

Applications et Services WEB: Architecture REST

Erick Stattner

Laboratoire LAMIA
Université des Antilles et de la Guyane
France
erick.stattner@univ-ag.fr

Guadeloupe
2014 - 2015



Description du cours

Objectif:

- Se familiariser avec la notion d'applications et de services WEB
- Être capables:
 - ▶ Savoir ce qu'est un service WEB
 - ▶ Développer et déployer un service
 - ▶ Développer des clients dédiés

Description du cours

Organisation de l'enseignement:

- CI: 15h
 - ▶ 8h: REST
 - ▶ 7h: SOAP
- TD/TP: 15H

Modalités dévaluation de l'UE:

- Mini-projet
- Projet global

Outline

- 1 Introduction
 - Le web en chiffre
 - Contexte
 - Solution
- 2 Services WEB
- 3 Architecture REST
- 4 JavaEE et JAX-RS

Introduction

Le web en chiffre

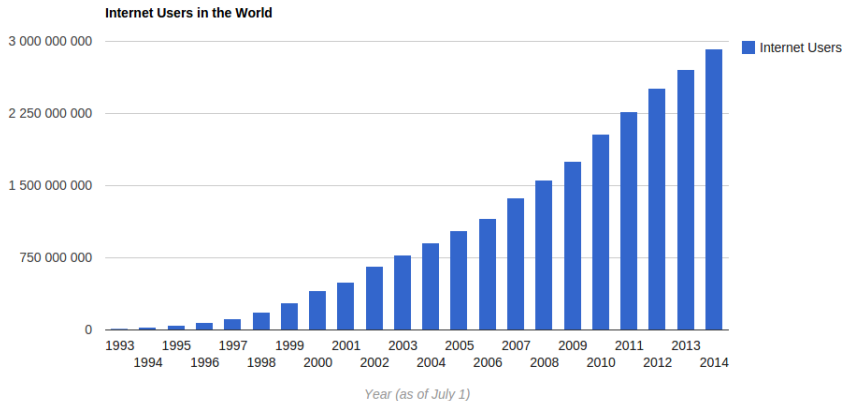


Figure: source: *internetlivestats* , *internetworldstats*

Introduction

Le web en chiffre

Vers une escalade des périphériques connectés

- Ordinateur/Serveur
- Téléphone/Tablette
- Montre
- Lunettes
- Télé
- Voiture
- Maison
- Electro-ménager
- ...



Introduction

Le web en chiffre

Vers une escalade des périphériques connectés



Figure: Source: Le Figaro 11/04/2013

Introduction

Contexte

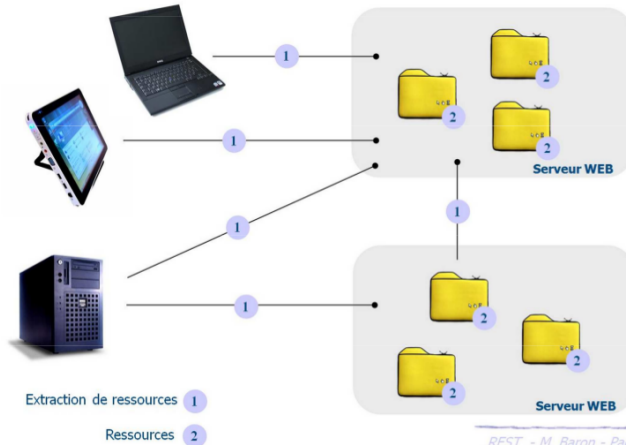
Contexte

- L'informatique d'entreprise est de plus en plus répartie
 - ▶ ex. classique: Agence de voyage (réserver billet, hôtel, voiture).
- Hétérogénéité des périphériques oblige à proposer des solutions adaptatives ("*Responsive design*").
 - ▶ ex. Consultation de son compte (pc, tel, tablette)
- Décentraliser les calculs
 - ▶ ex. problème consommation (tablette, tel, montre, lunettes, etc.)
- Différents besoins
 - ▶ ex. Affichage, statistiques, simulation, ...

Introduction

Contexte

Contexte



REST - M. Baron - Page 7

Introduction

Solution

Solution idéale:

- Architecture adaptative
 - ▶ ex. Appli bancaire communique avec différents clients
- Système de communication ouvert
 - ▶ ex. Information lisible et interprétable par tous les clients
- Client va traiter les données

Solution

Services WEB !

Outline

- 1 Introduction
- 2 Services WEB
 - Application VS Service WEB
 - Définitions
 - Exemples de service WEB
 - Architectures pour les WEB Services
- 3 Architecture REST
- 4 JavaEE et JAX-RS

Services WEB

Application VS Service WEB

Application VS Service WEB ?

Services WEB

Application VS Service WEB

Application

- Exploite plusieurs services
- Offre plusieurs fonctionnalités
- Propose généralement un rendu graphique

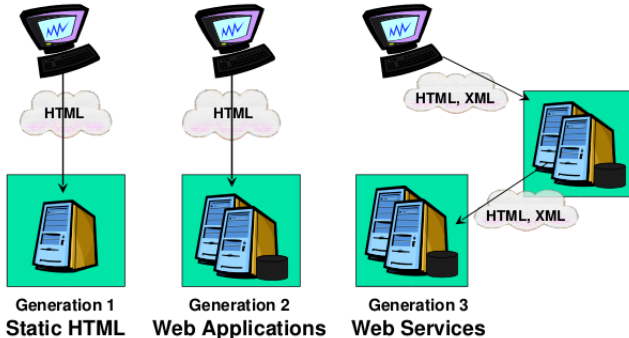
Service

- Limité à une opération élémentaire
- Fournit des données brutes ou formatées
- Généralement pas de rendu graphique
- Peut faire appel à d'autres services

Services WEB

Application VS Service WEB

Application VS Service WEB



Services WEB

Définitions

Définitions

- Selon le Wikipedia:

Une application Web (aussi appelée WebApp) est une application manipulable grâce à un **navigateur Web**.

Elle est généralement placée sur un serveur et se manipule **à travers une Interface** en actionnant des widgets à l'aide d'un navigateur Web, via un réseau informatique (Internet, intranet, réseau local, etc.).

- Exemples ?

Services WEB

Définitions

Exemple Application WEB

The screenshot displays the Ligo.com flight search results for a round trip from Pointe à Pitre (GP) to Paris (FR) on Saturday, September 13, 2014, to Saturday, September 20, 2014. The interface includes a sidebar with filters for budget, escalas, and departure times. The main content area shows a list of flight options, with the first result being a direct flight on Corsair for 458 € per passenger. The interface also features a comparison section for other travel services and a sidebar with additional flight information.

