



# Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec l'IA Gemini 3

Serge Tahé, janvier 2026

Ce site a été créé avec le convertisseur [Word ou ODT - > HTML] créé par l'IA Gemini 3 en janvier 2026 et décrit par ce document.

# 1. Introduction

On se propose dans cet article de mettre à disposition du lecteur un convertisseur Python de documents Word ou ODT vers un site statique HTML. Ce convertisseur a été construit par l'IA [Gemini 3](#). Il a fallu 288 itérations à Gemini pour produire le convertisseur de ce document. Cela a représenté un débogage de plusieurs semaines entre l'IA et moi. Il a fallu d'abord plusieurs dizaines d'itérations pour obtenir une première version à peu près correcte. Puis régulièrement j'ajoutais un nouveau problème posé par de nouveaux documents ODT. Et là, Gemini a souvent régressé. C'est-à-dire que ce qui marchait à l'étape N ne marchait plus aux étapes suivantes. J'ai alors procédé ainsi : dès que Gemini produisait un convertisseur qui résolvait un de mes problèmes, j'en faisais une version de référence et je le disais à Gemini. Puis lorsque je le voyais régresser trop longtemps je lui demandais de revenir à la dernière version de référence connue. C'est comme ça que peu à peu, on a construit ensemble ce convertisseur : moi en lui exprimant ce que je voulais, essentiellement en lui signalant les anomalies que je trouvais dans le site HTML produit, lui en produisant le code demandé.

J'ai utilisé [Gemini 3](#) avec une licence pro à 22 euros /mois. Si vous partez d'un document Word, ouvrez-le avec [LibreOffice](#) et sauvegardez-le au format ODT. N'utilisez pas l'option de Word qui permet de sauvegarder un document Word au format ODT. Pourquoi ? Parce que Word et LibreOffice ne génère pas le même XML pour un document ODT et que le convertisseur de Gemini a été entraîné sur des documents générés par LibreOffice.

Gemini va générer deux scripts Python :

- [convert] pour convertir le document ODT de [LibreOffice](#) en site [MkDocs](#) ;
- [build] pour convertir le site MkDocs en site HTML statique ;

Je n'ai jamais regardé le code généré. Je voulais les considérer comme des boîtes noires. Il n'y a pas besoin d'être un développeur Python pour suivre ce tutoriel ni même un développeur tout court.

Je voudrais faire un article court. Les interactions entre une IA et un utilisateur sont présentés dans l'article [\[Générer un script Python avec des outils d'IA\]](#) que j'appellerais désormais [\[ref1\]](#). Les interactions avec Gemini ne seront présentées qu'à la marge. De toute façon il était impossible de présenter les 288 itérations.

Le PDF du document dont le convertisseur Gemini 3 a produit le site HTML est [ICI](#).

## 2. Les exemples de ce document

Je vais présenter maintenant des exemples des particularités de mes documents ODT que le convertisseur Gemini gère correctement. C'est ce propre document que nous proposerons à la conversion HTML au script de Gemini. Nous verrons ce qu'il en fait.

### 2.1. Les listes

Le convertisseur Gemini sait gérer les listes à puces et les listes numérotées :

#### Listes à puces

- Élément 1 ;
- Élément 2 ;
- Élément 3 ;
  - Élément 3.1 ;
  - Élément 3.2 ;
- Élément 4 ;

#### Listes numérotées

1. Élément 1 ;
2. Élément 2 ;
3. Élément 3 ;

## 2.2. Les blocs de code

Mes cours contiennent beaucoup de blocs de code. Le convertisseur Gemini sait à peu près les gérer à l'aide de mots clés du langage mis dans un fichier de configuration. Lorsqu'il reconnaît un langage il insère un marqueur (fence) pour MkDocs afin qu'il adapte la coloration syntaxique du code au langage utilisé dans le bloc de code.

### Java

```
1. package istia.st.spring.core;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import org.springframework.context.ApplicationContext;
7. import org.springframework.context.support.ClassPathXmlApplicationContext;
8.
9. public class Demo01 {
10.
11.     @SuppressWarnings({ "unchecked", "resource" })
12.     public static void main(String[] args) {
13.         // récupération du contexte Spring
14.         ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15.         // on récupère les beans
16.         Personne p01 = ctx.getBean("personne_01", Personne.class);
17.         Personne p02 = ctx.getBean("personne_02", Personne.class);
18.         List<Personne> club = ctx.getBean("club", new
ArrayList<Personne>().getClass());
19.         Appartement appart01 = ctx.getBean(Appartement.class);
20.         // on les affiche
21.         System.out.println("personnes-----");
22.         System.out.println(p01);
23.         System.out.println(p02);
24.         System.out.println("club-----");
25.         for (Personne p : club) {
26.             System.out.println(p);
27.         }
28.         System.out.println("appartement-----");
29.         System.out.println(appart01);
30.         // les beans récupérés sont des singltons
31.         // on peut les demander plusieurs fois, on récupère toujours le même bean
32.         Personne p01b = ctx.getBean("personne_01", Personne.class);
33.         System.out.println(String.format("beans [p01,p01b] identiques ? %s", p01b ==
p01));
34.     }
35. }
```

### C#

```
1. using System;
2.
3. namespace Chap1 {
4.     class Impots {
5.         static void Main(string[] args) {
6.             // tableaux de données nécessaires au calcul de l'impôt
7.             decimal[] limites = { 4962M, 8382M, 14753M, 23888M, 38868M, 47932M, 0M
};
8.             decimal[] coeffR = { 0M, 0.068M, 0.191M, 0.283M, 0.374M, 0.426M,
0.481M };
9.             decimal[] coeffN = { 0M, 291.09M, 1322.92M, 2668.39M, 4846.98M,
6883.66M, 9505.54M };
10.
11.            // on récupère le statut marital
12.            bool OK = false;
13.            string reponse = null;
14.            while (!OK) {
15.                Console.Write("Etes-vous marié(e) (O/N) ? ");
16.                reponse = Console.ReadLine().Trim().ToLower();
17.                if (reponse != "o" && reponse != "n")
18.                    Console.Error.WriteLine("Réponse incorrecte.
Recommencez");
19.                else OK = true;
```

```

20.                     } //while
21.         bool marie = reponse == "o";
22.
23.         // nombre d'enfants
24.         OK = false;
25.         int nbEnfants = 0;
26.         while (!OK) {
27.             Console.WriteLine("Nombre d'enfants : ");
28.             try {
29.                 nbEnfants = int.Parse(Console.ReadLine());
30.                 OK = nbEnfants >= 0;
31.             } catch {
32.             } // try
33.             if (!OK) {
34.                 Console.WriteLine("Réponse incorrecte. Recommencez");
35.             }
36.         } // while
37.
38.         // salaire
39.         OK = false;
40.         int salaire = 0;
41.         while (!OK) {
42.             Console.WriteLine("Salaire annuel : ");
43.             try {
44.                 salaire = int.Parse(Console.ReadLine());
45.                 OK = salaire >= 0;
46.             } catch {
47.             } // try
48.             if (!OK) {
49.                 Console.WriteLine("Réponse incorrecte. Recommencez");
50.             }
51.         } // while
52.
53.         // calcul du nombre de parts
54.         decimal nbParts;
55.         if (marie) nbParts = (decimal)nbEnfants / 2 + 2;
56.         else nbParts = (decimal)nbEnfants / 2 + 1;
57.         if (nbEnfants >= 3) nbParts += 0.5M;
58.
59.         // revenu imposable
60.         decimal revenu = 0.72M * salaire;
61.
62.         // quotient familial
63.         decimal QF = revenu / nbParts;
64.
65.         // recherche de la tranche d'impôts correspondant à QF
66.         int i;
67.         int nbTranches = limites.Length;
68.         limites[nbTranches - 1] = QF;
69.         i = 0;
70.         while (QF > limites[i]) i++;
71.         // l'impôt
72.         int impots = (int)(coeffR[i] * revenu - coeffN[i] * nbParts);
73.
74.         // on affiche le résultat
75.         Console.WriteLine("Impôt à payer : {0} euros", impots);
76.     }
77. }
78. }
```

## Python

```

1. # -----
2. def affiche(chaine):
3.     # affiche chaine
4.     print("chaine=%s" % chaine)
5.
6.
7. # -----
8. def affiche_type(variable):
9.     # affiche le type de variable
10.    print("type[%s]=%s" % (variable, type(variable)))
11.
12.
```

```

13. # -----
14. def f1(param):
15.     # ajoute 10 à param
16.     return param + 10
17.
18.
19. # -----
20. def f2():
21.     # rend un tuple de 3 valeurs
22.     return "un", 0, 100
23.
24.
25. # ----- programme principal -----
26. # ceci est un commentaire
27. # variable utilisée sans avoir été déclarée
28. nom = "dupont"
29.
30. # un affichage écran
31. print("nom=%s" % nom)
32.
33. # une liste avec des éléments de type différent
34. liste = ["un", "deux", 3, 4]
35.
36. # son nombre d'éléments
37. n = len(liste)
38.
39. # une boucle
40. for i in range(n):
41.     print("liste[%d]=%s" % (i, liste[i]))
42.
43. # initialisation de 2 variables avec un tuple
44. (chaine1, chaine2) = ("chaine1", "chaine2")
45.
46. # concaténation des 2 chaînes
47. chaine3 = chaine1 + chaine2
48.
49. # affichage résultat
50. print("[%s,%s,%s]" % (chaine1, chaine2, chaine3))
51.
52. # utilisation fonction
53. affiche(chaine1)
54.
55. # le type d'une variable peut être connu
56. affiche_type(n)
57. affiche_type(chaine1)
58. affiche_type(liste)
59.
60. # le type d'une variable peut changer en cours d'exécution
61. n = "a changé"
62. affiche_type(n)
63.
64. # une fonction peut rendre un résultat
65. res1 = f1(4)
66. print("res1=%s" % res1)
67.
68. # une fonction peut rendre une liste de valeurs
69. (res1, res2, res3) = f2()
70. print("(res1,res2,res3)=[%s,%s,%s]" % (res1, res2, res3))
71.
72. # on aurait pu récupérer ces valeurs dans une variable
73. liste = f2()
74. for i in range(len(liste)):
75.     print("liste[%s]=%s" % (i, liste[i]))
76.
77. # des tests
78. for i in range(len(liste)):
79.     # n'affiche que les chaînes
80.     if type(liste[i]) == "str":
81.         print("liste[%s]=%s" % (i, liste[i]))
82.
83. # d'autres tests
84. for i in range(len(liste)):
85.     # n'affiche que les entiers >10
86.     if type(liste[i]) == "int" and liste[i] > 10:

```

```

87.         print("liste[%s]=%s" % (i, liste[i]))
88.
89. # une boucle while
90. liste = (8, 5, 0, -2, 3, 4)
91. i = 0
92. somme = 0
93. while i < len(liste) and liste[i] > 0:
94.     print("liste[%s]=%s" % (i, liste[i]))
95.     somme += liste[i] # somme=somme+liste[i]
96.     i += 1 # i=i+1
97. print("somme=%s" % somme)
98. # fin programme

```

## PHP

```

1.  <?php
2.
3. // types stricts pour les paramètres de fonctions
4. declare(strict_types=1);
5.
6. // constantes globales
7. define("PLAFOND_QF_DEMI_PART", 1551);
8. define("PLAFOND_REVENS_CELIBATAIRE_POUR_REDUCTION", 21037);
9. define("PLAFOND_REVENS_COUPLE_POUR_REDUCTION", 42074);
10. define("VALEUR_REDUC_DEMI_PART", 3797);
11. define("PLAFOND_DECOTE_CELIBATAIRE", 1196);
12. define("PLAFOND_DECOTE_COUPLE", 1970);
13. define("PLAFOND_IMPOT_COUPLE_POUR_DECOTE", 2627);
14. define("PLAFOND_IMPOT_CELIBATAIRE_POUR_DECOTE", 1595);
15. define("ABATTEMENT_DIXPOURCENT_MAX", 12502);
16. define("ABATTEMENT_DIXPOURCENT_MIN", 437);
17.
18. // définition des constantes locales
19. $DATA = "taxpayersdata.txt";
20. $RESULTATS = "resultats.txt";
21. $limites = array(9964, 27519, 73779, 156244, 0);
22. $coeffR = array(0, 0.14, 0.3, 0.41, 0.45);
23. $coeffN = array(0, 1394.96, 5798, 13913.69, 20163.45);
24.
25. // lecture des données
26. $data = fopen($DATA, "r");
27. if (!$data) {
28.     print "Impossible d'ouvrir en lecture le fichier des données [$DATA]\n";
29.     exit;
30. }
31.
32. // ouverture fichier des résultats
33. $résultats = fopen($RESULTATS, "w");
34. if (!$résultats) {
35.     print "Impossible de créer le fichier des résultats [$RESULTATS]\n";
36.     exit;
37. }
38.
39. // on exploite la ligne courante du fichier des données
40. while ($ligne = fgets($data, 100)) {
41.     // on enlève l'éventuelle marque de fin de ligne
42.     $ligne = cutNewLineChar($ligne);
43.     // on récupère les 3 champs marié:enfants:salaire qui forment $ligne
44.     list($marié, $enfants, $salaire) = explode(",", $ligne);
45.     // on calcule l'impôt
46.     $result = calculImpot($marié, (int) $enfants, (float) $salaire, $limites, $coeffR, $coeffN);
47.     // on inscrit le résultat dans le fichier des résultats
48.     $result = ["marié" => $marié, "enfants" => $enfants, "salaire" => $salaire] + $result;
49.     fputs($résultats, json_encode($result, JSON_UNESCAPED_UNICODE) . "\n");
50.     // donnée suivante
51. }
52. // on ferme les fichiers
53. fclose($data);
54. fclose($résultats);
55.
56. // fin
57. exit;
58.

```

```

59. // -----
60. function cut.NewLinechar(string $ligne): string {
61.     // on supprime la marque de fin de ligne de $ligne si elle existe
62.     $L = strlen($ligne); // longueur ligne
63.     while (substr($ligne, $L - 1, 1) === "\n" or substr($ligne, $L - 1, 1) === "\r") {
64.         $ligne = substr($ligne, 0, $L - 1);
65.         $L--;
66.     }
67.     // fin
68.     return($ligne);
69. }
70.
71. // calcul de l'impôt
72. // -----
73. function calculImpot(string $marié, int $enfants, float $salaire, array $limites, array
    $coeffR, array $coeffN): array {
74.     ...
75.     // résultat
76.     return ["impôt" => floor($impot), "surcôte" => $surcôte, "décôte" => $décôte, "réduction" =>
        $réduction, "taux" => $taux];
77. }
78.
79. // -----
80. function calculImpot2(string $marié, int $enfants, float $salaire, array $limites, array
    $coeffR, array $coeffN): array {
81.     ...
82.     // résultat
83.     return ["impôt" => $impôt, "surcôte" => $surcôte, "taux" => $coeffR[$i]];
84. }
85.
86. // revenuImposable=salaireAnnuel-abattement
87. // l'abattement a un min et un max
88. function getRevenuImposable(float $salaire): float {
89.     ...
90.     // résultat
91.     return floor($revenuImposable);
92. }
93.
94. // calcule une décôte éventuelle
95. function getDecote(string $marié, float $salaire, float $impots): float {
96.     ...
97.     // résultat
98.     return ceil($décôte);
99. }
100.
101. // calcule une réduction éventuelle
102. function getRéduction(string $marié, float $salaire, int $enfants, float $impots): float {
103.     ...
104.     // résultat
105.     return ceil($réduction);
106. }

```

## ECMAScript

```

1. 'use strict';
2. // ceci est un commentaire
3. // constante
4. const nom = "dupont";
5. // un affichage écran
6. console.log("nom : ", nom);
7. // un tableau avec des éléments de type différent
8. const tableau = ["un", "deux", 3, 4];
9. // son nombre d'éléments
10. let n = tableau.length;
11. // une boucle
12. for (let i = 0; i < n; i++) {
13.     console.log("tableau[", i, "] = ", tableau[i]);
14. }
15. // initialisation de 2 variables avec le contenu d'un tableau
16. let [chaine1, chaine2] = ["chaine1", "chaine2"];
17. // concaténation des 2 chaînes
18. const chaine3 = chaine1 + chaine2;
19. // affichage résultat

```

```

20. console.log([chaine1, chaine2, chaine3]);
21. // utilisation fonction
22. affiche(chaine1);
23. // le type d'une variable peut être connu
24. afficheType("n", n);
25. afficheType("chaine1", chaine1);
26. afficheType("tableau", tableau);
27. // le type d'une variable peut changer en cours d'exécution
28. n = "a changé";
29. afficheType("n", n);
30. // une fonction peut rendre un résultat
31. let res1 = f1(4);
32. console.log("res1=", res1);
33. // une fonction peut rendre un tableau de valeurs
34. let res2, res3;
35. [res1, res2, res3] = f2();
36. console.log("(res1,res2,res3)=", [res1, res2, res3]);
37. // on aurait pu récupérer ces valeurs dans un tableau
38. let t = f2();
39. for (let i = 0; i < t.length; i++) {
40.   console.log("t[i]=", t[i]);
41. }
42. // des tests
43. for (let i = 0; i < t.length; i++) {
44.   // n'affiche que les chaînes
45.   if (typeof (t[i]) === "string") {
46.     console.log("t[i]=", t[i]);
47.   }
48. }
49. // opérateurs de comparaison == et ===
50. if ("2" == 2) {
51.   console.log("avec l'opérateur ==, la chaîne 2 est égale à l'entier 2");
52. } else {
53.   console.log("avec l'opérateur ==, la chaîne 2 n'est pas égale à l'entier 2");
54. }
55. if ("2" === 2) {
56.   console.log("avec l'opérateur ===, la chaîne 2 est égale à l'entier 2");
57. } else {
58.   console.log("avec l'opérateur ===, la chaîne 2 n'est pas égale à l'entier 2");
59. }
60. // d'autres tests
61. for (let i = 0; i < t.length; i++) {
62.   // n'affiche que les entiers >10
63.   if (typeof (t[i]) === "number" && Math.floor(t[i]) === t[i] && t[i] > 10) {
64.     console.log("t[i]=", t[i]);
65.   }
66. }
67. // une boucle while
68. t = [8, 5, 0, -2, 3, 4];
69. let i = 0;
70. let somme = 0;
71. while (i < t.length && t[i] > 0) {
72.   console.log("t[i]=", t[i]);
73.   somme += t[i];
74.   i++;
75. }
76. console.log("somme=", somme);
77.
78. // arrêt du programme car il n'y a plus de code exécutable
79.
80. //affiche
81. //-----
82. function affiche(chaine) {
83.   // affiche chaine
84.   console.log("chaine=", chaine);
85. }
86.
87. //afficheType
88. //-----
89. function afficheType(name, variable) {
90.   // affiche le type de variable
91.   console.log("type[variable ", name, "]=", typeof (variable));
92. }
93.

