Services web

Katerina TZOMPANAKI (atzompan@u-cergy.fr)
Dan VODISLAV

Université de Cergy-Pontoise
Master Informatique M1
Cours IED

Plan

- Principes
- Services SOAP
  - WSDL
  - UDDI
  - Services SOAP en Java
- Services REST
  - Services REST en Java

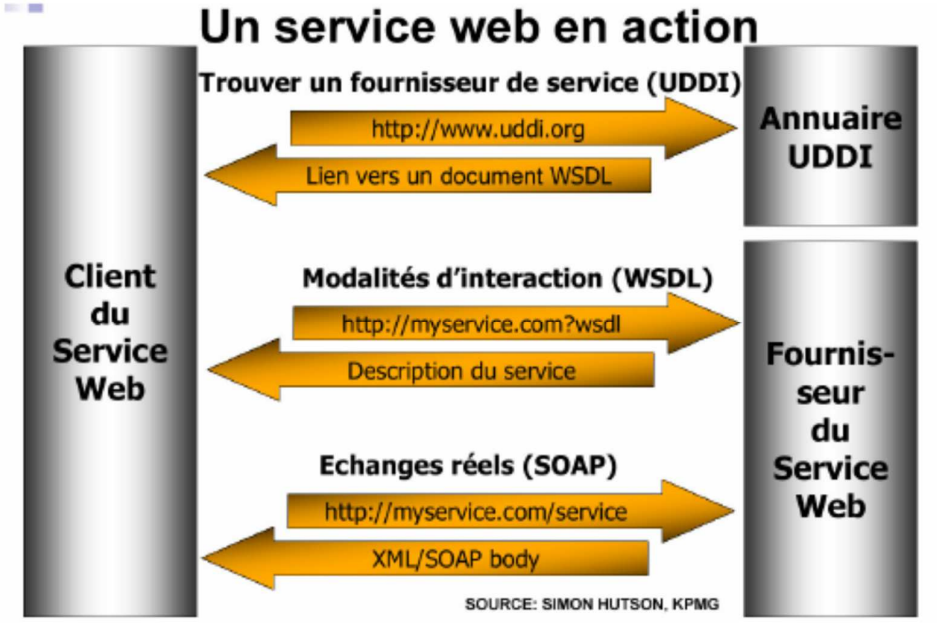
Services web

- Infrastructure pour le développement d'applications distribuées sur le web
- Le web → environnement distribué spécifique:
  - Contrôle limité des sites, débit faible
  - Utilisation des protocoles web (ex: HTTP) avec leurs limitations
  - Fonctionnalités, présentation moins riches (HTML)
  - Clients légers
- Objectif: réaliser des applications distribuées avec les contraintes imposées par le web

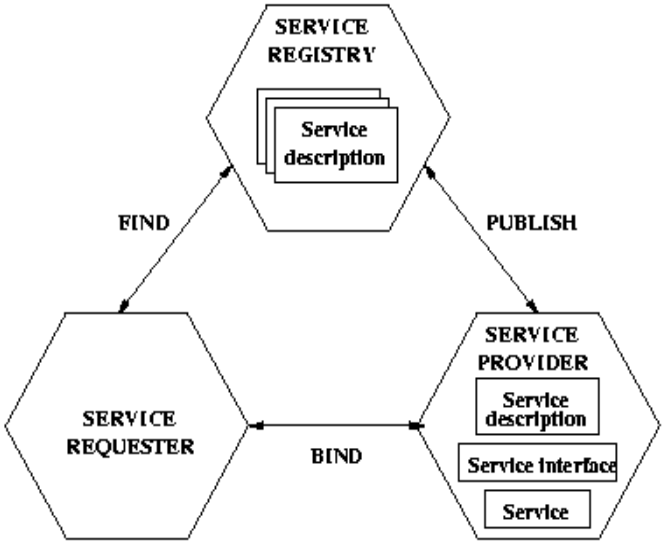
Scénario d'utilisation

1. Définition du service (fournisseur)
   - Description WSDL des entrées/sorties, des caractéristiques du service
2. Publication du service (fournisseur)
   - Publication de la description WSDL dans un annuaire (UDDI)
3. Recherche de service (client)
   - Recherche d'un service dans un annuaire → adresse du service choisi
4. Enregistrement au service web (client)(*optional*)
   - Enregistrement auprès du fournisseur pour accéder au service trouvé
5. Appel du service (client)
   - Exécution du service avec les paramètres fournis par le client
6. Composition (client, fournisseur)
   - Utilisation du résultat pour l'appel à d'autres services (client)
   - Appel d'un autre service lors de l'exécution du service appelé (fournisseur)

Scénario (suite)



Architecture orientée services (AOS)