Ligo.com

Pointe à Pitre GP samedi 13 sept. 2014 → Paris FR samedi 20 sept. 2014 **MODIFIER**

351/351 résultats affichés

Filtrer les résultats

Utilisez les filtres pour affiner votre recherche

Mon budget

Escalas

- ☒ Tous
- ☒ vols directs 458 €
- ☒ 1 escale 542 €
- ☒ 2 escales ou plus 1 399 €

Horaires du vol aller

- ☒ Décollage
- ☐ Atterrissage

Décollage: 06:45 - 20:40

Horaires du vol retour

Durée

Résumé des offres

Dates flexibles -10 €

Alerte de prix Votre email

Vols vers Paris dès 30€

www.edreams.fr/france
Offre limitée ! Réservez vite sur eDreams.fr
Tous les vols Low Cost Les promos en cours
Billet A/R dès 19 euros Réservez au meilleur prix

Prix total * en EUR Moyen de paiement Sans préférence **WEB**

448 € Economisez 10 € et plus avec les dates flexibles !

458 € 458 € / passager

Corsair 19:45 PTP → 09:50 ORY direct (8h05)
14:40 ORY → 17:25 PTP direct (8h45)

+ d'infos

Gotogate.com +4 autres sites marchands Détails Alerte de prix

458 € 458 € / passager

Corsair 16:45 PTP → 06:55 ORY direct (8h10)
14:40 ORY → 17:25 PTP direct (8h45)

+ d'infos

Gotogate.com +4 autres sites marchands Détails Alerte de prix

458 € 458 € / passager

Corsair 20:40 PTP → 10:50 ORY direct (8h10)
14:40 ORY → 17:25 PTP direct (8h45)

+ d'infos

Gotogate.com +4 autres sites marchands Détails Alerte de prix

Comparer ligo.com avec

- GOVOYAGES** Comparer
- opodo** Comparer
- eDreams** Comparer

Annonces Yahoo

Vol Pointe à Pitre - Pointe à Pitre
Billetsdiscount.com
Billetsdiscount.com
Tous les Vols sur la Guadeloupe Billet Pointe à Pitre dès 369 € TTC

Vol Pointe à Pitre - Pointe à Pitre
Billetsdiscount.com
Billetsdiscount.com
Tous les Vols sur la Guadeloupe Billet Pointe à Pitre dès 369 € TTC

Promo Classe Business
Tgv Air
Nos destinations
Les Bons Plans Vol

Vols Fort de France/PTP

Services WEB

Définitions

Exemple Application WEB

Services WEB

Définitions

Définitions

- Selon le Wikipedia:

Un service web est un programme informatique de la famille des technologies web permettant **la communication et l'échange de données** entre applications et systèmes hétérogènes dans des environnements distribués.

Il s'agit donc d'un **ensemble de fonctionnalités exposées sur internet** ou sur un intranet, par et pour des applications ou machines, sans intervention humaine, de manière synchrone ou asynchrone.

Services WEB

Définitions

Définitions

- Selon le W3C (World Wide Web Consortium)
A web service is a system designed to support **machine-to-machine interaction** over a network.
It has an interface that can be discovered dynamically and which is described in a **machine-processable format**.

Services WEB

Définitions

Définitions

- Selon le **Dico du net**

Un service web est une technologie permettant à des **applications de dialoguer** à distance via Internet, **indépendamment des plates-formes** et des langages sur lesquelles elles reposent.

Services WEB

Définitions

Définitions

- Pus généralement:

Un service Web est un programme sollicité via Internet par **différents type de clients**, permettant **l'échange de données** afin que l'application appelante puisse intégrer le résultat de cet échange à ses propres analyses.

Les requêtes et les réponses s'effectuent dans des **formats ouverts (HTML, XML, JSON ou text)** et transitent par Internet.

- Exemples ?

Services WEB

Définitions

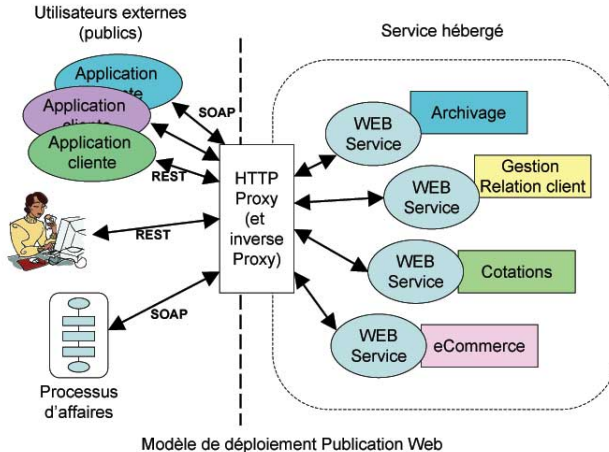
Principales caractéristiques:

- **Fonctionnalité** utilisable via Internet
- **Inter-opérables**: Interface publique décrite d'une manière interprétable par tous
- Systèmes **faiblement couplés**, client ne connaît pas forcément le fournisseur
- Le transport des données repose sur des "**protocoles du WEB**": HTTP, FTP, SMTP, ...
- **Standard ouvert**: Échange de données s'effectue dans un format standard
XML, JSON, HTML, Text, ...
- **Le client est chargé d'analyser**, traiter et/ou afficher les données reçues
- **Indépendante** des plates-formes et des langages

Services WEB

Définitions

Principe de base



Services WEB

Définitions

Client

- 1 Prend connaissance des interfaces publiques
- 2 Construit la requête et la normalise en respectant les interfaces (XML, JSON, Text)
- 3 Envoie la requête (protocole du WEB, ex. HTTP)
- 4 Reçoit les données, les interprète (XML)
- 5 Traite localement (affichage, calculs, etc.)

Services WEB

Définitions

Serveur

- 1 Définit ses interfaces (XML, WSDL, etc.)
- 2 Reçoit les requêtes
- 3 Les traduit et effectue le bon traitement
- 4 Normalise la réponse et envoie le résultat au client (HTML, XML, JSON)

Services WEB

Exemples de service WEB

Exemple 1: Actualités

The image displays two browser windows side-by-side. The left window shows the Clubic website (www.clubic.com) with a search bar and navigation links. The main content area features three news items: 'Samsung Galaxy Mega 6.3 : plus tablette que phablette', 'Téléphones Nokia : le best of depuis 25 ans', and 'Microsoft-Nokia: 5,4 milliards d'E et une stratégie multifacette'. Below these are sections for 'SUJETS RÉCENTS' and 'S'INFORMER' (Tokyo JO 2020 : une vitrine technologique pour le Japon) and 'S'ÉQUIPER' (PC Portable, Appareil photo numérique, etc.). The right window shows the source code of the RSS feed at 'view-source:com.clubic.feedsportal.com/c/33464/f/581979/index.rss'. The XML code includes a channel title 'Clubic : Actualité informatique, Comparatifs, Logiciels et Forum' and an item titled 'Tokyo JO 2020 : une vitrine technologique pour le Japon' with a description in French about the 2020 Tokyo Olympics and technological innovation.

