Conceptions des interfaces externes

M1 MIAGE Université Paris 1 Panthéon-Sorbonne

2020-2021

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Conception des interfaces externes

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Une fenêtre sur le monde extérieur

- Il est très probable que votre projet doit s'interfacer avec d'autres applications hors de notre contrôle.
- Exemple: une boutique en ligne s'interface avec la compta ou les stocks
- On peut vouloir créer des interfaces pour de multiples raisons
  - Lire des données externes pour traitement
  - Utiliser des applications externes pour effectuer un travail
  - Fournir des fonctionnalités à des applications tierces

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Inconvénient des interfaces externes

- Risque de stabilité
  - si l'application tierce est en maintenance, votre application n'est plus fonctionnelle
  - si l'interface de l'application tierce change, vous devez mettre à jour votre système.
- Développement plus complexe et difficile à tester
  - un bug peut venir de l'application tierce
  - requiert une modélisation des échanges

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Avant de vous lancer

Il est important d'arriver à prendre une décision sur ces aspects:

- Sélection du type d'interface (web services, queue de messages)
- export de services ou de fonctions?
- structure des types de données échangés
- Évènements déclenchant l'échange des données
- Gestion des erreurs et responsabilité.

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Consommer des données externes

De nombreuses stratégies sont possibles:

- Lecture directe dans une base de données externe
- Webservices REST ou SOAP
- Objets distants (CORBA, Enterprise Java Beans)
- Méthodes distantes (RPC)
- Queue de message

Nous allons voir dans cette section les avantages et inconvénients de chaque méthode

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Lecture directe dans une base de données externe

- Avantages ☺
  - C'est la méthode la plus simple et la moins chère à implémenter
  - Ne requiert aucun effort de la part de l'équipe tierce
  - les données sont toujours à jour

cette stratégie est surtout utilisée en début de projet, car elle est simple et utilise des outils déjà connus (JDBC, JPA)

- Inconvénients ☹
  - Rupture de compatibilité en cas de changement de structure de la base.
  - Risque de laisser des utilisateurs externes dégrader les performances de la base (read locking)

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Stratégie de l'Operational Data Store

- Le système va lire seulement une copie des données des applications tierces
- Plusieurs sources de données d'origine peuvent y être agrégées
- Le risque de rupture de compatibilité est déporté vers l'application d'agrégation/réplication.
- Exemple: en cas de changement de logiciel de comptabilité, seul l'export des données vers l'ODS est impacté, et pas le code métier de notre système.
- Différentes stratégies d'implémentation
  - à l'aide de bases de données génériques et de code
  - un système de Datawarehouse fournissant des fonctionnalités d'ETL (Extract Transform Load)

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Stratégie de l'Operational Data Store

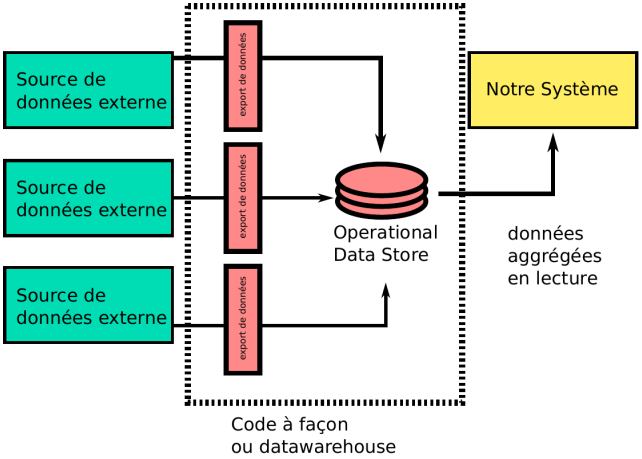


FIGURE 1 – Operational Data Store

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Webservices SOAP: présentation

- SOAP = Simple Object Access Protocol
- Il permet d'accéder à des objets distants via le réseau en utilisant un standard
- Permet d'accéder à des données ou des réaliser des appels de procédure distante (RPC)
- Écrit en XML et transporté via HTTP ou SMTP

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Webservices SOAP: Caractéristiques

- Définition des services documentée automatiquement avec WSDL (Web Service Description Language)
- Bon écosystème et outils disponibles
- Il permet le bon découplage et l'isolation des couches d'abstraction
- Indépendant de la plateforme et du langage de programmation
- Peut être publié de façon sécurisée sur Internet

