

L'objectif de ce TD est de créer une interface de consultation et de saisie/suppression des données de la table « matière » créée au TD n° 5.

Page web modèle : <https://eric.univ-lyon2.fr/jdarmont/docs/web/td67controleur.php>

Vue – Étape 1

- 1 Copier/coller le gabarit td5vue.tpl.html créé lors du TD n° 5 sous le nom td67vue-tab.tpl.html. Y ajouter un lien hypertexte de libellé « Nouvelle matière » sous le tableau d'affichage des matières. L'URL cible du lien est un champ nommé [onshow.cible] concaténé au paramètre « ?suivant=form_ajout » (« ? » indique un paramétrage en fin d'URL, « suivant » est le nom du paramètre et « form_ajout » sa valeur, cf. TD 3-4). L'idée est qu'après avoir cliqué sur le lien, on effectue une action « form_ajout » qui affiche un formulaire d'ajout de matière. Globalement, l'URL est « [onshow.cible]?suivant=form_ajout ».
- 2 Créer un nouveau gabarit nommé td67vue-form.tpl.html contenant un formulaire dont la description suit. Ne pas oublier de labelliser les champs du formulaire.
 - a. URL cible (attribut « action ») : champ [onshow.cible], méthode : get
 - b. Champ caché de nom « suivant » et de valeur « ajout » (cf. question 1)
 - c. Champ texte de nom « codemat » de taille 3 et de taille maximum 3
 - d. Champ texte de nom « libelle » de taille 20 et de taille maximum 20
 - e. Champ texte de nom « coef » de taille 3 et de taille maximum 3
 - f. Boutons « reset » et « submit »

Modèle – Étape 1

AccesMatiere
\$pdo : Object \$qmat : Object
__construct() liste() ajouterMatiere() supprimerMatiere()

- 3 Créer un nouveau fichier nommé td67modele.class.php. Y inclure le fichier td5modele.class.php.
- 4 Créer une classe AccesMatiere contenant deux attributs privés \$pdo (identifiant de connexion) et \$qmat (objet de classe RQ1), un constructeur qui initialise \$this->pdo à un paramètre \$param_pdo et \$this->qmat à la requête qui affiche la liste des matières (s'inspirer fortement de la création de la première requête dans le script td5controleur.php ; le constructeur de \$this->qmat aura besoin du paramètre \$param_tbs, qui sera directement injecté parmi les paramètres de new RQ1).
- 5 Ajouter à la classe AccesMatiere une méthode publique nommée liste() qui exécute la requête \$this->qmat, puis affiche le résultat (s'inspirer de nouveau de l'exécution et de l'affichage de la requête \$qmat dans le script td5controleur.php).

Contrôleur – Étape 1

- 6 Créer un nouveau script nommé `td67controleur.php`. Y inclure les fichiers `connect.inc.php`, `tbs_class.php` et `td67modele.class.php`. Créer également un objet gabarit `TinyButStrong`.
- 7 Reproduire depuis le script `td5controleur.php` la structure « try/catch », avec uniquement la partie création de l'objet PDO et l'initialisation de la gestion des erreurs dans la partie « try ».
- 8 Dans la partie « try », après la création de l'objet PDO et l'initialisation de `$etatConnexion`, initialiser la variable `$cible` à la valeur `$_SERVER["PHP_SELF"]` (cf. gabarit `td5vue.tpl.html` ; nous allons de nouveau boucler sur le même fichier `td67controleur.php` comme dans le TD 3-4).
- 9 Toujours dans le script `td67controleur.php`, créer un objet `$acccmat` de classe `AccesMatiere` et appeler la méthode `liste()` pour `$acccmat`.

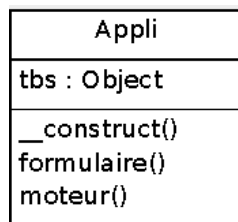
Test – Étape 1 (enfin !)

- 10 Téléverser les fichiers `td67vue-tab.tpl.html`, `td67vue-form.tpl.html`, `td67modele.class.php` et `td67controleur.php` sur le serveur `phpetu`.
- 11 Tester l'affichage de la liste des matières en vous connectant sur `td67controleur.php`. Une fois que cela fonctionne, supprimer l'appel à `liste()`.

Modèle – Étape 2

- 12 Dans `td67modele.class.php`, ajouter à la classe `AccesMatiere` une méthode publique nommée `ajouterMatiere()` prenant en paramètres un code matière, un libellé et un coefficient. Cette méthode doit insérer les valeurs transmises par les paramètres dans la table `MATIERE` (utiliser une requête paramétrée – T99).

Contrôleur – Étape 2



- 13 Dans un nouveau fichier `td67controleur.class.php`, créer une classe `Appli` contenant un attribut privé `$tbs` et un constructeur qui initialise `$this->tbs` à un paramètre `$param_tbs`.
- 14 Ajouter à la classe `Appli` une méthode privée (comme cette méthode ne sera pas appelée depuis l'extérieur de la classe, c'est plus sûr) nommée `formulaire()` qui affiche le gabarit du formulaire `td67form.tpl.html`.
- 15 Ajouter à la classe `Appli` une méthode publique nommée `moteur()` prenant en paramètre un objet `$acccmat` de classe `AccesMatiere`. Dans la méthode `moteur()`, définir une variable `$action` égale à l'action suivante demandée par l'utilisateur (`$_GET["suivant"]`) si cette dernière est définie (utiliser la fonction PHP `isset()`, qui retourne `true` si la variable passée en paramètre est définie), ou à une chaîne vide sinon.
- 16 Toujours dans la méthode `moteur()`, définir un « switch » sur la variable `$action`.
 - a Si l'action est « `form_ajout` », appeler la méthode `formulaire()`.
 - b Si l'action est « `ajout` », appeler la méthode `ajouterMatiere()` pour `$acccmat` en lui passant en paramètres les données transmises depuis le formulaire à l'aide de la méthode « `get` » (soit

`$_GET["codemat"]`, `$_GET["libelle"]` et `$_GET["coef"]`). Appeler ensuite la méthode `liste()` pour `$accmat`.

- c L'action par défaut est l'affichage de la liste des matières, effectué en appelant la méthode `liste()` pour `$accmat`.

Test – Étape 2

- 17 Téléverser le fichier `td67controleur.class.php`, puis l'inclure dans le fichier `td67controleur.php` après `td67modele.class.php`.
- 18 Dans la partie « try » du script `td67controleur.php`, après la création de l'objet `$accmat`, créer un objet `$appli` de classe `Appli` en lui passant en paramètre l'identifiant de l'objet TBS. Appeler ensuite la méthode `moteur()` pour `$appli` avec comme paramètre l'objet `$accmat` préalablement défini.
- 19 Tester l'ajout de quelques matières.

Vue – Étape 2

- 20 Dans le gabarit `td67vue-tab.tpl.html`, ajouter aux lignes du tableau une cellule `<td> </td>` contenant un lien « Effacer ». L'URL cible est le champ `[onshow.cible]` assorti des paramètres « `?suivant=suppr&codemat=[codemat.val;block=tr]` » (code de la matière courante ; cf. Question 1).

Modèle – Étape 3

- 21 Ajouter à la classe `AccesMatiere` une méthode publique nommée `supprimerMatiere()` prenant en paramètre un code matière. Cette méthode doit supprimer dans la table `MATIERE` le n-uplet correspondant au code matière (utiliser une requête paramétrée).

Contrôleur – Étape 3

- 22 Dans `td67controleur.class.php`, ajouter l'option suivante au switch de la méthode `moteur()`. Si la page suivante est « `suppr` », appeler la méthode `supprimerMatiere()` pour `$accmat` en lui passant en paramètre le code de matière transmis par l'URL (cf. Question 16b). Appeler ensuite la méthode `liste()` pour `$accmat`.

Test – Étape 3

- 23 Téléverser `td67vue-tab.tpl.html`, `td67modele.class.php` et `td67controleur.class.php` sur le serveur `phpetu`.
- 24 À partir de la page `td67controleur.php`, tester la suppression de quelques matières.
- 25 Vérifier la validité du code HTML et CSS de tous les éléments de l'application (affichage de la table `MATIERE` et formulaire de saisie/modification, notamment).
- 26 L'utilisation de la méthode « `get` » est-elle la plus sûre ? Comment remédier à ses inconvénients ?

Correction : Vue

```
<!DOCTYPE html> <!-- td67vue-tab.tpl.html -->
<html lang="fr">
  <head>
    <title>Mise à jour de base de données</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>[onshow.etatConnexion]</p>
    <table>
      <caption>Liste des matières</caption>
      <tr>
        <td>[codemat.val;block=tr]</td>
        <td>[libelle.val;block=tr]</td>
        <td>[coef.val;block=tr]</td>
        <td><a href="[onshow.cible]?suivant=suppr
          &codemat=[codemat.val;block=tr]">Effacer</a></td>
      </tr>
    </table>
    <p><a href="[onshow.cible]?suivant=form_ajout">Nouvelle matière</a></p>
  </body>
</html>
```

```
<!DOCTYPE html> <!-- td67vue-form.tpl.html -->
<html lang="fr">
  <head>
    <title>Mise à jour de base de données</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <form action="[onshow.cible]" method="get">
      <fieldset>
        <input type="hidden" name="suivant" value="ajout" />
        <p>
          <label for="codemat">Code matière</label>
          <input type="text" id="codemat" name="codemat" size="3"
            maxlength="3" />
        </p>
        <p>
          <label for="libelle">Libellé</label>
          <input type="text" id="libelle" name="libelle" size="20"
            maxlength="20" />
        </p>
        <p>
          <label for="coef">Coefficient</label>
          <input type="text" id="coef" name="coef" size="3"
            maxlength="3" />
        </p>
        <p>
          <input type="reset" value="Annuler" />
          <input type="submit" value="Valider" />
        </p>
      </fieldset>
    </form>
  </body>
</html>
```

Correction : Modèle

```
<?php // td67modele.class.php
```

```
require("td5modele.class.php");

class AccesMatiere {
    private $pdo; // Identifiant de connexion
    private $qmat; // Requête liste des matières

    function __construct($param_pdo, $param_tbs) {
        $this->pdo = $param_pdo;
        $this->qmat = new RQ1($this->pdo, $param_tbs,
            "SELECT * FROM MATIERE", "td67tab.tpl.html");
    }

    public function liste() {
        $this->qmat->executer();
        $this->qmat->afficher();
    }

    public function ajouterMatiere($codemat, $libelle, $coef) {
        $res = $this->pdo->prepare("INSERT INTO MATIERE VALUES(?, ?, ?)");
        $res->execute([$codemat, $libelle, $coef]);
    }

    public function supprimerMatiere($codemat) {
        $res = $this->pdo->prepare("DELETE FROM MATIERE WHERE codemat = ?");
        $res->execute([$codemat]);
    }
}
?>
```

Correction : Contrôleur

Fichier connect.inc.php

```
<?php // td67controleur.class.php
```

```
class Appli {
    private $tbs;

    function __construct($param_tbs) {
        $this->tbs = $param_tbs;
    }

    private function formulaire() {
        // Affichage du gabarit
        $this->tbs->LoadTemplate("td67form.tpl.html");
        $this->tbs->Show();
    }
}
```

```

public function moteur($accmat) {
    if (isset($_GET["suivant"])) $action = $_GET["suivant"];
    else $action = "";
    switch ($action) {
        case "form_ajout": // Formulaire d'ajout
            $this->formulaire();
            break;
        case "ajout": // Ajout
            $accmat->ajouterMatiere($_GET["codemat"], $_GET["libelle"],
                $_GET["coef"]);
            $accmat->liste();
            break;
        case "suppr": // Suppression
            $accmat->supprimerMatiere($_GET["codemat"]);
            $accmat->liste();
            break;
        default: // Liste des matières
            $accmat->liste();
            break;
    }
}
/* Question 28 : L'utilisation de la méthode "get" fait apparaître
toutes les données en URL, ce qui la surcharge et peut être gênant lorsqu'on
transfère des données délicates comme des mots de passe. De plus, un utilisateur
mal intentionné peut s'en servir pour insérer des valeurs à sa convenance, voire
tenter de pirater le système par injection de code SQL malicieux. Il est donc
préférable d'utiliser la méthode "post", et donc de remplacer les liens
"Nouvelle matière" et "Suppression" par des formulaires ne contenant qu'un
bouton. */
}
}
?>

```

```

<?php // td67contrôleur.php
require("connect.inc.php");
require("tbs_class.php");
require("td67modele.class.php");
require("td67contrôleur.class.php");
$tbs = new clsTinyButStrong;

try {
    // Connexion
    $c = new PDO("mysql:host=$host;dbname=$dbname", $login, $password);
    $etatConnexion = "Connexion OK";
    $cible = $_SERVER["PHP_SELF"];
    $c->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Moteur de l'application
    $accmat = new AccesMatiere($c, $tbs);
    //$accmat->liste();
    $appli = new Appli($tbs);
    $appli->moteur($accmat);

} catch(PDOException $erreur) {
    $etatConnexion = "Erreur : " . $erreur->getMessage();
}
?>

```