Moyen guide du nouveau développeur

A. Préparation de l'environnement de travail Eclipse

1. Créer un compte dans github
2. Installer Eclipse IDE for Java EE Developers (http://www.eclipse.org/downloads/)
    a. Sur Windows 8, Installer la version **Juno**
    b. Sur les autres systèmes d'exploitation, Installer la version **Juno ou Kepler**
    c. Tenez vous loin de Luna jusqu'à nouvel ordre
3. **S'assurer d'avoir un Workspace en UTF-8 (Défaut Windows en cp1252)**
4. **FACULTATIF :** Dans Eclipse, installer "Infinitest" à partir de Help>Eclipse Marketplace
5. Dans Eclipse, installer "Vaadin Plugin for Eclipse" à partir de Help>Eclipse Marketplace
6. Dans Eclipse, installer "Explore FS" à partir de Help>Install new Software > Add "http://junginger.biz/eclipse/" > choisir "Explore FS"
7. Dans Eclipse, installer "EGit - Git Team Provider" à partir de Help>Eclipse Market
8. Dans Eclipse, installer "Gradle Integration for Eclipse" à partir de Help>Eclipse Marketplace (En cas de problème, passer par l'update site)

9. Pour utiliser Eclipse ET IntelliJ (recommandé)
    a. Voir les sections G et C
10. Pour utiliser uniquement Eclipse
    a. Assurer vous d'avoir importé le repository de Github avant de continuer la configuration de Gradle (voir section B)
    b. Dans Eclipse, Window>Preferences>Gradle, ajouter le path du Gradle au "Gradle home to use" ce path doit pointer sur le dossier tools/gradle à la racine du project sur github
    ATTENTION : la version de graddle dans tools/gradle est la 1.11. Il y aura incomptatibilité. Télécharger Graddle dans sa dernière version (2.2.1), et choisir le dossier racine à la place
    c. Voir la sections C
11. Dans Eclipse, right-click Import… Preferences, choisir le fichier eclipse.preferences.epf du projet IntelliGID et sélectionner tous les éléments de configuration
12. Dans Eclipse, aller dans Preferences > Java > Code Style > Formatter > Edit IntelliGID dev team > OK > Apply

NOTE : Sous Windows 8, l'éditeur graphique de Vaadin ne fonctionne pas dans Eclipse Kepler. La meilleure solution est d'installer deux Eclipse: un Eclipse Kepler avec tout l'environnement de développement nécessaire, et un Eclipse Juno qui n'est utilisé que pour l'éditeur graphique de Vaadin. Il s'agit d'une solution temporaire en attendant qu'une mise à

jour d'Eclipse ou du plugin Vaadin règle le problème dans Kepler.

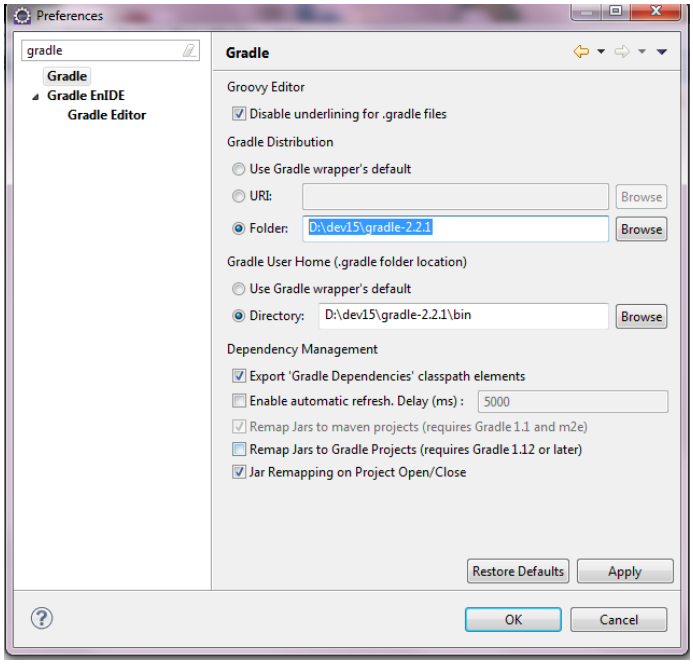
B. Importation du Git IntelliGID de Github

