

5.– 8. September 2011
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Aufgewacht

Entity Framework 4.1 vs. NHibernate 3.2

Thomas Haug

MATHEMA Software GmbH

Hallo

- ▶ Dipl.-Inf (Univ.)
- ▶ Senior Consultant, Architekt und Trainer (MATHEMA Software GmbH)
- ▶ 26+ Jahre Software Entwicklung
- ▶ 14+ Jahre Enterprise Entwicklung
- ▶ 8+ Jahre .Net Entwicklung

- ▶ Hauptinteressen
 - Software Architektur
 - Verteilte Systeme
 - Objekt-Relationales Mapping

- ▶ E-Mail: thomas.haug@mathema.de



Die Firma

- ▶ Erlangen, Deutschland
- ▶ 15+ Jahre
- ▶ Software Architektur
- ▶ Know-how Transfer
- ▶ Implementation
- ▶ Schwerpunkt auf
 - Java
 - .Net



Agenda

- ▶ (Architektur) Überblick
- ▶ Entitäten beschreiben
- ▶ CRUD
- ▶ Beziehungen
- ▶ Zusammenfassung



Agenda

- ▶ (Architektur) Überblick
- ▶ Entitäten beschreiben
- ▶ CRUD
- ▶ Beziehungen
- ▶ Zusammenfassung



Überblick NHibernate

- ▶ Aktuell Version 3.2 GA
- ▶ Kapselt ADO.Net und SQL vor Entwickler
- ▶ Unterstützt eine Vielzahl von Datenbanken (SQL Server, Oracle, DB2, Sybase, MySql, Postgres, ...)
- ▶ Transparente Persistenz für POCO Klassen (Entitäten)
- ▶ Unterstützung von Vererbungshierarchien und polymorphen Abfragen
- ▶ Transitive Persistenz für Objektgraphen
- ▶ Unterstützung für sog. Components
- ▶ Unterstützung für ‚Lazy Loading‘ und ‚Eager Fetching‘
- ▶ Optimistische und Pessimistische Sperr-Mechanismen
- ▶ Unterstützung von abgekoppelten Objekten
- ▶ Second Level Caching
- ▶ (SQL) Tracing Logging integraler Bestandteil

Überblick Entity Framework

- ▶ Aktuell Version 4.1 Update 1
- ▶ Kapselt ADO.Net und SQL vor Entwickler
- ▶ Unterstützt SQL Server „out-off-the-box“
- ▶ (Transparente) Persistenz für (POCO) Klassen
- ▶ Unterstützung von Vererbungshierarchien und polymorphen Abfragen
- ▶ Transitive Persistenz für Objektgraphen
- ▶ Unterstützung für sog. Complex Types
- ▶ Unterstützung für ‚Lazy Loading‘ vs ‚Eager Fetching‘
- ▶ Optimistische Locking
- ▶ Unterstützung von abgekoppelten Objekten
- ▶ SQL Tracing ist Bestandteil, aber nur für Abfragen
 - ▶ Abhilfe schafft EFProviderWrapper

