Introduction

Cette formation vous permettra de maîtriser la programmation objet en Java, la programmation concurrente et les outils de base du développement. Elle constitue le premier élément du parcours « formation de base Java EE ».

- Plan du cour
- Fil Rouge
- Fil Rouge Civadis

Plan du cour

Slide

1) introduction *(environ 1/2 de journée)*

- Qu'est-ce que Java, une JVM

- Le framework Java EE, les différents serveurs Applicatif J2EE ?

- Qu'est-ce qu'Eclipse ? Comment le configurer ?

- Le debuggage

Travaux pratiques

Installation du JDK, gestion de la variable d'environnement classpath. Installation puis utilisation de l'IDE Eclipse en perspective Java, utilisation de Eclipse en mode debug.

2) Les constructions de base du langage *(environ 3/4 de journée)*

- Les variables : déclaration et typage.

- Les méthodes : définition.

- Les expressions.

- Les instructions de contrôle : les instructions conditionnelles, de boucle, de branchement.

- Les tableaux.

- Les unités de compilation et packages : le contrôle de la visibilité des classes, le mécanisme d'import.

- Les imports statiques.

Travaux pratiques

Suite d'exercices simples permettant la prise en main de l'environnement de développement (Eclipse en perspective Java), notamment la documentation et la réalisation d'un programme

simple. Utilisation des packages.

3) Tests logiciels *(environ 3/4 de journée)*

- Pourquoi faire des tests ?

- Présentation des différents types de tests : tests unitaires, fonctionnels, de robustesse et de performance.

- Quels tests lancer et quand ?

- Utilité des objets "Mock" et "Fake" durant les tests unitaires. Couverture des tests unitaires.

Travaux pratiques

Installer JUnit sous Eclipse. Ajouter des tests unitaires avec JUnit sur les projets Eclipse déjà écrits.

4) Bonnes pratiques de conception d'une application *(environ 1/2 de journée)*

- Découpage en couches (données, métier, présentation).

- Présentation des enjeux d'un développement d'entreprise.

- Introduction à l'écosystème JAVA (JEE, Spring, Hibernate, JSF ou Struts...).

Travaux pratiques

Echanges quant au choix technique du framework Java EE Web Profile. Réflexion sur la conception en couche.

5) Les techniques Objet *(environ 1/2 journée)*

- Les principes généraux de la modélisation et de la programmation Objet.

- L'abstraction et l'encapsulation : les interfaces.

- Les différentes formes d'héritage, le polymorphisme.

- Introduction à la modélisation UML

Travaux pratiques

La notion d'identité simple seras étendue dans l'application de registre d'identité afin de faire apparaitre des particularités.

6) La gestion des versions : introduction à SVN *(environ 3/4 de journée)*

- Les concepts généraux liés à la gestion de versions.

- Les concepts SVN : dépôt, projets, révisions, tronc, branches et tags.

- Les principales opérations offertes au développeur. La gestion des conflits.

- La gestion des branches. Les perspectives SVN proposées par les plug-ins Eclipse.

Travaux pratiques

Installation d'une solution de versioning, installation du bon plug-in dans Eclipse. Création de projets associés à un repository. Gestion des versions de l'application développée – récupération d'une copie locale, modifications, fusion, commit.

7) Définition de la structure d'un projet avec Maven *(environ 3/4 de journée)*

- Définition de la structure d'un projet.

- Les conventions. Les dépendances entre projets. Les tâches prédéfinies : compilation, génération d'archives...

- Les perspectives Maven proposées par les plug-ins Eclipse.

Travaux pratiques

Utilisation de Maven sous Eclipse pour automatiser une succession de traitements notamment de test.

8) Quelques Design Pattern *(environ 1/4 de journée)*

- Les objectifs et les avantages.

- Les Design Patterns les plus populaires dans les architectures logicielles modernes.

9) mesures de la qualité *(environ 1/4 de journée)*

- Synthèse des mesures qualité.

- La convention de codage et la documentation.

- La couverture de tests et l'automatisation des procédures.

- Mise en place d'un tableau de bord de la qualité.

Fil Rouge

Contexte

L’ETNIC possède un site web dédié aux offres d’emploi (MonJob @ Etnic).