1. Ouvrir la perspective Git dans Eclipse
2. Dans la fenetre Git Repositories choisir Clone a Git Repository
3. Dans URI entrer l'adresse de Github : https://github.com/doculibre/intelligid-dev.git
4. Dans Authentification, User mettre vos informations de connection de Github
5. Selectionner Store in Secure Store
6. Selectionner la branche master
7. Dans Destination>Directory Choisisez un dossier pour le repo qui est différent de la workspace d'Eclipse

C. Importation et generation des projets Intelligid pour Eclipse

1. Dans Eclipse, File>Import…, Choisissez la catégorie Gradle>Gradle Project
2. Dans Root Folder, choisissez le path vers le dossier intelligid-dev, puis cliquez sur Build Model
    a. Si vous utilisez intellij, choisir le dossier de la workspace d'intelliJ
    b. Sinon, choisir le repository Github local créé à l'étape B

    Si problème avec la version de gradle faire le lien au bon répertoire, exemple :



3. Selectionner tous les projets puis finish

4. Dans le package explorer, cliquer a droite sur le fichier build.gradle du projet a compiler/tester/packager (**fort probablement le build.gradle du projet Constellio pour la première fois**) et choisir Run As>Gradle Build
5. Dans l'onglet Arguments, dans Java Home cocher Workspace JRE et sélectionner une jdk. Pour Windows, le build Gradle échouera s'il s'agit d'une JRE seulement. Si aucune jdk n'est offerte, aller dans Configure JREs > Add et parcourir jusqu'au répertoire racine de votre JDK.
6. Selectionner les task que vous voulez executer par défaut :
    a. build -> compile, test et crée le .jar final
    b. compileJava -> compile les sources

S'il y a des : Caused by: org.gradle.internal.UncheckedException: java.io.IOException: Cannot run program "java": CreateProcess error=206, The filename or extension is too long

Le problème vient de la compilation des widgetSet de Vaadin sous Windows

Lorsqu'on roule les tests, s'il y a des : java.lang.RuntimeException: 'C:\git\intelligid-dev\sdk\sdk.properties' does not exist in project 'sdk'.

Cela veut dire qu'il manque le sdk.properties dans le projet sdk
Il faut copier sdk.properties.fast vers sdk.properties (et possiblement modifier des propriétés comme l'ip/username/password ...)
Ensuite, faire un build gradle du projet sdk

S'il y a des : Caused by: org.apache.xerces.impl.io.MalformedByteSequenceException: Invalid byte 2 of 3-byte UTF-8 sequence.

S'assurer que le Workspace qu'utilise Eclipse (Windows/Preference/General/Workspace) est bien en UTF-8 et non cp1252 ou autre.
Par défaut, Eclipse sous Windows est configuré en cp1252.

E. Configuration pour tests Selenium avec Firefox

1. Télécharger et installer la version **23 de Firefox**
2. **S'assurer que Firefox est dans le path du système**
3. **Rouler le test "SeleniumTestFeaturesAcceptanceTest" pour vérifier**

G - Installation de IntelliJ

1. **Télécharger le programme d'installation d'IntelliJ.**
2. **Installer IntelliJ**
3. **Cliquer "Check out project from VCS", choisir Github et entrer vos informations de connexion. (si erreur "Cannot run program "git.exe"" installer git de** http://git-scm.com/download et l'ajouter au path)

4. IntelliJ devrait détecter qu'il s'agit d'un projet Gradle. Fournir le chemin vers votre implémentation Gradle 2 (option "Use local …"), et build.
5. **Facultatif :** Aller dans "Settings" > "Plugins" et chercher Infinitest. Cliquer sur "Install Plugin".
6. **Facultatif :** Aller dans "Project Structure" > "Facets" et ajouter une facet Infinitest liée au projet SDK.
7. Aller dans "Settings" > "Compiler" et cocher "Make projects automatically"
8. Dans "Project Structure" > "SDKs", cliquer sur "Edit" et pointer la SDK sur le répertoire de votre Java local.
9. Cliquer "Configure/Import Settings" puis importer "settings.jar" obtenu dans le Google Drive du projet.