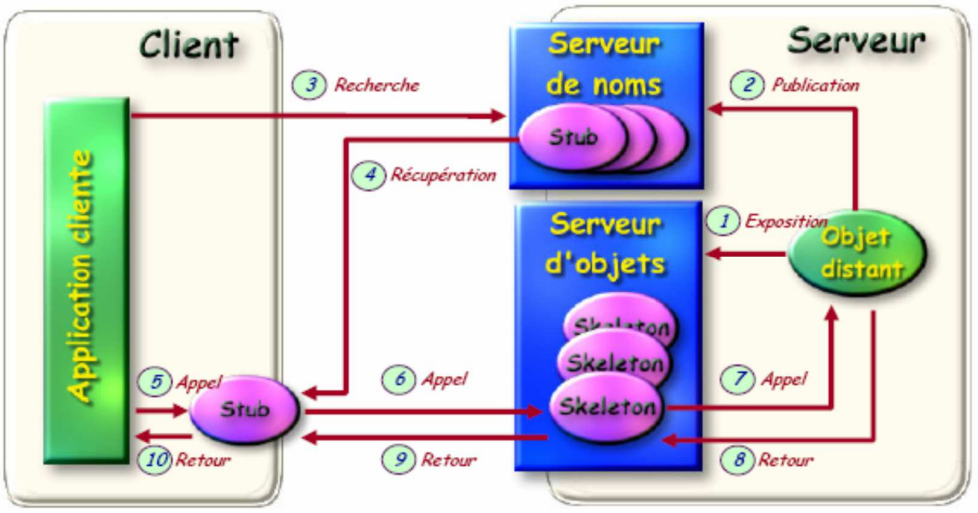


Source: G. Alonso

- Schéma proposé par IBM
  - Bas niveau, construction ascendante
  - D'autres architectures (ex. ebXML): haut niveau, construction descendante

Comparaison avec les solutions middleware

- Les mêmes notions existent dans CORBA, EJB, RMI



Source: R. Voyer

Technologies pour services web

- Technologies de base
  - SOAP : « Simple Object Access Protocol »
    - RPC par appel de service web
    - Protocole de communication à l'appel de services web
  - WSDL : « Web Service Description Language »
    - Langage de description de services web
    - Paramètres, type du résultat, opérations fournies par le service, points d'accès
  - UDDI : « Universal Description, Discovery and Integration »
    - Protocole de description et d'interaction avec des annuaires de services web
- Autres aspects: énormément de standards
  - Orchestration, composition: WSBPEL, WS-Coordination, WS-CDL
  - Sémantique: OWL-S, WSDL-S
  - Sécurité: WS-Security
  - …

SOAP

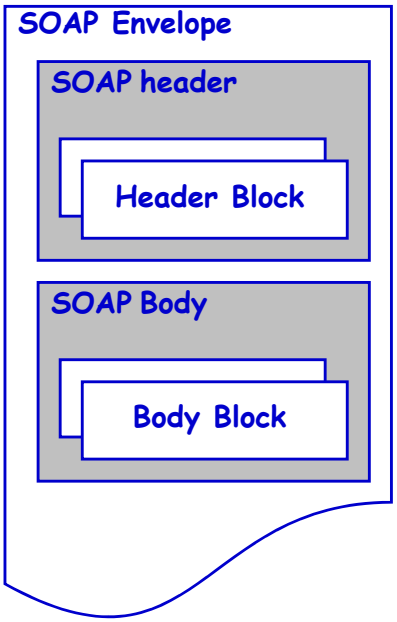
- Simple Object Access Protocol, norme W3C
  - SOAP 1.0: 1999, basé sur HTTP
  - SOAP 1.1: 2000, plus générique, autres protocoles
  - SOAP 1.2: recommandation W3C, 2007
- Couvre 4 aspects
  - Format XML des messages échangés
  - Comment un message SOAP est transporté sur le web par HTTP, SMTP.
  - Règles de traitement des messages SOAP
  - Conventions de transformation d'un appel RPC en SOAP et d'implémentation d'une communication RPC

SOAP: objectifs

- But initial : infrastructure minimale pour faire du RPC sur le web
  - Utilisation de XML pour les échanges
  - Structure de messages très simple
  - Basé sur HTTP pour résoudre le problème des pare-feu
- Raisons pratiques
  - Éviter les problèmes de CORBA sur le web, qui doit en pratique s'appuyer de toute façon sur HTTP à cause des pare-feu
  - Disposer d'une couche facile à mettre en œuvre au-dessus des plateformes middleware pour une intégration à travers le web
    - Interopérabilité system/plateforme
  - Transformer SOAP par la suite en support générique d'échange de messages sur le web, au-delà de RPC et de HTTP