```

```

94. //-----
95. function f1(param) {
96.   // ajoute 10 à param
97.   return param + 10;
98. }
99.
100. //-----
101.function f2() {
102.   // rend 3 valeurs
103.   return ["un", 0, 100];
104. }

```

## VBScript

```

1.      ' calcul de l'impôt d'un contribuable
2.      ' le programme doit être appelé avec trois paramètres : marié enfants salaire
3.      ' marié : caractère O si marié, N si non marié
4.      ' enfants : nombre d'enfants
5.      ' salaire : salaire annuel sans les centimes
6.
7.      ' déclaration obligatoire des variables
8.      Option Explicit
9.      Dim erreur
10.
11.     ' on récupère les arguments en vérifiant leur validité
12.     Dim marie, enfants, salaire
13.     erreur=getArguments(marie,enfants,salaire)
14.     ' erreur ?
15.     If erreur(0)<>0 Then wscript.echo erreur(1) : wscript.quit erreur(0)
16.
17.     ' on récupère les données nécessaires au calcul de l'impôt
18.     Dim limites, coeffR, coeffN
19.     erreur=getData(limites,coeffR,coeffN)
20.     ' erreur ?
21.     If erreur(0)<>0 Then wscript.echo erreur(1) : wscript.quit 5
22.
23.     ' on affiche le résultat
24.     wscript.echo "impôt=" & calculerImpot(marie,enfants,salaire,limites,coeffR,coeffN)
25.
26.     ' on quitte sans erreur
27.     wscript.quit 0

```

## HTML

```

1.  <!DOCTYPE HTML>
2.  <HTML>
3.    <head>
4.      <title>Laragon</title>
5.
6.      <link href="https://fonts.googleapis.com/css?family=Karla:400" rel="stylesheet"
7. type="text/css">
8.
9.      <style>
10.        HTML, body {
11.          height: 100%;
12.        }
13.
14.        body {
15.          margin: 0;
16.          padding: 0;
17.          width: 100%;
18.          display: table;
19.          font-weight: 100;
20.          font-family: 'Karla';
21.        }
22.
23.        .container {
24.          text-align: center;
25.          display: table-cell;
26.          vertical-align: middle;
27.        }

```

```

28.         .content {
29.             text-align: center;
30.             display: inline-block;
31.         }
32.
33.         .title {
34.             font-size: 96px;
35.         }
36.
37.         .opt {
38.             margin-top: 30px;
39.         }
40.
41.         .opt a {
42.             text-decoration: none;
43.             font-size: 150%;
44.         }
45.
46.         a:hover {
47.             color: red;
48.         }
49.     </style>
50. </head>
51. <body>
52.     <div class="container">
53.         <div class="content">
54.             <div class="title" title="Laragon">Laragon</div>
55.
56.             <div class="info"><br />
57.                 Apache/2.4.35 (Win64) OpenSSL/1.1.0i PHP/7.2.11<br />
58.                 PHP version: 7.2.11 <span><a title="phpinfo()" href="/?
q=info">info</a></span><br />
59.                 Document Root: C:/myprograms/laragon-lite/www<br />
60.
61.             </div>
62.             <div class="opt">
63.                 <div><a title="Getting Started" href="https://laragon.org/docs">Getting
Started</a></div>
64.             </div>
65.         </div>
66.
67.     </div>
68. </body>
69. </HTML>

```

## XML

```

1.    <?xml version="1.0" encoding="utf-8" ?>
2.    <configuration>
3.
4.        <configSections>
5.            <sectionGroup name="spring">
6.                <section name="context" type="Spring.Context.Support.ContextHandler, Spring.Core" />
7.                <section name="objects" type="Spring.Context.Support.DefaultSectionHandler,
Spring.Core" />
8.            </sectionGroup>
9.        </configSections>
10.
11.        <spring>
12.            <context>
13.                <resource uri="config://spring/objects" />
14.            </context>
15.            <objects xmlns="http://www.springframework.net">
16.                <object name="dao" type="Dao.DataBaseImpot, ImpotsV7-dao">
17.                    <constructor-arg index="0" value=" MySql.Data.MySqlClient" />
18.                    <constructor-arg index="1"
value="Server=localhost;Database=bdimpots;Uid=admimpots;Pwd=mdpimpots;" />
19.                    <constructor-arg index="2" value="select limite, coeffr, coeffn from tranches" />
20.                </object>
21.                <object name="metier" type="Metier.ImpotMetier, ImpotsV7-metier">

```

```
22.          <constructor-arg index="0" ref="dao"/>
23.          </object>
24.        </objects>
25.      </spring>
26.    </configuration>
```

## Résultats d'exécution

On notera que le code ne commence pas avec la ligne n° 1.

```
7.  C:\Data\st-2020\dev\python\cours-2020\python3-flask-2020\venv\Scripts\python.exe
    C:/Data/st-2020/dev/python/cours-2020/python3-flask-2020/bases/bases_01.py
8.  nom=dupont
9.  liste[0]=un
10. liste[1]=deux
11. liste[2]=3
12. liste[3]=4
13. [chaine1,chaine2,chaine1chaine2]
14. chaine=chaine1
15. type[4]=<class 'int'>
16. type[chaine1]=<class 'str'>
17. type[['un', 'deux', 3, 4]]=<class 'list'>
18. type[a changé]=<class 'str'>
19. res1=14s
20. (res1,res2,res3)=[un,0,100]
21. liste[0]=un
22. liste[1]=0
23. liste[2]=100
24. liste[0]=8
25. liste[1]=5
26. somme=13
27.
28. Process finished with exit code 0
```

Ce cas a nécessité des dizaines d'itérations. Le convertisseur ne trouvait jamais le numéro de la première ligne de code. Finalement j'ai demandé à Gemini comment formater le bloc de code pour qu'il le reconnaisse. Voici ce qu'il m'a répondu :

Dans LibreOffice :

1. Placez votre curseur sur la **première ligne** de votre bloc de code numéroté (celle qui doit être le n°5).
2. Faites un **Clic-Droit > Paragraphe > Paragraph...**
3. Allez dans l'onglet **Plan & numérotation** (ou *Outline & Numbering*).
4. Cochez la case **Recommencer la numérotation** (*Restart numbering at this paragraph*).
5. Cochez/Réglez **Commencer à** (*Start with*) sur **5**.
6. Cliquez sur **OK** et sauvegardez votre ODT.

**Pourquoi ça marche ?** Cette méthode oblige LibreOffice à inscrire un attribut explicite sur le paragraphe ou l'item de liste (`text:start-value="5"`), ce que le script détecte immédiatement. La méthode via l'icône de liste ("Personnaliser") modifie parfois seulement le style visuel sans changer la valeur logique sous-jacente dans le XML, surtout sur les documents importés de Word.

J'ai suivi cette démarche et ça a marché. Moi j'utilisais la méthode via l'icône de liste (Personnaliser). Donc on peut être amené à modifier certains éléments du document ODT pour obtenir ce qu'on veut.

## 2.3. Les liens

Le convertisseur Gemini sait conserver les liens externes du document ODT. Par exemple [Gemini 3](#) ou [Générer un script Python avec des outils d'IA].

Il sait gérer un lien vers un chapitre [Lien vers un chapitre](#)

Il ne sait pas gérer un lien vers une ancre (renvoi) : 2.1.

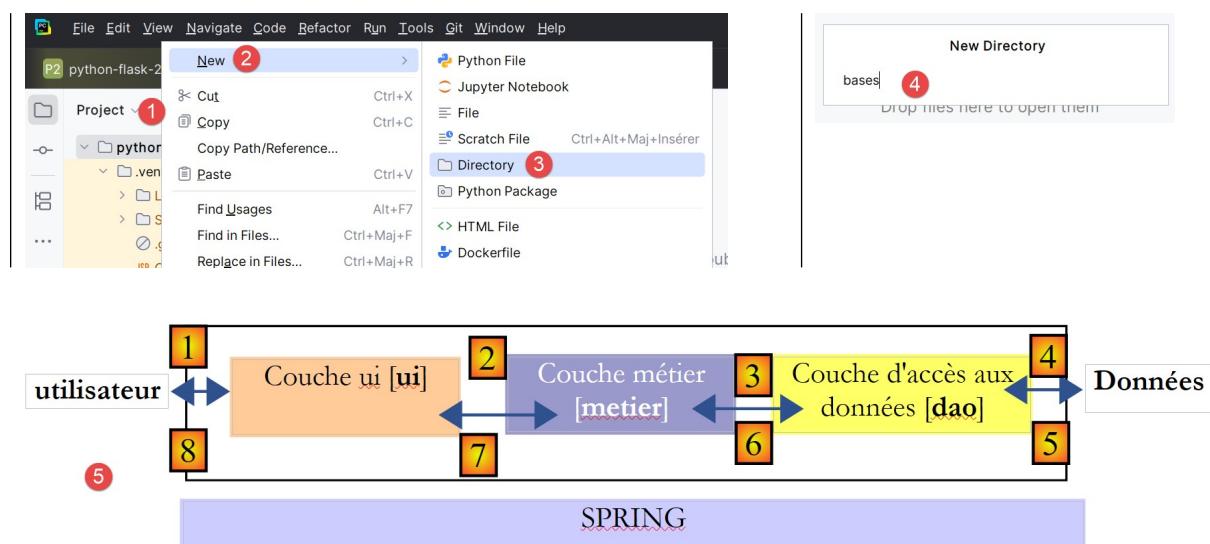
## 2.4. L'enrichissement de texte

Le convertisseur sait gérer le gras, l'italique, le souligné et le surlignage. Il respecte la couleur du surlignage.

Un texte avec des mots en **gras**, en *italiques*, soulignés ou **surlignés** ou surlignés ou **surlignés**.

## 2.5. Les images

Le convertisseur Gemini sait gérer les images et les tableaux d'images :



Dans les documents ODT il est fréquent d'avoir des dessins. Malgré des dizaines de tentatives, Gemini n'a pas réussi à générer un script qui générerait l'image (comme une copie d'écran) du dessin. Ainsi ci-dessus, l'image 5 est la copie d'écran d'un dessin d'un document ODT.

## 2.6. Les caractères à protéger

Un site MkDocs a des pages dont le contenu n'est pas du HTML mais du MarkDown. Si le document ODT contient des caractères qui existent en MarkDown, ils risquent d'être interprétés par MkDocs et donc ne pas avoir le rendu attendu. Voici deux exemples :

L'astérisque \* a une signification MarkDown. La ligne suivante peut être alors mal interprétée :

L'impôt I est alors égal à **0.15\*R – 2072.5\*nbParts**.

Un autre exemple est lorsque vous voulez insérer un bloc de code MarkDown dans votre document comme celui-ci :

```
1.      # Convertisseur Word/ODT vers Site HTML (MkDocs)
2.
3.       **[Voir le site de démonstration généré](https://stahe.github.io/word-odt-vers-html-janv-2026/)**
```

```

5.      ---
6.
7.      ## 📄 Description
8.
9.      On se propose dans ce projet de mettre à disposition du lecteur un convertisseur Python de
documents Word ou ODT vers un site statique HTML.
10.
11.     Lorsque le document ODT convient, le convertisseur produit un site HTML via **MkDocs** qui a
l'aspect professionnel des sites produits par Pandoc.
12.
13.     ## 📎 Contexte de création
14.
15.     Ce convertisseur a été entièrement construit par l'IA **Gemini 3** (avec un abonnement pro). Il
est le résultat d'itérations successives pour gérer finement la structure des documents ODT
(OpenDocument Text).
16.
17.     ## ✨ Fonctionnalités
18.
19.     Le script `convert.py` effectue les actions suivantes :
20.
21.     * **Conversion ODT vers Markdown** : Analyse le fichier `*.odt` (XML) pour en extraire la
structure.
22.     * **Gestion des Titres** : Génère automatiquement la Table des Matières (TOC) et la navigation
latérale.
23.     * **Blocs de Code** : Détection automatique des langages, coloration syntaxique et **gestion
précise de la numérotation des lignes** (attributs `start-value`).
24.     * **Listes** : Support des listes à puces et numérotées avec indentation correcte.
25.     * **Mise en forme** : Support du *gras*, *italique*, *souligné* et du *surlignage* (avec
respect des couleurs d'origine).
26.     * **Images** : Extraction et intégration automatique des images contenues dans le document.
27.     * **Configuration** : Personnalisation via un fichier `config.json` (Bas de page, Google
Analytics, etc.).
28.
29.     ## 🛠 Installation
30.
31.     ### Prérequis
32.
33.     * Python 3.x
34.     * Les bibliothèques suivantes :
35.
36.     ```bash
37.     pip install odfpy unidecode mkdocs mkdocs-material
38.
39.     ```


```

## 2.7. Les tableaux

Un tableau peut contenir différents contenus :

1      2



```

1.      package istia.st.spring.core;
2.
3.      import java.util.ArrayList;
4.      import java.util.List;
5.
6.      import
org.springframework.context.ApplicationContext;
7.      import
org.springframework.context.support.ClassPathXmlApplicationContext;
8.
9.      public class Demo01 {
10.
11.


```

3

4



```
1. # -----
2. def affiche(chaine):
3.     # affiche chaine
4.     print("chaine=%s" % chaine)
5.
6.
7. # -----
8. def affiche_type(variable):
9.     # affiche le type de variable
10.    print("type[%s]=%s" % (variable,
11.        type(variable)))
12.
```

### 3. Ce qui existe sur internet

Avant de commencer, on peut demander à l'IA ce qui existe sur le marché pour convertir un document Word ou ODT en site statique HTML et qui serait gratuit.

Posons la question à Gemini :

Existe-t-il sur internet un logiciel gratuit capable de convertir un document Word en site statique HTML ?

Voici sa réponse :

Oui, il existe plusieurs solutions gratuites, mais le choix dépend de votre définition d'un "site statique" (une simple page HTML vs un site complet avec navigation) et de vos compétences techniques.

Voici les meilleures options classées par méthode :

- **1. La solution "Tout-en-un" (Pour créer un vrai site navigable)**

Si vous voulez transformer un long document Word (avec chapitres et sous-chapitres) en un site web complet avec un menu de navigation à gauche, c'est la meilleure option.

- **Logiciel : HelpNDoc** (Version gratuite pour usage personnel)
- **Comment ça marche :** C'est un logiciel de création de documentation. Vous importez votre fichier Word, il analyse la structure (Titres 1, Titres 2...) et génère automatiquement un site HTML5 statique et "responsive" (adapté aux mobiles).
- **Les + :** Crée un vrai site avec menu, barre de recherche et index sans écrire une ligne de code.
- **Le - :** La version gratuite ajoute une petite bannière "Généré par HelpNDoc" en bas de page.
- **2. La solution "Nettoyeur en ligne" (Pour convertir page par page)**

Si vous avez un document court et que vous voulez juste obtenir le code HTML pour le mettre sur une page existante, les convertisseurs en ligne sont idéaux pour nettoyer le code "sale" généré par Word.

- **Outil : WordToHTML.net** (ou Word2CleanHTML)
- **Comment ça marche :**
  1. Vous copiez votre texte dans Word.
  2. Vous le collez dans l'éditeur visuel de gauche sur le site.
  3. Le site génère un code HTML propre (nettoyé des balises inutiles de Microsoft) dans la fenêtre de droite.

- **Les + :** Très simple, visuel, retire le "gras" du code Microsoft.
- **Le - :** Idéal pour une page unique, pas pour générer l'architecture d'un site entier.
- **3. La solution "Pro / Développeur" (La meilleure qualité)**

C'est la méthode recommandée par les experts pour obtenir un code ultra-propre et pérenne.

- **Logiciel : Pandoc** (Ligne de commande)
- **Comment ça marche :** C'est un petit programme sans interface graphique (à utiliser via le terminal/invite de commande). Il convertit les formats de manière très précise.
- **Commande type :** `pandoc mon_document.docx -t html -o mon_site.html`
- **Les + :** Code HTML5 parfait, gratuit, open-source, gère très bien les images et les formules mathématiques (LaTeX).
- **Le - :** Nécessite d'être à l'aise avec une invite de commande (écran noir).
- **4. La solution "Système D" (Microsoft Word)**

Vous pouvez le faire directement depuis Word, mais attention au résultat.

- **Méthode :** Fichier > Enregistrer sous > Choisir "**Page Web filtrée (\*.htm; \*.html)**".
- **Important :** Ne choisissez pas "Page Web à fichier unique (.mht)", choisissez bien "Filtrée". L'option "Filtrée" réduit (un peu) le code superflu que Word ajoute habituellement.
- **Le - :** Le code reste souvent lourd et difficile à modifier par la suite.

