Desarrollador de Aplicaciones Web
Programación Web III

Departamento de Ingeniería e Investigaciones Tecnológicas

LINQ

Ing. Matias Paz Wasiuchnik
Ing. Pablo Nicolás Sanchez
Ing. Mariano Juiz

¿Qué es?

- LINQ Language Integrated Query
- La consulta (query) como parte del lenguaje
- Surge a partir de .NET Framework 3.5

Sin LINQ…

- Objetos usando iteraciones y condiciones

```
foreach(Alumno a in alumnos)
  if (a.Localidad == “San Justo") ...
```

- Consultas a Tablas

```
SELECT *
FROM Alumno
Where Localidad = ‘San Justo’
```

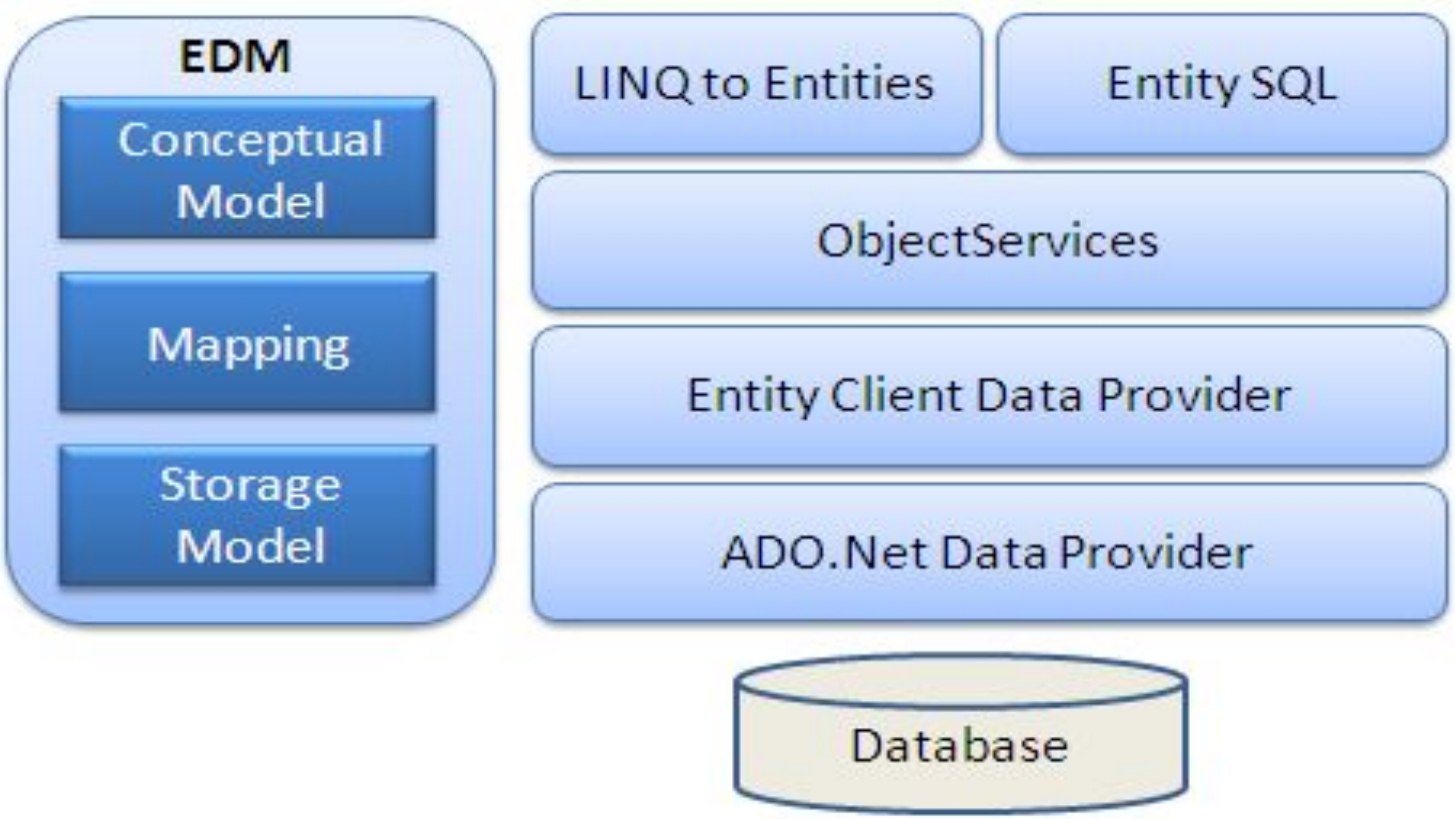
- Consultas a XML usando XPath o Xquery

```
//Alumnos/Alumno[@Localidad=‘San Justo']
```

