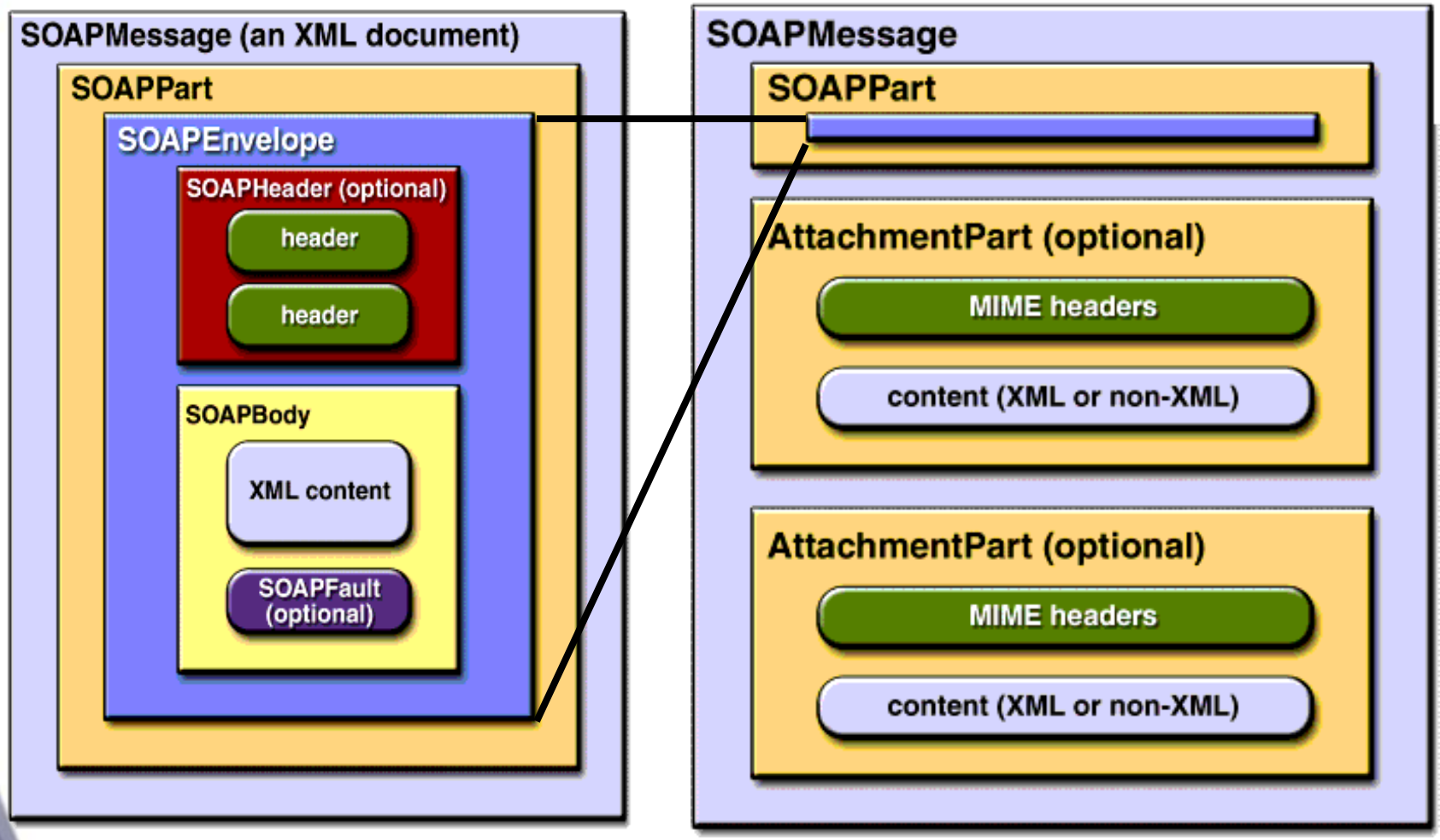
Sécurité

- ✓ Une sécurité « simple » mais efficace…
  - Basée sur la sécurité dans HTTP
    - HTTPS: assure une sécurité en point à point
    - S'intègre bien avec la politique des firewall
  - Pas de transfert de code applicatif
    - Uniquement des données
- ✓ … Mais ne répond pas à tous les problématiques
  - Repousse le problème des failles
    - Déni de service, failles applicatives, …
  - Pas un moyen de sécuriser de bout en bout dans des échanges
- ✓ Donc d'autres standards nécessaires
  - SAML 2.0: Security Assertion Markup Language (OASIS)
  - XACML 2.0: eXtensible Access Control Markup Language (OASIS)

Attachements avec SOAP

- ✓ SOAP with Attachments (SwA) ou MIME for Web Services
  - Utilise MIME
- ✓ MIME: Multipurpose Internet Mail Extensions
  - Pour le transport de gros fichiers
  - Placés en dehors de la partie XML
  - Avantages :
    - Simple
  - Inconvénients :
    - Impossible de faire du streaming
    - Le séparateur entre deux « attachements » MIME est une chaîne de caractères
    - Pas standardisé (juste une note du W3C)

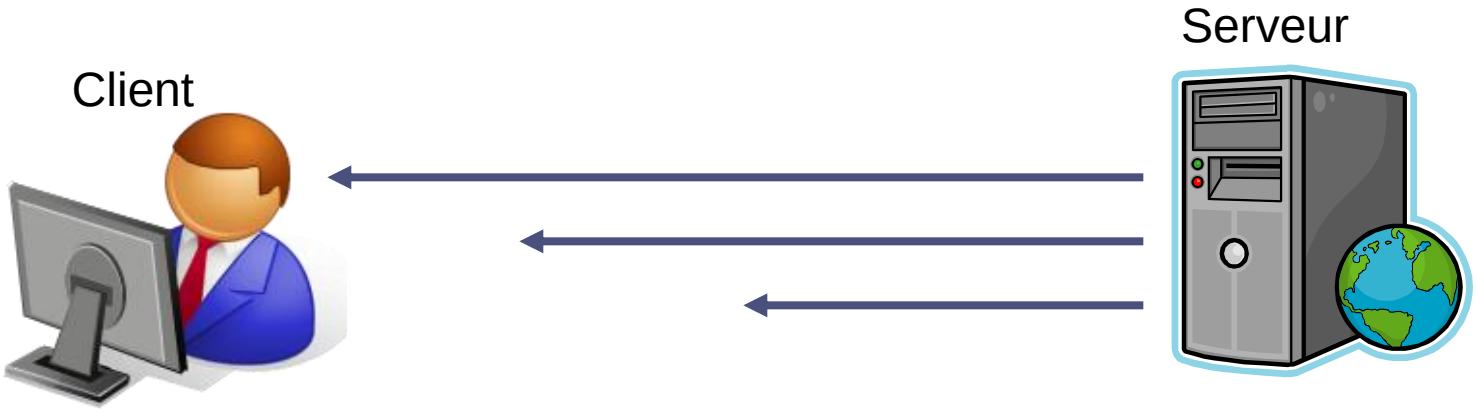
Les Attachements MIME



Les Attachements DIME

- ✓ DIME: Direct Internet Message Encapsulation
  - Proposé par Microsoft au début des années 2000
  - Pour le transport de flux continus
  - Placés en dehors de la partie XML
  - A chaque stream correspond un "DIME-record"
  - Un flag avertit le récepteur de la réception du dernier stream
    - Ainsi, il est possible de commencer à envoyer les premiers streams du flux avant même que tous les streams soient créés par la source
- ✓ Avantages
  - Simplicité
  - Permet la transmission de flux de données (binaires)
- ✓ Inconvénients
  - Impossible à débugguer (données binaires)
  - N'a jamais été standardisé (ni IETF, ni OASIS, ni W3C)

Autres extensions : les attachements DIME



Un DIME-Parser analyse et reconstitue le flux à partir des DIME-records envoyés par le serveur

Un DIME-generator crée les streams et constitue les DIME-records à envoyer au client

Les attachements MTOM

- ✓ Message Transmission Optimization Mechanism :
  - Méthode pour l'envoi efficace de données binaires
  - Recommandation du W3C
  - Supplante SwA et DIME pour les Web Services
- ✓ Détails:
  - Abstract SOAP Transmission Optimization Feature
  - Optimized MIME Multipart/Related Serialization of SOAP Messages (XOP)
  - HTTP SOAP Transmission Optimization Feature
- ✓ Avantages:
  - Fournit un moyen d'envoyer des données binaires dans leur format d'origine (évitant l'augmentation des infos transmises)
    - La transformation en base64 augmente la taille des données de 33%

Portée et Limitations de SOAP

- ✓ SOAP est simple et extensible...
  - Format XML over HTTP
  - Multi-langages
  - Multi-plateformes
- ✓ ... mais il ne couvre pas les fonctions suivantes :
  - Distributed garbage collection
  - Boxcarring or batching of messages
  - Objects-by-reference (qui requière distributed garbage collection)
  - Activation (qui requière objects-by-reference)

