Utilisation de Maven Lite et description de toutes les fonctionnalités

- See the English documentation
- Voir la documentation en PDF

Table des matières

Liste des options

- `-f`, `--file` : Permet de charger les options à partir d'un fichier de configuration.
- `-cr`, `--create` : Crée l'arborescence du projet ainsi qu'un fichier de configuration par défaut.
- `-mvc`, `--model-view-controller` : Spécifie à l'option '`--create`' de créer l'arborescence d'un projet MVC.
- `-c`, `--compilation` : Compile le projet.
- `-l`, `--launch` : Lance le projet.
- `-cl`, `--compile-launch` : Compile et lance le projet (équivalent à -c -l).
- `-lc`, `--launch-compile` : Compile et lance le projet (équivalent à -c -l).

- `-q`, `--quiet` : Supprime l'affichage de java dans le terminal lors de l'exécution du projet.
- `-v`, `--verbose` : Affiche les commandes exécutées.
- `-ex`, `--exclude` : Exclut des fichiers java et des dossiers de la compilation.
- `-cj`, `--compile-jar` : Crée un fichier jar du projet.
- `-lj`, `--launch-jar` : Lance un fichier jar exécutable.
- `-it`, `--integrate-test` : Intègre les tests unitaires au projet.
- `-s`, `--source` : Dossier contenant les fichiers java à compiler.
- `-t`, `--target` : Dossier de sortie des fichiers compilés.
- `-r`, `--resources` : Dossier contenant les fichiers ressources à copier dans le dossier de sortie des fichiers compilés.
- `-cp`, `--classpath` : Spécifie le classpath à utiliser lors de la compilation et du lancement.
- `-lib`, `--libraries` : Dossier contenant les fichiers jar utilisés par le programme.
- `-args`, `--arguments` : Tous les arguments à passer à la classe principale.
- `-m`, `--main` : Classe principale à lancer.
- `-e`, `--encoding` : Change l'encodage des fichiers java à compiler.
- `-exp`, `--export` : Crée un fichier jar exécutable pour lancer le projet sans installer MavenLite.
- `-mvn`, `--maven` : Convertit le projet en projet Maven.
- `-V`, `--version` : Affiche la version.
- `-h`, `--help` : Affiche l'aide et quitte.
- `-clr`, `--clear` : Supprime les fichiers dans le dossier de sortie des fichiers compilés.

Utilisation de base

- placer vous dans le dossier de votre projet java et exécuter la commande suivante :

```
mvnl [options] [arguments]
```

Ligne de commande

La ligne de commande est la méthode classique pour utiliser Maven Lite bien qu'elle soit moins pratique que l'utilisation d'un fichier de configuration.

Vous pouvez mettre autant d'options que vous le souhaitez et dans n'importe quel ordre.

il faut juste mettre les arguments avec des espaces entre guillemets, par exemple `-args "mon argument"`. sous windows il est impossible de mettre le caratère `"` dans les arguments.

- Elle ne prend pas en charge les commentaires.
- Il est possible de mettre des options en utilisant le format `--option` ou `-o`. Vous pouvez mettre n'inporte quel option cité dans la liste des options.
- Il est possible de passer des arguments avec des espaces à la class main de votre projet en utilisant des guillemets double, par exemple `mvnl -args "votre argument"`.
- Il est possible de mettre plusieurs options sur la même ligne en les séparant par un espace, par exemple `mvnl -q --verbose --arguments "votre arguments"` ou `mvnl -quiet -v -args "votre arguments"`.
- Il est possible d'échapper les caractères spéciaux avec un antislash `\`, par exemple `-args "exemp\"le"` sous Linux et Macos uniquement.

Exemple de ligne de commande

- Exemple d'une ligne de commande avec des arguments
    - Dans cette exemple, nous allons compiler et lancer un projet java avec comme source le dossier `src/main/java`, comme dossier de sortie des fichiers compilés le dossier `target`, comme dossier contenant les fichiers jar le dossier `src/main/resources/lib` et l'affichage des commandes éxécutées.

```
mvnl --source src/main/java --target target --libraries src/main/resources/lib --
verbose -cl
```

Fichier de configuration

L'utilisation d'un fichier de configuration se fait avec l'option `--file` ou `-f`.

Le fichier de configuration est un fichier unique à chaque projet qui permet de configurer les options que Maven Lite devra utiliser.

