

Les Architectures Orientées Services

SOA

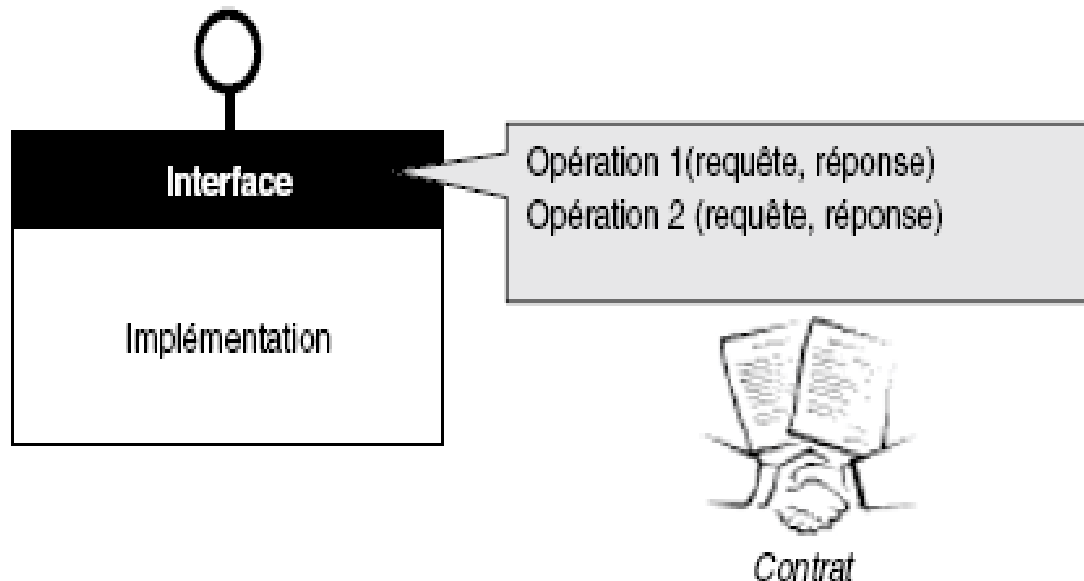
Contrat et Message de Service

Plan

- Contrat de service - Décrire
- Registre de services - Découvrir
- Message - Communiquer
- Bus de messages - Transporter

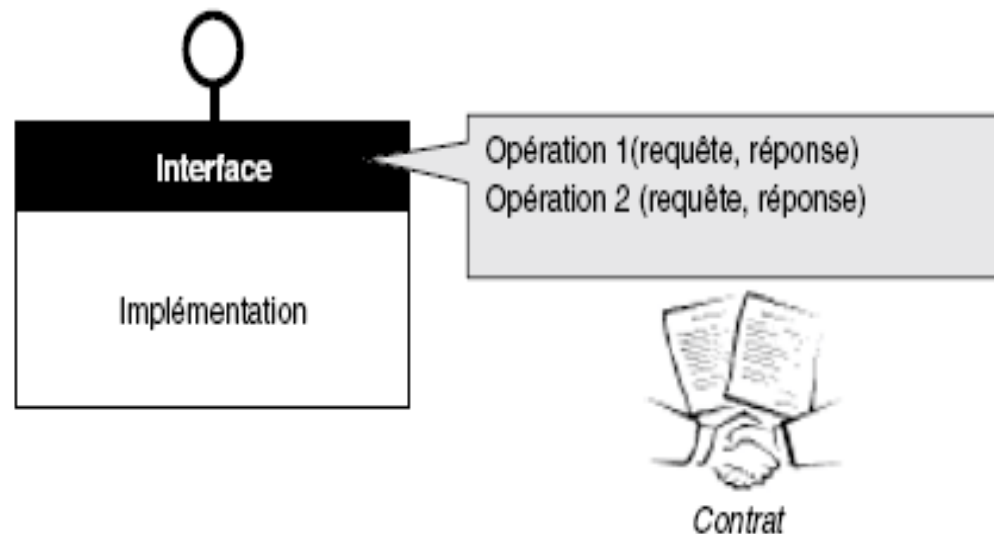
Rappel

- Le concept de **service** est la pierre angulaire de la démarche SOA.
- Le service est localisé sur une **ressource physique (adresse)** donnée, déployé sur un serveur.
- Chaque service a un **contrat de service** qui décrit son fonctionnement aux consommateurs.



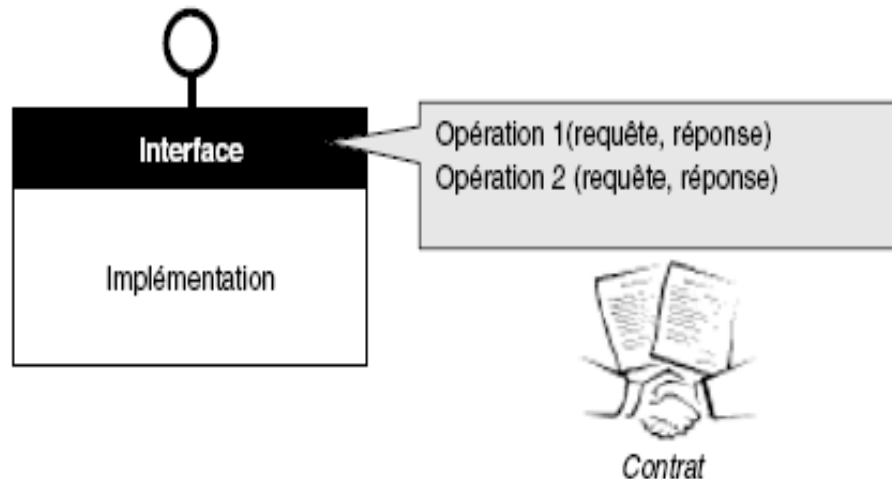
Contrat de service

- Le contrat liste les **opérations** offertes par le service et, pour chaque opération les paramètres et la sortie: **Signature** de l'opération
- Le contrat précise la **structure et le format** de chaque information échangée.
- Pour obtenir un service, de même que pour offrir un service, consommateur et fournisseur du service devront respecter ce contrat.



Contrat de service

- Un contrat doit garantir une **qualité de service** à ses consommateurs et savoir répondre à leurs exigences.
- Un fournisseur peut exiger en contre partie que le client respecte certaines contraintes et obligations d'utilisation. (**directives**)
- Un même contrat peut être associé à plusieurs implémentations. (Ex: version public et gratuite et version payante pour abonnés).



Contrat de Service

Un contrat de service satisfait les critères suivants :

- **Formalisé** : le contrat est défini dans un langage formalisé, WSDL par exemple.
- **Publié** : le contrat est publié, accessible et compréhensible par les consommateurs.
- **Concentré** sur une mission fonctionnelle et une seule.
- **Étanche** : le contrat ne fait pas référence à l'implémentation du service. Il est en ce sens une boîte noire.
- **Explicite** sur la Qualité de Service.