Comparaison

	RMI	RPC	DCOM	CORBA	SOAP
Qui	SUN	SUN/OSF	MicroSoft	OMG	W3C
Plate-formes	Multi	Multi	Win32	Multi	Multi
Langages de Programmation	Java	C, C++, …	C++, VB, VJ, OPascal, …	Multi	Multi
Langages de Définition de Service	Java	RPCGEN	ODL	IDL	XML
Réseau	TCP, HTTP, IIOP customisable	TCP, UDP	IP/IPX	GIOP, IIOP, Pluggable Transport Layer	RPC,HTTP SNMP
Firewall	Tunneling HTTP			HTTP Tunneling CORBA Firewall	HTTP
Nommage	RMI, JNDI,JINI	IP+Port	IP+Nom	COS Naming COS Trader	IP+Port, URL
Transaction	Non	Non	MTS	OTS, XA	Extension applicative dans le header
Extra	Chargement dynamique des classes			Services Communs Services Sectoriels	

Web Services Description Language

WSDL

WSDL

- ✓ Spécification
  - v1.1 pas « approuvée » par le W3C (note du 15-03-2001)
    - Soutenu par Ariba, IBM, Microsoft
  - v1.2 « Working Draft » du W3C (11-06-2003)
  - v2.0 recommandation du W3C (27-06-2007)
- ✓ Objectif
  - Interface publique d'accès à un Web Service
  - Comment communiquer pour utiliser le service (ensemble d'opérations et de messages abstraits reliés (bind) à des protocoles et des serveurs réseaux)
- ✓ Grammaire XML (schema XML)
  - Modulaire (import d'autres documents WSDL et XSD)
- ✓ Séparation entre la partie abstraite et concrète

WSDL 1.1

- ✓ <types>
  - Contient les définitions des types (utilise un système de typage comme XSD)
- ✓ <message>
  - Décrit les noms et types d'un ensemble de champs à transmettre
    - Paramètres d'une invocation, valeur du retour, ...
- ✓ <portType>
  - Décrit un ensemble d'opérations et les messages impliqués (0 ou 1 en entrée, 0 ou n en sortie). Partie la plus importante
- ✓ <binding>
  - Spécifie une liaison d'un <porttype> à un protocole concret (SOAP1.1, HTTP1.1, MIME, ...). Un portType peut avoir plusieurs liaisons !
- ✓ <port>
  - Spécifie un point d'entrée (endpoint) comme la combinaison d'un <binding> et d'une adresse réseau
- ✓ <service>
  - Pour aggréger un ensemble de ports

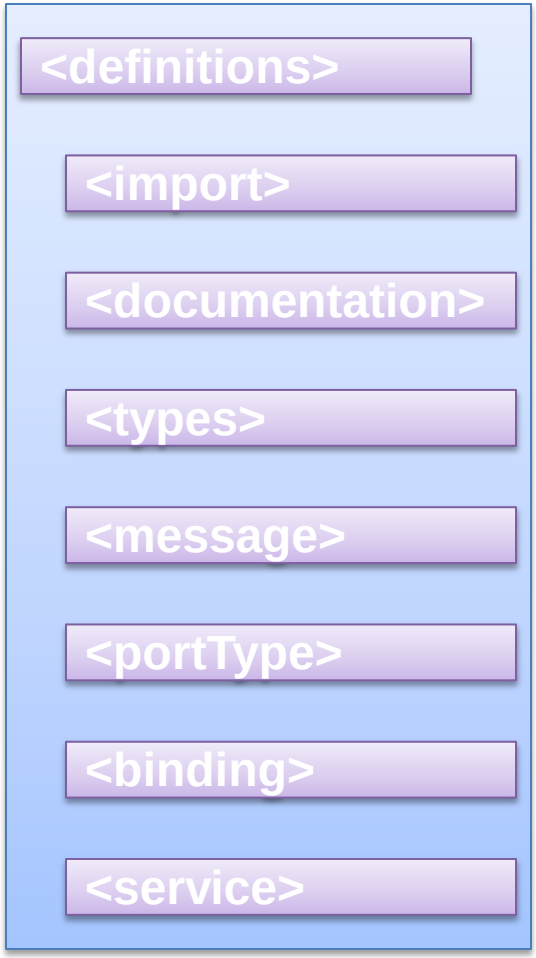
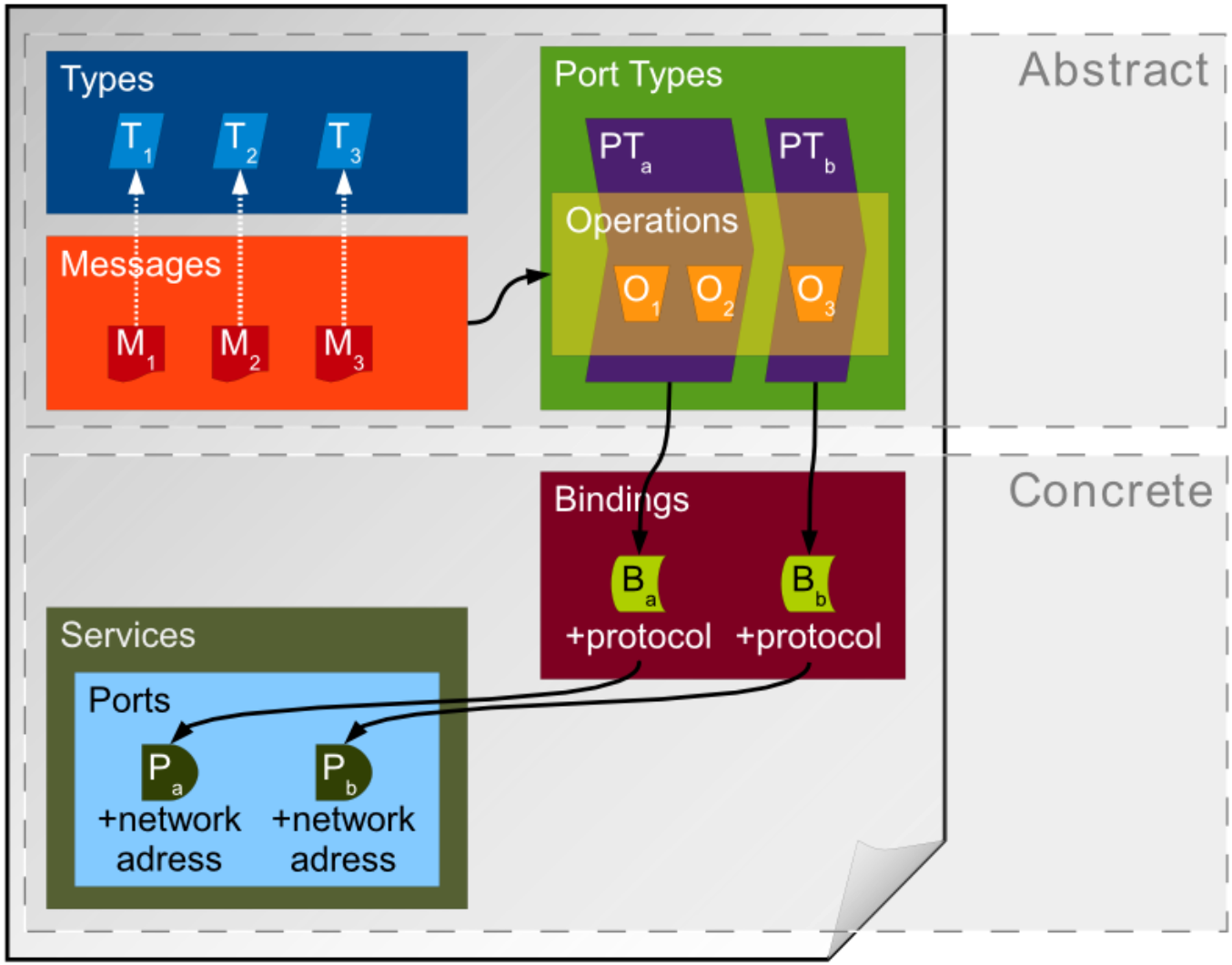


Schéma de WSDL 1.1



Élément <types>

- ✓ Contient les définition de types utilisant un système de typage (comme XSD).
- ✓ Exemple

```
<!– type defs –>
<types>
  <xsd:schema targetNamespace="urn:xml-soap-address-demo"
                          xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <xsd:complexType name="phone">
      <xsd:element name="areaCode" type="xsd:int"/>
      <xsd:element name="exchange" type="xsd:string"/>
      <xsd:element name="number" type="xsd:string"/>
    </xsd:complexType>
    <xsd:complexType name="address">
      <xsd:element name="streetNum" type="xsd:int"/>
      <xsd:element name="streetName" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
      <xsd:element name="state" type="xsd:string"/>
      <xsd:element name="zip" type="xsd:int"/>
      <xsd:element name="phoneNumber" type="typens:phone"/>
    </xsd:complexType>
  </xsd:schema>
</types>
```

