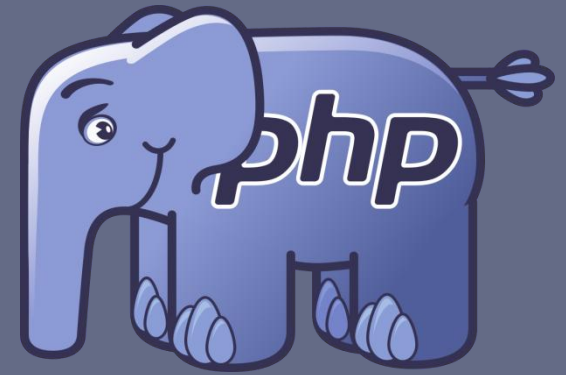


PHP framework



CODEIGNITER



PHP > 7.2

CodeIgniter 4.0.4

Présentation du Framework PHP



CI4_0

Gwénaél LAURENT – glaurent001@gmail.com

Plan

2

3

□ Framework CodeIgniter 4.0.4

8

□ L'architecture MVC, c'est quoi ?

13

□ PHP objet

16

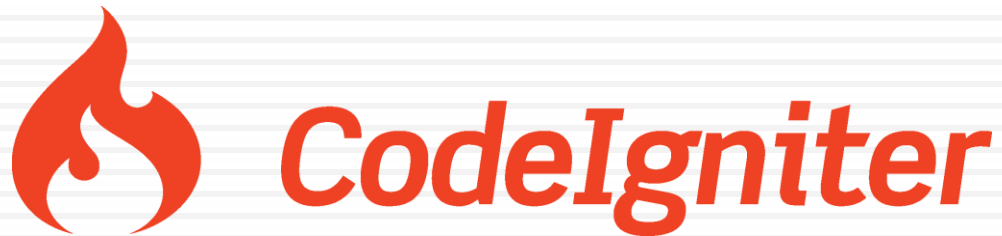
□ Les namespaces PHP

18

□ Les tableaux PHP (array)

3

Framework CodeIgniter 4.0.4



Un framework, c'est quoi ?