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Webservices SOAP: example WSDL

```
<?xml version="1.0"?>
<definitions name="Status" xmlns="*http://schemas.xmlsoap.org/wsdl/">*
    <types>
        <xsd:schema xmlns="*http://www.w3.org/2001/XMLSchema" targetNamespace="urn:Status">*
            <xsd:complexType name="StatusResult">
                <xsd:all>
                    <xsd:element name="message" type="xsd:string"/>
                    <xsd:element name="name" type="xsd:string"/>
                </xsd:all>
            </xsd:complexType>
        </xsd:schema>
    </types>
    <binding name="StatusBinding" type="typens:StatusPort">
        <soap:binding style="rpc" transport="*http://schemas.xmlsoap.org/soap/http"/>*
        <operation name="Status">
            <soap:operation soapAction="urn:StatusAction"/>
            <input>...</input>
            <output>...</output>
        </operation>
    </binding>
    <service name="StatusService">...</service>
</definitions>
```

Webservices SOAP: Échange de données

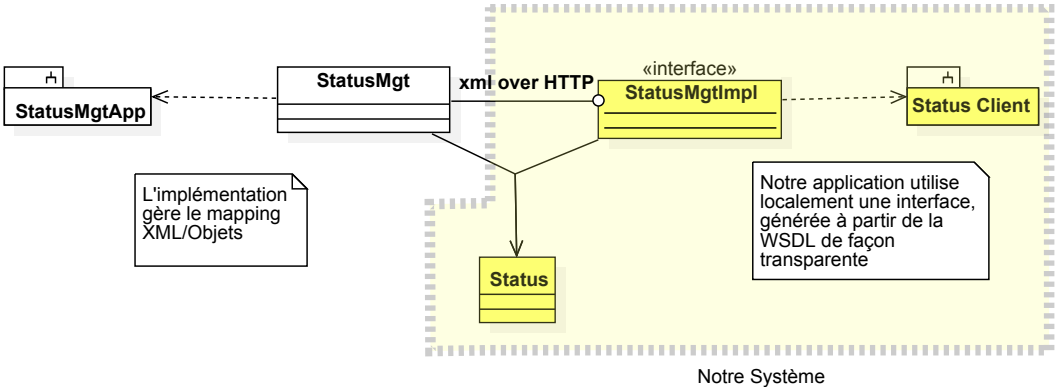


FIGURE 2 – Soap

Webservices SOAP: RPC (Remote Procedure Call)

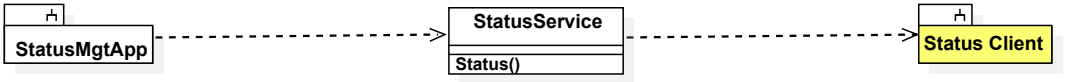


FIGURE 3 – Soap

- fournit une isolation entre le code métier et l'interface d'appel des méthodes
- permet le refactoring des deux côtés, tant que la WSDL reste inchangée

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Webservices SOAP: Incovénients ☹

- plus complexe à développer que les autres méthodes
- peu adaptées à de larges quantités de données (format XML lisible)
- rien n'est prévu en cas d'indisponibilité

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Webservices REST: tout est ressource

- L'architecture RESTful est dominante sur le marché des web services
- Les web services REST utilisent la sémantique des URL pour structurer chaque information sous forme de ressource.
- Dans l'approche RESTful, tout est ressource:

Webservices REST: tout est ressource

- L'architecture RESTful est dominante sur le marché des web services
- Les web services REST utilisent la sémantique des URL pour structurer chaque information sous forme de ressource.
- Dans l'approche RESTful, tout est ressource:

ressource	URL
Un billet d'avion	*http://ama.com/api/ticket/12351W*
Le vol AirFrance 252 du 3 mars	*http://ama.com/api/flight/AF253/march/03*
Un utilisateur	*http://miage.dev/user/1*
La page wikipedia sur les ragondins	*http://fr.wikipedia.org/wiki/Ragondins*

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Web services REST: Représentation des ressources

- Les ressources sont représentées sous forme “sérialisée” ou sous forme de flux d’octets.
  - Généralement, la sérialisation XML ou JSON est utilisée
  - Mais toutes les sérialisations sont possibles (CSV, HTML, Protobuf)
- La représentation des ressources (mime type) est négociée entre le client et le serveur (content negotiation [1]).