H - Configuration JRebel avec Intellij

1. Aller dans "Settings" > "Plugins" et chercher JRebel. Cliquer sur "Install Plugin".
2. Aller dans "Settings" > "JRebel", cliquer sur "Open activation dialog" et entrer la clé d'activation
3. Aller dans "View" > "Tools Windows" > JRebel
4. Cocher les modules ou JRebel vas fonctionner (app et sdk normallement)
5. Sélectionner un des modules et choisir Open rebel.xml
6. Assurer vous que la valeur du tag <dir name""> est la même que celle retrouvée dans Project structure > Module > [module sélectionner plus haut] > Paths > Output path
7. Dans "Settings" > "Keymap" > "Macros" assurez vous d'avoir save-format-compile mapper sur Ctrl+S

I - Configuration de Solr

1. Demander un package fonctionnel à quelqu'un qui l'a déjà d'installé ou suivre les étapes suivantes.
2. Télécharger le Solr qui se trouve dans le Google Drive (ou prendre la dernière version directement du site http://lucene.apache.org/solr/mirrors-solr-latest-redir.html).
3. Se placer au même niveau que "start.jar" dans le répertoire example de Solr.
4. Téléchargé "intelligid.zip" trouvable dans le Google Drive du projet. (En date du 2015-01-27, https://drive.google.com/a/doculibre.com/file/d/0B-QO2-V0wQkATXBTM2ZReW5KbTQ/view?usp=sharing)
5. Extraire le contenu d'intelligid.zip dans le répertoire example, tel que la relation suivante soit vraie : example/intelligid/solr.xml
6. Ajouter le script clear_and_start dans le dossier example de Solr. Voir section Z1 pour Windows.
7. Dans etc, changer le fichier jetty.xml et modifier <Call name="addConnector"> pour ajouter
   <Set name="requestHeaderSize">6291456</Set>

8. Changer le fichier solrconfig.xml pour chaque core -> <maxBooleanClauses>1000000</maxBooleanClauses>
9. Lancer le script clear_and_start approprié pour votre système d'exploitation.
10. Vérifier que les collections "records" et "vault" sont présente par l'interface Solr à http://localhost:8983/solr/#/

Remarque : En cas de problème avec solr et windows 64 bits, demander une autre version à un collègue.

J - Refresh gradle dependencies in Eclipse

1. Clique droite sur le projet -> Run as -> Gradle build… - > cleanEclipseClasspath puis eclipseclassPath

K - Refresh gradle dependencies in IntelliJ

1. Dans le panneau Gradle, clique sur le bouton refresh

U. Creer une branche sur Github par Eclipse

1. Ouvrir la prespective Git
2. Choisir le repo auquel on veux ajouter une branche
3. Dans la categorie Branches cliquez a droite, choisir Switch to>New Branch…
4. Cliquez a droite sur le repo et choisir Push Branch…
5. Choisir le remote origin: github
6. Cochez Configure upstream for push and pull et Merge upstream commits into local branch
7. Confirmer le push si le push result semble correcte

V. Ajouter une dependance dans Gradle

1. Faire une recherche avec: http://mvnrepository.com/ ou http://search.maven.org/#browse pour trouver le nom de la dépendance pour Gradle (le nom est semblable a 'commons-io:commons-io:2.4')
2. Ajouter le nom trouver plus haut dans le build.gradle du projet qui a cette dépendance, dans la section dependencies,

W. Normes de développement

1. Input et Output Streams
    a. Toujours fermer un Stream à l'endroit où on l'ouvre (dans la même méthode).

    b. Valider que tous les Streams sont fermés dans tous les cas possibles (entre l'ouverture et la fermeture, si une exception est lancée à n'importe quelle ligne, le Stream est quand même fermé).
2. Exceptions
    a. Ne jamais utiliser la même exception pour deux messages différents.
        i. Par exemple, si une méthode peut lancer une exception pour un problème de communication ou un fichier corrompu, elle doit lancer un type d'exception pour chacun.
    b. Toujours fournir l'nformation la plus pertinente possible dans le nom de la classe d'exception, et détaillée dans le message.

