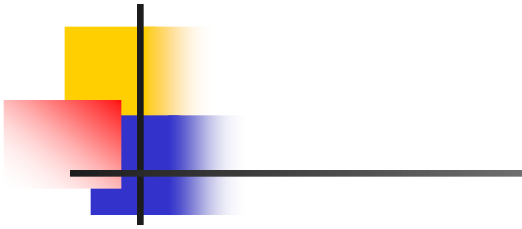
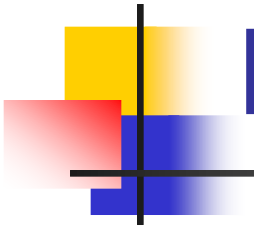


Serveur Web et Services Web

Jean-Yves Tigli BAT4, Applications Réparties

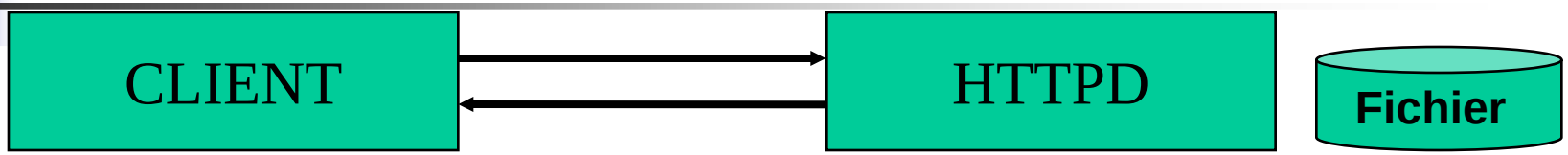
D'après le cours de Eric Garcia, Julien Henriet, LIFL



Les services Web

- Serveur Web
- Généralités
- Architecture
- SOAP
- WSDL
- UDDI
- Conclusion

Serveur Web : Récupération d'un Document Méthode GET



- GET /fichier

Port 80

Le Client envoie

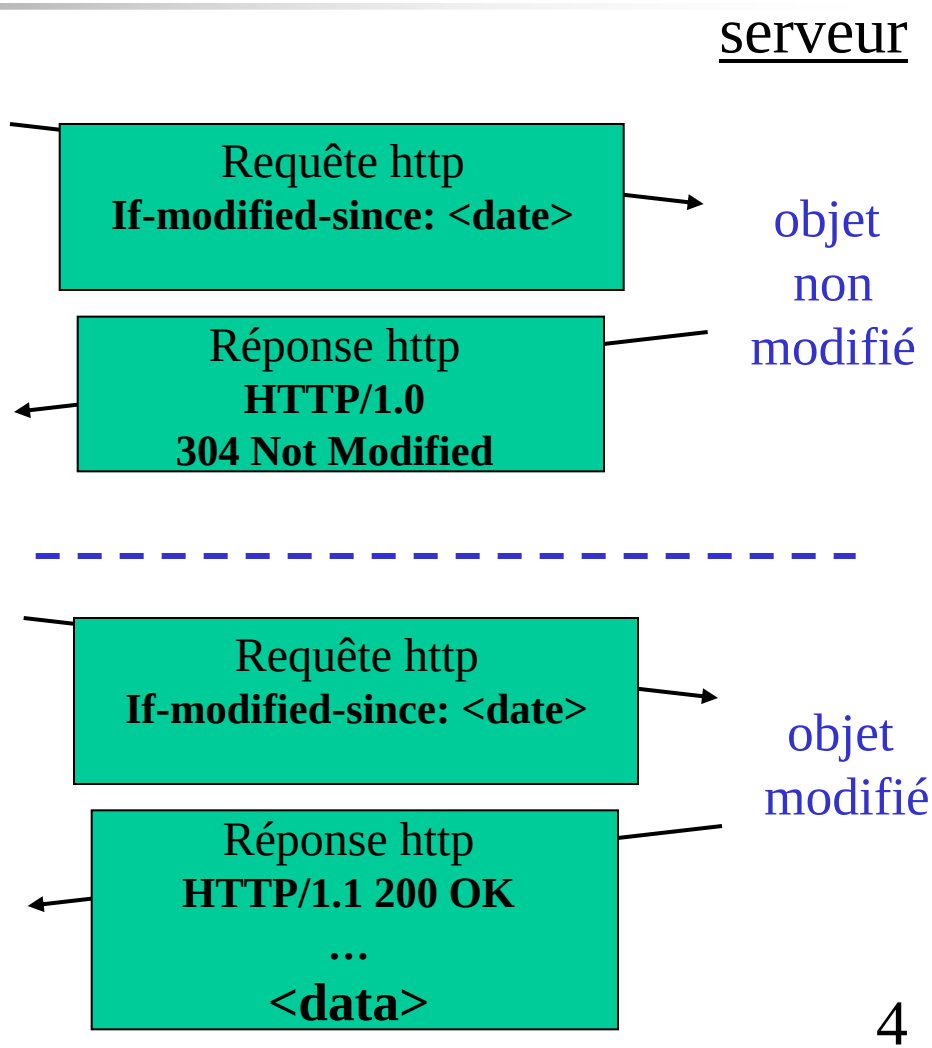
GET /docu2.html HTTP/1.0 *méthode,chemin,v*
Accept: www/source
documents acceptés
Accept: text/html
Accept: image/gif
User-Agent: Lynx/2.2 libwww/2.14
From: alice@pays.merveilles.net
* une ligne blanche *

le Serveur répond

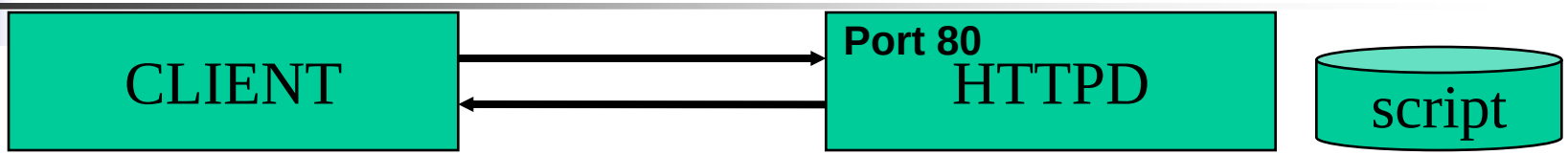
HTTP/1.0 200 OK *ligne de status*
Date: Wed, 02Feb97 23:04:12 GMT
Server: NCSA/1.1
MIME-version: 1.0
Last-modified: Mon,15Nov96 23:33:16 GMT
Content-type: text/html *type du document retourné*
Content-length: 2345 *sa taille*
* une ligne blanche *
<HTML><HEAD><TITLE> ...

Serveur Web : Récupération, Méthode GET conditionnelle

- Objectif : ne pas envoyer d'objet si le client à une version chargée à jour (en cache).
- client: spécifie la date de la copie en cache dans la requête :
- If-modified-since: <date>
- serveur: la réponse ne contient pas de données si l'objet est à jour :
- HTTP/1.0 304 Not Modified



Serveur Web : Soumission d'un Formulaire Méthode GET



- GET/script?name1=value1&name2=value2

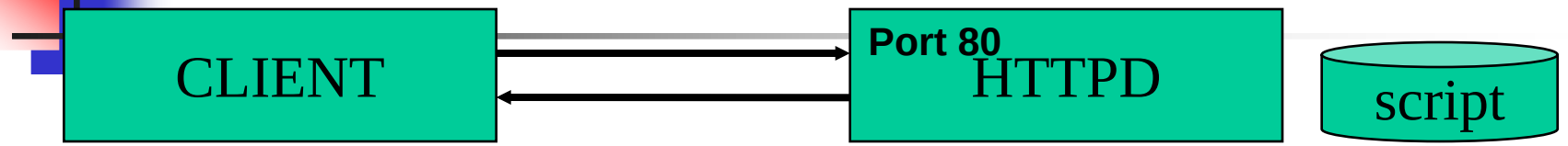
le Client envoie

GET /script?name1=value1 HTTP/1.0
Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent: Lynx/2.2 libwww/2.14
From: alice@pays.merveilles.net
* une ligne blanche *

le Serveur répond

HTTP/1.0 200 OK
Date: Wed, 02Feb97 23:04:12 GMT
Server: NCSA/1.1
MIME-version: 1.0
Last-modified: Mon,15Nov96 23:33:16 GMT
Content-type: text/html
Content-length: 2345
* une ligne blanche *
<HTML><HEAD><TITLE> ...