Le client supporte	Le serveur supporte
Header: Accept	Header: Content-type
application/json	application/xml
application/zip	text/csv
application/octet-stream	—
text/csv	—

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

1. https://tools.ietf.org/html/rfc7231

Webservices REST: anatomie d'une requête

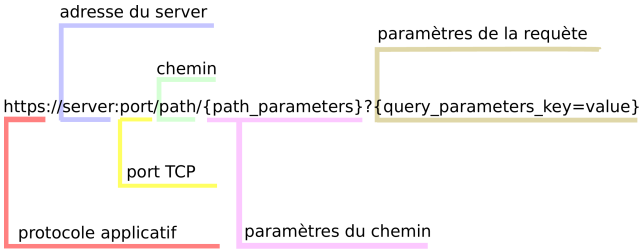


FIGURE 4 – URL

- Verbe HTTP: GET, POST, PUT, DELETE, OPTIONS, HEAD, CONNECT, PATCH
- Les Headers HTTP (utilisé pour l'authentification, le caching, négociation du contenu, Proxies, Redirect)
- Le corps de la requête (payload): XML,JSON,flux d'octet...

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Webservices REST : anatomie d'une réponse

- Code de retour (2xx, 3xx, 4xx, 5xx)
- Headers HTTP
- corps de la réponse (payload): XML, JSON, flux d'octet...

```
HTTP/1.1 301 Moved Permanently
Date: Tue, 03 Sep 2019 05:25:32 GMT
Server: Apache/2.4.18 (Ubuntu)
Location: https://nextnet.top/
Content-Length: 305
Content-Type: text/html; charset=iso-8859-1
X-Cache: MISS from box2-t25
X-Cache-Lookup: MISS from box2-t25:3128
< Connection: keep-alive
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Webservices REST : exemple d'échange

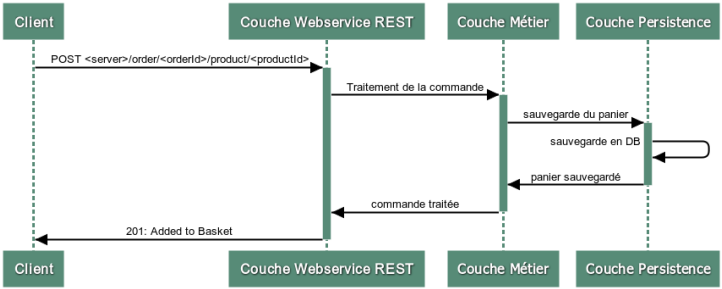


FIGURE 5 – Exemple cycle requête/réponse REST

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Webservices REST: avantages et inconvénient

- Avantages ☺
  - permet d'obtenir une bonne isolation entre le code métier et l'interface
  - indépendants de la plateforme et du langage de programmation
  - plus facile à mettre en place et à implémenter que les WS-SOAP
- Inconvénient ☹
  - rien n'est prévu en cas d'indisponibilité

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Queues de Messages

- C'est une méthode de communication asynchrone: la requête est publiée, mais la réponse n'arrive pas immédiatement.
- Le principal avantage est la garantie de service: si le message est publié alors notre système est en maintenance, il sera stocké et traité lors de la remise en route.
- La solution est également indépendante de la plateforme
- Elle peut être déployée en utilisant 2 principaux design patterns: point-to-point (PTP) et publish-and-suscribe (pub-sub)

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Queues de Messages: Point-to-Point

- Dans le modèle PTP, les producteurs de messages sont appelés les *senders* et les consommateurs les *receivers*.
- Ils échangent des messages à l'aide d'une destination, appelée Queue de message.
- Un message n'est consommé que par 1 seul receiver.

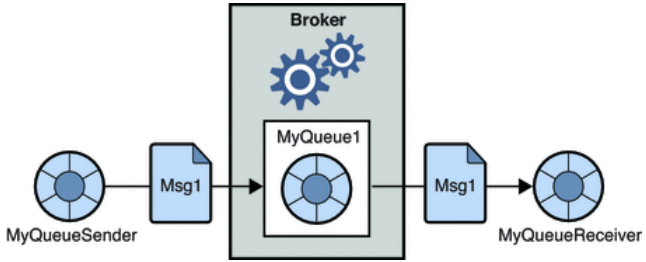


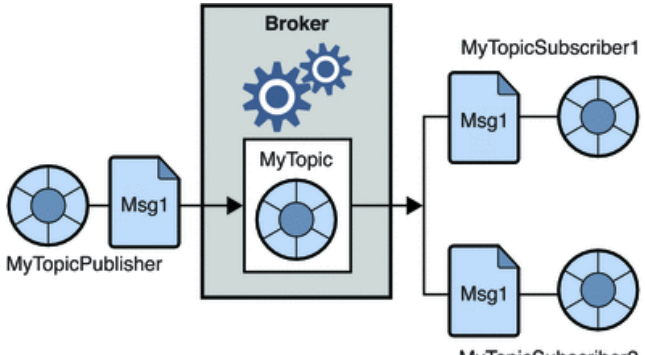
FIGURE 6 – Modèle PTP [2]

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

2. https://docs.oracle.com/cd/E19340-01/820-6424/6nhokb1ia/index.html

Queues de Message: Publish and Subscribe

- Dans le modèle pub-sub, les producteurs de messages sont appelés les *publishers* et les consommateurs les *subscribers*.
- Ils échangent des messages grâce à une destination appelée *topic*.