4

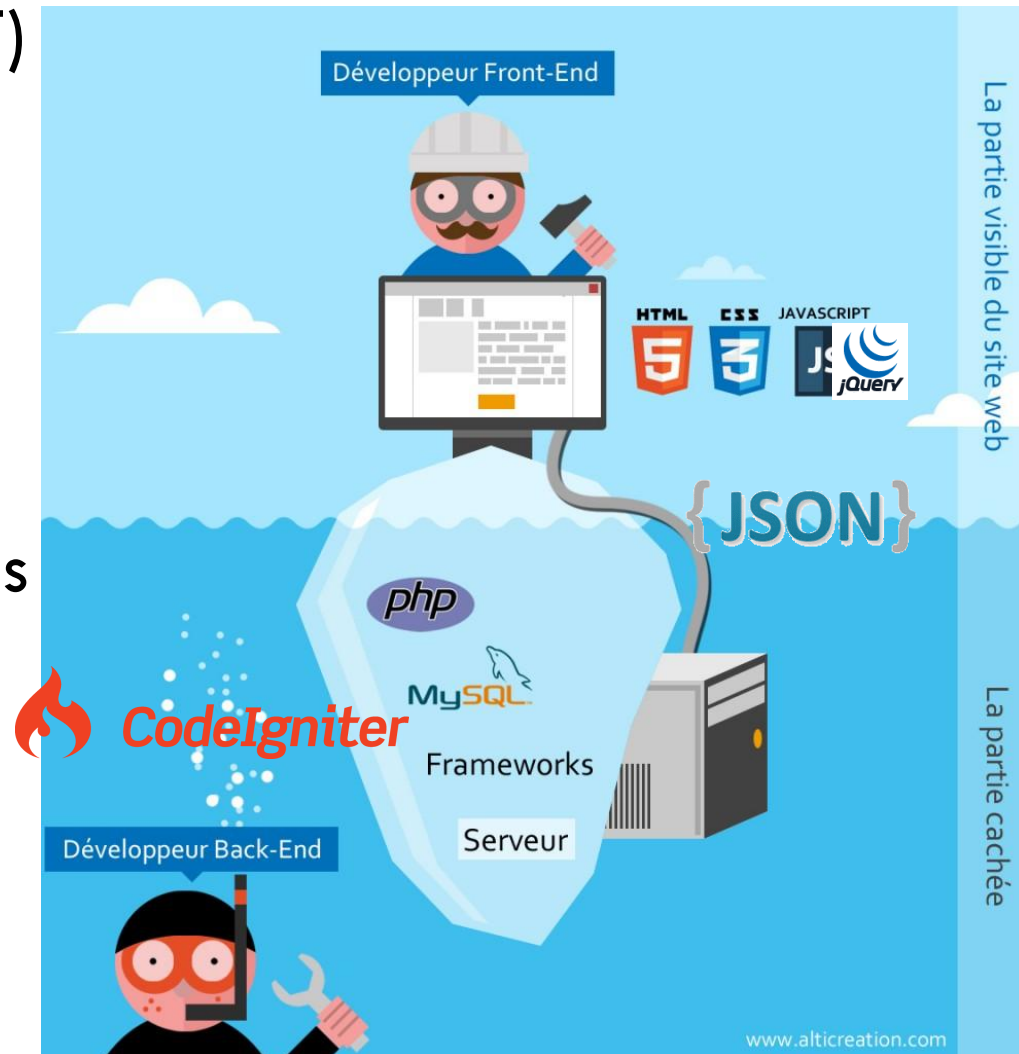
C'est un ensemble de bibliothèques et de conventions permettant le développement rapide d'applications.

- Objectifs :
 - ▣ Gain de temps
 - ▣ Performance
 - ▣ Mise à jour
 - ▣ Fiabilité

Le framework CodeIgniter 4.0.4

5

- Open Source (licence MIT)
- Langage PHP objet
 - ▣ Version >7.2 et PSR4
- Structure le **Back-end**
 - ▣ Architecture MVC
- Léger : minimum + options
- Rapide :
 - ▣ Temps d'apprentissage
 - ▣ Génération de la page
- Naturellement SEO
(Search Engine Optimization)



Bibliothèques et fonctions

6

□ En fonction des besoins, on peut charger :

▣ Des bibliothèques : **Libraries**

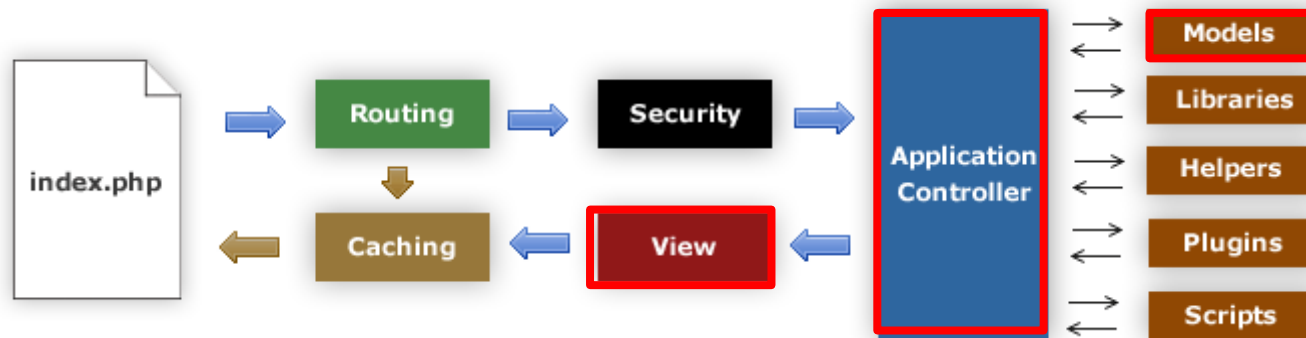
- Configuration, cache, accès aux BDD ...
- Pagination , validation de formulaires ...
- Fichiers, images,
- Sécurité, cryptage

▣ Des fonctions utilitaires : **Helpers**

- URL, fichiers, tableaux, email, captcha, ...