Services WEB

Exemples de service WEB

Exemple 1: Actualités



Services WEB

Exemples de service WEB

Exemple 2: Analyse des données

The screenshot shows a web browser with two panes. The left pane displays the DBLP Bibliography Server for Erick Stattner, showing a list of publications from 2013 and 2012. The right pane shows the XML data for the first publication, 'D2SNet: Dynamics of diffusion and dynamic human behaviour in social networks'.

Publications from 2013:

Year	Publication
2013	Erick Stattner, Martine Collard, Nicolas Vidot: D2SNet: Dynamics of diffusion and dynamic human behaviour in social networks. <i>Computers in Human Behavior</i> 29(2): 496-509 (2013)
2013	Erick Stattner, Martine Collard: From Frequent Features to Frequent Social Links. <i>IJISMD</i> 4(3): 76-98 (2013)
2013	Erick Stattner, Martine Collard: Towards a hybrid algorithm for extracting maximal frequent conceptual links in social networks. <i>ICIS</i> 2013: 1-8
2013	Erick Stattner, Wilfried Segretier, Martine Collard, Philippe Hunel, Nicolas Vidot: Song-based Classification techniques for Endangered Bird Conservation. <i>CoRR</i> abs/1306.5349 (2013)

Publications from 2012:

Year	Publication
2012	Erick Stattner, Martine Collard, Nicolas Vidot: Network-Based Modeling in

