

Formation Java Frameworks

Les Services Web

Sommaire

- Les services web: originalité et bénéfices
- La seconde vague
- Orchestration
- Interactivité
- Adaptation
- Administration

Les années « services web »

- Implication totale des grands acteurs.
- Apparition de nombreux acteurs dédiés, principalement dans le monde Java (CapeClear, The Mind Electric, Systinet, Bowstreet, Shinka etc.)
- Virage des ASP, des éditeurs d'EAI et de middleware classique.
- Support multi-plateformes, multi-langages (C, Perl, Smalltalk, Python, Cobol, PL1, Ada etc.)
- Explosion des séminaires, revues etc.

Rappel technique sur les services web

- Les services web constituent une solution, parmi d'autres, à un problème ancien: comment faire communiquer des programmes (potentiellement distants) entre eux ?
- Services web, RPC, objets distribués (Corba, RMI, DCOM, .NET Remoting etc.), MOM etc.

=> Variations sur un même thème.

D'où viens le succès ?

- Utilisation d'XML pour représenter les échanges entre applications et les interfaces des services Web.
- Émergence des architectures « orientées services ».
- Accord, pour la première fois, de toute l'industrie – compris Microsoft, autour d'un standard d'interopérabilité (SOAP).
- Annuaires mondiaux de services.

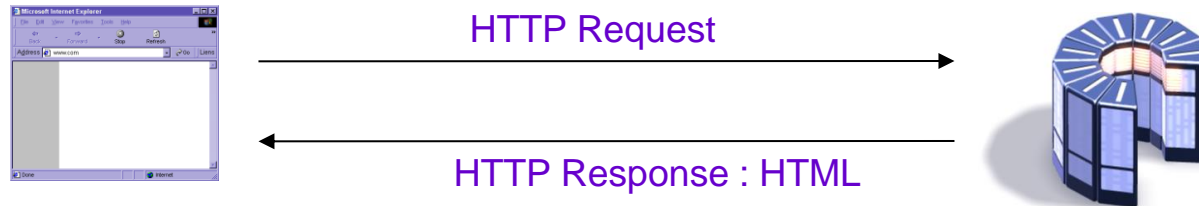
Bien que ces éléments soient importants, ils ne fondent pas aujourd'hui le succès de l'approche service web.

Une idée simple et astucieuse

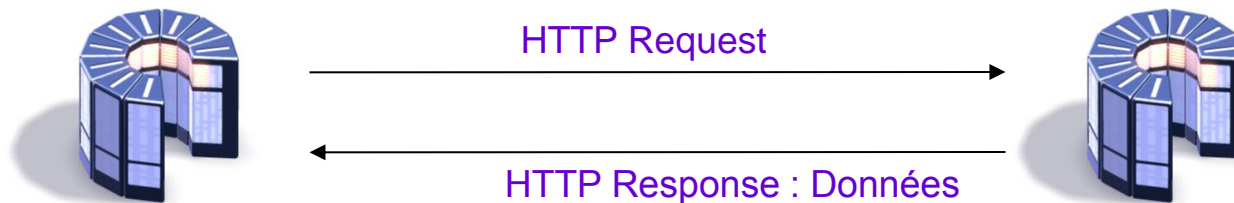
- ➡ Utiliser l'infrastructure web existante, c.a.d. tout ce qui tourne autour d'HTTP, comme support des interactions entre applications distribuées.

Services web: interactions programmatisques sur HTTP

Web classique interactif : interaction utilisateur/serveur



Services Web = Appels programmatiques sur le Web
Communication « Application to Application » (A2A)



Bénéfice: réduction radicale des coûts

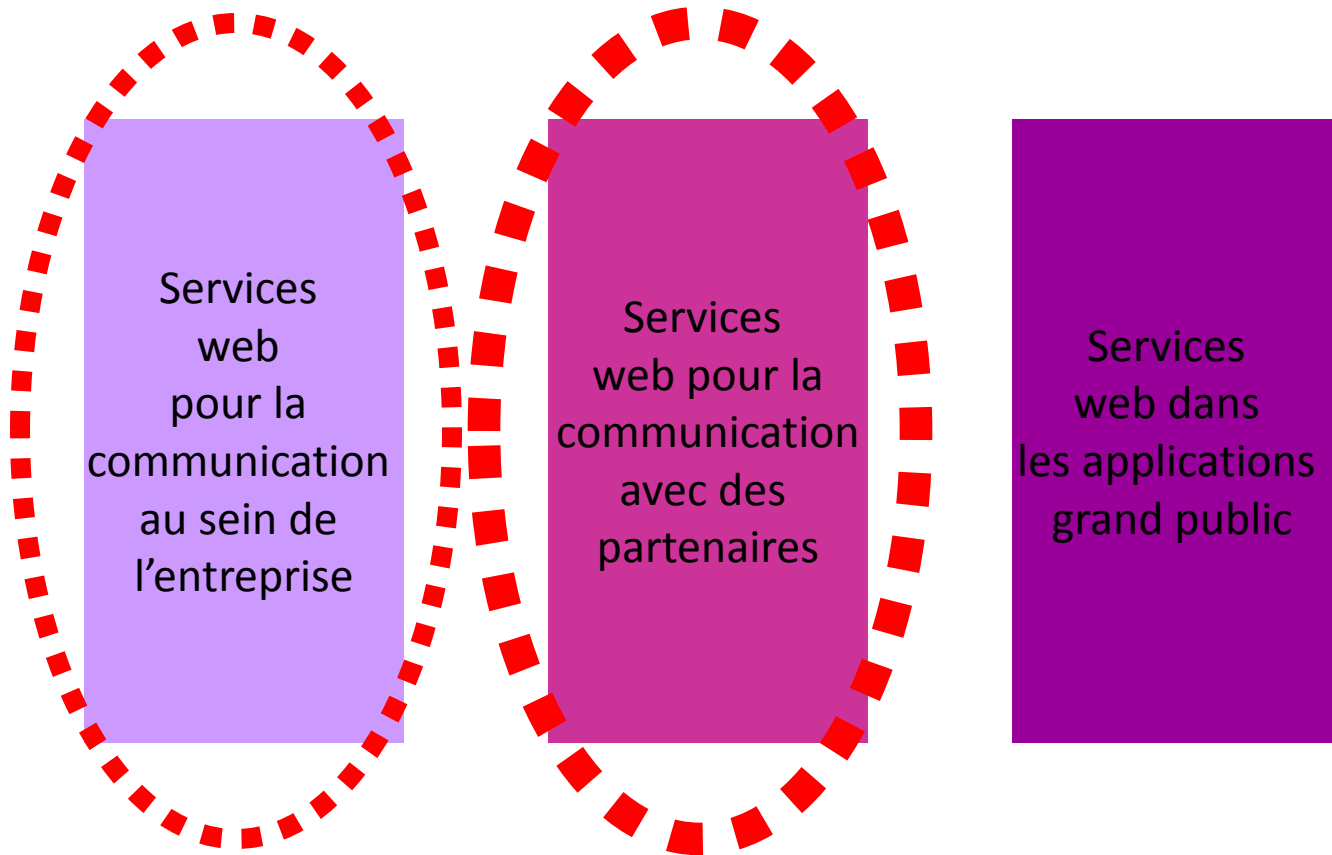
- Réutilisation des outils actuels, qu'il soient basés sur un serveur d'application, du scripting à la PHP, des programmes cgi etc.
- Réutilisation des compétences.
- Seuls les éléments touchant directement à la génération du HTML ou à la gestion d'une interaction utilisateur ne sont pas réutilisés.
- Coté client, il suffit de savoir émettre une requête HTTP !

