

# Equipe 2 - Roche Sébastien / Rodrigues Quentin



---

## Présentation et objectifs


Nous sommes l'**équipe 2**, composée de **Rodrigues Quentin** et de **Roche Sébastien**. Nous avons choisi la **mission 1**.

Afin de réaliser notre mission, nous avons deux tâches à accomplir :

- **Gestion des hôtels** : Cette tâche doit permettre **la création, la modification, la suppression** d'un **hôtel** à partir d'un formulaire. Il permet également d'obtenir la liste des hôtels. Il faudra réaliser les maquettes de ces formulaires, les faire valider par le chef de projet avant de commencer le codage.
- **Hôtellerie** : Cette tâche doit permettre de **loger un congressiste** dans un **hôtel**, le congressiste peut **demande à bénéficier du petit-déjeuner** ce qui occasionne un supplément. Cette tâche n'est pas possible si la facture a déjà été imprimée.


---

## Environnement de travail

Avant de découvrir notre travail, nous tenons à vous présenter notre environnement. Nous travaillons à l'aide du base de données , fournit dès le début du projet.

Par la suite, nous devons travailler sur un serveur WEB PHP. Pour ce faire nous utilisons Apache

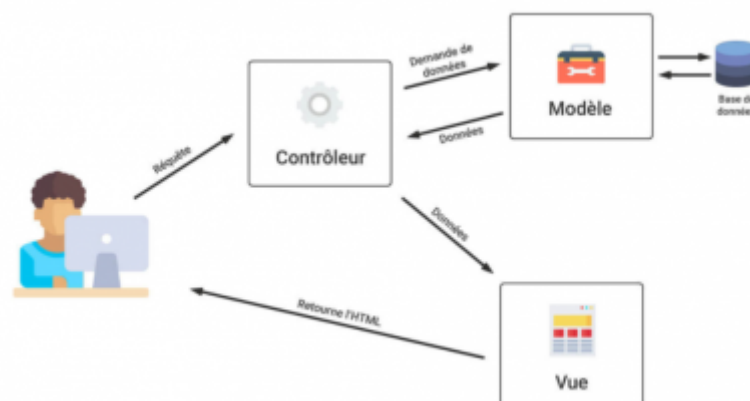


Chaque membre de l'équipe est libre de travail sur son IDE, cependant nous travaillons tous les deux sur Visual Studio Code .

## Utilisation de l'architecture MVC en PHP Objet

L'architecture MVC est l'une des architectures logicielles les plus utilisées pour les applications Web, elle se compose de 3 modules :

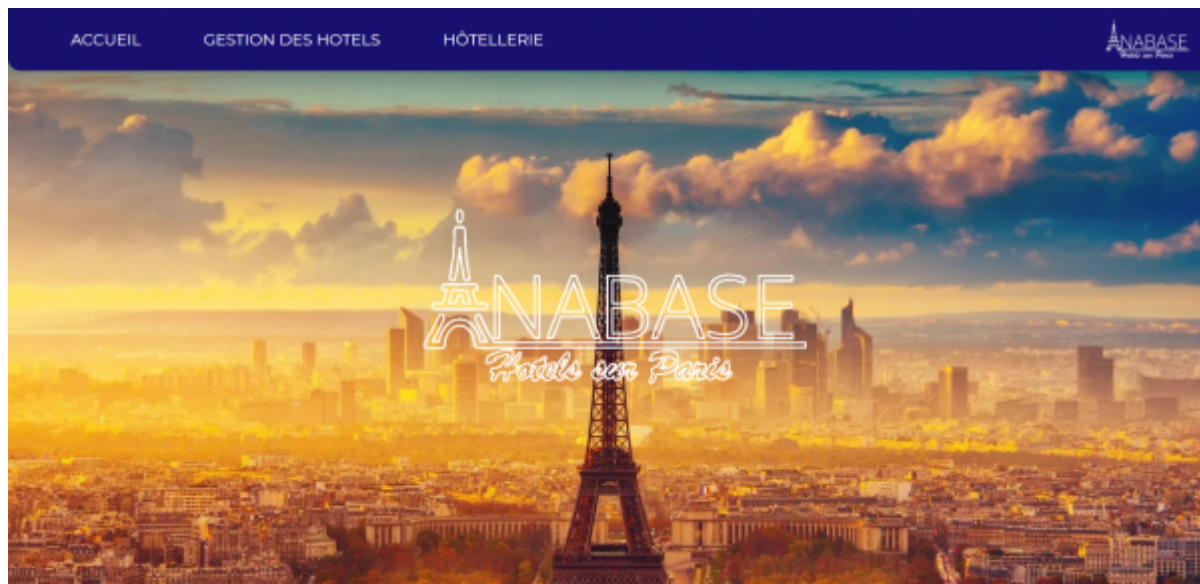
- **Modèle** : noyau de l'application qui gère les données, permet de récupérer les informations dans la base de données, de les organiser pour qu'elles puissent ensuite être traitées par le contrôleur.
- **Contrôleur** : partie principale de l'architecture, car c'est lui qui va faire le lien entre l'utilisateur, le modèle et la vue. Quand l'utilisateur souhaite afficher une page, c'est le contrôleur qui va recevoir cette requête et se charger ensuite de récupérer les données via le modèle, pour les envoyer à la vue qui sera affichée à l'utilisateur. Il est l'intermédiaire entre le modèle et la vue.
- **Vue** : composant graphique de l'interface qui permet de présenter les données du modèle à l'utilisateur au sein du code HTML.



Vous pouvez retrouver ci-contre l'intégration de PHP orienté Objet dans notre projet : [Découvrir PHP Objet](#).

## Apparence

La page d'accueil de notre site se présente ainsi :




L'onglet **GESTION DES HOTELS** affiche un tableau (Qui se gère au clique droit) :

Id	Nom	Adresse hôtel	Nombre étoile	Prix participant	Prix supplémentaire
1	Hôtels les angles morts	10 rue de la plage	4	172€	340€
3	Hôtel de la recherche	43 rue de la loi	5		58€
4	Hôtel chez Nasa	8 rue des lilas	4		35€
5	Hôtel Paradiso	9 rue des Inscriptions	1		7€
6	Hôtel des Utilisations	14 rue des diamants	3		16€
7	Hôtel la Pintache	3 place des lompes	2	94€	24€

Ajouter un hotel

Cet onglet permet de réaliser la première tâche, c'est à dire ajouter / modifier / supprimer un hôtel.

