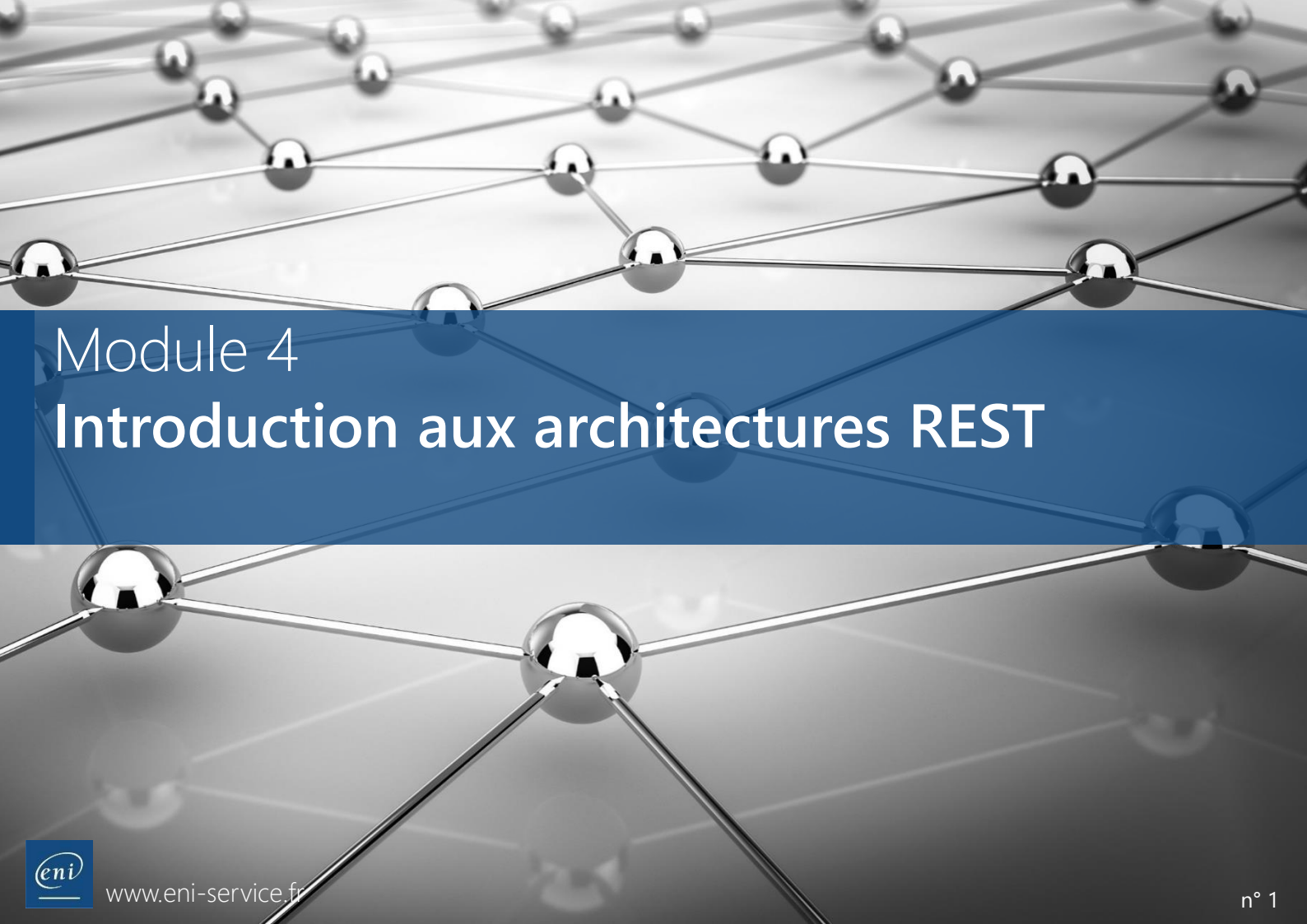

Module 4

Introduction aux architectures REST

Contenu du module

- Les concepts fondamentaux de REST (REpresentational State Transfer)
- Utilisation de REST dans une architecture informatique.
- Architecture client/serveur HTTP.
  - Structure d'une requête et d'une réponse HTTP.
- L'identification des ressources par URI.
- SOAP vs. REST
- Les formats de données utilisables
  - XML, HTML, JSON …