X. Règles d'utilisation de Git

1. Si on commit une version qui ne passe pas les tests, on doit l'indiquer dans le message ([#] Tests Fail).
2. Avant de faire un merge, exécuter tous les tests (unitaires et acceptation). Ils doivent passer pour procéder au merge.
3. Avant un merge, le code ne doit contenir aucun warning et respecter checkstyle.
4. On merge au moment où la tâche passe de "In process" à "To verify" sur le taskboard.
5. Au minimum un commit doit être fait pour le développement de chaque classe.
6. Au maximum un commit doit être fait par cycle de TDD.
7. À la fin d'un sprint, on supprime les branches de tâches (on ne garde que la branche du sprint).
8. On ne branch que depuis la branche principale du sprint, et on ne merge que dans celle-ci.
9. Voir les référence suivantes pour mieux comprendre le fonctionnement de Git
    a. The basics : https://www.youtube.com/watch?v=U8GBXvdmHT4 (great intro, but slow pace)
    b. Eclipse basics : https://www.youtube.com/watch?v=1XNVWpjPoio
    c. Revert basics : https://www.youtube.com/watch?v=g5h1VLgC85o
    d. Eclipse merging : https://www.youtube.com/watch?v=r5C6yXNaSGo
    e. Merging : https://www.youtube.com/watch?v=r5C6yXNaSGo
    f. eclipse + git merging (1:31 video): https://www.youtube.com/watch?v=ZK20jVt7XEc
    g. http://stackoverflow.com/questions/14894768/git-fetch-vs-pull-merge-vs-rebase
    h. http://git-scm.com/book/en/v2/Git-Branching-Rebasing
    i. https://www.youtube.com/watch?v=cSf8cO0WB4o
    j.

Y. Avant de commencer à développer

1. Lire Clean Code

2. Lire les slides sur Scrum
3. Lire les slides sur le TDD
4. Faire l'exercice TDD de la Pile (demander à Francis d'expliquer l'exercice)
5. Faire l'exercice TDD du presenter (demander à Francis d'expliquer l'exercice)
6. Lire tous les guides de développement
7. **S'assurer que tous les tests passent sur le poste de travail (en n'activant pas les options pour skipper des tests)**

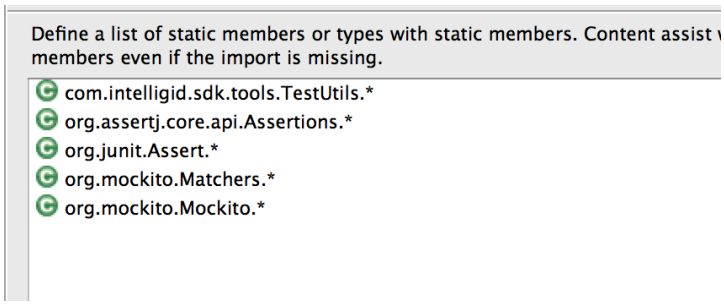
Remarque: si ne trouve pas sdk.properties en créer un en se basant sur sdk.properties.fast (faire une copie)

Z. Configuration des imports

1. Ouvrir Eclipse
2. Window > Preferences > Java > Editor > Content Assist > Favorites
3."New Type" et rentrer :

com.intelligid.sdk.tools.TestUtils
org.assertj.core.api.Assertions
org.junit.Assert
org.mockito.Matchers
org.mockito.Mockito

Vous devrez avoir ceci :



Z1. clear_and_start.bat

```
RD /S /Q %~dp0intelligid\default\data
RD /S /Q %~dp0intelligid\records\data
RD /S /Q %~dp0intelligid\contents\data
RD /S /Q %~dp0intelligid\events\data
RD /S /Q %~dp0intelligid\notifications\data
java -Xmx8096m -Dsolr.solr.home=intelligid -jar start.jar
```