#### • Tableau Comparatif Rapide

Solution	Type de résultat	Qualité du code	Facilité	Idéal pour...
<b>HelpNDoc</b>	Site complet (avec menu)	Bonne	Moyenne	Manuels, thèses, documentation
<b>WordToHTML</b>	Page unique	Excellente	Très Facile	Articles de blog, pages simples
<b>Pandoc</b>	Page unique (brute)	Parfaite	Difficile	Développeurs, archivage propre
<b>Word (Natif)</b>	Page unique	Mauvaise	Facile	Dépannage rapide

#### • Mon conseil

Si votre objectif est de publier ce document en ligne de façon professionnelle, je vous suggère **HelpNDoc** si vous êtes débutant et que vous voulez une structure de site immédiate.

Si vous souhaitez simplement le contenu propre pour le coller ailleurs (WordPress, etc.), utilisez **WordToHTML.net**.

**Souhaitez-vous que je vous détaillle la procédure pas-à-pas pour l'un de ces outils (par exemple WordToHTML) ?**

J'ai utilisé parfois [Pandoc] mais j'avais trouvé son usage difficile. Je me rappelle qu'il fallait « préparer » le document Word pour qu'il soit correctement géré par Pandoc. Cependant le résultat était excellent. **HelpNDoc** semble un logiciel attirant puisqu'il génère un site avec menu. Je l'ai testé sur un de mes documents Word et le résultat n'a pas été satisfaisant.

Maintenant cherchons s'il existe un logiciel de conversion pour les documents ODT :

Existe-t-il sur internet un logiciel gratuit capable de transformer un document ODT en site HTML statique ?

La réponse de Gemini est la suivante :

Oui, tout à fait. Comme pour Word, le format **ODT** (utilisé par LibreOffice et OpenOffice) dispose de ses propres solutions.

Voici les meilleures options gratuites spécifiques aux fichiers ODT, classées par méthode :

- **1. La solution "Extension" (La meilleure qualité intégrée)**

Si vous utilisez LibreOffice ou OpenOffice, l'export HTML de base est (comme pour Word) assez médiocre. Cependant, il existe une extension gratuite formidable conçue spécifiquement pour cela.

- **Outil : Writer2xhtml** (Extension pour LibreOffice)

- **Comment ça marche :**

1. Vous téléchargez et installez l'extension dans LibreOffice.
2. Vous faites Fichier > Exporter et une nouvelle option "XHTML 1.1 + MathML" (ou similaire) apparaît.

- **Les + :** Le code est beaucoup plus propre que l'export standard, et il gère très bien les formules mathématiques et la mise en forme complexe.

- **Le - :** Il faut installer une extension (manipulation technique unique).

- **2. La solution "Pro / Ligne de commande" (Universelle)**

C'est le même outil que pour Word, car il excelle aussi avec le format ODT.

- **Logiciel : Pandoc**

- **Comment ça marche :** Via le terminal de commande.

- **Commande :** pandoc mon\_fichier.odt -t html -o mon\_site.html

- **Les + :** C'est sans doute le convertisseur le plus puissant du marché pour le format ODT vers HTML. Il "comprend" la structure du document ODT mieux que quiconque.

- **3. La solution "Site Complet avec Navigation" (Via HelpNDoc)**

HelpNDoc (mentionné précédemment) est toujours la meilleure option gratuite pour créer un vrai site avec menu à gauche.

Cependant, HelpNDoc n'importe pas toujours nativement le format .odt aussi bien que le .docx.

- **L'astuce :**

1. Ouvrez votre fichier dans LibreOffice.
2. Faites Enregistrer sous... et choisissez "**Word 2007-365 (.docx)**".
3. Importez ce fichier .docx dans HelpNDoc pour générer votre site.

- **Pourquoi ?** Cela garantit que la hiérarchie de vos chapitres (Titres 1, Titres 2) est parfaitement reconnue pour créer le menu du site.

- **4. La solution en ligne (Nettoyage rapide)**

Certains outils en ligne acceptent directement le ODT pour en sortir du HTML propre.

- **Outil : WordToHTML.net** (Accepte aussi les ODT)

- **Comment ça marche :** Vous pouvez uploader votre fichier ODT directement ou coller votre texte.
- **Les + :** Très pratique pour convertir un article unique sans rien installer.
- **Résumé de mon conseil pour du ODT :**
  - Pour un vrai site (Documentation/Thèse) :** Convertissez votre ODT en **DOCX** via LibreOffice, puis utilisez **HelpNDoc** (c'est la méthode la plus fiable pour avoir un menu de navigation).
  - Pour une page simple et propre :** Utilisez l'extension **Writer2xhtml** si vous faites cela souvent, ou **Pandoc** si vous voulez automatiser la tâche.

Voulez-vous que je vous explique comment installer l'extension **Writer2xhtml** dans LibreOffice ?

J'ai testé l'extension de LibreOffice [Writer2xhtml] sur un de mes documents. Ca donne quelque chose de lisible mais pas suffisamment professionnel. On n'a pas de table des matières sur la gauche. Les blocs de code perdent leur coloration syntaxique.

Au final je pense que le convertisseur généré par Gemini est une alternative à ce qui est proposé ci-dessus. Le résultat est comparable à celui produit par Pandoc si le document ODT convient au convertisseur.

## 4. Le prompt initial à Gemini 3

On suppose ici que le lecteur a une première connaissance de l'IA [Gemini](#). Si ce n'est pas le cas, il lui est conseillé de lire le document [[ref1](#)]. Le prompt initial donné à Gemini l'a été sous la forme du fichier texte [Prompt User.txt] suivant :

```

1. Demandeuses utilisateur
2.
3. - crée un script Python [convert] pour créer un site MkDocs avec le thème material à partir
   d'un document ODT. Le nom du document ODT sera l'unique argument de convert.
4.
5. - puis crée un script Python [build] qui exploitera le fichier [mkdocs.yml] créé par convert
   pour créer un site statique HTML
6. - à la fin, le script [build] affichera l'accueil du site dans un navigateur.
7.
8. - le site aura deux colonnes indépendantes. Celle de gauche contiendra la table des matières,
   celle de droite le contenu du document ODT.
9.
10. - la table des matières reprendra la totalité des titres du document ODT (Titre1, Titre2,
    Titre3, etc...), leur numérotation et leur hiérarchie
11.
12. - les images sont la plupart du temps dans des tableaux et ont été redimensionnées.
13. Tu exploreras les tableaux à la recherche de ces images et tu respecteras leur
    redimensionnement.
14.
15. - il y a des blocs de code dans le document ODT. Ils peuvent être repérés par leurs styles
    [Source Code numéroté résultats, Source Code]
16. Les autres attributs du paragraphe de code doivent être ignorés. Par exemple, si le paragraphe
    de code est également une liste numérotée, tu dois ignorer cet attribut.
17. Tous ces blocs de code doivent être traités comme du code dans le site MkDocs. Les lignes de
    code doivent être numérotées dans le résultat HTML.
18.
19. - pour tous les blocs de code tu utiliseras la coloration syntaxique de Python.
20.
21. - tu ne dois pas toucher aux lignes de code. Notamment tu ne dois pas supprimer les
    indentations ou espaces qui précèdent le premier caractère de la ligne de code.
22.
23. - tu dois trouver toutes les listes à puces. ODT a plusieurs façons de les créer. Tu dois
    toutes les explorer. Les listes de puces sont
24. parfois imbriquées les unes dans les autres. Tu dois respecter cette imbrication.
25.
26. - le titre du site sera "Générer un script Python avec des outils d'IA"
27.
28. - tu logueras chaque image extraite du document ODT ainsi que chaque chapitre MkDocs créé
29.
30. - tu dois respecter les liens que tu trouves dans le document ODT. Ils doivent rester des
    liens dans le document HTML généré.

```

```
31.  
32. - tu dois respecter les textes en gras, soulignés ou en italiques. ils doivent être rendus  
      tels quels dans le document HTML  
33.
```

- ligne 19 : le test initial a été fait sur un cours publié précédemment qui contenait des codes Python ;
- ce prompt a permis de démarrer le chat. Au bout de quelques dizaines d'itérations il est devenu obsolète ;

## 5. Créer un environnement de travail Python

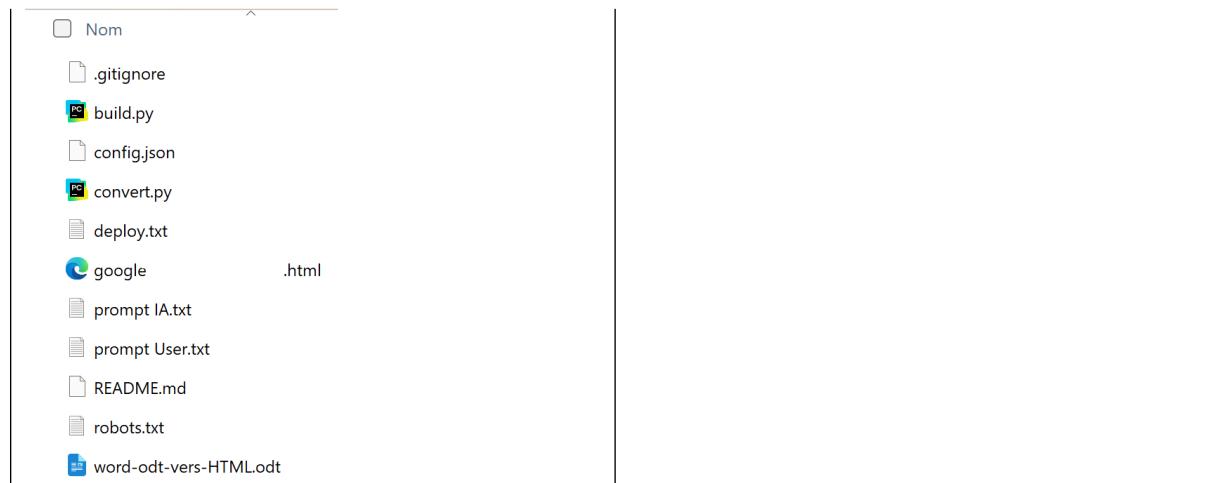
Gemini va générer deux scripts Python. Il nous faut donc un environnement Python :

- L'installation de Python est présenté au [paragraphe 3.1](#) de [ref1] ;
- L'installation de PyCharm est présenté au [paragraphe 3.2](#) de [ref1]. [ref1] présente une longue configuration de PyCharm. Ici, PyCharm ne sera utilisé que pour exécuter les scripts Python générés par Gemini. Donc toute configuration de PyCharm est inutile. PyCharm sera utilisé également pour présenter les fichiers nécessaires au convertisseur ;

## 6. Le dossier de travail du convertisseur Gemini 3

Vous pouvez télécharger le dossier de travail [ICI](#).

Le dossier de travail est le suivant :



- Nous allons ignorer pour l'instant les fichiers suivants [.gitignore, deploy.txt, google\*.html, README.md, robots.txt]. Ce sont des fichiers qui nous serviront au déploiement sur GitHub du site HTML généré localement ;
- [prompt IA.txt] est le prompt Expert que j'ai demandé à Gemini de générer lorsque j'ai estimé que le convertisseur généré était satisfaisant. Le but est de proposer ce prompt à une autre IA pour éviter les 288 itérations de Gemini. L'expérience s'est révélée négative. Elle a été proposée à chatGPT et ClaudeAI. Quand j'ai vu que les IA commençaient à diverger, j'ai arrêté la discussion. En fait le prompt IA généré par Gemini ne les aidait pas beaucoup. Même le proposer à Gemini dans une autre session n'empêche pas Gemini de faire les mêmes erreurs rencontrées dans les 288 itérations. Ce que j'ai trouvé le plus efficace pour aider une IA à démarrer, c'est de lui fournir le script généré par Gemini. A partir de là, on peut commencer à demander à l'IA de nouvelles fonctionnalités pour le convertisseur ;
- [prompt User.txt] est le prompt avec lequel j'ai commencé les itérations avec Gemini ;
- Le fichier [word.odt-vers-html.odt] est le document de cet article. Je l'ai d'abord écrit sous Word. Puis avec [[LibreOffice](#)], j'ai ouvert le fichier Word pour le sauvegarder au format ODT dans [word.odt-vers-html.odt]. Si vous avez produit un ODT via LibreOffice, vérifiez que celui-ci est bien lisible par Word. Ce dernier point est important. Il est courant d'inclure des dessins dans un document ODT. Or Word ne comprend pas ces dessins, ni le convertisseur généré par Gemini ;

Si votre document n'est pas parfaitement lisible aussi bien par Word que par LibreOffice, il est très probable que le convertisseur échouera à le convertir.

- Les scripts [convert.py] et [build.py] sont les deux scripts Python générés par Gemini ;
  - [convert] convertit un document ODT en site [[MkDocs](#)]. Un site MkDocs est un site statique dont les pages sont écrites selon la syntaxe [[MarkDown](#)]. [MkDocs] fournit un serveur capable d'afficher les sites MkDocs ;
  - [build] convertit le site MkDocs en site statique HTML standard. A la fin de cette conversion, il ouvre un navigateur pour afficher la racine du site ;

A aucun moment nous ne regarderons le code Python généré. On considérera les deux scripts comme deux boîtes noires. A aucun moment je n'ai modifié du code à la main. J'ai toujours demandé à Gemini de corriger lui-même son script. C'est pourquoi il n'est pas nécessaire de connaître Python pour utiliser le convertisseur.

- [config.json] est un fichier de configuration pour le script [convert]. Au départ [convert] n'avait pas de fichier de configuration. Puis en changeant de documents à convertir, je me suis aperçu peu à peu qu'il y avait des éléments que je demandais à Gemini de générer (par exemple le nom du site qui change pour chaque document) qui seraient mieux placés dans un fichier de configuration que l'utilisateur pourrait modifier lui-même. Je l'ai construit lui aussi par itérations avec Gemini ;

## 7. Le fichier de configuration du convertisseur Gemini 3

Le fichier de configuration [config.json] rassemble les informations qui peuvent changer d'un document à l'autre. Il commence ainsi :

```
1.      "mkdocs": {
2.          "site_name": "Convertir un document Word ou ODT vers un site statique HTML à l'aide de l'IA
Gemini",
3.          "site_url": "https://stahe.github.io/word-odt-vers-html-janv-2026/",
4.          "site_description": "Convertir un document Word ou ODT vers un site statique HTML à l'aide
de l'IA Gemini",
5.          "site_author": "Serge Tahé",
6.          "repo_url": "https://github.com/stahe/word-odt-vers-html-janv-2026",
7.          "repo_name": "GitHub",
8.          "use_directory_urls": false,
9.          "theme": {
10.              "name": "material",
11.              "custom_dir": "overrides",
12.              "features": [
13.                  "navigation.sections",
14.                  "navigation.indexes",
15.                  "navigation.expand",
16.                  "toc.integrate",
17.                  "navigation.top"
18.              ],
19.              "palette": [
20.                  {
21.                      "media": "(prefers-color-scheme: light)",
22.                      "scheme": "default",
23.                      "primary": "teal",
24.                      "accent": "purple",
25.                      "toggle": {
26.                          "icon": "material/brightness-7",
27.                          "name": "Passer au mode sombre"
28.                      }
29.                  },
30.                  {
31.                      "media": "(prefers-color-scheme: dark)",
32.                      "scheme": "slate",
33.                      "primary": "teal",
34.                      "accent": "purple",
35.                      "toggle": {
36.                          "icon": "material/brightness-4",
37.                          "name": "Passer au mode clair"
38.                      }
39.                  }
40.              ]
}
```

```

41.      },
42.      "markdown_extensions": [
43.        "admonition",
44.        "attr_list",
45.        "pymdownx.superfences",
46.        "pymdownx.mark",
47.        {
48.          "pymdownx.highlight": {
49.            "anchor_linenums": true,
50.            "linenums": true
51.          }
52.        }
53.      ],
54.      "extra_javascript": [
55.        "javascripts/focus.js"
56.      ],
57.      "extra_css": [
58.        "stylesheets/focus.css"
59.      ]
60.    },
61.    "footer": "{% block footer %}\n  <div class=\"md-footer-meta md-typeset\"\>\n    <div\n      class=\"md-footer-meta__inner\"\>\n        <div>\n          <a href=\"https://tahe.developpez.com\"\n            target=\"_blank\"\>\n            https://tahe.developpez.com\n          </a>\n          <br>\n          Ce cours-tutoriel écrit par <strong>Serge Tahé</strong> est mis à disposition du public selon les termes de la\n          <em>Licence Creative Commons Attribution - Pas d'Utilisation Commerciale -</em>\n          Partage dans les Mêmes Conditions 3.0 non transposé</em>.\n        </div>\n      </div>\n    </div>\n  </div>\n{%-\nendblock %}",
62.    "extra": {
63.      "analytics": {
64.        "provider": "google",
65.        "property": "G-XXXXXXX"
66.      }
67.    },

```

- lignes 1-67 : configuration pour la génération du fichier [mkdocs.yml]. Ce fichier est utilisé par le script [build] qui convertit le site MkDocs produit par le script [convert] en site statique HTML. Ce fichier est produit par le script [convert] ;
- ligne 2 : le nom que vous voulez donner à votre site ;
- ligne 3 : l'URL où sera déployé votre site, ici une URL GitHub ;
- ligne 4 : la description du site. J'ai repris ici le nom du site ;
- ligne 5 : le propriétaire du site ;
- ligne 6 : le dépôt GitHub sur lequel a été déployé le site HTML. Nous y revenons plus loin ;
- ligne 7 : le nom de ce dépôt ;
- ligne 8 : une ligne importante. Si elle est absente, lorsqu'on clique sur un élément de la table des matières, c'est le dossier de la page HTML qui est affiché et non le document HTML lui-même ;
- lignes 9-18 : le thème du site MkDocs ;
- ligne 10 : c'est Gemini lui-même qui m'a conseillé le thème [material] ;
- ligne 11 : cette ligne dit à MkDocs que certains éléments dont il a besoin seront dans un dossier [overrides].
- C'est le convertisseur qui va créer lui-même ce dossier. Il va y mettre deux fichiers :
  - [footer.html] qui sera le bas de page des documents HTML affichés ;
  - [analytics.html] utilisé pour faire le suivi Google Analytics du site HTML ;