Structure des messages SOAP

- Messages : « enveloppes » où l'application met les données à transmettre
  - Éléments XML avec des sous-éléments
- Structure
  - En-tête (optionnelle)
    - Niveau infrastructure
  - Corps (obligatoire)
    - Niveau application



Exemple

```
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
-------------------------------------------------------------------------------
  <SOAP-ENV:Header>                                                      H
        <t:Transaction                                                   E
          xmlns:t="some-URI"                                             A
          SOAP-ENV:mustUnderstand="true">                                D
            5                                                            E
        </t:Transaction>                                                 R
      </SOAP-ENV:Header>
-------------------------------------------------------------------------------
      <SOAP-ENV:Body>                                                    B
        <m:GetLastTradePrice xmlns:m="Some-URI">                         O
          <symbol>DEF</symbol>                                           D
        </m:GetLastTradePrice>                                           Y
      </SOAP-ENV:Body>
-------------------------------------------------------------------------------
</SOAP-ENV:Envelope>
```

En-tête SOAP

- Conteneur pour information indépendante de l'application
  - Ensemble de blocs (éléments XML)
  - Information de sécurité, de coordination, identificateurs de transaction, …
  - Chaque nœud sur le parcours source - destination peut traiter des blocs
- Qui traite les blocs d'en-tête?
  - Attribut « role »: qui a le droit de traiter le bloc
    - Rôles standard SOAP: *http://www.w3.org/2003/05/soap-envelope/role/...*
      - "*none*" : personne
      - "*next*" : tout nœud intermédiaire ou le destinataire final
      - "*ultimateReceiver*" : seul le destinataire final (par défaut)
    - Rôles spécifiques à l'application: chaque nœud connaît ses rôles applicatifs
  - Attribut « mustUnderstand »: obligation ou non de traiter l'élément
    - "true" : traitement obligatoire, si pas possible générer erreur
    - "false" : traitement optionnel (par défaut)
  - Attribut « relay » (SOAP 1.2): un bloc non traité doit être transmis au prochain nœud

Cheminement des messages SOAP

- Un message source → destination : chemin dans le réseau
  - Objectif: tenir compte de l'architecture du réseau (ex. middleware)
  - Attribut « role » dans chaque bloc d'en-tête (par défaut: "ultimateReceiver")
  - Chaque nœud dans le chemin regarde chaque bloc d'en-tête
    - Il traite (éventuellement) les blocs d'en-tête qui lui reviennent
      - Traitement obligatoire des parts avec « mustUnderstand » = "true"
    - Enlève éventuellement ces blocs de l'en-tête sauf pour les blocs non traités où l'attribut « relay » est présent
    - Retransmet le message avec l'en-tête mise-à-jour vers le nœud suivant
    - Si erreur, arrêt du cheminement et retour d'un message SOAP d'erreur
- Les modifications en chemin: seulement pour l'en-tête
- Le corps du message est traité seulement par le destinataire
- Remarque: il existe aussi une notion d' « intermédiaire actif », qui peut modifier le message reçu d'une façon dépendante de l'application

Corps SOAP

- Conteneur pour données spécifiques à l'application
- Divisé en blocs
  - Un bloc : traité seulement par le récepteur final
  - Certains blocs ont une signification prédéfinie
    - Pour traduire un appel RPC en SOAP
    - Pour signaler des erreurs
- Bloc d'erreur (« Fault ») : rapport d'erreur dans la traitement du message
  - Code: catégorie d'erreur (version, mustUnderstand, client, server)
  - Texte: explication textuelle, à afficher
  - Acteur: nœud à l'origine de l'erreur
  - Détail: information dépendante de l'application
  - D'autres éléments rajoutés dans SOAP 1.2

Appel RPC en SOAP

- RPC: deux messages SOAP (appel + réponse)
  - Message d'appel:
    - Élément : nom de fonction à appeler
    - Sous-éléments: paramètres d'appel de la fonction
  - Message de retour: deux possibilités
    - Résultat de l'appel OU
    - Élément « Fault » pour signaler une erreur de traitement
- Généralement réalisé par HTTP (mais pas nécessairement)
  - Appel: GET/POST
  - Retour: réponse HTTP

Exemple

- Message d'appel

```
<soapenv:Envelope      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <add soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <op1 xsi:type="xsd:int">2</op1>
            <op2 xsi:type="xsd:int">5</op2>
    </add>
  </soapenv:Body>
</soapenv:Envelope>
```

