

Framework Java EE

Résumé des TDs

François Robert



francois.robert@shipstone.org

Initialisation



Un peu de shell dans ce monde de brutes

1. Lancer la VM (login: **siad** - password : **javaeesaid**)
2. Ouvrez une console, et rendez vous dans le répertoire dev de votre home
3. créez un répertoire workplaces et placez vous y...
 1. astuce : `mkdir workplaces && cd $_`
4. Maintenant il faut cloner et initialiser le projet de TD
 1. executer **`git clone git@github.com:ptitbob/siad_m1_jee.git`**
 2. Vous placer dans le répertoire `siad_m1_jee`
 3. puis executer **`mvn clean compile`**
 1. Il devrait télécharger le web...
5. Lancer IntelliJ 13
 1. importer le projet

*Pour faire les mise à jour pour chaque cours : **`git pull origin`***



CDI

Module CDI

Exercice 0

- Apprentissage du système de logging
- Tous les classes de test CDI hérite de SiadTest
 - Mise en place du système de logging
 - Mise en place du container CDI (weld)
 - Preuve que cela ne fonctionne pas que dans JavaEE
- Utilisation de JUnit (framework de test le plus utilisé)
- Corriger la classe MiscService pour passer le test



CDI

Module CDI

Exercice 1

Attention :: dans le code, anglais == convention

- Instancier la classe (via le générateur de TestSiad)
- Utiliser l'injection afin que le test passe
 - La génération doit se faire une autre classe
 - Evidement new interdit
 - Sans addition de champs dans la classe ni modification du constructeur
- le nombre généré doit respecter la contrainte exprimé dans le test

*Cette dernière condition permet de mettre en place le développement selon la méthodologie **TDD**.*

CDI

Module CDI



Exercice 2

- Même problème que précédemment, mais...
- L'affectation de la classe de génération du nombre doit se faire via le constructeur de la classe `ServiceUser02`

Attention :: dans le code, anglais == convention

Bean Validation

Module Contrainte



Exercice 1

- Utilisez les contraintes standard pour valider :
 - Le nom est non null et ne dépasse pas 50 char
 - Le prénom ne dépasse pas 50 char
 - que la date de naissance est affecté
 - et que le nombre d'activités est supérieur à 2
- Le test `validatePerson` doit passer au vert.

Regardez les message renvoyés...



Bean Validation

Module Contrainte

Exercice 1.1

- Pour la classe `personne`, créer une contrainte `NameConvention`
 - Qui devra valider que le nom est non null
 - Qui le nom ne dépasse pas 50 char
 - Qui devra envoyer le message suivant:
 - « Le nom ne suit pas les conventions »
- Le test `validateNameConvention` doit passer au vert.

Astuce : combinez des contraintes standards



Bean Validation

Module Contrainte

Exercice 1.2

- Pour la classe `personne`, créer une contrainte `NameUpperCase`
 - Qui devra valider que le nom est en majuscule
 - Qui devra envoyer le message suivant:
 - « Le nom doit être en majuscule »
 - Tout en appliquant la contrainte de l'exercice 1.1
 - Le test `validateNameUpperCase` doit passer au vert.

Astuce : Regardez les options offerte par `common lang 3` (`StringUtils`)



Bean Validation

Module Contrainte

Exercice 2

Le but de cet exercice est de tester les contraintes imbriquées

La classe Customer doit satisfaire les contraintes suivantes :

- Le nom est obligatoire
- Le prénom est facultatif
- L'adresse doit être valide
- le premier champs de rue est obligatoire
- La ville doit être renseigné
- Le code postal doit faire 5 caractères
- La date de dernier achat doit être dans le passé
- L'identifiant est obligatoire

Mettez en place les tests décrivant ces contraintes et validez les

JPA

Module JPA



Préparation de la machine virtuelle

- Utilisation du serveur de base de donnée MySQL
 - Démarrer le service :
 - `sudo start mysql`
 - Arrêter le service
 - `sudo stop mysql`

JPA

Module JPA



Installation du client MySQL

- Via la logithèque ubuntu
- Faire une recherche sur « Client MySQL »
- Installer et executer (MySQLWorkbench)

Exercice 0

- Créer le schéma SIAD_M1
- Modifier le fichier persistance.xml
remplacer le mot de passe par javaeesiad
- Executer le test
- Vérifier que la table person a bien été créée et possède 1 enregistrement

JPA

Module JPA



Exercice 1 - Enrichir la classe person

- Le nom de la table doit être USER
- Le nom de chaque colonne doit être préfixé par USER_ et doit être en majuscule
- Le nom ne doit pas excéder 100 char ne doit pas être null
- Le prénom ne doit pas excéder 100 char, peut être null
- login , max 100 char et ne doit pas être null et doit être unique
- N'oubliez pas que l'identifiant (par convention : id et attention à la convention demande plus haut) qui ne pourra pas être l'objet d'un update.
- un champs genre, qui doit être un énuméré qui sera stocké sous forme de chaine.
- Executer votre code via les test et observez la DB.

Attention :: dans le code, anglais == convention

JPA

Module JPA



Exercice 1.1 - Enrichir la classe person

- Incluez le champs date de naissance
- Incluez la propriété âge qui ne sera pas persistée mais calculée

Attention :: dans le code, anglais == convention

JPA

Module JPA



Exercice 1.2 - Enrichir la classe Person

- Inclure les champs suivant dans la classe Person
 - Nom (Obligatoire)
 - Prénom
 - Ville
- Transférer les logins et mot de passe doivent être dans une classe extérieur à la classe Person mais persisté dans la table USER
- Transférer la ville dans une table secondaire nommé CITIES
- Utilisez les tests pour exécuter votre persistance et valider la création et l'alimentation des différentes tables

Attention :: dans le code, anglais == convention



JPA

Module JPA

Exercice 2

- Avec la classe Person
 - Que le test passe, tout simplement

Attention :: dans le code, anglais == convention

JPA

Module JPA



Exercice 2-1

- Avec la classe Person
 - Déterminez vous pour une politique de génération de clé pour que le test passe
 - Observez la base et expliquez ce que vous y voyez

Attention :: dans le code, anglais == convention