C'est Gemini qui a fixé le contenu de ces fichiers suite à mes prompts ;

- ligne 16 : cette ligne permet d'avoir la table des matières dans une unique colonne à gauche de l'écran. En son absence, MkDocs utilise deux colonnes pour afficher la table des matières :
  - à droite, les chapitres de niveau Titre 1 ;
  - à gauche : les sous-chapitres du chapitre sélectionné à gauche ;
- lignes 21-28 : configure les éléments visuels du site MkDocs ;
- lignes 23-24 : définissent les couleurs du site ;
- lignes 25-28 : configurent une icône dans la barre supérieure de chaque page qui permet de passer en mode sombre ;
- lignes 31-39 : même chose que les lignes 21-28 mais lorsque l'utilisateur a sélectionné le mode sombre ;
- lignes 42-52 : les extensions au langage [MarkDown] utilisé dans les pages MkDocs. Un site MkDocs n'est pas un site HTML. C'est une étape intermédiaire ou les documents du site MkDocs sont écrit avec le langage MarkDown. C'est Gemini qui a généré ce code suite à certains de mes prompts ;

- lignes 54-59 : j'ai demandé à Gemini d'inclure dans la barre supérieure du site MkDocs générée une icône pour montrer / cacher la table des matières. C'est le script javascript [focus.js] associé à la feuille de style [focus.css] qui assure ce rôle. Les deux fichiers sont générés par Gemini ;
- ligne 61 : le bas de page des documents MkDocs / HTML affichés. C'est là que vous pouvez customiser votre bas de page ;
- ligne 62-67 : la configuration pour assurer le suivi du site par Google Analytics ;

Le fichier [config.json] se poursuit de la façon suivante :

```

68.      "code": {
69.        "style_keywords": [
70.          "code"
71.        ],
72.        "default_language": "text",
73.        "detection_rules": {
74.          "csharp": [
75.            "using",
76.            "Console.WriteLine",
77.            "public static void Main",
78.            "WebMethod",
79.            "TryParse",
80.            "EventArgs",
81.            "String.Format",
82.            "System.Web.Services"
83.          ],
84.          "java": [
85.            "System.out.println",
86.            "public static void main(String",
87.            "package",
88.            "JUnitTest",
89.            "Class.forName",
90.            "PreparedStatement",
91.            "private static void",
92.            "private void"
93.          ],
94.          "html": [
95.            "<html>",
96.            "</div>",
97.            "<body>",
98.            "<script>",
99.            "href=",
100.           "<span>",
101.           "<p>",
102.           "<h2>",
103.           "<form>",
104.           "<table"
105.         ],
106.         "sql": [
107.           "SELECT",
108.           "INSERT INTO",
109.           "UPDATE",
110.           "DELETE FROM",
111.           "WHERE"
112.         ],
113.         "python": [
114.           "def",
115.           "import",
116.           "print(",
117.           "from"
118.         ],
119.         "xml": [
120.           "<?xml",
121.           "<project",
122.           "<version>",
123.           "<configuration>",
124.           "<build>",
125.           "<dependency>"
126.         ],
127.         "javascript": [
128.           "use strict",
129.           "console.log",
130.           "let",

```

```

131.      "constructor",
132.      "async",
133.      "export"
134.    ],
135.    "php": [
136.      "<?php",
137.      "declare",
138.      "require"
139.    ],
140.    "vbscript": [
141.      "Option",
142.      "Dim",
143.      "Explicit"
144.    ]
145.  },
146.},

```

- lignes 68-146 : la configuration pour que le convertisseur :
  - détecte bien les blocs de code ;
  - reconnaîsse le langage qui est dedans ;
- lignes 69-71 : les blocs de code sont repérés par leurs styles. Dans ce tableau mettez des mots clés qu'on peut trouver dans le style de vos blocs de code. Ainsi dans le prompt initial, j'avais indiqué les styles des blocs de code : [Source Code numéroté résultats, Source Code]. [code] est un mot clé qu'on retrouve dans les deux styles et dans le document je n'ai aucun autre style ayant ce mot clé. Donc dans ce cas particulier, la ligne 70 suffit ;
- ligne 72 : lorsque le convertisseur n'arrive pas à détecter le langage du bloc de code, il mettra 'text'. C'est un marqueur MkDocs qui indique la présence de code sans coloration syntaxique ;
- lignes 73-145 : pour chaque langage rencontré, on a mis des mots clés pour aider le convertisseur à les reconnaître. Vous pouvez ajouter d'autres langages ou d'autres mots clés à un langage donné. C'est là où il faut aller si vous découvrez que le convertisseur n'a pas reconnu un langage. On peut trouver un même mot clé dans plusieurs langages. Le convertisseur sélectionne le langage qui a le plus de mots clés détectés ;

La fin du fichier [config.json] est la suivante :

```

147.  "files_to_copy": [
148.    "googlexxxx.html",
149.    "robots.txt"
150.  ]

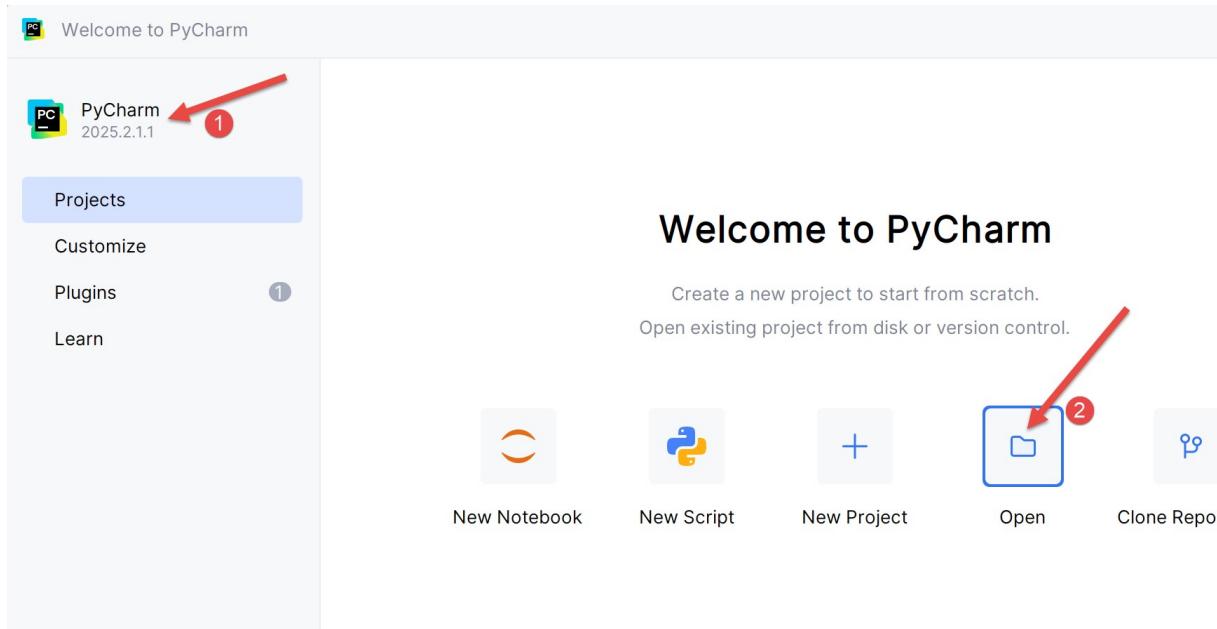
```

- lignes 147-150 : le nom des fichiers que le convertisseur doit copier à la racine du site MkDocs. Ces deux fichiers sont nécessaires au suivi du site par Google Analytics ;

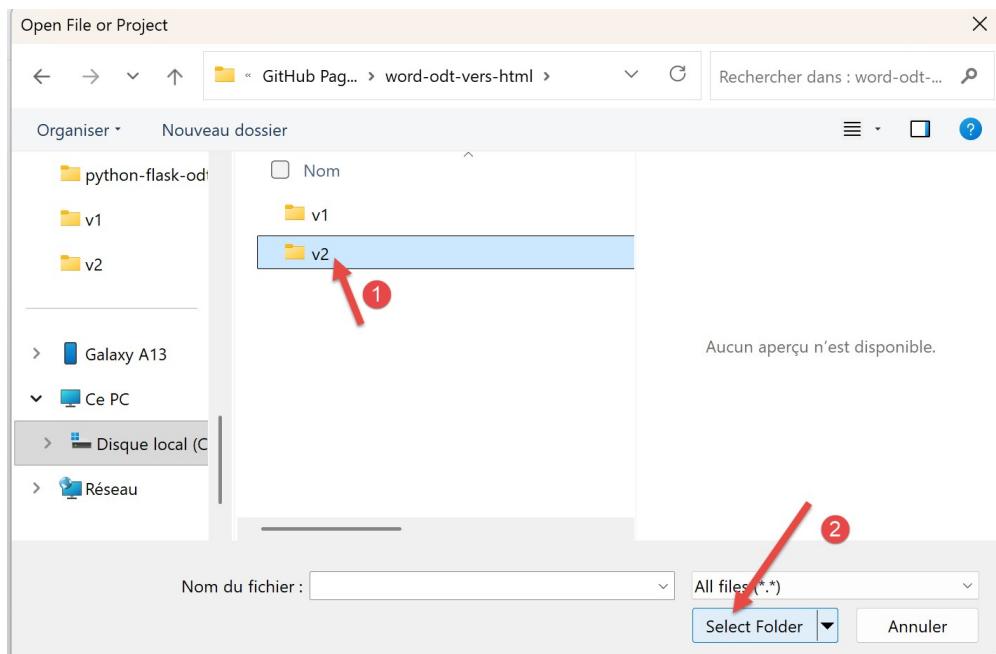
## 8. Utilisation du convertisseur Gemini 3

On rappelle que le dossier de travail peut être trouvé [ICI](#).

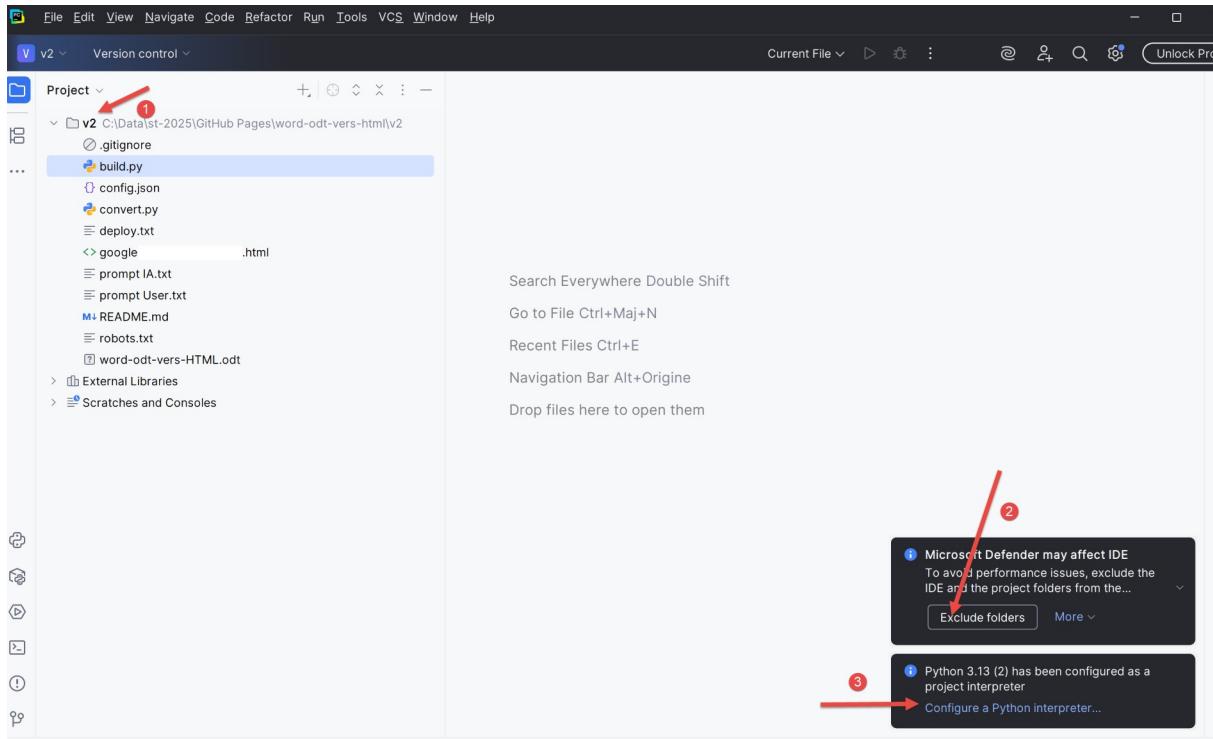
Lançons PyCharm :



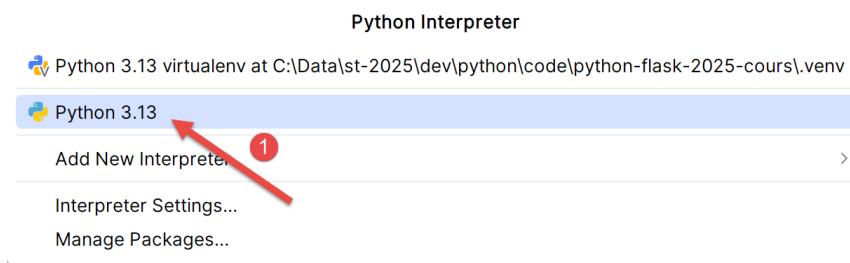
- En [2], ouvrez un nouveau projet ;



- En [1], sélectionner le dossier où vous avez sauvegardé le dossier de travail ;
- En [2], validez ce choix ;



- En [1], votre projet Python ;
- En [2], validez pour ne pas être embêté par le système de sécurité de Windows ;
- En [3], choisissez un interpréteur Python ;



- En [1], choisissez l'interpréteur Python que vous avez installé au paragraphe 5 ;

Le convertisseur Gemini utilise des dépendances Python que vous devez installer. On va demander lesquelles à Gemini :

rappelle moi toutes les dépendances Python nécessaires à ton convertisseur [convert]

La réponse de Gemini est la suivante :

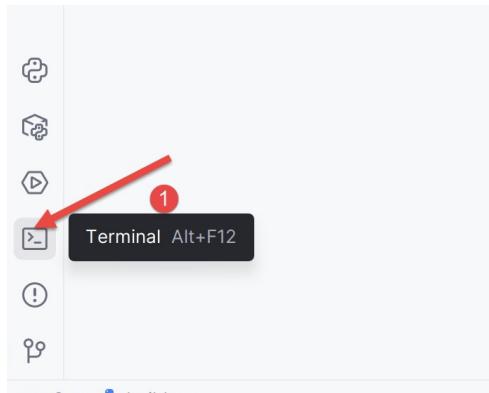
## Commande d'installation rapide

Vous pouvez installer tout le nécessaire en une seule commande dans votre terminal :

Bash

```
pip install odfpy unidecode mkdocs mkdocs-material
```

On va installer ces dépendances :



- En [1], ouvrez un terminal ;



- En [2], tapez la commande donnée par Gemini ;

La réponse est la suivante :

```
1.      PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> pip install odfpy unidecode mkdocs
mkdocs-material
2.      Defaulting to user installation because normal site-packages is not writeable
3.      Requirement already satisfied: odfpy in c:\users\serge\appdata\roaming\python\python313\site-
packages (1.4.1)
4.      Requirement already satisfied: unidecode in c:
\users\serge\appdata\roaming\python\python313\site-packages (1.4.0)
5.      Requirement already satisfied: mkdocs in c:\users\serge\appdata\roaming\python\python313\site-
packages (1.6.1)
6.      Requirement already satisfied: mkdocs-material in c:
\users\serge\appdata\roaming\python\python313\site-packages (9.7.0)
7.      Requirement already satisfied: defusedxml in c:
\users\serge\appdata\roaming\python\python313\site-packages (from odfpy) (0.7.1)
8.      Requirement already satisfied: click>=7.0 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (8.3.1)
9.      Requirement already satisfied: colorama>=0.4 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (0.4.6)
```

```

10. Requirement already satisfied: ghp-import>=1.0 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (2.1.0)
11. Requirement already satisfied: jinja2>=2.11.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (3.1.6)
12. Requirement already satisfied: markdown>=3.3.6 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (3.10)
13. Requirement already satisfied: markupsafe>=2.0.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (3.0.2)
14. Requirement already satisfied: mergedeep>=1.3.4 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (1.3.4)
15. Requirement already satisfied: mkdocs-get-deps>=0.2.0 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (0.2.0)
16. Requirement already satisfied: packaging>=20.5 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (25.0)
17. Requirement already satisfied: pathspec>=0.11.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (0.12.1)
18. Requirement already satisfied: pyyaml-env-tag>=0.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (1.1)
19. Requirement already satisfied: pyyaml>=5.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (6.0.2)
20. Requirement already satisfied: watchdog>=2.0 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (6.0.0)
21. Requirement already satisfied: babel>=2.10 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (2.17.0)
22. Requirement already satisfied: backrefs>=5.7.post1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (6.1)
23. Requirement already satisfied: mkdocs-material-extensions>=1.3 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (1.3.1)
24. Requirement already satisfied: paginate>=0.5 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (0.5.7)
25. Requirement already satisfied: pygments>=2.16 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (2.19.2)
26. Requirement already satisfied: pymdown-extensions>=10.2 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (10.17.2)
27. Requirement already satisfied: requests>=2.26 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (2.32.5)
28. Requirement already satisfied: python-dateutil>=2.8.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from ghp-import>=1.0->mkdocs) (2.9.0.post0)
29. Requirement already satisfied: platformdirs>=2.2.0 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-get-deps>=0.2.0->mkdocs) (4.5.0)
30. Requirement already satisfied: six>=1.5 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from python-dateutil>=2.8.1->ghp-import>=1.0->mkdocs) (1.17.0)
31. Requirement already satisfied: charset_normalizer<4,>=2 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from requests>=2.26->mkdocs-material) (3.4.3)
32. Requirement already satisfied: idna<4,>=2.5 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from requests>=2.26->mkdocs-material) (3.10)
33. Requirement already satisfied: urllib3<3,>=1.21.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from requests>=2.26->mkdocs-material) (2.5.0)
34. Requirement already satisfied: certifi>=2017.4.17 in c:
\users\serge\appdata\roaming\python\python313\site-packages (from requests>=2.26->mkdocs-material) (2025.8.3)
35. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2>

```

Sur mon PC, tout était déjà installé. Si ce n'est pas votre cas, toutes les dépendances demandées vont être installées.