Le nom par défaut du fichier de configuration est `LPOM.conf` et doit être à la racine du projet. Il est possible de le renommer mais dans ce cas il faudra préciser son nom l'ors de l'utilisation de l'option, par exemple `mvnl -f monFichier.extention`.

si vous voulez ajouté votre CLASSPATH système au CLASSPATH de Maven Lite dans le fichier de configuration, il faut utiliser le terme `$CLASSPATH` en majuscule.

Il est possible de mettre autant d'options que vous le souhaitez et dans n'importe quel ordre.

- Il prend en charge les commentaires en utilisant le caractère `#` au début de la ligne.
- Il est possible de mettre des options en utilisant le format `--option` ou `-o`. Vous pouvez mettre n'inporte quel option cité dans la liste des options sauf l'option `--file` et `-f`.
- Il est possible de passer des arguments avec des espaces à la class main de votre projet en utilisant des guillemets double, par exemple `-args "votre argument"`.
- Il est possible de mettre plusieurs options sur la même ligne en les séparant par un espace, par exemple `--quiet -v --arguments "votre arguments"` ou `-q --verbose -args "votre arguments"`.
- Il est possible d'échapper les caractères spéciaux avec un antislash `\`
    - Les caractères spéciaux dans le fichier de configuration sont : `\` et `"`. Voici des exemples d'utilisation :
        - `-args "exemp\"le"` devient `exemp"le`
        - `-args "exemp\\\\\"le` devient `exemp\"le`
        - `-args "\-exemple"` devient `\-exemple`
        - `-args "\--exemple"` devient `\--exemple`
        - `-args "exemp\\le"` devient `exemp\le`
        - `-args "exemp\le"` devient `exemp\le`
        - `-args "exemp\\\le"` devient `exemp\\le`
        - `-args "exemp\\\"le"` devient `exemp\"le`
        - `-args "exemp#le"` devient `exemp#le`
        - `-args "exemp\ le"` devient `exemp\ le`

Exemple de fichier de configuration

- Un exemple type de fichier de configuration est le suivant :

```
# Source du projet
--source src/main/java

# Dossier de sortie des fichiers compilés
--target target

# Liste des libraries à ajouter au classpath
--libraries src/main/resources/lib

# Affiche les commandes éxécutées
--verbose

# Ajoute le classpath système au classpath de Maven Lite
--classpath CLASSPATH

# Ajoute un argument à la classe principale
--arguments "argument 2 (dans config)"
-args "argument 3 (dans config)"

# Supprime l'affichage de java et affiche les commandes éxécutées
--quiet --verbose
```

Création d'un projet

L'utilisation d'un fichier de configuration se fait avec l'option `--create` ou `-cr`.

Il est impossible de créer un projet avec un espace dans le nom. Il est impossible de créer un projet avec un nom qui est déjà utilisé par un fichier ou un dossier.

Vous pouvez créer un projet dans le dossier courant grâce à la commande suivante :

```
mvnl --create ./
```

Par défaut, le nom du projet est `NewProject` et l'arborescence est la suivante

```
NewProject
├── LPOM.conf
├── src
│   ├── main
│   │   └── java
│   │       └── HelloWorld.java
│   └── resources
│       └── lib
└── target
```

Modèle Vue Contrôleur

L'utilisation de cette option se fait avec l'option `--model-view-controller` ou `-mvc`.

Permet de spécifier à l'option create de créer l'arborescence d'un projet MVC.

- Pour créer un projet MVC, vous pouvez utiliser la commande suivante

  ```
  mvnl --create NomProjet --model-view-controller
  ```

  - Arborescence du projet qui

    ```
    NomProjet
    ├── LPOM.conf
    ├── src
    │   ├── main
    │   │   └── java
    │   │       ├── controller
    │   │       │   └── Controller.java
    │   │       ├── model
    │   │       └── view
    │   └── resources
    │       └── lib
    └── target
    ```

Compilation d'un projet

L'utilisation de cette option se fait avec l'option `--compilation` ou `-c` ou `--compile-launch` ou `-cl` ou `--launch-compile` ou `-lc`.

Cette option permet de compiler le projet.

- Exemple d'utilisation de l'option Compilation

  - Dans cette exemple, nous allons compiler un projet java avec la source et le dossier de sortie des fichiers compilés par défaut.

```
mvnl -c
```

Lancement d'un projet

L'utilisation de cette option se fait avec l'option `--launch` ou `-l` ou `--compile-launch` ou `-cl` ou `--launch-compile` ou `-lc`.

