REST : Representational State Transfer

Abdelhak-Djamel Seriai

Page web versus Service Web

- Page web
  - Accès à des données via des pages Hypertext
  - Ces données sont « mélangées » avec des données liées à la présentation de ces données
    - CSS, HTML
  - Destinée à être exploitée « consommée» par des êtres humain
    - Exemple : Facebook et Tweeter

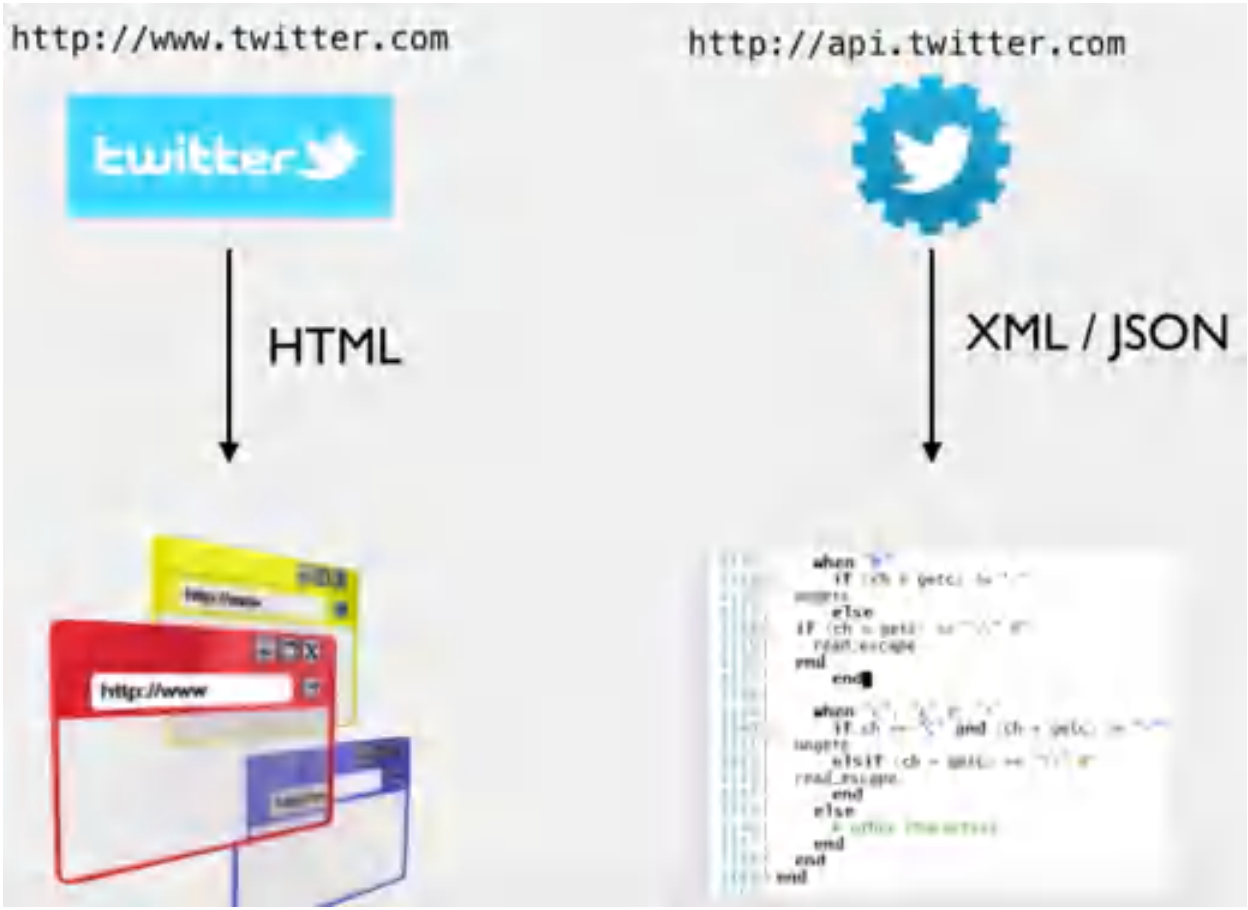
Page web versus Service Web

Page web versus Service Web

- Service Web
  - Est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués.
  - Service exposé sur internet pour un accès programmatique (via des programmes) via des API en lignes
    - Les fournisseurs des Web services publient ces services,(les mettent en accès en ligne).
    - Les clients utilisent (consomment ces services en accédant aux données rendues disponibles.
      - Exemple de Facebook et de Tweeter
  - Indépendant des plateformes
  - Indépendant des langages