Vergleich

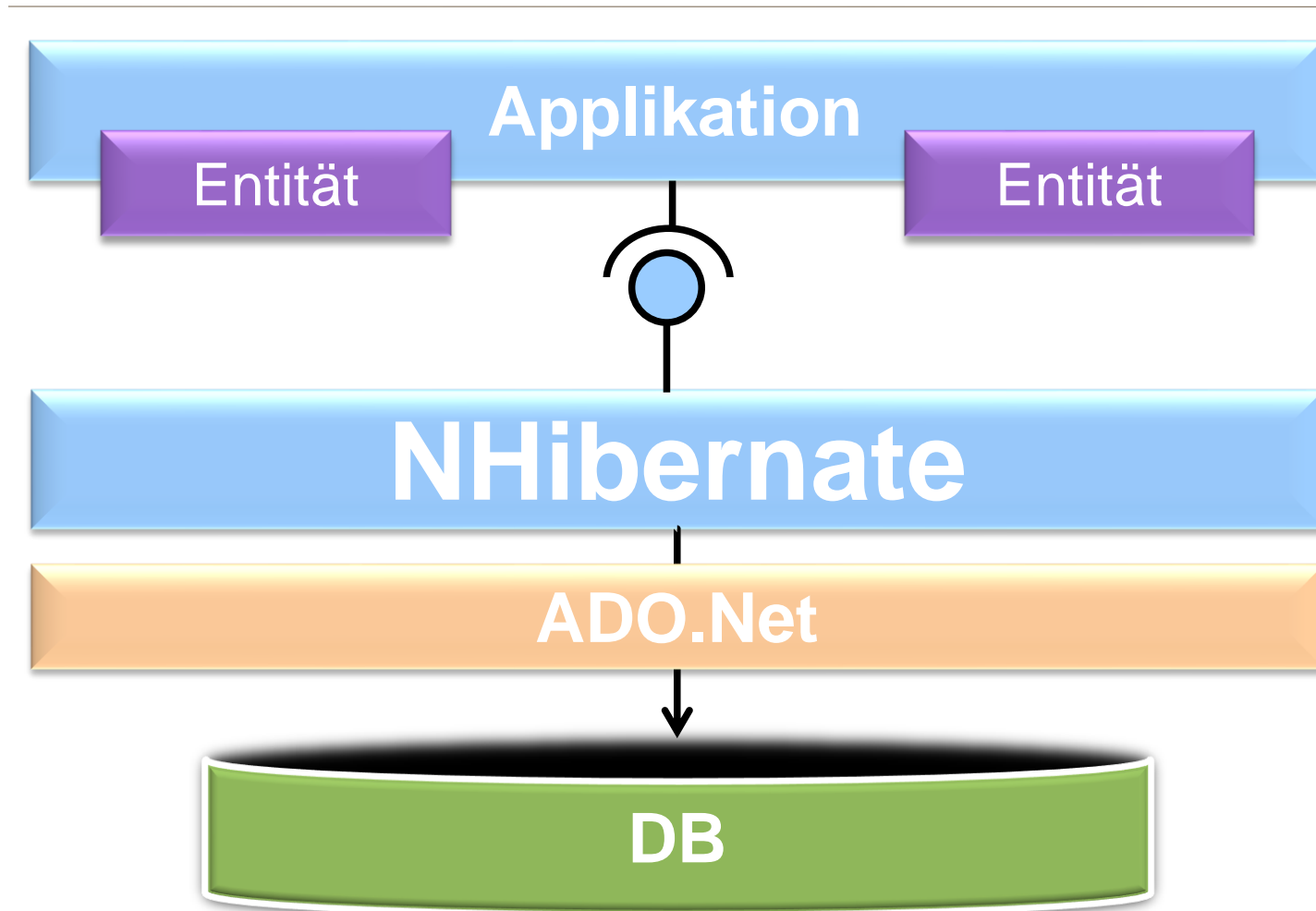
NHibernate

- ▶ Open Source
- ▶ Version 3.2 GA
- ▶ Unterstützt verschiedene Datenbanken
- ▶ NHibernate Attributes, Fluently NHibernate
- ▶ SQL Tracing leicht möglich

