Sommaire

1 Software tester **1**

A Exemple d'une sortie console **9**

Glossaire **17**

Table des matières **19**

Chapitre 1

Software tester

1.1 Travail réalisé

Tout au long de cette mission, ou plutôt ces missions (un test achevé laisse toujours la place à un autre), j'ai travaillé à implémenter des tests en Java permettant de valider automatiquement le comportement de WebI au niveau SDK.

L'objectif principal de cette mission était d'être, à terme, capable d'implémenter seul et dans les plus brefs délais un test automatique. La grande difficulté de début de cette mission a été de comprendre, d'une part, la logique du framework de test, et d'autre part les différentes choses que celui-ci me permettait de faire.

Les concepts à assimiler étaient nombreux et j'ai passé beaucoup de temps à essayer de comprendre le Framework sur lequel repose l'implémentation des tests. Les problèmes majeurs que j'ai rencontrés au début sont les différents points suivants :

— Comment intégrer un test à l'ensemble des tests exécutés ?

— Quel type de test mettre en place, statique ou dynamique ?

— Comment initialiser un test ?

— Comment gérer les sources de telle ou telle version du logiciel pour implémenter un test automatique ?

— Où trouver les fichiers de références nécessaires ?

J'ai éclairci tous ces points au fur et à mesure de mes missions et de mes expérimentations. Certains sont très simples à assimiler mais d'autres m'ont posés beaucoup de problèmes. Je détaille dans les parties suivantes comment est-ce que j'ai été confronté à ces différents problèmes et la manière dont je m'y suis pris pour les résoudre. Dans la partie qui suit je présenterai les différents problèmes que j'ai rencontré et la manière dont je m'y suis pris pour les résoudre. Dans le souci de rester clair dans mes explications je ne m'éparpillerai pas sur tous les tests que j'ai implémenté qui m'ont mis face à tel ou tel problème. Je partirai plutôt d'un cas simple de test automatique en considérant que

tous ces problèmes se sont succédés dans le cadre du même test.

1.1.1 Les recherches relatives aux tests automatiques dans le cadre du Framework utilisé

Ne connaissant rien ni de Web Intelligence ni du Framework de test, je me suis d'abord penché la manière d'utiliser ce Framework pour implémenter un code simple manipulant un document Web Inteliigence.

Pour se faire, j'ai choisi une fonctionnalité très simple, le déplacement d'une cellule dans un tableau, me basant sur un document de test mis à ma disposition (illustration de ce document figure 2).

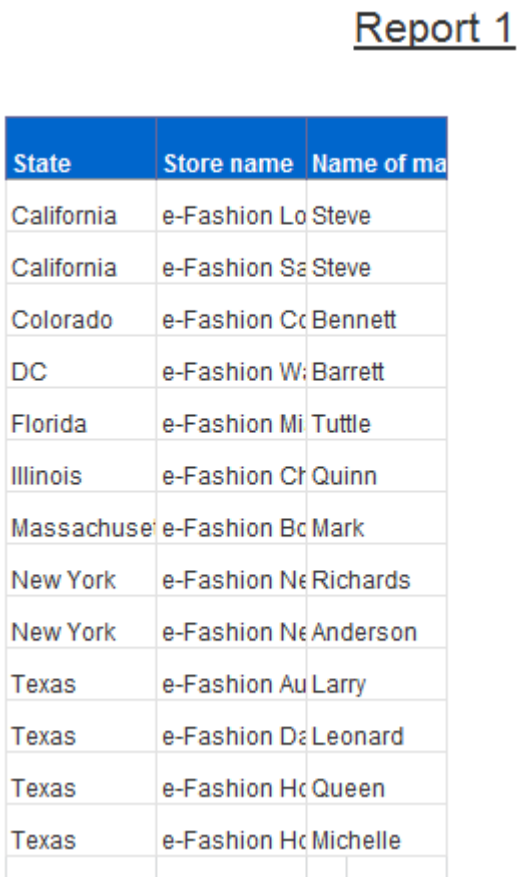

FIGURE 1.1 – Document sur lequel j'ai basé mes première expérimentations

Partant de ce document, comment devais-je m'y prendre pour automatiser une action sur celui-ci ? Pour se faire il me fallait me baser sur une implémentation de test auto-

matique déjà exitante. J'ai donc chercher un code dont la structure me permettait de manipuler un document Web Intelligence déjà existant. Il en existe beaucoup de types différents mais j'ai rapidement trouvé l'exemple d'un test qui se basait sur un document Web Intelligence. J'ai donc créé une nouvelle Classe pour mon test et ai collé le code épuré à l'intérieur, j'avais donc une Classe Java me permettant d'effectuer un test sur mon document. Mais à cette étape j'ai rencontré un certain nombre de problèmes m'empêchant de tester mon code qui ne compilait pas.
J'ai passé beaucoup de temps à chercher les solutions dans les messages d'erreurs que je reçevais en masse.

