

Java – Maven

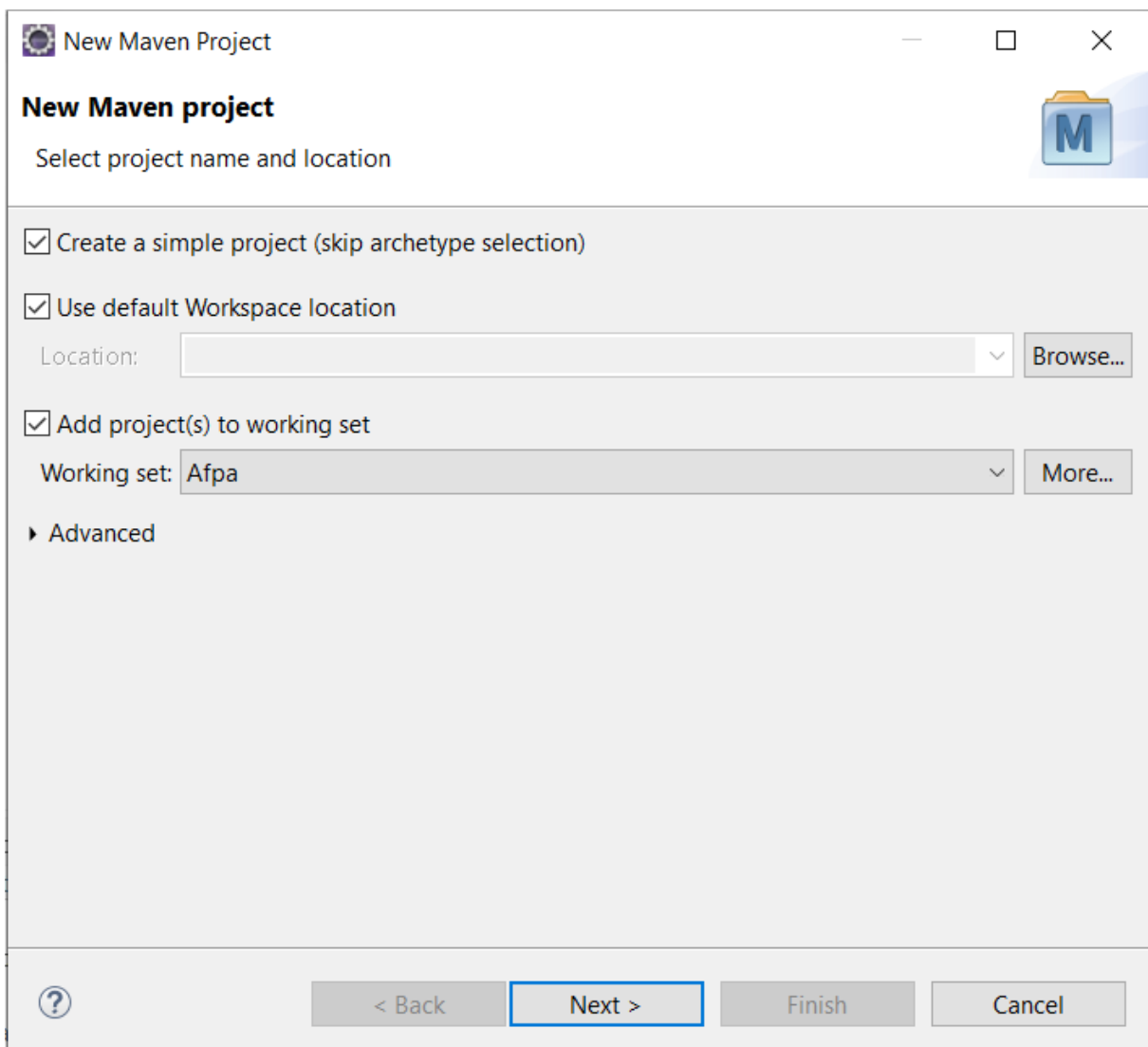
Exercice 1 : Gestionnaire de mails

Énoncé :

1 - Créer un projet Maven simple nommé « MailManager » qui sera packagé sous forme de jar avec les informations suivantes :

Group Id : fr.afpa.cda

Artifact Id : MailManager



New Maven Project

New Maven project

Select project name and location

☒ Create a simple project (skip archetype selection)

☒ Use default Workspace location

Location: Browse...