Contrat de Service

➔ Faible Couplage

Permet coté Fournisseur :

- Les modifications de l'implémentation de son service (par exemple pour suivre les réglementations, optimiser la performance, etc.)
- Une plus grande diversité de consommateurs (tous les cas de dialogues étant explicitement décrits dans le contrat);
- La garantie de la qualité de son service (respect des conditions d'utilisations, suivi de l'utilisation, etc.);

Contrat de Service

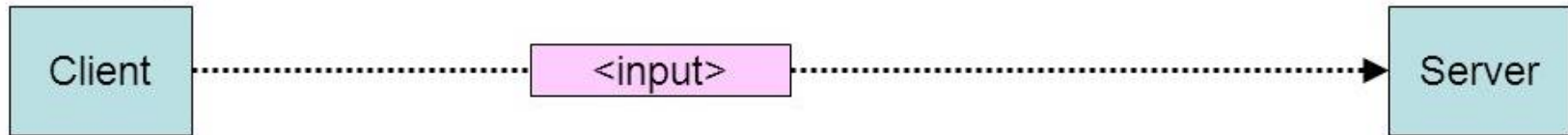
➔ Faible Couplage

Permet coté Consommateur :

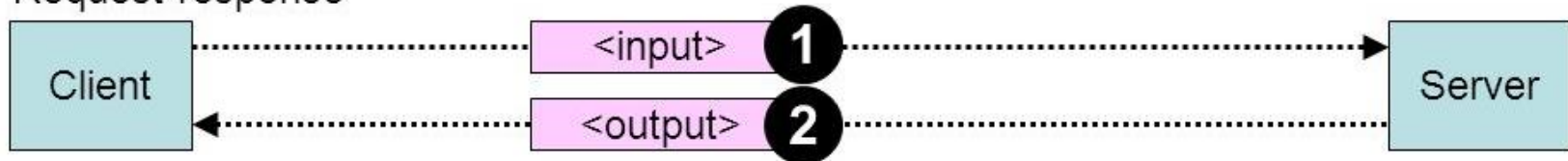
- Une facilité de développement (pas besoin de connaître l'implémentation);
- Une fiabilité de transaction (le contrat est explicite, il n'y a donc pas de surprise) ;
- Un changement possible de fournisseur (en cas de non satisfaction), si le fournisseur alternatif accepte de respecter le même contrat.