L'onglet **HÔTELLERIE** :

[ACCUEIL](#) [GESTION DES HOTELS](#) [HÔTELLERIE](#) 

### Outils d'hôtellerie

Choisir un congressiste ▾

Choisir un hôtel ▾

Petit-déjeuner ☐

Réserver

*Cet onglet permet de réaliser la seconde tâche, c'est à dire loger un congressiste dans un hôtel, et prendre en charge la gestion du petit-déjeuner.*

## Découvrir notre travail

### Gestion des hôtels :

- [Ajouter un hôtel](#)
- [Modifier un hôtel](#)
- [Supprimer un hôtel](#)
- [Bonus : Clic droit personnalisé.](#)

### Hôtellerie :

- [Placer un congressiste dans un hôtel](#)
- [Sélection du petit-déjeuner](#)

**Bonus :** [Utilisation de Ajax](#)

## Conclusion

Pour la réalisation de cet AP, nous avons utilisé le langage PHP orienté objet, ainsi, c'est la première fois que nous utilisons le langage PHP de cette manière, notamment en MVC, ce qui a permis d'améliorer notre niveau en programmation.

De plus, elle nous a permis de découvrir davantage AJAX pour obtenir un rendu de page plus propre et sans rechargement de page. On a également utilisé JQuery pour faciliter nos programmes en

Javascript.

From:

<https://ppe.boonum.fr/> - **AP.SIO**

Permanent link:

<https://ppe.boonum.fr/doku.php?id=slam:ws:2021:sio:ap3.1:equipe2:accueil>

Last update: **2021/11/09 17:01**



# Ajouter un hôtel

Afin d'ajouter un hôtel, j'ai choisi de créer un bouton "Ajouter un hôtel" en dessous du tableau :

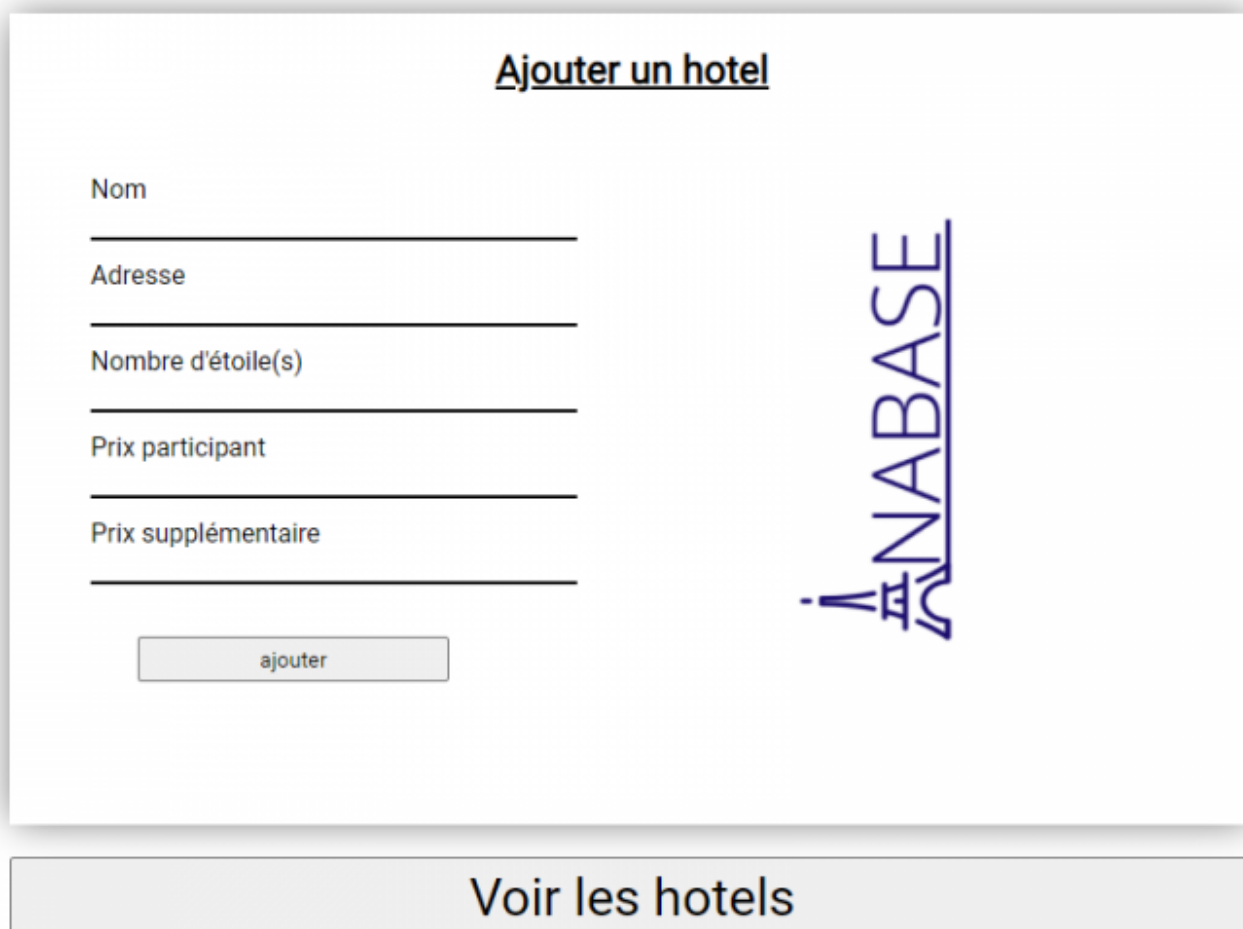


The screenshot shows a web application interface with a dark blue header containing the menu items 'ACCUEIL', 'GESTION DES HOTELS', and 'HÔTELLERIE', along with the 'ANABASE' logo. The main content area is titled 'Gestion des hôtels' and features a table with the following data:

Id	Nom	Adresse hôtel	Nombre étoile	Prix participant	Prix supplémentaire
1	Hôtels les angles morts	16 rue de la plage	4	172€	340€
3	Hôtel de la recherche	43 rue de la loi	5	214€	58€
4	Hôtel chez Nasa	8 rue des lilas	4	437€	95€
5	Hôtel Paradiso	9 rue des Inségnora	1	49€	7€
6	Hôtel des Utilisateurs	14 rue des diamants	3	128€	19€
7	Hôtel la Pétasche	3 place des lampes	2	94€	24€

Below the table is a button labeled 'Ajouter un hotel'. A red arrow points from the text above to this button.

Celui-ci redirige vers un formulaire d'ajout présenté comme ceci :



The screenshot shows a form titled 'Ajouter un hotel' with the following fields and a logo:

- Nom
- Adresse
- Nombre d'étoile(s)
- Prix participant
- Prix supplémentaire

Below the fields is a button labeled 'ajouter'. To the right of the form is the 'ANABASE' logo, which includes a stylized building icon.