Serveur Web : Soumission d'un Formulaire méthode POST



- POST /script

le Client envoie

POST /script HTTP/1.0
Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent: Lynx/2.2 libwww/2.14
From: alice@pays.merveilles.net
* une ligne blanche *
name1=value1&
name2=value2

le Serveur répond

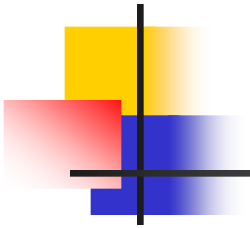
HTTP/1.0 200 OK
…
Content-length: 2345
* une ligne blanche *
<HTML><HEAD><TITLE> ...

6

Serveur Web : Comportement du Client / type du document retourné

- A partir du type MIME de Content-Type
  - Visualisation native
    - la fonction de visualisation est dans le noyau (core) du client text/html, image/jpeg
  - Visualisation par plugin
    - la fonction est présente dans un DLL, SO, ou un JAR
    - elle est liée dynamiquement pour réaliser la visualisation world/vrml, text/tex
  - Visualisation externe
    - la fonction n'est pas présente dans le client
    - le client rapporte le document et le sauvegarde dans un fichier temporaire video/mpeg, application/postscript.

7

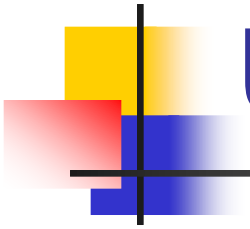


Les services Web

Généralités

Un Service Web, c'est quoi ?

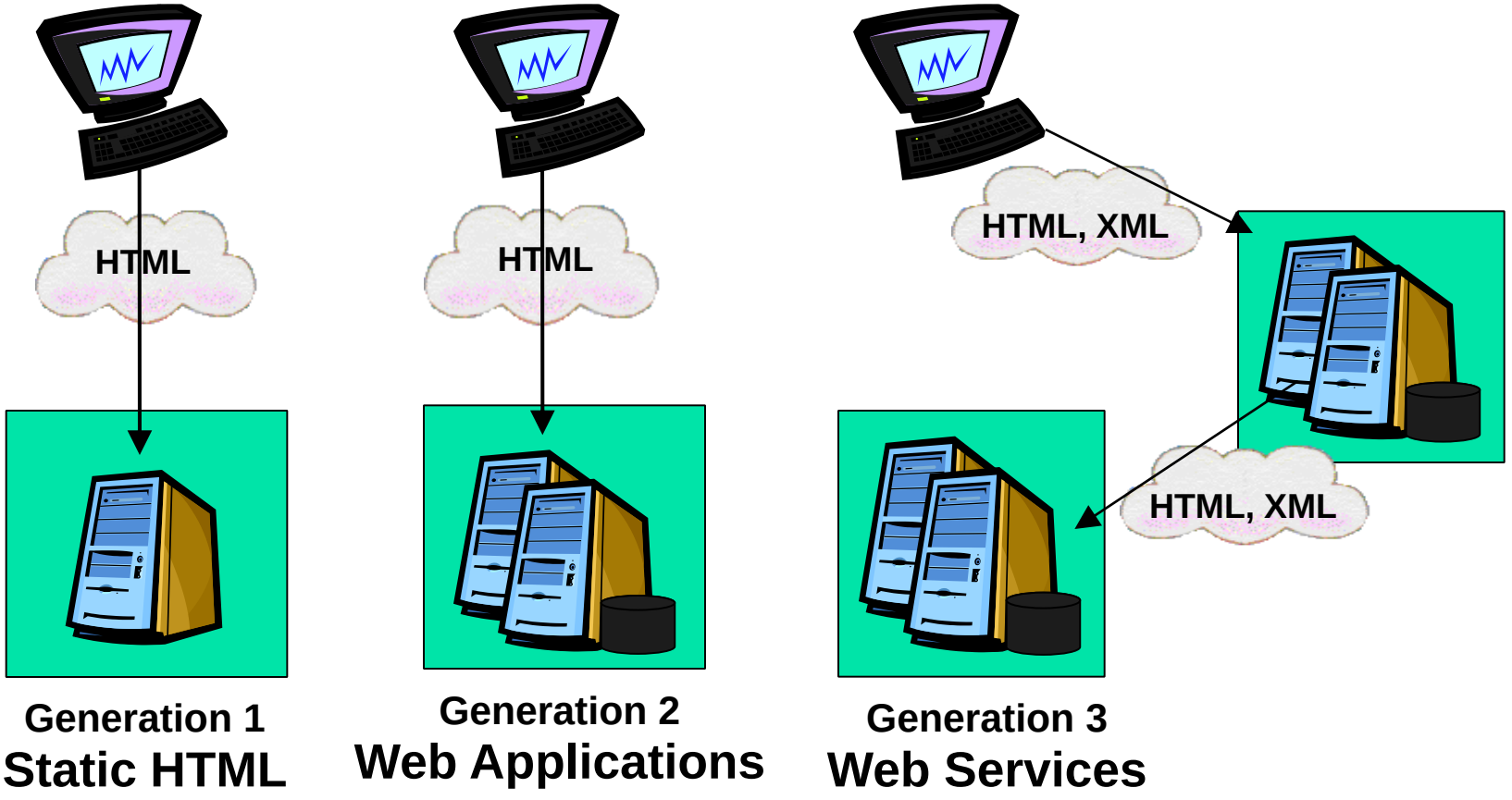
- Un service Web est une « unité logique applicative » accessible en utilisant les protocoles standard d'Internet
- Une «librairie» fournissant des données et des services à d'autres applications.
- Un objet métier qui peut être déployé et combiné sur Internet avec une faible dépendance vis-à-vis des technologies et des protocoles.
- Combine les meilleurs aspects du développement à base de composants et du Web.

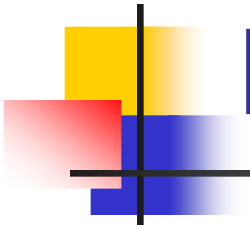


Un Service Web, c'est quoi ?

- Caractéristiques:
  - Réutilisable
  - Indépendamment de
    - la plate-forme (UNIX, Windows, …)
    - l'implémentation (VB, C#, Java, …)
    - la plate-forme de développement sous-jacente (.NET, J2EE, Axis…)

Evolution du Web

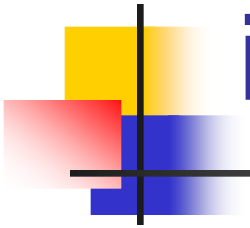




Pour quoi faire ?

- Les services Web permettent d'interconnecter :
  - Différentes entreprises
  - Différents matériels
  - Différentes applications
  - Différents clients

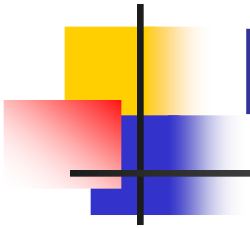
Un Service Web et interropérabilité …



- Plateform independent : A l'instar de Corba, les Services Web gèrent l'interropérabilité au niveau du protocole d'échange
- Plateform dependent : D'autres approches gèrent l'interropérabilité par le portage de la plate-forme d'exécution (ex. OSGi, RMI sur Java et historiquement COM/DCOM)

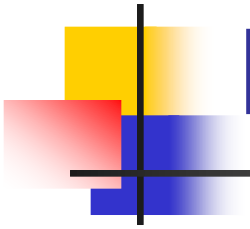
Quels objectifs ?

- Remplacer les outils actuels (RPC, DCOM, CORBA, RMI) par une approche entièrement ouverte et interopérable, basée sur la généralisation des serveurs Web avec scripts CGI (Common Gateway Interface).
- Faire interagir des composants hétérogènes, distants, et indépendants avec un protocole standard (SOAP).
- Dédiés aux applications B2B (Business to Business), EAI (Enterprise Application Integration), P2P (Peer to Peer).
- Passer les politiques de sécurité grâce en grande partie à une couche session basée sur HTTP (port 80).



