M1 SII- USTHB

2024-2025

E-commerce et services web

SERIE TP /TD WADL

Un fichier WADL (Web Application Description Language) est un format XML utilisé pour décrire les services RESTful. Il permet de spécifier les ressources d'une API, les méthodes HTTP disponibles (GET, POST, PUT, DELETE), et les paramètres associés à chaque méthode.

1. Exercice 1 :

1-1 Lecture d’un fichier WADL et extraction des ressources

- Soit le fichier WADL suivant pour une API REST. Identifiez les différentes ressources (path) et les méthodes HTTP associées.

Voici un exemple de fichier WADL pour une API REST fictive :

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2009/02"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <resources base="http://example.com/api/">
    <!-- /users resource -->
    <resource path="users">
         <!-- GET /users --><method name="GET">
        <response>
          <representation mediaType="application/json"/>
        </response>
      </method>
   <!-- POST /users -->
      <method name="POST">
        <request>
          <representation mediaType="application/json">
            <param name="name" style="plain" type="xs:string"
required="true"/>
            <param name="email" style="plain" type="xs:string"
required="true"/>
          </representation>
        </request>
        <response>
          <representation mediaType="application/json"/>
        </response>
```

```
      </method>

      <!-- /users/{id} resource -->
      <resource path="{id}">
        <param name="id" style="template" type="xs:int" required="true"/>
        <method name="GET">
          <response>
            <representation mediaType="application/json"/>
          </response>
        </method>
      </resource>

    </resource>
  </resources>
</application>
```

Le fichier WADL inclue les trois endpoints :

1. **POST** `/users` – Ajouter un utilisateur
2. **GET** `/users` – Récupérer la liste des utilisateurs
3. **GET** `/users/{id}` – Récupérer un utilisateur spécifique par son identifiant

Dans cet exemple, nous avons deux ressources :

- `/users` avec les méthodes `GET` et `POST`
- `/users/{id}` avec la méthode `GET`

1. 2 : Interagir avec l'API en utilisant les méthodes du WADL

- Utilisez l'outil `curl` pour interagir avec les ressources définies dans le fichier WADL ci-dessus.
- Faites une requête GET pour récupérer les utilisateurs et une autre pour récupérer un utilisateur spécifique (en remplaçant `{id}` par un ID valide).

Solution :

1. Pour récupérer la liste des utilisateurs (méthode GET sur `/users`) :

- `curl -X GET http://example.com/api/users`

- Pour récupérer un utilisateur spécifique (méthode GET sur `/users/{id}`), remplacez `{id}` par un identifiant valide :

- `curl -X GET http://example.com/api/users/123`
- Pour ajouter un nouvel utilisateur (méthode POST sur `/users`), créez un fichier `new_user.json` avec le contenu suivant :

```
{
  "name": "AMIN SAIDI",
  "email": "amin.saidi@example.com"
}
```

Puis, utilisez la commande suivante :

3. `curl -X POST http://example.com/api/users -H "Content-Type: application/json" -d @new_user.json`

Exercice 2 : Créer un fichier WADL pour une API REST simple

Objectif : Créez un fichier WADL pour une API REST qui gère des ressources de type "produits".

Énoncé :

- Créez un fichier WADL pour une API REST qui permet de gérer les produits.
- Les ressources disponibles seront :
  - `/products` : Liste des produits (GET), ajout d'un produit (POST)
  - `/products/{id}` : Détails d'un produit spécifique (GET), mise à jour d'un produit (PUT), suppression d'un produit (DELETE)

Solution :

- **base** : l'URL de base de votre API (ex. `http://example.com/api`)
- **/products** :
  - `GET` pour obtenir la liste des produits.
  - `POST` pour ajouter un nouveau produit.
- **/products/{id}** :
  - `GET` pour récupérer un produit spécifique.
  - `PUT` pour le modifier.
  - `DELETE` pour le supprimer.
- Les représentations utilisent ici `application/json`.

🔹 `/products` (GET)

Récupérer la liste des produits

Requête :

