Estrategias para cargar datos con Entity Framework:
DA1 - 2016 - Gabriel Pifaretti

Supongamos la siguiente relación entre las entidades Cliente y Pedido:

Cliente 1 ---------* Pedido
(Un cliente se relaciona con múltiples pedidos y un Pedido se relaciona a un único Cliente)

1) Eager Loading: "hacé todo el trabajo por adelantado".

Si quisiéramos cargar los pedidos de los clientes usando Eager Loading, cuando obtenga a cada cliente, EF realizaría la consulta SQL que obtiene toda la información de sus pedidos, en una sola consulta.

Se usa el método INCLUDE (System.Data.Entity), por ejemplo:

```
using (var context = new ExampleContext())
{
      var clientes = context.Clientes.Include(c =>
      c.Pedidos).ToList();
}
```

Aquí lo que estamos diciendo es que cuando obtenemos los clientes, también estamos obteniendo los pedidos asociados a cada uno de ellos.

También podemos ver otros ejemplos utilizando las entidades que intervienen en la guía que hacemos en clase de Entity Framework Code First (Blog y Post):

```
using (var context = new BloggingContext())
{
  // Cargamos los Blogs e incluimos los Posts asociados a los
mismos.
    var blogs1 = context.Blogs
                        .Include(b => b.Posts)
                        .ToList();

  // Cargamos UN SOLO Blog pero incluimos los Posts asociados a el
    var blog1 = context.Blogs
```

```
                        .Where(b => b.Name == "Blog de Música")
                        .Include(b => b.Posts)
                        .FirstOrDefault();
    // Cargamos todos los blogs con sus posts asociados,
    // pero utilizando un string para especificar la relación
    var blogs2 = context.Blogs
                          .Include("Posts")
                          .ToList();

    // Cargamos un solo blog con sus posts asociados
    // pero utilizando un string para especificar la relación
    var blog2 = context.Blogs
                        .Where(b => b.Name == "Blog de Música")
                        .Include("Posts")
                        .FirstOrDefault();
}
```

2) Lazy Loading: "no hagas el trabajo hasta que tengas que hacerlo".

Por otro lado, si usara Lazy Loading para los pedidos, cuando obtengo un cliente de la BD, Entity Framework generará la consulta que obtiene solo la información del cliente (sin los pedidos), generando una consulta aparte para el caso en el que yo quiera acceder a los pedidos de un cliente más adelante en mi código.

Para las entidades POCO que quiero que se carguen Lazy, se utiliza la palabra virtual para la property. En este caso el código sería algo así:

```
public class Cliente
{
      …
      public virtual ICollection<Pedido> Pedidos {get;set}
      …
}
```

De esa forma estamos indicando que cuando obtengamos un Cliente, obtendremos toda su información salvo la colección que referencia "Pedidos", ya que está señalada para que se obtenga con Lazy Loading. Cuando accedamos a sus Pedidos haciendo realizando "miCliente.Pedidos", será cuando se realice la consulta sobre la base de datos y se carguen los pedidos para el cliente miCliente.

Eager Loading vs. Lazy Loading:

Su uso varía de lo que me interese obtener en un cierto momento, y de la eficiencia. Por ejemplo, si quisiera llenar una tabla con la lista de clientes y pedidos:

- **Con Lazy Loading, tendría N+1 consultas (1 para el cliente y N para sus pedidos).**
- **Con Eager Loading, tendría una sola consulta para todo.**

3) Explicit Loading: "hacé todo el trabajo incluso si LL está deshabilitado".

Supongamos que tenemos Lazy Loading deshabilitado y que queremos tener un mayor control sobre la información que se quiere obtener. Se realiza una llamada explícita con el método Load, para una cierta entidad.

```
using (var context = new BloggingContext())
{
      var post = context.Posts.Find(2);
      //cargo el blog para ese post explícitamente
      context.Entry(post).Reference(p=>p.blog).Load();
}
```

Explicit loading se define como: cuando tenemos objetos retornados por un por una cierta consulta, los objetos relacionados no se cargan al mismo tiempo. Por defecto, estos no se cargan hasta que explícitamente se carguen usando la propiedad Load.

Tanto Lazy Loading como Explicit Loading son ejemplos de "Deferred Execution" ("no computes el resultado hasta que quién te llama realmente te use").