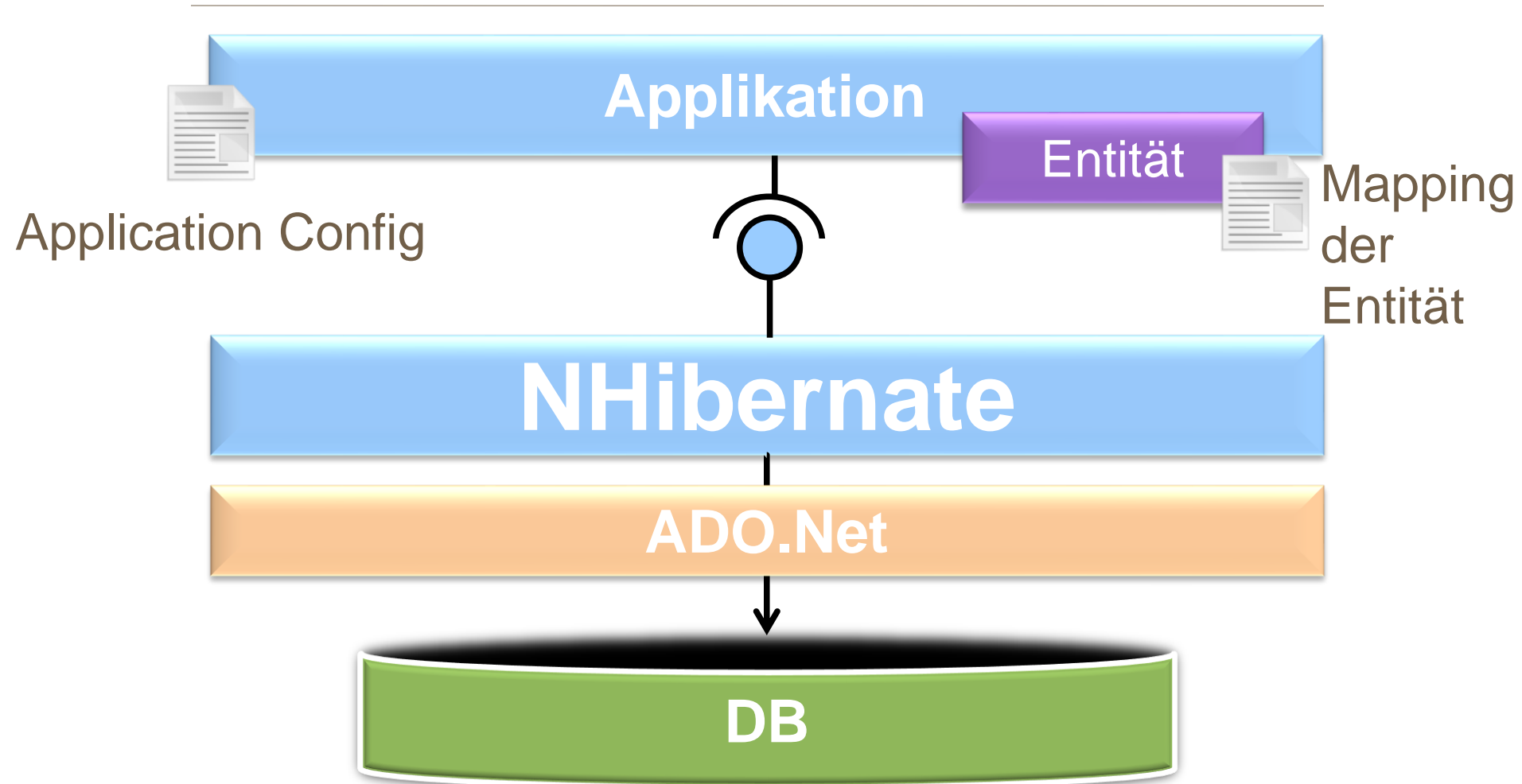
Entity Framework

- ▶ Closed Source
- ▶ Version 4.1 Update 1
- ▶ SQL Server, andere über Drittanbieter
- ▶ Code First Annotations, Code First Fluent API
- ▶ SQL Tracing eingeschränkt möglich
(`ToTraceString()`)