At the bottom of the page is a button labeled 'Voir les hotels'.

La redirection se fait à l'aide d'Ajax ([Voir page Ajax](#))

Pour compléter ce formulaire, j'ai utilisé la balise `<form>` d'HTML :

```
<fieldset class="fieldset-ajout-hotel">
  <h2 class="h2ajout">Ajouter un hotel</h2>
  <div class="justify-div">
    <form action="?controleur=hotels&todo=ajouter" method="post"
class="form-ajout-hotel" name="form-ajout">
      <label for="">Nom</label><br>
      <input class="input-ajout-hotel" type="text" name="nom"><br>
      <label for="">Adresse</label><br>
      <input class="input-ajout-hotel" type="text" name="adresse"><br>
      <label for="">Nombre d'étoile(s)</label><br>
      <input class="input-ajout-hotel" type="text"
name="nombreEtoiles"><br>
      <label for="">Prix participant</label><br>
      <input class="input-ajout-hotel" type="text"
name="prixParticipant"><br>
      <label for="">Prix supplémentaire</label><br>
      <input type="text" class="input-ajout-hotel" name="prixSuppl"><br>
      <input class="input-submit-ajout-hotel" type="submit"
value="ajouter" title="Ajouter" name="todo">
    </form>
    
  </div>
</fieldset>

<button class="voirHotels">Voir les hotels</button>
```

L'action du formulaire est : `'?controleur=hotels&todo=ajouter'` ce qui signifie qu'on passe par le contrôleur **hotels**, et on cherche la fonction **ajouter**.

Le formulaire est validé en [Ajax](#), ce qui ne change rien par rapport à un formulaire en PHP vanilla.

Voici la fonction **ajouter**, présente dans le contrôleur **hotels** :

```
public function todo_ajouter(){

    $this->modele->addHotel($_POST['nom'], $_POST['adresse'],
$_POST['nombreEtoiles'], $_POST['prixParticipant'], $_POST['prixSuppl']);
    echo "<div class='confirmationajout'><h2>L'hotel : ".$_POST['nom'].
vient d'être ajouté.</h2></div>";

    $this->data["hotels"] = $this->modele->getListeHotel();
    $this->vue = "afficherhotel";
}
```

Nous utilisons le langage PHP, orienté objet.

Comme nous pouvons le voir ci-dessus, la fonction appelle **addHotel()**, présente dans le modèle :

```

public function addHotel($nom, $adresse, $nombreEtoiles, $prixParticipant,
    $prixSuppl){
    $sql = "Insert into hotel VALUES(NULL, ?,?,?,?,?)";

    $req = $this->conn->prepare($sql);
    $req->bindValue(1, $nom);
    $req->bindValue(2, $adresse);
    $req->bindValue(3, $nombreEtoiles);
    $req->bindValue(4, $prixParticipant);
    $req->bindValue(5, $prixSuppl);

    $req->execute();
}

```

Après avoir ajouté un hôtel, un petit message de confirmation s'affiche dans le cas où l'opération a réussi :

Id	Nom	Adresse hôtel	Nombre étoile	Prix participant	Prix supplémentaire
1	Hôtels les angles morts	10 rue de la plage	4	172€	340€
3	Hôtel de la recherche	43 rue de la loi	5	214€	58€
4	Hôtel				35€
5	Hôtel				7€
6	Hôtel				16€
7	Hôtel la Pistache	3 place des lompes	2	94€	24€
21	Hôtel Pistache	1 rue de la rue	3	124€	19€

Ajouter un hotel

En cas d'erreur, par exemple si l'utilisateur entre une chaîne de caractère à la place d'un entier, il y aura une erreur.

Pour ce faire, j'utilise une fonction dans mon modèle, que j'ai découvert en PHP :

```

public function addHotel($nom, $adresse, $nombreEtoiles, $prixParticipant,
    $prixSuppl){
    settype($nombreEtoiles, "integer");
    settype($prixParticipant, "integer");
    settype($prixSuppl, "integer");

    if($nombreEtoiles == 0 or $prixParticipant == 0 or $prixSuppl == 0){
        $_POST['insert'] = false;
    } else{
        $sql = "Insert into hotel VALUES(NULL, ?,?,?,?,?)";

        $req = $this->conn->prepare($sql);
        $req->bindValue(1, $nom);
    }
}

```



```
$req->bindValue(2, $adresse);
$req->bindValue(3, $nombreEtoiles);
$req->bindValue(4, $prixParticipant);
$req->bindValue(5, $prixSuppl);

$req->execute();

$_POST['insert'] = true;
}

}
```

La fonction **settype()** va remplacer la valeur de la variable entrée en paramètre par un 0 si il n'est pas capable de devenir un entier. Car par défaut c'est un String.

Un input vide ne peut pas être un entier. Donc l'erreur fonctionne en cas de champ vide.

Ensuite, nous vérifions si aucune des trois valeurs qui doivent être un entier ne soit pas égale à 0, dans le cas contraire j'exécute la requête.


Pour savoir depuis le contrôleur si l'insertion a été effectuée, j'utilise une variable POST.

Dans le contrôleur :

J'ajoute cette condition :

```
if($_POST['insert']){
    echo "<div class='confirmationajout'><h2>L'hotel :  
".$_POST['nom']."' vient d'être ajouté.</h2></div>";
} else{
    echo "<div class='confirmationajout  
confirmationajoutfail'><h2>Une erreur s'est produite.</h2></div>";
}
```

Pour afficher le message de confirmation, ou bien l'erreur.

[ACCUEIL](#) [GESTION DES HOTELS](#) [HÔTELLERIE](#) 

### Gestion des hôtels

Id	Nom	Adresse hôtel	Nombre étoile	Prix participant	Prix supplémentaire
1	Hôtels les angles morts	10 rue de la plage	4	172€	34€
3	Hôtel de la recherche	43 rue de la loi	5	214€	58€
4	Hôtel				35€
5	Hôtel				7€

Une erreur s'est produite.