Introduction

- Les protocoles de communication Web sont un élément clé de l'infrastructure Web 2.0. Les deux approches principales sont REST et SOAP.
  - **REST** (*REpresentational State Transfer*) indique une façon d'échanger et de manipuler des données en utilisant simplement les *verbes* HTTP GET, POST, PUT et DELETE.
  - **SOAP** implique de poster à un serveur des requêtes XML comprenant une suite d'instructions à exécuter.
- Dans les deux cas, les accès aux services sont définis par une interface de programmation (API).
  - Souvent, l'interface est spécifique au serveur.
- Cependant, des interfaces de programmation Web standardisées (par exemple, pour poster sur un blog) sont en train d'émerger. La plupart, mais pas toutes, des communications avec des Services Web impliquent une transaction sous forme XML (*eXtensible Markup Language*).

Principes de REST

- Cette architecture part du principe selon lequel Internet est composé de ressources accessibles à partir d'une URL.
  - Par exemple, pour avoir le temps à Paris, un utilisateur pourrait utiliser une adresse de la forme http://www.meteo.fr/paris : Paris serait alors une ressource telle que définie par Météo France.
- A la requête de cet URL serait renvoyée une représentation de la ressource demandée (ex : paris.jsp). Cette représentation place l'application cliente dans un état (*state*) donné.
- Si l'application cliente lance un appel sur un des liens de la représentation en cours, une autre ressource est appelée, dont une représentation est envoyée. Ainsi, l'application cliente change d'état (*state transfer*) pour chaque représentation de ressource.

Caractéristiques de REST

- Utilise les standards de l'Internet
  - Architecture orientée ressource basée sur des URI (*Uniform Resource* Identifier)
  - Repose sur l'utilisation du protocole HTTP et de ses méthodes (POST, GET, PUT, DELETE)
- Echange de données multi-formats
  - en XML ou autres (JSON, objets sérialisés en binaire)
- Utilisation des standards hypermedia : HTML ou XML qui permettent de faire des liens vers d'autres ressources et d'assurer ainsi la navigation dans l'application REST
- Utilise les types MIME pour la représentation des ressources (text/xml, image/jpeg, application/pdf, text/html, ...)
- Implémentation restreinte : il faut comprendre l'architecture REST et ensuite concevoir des applications ou des services Web selon cette architecture
- Alternative à SOAP censée être plus simple
- Aucun langage standard de description de service
- N'est pas un standard

Principe de fonctionnement de REST

- Les méthodes du protocole HTTP servent à définir le type traitement à effectuer

Action	SQL	HTTP
Create	Insert	PUT
Read	Select	GET
Update	Update	POST
Delete	Delete	DELETE

- Donc pour l'URI `/client/123`, l'action sur cette ressource est décidée par la méthode HTTP utilisée.
  - Ce ne sont cependant que des conventions !
  - L'implémentation est à réaliser par le programmeur.

Règles de conception d'un Service REST

- Toutes les ressources devant être exposées au travers du service doivent être correctement identifiées, et de manière unique. Chaque ressource devra se voir assigner une URL.
  - Qui plus est, l'URL en question devra être de la forme http://www.site.com/contenus/003 plutôt que http://www.site.com/contenus?id=003.
- Les ressources doivent être catégorisées selon leurs possibilités offertes à l'application cliente
  - Ne peut-elle que recevoir une représentation (GET) ?
  - Ou bien peut-elle aussi modifier/créer une ressource (POST, PUT, DELETE) ?
- Chaque ressource devrait faire un lien vers les ressources liées
- La manière dont fonctionne le service sera décrite au sein d'un simple document HTML
  - Qui servira d'interface et de documentation d'API