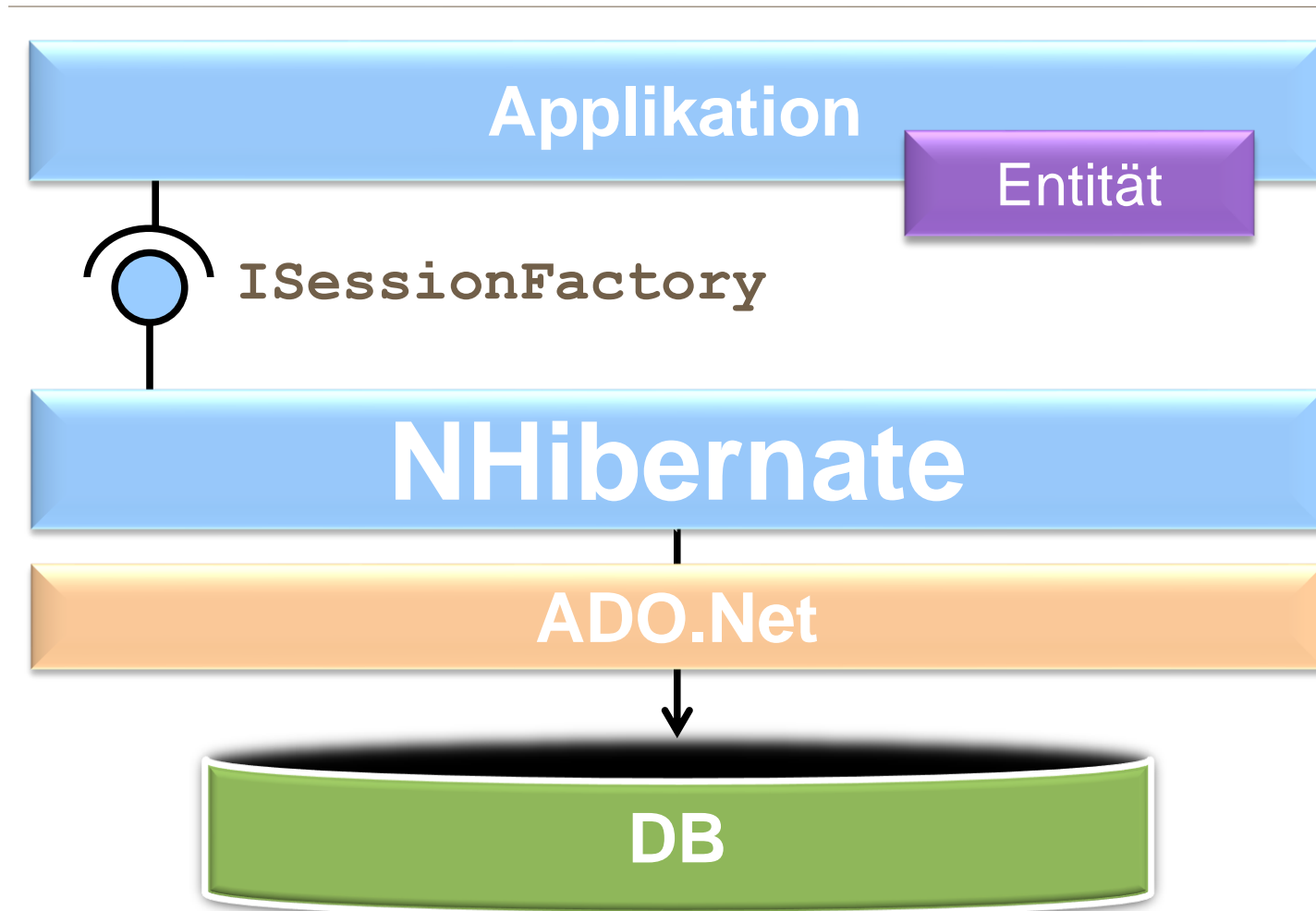
NHibernate Architektur