Ajouter un hotel

From:

<https://ppe.boonum.fr/> - **AP.SIO**

Permanent link:

<https://ppe.boonum.fr/doku.php?id=slam:ws:2021:sio:ap3.1:equipe2:ajouter>

Last update: **2021/11/10 08:12**

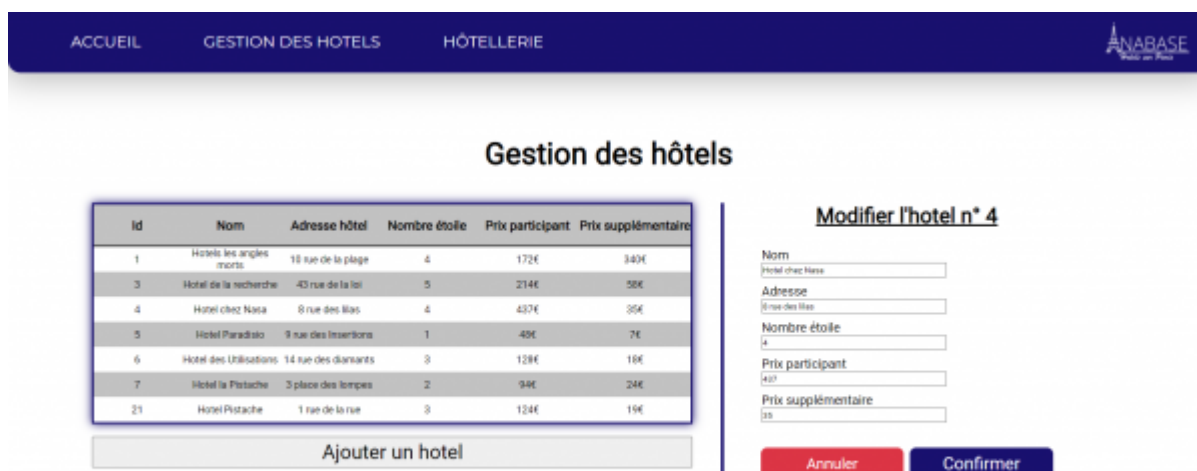


# Modifier un hôtel

Afin de modifier un hôtel, j'ai choisi de créer un bouton "Modifier" sur mon menu contextuel ([Au clic droit](#)) :



Celui-ci permet d'afficher un formulaire sur la droite du tableau présenté comme ceci :



Comme nous pouvons le voir il y a deux boutons, le premier permet d'annuler la modification. Et le deuxième permet de confirmer les changements effectués.

Lors de la confirmation, le processus est semblable à une insertion. Nous allons retourner dans le contrôleur dans le but d'effectuer une requête SQL. J'utilise encore [AJAX](#).

Mon formulaire est présenté comme ceci :

```
<div class='modifHotel'>
  <h2 id='h2modif'></h2>
  <form action='?controlleur=hotels&todo=modifier' class='formModif'
```

```
method="POST">
    <label for="">Nom</label><br>
    <input type="text" id="nomModif" name="nomModif"><br>
    <label for="">Adresse</label><br>
    <input type="text" id="adresseModif" name="adresseModif"><br>
    <label for="">Nombre étoile</label><br>
    <input type="text" id="etoileModif" name="etoileModif"><br>
    <label for="">Prix participant</label><br>
    <input type="text" id="prixParticipantModif"
name="prixParticipantModif"><br>
    <label for="">Prix supplémentaire</label><br>
    <input type="text" id="prixSuppModif" name="prixSuppModif">
    <input id="idModif" type="hidden" value="" name="idModif">
    <div class="confirmButton">
        <button type="button" id="cancelModif">Annuler</button>
        <input type="submit" value="Confirmer">
    </div>
</form>
</div>
```

L'action que récupère Ajax est donc '?controleur=hotels&todo=modifier'.

La fonction `todo_modifier()` dans le contrôleur est celle-ci :

```
public function todo_modifier(){
    $this->modele->updateHotel($this->post['idModif'],
    $this->post['nomModif'], $this->post['adresseModif'],
    $this->post['etoileModif'], $this->post['prixParticipantModif'],
    $this->post['prixSuppModif']);

    $this->data["hotels"] = $this->modele->getListeHotel();
    $this->vue = "afficherhotel";
}
```

On appelle la fonction `updateHotel()` du modèle, puis on réaffiche le tableau d'hôtels.

La requête présente dans le modèle se présente comme ci-dessous :

```
public function updateHotel($id, $nom, $adresse, $nombreEtoiles,
    $prixParticipant, $prixSuppl){

    $sql = "UPDATE hotel SET NOM_HOTEL = ?, ADRESSE_HOTEL = ?,
    NOMBRE_ETOILES = ?, PRIX_PARTICIPANT = ?, PRIX_SUPPL = ? WHERE NUM_HOTEL =
    ?";

    $req = $this->conn->prepare($sql);

    $req->bindValue(1, $nom);
    $req->bindValue(2, $adresse);
    $req->bindValue(3, $nombreEtoiles);
```

```
$req->bindValue(4, $prixParticipant);
$req->bindValue(5, $prixSuppl);
$req->bindValue(6, $id);

$req->execute();
}
}
```

Enfin, l'enregistrement a subit un changement, et il est affiché correctement au nouvel affichage du tableau.

En cas d'erreur, par exemple si l'utilisateur entre une chaîne de caractère à la place d'un entier, il y aura une erreur.

Pour ce faire, j'utilise une fonction dans mon modèle, que j'ai découverte en PHP :

```
public function updateHotel($id, $nom, $adresse, $nombreEtoiles,
    $prixParticipant, $prixSuppl){

    settype($nombreEtoiles, "integer");
    settype($prixParticipant, "integer");
    settype($prixSuppl, "integer");

    if($nombreEtoiles == 0 or $prixParticipant == 0 or $prixSuppl == 0){
        $_POST['insert'] = false;
    } else{
        $sql = "UPDATE hotel SET NOM_HOTEL = ?, ADRESSE_HOTEL = ?,
NOMBRE_ETOILES = ?, PRIX_PARTICIPANT = ?, PRIX_SUPPL = ? WHERE NUM_HOTEL =
?";

        $req = $this->conn->prepare($sql);

        $req->bindValue(1, $nom);
        $req->bindValue(2, $adresse);
        $req->bindValue(3, $nombreEtoiles);
        $req->bindValue(4, $prixParticipant);
        $req->bindValue(5, $prixSuppl);
        $req->bindValue(6, $id);

        $req->execute();

        $_POST['insert'] = true;
    }
}
```

La fonction **settype()** va remplacer la valeur de la variable entrée en paramètre par un 0 si il n'est pas capable de devenir un entier. Car par défaut c'est un String.

Un input vide ne peut pas être un entier. Donc l'erreur fonctionne en cas de champ vide.

Ensuite, nous vérifions si aucune des trois valeurs qui doivent être un entier ne soit pas égale à 0,

dans le cas contraire j'exécute la requête.


Pour savoir depuis le contrôleur si l'insertion a été effectuée, j'utilise une variable POST.

Dans le contrôleur :

J'ajoute cette condition :

```
if($_POST['insert']){  
    echo "<div class='confirmationajout'><h2>L'hotel :  
    ".$_POST['nomModif']." vient d'être ajouté.</h2></div>";  
} else{  
    echo "<div class='confirmationajout  
confirmationajoutfail'><h2>Une erreur s'est produite.</h2></div>";  
}
```

Pour afficher le message de confirmation, ou bien l'erreur.