Réutilisation de l'infrastructure web

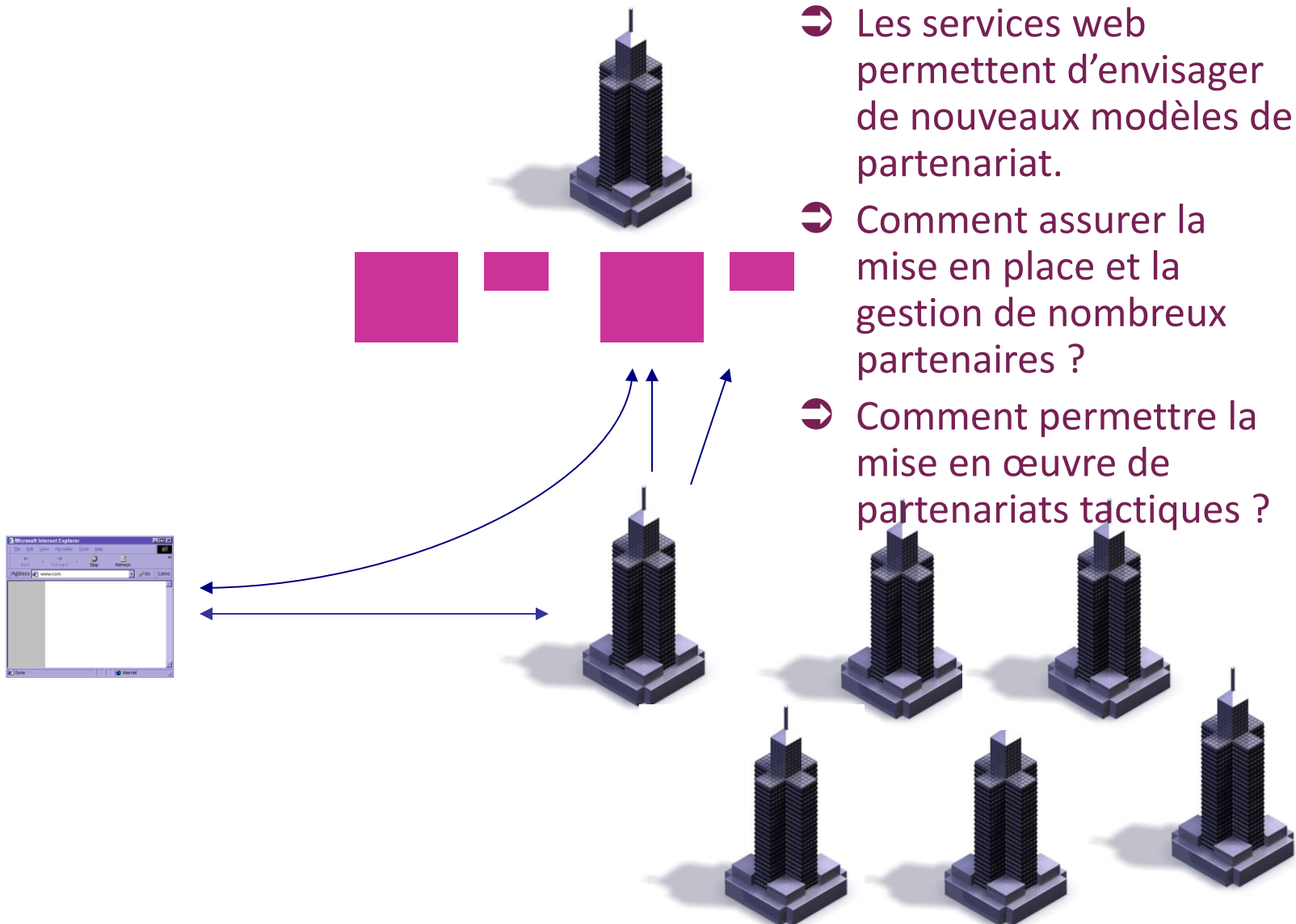
- Mise en ligne d'un service web
- Authentification
- Cryptage
- Firewalls
- Scalabilité
- Load-balancing, clustering, fault-tolerance
- Environnement d'exécution
- Administration / Logs
- Semantic web & nouveaux standards du web: approche REST