Ce site se sépare en trois parties front-end, backend-utilisateur, backend-administrateur.

La partie front end permet :

- De savoir ce qu’est MonJob,
- D’avoir un mail/téléphone de contact à l’ETNIC,
- D’avoir accès à une liste de questions les plus souvent posées,
- De parcourir les offres d’emploi disponibles sous forme de liste (description rapide des postes) et d’avoir par poste une fiche détaillée,
- De se connecter au backend-utilisateur ou se créer un compte.

La partie backend utilisateur permet

- De créer un CV composé de données personnels (date de naissance, lieu, sexe..), de ses certificats, de ses diplômes, de ses expériences,
- De parcourir les offres d’emploi disponibles et de postuler à ces offres.

La partie backend administrateur permet

- De se connecter,
- De créer une offre d’emploi,
- De voir les candidatures ayant postulé à une offre,
- D’accepter ou de refuser une offre.

Risques

Cette application n’est pas une application critique (elle n’empêche pas l’ETNIC de fonctionner en cas d’arrêt), mais c’est un cas typique d’application dont la qualité doit être élevée car :

- Les bugs divers de l’application donnent une mauvaise image de l’ETNIC,
- Cette application contient des données personnelles de candidate. Un bug sur l’application peut entraîner des problèmes législatifs (RGPD).

Il est donc essentiel pendant la formation et ce fil rouge de faire les tests suffisants pour éviter tous problèmes

Parcours Utilisateurs

Il y a trois parcours utilisateurs :

- Le premier parcours est celui du curieux. La personne curieuse tombe par hasard sur le site MonJob. Il va regarder rapidement les offres d'emplois. Enfin, il quitte le site.
- Le parcours du postulant est différent. Il doit en premier se créer un compte en donnant un login et un password. Enfin, il reçoit un email validant son inscription. Après la finalisation de son inscription, il remplit son CV et postule à une offre. Enfin, il se déconnecte de MonJob et attend avec ou sans angoisse de passer à l'étape 2 du processus de recrutement.
- Le parcours administrateur est singulier. L'administrateur se connecte sur MonJob soit pour créer un Job soit pour voir les personnes ayant postulé. Il est au courant qu'une personne a postulé à un Job car il reçoit un email à chaque candidature. Sur les postulants, l'administrateur peut accepter la candidature (un mail heureux part vers le candidat et un autre part vers le responsable de l'offre d'emploi) soit la refuser (un mail malheureux part vers le candidat).

Réalisation

Le but de ce fil rouge n'est pas de refaire MonJob en entier dans le cadre de la formation.

Ci-après les objectifs plus ou moins atteignables lors de la formation :

Objectif : Niveau 1 je pratique le Java/J2EE et je suis à l'aise

Être à l'aise avec Java/J2EE est de pouvoir faire quelques classes, construire une application Web et mettre en place les éléments de qualité logiciel me garantissant que cela fonctionne.

Je dois faire l'application suivante :

- Une page Web permettant de créer un Jobs pour l'administrateur

- Une page Web permettant de visualisez les Jobs sous forme de table et sous forme détaillé
- Une page Web d'info utile.

Ce que je dois comprendre est :

- Faire des classes simples en Java et faire des pages web simples pour les gérer,
- **Comment tester cette application, Identifier les points durs en termes de qualité logiciel. Je dois savoir faire les tests unitaires de ces classes, d'en faire les tests graphiques.**
- **Ma couverture de code de mes tests est de 60%.**

Objectif : Niveau 2 je connais le Java/J2EE et je suis plus qu'à l'aise

Je suis assez à l'aise pour monter une architecture plus complexe et en faire une application Web. Je comprends que la mise en place des tests est une nécessité pour maintenir mon architecture Web stable.

Je dois faire l'application suivante :

- Une page Web permettant de créer des Jobs.
- Une page Web permettant de créer un Job pour l'administrateur.
- Une page Web d'info utile.
- La création d'un compte candidat (avec login/password et gestion du mail d'activation).
- Le candidat peut remplir quelques données personnelles simples
- Le candidat peut postuler à un Job.

