

Objectifs du TP

Préambule:

Dans un projet Java, il arrive fréquemment que vous ayez à intégrer des bibliothèques externes. Vous ne pouvez pas tout développer, et ce n'est d'ailleurs pas une bonne pratique de le faire.

Définition:

Le lien entre votre projet et une bibliothèque s'appelle une dépendance.

Gestion des dépendances:

Vous avez plusieurs manières de gérer ces **dépendances**:

- manuellement en téléchargeant la bibliothèque et en l'intégrant dans STS
- en utilisant un outil comme MAVEN qui va régler un certain nombre de problèmes, notamment de dépendance transitive, comme on va le voir dans ce TP.

Contenu du TP:

Dans ce TP, nous allons bâtir un petit projet sans aucun outil de gestion de dépendance, à l'ancienne !!!

Etapes du TP

Dans ce TP nous allons apprendre à utiliser une librairie de Logging.

Ce genre de librairie est utilisée sur tous les projets. Elles permettent de remplacer le fameux ***System.out.println*** dont l'utilisation est proscrite.


ETAPES:

- 1) Créez un projet Java dans STS (Eclipse)
 - * Clic droit >> New >> Java project
 - * Nommez votre projet comme vous le souhaitez
- 2) Créez un package **fr.diginamic**
- 3) Créer une classe **exécutable TestLogger** dans ce package
- 4) Ouvrez votre navigateur et allez sur Google
 - a) Recherchez la librairie suivante: **slf4j-api-1.7.18.jar**
 - b) Normalement vous devriez obtenir le lien suivant:
<https://mvnrepository.com/artifact/org.slf4j/slf4j-api/1.7.18>
 - c) Cliquez dessus, vous arrivez sur **MAVEN Central Repository** qui est un énorme dépôt de librairies:

Home » org.slf4j » slf4j-api » 1.7.18

Note: There is a new version for this artifact

New Version	1.8.0-alpha2
-------------	--------------



SLF4J API Module » 1.7.18
The slf4j API

License	MIT
Categories	Logging Frameworks
HomePage	http://www.slf4j.org
Date	(Feb 26, 2016)
Files	Download (JAR) (54 KB)
Repositories	Central Sonatype Releases
Used By	23,856 artifacts

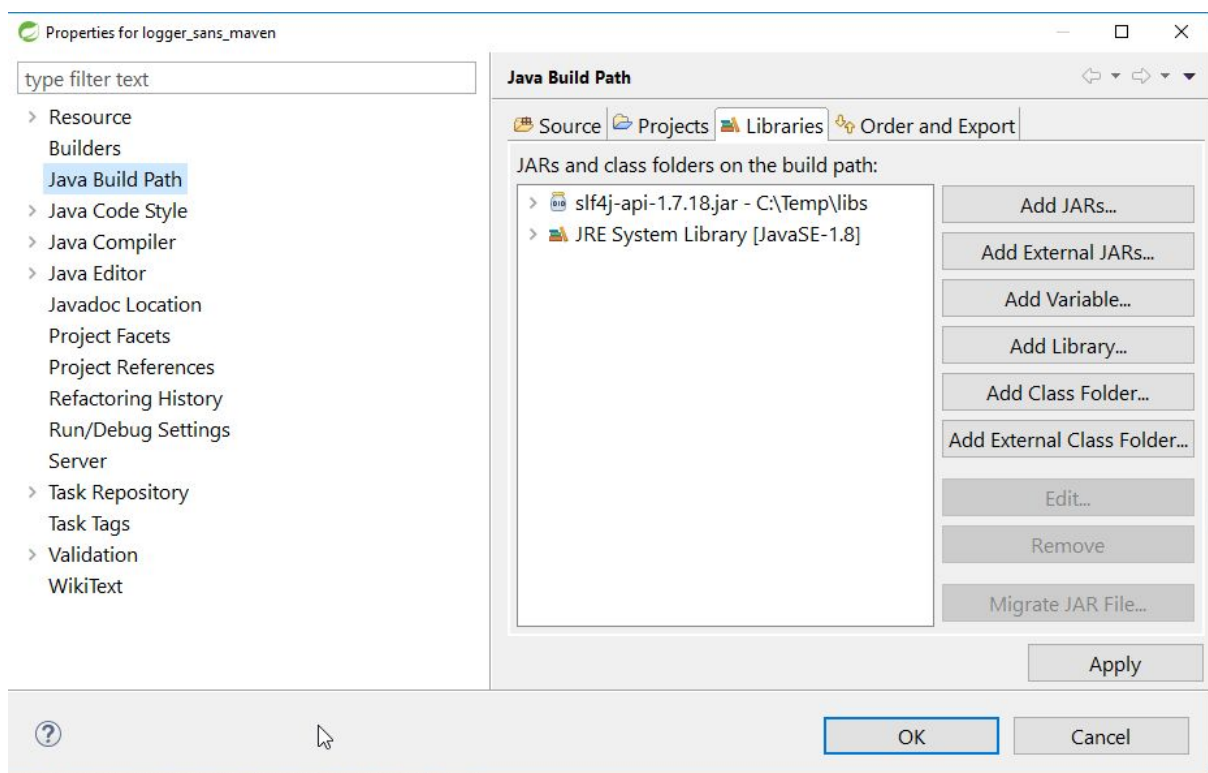
[Maven](#) [Gradle](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

```
<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api -->
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.7.18</version>
</dependency>
```

d) Dans la page vous avez un bouton “Download (JAR)” assez discret:

License	MIT
Categories	Logging Frameworks
HomePage	http://www.slf4j.org
Date	(Feb 26, 2016)
Files	Download (JAR) (54 KB)
Repositories	Central Sonatype Releases
Used By	23,856 artifacts

- e) Cliquez sur ce bouton et téléchargez la librairie sur votre PC
- f) Une fois téléchargée, allez dans Eclipse pour l’intégrer à votre projet.
- g) Faites un clic droit sur votre projet et sélectionnez **properties**
- h) La fenêtre ci-dessous s’ouvre :



- i) Sélectionnez **Java Build Path** dans le menu de gauche
- j) Dans la partie droite, cliquez sur l’onglet “**Libraries**”
- k) Ensuite cliquez sur le bouton **Add external JARS...**
- l) Parcourez votre disque dur afin de sélectionner la librairie que vous avez téléchargé puis cliquez sur Ouvrir

m) Cette action va avoir pour effet d'ajouter cette librairie à votre classpath, c'est-à-dire à la liste des ressources utilisables (classes, fichiers de conf, etc.) dans votre projet

5) Modifiez votre classe **TestLogger** comme suit et testez là:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class TestLogger {

    private static final Logger LOG = LoggerFactory.getLogger(TestLogger.class);

    public static void main(String[] args) {
        LOG.info("Bonjour Slf4J !");
        LOG.info("Implémentation Logback");
    }
}
```

6) Normalement à ce niveau, votre logger ne fonctionne pas encore car il manque des librairies. Vous devez obtenir le message d'erreur suivant:

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
```

C'est normal car la librairie que vous avons ajouté est l'API (qui contient les interfaces) de Logging SLF4J. Nous allons devoir ajouter une librairie d'implémentation (qui contient les classes)...

Passons à l'étape suivante...

7) Nous allons intégrer cette implémentation de SLF4J.

Répétez l'étape 4 avec la librairie suivante: **logback-classic.1.2.2.jar**

8) Tester désormais le code précédent. Qu'obtenez-vous ?

Normalement vous obtenez encore une erreur, une `ClassNotFoundException` qui indique qu'au moins une classe est manquante:

```
Failed to instantiate SLF4J LoggerFactory
Reported exception:
java.lang.NoClassDefFoundError: ch/qos/logback/core/joran/spi/JoranException
    at org.slf4j.LoggerFactory.bind(LoggerFactory.java:147)
    at org.slf4j.LoggerFactory.performInitialization(LoggerFactory.java:122)
    at org.slf4j.LoggerFactory.getILoggerFactory(LoggerFactory.java:378)
    at org.slf4j.LoggerFactory.getLogger(LoggerFactory.java:328)
    at org.slf4j.LoggerFactory.getLogger(LoggerFactory.java:349)
    at sans_maven.TestLogger.<clinit>(TestLogger.java:8)
```

Là encore c'est normal car la librairie d'implémentation a encore besoin d'une autre librairie pour fonctionner. Elle a donc elle même une dépendance vers une autre librairie !!! **C'est ce qu'on appelle un problème de dépendance transitive.**

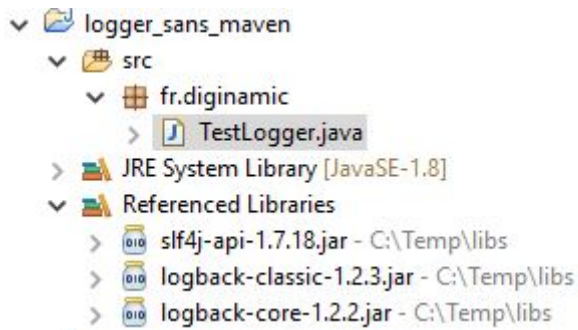
A ce stade si nous avons utilisé MAVEN nous n'aurions pas ce problème de dépendance transitive. En effet MAVEN est capable de régler ce genre de souci, comme vous le verrez par la suite.

Passons à la dernière étape.

7) Nous allons intégrer cette dépendance dont a besoin l'implémentation.

Répétez l'étape 4 avec la librairie suivante: **logback-core.1.2.2.jar**

8) Après cette étape, voici la structure du projet que vous devez obtenir:



Dans Referenced libraries vous voyez apparaître les 3 librairies que vous avez ajouté.

10) Tester désormais le code précédent. Qu'obtenez-vous ?

Normalement vous obtenez le résultat suivant:

```
18:38:15.367 [main] INFO sans_maven.TestLogger - Bonjour Slf4J !
18:38:15.371 [main] INFO sans_maven.TestLogger - Implémentation
```

LE TP EST FINI !!

Quelques remarques à l'issue de ce TP:

- Lorsqu'on gère des dépendances manuellement on a très souvent des problèmes de dépendances transitives.
- Dans le TP je vous ai épargné les problèmes de version de librairies qui peuvent se poser.