J'avais pris le soin de coller mon document Web Intelligence précisément dans le dossier qu'il fallait mais pourtant, les messages d'erreur me disait que le document était introuvable. En investiguant tous les documents rentrant en jeu dans l'exécution du test et en réfléchissant à la signification de chacune des variables, je me suis rendu compte que je n'avais pas modifié le « script ID » du fichier script.xml. Celui-ci permet d'identifier un plan de test, qui lui-même identifie une ou plusieurs Classes de test. Une fois la modification effectuée, en prenant soin de correctement renseigner le nom de mon test, je suis tombé une fois de plus sur une erreur similaire : le document demeure introuvable. Le plan de test étant exécuté, cette fois-ci, l'erreur ne pouvait se trouver que dans ma Classe de test. J'ai rapidement identifié le problème puisque c'est dans cette même Classe que le chemin d'accès à mon fichier est renseigné. Les répertoires étant nommés différemment en fonction de la suite de test à laquelle ils appartiennent, et, le test que je voulais exécuter ne faisant pas partie de la même suite, lors de l'exécution Java n'allais pas chercher mon document dans le bon répertoire. J'ai donc corrigé cette erreur et ai renseigné la bonne valeur.
En exécutant de nouveau, je constate qu'il n'y a plus de problèmes d'accès à un fichier. Cette fois-ci les messages d'erreur m'annoncent qu'il est impossible d'accéder au CMS. Je me dirige donc vers le CMS en question pour récupérer son adresse puis vers le fichier parameters.xml dans lequel l'url du CMS est renseignée, effectivement ce n'était pas la même, mais d'autre part en essayant d'accéder au CMS dont il était question je constatais que celui-ci n'existait plus, je ne pouvais pas y accéder. J'ai donc corrigé l'adresse du CMS sur lequel je voulais faire mon test.
En exécutant une fois de plus j'ai été ravi de voir que mon test compilais, celui-ci ne faisait encore rien mais je pouvais alors commencer à manipuler mon document Web Intelligence en implémentant le code de mon test automatique.

La question qui se posa alors était de savoir comment y accéder et de quelle manière m'y prendre pour le modifier.
En parcourant les tests automatiques existants et en cherchant les informations qui me manquaient, j'ai constaté avec surprise que nul part dans le code il n'y avait de commentaires pour expliquer la démarche utilisée ou la logique appliquée. Ce qui m'a troublé, puisque l'on m'avais toujours enseigné que les commentaires sont essentiels dans toute implémentation. J'ai évidemment posé la question pour en connaitre la raison. Les équipes

de développement et de test de Web Intelligence travaillant en suivant la méthode agile, celles-ci sont censées écrire des codes clairs au point que le code lui-même suffit à sa compréhension. Les quelques rares commentaires servant à expliquer le fonctionnement général d'une grande portion de code. Je comprenais ce principe mais j'étais dérangé par le fait que chacun des objets utilisés dans le code m'étaient inconnus, ce qui augmentait beaucoup le temps qu'il me fallait pour comprendre une certaine implémentation.

En cherchant parmi les tests automatiques existants j'ai trouvé le code qui me permettrait de récupérer mon document :

```
IRSReport monRapport = docCalc.getReportAurora(0);
```

L'objet « docCalc » appartenant à la super Classe de ma Classe de test, contient un grand nombre de méthodes permettant de manipuler un document Web Intelligence. L'argument « 0 » que je passe à la méthode me permet de récupérer le premier rapport que contient mon document. Pour compléter ce code ci-dessus et le faire ressembler à test il faut marquer le début et la fin du test. Le code complété étant le suivant :

```
IRSReport reportAurora = docCalc.getReportAurora(0);
startTestcaseStep("monTest");
stopTestcaseStepWithAction();
```

En observant les logs de l'exécution de ce test, on remaque que la première méthode « startTestcaseStep("") ; » permet de marquer le début des tests, celle-ci ne fait rien d'autre que d'ajouter un log. Ensuite, « stopTestcaseStepWithAction() ; » marque non seulement la fin du test dans les logs mais aussi exécute une série d'actions qui sont décritent dans les logs. Il est intéressant de les noter ici car ces actions peuvent nous être très utiles.
L'étude des ces logs m'a été d'une grande aide car ceux-ci décrivent précisément l'exécution du test.

