
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

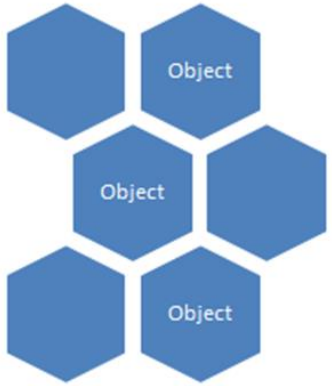
- Introducir conceptos y actividades que serán utilizados a lo largo del curso
- Explicar que es un ORM, que problema resuelve
- Realizar un primer acercamiento a **NHibernate**
  - Introducción teórica
  - Mapeo básico
  - Práctica: Utilización de entorno de trabajo y un “Hola Mundo”

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

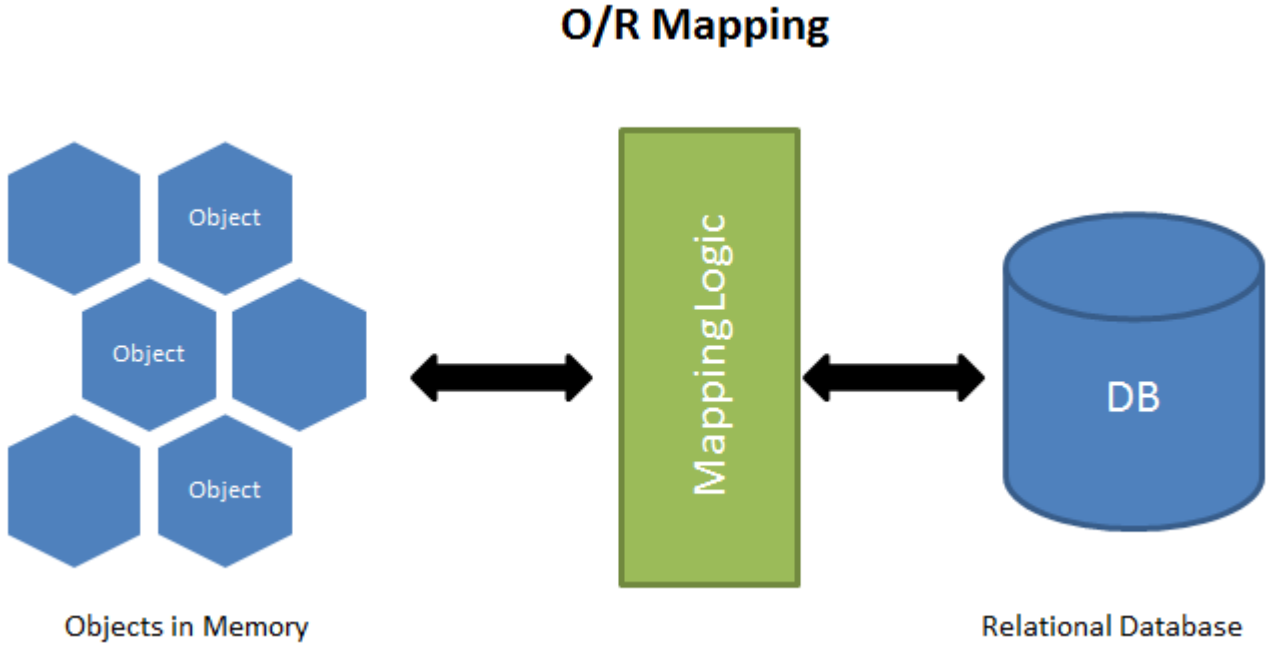
Relational Database

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

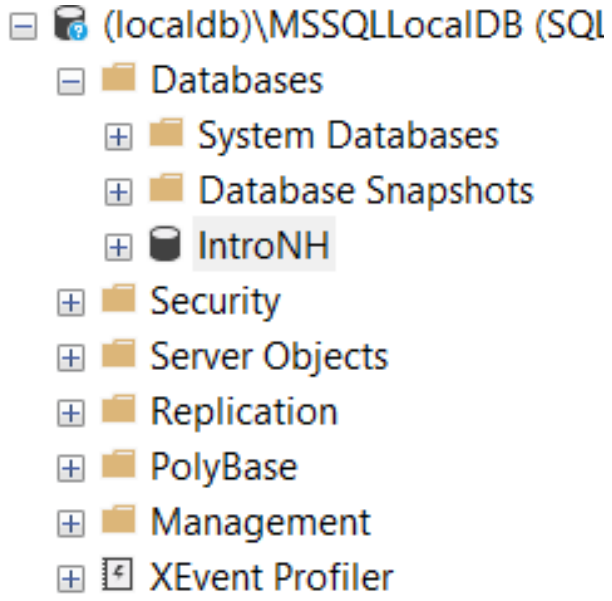
- Visual Studio
- SQL Server Management Studio
- SQL Server Profiler

