**Ejercicio 1**

Listar los dias del año actual que correspondan a fines de semana

Formato del listado

Fines de semana del año yyyy

Fin de semana #1 - Sabado dd de MMM

Fin de semana #1 - Domingo dd de MMM

Fin de semana #2 - Sabado dd de MMM

. . .

Cantidad de dias que corresponden a fines de semana: nnn

(si el primer fin de semana empieza el domingo, obviamente omitimos el primer sabado)

Objetivo: uso de estructuras de control e iteracion, tipos de datos, uso de MSDN para examinar propiedades disponibles para el tipo DateTime

**Ejercicio 2**

Crear una clase Persona que contenga 3 campos o atributos: Nombre, Nacimiento, DNI. Seleccionar el tipo de dato mas apropiado para cada campo. Escribir un constructor que permita iniciar una instancia de esta clase pasandole como argumentos los 3 valores iniciales de los campos.



Luego de esto, crear una funcion TryParsePersona (no como un metodo de instancia sino como una funcion comun al mismo nivel de Main) con la misma filosofia de, por ejemplo, int.TryParse, es decir, que acepte un string conteniendo los datos de un objeto y que retorne true o false dependiendo si con los datos se puede generar un objeto tipo Persona. Como argumento **out** y si el proceso de conversion tiene éxito, deberiamos obtener una nueva instancia de Persona.

La firma de la funcion seria:

bool TryParsePersona( string strPersona, **out** Persona resultado )

Una cadena de ingreso tendria el siguiente formato:

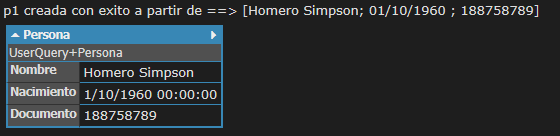
“Homero J Simpson ; 01/01/1960 ; 14879898”

Cada valor esta separado por punto y coma.

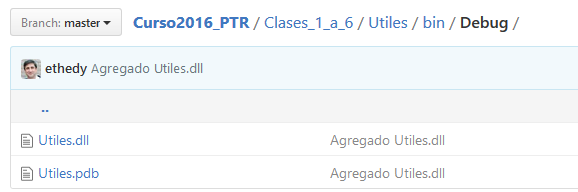
Entre otras cosas debemos validar que:

* Cada cadena contenga exactamente 3 campos
* El segundo campo sea convertible a fecha
* El tercer campo sea convertible a un entero
* Adicionalmente, podemos chequear que la fecha de nacimiento no sea menor que un valor dado, o que el documento sea mayor que cierto numero
* Otras validaciones que se les puedan ocurrir…

El resultado deberia ser como el que sigue:



Para ayudarnos con el procesamiento de la cadena, podemos referenciar desde LINQPad el assembly Utiles.dll (podemos bajarlo de GitHub)



Luego de referenciar el assembly, hay que agregar el namespace Utiles en la consulta LINQPad y a partir de entonces veremos que cualquier string tiene dos funciones nuevas:

int Campos(char sep = ‘;’)

string GetCampo(int campo, char sep = ‘;’)

La primera retorna la cantidad de campos en los que se puede descomponer una cadena, si usamos como separador el segundo argumento (por defecto es punto y coma, por lo que no habria que utilizar este parametro)

La segunda nos devuelve, cada uno de los campos en que se descompone la cadena, tambien usando el separador del segundo parametro. El primer parametro se accede con el indice “1”.

Por ejemplo:

string prueba = "Campo 1 ; Campo 2 ; Campo 3 ; Campo 4 ; Campo 5";

prueba.Campos(); // retorna 5

prueba.GetCampo(3); // retorna [ Campo 3 ]

Objetivos: escribir clases, constructores, manejar funciones y posibles errores, librerias externas