Recherche effectuées pour connaître les différentes actions de l'exécution d'un test

Les logs sont nombreux et j'ai passé beaucoup de temps à les analyser. J'ai pu en conclure les trois étapes essentielles, et évidentes, de l'exécution d'un test : l'avant test, le pendant et l'après. Un exemple d'une de ces sorties consoles que j'ai ainsi étudié est disponible annexe A page 9 dans laquelle j'ai distingué les différentes parties depuis lesquelles les logs étaient générés.
Pour faire ces obsservations j'ai simplement ajouté des logs au moments clés de l'exécution : au début et à la fin du constructeur et de mon test (en distinguant la méthode de test et l'implémentation du test). Cela m'a permis d'observer et ainsi de comprendre les différentes étapes du test tels qu'ils étaient dans le Framework que j'utilisais. Les tests que j'implémentais étaient exécutés dans un cycle propre au Framework, cycle qui m'était inconnu et sans documentation. De cette manière j'ai pu, d'une part, observer

dans quelle exécution s'inscrivait mon test, d'autre part, étudier l'exécution de mon test lui-même et comprendre l'utilité de certaines fonctions. Ci-dessous la liste déscriptive de ce que j'ai pu déduire de cette étude (la méthode « run() » dont je parle et la méthode contenant l'implémentation de mon test automatique) :

1. Avant la méthode « run() »
   - Avant toute chose, ce sont les informations contenues dans le fichier parameters.xml qui sont récupérées, autrement dit, le CMS sur lequel le test devra être effectué
   - Vérifie l'existence du dossier contenant les ressources utilisés dans le cadre de tout test
   - Création du fichier dans lequel seront enregistrés les logs
   - Connexion au serveur (CSM)
   - Recherche le dossier FavoritesFolder sur le CMS
   - Supprime, en local, les dossier où sont enregistrés les résultats (\res\et \ref\)
2. Pendant la méthode « run() » et avant la méthode « stopTestcaseStepWithAction() ; »
   - Recherche de l'univers mentionné dans l'implémentation du test automatique
   - Création d'un nouveau document sur les CMS, celui-ci est suivi du nom que j'ai donné à l'exécution de mon test (« startTestcaseStep("monTest") ; ») de telle sorte que je vais retrouvé sur le CMS le document que j'avais en local.
3. Pendant la méthode « stopTestcaseStepWithAction() ; » (cette méthode étant comprise dans la méthode « run() »)
   - Création du dossier où les résultats seront enregistrés
   - Recherche le dossier « Personnal Documents » sur le CMS
   - Recherche sur le CMS du document concerné par mon test (ce document se retrouve en deux endroits et dans deux états différents, sur le CMS et en local, avant test et après test)
   - Sauvegarde du document en local
4. Après la méthode « run() »
   - Sauvegarde du document dans le personal folder du CMS
   - Copie la référence du cube (référence que j'ai moi-même déposé) vers le dossier \ref\
   - Crée le dossier qui recevra les références
   - Compare le document créé avec la référence

Durant cette partie de mon travail, où l'investigation c'est étalée sur plusieurs semaines et portant sur plusieurs tests à implémenter, j'ai découvert énormément de choses aussi bien en Java que sur l'utilisation du Framework de test. J'ai pu mettre au point des méthodes de résolution de mes problèmes et ai pu découvrir la majeure partie des objets utilisés lors de l'implémentation des tests automatiques de Web Intelligence.

Manipulation des différents objets du Framework

Ayant maintenant étudié toute l'exécution du test, je suis plus à même de situer le test automatique dans sa globalité.
La question qui se pose maintenant est de savoir quoi mettre dans le test, avant de pouvoir se poser la vraie question : « Comment tester automatiquement une fonctionnalité ? ».
Au début, je considérai que toute modification applicable depuis l'interface graphique pouvait être reproduite depuis le test. J'avais tord. Mes tests automatiques étant des tests au niveau SDK, les fonctionnalités auxquelles j'avais accès et que je pouvais tester étaient donc limitées à celui-ci.

