LA DOCUMENTATION DES CLASSES

Trigger Name: OrderTriggers.

Fonctionnalité: C'est un déclencheur (trigger) qui se déclenche avant la mise à jour et après l'insertion ou la suppression d'un enregistrement Order.

- "handleBeforeUpdate" Si le déclencheur est déclenché par une mise à jour,il appelle

la méthode de la classe "OrderTriggersHandler".

- la methode "handlerAfterDelete":Si le déclencheur est déclenché par une insertion ou une suppression,il appelle la méthode de la classe "OrderTriggersHandler".
- En cas d'erreur, il affiche un message de débogage.

Classe Name : OrderTriggersHandler.

Fonctionnalités :

Cette classe contient deux méthodes :

- "handleBeforeUpdate" : Cette méthode est appelée avant la mise à jour des enregistrements Order.
  Elle vérifie si le nouveau statut de l'Order est "Activated" et que l'ancien statut n'était pas "Activated".Si c'est le cas :
  elle vérifie si l'Order a des produits associés.
  Si l'Order n'a pas de produits associés, elle ajoute une erreur.
  Sinon, elle ajoute l'Order à la liste des Orders à mettre à jour.
  Si tous les Orders ont des produits associés, elle les met à jour.
  En cas d'erreur, elle affiche un message de débogage.
- "hasAssociatedProducts" : Cette méthode vérifie si un Order a des produits associés. Elle retourne "True" s'il y a des produits associés et "False" sinon.

- "handlerAfterDelete" : Cette méthode est appelée après la suppression d'un enregistrement Order.

  Elle récupère tous les Accounts, vérifie s'ils ont des Orders associés et met à jour leur champ "Active__c" en conséquence
  En cas d'erreur, elle affiche un message de débogage.

LA DOCUMENTATION DES CLASSES

class Name :OrderTrrigerHandlerTest

fonctionnalité : c' est une class de tests unitaires pour la classe OrderTriggersHandler.
Ces tests permettent de vérifier le comportement de la classe OrderTriggersHandler
dans différentes situations, en s'assurant que les méthodes de la classe fonctionnent
correctement et renvoient les résultats attendus.

La classe de test orderTriggersHandlerTest contient trois méthodes de test annotées @isTest, chacune testant une méthode différente de la classe OrderTriggersHandler.

Méthodes de test :

- Le premier test, handlerAfterDeleteWithproductTest, crée 50 comptes et 50 commandes associées pour chaque compte, définit le statut de toutes les commandes comme 'Activated', appelle la méthode handlerAfterDelete de la classe OrderTriggersHandler, et vérifie que le champ Active__c de chaque compte est vrai.

- Le deuxième test, handlerAfterDeleteWithNoproductTest, crée 50 comptes sans aucune commande associée, crée une liste de commandes vide, appelle la méthode handlerAfterDelete de la classe OrderTriggersHandler, et vérifie que le champ Active__c de chaque compte est faux.

- Le troisième test, handleBeforeUpdateTest, crée 50 comptes et 50 commandes associées pour chaque compte, définit le statut de toutes les commandes comme 'Activated', met à jour les commandes, appelle la méthode handleBeforeUpdate de la classe OrderTriggersHandler, et vérifie que le statut de la commande est maintenant 'Activated'.

Class Name : CreateReminderBatch.

Focntionnalité :c'est une implémentation de l'interface Database.Batchable<sObject>, qui permet de traiter de gros volumes de données en mode asynchrone. Elle définit trois méthodes : start, execute et finish.

Methodes :

- La méthode start retourne un objet de type Database.QueryLocator qui permet d'identifier les enregistrements à traiter. la méthode interroge les comptes de l'organisation et leurs commandes, ainsi que les tâches d'appel non démarrées, en utilisant une requête SOQL. Les résultats sont renvoyés sous forme de QueryLocator.
- La méthode execute est appelée pour chaque lot d'enregistrements renvoyé par la méthode start. la méthode crée des tâches d'appel pour les comptes qui n'ont pas de commande et n'ont pas déjà de tâches d'appel non démarrées. Les nouvelles tâches sont stockées dans une liste et insérées à la fin de la méthode.

- La méthode finish est appelée une fois que tous les lots ont été traités. elle est vide, car aucune action supplémentaire n'est requise à ce stade.

Class Name : CreateReminderBatchShledule.

Fonctionnalité :c' est une implémentation de l'interface Schedulable, qui permet de planifier l'exécution d'une classe ou d'une méthode à une date et une heure spécifiques. Elle définit une méthode execute qui appelle la classe CreateReminderBatch en tant que batch.

Les methodes :

La méthode execute est appelée par le planificateur pour exécuter la tâche planifiée. la méthode crée une instance de la classe CreateReminderBatch et exécute la tâche en tant que batch en appelant la méthode executeBatch de la classe Database. Si une exception est levée lors de l'exécution, le message d'erreur est enregistré dans le journal de débogage.

Class Name "CreateReminderBatchTest :

Fonctionnalités :Cest une classe de test pour la classe Apex "CreateReminderBatch" qui implémente l'interface "Database.Batchable" pour créer des tâches de suivi pour les comptes Salesforce.

Il y a deux methodes tests dans cette classe.

- Le premier test, "TestCreateReminderBatchWithOrders", vérifie que des tâches sont créées uniquement pour les comptes qui n'ont ni commandes ni tâches existantes. Pour cela, il crée d'abord 80 comptes et 1 commande pour chaque compte. Il exécute ensuite le job batch "CreateReminderBatch" à l'aide de la méthode "Database.executeBatch". Enfin, il vérifie que les tâches ont été créées uniquement pour les comptes qui n'ont ni commandes ni tâches existantes.

- Le deuxième test, "TestCreateReminderBatchWithNoOrders", vérifie que des tâches sont créées pour tous les comptes qui n'ont pas de commande. Pour cela, il crée d'abord 80 comptes sans commande. Il exécute ensuite le job batch "CreateReminderBatch" à l'aide de la méthode "Database.executeBatch".

Enfin, il vérifie que chaque compte sans commande a bien une tâche créée pour lui. Les

deux tests utilisent la méthode "TestDataFactory" pour créer des données de test.

Class Name : CreateReminderBatchShleduleTest .

Fonctionnalités :Cette classe de test teste la planification d'un travail par lots à l'aide de la classe CreateReminderBatchShledule.

Elle contient une seule methode de test :

- La méthode ReminderBatchShleduleTest crée 80 comptes et 1 commande pour chaque compte à l'aide de la classe TestDataFactory.

La méthode démarre ensuite un test, planifie l'exécution du travail par lots tous les lundis du mois à 23 h 00 à l'aide de la méthode System.schedule et arrête le test.

la méthode interroge les tâches avec le sujet "Call" et vérifie qu'aucune n'a été créée.

Le but de cette classe de test est de s'assurer que le travail planifié s'exécute comme prévu sans créer de tâches indésirables. Il est important de noter que cette classe de test ne teste pas la fonctionnalité réelle du travail par lots, mais plutôt sa planification.

Classe Name : TestDataFactory .

Fonctionnalités :Cette classe est responsable de la création des données de test pour l'application.

Il inclut des méthodes de création d'enregistrements "Compte" et "Commande" à des fins de test.

De plus, il crée les enregistrements Product et PricebookEntry requis pour créer un enregistrement Order.

La classe TestDataFactory est utilisée pour créer des données de test pour l'application, elle inclut des méthodes pour créer des enregistrements "Compte" et "Commande" pour les tests, ainsi que des enregistrements Product et PricebookEntry nécessaires pour créer des enregistrements Order.