Types d'utilisation

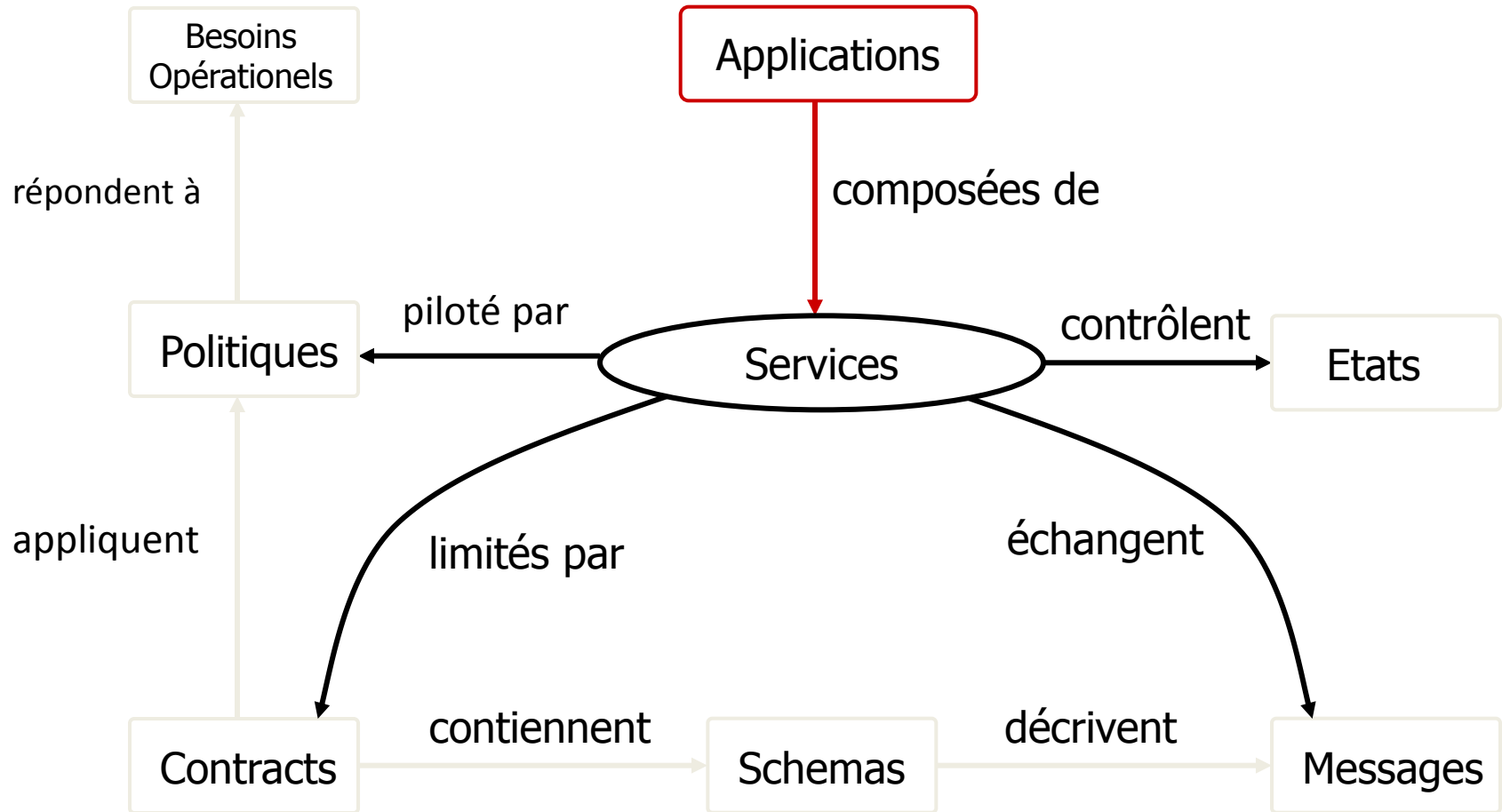
Orchestration - Adaptation - Administration



Partenariats multiples



Structure Fonctionnelle



Composants des Web services

① Acteurs

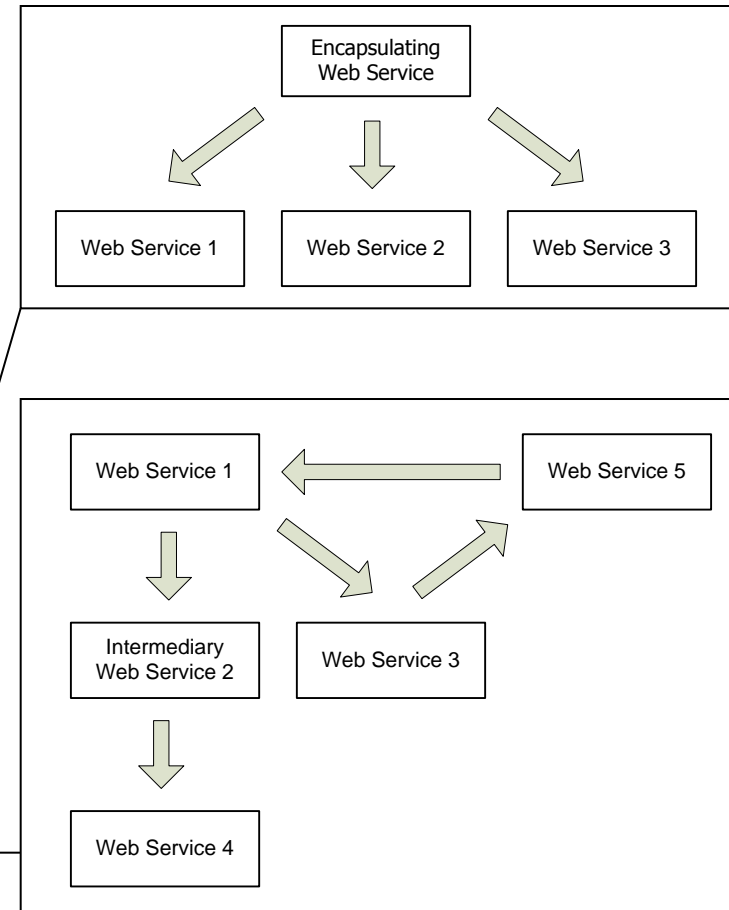
- Utilisateurs : individus utilisant une interface d'abstraction
- Requesters : "Clients" des Web Services
- Intermediary : capable de traiter une partie de la requête
- Providers : servent la requête

