

# IFT1147





# CONTACTER

- FORUM SUR STUDIUM
- COURRIEL: [JOANIE.BIRTZ@UMONTREAL.CA](mailto:JOANIE.BIRTZ@UMONTREAL.CA)



# COMPTE DIRO

- ACTIVATION  
COMPTE
- CHANGER MOT  
DE PASSE

INSTRUCTION A SUIVRE :







[https://support.iro.umontreal.ca/doku.php?id=modifier\\_mdp](https://support.iro.umontreal.ca/doku.php?id=modifier_mdp)

VIDEO EXPLICATIVE

<https://support.iro.umontreal.ca/lib/exe/fetch.php?media=guide:activationcomptediro.mp4>



# SERVEUR LOCAL

- XAMPP    [Lien ici](#)
- MAMP   [Lien ici](#)
- WAMP  [Lien ici](#)

# XAMPP

**X** = Multiplateforme (Windows, Mac, Linux)

**A** = Apache (serveur web)

**M** = MySQL (base de données)

**P** = PHP

**P** = Perl

# XAMPP

XAMPP est une solution tout-en-un qui inclut Apache (serveur web), MySQL (serveur de base de données), PHP, et Perl. Pour commencer avec PHP sur XAMPP :

**1. Téléchargement de XAMPP :**

- Téléchargez XAMPP depuis <https://www.apachefriends.org>.
- Installez-le sur votre système.

**2. Lancer XAMPP :**

- Ouvrez le **XAMPP Control Panel**.
- Activez **Apache** et **MySQL**.

**3. Configurer le dossier de travail :**

- Par défaut, placez vos fichiers PHP dans le dossier C:\xampp\htdocs\.
- Créez un fichier PHP comme index.php dans ce dossier pour tester vos scripts.

**4. Accéder à vos pages PHP :**

- Dans votre navigateur, tapez `http://localhost/nom_du_fichier.php` pour exécuter votre script.



# VISUAL STUDIO CODE (VSC)



Visual Studio Code est un  
éditeur de code extensible



Download: <https://code.visualstudio.com/download>

**Extension util :**

PHP Intelephense, HTML CSS Support, Prettier - Code  
formatter, Auto Rename Tag

# GIT



- **logiciel de gestion de versions** (version control system).
- Il s'installe **en local sur ton ordinateur** (Windows, Mac, Linux).

Exemples de commandes Git :

```
git init          # créer un dépôt local
git add .         # ajouter des fichiers
git commit -m "message" # enregistrer un commit
git branch        # gérer les branches
git merge         # fusionner des branches
```

<https://git-scm.com/downloads>

# GitHub



- C'est un **service en ligne (hébergé dans le cloud)** basé sur Git.
- Il sert à **héberger tes dépôts Git** pour collaborer avec d'autres personnes.
- Il offre des outils supplémentaires : interface web, gestion de projets, suivi d'issues, pull requests, actions CI/CD, etc.
- Tu utilises GitHub pour partager ton code, contribuer à des projets open source ou travailler en équipe.

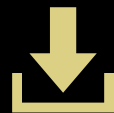
<https://github.com/>  
<https://desktop.github.com/download/>



# JQUERY

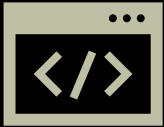


jQuery est une bibliothèque JavaScript  
contracter

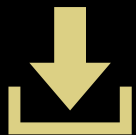


Download :

<https://jquery.com/download/>



Bootstrap est un framework open-source et gratuit d'outils (HTML, CSS, JavaScript) pour la création de sites web et d'applications adaptatifs et responsives