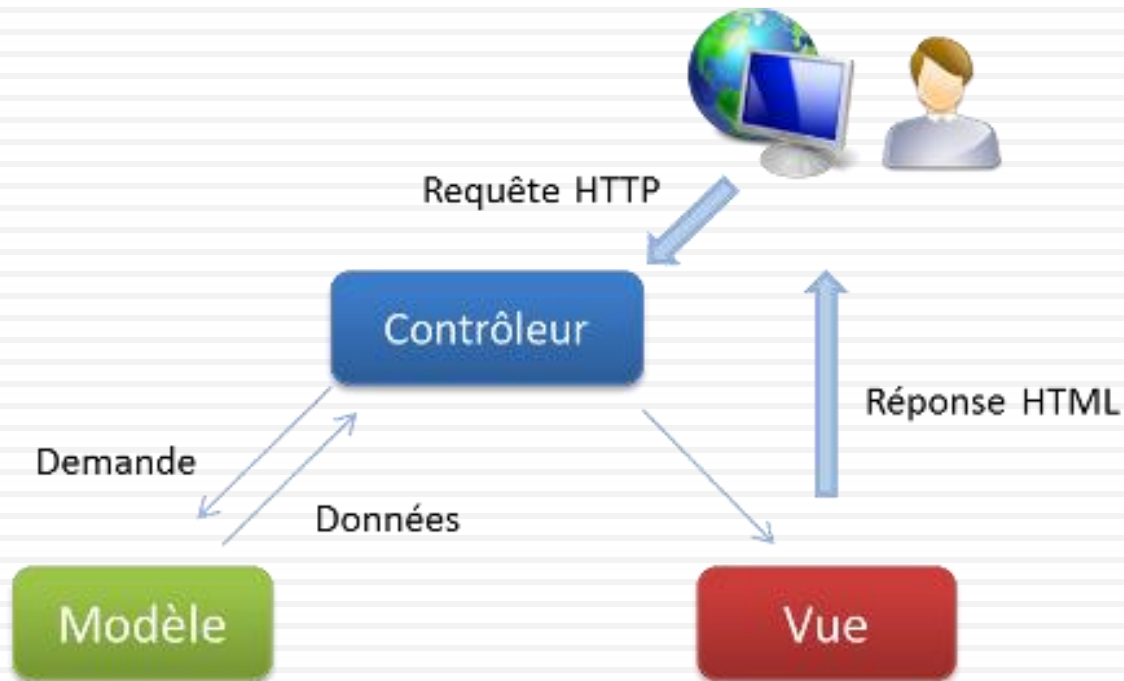
Fonctionnement de CodeIgniter

7



1	index.php	Ce sera toujours le fichier index.php, situé à la racine, qui sera appelé en premier.
2	Routing	C'est le routeur du framework. C'est lui qui récupérera l'URL et la décomposera en actions. Il a pour rôle de trouver le contrôleur à appeler.
3	Caching	Une fois que le routeur a fini son travail, le module de cache va regarder s'il existe des fichiers mis en cache pour cette action. Dans le cas d'une réponse positive, ces fichiers vont être renvoyés au navigateur.
4	Security	Cette partie va sécuriser toutes les données entrantes : cookies, variables get, post, etc. C'est la dernière étape avant le contrôleur.
5	Application Controller	Le contrôleur analyse la requête
6	Models, Libraries & Helpers	Le contrôleur peut fait appel à différents éléments (base de données, bibliothèques)
7	View	Le contrôleur renvoie une vue. Les vues sont les fichiers qui contiennent le code HTML.

