Validation

- La validation est importante pour protéger les données de formulaire contre les hackers

- La fonction **htmlspecialchars()** convertit les **caractères spéciaux** en **entités** HTML pour sécuriser les valeurs

=> Evite l'exécution de codes malveillants tapés dans les champs de saisie du formulaire

Validation avec PHP

- Passer toutes les variables via la fonction htmlspecialchars()
- Si un hacker insère le code JavaScript suivant: <script>location.href('http://www.hacked.com')</script>
- Ce script ne sera pas exécuté car il sera sauvegardé comme suit: <script>location.href('http://www.hacked.com')</script>

- La fonction **trim()** supprime les caractères **inutiles** (espaces, tabulations, saut de ligne, retour chariot, etc)

Syntaxe:

trim(chaine à **traiter**, chaine à **supprimer**): string

Ex. Trim(" HTML, PHP ") // supprime les espaces

Trim("HTML, et PHP", "et")// supprime la chaine de caractères 'et'

=> Sortie : " HTML, PHP "

- La fonction **stripslashes()** enlève les antislashs (\)
- On peut créer une fonction qui fait tout:

```
function validation($data) {
        $data = trim($data);
        $data = stripslashes($data);
        $data = htmlspecialchars($data);
      return $data;
      }
```

Exemple

- La variable **$_SERVER["PHP_SELF"]** est une variable superglobale qui renvoie le **nom** du script en cours d'exécution

- **Code vulnérable** aux attaques

```
<form method="POST" action="<?php echo
$_SERVER["PHP_SELF"];?>">
```

- **Code sûr**

```
<form method="POST" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

=> Cela empêche l'injection de code malveilant

Programmation orientée objet

- Une **classe** est un **moule**, **prototype** ou un **modèle** pour les objets

- Une classe possède des **propriétés** (*attributs*) et des **méthodes** (*comportements*)

- Une classe définit **l'état** et les **comportements** d'un objet

- Un **objet** est une **instance** d'une classe

- Les objets héritent de toutes les propriétés et de tous les comportements de la classe

- Chaque objet aura des valeurs différentes pour les propriétés

- Dans une classe, les **variables** sont appelées propriétés et les **fonctions** sont appelées méthodes

Exemples

- **Classe**: Voiture

  Objets: Nissan, Toyota, Renault, …etc

- **Classe**: Animal

  Objets: Chat, Chien, Singe, …etc

- **Classe**: Fruit

  Objets : Pomme, Banane, Orange, ..etc

- En PHP, une classe est définie par le mot clé **class** suivi par le **nom** de la classe, une accolade ouvrante, les propriétés et méthodes puis une accolade fermante)
- **Ex**.

```php
<?php
    class Personne {
    // Proprietés
    public $nom;
    public $adresse;
    // Méthodes
    function afficher_nom() {
      return $this->nom;
     } }
?>
```

- $this fait référence à l'objet courant, et n'est disponible qu'à l'intérieur des méthodes

- Les objets d'une classe sont créés à l'aide du mot clé **new**

```
<?php
    $etudiant = new Personne();

    echo $etudiant->afficher_nom();

    ?>
```

- La valeur d'une propriété peut changer soit:
  - À l'intérieur d'une classe en utilisant *$this* dans une méthode

```
<?php
    class Personne {
    public $nom;
   function set_name($name) {
      $this->nom = $name;  }}
  $etudiant = new Personne();
  $etudiant->set_name("Jean");
  echo $etudiant->nom;
?>
```

  - Directement à l'extérieur d'une classe

```
<?php
    class Personne {
    public $nom;}
   $etudiant = new Personne();
  $etudiant->nom = "Jean";
  echo $etudiant->nom;
 ?>
```

Gestion de base de données MySQL

MySQL est un SGBDR

- Les étapes de gestion d'une base de données MySQL avec PHP sont:

1. **Connexion** au serveur MySQL
2. **Envoi** des **requêtes** SQL au serveur
3. **Récupération** et traitement des résultats des requêtes
4. **Fermeture** de connexion

- La gestion de BDD peut se faire avec:

  - MySQLi: spécifique à MySQL
    - Accès procédural
    - Accès objet

  - PDO (PHP Data Objects): supporte **différentes** base de données (MySQL, SQLite, PostgreSQL, oracle, etc)

❑ MySQLi: Accès procédural

- Connexion au serveur MySQL: mysqli_connect

  $idcon= mysqli_connect($host, $user, $pass, [$db]);

- Envoi des requêtes SQL au serveur:

  $res=mysqli_query($requête, [idcon])

- Récupération et traitement des résultats des requêtes: mysqli_fetch_array($res), mysqli_fetch_object

  mysqli_fetch_assoc($res) et mysqli_fetch_row($res)

- Fermeture de connexion: mysqli_close($idcon)

❑ MySQLi: Accès objet

- Connexion au serveur MySQL: $mysqli=new mysqli()

  $idcon =new mysqli($host, $user, $pass, [$db]);
- Envoi de requêtes SQL au serveur:

  $res=$idcon->query($requete)
- Récupération et traitement des résultats des requêtes: $res-> fetch_array(), $res-> fetch_assoc(), $res-> fetch_row(), $res–>fetch_object()
- Fermeture de connexion: $idcon->close()

❑ Accès PDO à MySQL

- Connexion au serveur MySQL:

```
$idcon= new PDO("mysql:host=$host;dbname=$base ,$user, $pass");
```

- Envoi de requêtes SQL au serveur:

```
$res= $idcon->query($requête), $res= $idcon->exec($requête)
```

- Récupération et traitement des résultats des requêtes: $res->fetch(), $res->fetchAll(), $res->fetchObject()
- Fermeture de connexion: $idcon=NULL