[ACCUEIL](#) [GESTION DES HOTELS](#) [HÔTELLERIE](#) 

### Gestion des hôtels

id	Nom	Adresse hôtel	Nombre étoile	Prix participant	Prix supplémentaire
1	Hôtels les angles morts	10 rue de la plage	4	172€	34€
3	Hôtel de la recherche	43 rue de la loi	5	214€	58€
4	Hôtel				95€
5	Hôtel				7€

Une erreur s'est produite.

Ajouter un hotel

From:

<https://ppe.boonum.fr/> - **AP.SIO**

Permanent link:

<https://ppe.boonum.fr/doku.php?id=siam:ws:2021:sio:ap3.1:equipe2:modifier>

Last update: **2021/11/10 08:13**



# Supprimer un hôtel

Afin de supprimer un hôtel, j'ai choisi de créer un bouton "Supprimer" sur mon menu contextuel (Au clic droit) :

The screenshot shows a web application interface with a dark blue header containing the menu items: ACCUEIL, GESTION DES HOTELS, and HÔTELLERIE. The main content area is titled 'Gestion des hôtels' and features a table with the following data:

Id	Nom	Adresse hôtel	Nombre étoile	Prix participant	Prix supplémentaire
1	Hôtels les angles morts	10 rue de la plage	4	172€	340€
3	Hôtel de la recherche	43 rue de la loi		214€	58€
4	Hôtel chez Nasa	8 rue des lilas		437€	35€
5	Hôtel Paradiso	9 rue des Imberts		49€	7€
6	Hôtel des Utilisations	14 rue des diamants		128€	18€
7	Hôtel la Pistache	3 place des loupes	2	94€	24€
21	Hôtel Pistache	1 rue de la rue	3	124€	19€

Below the table is a button labeled 'Ajouter un hotel'. A context menu is open over the table, showing options: Actualiser, Modifier, Supprimer, and Surligner.

Celui-ci permet de supprimer l'enregistrement du tableau.

Pour faire la suppression de l'enregistrement, j'ai dû utiliser JQuery pour récupérer toute une ligne.

Tout d'abord, Nous récupérerons l'ID de ma ligne dans mon tableau sur l'évènement Hover,.

```
$('.rowhotel').mouseover(function () {
    index = this.id;
});
```

Ensuite, j'ai choisi de mettre les données du tableau dans un variable, qui est un tableau. Les données vont dans la variable une fois que le clic sur supprimer a été effectué.

```
var content = $('#'+index).children().each(function(index){
    datas[index] = $(this).text();
});
```

Puis, nous allons de nouveau utiliser AJAX pour entrer dans le contrôleur nécessaire à l'exécution de la requête.

```
$.ajax({
    url: '?controleur=hotels&todo=supprimer', // La ressource ciblée
    type: 'POST', // Le type de la requête HTTP,
    data: 'idHotel='+datas[0], // datas[0] === ID
    dataType: 'html', // Le type de données à recevoir, ici, du PHP.

    success: function(response) {
        $(".content").append(response);
    }
});
```

```
    },  
    });
```

Nous voici dans le contrôleur, et dans la fonction supprimer :

```
public function todo_supprimer(){  
    $this->modele->deleteHotel($this->post['idHotel']);  
  
    $this->data["hotels"] = $this->modele->getListeHotel();  
    $this->vue = "afficherhotel";  
}
```

Nous demandons l'exécution de la fonction deleteHotel() présente dans le modèle, avant de réafficher le tableau des hôtels.

La modèle se présente comme ceci :

```
public function deleteHotel($id){  
    $sql = "Delete from hotel Where NUM_HOTEL = ?";  
  
    $req = $this->conn->prepare($sql);  
    $req->bindValue(1, $id);  
  
    $req->execute();  
}
```

Et voilà, la fonctionnalité de suppression est fonctionnelle. Maintenant, le site dirige vers le tableau et on peut constater que l'enregistrement sélectionné n'est plus présent.

From:  
<https://ppe.boonum.fr/> - **AP.SIO**

Permanent link:  
<https://ppe.boonum.fr/doku.php?id=slam:ws:2021:sio:ap3.1:equipe2:supprimer>

Last update: **2021/11/10 08:14**





# Le clic droit personnalisé

Pour que le projet de gestion des hôtels soit digne d'une œuvre de Mozart, nous avons choisi de faire un menu contextuel personnalisé pour la gestion des hôtels.

Le fonctionnement est très simple :

1. L'utilisateur fait un clic droit
2. JQuery va empêcher l'apparition du menu par défaut
3. On affiche une div sous forme de menu contenant les boutons
4. L'utilisateur choisi le bouton souhaité
5. Le menu disparaît

Mais comment cela se déroule en programmation ?

Dans un premier temps, nous demandons si l'utilisateur fait un clic droit sur le tableau :

```
$('.rowhotel').contextmenu((e)=> {})
```

Puis j'ajoute l'affichage de ma div de menu au coordonnées de la souris de l'utilisateur, et j'empêche l'apparition du menu par défaut avec preventDefault() :

```
$('.rowhotel').contextmenu((e)=> {
    e.preventDefault();

    $('.menuperso').css('display', 'block');
    $('.menuperso').css('top', `${e.clientY}px`);
    $('.menuperso').css('left', `${e.clientX}px`);
})
```

Et nous devons ajouter le fait que si l'utilisateur clic ailleurs, le menu s'enlève, que ce soit un clic droit ou gauche, ou également sur un bouton du menu (Qui porte la class 'btn') :

```
$(document).click(function() {
    $('.menuperso').css('display', 'none');
})

$('.btn').click(function() {
    $('.menuperso').css('display', 'none');
})
```

Voici la division qui affiche le menu :

```
<div class="menuperso">
<button class="btn b0">Actualiser<i class="fas fa-sync-alt" aria-
hidden="true" style="float: right; margin-right: 12px"></i></button>
<div class="contextline"></div>
<button class="btn b1">Modifier<i class="fas fa-tools" aria-hidden="true"
style="float: right; margin-right: 12px"></i></button>
```

```
<button class="btn b2">Supprimer<i class="fas fa-trash-alt" aria-  
hidden="true" style="float: right; margin-right: 12px"></i></button>  
<div class="contextline"></div>  
<button class="btn b3">Surligner<i class="fas fa-highlighter" aria-  
hidden="true" style="float: right; margin-right: 12px"></i></button>  
</div>
```