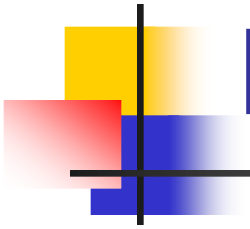
Et plus concrètement ?

- Une nouvelle technologie des objets distribués ?
  - Invocation distante des services Web : **SOAP**
  - Description des services Web : **WSDL**
  - Enregistrement et découverte de services Web : **UDDI**



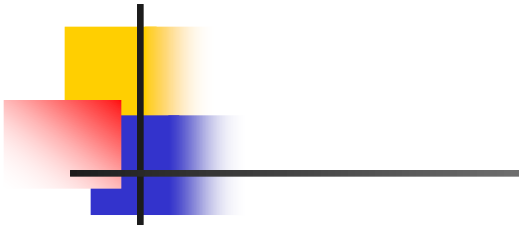
Et plus concrètement ?

- Basés sur des standards
  - Standards du W3C : XML, SOAP, WSDL
  - Web Service Discovery Tool
    - Standards industriels : UDDI, ebXML
    - Propriétaires : DISCO, WSDD, WSFL, ASMX, …



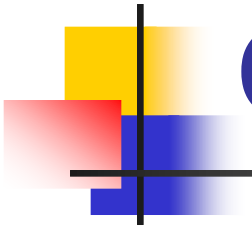
Et plus concrètement ?

- Implémentations actuelles :
  - Microsoft .Net
  - Sun JavaONE : J2EE + Web services (WSDP = JAXP,JAX-RPC,JAXM…)
  - Apache SOAP / Axis, IBM WSTK
  - Oracle, Bea, Iona, Enhydra …



Les services Web

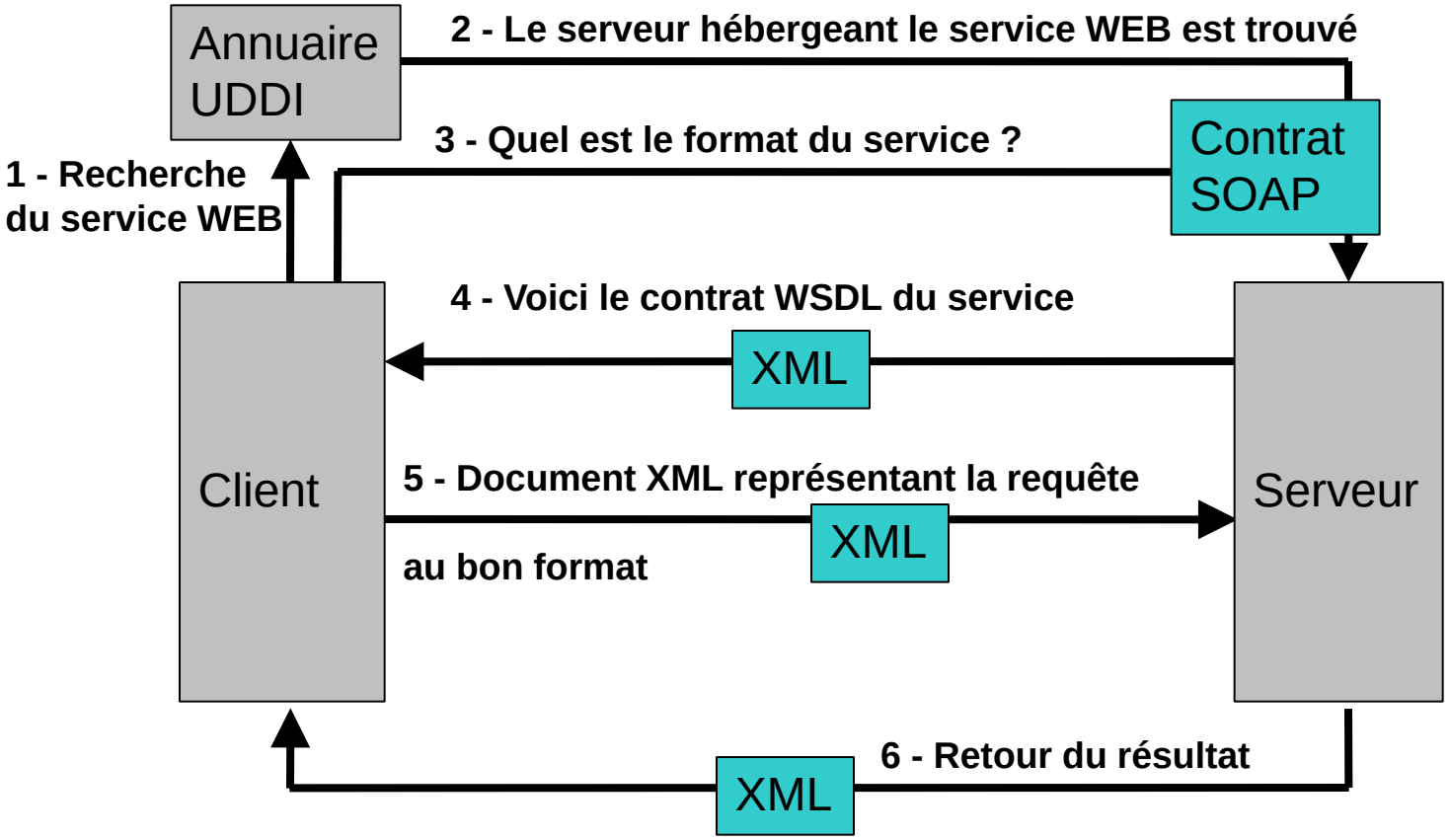
Architecture

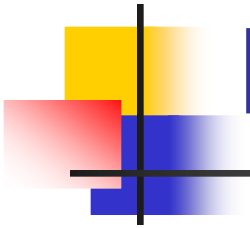


Cycle de vie complet

- 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

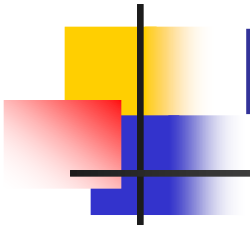
Cycle de vie complet





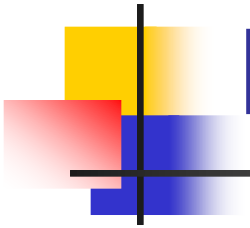
PART II : SOAP

- Simple Object Access Protocol
- Page sur SOAP du W3C : http://www.w3.org/2002/07/soap-translation/soap12-part0.html
- XML-RPC, l'ancêtre de SOAP