Ce que je dois comprendre :

- **Comment monter une architecture complexe de classe Java et de mettre en place un début d'authentification utilisateur.**
- **Je sais monter, tout au long de mon travail, les tests unitaires qui garantissent mes développements futurs.**
- **Je monte des tests fonctionnels depuis l'interface graphique en bouchonnant ou pas mes éléments.**
- **Ma couverture de code de mes tests est de 70%**

Objectif : Niveau 3 Le Java/J2EE n’est pas le problème ici

Je pense déjà au futur de cette application. J’ai compris comment monter une architecture très complexe, stable dans le temps et suffisamment performante pour accueillir plein de candidats.

Je dois faire l’application suivante :

- Une page Web permettant de créer des Jobs.
- Une page Web permettant, à l’administrateur, de créer un Job et d’associer un responsable du Job.
- Une page Web d’info utile.
- La création d’un compte candidat (avec login/password et gestion du mail d’activation).
- Le candidat peut remplir toutes données personnelles.
- Il peut remplir ses CV, ses certificats et ses expériences . Les opérations suivantes sont possibles :

A minima, il est possible de préciser un titre au CV, donner un texte libre et uploader du CV. Le format de l’upload est du PDF d’une taille maximal de 200ko.

- Le candidat peut postuler à un Job.
- L’administrateur visualise pour chaque Job et postulant. Cela lui permet d’accepter une candidature ou de la refuser.

Je comprends largement comment faire cette application. J’ai monté les tests unitaires et les tests fonctionnels. Ma couverture de code est bonne (70%), mes tests fonctionnels sont complets et je me prépare à monter des tests de performances et des tests de sécurité.

Objet du système

Nous donnons ici les caractéristiques des objets du système

Un Job est défini par :

- Niveau 1,2, 3 :D’un titre, d’un résumé, d’une date limite de candidature et d’une description détaillée
- Niveau 2,3 : D’une liste de candidat

Un Candidat est défini par :

- Niveau 2,3 d’un login/password/email et de quelques données personnels (nom, prénom, adresse, nationalité)
- Niveau 3 : de diplôme (nationalité du diplôme, titre, niveau), de certificat (description) et de compétence (description).

Fil Rouge Civadis

Gestion Simplifié de recrutement

Contexte

Le but de l'application est de fournir à Civadis un logiciel de gestion simplifié du recrutement.

Ce site se sépare en trois parties front-end, backend-utilisateur, backend-administrateur.

La partie front end permet :

- De pouvoir positionner une Vitrine
- De parcourir les offres d'emploi disponibles sous forme de liste (description rapide des postes) et d'avoir par poste une fiche détaillée,
- De se connecter au backend-utilisateur via un email/password ou se créer un compte.
- En fonction de l'email, l'utilisateur à accès aux fonctionnalités RH (backend Civadis) ou pas (Backend Invité).

La partie Backend Invité permet

- De créer un CV composé de données personnels (date de naissance, lieu, sexe..), de ses certificats, de ses diplômes, de ses expériences,
- De parcourir les offres d'emploi disponibles et de postuler à ces offres.

La partie backend administrateur permet

- De se connecter,
- De créer une offre d'emploi,

- De voir les candidatures ayant postulé à une offre,
- De sélectionner les candidatures

Risques

Cette application n'est pas une application critique (elle n'empêche pas Civadis de fonctionner en cas d'arrêt), mais c'est un cas typique d'application dont la qualité doit être élevée car :

- Les bugs divers de l'application donnent une mauvaise image de Civadis,

- Cette application contient des données personnelles de candidate. Un bug sur l'application peut entraîner des problèmes législatifs (RGPD).

Il est donc essentiel pendant la formation et ce fil rouge de faire les tests suffisants pour éviter tous problèmes

Parcours Utilisateurs

Il y a trois parcours utilisateurs :

- Le premier parcours est celui du curieux. La personne curieuse tombe par hasard sur le site de recrutement. Il va regarder rapidement les offres d'emplois. Enfin, il quitte le site.