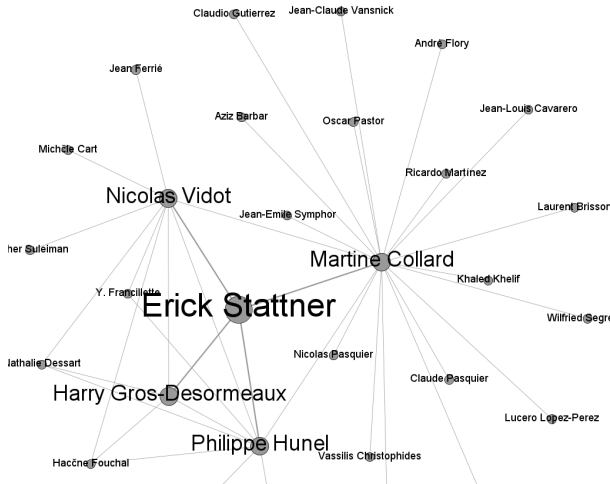
XML Data for the first publication:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<dblp:person name="Erick Stattner" n="23">
  <person key="homepages/28/8138" mdate="2010-10-05">
    <author>Erick Stattner</author>
  </person>
  <article key="journals/chb/StattnerCV13" mdate="2013-01-23">
    <author>Erick Stattner</author>
    <author>Martine Collard</author>
    <author>Nicolas Vidot</author>
    <title>
      D2SNet: Dynamics of diffusion and dynamic human behaviour in social networks.
    </title>
    <pages>496-509</pages>
    <year>2013</year>
    <volume>29</volume>
    <journal>Computers in Human Behavior</journal>
    <number>2</number>
    <ee>http://dx.doi.org/10.1016/j.chb.2012.06.004</ee>
    <url>db/journals/chb/chb29.html#StattnerCV13</url>
  </article>
  <article key="journals/ijismd/StattnerC13" mdate="2013-09-06">
    <author>Erick Stattner</author>
    <author>Martine Collard</author>
    <title>From Frequent Features to Frequent Social Links.</title>
    <pages>76-98</pages>
    <year>2013</year>
    <volume>4</volume>
    <journal>IJISMD</journal>
    <number>3</number>
    <ee>http://dx.doi.org/10.4018/ijismd.2013070104</ee>
    <url>db/journals/ijismd/ijismd4.html#StattnerC13</url>
  </article>
  <inproceedings key="conf/ccis/StattnerC13" mdate="2013-08-26">
    <author>Erick Stattner</author>
    <author>Martine Collard</author>
    <author>Nicolas Vidot</author>
    <title>Towards a hybrid algorithm for extracting maximal frequent conceptual links in social networks.</title>
    <pages>1-8</pages>
    <year>2013</year>
    <volume>1</volume>
    <series>Conference on Information Systems</series>
    <url>db/conf/ccis/StattnerC13.html#StattnerCV13</url>
  </inproceedings>
</dblp:person>
```

Services WEB

Exemples de service WEB

Exemple 2: Analyse des données



Services WEB

Exemples de service WEB

Autres exemples:

- Réduction d'URL (goo.gl, TinyURL, bit.ly, etc.)

http://www.erickstattner.com/?page_id=21



goo.gl/sKV0Hf

- Analyse d'adresses mail

es@es → INCORRECT

- Association de codes postaux

97190 → Le Gosier

Services WEB

Exemples de service WEB

De nombreux services proposés par Google

<https://developers.google.com/apis-explorer/#p/>

- Analyse de données
- Traduction
- Réduction d'URL
- Statistique fréquentation site web
- etc.

Services WEB

Architectures pour les WEB Services

Deux grandes familles d'architecture pour les services WEB

● SOAP-based architecture

- ▶ Protocole de communications pour des systèmes distribués.
- ▶ Il est décrit en XML et standardisé par le W3C.
- ▶ Il encapsule les données échangées dans une enveloppe qui peut-être chiffrée et contenir des pièces jointes.

● RESTFul architecture

- ▶ REST (Representational State Transfer) est un style architecture de services Web
- ▶ Introduite en 2000 par Roy Fiedling dans sa thèse de doctorat.
- ▶ REST est une "méthodologie" pour la construction d'une application pour les services WEB.

Services WEB

Architectures pour les WEB Services

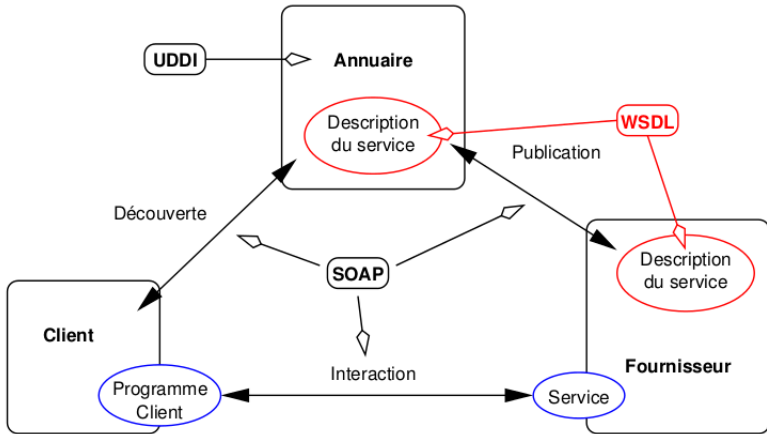
SOAP-based architecture

- SOAP: *Simple Object Access Protocol*
- Basé sur 3 acteurs principaux
 - ▶ Fournisseur de service
 - ▶ Annuaire
 - ▶ Client

Services WEB

Architectures pour les WEB Services

SOAP-based architecture



Services WEB

Architectures pour les WEB Services

Service provider

- Définit le service et ses interfaces
- Publie sa description dans l'annuaire
- Effectue le traitement
- Renvoie la réponse

Services WEB

Architectures pour les WEB Services

Annuaire

- Maintient à jour une liste de services
- Reçoit et enregistre la description des services
- Reçoit et répond aux recherches de service

Services WEB

Architectures pour les WEB Services

Programme client

- Obtient la description du service
- Fait la requête auprès du fournisseur de service
- Reçoit et traite le réponse

Services WEB

Architectures pour les WEB Services

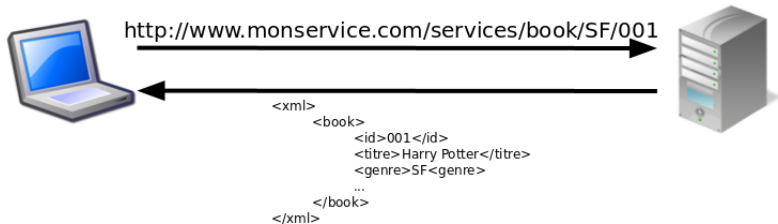
RESTFul architecture

- REST: REpresentational State Transfer
- Façon de concevoir une application
- Définit un ensemble de contraintes pour l'accès et la manipulation des données
- Structure l'application en différentes **ressources**
- Architecture **orientée ressource**
- Se base uniquement sur le protocole HTTP
- Basé sur le principe client/serveur

Services WEB

Architectures pour les WEB Services

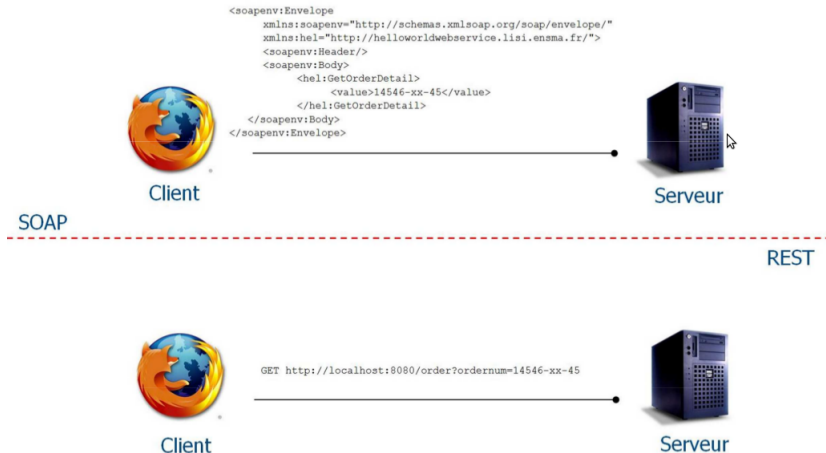
RESTful architecture



Services WEB

Architectures pour les WEB Services

SOAP VS REST



Services WEB

Architectures pour les WEB Services

SOAP VS REST

#	SOAP	REST
1	A XML-based message protocol	An architectural style protocol
2	Uses WSDL for communication between consumer and provider	Uses XML or JSON to send and receive data
3	Invokes services by calling RPC method	Simply calls services via URL path
4	Does not return human readable result	Result is readable which is just plain XML or JSON
5	Transfer is over HTTP. Also uses other protocols such as SMTP, FTP, etc.	Transfer is over HTTP only
6	JavaScript can call SOAP, but it is difficult to implement	Easy to call from JavaScript
7	Performance is not great compared to REST	Performance is much better compared to SOAP - less CPU intensive, leaner code etc.

Services WEB

Architectures pour les WEB Services

Bilan SOAP

Inconvénients

- Performances (encapsulation SOAP des données)
- Complexité, lourdeur de la mise en place/maintenance
- Cible l'appel de service

Avantages

- Standardisé
- Interopérabilité
- Sécurité (Possibilité de chiffrer échange SOAP)

Services WEB

Architectures pour les WEB Services

Bilan REST

Inconvénients

- Pas vraiment de standard
- Sécurité restreinte par l'emploi du HTTP, format des données

Avantages

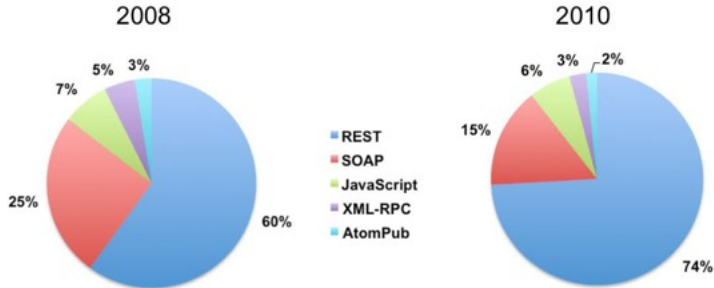
- Simplicité mise en place/maintenance
- Repose sur des principes éprouvés (WEB)
- Services facilement identifiables

Services WEB

Architectures pour les WEB Services

SOAP VS REST

REST vs. SOAP: Simplicity wins again



Distribution of API protocols and styles

Based on directory of 2,000 web APIs listed at ProgrammableWeb, May 2010

Outline

- 1 Introduction
- 2 Services WEB
- 3 **Architecture REST**
 - Présentation
 - Ressources
 - Interfaces et représentations
 - Outils
- 4 JavaEE et JAX-RS

Architecture REST

Présentation

Présentation

- REST: *REpresentational State Transfer*
- Décrit par Roy T. Fielding dans sa thèse de Doctorat en 2000, Chap 5
"Architectural Styles and the Design of Network-based Software Architectures"
<http://goo.gl/q68TE4>
 - ▶ Principaux acteurs de la spéf. du HTTP
 - ▶ Membre fondateur de la fondation Apache
 - ▶ Dev. du serveur WEB Apache
- Style d'architecture
 - ▶ \neq protocole de comm. (SOAP)
 - ▶ Repose sur le protocole HTTP
 - ▶ Spécifie des contraintes (interface uniforme)
 - ▶ Utilisé pour développer des Services WEB

Architecture REST

Présentation

Présentation

- Dans une architecture REST
 - ▶ *TOUT* est ressource (*archi. orientée ressource*)
 - ▶ Ressources identifiées par une URI (*Uniform Resource Identifier*)
 - ▶ Ressources manipulables via interface commune
i.e. Ressources supportent les mêmes opérations
 - ▶ Ressources ont différentes représentations: XML, TEXT, JSON, etc.
 - ▶ Serveur REST: fournit un accès aux ressources
 - ▶ Client REST: exploite les ressources, selon le format voulu
- Quand une application respecte ces principes: **RESTful**

Important

- Les services WEB avec REST sont sans états (Stateless)
 - ▶ Pas de mémoire des requêtes antérieures
 - ▶ Chaque requête envoyée doit contenir toutes les informations nécessaire au traitement

Architecture REST

Présentation

3 concepts majeurs:

- ❶ Ressource (Identifiant)
 - ▶ Identifiée par une URI
 - ▶ Exemple: `http://achats.com/livre/SF/Harry-Potter`
- ❷ Opération (Interface)
 - ▶ Action à effectuer sur la ressource
 - ▶ Méthodes HTTP: GET, POST, PUT et DELETE
- ❸ Représentation (Vue de la ressource ou de son état)
 - ▶ Informations échangées avec le service
 - ▶ TEXT, XML, JSON, ...

Architecture REST

Ressources

Ressource

- Tout ce qui est identifiable/manipulable dans le système
 - ▶ Document, Image, Personne, Le montant du compte d'un client, etc.
- Identifié de manière unique par un lien (URI)
- Une ressource peut avoir plusieurs URI
- Une URI identifie une seule ressource (ou un seul groupe de ressources)
- Construite de façon hiérarchique
- La représentation d'une ressource peut évoluer avec le temps
 - ▶ Lié au temps: ex. Dernier article
 - ▶ Modification structure: ex. Ajout d'un champ

Architecture REST

Ressources

Structure classique

- Structure hiérarchique
- Construction classique
 - ▶ `http://domaine.com/<plus général>/../<plus spécifique>`

Exemples d'URIs

- `/livre/SF`
- `/livre/SF/harrypotter/`
- `/livre/SF/harrypotter/5`
- `/livre/SF/harrypotter/l_ordre_du_phenix`
- `/livre/SF/harrypotter/5/année`
- `/livre/aventure/meilleur_vente`
- `/livre/search/50_nuances`

Architecture REST

Interfaces et représentations

Interfaces

- REST fournit une interface uniforme
- Chaque ressource supporte 4 opérations de base (CRUD)
 - ▶ Create
 - ▶ Read
 - ▶ Update
 - ▶ Delete
- REST s'appuie sur les méthodes HTTP pour ces opérations
 - ▶ GET:
Définit un accès en lecture. La ressource n'est pas modifiée.
 - ▶ POST:
Met à jour une ressource existante
 - ▶ PUT:
Crée une nouvelle ressource
 - ▶ DELETE:
Supprime une ressource

Architecture REST

Interfaces et représentations

Exemples:

- GET: /livre/SF/harrypotter/
- POST: /livre/SF/harrypotter/
- PUT: /livre/SF/harrypotter/2
- DELETE: /livre/SF/harrypotter/2

Attention

Toutes les méthodes ne sont pas obligés d'être implémentées !

Ex. POST: /livre/SF

Architecture REST

Interfaces et représentations

Rappel:

METHODE URL VERSION

EN-TETE : Valeur

...

EN-TETE : Valeur

Ligne vide

CORPS DE LA REQUETE

Exemple:

GET http://www.commentcamarche.net HTTP/1.0

Accept : text/html

If-Modified-Since : Saturday, 15-January-2000 14:37:11 GMT

User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)

Architecture REST

Interfaces et représentations

Exemple de requetes:

```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

Diagram illustrating the structure of an HTTP Request Message:

- Request Line:** GET /doc/test.html HTTP/1.1
- Request Headers:** Host: www.test101.com, Accept: image/gif, image/jpeg, */*, Accept-Language: en-us, Accept-Encoding: gzip, deflate, User-Agent: Mozilla/4.0, Content-Length: 35
- Request Message Header:** (Grouped with Request Headers)
- A blank line separates header & body**
- Request Message Body:** bookId=12345&author=Tan+Ah+Teck

Important

Rappel sur les requêtes HTTP: <http://goo.gl/uoN0IT>
source: *openclassrooms*

Architecture REST

Interfaces et représentations

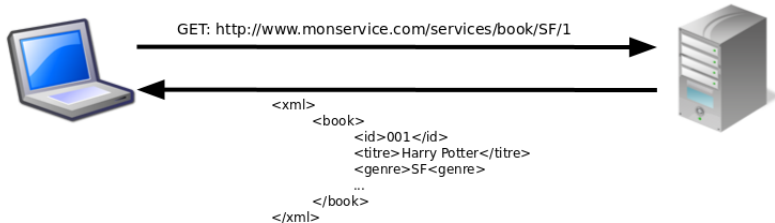
Représentation:

- Le client et le serveur échange des "*représentations*" des ressources en utilisant une interface et un protocole standardisé
 - ▶ Le client: lorsqu'il reçoit avec GET
 - ▶ Le serveur: lorsqu'il reçoit avec PUT ou POST
- Généralement: TEXT, JSON, XML, HTML, CSV, Défini par l'utilisateur
- Une même ressource peut-être proposée sous différents formats

Architecture REST

Interfaces et représentations

Exemple:



Architecture REST

Outils

Outils pour le dev.

- **Advanced REST client**: Chrome
- Postman: Chrome
- Poster: Firefox

API pour mettre en place Service REST

- PHP (Voir liste API: <http://goo.gl/OYITjY>)
 - ▶ CURL
 - ▶ HttpFul
 - ▶ Epiphany
 - ▶ ...
- JavaEE (JAX-RS, JSR311)
 - ▶ **Jersey**
 - ▶ Spring
 - ▶ RESTeasy
 - ▶ Restlet
 - ▶ ...