② Ressources

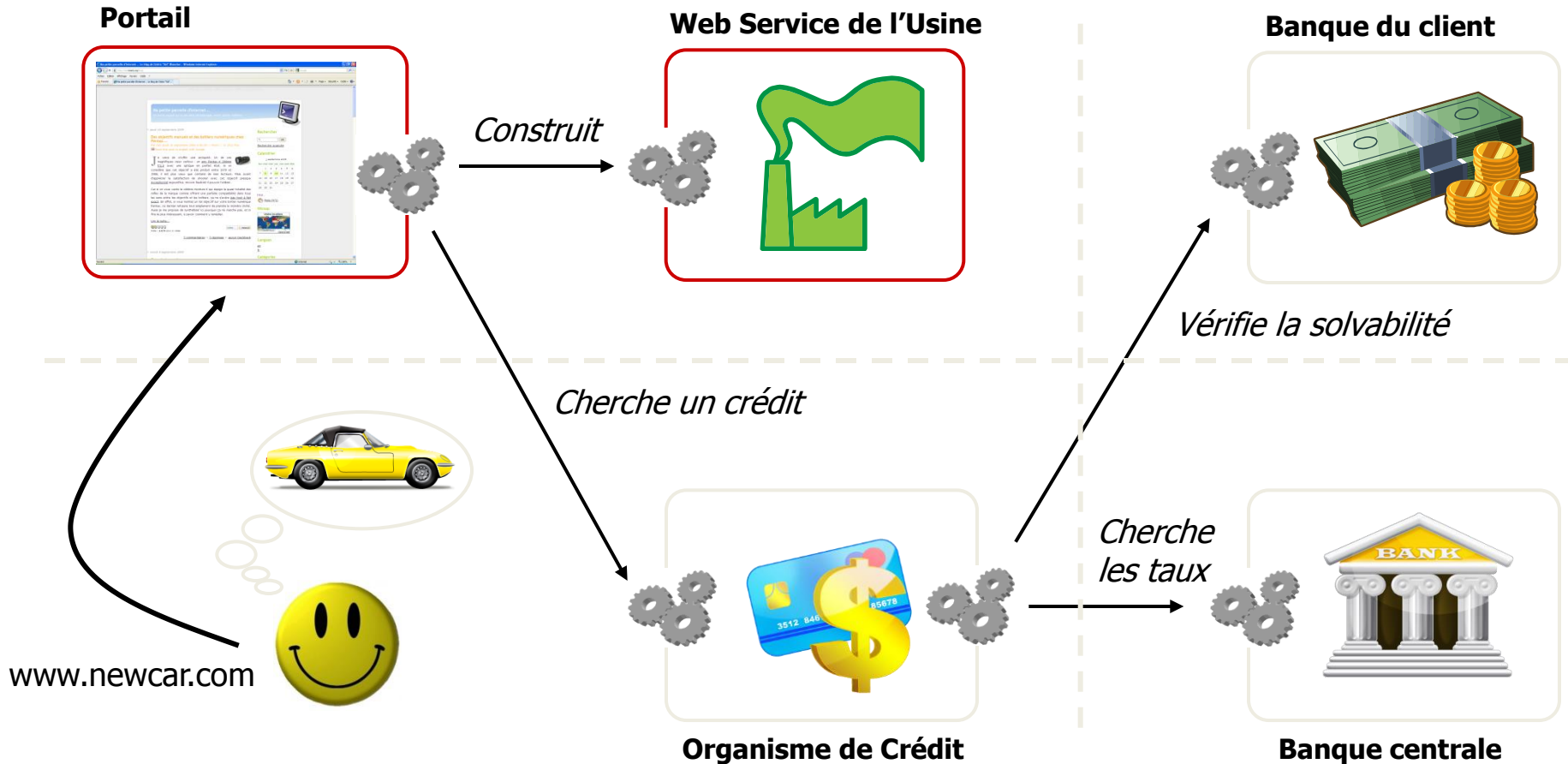
- Registres : fournissent la description et les points d'accès
- Portail : Frontal des "Requester" pour les utilisateurs
- Communication : Basée entièrement sur SOAP

③ Coordination

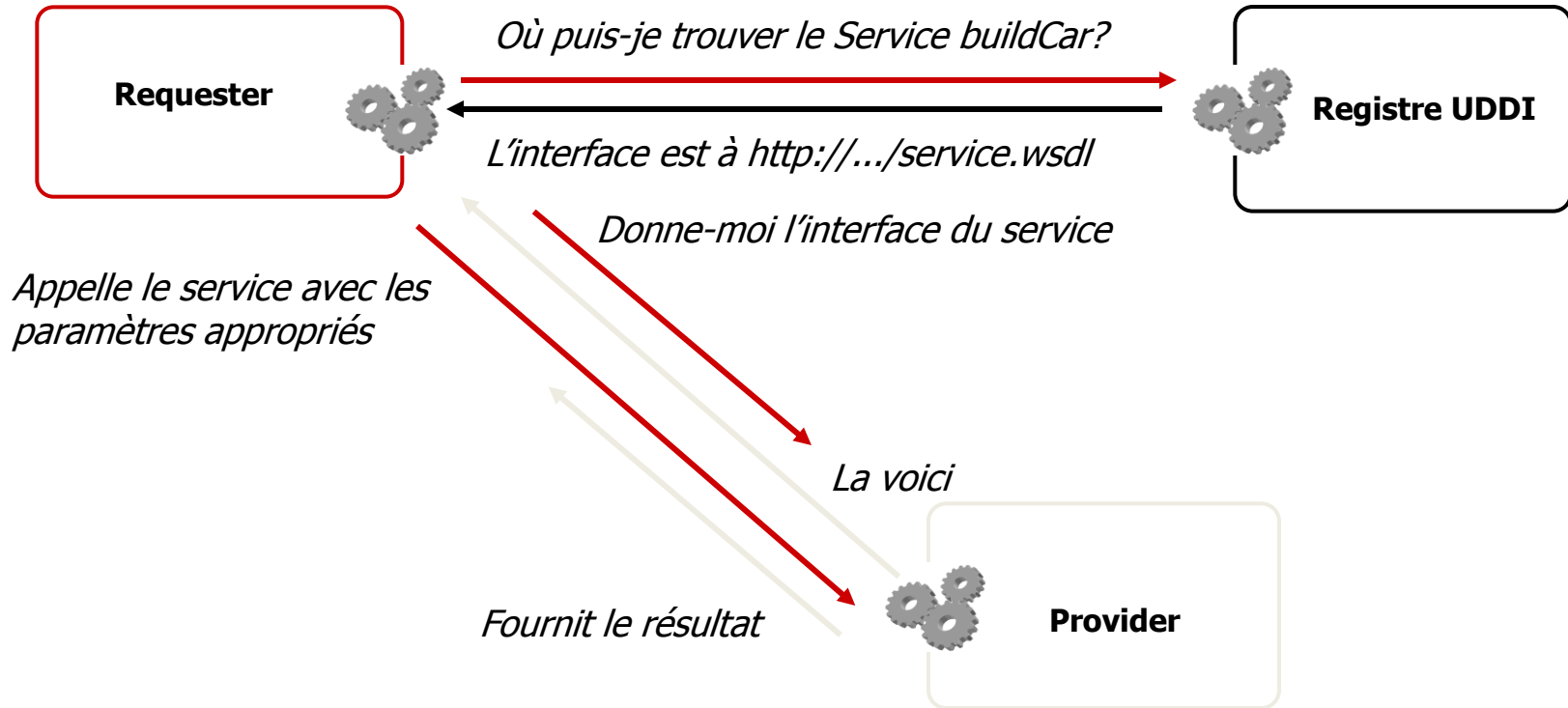
- Organise le traitement entre "providers"
- Orchestration : 1 service appelle les autres
- Chorégraphie : plusieurs services en appellent d'autres