Voilà le menu, pour une touche de beauté, il faut ajouter du css :

```
.menuperso{  
    position: absolute;  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    flex-direction: column;  
    width: 120px;  
    height: auto;  
    display: none;  
    z-index: 10;  
    background-color: #efefef;  
    padding: 2px;  
    border-radius: 3px;  
    -webkit-box-shadow: 0px 0px 17px -7px #000000;  
    box-shadow: 0px 0px 17px -7px #000000;  
    border: 1px solid #00000063;  
    opacity: .95;  
}
```

Le menu fonctionne. Voici un petit exemple pour mettre un écouteur sur un bouton du menu, par exemple Surligner :

```
$('.b3').click(function(){  
  
    var content = $('#'+index).children();  
    $(content).css('background-color', '#19106F');  
    $(content).css('color', 'white');  
})
```

*J'utilise le class b3 que j'ai mis sur le bouton surligner. (Ce bouton permet de surligner un enregistrement)*

From:  
<https://ppe.boonum.fr/> - AP.SIO

Permanent link:  
<https://ppe.boonum.fr/doku.php?id=slam:ws:2021:sio:ap3.1:equipe2:clidroit>

Last update: 2021/11/10 08:14



# Placer un congressiste dans un hôtel

Lors de notre arrivée sur la page nous sommes face à un formulaire composé de 2 files déroulantes, une pour les congressistes et une autre pour les hôtels. Nous avons également une case à cocher si nous souhaitons prendre le supplément petit déjeuner.



Voici la structure (conséquente) HTML/PHP de ce formulaire

```
<form action="./?controleur=hotellerie&todo=confirmer" method="post"
class="form_hotellerie">
    <select name="selectcongre" id="select_congre_id"
class="selec_hotellerie">
        <option value="no" disabled selected>Choisir un
congressiste</option>
        <?php
        foreach ($c->data["congressistes"] as $variable) {
            <?php
            <option value="<?php echo $variable->NUM_CONGRESSISTE."
            ".$variable->PRENOM_CONGRESSISTE." ".$variable->NOM_CONGRESSISTE ?>"><?php
            echo $variable->NUM_CONGRESSISTE." | ".$variable->PRENOM_CONGRESSISTE."
            ".$variable->NOM_CONGRESSISTE ?></option>
            <?php
            }
        }
    </select>
    <input class="inputhidden" type="text" name="idcongre" value="">
```

```
<select name="select_hotel" id="select_hotel_id"
class="selec_hotellerie">
    <option disabled selected value="no">Choisir un hôtel</option>
    <?php
    foreach ($c->data["hotels"] as $variable) {
    ?>
        <option value="<?php echo $variable->NUM_HOTEL."
". $variable->NOM_HOTEL ?>"><?php echo $variable->NUM_HOTEL." |
". $variable->NOM_HOTEL ?></option>
        <?php
        }
    ?>
    </select>
    <div>
        <span class="dej_hotellerie"> Petit-déjeuner </span>
        <input class="checkbox_hotellerie" type="checkbox" name=""
id="checkbox">
        <input class="inputhidden" type="text" name="input_prix"
value="">
    </div>
    <div class="cost_hotellerie"></div>
    <input type="submit" value="Réserver" class="submit_hotellerie"
name="submit_reserv">
    <i class="fas"></i>
</form>
```

Pour le remplissage des balises **<SELECT>**, toutes les informations sont récupérées grâce au **todo\_initialiser** qui se fait de base **au chargement de la page** :

```
public function todo_initialiser(){
    $this->data["hotels"] = $this->modele->getListeHotel();
    $this->data["congressistes"] =
    $this->modele->getListeCongressiste();
}
```

Ensuite si nous cliquons sur le bouton de type submit après avoir rempli le formulaire (voir les cas erreurs), un récapitulatif ainsi qu'un système de confirmation s'affiche sur la droite du formulaire. De plus, on peut voir que des petites modifications s'opèrent sur le formulaire à gauche (La valeur du submit change et un icon de chargement apparaît) :

### Outils d'hotellerie

3 | André Romuald

3 | Hotel de la recherche

**Petit-déjeuner** ☒

**Le prix est de : 58 €**

Confirmation en cours

### Récapitulatif

**Informations :**  
 Congressiste → 3 André Romuald  
 Hotel → 3 Hotel de la recherche

Supplément : 58 €

Annuler

Confirmer

Tout ceci est réalisé sans rechargement grâce à [AJAX](#) dont voici le code :

```
const form = $(".form_hotellerie");
form.submit(function (e) {
  e.preventDefault();
  var url = form.attr("action");
  var prix = "0€";
  var select1 = selectselect1.value;
  var select2 = selectselect2.value;

  if (select1 == "no") {
    // [...] Ceci correspond aux cas erreurs que nous verrons plus tard
  } else {
    $.ajax({
      url: url, // La ressource ciblée ici le formulaire
      type: "POST", // Le type de la requête HTTP,
      data: form.serialize(),
      dataType: "html", // Le type de données à recevoir, ici, du PHP

      success: function (response) {
        $(".hotellerie_contain_part1").css("width", "50%");
        $(".hotellerie_contain_part2").empty();
        $(".hotellerie_contain_part2").css("width", "50%");
        $(".hotellerie_contain_part2").css(
          "border-left",
          "2px solid #19106f"
        );
        $(".submit_hotellerie").val("Confirmation en cours");
        $(".form_hotellerie").css("pointer-events", "none");
        $(".hotellerie_contain_part1").attr(
          "title",
          "Annuler votre confirmation"
        );
        $(".fas").addClass("fa-spinner");
      }
    });
  }
});
```

```
$(".submit_hotellerie").addClass("no_click");
$(".hotellerie_contain_part1").mouseover(function () {
    $(".hotellerie_contain_part1").css("cursor", "not-allowed");
});

$(".hotellerie_contain_part2").append(response).hide().show("slow");
var confirm_no = document.querySelector(".confirm-no");
var confirm_yes = document.querySelector(".confirm-yes");

// Système de confirmation également ajouté avec AJAX
confirm_no.addEventListener("click", function check() {
    $(".hotellerie_contain_part1").css("width", "100%");
    $(".hotellerie_contain_part2").css("width", "0%");
    $(".hotellerie_contain_part2").empty();
    $(".hotellerie_contain_part2").css("border-left", "none");
    $(".submit_hotellerie").val("Confirmer");
    $(".form_hotellerie").css("pointer-events", "unset");
    $(".hotellerie_contain_part1").removeAttr("title");
    $(".fas").removeClass("fa-spinner");
    $(".submit_hotellerie").removeClass("no_click");//Ceci
correspond aux cas erreurs que nous verrons plus tard

    $(".hotellerie_contain_part1").mouseover(function () {
        $(".hotellerie_contain_part1").css("cursor", "auto");//Ceci
correspond aux cas erreurs que nous verrons plus tard
    });
});

confirm_yes.addEventListener("click", function check() {
    e.preventDefault();
    $.ajax({
        url: "?controleur=hotellerie&todo=confirmation", // La
ressource ciblée
        type: "POST", // Le type de la requête HTTP,
        data: form.serialize(),
        // data: { 'congredata=': select1, 'hoteldata=': select2 },
        dataType: "html", // Le type de données à recevoir, ici, du
PHP.$

        success: function (response) {
            $(".hotellerie_contain_part2")
                .append(response)
                .hide()
                .show("slow");

            $(".hotellerie_contain_part1").mouseover(function () {
                $(".hotellerie_contain_part1").css("cursor", "auto");//Ceci
correspond aux cas erreurs que nous verrons plus tard
            });
        }
    });
});
```

```

    },
  });
});
},
});
}
});