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”

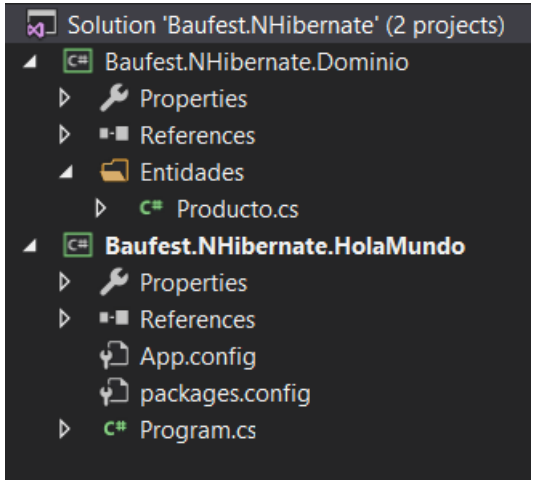


Práctica

¡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);
    }
}
```

baufest

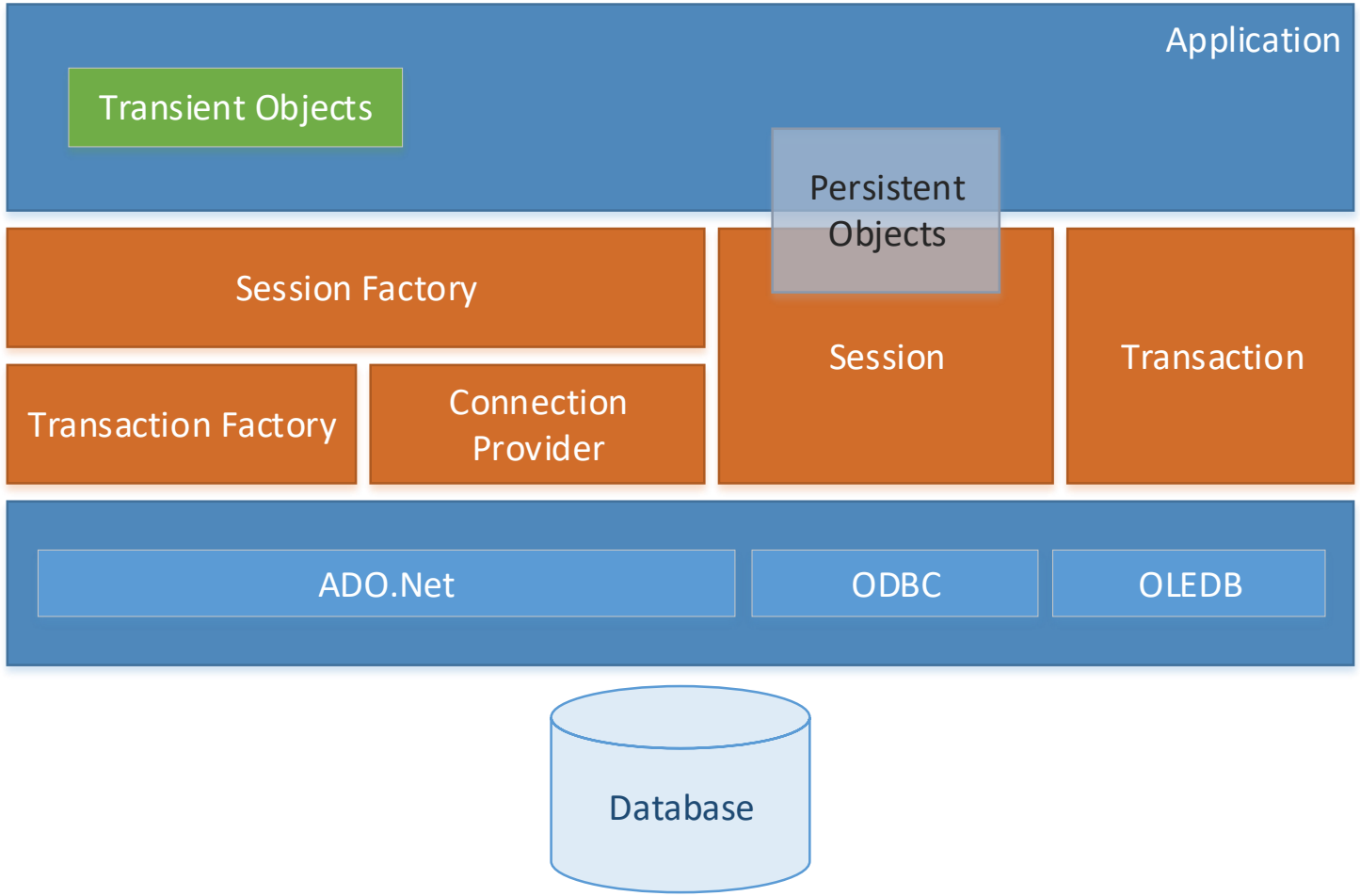
Día 2 – Objetivos del día

- Dudas o consultas de la clase anterior
- Introducir los conceptos de la arquitectura de NHibernate
- Comprender la configuración con sus diferentes opciones
- Comprender el manejo de la sesión
- Realizar operaciones CRUD básicas

Arquitectura

Componentes principales

baufest



Arquitectura

Componentes principales

baufest

- **ISessionFactory**: es una cache con toda la configuración y mapeos “compilados”. Es la Factory para crear las ISession. Se usa una instancia por aplicación.
- **ISession**: representa la conversación entre la aplicación y la base de datos. Se usa una instancia por “operación de negocio” (unit of work). Tiene un tiempo de vida corto.
- **Persistent Objects**: Son objetos que contienen estado persistente (una correlación don datos de la base) y funciones de negocio. Pueden ser POCOs pero están asociados a una única sesión.
- **Transient Objects**: son objetos que no están actualmente asociados a una sesión. Su estado aún no está persistido y no se se persistirá salvo que los asociemos a una sesión y se conviertan en “persistent”.
- **ITransaction**: representa una unidad de trabajo atómica. Es una abstracción de la transacción ADO.Net subyacente.

Configuración

ISessionFactory

baufest