Élément <message>

- ✓ Décrit les noms et types d'un ensemble de champs à transmettre
  - Paramètres d'une invocation, valeur du retour, ...
- ✓ Exemple

```
<!-- message declns -->
<message name="AddEntryRequest">
   <part name="name" type="xsd:string"/>
   <part name="address" type="typens:address"/>
</message>

<message name="GetAddressFromNameRequest">
   <part name="name" type="xsd:string"/>
</message>

<message name="GetAddressFromNameResponse">
   <part name="address" type="typens:address"/>
</message>
```

Élément <porttype>

- Décrit un ensemble d'opérations (peut être vu comme une librairie, un module ou une classe)
- Plusieurs types d'opérations
  - One-way
    - Le point d'entrée reçoit un message (<input>) mais sans réponse
  - Request-response
    - Le point d'entrée reçoit un message (<input>) et retourne un message corrélé (<output>) ou un ou plusieurs messages de faute (<fault>).
  - Solicit-response
    - Le point d'entrée envoie un message (<output>) et reçoit un message corrélé (<input>) ou un ou plusieurs messages de faute (<fault>).
      - Binding HTTP : 2 requêtes HTTP par exemple
  - Notification
    - Le point d'entrée envoie un message de notification (<output>)
- Paramètres
  - Les champs des messages constituent les paramètres (in,out, inout) des opérations

Exemple <porttype>

- Exemple

```
<!– port type declarations –>
<portType name="AddressBook">

  <!– One way operation –>
  <operation name="addEntry">
    <input message="AddEntryRequest"/>
  </operation>

  <!– Request-Response operation –>
  <operation name="getAddressFromName">
     <input message="GetAddressFromNameRequest"/>
     <output message="GetAddressFromNameResponse"/>
  </operation>

</portType>
```

Élément <binding>

- ✓ Spécifie une liaison d'un <portType> à un protocole concret (SOAP1.1, HTTP1.1, MIME, ...)

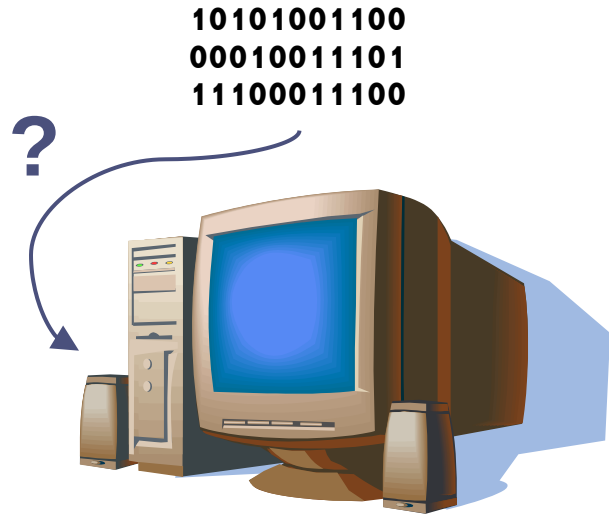
```
<!– binding declns –>
<binding name="AddressBookSOAPBinding" type="AddressBook">
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="addEntry">
  <soap:operation soapAction=""/>
  <input> <soap:body use="encoded" namespace="urn:AddressFetcher2"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/> </input>
  <output> <soap:body use="encoded" namespace="urn:AddressFetcher2"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/> </output>
 </operation>
 <operation name="getAddressFromName">
  <soap:operation soapAction=""/>
  <input> <soap:body use="encoded" namespace="urn:AddressFetcher2"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
  <output> <soap:body use="encoded" namespace="urn:AddressFetcher2"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
 </operation>
</binding>
```

Élément <service>

- ✓ Une collection de points d'entrée (endpoint) relatifs
- ✓ Exemple

```
<?xml version="1.0" ?>
<definitions name="urn:AddressFetcher"
        targetNamespace="urn:AddressFetcher2"
        xmlns:typens="urn:xml-soap-address-demo"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
            xmlns="http://schemas.xmlsoap.org/wsdl/">

  ...
  <!-- service decln -->
  <service name="AddressBookService">
    <port name="AddressBook" binding="AddressBookSOAPBinding">
      <soap:address location="http://www.mycomp.com/soap/servlet/rpcrouter"/>
    </port>
  </service>
</definitions>
```



Environnements de Programmation

Un exemple .NET C#

Des Environnements pour Programmer les Web Services

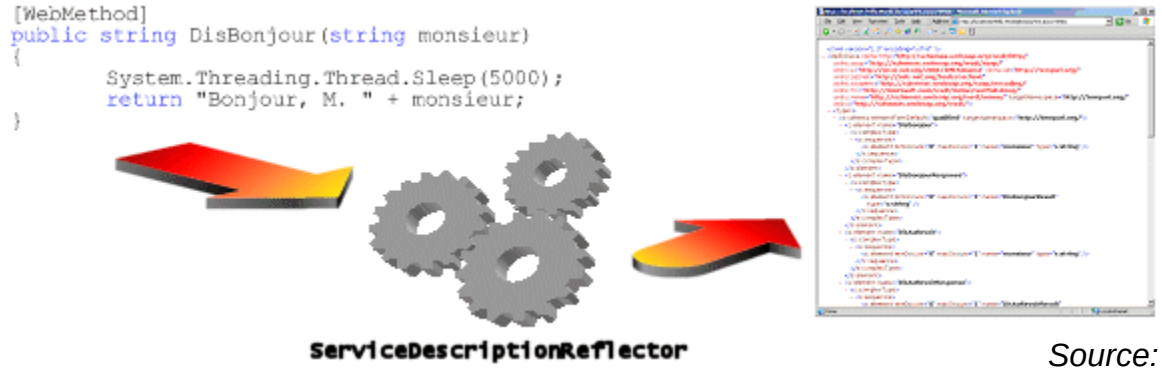
- ✓ Pour une mise en pratique des Web Services
  - TD 3: ASP .NET
    - http://dotnet.developpez.com/cours/?page=csharp#servicewebcs
  - TD 4: Java AXIS
    - http://javaweb.developpez.com/cours/?page=architectures-web#webservices
  - TD 5: gSOAP C/C++
    - http://www.cs.fsu.edu/~engelen/soap.html
- ✓ Permet de simplifier la mise en œuvre
  - Génère automatiquement le WSDL
  - Crée le binding avec SOAP
  - ... mais comment étendre le comportement par défaut ??

Génération du WSDL par le Framework WS de ASP.NET

- ✓ **Ajout d'un attribut** `[WebMethod]`

```
[WebMethod]
public string DisBonjour(string monsieur) {
    System.Threading.Thread.Sleep(5000);
    return "Bonjour, M. " + monsieur + ". Nous sommes " +
  DateTime.Now.DayOfWeek;
}
```

- ✓ **Accès dynamique à la description du Web Service**
  - http://localhost/BonjourWS. asmx?WSDL
  - Introspecte la classe BonjourWS grâce à ServiceDescriptionReflector



Source: Sébastien Bouchet

Attributs de l'Attribut [WebMethod]

- ✓ [WebMethod *attribut=valeur*]
  - (EnableSession=false)
    - Aucune donnée persistante entre le consommateur et le service
  - (BufferResponse=true)
    - Créé une mémoire tampon stockant la réponse
  - (CacheDuration=*nombre de secondes*)
    - Garder les réponse en cache pendant une certaine durée
  - (Description="*une description*")
    - Donne une description à la méthode
  - (MessageName="*alias nom de méthode*")
    - Surcharge de méthode du service
  - (TransactionOption=TransactionOption.[Disabled|Required|Supported|NotSupported|RequiresNew])
    - Gère les transactions

Exemple d'Extension du « Contrat WDSL par défaut »

- ✓ Exemple:
  - Mécanisme de mise en cache
  - Cacher les réponses du Web Service (côté consommateur)
    - Eviter au client de refaire une requête si déjà faire dans un lapse de temps donnée
- ✓ But:
  - Exemple pédagogique plus que réel
  - Illustrer la technique d'extension (d'autres approches sont possibles pour réaliser le cache)

Introspection pour Générer le WSDL

- ✓ **Lors de l'exécution du** `ServiceDescriptionReflector`
  - **Pour chaque attribut** `[WebMethod]` **rencontrée**
  - **Appel** `ReflectMethod` **de tous les** `SoapExtensionReflector`

```
public class MyReflector : SoapExtensionReflector
{
  public override void ReflectMethod()
  {
    ProtocolReflector refl = this.ReflectionContext;
    //Faire quelque chose avec refl, comme insérer
    //des extensions
  }
}
```

- ✓ **Modifier le WSDL généré:**
  - **comment représenter une méta-donnée (et accessoirement quel est son contenu)**
  - **où l'insérer dans le contrat**
  - **comment définir la nouvelle propriété de la méthode web**