Outline

- 1 Introduction
- 2 Services WEB
- 3 Architecture REST
- 4 JavaEE et JAX-RS
 - Introduction
 - Fonctionnement JavaEE
 - Framework Jersey
 - Ressources

JavaEE et JAX-RS

Introduction

JavaEE

- Java Enterprise Edition, ou JavaEE (anciennement J2EE)
- Version de JAVA pour les entreprises
- Plus particulièrement à destination des appli et services WEB
- Première spécification proposée en 1999
- Basée sur la notion de **Servlet**
- Nécessite un serveur d'applications ("*serveur Java*")
 - ▶ **Apache Tomcat (Apache)**
 - ▶ GlassFish Server (Oracle)
 - ▶ Google App Engine (Google)
 - ▶ JBoss App. Server (Red Hat)
 - ▶ ...

JavaEE et JAX-RS

Introduction

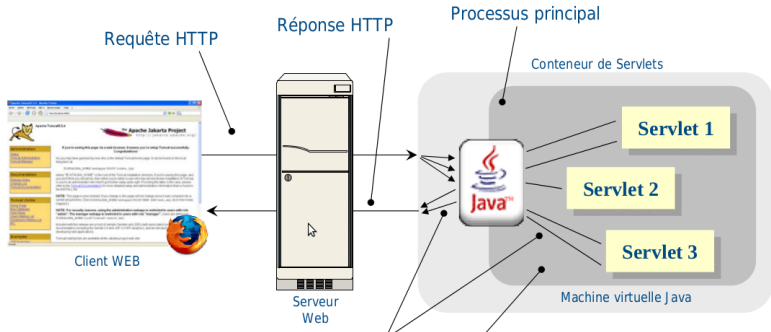
Servlet

- Composant logiciel écrit en Java fonctionnant coté serveur
 - ▶ Assimilable a: PHP, ASP, etc.
- Reçoit et traite les requêtes HTTP
- Fournit au client une réponse HTTP
- Une Servlet s'exécute dans un moteur (ou conteneur) de Servlets
 - ▶ Établit le lien entre la Servlet et le serveur Web
 - ▶ Associe à des URL virtuelles une Servlet
- Tout comme un programme JAVA, la Servlet s'exécute par l'intermédiaire d'une machine virtuelle
 - ▶ Nécessite que java soit installé sur le serveur
 - ▶ Pas nécessaire sur le client

JavaEE et JAX-RS

Introduction

Principe



JavaEE et JAX-RS

Introduction

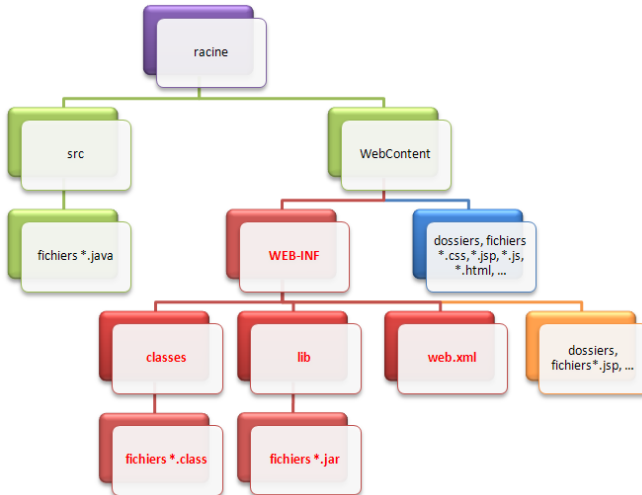
Intérêt

- Puissance du JAVA
 - ▶ Disponibilités de l'API Java et des toutes les API liées
 - ▶ Traitement d'images, de sons, connexion aux bases de données, chiffrement, graphisme, etc.
 - ▶ Gestion des erreurs par exception
 - ▶ Typage fort de JAVA
 - ▶ Technologie portable
- Ajout de JavaEE
 - ▶ Une servlet est chargée une seule fois
 - ▶ Servlet peut conserver son état

JavaEE et JAX-RS

Introduction

Structure d'une application JavaEE



JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur

- Mise en place d'une servlet nécessite deux étapes:
 - 1 Routage des requêtes à l'aide du fichier **web.xml**
 - 2 Traitement des informations par la Servlet

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 1) web.xml

- **OBLIGATOIREMENT** situé a la racine du dossier WEBINF
- **ATTENTION** Un seul fichier par application/projet
- Regroupe l'ensemble des informations de fonctionnement de l'application
- Permet d'associer une chemin (URL) à une servlet
- La servlet ainsi spécifiée sera chargée du traitement de la requête

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 1. web.xml

- Squelette classique fichier web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0">
  <servlet>
    <servlet-name>test</servlet-name>
    <servlet-class>services.Response</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>test</servlet-name>
    <url-pattern>/toto</url-pattern>
  </servlet-mapping>
</web-app>
```

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 1. web.xml

- Association de plusieurs chemins
- Deux pointeurs vers une même servlet

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ...>

  <display-name>Servlets affichant différemment le message
    HelloWorld
  </display-name>

  {
    <servlet>
      <servlet-name>HelloWorldServlet</servlet-name>
      <servlet-class>HelloWorld</servlet-class>
    </servlet>
    <servlet>
      <servlet-name>HelloWorldPrintWriter</servlet-name>
      <servlet-class>HelloWorldPrintWriter</servlet-class>
    </servlet>
  }

  {
    <servlet-mapping>
      <servlet-name>HelloWorldServlet</servlet-name>
      <url-pattern>*.toutpourservlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
      <servlet-name>HelloWorldServlet</servlet-name>
      <url-pattern>/msg.hello</url-pattern>
    </servlet-mapping>
  }

  {
    <servlet-mapping>
      <servlet-name>HelloWorldPrintWriter</servlet-name>
      <url-pattern>/printwriter.html</url-pattern>
    </servlet-mapping>
  }
</web-app>
```

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 1. web.xml

- Utilisation de meta-caractères

```
...  
  <servlet-mapping>  
    <servlet-name>HelloWorldServlet</servlet-name>  
    <url-pattern>/HelloWorldServlet/msg.hello</url-pattern>  
  </servlet-mapping>  
  <servlet-mapping>  
    <servlet-name>HelloWorldServlet</servlet-name>  
    <url-pattern>*.toutpourservlet</url-pattern>  
  </servlet-mapping>  
  <servlet-mapping>  
    <servlet-name>HelloWorldServlet</servlet-name>  
    <url-pattern>/index.html</url-pattern>  
  </servlet-mapping>  
...
```

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 1. web.xml

Comment l'utiliser dans le cas d'un service WEB REST ?