Types d'opération - Message patterns

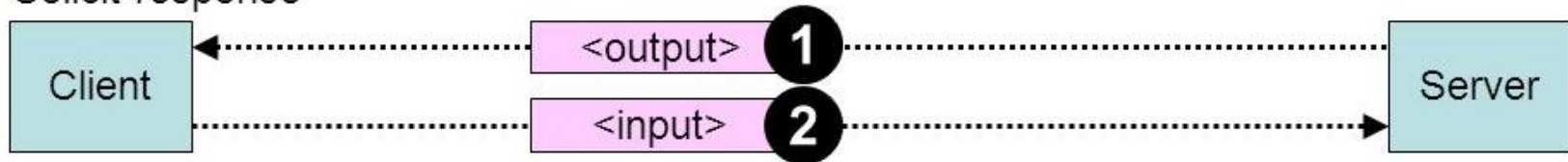
One-way



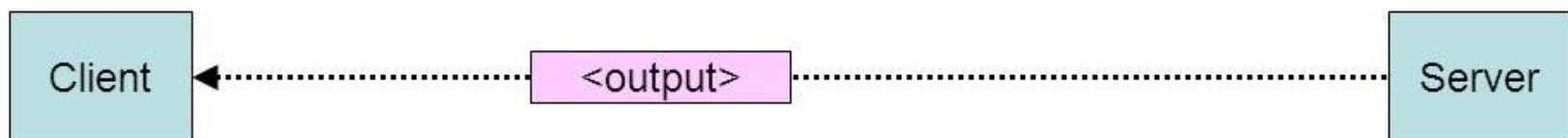
Request-response



Solicit-response



Notification



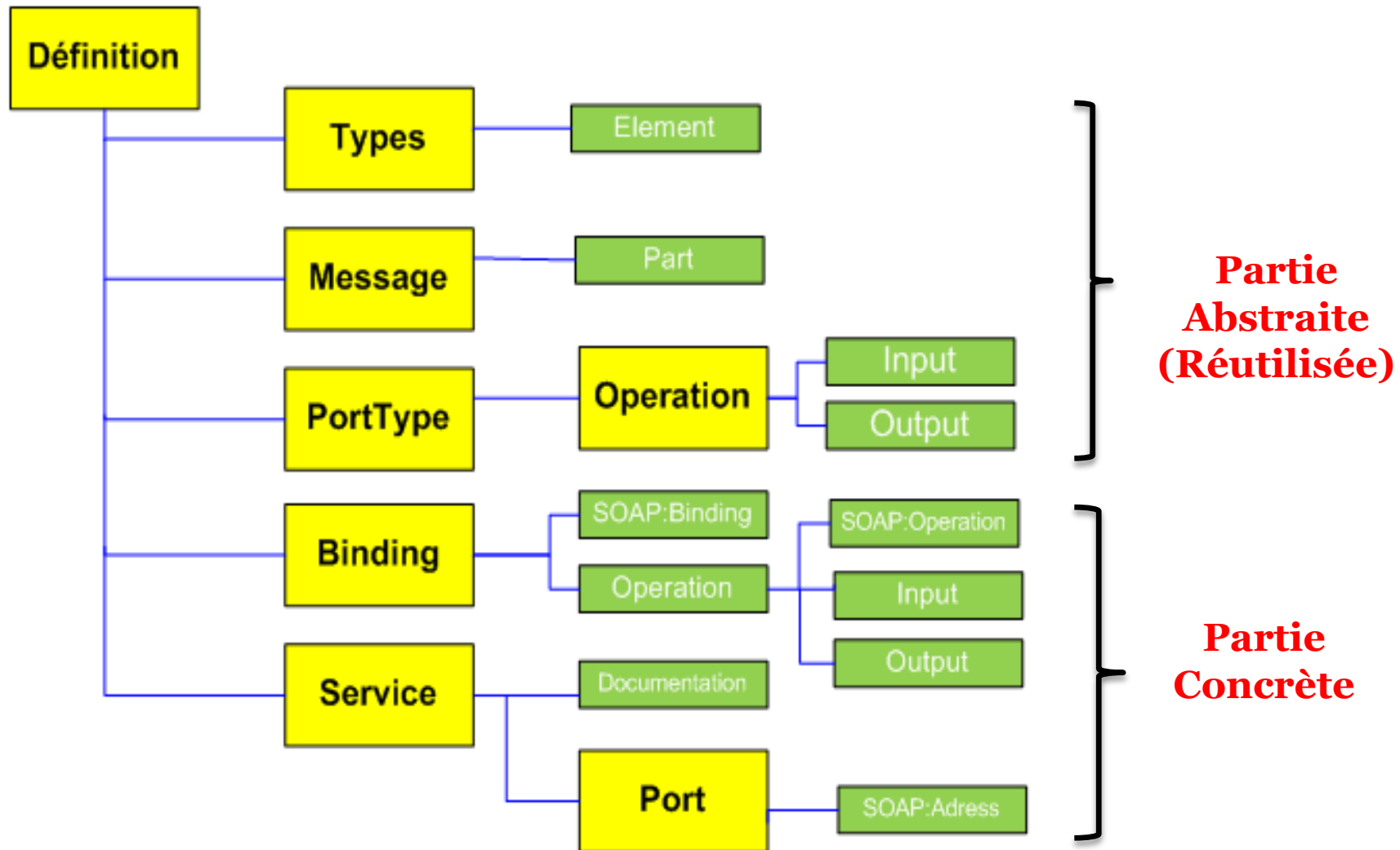
Contrat de service - Exemple

Document WSDL dans les Services Web

- WSDL pour *Web Service Description Language*.
- Est un langage normalisé basé XML pour décrire le mode de fonctionnement d'un Web Service en cachant ses détails d'implémentation.
- Il permet ainsi de décrire les modalités d'invocation distante d'un Web Service, en particulier :
 - Les opérations possibles au sein du service.
 - Les paramètres d'entrée et sortie de ces opérations.
 - Le typage de ces paramètres.
 - Les points d'entrée (URL) des opérations.

Contrat de service - Exemple

WSDL : Structure



Contrat de service - Exemple

WSDL : <types>

- L'élément <types> contient la définition des types utilisés pour décrire la structure des messages échangés par le service web.
- Le système de typage est généralement un **fichier SchemaXML - XSD** accessible au même titre que le document WSDL.
- Cette séparation permet :
 - ✓ de réutiliser des types dans plusieurs WSDL différents
 - ✓ d'éviter d'alourdir le document WSDL
- Pouvant contenir des types simples (String, double) et complexes (Person, Client) .

Contrat de service - Exemple

WSDL : <types>

Exemple : Définition des types pour HelloWorld web service – WSDL et XSD

Contrat de service - Exemple

WSDL : <message>

- L'élément <message> permet de décrire les messages échangés par les services
 - ✓ Paramètres d'entrées et de sorties des opérations
 - ✓ Exception
- Chaque <message> est identifié par un nom (attribut name) et est constitué d'un ensemble d'éléments <part>.
- En quelque sorte un élément <part> correspond à un paramètre d'une opération.
- L'élément <part> est défini par : un nom (attribut name) et un type (attribut type)

Contrat de service - Exemple

WSDL : <portType> et <operation>

- Un élément <portType> est un regroupement d'opérations et peut être comparé à une interface Java.
- Caractéristiques d'un élément <portType>
 - ✓ Identifiable par un nom (attribut name)
 - ✓ Composé de sous élément <operation>
- Une opération est comparable à une méthode Java
 - ✓ Identifiable par un nom (attribut name)
- Une opération exploite les messages via les sous éléments
 - ✓ <input> : message transmis au service
 - ✓ <output> : message produit par le service
 - ✓ <fault> : message d'erreur (très proche des exceptions)

Contrat de service - Exemple

WSDL : <binding>

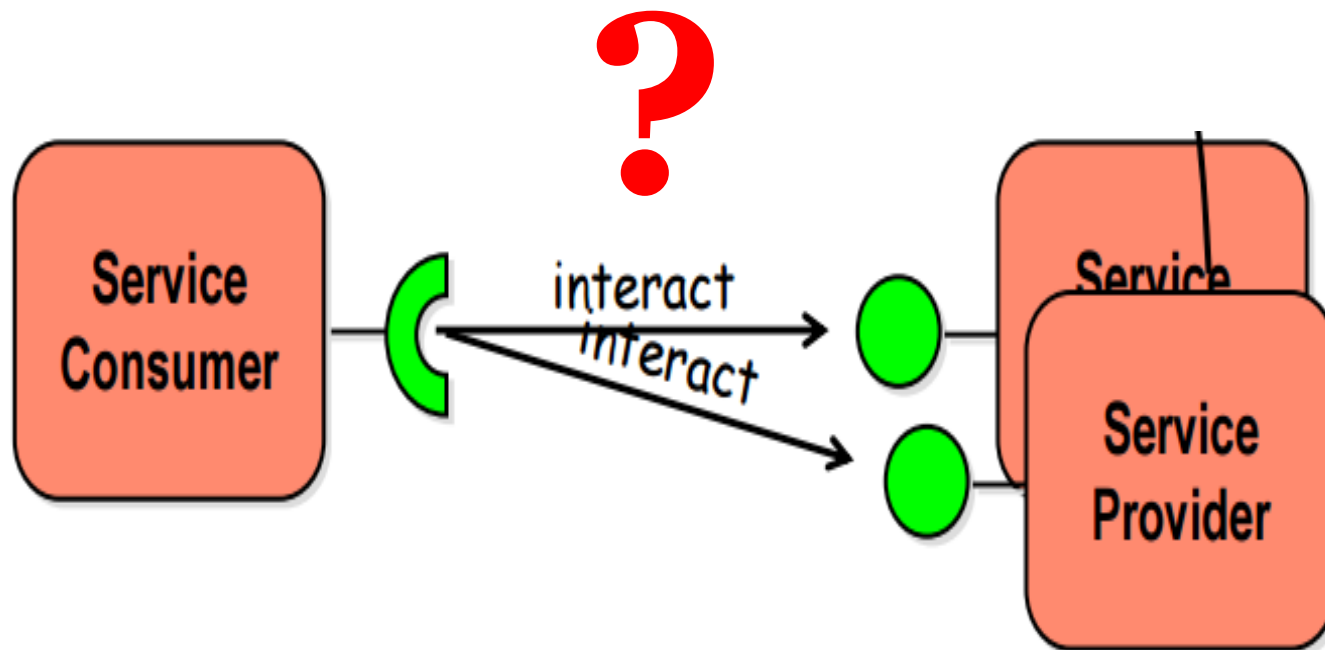
- Un élément <binding> permet de réaliser la partie concrète d'un élément <portType> :
 - un nom (attribut name)
 - un portType (attribut type)
- Il décrit précisément le protocole à utiliser pour manipuler un élément <portType>:
 - SOAP 1.1 et 1.2
 - HTTP GET & Post (pour le transfert d'images par exemple)
 - MIME
- Plusieurs éléments <binding> peuvent être définis de sorte qu'un élément portType peut être appelé de différentes manières.
- La structure de l'élément <binding> dépend du protocole utilisé.

Contrat de service - Exemple

WSDL : <service> et <port>

- Un élément service définit l'ensemble des points d'entrée du service web, en regroupant des éléments <port>.
- L'élément <port> permet de spécifier une adresse pour un binding donné.
- Un port est défini par deux attributs:
 - ✓ name : nom du port
 - ✓ binding : nom du binding (défini précédemment)
- Le corps de l'élément <port> est spécifique au protocole utilisé pour définir le binding.
- Dans le cas d'un binding de type SOAP, un élément <soap:address> précise l'URI du port. Il est par conséquent possible d'appeler un service à des endroits différents (plusieurs éléments port).

Comment découvrir/rechercher un service ?

