

Collège Technique **Aumôniers du Travail**

Enseignement de **promotion sociale**

185 Grand’rue 6000 Charleroi

Tel. 071.285.905

**Projet de développement WEB**

Code de l’unité de formation : ***7534 30 U32 D1***

Partie(s) concernée(s) : ***Exercices***



Domaine de formation :

* ***Bachelier en informatique de gestion***

Nom du chargé de cours : ***Mr De Guglielmo***

Table des matières

[Introduction 4](#_Toc514430651)

[Serie 1 : Hello World 5](#_Toc514430652)

[Serie 2 : Découpe de l’examen 5](#_Toc514430653)

[Serie 3 : Les boucles 5](#_Toc514430654)

[Exercice 1 : 5](#_Toc514430655)

[Exercice 2 : 5](#_Toc514430656)

[Exercice 3 : 5](#_Toc514430657)

[Exercice 4 : 6](#_Toc514430658)

[Série 4 : Les tableaux PHP 7](#_Toc514430659)

[Exercice 1 : 7](#_Toc514430660)

[Exercice 2 : 7](#_Toc514430661)

[Exercice 3 : 8](#_Toc514430662)

[Série 4 : Mise en place de Git 9](#_Toc514430663)

[Série 5 : Formulaires et Session 10](#_Toc514430664)

[Exercice 1 10](#_Toc514430665)

[Exercice 2 11](#_Toc514430666)

[Exercice 3 11](#_Toc514430667)

[Exercice 4 11](#_Toc514430668)

[Exercice 5 11](#_Toc514430669)

[Série 6 : Les classes 12](#_Toc514430670)

[Exercice 1 12](#_Toc514430671)

[Exercice 2 13](#_Toc514430672)

[Série 7 : Mise en place du MVC et connexion à la DB 14](#_Toc514430673)

[Mise en place du MVC 14](#_Toc514430674)

[Connexion à la DB 15](#_Toc514430675)

[Série 8 : Mise en place de la structure pour afficher de l’information 16](#_Toc514430676)

[Série 9 : Mise en place de l’accès DB 18](#_Toc514430677)

[Série 10 : Recherches 19](#_Toc514430678)

[Série 11 : Exercices JQuery 20](#_Toc514430679)

[Exercices 20](#_Toc514430680)

[Exercice à mettre en place dans le projet 20](#_Toc514430681)

[Série 12 : Mise en place d’Ajax pour activer et désactiver un utilisateur 22](#_Toc514430682)

[Dans le contrôleur, corrigez la page5.php 22](#_Toc514430683)

[Dans le contrôleur, dupliquez tabletest\_tab.php en utilisateurs\_tab.php 22](#_Toc514430684)

[Dans la page qui génère le tableau HTML, ajouter les différentes informations ID et PK  ( vue/utilisateurs\_tab): 23](#_Toc514430685)

[Poser un listener Ajax sur la colonne Actif : /js/employees.js 24](#_Toc514430686)

[Dans le contrôleur, créer un Php récepteur du formulaire Ajax : /controler/utilisateur\_active.php 25](#_Toc514430687)

[Dans le Modele, adapter la classe utilisateurs : 26](#_Toc514430688)

[Créer la procédure stockée 27](#_Toc514430689)

[*Récapitulatif HTML5* 28](#_Toc514430690)

# Introduction

En vue du développement de votre projet, vous allez avoir besoin de plusieurs outils :

-Wamp : Le serveur WEB.

-GIT : Le gestionnaire de sources.

Pour télécharger Wamp :

<http://www.wampserver.com/>

Créer un compte sur GitHUB :

<https://github.com/>

Tous les exercices doivent être créés dans la racine de votre site

Chaque série correspond à une page web. Il vous sera précisé à chaque fois le nom de la page Web ainsi que son répertoire (vous comprendrez par la suite l’importance de cette arborescence).

Exemple 🡺 Série 1doit correspondre à ceci :

***projetweb2017/serie1/index.php***

NB : Etc. Il vous est obligatoire que vos travaux soient synchronisés avec le Cloud (Git) de manière à ne **JAMAIS** rien perdre !

# Serie 1 : Hello World

1 A la découverte de PHP, effectuez un echo de hello World

2 Créez une seconde page et utilisez la fonction require

# Serie 2 : Découpe de l’examen

Reprenez l’examen d’HTML et effectuez une découpe selon le MVC.