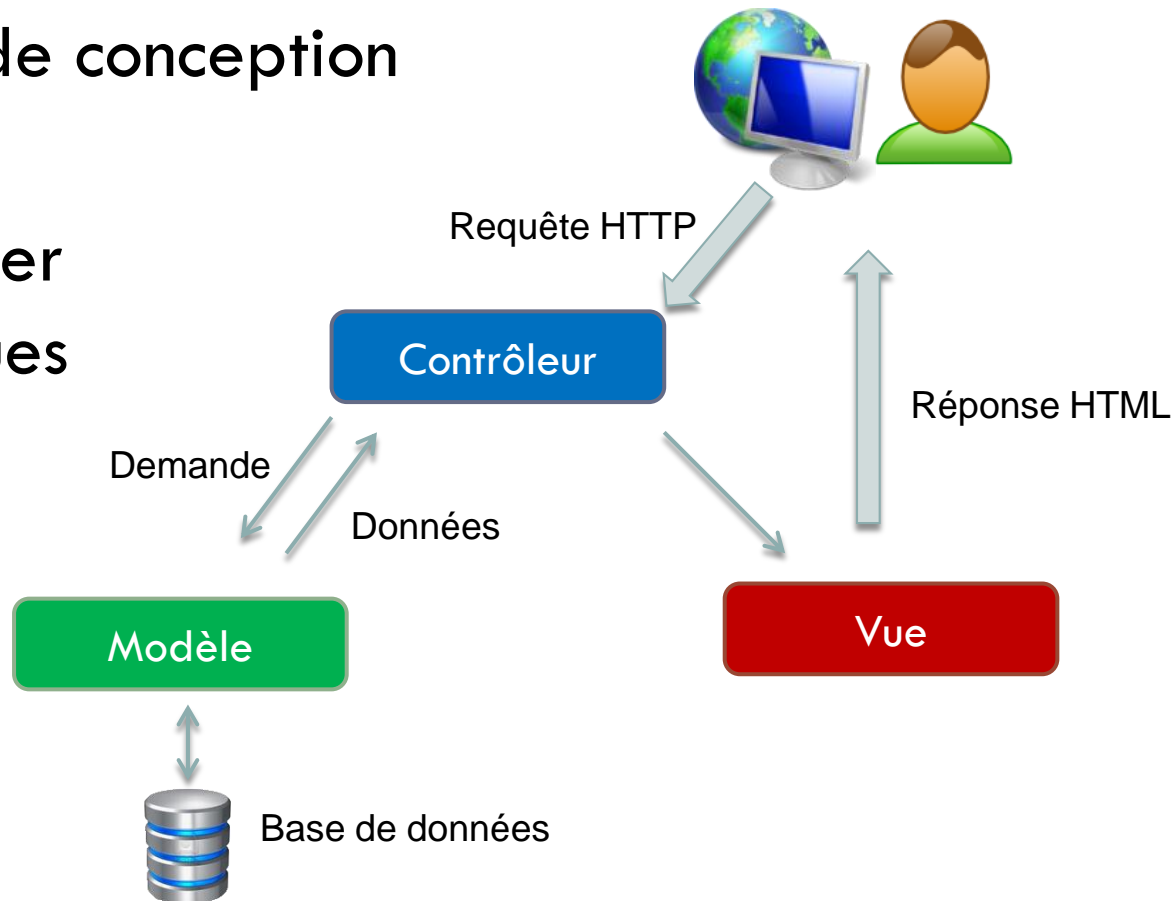
L'architecture MVC, c'est quoi ?



MVC, c'est quoi ?