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

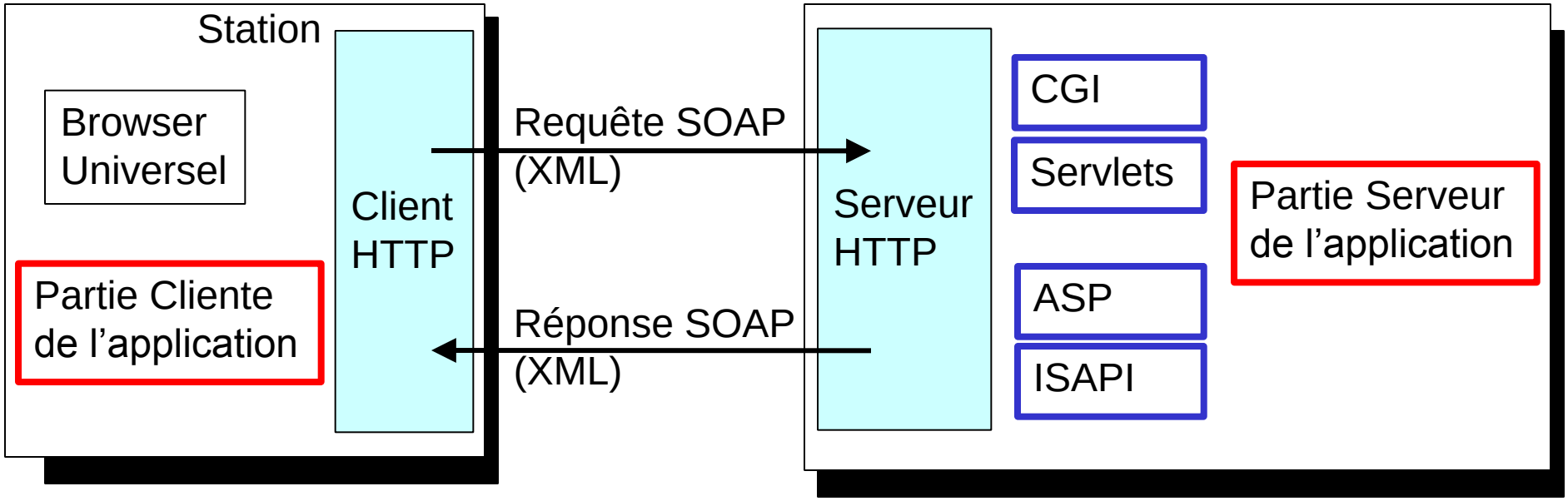


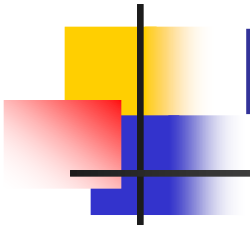
Les 3 aspects d'un appel SOAP

- 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





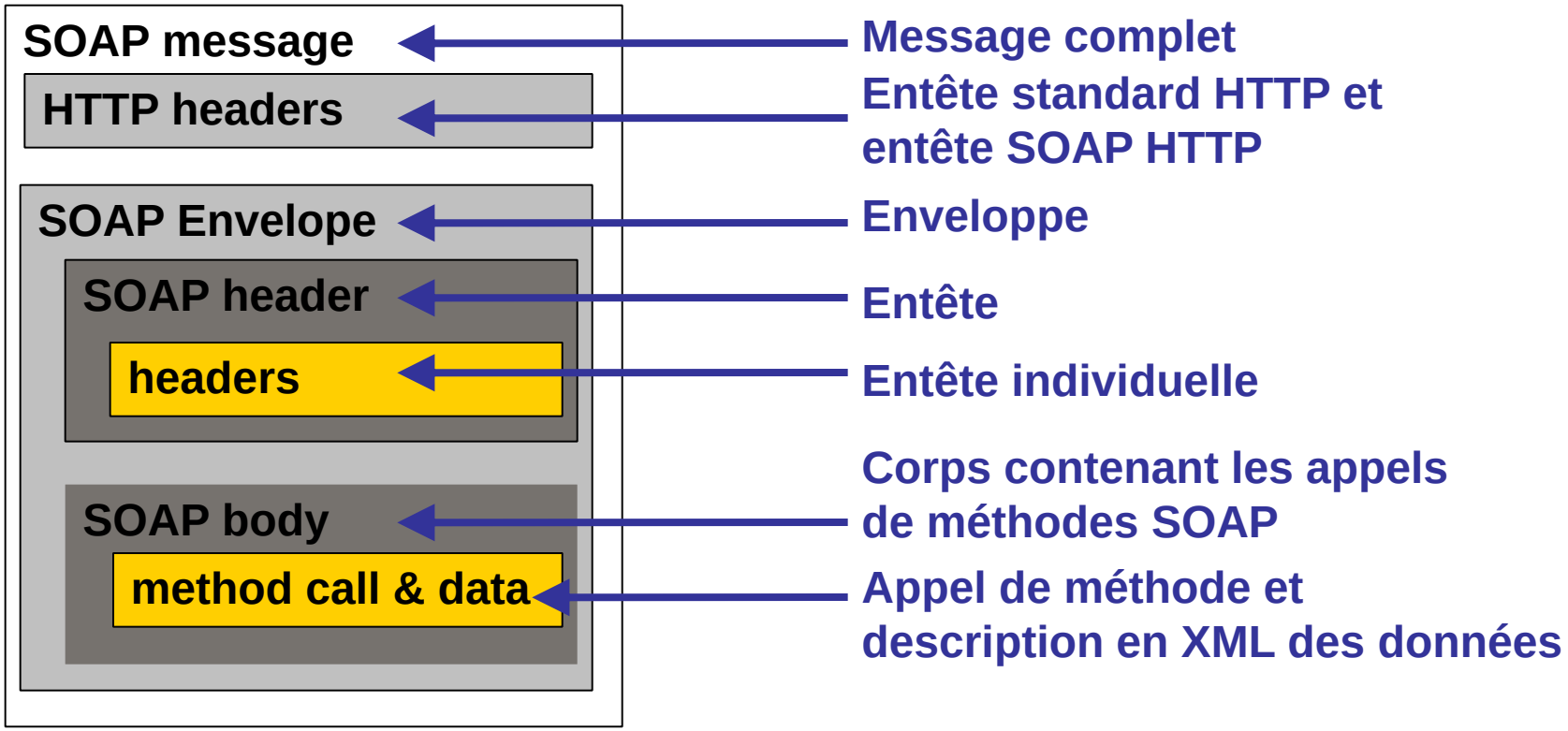
Pourquoi utiliser HTTP ?

- HTTP est devenu de facto le protocole de communication d'Internet
- HTTP est disponible sur toutes les plate-formes
- HTTP est un protocole simple, qui ne requiert que peu de support pour fonctionner correctement
- HTTP 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 : plate-formes, 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



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

```
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>
```

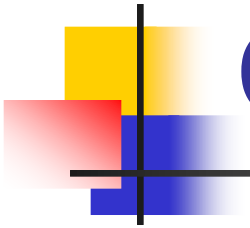
Modèle de message

- SOAP permet une communication par message
  - d'un expéditeur vers un récepteur
- Structure d'un message
  - Enveloppe / envelope
  - Entête / header
    - Elément optionnel
    - Contient des entrées non applicatives (transactions, session, …)
  - Corps / body
    - 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

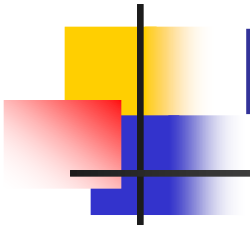
- Contient des entrées non applicatives
  - Transactions, sessions, …
- L'attribut mustUnderstand
  - Si absent ou =0 l'élément est optionnel pour l'application réceptrice
  - si =1, l'élément doit être compris par l'application réceptrice sinon le traitement du message par le récepteur doit échouer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

```
<element name= '' price'' type= '' float'' />
<element name= '' greeting'' type= '' xsd:string'' />
<price>15.57</price>
<greeting id= '' id1'' >Hello</greeting>
```

- **Structures**

```
<element name= '' book'' ><complexType>
<element name= '' autor'' type= '' xsd:string'' />
<element name= '' title'' type= '' xsd:string'' />
</complexType></element>
<e:book>
<autor>J.R.R. Tolkien</autor>
<title>A hobbit story</title>
</e:book>
```

Règles d'encodage

- **Enumération**

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

Règles d'encodage

- **Références**

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

Règles d'encodage

- **Tableaux**

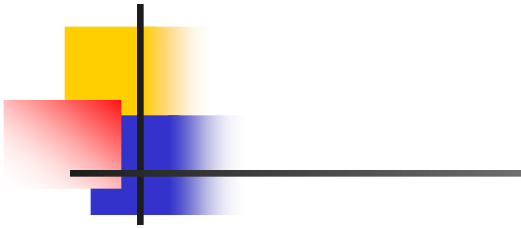
```
<SOAP-ENC:Array id= '' id3' '  SOAP-ENC:arrayType=xsd:string[2,2]₩>
   <item>r1c1</item>
   <item>r1c2</item>
   <item>r2c1</item>
   <item>r2c2</item>
```

- **Tableaux d 'octets**

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

- **Tableaux creux**

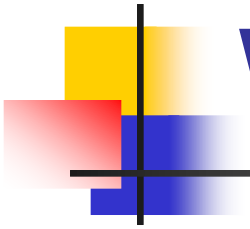
```
<SOAP-ENC:Array id= '' array-1' '  SOAP-ENC:arrayType=xsd:string[10,10]>
   <item SOAP-ENC:position= '' [2,2]' ' >Third row, third col</item>
   <item SOAP-ENC:position= '' [7,2]' ' >Eigth row, third col</item>
</SOAP-ENC:Array>
```



PART II : Les services Web

WSDL : Web Services Description Language

Ou le premier niveau de Contrat



WSDL

- Spécification (09/2000)
  - Ariba, IBM, Microsoft
  - TR W3C v1.1 (25/03/2001)
- Objectif
  - Décrire les services comme un ensemble d'opérations et de messages abstraits relié (*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

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.
- Plusieurs types d'opérations
  - **One-way**
    - Le point d'entrée reçoit un message (<input>).
  - **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 recoit 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

Élément <porttype>

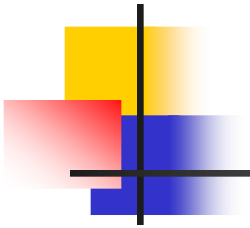
- Exemple

```
<!-- port type declns -->
<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>

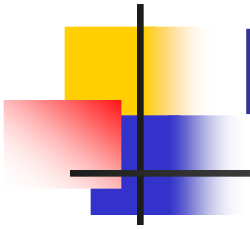
- Spécifie une liaison d'un <porttype> à un protocole concret (SOAP1.1, HTTP1.1, MIME, …).