Page web versus Service Web

- Service Web



Caractéristiques des services web

- HTTP : Hyper Text Transfer Protocol
  - Hyper Text : Des textes avec des Hyper Links; des textes qui références d'autres textes
- Architecture client– serveur
- Protocole : Format des messages échangés entre le client et le serveur
  - Par exemple SOAP : format spécifique des messages échangés
- La définition des services
  - Nom du service, type des données retournées, types de paramètres en entrées :
    - l'API-SOAP : WSDL

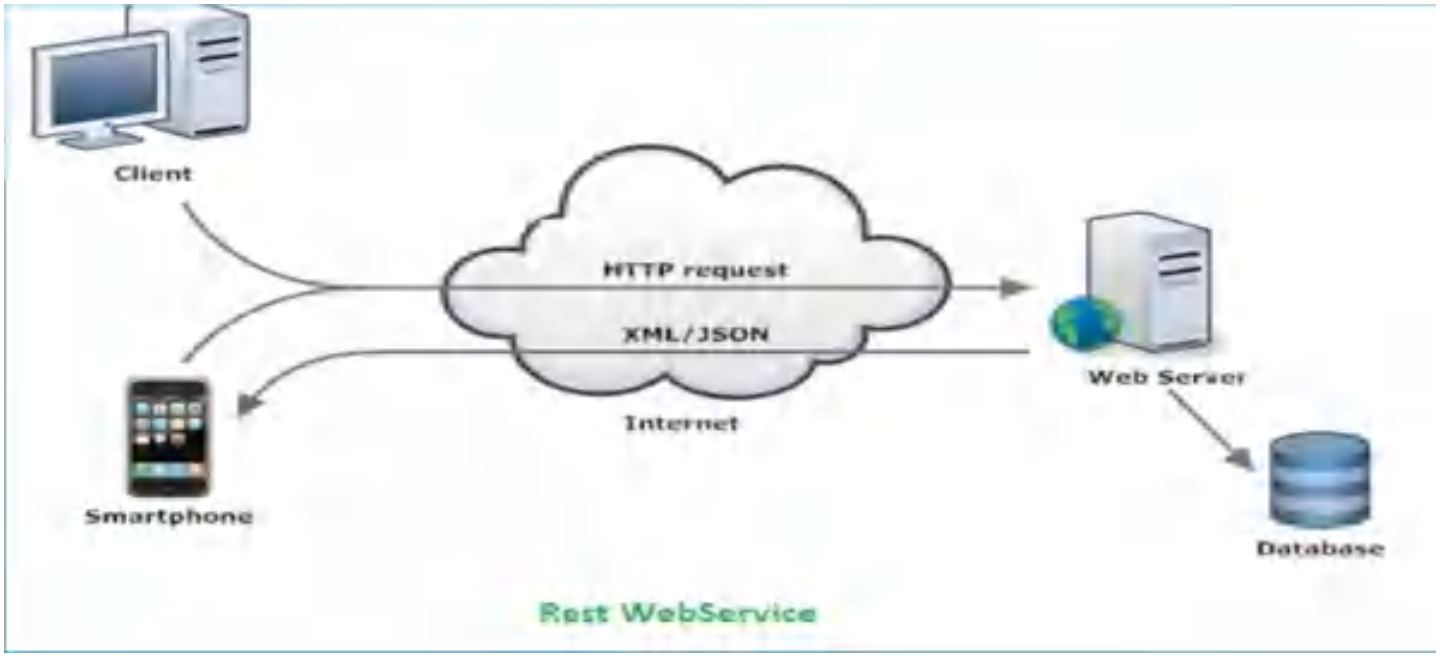
Inconvénients des premiers modèles des Services web

- HTTP est utilisé uniquement comme moyen de transport.
  - Les seuls messages utilisés de HTTP sont GET et POST.
  - Chaque Service web dispose d'une interface spécifique, encapsulé directement dans HTTP
    - Dans SOAP, décrite en XML par le langage WSDL.
- Architecture SOA : la mise en œuvre réelle est complexe, les normes volumineuses et difficiles à maîtriser.