Exemple de Service REST

- Prenons une entreprise de jouets qui veut permettre à ses clients d'obtenir une liste des jouets disponibles à la vente et d'obtenir des informations sur un jouet précis.
- La liste des jouets est disponible à l'URL suivante : http://www.jktoys.com/jouets
- Le client reçoit une réponse sous la forme suivante :

```
<?xml version="1.0"?>
<p:Jouets xmlns:p="http://www.jktoys.com/"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <Jouet id="0001"
xlink:href="http://www.jktoys.com/jouets/0001"/>
    <Jouet id="0002"
xlink:href="http://www.jktoys.com/jouets/0002"/>
    <Jouet id="0003"
xlink:href="http://www.jktoys.com/jouets/0003"/>
[...]
</p:Jouets>
```

- La liste des jouets contient des liens pour obtenir des informations sur chaque jouet. C'est là la clef de REST : le lien entre les ressources. Le client peut ensuite choisir parmi les liens proposés pour aller plus loin.

Exemple de Service REST - suite

- Les détails d'un jouet se trouvent à l'URL : http://www.jktoys.com/jouets/0002
- Ce qui renvoi la réponse :
  - ```
    <?xml version="1.0"?>
    <p:Jouet xmlns:p="http://www.jktoys.com"
    xmlns:xlink="http://www.w3.org/1999/xlink">
        <Jouet-ID>0002</Jouet-ID>
        <Nom>Bisounours : Gros Câlin</Nom>
        <Description>Coeur sur le ventre</Description>
        <Details xlink:href=
    "http://www.jktoys.com/jouets/0002/details"/>
        <CoutUnitaire monnaire="EUR">30</CoutUnitaire>
        <Quantite>37</Quantite>
    </p:Jouet>
    ```
- A nouveau, d'autres ressources sont accessibles grâce à un lien...
    ```

Format JSON

```
[{
   "id": 1, "nom": "Troadec", "prenom": "Nolwenn", "comptes": [{
    "id":1, "numero":"12345678", "intitule":"Compte courant", "operations":[...]
   },{
    "id":2, "numero":"23456789", "intitule":"Livret A", "operations":[{
     "date":"2015-01-03", "libelle":"Intérêts 2014", "montant":125.68, "type":"Credit"
    },{
     "date":"2015-03-14", "libelle":"Placement", "montant":300, "type":"Credit"
    }]
   }]
  },{
   "id": 2, "nom": "Leblanc", "prenom": "Marc"
  },{
   "id": 3, "nom": "Durand", "prenom": "Marie"
  },{
   "id": 4, "nom": "Meyer", "prenom": "Nils"
  },{
   "id": 5, "nom": "Lebreton", "prenom": "Louann"
  }]
```

L'essentiel de REST : HTTP

- Comprendre les mécanismes du protocole HTTP est essentiel pour comprendre le fonctionnement des architectures REST !
- La requête !
  - Les méthodes/verbes HTTP
    - GET, POST, PUT, DELETE
    - Leur signification, différences, usages et objectifs dans une API REST
  - L'identification unique des ressources
    - Associée à une méthode/un verbe HTTP !
- La réponse !
  - Les codes de réponse !
    - 1xx, 2xx, 3xx, 4xx, 5xx !
    - Pour savoir les utiliser à bon escient !

Quand utiliser SOAP ou REST ?

- Utiliser SOAP quand :
  - Un contrat doit être définit entre le fournisseur de service et les clients via l'utilisation de WSDL
  - Il devient nécessaire d'introduire des notions de transaction, de gestion d'état, de compression, de cryptage, …
  - Le client souhaite invoquer le service de manière asynchrone en plus des appels synchrones

- Préférer REST si :
  - Le fournisseur de service et les clients ont une connaissances mutuelle des informations à échanger et de leurs sémantiques
  - La bande passante est limitée
  - Les données doivent être agrégées dans un site Web via AJAX