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 – Son 3 enfoques : code first ,model first y database first.**
   2. Una de las ventajas de EF es que no soporta herencia. **Verdadero**
   3. LINQ solo se puede aplicar sobre el origen de los datos EF (contexto). **Falso - Tambien se puede aplicar LINQ to Objects (cadenas,números, objetos, etc) , LINQ to XML ((Documentos XML) , LINQ to SQL (Bases de datos SQL Server) ,** **LINQ to Entities (Queries contra el modelo de objetos, Retorna Entidades del Modelo Conceptual), LINQ To DataSets ( Conjuntos de datos ADO .Net).**
   4. En LINQ to entitis existen 2 tipos de consultas: 1) Sintaxis de expresiones de consulta (LINQ) y 2) Sintaxis de consulta basada en eventos (con expresiones gamma). **Falso** **- 2) sintaxis de consulta basada en método(con expresiones Lambda).**
2. Explique con sus palabras que significa el término “Lazy Loading” o “Carga Perezosa”.  
   **Rta. Lazy loading es un patron de diseño comunmente usado en programación a propósito de posponer la inicialización de un objeto hasta el momento en el cual este es necesitado. El objetivo es contribuir a la eficiencia; de esta manera los distintos objetos de una clase se irán cargando a medida que los vamos usando.**
   1. Defina ORM.  
      **Rta**. Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia. Un ORM aporta ayuda en la resolución del denominado desajuste de impedancia, el desajuste de impedancia refiere a las diferencias existentes entre los modelos relacionales (base de datos) y los modelos conceptuales implementados en lenguajes de programación orientados a objetos.
   2. Indique y explique cuáles son los enfoques de EF  
      **Rta.** Existen 3 enfoques:  
      ***•Data Base First***: A partir de un modelo relacional genero el EDM. Se trabaja directamente con DbContext.

***• Model First*** A partir del modelo conceptual (EDM) genero el modelo relacional. Se trabaja directamente con DbContext.

***• Code First:*** No se trabaja con el EDMX designer. Se parte de un modelo puro de entidades en programación orientada a objetos (clases POCO) y a partir de ahí se genera el modelo de base de datos relacional. Incorpora dos nuevos tipos: DbContext, que es una alternativa simplificada de DbContext, y DBset en vez de ObjectSet(Of TEntity). Solo Disponible a partir de Entity Framework 4.1

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.**

**Difiero con el articulo ya que unas de las ventajas que brinda un ORM es la facilidad y velocidad de uso, es verdad que puede verse como una desventaja el aprendizaje de un ORM , pero gracias a ello su objetivo es reducir la cantidad de código y el mantenimiento necesarios para las aplicaciones orientadas a datos.**

1. Web Services:
   1. En ASP .Net las propiedades se declaran con el atributo [WebMethod] sobre el nombre de cada propiedad. **Falso - los webservices exponen métodos, no propiedades y esos métodos son los q se marcan con WebMethod**
   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**. Falso - El servicio se crea fuera del proyecto MVC y se referencia como un proyecto mas de la solución. Además el ASMX es la declaración, pero hay mas archivos q tienen la implementación**
   4. No pueden utilizarse Mediante protocolo HTTP, ya que sería muy peligroso. **Falso – Esta basado en HTTP**

**Practica**

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();  
**Rta. SQL**

var jg = FROM j in ctx.Jugadores WHERE Edad <=20 SELECT j;

List<Jugadores> jugadores = jg . ToList();

1. 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 ();  
**Rta. SQL**

var jg = FROM j in ctx.Jugadores WHERE Nombre = “Pepe” AND Apellido=”Argento” SELECT j;

Jugadores jugador = jg . FirstOrDefault ();

1. Escriba un ejemplo de código utilizado Eager Loading.  
   **Rta.**  
   Eager Loading es lo opuesto a Lazy Loading, es decir los objetos relacionados se cargaran o recuperaran la primera vez, en el momento de ejecutar la consulta. Para eager Loading debemos explicitar el metodo Include(), pasando como parámetro la-s entidad-es que queremos se recuperen con la consulta.

Ej:  
Context ctx = new Context();  
Cliente cli = ctx.Cliente.Include(c=>c.Direccion).FirstOrDefault();

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();  
     
   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 código lo que está haciendo es dar de alta/crear un nuevo proveedor.
   2. Explique ayudándose con el código/pseudocodigo como hacer que la funcionalidad logre el objetivo inicial, es decir, la modificación correcta del proveedor.  
      **Rta.**   
      Contexto ctx = new Contexto();  
      Proveedor prov = ctx.Proveedor.Where(x=>x.Id == 20).FirstOrDefault();

prov.RazonSocial = “Carlos SRL”;  
ctx.SaveChanges();

* 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 nombre del producto , ejemplo:  
     Si se quiere dar de alta el producto “Zapatalla”, se debe asociarse a este producto con todos los proveedores cuya razón social contenga “Zapatilla”, por ejemplo “zapatillas Haedo SRL”.  
     **Rta.**   
     public void crearProducto( string NombreProducto)

{

Contexto ctx = new Contexto();

Producto pr = new Producto();

Pr.Nombre = NombreProducto;

List<Proveedor> prov = ctx.Proveedores.Where(x=>x.Nombre.Contains(NombreProducto)).ToList();

foreach (Proveedor p in prov)

{

Pr.Proveedores.Add(p);

}

Ctx.Productos.Add(pr);  
ctx.SaveChanges();

}