```
Fluently
 .Configure()
 .Database(...) // Configuración de la base y versión
 .Mappings(...) // Configuración de mapeos
 .ExposeConfiguration(...) // Configuraciones adicionales (opcional)
 .BuildSessionFactory();
```

```
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();
}
```

Configuración

ISessionFactory

baufest

Database permite configurar el motor de base de datos a utilizar:

- **Motor/Versión**: NHibernate soporta múltiples motores de bases de datos en casi todas sus versiones
- **Connection String**: Se puede especificar el connection string o la clave con la que se lo configuró en el web.config o app.config.
- **Driver**: permite configurar el driver del motor de base de datos a utilizar (opcional)
- **Dialect**: permite configurar un dialecto de SQL específico del motor y versión a utilizar (opcional).

```
return Fluently
    .Configure()
    .Database(
      MsSqlConfiguration.MsSql2012
          .Dialect<MsSqlAzure2008Dialect>()
          .ConnectionString(x => x.FromConnectionStringWithKey("IntroNH")))
```

Configuración

ISessionFactory

baufest

Mappings permite configurar los mapeos

- Se puede especificar automapping, las convenciones, overrides y el conjunto de clases a mapear

```
.Mappings(m => m.AutoMappings.Add(
          AutoMap.AssemblyOf<Producto>()
              .Where(t => t.Namespace == typeof(Producto).Namespace)
              .Conventions.AddFromAssemblyOf<ForeignKeyNameConvention>()
              .UseOverridesFromAssemblyOf<VentaMappingOverride>()
              ))
```

- También se pueden especificar los “fluent mappings” en el caso de querer hacerlos manualmente

```
.Mappings(m => m.FluentMappings.AddFromAssemblyOf<ProductoMap>())
```

**Ejemplo*

Configuración

ISessionFactory

baufest

ExposeConfiguration permite hacer configuraciones adicionales

- Habilitar la generación del modelo de la base
- Opciones de quoting de nombres de columnas tablas
- Modificar la forma de generar las consultas
- Configuración de event listeners
- Configuraciones de ADO.Net
- Conexiones, transacciones, cache, logging, etc.

