# Fiche procédure PHP

## Table des matières



## Installation de PHP

## Sous Windows

- Télécharge le package PHP sur php.net/downloads.php.
- Décompresse l'archive et ajoute le dossier PHP à la variable d'environnement `PATH`.
- Installe un serveur local comme XAMPP ou WampServer pour faciliter le développement.

## Sous macOS

- Utilise Homebrew :
- `bash`

```
brew install php
```

- 
- Ou installe MAMP pour un environnement tout-en-un.

## Sous Linux

- Utilise le gestionnaire de paquets :
- `bash`

```
sudo apt update
sudo apt install php
```

- 

Pour vérifier l'installation, tape dans le terminal :
`bash`

```
php -v
```

## Création d’un projet PHP

1. Crée un dossier pour ton projet, par exemple : `mon-projet-php`.
2. Crée un fichier principal, par exemple : `index.php`.
3. Place ce dossier dans le répertoire web de ton serveur local :
   - Sous XAMPP : `C:\xampp\htdocs\mon-projet-php`
   - Sous WAMP : `C:\wamp64\www\mon-projet-php`
   - Sous MAMP : `/Applications/MAMP/htdocs/mon-projet-php`
4. Lance le serveur web (Apache) via XAMPP/WAMP/MAMP ou utilise le serveur PHP intégré :
5. `bash`

```
php -S localhost:8000
```

6. 
7. Accède à ton projet via : http://localhost/mon-projet-php (ou http://localhost:8000 si tu utilises le serveur intégré).

## Utilisation de PHP

- Un fichier PHP contient du code HTML et du code PHP entouré de balises `<?php ... ?>`.
- PHP s’exécute côté serveur et génère du HTML envoyé au navigateur.
- Tu peux inclure des variables, des boucles, des conditions, etc.

Exemple de base :
`php`

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Mon premier script PHP</title>
</head>
<body>
    <h1>
        <?php
            echo "Bienvenue sur mon site PHP !";
        ?>
    </h1>
```

```
    <p>Aujourd'hui, nous sommes le <?php echo date('d/m/Y'); ?>.</p>
</body>
</html>
```

## Bonnes pratiques

- Sépare le code PHP et HTML pour plus de clarté.
- Utilise des noms de fichiers explicites et une structure de dossiers logique.
- Commente ton code pour faciliter la maintenance.
- Valide les entrées utilisateur pour éviter les failles de sécurité.
- Utilise les fonctions natives de PHP pour manipuler les chaînes, tableaux, fichiers, etc.
- Teste ton code régulièrement sur différents environnements.
- Gère les erreurs avec `try/catch` (PHP 7+) ou `error_reporting`.

## Exemple de code PHP

php

```php
<?php
// Déclaration d'une variable
$nom = "Luca";

// Fonction pour afficher un message
function saluer($prenom) {
    return "Bonjour, $prenom !";
}

// Utilisation de la fonction
$message = saluer($nom);
?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Exemple PHP</title>
</head>
<body>
```

```
    <h2><?php echo $message; ?></h2>
</body>
</html>
```