☒ Add project(s) to working set

Working set: More...

▶ Advanced

? < Back Next > Finish Cancel

New Maven Project

Configure project

Artifact

Group Id: fr.afpa.cda

Artifact Id: MailManager

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name:

Description: Projet de gestion d'une boîte mail

Parent Project

Group Id:

Artifact Id:

Version: Browse... Clear

Advanced

< Back Next > Finish Cancel

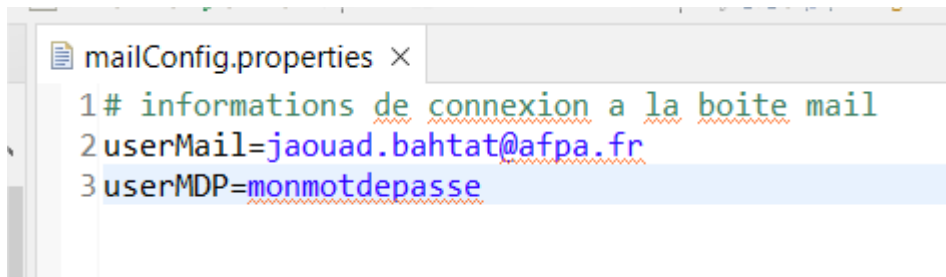
2- Créer un package « fr.afpa.cda.metier » dans les dossier sr/main/java et src/test/java

3- Rechercher sur internet les informations et ajouter les dépendances suivantes dans le pom.xml dans leur dernière version stable

- Junit (avec le scope « test »)
- slf4j

Vérifier que les dépendances ont bien été téléchargées dans le dossier « Maven Dependencies »

4- Créer un fichier de propriétés dans le dossier src/main/resources nommé « mailConfig.properties » et y mettre les propriétés « user » et « mdp » (avec des valeurs de tests)



```
mailConfig.properties ×  
1# informations de connexion a la boite mail  
2userMail=jaouad.bahtat@afpa.fr  
3userMDP=monmotdepasse
```