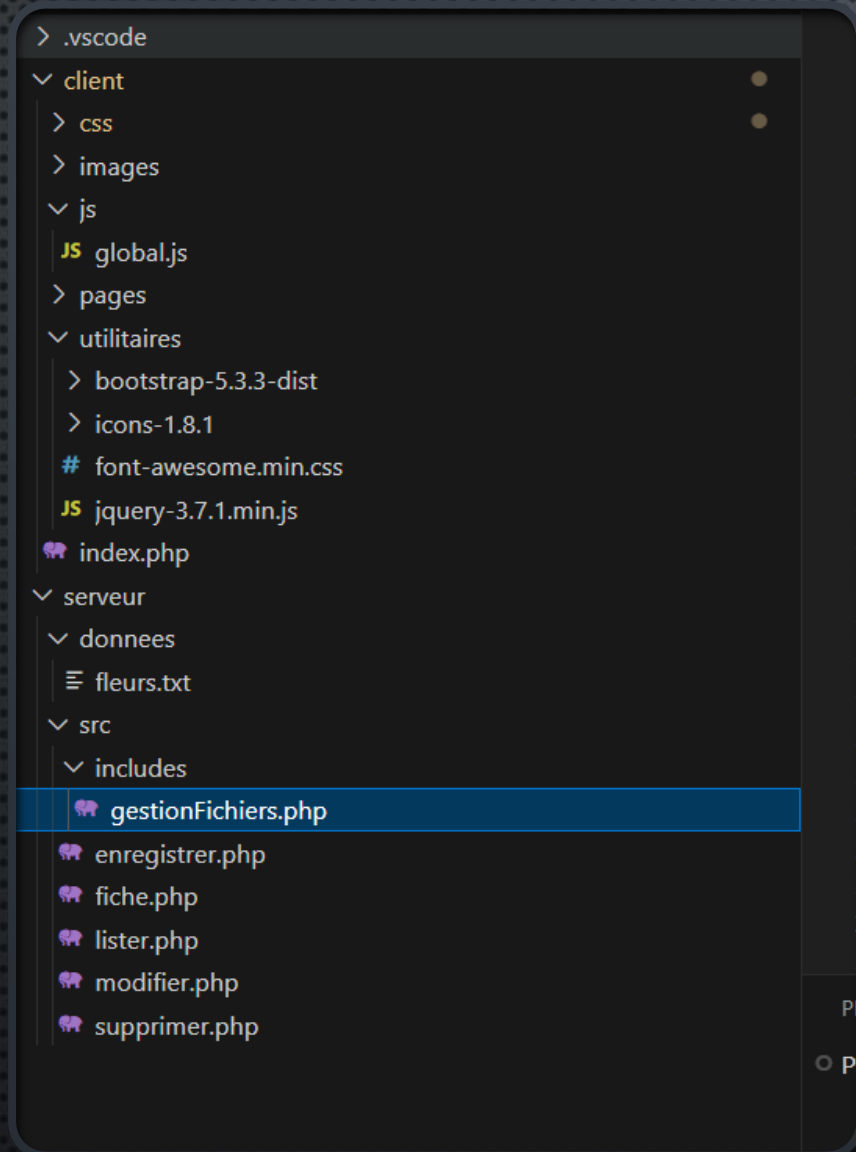


Download :  
<https://getbootstrap.com/docs/5.3/getting-started/download/>

# BOOTSTRAP



# EXEMPLE STRUCTURE DE PROJET



## EXERCICE 1 : VARIABLES ET AFFICHAGE

Écris un script profil.php qui :

- déclare trois variables : \$nom, \$age, \$ville
- affiche : « *Je m'appelle [nom], j'ai [age] ans et j'habite à [ville].* »



```
1  <?php
2  $nom = "JOE";
3  $age = 24;
4  $ville = "Montreal";
5
6  echo "Je m'appelle $nom, j'ai $age ans et j'habite à $ville.";
7  ?>
8
```

## EXERCICE 2 : STRUCTURES CONDITIONNELLES

Crée un fichier majeur.php qui :  
demande un âge (par exemple via  
une variable ou un formulaire).  
affiche :

- « *Vous êtes mineur.* » si  $<$   
18
- « *Vous êtes majeur.* » si  $\geq$   
18



```
1  <?php
2  $age = 17;
3
4  if ($age < 18) {
5      echo "Vous êtes mineur.";
6  } else {
7      echo "Vous êtes majeur.";
8  }
9  ?>
```

## EXERCICE 3 : BOUCLES

Affiche dans boucles.php :

- les nombres de 1 à 20 avec une boucle for
- les couleurs d'un tableau (["rouge", "vert", "bleu"]) avec foreach



```
<?php
// Boucle for
for ($i = 1; $i <= 20; $i++) {
    echo "Nombre : $i <br>";
}

// Boucle foreach
$couleurs = ["rouge", "vert", "bleu"];
foreach ($couleurs as $c) {
    echo "Couleur : $c <br>";
}
?>
```

## EXERCICE 4 : FONCTIONS

Écris une fonction  
`carre($nombre)` qui retourne le  
carré du nombre passé en  
paramètre.