Maintenant, nous pouvons utiliser le convertisseur. Dans le terminal ouvert, tapez la commande suivante :

```

1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\convert.py .\word-odt-vers-
HTML.odt config.json
2. --- ODT to MkDocs Converter V288 ---
3. Traitement de .\word-odt-vers-HTML.odt avec config config.json...
4. Copié : googlexxxxxxxx.html
5. Copié : robots.txt
6. Génération Markdown...
7. >>> CHAPITRE: Introduction (Niv 1)
8. >>> CHAPITRE: Les exemples (Niv 1)

```

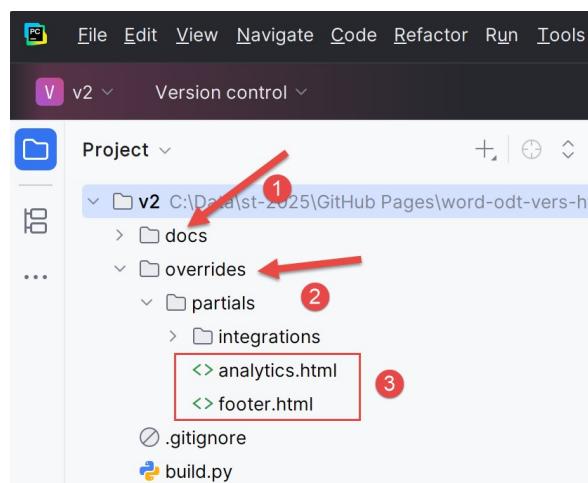
```

9.      >>> CHAPITRE: Les listes (Niv 2)
10.     >>> CHAPITRE: Les blocs de code (Niv 2)
11.     >>> CHAPITRE: Les liens (Niv 2)
12.     >>> CHAPITRE: L'enrichissement de texte (Niv 2)
13.     >>> CHAPITRE: Les images (Niv 2)
14.     >>> CHAPITRE: Ce qui existe (Niv 1)
15.     >>> CHAPITRE: Le prompt initial (Niv 1)
16.     >>> CHAPITRE: Créer un environnement de travail (Niv 1)
17.     >>> CHAPITRE: Le dossier de travail (Niv 1)
18.     >>> CHAPITRE: Le fichier de configuration [config.json] (Niv 1)
19.     Terminé.

```

- Ligne 1 : la commande qui convertit le document ODT en site MkDocs [**python .\convert.py .\word-odt-vers-HTML.odt config.json**]. Le premier paramètre du convertisseur est le document ODT à convertir, le second le fichier de configuration du convertisseur ;
- lignes 4-5 : les deux fichiers que le convertisseur copie à la racine du site MkDocs qu'il crée ;
- lignes 7-18 : j'ai demandé à Gemini de loguer tous les chapitres qu'il rencontrait ;

Cette exécution a modifié votre dossier de travail :



- En [1], [docs] est le site MkDocs que le convertisseur Gemini a créé. Vous pouvez avoir la curiosité de le visiter ;
- en [2], un dossier [overrides] a été créé. Il va être utilisé par le constructeur [build] du site HTML ;
- en [3] : [analytics.html] sera utilisé pour le suivi du site par Google Analytics. [footer.html] est le bas de page que vous avez défini dans le fichier [config.json] ;

On pourrait utiliser le site MkDocs dès maintenant. La commande [**python -m mkdocs serve**] permet de le visualiser. Vous pouvez essayer :

```

1.  PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python -m mkdocs serve
2.  INFO    - Building documentation...
3.  INFO    - Cleaning site directory
4.  INFO    - Doc file 'les-exemples.md' contains a link '#_Les_exemples', but there is no such
anchor on this page.
5.  INFO    - Documentation built in 0.41 seconds
6.  INFO    - [15:46:06] Serving on http://127.0.0.1:8000/word-odt-vers-html-janv-2026/

```

Ctrl-cliquez sur le lien de la ligne 6. Le site MkDocs doit apparaître :

Convertir un document Word ou ODT vers un site statique HTML ...

Accueil

Convertir un document Word ou ODT vers un site statique HTML à l'aide de l'IA Gemini

Accueil

1. Introduction

2. Les exemples

3. Ce qui existe

4. Le prompt initial

5. Créer un environnement de travail

6. Le dossier de travail

7. Le fichier de configuration [config.json]

8. Utilisation du convertisseur

Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec l'IA Gemini 3

Serge Tahé, janvier 2026

Voici le fruit d'un long travail. Pour arrêter le serveur de MkDocs, faites simplement Ctrl-C dans le terminal qui l'a lancé.

## 9. Construction du site HTML statique

Maintenant construisons le site HTML. Cela se fait avec le script [build]. Toujours dans le terminal, tapez la commande suivante :

```
1.      PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\build.py
2.      Démarrage de la construction du site MkDocs...
3.      Construction réussie ! Le site est dans : C:\Data\st-2025\GitHub Pages\word-odt-vers-
html\v2\site
4.      Ouverture de C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site\index.html dans le
navigateur...
```

- ligne 1 : la commande [python build] crée le site HTML à partir du site MkDocs ;
- ligne 4 : le site est affiché dans un navigateur ;

Le résultat est le même que pour le site MkDocs :

Convertir un document Word ou ODT vers un site statique HTML à l'aide de l'IA Gemini

Accueil

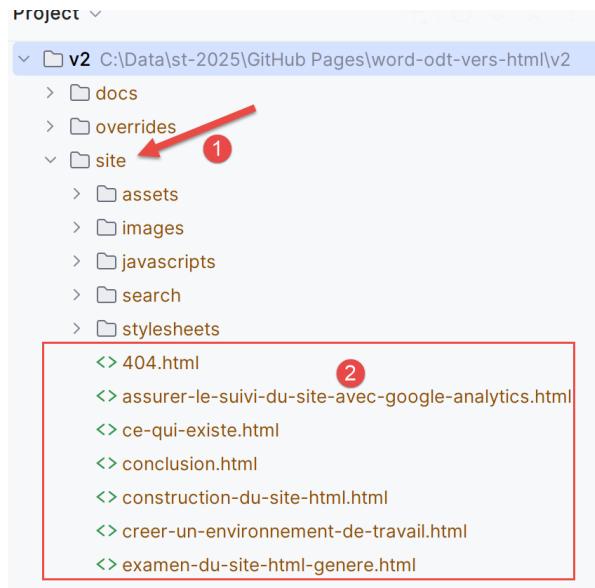
Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec l'IA Gemini 3

Serge Tahé, janvier 2026

- En [1], on voit que c'est une page HTML qui est affichée ;

- En [2], la table des matières du document HTML ;
- En [3], la première page affichée est [Accueil]. Le contenu de cette page est le contenu du document ODT qui précède le premier chapitre de niveau Titre 1 ;

L'exécution de [build] a modifié votre dossier de travail :



- En [1], un nouveau dossier [site] a été créé. Il contient votre site statique HTML ;
- En [2], mes pages de votre site ;

## 10. Examen du site HTML généré

Nous allons examiner maintenant le rendu HTML de ce document ODT. On a déjà vu que le convertisseur respectait la table des matières.

Maintenant regardons la barre supérieure du site :



- En [1], le nom du site défini dans [config.json] ;
- En [2], l'icône qui permet de passer au mode sombre ou clair ;
- En [3], l'icône qui est un lien sur le dépôt GitHub où sera exporté le site HTML. Aussi défini dans [config.json] ;
- En [4], l'icône qui permet de cacher / montrer la table des matières ;

Regardons maintenant le bas de page :



C'est le bas de page défini dans le fichier [config.json].

### Le contenu de la page d'accueil

# Accueil



## Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec l'IA Gemini 3

Serge Tahé, janvier 2026

Le convertisseur Gemini 3 met dans la page Accueil tout ce qui précède, dans le document ODT, le premier titre de niveau, de style 'Titre 1' donc. Si vous y mettez des images comme ci-dessus, il va les afficher. Vous pouvez donc imaginer donner une couverture à votre site comme avec un livre réel. En [1], c'est le titre principal du document. Son rendu est contrôlé par le lignes suivantes du fichier de configuration [config.json] :

```
1.     "document_title": {
2.         "style_names": [
3.             "Titre principal",
4.             "Title",
5.             "Titre",
6.             "P1"
7.         ],
8.         "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2; color: #2c3e50;"
9.     },
```

- lignes [2-7] : la liste des styles possibles pour le titre de votre document. Quand je regarde ce document, le style LibreOffice de mon titre est 'Titre principal'. Or le convertisseur Gemini ne le trouvait pas. Il a fait des logs des styles qu'il rencontrait et cela a affiché [P1]. C'est un gros souci avec LibreOffice : les noms affichés des styles ne correspondant pas aux noms internes utilisés par LibreOffice. Ils ne sont là que pour s'adapter à la langue de l'utilisateur. Donc lignes [2-7], seule la ligne 6 est utile ;
- ligne 8 : une fois que le titre principal a été détecté, vous pouvez paramétriser son rendu. Moi je voulais je voulais une police de taille 28 (font-size: 28px;) et du gras (font-weight: bold) ;

Avec les images et le style du titre, vous pouvez faire une page d'accueil attrayante.

Le titre principal de votre document n'aura peut-être pas un des styles définis aux lignes [3-6]. Pour trouver le style de votre titre principal, utilisez la ligne suivante du fichier [config.json]

```
1.     "debug": true
```

Avec la valeur [true], le style des paragraphes qui précèdent le premier titre de niveau 1, donc les paragraphes de la page d'accueil, va être affiché lors de l'exécution du convertisseur Gemini. Ainsi pour un autre document que celui-ci, j'ai obtenu les logs suivants :

```
1. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
2. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
3. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
4. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
5. [DEBUG PRE-H1] Style='P3' (Clean='p3') | Texte='...'
6. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
7. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
8. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
9. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
```

```
10. [DEBUG PRE-H1] Style='P4' (Clean='p4') | Texte='Introduction au langage PHP7 par l'exemp...'
11. >>> TITRE DOCUMENT DETECTÉ : Introduction au langage PHP7 par l'exemple
12. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
13. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
14. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
15. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
16. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
17. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
18. [DEBUG PRE-H1] Style='P5' (Clean='p5') | Texte='Serge Tahé, juillet 2019...'
19. [DEBUG PRE-H1] Style='P5' (Clean='p5') | Texte='...'
20. [DEBUG PRE-H1] Style='P6' (Clean='p6') | Texte='...'
21. [DEBUG PRE-H1] Style='Heading 1' (Clean='heading 1') | Texte='Introduction au langage PHP 7...'
```

- ligne 10, le titre du document a le style 'P4' ;

Dans le fichier [config.json], j'ai mis les lignes suivantes :

```
1.     "document_title": {
2.         "style_names": [
3.             "P4"
4.         ],
5.         "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2; color:
#2c3e50;"
6.     },
```

- ligne 3, le style que je cherchais ;

C'est pour cette raison que le débogueur affiche les lignes :

```
10. [DEBUG PRE-H1] Style='P4' (Clean='p4') | Texte='Introduction au langage PHP7 par l'exemp...'
11. >>> TITRE DOCUMENT DETECTÉ : Introduction au langage PHP7 par l'exemple
```

Il a rencontré le style 'P4' et affiche alors que le titre du document a été trouvé. Lorsque celui-ci a été trouvé, vous pouvez mettre la clé [debug] à [false] dans [config.json] :

```
1.     "debug": false
```

Regardons maintenant le chapitre [Exemples] qui regroupe les exemples que le convertisseur Gemini sait gérer :

## Les listes



## 2. Les exemples

Convertir un document Word ou ODT vers un site statique HTML à l'aide de l'IA Gemini

Accueil

1. Introduction

2. Les exemples

2.1. Les listes

2.2. Les blocs de code

2.3. Les liens

2.4. L'enrichissement de texte

2.5. Les images

3. Ce qui existe

4. Le prompt initial

5. Créer un environnement de travail

6. Le dossier de travail

7. Le fichier de configuration  
[config.json]

8. Utilisation du convertisseur

### 2.1. Les listes

Le convertisseur Gemini sait gérer les listes à puces et les listes numérotées :

#### Listes à puces

- Elément 1 ;
  - Elément 2 :
  - Elément 3 ;
    - Elément 3.1 ;
    - Elément 3.2 ;
  - Elément 4 ;
- Listes numérotées
- 1. Elément 1 ;
  - 2. Elément 2 ;
  - 3. Elément 3 ;

## Code Java

### 2.2. Les blocs de code

Mes cours contiennent beaucoup de blocs de code. Le convertisseur Gemini sait à présent détecter le langage mis dans un fichier de configuration. Lorsqu'il reconnaît un langage il insère les balises correspondantes et adapte la coloration syntaxique du code au langage utilisé dans le bloc de code.

#### Java

```
1 package istia.st.spring.core;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.springframework.context.ApplicationContext;
7 import org.springframework.context.support.ClassPathXmlApplicationC
8
9 public class Demo01 {
10
11     @SuppressWarnings({ "unchecked", "resource" })
12     public static void main(String[] args) {
13         // récupération du contexte Spring
14         ApplicationContext ctx = new ClassPathXmlApplicationContext(
15             // on récupère les beans
16             "classpath:beans.xml");
17
18         // on utilise les beans
19         // ...
20     }
21 }
```

## Code C#

## C#

```
1  using System;
2
3  namespace Chap1 {
4      class Impots {
5          static void Main(string[] args) {
6              // tableaux de données nécessaires au calcul de l'impôt
7              decimal[] limites = { 4962M, 8382M, 14753M, 23888M, 38868M, 47932M, 0M };
8              decimal[] coeffR = { 0M, 0.068M, 0.191M, 0.283M, 0.374M, 0.426M, 0.481M };
9              decimal[] coeffN = { 0M, 291.09M, 1322.92M, 2668.39M, 4846.98M, 6883.66M, 9505.54M };
10
11             // on récupère le statut marital
12             bool OK = false;
13             string reponse = null;
14             while (!OK) {
```

## Code Python

### Python

```
1  # -----
2  def affiche(chaine):
3      # affiche chaine
4      print("chaine=%s" % chaine)
5
6
7  # -----
8  def affiche_type(variable):
9      # affiche le type de variable
10     print("type[%s]=%s" % (variable, type(variable)))
11
12
13 # -----
14 def f1(param):
15     # ajoute 10 à param
16     return param + 10
17
```

## Code PHP

### PHP

```
1  <?php
2
3  // types stricts pour les paramètres de fonctions
4  declare(strict_types=1);
5
6  // constantes globales
7  define("PLAFOND_QF_DEMI_PART", 1551);
8  define("PLAFOND_REVENUS_CELIBATAIRE_POUR_REDUCTION", 21037);
9  define("PLAFOND_REVENUS_COUPLE_POUR_REDUCTION", 42074);
10 define("VALEUR_REDUC_DEMI_PART", 3797);
11 define("PLAFOND_DECOTE_CELIBATAIRE", 1196);
12 define("PLAFOND_DECOTE_COUPLE", 1970);
13 define("PLAFOND_IMPOT_COUPLE_POUR_DECOTE", 2627);
14 define("PLAFOND_IMPOT_CELIBATAIRE_POUR_DECOTE", 1595);
15 define("ABATTEMENT_DIXPOURCENT_MAX", 12502);
16 define("ABATTEMENT_DIXPOURCENT_MIN", 437);
17
18 // définition des constantes locales
19 $DATA = "taxpayersdata.txt";
20 $RESULTATS = "resultats.txt";
```

## Code ECMASCIPT 6

### ECMAScript

```
1 'use strict';
2 // ceci est un commentaire
3 // constante
4 const nom = "dupont";
5 // un affichage écran
6 console.log("nom : ", nom);
7 // un tableau avec des éléments de type différent
8 const tableau = ["un", "deux", 3, 4];
9 // son nombre d'éléments
10 let n = tableau.length;
11 // une boucle
12 for (let i = 0; i < n; i++) {
13     console.log("tableau[", i, "] = ", tableau[i]);
14 }
15 // initialisation de 2 variables avec le contenu d'un tableau
16 let [chaine1, chaine2] = ["chaine1", "chaine2"];
```

## Code VBScript

---

### VBScript

```
1 ' calcul de l'impôt d'un contribuable
2 ' le programme doit être appelé avec trois paramètres : marié enfants salaire
3 ' marié : caractère O si marié, N si non marié
4 ' enfants : nombre d'enfants
5 ' salaire : salaire annuel sans les centimes
6
7 ' déclaration obligatoire des variables
8 Option Explicit
9 Dim erreur
10
11 ' on récupère les arguments en vérifiant leur validité
12 Dim marie, enfants, salaire
13 erreur=getArguments(marie,enfants,salaire)
14 ' erreur ?
15 If erreur(0)<>0 Then wscript.echo erreur(1) : wscript.quit erreur(0)
16
```

## Code HTML

## HTML

```
1 <!DOCTYPE HTML>
2 <HTML>
3     <head>
4         <title>Laragon</title>
5
6         <link href="[https://fonts.googleapis.com/css
7
8             <style>
9                 HTML, body {
10                     height: 100%;
11                 }
12
13                 body {
14                     margin: 0;
15                     padding: 0;
16                     width: 100%;
17                     display: table;
18                     font-weight: 100;
19                     font-family: 'Karla';
20                 }
21
22             .container {
23                 text-align: center;
24                 display: table-cell;
25                 vertical-align: middle;
26             }

```

## Code XML

### XML

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <configuration>
3
4     <configSections>
5         <sectionGroup name="spring">
6             <section name="context" type="Spring.Context.Support.ContextHandler, Spring.Core" />
7             <section name="objects" type="Spring.Context.Support.DefaultSectionHandler, Spring.Core" />
8         </sectionGroup>
9     </configSections>
10
11     <spring>
12         <context>
13             <resource uri="config://spring/objects" />
14         </context>
15         <objects xmlns="http://www.springframework.net">
```

Un résultat d'exécution avec une première ligne qui ne commence pas à 1 :

## Résultats d'exécution

On notera que le code ne commence pas avec la ligne n° 1.