- Message réponse

```
<soapenv:Envelope      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <addResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <addReturn xsi:type="xsd:int">7</addReturn>
    </addResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

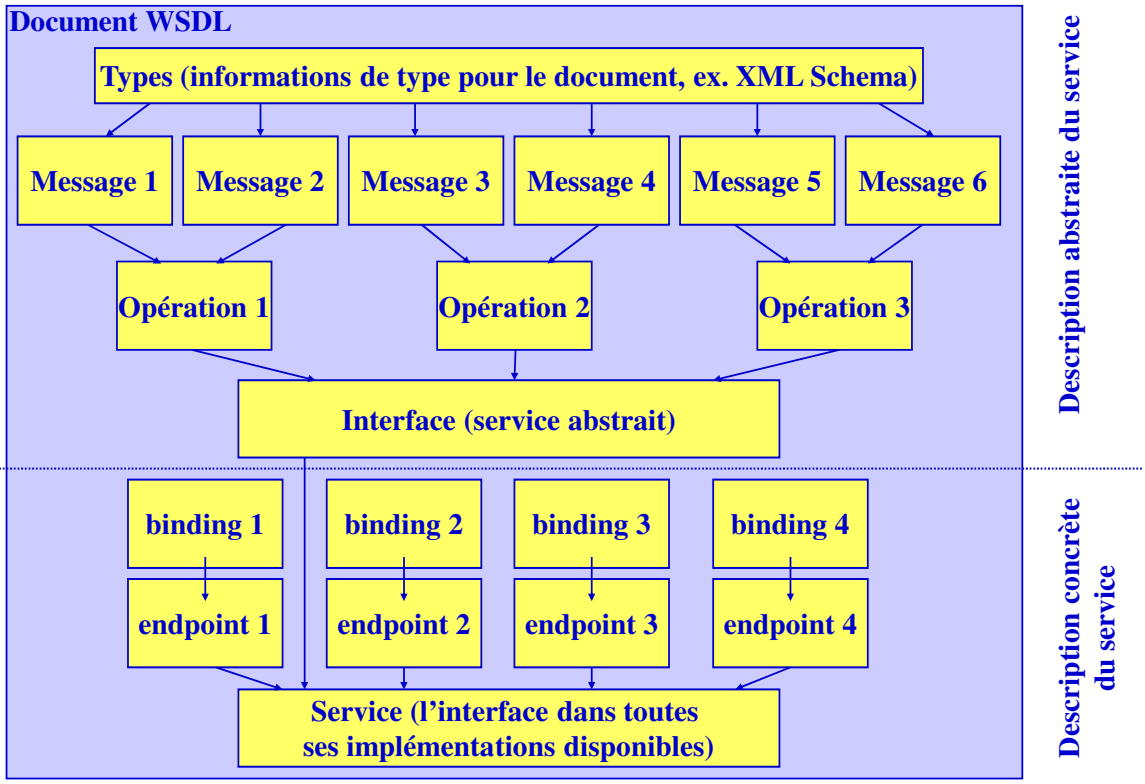

SOAP et HTTP

- « Binding » SOAP
  - Description de la façon dont les messages SOAP sont envoyés en utilisant un protocole de transport donné
  - « Binding » typique pour SOAP : HTTP
- SOAP dans HTTP: utilise GET ou POST
  - GET: l'appel n'est pas un message SOAP, seule la réponse l'est
  - POST (préféré): l'appel et la réponse sont des messages SOAP
- Inclusion dans les messages HTTP
  - Appel POST: enveloppe SOAP = contenu du message POST
  - Réponse: enveloppe SOAP = contenu de la réponse
  - SOAP utilise les mêmes codes d'erreur et d'état que HTTP → une réponse HTTP peut être directement interprétée par un module SOAP

WSDL

- Web Services Description Language
  - Description des différentes parties d'un service web
- Description à deux niveaux
  - Abstrait
    - Les types (XML Schema) des paramètres et des résultats des messages
    - Les messages manipulés dans le service
    - Les opérations individuelles: suite d'échange de messages
    - Une interface de service abstrait, qui groupe les opérations individuelles
  - Concret
    - Le « binding » de l'interface (des opérations) à un protocole de transport
    - Les points d'accès (adresses réseau) pour chaque opération
    - Service = ensemble de « bindings » avec leurs points d'accès

Éléments WSDL



Source: G. Alonso

Version courante: WSDL 2.0

- WSDL 2.0 (2007)
  - Points d'accès ("endpoints")
  - Interfaces
  - Héritage d'interfaces
  - Redéfinition d'opérations enlevée
  - Messages définis par des types
  - Opérations définies dans les interfaces
  - Points d'accès définis dans les "Bindings"
  - 8 motifs d'échange de messages (3 principaux, 5 additionnels)
  - Quelques nouveaux éléments
- WSDL 1.1 (2001)
  - Ports
  - "PortTypes"
  - Redéfinition d'opérations
  - Messages composés de parts
  - 6 éléments de premier niveau: Messages, Opérations, « PortTypes », « Bindings », Ports et Services
  - 4 primitives de transmission : « One-way », « Request-Response », « Solicit-Response », « Notification »

Exemple : service de réservation