```
.ExposeConfiguration(cfg =>
{
    cfg.SetProperty(Environment.Hbm2ddlKeyWords, "auto-quote");
    cfg.SetProperty(Environment.ConnectionDriver, typeof(SqlClientDriver).AssemblyQualifiedName);
    cfg.LinqToHqlGeneratorsRegistry<MyLinqtoHqlGeneratorsRegistry>();
})
.ExposeConfiguration(RegisterEventListeners)
.ExposeConfiguration(BuildSchema)
.BuildSessionFactory();
```

ISession

Operaciones

baufest

El objeto a través del cual se hace la manipulación de objetos persistentes es la ISession.

Funciona como un contenedor. Cuando se carga un objeto, se guarda en el contenedor de manera que NHibernate registrará cualquier cambia que se le haga al objeto.

Las operaciones principales son:

- **Get** y **Load**: permiten obtener un único objeto por clave primaria
- **Query**: permite hacer una consulta con LINQ sobre el modelo persistente.
- **Add**: adjunta un objeto a la sesión para que sea persistido.
- **Update**: Re-adjunta un objeto a la sesión para actualizar su estado en el modelo persistente.
- **Delete**: elimina un objeto de la sesión para ser eliminado del modelo persistente.

ISession

Estados de una objeto

baufest

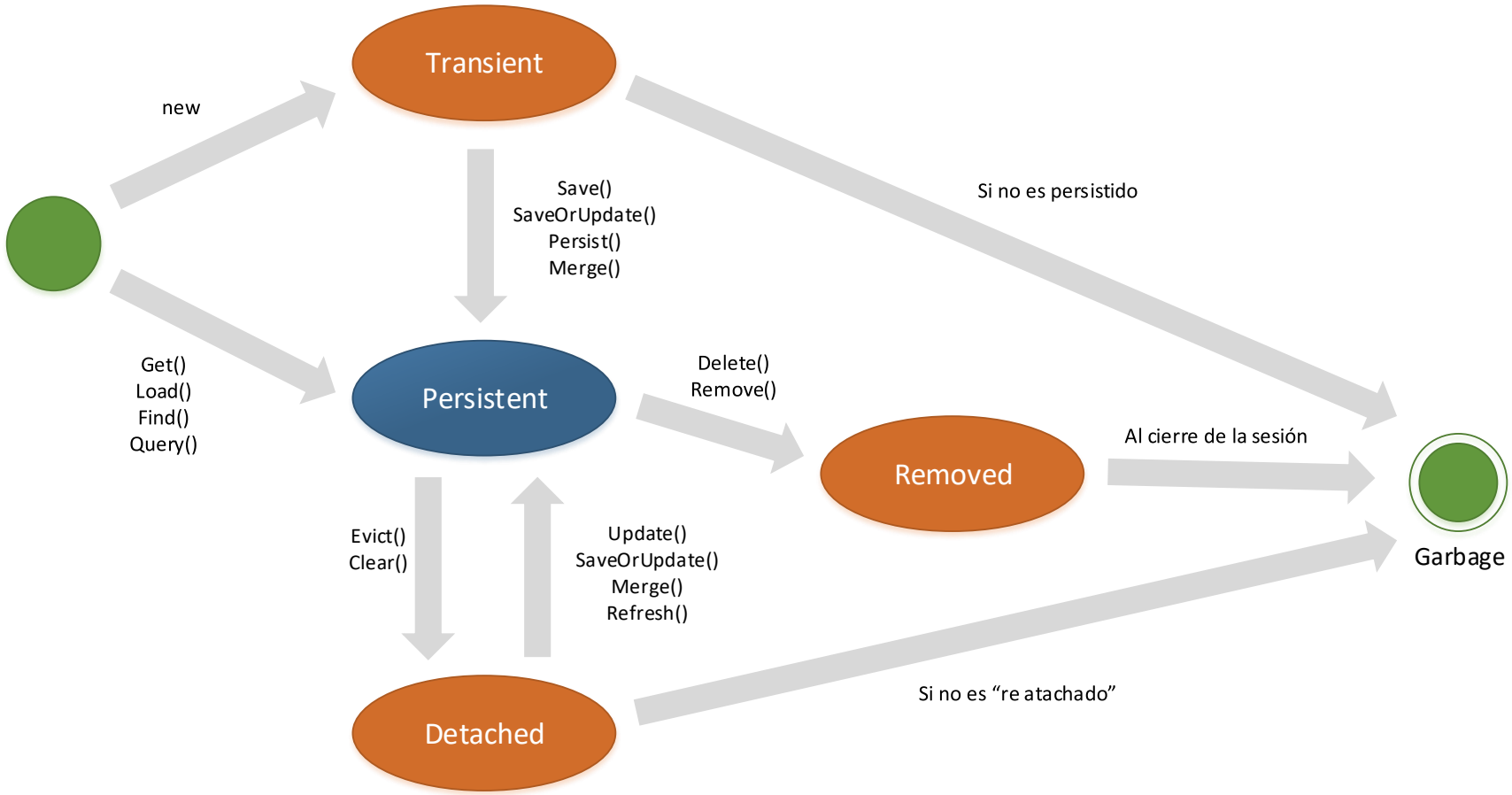
Una instancia de una clase persistente puede pasar por tres diferentes estados con respecto al contexto de persistencia o sesión (ISession).

- **Transient**: el objeto no esta asociado a una sesión y nunca lo estuvo. No tiene un identificador de persistencia (PK)
- **Persistent**: el objeto esta actualmente asociado a una sesión. Tiene un identificador de persistencia (PK), y posiblemente, una registro correspondiente en la base d e datos. Para una sesión particular NHibernate garantiza que dos objetos con la misma clave primaria son el mismo objeto en memoria.
- **Detached**: El objeto estuvo asociado a una sesión.

ISession

Ciclo de vida

baufest



ISession

Flush Mode

baufest

Los cambios que se hacen en la sesión no necesariamente se impactan en la base de datos en ese momento. NHibernate puede diferir un INSERT o UPDATE en la base de datos para agrupar sentencias y optimizar los accesos a la base de datos.

La acción de impactar los cambios de la sesión en la base de datos de llama **Flush**. Se puede configurar cambiando el **FlushMode**:

- **Never**: Los cambios no se sincronizan automáticamente, se deben hace manualmente llamando al método **Flush**
- **Commit**: Los cambios se sincronizan cuando la transacción en curso hace commit
- **Auto**: los flush se hacen automáticamente durante la vida de la sesión para evitar “dirty reads”.
- **Always**: Se hace un Flush automáticamente ante la ejecución de cada consulta

ISession

Flush

baufest

En ocasiones Nhibernate necesita hacer un Flush para sincronizar con la base de datos para mantener la consistencia:

- Cuando se listan registros de la base mediante una consulta
- Cuando se hace commit de una transacción
- Cuando se hace un flush explícito

ISession