```
GET /products HTTP/1.1
Accept: application/json
```

Réponse (200 OK) :

```
[
  {
    "id": "1",
    "name": "Ordinateur portable",
    "price": 999.99,
    "stock": 15
  },
  {
    "id": "2",
    "name": "Souris sans fil",
    "price": 25.50,
    "stock": 100
  }
]
```

`/products` (POST)

Ajouter un nouveau produit

Requête :

```
POST /products HTTP/1.1
Content-Type: application/json

{
  "name": "Clavier mécanique",
  "price": 79.99,
  "stock": 30
}
```

Réponse (201 Created) :

```
{
  "id": "3",
  "name": "Clavier mécanique",
  "price": 79.99,
  "stock": 30
}
```

`/products/{id}` (GET)

Récupérer un produit spécifique

Requête :

```
GET /products/1 HTTP/1.1
Accept: application/json
```

Réponse (200 OK) :

```
{
  "id": "1",
  "name": "Ordinateur portable",
  "price": 999.99,
  "stock": 15
}
```

`/products/{id}` (PUT)

Mettre à jour un produit

Requête :

```
PUT /products/1 HTTP/1.1
Content-Type: application/json

{
  "name": "Ordinateur portable - Nouvelle version",
  "price": 1099.99,
  "stock": 10
}
```

Réponse (200 OK) :

```
{
  "id": "1",
  "name": "Ordinateur portable - Nouvelle version",
  "price": 1099.99,
  "stock": 10
}
```

`/products/{id}` (DELETE)

Supprimer un produit

Requête :

```
DELETE /products/1 HTTP/1.1
```

Réponse (204 No Content) :

```
<aucun contenu>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2009/02"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <resources base="http://example.com/api">

        <!-- /products -->
        <resource path="products">

            <!-- GET : liste des produits -->
            <method name="GET">
                 <response>
                    <representation mediaType="application/json"/>
                </response>
            </method>

            <!-- POST : ajout d’un produit -->
            <method name="POST">
                  <request>
                    <representation mediaType="application/json"/>
                </request>
                <response status="201">
                    <representation mediaType="application/json"/>
                </response>
            </method>

        </resource>

        <!-- /products/{id} -->
        <resource path="products/{id}">
            <param name="id" style="template" type="xs:string" required="true"/>

            <!-- GET : détails d’un produit -->
            <method name="GET">
                 <response>
                    <representation mediaType="application/json"/>
                </response>
            </method>

            <!-- PUT : mise à jour d’un produit -->
            <method name="PUT">
                 <request>
                    <representation mediaType="application/json"/>
                </request>
                <response status="200">
                    <representation mediaType="application/json"/>
                </response>
            </method>

            <!-- DELETE : suppression d’un produit -->
            <method name="DELETE">
                <!--
                Réponse : 204 No Content
                -->
                <response status="204"/>
            </method>

        </resource>
```

```
    </resources>
</application>
```

Exercice 3 : Modifier un fichier WADL pour ajouter une nouvelle ressource

Objectif : Ajoutez une nouvelle ressource à un fichier WADL.

Énoncé :

- Modifiez le fichier WADL existant (par exemple, celui de l'exercice pour les produits) en ajoutant une nouvelle ressource `/categories` qui permet de gérer les catégories de produits.
- La ressource `/categories` doit avoir les méthodes suivantes :
    - `GET` pour lister toutes les catégories de produits
    - `POST` pour ajouter une nouvelle catégorie
    - `GET /categories/{id}` pour obtenir les détails d'une catégorie
    - `PUT /categories/{id}` pour mettre à jour une catégorie
    - `DELETE /categories/{id}` pour supprimer une catégorie

Solution :

Voici la modification du fichier WADL pour ajouter la ressource `/categories` :

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2009/02">
  <resources base="http://example.com/api">
    <resource path="/products">
      <method name="GET">
        <response status="200">
          <representation mediaType="application/json"/>
        </response>
      </method>
      <method name="POST">
        <request>
          <representation mediaType="application/json"/>
        </request>
      </method>
    </resource>
    <resource path="/products/{id}">
      <method name="GET">
        <response status="200">
          <representation mediaType="application/json"/>
        </response>
      </method>
      <method name="PUT">
        <request>
          <representation mediaType="application/json"/>
        </request>
      </method>
```

```
      <method name="DELETE">
        <response status="204"/>
      </method>
    </resource>
    <!-- Nouvelle ressource catégories -->
    <resource path="/categories">
      <method name="GET">
        <response status="200">
          <representation mediaType="application/json"/>
        </response>
      </method>
      <method name="POST">
        <request>
          <representation mediaType="application/json"/>
        </request>
      </method>
    </resource>
    <resource path="/categories/{id}">
      <method name="GET">
        <response status="200">
          <representation mediaType="application/json"/>
        </response>
      </method>
      <method name="PUT">
        <request>
          <representation mediaType="application/json"/>
        </request>
      </method>
      <method name="DELETE">
        <response status="204"/>
      </method>
    </resource>
  </resources>
</application>
```