```
7 C:\Data\st-2020\dev\python\cours-2020\python3-flask-2
8 nom=dupont
9 liste[0]=un
10 liste[1]=deux
11 liste[2]=3
12 liste[3]=4
13 [chaine1,chaine2,chaine1chaine2]
14 chaine=chaine1
15 type[4]=<class 'int'>
16 type[chaine1]=<class 'str'>
17 type[['un', 'deux', 3, 4]]=<class 'list'>
18 type[a changé]=<class 'str'>
```

## Les liens

### 2.3. Les liens

Le convertisseur Gemini sait conserver les liens externes du document ODT. Par exemple [Gemini 3](#) ou [Générer un script Python avec des outils d'IA](#).

Il sait gérer un lien vers un chapitre [Lien vers un chapitre](#)

Il ne sait pas gérer un lien vers une ancre (renvoi) : 2.1.

## L'enrichissement de texte

### 2.4. L'enrichissement de texte

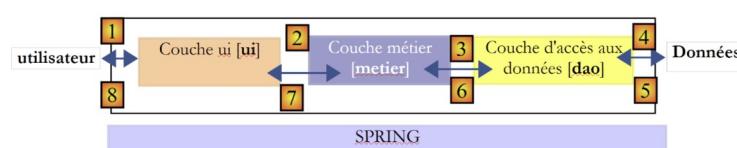
Le convertisseur sait gérer le gras, l'italique, le souligné et le surlignage. Il respecte la couleur du surlignage.

Un texte avec des mots en **gras**, en *italiques*, soulignés ou **surlignés** ou **surlignés** ou **surlignés**.

## Les images

### 2.5. Les images

Le convertisseur Gemini sait gérer les images et les tableaux d'images :



Il faut noter que le convertisseur Gemini respecte les redimensionnements des images, faits dans le document ODT.

### Les caractères protégés

L'astérisque \* a une signification MarkDown. La ligne suivante peut être alors mal interprétée :

L'impôt I est alors égal à **0.15\*R – 2072.5\*nbParts.**

Un autre exemple est lorsque vous voulez insérer un bloc de code MarkDown dans votre document

```
1 # Convertisseur Word/ODT vers Site HTML (MkDocs)
2
3  **[Voir le site de démonstration généré](https://stahe.github.io/word-odt-v)
4
5 ---
6
7 ##  Description
8
9 On se propose dans ce projet de mettre à disposition du lecteur un convertisseur
10
11 Lorsque le document ODT convient, le convertisseur produit un site HTML via **M
12
```

- En [1-3], les astérisques du texte ont été préservées ;
- En [4], le code MarkDown a été préservé ;

### Les tableaux

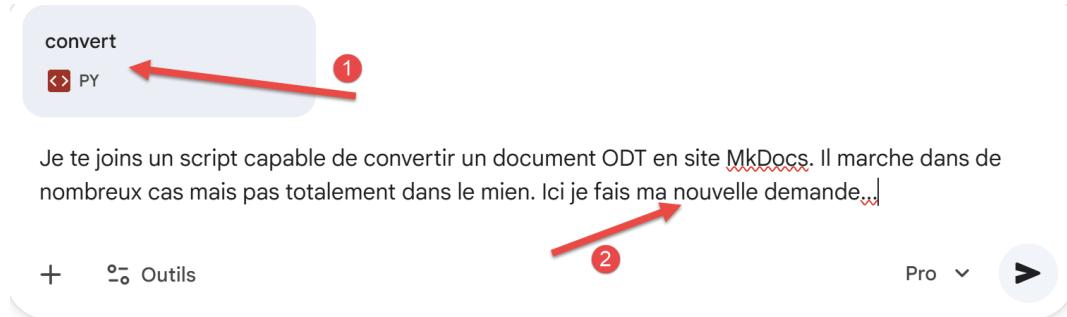
## 2.7. Les tableaux

Un tableau peut contenir différents contenus :

1	2		<pre>1 package istia.st.spring.core; 2 3 import java.util.ArrayList; 4 import java.util.List; 5 6 import org.springframework.context.ApplicationContext; 7 import org.springframework.context.support.ClassPathXmlApplicationContext; 8 9 public class Demo01 {</pre>
3	4		<pre>1 ----- 2 def affiche(chaine): 3     # affiche chaine 4     print("chaine=%s" % chaine) 5 6 7 -----# 8 def affiche_type(variable): 9     # affiche le type de variable 10    print("type[%s]=%s" % (variable, type(variable)))</pre>

Ce tableau qui avait différentes sortes de contenus a été correctement rendu par le convertisseur Gemini.

C'est tout. Si votre document utilise d'autres particularités que celles évoquées précédemment, il est hautement probable que celles-ci ne seront pas prises en compte par le convertisseur Gemini. Que faire alors ? Vous pouvez faire part de votre nouvelle demande à Gemini en lui donnant le convertisseur actuel :



- En [1], je joins le convertisseur de ce document ;
- En [2], je fais ma nouvelle demande ;

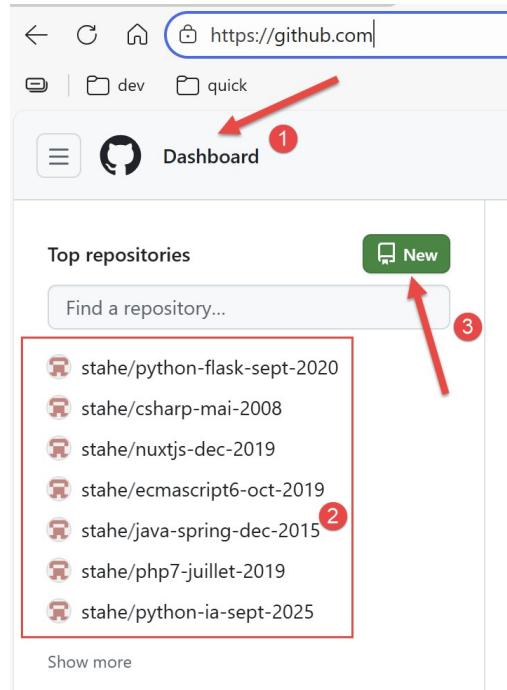
Vous allez partir probablement pour de nombreuses itérations. Lorsqu'une version est stable, notez son numéro pour pouvoir le redonner à Gemini en cas de régression. Il est conseillé également de faire une copie de chaque version stable. Un inconvénient majeur de Gemini est qu'il régresse assez facilement. Il suffit de lui demander une nouvelle fonctionnalité pour qu'il casse le code qui marchait auparavant. D'où l'importance de noter le numéro de version des versions stables afin de pouvoir y revenir.

## 11. Héberger le site HTML sur GitHub

C'est Gemini lui-même qui m'a proposé d'héberger le site HTML produit par ses deux scripts sur GitHub. Je ne savais pas que c'était possible. GitHub est un site qui héberge des projets de développement. Y exporter des cours de programmation paraît naturel.

Il faut d'abord que vous ayez un compte [GitHub](#). Si besoin est, créez le.

Connectez-vous à votre compte GitHub :



- En [2], vos dépôts (repository) existants si vous en avez ;

- En [3], créez un nouveau dépôt GitHub ;

## Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).  
Required fields are marked with an asterisk (\*).

### 1 General

Owner \* Repository name \*

 stahé /  ①

word-odt-vers-html-janv-2026 is available.

Great repository names are short and memorable. How about [sturdy-telegram](#)?

### Description

②

84 / 350 characters

### 2 Configuration

#### Choose visibility \*

Choose who can see and commit to this repository

 Public ▾

#### Add README

READMEs can be used as longer descriptions. [About READMEs](#)

Off

#### Add .gitignore

.gitignore tells git which files not to track. [About ignoring files](#)

No .gitignore ▾

#### Add license

Licenses explain how others can use your code. [About licenses](#)

No license ▾

③

**Create repository**

- En [1], utilisez le nom que vous avez mis dans [config.json] :

1. "repo\_url": "https://github.com/stahé/**word-odt-vers-html-janv-2026**",

- En [2], vous pouvez également mettre la même chose que dans [config.json] :

1. "site\_description": "Convertir un document Word ou ODT vers un site statique HTML à l'aide de l'IA Gemini",

- En [3], validez la création de votre dépôt GitHub ;

**word-odt-vers-html-janv-2026** Public

**Start coding with Codespaces**  
Add a README file and start coding in a secure, configurable, and dedicated development environment.

[Create a codespace](#)

**Quick setup — if you've done this kind of thing before**

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/stahe/word-odt-vers-html-janv-2026.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# word-odt-vers-html-janv-2026" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/stahe/word-odt-vers-html-janv-2026.git
git push -u origin main
```

Il est conseillé pour chaque dépôt GitHub de créer un fichier [README.md] qui sera affiché à la place de l'écran ci-dessus. Il doit être écrit en MarkDown, un truc pas simple. On va donc écrire un fichier texte avec dedans ce qu'on voudrait voir dans le README. Puis on demandera à Gemini de nous donner le README.md qui correspond.

Le fichier texte [readme.txt] sera le suivant :

1. On se propose dans cet article de mettre à disposition du lecteur un convertisseur Python de documents Word ou ODT vers un site statique HTML.
- 2.
3. Ce convertisseur a été construit par l'IA Gemini 3.
- 4.
5. Lorsque le document ODT convient, le convertisseur produit un site HTML via MkDocs qui a l'aspect professionnel des sites produits par Pandoc.
- 6.
7. J'ai utilisé Gemini 3 avec un abonnement pro.

Le prompt à Gemini est alors le suivant :

readme

1

2

Peux-tu produire le fichier GitHub README.md correspondant au texte que je te donne. Tu commenceras le README par un lien sur le site HTML [<https://stahe.github.io/word-odt-vers-html-janv-2026/>]

+ ⚙️ Outils Pro >

- En [1], le fichier attaché [readme.txt] ;
- En [2], le prompt ;

Copiez la réponse que Gemini vous donne dans le fichier [README.md] de votre dossier de travail. Voici la réponse que Gemini m'a donnée :

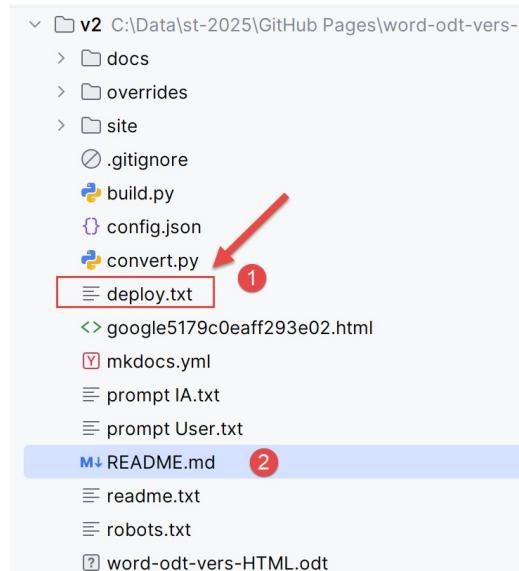
```
1.      # Convertisseur Word/ODT vers Site HTML (MkDocs)
2.
3.      ⚡ **[Voir le site de démonstration généré](https://stahe.github.io/word-odt-vers-html-janv-2026/)**
4.
5.      ---
6.
7.      ## 📄 Description
8.
9.      On se propose dans ce projet de mettre à disposition du lecteur un convertisseur Python de documents Word ou ODT vers un site statique HTML.
10.
11.     Lorsque le document ODT convient, le convertisseur produit un site HTML via **MkDocs** qui a l'aspect professionnel des sites produits par Pandoc.
12.
13.     ## 📸 Contexte de création
14.
15.     Ce convertisseur a été entièrement construit par l'IA **Gemini 3** (avec un abonnement pro). Il est le résultat d'itérations successives pour gérer finement la structure des documents ODT (OpenDocument Text).
16.
17.     ## ✨ Fonctionnalités
18.
19.     Le script `convert.py` effectue les actions suivantes :
20.
21.     * **Conversion ODT vers Markdown** : Analyse le fichier ` `.odt` (XML) pour en extraire la structure.
22.     * **Gestion des Titres** : Génère automatiquement la Table des Matières (TOC) et la navigation latérale.
23.     * **Blocs de Code** : Détection automatique des langages, coloration syntaxique et **gestion précise de la numérotation des lignes** (attributs `start-value`).
24.     * **Listes** : Support des listes à puces et numérotées avec indentation correcte.
25.     * **Mise en forme** : Support du *gras*, *italique*, *souligné* et du *surlignage* (avec respect des couleurs d'origine).
26.     * **Images** : Extraction et intégration automatique des images contenues dans le document.
27.     * **Configuration** : Personnalisation via un fichier `config.json` (Bas de page, Google Analytics, etc.).
28.
29.     ## 🔒 Installation
30.
31.     ### Prérequis
32.
33.     * Python 3.x
34.     * Les bibliothèques suivantes :
35.
36.     ```bash
37.     pip install odfpy unidecode mkdocs mkdocs-material
38.
39.     ```
40.
41.     ### Structure du projet
42.
43.     Assurez-vous d'avoir les fichiers suivants :
44.
45.     * `convert.py` : Le script de conversion.
46.     * `config.json` : Votre fichier de configuration.
47.     * `votre-document.odt` : Le document source.
48.
49.     ## 💻 Utilisation
50.
51.     1. **Conversion**
52.     Lancez le script en indiquant le fichier ODT source et le fichier de configuration :
53.     ```bash
54.     python convert.py votre-document.odt config.json
55.
56.     ```
57.
58.
59.     *Cela va générer un dossier `docs/` contenant les fichiers Markdown et un fichier `mkdocs.yml`.*
60.     2. **Prévisualisation**
```

```

61.     Pour voir le site en local :
62.         ```bash
63.             mkdocs serve
64.
65.             ...
66.
67.
68.     3. **Génération**
69.     Pour construire le site statique (dossier `site/`) :
70.         ```bash
71.             mkdocs build
72.
73.             ...
74.
75.
76.
77.     ## 🛡 Configuration (`config.json`)
78.
79.     Le fichier `config.json` permet de contrôler l'apparence du site :
80.
81.     * **mkdocs** : Paramètres généraux du site (titre, description, thème Material).
82.     * **footer** : Code HTML complet pour personnaliser le pied de page.
83.     * **code** : Règles de détection des langages pour la coloration syntaxique.
84.     * **extra** : Configuration de Google Analytics (GA4).
85.
86.     ## 📄 Licence
87.
88.     Ce cours tutoriel écrit par **Serge Tahé** est mis à disposition du public selon les termes de
la :
89.     *Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes
Conditions 3.0 non transposé.*
```

C'est un README très complet qu'il m'a fourni. C'est parce que Gemini connaît très bien ce projet sur lequel on travaille depuis des semaines. Vous aurez sans doute un [README.md] moins détaillé.

Maintenant revenons à notre dossier de travail :



- En [2], le fichier README que vous venez de modifier ;
- En [1], le fichier [deploy.txt] explique comment exporter votre site HTML sur votre dépôt GitHub ;

Le contenu du fichier [deploy.txt] est le suivant :

```

1. git init
2. git branch -M main
3. # vérifier .gitignore avant !
4. git add .
5. # premier commit
```

```
6.     git commit -m "Initial commit: Source MkDocs"
7.     git remote add origin https://github.com/stahe/word-odt-vers-html-janv-2026.git
8.     git push -u origin main
9.     python -m mkdocs gh-deploy
```

C'est la suite de commandes qui va exporter votre site HTML sur votre dépôt GitHub. Vous aurez à modifier la ligne 7 avec l'URL de votre propre dépôt GitHub, présente dans le fichier [config.json] :

```
1.     "repo_url": "https://github.com/stahe/word-odt-vers-html-janv-2026",
```

La suite de commandes de [deploy.txt] utilise un logiciel appelé Git. Vous devez l'installer [[Git - Install for Windows](#)].