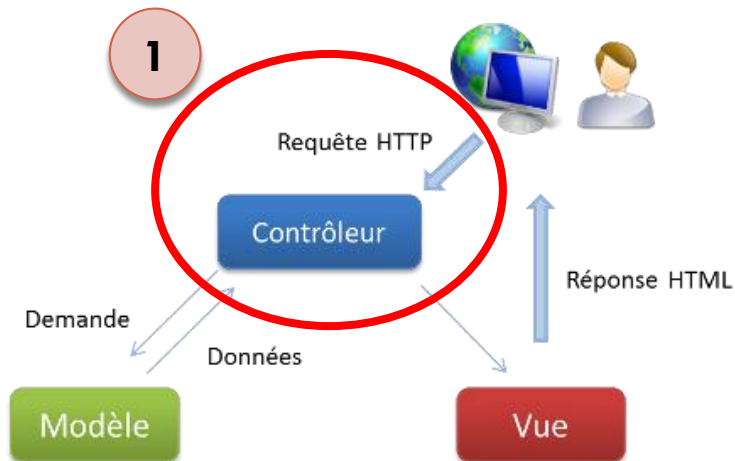
9

- MVC = **Modèle-Vue-Contrôleur**
- C'est un patron de conception (*design pattern*)
- Permet de séparer les problématiques



1 - Le Contrôleur

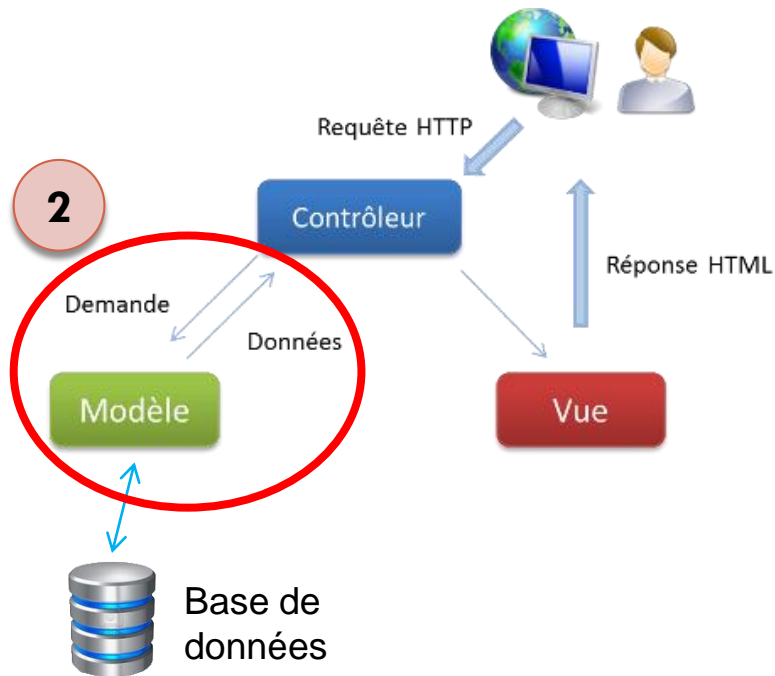
10



- Appelé en premier
- Appelé par l'URL
- Récupère les données envoyées par l'utilisateur
- Fait des vérifications
- Appelle tous les composants nécessaires
- Fait appel au modèle pour récupérer les données
- Renvoie la vue