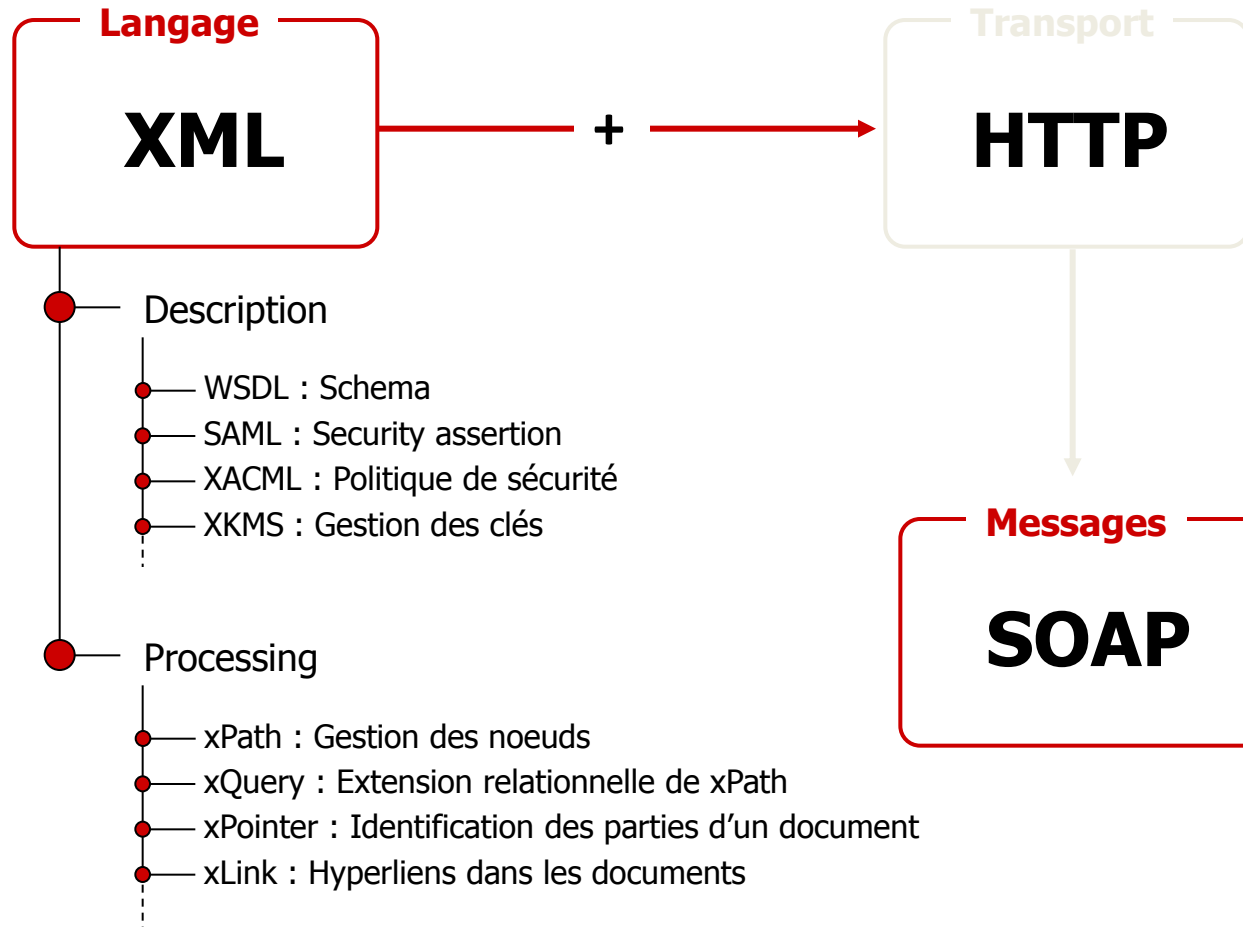
Exemple de Web Service



UDDI : Registres de Web Services

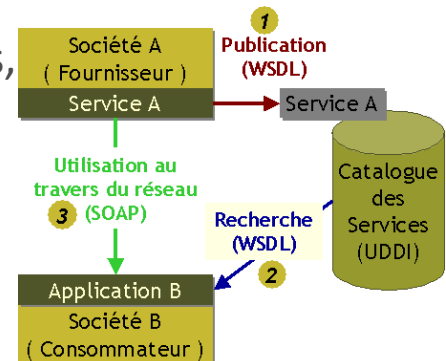


Technologies



Le socle technologique de base

- L'Architecture Web Services met en œuvre conjointement les spécifications :
 - SOAP : Simple Object Access Protocol
 - ▶ Protocole de type RPC utilisant XML pour la structuration de ses messages
 - ▶ Initialement proposé par Microsoft, désormais géré par le W3C
 - WSDL : Web Service Description Language
 - ▶ Il faut être capable de décrire de manière unifiée les services pour pouvoir les invoquer
 - ▶ WSDL est une spécification de description des Web Services
 - ▶ WSDL est un complément de SOAP (peut être vu comme l'IDL de CORBA)
 - UDDI : Universal Description, Discovery and Integration
 - ▶ Annuaire des Services Web mis à disposition par les entreprises, la sélection et la mise à disposition des descriptions de services



SOAP - Le protocole

- Est un protocole entièrement **basé sur le langage XML** :
 - Définit la structure du message (l'enveloppe) et les données véhiculées (le corps)
- Utilise des **protocoles standards de l'Internet** : HTTP, SMTP ou encore FTP :
 - Le choix du protocole est guidé par les contraintes techniques du système ou encore le mode de communication désiré (synchrone ou asynchrone)
- **Est extensible**, il peut être complété par d'autres spécifications XML pour apporter des services de plus haut niveau tels que :
 - Les pièces jointes
 - Le routage et les intermédiaires
 - La garantie de délivrance
 - La sécurité
 - Le contexte et la confidentialité
 - Les transactions
 - La qualité de Service (QoS)
- Le protocole SOAP peut être considéré comme un « **standard de fait** » de par son adoption par un grand nombre d'éditeurs et sa prise en main par le W3C