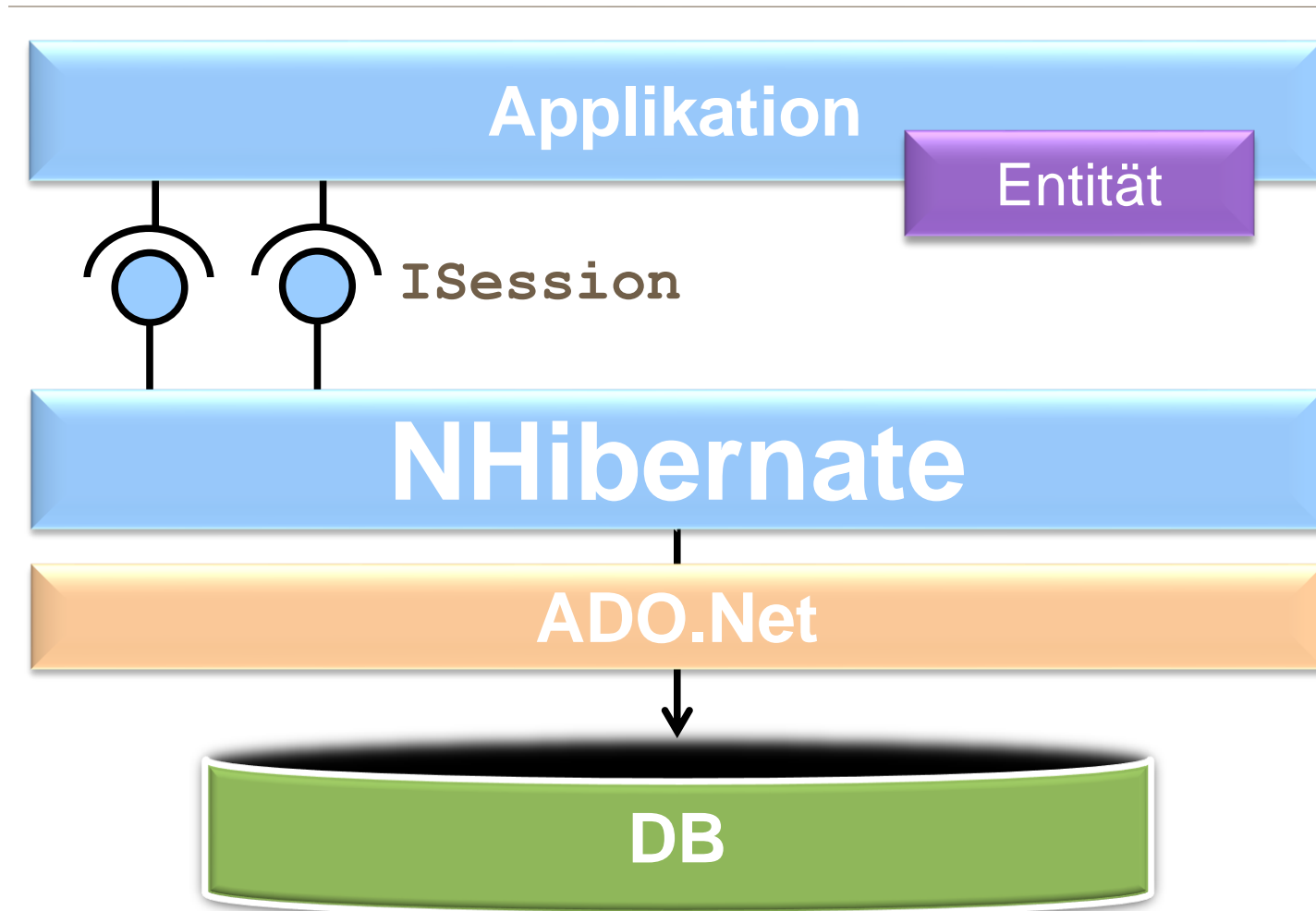
NHibernate Architektur



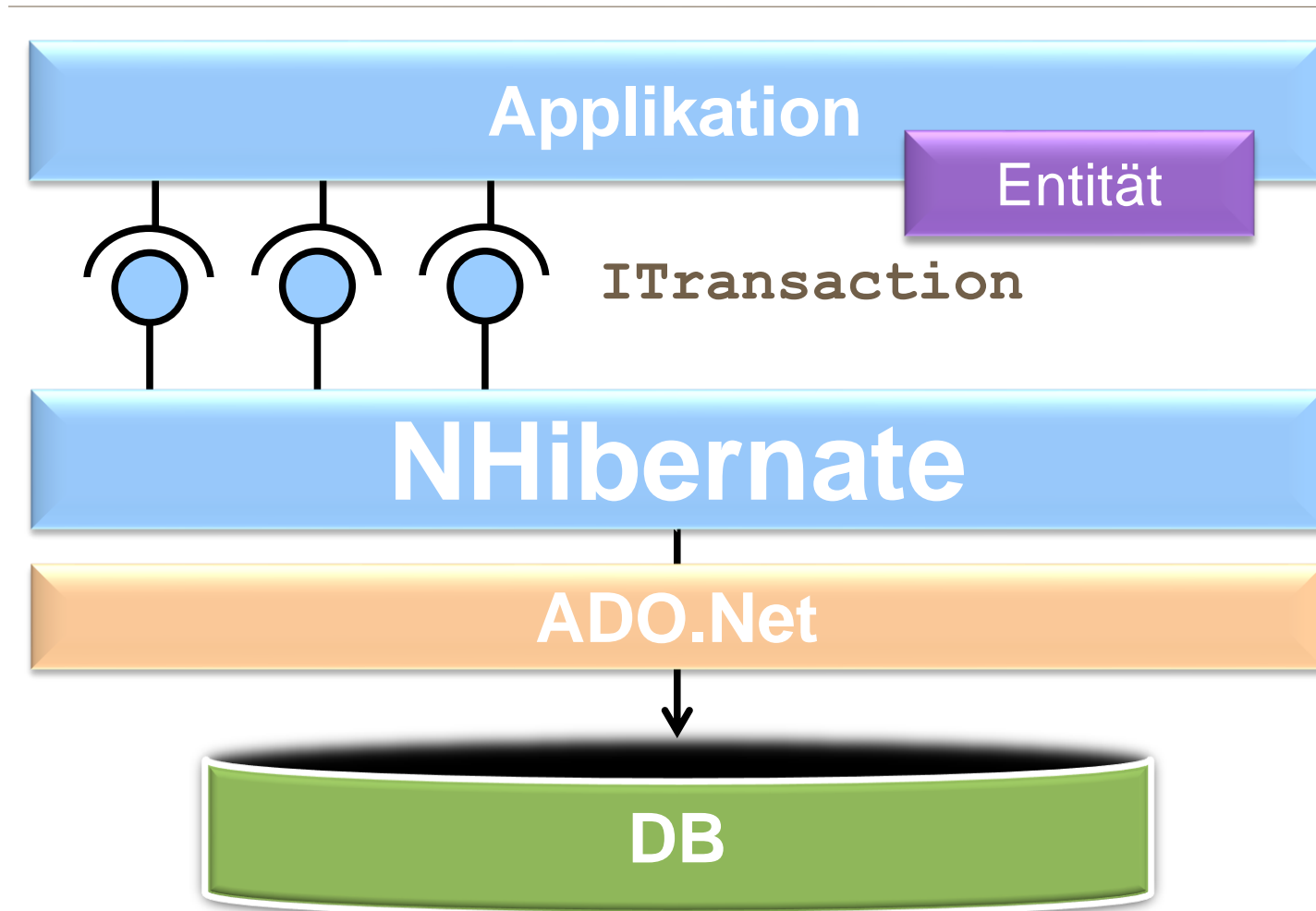
