Code documentation for "Mini project 2"

This time the task was to update the previous version of the inventory register to work with a database in the background via Entity Framewrok (EF). I have chosen to create as general a database structure as possible so that users have the opportunity to add different types of categories with different extra properties such as e.g. smart phone.

Furthermore, I have tried to get it as normalized as possible in the database via the classes, all to facilitate in database administration but also in the extension code. At the time of writing, it is not possible to edit properties for a category or add a new one. The goal is for this to be added in the near future, so there may be some extra code for just this. Have had some code crashes, so to speak, so I do not have a very good idea of the code.

I chose to start with the basis of the code for version 1 of the code, so here in retrospect it would have been smarter to start over from scratch and snot functions etc. that were needed in this new code. It is always a chance to do this, with a little luck you can save some jobs.

Below are the various folders and files and their functions included in them.

Classes

Folder that contains all definitions for all classes used by the program. What is common to all classes is in the base class Inventory.

Category

Category

A collection class for managing different properties for the category class. There are links to category properties and texts.

CategoryText

An intermediate class for handling the name of the inventory group, also has handling about the connection between different extra properties with associated texts.

CategoryProp

A class for handling different properties for the linked inventory category.

Inventory

Inventory

Class that acts as the main class for an inventory, has connections to miscellaneous under class everything to normalize the information.

InventoryExtraData

An underclass to save unique properties for that particular inventory.

MasterSettings

MasterSettings

Class that indicates some general settings in the service.

MastersettingsInventory

Class to make some default settings about how inventory ID should be designed.

ValidCurrency

Class to handle valid currencies in the program, of course we could use the information from freecurrency therapy instead of hard-coding these currencies that are included in the service. Currency conversion came in a little later in the coding and I feel that there is no point in adjusting this as it will not be a commercial product.

ValidOffice

Class to handle our various offices in the service

Control

CheckItemsOffice

CheckItemsOffice ( )

Out parameter: True or False

Function for checking if an office has some connected equipment when closing an office. False returns if there is connected equipment connected to the office.

CheckSyntax

A class with a number of common private variables and private functions to verify the syntax for different SKUs or category ids that exist.

ClassExist (string className)

A parameter

string className, the class name we are going to verify exists

Out parameter:

Bool, true if the class exists, otherwise false.

convertFieldsToDictonary ( )

A parameter: None a parameter

Out parameter: No out parameter

Converts a fieldInfo variable type to a dictonary to facilitate further coding.

checkForceHyphen ( )

A parameter: None a parameter

From parameter: bool

Function to verify if received string to verify contains the correct number of hyphens, etc. Returns true if everything is okay, otherwise false.

checkMinLengthLeftSide ( )

A parameter: None a parameter

Ut bool

Function to verify the minimum length of category id or for the left part if we have an article number that consists of two different parts that are tied together with a hyphen.

check (string className = null, string InputString = null)

In parameters:

className classname that we work by and its associated parameters

InputString close as we check the syntax

From parameter: bool

Main function for verifying the syntax for a category ID or inventory ID that we want to verify. Returns true if all conditions are correct otherwise false.

InventoryContext

Class to create a kind of "connection" to the database. There are also some other independent connections for different classes as it has been easier in terms of code

Here are also the settings for connecting to the database.

RemoveInventory

Function for deleting an inventory, starts with deleting the unique properties of that particular inventory before the inventory itself is deleted.

Migrations

Catalog of database migrations used internally by Visual Studio

Model

CommonFunction

Class to handle various common functions in the service.

FirstLetterToUpper (string str)

In parameter: String that should be capitalized on the first character in the string

Out parameter: Close with the first character as a capital letter

Function for capitalizing the first character in the string that enters the function.

GetMonthsBetween (DateTime from, DateTime to)

A parameter:

DateTime from. Start date that we will compare from

DateTime to: End date to compare with

Out parameter: The difference between two incoming dates in months.

GetDifferenceInYears (DateTime startDate, DateTime endDate)

A parameter:

DateTime startDate, start date being compared

DateTime endDate end date that we will compare against

Out parameter: Number of years between the specified dates

Function that calculates the number of years between two dates and takes into account the month.

ExchangeRates

Class for retrieving the current exchange rates online, ie the rates apply internationally and not at the companies that exchange currency.

getExchangeRates ( )

A parameter: None a parameter

Out parameter: No out parameter

Trying to retrieve the current exchange rates and put these in a public variable that can be used in different places in the service.

MoveInventorysToOffice

Class for moving equipment connected from one office to another.

moveInventorysToOffice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for moving all equipment in one office to another, uses global memory variable to keep track of from which office the equipment should be moved.

SaveEditInventorys

Class to save the edited inventory to the database.

saveEditInventory ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that updates the information in our register based on information in global memory variables.

SaveEditOffice

Class to save office editing.

saveEditOffice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function to save editing properties around an office.