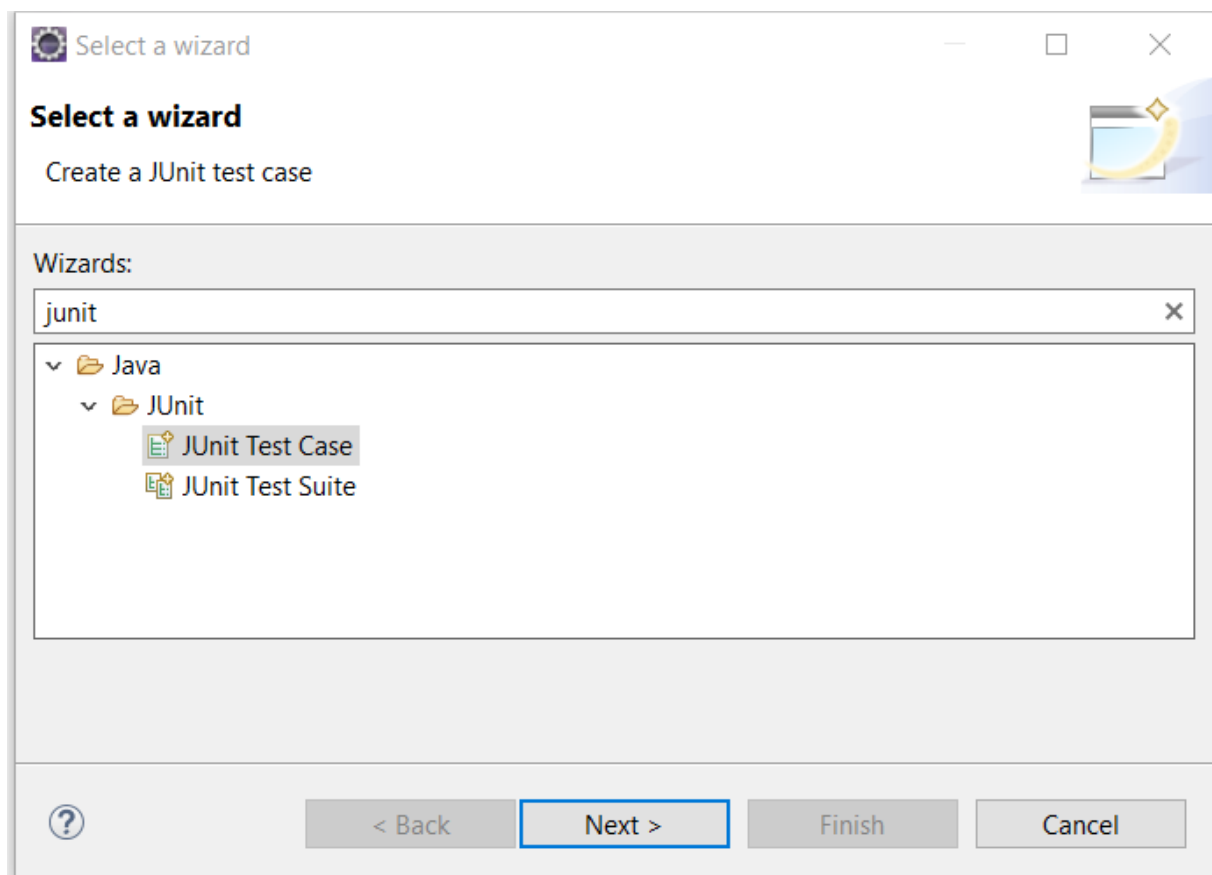
5- Créer une classe « ConnexionManager » avec une méthode nommée « validationMDP » qui prend en paramètre un String représentant un mot de passe et dont le rôle est de vérifier qu’il contient au minimum 8 caractères.

Si c’est le cas elle retourne un boolean vrai sinon false

6- Créer une classe de test Junit (dans le bon emplacement) pour tester la méthode « validationMDP » avec des assertions vérifiant les 2 cas (mot de passe valide et mot de passe invalide) à l’aide des méthode « assertTrue » et « assertFalse »

Exemple : `assertTrue(validationMDP("motdepasse"))` ;

Puis exécuter le test unitaire





New JUnit Test Case



JUnit Test Case



Select the name of the new JUnit test case. Specify the class under test to select methods to be tested on the next page.

☐ New JUnit 3 test ☒ New JUnit 4 test ☐ New JUnit Jupiter test

Source folder:

Package:

Name:

Superclass:

Which method stubs would you like to create?

- ☐ @BeforeClass setUpBeforeClass() ☐ @AfterClass tearDownAfterClass()
☒ @Before setUp() ☐ @After tearDown()
☐ constructor

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Class under test:

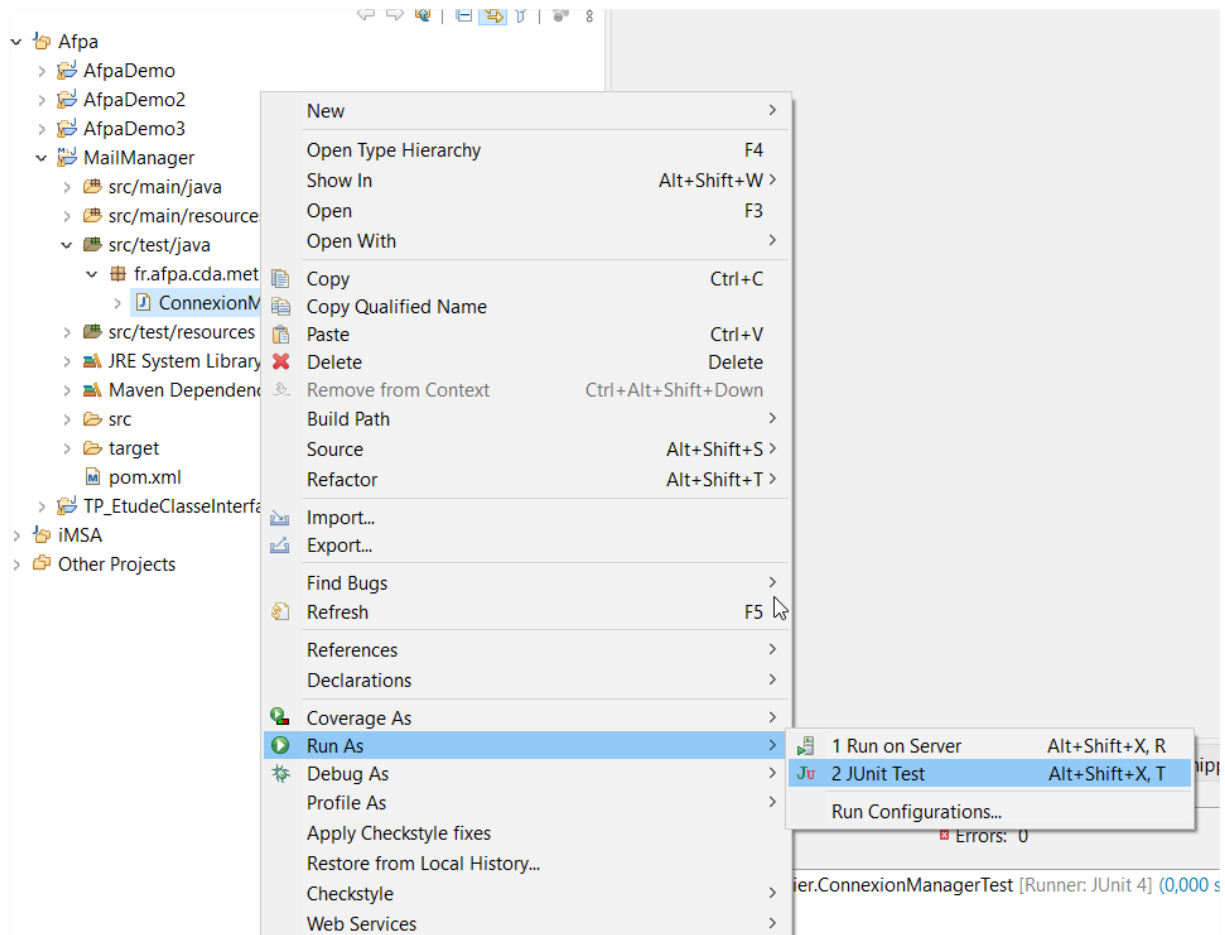


< Back

Next >

Finish

Cancel



7- Créer une autre méthode « connexionMail » qui récupère les informations du fichier properties et fait appel à « validationMDP » avec le mot de passe récupéré, si celle-ci retourne false déclencher une exception interne préalablement créée nommée « InvalidMDPException » avec le message « Le mot de passe est invalide »

8- Créer une autre classe nommée « ConnexionLauncher » qui contient une méthode main qui fait appel à la méthode « connexionMail » et qui en cas d'exception la capture pour afficher le message d'erreur qu'elle contient dans un fichier de log

Exemple d'utilisation du logger :

```
private static final Logger LOGGER =
    LoggerFactory.getLogger(ConnexionLauncher.class);
```

Puis dans la méthode :

```
    } catch (final Exception e) {
        LOGGER.error(e.getMessage(), e);
    }
```

9- Exécuter le programme en ligne de commande

10- Exclure les fichiers properties du jar généré

11- Exécuter le programme en ligne de commande avec les fichiers externalisés

12- Un message ne devant pas être stocké en dur dans un fichier (ni dans une BDD) dans un premier temps l'encoder en base 64 via un outil Web, mettre le résultat dans le properties, puis le décoder dans le code Java avant de l'afficher dans la log en niveau TRACE

13- Maintenant se connecter à une vraie boîte mail laposte

Ajouter la propriété serveur au fichier properties et renseigner les informations de connexion suivantes dans ce properties (ne pas oublier d'encoder le mot de passe) :