```
<!-- 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>
```



Les services Web

Conclusion



Les services Web

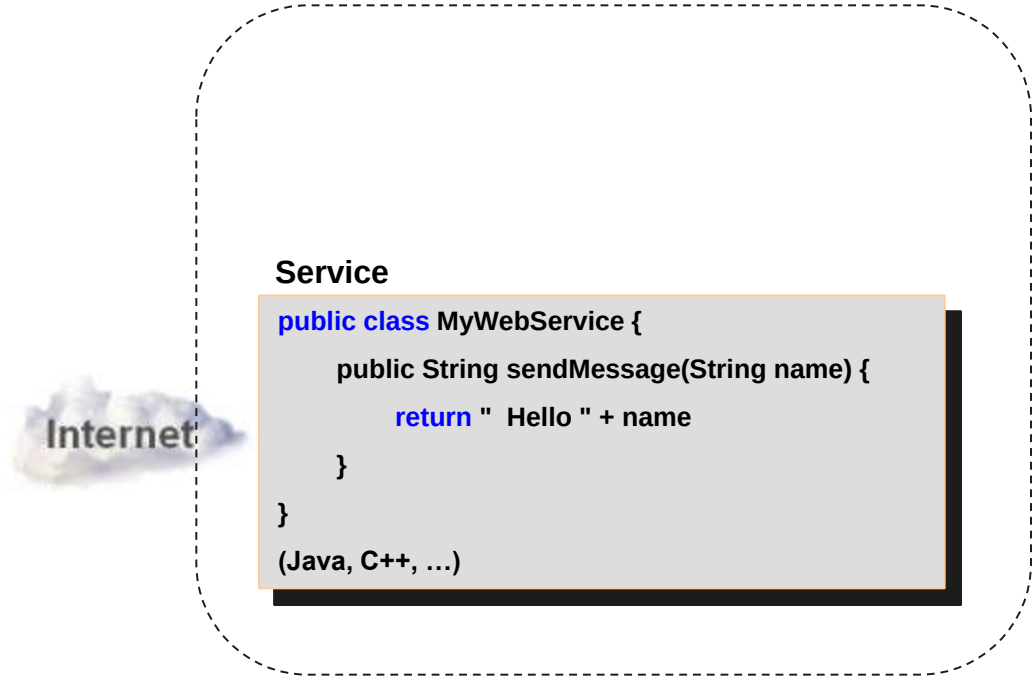
- La 3ème génération du Web
- Technologies Standards du Web
  - SOAP 1.1 (puis SOAP 1.2)
  - WSDL
- Technologies non standardisées
  - UDDI, DISCO
  - GXA (Global XML Architecture)
  - WSDD, WSFL, ASMX, …

Cinématique générale

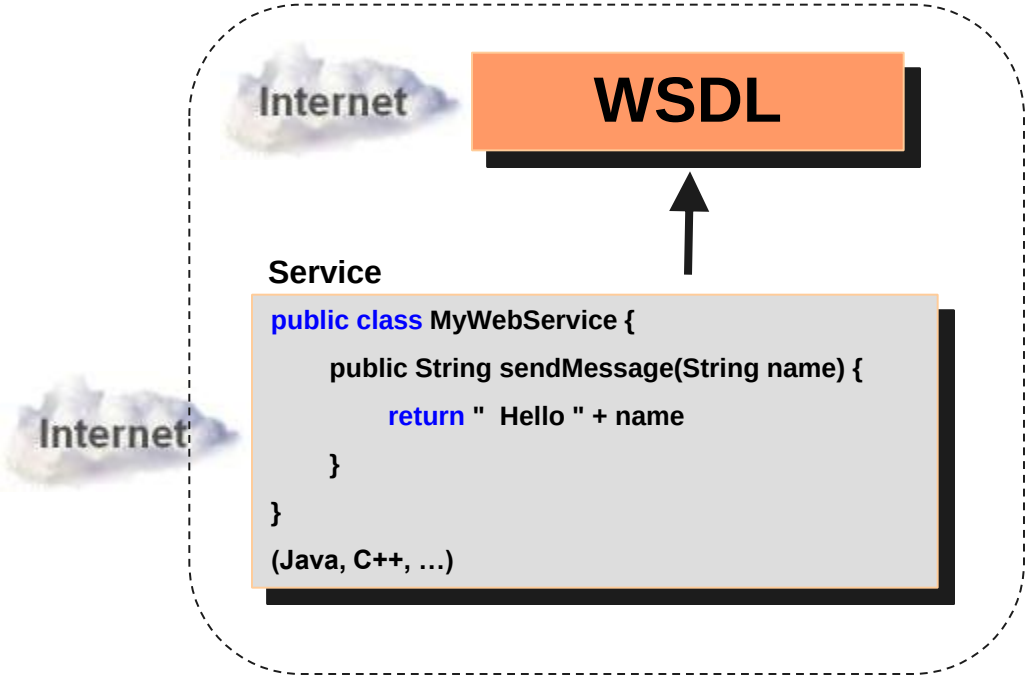
Service