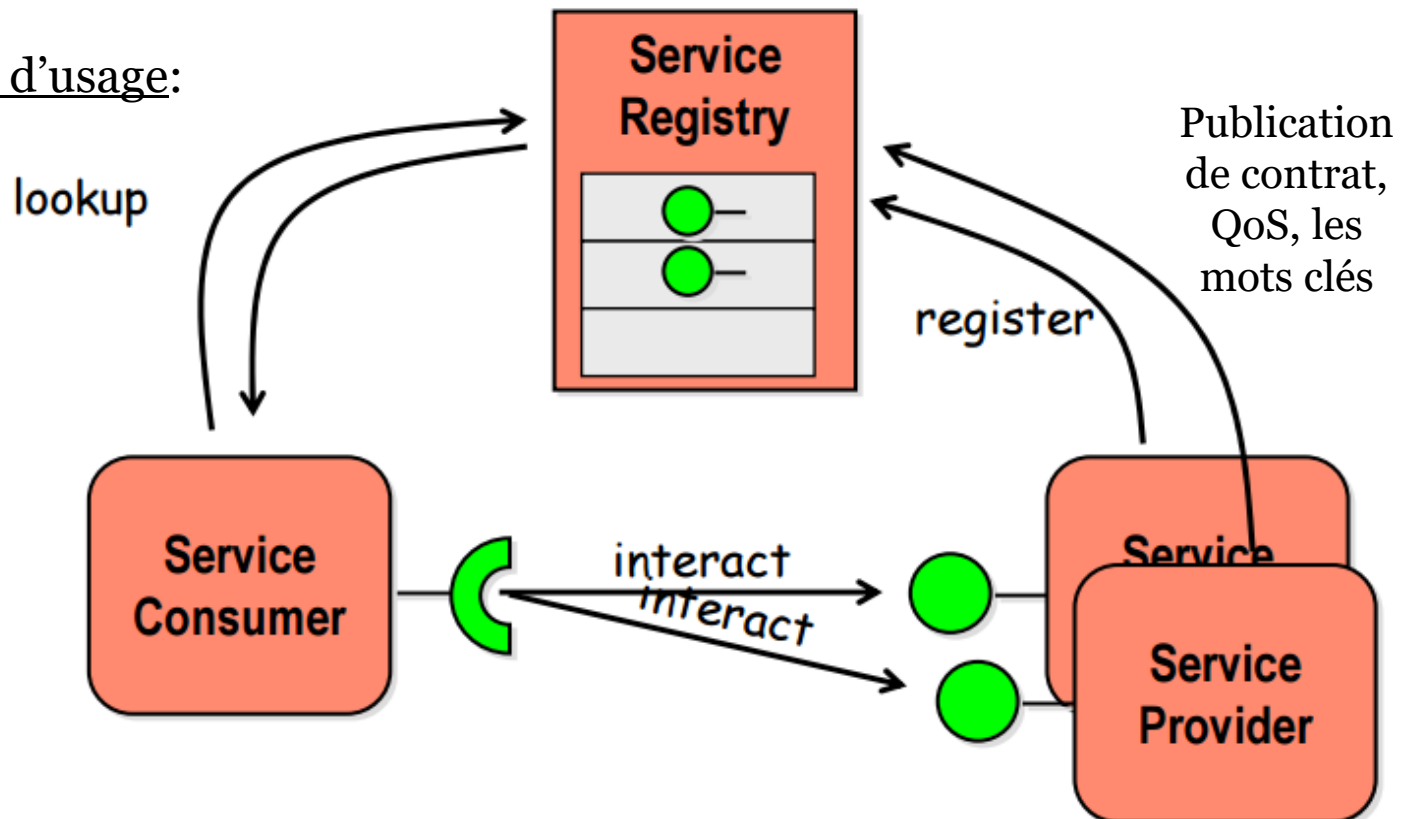


Registre de services

Registre (annuaire ou registry) de services

Les développeurs d'applications consommatrices ont besoin d'accéder aux contrats décrivant les services pour en prendre connaissance.

Cas d'usage:



Registre de services

Registre de services

Les développeurs d'applications consommatrices ont besoin d'accéder aux contrats décrivant les services pour en prendre connaissance.

Cas d'usage :

- ☐ Le fournisseur **publie** la description de son service dans le registre (contrat, QoS, les mots clés qui en faciliteront la recherche, etc);
- ☐ Le consommateur **recherche/découvre** un service correspondant à son niveau d'exigences auprès du registre;
- ☐ Le registre transmet au consommateur la ou les descriptions correspondant à sa recherche.

Registre de services

Registre de services

Les principales fonctions à attendre d'un tel registre sont donc :

- La publication des contrats
- La classification des services
 - ❑ Documentation pour les consommateurs.
 - ❑ Indexation sur le plan technique (messages, interfaces, protocoles acceptables, QoS, etc.).
 - ❑ Indexation sur le plan métier (domaine sectoriel, couverture géographique, tarification, réglementation, etc.).

Registre de services

Registre de services

Les principales fonctions à attendre d'un tel registre sont donc :

- La gestion des services. (repository)
 - ☐ Stockage des implémentations.
 - ☐ Gestion des modifications, des versions, des variantes
 - ☐ Gestion des dépendances entre services.

- La recherche d'un service
 - ☐ Recherche interactive.
 - ☐ Recherche programmatique (requêtes de sélection).

Registre de services - Exemple

UDDI

- UDDI pour *Universal Description, Discovery, and Integration*.
- Un système d'annuaire qui permet à un fournisseur de décrire son Service, puis au client de le découvrir et de s'y connecter.
- UDDI est à l'origine une norme du W3C écrite en 2000. L'OASIS a repris sa gestion et a sorti la version 3 en 2004.
- Un annuaire UDDI est construit sur la base des normes WSDL et SOAP. Basé XML.
- Les services y sont décrits en WSDL et on accède à l'annuaire via des requêtes SOAP.