SERVICE REST

- Créé en 2000 par Roy Fielding - Thèse de doctorat
  - Projet Waka
  - Principal auteur de la spécification HTTP
  - Membre fondateur de la fondation Apache
- REST est l'acronyme de Representational State Transfer
- Créé pour les systèmes distribués
  - Appliqué aux web services = RestFull web application
- Est un style architectural réseau
  - Un ensemble de guides
    - Client / Serveur
    - Utilise le Protocole HTTP pour l'échange de données
      - Met l'accent sur la définition de ressources identifiées par des Urs
      - Utilise les messages du protocole HTTP pour définir la sémantique de la communication client/serveur
    - Stateless – Sans état
  - REST n'est pas un standard , pas de spécifications de la W3C

Architecture REST



REST Aujourd'hui

Les concepts clés de l'API REST

REST et HTTP

- Inventeur de REST est un des auteurs de la spécification de HTTP
- Pas de protocole
  - Différents format de données échangées : XML, JSON, Text (Mime type)
- Pas de règles strictes pour l'utilisation des méthodes HTTP
  - Plutôt des guides et conventions
    - Rien n'oblige d'utiliser l'ensemble des méthodes
- Pas de définition de service
  - Il peut exister de la documentation sous format consultable par les êtres humains pour les services (une page web) mais pas destinées à être exploitée par un programme
- Pas de spécification de REST : c'est un concept (au contraire de SOAP : spécification W3C)

Avantages et inconvénients de REST

- Avantages
  - Facile à comprendre et à implémenter
    - Framework dans plusieurs langages : Java – Python – Php
  - Un client HTTP suffit pour accéder à un service RESTful.
  - Interopérabilité des langages
  - Architecture scalable : Possibilité de répartir les requêtes sur plusieurs serveurs
    - Conséquence de « stateless ».
  - L'utilisation de formats standards comme JSON ou XML assure la compatibilité dans le temps.
- Inconvénients
  - La sécurité est inexistante – Utilisation d'HTTPS + Authentification
  - Le client doit conserver des données localement (stateless)

Les principaux éléments de HTTP pour REST

- Localisation (identification) des ressources
  - Adresse d'un élément (d'une entité)
    - Adresse d'une page web: adresse qui n'est pas basée sur l'identification des ressources mais « action-based »:
      - http://www.lirmm.fr/~seriai/index.php?n=Main.Software
      - http://weatherapp.com/weatherLookup.do?zipcode=12345
    - URI basée sur l'identification des ressources (Ressource based URI) :
      - » http://weatherapp.com/zipcode/12345
      - » http://weatherapp.com/zipcode/56789
      - » http://weatherapp.com/countries/brazil
      - » http://free-web-services.com/web-services/geo/weather/
      - » https://developer.worldweatheronline.com/api/docs/

Les principaux éléments de HTTP pour REST

- Les méthodes HTTP
  - GET, POST, PUT, DELETE
    - Un service web bien conçu doit être se basé sur l'ensemble de ces méthodes
  - Les méta données:
    - Header (Header response)
    - HTTP Status Codes
      - Important pour comprendre la nature des données reçues et agir en fonction
        - » 200 : success
        - » 500 : server error
        - » 404 : Not found
      - Format de message
        - » Content Types : pour savoir quel type de données
          - XML, JSON, Text
        - » Un serveur peut retourner les mêmes données sous plusieurs formats : XML, JSON, etc.
          - Le client peut décider du format de réception des données souhaité (Content negotiation)

Les étapes pour créer un service web REST

- Identifier et concevoir les URIs des ressources

  - Les URI sont utilisées pour les applications web
    - Pour une page web : l'utilisateur n'a pas besoin de connaître tous les URIs pour accéder aux pages web
      - Besoin uniquement de savoir l'URI de la page principale (home Page) et après il navigues aux autres pages via les liens hypertext

    - Pour REST : besoin d'accéder directement aux URIs dans les applications
      - besoin d'une convention de définition de ces URIs

  - Convention pour l'identification des ressources:
    - Chaque entité accessible doit être définie via une URI unique
    - Resource based URI
      - Des noms et non des verbes
        - » Exemples : document, messages, profile et pas getMessage, getDocument

      - Imbrication des noms de ressources : dossier et sous dossier et enfin la ressource
        - » /profiles/{profileNames}
        - » /messages/{messageId}
        - » /messages/1
        - » /messages/2

Les étapes pour créer un service web REST

- Identifier et concevoir les URIs des ressources :
  - Convention pour l'identification des ressources:
    - Expliciter les relations entre ressources
      - Exemples : Les commentaires sur un message donné
        - » /messages/1/comments/2
        - » /messages/{messageId}/comments/{commentId}
        - » /messages/15/comments/4
        - » /messages/{messageId}/likes/{likeId}
    - Distinguer les ressources uniques et les collections de ressources
      - Exemples:
        - » /messages : tous les messages
        - » /messages/15/comments : tous les commentaires du message 15.
        - » /messages/comments/
        - » /comments/
        - » profiles/messages/comments/
    - Utilisation de paramètres de filtrage et de pagination
      - Filtrer les résultats désirés
        - » Exemples:
          - /messages?offset=30&limit=10
          - /messages?year=2014