2- Le Modèle

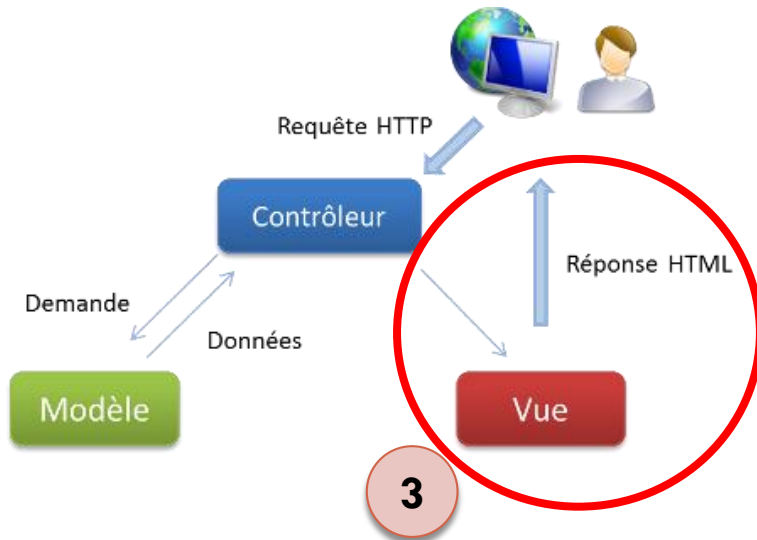
11



- appelé par le contrôleur
- **Accède aux données de la base**
- peut comporter plusieurs méthodes (CRUD)
- Renvoie les données au contrôleur

3- La Vue

12



- appelée par le contrôleur
- **Contient le code HTML/CSS**
- Intègre les données reçues du contrôleur pour les afficher
- Est renvoyée vers le navigateur web

13

PHP objet

Depuis PHP 5 ...

Création d'une classe en PHP

14

Voiture
-km
+ __construct() + LireKm()

Création de la classe

```
<?php
class Voiture {

    private $km;

    public function __construct() {
        $this->km = 0;
    }

    public function LireKm() {
        return $this->km;
    }

}
```

Déclaration d'attribut en public, private ou protected

Constructeur (il existe aussi __destruct())

Utiliser l'attribut avec \$this (et sans \$ devant le nom de l'attribut)

Instanciation

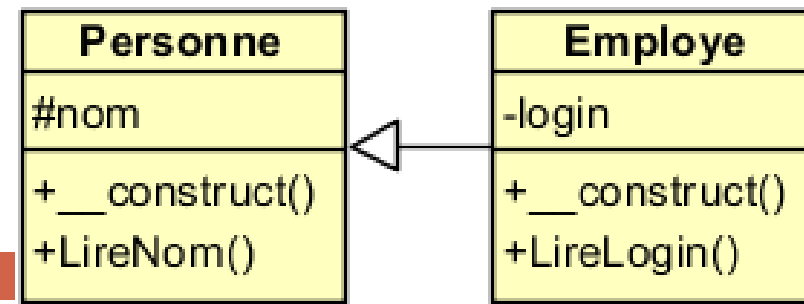
```
$objVoiture = new Voiture();
$val = $objVoiture->LireKm();
```

Création de l'objet avec new

Symbole -> pour accéder aux membres de l'objet

L'héritage en PHP

15



Création de la classe mère

```
class Personne {
    protected $nom = "";
    public function __construct() {
        $this->nom = "anonyme";
    }
    public function LireNom() {
        return $this->nom;
    }
}
```

Création de la classe fille

```
class Employe extends Personne {
    private $login = "abcd";
    public function __construct() {
        parent::__construct();
    }
    public function LireLogin() {
        return $this->login;
    }
}
```

Appel du constructeur parent

Les constructeurs parents ne sont pas appelés implicitement si la classe enfant définit un constructeur.

Instanciation

```
$objEmploye = new Employe();
$nom = $objEmploye->LireNom(); //"anonyme"
$login = $objEmploye->LireLogin(); //"abcd"
```

16

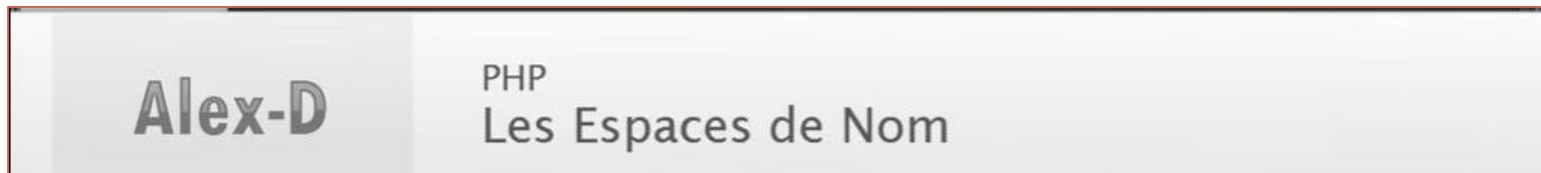
Les namespaces PHP

Les namespaces PHP

17

- Suivez le tuto en vidéo (11'42«)