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 2. Servlet

- Une Servlet doit hériter de HttpServlet
- Reçoit et traite les requêtes
 - ▶ Lit la méthode HTTP dans le paquet
 - ▶ Transmet à la méthode appropriée
- Squelette classique:

```
1
2 public class maServlet extends HttpServlet
3     public void doGet(HttpServletRequest req, HttpServletResponse res){
4         PrintWriter out = res.getWriter();
5         out.println("<HTML><BODY>");
6         out.println("Bonjour tout le monde");
7         out.println("</BODY></HTML>");
8     }
9 }|
10
```

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 2. Servlet

- HttpServlet fournit plusieurs **méthodes de traitement des requetes**
- Nom construit sur la meme base **doXXX(...)**
 - ▶ **doGet(...)**: pour les requêtes de type GET
 - ▶ **doPost(...)**: pour les requêtes de type POST
 - ▶ **doPut(...)**: pour les requêtes de type PUT
 - ▶ **doDelete(...)**: pour les requêtes de type DELETE
- Si non redéfinies, les méthodes doXXX(...) renvoie une erreur de type HTTP 405

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 2. Servlet

- Quelle que soit la méthode, on dispose toujours de deux types d'objets pour chaque requête
 - ▶ **Un objet de requête**: contexte de la requête (entete, parametre, url, navig, etc.) + info sur le client (navigateur, IP, etc.)
 - ▶ **Un objet de réponse**: qui permet de renvoyer des données au client (type de contenu, code de retour, données, etc.)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SampleServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        ...
    }

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        ...
    }
}
```


JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 2. Servlet

- Méthodes utiles sur l'objet requête: (<http://goo.gl/eiGhgK>)
 - ▶ `getMethod()`
 - ▶ `getParameter(String name)`
 - ▶ `getRequestURI()`
 - ▶ `getRequestURL()`
 - ▶ `getQueryString()`
 - ▶ `getParameterNames()`
 - ▶ ...
- Méthodes utiles sur l'objet reponse: (<http://goo.gl/LRlllh>)
 - ▶ `getWriter()`
 - ▶ `setStatus()`
 - ▶ `setContentType()`
 - ▶ `sendRedirect()`
 - ▶ ...


JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 2. Servlet et cycle de vie

- Exemple de servlet traitant les informations reçues

```
1
2 public class maServlet extends HttpServlet
3     public void doPost(HttpServletRequest req, HttpServletResponse res){
4         //Récupération des paramètres
5         String prenom = req.getParameter("prenom");
6         String nom = req.getParameter("nom");
7
8         //Renvoi des informations au clients
9         PrintWriter out = res.getWriter();
10        out.println("Bonjour "+prenom+" "+nom);
11    }
12 }
13
```



JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 2. Servlet et cycle de vie

- La servlet n'est créée qu'une fois, i.e. une seule instance
- A chaque fois que le serveur est relancé, le conteneur de servlet est réinitialisé et il y a création d'une nouvelle instance
- Cette instance traite toutes les requetes
- Avantages
 - ▶ Rapidité: pas cout lié à la création d'un nouvel objet à chaque requete
 - ▶ Possibilité de conserver des données entre les requetes

Exemple

```
public class SimpleCounterServlet extends HttpServlet {  
    private int count = 0;  
  
    protected void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/plain");  
        PrintWriter out = res.getWriter();  
        count++;  
  
        out.println("Depuis son chargement, on a accédé à cette Servlet " +  
            count + " fois.");  
    }  
}
```

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté serveur: 2. Servlet

Comment l'utiliser dans le cas d'un service WEB REST ?

JavaEE et JAX-RS

Framework Jersey

Coté client

- Difficulté: **Comment modifier la méthode HTTP?**
- **Habitués à deux types de méthodes ?**

Rappel PHP

Exemple PHP:

- ▶ `<form action="..." method="POST ou GET">`
...
`< /form>`

- **Problème:** Vous ne pouvez pas appeler **PUT** ou **DELETE** avec votre navigateur habituel
- Paradoxalement, les serveurs sont capables de gérer les appels a PUT ou DELETE
- Mais pas de procédure standard en HTML classique

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté client

- Plusieurs solutions:
 - ❶ Inventer une convention pour représenter PUT et DELETE
 - **Passage de paramètres**
Ex. /livre/SF/parrypotter/2/?action=maj ou supp
 - **Ajout dans l'url**
Ex. /maj/livre/SF/parrypotter/2
Ex. /supp/livre/SF/parrypotter/2
 - ❷ Construire entièrement la requête HTTP à envoyer
 - En utilisant la classe `URLConnection`
 - ❸ Utiliser une API capable de modifier, de façon transparente, les méthodes HTTP
 - JAX-RS: Jersey, RestLet, Spring, etc.

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté client: exemple GET

```
2 import java.io.*;
3 import java.net.*;
4 public class Client {
5     public static void main(String args[]) throws Exception{
6         //Adresse du service
7         URL url = new URL("http://localhost:8080/AppliWeb/");
8         HttpURLConnection conn = (HttpURLConnection) url.openConnection();
9         //Deinition de la méthode et des propriétés des de l'échange
10        conn.setRequestMethod("GET");
11        conn.setRequestProperty("Accept", "application/xml");
12        //Ouverture du flux
13        BufferedReader br = new BufferedReader(new InputStreamReader((conn.getInputStream())));
14        //Lecture des informations reçues du serveur
15        String output;
16        System.out.println("Output from Server .... \n");
17        while ((output = br.readLine()) != null) {
18            System.out.println(output);
19        }
20        conn.disconnect();
21    }
22 }
23
24
```

JavaEE et JAX-RS

Fonctionnement JavaEE

Coté client: exemple POST