Pour en revenir à l'implémentation de mon test de base, devant déplacer une cellule, la manière la plus simlple était de trouver quelque part dans les tests existants comment cette fonctionnalité était utilisée. Je n'ai pas trouvé d'exemple me permettant d'appliquer rapidement ce changement à mon document mais j'ai trouvé des pistes qui paraîssaient prometteuses. La recherche du mot « drop » dans le Framework m'a retourné beaucoup de résultats dont un qui a retenu mon attention, il existe un objet « DropHelperImpl ».
On en déduit assez facilement, à partir de son nom, que celui à quelque chose à voir avec le déplacement d'une cellule. Il me fallait trouver un exemple d'utilisation de cet objet pour faciliter mon travail, fort heureusement il existe une fonctionnalité d'Eclipse que je ne connaissais pas avant (et que, depuis lors, je me suis mis à utiliser systématiquement) : « get call heirarchy », me permettant d'avoir le détail de tous les emplacements où cette Classe est instanciée. De cette manière j'ai rapidement trouvé des exemple de codes me permettant d'en déduire les quelques informations qui m'étaient nécessaires.

J'ai comprise que je ne pouvais pas faire ce que je voulais directement, il me fallait préparer le terrain et récupérer tous les objets qui entraient en jeu dans le déplacement de la cellule.

Il fallait d'abord récupérer, évidemment, le corps de mon document, cela se fait simplement grâce à l'une des Méthodes propres au rapport :

```
1 reportAurora.getPageZone(PageZoneType.BODY);
```

L'étape suivante était de récupérer le tableau contenu dans le corps du document, le problème étant que pour se faire il faut appeler l'objet par son nom. Comment savoir comment ce tableau a été appelé ? Car je pouvais accéder à mon document depuis Web Intelligence mais, depuis l'interface, je n'avais aucun moyen de savoir ce qui se passait derrière, côté logiciel.
La méthode que j'ai utilisé à ce moment là et que j'ai toujours continué à utiliser était de parcourir l'ensemble des éléments contenu dans le corps du rapport et d'en afficher les propriétés. Cela me permettais, non seulement, d'obtenir son nom, son identifiant et tout ses paramètres, mais aussi, d'obtenir son type. Au fur est à mesure du temps et que j'implémentais des tests j'ai découvert beaucoup de types différents, leurs imbriquations, utilités et correspondances avec les éléments graphiques du document Web Intelligence.

Grâce à cette méthode, j'ai pu déterminer le nom de ce que je cherchais.
Ensuite, de cette table il me fallait récupérer les colonnes, pour pouvoir les manipuler. Dans les différents tests que j'avais pu étudier auparavant, j'avais remarqué l'utilisation d'une Méthode générique permettant de récupérer une cellule d'un tableau. J'ai utilisé cette Méthode de la même manière que celle utilisée dans les tests existants.
Alors j'ai pu implémenter le code suivant qui me permettait de déplacer une colonne :

```
IRSPageZone pageZone = reportAurora.getPageZone(PageZoneType.BODY);
IRSReportElement findReportElement = docCalc.findReportElement(pageZone.
    getChildren(), "Block 1");
IRSDynamicBlock table = (IRSDynamicBlock) findReportElement;
IRSCell cell1 = docCalc.getCellFromTable("State",(IRSTable) table);
IRSCell cell2 = docCalc.getCellFromTable("Store name ",(IRSTable) table);
IRSCell cell3 = docCalc.getCellFromTable("Name of manager ",(IRSTable) table);

DropHelperImpl dropHelp = new DropHelperImpl();
List<IRSCell> cells = new ArrayList<IRSCell>();
cells.add(cell1);
cells.add(cell2);
dropHelp.dropCells(cells, cell3, CellZone.RIGHT);
docCalc.applyFormat();
```

L'utilisation de la Méthode « dropCells() » n'est pas instinctive, au premier abord, et j'ai fais plusieurs essais avant de comprendre son fonctionnement. Cette Méthode prends trois paramètres : un tableau de cellules, une cellule et une position. Le changement appliqué par cette Méthode est que le contenu du tableau de cellule est déplacé comme désiré, ici, à droite de la cellule spécifiée.

