TEMA 1 y 2 PARACIAL 2 1C 2017 PW3

1. Entity Framework:
   1. Los enfoques de EF son 2, model first y database first, en este último se debe partir de un modelo relacional correcto. **Falso**

**Rta**: Los tres enfoques son, model first, database first y code first.

* 1. Una de las desventajas de EF es que no soporta herencia. **falso**
  2. LINQ solo se puede aplicar sobre el origen de los datos EF (contexto). **Falso**

**Rta**: Se puede aplicar sobre Sql, objetos, xml, y datasets.

* 1. En LINQ to entities existen 2 tipos de consultas: 1) Sintaxis de expresiones de consulta (LINQ) y 2) Sintaxis de consulta basada en eventos (con expresiones gamma). **Falso**

**Rta**: Son expresiones Lambda y es basada en métodos.

1. Explique con sus palabras que significa el término “Lazy Loading” o “Carga Perezosa”.

**Rta**: Es un patrón de diseños que pospone la inicialización de los objetos hasta el momento en que estos tengan que ser utilizados.

Es la manera de tratar objetos sin hacer la carga completa de sus atributos específicos hasta el momento que estos sean requeridos.

* 1. Defina ORM.

**Rta**: Es una técnica de programación para convertir datos entre el sistema de tipos utilizando un lenguaje de programación orientado a objetos y el utilizado en un modelo relacional de base de datos. Resuelve el desajuste de impedancia (diferencias existentes entre los modelos relacionales y los modelos conceptuales)

* 1. Indique y explique cuáles son los enfoques de EF

**Rta**: Son tres enfoques, **“Database first”** a partir de un modelo relaciónal genero el EDM. El **“Model First”**, a partir de un modelo conceptual genero el modelo relacional. Y el **“Code First”** donde se parte de un modelo puro de entidades en programación orientada a objetos y se genera el modelo relacional.

1. Un Articulo de internet comenta que: “Usar un ORM tiene las siguientes desventajas:
   1. Las Aplicaciones pueden ser más lentas.
   2. Se puede necesitar una inversión de tiempo de aprendizaje del ORM (como operarlo, como hacer consultas, etc).

Por lo tanto, en general se desaconseja su uso.”

Indique si está de acuerdo justificando claramente con sus palabras.

**Rta**: La respuesta a estas afirmaciones varían según el contexto ha aplicar una resolución, donde existen millones de registros con los cuales trabajar, y donde el tiempo de respuesta sea crítico, se podría aconsejar usar SQL puro para optimizar consultas complejas, ya que utilizar un ORM tendría un retardo de tiempo. Con respecto a la inversión de tiempo que lleva aprender a utilizar un ORM, podes comentar que la curva de aprendizaje y el tiempo que conlleva es menor a tener que aprender SQL puro para cada motor de base de datos, y además dicho tiempo se recupera y se ganá aun más, al momento del desarrollo, ya que es muy considerable la diferencia de código ha escribir con un ORM (mucho menor).

1. Web Services:
   1. En ASP .Net las propiedades se declaran con el atributo [WebMethod] sobre el nombre de cada propiedad. **Verdadero**
   2. Desde .Net puedo consumir un web Service construido en PHP. **Verdadero**
   3. En ASP .Net se crean dentro de un archivo extensión .asmx y debe estar si o si dentro del mismo proyecto MVC. **Verdadero**

Rta:

* 1. No pueden utilizarse Mediante protocolo HTTP, ya que sería muy peligroso. Falso

**Rta**: Se utiliza HTTP por ser muy difundido

**Practica**

1. Escriba la siguiente query en LNQ o Lambda.  
   SELECT \* FROM Jugadores where Edad <=20  
   **Rta. Lambda:**Context ctx = new Context();  
   List<Jugadores> Jugadores = ctx. Jugadores.Where(x=>x.Edad<=20).ToList();
2. Escriba la siguiente query en LINQ o Lambda.  
   SELECT TOP 1 FROM Jugadores WHERE Nombre = “Pepe” AND Apellido=”Argento”

**Rta. Lambda**Context ctx = new Context();  
Jugadores jugador = ctx. Jugadores.Where(x=>x.Nombre==”Pepe”&& x.Apellido=”Argento”). FirstOrDefault ();

1. Escriba un ejemplo de código utilizado Eager Loading.

Var jdrs = from jugadores in ctx.Jugadores.Include(“Domicilio”)

Select jugadores;

Foreach (jug in jdrs){

String calle = jug.Domicilio.Calle;

1. Partiendo de un model.edmx correcto, el cual el proveedor tiene dos propiedades Id y RazónSocial :  
     
   Se requiere modificar el Proveedor “Carlos” con la razón social “Carlos SRL”, para tal fin, se sabe a priori que el id del proveedor es 20. El código usado para esta modificación es el siguiente:  
     
   Contexto ctx = new Contexto();  
   Proveedor prov = new Proveedor();  
   prov.Id = 20;  
   prov.RazonSocial = “Carlos SRL”;  
   ctx.Proveedores.Add(prov);  
   ctx.SaveChanges();

**Resolucion:**

**Contexto ctx = new Contexto();**

**Proveedor prov = ctx.Proveedor.Single(x => x.id == 20);**

**prov.RazonSocial = “Carlos SRL”;**

**ctx.SaveChanges();**

Lamentablemente el resultado de la funcionalidad no es el esperado.

* 1. Explique con sus palabras lo que está sucediendo, es decir identifique cual es el problema.

Rta: El problema es que en vez de modificar el proveedor existente con dicho ID y razón social, se esta intentando agregar uno nuevo.

* 1. Explique ayudándose con el código/pseudocodigo como hacer que la funcionalidad logre el objetivo inicial, es decir, la modificación correcta del proveedor.

1. Basándose en el ejercicio anterior, asumiendo que :  
   - Existe la tabla Producto con las columnas Id y Nombre , y que la misma fue agregada correctamente al .edmx   
   - Existe una relación N a N (Muchos a Muchos) entre para la entidad Producto y Proveedor.  
   Escriba el código para al momento de crear un Producto, se dé le debe asociar todos los proveedores cuya razón social contenga el nombre del producto , ejemplo:  
   Si se quiere dar de alta el producto “Zapatilla”, se debe asociarse a este producto con todos los proveedores cuya razón social contenga “Zapatilla”, por ejemplo “zapatillas Haedo SRL”.

**Rta:**

**Contexto ctx = new Contexto();**

**Producto prod = New Producto();**

**prod.Nombre = “Zapatilla”;**

**List<Proveedor> provs = ctx.Proveedor.where(x => x.Nombre.Contains(prod.Nombre));**

**prod.Proveedores.add(provs);**