Les étapes pour créer un service web REST

- Spécifier les méthodes HTTP correspondant aux URIs
  - Les méthodes HTTPs
    - Les plus communes : GET, POST, PUT, DELETE
      - opérations CRUD
    - Les moins communes : HEAD, OPTIONS

  - GET : récupérer les données liées à une ressources ou une collection de ressources (une méthode Read-only)
    - Exemples :
      - /messages/20 :
        - » retourner le message 20
      - /messages/20/comments
        - » retourner tous les commentaires associé du message 20
      - /messages
        - » retourner tous les messages

Les étapes pour créer un service web REST

- Spécifier les méthodes HTTP correspondant aux URIs
  - POST : créer une ressource ou une collection de ressources (une méthode d'écriture)
    - Exemple :
      - /messages
        - » créer une nouveau message
      - /messages/20/comments
        - » créer un nouveau commentaire associé au message 20
  - PUT : remplacer une ressource (une méthode d'écriture)
    - /messages/20 : remplacer le message dont l'id est 20
    - /messages/20/comments/10 : remplacer le commentaire dont l'id est 10
    - /messages/20/comments : remplacer tous les commentaires du message 20
  - DELETE : supprimer une ressource (une méthode d'écriture)
    - /messages/20 : supprimer le message dont l'id est 20
    - /messages/20/comments/10 : supprimer le commentaire dont l'id est 10
    - /messages/20/comments : supprimer tous les commentaires du message 20

Les étapes pour créer un service web REST

- Spécifier les méthodes HTTP correspondants aux URIs
  - les méthodes GET, PUT et DELETE sont « idempotent »
    - Une même requête peut être invoquée plusieurs fois sans effet supplémentaire côté serveur par rapport à la première invocation
      - GET —> /messages/20 : retourner le message 20
      - GET —> /messages/20 : retourner le message 20
      - GET —> /messages/20 : retourner le message 20
      - PUT—> /messages/20 : remplacer le message dont l'id est 20
      - PUT—> /messages/20 : remplacer le message dont l'id est 20
      - PUT—> /messages/20 : remplacer le message dont l'id est 20
      - DELETE—> /messages/20 : supprimer le message dont l'id est 20
      - DELETE—> /messages/20 : supprimer le message dont l'id est 20
      - DELETE—> /messages/20 : supprimer le message dont l'id est 20
  - La méthode POST est « non-Idempotent »
    - La même requête invoquée plusieurs fois a un effet différent côté serveur à chaque invocation
      - POST —> /messages : créer une nouveau message —> /messages/21
      - POST —> /messages : créer une nouveau message —> /messages/22
      - POST —> /messages : créer une nouveau message —> /messages/23

Les étapes pour créer un service web REST

- Spécifier la réponse REST
  - Plusieurs formats de réponse sont possibles pour REST : JSON, TEXT, XML, etc.
    - C'est pourquoi on parle de REPRESENTAL
  - Plusieurs formats de réponse peuvent être possibles pour une même requête
  - Pour indiquer le type de retour d'une réponse : utilisation de la partie Headers d'une requête HTTP (l'autre partie est Message Body).
    - Headers définit : Message length, Date, et Content Type.

Les étapes pour créer un service web REST

- Spécifier le code Status (Status Codes)
  - Les codes sont classés en 5 catégories : de 1XX à 5XX
    - 1XX : code informationnel
    - 2XX success
      - 200 OK
      - 201 : created
      - 204 : No Content
    - 3XX : Redirection
      - 302 Found
      - 304 : Not Modified
      - 307 : Tempororary Rederict
    - 4XX : Client Error
      - 400 Bad request
      - 401 : Unauthorized
      - 403 : Forbidden
      - 404 : Not Found
      - 415 : Unsupported Media Type
    - 5XX : server Error
      - 500 : Internal Server Error

HATEOAS: Hypermedia As The Engine Of Application State

- C'est une manière de pallier à l'absence de définition de service
  - Problème : le client doit connaître tous les URIs pour accéder aux ressources
    - Dans une page web, il suffit de connaître la page racine et par navigation en accède aux autres pages
    - Faire pareil pour les réponse REST
      - Exemple JSON

```
{
« id »: « 20 »,
« message »: « hello world »,
« date » : « 24Apr2016 »,
« author » : « seriai »,
« href »: « /messages/1 »,
« comments-href »: api/messages/20/comments »,
« likes-href » : « api/messages/20/likes » »,
« shares-href »: « api/messages/20/shares »,
« profile-href »: « /messages/1/comments »
}
```