1.1.2

Annexe A

Exemple d'une sortie console

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Avant la methode run

Check parameters first in: C:\p4\service.td.webi_LVLD60232305A\depot\
    SoftwareTesting\WebISDK\Workspace_Aurora\rebean\parameters.xml
Resources parameters: C:\p4\service.td.webi_LVLD60232305A\depot\SoftwareTesting
    \cube\Resources_AURORA\
IsClientServerDeployment value changed (replaced by parameters): false
Log file.C:\p4\service.td.webi_LVLD60232305A\depot\SoftwareTesting\WebISDK\
    Workspace_Aurora\rebean\logs\jeu_de_test_pierre\TestcaseCreateANewDocument.
    txt=
[<]       -Testcase.java(startMSGStep) : Testcase.java(startTestcase) :
    Testcase TestcaseCreateANewDocument [START]
[O]           -Testcase.java(okMSG) : Testcase.java(startTestcase) : Started at
     Mon Jul 28 14:32:20 CEST 2014   :: OK
[<]           -initSession [START]
[O]               -Session Manager inited   :: OK
[<]               -getClusterName [START]
[>]               -getClusterName [END]
[<]               -getWebiServerName [START]
[O]                   -AdaptiveProcessingServer to be monitored is MySIA.
    AdaptiveProcessingServer   :: OK
[>]               -getWebiServerName [END]
[O]               -Login with user "Administrator" on DEWDFTV01232.DHCP.PGDEV.
    SAP.CORP:6400   :: OK
[>]           -initSession [END]
[<]           -initRepoProxy [START]
[O]               -Repository Proxy inited   :: OK
[>]           -initRepoProxy [END]
[<]           -initInfoStore [START]
[>]           -initInfoStore [END]
[<]           -initDefaults [START]
```

```
[<]               -findInfoObjects [START]
[O]                   -Try to find InfoObjects with name = "Administrator" and
    with type = "FavoritesFolder"   :: OK
[<]                   -executeQuery [START]
[O]                       -Try to execute query 'SELECT * FROM CI_INFOOBJECTS
    WHERE SI_KIND='FavoritesFolder' AND SI_NAME='Administrator''   :: OK
[>]                   -executeQuery [END]
[O]                   -1 InfoObject(s) found   :: OK
[>]               -findInfoObjects [END]
[<]               -executeQuery [START]
[O]                   -Try to execute query 'SELECT * FROM CI_INFOOBJECTS WHERE
     SI_KIND='Folder' AND SI_PARENTID=0'   :: OK
[>]               -executeQuery [END]
[<]               -executeQuery [START]
[O]                   -Try to execute query 'SELECT * FROM CI_INFOOBJECTS WHERE
     SI_ID='18''   :: OK
[>]               -executeQuery [END]
[>]           -initDefaults [END]
[<]           -waitForWebiAlive [START]
[>]           -waitForWebiAlive [END]
[<]           -initWIReportEngine [START]
[O]               -In case of Error, check that sdk.core.config is in classpath
    .   :: OK
[<]               -getGlobalContext [START]
[>]               -getGlobalContext [END]
[O]               -report engine is ready   :: OK
[>]           -initWIReportEngine [END]
[<]           -getConnections [START]
[<]               -executeQuery [START]
[O]                   -Try to execute query 'SELECT SI_ID, SI_NAME FROM
    CI_APPOBJECTS WHERE SI_KIND='CCIS.DataConnection''   :: OK
[>]               -executeQuery [END]
[O]               -34 connection(s) found   :: OK
[>]           -getConnections [END]
[<]           -getUniverses [START]
[<]               -executeQuery [START]
[O]                   -Try to execute query 'SELECT SI_ID, SI_NAME FROM
    CI_APPOBJECTS WHERE SI_KIND='Universe''   :: OK
[>]               -executeQuery [END]
[O]               -26 universe(s) found   :: OK
[>]           -getUniverses [END]
[<]           -getWIReportServerPID [START]
[O]               -Requesting MySIA.WebIntelligenceProcessingServer PID.
    Current time is: Mon Jul 28 14:32:29 CEST 2014   :: OK
[O]               -MySIA.WebIntelligenceProcessingServer PID is: 3132. Current
    time is: Mon Jul 28 14:32:29 CEST 2014   :: OK
[>]           -getWIReportServerPID [END]
[<]           -cleanResultStorage [START]
[O]               -Clear result folder: C:\p4\service.td.webi_LVLD60232305A\
```