Serveur : imap.laposte.net
Mail : formationcda@laposte.net
Password : [FormationAfpa2021!](#)

Rechercher dans le repo Maven et ajouter les dépendances suivantes : « JavaMail API (compat) » et « Jakarta Mail API IMAP Provider »

Déclarer les 2 attributs de classe suivants dans la classe ConnexionManager :

```
/** elements boite mail */  
protected static Store store;  
protected static Folder inbox;
```

Récupérer dans le code toutes les propriétés de connexion et se connecter (avec le protocole IMAP) avec le code suivant (toujours dans la méthode connexionMail) :

```
// init session serveur mail  
Properties props = new Properties();  
Session session = Session.getDefaultInstance(props, null);  
store = session.getStore("imaps");  
store.connect(server, userMail, userPassword);  
  
// chargement du dossier des messages  
inbox = store.getFolder("inbox");  
  
// connexion et recuperation des mail  
// use READ_ONLY if you don't wish the messages  
// to be marked as read after retrieving its content
```

```
//inbox.open(Folder.READ_ONLY);
inbox.open(Folder.READ_WRITE);
```

Puis afficher le nombre de mails total, non lus, lus, et effacés

Exemple : `LOGGER.info("Total mails : " + inbox.getMessageCount());`

14- Créer une classe de transport (BO) nommée « LocalMail » qui servira à transporter les mails avec les attributs :

- sujet : sujet du mail
- expéditeur : adresse mail de l'expéditeur
- dateReception : date de reception du mail
- flag : caractère pouvant prendre la valeur 'R' (pour read), 'U' (pour unread), 'D' (pour delete)
- identifiant : numéro unique permettant d'identifier chaque mail

Puis générer les accesseurs et le constructeur avec paramètres en gérant l'incrémentement de l'id unique.

15- Ajouter un attribut de liste d'objets LocalMail dans la classe ConnexionManager pour y mettre les mails.

Puis dans la méthode connexionMail alimenter cette liste en extrayant les mails depuis l'objet inbox (qui représente le dossier principale de la boîte mail)

Exemple pour les mails lus :

```
// creation du flag de recherche des mails lus
Flags seenFlag = new Flags(Flags.Flag.SEEN);
FlagTerm seenFlagTerm = new FlagTerm(seenFlag, true);

// extraction des messages lus a l'aide du flag
Message[] readMessages = inbox.search(seenFlagTerm);
```

16- Créer une base de données MAILS_DB avec une table « mails » avec les colonnes correspondant aux attributs d'un objet mail. Puis implémenter une classe « MailDAOJdbcImpl » avec les méthodes :

- selectAll : retourne la liste de tous les mails en base
- selectByFlag : retourne la liste des mails selon le flag R, U ou D
- insert : ajoute un mail dans la base

17- A la fin de la méthode connexionMail insérer tous les mails récupérés dans la bdd

18- Créer une autre classe de traitement nommée « MailReader » dans sa méthode main faire ces 2 actions :

- appeler la méthode connexionMail de la classe ConnexionManager
- Demander à l'utilisateur de choisir les mails à lui afficher : « Quels mails voulez-vous consulter ? 1 : mails lus, 2 : mails non lus, 3 mails effacés, 4 tous les mails » puis les afficher en les récupérant de la base dans la log et la console avec le logger en fonction du choix de l'utilisateur en implémentant une méthode toString() dans la classe LocalMail

19- Ajouter la récupération et le stockage du contenu des mails puis demander à l'utilisateur à travers une boîte de dialogue de saisir l'identifiant du mail qu'il souhaite consulter et l'afficher dans une popup

20- Implémenter une méthode sendMail pour envoyer un mail

21- Implémenter les méthodes faites en JDBC avec JPA et Hibernate :

- supprimer tous les mails de la table (à appeler avant la sauvegarde des mails)
- sauvegarder les mails
- lire tous les mails
- lire les mails selon le flag
- consulter le corps d'un mail selon son id

22- Ajouter la gestion de la suppression d'un mail, implémenter une méthode qui :

- supprime un mail de la boîte mail
- si ok supprime le mail de la base selon son id