```
<description  xmlns="http://www.w3.org/2006/01/wsdl"
   targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
   xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
   xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
   xmlns:wsoap= "http://www.w3.org/2006/01/wsdl/soap"
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
<documentation> This document describes … </documentation>
<types>
      <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
         targetNamespace="http://greath.example.com/2004/schemas/resSvc"
         xmlns="http://greath.example.com/2004/schemas/resSvc">
        <xs:element name="checkAvailability" type="tCheckAvailability"/>
        <xs:complexType name="tCheckAvailability">
          <xs:sequence>
            <xs:element name="checkInDate" type="xs:date"/>
            <xs:element name="checkOutDate" type="xs:date"/>
            <xs:element name="roomType" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
        <xs:element name="checkAvailabilityResponse" type="xs:double"/>
        <xs:element name="invalidDataError" type="xs:string"/>
      </xs:schema>
</types>
```

Exemple (suite)

```
<interface name = "reservationInterface" >
      <fault name = "invalidDataFault" element = "ghns:invalidDataError"/>
      <operation name="opCheckAvailability"
         pattern="http://www.w3.org/2006/01/wsdl/in-out"
         style="http://www.w3.org/2006/01/wsdl/style/iri">

         <input messageLabel="In" element="ghns:checkAvailability" />
         <output messageLabel="Out" element="ghns:checkAvailabilityResponse" />
         <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
      </operation>
</interface>
```

- Interface : définition abstraite du service
  - Composée d'opérations et d'erreurs
  - Peut hériter d'autres interfaces
- Opérations
  - Ensemble de messages et d'erreurs
  - Enchaînement des messages défini par un *motif* (« pattern »)
  - Style d'opération: contraintes sur l'opération et les messages
    - RPC, IRI (internationalized resource identifier), etc.

Motifs d'échange de messages

- Motifs de base
  - IN-ONLY : un seul message d'entrée, sans erreurs
  - ROBUST IN-ONLY : pareil, mais avec erreur possible
  - IN-OUT
    - Message d'entrée reçu en provenance d'un noeud N
    - Message de sortie envoyé au noeud N
    - Erreurs, qui si elles apparaissent, remplacent le message de sortie
- Motifs supplémentaires (en dehors de la norme)
  - IN-OPTIONAL-OUT : pareil, mais le message de sortie est optionnel
  - OUT-ONLY : un seul message de sortie, sans erreurs
  - ROBUST OUT-ONLY : pareil, mais avec erreur possible
  - OUT-IN
    - Message de sortie envoyé au noeud N
    - Message d'entrée reçu en provenance d'un noeud N
    - Erreurs, qui si elles apparaissent, remplacent le message d'entrée
  - OUT-OPTIONAL-IN : l'opposé de IN-OPTIONAL-OUT

Exemple (fin)

```
<binding name="reservationSOAPBinding"
   interface="tns:reservationInterface"
   type="http://www.w3.org/2006/01/wsdl/soap"
   wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">

      <fault ref="tns:invalidDataFault" wsoap:code="soap:Sender"/>
      <operation ref="tns:opCheckAvailability"
         wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
</binding>

<service name="reservationService" interface="tns:reservationInterface">
      <endpoint name="reservationEndpoint"
         binding="tns:reservationSOAPBinding"
         address ="http://greath.example.com/2004/reservation"/>
 </service>

