Coding Challenge Ángel Vela Martínez

2 JUNE

adidas

Created by Ángel Vela

Introduction

Objetive

The objective of this document is the technical detail's itemization of the proposed exercises in the Coding Challenge by Adidas. In the next pages these exercises had been detailed one by one, quoting the elements which are used the implementation.

Nomenclature

A standardize naming has been used as good practise, based on the use of an item's abbreviation before the custom component's name. Example: **CO_** for Custom Object or **AC_** for Apex Class.

Index

Exercise 1

Objetive

Given two custom objects (Student & Mark) related with a lookup field, calculate the average mark of the student considering that the mark can be added, modified, or deleted.

Custom Components

Custom Objects	
Name	**API Name**
Student	CO_Student
Mark	CO_Mark

Custom Fields			
Name	**API Name**	**Object**	**Type**
Average Mark	CF_AverageMark__c	CO_Student	Number
Student	CF_Student__c	CO_Mark	Lookup (CO_Student)
Mark	CF_Mark__c	CO_Mark	Number

Custom Apex Triggers				
Object	**API Name**	**Description**	**Events**	**Invoked methods**
CO_Mark__c	AT_Mark	Trigger launched when a Mark is created, updated or deleted. This trigger refresh Average Mark's values in the related Student record.	After Insert After Update After Delete	AC_MarkHandler > AC_MarkUtils > refreshAverageMark

Custom Apex Classes

1. AC_MarkHandler

This class is invoked from CO_Mark trigger (AT_Mark) to handle complex logic used in mark processing.

Methods		
Name	**Description**	**Invoked methods**
OnAfterInsert	Invoke the methods which apply after a mark's insertion, as the average mark's refresh.	AC_MarkUtils > refreshAverageMark
OnAfterUpdate	Invoke the methods which apply after a mark's edition, as the average mark's refresh.	AC_MarkUtils > refreshAverageMark
OnAfterDelete	Invoke the methods which apply after a mark's deletion, as the average mark's refresh.	AC_MarkUtils > refreshAverageMark

2. AC_MarkUtils

This class contains reusable logic for Mark records management, which could be invoked from any platform class.

Methods		
Name	**Description**	**Invoked methods**
refreshAverageMark	This class calculates the average of mark's values from a student when one of his marks was created, updated or deleted.	

3. AC_MarkUtilsTest

This test class covered the code used in AT_Mark, AC_MarkHandler and AC_MarkUtils.

Methods		
Name	**Description**	**Invoked methods**
refreshAverageMarkTest	This method covers the main functionability of average mark's refresh executed by AT_Mark trigger.	AT_Mark > AC_MarkHandler > AC_MarkUtils > refreshAverageMark

Exercise 2

Objetive

Create a new apex class to given a disorganised list of contacts, where one of the fields contains a numeric value (CF_Value__c), return the same list but with interlocking and incremental odd and even values, starting with an even value.

Custom Components

Custom Fields			
Name	**API Name**	**Object**	**Type**
Value	CF_Value__c	Contact	Decimal

Custom Apex Classes

1. AC_ContactOrganizer

This class has the responsibility of organise contacts by Value field interlocking and incremental odd and even values, starting with an even value.

Methods		
Name	**Description**	**Invoked methods**
contactOrganizer	Method which organises contacts interlocking and incremental odd and even values.	AC_ContactWrapper > compareTo

2. AC_ContactWrapper

This global class contains reusable logic for organise contacts incrementally using Comparable interface implementation, adding sorting support for lists of this type of records. This class is invoked when a list of records of this class is sorted.

Methods		
Name	**Description**	**Invoked methods**
AC_ContactWrapper	Constructor which uses a Contact input variable	
compareTo	Method inherited of Comparable interface	

3. AC_ContactOrganizer_Test

This test class covered the code used in AC_ContactOrganizer and AC_ContactWrapper.

Methods		
Name	**Description**	**Invoked methods**
contactOrganizerTest	This method covers the main functionability of AC_ContactOrganizer and AC_ContactWrapper.	AC_ContactOrganizer > contactOrganizer > AC_ContactWrapper

Exercise 3

Objetive

Given a text as an input, return a map<String,Integer> with the number of times each word is used.

Custom Apex Classes

1. AC_WordMeter

This class gets a count of each word which are contained in a text.

Methods		
Name	**Description**	**Invoked methods**
wordMeter	A method which refines an input text and count each word which are contained in it.	

2. AC_WordMeter_Test

This test class covered the code used in AC_WordMeter.

Methods		
Name	**Description**	**Invoked methods**
wordMeterTest	This method covers the main functionability of AC_WordMeter.	AC_WordMeter > wordMeter

Exercise 4

Objetive

Given a text as an input, return a map<String,Integer> with the number of times each word is used.

Custom Components

Custom Lightning Web Components		
Name	**Description**	**Invoked methods**
myComponent	Custom search bar which can find accounts, contacts, campaigns, leads and opportunities from a word	AC_MyApexController > searchRecords

Custom Apex Classes

1. AC_MyApexController

Apex controller which gives the search power to myComponent LWC.

Methods		
Name	**Description**	**Invoked methods**
searchRecords	A method which searches records related to the term inserted in myComponent LWC and return a sObject's list with the search result.	

2. AC_MyApexController_Test

This test class covered the code used in AC_MyApexController.

Methods		
Name	**Description**	**Invoked methods**
searchRecordsTest	This method covers the main functionability of AC_MyApexController.	AC_MyApexController > searchRecords