Client JS JSON - REST avec JQuery

Serveur REST en Silex - Client Onepage Ajax

Exercice 1. Contexte

Dans ce TD, vous allez mettre en place un client JavaScript *riche* parfois appelé *RIA* à l'aide de la librairie JQuery à une application REST codée en PHP Silex. Cette architecture permet de réaliser des applications de type *onepage* en reportant sur le client une bonne partie de la logique métier. Les données du serveur sont envoyées en JSON : JavaScript Object Notation, c'est à dire des objets directement utilisables en JS.

Typiquement une architecture REST doit vérifier les propriétés principales suivantes côté serveur :

1. Une séparation claire avec la partie client
2. Etre sans état (sans mémoire entre 2 requêtes client)
3. On peut marquer ou pas des ressources comme pouvant être mises en cache
4. Elle doit mettre en place un protocole simple et cohérent, généralement basé sur HTTP
5. Elle peut (et souvent elle doit) fonctionner avec une authentification des clients

Un flux de news pourra ainsi offrir par exemple une ressource du type : `/api/news/36467` qui permettra aux clients de récupérer la news numéro 36467 en JSON ou en XML en employant la méthode HTTP GET. Dans cet exemple, la news est ici la ressource ou *élément* manipulée dans l'API. La méthode GET sera employée pour récupérer des *éléments individuellement* ou par *Collections*. La méthode POST sera quand à elle employée pour envoyer vers le serveur une collection d'éléments. D'autres méthodes HTTP pour créer ou modifier (PUT) des éléments ou les effacer (DELETE) seront également disponibles dans l'API. La méthode DELETE pourra également servir à supprimer une collection d'éléments.

Exercice 2. Les bases de JSON

Construisons les objets JavaScript suivants :

```
1 class Personne{
2     constructor(id, prenom, nom){
3       this.id = id;
4       this.nom = nom;
5       this.prenom = prenom;
```

```
 6      }
 7      toString(){
 8          return prenom +' '+ nom;
 9      }
10  };
11  let p1 = new Personne(1,'John', 'Doe');
12  console.log(p);
13
14   class Contacts{
15     constructor(nom){
16      this.nom=nom;
17      this.contacts= Array();
18   }
19      add(pers){
20          this.contacts.push(pers);
21      }
22  };
23  let carnet = new Contacts("MesContacts");
24  let p2 = new Personne(2,'Alice', 'Spring');
25  carnet.add(p1);
26  carnet.add(p2);
27  console.log(carnet);
28  console.dir(carnet);
29  console.log(JSON.stringify(carnet));
```

L'affichage que vous obtenez est ce qu'on appelle du JSON dans le dernier cas.

- Ajoutez un champ date de naissance et un autre ville aux personnes
- Ajoutez une méthode toString() à la classe Contacts
- Ajoutez des méthodes get et set sur la ville et la date de naissance d'une personne.
- Testez

Exercice 3. Template d'affichage des questions en provenance de la base

On part du template flex.html fourni associé au style flex.css. Le template contacts.html suivant va en hériter. Il contiendra un ensemble de fonctions JS permettant de récupérer un carnet, des personnes et de les éditer puis de les modifier, effacer, etc

```
{% extends "flex.html" %}
{% block js %}
{{ parent() }}
<script>
$(function() {
    $("#button").click(refreshContactsList);

    function refreshContactsList(){
        console.log('refreshContactsList');
    }

    // Objet Personne en JS
    class Personne{
    constructor(id, prenom, nom){
      this.id = id;
      this.nom = nom;
      this.prenom = prenom;
    }
    toString(){
        return prenom +' '+ nom;
    }
};

    $("span#tool1").on("click", newpers);
</script>
{% endblock %}

{% block contacts %}
    <input id="button" type="button" value="Recuperer les
       contacts" />
    <div id="carnet">

    </div>
{% endblock %}

 {% block editor %}
<div id="currentpers">

</div>
 {%endblock %}
```

— Commençons par compléter la fonction `refreshContactsList()` en utilisant la fonc-

tion ajax de JQuery :

```
function refreshContactsList(){
        $.ajax({
            url: "http://localhost/~login/silex/api/v/
               contacts",
            type: "GET",
            // Paramètres optionnels transmis
            //data : ,
            dataType : "json",
            success: function(data) {
                console.log(JSON.stringify(data.contacts
                   ));
                $('#carnet').empty();
                $('#carnet').append($('<ul>'));
                for(var i=0;i<data.contacts.length;i++){
                    console.log(data.contacts[i]);
                    $('#carnet ul')
                    .append($('<li>')
                    .append($('<a>')
                        .text(data.contacts[i].nom)
                      ).on("click", data.contacts[i],
                         details)
                    );
                        }
                    },
            error: function(req, status, err) {
                      $("#carnet").html("<b>Impossible
                         de récupérer les contacts !</b>"
                         );
                    }
                  });
            }
```

Faisons en sorte, en utilisant JQuery d'afficher la question complète qui sera sélectionnée sur la barre latérale de gauche.

— Puis complétons la méthode details() :

```
function details(event){
        $("#currentpers").empty();
        //alert(event.data.nom);
        for (p in event.data){
            $("#currentpers")
            .append($('<ul>')
```

```
7                   .append($('<li>')
8                   .text(p+": "+event.data[p])
9                   ))
10              }
11          }
```

— La méthode `newpers()` qui présente un formulaire d'ajout de personnes et appelle `saveNewPers()` à la validation.

```
1  function newpers(){
2          $("#currentpers").empty();
3          $("#currentpers")
4              .append($('<span>Nom<input type="text" id="
                   nom"><br></span>'))
5              .append($('<span>Prénom<input type="text" id
                   ="prenom"><br></span>'))
6              .append($('<span>Naissance<input type="date"
                    id="naissance"><br></span>'))
7              .append($('<span><input type="button" value
                   ="Save Pers"><br></span>').on("click",
                   saveNewPers));
8          }
```

— La méthode `saveNewPers()` proprement dite qui instancie une nouvelle personne puis la transmet en ajax au serveur via la méthode HTTP POST :

```
1  function saveNewPers(){
2          let pers = new Personne(
3              $("#currentpers #nom").val(),
4              $("#currentpers #prenom").val(),
5              $("#currentpers #naissance").val()
6              );
7          console.log(JSON.stringify(pers));
8          $.ajax({
9              url: 'http://localhost/~login/silex/api/v1
                   .0/tasks',
10             type: 'POST',
11             contentType: 'application/json',
12             data: JSON.stringify(pers),
13             dataType: 'json'
14             });
15         refreshContactsList();
16     }
17 });
```

— Testez

Exercice 4. A suivre

— Ajoutez la possibilité de supprimer une personne en utilisant la méthode HTTP DELETE

— Puis celle de modifier une personne en utilisant la méthode PUT.