

# TP02 : PRÉSENTATION DE L'ENVIRONNEMENT DE TRAVAIL – VS CODE

Nous verrons dans ce TP l'utilisation d'un logiciel qui nous servira pour le développement d'application, à savoir VSCode.

VSCode est un IDE : **integrated development environment**. En français : **Environnement de Développement Intégré (EDI)**.

Un IDE est bien plus qu'un éditeur de texte. Un IDE permet de développer de façon beaucoup plus efficace en intégrant de nombreuses options :

- Coloration syntaxique
- Refactor du code
- Auto-complétions
- Lien avec des outils de version (GitHub)
- Permet de coder dans de nombreux langages : Java, JS, PHP, HTML, CSS et bien d'autres
- Le débogage, etc ...

## CRÉATION D'UN PROJET

Pour développer une application HTML-CSS-JS, c'est extrêmement simple : il suffit d'ouvrir un dossier. Nous avons déjà fait cette opération dans le TP1. Refaisons le pour ce TP.

A faire : créer un dossier TP2-Web. Ouvrir ce dossier avec VSCode. Ajouter une page « index.html »

## AJOUTER L'EXTENSION « LIVE PREVIEW »

Dans le TP précédent nous avons utilisé le serveur WEB de python pour des raisons essentiellement pédagogique. C'était afin de montrer le rôle d'un serveur WEB, et de comprendre le modèle CLIENT-SERVEUR basé sur des requêtes – réponses.

Ce dernier présentait néanmoins quelques inconvénients que nous allons corriger avec cette extension que nous allons installer.

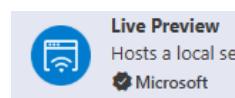
En effet « Live Preview » permet :

- D'afficher une page web **en temps réel**
- Avec **rechargement automatique** à chaque modification
- Et comme « Simple Python serveur » c'est un **serveur HTTP** (donc pas besoin d'installer Apache, nginx ou autres)

Mais contrairement à « Simple Python serveur », « Live Preview » ne permet pas :

- De visualiser les requêtes et réponses http
- Une connexion distante : juste du localhost (127.0.0.1)

A faire : dans « Extensions » chercher l'extension « certifié Microsoft » nommé « Live Preview ». Redémarrer VSCode si nécessaire pour qu'elle soit prise en compte.



A faire : Maintenant que l'extension est installé plusieurs possibilités sont offertes pour visualiser la page WEB :

- Dans la partie « Explorer » (bandeau latéral gauche), clic droit sur le fichier « index.html », puis « Show Preview ».
- Dans l'éditeur de code cliquez sur le bouton « Show Preview » en haut à droite
- Dernière option, clic droit dans l'éditeur de code puis « Show Preview »



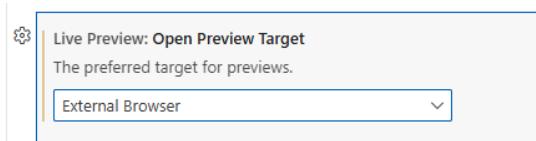
A faire : L'opération précédente permet de prévisualiser le code html de votre page Web dans VSCode.

Mais il peut être beaucoup plus pratique de l'ouvrir dans le navigateur. Dans la fenêtre de prévisualisation il est possible d'ouvrir la page dans un navigateur. Cliquez sur le menu « hamburger » puis « Open in Browser »



A faire : l'intérêt de cette extension réside dans le rendu visuel de la page en temps réel. Modifier votre page dans VSCode tout en ayant le navigateur ouvert. Vous pouvez visualiser les changements en direct.

A faire : chaque extensions VSCode peut être paramétré selon ses besoins. Ouvrir la page de l'extensions « Live Preview », puis cliquez sur l'icône pour modifier les paramètres (« Manage »). Dans la page « Settings » vous pouvez modifier le navigateur utilisé pour prévisualiser les pages, modifier le port d'écoute (3000 par défaut), et nous allons ici modifier le mode de prévisualisation. En choisissant « External Brower » la prévisualisation de notre page se fera directement dans le navigateur de votre choix.



## UTILISER GIT

Git permet de gérer les versions d'un projet et d'en conserver l'historique. Il facilite le suivi des modifications et le travail propre sur un dossier. Dans ce TP, il sera utilisé pour publier la production finale sur un dépôt GitHub.

<https://fr.wikipedia.org/wiki/Git>

Il existe un mode graphique, mais pour comprendre le fonctionnement il faut passer par la ligne de commande où l'on comprend beaucoup mieux les opérations réalisées.

A faire : avant toute manipulation GIT, nous allons ajouter un fichier « test.html »

### CREER LE DEPOT LOCAL.

Le dépôt Git sert à enregistrer et conserver l'historique des versions d'un projet, afin de suivre et retrouver les modifications apportées aux fichiers.

A faire : dans le terminal entrer la commande « **git init** » permettant créer un nouveau dépôt (repository). Dans l'explorateur de fichier vous avez maintenant un dossier caché « **.git** ». Dans le terminal vous pouvez taper « **dir -Force** » pour voir les fichiers cachés.