- ▣ <https://www.grafikart.fr/tutoriels/namespaces-279>



- Autres docs :

- ▣ <https://www.php.net/manual/fr/language.namespaces.rationale.php>

- ▣ <https://www.grafikart.fr/tutoriels/namespace-php-1143>

Les tableaux PHP (*array*)

http://www.w3schools.com/php/php_arrays.asp









Les tableaux indexés : [index]

19

- Un array stocke plusieurs valeurs dans une variable unique

```
<?php
$cars = array("Aston Martin", "Bentley", "Audi");
echo "J'aime " . $cars[0] . ", " . $cars[1] . " et "
      . $cars[2] . ".";
?>
```

- Chaque case du tableau est identifiée par un numéro qu'on appelle **index** ou **indice**

 \$cars	array[3]		
 [0]	string		"Aston Martin"
 [1]	string		"Bentley"
 [2]	string		"Audi"

Parcourir un tableau indexé (for)

20

- Le nombre d'éléments dans un tableau est donné par la fonction **count()**

```
<?php
$cars = array("Aston Martin", "Bentley", "Audi");
$taille = count($cars);

for($x = 0; $x < $taille; $x++) {
    echo $cars[$x];
    echo "<br/>";
}
?>
```

Parcourir un tableau indexé (foreach)

21

- **foreach** fournit une façon simple de parcourir des tableaux

```
<?php
$cars = array("Aston Martin", "Bentley", "Audi");

foreach ($cars as $car) {
    echo $car;
    echo "<br/>";
}
?>
```

Aston Martin
Bentley
Audi









Les tableaux associatifs : ['clé']

22

- Au lieu de numéroter les cases, on va les étiqueter en donnant à chacune un nom différent (clés nommées)

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

- Chaque case du tableau est identifiée par un nom qu'on appelle **clé**

 \$age	array[3]		
 [Peter]	string		"35"
 [Ben]	string		"37"
 [Joe]	string		"43"

Créer un tableau associatif

23

- Il existe 2 façons de créer un tableau associatif

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

OU

```
$age['Peter'] = "35";  
$age['Ben'] = "37";  
$age['Joe'] = "43";
```

Parcourir un tableau associatif (foreach)

24

- **foreach** fournit une façon simple de parcourir des tableaux associatifs

```
foreach ($age as $x => $x_value) {  
    echo "Key=" . $x . ", Value=" . $x_value;  
    echo "<br>";  
}
```

```
Key=Peter, Value=35  
Key=Ben, Value=37  
Key=Joe, Value=43
```


Insérer des éléments dans un tableau

25

```
$tab = array("orange", "banane");  
  
//ajouter un élément à la fin du tableau  
array_push($tab, "framboise");  
$tab[] = "fraise"; //préférez array_push()  
  
//ajouter un élément au début du tableau  
array_unshift($tab, "pomme");  
  
//ajouter un élément après le 2ème élément  
array_splice($tab, 2, 0, "poire");
```

Supprimer des éléments d'un tableau

26

```
$input = array("a", "b", "c", "d", "e");  
  
//supprimer un élément de la fin du tableau  
array_pop($input);  
  
//supprimer un élément du début du tableau  
array_shift($input);  
  
//supprimer le 2è élément  
unset($input[1]); //efface l'élément du tableau  
$input = array_values($input); // Ré-indexation  
  
//Supprimer un tableau  
unset($input);
```

