Jenga2 – Utils

Le module **Utils** fournit un ensemble d'outils **transversaux** et **réutilisables** utilisés par toutes les autres parties de Jenga2 (Core, Commands, ...).
Ces utilitaires couvrent :

- L'affichage console **coloré** et **structuré**.
- Les opérations **système de fichiers** (copie, suppression, hash, recherche).
- L'exécution de **processus externes** avec capture.
- La génération de **rapports de build/test**.
- L'affichage de **barres de progression**, **tableaux**, **arbres**, **spinners**.

Sommaire

Philosophie

- **Ne pas réinventer la roue** : Utiliser les bibliothèques standard ou des wrappers légers.
- **Cohérence** : Tous les modules exposent des classes avec des méthodes statiques en `PascalCase`.
- **Ergonomie** : Fournir des raccourcis DSL (lowercase, one word) dans `__init__.py` pour les scripts utilisateur.
- **Robustesse** : Gérer proprement les cas d'erreur, les environnements Windows/Unix, l'absence de dépendances optionnelles.

Modules

1. `Colored` – Console colorée

Classe utilitaire pour produire du texte coloré et stylisé sur les terminaux compatibles ANSI.

Fonctionnalités :

rihen.universe@gmail.com • teuguiasederis@gmail.com
(+237) 693-761-773

- Détection automatique du support des couleurs.
- Forçage du support Windows via `EnableWindowsColor()`.
- Application de couleurs (foreground/background) et de styles (gras, italique, souligné, clignotant, ...).
- Impression directe avec `Print()`, `PrintError()`, `PrintSuccess()`, `PrintWarning()`, `PrintInfo()`.
- Suppression des codes ANSI (`StripColors`, `LenWithoutColors`).
- Formatage de tableaux (`FormatTable`) avec couleurs optionnelles.

Exemple :

```
from Jenga2.Utils import Colored

Colored.PrintSuccess("Build succeeded!", bold=True)
table = Colored.FormatTable(
    rows=[["1", "Alice"], ["2", "Bob"]],
    headers=["ID", "Name"],
    headerColors=["yellow", "cyan"]
)
print(table)
```

---

2. `FileSystem` – Système de fichiers

Wrapper autour de `pathlib` et `shutil` avec des opérations fréquentes en build.

Fonctionnalités :

- Test d'existence, type (fichier/répertoire).
- Chemins absolus/relatifs, normalisation, jointure.
- Création, suppression (fichier, répertoire vide, récursive).
- Copie, déplacement (fichier, répertoire).
- Listage de fichiers/répertoires avec `glob` récursif, filtres d'ignorance (hidden).
- Lecture/écriture texte et binaire.
- Métadonnées (taille, mtime).
- Hachage (MD5, SHA1, SHA256) de fichiers et de chaînes.
- Fichiers/répertoires temporaires.
- Recherche d'exécutable dans le PATH.
- Détection intelligente du fichier workspace (`FindWorkspaceEntry`).

Exemple :

```
from Jenga2.Utils import FileSystem

if FileSystem.PathExists("build/"):
    FileSystem.RemoveDirectory("build/", recursive=True)
```

```
hash = FileSystem.ComputeFileHash("main.cpp", "sha256")
files = FileSystem.ListFiles(".", "*.cpp", recursive=True)
```

---

3. `Process` – Processus externes

Exécution de commandes système avec contrôle fin.

Fonctionnalités :

- `ExecuteCommand()` : paramètres complets (cwd, env, timeout, capture stdout/stderr, shell, check).
- `Run()` : exécution simple, retourne le code de sortie.
- `Capture()` : exécute et retourne stdout (lève une exception si échec).
- `CaptureLines()` : pareil, retourne une liste de lignes.
- `RunBackground()` : lance un processus en arrière-plan (objet `Popen`).
- `Which()` : localise un exécutable.
- Manipulation des variables d'environnement (`SetEnvironmentVariable`, `GetEnvironmentVariable`, `UnsetEnvironmentVariable`).

Exemple :

```
from Jenga2.Utils import Process

result = Process.ExecuteCommand(["g++", "-c", "main.cpp"], captureOutput=True)
if result.returnCode == 0:
    print(result.stdout)

lines = Process.CaptureLines(["git", "status", "--porcelain"])
for line in lines:
    print(line)
```

---

4. `Reporter` – Rapports de build/test

Système de rapport structuré pour les builds et les tests.

Composants :

- `Report` : conteneur de base, sections, sérialisation JSON, export texte.
- `BuildReport` : spécialisé pour les builds (projets, succès/échec, temps).
- `TestReport` : spécialisé pour les tests unitaires (cas, résultats, durée).
- `Reporter` : classe statique pour logger avec verbosité, timing, sections.

Fonctionnalités :

- Ajout de sections (`AddSection`).
- Sauvegarde/chargement JSON.
- Génération de rapports texte avec couleurs.

- Export JUnit XML (`ExportJUnitXml`) pour intégration CI.