SOAP - Un exemple

POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-type: text/xml; charset="utf-8"
Content-length: nnnn
SOAPAction: "Some URI"

REQUETE





```
<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">
      <symbole>DIS</symbole>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

HTTP/1.1 200 OK
Content-type: text/xml; charset="utf-8"
Content-length: nnnn

REPONSE

```
<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:GetLastTradePriceResponse xmlns:m="Some URI">
      <Price>34.5</Price>
    </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP – Données échangées

- SOAP Véhicule des données au format XML
 - Enveloppe, en-tête et corps
 - Données échangées dans le cadre de l'appel du service (Contenu du corps, Pièces jointes éventuelles)
- Ces données peuvent être :
 - Des données quelconques  ***Document libre (forme et contenu)***
 - Des données XML  ***Document libre (contenu)***
 - Des données XML + Schéma (XSD)  ***Document métier définition externe***
 - Définies dans le Contrat (WSDL)  ***Définition interne***
- Du choix technique et de la granularité de description dépend :
 - Le contrôle sur la qualité des données échangées (typage +/- fort)
 - Le travail d'analyse des données en réception
 - Requête → Fournisseur de service
 - Réponse → Consommateur de service
 - Le couplage technique entre consommateurs et fournisseurs
 - Le couplage métier entre consommateurs et fournisseurs

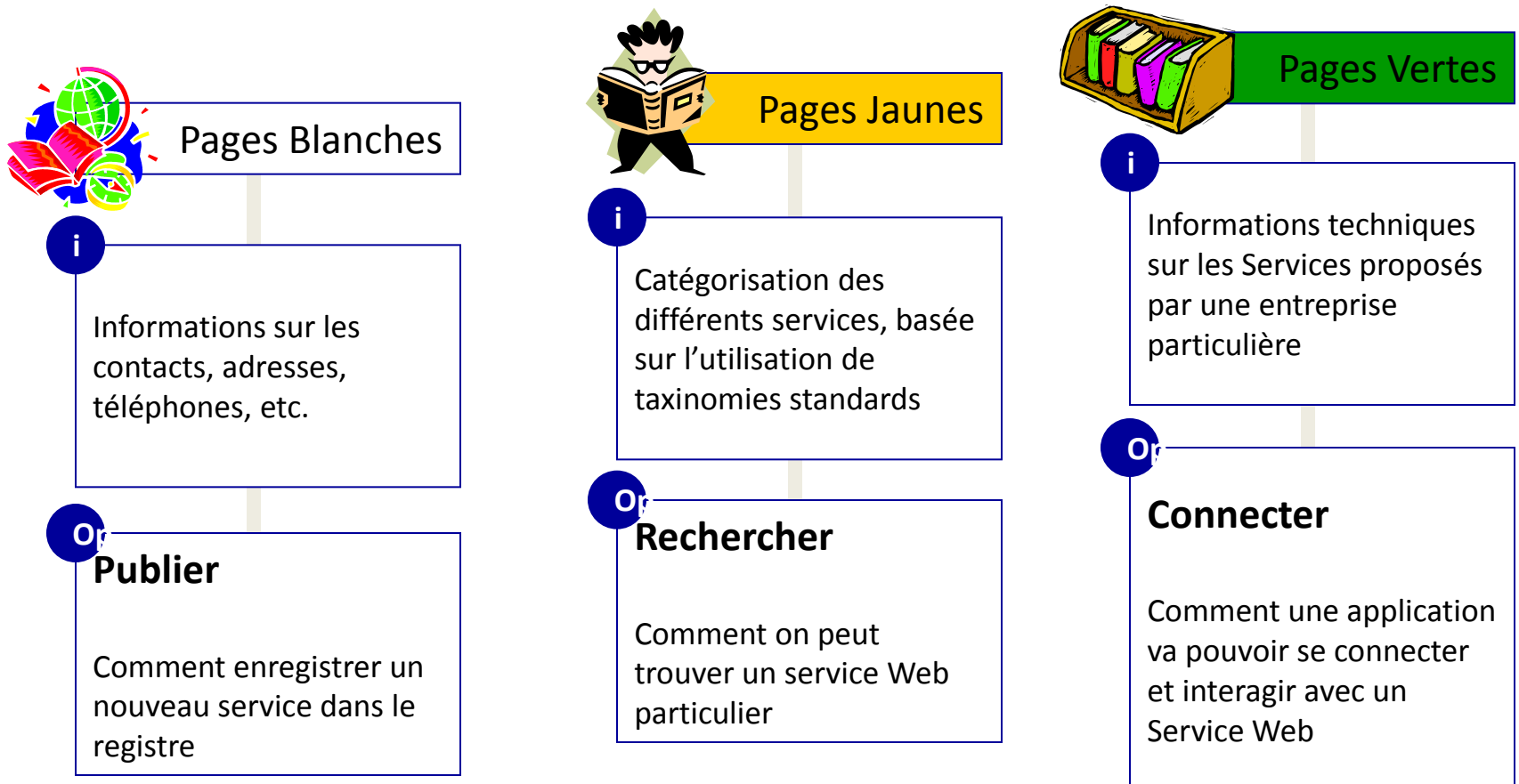
L'approche par document validé par un schéma combine :
grand degrés de liberté, qualité des contrôles et interopérabilité

WSDL

- WSDL est un **langage XML de description des Web Services**
- Un document WSDL décrit :
 - Ce que fait un Web Service
 - Où il se situe (i.e. quelles URLs et quels protocoles vont permettre son invocation)
 - Comment l'invoquer (i.e. quelles sont les méthodes disponibles et leurs paramètres, les types de données sont définies à base de XML Schema)
- Le rôle de WSDL est essentiel, puisque ce sont les documents WSDL qui seront échangés entre les partenaires de manière à ce qu'ils puissent techniquement mettre en œuvre la communication basée sur les Web Services
- L'intérêt de WSDL réside dans les quatre points suivants :
 - Le langage WSDL peut être utilisé pour définir complètement l'interface d'accès d'un service distant
 - Côté serveur, le fichier WSDL peut être généré automatiquement par introspection des classes qui implémentent le service
 - Côté client, le fichier WSDL peut être utilisé pour générer automatiquement un proxy (java, C#...) permettant d'invoquer le service
 - Le fichier WSDL peut être exporté dans un **annuaire UDDI** permettant ainsi qu'il soit découvert par interrogation de cet annuaire

