
NHIBERNATE

baufest

Sobre el instructor

Diego Tubello

dtubello@baufest.com

Technical Expert

13+ años de experiencia en desarrollo de Software

baufest

baufest

Objetivos del curso

- Inicialmente existirá una breve introducción que permitirá incorporar conceptos que se utilizarán a lo largo del curso, para luego, ir evolucionando en el conocimiento del framework
- Se aclara que el curso será exclusivamente sobre **NHibernate**, descartando entrar en detalle de otros conceptos de arquitectura en capas o desarrollo Full Stack, sobre los que por supuesto habrá menciones pero no se entrará en detalle
- El curso irá evolucionando día a día hasta poder completar un ejercicio integrador final

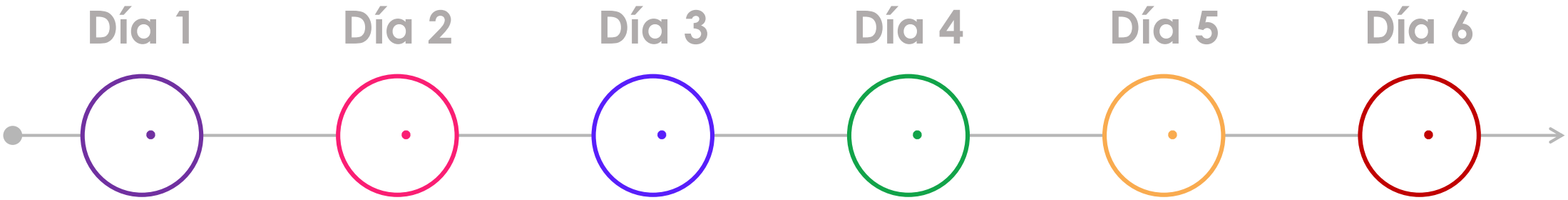
baufest

Organización del curso

Organización del curso (1)

Temario por día

baufest



- Día 1
  - Presentación
  - Introducción
  - Mapeo básico
  - Entorno de desarrollo
- Día 2
  - Arquitectura
  - Configuración
  - Session
  - Operaciones CRUD
- Día 3
  - Convenciones
  - Mapeo de claves
  - Mapeo de relaciones
  - Otros mapeos
- Día 4
  - Transacciones
  - Operaciones CRUD con modelo completo
  - Revisión de SQL generado
  - Consultas
- Día 5
  - SQL Nativo
  - Buenas prácticas
  - Uso en aplicaciones Web
- Día 6
  - Ejercicio integrador
  - Evaluación
  - Despedida

Organización del curso (2)

Eventos en común para todos los días

baufest

Coffee Break

Resolución de dudas de los días anteriores

baufest

Día 1 – Objetivos del día

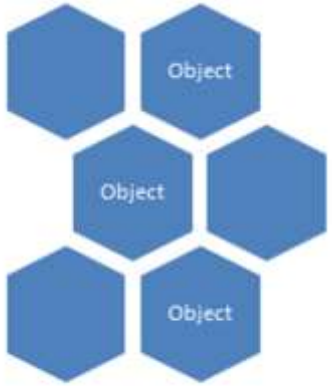
- Introducir conceptos y actividades que serán utilizados a lo largo del curso
- Explicar que es un ORM, que problema resuelve
- Realizar un primer acercamiento a **NHibernate**
  - Introducción teórica
  - Mapeo básico
  - Práctica: Utilización de entorno de trabajo y un “Hola Mundo”

Conceptos preliminares

Diferentes modelos

baufest

- Modelo de Objetos
  - Objetos
  - Propiedades
  - Agregación, Composición, Herencia



Objects in Memory

- Modelo Relacional
  - Tablas
  - Campos
  - Registros
  - Relaciones por Foreign Keys

