Configuration et paramètres de JVM

Java DataBase Connectivity

Programme détaillé ou sommaire

Fichiers de configuration
Paramètres de JVM

Objectifs pédagogiques

À l'issue de cette formation, vous serez en mesure de :

- ✓ Créer des fichiers de configuration pour l'accès à des ressources externes
- ✓ Régler les principaux paramètres d'une JVM: encodage et mémoire

Chapitre 1
Fichiers de configuration

Pourquoi un fichier de configuration

- On ne met pas en dur, dans le code, les informations d'accès à la base de données, ou plus généralement à des ressources externes.

- **Comment ferez vous lorsque vous passerez votre application en production ?**

Avantage du fichier de configuration

- L'équipe qui met **en production** votre application à juste à modifier le fichier de configuration pour renseigner le **user** et **mot de passe de la base de production**.

Qu'utilise t'on en Java ?

- Il existe plusieurs formats de fichiers de configuration
  - Le fichier **properties** utilisé notamment par le framework SPRING
  - Le fichier **XML**
  - Le fichier **YAML**

- Dans tous les cas ce sont des fichiers de type texte

- On les stocke dans les ressources du projet : **src/main/resources**

Chapitre 2
Fichier properties

Le fichier properties

- Le fichier de configuration **le plus simple** à utiliser en Java est le fichier **properties**
  - Celui utilisé par le framework SPRING
  - Stockage sous la forme **clé** = **valeur**.

Le fichier properties

- Un fichier **properties** est un fichier :
  - de type texte
  - avec une extension **.properties** au lieu de .txt

- On le stocke dans les ressources du projet : **src/main/resources**

Exemple de fichier properties

- Exemple de contenu d'un fichier

Contenu de **database.properties**

```
database.driver=org.mariadb.jdbc.Driver
database.url=jdbc:mariadb://localhost:3306/pizzeria
database.user=user127
database.password=wsgskj74&!
```

Lire un fichier properties

- Utilisation de la classe **java.util.ResourceBundle**

- On récupère le fichier properties :
  - en passant son **nom** en paramètre de la méthode static getBundle.
  - on ne précise pas l'extension .properties car elle est implicite.

```
ResourceBundle config = ResourceBundle.getBundle("database");
```

Extraire une valeur dans un fichier properties

- On récupère une valeur avec la méthode getString(key):

```
ResourceBundle config = ResourceBundle.getBundle("database");
String driverName = config.getString("database.driver");
```

org.mariadb.jdbc.Driver

Chapitre 3
Apache commons-configuration2

Fondation Apache

- **Apache Software Foundation** est une communauté de développeurs qui développe et maintient des projets open-source.

- On trouve parmi ces projets des librairies et frameworks de portée internationale
  - ActiveMQ (serveur de messagerie, i.e. message broker)
  - Hadoop (Applications distribuées)
  - Commons: ensemble de librairies Java
  - Maven (configuration)
  - Cassandra (base NoSQL orientée colonne)
  - Spark (framework de calcul distribué, surcouche à Hadoop)
  - Kafka (Data pipeline et architectures micro-services)
  - Tomcat (serveur d'applications)
  - Cordova (développement mobile)

Fondation Apache

- **Projet commons**

- Contient de nombreux sous-projets
  - Commons-digester : cryptage
  - Commons-lang : classes utilitaires (chaines de caractères, nombres, etc.)
  - Commons-io : accès aux fichiers
  - Commons-configuration2 : configuration
- Lien : https://commons.apache.org/

Accès à un fichier properties

- **Fournit une classe permettant d'accéder à un fichier properties**

```
// Permet de lire plusieurs configurations
Configurations configs = new Configurations();
try {
    Configuration config = configs.properties("config.properties");
} catch (ConfigurationException e) {
    e.printStackTrace();
}
```