UDDI

- UDDI (*Universal Description, Discovery, and Integration*) distingue trois types de registres :



JAX-RPC: standard mais lourd

```
import java.net.*;
import javax.xml.namespace.*;
import javax.xml.rpc.*;
...
URL WSDLLocation = new URL("http://test.on.com/h/s1?WSDL");
QName serviceName = new QName("http://www.on.com/Message.xsd", "ONWS");
QName portName = new QName("http://www.on.com/Message.xsd", "ONWS");
ServiceFactory serviceFactory = ServiceFactory.newInstance();
Service service = serviceFactory.createService(WSDLLocation, serviceName);
Call call = service.createCall(portName, "HelloWorld");
call.invoke(new Object[]{});
```

```
import electric.registry.*;
...
Registry.bind("http://test.on.com/h/s1?WSDL").invoke("HelloWorld", new Object[]{});
```

Adaptation

- Les services web peuvent intégrer une technologie d'adaptation.
- Le comportement des services web est configuré différemment selon les différents contextes d'appels (partenaires).
- Les outils d'adaptation doivent être utilisables par des hommes métiers.
- Tests.
- Adaptation en masse.
- Impact méthodologique.

Adaptation

- Deux types d'adaptation
 - Adaptation graphique (uniquement pertinente pour les services web interactifs).
 - Adaptation métier.
- L'adaptation implique les différents partenaires.
- Délégation de certains droits d'adaptation par le propriétaire du service web.
- Plateforme commune d'adaptation.

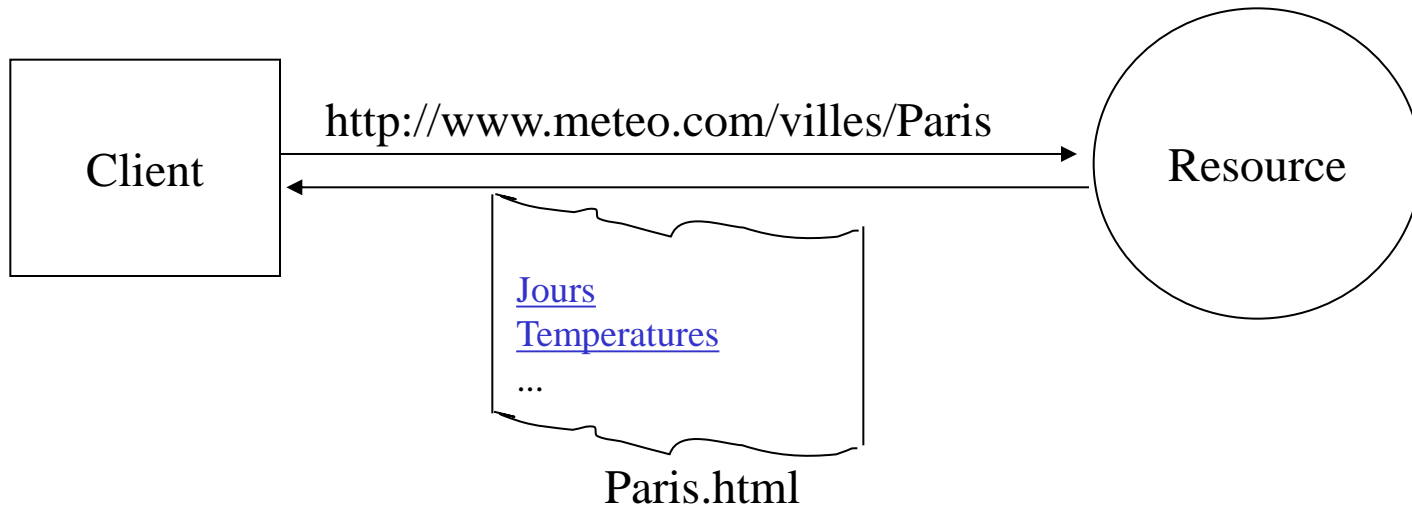
REST c'est quoi ?

- Thèse de Roy Fielding en 2000
- Un style d'architecture
- Un ensemble de contraintes
 - Client /serveur
 - Sans états (Stateless)
 - Cache
 - Interface uniforme
- La plus connue des implémentations de REST est HTTP

Les principes clefs

- Une ressource
- Un identifiant de ressource
- Une représentation
- Interagir avec les ressources
 - Exemple avec HTTP : GET, POST, PUT et DELETE