Nouvel Attribut à Créer

```
public enum CachingMode
{
    Rotating,Sliding
}

public enum CachingPeriod
{
    ToNextHour,ToNextDay,ToNextWeek,ToNext
    Month,ToNextYear
}

[AttributeUsage(AttributeTargets.Method)]
public class
    ClientSideCacheabilityAttribute :
    Attribute
{
    private CachingMode mode =
    CachingMode.Rotating;
    private CachingPeriod period =
    CachingPeriod.ToNextDay;
    private long validity;

    public
    ClientSideCacheabilityAttribute(){}

    public
    ClientSideCacheabilityAttribute(Cachin
    gPeriod p)
    {
        mode = CachingMode.Rotating;
        period = p;
    }

    public
    ClientSideCacheabilityAttribute(long
    validityTimeSpanMs)
    {
        mode = CachingMode.Sliding;
        validity = validityTimeSpanMs;
    }

    public CachingMode Mode {.}

    public CachingPeriod Period {.}

    public long Validity {.}
}
```

Ajout du Nouvel Attribut et Impact sur le WSDL

- ✓ **Ajout du Nouvel Attribut:**

```
[WebMethod]
[ClientSideCacheability(CachingPeriod.ToNextDay)]
public string DisBonjour(string monsieur)
{
    System.Threading.Thread.Sleep(5000);
    return "Bonjour, M. " + monsieur + ". Nous sommes " +
DateTime.Now.DayOfWeek;
}
```

- ✓ **Ce nouvel attribut doit se retrouver dans le WSDL**
  - **Où ajouter cette information ???**

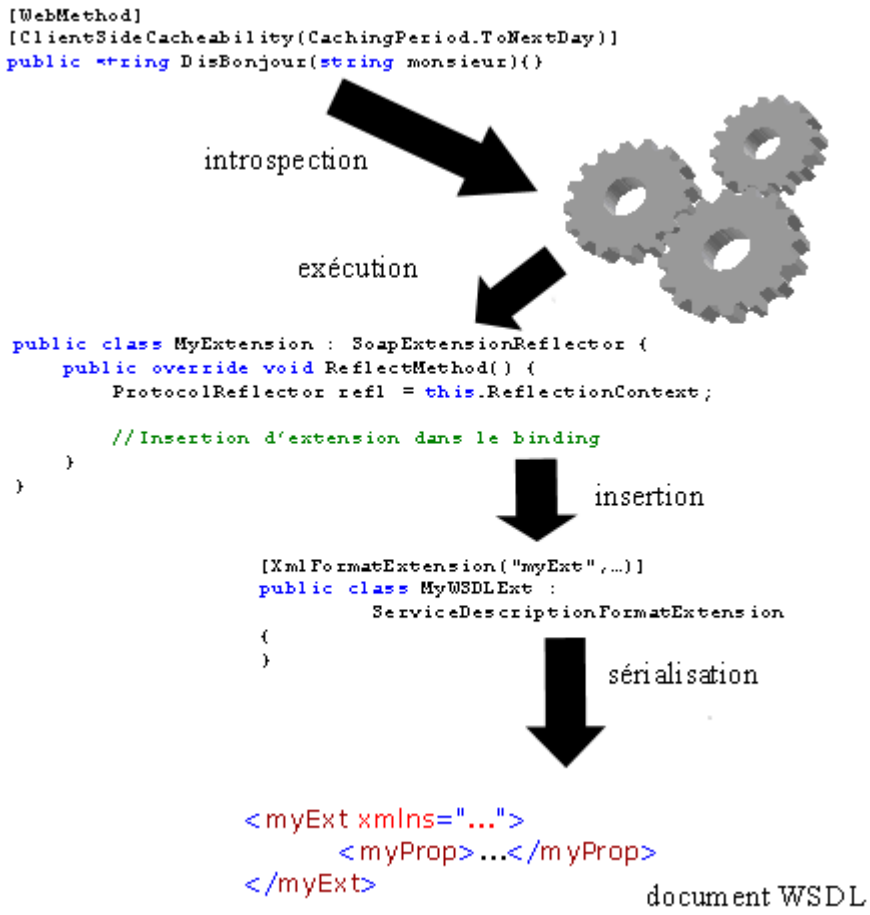
A Ajouter dans le binding WSDL

- La spécification WSDL impose de le faire au niveau du binding

```
<operation name="DisBonjour">
    <soap:operation soapAction="http://tempuri.org/DisBonjour"
                    style="document" />
    <dng:cachePolicy >
        <dng:Mode>Rotating</dng:Mode>
        <dng:Period>ToNextDay</dng:Period>
        <dng:Validity>0</dng:Validity>
    </dng:cachePolicy>
    <input>
        <soap:body use="literal"/>
    </input>
    <output>
        <soap:body use="literal"/>
    </output>
</operation>
```

Comment le Générer ?

- ✓ Créer une classe ServiceDescriptionFormatExtension
  - Pour créer les méta-données
- ✓ La classe CachingExtension
  - Possède les mêmes propriétés que l'attribut ClientSideCacheabilityAttribute




```
[XmlFormatExtension("cachePolicy","urn:dng-articles.org:seb:cachext",typeof(OperationBinding))]
[XmlFormatExtensionPrefix("dng", "urn:dng-articles.org:seb:cachext")]
public class CachingExtension : ServiceDescriptionFormatExtension {... }
```

Reste à Compléter ReflectMethod

- ✓ Ecrire SoapExtensionReflector
  - qui va insérer cette extension au document WSDL en cours de génération pour chaque méthode cacheable

```
public override void ReflectMethod(){
    ProtocolReflector refl = this.ReflectionContext;
    ClientSideCacheabilityAttribute policy =
        refl.Method.GetCustomAttribute(typeof(ClientSideCacheabil
ityAttribute))
            as ClientSideCacheabilityAttribute;
    if(policy != null){
         refl.OperationBinding.Extensions.Add(new
CacheabilityExtension(policy));
    }
}
```

- ✓ Que reste-t-il à faire pour que cela fonctionne ?

Modifier le Comportement du Générateur de Proxy

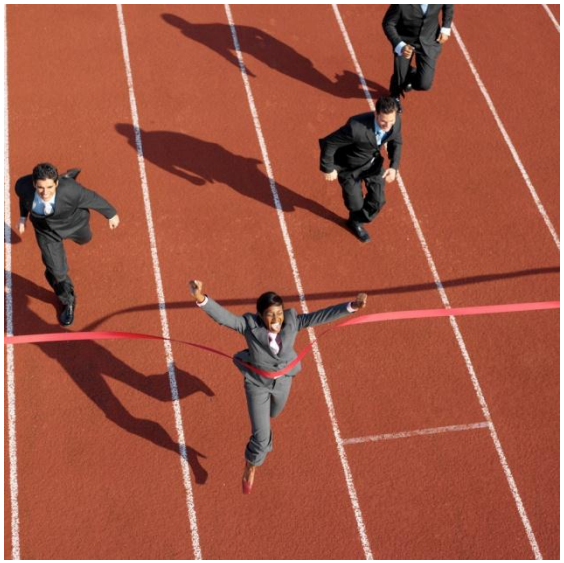
- ✓ Consommation d'un Service Web
  - Génération d'un proxy pour le service
    - wsdl http://localhost/Service?WSDL /out:lenomdelaclasseproxy.cs

```
C:\>wsdl http://127.0.0.1/PetitExemple/Exemple.asmx?wsdl /out:petitExemple.cs
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 1.1.4322.573]
Copyright (C) Microsoft Corporation 1998-2002. All rights reserved.

Writing file 'petitExemple.cs'.

C:\>
```

- ✓ Voir la suite de l'exemple:
  - http://www.dotnetguru.org/articles/dossiers/extensionsWSDL/ExtensionsWSDL.htm



Conclusion

Les Services Web

Les Services Web

- ✓ La 3ème génération du Web
- ✓ Technologies Standards du Web
  - SOAP (1.1 puis SOAP 1.2)
  - WSDL (1.1, 1.2 puis 2.0)
- ✓ Technologies non standardisées
  - UDDI, DISCO
  - GXA (Global XML Architecture)
  - WSDD, WSFL, ASMX, ...