```
public class MyWebService {
        public String sendMessage(String name) {
                return "  Hello " + name
        }
}
(Java, C++, …)
```

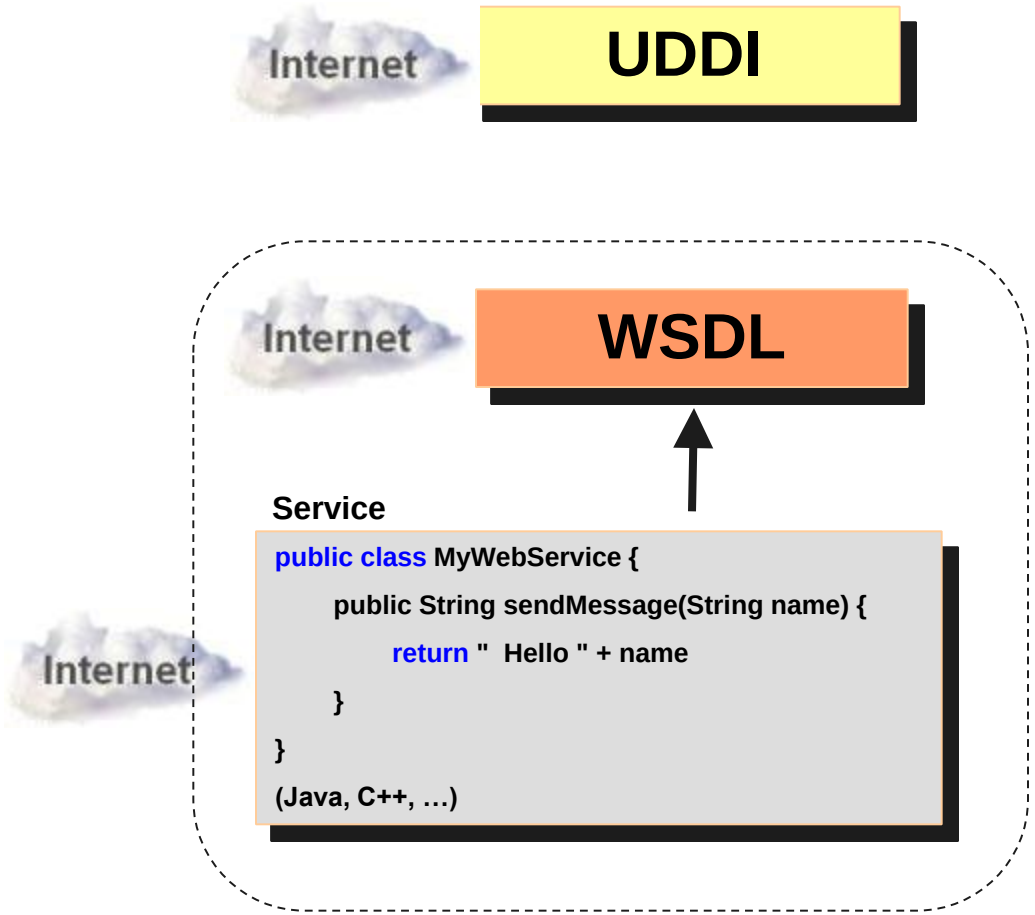
Cinématique générale



Cinématique générale



Cinématique générale



Cinématique générale

Internet