Registre de services - Exemple

UDDI

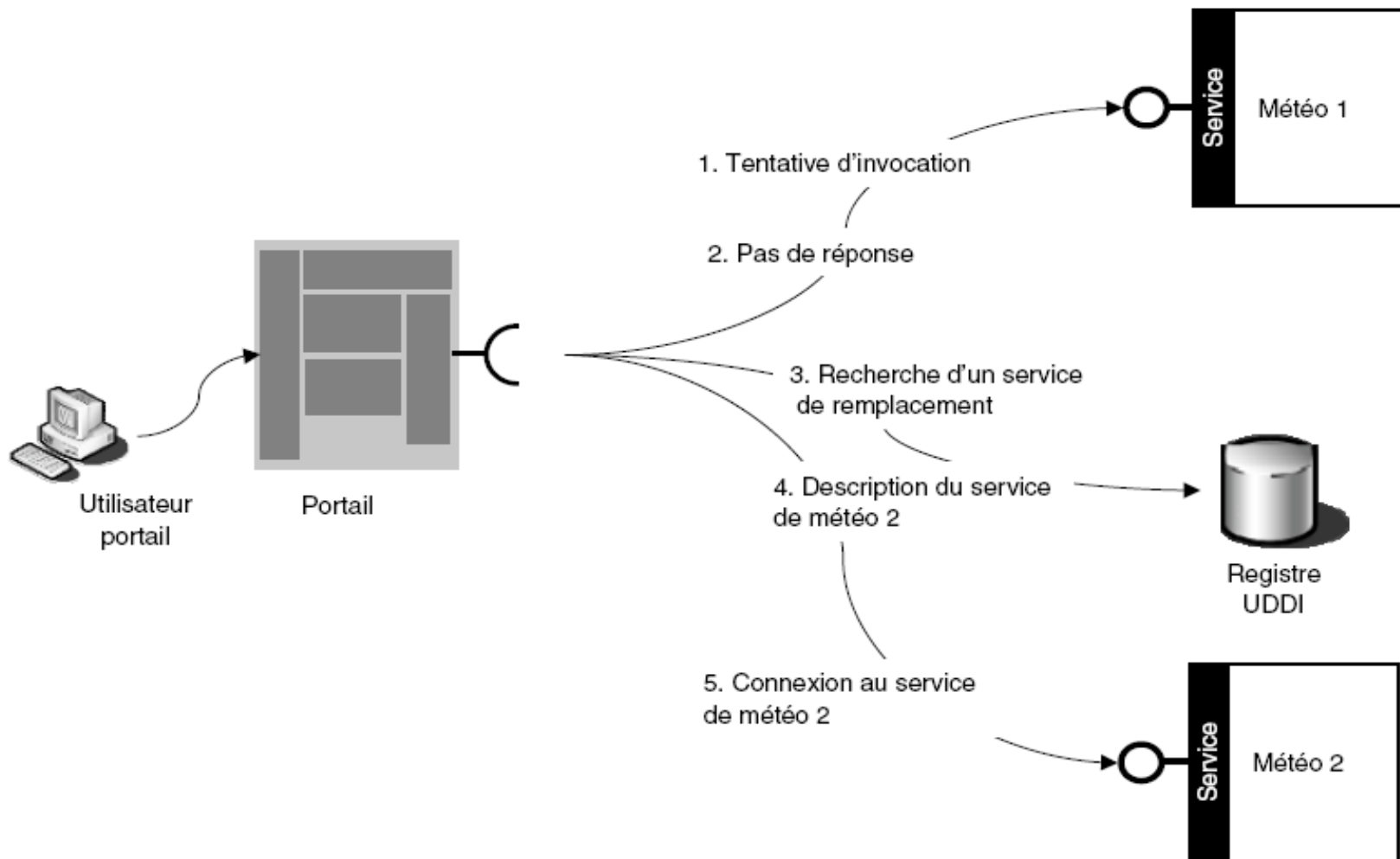
On y trouve trois types d'informations :

1. Les Pages Blanches, qui décrivent les fournisseurs de services (pour rechercher un service via contact, nom, adresse, etc.).
2. Les Pages Jaunes, qui décrivent les catégories de services/fournisseurs.
3. Les Pages Vertes, qui décrivent les contrats WSDL (caractéristiques techniques d'un service).

UDDI complète donc le contrat WSDL par des métadonnées sur le fournisseur du service.

Registre de services - Exemple

UDDI a été conçu pour permettre la découverte et l'**intégration automatique** d'un service.



Registre de services - Exemple

UDDI

On distingue ainsi trois types de registres :

- **Registre d'entreprise** : ces annuaires sont situés sur un réseau privé, et servent de référentiels des services disponibles en interne. C'est aujourd'hui le principal usage des annuaires UDDI.
- **Registre fédéré avec des partenaires** : ces annuaires peuvent avoir une gestion déléguée à un tiers ou bien être synchronisés avec d'autres référentiels.
- **Registre public** : ces annuaires devraient théoriquement être liés les uns aux autres à la manière de DNS afin de constituer un registre mondial des services invocables depuis Internet. Dans la pratique, ils sont inexistant : seuls quelques annuaires de tests sont proposés par Microsoft et IBM.

Registre de services - Exemple

UDDI

- La mise en œuvre d'un annuaire publique de services à l'échelle d'Internet n'a donc pas été atteint.
- Services payants ➔ Problème dans l'automatisation.
- Il n'est pas envisageable de se connecter à un prestataire dont on ignore tout.
- Dans le contexte d'un système d'information s'orientant vers une architecture SOA, il est préférable de mettre en œuvre un registre interne des services.
- Ce registre constitue un référentiel d'entreprise au même titre qu'une base de clients.
- Il permet de décrire et localiser les services avec leurs différentes versions.

Registre de services - Exemple

UDDI

Aspects techniques :

- Le fournisseur de service publie son contrat WSDL dans l'annuaire UDDI en l'associant à la description de la société et à la catégorie de service concernée.
- Chaque service de l'annuaire est identifié par une clef unique, sorte de clé primaire du registre.
- Il est possible de signer électroniquement un contrat WSDL lors de sa publication afin de certifier qu'il émane bien d'une société connue et non d'un pirate malveillant.

Registre de services - Exemple

UDDI

Aspects techniques :

- Il inclut un schéma XML qui décrit les structures de données suivantes :
 - businessEntity
 - businessService
 - bindingTemplate
 - tModel
 - publisherAssertion

Registre de services - Exemple

UDDI

businessEntity Data Structure : Représente le fournisseur de WS.

Exemple :

Registre de services - Exemple

```
<businessEntity businessKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40"  
  operator="http://www.ibm.com" authorizedName="John Doe">  
  <name>Acme Company</name>  
  <description>  
    We create cool Web services  
  </description>  
  
  <contacts>  
    <contact useType="general info">  
      <description>General Information</description>  
      <personName>John Doe</personName>  
      <phone>(123) 123-1234</phone>  
      <email>jdoe@acme.com</email>  
    </contact>  
  </contacts>  
  
  <businessServices>  
    ...  
  </businessServices>  
  <identifierBag>  
    <keyedReference tModelKey="UUID:8609C81E-EE1F-4D5A-B202-3EB13AD01823" name="D-U-N-S" value="123456789" />  
  </identifierBag>  
  
  <categoryBag>  
    <keyedReference tModelKey="UUID:C089FE13-179F-413D-8A5B-5004DB8E58B2" name="NAICS" value="111336" />  
  </categoryBag>  
</businessEntity>
```

Registre de services - Exemple

UDDI

businessService Data Structure : Représente un service Web individuel fourni par une businessEntity.

Exemple :

```
<businessService serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
  businessKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">

  <name>Hello World Web Service</name>
  <description>A friendly Web service</description>
  <bindingTemplates>
    ...
  </bindingTemplates>
  <categoryBag />
</businessService>
```


Registre de services - Exemple

UDDI

bindingTemplate Data Structures : Sont les descriptions techniques des services web représentés par une structure businessService. Un seul businessService peut avoir plusieurs bindingTemplate. Le bindingTemplate représente l'implémentation réelle du service Web.