Ceci fait, vérifiez le fichier s'appelant [.gitignore] dans votre dossier de travail. Il indique à Git les fichiers qu'il doit ignorer. Mon fichier [.gitignore] est le suivant :

```
1.     # 1. On ignore TOUS les fichiers et dossiers du projet
2.     *
3.
4.     # 3. On fait une exception (!) pour le seul fichier que vous voulez garder
5.     !README.md
```

Il est extrêmement simple. On ignore tous les fichiers (ligne 2) sauf le fichier [README.md] (ligne 5). GitHub est fait pour héberger les projets de développement. C'est en général tout le projet de développement qui est exporté sur GitHub. Nous on cherche simplement à exporter un site HTML, pas un projet de développement. Le seul fichier qu'on veut exporter sur notre dépôt GitHub est le fichier [README.md] qui explique aux visiteurs ce que contient notre site HTML.

Maintenant, dans votre terminal, tapez la séquence de commandes suivantes dans l'ordre indiqué par [deploy.txt] en allant jusqu'à la commande 8. N'exécutez pas la commande 9 pour l'instant.

```
1.     PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git init
2.     Initialized empty Git repository in C:/Data/st-2025/GitHub Pages/word-odt-vers-html/v2/.git/
3.     PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git branch -M main
4.     PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git add .
5.     PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git commit -m "Initial commit: Source
MkDocs"
6.     [main (root-commit) 7cba5b1] Initial commit: Source MkDocs
7.       1 file changed, 89 insertions(+)
8.       create mode 100644 README.md
9.     PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git remote add origin
https://github.com/stahe/word-odt-vers-html-janv-2026.git
10.    PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git push -u origin main
11.    Enumerating objects: 3, done.
12.    Counting objects: 100% (3/3), done.
13.    Delta compression using up to 8 threads
14.    Compressing objects: 100% (2/2), done.
15.    Writing objects: 100% (3/3), 1.70 KiB | 1.70 MiB/s, done.
16.    Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
17.    To https://github.com/stahe/word-odt-vers-html-janv-2026.git
18.      * [new branch]      main -> main
19.    branch 'main' set up to track 'origin/main'.
```

Faites un Ctrl-clic sur l'URL de la ligne 17. Cela va vous emmener sur votre dépôt GitHub :

The screenshot shows a GitHub repository page. At the top, the URL is https://github.com/stahe/word-odt-vers-html-janv-2026. A red arrow labeled '1' points to the URL bar. Below the URL bar, there's a sidebar with links like 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. To the right of the sidebar is a search bar with the placeholder 'Type / to search'. The main content area shows a commit history for the 'main' branch. The first commit is from 'stahe' titled 'Initial commit: Source MkDocs' made 4 minutes ago. Below the commit history is a file list. A red arrow labeled '2' points to the 'README' file. The 'README' file has a green 'Code' button next to it. The file content is visible, showing the title 'Convertisseur Word/ODT vers Site HTML (MkDocs)' and a link to 'Voir le site de démonstration générée'.

- En [1], l'URL de votre dépôt GitHub ;
- En [2], le nouveau README.md ;

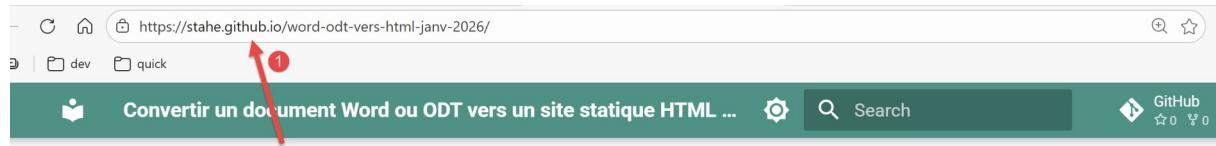
Maintenant passons à la commande 9 du fichier [deploy.txt]. C'est elle qui exporte le site HTML sur GitHub :

```

1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python -m mkdocs gh-deploy
2. INFO    - Cleaning site directory
3. INFO    - Building documentation to directory: C:\Data\st-2025\GitHub Pages\word-odt-vers-
html\v2\site
4. INFO    - Doc file 'les-exemples.md' contains a link '#_Les_exemples', but there is no such
anchor on this page.
5. INFO    - Documentation built in 1.79 seconds
6. WARNING - Version check skipped: No version specified in previous deployment.
7. INFO    - Copying 'C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site' to 'gh-pages'
branch and pushing to GitHub.
8. Enumerating objects: 91, done.
9. Counting objects: 100% (91/91), done.
10. Delta compression using up to 8 threads
11. Compressing objects: 100% (85/85), done.
12. Writing objects: 100% (91/91), 1.64 MiB | 2.01 MiB/s, done.
13. Total 91 (delta 9), reused 0 (delta 0), pack-reused 0 (from 0)
14. remote: Resolving deltas: 100% (9/9), done.
15. remote:
16. remote: Create a pull request for 'gh-pages' on GitHub by visiting:
17. remote:     https://github.com/stahe/word-odt-vers-html-janv-2026/pull/new/gh-pages
18. remote:
19. To https://github.com/stahe/word-odt-vers-html-janv-2026.git
20.   * [new branch]      gh-pages -> gh-pages
21. INFO    - Your documentation should shortly be available at: https://stahe.github.io/word-odt-
vers-html-janv-2026/

```

Faites un Ctrl-clic sur l'URL de la ligne 21. Cela va vous amener à votre nouveau site HTML sur GitHub :



Convertir un document Word ou  
ODT vers un site statique HTML  
à l'aide de l'IA Gemini

## Accueil

- Accueil
- 1. Introduction
- 2. Les exemples
- 3. Ce qui existe
- 4. Le prompt initial
- 5. Créer un environnement de travail
- 6. Le dossier de travail
- 7. Le fichier de configuration [config.json]
- 8. Utilisation du convertisseur
- 9. Construction du site HTML
- 10. Examen du site HTML généré
- 11. Héberger le site HTML sur GitHub

Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec l'IA Gemini 3

Serge Tahé, janvier 2026

GitHub  
☆ 0 ⚡ 0

- En [1], on voit que vous affichez un site web sur GitHub ;

Il est extrêmement facile de se tromper dans l'exécution des commandes de [deploy.txt]. Il est alors difficile de revenir en arrière car Git garde la mémoire de ce qu'on a fait (mal). Pour repartir de zéro, visualisez le dossier de travail :

Nom	Modifiée
.git	06/01/2
.idea	06/01/2
docs	06/01/2
overrides	06/01/2
site	06/01/2
.gitignore	27/12/2
build.py	25/12/2
config.json	06/01/2
convert.py	06/01/2
deploy.txt	06/01/2
google!.html	09/12/2
mkdocs.yml	06/01/2
prompt IA.txt	26/12/2
prompt User.txt	25/12/2
README.md	06/01/2
readme.txt	06/01/2
robots.txt	14/12/2
word-odt-vers-HTML.odt	06/01/2



- En [1], supprimez le dossier [.git] ;

Revenez ensuite dans PyChar et recommencez la série de commandes de [deploy.txt].

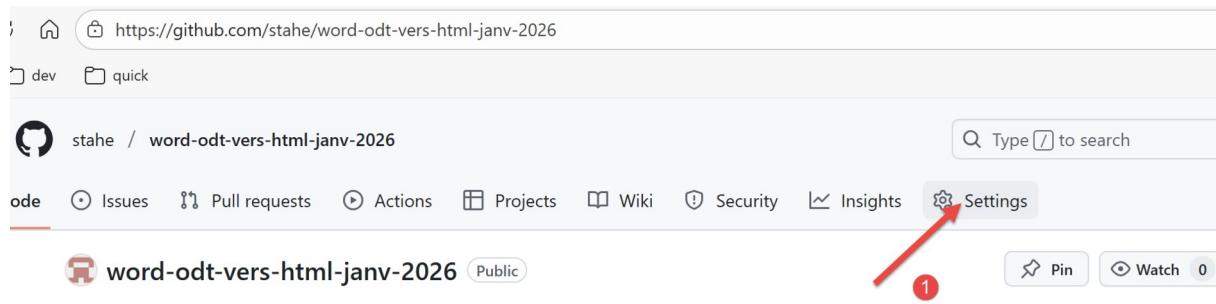
Que faire si vous modifiez votre document ODT ? Faites les 3 choses suivantes :

1. convertissez de nouveau votre document ODT avec [convert] ;
2. construisez le site HTML avec [build] ;
3. exportez le site HTML sur GitHub avec la commande [python -m mkdocs gh-deploy]. Cette commande suffit tant que vous ne modifiez pas le fichier [README.md]. Si vous modifiez le fichier [README.md], vous aurez un peu plus de commandes à émettre :

```
1. git add README.md
2. # commit
3. git commit -m "modif README"
4. git push -u origin main
5. python -m mkdocs gh-deploy
```

Si vous avez seulement modifié le README, seules les commandes 1, 3, 4 sont nécessaires. La commande 4 est inutile si vous avez déjà déployé le site HTML et qu'il n'a pas bougé depuis.

Que faire si vous voulez repartir de zéro parce que les choses sont parties en vrille ? Vous pouvez supprimer votre dépôt GitHub et refaire toutes les opérations du chapitre 11. L'option de suppression d'un dépôt GitHub est bien cachée :



- En [1], allez dans les paramètres de configuration du dépôt ;

## Danger Zone

The screenshot shows the 'Danger Zone' section of a GitHub repository settings page. It contains five items:

- Change repository visibility**: This repository is currently public. **Change visibility** button.
- Disable branch protection rules**: Disable branch protection rules enforcement and APIs. **Disable branch protection rules** button.
- Transfer ownership**: Transfer this repository to another user or to an organization where you have the ability to create repositories. **Transfer** button.
- Archive this repository**: Mark this repository as archived and read-only. **Archive this repository** button.
- Delete this repository**: Once you delete a repository, there is no going back. Please be certain. **Delete this repository** button. A red arrow labeled '2' points to this button.

Allez tout en bas de la page des [settings]. Vous y trouverez le bouton pour supprimer votre dépôt [2].

## 12. Assurer le suivi du site avec Google Analytics et Google Search Console

Si vous hébergez votre site sur GitHub vous serez probablement intéressé par connaître le nombre de visiteurs du site. Il vous faut pour ça un marqueur Google Analytics (GA). Nous demandons à ChatGPT comment créer un marqueur GA :

ChatGPT 5.2 ▾

comment obtient-t-on un marqueur Google Analytics pour un site web ?

Pour un site web, le **marqueur Google Analytics** correspond aujourd'hui à l'**identifiant de mesure GA4** (format `G-XXXXXX`). Voici la procédure complète et à jour pour l'obtenir.

### 1. Créer ou utiliser un compte Google Analytics

1. Accédez à Google Analytics et connectez-vous avec votre compte Google.
2. Si aucun compte n'existe encore :
  - Cliquez sur **Créer un compte**
  - Donnez un nom au compte (ex. : *Entreprise X*)
3. Si un compte existe déjà, sélectionnez-le.

## 2. Créer une propriété GA4

1. Dans Administration (icône engrenage, en bas à gauche),

2. Colonne Propriété → Créez une propriété,

3. Renseignez :

- Nom de la propriété
- Fuseau horaire
- Devise

4. Validez.

Depuis plusieurs années, **GA4 est le seul type de propriété disponible**. Les anciennes propriétés Universal Analytics ne sont plus utilisables.

## 3. Créez un flux de données Web

1. Toujours dans Administration,

2. Colonne Propriété → Flux de données,

3. Cliquez sur Ajouter un flux → Web,

4. Saisissez :

- L'URL du site (ex. `https://www.monsite.fr`)
- Un nom de flux

5. Validez.



- En [1], mettez l'URL de votre site web sur GitHub ;

## 4. Récupérer le marqueur (ID de mesure)

Après la création du flux web :

- Un ID de mesure est affiché, au format :

css Copier le code

👉 C'est ce marqueur Google Analytics

Il identifie de manière unique votre site dans GA4.

- En [2], notez le marqueur de votre site ;

Mettez ce marqueur dans le fichier de configuration [config.json] :

```
1.     "extra": {
2.         "analytics": {
3.             "provider": "google",
4.             "property": "G-XXXXXX"
5.         }
6.     },
```

Ligne 4, inscrivez votre marqueur. MkDocs le mettra automatiquement dans toutes vos pages HTML lorsqu'il va générer le site HTML. Vous n'avez donc pas besoin de suivre les autres instructions de ChatGPT qui montrent comment installer le marqueur GA dans vos pages HTML.

On va demander à ChatGPT des instructions précises pour vérifier que le marqueur fonctionne :

donne moi des instructions plus précises pour vérifier que le marqueur fonctionne. Donne moi l'URL à utiliser.

Voici une procédure **opérationnelle et vérifiable pas à pas** pour contrôler que votre marqueur GA4 fonctionne, avec l'URL exacte à utiliser.

### Méthode recommandée : vérification via le rapport *Temps réel*

#### 1. Accéder à Google Analytics

Ouvrez directement l'interface de [Google Analytics](https://analytics.google.com/) à l'adresse suivante :

[https://analytics.google.com/ ↗](https://analytics.google.com/)

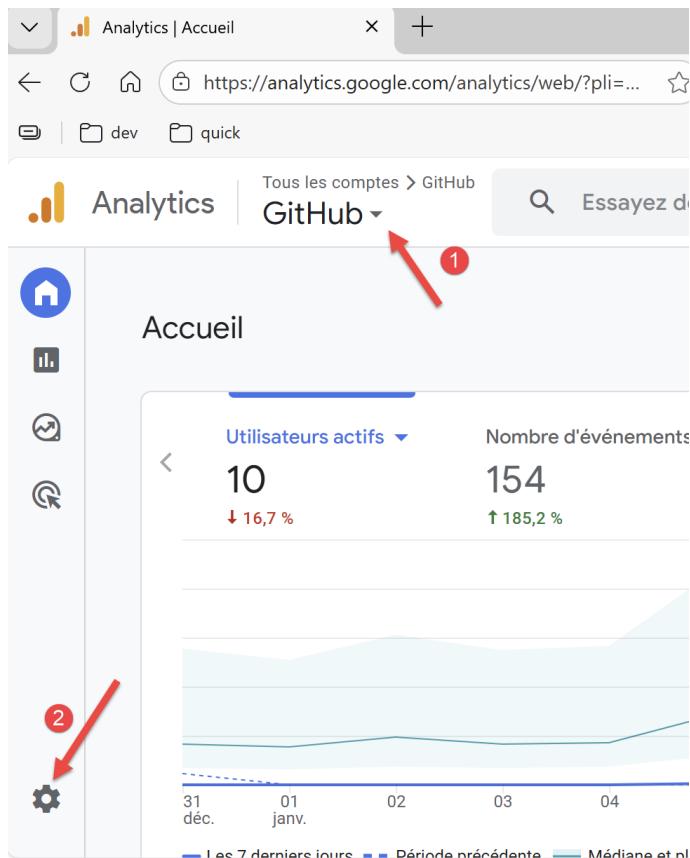
Connectez-vous avec le compte Google qui possède la propriété GA4.

#### 2. Sélectionner la bonne propriété

En haut à gauche de l'interface :

1. Vérifiez que la **propriété GA4** correspondant à votre site est bien sélectionnée.
2. Si nécessaire, changez-la via le sélecteur de compte/propriété.

L'étape précédente n'est pas assez claire. Voici quelques copies d'écran pour vous aider :



- En [1], sélectionnez le compte que vous voulez suivre (au cas où vous en auriez plusieurs) ;
- En [2], sélectionnez le bouton d'administration ;

The screenshot shows the Google Analytics Admin interface. The left sidebar has 'Administration' selected. Under 'PARAMÈTRES DU COMPTE', the 'Compte' section is active. In the main content area, there are two sections: 'Compte' and 'Propriété'. The 'Compte' section contains links for 'Détails du compte', 'Gestion des accès au compte', and 'Historique des modifications du compte'. The 'Propriété' section contains links for 'Détails de la propriété' and 'Gestion des accès à la propriété'. Below these are sections for 'Collecte et modification des données' and 'Associations de produits'. A red box highlights the 'Flux de données' link under 'Collecte et modification des données', with a red number '3' indicating it's the target for step [3].

- En [3], sélectionnez [Flux de données] ;

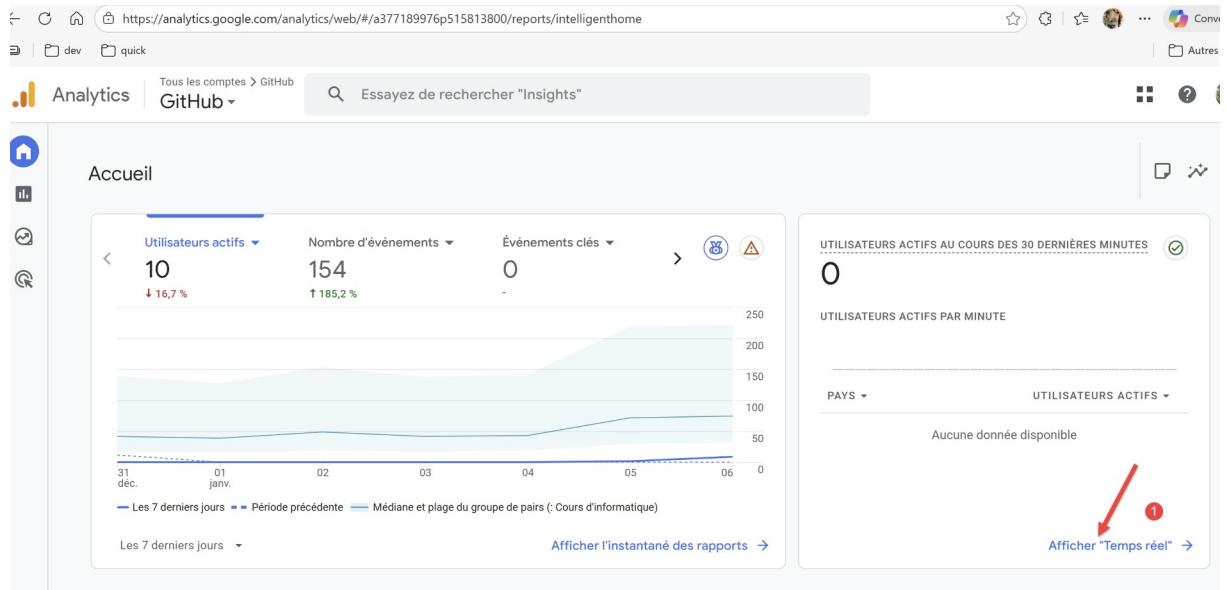
The screenshot shows the 'Flux de données' (Data Stream) page. It lists a single stream for 'GitHub' with the URL 'https://stahe.github.io'. The stream has an ID '13112486249' and a status message 'La propriété a reçu du trafic au cours des dernières 48 heures.' A red box highlights the 'GitHub' row, with a red number '4' indicating it's the target for step [4].