⇒ **Le dépôt est créé, mais aucun fichier n'a encore été ajouté.**

A faire : noter le symbole « **U** » devant les fichiers HTML pour « **Untracked** ». Cela signifie que ces fichiers ne sont pas encore gérés (ou suivis) par Git.

### AJOUTER LES FICHIERS AU DEPOT.

Nous allons ajouter nos pages html dans le dépôt. La commande « **add** » permet de préparer le fichier pour l'enregistrer dans le prochain commit. Il **prépare le fichier** dans ce qu'on appelle la **staging area ou index**.

A faire : dans le terminal taper la commande « **git add index.html** », mais ne pas le faire pour le fichier « **test.html** »

A faire : noter le changement dans le symbole utilisé : « **A** » pour « **index Added** »

### REALISER UN COMMIT.

Le terme « **commit** » vient du monde SQL, utilisé pour valider une transaction.

Cette commande **enregistre définitivement** les fichiers qui ont été ajoutés avec « **git add** ».

Chaque commit est **une version datée du projet**, avec un **message** pour expliquer ce qui a été fait sur les fichiers. C'est comme prendre **une photo du projet à un instant donné**.

A faire : taper la commande « **git commit** ». Un message est affiché vous demandant de taper 2 commandes au préalable :

« **git config --global user.email** » et « **git config --global user.name** » permettant de garder une trace de l'auteur. N'oublions pas que « **Git** » est conçu pour le **travail collaboratif** (quand plusieurs personnes travaillent sur le même projet). Chaque commit conserve :

- **Qui l'a fait** (user.name / user.email)
- **Quand il a été fait** (date et heure)

A faire : relance la commande « git commit », un éditeur de texte se lance pour que vous puissiez indiquer le message associé à ce commit. Indiquer le message « First Commit ». Valider, vous remarquez que le « A » associé au fichier « index.html » a disparu.

## REVENIR SUR UNE VERSION ANTERIEUR.

A faire : dans le fichier « index.html » ajouter du code html. Vous remarqué que le fichier passe en statut « M » pour « Modified ».

A faire : ajouter un nouveau fichier « test2.html » (« test.html » est toujours en « U »), et l'ajouter au dépôt. Pensez à ajouter le fichier « index.html » au dépôt également étant donné qu'il a été modifié, en effet, **Git ne commit que ce qui est dans l'index**.

A faire : refaire un commit cette fois ci avec l'option -m pour indiquer le message. Taper la commande : « git commit -m "Ajout de code dans index.html, et création de la page test2.html" ». Noter le numéro associer à ce commit → [master 37c15c5]

A faire : lancer la commande « git reset --hard 37c15c5 » (adapter le numéro à votre contexte), noter que cette commande est revenue à une situation antérieure. Le commit est annulé **et les fichiers retournent à l'état précédent**.

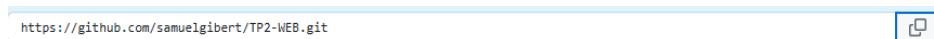
A faire : relever le numéro du tout premier commit (avec le message « First Commit ») et lancer la commande « git reset --hard numéro » (adapter le numéro à votre contexte). Vous êtes revenu à l'état initial.

## GITHUB

Nous allons maintenant « pousser » notre dépôt local vers un dépôt distant qu'est GitHub.

A faire : si ce n'est pas déjà fait créer un compte sur GitHub.

A faire : créer un nouveau dépôt sur GitHub nommé TP2-WEB. Laisser les options par défaut pour README, .gitignore et licence. A la fin de l'opération copié le lien de votre dépôt GitHub.



A faire : nous allons maintenant lier le dépôt local au dépôt GitHub. Dans le **terminal VS Code**, taper la commande suivante : « git remote add origin https://github.com/ton-compte/TP2-WEB.git », origin étant le nom standard du dépôt distant.

Vérifier que le lien à fonctionner avec la commande suivante : « git remote -v »

A faire : Nous allons pousser notre dépôt local vers le dépôt GitHub. Lancer la commande suivante : « git push -u origin master ». L'option -u mémorise la destination pour les prochains push. Git va demander une **authentification GitHub**, entrer vos identifiants et autoriser le dépôt. Vérifier sur le site que votre dépôt est effectif.

## GITHUB PAGES

Nous allons maintenant publier notre site de notre dépôt GitHub.

A faire : se placer dans le dépôt TP2-WEB, puis Settings, et enfin pages. Choisir la branche « master » puis sauvegarder. Cela prend quelques minutes avant d'avoir votre lien vers votre site web. Vous n'avez plus qu'à visiter votre site.

### GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at <https://samuelgibert.github.io/TP2-WEB/>  
Last deployed by samuelgibert 2 minutes ago

[Visit site](#)

[Unpublish site](#)

Attention pour que les pages soit accessible le dépôt doit être public.