Service Web Java

Architecture SOA (Service Oriented Architecture)

- Collaboration pour des tâches communes
- Distants géographiquement mais interconnectés
- Hétérogène

Service

Un service est un composant logiciel *distribué*, exposant les fonctionnalités à forte valeur ajoutée d'un dommaine métier. Ils sont basés sur les langages et protocoles web HTTP, XML, TCP/IP qui ne nécéssitent pas de configuration particulière. Ils possèdent un contrat de fonctionnement qui content les informations nécessaires à leuur fonctionnement. Le service peut être carractérisé par 8 aspects:

- Contrat standardisé
- Couplage faible
- Abstraction
- Réutilisabilité
- Autonomie
- Sans état
- Découvrabilité

- Composabilité

Contrat Standardisé :

Entre le consommateur et le fournisseur de données. Il est lié à :

- La Syntaxe (*ex: attends un entier*)
- La sémantique (= définition des règles (*ex: Nombre de compte pas négatif*))
- La qualité du service (temps de réponse, tolérence aux pannes)

Couplage faible :

Echange de messages et passage par une interface —> pas d'accès dirrect aux classes métier

Abstraction :

Le contract ne dois contenir que les infos pertinentes. Fonctionne en "boite noire" : le fonctionnement ne dois pas être visible. Pas de variation dans le comportement du service.

Réutilisabilité *(peu utilisé en vrai)* :

Service accessible depuis un annuaire. le fournisseur doit déposer et mettre à jour le service. Utilise les standards (UDDI, ebXML)

Autonome/sans état :

Sans état : garantie plus de performance. Autonomie = prédicabilité

Composabilité :

Doit fonctionner de façon modulaire et non pas intégrée. Décompose un système complexe. S'inscrit dans une logique de compsition de services.

Technos disponibles

- Webservices étendus : s'appuient sur les standards UDDI/WSDL(fichier XML de definition de service)/SOAP
- Webservices REST (Representationnal State Transfer) : Utilise dirrectement HTTP au lieu de passer par SOAP. Utilise URI pour nommer et identifier une ressource.

Services web REST

Exploités pour les architectures orientées données (DOA). Rest n'est pas un standard. REST est un style d'architecture basé sur un mode de compréhension du web. Rest s'appuie sur les standards du web :

- Protocole HTTP

- URLs
- Formats de fichiers
- Sécurité via SSL

Fournisseurs de service web

Deux types de fournisseurs:

- Webservices orientés web (public)
- Webservices entreprise (privé)

Les grands noms du web (Twitter, Google, Facebook...) fournissent des webservices REST.

Plateformes de développement

La pluspart des plateformes de dev suppoertent les webservices (.NET, Java, PHP, C++, Python...). ces outils permettent de manipuler des messages SOAP, de données XML, mapper du XLM/classe, accéder à la couche HTTP.

Protocoles internet

URI : (Uniform Ressource Identifier) Mécanisme permettant aux utilisateurs et aux programmes de localiser des ressources web. L'URI est utilisé par HTTP, FTP, namespaces, XML, SMIL (Synchronized Multimedia Integration Language), SVG (Scalable Vector Graphic). L'URI est toujours basée sur le même modèle:

```
<modèle>://<autorité><chemin> ?<requete>
```

URL et URN sont des sous classes de URI

Exemple d'URL : http://www.google.com ; file://home/documents/index.html ; ftp://host@adress/dir

---

MIME

Multipurpose Internet Mail Extensions. Standard internet qui étend le format de données des courriels pour supporter des textes en différents codage de caractères autres que l'ASCII, des contenus non textuels, des contenus multiples, et des informations d'en-tête en d'autres codages que l'ASCII.

HTTP1.1

Utilise un jeu de requêtes/réponses entre un client et un serveur. La communication peut être directe mais elle peu aussi passer par :

- Un proxy

- Une passerelle
- Un tunnel

Post : Pas d'info dans l'url, prends en compte une taille de données plus importante.

Get : Paramètres dans l'url. Methode par défaut.