```

Si l'on clique sur **Non** lors de la confirmation, la div de droite contenant le **récapitulatif se vide** avec ce code (présent dans le code ci-dessus) :

```

confirm_no.addEventListener("click", function check() {
    $(".hotellerie_contain_part1").css("width", "100%");
    $(".hotellerie_contain_part2").css("width", "0%");
    $(".hotellerie_contain_part2").empty();
    $(".hotellerie_contain_part2").css("border-left", "none");
    $(".submit_hotellerie").val("Confirmer");
    $(".form_hotellerie").css("pointer-events", "unset");
    $(".hotellerie_contain_part1").removeAttr("title");
    $(".fas").removeClass("fa-spinner");
    $(".submit_hotellerie").removeClass("no_click");

    $(".hotellerie_contain_part1").mouseover(function () {
        $(".hotellerie_contain_part1").css("cursor", "auto");
    });
});

```

Si l'on clique sur **Oui** lors de la confirmation, **la requête d'insertion** s'effectue (avec [AJAX](#)) :

```

confirm_yes.addEventListener("click", function check() {
    e.preventDefault();
    $.ajax({
        url: "?controleur=hotellerie&todo=confirmation", // La
ressource ciblée
        type: "POST", // Le type de la requête HTTP,
        data: form.serialize(),
        // data: { 'congredata=': select1, 'hoteldata=': select2 },
        dataType: "html", // Le type de données à recevoir, ici, du
PHP.$

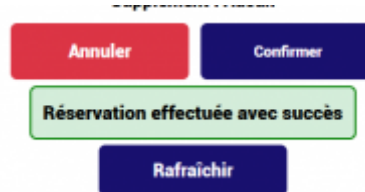
        success: function (response) {
            $(".hotellerie_contain_part2")
                .append(response)
                .hide()
                .show("slow");

            $(".hotellerie_contain_part1").mouseover(function () {
                $(".hotellerie_contain_part1").css("cursor", "auto");
            });
        },
    },

```

```
});  
});
```

Voici l'insertion de la **requête SQL** du modèle ainsi que l'affichage d'un bouton pour rafraichir grâce à la **vue confirmé** appelé dans le **todo\_confirmation** ci-après :



```
public function reservations($hotel,$congressiste){  
    $req1 = "UPDATE congressiste SET NUM_HOTEL = ? WHERE  
NUM_CONGRESSISTE = ?";  
    $sql = $this->conn->prepare( $req1 );  
    $sql->bindValue(1, $hotel);  
    $sql->bindValue(2, $congressiste);  
    $sql->execute();  
}  
<?
```

```
$congres = $this->post['congredata'];  
$hotels = $this->post['hoteldata'];  
$nom_congre_split = explode(" ", $congres); //Sépare le contenu de  
$congres sous forme d'array en fonction avec comme délimiteur : les espaces  
$hotels_split = explode(" ", $hotels); //Sépare le contenu de $hotels  
sous forme d'array en fonction avec comme délimiteur : les espaces  
$num_congre = $nom_congre_split[0]; //puis nous prenons la position  
1 du tableau soit l'id congressiste  
$num_hotels = $hotels_split[0]; // nous prenons la position 1 du  
tableau soit l'id hotel  
  
echo "<script>alert(\"$num_congre$num_hotels\")</script>";  
$this->data["hotelsconfirmation"] =  
$this->modele->reservations($num_hotels,$num_congre);  
$this->vue = "confirmé";
```

## Gestion des cas erreurs

- Impossible de confirmer si les **<SELECT>** sont vides



1 | Thierry Michel

Choisir un hôtel

Petit-déjeuner ☐

Sélectionner un hotel avant de confirmer.

Réserver

Choisir un congressiste

6 | Hotel des Utilisations

Petit-déjeuner ☐

Sélectionner un congressiste avant de confirmer.

Réserver

```

const form = $(".form_hotellerie");
form.submit(function (e) {
  e.preventDefault();
  var url = form.attr("action");
  var prix = "0€";
  var select1 = selectselect1.value; //Select congressiste
  var select2 = selectselect2.value; //Select Hotel

  if (select1 == "no") { //Si le select des congressistes est vide.
    $(".cost_hotellerie").empty();
    $(".cost_hotellerie")
      .append("Sélectionner un congressiste avant de confirmer.")
    //Affiche le message d'erreur pendant 3 secondes
      .hide()
      .show("slow");
    setTimeout(() => {
      $(".cost_hotellerie").empty();
    }, 3000);
  } else if (select2 == "no") { //Si le select des hotels est vide.
    $(".cost_hotellerie").empty();
    $(".cost_hotellerie")
      .append("Sélectionner un hotel avant de confirmer.") //Affiche le
    message d'erreur pendant 3 secondes
      .show("slow");
    setTimeout(() => {
      $(".cost_hotellerie").empty();
    }, 3000);
  } else {

```

### -Impossible de Modifier le formulaire une fois l'option de confirmation afficher

Lors de la pression du bouton de Réservation, la ligne permet d'empêcher à l'utilisateur d'interagir

avec le formulaire :

```
$(".form_hotellerie").css("pointer-events", "none");
```

From:

<https://ppe.boonum.fr/> - **AP.SIO**

Permanent link:

<https://ppe.boonum.fr/doku.php?id=slam:ws:2021:sio:ap3.1:equipe2:congressiste>

Last update: **2021/11/10 08:15**



# Sélection du petit-déjeuner

Dans le formulaire, si nous **sélections un hôtel** et que nous **cochons la case du supplément**, le prix du petit déjeuner **correspondant à l'hôtel** choisi s'affiche en dessous.