```
    depot\SoftwareTesting\WebISDK\Workspace_Aurora\rebean\results\Your Build
    Number\res\jeu_de_test_pierre\TestcaseCreateANewDocument\   :: OK
[<]               -emptyFolder [START]
[<]                   -emptyFolder [START]
[O]                       -clean Folder: C:\p4\service.td.webi_LVLD60232305A\
    depot\SoftwareTesting\WebISDK\Workspace_Aurora\rebean\results\Your Build
    Number\res\jeu_de_test_pierre\TestcaseCreateANewDocument\createdoc   :: OK
[>]                   -emptyFolder [END]
[O]                   -clean Folder: C:\p4\service.td.webi_LVLD60232305A\depot\
    SoftwareTesting\WebISDK\Workspace_Aurora\rebean\results\Your Build Number\
    res\jeu_de_test_pierre\TestcaseCreateANewDocument\   :: OK
[>]               -emptyFolder [END]
[O]               -Clear result folder: C:\p4\service.td.webi_LVLD60232305A\
    depot\SoftwareTesting\WebISDK\Workspace_Aurora\rebean\results\Your Build
    Number\ref\jeu_de_test_pierre\TestcaseCreateANewDocument\   :: OK
[<]               -emptyFolder [START]
[<]                   -emptyFolder [START]
[O]                       -clean Folder: C:\p4\service.td.webi_LVLD60232305A\
    depot\SoftwareTesting\WebISDK\Workspace_Aurora\rebean\results\Your Build
    Number\ref\jeu_de_test_pierre\TestcaseCreateANewDocument\createdoc   :: OK
[>]                   -emptyFolder [END]
[O]                   -clean Folder: C:\p4\service.td.webi_LVLD60232305A\depot\
    SoftwareTesting\WebISDK\Workspace_Aurora\rebean\results\Your Build Number\
    ref\jeu_de_test_pierre\TestcaseCreateANewDocument\   :: OK
[>]               -emptyFolder [END]
[>]           -cleanResultStorage [END]
[<]           -newAuroraDocument [START]
[>]           -newAuroraDocument [END]


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Pendant la methode run


[<]           -updateStructureAurora [START]
[>]           -updateStructureAurora [END]
[<]           -Testcase step: createdoc [START]
[<]               -newDocument based on unv BEACH [START]
[<]                   -findUniverseInRoot [START]
[O]                       -Try to find InfoObjects with name = "BEACH" and with
     type = "Universe"   :: OK
[<]                       -executeQuery [START]
[O]                           -Try to execute query 'SELECT SI_ID FROM
    CI_APPOBJECTS WHERE SI_NAME='Universes' AND SI_KIND='Folder' AND
    SI_PARENTID=95'   :: OK
[>]                       -executeQuery [END]
[O]                       -1 InfoObject(s) found   :: OK
[<]                       -executeQuery [START]
```

```
[O]                          -Try to execute query 'SELECT * FROM
    CI_APPOBJECTS WHERE SI_KIND='Universe' AND SI_NAME='BEACH' AND SI_PARENTID
    =532'   :: OK
[>]                      -executeQuery [END]
[O]                      -1 InfoObject(s) found   :: OK
[>]                  -findUniverseInRoot [END]
[<]                  -createUniverseDataSourceInfo unv cuid Af7KxGg_CDpFp.Uq.
    gt3D58 [START]
[O]                      -- [DSL] create DataSourceInfo : OK   :: OK
[>]                  -createUniverseDataSourceInfo unv cuid Af7KxGg_CDpFp.Uq.
    gt3D58 [END]
[<]                  -DSLaddDataProvider [START]
[O]                      -- [DSL] add new data provider : OK   :: OK
[>]                  -DSLaddDataProvider [END]
[O]                  -add a simple flat query   :: OK
[>]              -newDocument based on unv BEACH [END]
[<]              -createAuroraReport My Report [START]
[O]                  -report.setName My Report   :: OK
[O]                  -_repStruct.getChildren() - size: 0   :: OK
[>]              -createAuroraReport My Report [END]
[<]              -createReportBody(Aurora) [START]
[>]              -createReportBody(Aurora) [END]
[<]              -applyFormat [START]
[>]              -applyFormat [END]