Exemple :

```
<bindingTemplate serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
  bindingKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">

  <description>Hello World SOAP Binding</description>
  <accessPoint URLType="http">http://localhost:8080</accessPoint>
  <tModelInstanceDetails>
    <tModelInstanceInfo tModelKey="uuid:EB1B645F-CF2F-491f-811A-4868705F5904">
      <instanceDetails>
        <overviewDoc>
          <description>
            references the description of the WSDL service definition
          </description>

          <overviewURL>
            http://localhost/helloworld.wsdl
          </overviewURL>
        </overviewDoc>
      </instanceDetails>
    </tModelInstanceInfo>
  </tModelInstanceDetails>
</bindingTemplate>
```

Registre de services - Exemple

UDDI

tModel Data Structures : est le moyen de décrire les différentes structures de business, services, et templates stockées dans le registre UDDI.

Tout concept abstrait peut être enregistré dans l'UDDI comme un tModel.

Exemple :

Registre de services - Exemple

UDDI

Exemple : si on définit un nouveau type de port WSDL, on peut définir un tModel qui représente ce type de port dans UDDI. Puis, l'associer à un service.

```
<tModel tModelKey="uuid:xyz987..." operator="http://www.ibm.com"   authorizedName="Jc
  <name>HelloWorldInterface Port Type</name>
  <description>
    An interface for a friendly Web service
  </description>

  <overviewDoc>
    <overviewURL>
      http://localhost/helloworld.wsdl
    </overviewURL>
  </overviewDoc>
</tModel>
```

Registre de services - Exemple

UDDI

publisherAssertion Data Structures : Il s'agit d'une structure relationnelle mettant en association deux ou plusieurs structures businessEntity selon un type spécifique de relation, comme une filiale ou un département.

Exemple :

```
<element name="publisherAssertion" type="uddi:publisherAssertion" />
<complexType name="publisherAssertion">
  <sequence>
    <element ref="uddi:fromKey" />
    <element ref="uddi:toKey" />
    <element ref="uddi:keyedReference" />
  </sequence>
</complexType>
```

Registre de services - Exemple

UDDI

Aspects techniques :

- Comment y accéder ?
- ✓ Deux interfaces pour les consommateurs de services et les fournisseurs de services d'interagir avec le registre.
- ✓ Les consommateurs utilisent Inquiry Interface pour rechercher un service ;
- ✓ Et les fournisseurs utilisent Publisher Interface pour ajouter et répertorier un service.

Registre de services - Exemple

UDDI

Aspects techniques :

- ✓ Les fournisseurs utilisent Publisher Interface pour ajouter et répertorier un service.

16 opérations :

- save_business / delete_business
- save_service / delete_service
- save_binding / delete_binding
- Etc.

Registre de services - Exemple

UDDI

Aspects techniques :

- ✓ Les consommateurs utilisent Inquiry Interface pour rechercher un service.

10 opérations :

- find_business
- get_serviceDetail
- find_binding
- Etc.

Registre de services - Exemple

UDDI

Implémentations :

- ✓ Java : jUDDI, UDDI4J (UDDI for Java)
- ✓ Python : UDDI4Py
- ✓ Perl : UDDI::Lite
- ✓ Ruby : UDDI4r
- ✓ Etc.

Registre de services - Exemple

Cas d'usage - Publication d'un Service Web avec jUDDI:

jUDDI Console (Beta)

jUDDI API (proprietary)

[get_registryInfo](#)

[find_publisher](#)

[get_publisherDetail](#)

[save_publisher](#) [delete_publisher](#)

UDDI Inquiry API

[find_business](#)

[find_service](#)

[find_binding](#)

[find_tModel](#)

[find_relatedBusinesses](#)

[get_businessDetail](#)

[get_businessDetailExt](#)

[get_serviceDetail](#)

[get_bindingDetail](#)

[get_tModelDetail](#)

UDDI Publish API

[get_authToken](#)

[get_registeredInfo](#)

[discard_authToken](#)

[save_business](#)

[save_service](#)

[save_binding](#)

[save_tModel](#)

[delete_business](#)

[delete_service](#)

<http://localhost:8080/juddi/console/>

Registre de services - Exemple

Cas d'usage - Publication d'un Service Web avec jUDDDI:

Etape 1 : Authentification - authToken

jUDDI Console (Beta)

get_authToken

The [get_authToken](#) API call is used to obtain an authentication token ([authToken](#)). Authentication tokens are opaque values that are required for all other publisher API calls. If an error occurs while processing this API call, a [dispositionReport](#) element will be returned to the caller within a [SOAP Fault](#) containing information about the [error](#) that was encountered.

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <get_authToken generic="2.0" xmlns="urn:uddi-org:api_v2"
      userID="camelia"
      cred="****"/>
  </soapenv:Body>
</soapenv:Envelope>
```

Time: 1223 milliseconds

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmlns:soapenv="http://
  <soapenv:Body>
    <authToken generic="2.0" operator="jUDDI-org" xmlns="urn:uddi-org:api_v2">
      <authInfo>authToken:9DF6FC10-3AF6-11E0-AD97-FC8CC7E394E4</authInfo>
    </authToken>
  </soapenv:Body>
</soapenv:Envelope>
```

jUDDI API (proprietary)

[get_registryInfo](#)
[find_publisher](#)
[get_publisherDetail](#)
[save_publisher](#)
[delete_publisher](#)

UDDI Inquiry API

[find_business](#)
[find_service](#)
[find_binding](#)
[find_tModel](#)
[find_relatedBusinesses](#)
[get_businessDetail](#)
[get_businessDetailExt](#)
[get_serviceDetail](#)
[get_bindingDetail](#)
[get_tModelDetail](#)

UDDI Publish API

[get_authToken](#)
[get_registeredInfo](#)
[discard_authToken](#)
[save_business](#)
[save_service](#)
[save_binding](#)
[save_tModel](#)
[delete_business](#)
[delete_service](#)
[delete_binding](#)
[delete_tModel](#)
[add_publisherAssertions](#)
[set_publisherAssertions](#)

Registre de services - Exemple

Cas d'usage - Publication d'un Service Web avec jUDDDI:

Etape 2 : save ou update des informations sur un élément **businessEntity**

jUDDI Console (Beta)

save_business

The [save_business](#) API call is used to save or update information about a complete [businessEntity](#) element. If an error occurs while processing this API call, a [dispositionReport](#) element will be returned to the caller within a [SOAP Fault](#) containing information about the [error](#) that was encountered.