Exemple :

```
from Jenga2.Utils import Reporter, CreateTestReport

report = CreateTestReport()
report.AddTestCase("test_addition", "pass", 0.001)
report.Print(colored=True)

Reporter.Section("Building...")
Reporter.Success("All good.")
```

---

5. `Display` – Affichage structuré

Outils pour une console moderne et interactive.

Fonctionnalités :

- `PrintTree()` : affichage d'arbres avec connecteurs.
- `ProgressBar` : barre de progression avec pourcentage, temps écoulé.
- `Spinner` : indicateur d'activité indéterminé.
- `PrintTable()` : tableau formaté (wrapper de `Colored.FormatTable`).
- `PrintHeader()`, `PrintSeparator()` : décorations.
- Raccourcis `Success()`, `Error()`, `Warning()`, `Info()`, `Section()`, `Subsection()`, `Detail()`, `Debug()` – avec émoticônes et couleurs.

Exemple :

```
from Jenga2.Utils import Display, ProgressBar

pb = ProgressBar(total=100, prefix="Compiling")
for i in range(100):
    pb.Update(1, f"file_{i}.cpp")
pb.Finish()

Display.Section("Results")
Display.Success("Tests passed: 42")
```

📐 Conventions de nommage

Le module `Utils` suit les mêmes conventions que le reste de Jenga2 :

Élément	Convention	Exemple

Élément	Convention	Exemple
Classes	PascalCase	`Colored`, `FileSystem`
Méthodes publiques (statiques)	PascalCase	`PrintSuccess`, `CopyFile`
Méthodes privées	_PascalCase	`_StripAnsiCodes`
Attributs privés	_camelCase	`_supportsColor`
Constantes	UPPER_SNAKE_CASE	`_CACHE_ROOT`

Raccourcis DSL (lowercase, one word) :
Dans `__init__.py`, des alias sont exposés pour les scripts utilisateur :

```
printcolor = Colored.Print
printerror = Colored.PrintError
printsuccess = Colored.PrintSuccess
printwarning = Colored.PrintWarning
printinfo = Colored.PrintInfo
```

Ces alias sont utilisables sans importer les classes.

Dépendances

Le module `Utils` est **autonome** (ne dépend d'aucun autre module Jenga2).
Il peut être importé par `Core`, `Commands`, etc.

Dépendances externes optionnelles :

- `watchdog` : pour `FileWatcher` (dans `Core`, pas dans `Utils`).
- `colorama` : automatiquement utilisé sur Windows si présent.

Bonnes pratiques

1. **Toujours utiliser `FileSystem` pour les opérations de fichiers** – ne pas appeler directement `shutil` ou `os`.
2. **Pour les logs utilisateur** : préférer `Reporter` ou `Display` plutôt que `print()` direct.
3. **Capturer les sorties de processus** : utiliser `Process.ExecuteCommand(captureOutput=True)` plutôt que `subprocess.run()`.
4. **Vérifier la disponibilité du terminal** : `Colored.SupportsColor()` avant d'utiliser des codes ANSI.
5. **Éviter les prints dans les composants internes** : utiliser `Reporter.Detail` ou `Display.Detail` avec le flag `verbose`.

Exemples d'utilisation

Nettoyage de dossier

```
from Jenga2.Utils import FileSystem, Colored

build_dir = Path("Build")
if build_dir.exists():
    FileSystem.RemoveDirectory(build_dir, recursive=True)
    Colored.PrintSuccess(f"Removed {build_dir}")
```

Exécution d'une commande avec timeout

```
try:
    out = Process.Capture(["make", "-j4"], timeout=60)
except TimeoutError:
    Colored.PrintError("Build timed out")
```

Rapport de build minimal

```
from Jenga2.Utils import Reporter, BuildReport

report = BuildReport()
report.AddProjectResult("core", success=True, duration=2.5)
report.AddProjectResult("game", success=False, duration=1.2, errors=["Link
error"])
report.Print()
```

Barre de progression

```
from Jenga2.Utils import Display

pb = Display.ProgressBar(total=10)
for i in range(10):
    time.sleep(0.1)
    pb.Update(1)
pb.Finish()
```

🐞 Dépannage

Problème	Cause probable	Solution
Les couleurs ne s'affichent pas sur Windows	Terminal non compatible VT	Appeler `Colored.EnableWindowsColor()` ou utiliser `colorama`

Problème	Cause probable	Solution
`FindWorkspaceEntry` retourne `None`	Aucun fichier `.jenga` avec workspace	Vérifier que le fichier contient `with workspace(`
`Process.ExecuteCommand` lève une exception	Commande introuvable ou timeout	Vérifier le PATH, augmenter le timeout
`Reporter.verbose` ne fait rien	La variable de classe n'est pas activée	`Reporter.verbose = True` avant d'appeler les commandes

🔗 Liens connexes

- Documentation Core – utilise intensivement `Utils`
- Documentation Commands – utilise `Utils` pour l'UI
- Guide utilisateur

Ce document est maintenu par l'équipe Jenga. Toute contribution est la bienvenue.