Cinématique générale

Service

```
public class MyWebService {
        public String sendMessage(String name) {
                return "  Hello " + name
        }
}
```

(Java, C++, …)

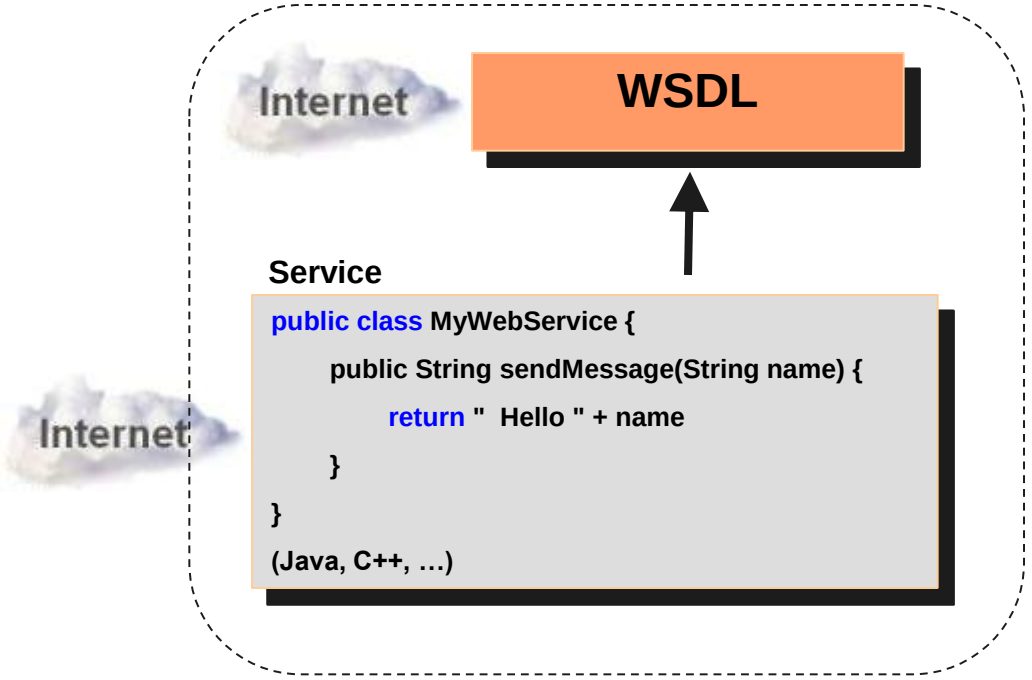
Cinématique générale

Service

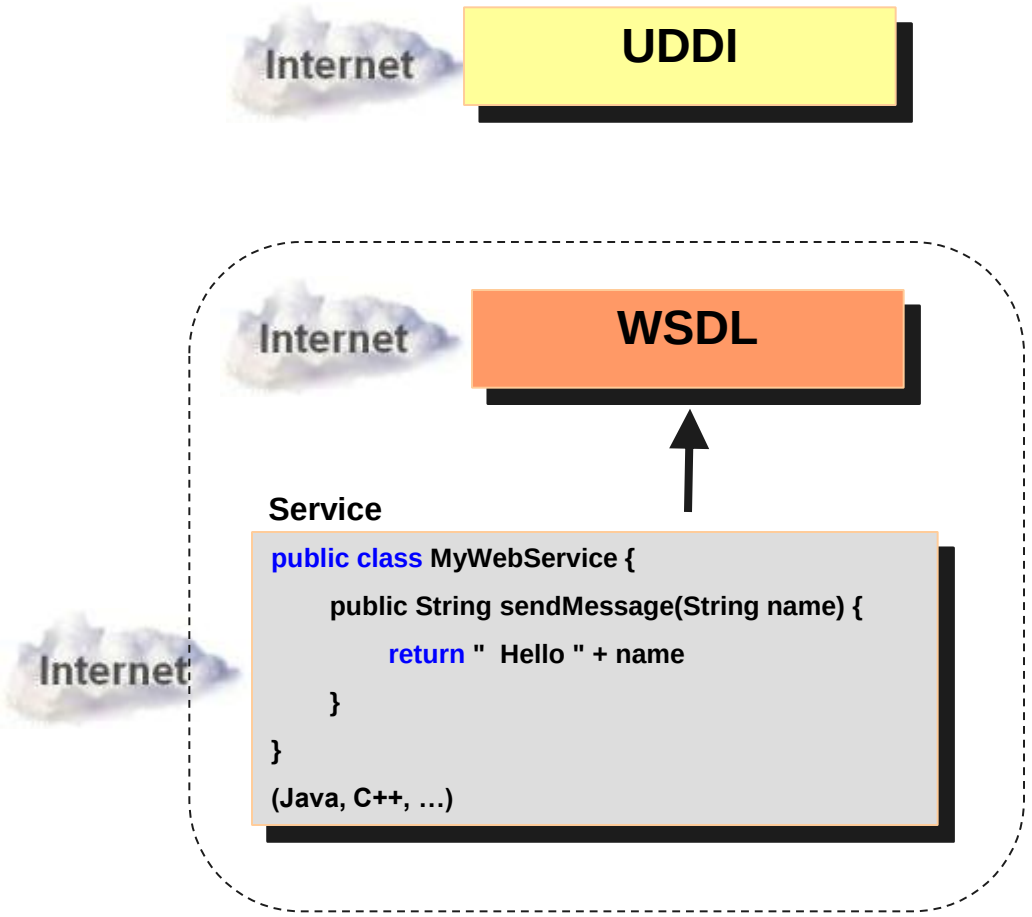
```
public class MyWebService {
        public String sendMessage(String name) {
                return "  Hello " + name
        }
}
```

(Java, C++, …)