- En [4], cliquez sur le flux de données ;

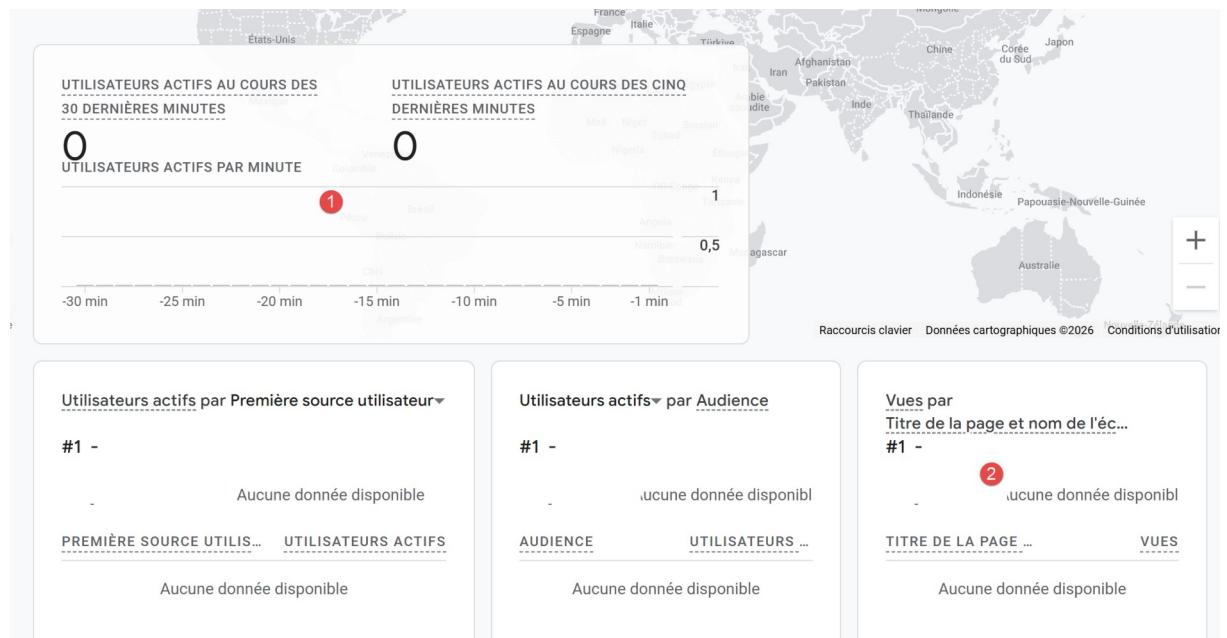
The screenshot shows the 'Détails du flux Web' (Web Data Stream Details) page. It displays the stream name 'GitHub', URL 'https://stahe.github.io', and ID '13112486249'. A green checkmark indicates data collection was active in the last 48 hours. The 'ID DE MESURE' field contains 'G-EBZC: 5', which is highlighted with a red box and a red number '5'.

- En [5], vous retrouvez le marqueur GA4 que vous avez créé ;
- En [6], l'ID de flux. C'est une notion différente de l'ID de mesure ;

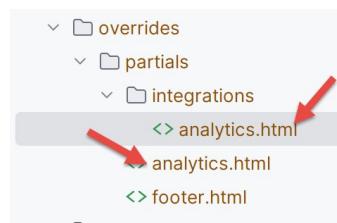
Revenons à l'accueil de GA :



- En [1], demandez l'affichage temps réel des visites ;



- En [1], vous voyez vos visiteurs ;
- En [2], la page qui a été visitée. J'ai demandé au convertisseur Gemini d'associer le nom du site plutôt que le nom de page au marqueur GA4. C'est le fichier [analytics.html] qui assure ce changement de nom associé au GA4 :



Maintenant on peut continuer avec les explications de ChatGPT :

#### 4. Générer une visite test propre

Pour éviter toute ambiguïté :

1. Ouvrez une fenêtre de navigation privée (Ctrl+Maj+N ou Ctrl+Maj+P).
2. Saisissez l'URL exacte de votre site (ex. <https://www.monsite.fr> ).
3. Naviguez sur une ou deux pages.
4. Attendez 5 à 30 secondes.



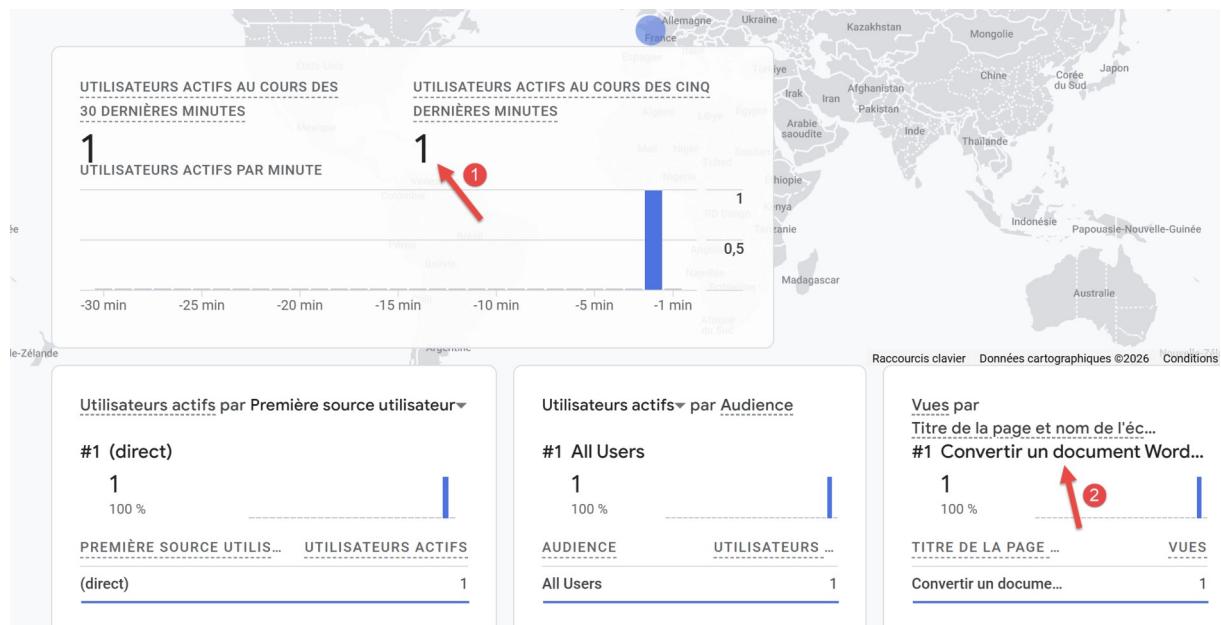
#### 5. Résultat attendu

Dans le rapport *Temps réel*, vous devez voir au minimum :

- 1 utilisateur actif
- Une page consultée
- Une source de trafic (souvent *Direct*)

Si ces éléments apparaissent → le marqueur fonctionne correctement.

J'ai fait ce qui est demandé en [4] sans passer par une fenêtre privée. Et j'ai obtenu ceci :



Le marqueur GA4 de Google Analytics est bien fonctionnel.

Maintenant nous allons présenter un autre outil de suivi de votre site [Google Search Console]. Demandons à ChatGPT à quoi sert cet outil :

### Conclusion

Google Search Console sert à :

- comprendre comment Google perçoit votre site,
- améliorer votre référencement naturel (SEO),
- détecter rapidement les problèmes bloquants,
- piloter votre visibilité dans Google Search.

Je ne me sers quasiment jamais de cet outil. Je ne le fais que pour inciter le moteur de recherche Google à l'explorer et à l'indexer afin que des internautes puissent le trouver.

L'URL de l'outil [<https://search.google.com/search-console>] :

The screenshot shows the Google Search Console interface. On the left, there's a sidebar with a dropdown menu set to 'https://stahe.github.io/...', a 'Vue d'ensemble' button, and an 'Insights' button. A red arrow labeled '1' points to the dropdown menu. On the right, the main area has a header 'Google Search Console' and a sub-header 'Propriété de recherche'. It shows two sections: 'DEVELOPPEZ.COM' and 'STAHE.GITHUB.IO'. Under 'STAHE.GITHUB.IO', there are several URLs listed, each with a small circular icon. One URL, 'https://stahe.github.io/python-ia-sept-2025/', is highlighted with a red arrow labeled '3'. At the bottom of this list, there's a button labeled '+ Ajouter la propriété' with a red arrow labeled '4' pointing to it.

- En [1], la liste des sites qui ont été déclarés à Google ;
- En [3], la liste de vos sites ;
- En [4], ajoutez votre nouveau site ;

## Sélectionnez le type de propriété

The screenshot shows the 'Select property type' step in Google Search Console. It offers two choices: 'Domaine' (Domain) and 'Préfix de l'URL' (URL prefix). The 'Préfix de l'URL' option is highlighted. A red arrow points to the URL input field where 'iub.io/word-odt-vers-html-janv-2026/' is typed. Another red arrow points to the 'CONTINUER' button below the URL field.

- En [5], tapez l'URL de votre site. Elle est dans le fichier [config.json] :

```
1.     "site_url": "https://stahe.github.io/word-odt-vers-html-janv-2026/",
```

- en [2], passez à l'étape suivante ;

Normalement à cette étape, Google Search Console vous propose un fichier [googlexxxxx.html] que vous devez télécharger. Je n'ai pas eu cet écran car j'avais déjà ce fichier dans le site déployé. C'est l'un des deux fichiers déclarés dans [config.json]

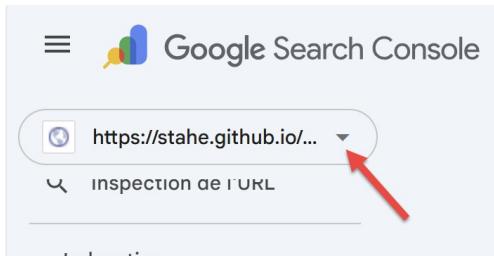
```
1.     "files_to_copy": [
2.       "google5179xxxxx.html",
3.       "robots.txt"
4.     ]
```

Mettez le fichier [googlexxxxx.html] à la racine de votre dossier de travail et inscrivez son nom à la ligne 2 ci-dessus du fichier [config.json]. Le script [convert] s'occupe de mettre les deux fichiers ci-dessus à la racine du site MkDocs qu'il génère. Le script [build] lui les mettra à la racine du site HTML généré.

Lorsque vous avez copié le fichier [googlexxxxx.html] à la racine de votre dossier de travail, redéployez votre site web sur GitHub en tapant les trois commandes suivantes dans votre terminal Python :

```
1.     python .\convert.py .\word-odt-vers-HTML.odt config.json
2.     python .\build.py
3.     python -m mkdocs gh-deploy
```

Ceci fait, retournez dans [Google Serach Console] et sélectionnez la propriété que vous avez créée précédemment :



Google Search Console

https://stahe.github.io/... ▾

inspection de l'URL

Sitemaps

Ajouter un sitemap

https://stahe.github.io/word-odt-vers-html-janv-2026/  ②

ENVOYER ③

- En [1], sélectionnez l'option [Sitemaps] ;
- en [2], tapez [sitemap.xml] ;
- En [3], validez l'URL ;

Le fichier [sitemap.xml] est un fichier que Mkdocs a mis à la racine du site HTML exporté sur GitHub. Vous pouvez vérifier sa présence en tapant directement son URL :

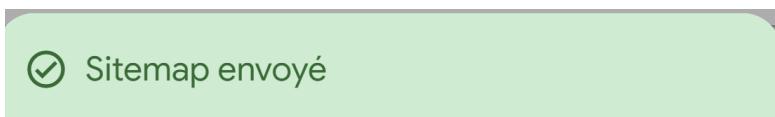


This XML file does not appear to have any style information associated with it. The document tree is shown below

```
▼<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  ▼<url>
    <loc>https://stahe.github.io/word-odt-vers-html-janv-2026/index.html</loc>
    <lastmod>2026-01-07</lastmod>
  </url>
  ▼<url>
    <loc>https://stahe.github.io/word-odt-vers-html-janv-2026/ce-qui-existe.html</loc>
    <lastmod>2026-01-07</lastmod>
  </url>
  ▼<url>
    <loc>https://stahe.github.io/word-odt-vers-html-janv-2026/construction-du-site-html.html</loc>
    <lastmod>2026-01-07</lastmod>
  </url>
  ▼<url>
```

Le fichier [sitemap.xml] liste toutes les pages HTML du site.

Si tout va bien, vous obtenez l'écran suivant :



Google le traitera régulièrement pour chercher des modifications. Si un problème se pose à l'avenir, vous en serez averti.

Masquer

C'est tout. Vous devrez indexer les pages du site vous-même :

The screenshot shows the Google Search Console interface. On the left, there's a sidebar with links like 'Vue d'ensemble', 'Insights', 'Performances', 'Inspection de l'URL', 'Indexation' (with 'Pages', 'Sitemaps' highlighted with a red arrow labeled '1', and 'Suppressions'), and 'Expérience'. The main area has a search bar at the top with the URL 'https://stahe.github.io/word-odt-vers-html-janv-2026/introduction.html'. Below it is a section titled 'Ajouter un sitemap' with a text input field containing the same URL. Further down is a table titled 'Sitemaps envoyés' with one entry: 'sitemap.xml' (Type: Inconnu, Last Read: 7 janv. 2026, Status: Impossible de récupérer le sitemap). A red arrow labeled '2' points to the URL input field.

- En [2], tapez l'URL d'une des pages de votre site ;

Ici, le résultat est le suivant :

The screenshot shows the 'Inspection de l'URL' page. At the top, there's a button 'TESTER L'URL ACT'. Below it, a message box contains an information icon and the text 'Cette URL n'a pas été indexée par Google'. It explains that non-indexed pages can't be displayed on Google and provides a link 'En savoir plus'. At the bottom of the message box are two buttons: 'AFFICHER LA PAGE EXPLORÉE' and 'Page modifiée ? DEMANDER UNE INDEXATION'. A red arrow labeled '2' points to the 'DEMANDER UNE INDEXATION' button.

- En [1], Google dit que la page n'est pas encore indexée ;
- En [2], vous pouvez demander son indexation ;

Google Search Console vérifie alors que la page demandée peut être indexée. Si oui, on obtient le message suivant :

The screenshot shows a green success message box with a checkmark icon and the text 'Indexation demandée'. Below it is a note: 'Cette URL a été ajoutée à une file d'attente d'exploration prioritaire. Le fait d'envoyer une page à plusieurs reprises ne modifiera pas sa priorité ni sa position dans la file d'attente. [En savoir plus](#)'. At the bottom right is a blue 'Masquer' button.

Vous pouvez faire cela pour toutes les pages de votre site afin d'être sûr qu'il est bien indexé par le moteur de recherche de Google.

## 13. Conclusion

Nous avons montré dans ce document un convertisseur de documents ODT produits par [LibreOffice](#) en un site statique HTML de très bonne qualité. Pour les documents Word, nous avons expliqué qu'il fallait les ouvrir avec LibreOffice et les sauvegarder au format ODT.

Nous avons présenté dans le chapitre [Les exemples de ce document] les structures gérées correctement par la convertisseur Gemini. En dehors de ces structures, rien n'est garanti. Il est probable que le convertisseur montrera des anomalies. Il faudra alors de nouveau commencer un chat avec Gemini pour résoudre des anomalies.

Nous avons montré également comment exporter sur GitHub les sites HTML créés par le convertisseur (cf chapitre Héberger le site HTML sur GitHub)

Pour ma part, j'ai converti plusieurs documents ODT produits par LibreOffice, dont nombre d'entre-eux étaient à l'origine des documents Word :

- [\[Générer un script Python avec des outils d'IA \(2025\)\]](#) ;
- [\[Introduction au langage Python et au framework web Flask par l'exemple \(2020\)\]](#) ;
- [\[Introduction au langage PHP7 par l'exemple \(2019\)\]](#) ;
- [\[Introduction au framework VUE.JS par l'exemple \(2019\)\]](#) ;
- [\[Introduction au framework NUXT.JS par l'exemple \(2019\)\]](#) ;
- [\[Introduction au langage ECMASCIPT6 par l'exemple \(2019\)\]](#) ;
- [\[Introduction au langage VBScript \(2002\)\]](#) ;
- [\[Introduction au langage Java et à l'écosystème Spring au travers d'une étude de cas \(2015\)\]](#) ;
- [\[Apprentissage du langage C# Version 3.0 avec le Framework .NET 3.5 \(2008\)\]](#) ;

Ce sont pour la plupart des anciens cours de plusieurs centaines de pages. La taille du document ODT n'importe pas au convertisseur Gemini. Ce qui lui importe ce sont les structures que l'on y trouve (cf. chapitre : Les exemples de ce document).