NHibernate Architektur



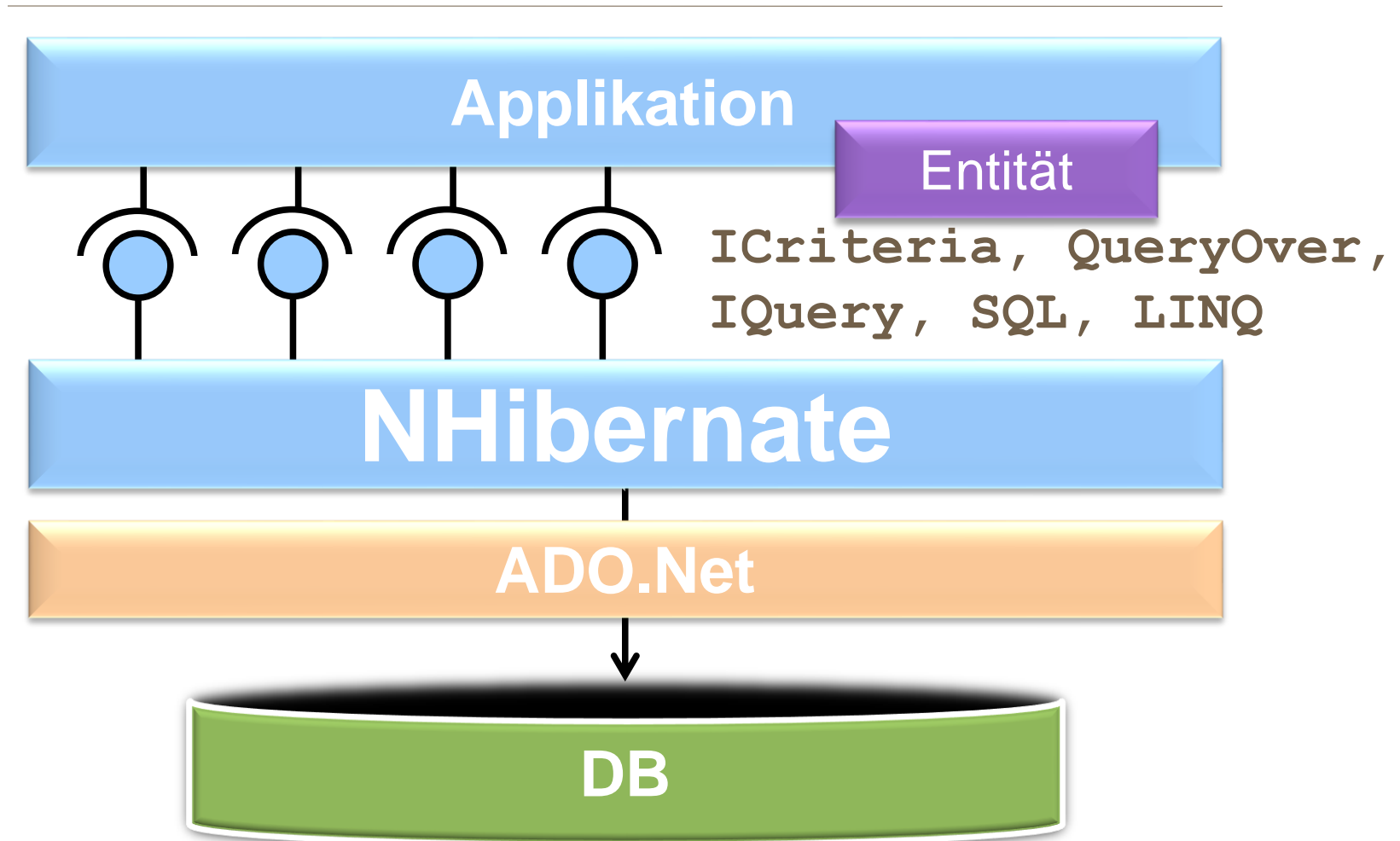