LINQ

- Ejecución a tiempo
  - Las consultas LINQ no se ejecutan hasta que sea el momento indicado
- El Acceso a Datos es directo
  - No hay que codificarlo
- La sintaxis y el esquema se verifican en tiempo de compilación (se puede usar intellisense)
- Múltiples DataSources
  - Se puede hacer Joins de diferentes fuentes

Arquitectura Entity Framework



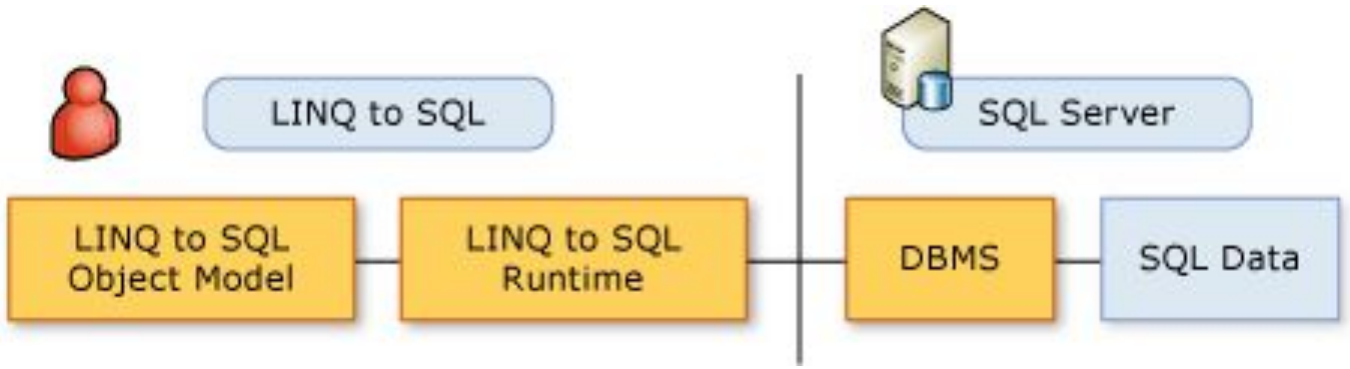
LINQ

Distintos Orígenes de Datos

- **LINQ to SQL** (Bases de datos SQL Server)
- **LINQ to Objects** (Cadenas, numeros, objetos, etc)
- **LINQ to XML** (Documentos XML)
- **LINQ to Entities** (Queries contra el modelo de objetos. Retorna Entidades del Modelo Conceptual)

Herramienta muy útil: http://www.linqpad.net/

LINQ To SQL



- Solo conexiones con SQL Server
- Mapea cada tabla/vista como una clase o "entidad" sin permitir herencia, entidades lógicas o entidades de negocio
- Permite uso de Stored Procedure
- Menos performante que Entity Framework
- Microsoft no desarrollará mejoras

LINQ To Objects

- Se pueden consultar fuentes IEnumerable<T> (incluyendo arrays, List<T>, Dictionary, etc)
- Varios operadores disponibles (Sum, Max, Min, Distinct, Intersect, etc.)
- Nuevos operadores usando “extension methods”
- Los registros (sin aplicar el SELECT y el WHERE) se retornan de la base de datos y luego se aplican en memoria

LINQ To XML

- Similar a LINQ To SQL, pero sobre documentos XML

LINQ To Entities

- Son “queries” sobre el modelo conceptual de EntityFramework
- Las “queries” se organizan en una estructura de árbol
- El Select y el Where se ejecutan en el Motor de Base de Datos
- Más performante
- Devuelve objetos IQueryable<T>

LINQ To Entities

LINQ se usa en este caso para escribir consultas hacia el EDM. Devuelve las entidades definidas en el modelo conceptual

En este caso usamos la **sintaxis de expresiones de consulta**

LINQ Sintáxis Query

```
private void ListarProductos(int idCliente)
{
    //1) Origen de Datos
    EFPW3 context = new EFPW3();

    //2) Consulta: Sintaxis de consulta
    var productos = from p in context.Producto
                    where p.Cliente.Id == idCliente
                    select p;

    //3) Ejecución de Consulta
    foreach (Producto p in productos)
    {
        lblMensaje.Text += p.Nombre + " - ";
    }
}
```