%%%%%%%%%%%%%%%%%%%%%%%% Pendant la methode StopTestcaseStepWithAction()


[<]              -applyFormat [START]
[>]              -applyFormat [END]
[<]              -verifyWithRef - PageMode Listing [START]
[<]                  -prepareResFiles [START]
[<]                      -createPath [START]
[O]                          -Directory created: C:\p4\service.td.
    webi_LVLD60232305A\depot\SoftwareTesting\WebISDK\Workspace_Aurora\rebean\
    results\Your Build Number\res\jeu_de_test_pierre\TestcaseCreateANewDocument
    \createdoc   :: OK
[>]                      -createPath [END]
[<]                      -checkSupportedViews [START]
[>]                      -checkSupportedViews [END]
[<]                      -saveAsXMLaurora [START]
[<]                          -saveReportAsCharacterViewAurora [START]
[O]                              -set LISTING for pagination mode   :: OK
[O]                              -navigate to report '1'   :: OK
[O]                              -get CharacterView view for report: My Report
       :: OK
[>]                          -saveReportAsCharacterViewAurora [END]
```

```
[O]                          -save as xml: C:\p4\service.td.webi_LVLD60232305A
    \depot\SoftwareTesting\WebISDK\Workspace_Aurora\rebean\results\Your Build
    Number\res\jeu_de_test_pierre\TestcaseCreateANewDocument\createdoc   :: OK
141 [>]                       -saveAsXMLaurora [END]
    [<]                       -XML2TXT [START]
143 [>]                       -XML2TXT [END]
    [<]                       -saveDocInPersonal [START]
145 [<]                           -search (Folder) [START]
    [<]                               -findInfoObjects [START]
147 [O]                                   -Try to find InfoObjects with name = "
    Personal Documents" and with type = "Folder"   :: OK
    [<]                                   -executeQuery [START]
149 [O]                                       -Try to execute query 'SELECT * FROM
    CI_INFOOBJECTS WHERE SI_KIND='Folder' AND SI_NAME='Personal Documents''
    :: OK
    [>]                                   -executeQuery [END]
151 [O]                                   -1 InfoObject(s) found   :: OK
    [>]                               -findInfoObjects [END]
153 [>]                           -search (Folder) [END]
    [<]                           -saveDocument_aurora [START]
155 [<]                               -search (Folder) [START]
    [<]                                   -findInfoObjects [START]
157 [O]                                       -Try to find InfoObjects with name =
    "Personal Documents" and with type = "Folder"   :: OK
    [<]                                       -executeQuery [START]
159 [O]                                           -Try to execute query 'SELECT *
    FROM CI_INFOOBJECTS WHERE SI_KIND='Folder' AND SI_NAME='Personal Documents'
    '   :: OK
    [>]                                       -executeQuery [END]
161 [O]                                       -1 InfoObject(s) found   :: OK
    [>]                                   -findInfoObjects [END]
163 [>]                               -search (Folder) [END]
    [<]                               -search (Document) [START]
165 [<]                                   -search (Folder) [START]
    [<]                                       -findInfoObjects [START]
167 [O]                                           -Try to find InfoObjects with
    name = "Personal Documents" and with type = "Folder"   :: OK
    [<]                                           -executeQuery [START]
169 [O]                                               -Try to execute query 'SELECT
     * FROM CI_INFOOBJECTS WHERE SI_KIND='Folder' AND SI_NAME='Personal
    Documents''   :: OK
    [>]                                           -executeQuery [END]
171 [O]                                           -1 InfoObject(s) found   :: OK
    [>]                                       -findInfoObjects [END]
173 [>]                                   -search (Folder) [END]
    [<]                                   -findInfoObjects [START]
175 [O]                                       -Try to find InfoObjects with name =
    "TestcaseCreateANewDocument_createdoc" and with type = "Webi"   :: OK
    [<]                                       -executeQuery [START]
```

```
177 [O]                                          -Try to execute query 'SELECT *
    FROM CI_INFOOBJECTS WHERE SI_KIND='Webi' AND SI_NAME='
    TestcaseCreateANewDocument_createdoc''   :: OK
    [>]                                     -executeQuery [END]
179 [O]                                     -1 InfoObject(s) found   :: OK
    [>]                                 -findInfoObjects [END]
181 [>]                             -search (Document) [END]
    [O]                             -Document
    TestcaseCreateANewDocument_createdoc saved in Personal Documents   :: OK
183 [>]                         -saveDocument_aurora [END]