</description>
```

- « Binding »: format des messages et détails de protocole par opération
  - Une même opération peut avoir plusieurs bindings
- Service: ensemble de points d'accès = couples (binding, adresse réseau)

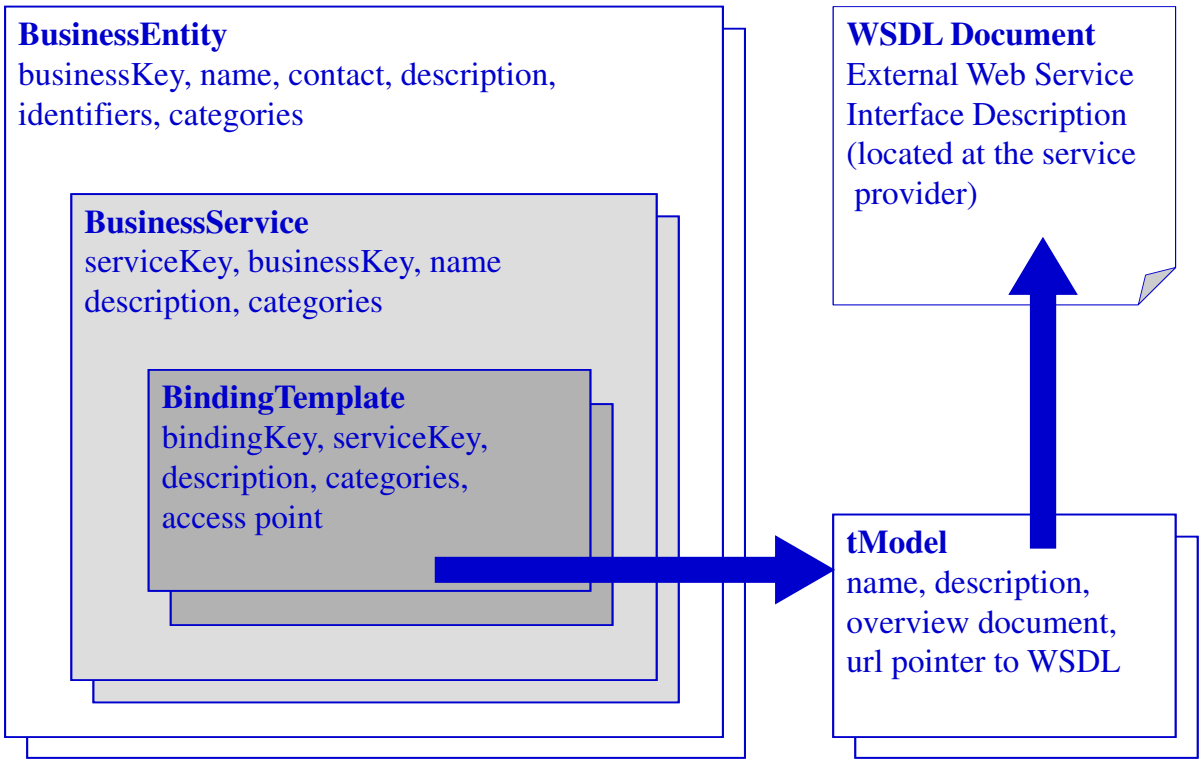
UDDI

- Universal Description, Discovery and Integration
- Historique
  - À l'origine: annuaire universel pour les services web (à la Google)
  - Aujourd'hui: vise plutôt les environnements privés, à petite échelle
  - Raisons: peu d'annuaires généraux UDDI (IBM, Microsoft, …), contenu pauvre et non fiable
    - Meilleure fiabilité en environnements contraints, privés (~EAI)
    - Élément d'infrastructure qui aide aussi à stocker des infos absentes en WSDL
- Versions
  - Version 1: les bases d'un annuaire de services
  - Version 2: adaptation à SOAP et WSDL
  - Version 3: redéfinition du rôle UDDI, accent sur les implémentations privées, sur l'interaction entre annuaires privés et publics

Modèle de données UDDI

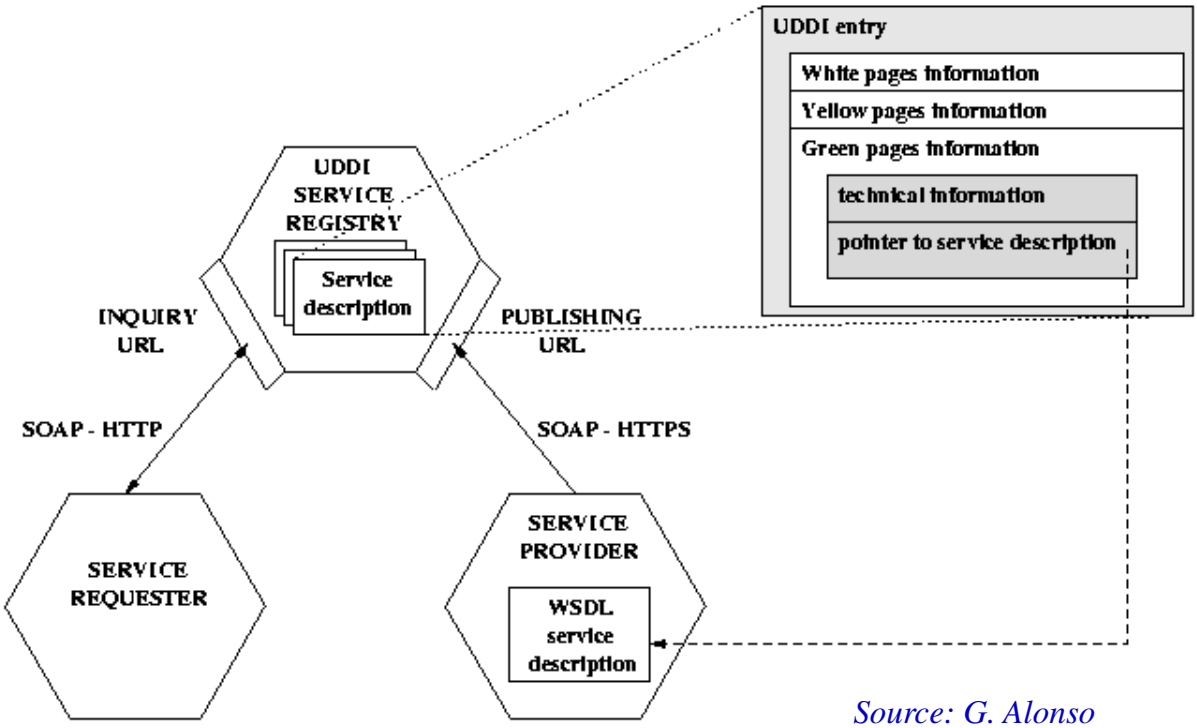
- Entrée d'annuaire UDDI = document XML composé d'éléments
  - *businessEntity*: organisation qui offre le service
  - *businessService*: liste des services web offerts par l'organisation
  - *bindingTemplate* : aspects techniques du service offert
  - *tModel*: élément générique pour info supplémentaire sur le service
- Types d'information
  - Pages blanches: données sur le fournisseur du service (nom, adresse, …)
  - Pages jaunes: classification du type de service, basée sur des standards
  - Pages vertes: info technique sur l'utilisation du service
    - Pointeurs sur les descriptions WSDL, qui ne font pas partie de l'annuaire

Schéma du modèle de données UDDI



Source: G. Alonso

UDDI, WSDL et SOAP



Source: G. Alonso

Interaction avec UDDI

- APIs pour l'accès à UDDI
  - UDDI Inquiry: rechercher des entrées UDDI dans l'annuaire (mots clés)
  - UDDI Publication: publier et modifier des entrées UDDI dans l'annuaire
  - UDDI Security: contrôle d'accès à l'annuaire
  - UDDI Subscription: souscription à des modifications d'entrées UDDI
  - UDDI Replication: dupliquer des entrées sur plusieurs nœuds
  - UDDI Custody and Ownership transfer: modifier le propriétaire d'une entrée UDDI
  - UDDI Subscription Listener: pour le client qui souscrit aux modifications
  - UDDI Value Set: pour valider l'information à publier dans l'annuaire

Services REST

- REST = Representational State Transfer
  - Pas un standard (comme SOAP), mais un style d'architecture applicative