```
1 import java.io.*;
2 import java.net.*;
3 public class Client {
4     public static void main(String args[]) throws Exception{
5         //Adresse du service
6         URL url = new URL("http://localhost:8080/AppliWeb/");
7         HttpURLConnection conn = (HttpURLConnection) url.openConnection();
8         //Definition de la méthode et des propriétés de l'échange
9         conn.setRequestMethod("POST");
10        conn.setDoInput(true);
11        conn.setDoOutput(true);
12        conn.setRequestProperty("Content-Type", "application/json");
13        //Parametres
14        String input = "organisation=UAG&section=Master2";
15        //Ouverture du flux et envoi des paramètres
16        OutputStream os = conn.getOutputStream();
17        os.write(input.getBytes());
18        os.flush();
19        //Lecture des informations reçues par le serveur
20        BufferedReader br = new BufferedReader(new InputStreamReader((conn.getInputStream())));
21        String output;
22        System.out.println("Output from Server .... \n");
23        while ((output = br.readLine()) != null) {
24            System.out.println(output);
25        }
26        conn.disconnect();
27    }
```


JavaEE et JAX-RS

Framework Jersey

JAX-RS:

- JAX-RS: Java API for RESTful Web Services
fournit un support pour la création de services WEB avec une architecture REST
- Il est défini dans la **JSR 311** (*Java Specification Requests*)
<http://goo.gl/sg1g6k>
- JAX-RS introduit un système de **d'annotations** pour la création de services
- Plusieurs implémentations:
 - ▶ **Jersey**
 - ▶ RESTEasy
 - ▶ Restlet
 - ▶ ...

JavaEE et JAX-RS

Framework Jersey

Jersey

- Implémentation de référence de JAX-RS
- Framework open-source développé par Oracle
- Fournit un ensemble de fonction pour implémenter des services WEB REST dans un conteneur de servlets
- Au niveau Serveur
 - ▶ Fournit une implémentation de Servlet qui parcourt automatiquement les classes pour identifier les ressources
 - ▶ Elle doit être définie dans le *web.xml*
- Au niveau Client
 - ▶ Fournit une API pour communiquer avec un service

JavaEE et JAX-RS

Framework Jersey

Jersey

- Mise en place du service s'effectue en deux étapes
 - ❶ Implémenter les classes qui répondent aux requêtes à l'aide des **annotations**
 - ❷ Définir le **Jersey Servlet Dispatcher**

JavaEE et JAX-RS

Framework Jersey

1) Annotations

- JAX-RS repose sur un système d'annotations
 - ▶ **@PATH(your-path)**: Définit le chemin à partir de l'URL de base (i.e. celle définie dans le *web.xml*)
 - ▶ **@GET**: Indique que la méthode qui suit traite les requêtes avec la méthode GET
 - ▶ **@POST**: Idem pour méthode POST
 - ▶ **@PUT**: Idem pour méthode PUT
 - ▶ **@DELETE**: Idem pour méthode DELETE
 - ▶ **@Produces**: Définit le type de représentation produit par la méthode
 - ▶ **@Consumes**: Définit le type de représentation accepté par la méthode
 - ▶ **@PathParam**: Utiliser pour récupérer paramètres dans l'URL

Attention

Il en existe bcp d'autres: @queryparam, @provider, ...

Qqs exemples d'utilisation ici: <http://goo.gl/MoRz3e>

JavaEE et JAX-RS

Framework Jersey

1) Annotations: Exemple @GET et @Produces

- Précise le type de représentation produite par le serveur

```
1 | @GET
2 | @Produces("application/xml")
3 | public Contact getXML() {
4 |     ...
5 | }

1 | @GET
2 | @Produces("application/json")
3 | public Contact getJSON() {
4 |     ...
5 | }
```

JavaEE et JAX-RS

Framework Jersey

1) Annotations: Exemple @PATH

- Précise l'URI de la ressource

```
1  @GET
2  @Produces("application/xml")
3  @Path("xml/{firstName}")
4  public Contact getXML() {
5      ...
6  }
```

- Exemple de templates d'URI

URI PathTemplate	URI After Substitution
http://example.com/{name1}/{name2}/	http://example.com/james/gatz/
http://example.com/{question}/ {question}/{question}/	http://example.com/why/why/why/
http://example.com/maps/{location}	http://example.com/maps/Main%20Street
http://example.com/{name3}/home/	http://example.com//home/

JavaEE et JAX-RS

Framework Jersey

1) Annotations: Exemple @PathParam

- Pour le récupération de parametre dans l'URL

```
1  @GET
2  @Produces("application/xml")
3  @Path("xml/{firstName}")
4  public Contact getXML(@PathParam("firstName") String firstName) {
5      Contact contact = contactService.findByFirstName(firstName);
6      return contact;
7  }

1  @GET
2  @Produces("application/json")
3  @Path("json/{firstName}")
4  public Contact getJSON(@PathParam("firstName") String firstName) {
5      Contact contact = contactService.findByFirstName(firstName);
6      return contact;
7  }
```

JavaEE et JAX-RS

Framework Jersey

2) Jersey Servlet Dispatcher

- Parcourt les classes d'un package pour identifier les ressources
- Fait le lien entre la requête et la classe
- Initialisé dans le fichier *web.xml*
- Le paramètre ***com.sun.jersey.config.property.package*** définit dans quel package le dispatcher doit rechercher les classes

```
1 </web-app>
2   <servlet>
3       <servlet-name>jerseyDispatcher</servlet-name>
4       <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
5       <init-param>
6           <param-name>jersey.config.server.provider.packages</param-name>
7           <param-value><b>PACKAGE CONTENANT VOS CLASSES</b></param-value>
8       </init-param>
9   </servlet>
10
11   <servlet-mapping>
12       <servlet-name>jerseyDispatcher</servlet-name>
13       <url-pattern>/*</url-pattern>
14   </servlet-mapping>
15 </web-app>
```


JavaEE et JAX-RS

Ressources

WEB Services:

- SOAP vs REST: Choisir la bonne architecture web service
<http://goo.gl/MrRfCw>
- Comprendre le style d'architecture REST
<http://goo.gl/MKtKBp>
- Comment j'ai expliqué REST à ma femme
<http://goo.gl/o1o2dc>
- Apprendre REST - un style d'architecture du Web
<http://goo.gl/30gJir>
- Pour ne plus être en REST, comprendre cette architecture
<http://goo.gl/M3ISvg>
- L'architecture orientée ressource pour faire des services web RESTful
<http://goo.gl/kZ7qpB>

JavaEE et JAX-RS

Ressources

Sur l'utilisation de JAX-RS:

- Building RESTful Web Services with JAX-RS
<http://goo.gl/WVFhfp>
- REST with Java (JAX-RS) using Jersey - Tutorial
<http://goo.gl/dK4iM>
- JAX-RS: REST coté serveur avec JAVA
<http://goo.gl/X8uOZ>
- Développer des services web REST avec JAVA: JAX-RS
<http://goo.gl/4Tzgh8>
- JAX-RS, le spécification Java pour implémenter les services REST
<http://goo.gl/7Y24UI>
- Building a RESTful Web Service with Spring Framework
<http://goo.gl/LdfyjL>
- RESTful Web Service - JAX-RS Annotations
<http://goo.gl/MoRz3e>