SetupDefaultOffice

Class for setting up a number of standard offices if there are none already in the service, as well as some standard data and categories etc.

setupDefaultOffice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that sets up a number of standard offices in the service, etc.

View

AddInventory

Class to add inventory to our register.

showInputInventoryID ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for presenting which inventory ID to enter for the product. It retrieves the highest ID found in the register and adds 1 if possible.

showPurchaseDate ( )

A parameter: None a parameter

Out parameter: No out parameter

Function to present to the user that he must enter the date of purchase of the equipment. Uses old value if possible, otherwise today's date.

showBrand ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for specifying the manufacturer of the equipment, if possible using previously used manufacturers from the global memory variable.

showModel ( )

A parameter: None a parameter

Out parameter: No out parameter

Function to specify model of inventory, if possible use previously used model if the global variable is set!

showOffice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for indicating which office the inventory is located in. Uses value from global memory if the variable is set!

showPurchasePrice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for entering the purchase price in the currency that applies to the particular office. Uses value from global memory if the variable is set!

showNotebook ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that is called if you enter a computer in the inventory register. It asks if it is a laptop, the default is based on global memory variable.

showDualSIM ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that is called if you enter a mobile phone in the inventory register. It asks if it supports dual SIM cards, the default is based on global memory variable

showPrintPin ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that is called if you enter a printer in the inventory register. It asks if it supports printing PIN code

to then retrieve the print queue from the print server, the default is based on global memory variable

GetProperties ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for retrieving all defined variables in a class, as well as the base class itself. Used to build the order in which different input options are presented to the user.

showAddArticle ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that presents the different options when an inventory is to be registered. The different options presented are based on the different variables that exist in the different classes.

resetMemParameters ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that resets some common memory parameters. Called when we switch between different types of equipment to be registered in the service.

addArticle ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that is called when we are to add new inventory to our register. The first thing it asks of the user is which category of inventory to register.

EditInventory

Class for editing inventory in our register.

editInputInventoryID ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for changing inventory ID on a product.

editInventory ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that calls up some sub-functions when we have to edit an inventory in the system. This is based on the various input variables that belong to the selected inventory ID.

ListInventorys

Class used to list our various inventory by selected sort from the user.

listInventorys ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that presents options for how we want to sort our equipment in the service.

Based on the choice, it asks various LAMDA questions whose results are saved in a variable. For the sake of readability of the code, the print itself is in a separate function.

ManageInventoyryCategory

Class with functions for editing different properties for an inventory group. All code is not complete yet, just get the right side of some bugs around the database and categories.

editInventoryCategoryName ( )

A parameter: None a parameter

Out parameter: No out parameter

Function to ask the user to put a new name on the inventory group.

editInventoryProperties ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for editing different properties for all inventory categories.

editInvenyotyCategory ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for selecting which inventory category to edit.

addInvenyotyCategory ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for adding a new inventory category to our register.

deleteInvenyotyCategory ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for deleting an inventory category in our register

manageInventoyryCategory ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for building up a menu system to influence different properties for the inventory categories.

ManageOffice

Class with functions for managing the various offices in the service.

addOffice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for adding a new office, it is the very name of the office that is meant.

showCurrency ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for specifying which currency applies to that particular office.

editOffice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for selecting an office to edit.

showValidateEmptyOffice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that is called if an office to be deleted has equipment, you are asked to specify which office the equipment is to be transferred to.

deleteOffice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function called when an office is to be deleted. Request information from the user which office is deleted from the inventory register.

showCountry ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that asks the user which country the office should be connected to.

editOfficeMenu ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for calling the various sub-functions when we are editing an office.

showAddOffice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for calling the various sub-functions when we are adding an office.

manageOffice ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that presents a menu system for managing everything around offices.

Menu

Class to present our menu system for the service.

displayMenu ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for presenting a flexible menu system as it is based on a list is defined in the function.

RemoveInventory

Class to present information about deleting an inventory.

removeInventory ( )

A parameter: None a parameter

Out parameter: No out parameter

Function to present which equipment is to be deleted. Perform some checks around the entered string.

SearchInventory

Class to search for certain text string in our article index.

searchInventory ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for presenting which string to search for and also performs the actual search and saves the result in a variable to be presented to the user later.

ShowListInventorys

Class for presenting the contents of incoming variables with data.

showListInventorys (List <Inventory> sortedListCategory, bool showReadKey = true)

A parameter:

sortedListCategory the list to be presented to the user

showReadKey if we are to present a string that you have to press a key to move on.

Out parameter: No out parameter

Function that presents a list of all equipment that has entered the function. Marks equipment that is beginning to approach the position for exchange.

Program

Main file for the program itself. Initially, a number of global variables used by the program are defined in different places

Main ( )

A parameter: None a parameter

Out parameter: No out parameter

The main function itself that is called when starting the program. Starts with downloading the exchange rates, then reads different classes from the hard drive. Finally, the menu for the service is presented.