- Architecture orientée données (ressources)
  - SOAP : architecture orientée services
- Idées
  - Directement sur HTTP (pas de surcouche comme SOAP)
  - Interface uniforme = méthodes HTTP
  - Manipulation de données/ressources identifiées par de URI
  - Services sans état : tous les informations nécessaires se trouvent dans les paramètres d'appel
  - Orienté données: actions de base sur les données/ressources (consultation, création, mise à jour, suppression)
    - SOAP: orienté service abstrait, fonctionnalités potentiellement complexes

REST vs. SOAP

- SOAP (Simple Object Access Protocol)
  - Principal standard W3C pour les services web
  - Associé à WSDL pour la description du service
  - Protocole de communication → échange de messages XML
  - Services appelables à travers des points d'accès sur le web
- REST (Representational State Transfer)
  - Appel de services web directement en HTTP
  - Messages HTTP : POST, GET, PUT, DELETE
    - Utilisation codes d'erreur, options d'appel HTTP, caching
  - Description WADL (peu utilisée)
  - Tout objet (ressource) manipulé par le service a une URI
  - Services sans état: exécution indépendante des appels précédents

Caractéristiques REST

- Quatre actions primitives de base
  - POST – pour des créations d'objets (ressources)
  - GET – pour de la consultation d'objets
  - PUT – pour des mises à jour d'objets
  - DELETE – pour des suppressions d'objets
- Tout objet créé / consulté / modifié / supprimé a une unique URI
  - Objets directement adressables sur le web
- Un objet peut avoir plusieurs représentations
  - XML, JSON, CSV, XHTML, …
  - Le format est spécifié aux HTTP HEADER (CONTENT-TYPE, ACCEPT)

REST CRUD

- En pratique: principes pas toujours respectés
  - Ceux qui les respectent → services REST CRUD (création, recherche, update, delete)
- En pratique on peut:
  - Programmer des opérations autres que CRUD
  - Associer des opérations CRUD à d'autres méthodes HTTP
  - Ne pas exposer les objets à travers des URL
- Ce qui reste:
  - On associe des opérations à des méthodes HTTP et à des URL / requêtes
  - Le type de retour peut être paramétré selon différents critères
  - Ca reste une communication HTTP

Résumé

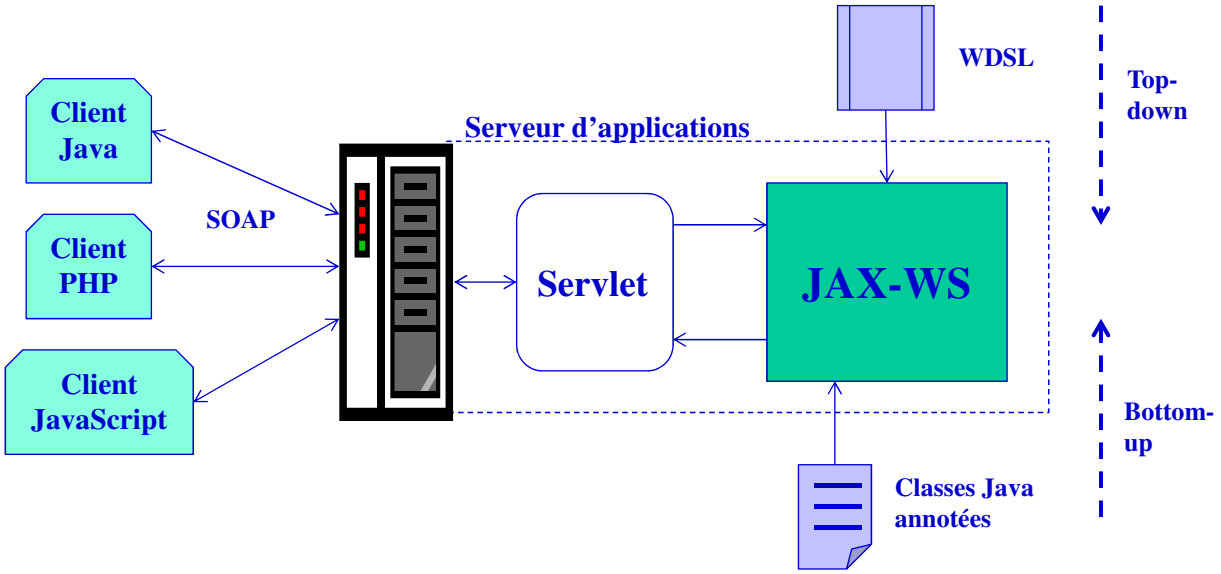
- SOAP
  - Plus évolué
  - Indépendant du protocole qui achemine les messages
  - Standards associés pour la sécurité, la fiabilité, les transactions, etc.
  - Permet aux applications d'exposer un minimum de leur fonctionnement
  - Peut garder un état au niveau du service suite aux appels
- REST