Cette option permet de lancer le projet.

- Exemple d'utilisation de l'option Lancement
    - Dans cette exemple, nous allons lancer un projet java avec la source et le dossier de sortie des fichiers compilés par défaut.

```
mvnl -l
```

Silencieux

L'utilisation de cette option se fait avec l'option `--quiet` ou `-q`.

Cette option permet de supprimer l'affichage de java dans le terminal lors de l'exécution du projet.

- Exemple d'utilisation de l'option Silencieux
    - Dans cette exemple, nous allons compiler et lancer un projet java avec l'affichage de java dans le terminal désactivé ainsi que le dossier source et le dossier de sortie des fichiers compilés par défaut.

```
mvnl -q -cl
```

Verbeuse

L'utilisation de cette option se fait avec l'option `--verbose` ou `-v`.

Cette option permet d'afficher les commandes Java exécutées par Maven Lite.

- Exemple d'utilisation de l'option Verbeuse
    - Dans cette exemple, nous allons compiler et lancer un projet java avec l'affichage des commandes Java exécutées par Maven Lite ainsi que le dossier source et le dossier de sortie des fichiers compilés par défaut.

```
mvnl -v -cl
```

Exclusion de fichiers et de dossiers

L'utilisation de cette option se fait avec l'option `--exclude` ou `-ex`.

Cette option permet d'exclure des fichiers java et des dossiers de la compilation.

Si vous voulez exclure un dossier, il faut mettre le chemin relatif du dossier à partir du dossier source du projet et sans utiliser `./` ou `../`.

Si vous excluez un dossier, tous les fichiers et dossiers qu'il contient seront aussi exclus.

Attention, si vous excluez le dossier `test`, tout les fichiers ou dossiers qui contiennent le mot `test` dans leur nom seront aussi exclus.

- Exemple d'utilisation de l'option Exclusion

    - Dans cette exemple, nous allons compiler et lancer un projet java avec l'exclusion du fichier `Main2.java` et de tout les fichiers et dossier que contient le dossier `tests` ainsi que le dossier source et le dossier de sortie des fichiers compilés par défaut.

```
mvnl -ex Main2.java tests -cl
```

Compilation en fichier jar

L'utilisation de cette option se fait avec l'option `--compile-jar` ou `-cj`.

Cette option permet de créer un fichier jar exécutable de votre projet.

Les conventions de nommage des fichiers jar sont les suivantes : `<nom>-<M>.<m>.<b>.jar`.

- `<nom>` : Nom du projet.
- `<M>` : Numéro de la version majeur. Il commence à 1 et est incrémenté à chaque nouvelle version non compatible avec la précédente et avec de grosses modifications.
- `<m>` : Numéro de la version mineur. Il commence à 0 et est incrémenté à chaque nouvelle version compatible avec la précédente et avec de petites modifications et retourne à 0 à chaque changement de version majeur.
- `<b>` : Numéro de la version de build. Il commence à 0 et est incrémenté à chaque correction de bug et retourne à 0 à chaque changement de version mineur ou majeur.
- Exemple d'utilisation de l'option Compilation en fichier jar
    - Dans cette exemple, nous allons créer un fichier jar du projet java avec le dossier de sortie des fichiers compilés par défaut.

```
mvnl -cj
```

Lancement d'un fichier jar

L'utilisation de cette option se fait avec l'option `--launch-jar` ou `-lj`.

Cette option permet de lancer le fichier jar exécutable de votre projet.

- Exemple d'utilisation de l'option Lancement d'un fichier jar
    - Dans cette exemple, nous allons lancer le fichier jar exécutable `MonProjet-1.0.0.jar` qui se trouve dans le dossier `target`.

```
mvnl -lj target/MonProjet-1.0.0.jar
```

Intégration des tests unitaires

L'utilisation de cette option se fait avec l'option `--integrate-test` ou `-it`.

Cette option permet d'intégrer les tests unitaires au projet, c'est à dire de créer l'arborescence des tests unitaires ainsi que les fichiers de test unitaire et copie également les fichiers jar utilisés par JUnit dans le dossier des librairies.

Si vous l'utilisez en même temps que l'option `-cr` ou `--create`, Maven Lite créera l'arborescence des tests unitaires ainsi qu'un fichier de test par défaut qui contient un test unitaire de la classe principale.