Pour résumer



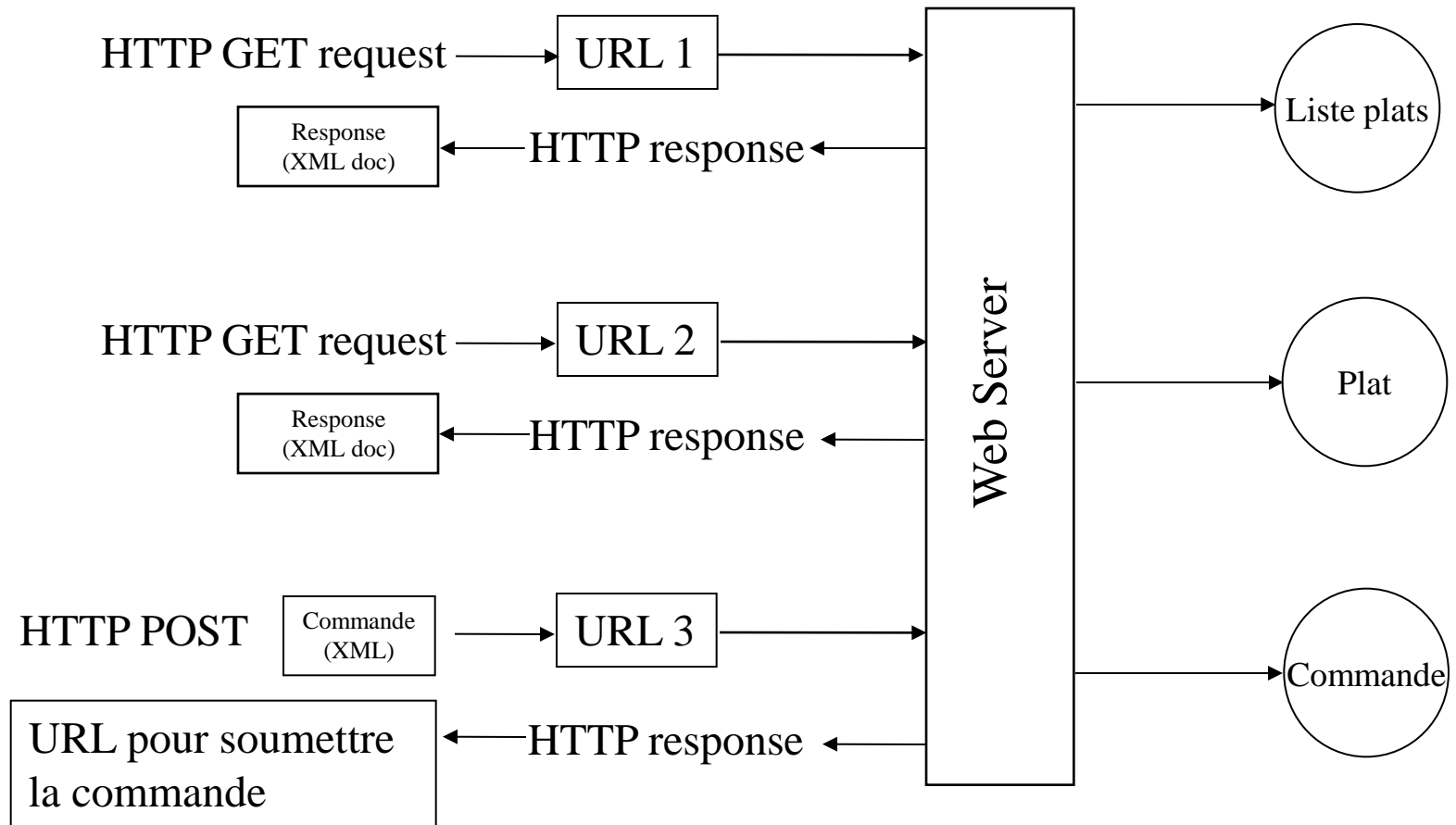
Un service RESTful

- Identifier les ressources
- Définir les URIs
- Spécifier les méthodes des interfaces
- Lier les ressources

Exemple (1/6)

- Un traiteur propose sur son site plusieurs services à ses clients :
 - Obtenir la liste des plats disponibles
 - Obtenir des informations sur un plat précis
 - Passer une commande

Exemple (2/6)



Exemple (3/6)

- La liste des plats est disponible à l'URL suivante : <http://www.monresto.com/plats/>
- Le client reçoit une réponse sous la forme suivante :

```
<?xml version="1.0"?>
  <p:Plats xmlns:p="http://www.monresto.com/"
    xmlns:xlink="http://www.w3.org/1999/xlink">
    <Plat id="0001" xlink:href="http://www.
monresto.com/Plats/0001"/>
    <Plat id="0002" xlink:href="http://www.
monresto.com/Plats/0002"/>
    <Plat id="0003" xlink:href="http://www.
monresto.com/Plats/0003"/>
  [...]
</p:Plats>
```


Exemple (4/6)

- Les détails d'un plat se trouvent à l'URL : <http://www.monresto.com/plats/0002>
- D'où la réponse :

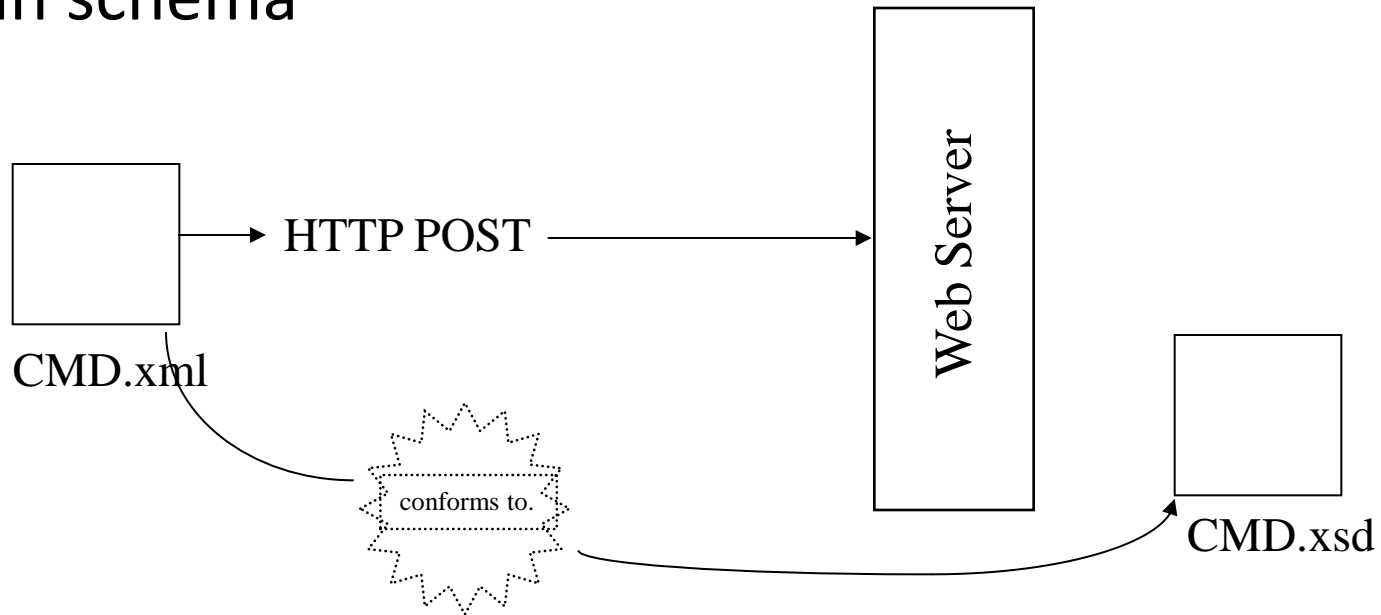
```
<?xml version="1.0"?>
  <p:Plat xmlns:p="http://www.monresto.com"
          xmlns:xlink="http://www.w3.org/1999/xlink">

    <Plat-ID>0002</Plat-ID>
    <Nom>Rouleaux de Printemps</Nom>
    <Description>Entrée</Description>
    <Details
xlink:href="http://www.monresto.com/plats/00002/details"/>
      <CoutUnitaire monnaie="EUR">3</CoutUnitaire>

    </p:Plat>
```

Exemple (5/6)

- Le service « Passer commande »
 - Créer une instance de « commande » conforme à un schéma



Exemple (6/6)

- Le service « Passer une commande » répond par une URL vers la commande soumise.

