Les API de logging

1. Le logging

Le logging (journalisation en français) consiste à ajouter des instructions dans les applications pour émettre et stocker des messages suite à des événements.

Le logging écrit dans des fichiers log les informations nécessaires pour :

- Déboguer : pratique lorsque la mise en œuvre d'un débogueur n'est pas facile.
- Obtenir des traces d'exécution (démarrage/arrêt, informations, avertissements, erreurs d'exécution, ...
- Faciliter la recherche d'une source d'anomalie (stacktrace, ...)
- Comprendre ou vérifier le flux des traitements exécutés : traces des entrées/sorties dans les méthodes, affichage de la pile d'appels, ...

Une API de logging fait intervenir 3 composants :

- Logger : invoqué pour émettre grâce au framework un message généralement avec un niveau de gravité associé
- Formatter : utilisé pour formater le contenu du message
- Appender : utilisé pour envoyer le message à une cible de stockage (console, fichier, base de données, email, ...)

2. Les différences API de logging java :

- Log4j et log4j2
- Java Logging
- Jlog
- Protomatter
- SLF4J
- LogBack

3. L'API Java Logging

C'est l'API fournie avec le JDK

- Import : java.util.logging.Logger
- Les niveaux les plus utilisés sont :
  - Info
  - Warning
  - Severe

- Implémentation

- Créer une classe ayant

- Une variable/attribut de classe (constante) de la classe Logger

  public static final Logger LOGGER =

  Logger.getLogger(NomDeLaClasseLogger.class.getName());

- Une variable de classe de la classe FileHandler
  private static FileHandler fh = null;

- Dans une méthode de classe :

- Implémenter un objet de la classe FileHandler (gestionnaire de fichier) ayant comme valeurs d'attributs le nom du fichier
- Gestion de l'ouverture du fichier : ajout (append) ou non
  fh = new FileHandler("LogAppli.log", true);

- Supprimer tous les handlers (gestionnaires) si on ne veut pas de log dans la console car le handler console est par défaut
  LOGGER.setUseParentHandlers(false);

- Rattacher votre gestionnaire de fichier à notre logger
  LOGGER.addHandler(fh);

- Rattacher de gestionnaire de ficher au formatter
  fh.setFormatter(new FormatterLog());

- Créer une classe permettant de formatter le message

Exemple de formatage :

```
public class FormatterLog extends Formatter {
    @Override
    public String format(LogRecord record) {
        DateFormat format = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
        StringBuilder result = new StringBuilder();
         result.append(format.format(new Date()));
        result.append(" Level : " );
        result.append(record.getLevel());
```

```
        result.append(" / Message : ");
        result.append(record.getMessage());
        result.append(" / Classe : ");
        result.append(record.getSourceClassName());
        result.append(" / Méthode : ");
        result.append(record.getSourceMethodName());
        result.append("\n");
        return result.toString();
    }
```

Dans la classe Main

- Importer la variable de classe

```
import static com.company.utilitaires. NomDeLaClasseLogger.LOGGER;
```

- Utilisation :

```
NomLogger.log(level.niveau,message)
```