Flush

baufest

¿En qué orden se impactan los cambios en un Flush?

1. Todos los inserts, en el mismo orden en que se guardaron usando ISession.Save()
2. Todos las actualizaciones de entidades
3. Todos los borrados de colecciones
4. Todos los inserts y actualizaciones de elementos de las colecciones
5. Todos los inserts de colecciones
6. Borrado de entidades en el mismo orden en que fueron eliminados con ISession.Delete()

Una excepción son las entidades que usan ID identity que se insertan en el momento de ejecutar Save.

ISession

Flush

baufest

Salvo que explícitamente se llame al método Flush() no hay garantías de cuándo la Session ejecuta las llamadas a ADO.Net.

Sin embargo, NHibernate **SI** garantiza:

- El orden en que se ejecutan las sentencias en la base de datos
- Las consultas siempre retornan datos “consistentes” y “actualizados”, nunca devolverán datos incorrectos.

ISession

Lazy Loading

baufest

Cuando se obtiene un objeto de la base de datos, por defecto, NHibernate no carga los objetos relacionados.

Recién obtiene el objeto relacionado al acceder a la propiedad del objetos que se obtuvo en primero lugar.

- En la clase de ejemplo, recién obtendrá el objeto Categoría de la base de datos al invocar *producto.Categoria*

Para que un objeto pueda ser cargado de forma Lazy, todas sus propiedades y métodos deben ser virtual (NH genera proxies dinámicos)

Funciona dentro de la sesión

- Una vez cerrada la sesión ya no se pueden cargar relaciones de forma Lazy

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

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

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

    5 references
    public virtual decimal Precio { get; set; }

    7 references
    public virtual Categoria Categoria { get; set; }
}
```

ISession

Cierre

baufest

Siempre se debe cerrar la sesión para marcar el final de la “conversación” y liberar los recursos ADO.Net.

Se puede hacer explícitamente o con la construcción “using” que hace el Dispose de la Session al salir del scope.

```
var session = sessionfactory.OpenSession();
try
{
    var categoria = new Categoria { Nombre = "Notebooks" };
    session.Save(categoria);
}
finally
{
    session.Close();
}
```

```
using (var session = sessionfactory.OpenSession())
{
    var categoria = new Categoria { Nombre = "Notebooks" };
    session.Save(categoria);
}
```

Práctica

WEB API

baufest

- Descargar la solución de ejemplo de github https://github.com/baufest-ms/Curso-NHibernate-AySA
- Dentro de la carpeta Practica/API hay una solución WEB API base para iniciar los ejercicios
- Recorremos la solución: proyectos, frameworks, dependencias, nswag (swagger)
- Demo de swagger
- Agregar las operaciones CRUD para las entidades que usamos la primera clase
  - Comenzar con la clase Categoria
  - Update detached/attached
  - Hacer lo mismo con la clase Producto. Funciona?
  - DTOs & Lazy Loading