NHibernate Architektur



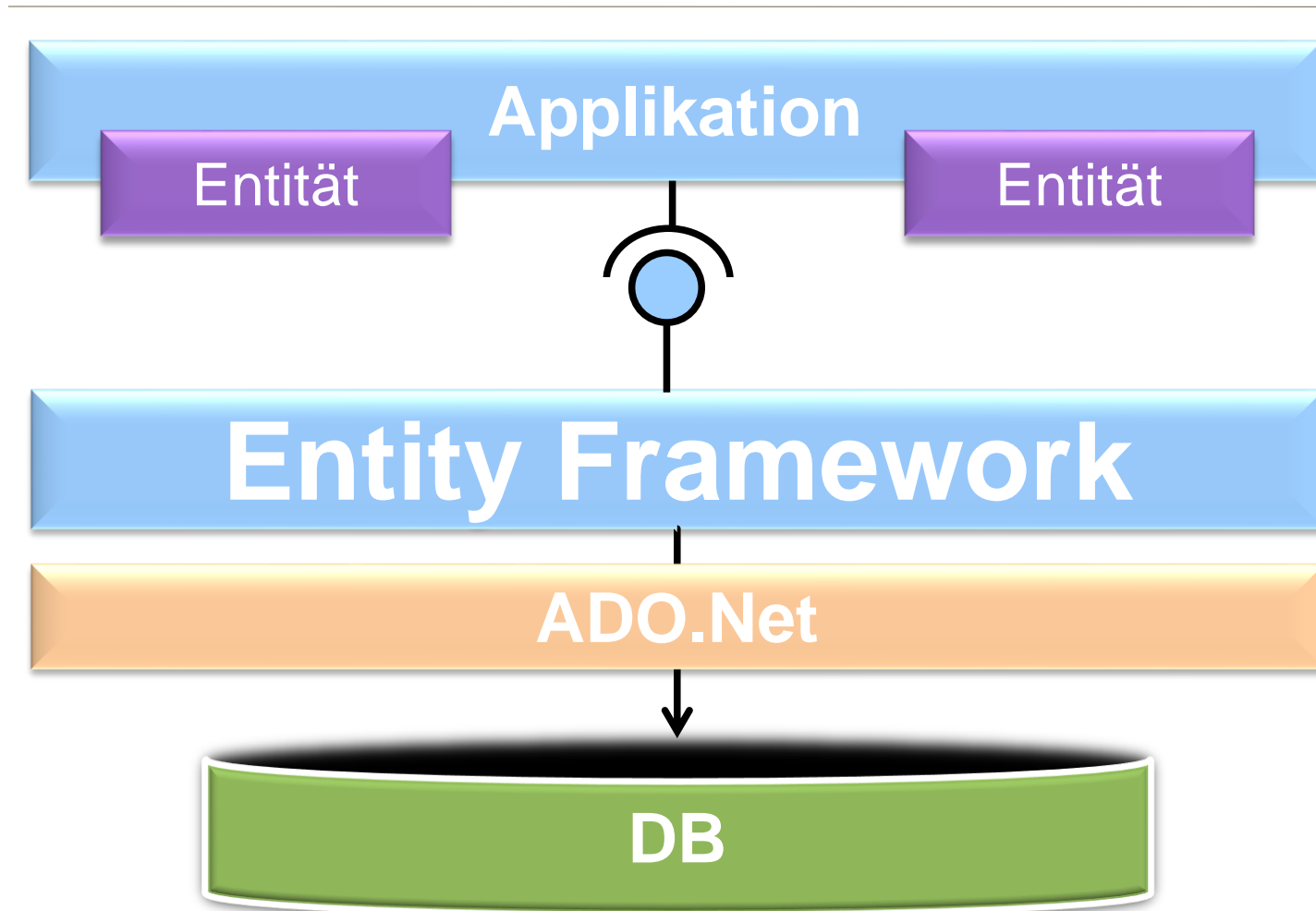
NHibernate