ISEN | école d'ingénieurs
ALL IS DIGITAL!

Communications Web

REST

ISEN Yncréa Ouest

thibault.napoleon@isen-ouest.yncrea.fr

Sommaire I

ISEN ALL IS DIGITAL! OUEST

Sommaire I

ISEN ALL IS DIGITAL! OUEST

REST

Définition

ISEN ALL IS DIGITAL! OUEST

Définition

REST, pour *REpresentational State Transfer*, est une architecture de développement Web permettant de construire des applications Web « orientées ressource » (c.-à-d. *Resource-oriented architecture*). Les applications qui respectent cette architecture sont dites « *RESTful* ».

Normalisation :

Même si l'architecture *REST* a été créée en 2000 par Roy Fielding (*HTTP*, *Apache*) et largement utilisée depuis 2012-2013 :

Celle-ci n'est pas standardisée !

Il existe par exemple des divergences de point de vue sur le rôle des verbes *POST* et *PUT* (voir diapositive 10).

REST

Architecture

ISEN
ALL IS DIGITAL!
OUEST

Principes architecturaux :

Pour standardiser les échanges client/serveur. L'architecture *REST* est basée sur les principes suivants :

- **Adressabilité** : Les ressources sont identifiées par des *URI* manipulés grâce à leurs représentations (voir diapositive 7).
- **Uniformité des interfaces** : Toute requête commence par un verbe lui-même basé sur les méthodes *HTTP* (voir diapositives 9 et 10).
- **Communication *stateless*** : Aucune gestion des états côté serveur (voir diapositive 5).

REST

Sans état

ISEN ALL IS DIGITAL! OUEST

Définition

C'est une architecture qui maintient un serveur « sans état » (pas de session, pas de *cookies*...). Le serveur ne tient pas compte du contexte dû à des requêtes précédentes.

Chaque requête envoyée vers le serveur doit contenir toutes les informations nécessaires à son traitement. L'état est alors conservé dans les ressources.

Pourquoi ?

Cela permet la minimisation des ressources système, car il n'y a pas de notion de session ni d'état. On dit que les requêtes sont autonomes (c.-à-d. *self-descriptive*).

Sommaire I

Les *URI*

ISEN ALL IS DIGITAL! OUEST

Définition

Un *URI* (c.-à-d. *Uniform Resource Identifier*) identifie toujours une ressource de manière unique sur le serveur.

Une ressource est une chose qui est identifiable dans un système : personne, livre, sondage...

`http://serveur:port/script.php/ressource/id?c1=v1&c2=v2`

protocole + serveur — *script PHP* — *chemin vers la ressource* — *paramètres de la requête*

Exemples :

```
http://localhost/php/request.php/polls/?login=cir2
http://localhost/php/request.php/polls/12
```

Respectivement les URL *représentant les sondages pour l'utilisateur* cir2 *et le sondage n°12.*

Sommaire I

Les verbes

Quatre verbes

Les verbes *CRUD* :

Il est possible d'appliquer quatre opérations de base à chacune des ressources hébergées sur le serveur. Elles sont représentées par les verbes dont l'acronyme est ***CRUD*** :

- ***C****reate* (Créer)
- ***R****etrieve* (Récupérer)
- ***U****pdate* (Mettre à jour)
- ***D****elete* (Supprimer)

Parfois on utilise l'acronyme ***SCRUD*** où le *S* représentera l'opération ***S****earch* (Rechercher).

Les verbes

Les méthodes *HTTP*

ISEN ALL IS DIGITAL! OUEST

Les méthodes *HTTP* associées aux verbes :

L'architecture *REST* s'appuie sur le protocole *HTTP* pour faire des requêtes. On utilise donc des méthodes *HTTP* pour implémenter les différentes actions applicables aux ressources :

- *Create* : méthode ***POST***
- *Retrieve* : méthode ***GET***
- *Update* : méthode ***PUT***
- *Delete* : méthode ***DELETE***

Une ambiguïté est présente entre les méthodes *POST* et *PUT* ! ! Vous trouverez l'inverse dans certains *frameworks*.

Les verbes

Create → *POST*

ISEN ALL IS DIGITAL! OUEST

Définition

La méthode ***POST*** permet de créer une ressource. Elle est non idempotente (plusieurs créations de la même ressource).

Création *du sondage de titre* s1 *par* tibo.

Les verbes

Create → *POST*

ISEN ALL IS DIGITAL! OUEST

Définition

La méthode ***POST*** permet de créer une ressource. Elle est non idempotente (plusieurs créations de la même ressource).