Si vous l'utilisez après avoir créé le projet, Maven Lite créera l'arborescence des tests unitaires ainsi qu'un fichier de test unitaire pour chaque fichier java du projet.

- Exemple d'utilisation de l'option Intégration des tests unitaires

    - Dans cette exemple, nous allons créer un projet java avec l'intégration des tests unitaires.

```
mvnl -cr -it
```

    - Arborescence du projet

```
NewProject
├── LPOM.conf
├── src
│   ├── main
│   │   └── java
│   │       └── HelloWorld.java
│   ├── resources
│   │   └── lib
│   │       ├── hamcrest-core-1.3.jar
│   │       └── junit-4.13.2.jar
│   └── test
│       └── java
│           └── HelloWorldTest.java
└── target
```

- Exemple d'utilisation de l'option Intégration des tests unitaires avec un projet MVC

    - Dans cette exemple, nous allons créer un projet java MVC avec l'intégration des tests unitaires.

```
mvnl -cr -mvc -it
```

    - Arborescence du projet

```
NewProject
├── LPOM.conf
├── src
│   ├── main
│   │   └── java
│   │       ├── controller
│   │       │   └── Controller.java
│   │       ├── model
│   │       └── view
│   ├── resources
│   │   └── lib
│   │       ├── hamcrest-core-1.3.jar
│   │       └── junit-4.13.2.jar
│   └── test
│       └── java
│           ├── controller
│           │   └── ControllerTest.java
│           ├── model
│           └── view
└── target
```

Source du projet

L'utilisation de cette option se fait avec l'option `--source` ou `-s`.

Cette option permet de spécifier le dossier contenant les fichiers java à compiler.

Le dossier source n'est pas la racine du projet mais le premier dossier contenant les fichiers java à compiler. Les fichiers java qui sont dans se dossier n'ont pas de package.

- Exemple d'utilisation de l'option Source du projet

- Dans cette exemple, nous allons compiler et lancer un projet java avec `src` comme dossier source et le dossier de sortie des fichiers compilés par défaut.

```
mvnl -s src -cl
```

Destination des fichiers compilés

L'utilisation de cette option se fait avec l'option `--target` ou `-t`.

Cette option permet de spécifier le dossier de sortie des fichiers compilés, le dossier d'entrée des fichiers à lancer, le dossier de sortie des fichiers jar ainsi que le dossier d'entrée des fichiers .class à mettre dans le jar.

Vous n'avez pas besoin de créer le dossier de sortie des fichiers compilés, Maven Lite le fera pour vous.

Vous n'avez pas besoin d'ajouter le dossier de sortie des fichiers compilés au classpath, Maven Lite le fera pour vous.

- Exemple d'utilisation de l'option Destination des fichiers compilés
  - Dans cette exemple, nous allons compiler et lancer un projet java avec `src` comme dossier source et `target` comme dossier de sortie des fichiers compilés.

```
mvnl -s src -t target -cl
```

- Exemple d'utilisation de l'option target pour créer un fichier jar
  - Dans cette exemple, nous allons créer un fichier jar du projet java avec `target` comme dossier de sortie des fichiers compilés. Vous pourrez trouver le fichier jar dans le dossier `target` ainsi que lancer le fichier jar avec la commande `mvnl -lj target/MonProjet-1.0.0.jar`.

```
mvnl -t target -cj
```

Ressources

L'utilisation de cette option se fait avec l'option `--resources` ou `-r`.

Cette option permet de spécifier le dossier contenant les fichiers ressources à copier dans le dossier de sortie des fichiers compilés lors de la création d'un fichier jar.

- Exemple d'utilisation de l'option Ressources
  - Dans cette exemple, nous allons créer un fichier jar du projet java avec `src/main/resources` comme dossier contenant les fichiers ressources à copier dans le dossier de sortie des fichiers compilés.

```
mvnl -r src/main/resources -cj
```

Classpath

L'utilisation de cette option se fait avec l'option `--classpath` ou `-cp`.

Cette option permet de spécifier le classpath à utiliser lors de la compilation et du lancement.

Vous pouvez ajouter le classpath système en utilisant le terme `$CLASSPATH` en majuscule dans le fichier de configuration.

Dans la ligne de commande, vous pouvez utiliser le classpath système en utilisant le terme `$CLASSPATH` sous Linux et Macos. Sous Windows, vous pouvez utiliser le terme `%CLASSPATH%`.