Accès à un fichier properties

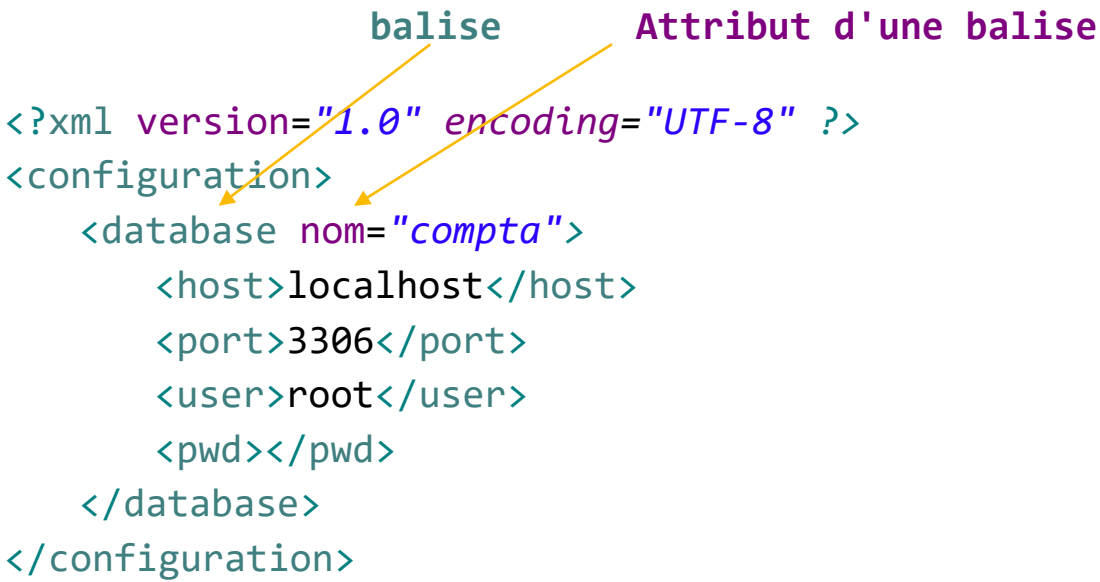
❑ Méthode getString

```
Configurations configs = new Configurations();
Configuration config = configs.properties("config.properties");
String nomBase = config.getString("database.nom");
```

Chapitre 4
Fichier de configuration XML

Accès à un fichier properties

- **La syntaxe XML utilise un système de balises qui ressemble à HTML**
- Exemple fichier **config.xml** situé sans **src/main/resources**

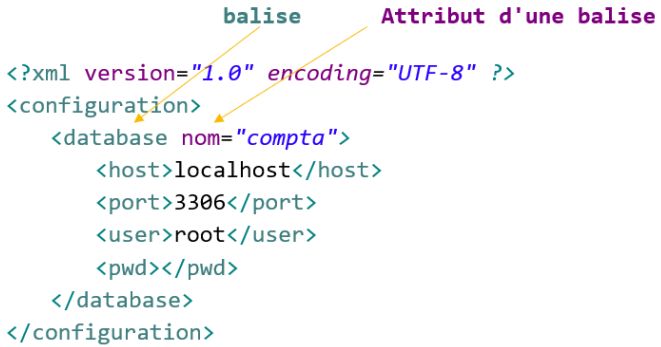


Accès au fichier XML

- **Commons-configuration2 fournit une classe permettant d'accéder à un fichier XML**

```
// Permet de lire plusieurs configurations
Configurations configs = new Configurations();
try {
    XMLConfiguration config = configs.xml("config.xml");
} catch (ConfigurationException e) {
    e.printStackTrace();
}
```

Accès à un fichier XML



- Méthode **getString** avec une notion de **path**

```
Configurations configs = new Configurations();

XMLConfiguration config = configs.xml("config.xml");
String nomBase = config.getString("database[@nom]");
String host = config.getString("database.host");
```

- La notation **database**[@nom] permet d'accéder à la valeur de l'attribut nom appartenant à la balise **database**.
- La notation database.host permet d'accéder à la valeur de la balise host appartenant à la balise database

Atelier (TP)

OBJECTIFS : Créer un fichier de configuration et y accéder

DESCRIPTION :

- Dans le TP n°1 vous allez mettre en place 2 fichiers de configuration et accéder aux données qui s'y trouvent.