Création *du sondage de titre* s1 *par* tibo.

Les verbes

Retrieve → *GET*

ISEN ALL IS DIGITAL! OUEST

Définition

La méthode ***GET*** permet de récupérer la représentation d'une ressource. Elle est idempotente.

Récupération *de tous les sondages.*

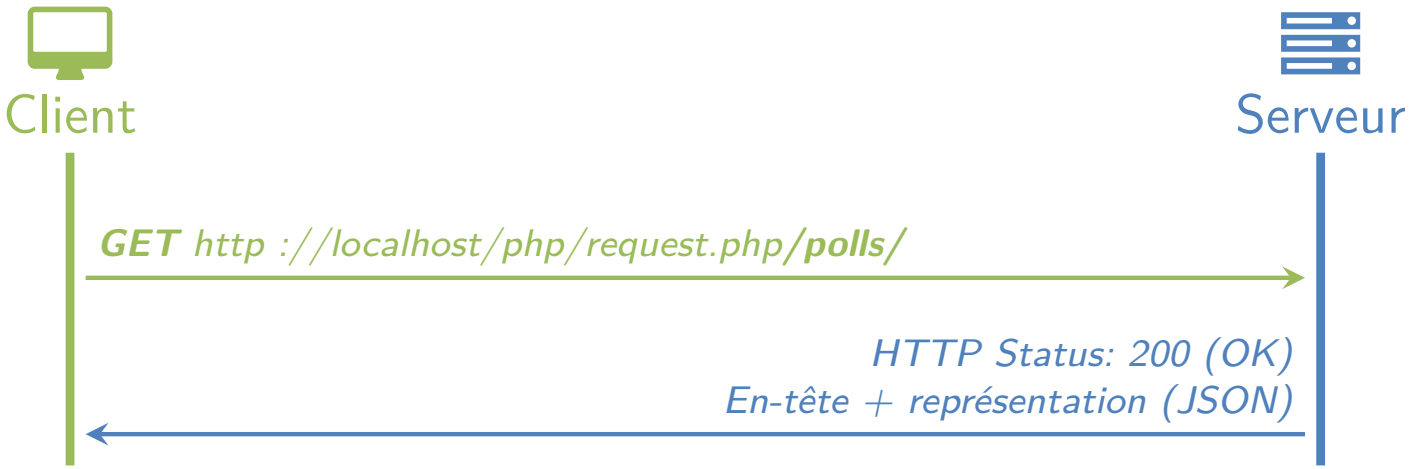
Les verbes

Retrieve → *GET*

ISEN ALL IS DIGITAL! OUEST

Définition

La méthode ***GET*** permet de récupérer la représentation d'une ressource. Elle est idempotente.



Récupération *de tous les sondages.*

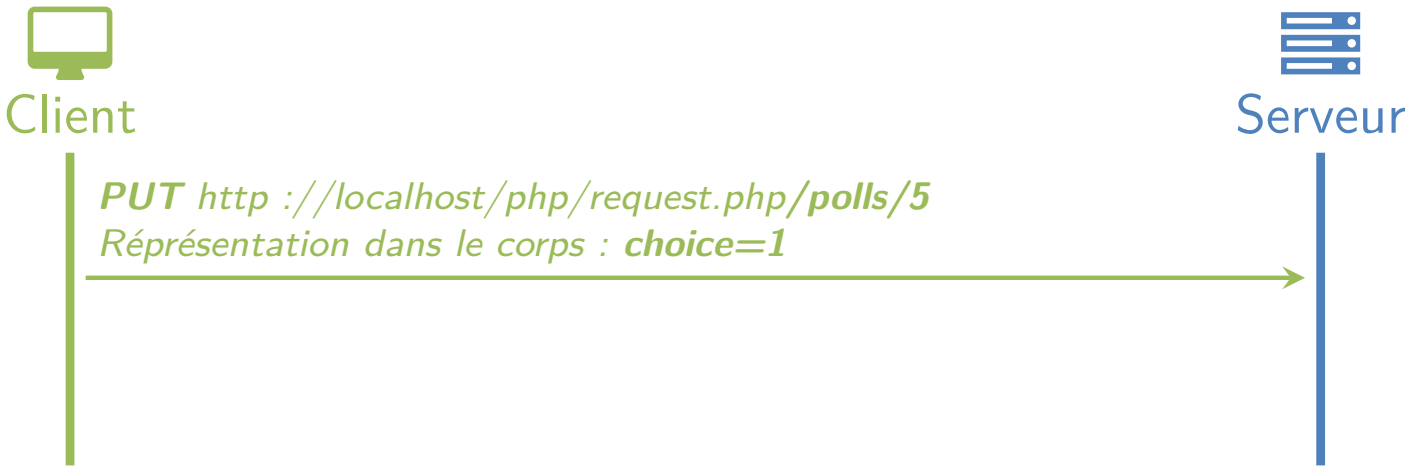
Les verbes

Update → *PUT*

ISEN ALL IS DIGITAL! OUEST

Définition

La méthode ***PUT*** permet de mettre à jour une ressource. Elle est idempotente.