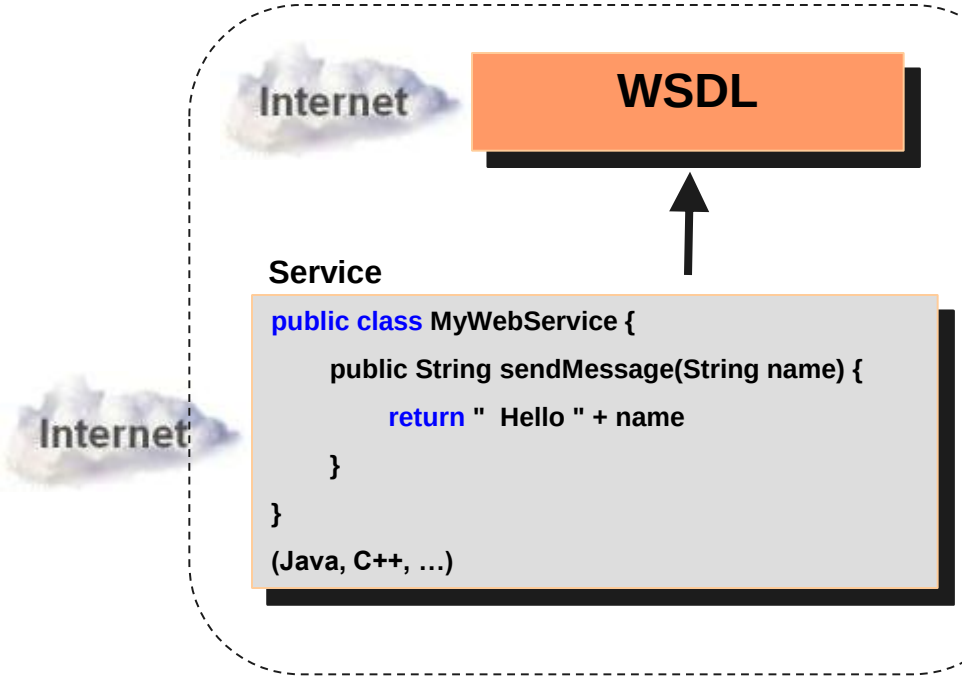
Cinématique générale



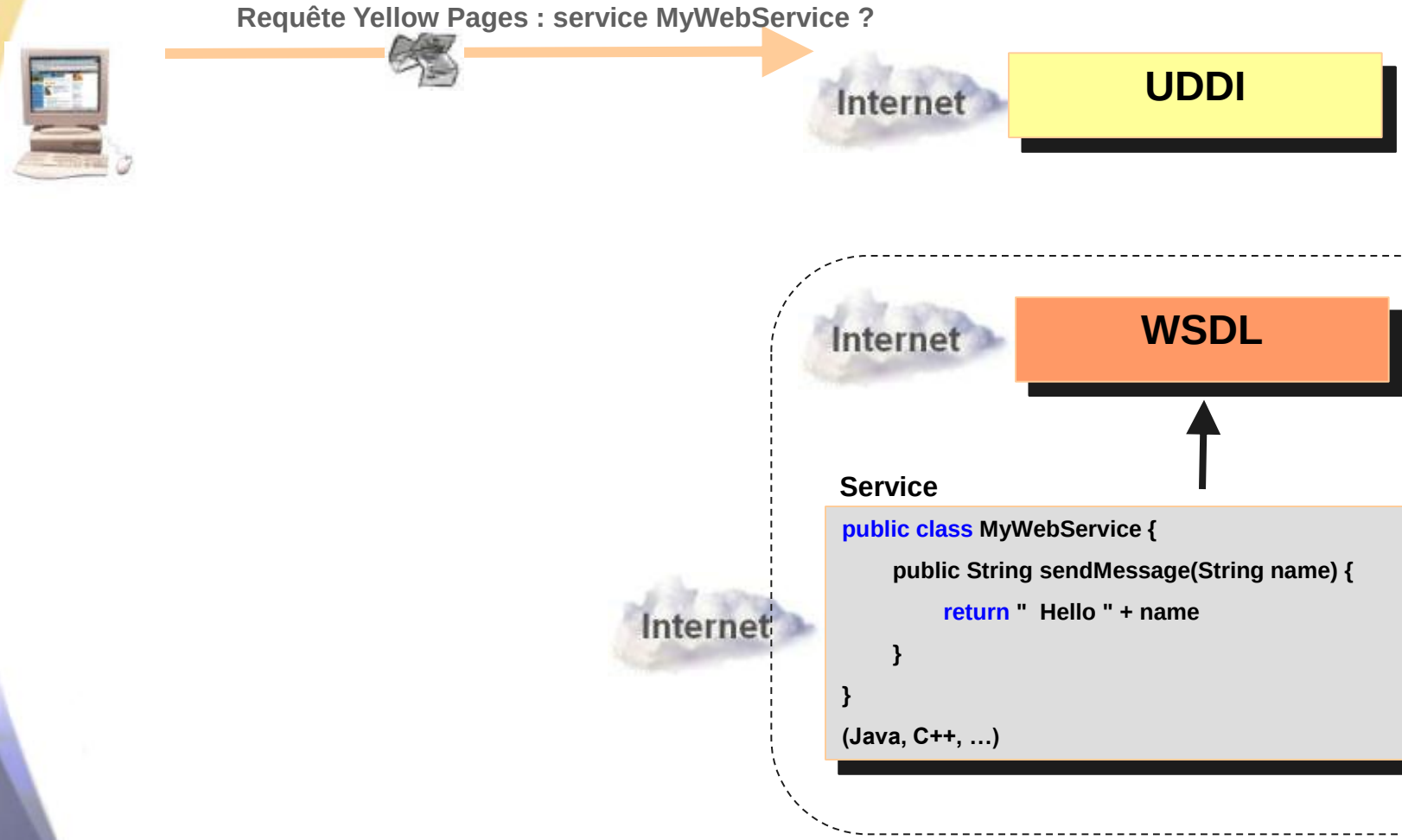
Cinématique générale



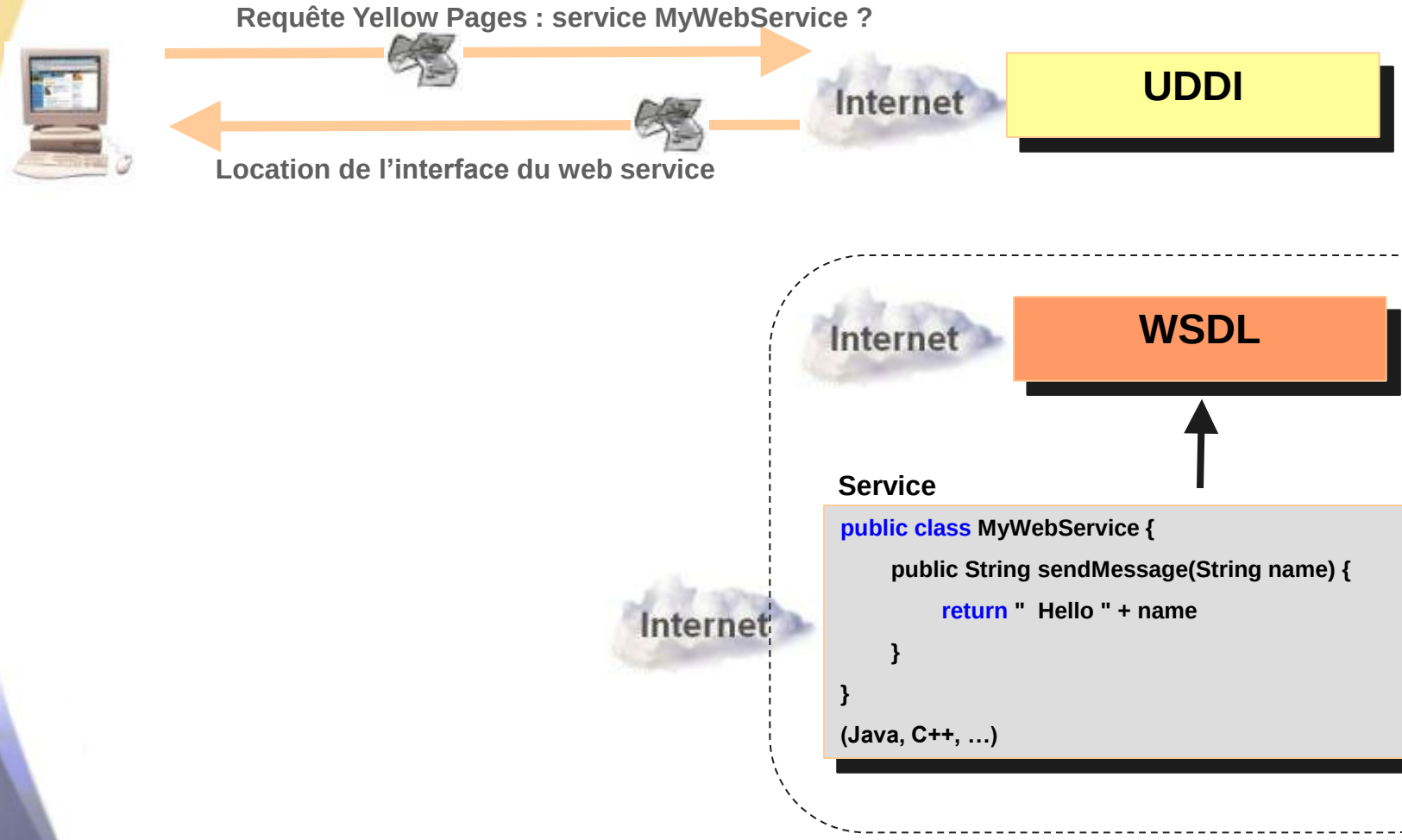
Cinématique générale



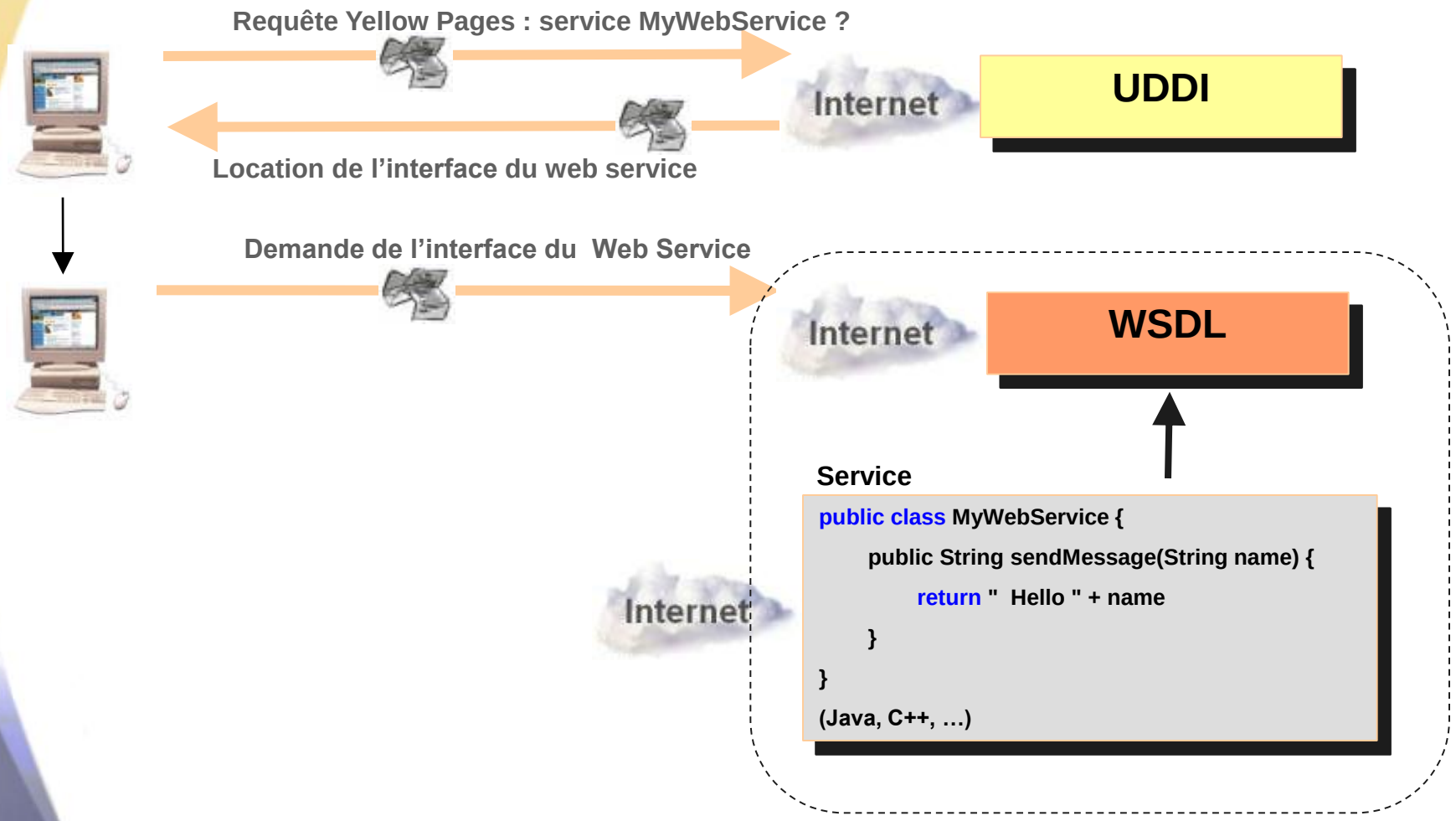
Cinématique générale



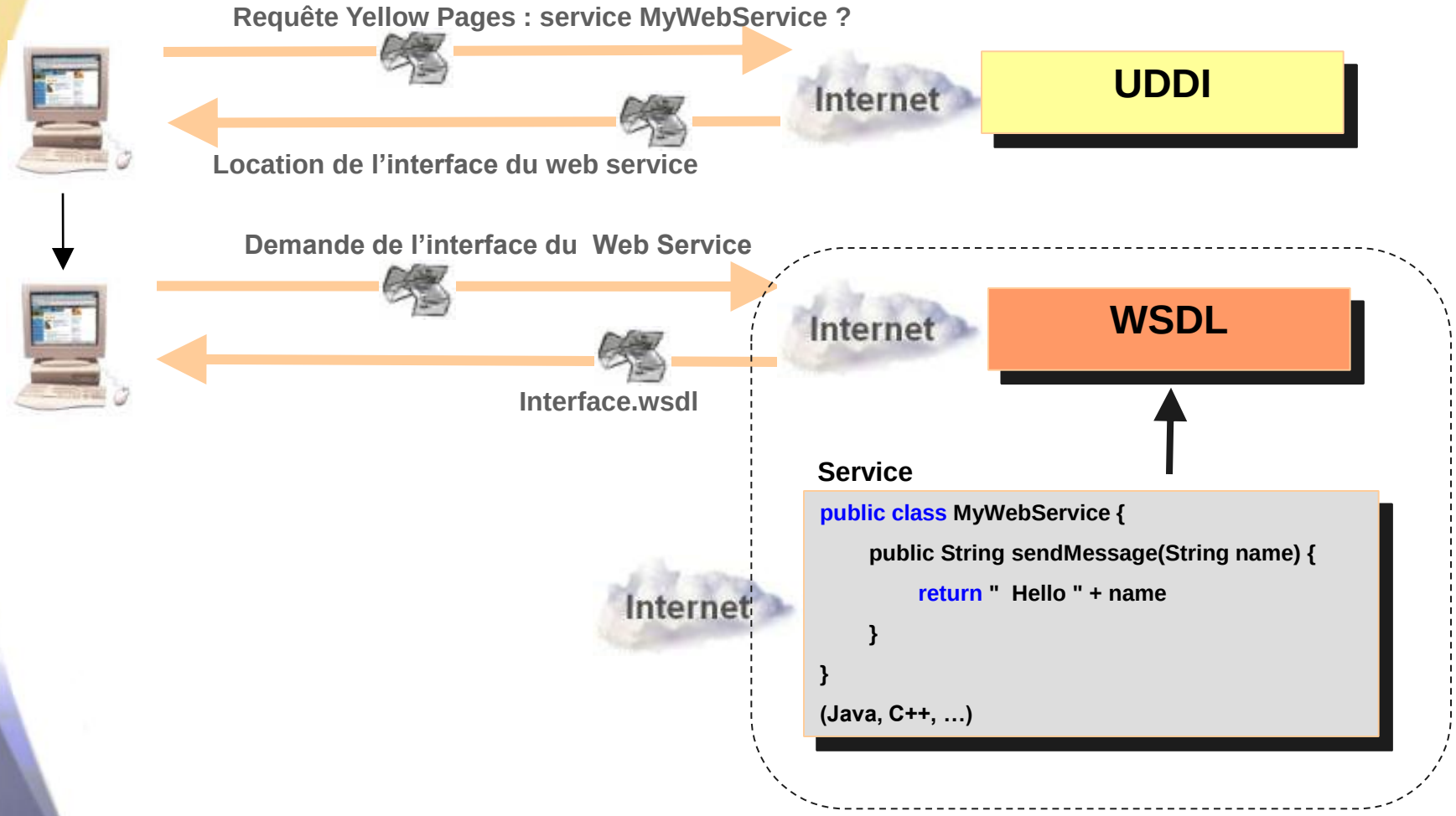
Cinématique générale



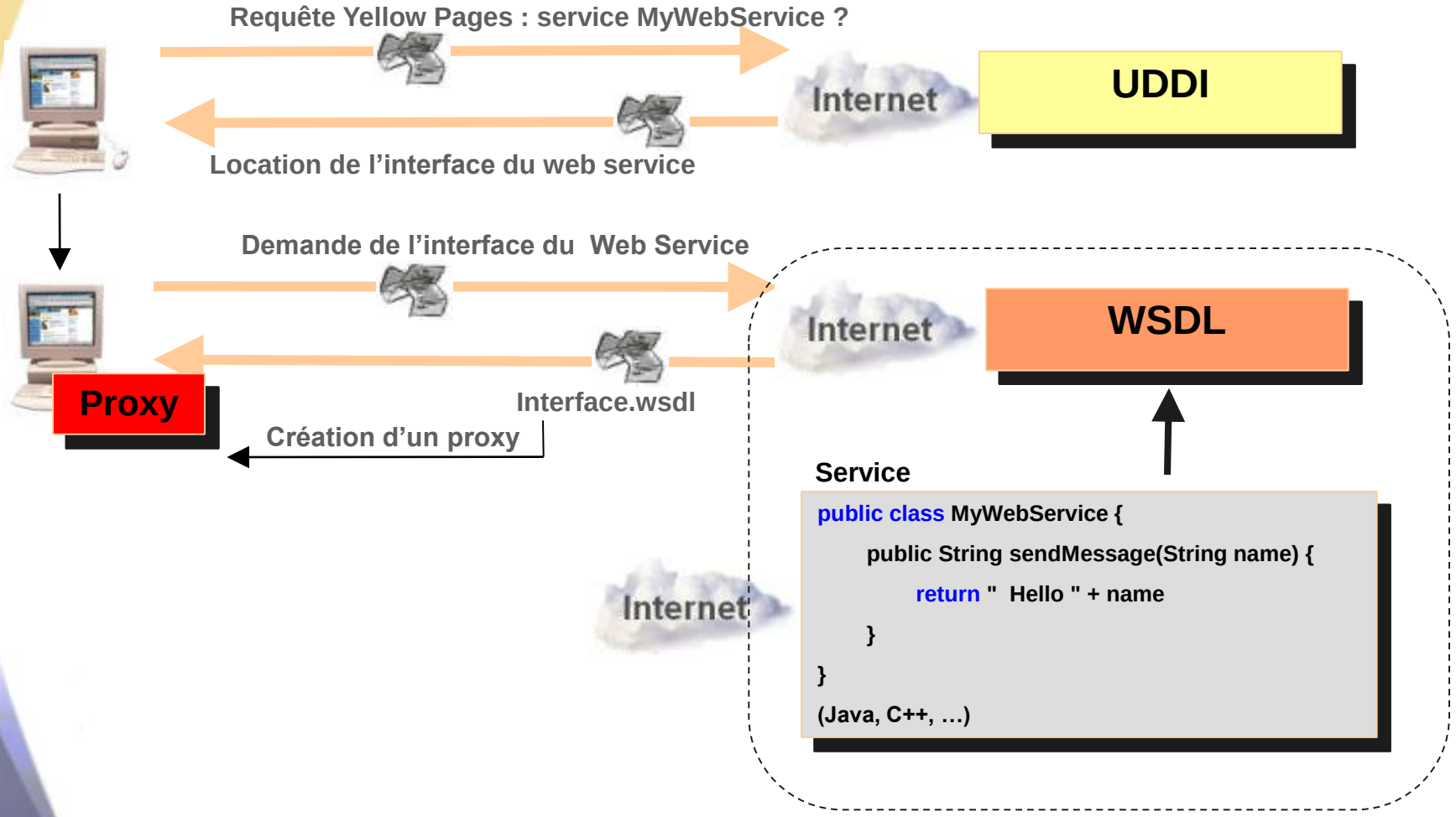
Cinématique générale



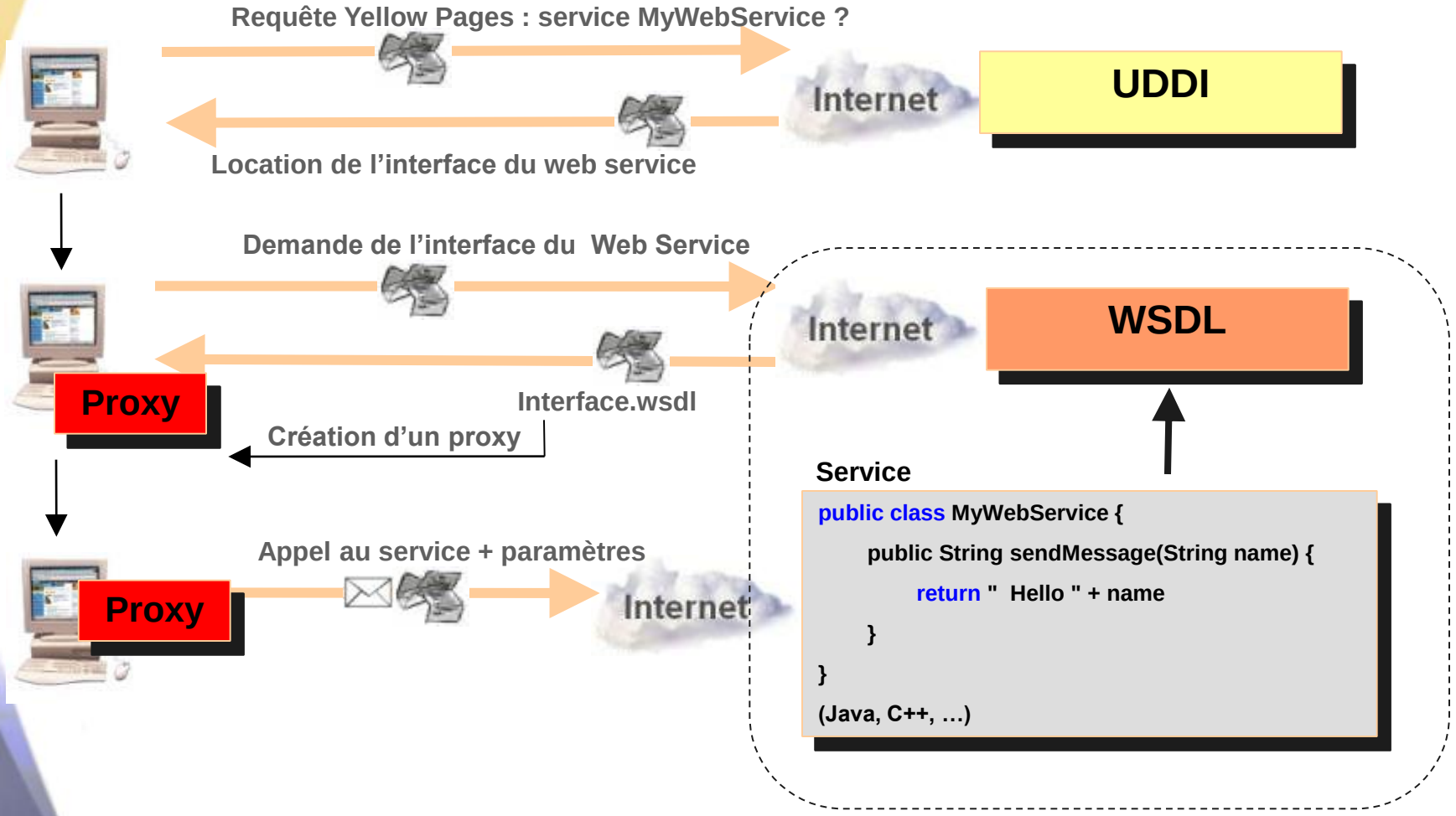
Cinématique générale



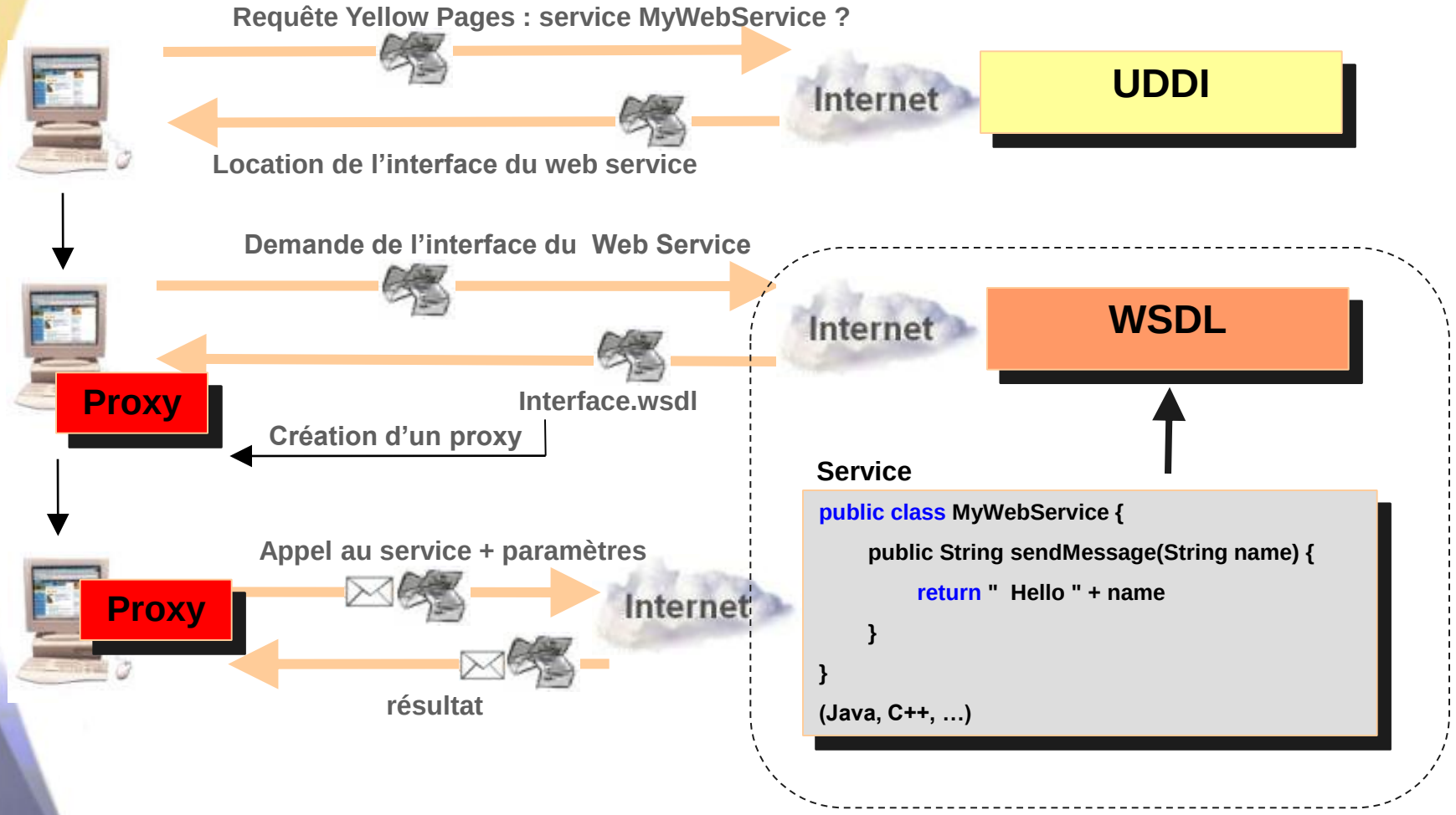
Cinématique générale



Cinématique générale



Cinématique générale



Université Nice SOPHIA ANTIPOLIS

UDDI : Universal Description, Discovery and Integration

Les Services Web

UDDI

- ✓ Spécification (09/2000)
  - Ariba, IBM, Microsoft +260 autres sociétés
- ✓ Objectifs
  - annuaire mondial d'entreprises pour permettre d'automatiser les communications entre prestataires, clients, etc.
  - plusieurs entrées indexées : nom, carte d'identité des sociétés, description des produits, services applicatifs invocables à distance (références des connexions)