SSL & TLS

SSL *(Secure Socket Layer)* et TLS *(Transport Layer Security)* sont deux protocoles cryptographiques qui permettent l'authentification, et le chiffrement des données qui transitent entre des serveurs, des machines et des applications en réseau (notamment lorsqu'un client se connecte à un serveur Web). Le SSL est le prédécesseur du TLS. Au fil du temps, de nouvelles versions de ces protocoles ont vu le jour pour faire face aux vulnérabilités et prendre en charge des suites et des algorithmes de chiffrement toujours plus forts, toujours plus sécurisés.

XML

(eXtended Markup Language)

C'est un métalangage informatique de balisage générique. Sa syntaxe est dite « extensible » car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire. Elle est reconnaissable par son usage des chevrons encadrant les noms des balises. L'objectif initial de XML est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche...) entre systèmes d'informations hétérogènes.

Exemple de code :

```
<note>
    <to>You</to>
    <from>Me</from>
    <heading>Reminder</heading>
    <body>Don't forget about XML</body>
</note>
```

DTD

La DTD *(document type definition)*, ou définition de type de document, est, soit un fichier, soit une partie d'un document SGML ou XML, qui décrit ce document ou une classe de documents.

Exemple de code (programme TV)

```
<!DOCTYPE TVSCHEDULE [

<!ELEMENT TVSCHEDULE (CHANNEL+)>
<!ELEMENT CHANNEL (BANNER,DAY+)>
```

```
<!ELEMENT BANNER (#PCDATA)>
<!ELEMENT DAY (DATE,(HOLIDAY|PROGRAMSLOT+)+)>
<!ELEMENT HOLIDAY (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT PROGRAMSLOT (TIME,TITLE,DESCRIPTION?)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>

<!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
<!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
<!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
<!ATTLIST TITLE RATING CDATA #IMPLIED>
<!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
]>
<!-- #PCDATA  = Donnée alpha numérique -->
<!-- !ELEMENT nom valeur -->
<!-- !ATTLIST nom attribut type défaut -->
```

Namespace

Les namespaces désignent un lieu abstrait conçu pour accueillir des ensembles de termes appartenant à un même répertoire. Ils peuvent permettre d'utiliser plusieurs DTD au sein d'un même document. (Peu de chance d'en manipuler)

Dans l'exemple ci dessous context est ne namespace.

```
<context:composant-scan>
```

On peut définir des namespaces pour s'en servir plus tard ex:

```
<beans xmlns:ct="http://adresse">
    <ct:composant-scan base-package="package.name" />
</beans>
```