  - Simple – prise en main rapide
  - Basé directement sur HTTP, plus performant que SOAP
  - Accès uniforme aux ressources / objets
  - Limité à seulement quatre opérations
  - Expose les objets manipulés à un adressage direct sur le web
  - Plusieurs représentations des objets disponibles

Aujourd'hui: 85% des services disponibles sont REST

Services SOAP en Java avec JAX-WS

- Java API for XML Web Services (JAX-WS)
  - Standard et implémentation (Metro = implémentation de référence)
  - WSIT (Web Services Interoperability Technologies): complément pour gérer les services web avancés
  - JAXB pour marshalling/unmarshalling XML
- Intégrée à la JRE / JDK depuis la version 6. Retiré du JDK depuis la version 9.
- Idée générale d'utilisation
  - Créer des services web à partir de classes Java + annotations @WebService (création bottom-up de services web)
  - Créer des services web à partir d'une description WSDL (top-down)
  - Outils d'assemblage et déploiement des services web
  - Création de client Java de services web JAX-WS à partir du WSDL

Architecture



Génération de clients avec JAX-WS

- Utilisation de la commande **wsimport**
  - Génération classes Java pour le client
    - Classe service – donne accès au service web distant
    - Interface d'accès aux opérations du service
    - Classes implémentant les messages échangés

  Ex. **wsimport** -keep -p *packageClient URIversWSDL*

- Utilisation des classes générées dans l'application Java cliente

  Ex. Pour un service web *ExempleService* d'interface (PortType) *Exemple* offrant entre autres une opération *somme*

  - **wsimport** produit la classe service *ExempleService*, l'interface *Exemple*

```
ExempleService service = new ExempleService();
Exemple e = service.getExemplePort();
int resultat = e.somme(2, 3);
```

Services REST en Java avec JAX-RS

- Java API for REST Web Services (JAX-RS)
  - Standard et implémentation (Jersey= implémentation de référence)
  - JAXB content-handler pour XML
- Intégrée à la JRE / JDK depuis la version 6. Retiré du JDK depuis la version 9.
- Idée générale d'utilisation
  - Créer des services web à partir de classes Java + annotations @Path
    - Chaque ressource correspond à une classe Java
    - @Path est le URI (relative au URL du système) de la ressource qui est exposé à l'internet
    - Annotations pour les opérations HTTP qui sont utilisé pour gérer la ressource (e.g., @GET, @POST, etc).
  - Outils d'assemblage et déploiement des services web
  - Création de client Java de services web JAX-RS