Universidad Nacional de La Matanza Materia: Programación Web 3 2C 2021 – 2do Parcial Recuperatorio

Apellido y Nombre: El Barto DNI: \_

Teoría

1. **Un artículo en redes sociales afirma que usar un ORM podría perjudicar la performance de una aplicación por lo tanto NO recomienda su uso. Ud. está de acuerdo? Tiene algo de veracidad la afirmación? Justifique en todos los casos.**

Depende del proyecto, es verdad que usar ORM tiene beneficios de velocidad, es legible e incluso puede llegar a quitar código repetido, además de eso es rápido de aprender (Mucho más que SQL). Pero cuando ya cuando el proyecto tiene un entorno de mucho mayor volumen perderá rendimiento, en esos casos es mejor usar SQL puro para optimizar consultas complejas, porque en esos casos ORM tardaría más.

1. Asumiendo que existe una variable Contexto EF llamada “context” y que existe una entidad Cliente correctamente creada con EF, se quiere Modificar la razón social de un cliente existente (cuyo id es recibido por parámetro), utilizando el siguiente código, sin lograr el resultado esperado:

public ActionResult ModificarRazonSocialCliente(int id)

{

//Punto b) 🡪 De manera “ Modificacion Parcial ”

~~Cliente~~ var cliente = new Cliente();

cliente.Id = id;

cliente.RazonSocial = "Cliente Modificado";

context.Cliente.Atach(cliente); //Para que le preste atención

context.Entity(cliente).Property(x=> x.RazonSocial).isModified=true;

context.SaveChanges();

return RedirectToAction(~~nameof(~~Index~~)~~);

}

1. Describa que está sucediendo, es decir identifique y describa el/los problema/s

Esta creando un cliente en la función, guardando su id y razón social. Luego guardas cambios en el contexto, aunque en ningún momento hubo un cambio en el contexto. Por lo que veo intento hacerlo con el método de “Modificación Parcial”, además que falta la etiqueta de [HttpPost], por ende, necesita traer desde el contexto al cliente

1. Corrija el código anterior para que finalmente cumpla el objetivo de modificación.
2. **A partir del siguiente código:**

public string LINQ()

{

//El origen del dato

int[] fibo = new int[] { 1, 2, 3, 5, 8, 13, 21, 34, 55 };

//La creacion de la consulta

var fiboOJO = fibo.Where(n => n >= 8);

fibo[0] = 100; //Linea 3

string salida = "";

//Ejecucion

foreach (int num in fiboOJO)

{

salida += $"{num} - ";

}

return salida;

}

1. Identifique las partes o pasos de una consulta en LINQ.

LINQ tiene 3 pasos, buscar el origen del dato, consultar el dato (Creación de la consulta) y ejecutarlo

1. Indique cual es la salida del método y justifique en relación a la Línea 3. No entendí bien jaja

La salida del método será un string en forma de cadena de los números que había en el array “fibo” pero solo los que sean mayor e igual a 8, y cambiando el primero de su fila con un 100

Algo así “ 100 – 8 – 13 – 21 – 34 – 55 “

1. **Verdadero o falso EF. Justifique en los casos que sea falso:**

1. Luego de recuperar un objeto del contexto de EF, el mismo tiene estado “**recovered**”, ya que está vinculado al contexto.

Falso, recibe el nombre de “Unchanged”

1. Una de las limitaciones de EF Core 5 es que NO existe forma de agregar validaciones DataAnnotations a las entidades.

Falso, se puede agregar validaciones haciendo clases parciales de la entidad misma

1. En EF Core 5 es mandatorio/obligatorio usar el enfoque DataBaseFirst, debido a que ningún otro enfoque tiene soporte por Microsoft.

Se pueden usar varios tipos de enfoques, como Code First, pero no Model First (En Core 6 sí)

1. LazyLoading es el patrón o mecanismo que permite postergar la inicialización de objetos relacionados hasta el momento que son utilizados y se encuentra activo por defecto en Net 5 / EF Core 5.

Lo primero es correcto, excepto que no está activo por defecto

1. **Verdadero o falso Web Services y Sesión. Justifique en los casos que sea falso:**
2. ASPNET Web API se basa en convenciones respecto al nombre de sus métodos y NO permite bindear parámetros complejos (clases)

Falso, permite bindear tanto primitivos (int,string,etc), como también complejos (En este caso clases)

1. Las variables de sesión en ASP.NET Core requieren habilitarse explícitamente ya que no están activas por defecto como en ASP.NET 4.x

¿Esta es del anterior parcial? Las variables de sesión están habilitadas pero no se encuentran activas (Pero no recuerdo en que version)

1. Un cliente que invoca a un servicio ASPNET MVC Web api debe usar serialización XML.

Falso, ASPNET MVC Web tiene por default formato JSON, si quiere serializarlo con XML tiene que cambiarlo

1. ASPNET MVC Web Api NO mantiene el estado entre llamadas cliente y el servidor.

No se si es exactamente así, ya que la arquitectura REST quien no mantiene el estado entre llamadas de cliente y el servidor