Exercice 4 : Analyser un fichier WADL pour identifier les erreurs dans la description de l'API

Soit le fichier WADL avec des erreurs :

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2009/02">
  <resources base="http://example.com/api">
    <resource path="/users">
      <method name="GET">
        <!-- Il manque une balise response pour la méthode GET -->
      </method>
      <method name="POST">
        <request>
          <representation mediaType="application/json"/>
        </request>
      </method>
    </resource>
    <resource path="/users/{id}">
      <method name="GET">
```

```
        <response status="200">
          <representation mediaType="application/json"/>
        </response>
      </method>
      <method name="POST">
        <!-- La méthode POST n'est pas valide ici. -->
        <request>
          <representation mediaType="application/json"/>
        </request>
      </method>
    </resource>
  </resources>
</application>
```

- Analysez le fichier WADL ci-dessus et identifiez les erreurs :
    1. Il manque une balise `response` pour la méthode `GET` de la ressource `/users`.
    2. La méthode `POST` de la ressource `/users/{id}` ne doit pas exister.
    3. Il manque une représentation pour la méthode `PUT` de la ressource `/users/{id}`.

Solution :

Voici la version corrigée du fichier WADL :

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2009/02">
  <resources base="http://example.com/api">
    <resource path="/users">
      <method name="GET">
        <response status="200">
          <representation mediaType="application/json"/>
        </response>
      </method>
      <method name="POST">
        <request>
          <representation mediaType="application/json"/>
        </request>
      </method>
    </resource>
    <resource path="/users/{id}">
      <method name="GET">
        <response status="200">
          <representation mediaType="application/json"/>
        </response>
      </method>
      <method name="PUT">
        <request>
          <representation mediaType="application/json"/>
        </request>
      </method>
      <method name="DELETE">
```

```
        <response status="204"/>
      </method>
    </resource>
  </resources>
</application>
```

Exercice 5 :

Énoncé :

- Ajoutez un paramètre de requête à la ressource `/products`, permettant de filtrer les produits en fonction d'un prix minimum (`minPrice`).

Solution :

1. **Modification du fichier WADL pour ajouter un paramètre de requête `minPrice` à la ressource `/products` :**

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2009/02">
  <resources base="http://example.com/api">
    <resource path="/products">
      <method name="GET">
        <request>
          <param name="minPrice" style="query" type="xsd:decimal"
required="false"/>
        </request>
        <response status="200">
          <representation mediaType="application/json"/>
        </response>
      </method>
    </resource>
  </resources>
</application>
```

2. **Tester la requête dans Postman :**
    - Ouvrez Postman et créez une nouvelle requête GET.
    - L'URL serait : `http://example.com/api/products?minPrice=50`
    - Dans Postman, vous pouvez également ajouter un paramètre de requête en cliquant sur l'onglet "Params" et en entrant `minPrice` comme clé et `50` comme valeur.
    - Cliquez sur "Send" pour envoyer la requête et tester la réponse.