Test : afficher `carre(5)` doit  
donner 25



```
<?php
function carre($nombre) {
    return $nombre * $nombre;
}

echo carre(5); // Affiche 25
?>
```

## EXERCICE 5 : FORMULAIRE ET TRAITEMENT

Crée deux fichiers :

- `formulaire.html` avec un champ texte pour entrer un prénom.
- `traitement.php` qui affiche « *Bonjour [prénom]* » en utilisant `$_POST`.  
Protège l'entrée avec `htmlspecialchars()` pour éviter les injections



```
<form action="traitement.php" method="POST">
    Prénom : <input type="text" name="prenom">
    <input type="submit" value="Envoyer">
</form>
```

php

 Copy code

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $prenom = htmlspecialchars($_POST['prenom']);
    echo "Bonjour $prenom !";
}
?>
```

## EXERCICE 6 : HTTP – THÉORIE

Explique en une phrase la différence entre :

une requête **GET**

une requête **POST**

Donne un exemple de cas où tu utiliserais chacune.



**GET** : envoie les données dans l'URL (?cle=valeur).

→ Exemple : rechercher un produit sur un site e-commerce.

**POST** : envoie les données dans le corps de la requête (plus sécurisé, pas visible dans l'URL).

→ Exemple : formulaire de connexion avec mot de passe

## EXERCICES 7: ANALYSE DE REQUÊTE HTTP

*GET /produits.html HTTP/1.1*

*Host: www.maboutique.com*

*User-Agent: Mozilla/5.0*

- Quelle est la méthode ?*
- Quelle ressource est demandée ?*
- Quel en-tête indique le domaine ?*



MÉTHODE : GET

RESSOURCE : /PRODUITS.HTML

DOMAINE : WWW.MABOUTIQUE.COM

## EXERCICE 8 : CODES DE STATUT

Associe le code HTTP avec sa signification :

200

404

500

- a) Page non trouvée
- b) Requête réussie
- c) Erreur interne du serveur



200 → B) REQUÊTE RÉUSSIE

404 → A) PAGE NON TROUVÉE

500 → C) ERREUR INTERNE DU  
SERVEUR

## EXERCICE 9 : DNS

Explique le rôle du DNS dans la navigation web.

⚠ Tu peux faire un schéma simple (nom de domaine → serveur DNS → adresse IP).



LE DNS TRADUIT UN NOM DE DOMAINE (EX.  
WWW.GOOGLE.COM ) EN ADRESSE IP  
(EX. 142.250.72.206) POUR QUE TON NAVIGATEUR  
SACHE QUEL SERVEUR CONTACTER

www.exemple.com → Serveur DNS → 192.168.1.10

# EXERCICE 10 : FORMULAIRE ET HTTP

```
1 <form action="enregistrer.php" method="POST" enctype="multipart/form-data">
2   <input type="text" name="nom">
3   <input type="file" name="photo">
4   <input type="submit">
5 </form>
6
```

Pourquoi la méthode est POST et non GET ?

Pourquoi utilise-t-on multipart/form-data ?



**GET** → pour afficher, rechercher ou consulter des données (lecture).

**POST** → pour tout le reste (ajouter, modifier, supprimer), car HTML ne permet pas d'utiliser PUT/DELETE directement dans <form>.

On utilise multipart/form-data car il y a un fichier à transférer : le navigateur doit envoyer le texte et le fichier en plusieurs parties

html

```
<form method="POST" action="traitement.php">
  <input type="text" name="nom" value="Alice">
  <input type="text" name="age" value="25">
  <input type="submit">
</form>
```

CAS NORMAL (SANS  
FICHIER)  
(APPLICATION/X-WWW-  
FORM-URLENCODED)

```
<form method="POST" action="upload.php" enctype="multipart/form-data">
  Nom : <input type="text" name="nom">
  Photo : <input type="file" name="photo">
  <input type="submit">
</form>
```

LE CAS AVEC UN  
FICHIER  
(MULTIPART/FORM-  
DATA)