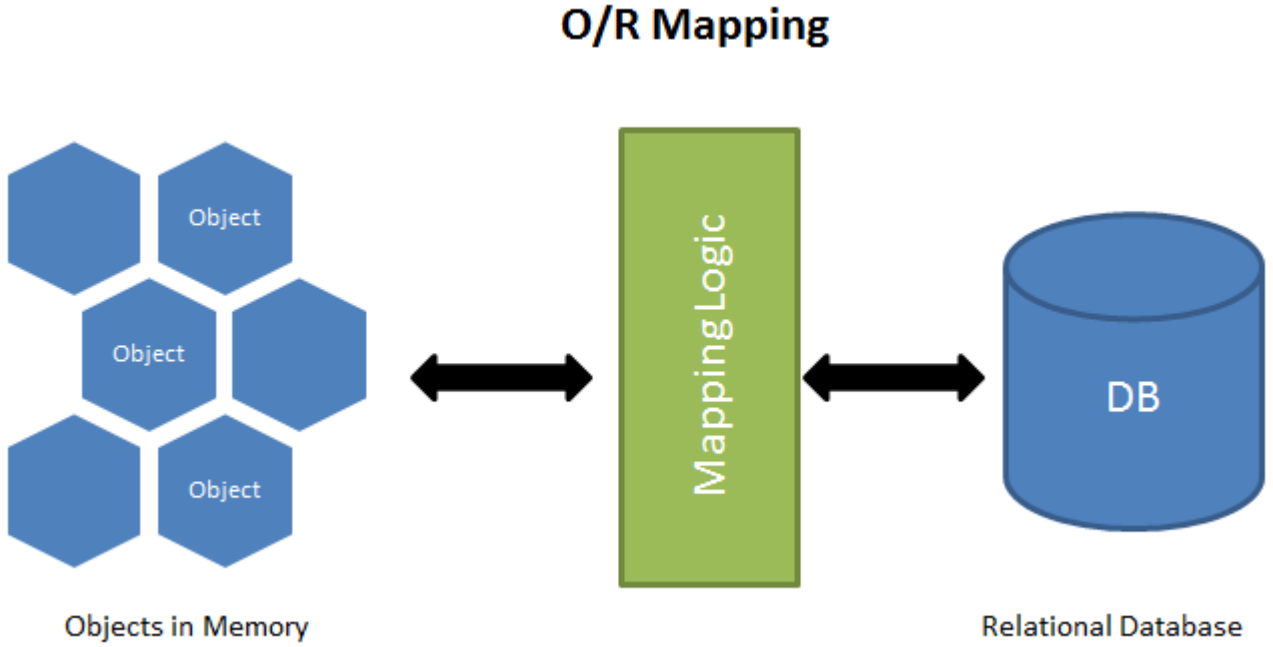
Relational Database

Conceptos preliminares

ORM: Object/Relational Mapper

baufest

- Permite mapear un modelo de objetos a un modelo relacional.
- Se mapea un conjunto de objetos que colaboran entre si a un conjunto de tablas relacionadas.
- Permite persistir objetos de una forma “transparente”
- Se manipulan los datos como si tuviéramos colecciones de objetos en memoria



Object/Relational Mapper

¿Por qué usar un ORM?

baufest

- Permite modelar con objetos abstrayéndonos del modelo de datos relacional.
- Se reducen los tiempos de desarrollo al no tener que escribir código de base de datos.
- Se manipulan los “datos” como si tuviéramos colecciones de objetos en memoria.
- Se facilitan los *refactors*, ya que al modificar el modelo de objetos el ORM adapta todas las consultas automáticamente eliminando la necesidad de modificar manualmente consultas y stored procedures.
- El esquema de carga de objetos de de la base suele ser muy configuragle: cache, lazy/eager loading, batch
- Soporte para múltiples motores de bases de datos y versiones (por configuración)

Abstraerse de la base de datos no implica olvidar que existe

Object/Relational Mapper

Implementaciones

baufest

Object/Relational Mapper

¿Por qué NHibernate?

baufest

- Nació en 2006 como un port del framework Hibernate para Java, pero con el tiempo evolucionó de forma distinta
- Es open-source (LGPL)
- Soporta la mayoría de los motores de bases de datos y sus versiones
- Es flexible y posee muchos puntos de extensión
- Sigue activamente en desarrollo
- Soporte para .Net Core a partir de la versión 5