LINQ To Entities

- Consulta basadas en Métodos y expresiones lambdas

```
var result = context.Alumno.Join( localidad,   // Tabla a Relacionar
( a => a.IdLocalidad ),                  //Campo Join de Tabla “left”
( l => l.Id),                  // Campo Join de Tabla “right”
( ( a, l ) => new { Alumno = a, Localidad = l } ))  // asignación
.Where(x => x.l.Descripcion = ‘San Justo’ && x.a.Edad > 30) //Filtro
OrderBy(x => x.a.NomyApel)                  //Ordenación
.Select(x => new { x.a.NomYApel, x.a.DNI, x.l.descripcion});  //Select
```

LINQ To Entities

El mismo ejemplo anterior, pero usando **sintaxis de consulta basada en métodos**

Es importante destacar que en este tipo de sintaxis se usan las llamadas **expresiones lambdas**, observar: **(p => p.Cliente.Id == idCliente)**

LINQ Sintáxis de Métodos

```
private void ListarProductos(int idCliente)
{
    //1) Origen de Datos
    EFPW3 context = new EFPW3();

    //2) Consulta: Sintaxis de Metodo, con expresión lambda
    var productos = context.Producto.Where(p => p.Cliente.Id == idCliente).Select(p1 => p1);

    //3) Ejecución de Consulta
    foreach (Producto p in productos)
    {
        lblMensaje.Text += p.Nombre + " - ";
    }
}
```

LINQ To Entities

- Consulta SQL

```
SELECT *
FROM Alumno
WHERE Localidad = 'San Justo'
      AND Edad > 30
```

- Consulta basada en Queries

```
var result = from a Alumno
             where a.Localidad = 'San Justo'
                   AND Edad > 30
             select a;
```

- Consulta basadas en Métodos y expresiones lambdas

```
var result = context.Alumno.Where(a => a.Localidad='San Justo'
      and a.Edad>30).select(a1 => a1);
```

LINQ To Entities

- Consulta SQL

```
SELECT a.*
FROM Alumno a
      Inner Join Localidad l on (a.idLocalidad = l.id)
WHERE l.descripcion = 'San Justo'  AND Edad > 30
```

- Consulta basada en Queries

```
var result = from a Alumno
join l Localidad on a.idLocalidad equals l.id
where l.descripcion = 'San Justo'  AND Edad > 30
            select a;
```

Lazy Loading (Carga Perezosa)

Lazy loading es un patron de diseño comumente 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.

Estudiemos el siguiente ejemplo: si tenemos una entidad **Empleado,** que tiene como propiedad una entidad **DireccionEmpleado**, y hacemos la siguiente consulta LINQ to entities:

```
EFPW3 ctx = new EFPW3();

var emps = (from em in ctx.Empleados
            select em).ToList();

foreach (Empleado em in emps)
{
    string nombre = em.Nombre;
    string calle = em.DireccionEmpleado.Calle;
}
```

En la siguiente consulta no se recupera información para el objeto **DireccionEmpleado**

Al momento que se quiere acceder a la propiedad **DireccionEmpleado**, es justo en ese preciso instante cuando se recupera su información desde la BD.

```
public EFPW3() : base("name=EFPW3", "EFPW3")
{
    this.ContextOptions.LazyLoadingEnabled = true;
    OnContextCreated();
}
```

Constructor del Contexto:
Por defecto EF trabaja en esta modalidad **LazyLoading (ver)**

Eager Loading (Carga Temprana)

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.

```
EFPW3 ctx = new EFPW3();

var emps = from em in ctx.Empleados.Include("DireccionEmpleado")
           select em;

foreach (Empleado em in emps)
{
    string nombre = em.Nombre;
    string calle = em.DireccionEmpleado.Calle;
}
```

Se explicita que se recuperará la entidad DireccionEmpleado en la consulta.

Podemos recuperar la consulta que EF ejecuta contra la base de datos con la siguiente línea de código:

```
((System.Data.Objects.ObjectQuery)emps).ToTraceString();
```

Anexo – Queries en EDM

Podemos ejecutar consultas sobre el EDM mediante:

- **LINQ to Entities**

- **Entity SQL**

  Contexto.CreateQuery<Entidad>(“QUERY_SQL”);

- **Native SQL**

  Contexto.Entidad.SqlQuery(“QUERY_SQL”)

Desarrollador de Aplicaciones Web
Programación Web III

Departamento de Ingeniería e Investigaciones Tecnológicas

Muchas gracias

Ing. Matias Paz Wasiuchnik
Ing. Pablo Nicolás Sanchez
Ing. Mariano Juiz