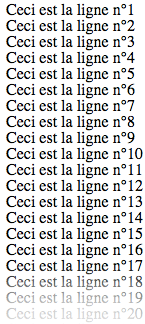
Rendez dynamique le TITLE de votre page en utilisant des variables. Chaque page doit avoir un TITLE différent.

# Serie 3 : Les boucles

* Dupliquez votre répertoire série 2 vers série 3 ;
* Ajoutez une page3.php et ajoutez cette nouvelle page dans le menu ;
* Cette nouvelle page va pouvoir accueillir le résultat des exercices suivants.

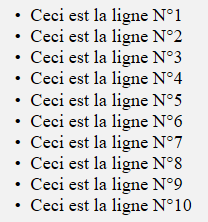
## Exercice 1 :

Ecrire une boucle qui permet d’afficher dix phrases :



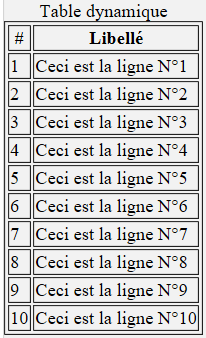
## Exercice 2 :

Ecrire une boucle qui crée la liste à puce suivante :



## Exercice 3 :

Ecrire une boucle qui génère le tableau dynamique suivant (pas forcément avec le même look)



## Exercice 4 :

Ecrire le tableau dynamique suivant :

-Les lignes paires doivent apparaitre en rouge, les autres en bleu.



# Série 4 : Les tableaux PHP

Dupliquez la série 3.

Ajoutez un nouveau point de menu « Tableaux »

C’est dans cette nouvelle page que vous allez effectuer les exercices.

## Exercice 1 :

Créez un tableau indicé 1 dimension PHP reprenant les jours de la semaine et affichez le dans un tableau HTML.

|  |
| --- |
| Jour |
| Lundi |
| Mardi |
| Mercredi |
| Jeudi |
| Vendredi |
| Samedi |
| Dimanche |

## Exercice 2 :

Créez un tableau associatif pour chaque jour de la semaine.

🡺Dans les éléments ‘lundi’ jusque ‘vendredi’, je dois retrouver la valeur école, pour samedi et dimanche on retrouvera ‘maison’.

Le résultat doit être affiché dans la page :

|  |  |
| --- | --- |
| Jour | Ce que je fais |
| Lundi | Ecole |
| Mardi | Ecole |
| Mercredi | Ecole |
| Jeudi | Ecole |
| Vendredi | Ecole |
| Samedi | Maison |
| Dimanche | Maison |

Le contenu de la colonne « Jour » est l’association, Ecole est la valeur stockée dans la zone.

## Exercice 3 :

Créez un tableau indicé de 52 éléments (= 52 semaines), dans chaque élément, on retrouvera tous les jours de la semaine comme à l’exercice précédent.

|  |  |  |
| --- | --- | --- |
| Semaine | Jour | Ce que je fais |
| 1 | Lundi | Ecole |
| Mardi | Ecole |
| Mercredi | Ecole |
| Jeudi | Ecole |
| Vendredi | Ecole |
| Samedi | Maison |
| Dimanche | Maison |
| 2 | Lundi | Ecole |
| Mardi | Ecole |
| Mercredi | Ecole |
| Jeudi | Ecole |
| Vendredi | Ecole |
| Samedi | Maison |
| Dimanche | Maison |

# Série 4 : Mise en place de Git

Pour la mise en place de la suite du cours, nous allons utiliser le gestionnaire de codes sources GitHub.

Pour cela, nous avons besoin de créer un compte sur la plateforme :

🡺 [www.github.com](http://www.github.com)

Nous aurons aussi besoin d’un logiciel jouant le rôle de client GitHub de manière à gérer les interactions :

🡺 <https://desktop.github.com/>

Lexique de Git :

🡺 <http://www.robusta.io/content/tutoriel/git/start-git.html>

Le tutoriel pour découvrir GitHub :

🡺 <https://openclassrooms.com/courses/gerer-son-code-avec-git-et-github>

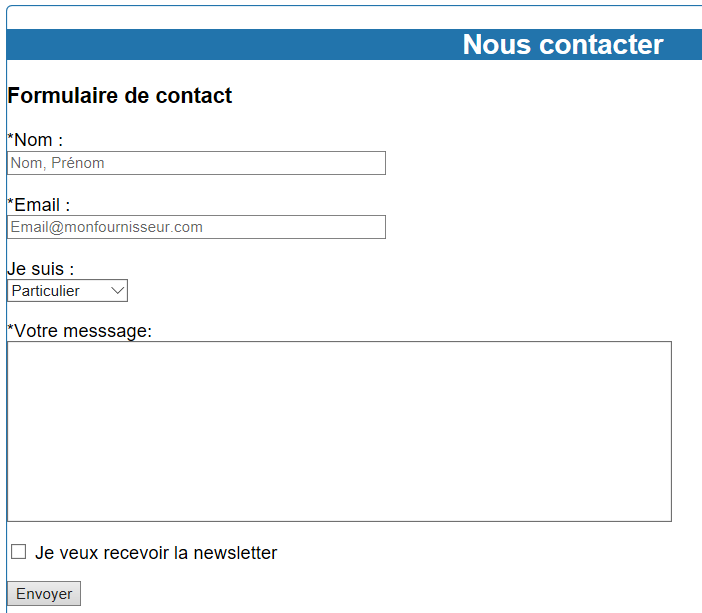
# Série 5 : Formulaires et Session

Dupliquez la série 4.

## Exercice 1

Ajoutez une nouvelle page contenant un formulaire :

Les zones avec un « \* » sont obligatoires



JESUIS

EMAIL

NOM

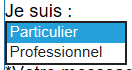
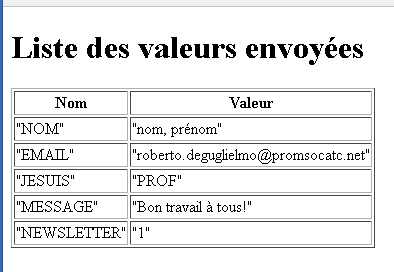
NEWSLETTER

MESSAGE

Valeurs de la liste déroulante : Valeurs envoyées par le formulaire :

PART

PROF



Votre formulaire doit pointer sur cette nouvelle page.

Lorsque votre page va recevoir le formulaire, elle va recharger les informations dans le formulaire.

De cette manière, vous pourrez recharger votre formulaire en cas d’erreur.

## Exercice 2

Les tests suivants doivent être faits :

-La zone nom doit reprendre une virgule : XXX , XXX

-La zone Email doit reprendre un @ ainsqi qu’un point.

Pour chaque erreur repérée, un libellé en rouge doit apparaitre en dessous de la zone testée.

## Exercice 3

Si le formulaire est correct, la zone NOM doit être enregistrée dans la session dans la zone « UTILISATEUR\_NOM ». La zone « UTILISATEUR\_OK » doit être = 1.

## Exercice 4

Dans le Header :

-Si la zone « UTILISATEUR\_OK » est affectée à 1 , affichez « Bienvenue » ainsi que le contenu de la zone « UTILISATEUR\_NOM ».

## Exercice 5

Ajoutez un bouton « se déconnecter » juste à côté du « Bienvenue NOM, Prénom ».

Ce bouton est un formulaire qui va pointer vers une nouvelle page php réceptrice de votre formulaire doit détruire la session et faire un require sur la première page de votre site.

# Série 6 : Les classes

Comme à l’habitude, dupliquez le projet 5.

## Exercice 1

Ajoutez une nouvelle page (avec son point de menu, etc…) dans laquelle vous allez placer votre combat.

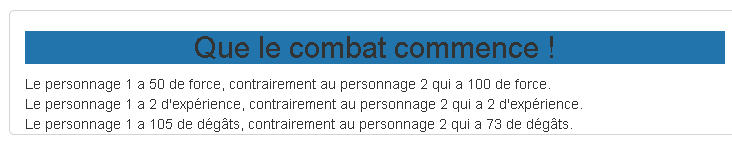
Vous allez aussi créer une nouvelle classe PHP Personnage.

🡺Pour rappel, 1 classe php = un fichier php !

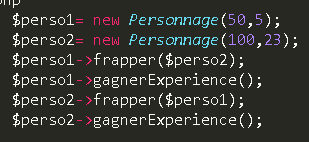
Utilisez le tutoriel classroom pour faire combattre les deux personnages.

<https://openclassrooms.com/courses/programmez-en-oriente-objet-en-php/utiliser-la-classe>

Le résultat attendu est celui-là :



Voici l’enchainement des opérations :



echo 'Le personnage 1 a ', $perso1->getforce(), ' de force, contrairement au personnage 2 qui a ', $perso2->getforce(), ' de force.<br />';

echo 'Le personnage 1 a ', $perso1->getexperience(), ' d\'expérience, contrairement au personnage 2 qui a ', $perso2->getexperience(), ' d\'expérience.<br />';

echo 'Le personnage 1 a ', $perso1->getdegats(), ' de dégâts, contrairement au personnage 2 qui a ', $perso2->getdegats(), ' de dégâts.<br />';

## Exercice 2

Vous allez créer une nouvelle classe. Celle-ci va vous permettre de générer le code HTML de vos formulaires. Il s’agit d’une classe qui sera instanciée. Cela va vous permettre d’avoir différents objets formulaires.

Nous allons donc avoir besoin d’une classe « Form ». Lors de son instanciation, les paramètres suivant doivent être passés : name ; id ; méthode ; action.

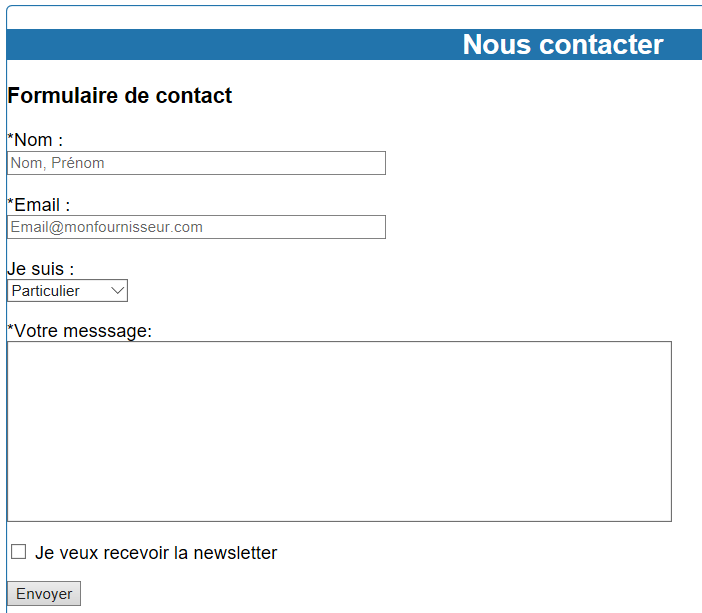
Pour ajouter des éléments au formulaire, nous allons avoir besoin de fonctions telles que setText et setEmail donc le prototype est le suivant : label ;

Name ; id ; required ; placeholder ; value.

Pour disposer d’un bouton submit, vous allez créer une fonction setSubmit dont le prototype est : name ; value.

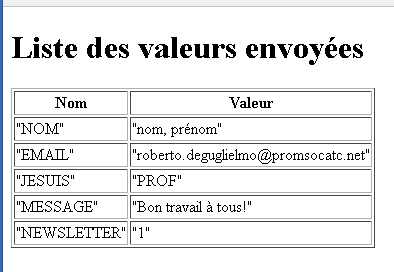
Une fonction getForm va vous permettre de récupérer le code HTML correspondant au formulaire ainsi généré.

Utilisez cette classe pour créer un nouveau formulaire contenant une zone nom et Email idem à la série 5.



NOM

EMAIL



# Série 7 : Mise en place du MVC et connexion à la DB

🡺Dupliquez la série 6 en série 7.

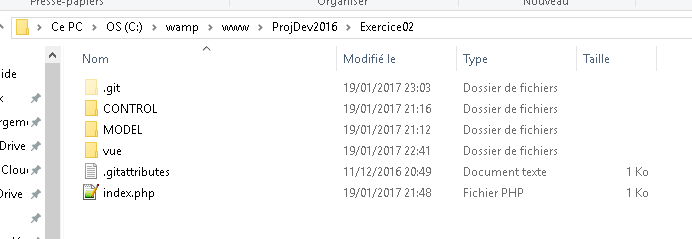
## Mise en place du MVC

Etape 1 : Déplacer le contenu du répertoire dans un nouveau nommé « VUE »

Etape 2 : Créer les répertoires CONTROL et MODEL au même niveau que la vue

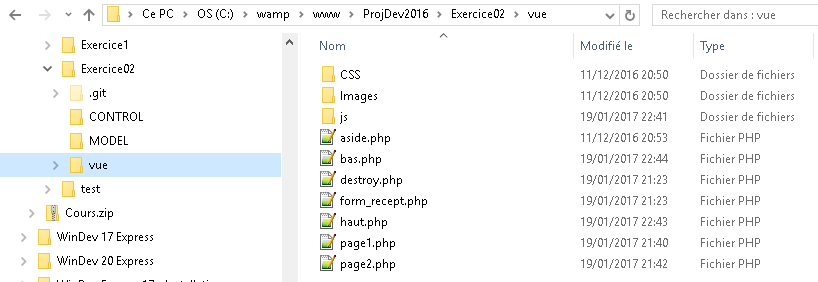
Etape 3 : Les page1.php et page2.php n’ont plus besoin de leurs include « Haut/up.php » et « bas/down.php »

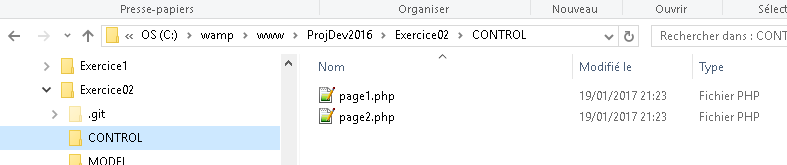
Vous devez après cette étape avoir cette arborescence :



Etape 4 : Dans vue, créer les fichiers page1.php et page2.php. Ces pages se chargeront de faire les includes nécessaires des fichiers se trouvant dans la vue.

Après cette étape, vous devez avoir l’arborescence suivante :





Etape 6 : Du fait du changement d’arborescence, tous les liens doivent être corrigés :

* Le menu : Les liens doivent maintenant pointer vers le contrôleur  (Exemple: « ../control/page1.php » )
* Les liens vers les images, et CSS doivent être « ../vue/images/image.jpg »

## Connexion à la DB

Vous disposez d’une archive ZIP reprenant différents fichiers. Il s’agit du Relational Mapping qui est à intégrer dans votre projet.

Vous retrouverez donc les fichiers php nécessaires pour faire fonctionner un MVC, mais aussi le répertoire MODEL permettant d’accéder à la base de données.

Faites dérouler votre script SQL sur votre phpMyAdmin, configurez les users et mot de passe dans Model.php et faites fonctionner l’exemple.

Une fois l’exemple fonctionnel, regardez à l’intégrer dans votre projet série 7 (Dans la dernière page de votre menu).

# Série 8 : Mise en place de la structure pour afficher de l’information

Vous allez dupliquer la série 7.

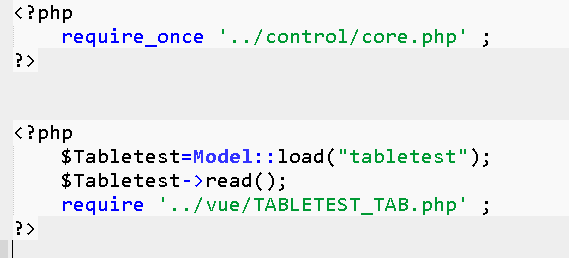
L’objectif de cette série est de structurer votre code pour standardiser la manière dont vous allez utiliser votre contrôleur pour accéder à l’information (grâce au modèle) et l’afficher (grâce à la vue).

Exemple :

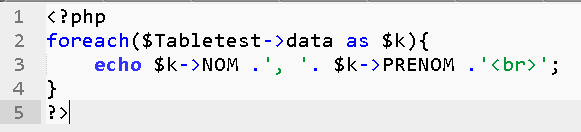
Pour un fichier donné « TABLETEST », vous allez donc avoir une classe dans le modèle (jusque-là rien ne change). L’objectif est d’afficher le contenu de cette table dans une page.

Pour cela, nous allons créer un fichier « TABLETEST\_TAB.php » dans le contrôleur ainsi que dans la vue.

Dans le contrôleur, nous allons avoir ceci « TABLETEST\_TAB.php » :



Et dans la vue « TABLETEST\_TAB.php »:

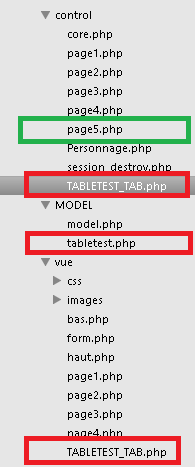


De cette manière, il vous suffira d’ajouter un require de votre /control/TABLETEST\_TAB.php pour disposer de l’affichage du contenu de votre table.

Ajoutez un nouveau point de menu ainsi qu’une nouvelle page dans le contrôleur (ainsi que dans le point de menu et faites un require de votre nouvelle page.

Voilà à quoi cela doit ressembler au final :

Ma nouvelle page 5 qui est dans mon point de menu, ainsi que mes fichiers TABLETEST\_TAB + tabletest.php



# Série 9 : Mise en place de l’accès DB

Dupliquez la série 8 !

Sur base du document fourni en annexe (Gestion des sessions et authentification):

1. Nous allons devoir créer la table utilisateurs dans la base de données

CREATE TABLE `utilisateurs` (  
`utilisateur` varchar(255) NOT NULL,  
`code` varchar(255) DEFAULT NULL,  
`nom` varchar(255) DEFAULT NULL,  
`prenom` varchar(255) DEFAULT NULL,  
`admin` int(11) DEFAULT NULL,  
`actif` int(1) DEFAULT NULL,  
PRIMARY KEY (`utilisateur`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

INSERT INTO `test`.`utilisateurs` (`utilisateur`, `code`, `nom`, `prenom`, `admin`, `actif`) VALUES ('Administrateur', 'code', 'De Guglielmo', 'Roberto', '1', '1');

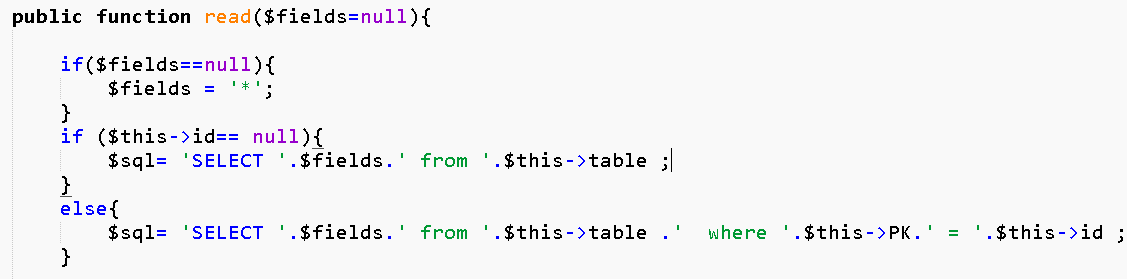
1. Ajouter un enregistrement pour l’utilisateur Administrateur
2. Créer le fichier utilisateurs dans le répertoire model.
3. Créer le fichier login.php dans le répertoire vue.
   1. Il reprendra le formulaire nécessaire pour l’authentification
4. Créer le fichier login.php dans le répertoire contrôleur
   1. C’est lui qui va recevoir le formulaire de login, il se chargera de vérifier l’authentification et ainsi de préparer la session de manière à débloquer l’accès au site.
5. Modifier core.php
   1. Démarrer la session
   2. Si l’url courante n’est pas \*login.php et que la variable ‘UTILISATEUR’ de la session n’est pas garnie, redirection automatique vers control/login.php
6. Modifier vue/haut.php
   1. Conditionner l’affichage du menu uniquement si la variable UTILISATEUR’ de la session est pas garnie
7. Ajouter deux fonctions statiques à la classe Utilisateur pour se charger de la vérification de l’utilisateur, ainsi que de son niveau d’autorisation.

# Série 10 : Recherches

Nous allons maintenant faire évoluer notre application de manière à pouvoir effectuer des recherches.

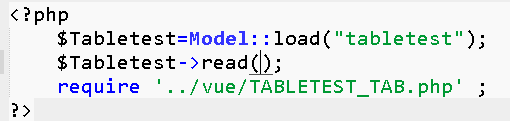
Pour cela, nous allons tout d’abord devoir faire évoluer notre classe **Model**

🡺Elle doit permettre d’accueillir une clause **Where.** Pour cela, nous allons ajouter un paramètre $where à la fonction READ.



Il doit être ajouté une condition supplémentaire dans le cas ou l’ID est null. Si le contenu de la variable $where est différent de null et ‘’ , on ajoute la clause where ainsi que le contenu de la variable dans $sql.

Il est conseillé à ce point d’effectuer un test pour vérifier que la classe fonctionne correctement. Pour cela, ajoutez un paramètre dans le read présent dans tabletest\_tab.php de manière à demander « prenom = ‘roberto’ »



Visualisez le point de menu de manière à vérifier que le contenu affiché correspond.

Maintenant que cela fonctionne, nous allons ajouter un formulaire dans cette même page.

(Veillez à utiliser l’objet form !). Le formulaire doit pointer vers la page5.php.

Ensuite, veillez à ajouter la gestion de la récupération du formulaire de manière créer la String SQL et la passer à la fonction *read()* .

🡺$where.=" upper(prenom) like upper('%".$VarPrenom."%' )";

…Voilà, cela devrait fonctionner !

# Série 11 : Exercices JQuery

## Exercices

Ressources :

<https://www.w3schools.com/jquery/jquery_events.asp>

<https://openclassrooms.com/courses/un-site-web-dynamique-avec-jquery?status=published>

Sur base de la feuille Excel, effectuez l’exercice et consignez les résultats dans celle-ci.

Pour mener à bien l’exercice, vous devez aller à l’URL :

<http://dero-promsocatc.alwaysdata.net/jquery/>

## Exercice à mettre en place dans le projet

Etape 1 : Créer le répertoire JS dans la vue. Ce répertoire va contenir tous nos scripts JS

Etape 2 : Créer une Init.js dans ce nouveau répertoire. Celui-ci reprendra la déclaration de nos bibliothèques ainsi qu’un include des JS dont nous allons avoir besoin.

Etape 3 : Ajouter le require d’init.js dans notre « bas/down.php »

-Lors d’un clic sur H2, je veux qu’en bas du ASIDE soit ajouté une ligne reprenant la classe et le contenu du titre.

-Lors d’un clic sur un paragraphe, emballer celui-ci pour qu’il soit en italique.

-Lors d’un clic sur une image :

* Ajouter une nouvelle <div> au body
* Effectuer un Fade sur les sections et aside
* Effectuer un fadein sur la div
* Dans la div, ajouter l’image
* Ajouter une zone cliquable permettant de supprimer cette div et annuler le Fade sur section + aside.

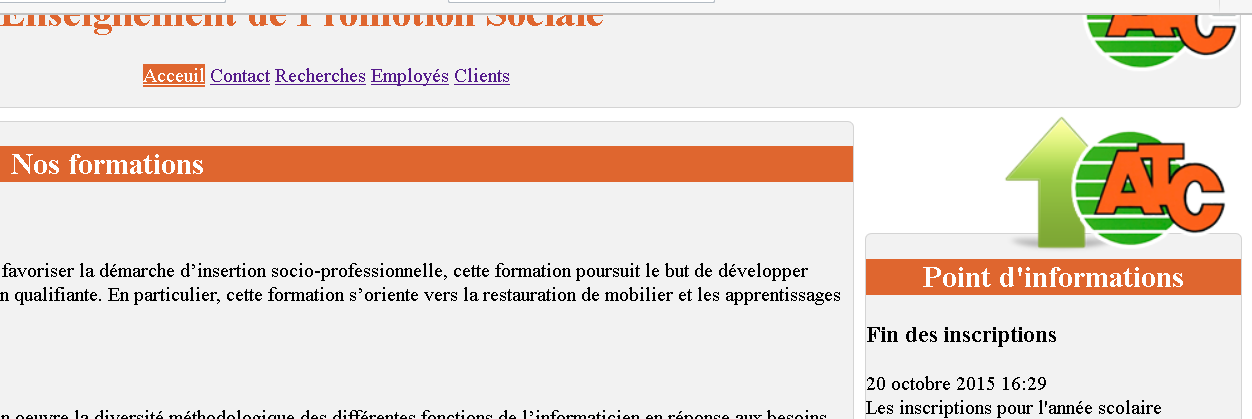


Intégrer dans sa page un lien pour remonter en haut de page.

Vous allez intégrer dans votre code sur base des informations disponibles sur le site suivant.

<http://www.paulund.co.uk/how-to-create-an-animated-scroll-to-top-with-jquery>

🡺Ajoutez-y le Logo ATC une fois l’intégration terminée



# Série 12 : Mise en place d’Ajax pour activer et désactiver un utilisateur

L’objectif de l’exercice est de permettre l’activation ou la désactivation d’un utilisateur en utilisant la technologie Ajax.

Dans cet exercice, nous allons avoir un exemple complet. Nous allons traverser la pile complète MVC.

Un listener va être posé sur l’interface graphique (via JQuery)pour effectuer un call Ajax, arriver au contrôleur d’activation ou désactivation pour arriver à la classe Utilisateurs du modèle.

Cette mise à jour s’effectue via procédure stockée.

Il vous suffira d’appliquer chaque étape pour intégrer cette fonctionnalité dans votre projet.

## Dans le contrôleur, corrigez la page5.php

<?php

$Montitle= 'Mon Title 2';

require\_once '../control/core.php' ;

require '../vue/haut.php' ;

require '../control/utilisateurs\_tab.php' ;

require '../vue/bas.php' ;

?>

## Dans le contrôleur, dupliquez tabletest\_tab.php en utilisateurs\_tab.php

<?php

require\_once '../control/core.php' ;

?>

<?php

$VarPrenom ="";

$where ="";

if (isset($\_POST['PRENOM'])){

$VarPrenom=$\_POST['PRENOM'];

if($where!=""){

$where.=" and ";

}

$where.=" upper(prenom) like upper('%".$VarPrenom."%' )";

}

$monFormulaire = new Form('Formulaire','post','?');

$monFormulaire->addText('Prénom :','PRENOM','PRENOM',$VarPrenom,false,'Entrez ici le prénom recherché');

/\*$monFormulaire->addSubmit('VALIDER','Valider');\*/

echo $monFormulaire->getForm();

$Utilisateurs=Model::load("utilisateurs");

$Utilisateurs->read(null,$where);

require '../vue/utilisateurs.php' ;

?>

## Dans la page qui génère le tableau HTML, ajouter les différentes informations ID et PK  ( vue/utilisateurs\_tab):

<?php

$out = "";

$titre= '<tr>';

$titre\_trt= false;

foreach($Utilisateurs->data as $key => $element){

$out .= '<tr id ="'.$element->utilisateur.'" >';

foreach($element as $subkey => $subelement){

if($titre\_trt==false){

$titre .= '<th>'.$subkey.'</th>' ;

}

$out .= '<td id="'.$subkey.'">'.$subelement.'</td>' ;

}

if($titre\_trt==false){

$titre.= '</tr>';

}

$titre\_trt= true;

$out .= "</tr>";

}

$out = '<table id ="utilisateurs" >'.$titre.$out.'</table>';

echo $out;

?>

## Poser un listener Ajax sur la colonne Actif : /js/employees.js

NB : Ne pas oublier de déclarer jquery ainsi que le script suivant dans vue/bas.php

var clicActive = function(){

if($(this).text()=="0"){

$destination='utilisateur\_active.php';

}else{

$destination='utilisateur\_desactive.php';

};

var elem = $( this );

$.post( $destination, { code : $(this).parent().attr('id') } )

.done(function( data ) {

elem.empty();

elem.html(data);

}

);

};

$(function(){

$('#utilisateurs #actif').on('click',clicActive);

});

## Dans le contrôleur, créer un Php récepteur du formulaire Ajax : /controler/utilisateur\_active.php

Une fois la page active créée, la dupliquer pour créer le récepteur « utilisateur\_active.php »

<?php

require\_once '../control/core.php' ;

if( isset($\_POST['code'])){

$tmp\_utilisateur=Model::load("utilisateurs");

$tmp\_utilisateur->utilisateur\_active($\_POST['code']);

echo '1';

}else{

echo '0';

}

?>

## Dans le Modele, adapter la classe utilisateurs :

NB : Vous aurez besoin d’ajouter la fonction suivante dans la classe Model

<?php

class utilisateurs extends Model{

protected $table = "utilisateurs";

protected $PK = "utilisateur";

function utilisateur\_active($Utilisateur){

$sql= "call Utilisateur\_active('".$Utilisateur."') ";

$this->connection->query($sql);

}

function utilisateur\_desactive($Utilisateur){

$sql= "call Utilisateur\_desactive ('".$Utilisateur."') ";

$this->connection->query($sql);

}

}

?>

## Créer la procédure stockée

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `utilisateur\_active`(IN `idUtilisateur` varchar(255))

update utilisateurs set actif = 1 where utilisateur = idUtilisateur and actif = 0$$

DELIMITER ;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `utilisateur\_desactive`(IN `idUtilisateur` varchar(255))

update utilisateurs set actif = 0 where utilisateur = idUtilisateur and actif = 1$$

DELIMITER ;

# Récapitulatif HTML5