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  <soapenv:Body>
    <save_business generic="2.0" xmlns="urn:uddi-org:api_v2">
      <authInfo>authToken:9DF6FC10-3AF6-11E0-AD97-FC8CC7E394E4</authInfo>
      <businessEntity businessKey="">
        <name>camben</name>
        <description>camelia benchegroun</description>
        <contacts>
          <contact useType="***">
            <personName>Camelia</personName>
            <phone>0614409383</phone>
            <email>cam@gmail.com</email>
          </contact>
        </contacts>
      </businessEntity>
    </save_business>
  </soapenv:Body>
</soapenv:Envelope>
```

Validate Submit Reset

Time: 900 milliseconds

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmlns:soapenv="http://
  <soapenv:Body>
    <businessDetail generic="2.0" operator="jUDDI-org" xmlns="urn:uddi-org:api_v
    <businessEntity authorizedName="ben" businessKey="44C1A770-3AF7-11E0-AD97-0
      <discoveryURLs>
        <discoveryURL useType="businessEntity">http://localhost:8080/juddi/uddig
      </discoveryURLs>
      <name>camben</name>
      <description>camelia benchegroun</description>
```

jUDDI API (proprietary)

[get_registryInfo](#)
[find_publisher](#)
[get_publisherDetail](#)
[save_publisher](#)
[delete_publisher](#)

UDDI Inquiry API

[find_business](#)
[find_service](#)
[find_binding](#)
[find_tModel](#)
[find_relatedBusinesses](#)
[get_businessDetail](#)
[get_businessDetailExt](#)
[get_serviceDetail](#)
[get_bindingDetail](#)
[get_tModelDetail](#)

UDDI Publish API

[get_authToken](#)
[get_registeredInfo](#)
[discard_authToken](#)
[save_business](#)
[save_service](#)
[save_binding](#)
[save_tModel](#)
[delete_business](#)
[delete_service](#)
[delete_binding](#)
[delete_tModel](#)

Registre de services - Exemple

Cas d'usage - Publication d'un Service Web avec jUDDDI:

Etape 3 : add ou update des éléments tModel

jUDDI Console (Beta)

save_tModel

The [save_tModel](#) API call is used to add or update one or more registered [tModel](#) elements. If an error occurs while processing this API call, a [dispositionReport](#) element will be returned to the caller within a [SOAP Fault](#) containing information about the [error](#) that was encountered.

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <save_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
      <authInfo authToken="9DF6FC10-3AF6-11E0-AD97-FC8CC7E394E4"/>
      <tModel tModelKey="">
        <name>http://localhost:8080/axis/StockQuoteService.jws?wsdl</name>
        <description>***</description>
        <overviewDoc>
          <description>***</description>
          <overviewURL>http://localhost:8080/axis/StockQuoteService.jws?wsd
        </overviewDoc>
        <identifierBag>
          <keyedReference tModelKey="UUID:C1ACF26D-9672-4404-9D70-39B756E62"
        </identifierBag>
      </tModel>
    </save_tModel>
  </soapenv:Body>
</soapenv:Envelope>
```

Validate

Submit

Reset

Time: 52 milliseconds

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmlns:soapenv="http://
<soapenv:Body>
  <tModelDetail generic="2.0" operator="jUDDI-org" xmlns="urn:uddi-org:api_v2">
    <tModel authorizedName="ben" operator="jUDDI-org" tModelKey="uuid:BC4E5C20-
    <name>http://localhost:8080/axis/StockQuoteService.jws?wsdl</name>
    <description>***</description>
    <overviewDoc>
      <description>***</description>
      <overviewURL>http://localhost:8080/axis/StockQuoteService.jws?wsdl</overv
```

jUDDI API (proprietary)

[get registryInfo](#)

[find_publisher](#)

[get_publisherDetail](#)

[save_publisher](#)

[delete_publisher](#)

UDDI Inquiry API

[find_business](#)

[find_service](#)

[find_binding](#)

[find_tModel](#)

[find_relatedBusinesses](#)

[get_businessDetail](#)

[get_businessDetailExt](#)

[get_serviceDetail](#)

[get_bindingDetail](#)

[get_tModelDetail](#)

UDDI Publish API

[get_authToken](#)

[get_registeredInfo](#)

[discard_authToken](#)

[save_business](#)

[save_service](#)

[save_binding](#)

[save_tModel](#)

[delete_business](#)

[delete_service](#)

[delete_binding](#)

[delete_tModel](#)

Registre de services - Exemple

Cas d'usage - Publication d'un Service Web avec jUDDDI:

Etape 4 : add ou update des éléments **BusinessService**

jUDDI Console (Beta)

save_service

The [save_service](#) API call is used to add or update one or more [businessService](#) elements. If an error occurs while processing this API call, a [dispositionReport](#) element will be returned to the caller within a [SOAP Fault](#) containing information about the [error](#) that was encountered.

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <save_service generic="2.0" xmlns="urn:uddi-org:api_v2">
      <authInfo>authToken:9DF6FC10-3AF6-11E0-AD97-FC8CC7E394E4</authInfo>
      <businessService businessKey="44C1A770-3AF7-11E0-AD97-CE8242B9869F">
        <name>Sommer</name>
        <description>Sommer a et b</description>
        <bindingTemplates>
          <bindingTemplate bindingKey="">
            <accessPoint URLType="http">http://localhost:8080/axis/sommer.j</accessPoint>
            <tModelInstanceDetails>
              <tModelInstanceInfo tModelKey="uuid:BC4E5C20-3AF7-11E0-AD97-F</tModelInstanceInfo>
              <instanceDetails>
                <overviewDoc>
```

Validate Submit Reset

Time: 156 milliseconds

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmlns:soapenv="http://
<soapenv:Body>
  <serviceDetail generic="2.0" operator="jUDDI-org" xmlns="urn:uddi-org:api_v2">
    <businessService businessKey="44C1A770-3AF7-11E0-AD97-CE8242B9869F">
      <name>Sommer</name>
      <description>Sommer a et b</description>
      <bindingTemplates>
        <bindingTemplate bindingKey="2DDB92E0-3AF8-11E0-AD97-91D36E9030F4">
          <accessPoint URLType="http">http://localhost:8080/axis/sommer.jws</accessPoint>
```

jUDDI API (proprietary)

[get registryInfo](#)
[find publisher](#)
[get publisherDetail](#)
[save publisher](#)
[delete publisher](#)