HATEOAS: Hypermedia As The Engine Of Application State

```
{
« id »: « 20 »,
« message »: « hello world »,
« date » : « 24Apr2016 »,
« author » : « seriai »,
« links » : [
        {
        « href »: « /messages/1 »,
        « rel » : « self »
        }
        {
        « href »: « comments-href »: « /messages/20/comments »,
        « rel » : « comments »
        }
        {
        « href »: « likes-href » : «/messages/20/likes » »,
        « rel » : « likes »
        }
        {
        « href »: « shares-href »: «/messages/20/shares »,
        « rel » : « shares »
        }
        {
        « href »: « profile-href »: « /messages/1/comments »
        « rel » : « author »
        }
]
```

Richardson Maturty Model

- Est une classification des applications REST en fonction de leur respect du style REST
  - Niveau 0 :
    - Style qui n'est pas RestFull
      - PlainXML: pas de ressource URI ;
        - » par exemple SOAP : tout est dans le même URI et tout est dans le le message
  - Niveau 1 : utilisation de URI
  - Niveau 2 : utilisation des méthodes HTTP
  - Niveau 3: HATEOAS

WADL : Web Application Description Language

- Standard du W3C : spécification W3C initié par SUN (www.w.org/Submission/wadl)
- Est un langage de description des services REST au format XML.
- Il fournit les informations descriptives d'un service permettant de construire des applications clientes exploitant les services REST.
- Il décrit les éléments à partir de leur type (Ressources, Verbes, Paramètre, type de requête, Réponse).
- Permet d'interagir de manière dynamique avec les applications REST.
- Moins exploité que le WSDL pour les Services SOAP

WADL : Web Application Description Language

Principes d'implémentation d'une API REST

- Définition des ressources manipulées:
  - Collection de ressources(liste)
  - Ressource unique.
- Codage de la représentation des ressources
  - Quels sont les attributs d'une ressource ?
  - Quel format à utiliser ?
- Sémantique des messages
  - Les actions possibles sur les ressources sont indiquées par les messages du protocole detransport, ce qui donne pour HTTP :
    - GET : récupération de la représentation d'une ressource ou d'une liste de ressource.
    - PUT : mise à jour d'une ressource existante, création d'une ressource en spécifiantl'URI de la ressource.
    - POST : création d'une sous ressource (le serveur décide de l'URI), ajoutd'information à une ressource existante.
    - DELETE : effacement.
    - HEAD : informations sur une ressource.
  - Une ressource donnée ne sera pas obligatoirement manipulable par tous les messages.
    - Par exemple, une ressource accessible en lecture seulement peut n'être accessible que par les messages de type GET.

Définir une API REST en .NET

```
namespace testRest3.Controllers
{
    [Route("api/[controller]")]
    public class ValuesController : Controller
    {
        // GET api/values
        [HttpGet]
        public IEnumerable<string> Get()
        {
            return new string[] { "value1", "value2" };
        }

        // GET api/values/5
        [HttpGet("{id}")]
        public string Get(int id)
        {
            return "value";
        }

        // POST api/values
        [HttpPost]
        public void Post([FromBody]string value)
        {
        }

        // PUT api/values/5
        [HttpPut("{id}")]
        public void Put(int id, [FromBody]string value)
        {
        }

        // DELETE api/values/5
        [HttpDelete("{id}")]
        public void Delete(int id)
        {
        }
    }
}
```

Définir une API REST en JAVA

- MÉTHODES HTTP : @GET, @POST, @PUT, @DELETE
  - L'annotation des méthodes Java permet de traiter des requêtes HTTP suivant le type de méthode (GET,POST..)
  - Opération CRUD sur les ressources
  - Annotation disponibles :@GET, @POST, @PUT, @DELETE et @HEAD
    - Uniquement utilisable sur des méthodes Java et non sur des classes
    - Le nom de la méthode importe peu.
      - C'est l'annotation qui importe et qui permet d'aiguiller la requête.

```
@GET
@Path(« author/{author} »)
public void
getByAuthor(@PathParam(«
author » String author){
// Do Something }
```

Conclusion

- SOAP
  - Avantages
    - Standardisé
    - Interopérabilité
    - Sécurité (WS-Security)
  - Inconvénients
    - Performances (enveloppe SOAP supplémentaire)
    - Complexité, lourdeur
- REST
  - Avantages
    - Simplicité de mise en œuvre
    - Lisibilité par un humain
    - Evolutivité
    - Repose sur les principes du web
    - Représentations multiples (XML, JSON,…)
  - Inconvénients
    - Sécurité restreinte par l'emploi des méthodes HTTP