185


187


189 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Apres la methode run

191


193 [O]                         -document saved in default personal folder:
    TestcaseCreateANewDocument_createdoc   :: OK
    [>]                     -saveDocInPersonal [END]
195 [>]                 -prepareResFiles [END]
    [<]                 -prepareRefFiles [START]
197 [<]                     -getFileNamesWithExtension [START]
    [O]                         -Get 1 files with extension "txt"   :: OK
199 [>]                     -getFileNamesWithExtension [END]
    [<]                     -copyFile [START]
201 [O]                         -copy file from C:\p4\service.td.
    webi_LVLD60232305A\depot\SoftwareTesting\cube\Resources_AURORA\storage\auto
    \jeu_de_test_pierre\TestcaseCreateANewDocument\txt\
    TestcaseCreateANewDocument_createdoc1.txt to C:\p4\service.td.
    webi_LVLD60232305A\depot\SoftwareTesting\WebISDK\Workspace_Aurora\rebean\
    results\Your Build Number\ref\jeu_de_test_pierre\TestcaseCreateANewDocument
    \createdoc\TestcaseCreateANewDocument_createdoc1.txt in 0s   :: OK
    [>]                     -copyFile [END]
203 [>]                 -prepareRefFiles [END]
    [<]                 -createPath [START]
205 [O]                     -Directory already exists: C:\p4\service.td.
    webi_LVLD60232305A\depot\SoftwareTesting\WebISDK\Workspace_Aurora\rebean\
    results\Your Build Number\ref\jeu_de_test_pierre\TestcaseCreateANewDocument
    \createdoc   :: OK
    [>]                 -createPath [END]
207 [<]                 -getFileNamesWithExtension [START]
    [O]                     -Get 1 files with extension "txt"   :: OK
209 [>]                 -getFileNamesWithExtension [END]
    [<]                 -comparePreparedFiles [START]
211 [<]                     -compareTXTFiles [START]
    [O]                         -C:\p4\service.td.webi_LVLD60232305A\depot\
```

```
    SoftwareTesting\WebISDK\Workspace_Aurora\rebean\results\Your Build Number\
    res\jeu_de_test_pierre\TestcaseCreateANewDocument\createdoc\
    TestcaseCreateANewDocument_createdoc1.txt and C:\p4\service.td.
    webi_LVLD60232305A\depot\SoftwareTesting\WebISDK\Workspace_Aurora\rebean\
    results\Your Build Number\ref\jeu_de_test_pierre\TestcaseCreateANewDocument
    \createdoc\TestcaseCreateANewDocument_createdoc1.txt are SAME   :: OK
213 [>]                       -compareTXTFiles [END]
    [>]                   -comparePreparedFiles [END]
215 [>]               -verifyWithRef - PageMode Listing [END]
    [>]           -Testcase step: createdoc [END]
217 [<]           -closeDocument [START]
    [>]           -closeDocument [END]
219 [<]           -closeAll [START]
    [>]           -closeAll [END]
221 [<]           -logoff [START]
    [0]               -close report engines   :: OK
223 [>]           -logoff [END]
    [<]           -logoff [START]
225 [0]               -Close session   :: OK
    [>]           -logoff [END]
227 [0]           -Total Time Spent for Testcase TestcaseCreateANewDocument =
    22.083 second(s)   :: OK
    [0]           -Finished at Mon Jul 28 14:32:42 CEST 2014   :: OK
229 [>]       - Testcase TestcaseCreateANewDocument [END]
    [TestcaseCreateANewDocument]    --> PASSED : 'RebeanTestPlanDefinition.java(
    StressPF) : Min=1.4065508E12|Max=1.4065508E12|Average=1.4065508E12|Count=1|
    CPUTime=0|PeakVM=0|LeakVM=0'
231 Passed.TestcaseCreateANewDocument=
```

Glossaire

CMS Le . 3

Table des matières

1 Software tester **1**
1.1 Travail réalisé 1
1.1.1 Les recherches relatives aux tests automatiques dans le cadre du Framework utilisé 2
1.1.2 7

A Exemple d'une sortie console **9**

Glossaire **17**

Table des matières **19**