- Le parcours du postulant est différent. Il doit en premier se créer un compte en donnant un login et un password. Enfin, il reçoit un email validant son inscription. Après la finalisation de son inscription, il remplit son CV et postule à une offre. Enfin, il se déconnecte de l'applicatif et attend avec ou sans angoisse de passer aux étapes ultérieures du processus de recrutement. Il est informé du passage des étapes (épreuves 1, épreuves 2, décision) par courriel.
- Le parcours administrateur est singulier. L'administrateur se connecte sur l'applicatif soit pour créer un Job soit pour voir les personnes ayant postulé ou ceux en épreuves. Il est au courant qu'une personne a postulé à un Job car il reçoit un email à chaque candidature. Sur les postulants, l'administrateur peut accepter la candidature (un mail heureux part vers le candidat et un autre part vers le responsable de l'offre d'emploi) soit la refuser (un mail malheureux part vers le candidat). Dans le cas où la candidature est acceptée, cette dernière positionne le candidat en mode passage d'épreuves. A chaque épreuve réussie, le RH peut modifier l'état du candidat, ce qui provoque une notification.

Réalisation

Le but de ce fil rouge n'est pas de refaire l'applicatif de recrutement en entier dans le cadre de la formation.

Ci-après les objectifs plus ou moins atteignables lors de la formation :

Objectif : Niveau 1 je pratique le Java et je suis à l'aise

Être à l'aise avec Java est de pouvoir faire quelques classes, construire une application Web et mettre en place les éléments de qualité logiciel me garantissant que cela fonctionne.

Je dois faire l'application suivante :

- Une API REST permettant de lister des Jobs en liste en fonction d'un statut (En cours d'élaboration, Validée, publiée)
- Une API REST pour afficher un job en détail.

- Une API REST pour modifier un job
- Une API REST permettant de créer un Job pour l'administrateur.

Afin de pouvoir créer une offre, il est nécessaire de pouvoir encoder pour créer une offre (A titre d'exemple) :

- Numéro de l'offre
- Titre de l'offre

- Statut : (En cours d'élaboration, Validée, publiée)
- Type engagement
- Lieu travail
- Régime de travail
- Diplôme requis

- Vos responsabilités
- Votre profil
- Notre offre

En termes d'architecture, il est proposé mais non imposé l'approche suivante :

- Faire un cœur applicatif sans authentification ni base de données

- Rajouter la base de données et le mode REST sur le cœur applicatif
- Rajouter l'authentification.

Ce que je dois comprendre est :

- Faire des classes simples en Java et faire des pages web simples pour les gérer,
- **Comment tester cette application, Identifier les points durs en termes de qualité logiciel. Je dois savoir faire les tests unitaires de ces classes, d'en faire les tests graphiques.**

- **Ma couverture de code de mes tests est de 60%.**

Objectif : Niveau 2 je connais le Java et je suis plus qu'à l'aise

Je suis assez à l'aise pour monter une architecture plus complexe et en faire une application Web. Je comprends que la mise en place des tests est une nécessité pour maintenir mon architecture Web stable.

Je dois faire l'application suivante :

- Une API REST permettant de lister des Jobs en liste en fonction d'un statut (En cours d'élaboration, Validée, publiée)

- Une API REST pour afficher un job en détail.
- Une API REST pour modifier un job
- Une API REST permettant de créer un Job pour l'administrateur.

- Une API REST permettant d’initialiser le compte candidat (avec login/password et gestion du mail d’activation).
- Une API REST permettant de créer un candidat.

- Une modification de l’API REST d’un Job afin d’affilier un candidat à un Job publié.

La candidature d’une personne peut contenir à titre d’exemple :

- Données personnelles

- Nom

- Prénom
- Sexe
- Date de naissance
- Nationalité
- Langue

- Email
- Numéro téléphone

- Adresse

- Rue
- Numéro

- Code postal
- Ville
- Pays

- Connaissance

- Diplôme

- Langue

- Français (pas de connaissance, Notions, Bonne Très bonne, langue maternelle)
- Néerlandais (pas de connaissance, Notions, Bonne Très bonne, langue maternelle)
- Anglais (pas de connaissance, Notions, Bonne Très bonne, langue maternelle)

- Informations complémentaires

- CV
- Motivation
- URL profil linkedin

En termes d'architecture, il est proposé mais non imposé l'approche suivante :

- Faire un cœur applicatif sans authentification ni base de données
- Rajouter la base de données et le mode REST sur le cœur applicatif
- Rajouter l'authentification.