Norme traitement de cocument

JAX-Processing

- Java.xml.parser
- Java.xml.transform

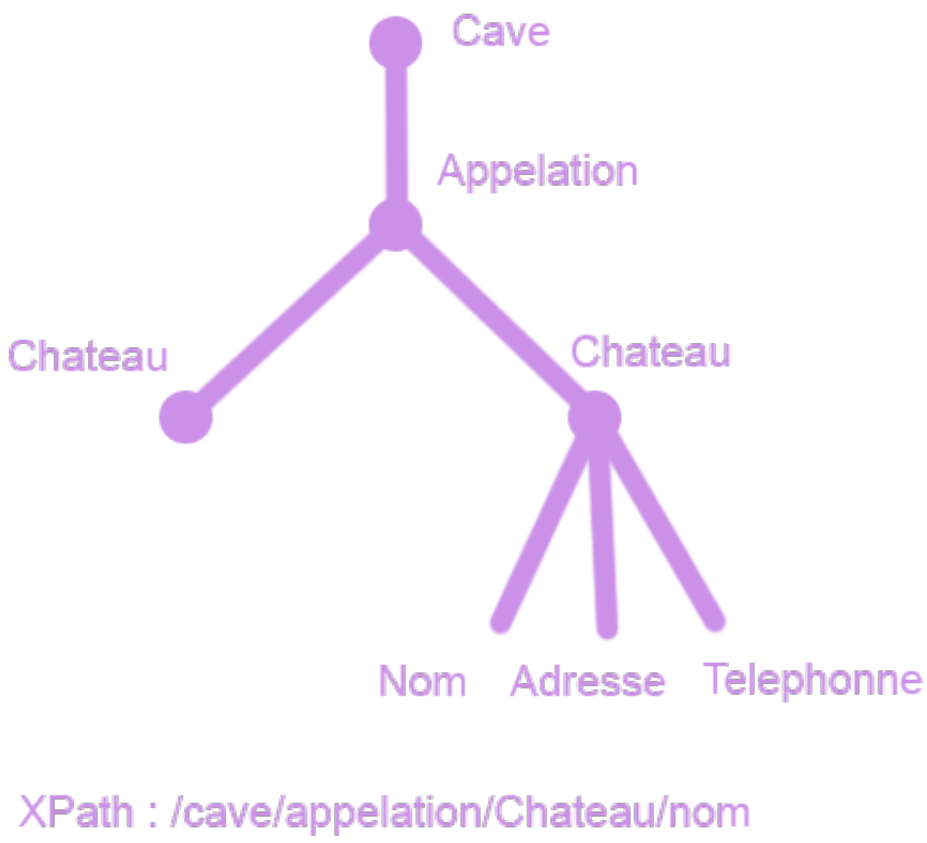
JAX-Binding

- représentation d'objet Java en XML (serialization, deserialization)

DOM

Document Object Model. Représentation en arbre de documents.

Exemple de représentation dom avec exemple de spath :



SOAP : généralités + exemple de code

Pros :

- Utile pour débugger une appli
- Utile pour réaliser des tests via SOAP UI
- Intersepter des messages bas niveau

Cons :

- APIs fourinissent une abstraction des messages SOAP

SOAP *(Simple Object Access Protocol)* : protocole de communication basé sur le language XML.

Concept des messages SOAP : Utilisés pour envoyer et recevoir. Il peut êytre transmis à plusieurs intermediares avant d'atteindre le destinataire final. Il est véhiculé via un protocole de transfert (dans la palupart des cas: HTTP).

En-tête (Header) : Contient les infos pour authentifier l'émetteur, contextualise la transaction, peut permettre d'identifier l'émetteur.

Corps (body): Il contient le message pour le destinataier ou une erreur.

Exemple:

Message soap pour appeler l'opération HelloWorld contenant un paramètre:

```
<soapenv:Enveloppe
                    xlmns:soapenv="http://schemas.xmlsoap.org/soap/envelope"
                    xlmns:hel="http://helloworldwebservice.lisi.ensma.fr" >
    <soapenv:Header />
    <soapenv:Body>
        <hel:makeHelloWorld>
            <value>Bonjour</value>
        </hel:makeHelloWorld>
    </soapenv:Body>
</soapenv:Enveloppe>
```

Message SOAP pour appeler l'operation simpleHelloWorldne qui ne contient pas le paramètre :

```
<soapenv:Enveloppe
                    xlmns:soapenv="http://schemas.xmlsoap.org/soap/envelope"
                    xlmns:hel="http://helloworldwebservice.lisi.ensma.fr" >
    <soapenv:Header />
    <soapenv:Body>
        <hel:simpleHelloWorld />
    </soapenv:Body>
</soapenv:Enveloppe>
```

WSDL : Généralités + exemple

Web Service Description Language

Basé sur XML et fournit une description indépendante du langage de la plateforme. Les concepts du document WSDL:

- Types : Définition des types de données
- Messages : Definition abstraite des données en cours de transmission
- PortType : Ensemble d'opérations. Chaque opération peut avoir 0 ou 1 message en entrée et 0 ou n en sortie (ou erreur)
- Binding : Liaisoon entre PortType et un protocole concret (SOAP, HTTP...) (génrés automatiquement par les outils.)
- Port : Point d'accès du service
- Opération : description d'une action proposée dans le port

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>
```

```
<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<!-- n this example the <portType> element defines "glossaryTerms" as the
name of a port, and "getTerm" as the name of an operation.
The "getTerm" operation has an input message called "getTermRequest" and an
output message called "getTermResponse".
The <message> elements define the parts of each message and the associated
data types. -->
```