Mapeo de objetos

Alternativas: Xml, Fluent, Auto-mappings

baufest

Archivos XML

- Mecanismo original de mapeo portado de Hibernate (de la época dorada del XML)

Fluent NHibernate

- Surge como un proyecto separado con la idea de genera mapeos type-safe
- Permite hacer mapeos manuales o “automáticos”
- Convention over configuration

Mapeo de objetos

Objetos

baufest

NHibernate es capaz de mapear objetos “comunes” (POCO):

- No se requiere implementar una interface y extender una clase
- Tampoco deben tener ningún código relacionado a NH

Solo requiere dos cosas:

- La clase debe tener un constructor por defecto sin parámetros (o no tener ningún constructor)
- Todos los métodos y propiedades de la clase deben ser *virtual*

```
namespace Intro.NHibernate.Entities
{
    public class Product
    {
        public virtual int Id { get; set; }
        public virtual string Code { get; set; }
        public virtual string Description { get; set; }
        public virtual decimal RetailPrice { get; set; }
        public virtual Category Category { get; set; }
    }
}
```

Mapeo de objetos

XML

baufest

El mapeo de objetos puede definirse en un archivo xml de la siguiente manera.

Todos los objetos deben tener un atributo como identificador.

```xml
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
  assembly="Intro.NHibernate"
  namespace="Intro.NHibernate.Entities" default-lazy="false">

  <class name="Product">
    <id name="Id">
      <generator class="identity" />
    </id>
    <property name="Code" />
    <property name="Description" />
    <property name="RetailPrice" />
    <many-to-one name="Category" column="Category_id" />
  </class>

</hibernate-mapping>
```

```csharp
namespace Intro.NHibernate.Entities
{
    public class Product
    {
        public virtual int Id { get; set; }
        public virtual string Code { get; set; }
        public virtual string Description { get; set; }
        public virtual decimal RetailPrice { get; set; }
        public virtual Category Category { get; set; }
    }
}
```

Mapeo de objetos

Fluent Nhibernate: Fluent mappings

baufest

- Ofrece una alternativa al mapeo de archivos xml.
- En lugar de documentos xml, se escriben los mapeos en código fuertemente tipado.
- Al ser mapeos tipados, facilita la refactorización de código y hace el mapeo mas legible.

```
public class ProductMap : ClassMap<Product>
{
    public ProductMap()
    {
        Id(x => x.Id).GeneratedBy.Identity();
        Map(x => x.Code);
        Map(x => x.Description);
        Map(x => x.RetailPrice);
        References(x => x.Category).Cascade.None();
    }
}
```

```
namespace Intro.NHibernate.Entities
{
    public class Product
    {
        public virtual int Id { get; set; }
        public virtual string Code { get; set; }
        public virtual string Description { get; set; }
        public virtual decimal RetailPrice { get; set; }
        public virtual Category Category { get; set; }
    }
}
```

Mapeo de objetos

Fluent Nhibernate: auto-mappings

baufest

- Permite mapear un modelo de objetos sin escribir código de mapeo, con muy poco código.
- Se base en el concepto de convenciones sobre configuración (convention over configuration)
- Requiere que nuestro modelo respete ciertas convenciones
- Si las convenciones por defecto no se adaptan a nuestras convenciones, podemos definir y utilizar nuestras propias convenciones
- Para modificar mapeos puntuales tiene el concepto de overrides

Entorno de desarrollo

¿Qué herramientas vamos a utilizar?