Afin de mettre en place la création d'utilisateur qui nécessite une validation d'email, il est possible d'utiliser les composants suivants :

- Validation de utilisateurs par email : https://www.baeldung.com/registration-verify-user-by-email
- Sauvegarder des informations semi structurée en base de données : https://www.baeldung.com/hibernate-persist-json-object
- Un serveur smtp bouchonné en userland : https://github.com/maildev/maildev

Ce que je dois comprendre :

- **Comment monter une architecture complexe de classe Java et de mettre en place un début d'authentification utilisateur.**
- **Je sais monter, tout au long de mon travail, les tests unitaires qui garantissent mes développements futurs.**
- **Je monte des tests fonctionnels depuis les API REST en bouchonnant ou pas mes éléments.**
- **Ma couverture de code de mes tests est de 70%**

Objectif : Niveau 3 Le Java n'est pas le problème ici

Je pense déjà au futur de cette application. J'ai compris comment monter une architecture très complexe, stable dans le temps et suffisamment performante pour accueillir plein de candidats.

Je dois faire l'application suivante :

- Une API REST permettant de lister des Jobs en liste en fonction d'un statut (En cours d'élaboration, Validée, publiée, supprimé)
- Une API REST pour afficher un job en détail.
- Une API REST pour modifier un job. La modification pour cette partie consiste principalement a rajouter un email de contact pour le Job.
- Une API REST permettant de créer un Job pour l'administrateur.
- Une API REST permettant d'initialiser le compte candidat (avec login/password et gestion du mail d'activation).

- Une API REST permettant de créer un candidat.
- Une modification de l'API REST d'un Job afin d'affilier un candidat à un Job publié.
- Une API REST est disponible pour pouvoir pousser des pièces jointes (certificats, diplôme, CV …) sur un candidat.
- Une API REST pour modifier l'état d'un candidat au regard de ses épreuves.

Pour chacune des offres les candidats doivent passer deux épreuves, une première épreuve écrite et dans le cas où l'épreuve écrite est réussie, une épreuve orale.

L'application doit permettre au gestionnaire RH de sélectionner à chaque étape les candidats retenus pour l'étape suivante. A chaque passage d'une étape à l'autre, les candidats non retenus doivent recevoir un mail et les autres candidats reçoivent un autre mail pour les inviter à l'étape suivante. Enfin pour chaque candidat avançant dans le process, il est possible d'envoyer un courriel à la personne de contact.

Lorsqu'un candidat est retenu, le Job est dépublié automatiquement.

Afin de mettre en place la fonctionnalité d'Upload de pièce d'un utilisateur, il est possible de s'inspirer de :

- Uploading Files : https://spring.io/guides/gs/uploading-files/

Je comprends largement comment faire cette application. J'ai monté les tests unitaires et les tests fonctionnels. Ma couverture de code est bonne (70%), mes tests fonctionnels sont complets et je me prépare à monter des tests de performances et des tests de sécurité de base.

Objectif : Je pratique de l'Angular

Cet objectif est transverse aux objectifs Java. Le but ici est de monter un applicatif Angular avec PrimeNG afin :

- Permettre de lister des Jobs en liste en fonction d'un statut (En cours d'élaboration, Validée, publiée, supprimé)
- Permettre pour afficher un job en détail.
- Permettre de créer/modifier un Job pour l'administrateur.
- Permettre d'initialiser le compte candidat (avec login/password et gestion du mail d'activation).

- Permettre de créer un candidat.
- Permettre d’affilier un candidat à un Job publié.

De façon non exhaustive et non obligatoire, il est possible de s’inspirer des sketch suivants fortement inspiré de :

https://www.primefaces.org/primeng/showcase/#/dataview

Une deuxième page permet d’afficher un job en détail :

En fonction de l’état de l’utilisateur (connecté ou pas), il est possible de postuler à l’offre.

L’interface backend RH permet pour chaque Job d’avoir la liste des candidats d’un job.