UDDI

Internet

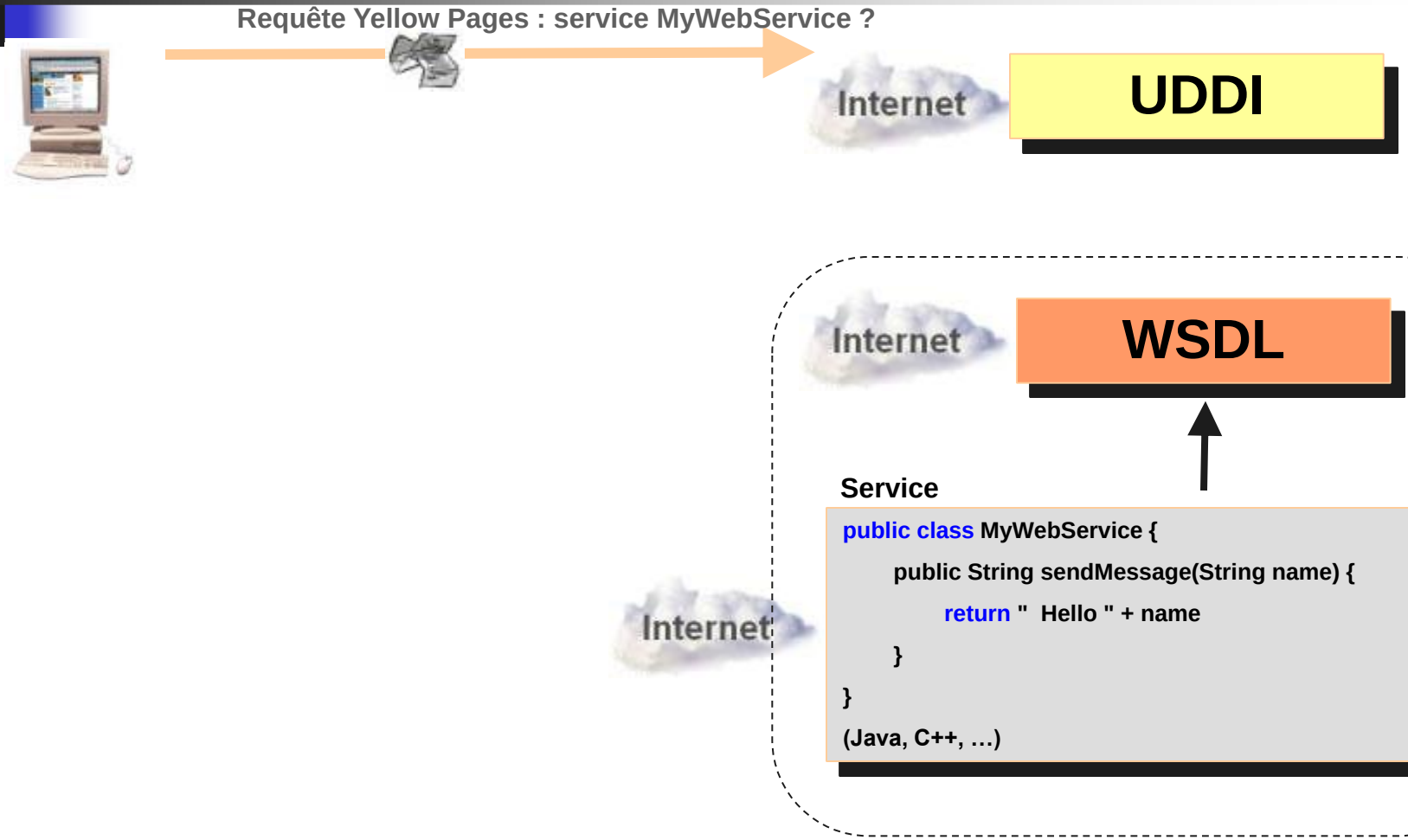
WSDL

Service

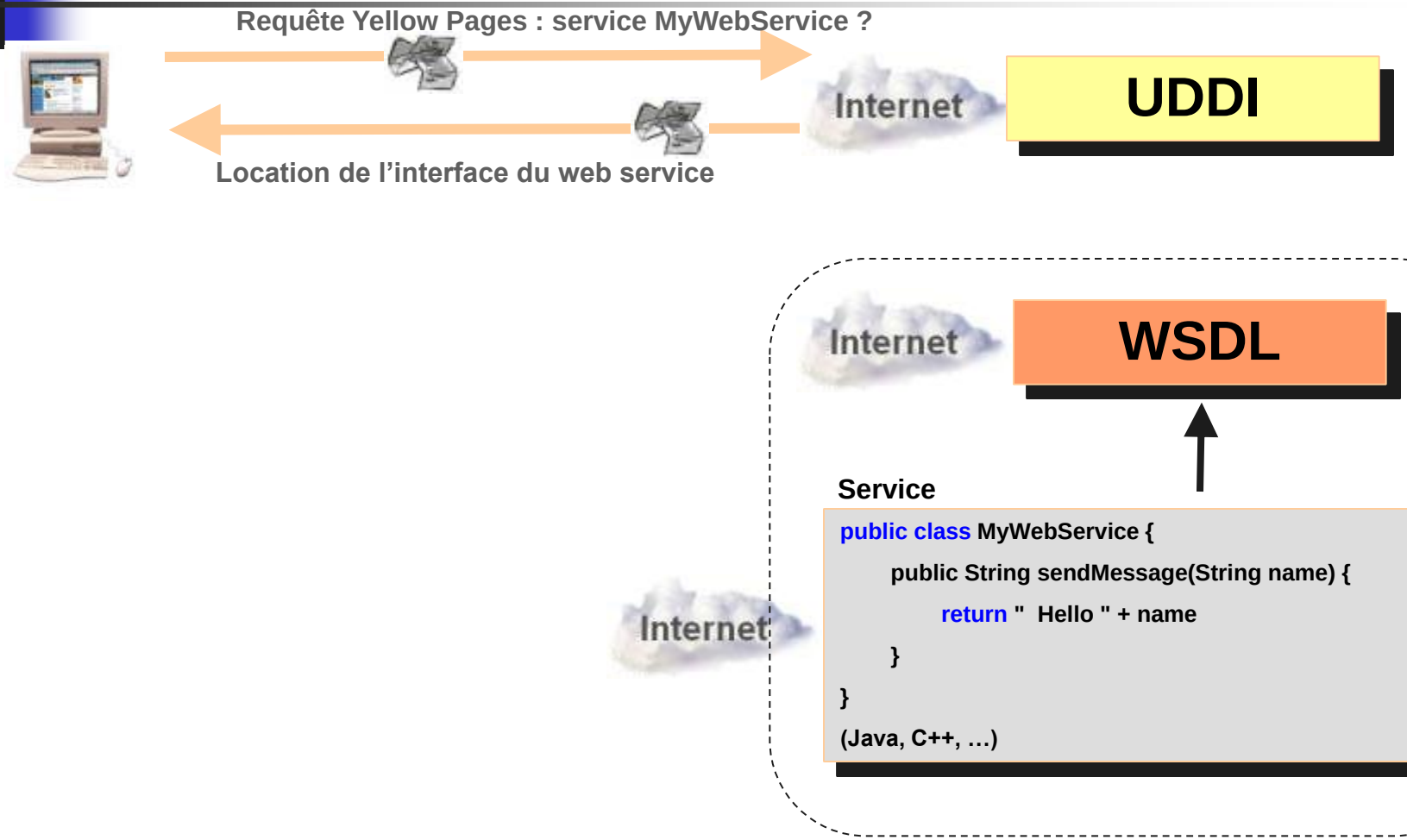
```
public class MyWebService {
        public String sendMessage(String name) {
                return "  Hello " + name
        }
}
(Java, C++, …)
```

Internet

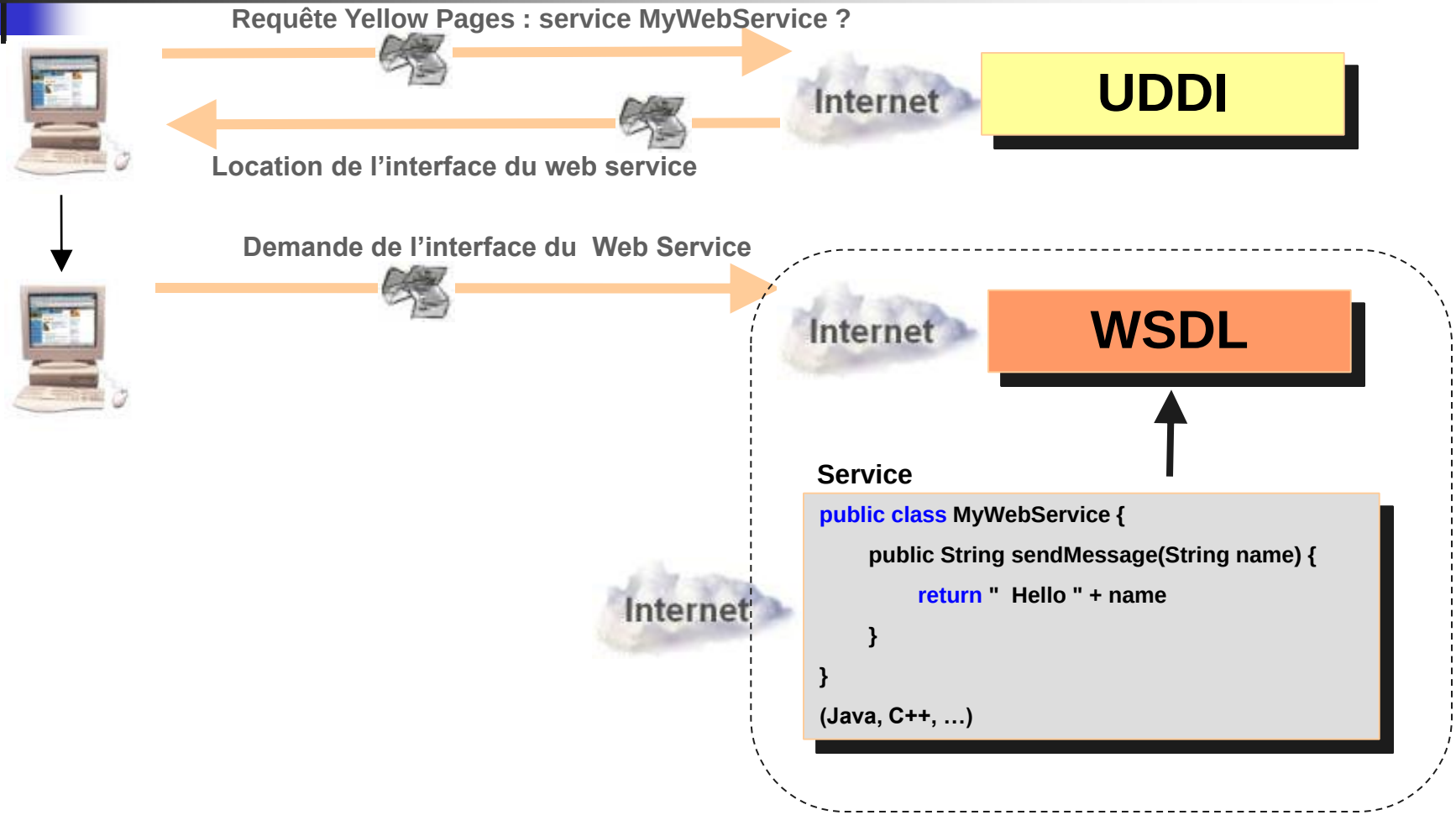