- Les publishers envoient des messages à un topic, alors que les *subscribers* consomment leurs messages d'un topic.



Queues de Message: Avantages et Inconvénients ☹

- Avantages ☺
  - Garantie de livraison
  - Indépendant de la plateforme
    - Advance. Message Queuing Protocol (AMQP) norme OASIS et ISO [4]
    - Message Queuing Telemetry Transport (MQTT) norme ISO [5]
    - Java Messaging Système (JSM) avec des middlewares compatibles [6]
- Inconvénients ☹
  - Ajoute une brique à l'architecture
  - Pas de génération de la documentation sur les messages

4. https://www.iso.org/standard/64955.html
5. https://www.iso.org/standard/69466.html
6. https://jcp.org/en/jsr/detail?id=343

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

CORBA/ORB

- Caractéristiques
  - Common Object Request Broker Architecture (CORBA) = standard OMG [7]
  - ORB = Object Request Broker
  - Rend disponible à distance un objet de votre système et supporte ses invocations
  - On peut fait communiquer 2 ORB avec TCP/IP à l'aide du protocole IIOP (Internet Inter-ORB Protocol)
- Avantages ☺
  - Standard indépendant de la plateforme
  - Utilisation naturelle en OOP
- Inconvénients ☹
  - Architecture lourde à configurer
  - rien n'est prévu en cas d'indisponibilité

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

7. https://www.omg.org/spec/CORBA

EJB (Enterprise Java Beans)

- EJB = CORBA pour JAVA + Best practice
- Standard JAVA : JSR 345
- Avantages ☺
  - Génération native de la documentation au travers d'Interfaces Java
  - Nombreux outils disponibles
  - Très bonne isolation des couches logiques
- Inconvénients ☹
  - Uniquement JAVA
  - rien n'est prévu en cas d'indisponibilité

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Erreurs fréquentes ☹

- Utilisation d'un modèle asynchrone alors qu'une réponse est requise
  - ajoute un temps de latence et complexifie grandement le déploiement et la programmation
- Utiliser un partage de fichier
  - Deux applications peuvent utiliser un partage de fichier pour échanger des informations.
  - Une application écrit un fichier sur un File System Distribué
  - Une autre application vérifie périodiquement la présence/mise à jour du fichier
  - Cette solution est simple à mettre en place, mais possède tous les inconvénients des Queues de messages (nouveau composant à ajouter dans l'archi, modèles de programmation asynchrone) sans les avantages (disponibilité)
- Utiliser une méthode plus complexe à déployer alors qu'une méthode plus simple est disponible.
- Envoyer directement des objets Java sérialisés par la Sérialisation native.

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Conseils

- Un certain nombre de bonnes pratiques *doivent* être mises en place dans vos architectures
  - Il est capital de documenter toutes les interfaces externes
  - Pour les interfaces vers des systèmes légataires, utiliser de type de données simples (CSV)
  - Logger tout accès à vos interfaces externes (+ identifiant du client)
  - Écrire des tests unitaires ☺

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE