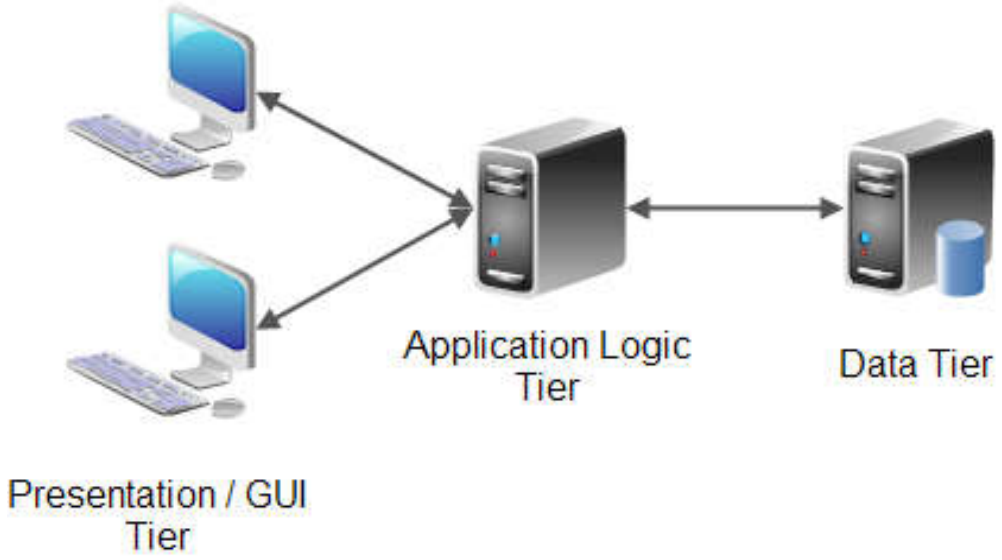
Pilotage d'une page web

Sommaire

Architecture n-tiers



Vient de l'anglais tiers → étage / niveau

Modélisation d'une application comme un empilement de 3 trois couches (architecture 3-tiers)

Rôle clairement défini :

- Présentation ;
- Traitement ;
- Accès aux données.

Couche présentation :

Correspond à l'affichage, la restitution sur le poste de travail, le dialogue avec l'utilisateur (HTML/CSS/JS) ;

Couche traitement :

Correspond à la mise en œuvre de l'ensemble des règles de gestion et de la logique applicative (PHP) ;

Couche accès aux données :

Correspondant aux données qui sont destinées à être conservées sur la durée, voire de manière définitive (SQL).

Les couches communiquent entres elles à travers un « modèle d'échange » que l'on peut assimiler à une liste de fonctions ou URL ;

Chaque fonction ou URL permet de rendre un service précis ;

Une couche N ne peut communiquer qu'avec la couche N-1 ou N+1 ;

Comme chaque service est clairement défini, une couche peut évoluer sans impacter ses voisins ;

L'ajout d'une fonctionnalité dans l'application peut avoir des impacts dans plusieurs couches.

L'architecture n-tiers a plusieurs objectifs :

• alléger la charge de travail des applications clientes ;

• gérer des clients hétérogènes (systèmes d'exploitations différents) ;

• introduire la notion de clients légers (notion de web) ;

• Améliorer la sécurité en coupant l'accès direct aux données, la couche intermédiaire doit vérifier l'intégrité des données en provenance du client ;

• rompre le lien de propriété entre l'application et les données (normalisation du modèle de base de données).

Patrons de conception

Un patron de conception ou design pattern est un arrangement caractéristique des parties d'une application ;

L'utilisation d'un patron de conception permet de respecter un certain nombre de bonnes pratiques en développement logiciel ;

Un patron de conception permet de répondre à une problématique (accès au données, arrangement architectural, ...) ;

L'élaboration de ces patrons se fait, la plupart du temps, d'après le retour d'expérience de concepteurs logiciel confirmés ;

Le patron ***Modèle Vue Contrôleur*** (MVC)

Modèle :

- contient les données et la logique en rapport avec les données ;
- il peut contenir une simple valeur ou une structure de données ;
- le modèle représente l'univers dans lequel s'inscrit l'application ;

Contrôleur :

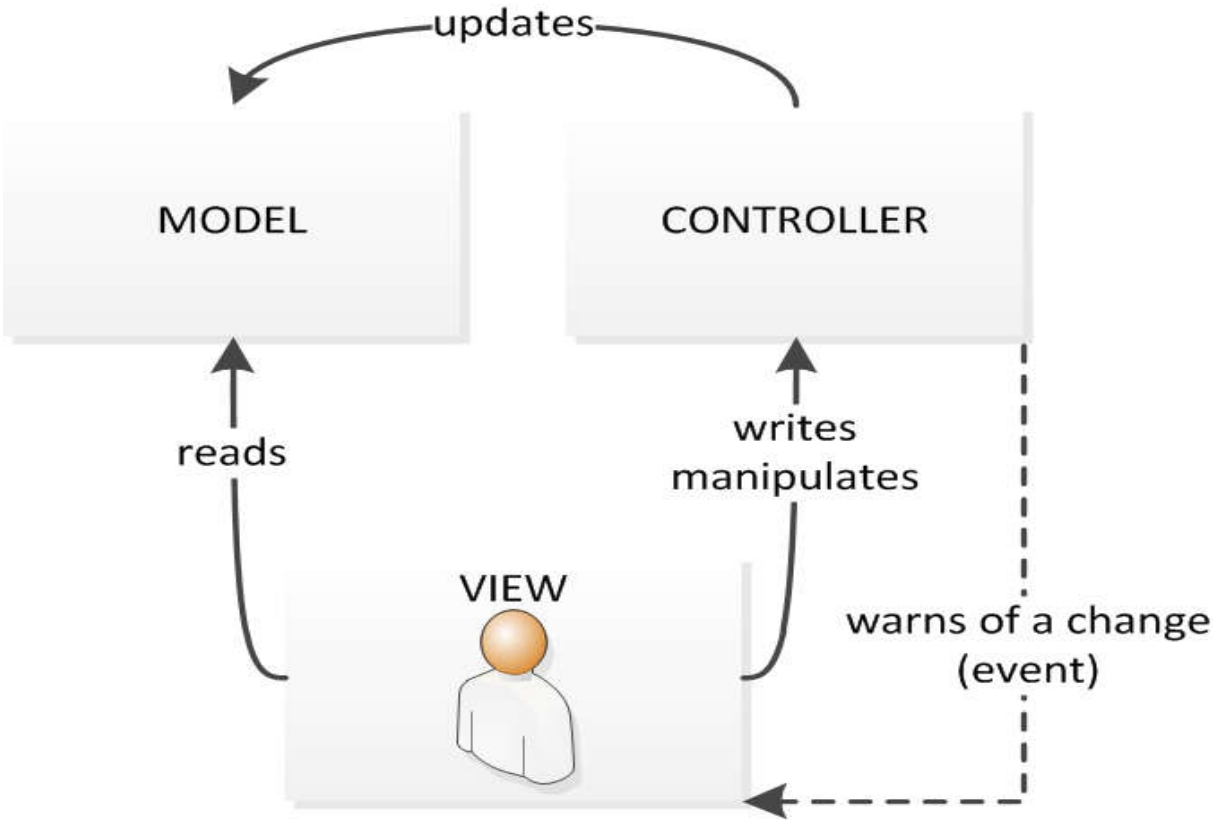
- module qui traite les actions de l'utilisateur ;
- modifie les données du modèle et de la vue ;

Le patron ***Modèle Vue Contrôleur*** (MVC)

Vue :

- partie visible (Graphical User Interface) ;
- la vue se sert du modèle qu'elle expose sous forme de formulaire, boutons, etc...
- elle contient des éléments visuels ainsi que le moyen d'afficher les données provenant du modèle ;
- elle obtient les données à afficher à travers une interface gérée par le contrôleur.
- elle met à jour le modèle en envoyant des messages (**A**pplication **P**rogramming **I**nterface) ;

Le patron ***Modèle Vue Contrôleur*** (MVC):



Le patron ***Database Access Object*** (DAO):

- les objets / variables en mémoire vive sont liés à des données persistantes (base de données, fichiers, …) ;
- le modèle DAO consiste à regrouper le code relatif à l'accès aux données dans une classe / fichier à part ;
- cela permet de ne pas « disperser » le code d'accès aux données dans la partie graphique ou métier ;
- ce modèle vient en complément du modèle MVC ;

Si on prend l'exemple d'un objet personne avec les attributs login et password, le DAO *DaoPersonne* **pourra** contenir les méthodes suivantes :

- createPersonne(login, password) → qui renvoie l'identifiant unique ;
- getPersonneById(id) → qui renvoie un objet personne ;
- modifyPersonne(id, login, password) → qui renvoie vrai ou faux ;
- deletePersonneById(id) → qui renvoie vrai ou faux.

C'est ce que l'on appelle le **CRUD** : **C**reate, **R**ead, **U**pdate, **D**elete !

Connecteur PDO

- **P**HP **D**ata **O**bjects (**PDO**) est une extension définissant l'interface pour accéder à une base de données avec PHP ;
- Elle constitue une couche d'abstraction entre l'application PHP et la base de données ;
- Elle permet de séparer la partie traitement (DAO) de la base de données ;
- Elle facilite donc la migration vers une autre base de données ;
- Elle est plus lente que l'utilisation du connecteur spécifique ;

```
try{
   $db = new PDO('mysql:host=localhost;dbname=wis', 'root', 'password');
}catch(Exception $e){
   echo "Échec : " . $e->getMessage();
}
```

- Les requêtes à la base de données se font au travers d'instructions paramétrées (*prepared statements*) !
- Les instructions paramétrées permettent d'éviter les attaques par injection SQL (attaques encore très courantes) ;
- Les instructions paramétrées permettent d'écrire une requête SQL avec des paramètres d'entrées non spécifiés ;
- Les noms des paramètres sont précédés du caractère ':' ;
- Les paramètres sont remplacés dans l'ordre SQL, avant son exécution ;
- Les valeurs des paramètres sont transférées à PDO par le biais d'un tableau associatif.

Voici un exemple de requête préparée :

```
try{
    $db = new PDO('mysql:host=localhost;dbname=wis', 'root', 'password');
    $select = $db->prepare('SELECT * FROM personnes WHERE id=:id');
    $select->execute(array('id' => $id));
    $result = $select->fetchAll();
    var_dump($result[0]);
}catch(Exception $e){
    echo "Échec : " . $e->getMessage();
}
```

Ou encore :

```
...
$update=$db->prepare('UPDATE personnes SET password=:password WHERE
id=:id');
$update->execute(array('password' => $password,'id'=>$id));
...
```

Traitement des informations utilisateur

Comment envoyer les informations utilisateur ?

→ grâce à un formulaire

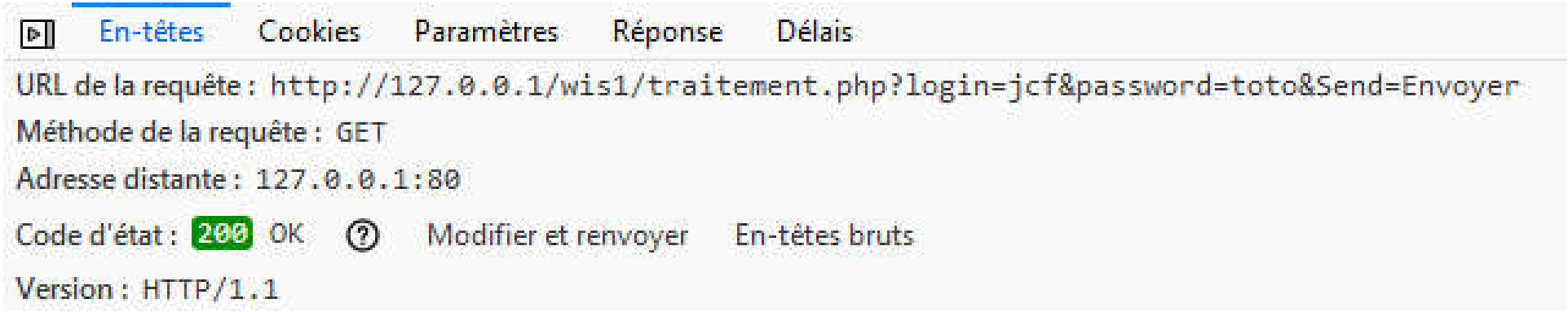
- GET
- POST

Le champ ***method*** permet de choisir entre GET et POST ;

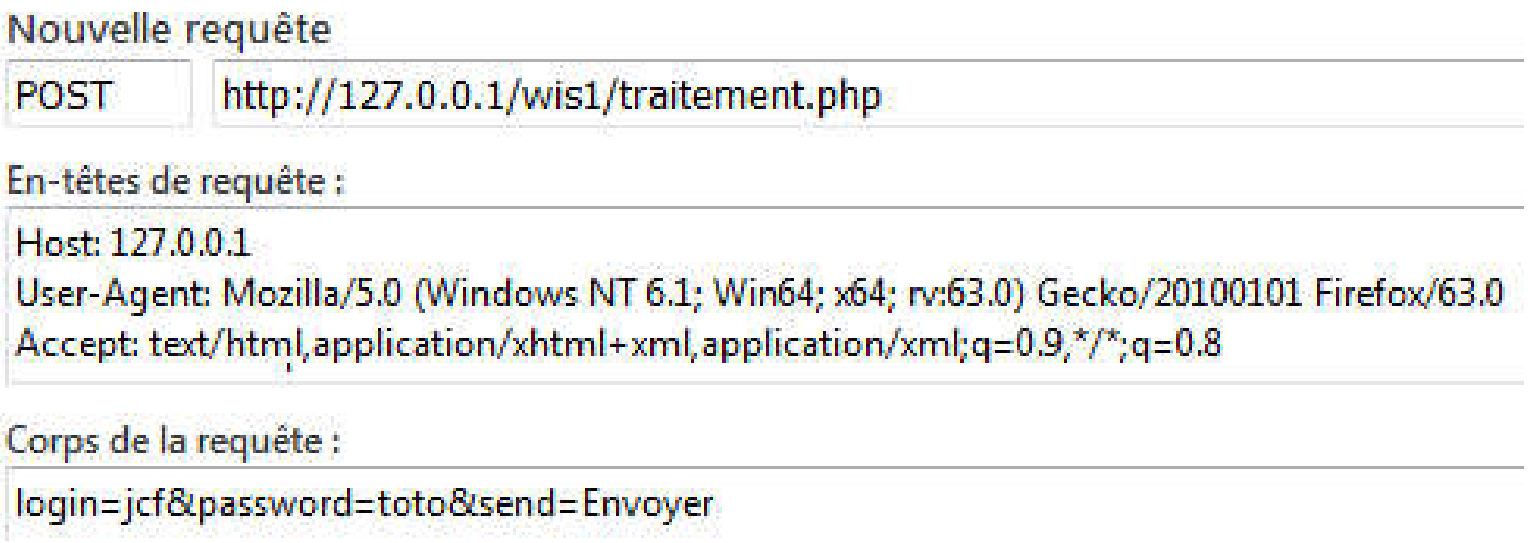
Le champ ***action*** permet de choisir la page où seront envoyées les données ;

```
<form method="POST" action="traitement.php">
    Login:
    <input type="text" name="login">
    <br>
    Password:
    <input type="password" name="password">
    <br>
    <input type="submit" name="send">
</form>
```

GET : les données passent dans l'URL



POST : les données passent dans le corps de la requête



Comment récupérer les informations utilisateur ?

→ grâce à du PHP :

GET → les données sont dans le tableau $_GET
POST → les données sont dans le tableau $_POST

Ce sont des tableaux associatifs (ensemble clé / valeur)

Il y a autant d'éléments dans le tableau que de champs ***input*** dans le formulaire

La clé correspond à la valeur de l'attribut ***name***

La valeur est celle de l'attribut ***value***

```
<input type="submit" name="send" value="titi">
<input type="submit" name="send" value="tata">
```

La récupération se fait en utilisant le tableau :

```
$login = $_POST["login"];
$password = $_POST["password"];
```

La valeur peut être vide…

Notice: Undefined index: login in C:\wamp64\www\wis1\traitement.php on line *3*

Call Stack

#	Time	Memory	Function	Location
1	0.0000	362504	{main}( )	...\traitement.php:0

Notice: Undefined index: password in C:\wamp64\www\wis1\traitement.php on line *4*

Call Stack

#	Time	Memory	Function	Location
1	0.0000	362504	{main}( )	...\traitement.php:0

Cela met en évidence l'importance des tests !

On a trois possibilités dans le cas de la récupération de la variable:

- elle n'existe pas ;
- elle existe mais n'a pas de valeur ;
- elle existe et possède une valeur ;

Si on veut attribuer une valeur à chacun des états on peut procéder de la sorte :

- n'existe pas → false ;
- existe mais vide → true ;
- possède une valeur → la valeur ;

Ce qui signifie, que pour chaque variable, il faudrait faire les tests suivants :

```
$login = false ;
if (isset ( $_GET ["login"] )) {
    if(!empty($_GET ["login"])){
        $login = $_GET ["login"] ;
    }else{
        $login = true ;
    }
}
...
```

→ la quantité de code qu'il faut écrire, si on veut faire un traitement efficace des valeurs, est colossale.

On peut externaliser ce code dans une fonction :

```
function getVar($name) {
    if (isset ( $_GET [$name] )) {
        if (! empty ( $_GET [$name] )) {
            return $_GET [$name];
        }
        return true;
    }
    return false;
}

$login = getVar("login");
$password = getVar("password");
```