**3 | Hotel de la recherche**

## Petit-déjeuner ☒

### Le prix est de : 58 €

	NUM_HOTEL	NOM_HOTEL	ADRESSE_HOTEL	NOMBRE_ETOILES	PRIX_PARTICIPANT	PRIX_SUPPL
<input type="checkbox"/> <small>Editer</small> <small>4 Copier</small> <small>Supprimer</small>	1	Hôtels les angles morts	10 rue de la plage	4	172	34
<input type="checkbox"/> <small>Editer</small> <small>4 Copier</small> <small>Supprimer</small>	2	Hôtel Morisy ID Beach	12 place des anges	3	120	14
<input type="checkbox"/> <small>Editer</small> <small>4 Copier</small> <small>Supprimer</small>	3	Hotel de la recherche	45 rue de la loi	5	214	58

Cette affichage se fait sans **aucun rafraichissement** de page grace à l'outil [AJAX](#) et également grace au système controleur/modele de PHP orienté objet avec le **todo\_getprix** qui appelle cette **fonction** :

```
public function getPrixforHotel($hotel){
    $sql = "select PRIX_SUPPL from hotel where NUM_HOTEL = ?";
    $req = $this->conn->prepare( $sql );
    $req->bindValue(1, $hotel);
    $req->execute();

    return $req->fetch(PDO::FETCH_OBJ);
}
```

## Gestion des cas erreurs

- Impossible de sélectionner un prix sans avoir d'hôtel

1 | Thierry Michel

Choisir un hôtel

**Petit-déjeuner ☒**  
*Veuillez d'abord sélectionner un hotel.*

```
checkbox.addEventListener("change", function check() {
```

```
if (document.getElementById("checkbox").checked) {  
    //Si la checkbox est checked  
    var select = selectselect2.value;  
    if (select == "no") { //Si le select des hôtels est vide alors :  
        $(".cost_hotellerie").empty(); //Vide la div  
        $(".cost_hotellerie")  
            .append("Veuillez d'abord sélectionner un hotel.") //Affiche le  
message d'erreur pendant 3 secondes  
            .hide()  
            .show("slow");  
        setTimeout(() => {  
            $(".cost_hotellerie").empty();  
        }, 5000);  
    } else {  
        $.ajax({  
            url: "?controleur=hotellerie&todo=getprix", // La ressource ciblée  
            type: "POST", // Le type de la requête HTTP,  
            data: "hotels=" + select,  
            dataType: "html", // Le type de données à recevoir, ici, du PHP.  
            success: function (response) {  
                $(".cost_hotellerie").append(response).hide().show("slow");  
            },  
        });  
    }  
} else {  
    $(".cost_hotellerie").empty();  
}  
});
```

From:

<https://ppe.boonum.fr/> - **AP.SIO**

Permanent link:

<https://ppe.boonum.fr/doku.php?id=slam:ws:2021:sio:ap3.1:equipe2:petitdej>

Last update: **2021/10/12 22:31**



# Utilisation de AJAX



Lors de ce PPE nous avons fréquemment utilisé l'outil **AJAX** de **jQuery** pour nos requêtes PHP. Mais laissez nous vous présenter ce qu'il en est.

**AJAX** pour **A**synchronous **J**avaScript and **X**ML correspond à un groupe de méthodes et de moyens visant à permettre d'établir **une communication asynchrone** entre le **navigateur** et le **serveur**, c'est à dire **sans rechargement de page**, ce qui représente un véritable **gain de fluidité** et de propreté lors de la navigation sur le site web. L'objet que AJAX utilise est le **XMLHttpRequest**, cet objet peut être utilisé en **Javascript natif**, or il est relativement lourd à comprendre et à utiliser nous avons donc opté pour la méthode de **jQuery** qui est beaucoup plus simple et compréhensible. Elle se présente sous cette forme :

```
$('.b0').click(function(){
    $.ajax({
        url: '?controleur=hotels&todo=afficher',
        type: 'POST',
        data: '',
        dataType: 'html',

        success: function(response) {
            $(".content").replaceWith(response);
        },
    });
})
```

Dans **AJAX**, l'**url** ci dessus correspond à la **ressource ciblée** (c'est à dire une page et ici une action à effectuer avec le modèle), ensuite le **type** correspond à **celui de la requête HTTP que nous voulons effectuer** (POST ou GET), les **data** correspondent **au données que l'on veut faire passer en requête HTTP**, et le **dataType** correspond **au mode d'affichage de la réponse** (c'est à dire sous forme HTML ici). Si la requête **AJAX** est un **succès** alors il affiche la **réponse** à cette dernière (dans la div avec la class "content ici).

## Notre utilisation d'AJAX - Suppression d'un hôtel :

Voici un exemple de notre utilisation d'**AJAX**, ici pour la **suppression d'un hôtel**. Dans l'**url**, nous appelons le **contrôleur** ainsi que le **modèle** pour supprimer un hôtel. Le **data** correspond à l'id de l'hôtel (soit la ligne que nous avons sélectionné dans le tableau des hôtels).

```
$.ajax({  
    url: '?controleur=hotels&todo=supprimer', // La ressource ciblée  
    type: 'POST', // Le type de la requête HTTP,  
    data: 'idHotel='+datas[0], // datas[0] === ID  
    dataType: 'html', // Le type de données à recevoir, ici, du PHP.  
  
    success: function(response) {  
        $(".content").append(response);  
    },  
});
```

Voici le **todo** pour **supprimer**, on peut voir que à la fin de la suppression, il affiche la vue **afficherhotel**, ce qui correspond au **rechargement de la page**, or tout ceci est réalisé en **AJAX** donc sans **rechargement**.

```
public function todo_supprimer(){  
    $this->modele->deleteHotel($this->post['idHotel']);  
  
    $this->data["hotels"] = $this->modele->getListeHotel();  
    $this->vue = "afficherhotel";  
}
```

## Notre utilisation d'AJAX - Gestion hôtellerie :

Voici un second exemple, ici pour l' **affichage du prix lorsque la croix est cochée**. Dans l'**url**, nous appelons le **contrôleur** ainsi que le **modèle** pour afficher le prix. Le **data** correspond à la valeur du select (soit l'id de l'hôtel sélectionné).

```
$.ajax({  
    url: "?controleur=hotellerie&todo=getprix", // La ressource ciblée  
    type: "POST", // Le type de la requête HTTP,  
    data: "hotels=" + select,  
    dataType: "html", // Le type de données à recevoir, ici, du PHP.  
    success: function (response) {  
        $(".cost_hotellerie").append(response).hide().show("slow");  
    },  
});
```

}

From:  
<https://ppe.boonum.fr/> - **AP.SIO**

Permanent link:  
<https://ppe.boonum.fr/doku.php?id=slam:ws:2021:sio:ap3.1:equipe2:ajax>

Last update: **2021/11/10 08:12**