UDDI Inquiry API

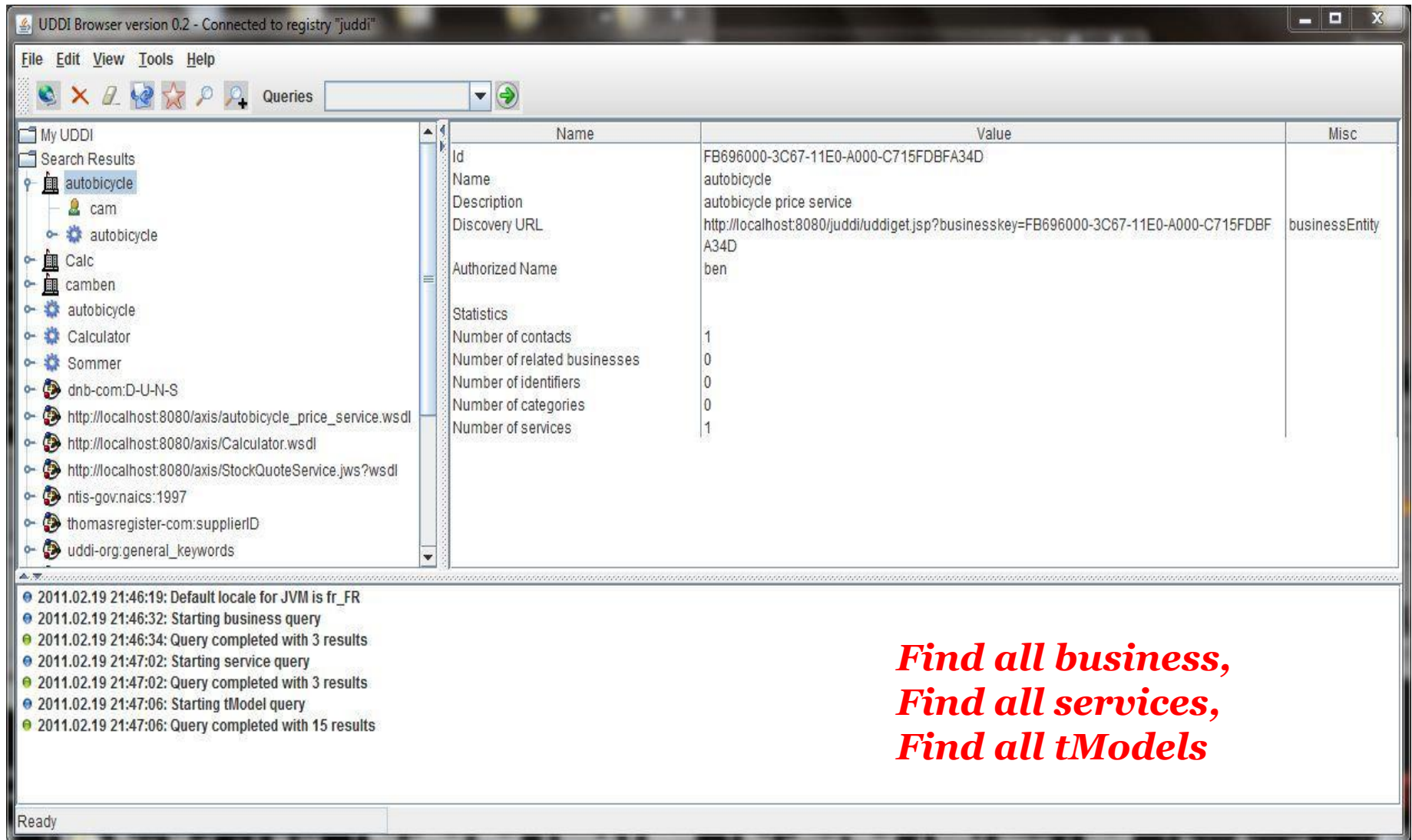
[find business](#)
[find service](#)
[find binding](#)
[find tModel](#)
[find relatedBusinesses](#)
[get businessDetail](#)
[get businessDetailExt](#)
[get serviceDetail](#)
[get bindingDetail](#)
[get tModelDetail](#)

UDDI Publish API

[get authToken](#)
[get registeredInfo](#)
[discard authToken](#)
[save business](#)
[save service](#)
[save binding](#)
[save tModel](#)
[delete business](#)
[delete service](#)
[delete binding](#)
[delete tModel](#)

Registre de services - Exemple

Cas d'usage - Recherche d'un Service Web avec UDDI Browser:



The screenshot shows the UDDI Browser interface. The left pane displays a tree view of search results under 'My UDDI' and 'Search Results'. The 'autobicycle' entry is selected, showing a list of related services including 'cam', 'autobicycle', 'Calc', 'camben', 'autobicycle', 'Calculator', 'Sommer', 'dnb-com:D-U-N-S', 'http://localhost:8080/axis/autobicycle_price_service.wsdl', 'http://localhost:8080/axis/Calculator.wsdl', 'http://localhost:8080/axis/StockQuoteService.jws?wsdl', 'ntis-gov:naics:1997', 'thomasregister-com:supplierID', and 'uddi-org:general_keywords'.

The right pane displays the details for the selected 'autobicycle' service. The details are organized into a table with columns: Name, Value, and Misc.

Name	Value	Misc
Id	FB696000-3C67-11E0-A000-C715FDBFA34D	
Name	autobicycle	
Description	autobicycle price service	
Discovery URL	http://localhost:8080/juddi/uddiget.jsp?businesskey=FB696000-3C67-11E0-A000-C715FDBF	businessEntity
Authorized Name	A34D ben	
Statistics		
Number of contacts	1	
Number of related businesses	0	
Number of identifiers	0	
Number of categories	0	
Number of services	1	

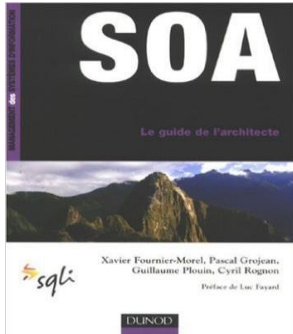
The bottom pane shows a log of events:

- 2011.02.19 21:46:19: Default locale for JVM is fr_FR
- 2011.02.19 21:46:32: Starting business query
- 2011.02.19 21:46:34: Query completed with 3 results
- 2011.02.19 21:47:02: Starting service query
- 2011.02.19 21:47:02: Query completed with 3 results
- 2011.02.19 21:47:06: Starting tModel query
- 2011.02.19 21:47:06: Query completed with 15 results

Ready

***Find all business,
Find all services,
Find all tModels***

Ressources



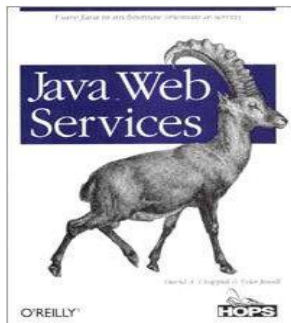
Le guide de l'architecte du SI

- ✓ Auteur : Xavier Fournier-Morel, Pascal Grosjean, ...
- ✓ Éditeur : Dunod
- ✓ Edition : Octobre 2006 - 302 pages - ISBN : 2100499726



SOA Principles of Service Design

- ✓ Auteur : Thomas Erl
- ✓ Éditeur : Prentice Hall Ptr
- ✓ Edition : Juillet 2007 - 608 pages - ISBN : 0132344823



Java Web Services

- ✓ Auteur : David Chappell & Tyler Jewell
- ✓ Éditeur : O'Reilly
- ✓ Edition : Mars 2002 - 276 pages - ISBN : 0-596-00269-6

Ressources

Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing

- ✓ Auteur : Armando Fox and David Patterson
- ✓ Éditeur : Strawberry Canyon LLC
- ✓ Edition : Aout 2012 - 412 pages - ISBN : 0984881212

Livre blanc SOA : Architecture Logique : Principes, structures et bonnes pratiques Auteur: Gilbert Raymond.Version 2.

Cours – Mickael Baron – SOA et Microservices

- ✓ http://mbaron.developpez.com/#page_soa

Cours – Camélia Benchaqroun - *Publication des services web (juddi)*

- ✓ <https://fr.scribd.com/document/50228577/Publication-des-services-web-JUDDI>