Mise à jour *du sondage n°*5 *avec le choix* 1.

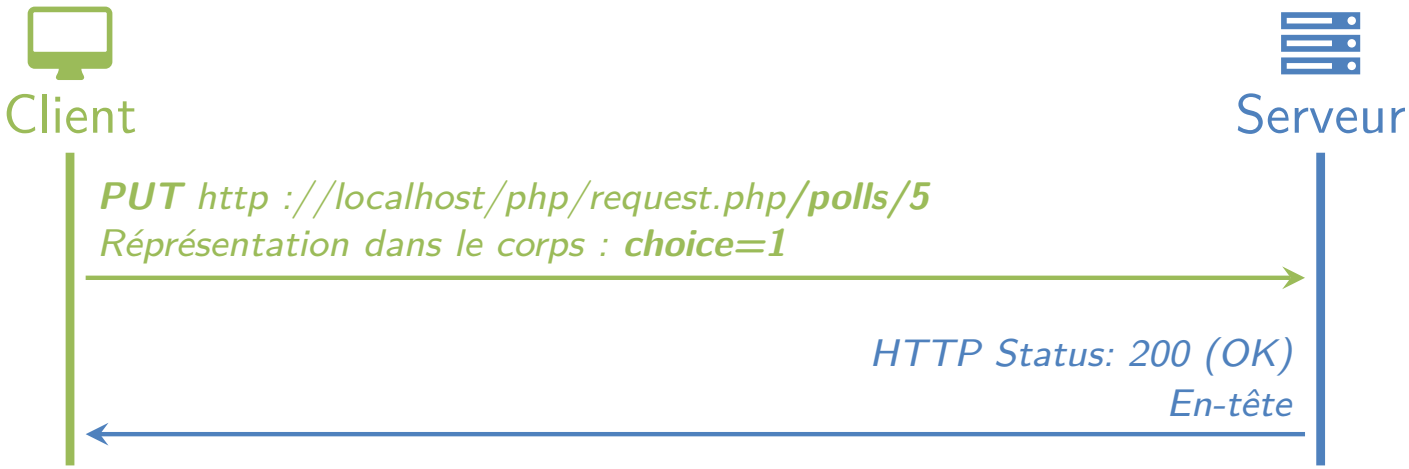
Les verbes

Update → *PUT*

ISEN ALL IS DIGITAL! OUEST

Définition

La méthode ***PUT*** permet de mettre à jour une ressource. Elle est idempotente.



Mise à jour *du sondage n°*5 *avec le choix* 1.