baufest

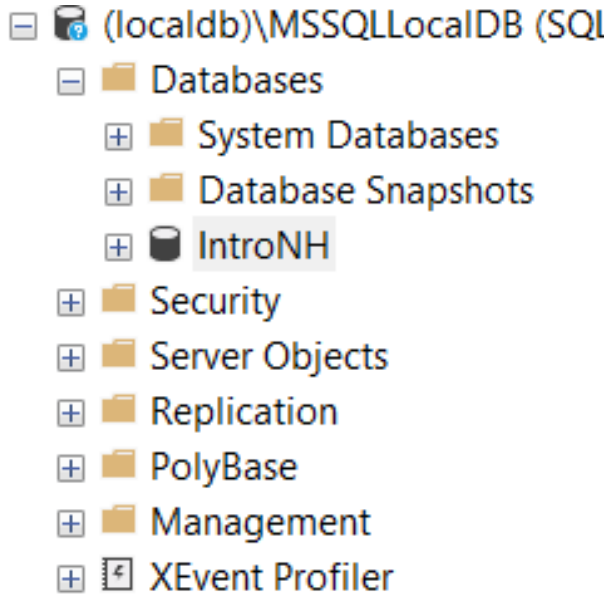
- Visual Studio
- SQL Server Management Studio
- SQL Server Profiler

Práctica

¡Hola Mundo!

baufest

- En SQL Management Studio
  - Conectarse a la base de datos local
  - Crear una nueva base de datos con el nombre “IntroNH”

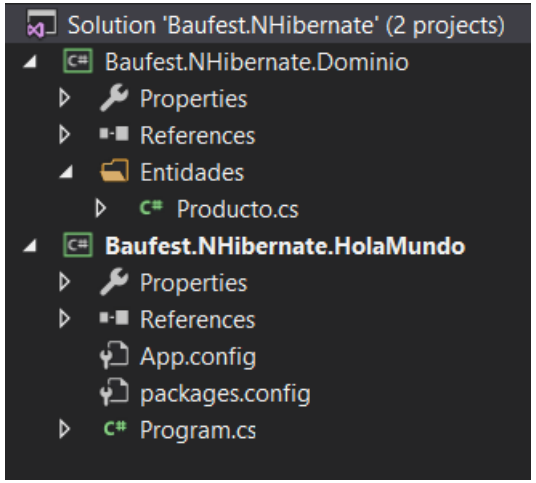


Práctica

¡Hola Mundo!

baufest

- En Visual Studio
  - Crear una nueva solución del tipo blank solution con el nombre “Baufest.Nhibernate”
  - Agregar un proyecto del tipo “Console Application” con el nombre “Baufest.NHibernate.HolaMundo”
  - Agregar un nuevo proyecto del tipo “Class Library” con el nombre “Baufest.Nhibernate.Dominio”
    - Eliminar el archivo Class1.cs
    - Agregar la carpeta “Entidades”
    - Agregar la clase “Producto” dentro de la carpeta Entidades
  - Instalar los paquetes NuGet en el proyecto HolaMundo
    - NHibernate
    - FluentNHibernate




```
0 references
public class Producto
{
    0 references
    public virtual int Id { get; set; }

    0 references
    public virtual string Nombre { get; set; }

    0 references
    public virtual string Descripcion { get; set; }

    0 references
    public virtual decimal Precio { get; set; }
}
```

Práctica

¡Hola Mundo!

baufest

- En Visual Studio
  - En la clase Program crear el método “CrearSessionFactory” con la configuración de NHibernate
  - Dentro del método Main crear un producto y guardarlo en la base de datos
  - Ejecutar el proyecto!

```
public static ISessionFactory CrearSessionFactory()
{
    return Fluently
          .Configure()
          .Database(
             MsSqlConfiguration.MsSql2012.ConnectionString(
                      x => x.FromConnectionStringWithKey("BaufestNH")))
          .Mappings(m => m.AutoMappings.Add(
                     AutoMap.AssemblyOf<Producto>()
                         .Where(t => t.Namespace == typeof(Producto).Namespace)))
          .ExposeConfiguration(cfg => new SchemaUpdate(cfg).Execute(false, true))
          .BuildSessionFactory();
}
```

```
static void Main(string[] args)
{
    var sessionfactory = CrearSessionFactory();
    using(var session = sessionfactory.OpenSession())
    {
        var producto = new Producto
        {
            Nombre = "Lenovo T470",
            Descripcion = "Laptop Lenovo T470 Core i5, 16GB RAM, SSD 128GB",
            Precio = 500
        };

        session.Save(producto);
    }
}
```