Chapitre 2
Paramètres de JVM

A quoi servent les paramètres de JVM ?

- Ils servent à adapter l'environnement d'exécution de l'application:
  - Mémoire utilisée par l'application
  - Caractères UTF-8 pour une application internationale
  - Locale: paramètre culturel

La mémoire

- Paramètres pour la gestion de la mémoire

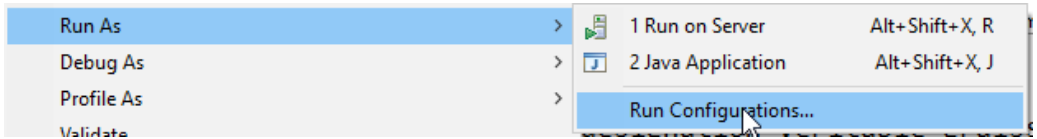
Nom du paramètre	Description
-Xms	allocation mémoire initiale
-Xmx	allocation mémoire maximum

- Exemple en ligne de commande

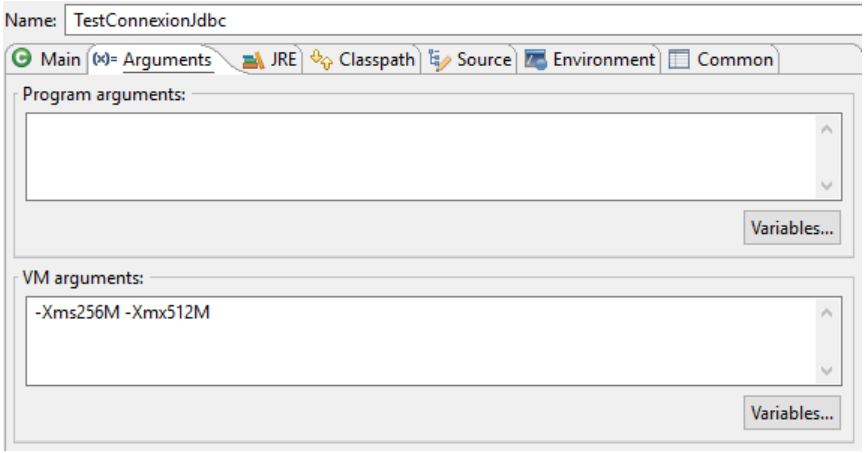
```
java -Xms256M -Xmx512M fr.diginamic.MonProgrammeJava
```

La mémoire depuis STS (i.e. Eclipse)

- Clic droit sur l'application Java puis Run As >> Run Configurations



- Cliquez sur l'onglet **Arguments** puis saisissez les paramètres dans la zone de texte "VM arguments"



L'encodage des caractères

- Il permet à l'application de pouvoir afficher des caractères internationaux.

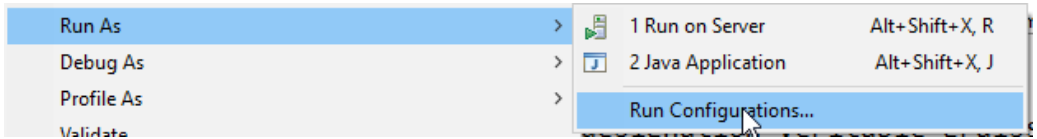
Nom du paramètre	Description	Valeur
file.encoding	Encodage des caractères	UTF-8