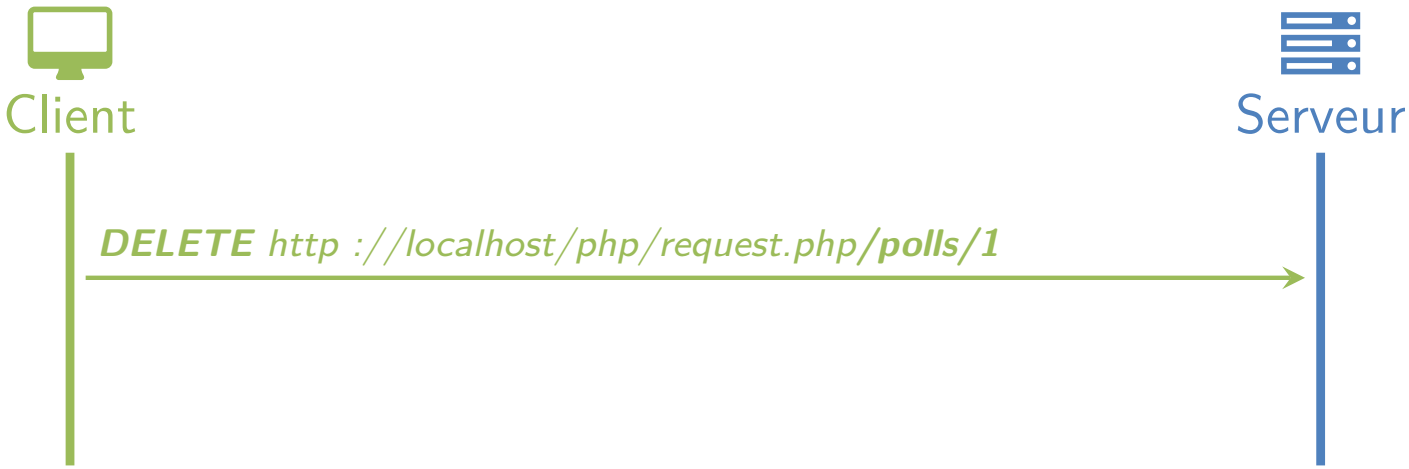
Les verbes

Delete → *DELETE*

ISEN ALL IS DIGITAL! OUEST

Définition

La méthode ***DELETE*** permet de supprimer une ressource. Elle est idempotente.



Suppression *du sondage n°1.*

Les verbes

Delete → *DELETE*

ISEN ALL IS DIGITAL! OUEST

Définition

La méthode ***DELETE*** permet de supprimer une ressource. Elle est idempotente.

Suppression *du sondage n°1.*

Sommaire I

ISEN ALL IS DIGITAL! OUEST

Représentation

Définition

ISEN ALL IS DIGITAL! OUEST

Définition

Ce sont des données additionnelles associées à une requête ou à sa réponse.

Pour qui ?

La représentation peut être :

- À destination du serveur lors d'un ***GET***, ***POST***, ***PUT*** ou ***DELETE***.
- À destination du client lors d'un ***GET***.

Sous quelle forme ?

Il existe deux formes de représentations :

- Dans l'*URI* directement (pour les requêtes ***GET*** ou ***DELETE***).
- Dans le corps de la requête ou de la réponse (dans les autres cas). Souvent avec la notation *JSON*.

Représentation

Représentation dans l'*URI*

Syntaxe :

Pour associer une représentation à une requête directement dans l'*URI*, on concatène la représentation à celle-ci. Pour cela, il est nécessaire d'ajouter le séparateur **?** entre l'*URI* de la ressource et la représentation.

S'il y a plusieurs informations dans la représentation, on les sépare par un **&**.

En *PHP* on utilise la syntaxe suivante (même pour un *DELETE*) :

```php
$_GET["key"];
```

Exemple :

```
GET http://localhost/php/request.php/polls/?login=tibo&delay=10d
```

Récupération de l'ensemble des sondages de l'utilisateur *tibo* et qui datent de moins de *10 jours*.

Représentation

ISEN ALL IS DIGITAL! OUEST

Représentation dans le corps de la requête

Syntaxe *JavaScript* :

Pour pouvoir envoyer une représentation dans le corps d'une requête *AJAX*, grâce à *send(data)*, il faut ajouter l'entête suivant :

```
1 xhr.setRequestHeader('Content-Type',
2   'application/x-www-form-urlencoded');
```

Syntaxe *PHP* :

Pour récupérer cette représentation en *PHP*, il faut utiliser lors d'un ***POST*** :

```
1 $_POST["key"];
```

En revanche, pour un ***PUT*** il faut utiliser :

```
1 parse_str(file_get_contents('php://input'), $_PUT);
2 $_PUT["key"];
```

Sommaire I

ISEN ALL IS DIGITAL! OUEST

Conclusion

Avantages et inconvénients

Avantages :

- L'intelligence est du côté client :
  $\Rightarrow$ Le serveur répond à des requêtes simples pour récupérer, créer, mettre à jour ou supprimer une ressource.
- Le serveur est peu sollicité.
- Les *URI* sont réécrits pour nommer plus naturellement les ressources.
- L'architecture *REST* est « sans état ».

Inconvénients :

- Nécessite un serveur capable de prendre en compte cette architecture.
- Augmente la charge du client qui doit construire le rendu visuel.
- Alourdit le code côté client.

Conclusion

Les requêtes du mini-projet

Authentification :

- `GET request.php/authenticate`

Ressources photo :

- `GET request.php/photos/`
- `GET request.php/photos/i`

Ressources commentaires :

- `GET request.php/comments/i`
- `POST request.php/comments/i + comment=...`
- `DELETE request.php/comments/i`

ISEN | école d'ingénieurs
ALL IS DIGITAL!

Avez vous des questions ?