Cinématique générale



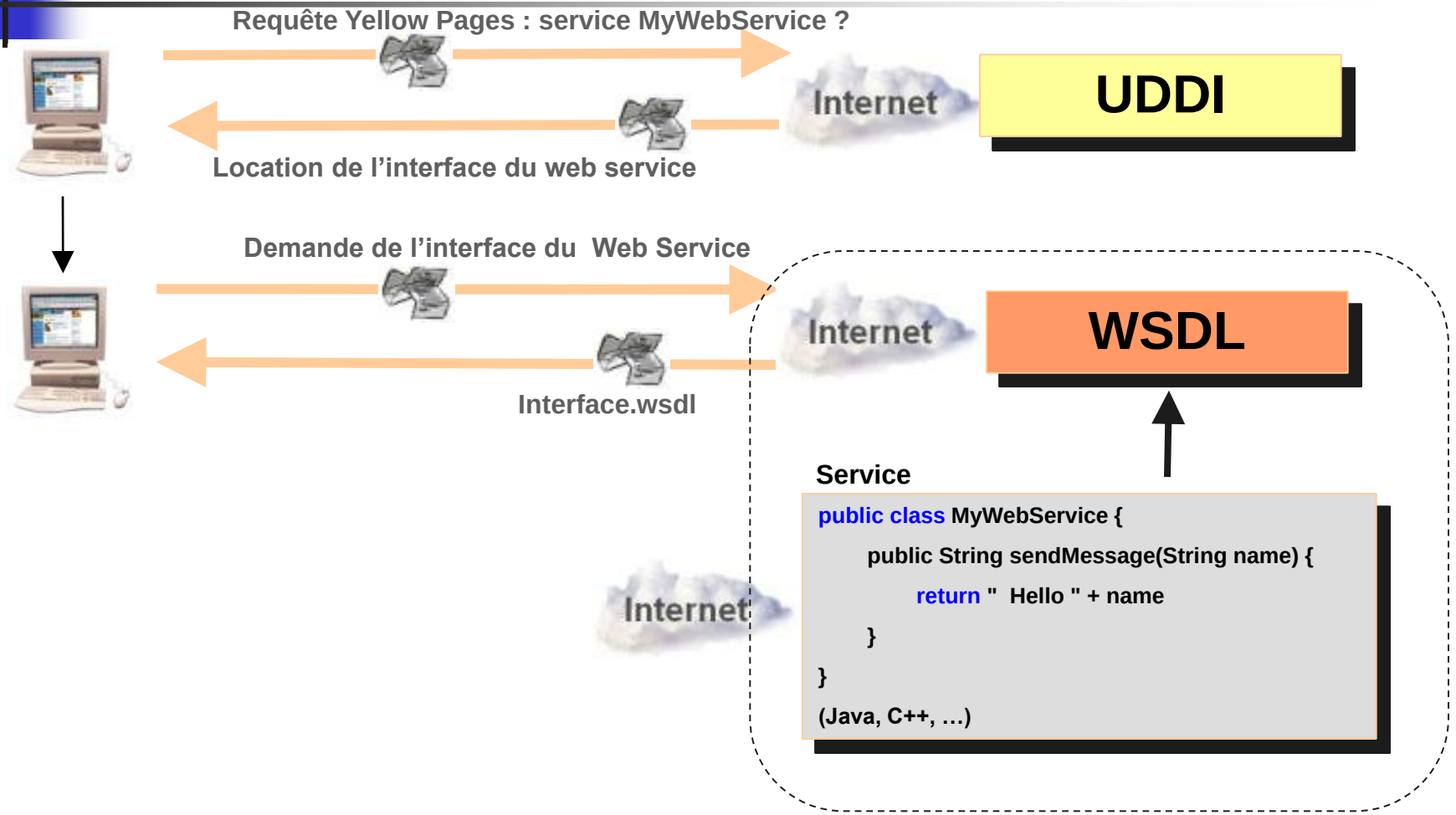
Cinématique générale



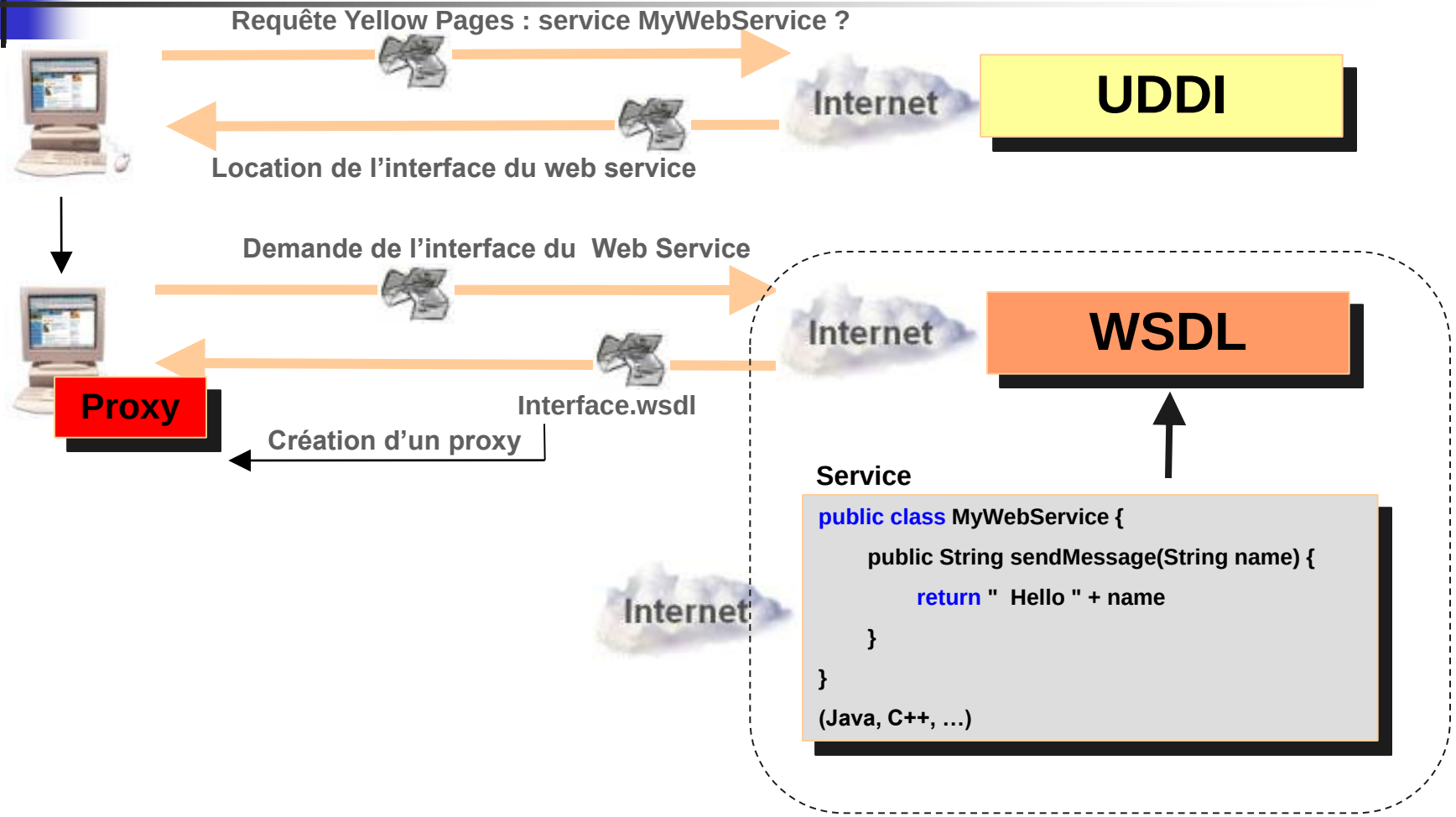
Cinématique générale



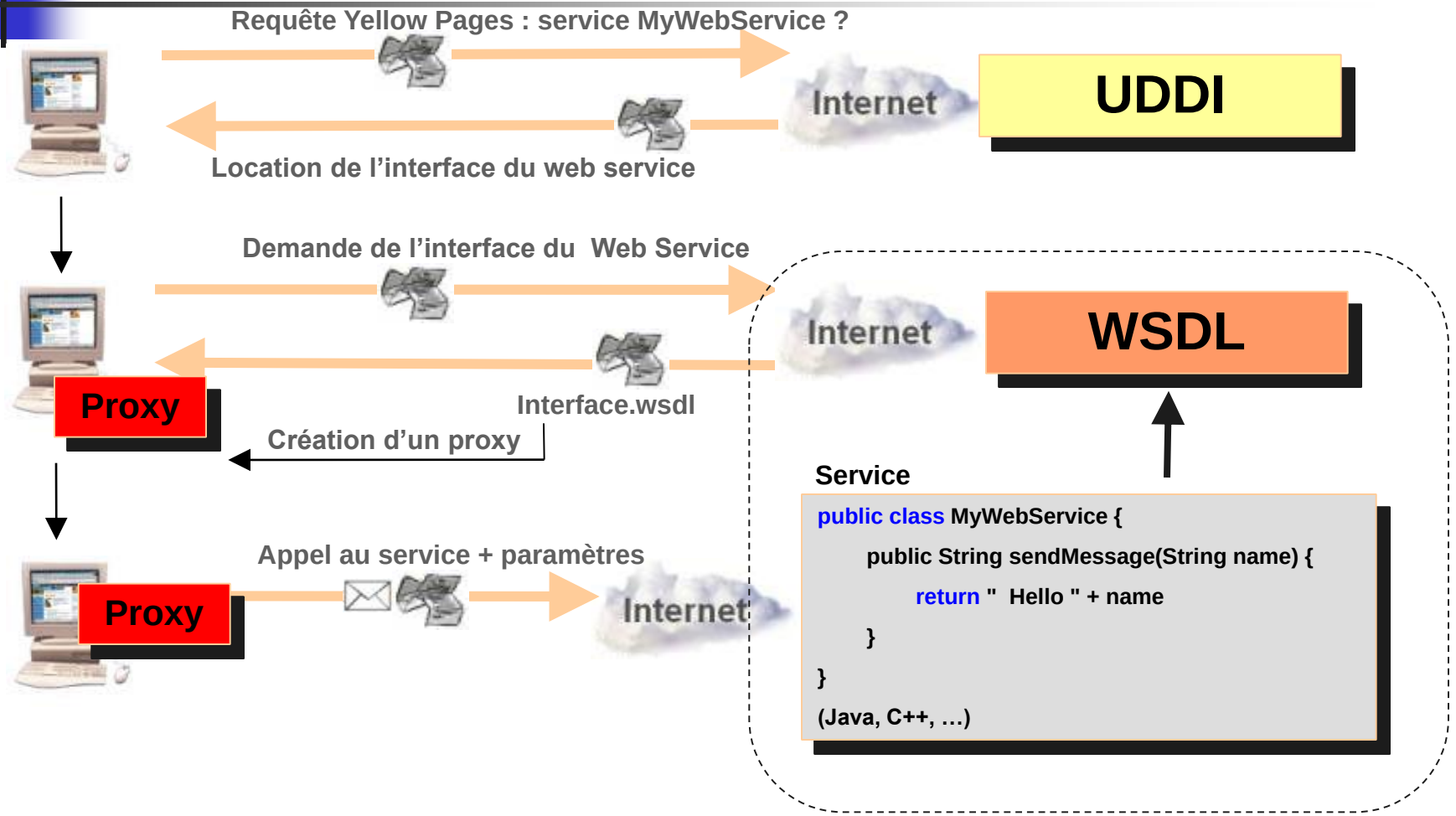
Cinématique générale



Cinématique générale



Cinématique générale



Cinématique générale