- Exemple en ligne de commande
  - -D en préfixe
  - Signe = entre clé et valeur

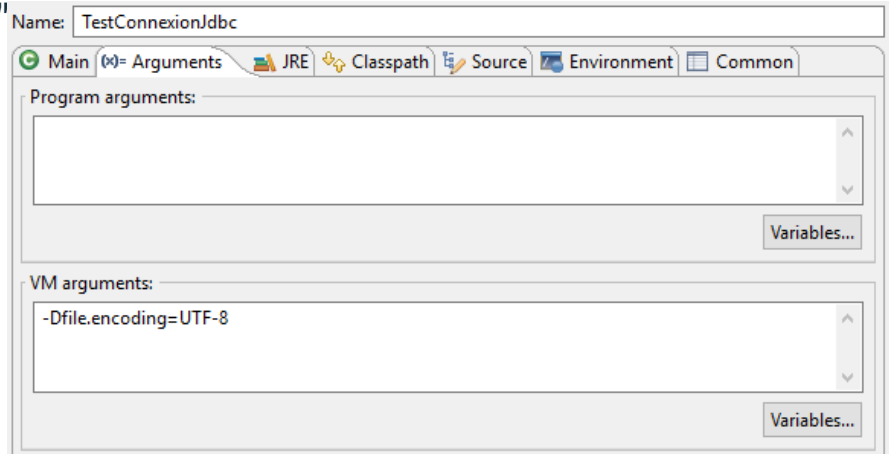
```
java -Dfile.encoding=UTF-8 fr.diginamic.MonProgrammeJava
```

L'encodage des caractères depuis STS (i.e. Eclipse)

- Clic droit sur l'application Java puis Run As >> Run Configurations



- Cliquez sur l'onglet **Arguments** puis saisissez les paramètres dans la zone de texte "VM arguments"



Paramètres de culture

- Ils permettent de régler ce qu'on appelle la Locale par défaut de l'application Java.
- Ces paramètres influent sur le formattage par défaut des dates et des nombres.

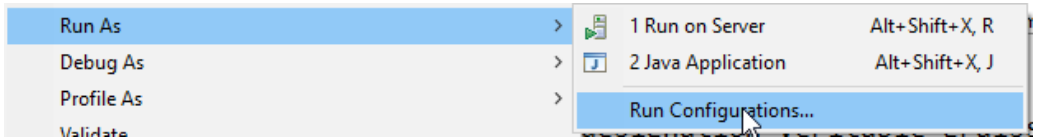
Nom du paramètre	Description	Valeur
user.country	Code du pays	UK, CA, US, FR, etc..
user.language	Code de la langue	en, fr

- Exemple en ligne de commande

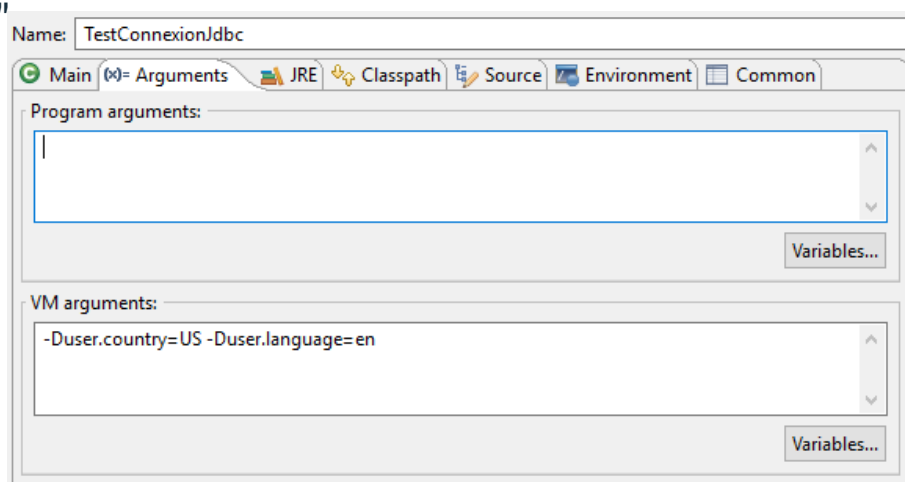
```
java -Duser.country=US -Duser.language=en fr.diginamic.MonProgrammeJava
```

L'encodage des caractères depuis STS (i.e. Eclipse)

- Clic droit sur l'application Java puis Run As >> Run Configurations



- Cliquez sur l'onglet **Arguments** puis saisissez les paramètres dans la zone de texte "VM arguments"



Atelier (TP)

OBJECTIFS : Tester des paramètres de JVM

DESCRIPTION :

- Dans le TP n°2 vous allez tester différents paramétrages de JVM