- Exemple d'utilisation de l'option Classpath

- Dans cette exemple, nous allons compiler et lancer un projet java avec le classpath système, le dossier `src/main/resources/lib` et le dossier `target` dans le classpath du projet.

```
mvnl -cp $CLASSPATH target src/main/resources/lib -cl
```

Libraries

L'utilisation de cette option se fait avec l'option `--libraries` ou `-lib`.

Cette option permet de spécifier le dossier contenant les fichiers jar utiliser par le programme. Tout les fichiers jar seront ajoutés au classpath lors de la compilation et du lancement.

Vous pouvez créer des sous-dossiers dans le dossier des librairies pour mieux organiser vos fichiers jar, Maven Lite les prendra en compte.

- Exemple d'utilisation de l'option Libraries
  - Dans cette exemple, nous allons compiler et lancer un projet java avec le dossier `src/main/resources/lib` comme dossier contenant les fichiers jar utilisés par le programme.

```
mvnl -lib src/main/resources/lib -cl
```

Arguments

L'utilisation de cette option se fait avec l'option `--arguments` ou `-args`.

Cette option permet de spécifier tous les arguments à passer à la classe principale de votre projet.

Ces arguments seront passés à la classe principale dans l'ordre où ils sont passés à Maven Lite.

Attention, sous Windows, il est impossible de mettre le caractère `"` dans les arguments passés dans la ligne de commande, il devra donc être passé dans le fichier de configuration.

Exemple d'utilisation de l'option Arguments

- Exemple d'une ligne de commande avec des arguments et un fichier de configuration
  - Dans cette exemple, nous allons compiler et lancer un projet java avec des arguments en utilisant la ligne de commande et un fichier de configuration.

```
mvnl -args "argument 1" -f --arguments "argument 3" "argument 4" -args -cl
```

  - Fichier de configuration

```
# Ajoute un argument à la classe principale
-args "argument 2 (dans config)"
```

  - Arguments passés à la classe principale

```
String args = new String[]{"rgument 1", "argument 2 (dans config)", "argument
3", "argument 4"};
```

Arguments dans la ligne de commande

Ligne de commande sous linux et Macos

- pour passer des arguments à la class main de votre projet avec l'option `-args` et `--arguments` pour éviter tous problèmes d'échappement.
    - Les caractères spéciaux dans la ligne de commande sont : `\`, `"`
        - `-args '"exemp\"le"'` devient `exemp"le`
        - `-args '"-exemple"'` devient `-exemple`
        - `-args '"--exemple"'` devient `--exemple`
        - `-args '"exemp\\le"'` devient `exemp\le`
        - `-args '"exemp\\\"le"'` devient `exemp\"le`.

Ligne de commande sous windows

// TODO : vérifier que les caractères spéciaux fonctionnent sous windows

- pour passer des arguments à la class main de votre projet avec l'option `-args` et `--arguments` pour éviter tous problèmes d'échappement.
    - Les caractères spéciaux dans la ligne de commande sont : `\`
        - `-args '"exemp\\\"le"'` devient `exemp\"le`
        - `-args '"-exemple"'` devient `-exemple`
        - `-args '"--exemple"'` devient `--exemple`
        - `-args '"exemp\\le"'` devient `exemp\le`
        - `-args '"exemp\le"'` devient `exemp\le`
        - `-args '"exemp\\\le"'` devient `exemp\\le`
        - `-args '"exemp\\\\\"le"'` devient `exemp\"le`
        - `-args '"exemp#le"'` devient `exemp#le`
        - `-args '"exemp\ le"'` devient `exemp\ le`

Arguments dans le fichier de configuration

// TODO : vérifier les caractères spéciaux

- pour passer des arguments à la class main de votre projet avec l'option `-args` et `--arguments` pour éviter tous problèmes d'échappement.
    - Les caractères spéciaux dans le fichier de configuration sont : `\`, `"`
        - `-args "exemp\"le"` devient `exemp"le`
        - `-args "\-exemple"` devient `\-exemple`
        - `-args "\--exemple"` devient `\--exemple`
        - `-args "exemp\\le"` devient `exemp\le`
        - `-args "exemp\le"` devient `exemp\le`
        - `-args "exemp\\\le"` devient `exemp\\le`
        - `-args "exemp\\\"le"` devient `exemp\"le`
        - `-args "exemp#le"` devient `exemp#le`
        - `-args "exemp\ le"` devient `exemp\ le`

Class main

L'utilisation de cette option se fait avec l'option `--main` ou `-m`.