    - Indexation des catalogues propriétaires (ebXML, RosettaNet, Ariba, Commerce One, etc.)
- ✓ Grammaire XML (schéma XML)
  - Soumission/interrogation basées sur SOAP et WSDL

White, yellow and green pages

- « White pages »
  - adresse, contact, et identifiants connus
- « Yellow pages »
  - catégorie industrielle
    - Industrie: NAICS (Industry codes - US Govt.)
    - Produit/Services: UN/SPSC (ECMA)
    - Location: taxinomie géographique
- « Green pages »
  - informations techniques sur les services

UDDI : fonctionnement : tModel

- ✓ tModel = Technology Model
- ✓ Structure générique de méta-data pour représenter de manière unique les concepts ou les outils de construction
- ✓ Inclus aussi les définitions des protocoles
- ✓ Puissant système de modélisation abstraite

UDDI : le modèle de données

Provider: Information sur l'entité proposant le service

0…n

Service: Informations descriptives sur la famille de produits offerts

0…n

Binding: Informations techniques sur le point d'entrée du service et sur les specs de construction du service

tModel: Descriptions des spécifications des services

1…n

Les Bindings contiennent les références aux tModels. Ces références désignent les spécifications de l'interface pour un service.

Providers, Services et Bindings

- ✓ Providers
  - Exemples: Accounting Department, Corporate Application Server
  - Nom, Description, autres informations pour les contacts
  - Informations sur les identifications et caractéristiques
- ✓ Services
  - Exemples: Purchase Order services, Payroll services
  - Nom, Description(s)
  - Catégorie, caractéristiques
- ✓ Bindings
  - Description(s), points d'accès, paramètres
  - Exemples: point d'accès (http://...) pour le Web Service

UDDI : les caractéristiques importantes

- ✓ Neutre en terme de protocole – comme tout registre, il peut y avoir des pointeurs sur n'importe quoi
- ✓ Possibilité de faire des recherches par domaine d'activité, service, Web Service, binding
- ✓ Nœuds privés et publics autorisés