Cette option permet de spécifier la classe principale à lancer avec le package sous la forme `package.MainClass`. Cette option est utile uniquement si vous avez plusieurs classes main dans votre projet, si se n'est pas le cas, Maven Lite trouvera et utilisera automatiquement la classe main du projet.

- Exemple d'utilisation de l'option Class main
    - Dans cette exemple, nous allons compiler et lancer un projet java avec la classe main `Main2` du package `com.example` et le dossier de sortie des fichiers compilés par défaut.

```
mvnl -m com.example.Main2 -cl
```

Encodage

L'utilisation de cette option se fait avec l'option `--encoding` ou `-e`.

Cette option permet de spécifier l'encodage des fichiers java à compiler.

- Exemple d'utilisation de l'option Encodage
    - Dans cette exemple, nous allons compiler et lancer un projet java avec l'encodage `ANSI` et le dossier de sortie des fichiers compilés par défaut.

```
mvnl -e ANSI -cl
```

Exportation

L'utilisation de cette option se fait avec l'option `--export` ou `-exp`.

Cette option permet de créer un fichier .class exécutable paramétrer pour votre projet permettant de compiler et lancer votre projet sans avoir installer MavenLite.

Les seuls options qu'accepte le fichier .class sont les options de la class main, `--arguments`, `-args`, les options de lancement et de compilation, `--launch`, `-l`, `--compile`, `-c`, `--compile-launch`, `-cl`, `--launch-compile`, `-lc` et l'option du fichier de config `--file`, `-f` et c'est tout. Les autres options ne seront pas prises en compte.

Il est important d'utiliser le fichier de configuration lors de l'utilisation de l'option `--export` ou `-exp` car il permet de spécifier les paramètres qu'utilisera le fichier .class exécutable.

Le but de cette option est de pouvoir lancer votre projet via le lancement d'un seul fichier sans argument et sans avoir à installer Maven Lite. Cela permet notamment de permettre à des personnes qui n'ont pas Maven Lite d'utiliser votre projet.

- Exemple d'utilisation de l'option Exportation
    - Dans cette exemple, nous allons créer un fichier .class exécutable paramétrer pour notre projet permettant de compiler et lancer notre projet sans avoir installer MavenLite.

```
mvnl -f -exp
```

    - Fichier de configuration

```
# Source
--source src/main/java

# Dossier de sortie des fichiers compilés
--target target

# Liste des libraries à ajouter au classpath
--libraries src/main/resources/lib

# Affiche les commandes éxécutées
--verbose
```

Transformation en projet Maven

L'utilisation de cette option se fait avec l'option `--maven` ou `-mvn`.

Cette option permet de convertir votre projet en projet Maven en créant un fichier pom.xml et en déplaçant les fichiers si nécessaire.

Version

L'utilisation de cette option se fait avec l'option `--version` ou `-V`.

Cette option permet d'afficher la version de Maven Lite ainsi que la localisation du fichier principale de Maven Lite, la version de java, le type de build, le runtime java utilisé, la langue utilisée par le système, la plateforme d'encodage utilisée par Maven Lite, le nom du système d'exploitation, la version du noyaux du système d'exploitation et l'architecture du système.

- Exemple

```
Maven Lite 2.0.0
Maven Lite home : /usr/local/etc/maven-lite/
Java version : 17.0.9, vendor : Private Build, runtime : OpenJDK Runtime Environment
Default locale : fr_FR, platform encoding : UTF-8
OS name : "Linux", version : "6.5.0-14-generic", architecture : "amd64"
```

Aide

L'utilisation de cette option se fait avec l'option `--help` ou `-h`.

Cette option permet d'afficher la liste des options ainsi que leur description, le nombre d'arguments qu'elles prennent et leur valeur par défaut si elle en ont une ainsi que le lien vers la documentation.

Supprimer les fichiers compilés

L'utilisation de cette option se fait avec l'option `--clear` ou `-clr`.

Cette option permet de supprimer tous les fichiers dans le dossier de sortie des fichiers compilés. Cette option permet de libérer de l'espace disque, ainsi que de faire un un fichier jar propre.

Exemple, fonctionnaliés et limites

Retour