Architektur



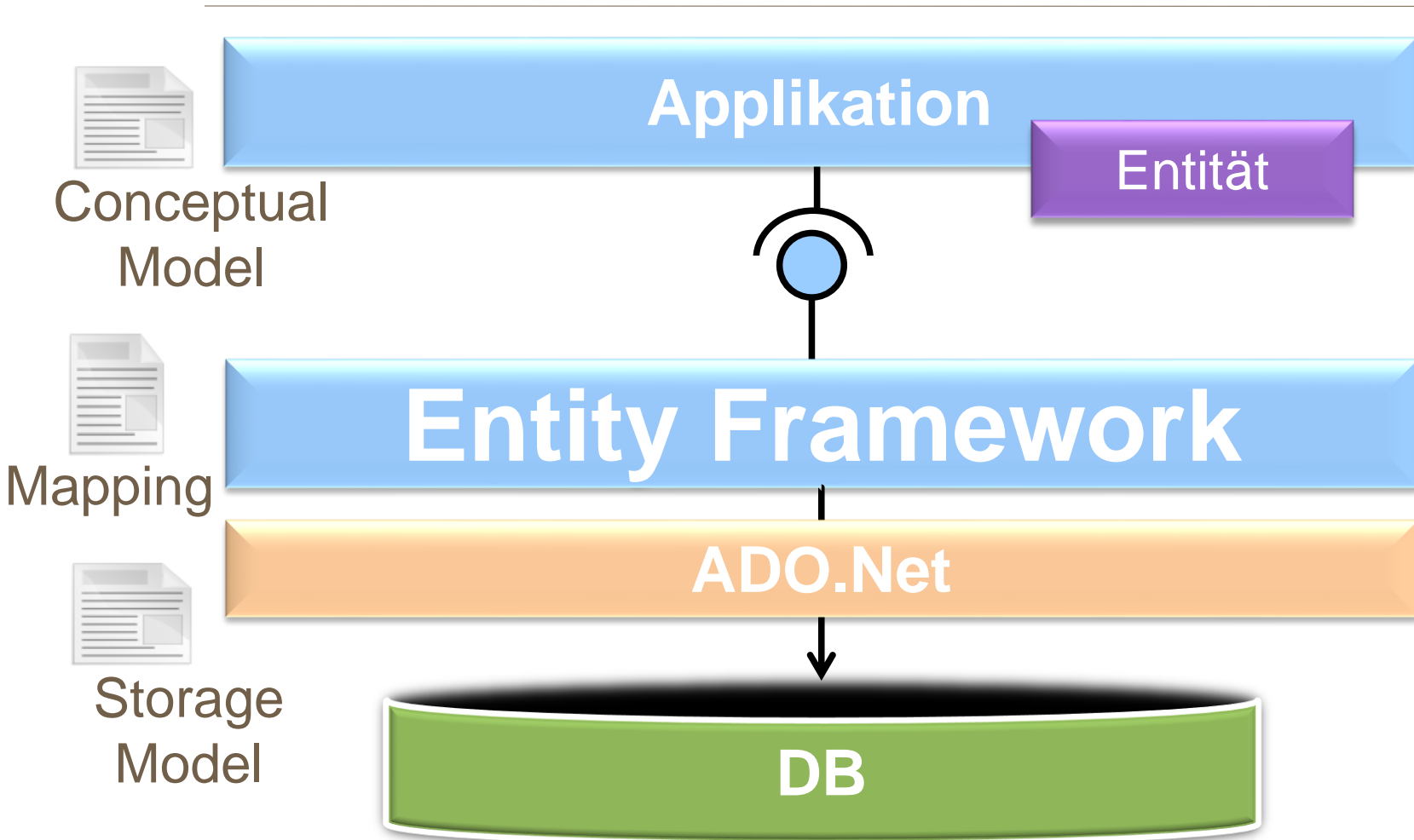
NHibernate Architektur



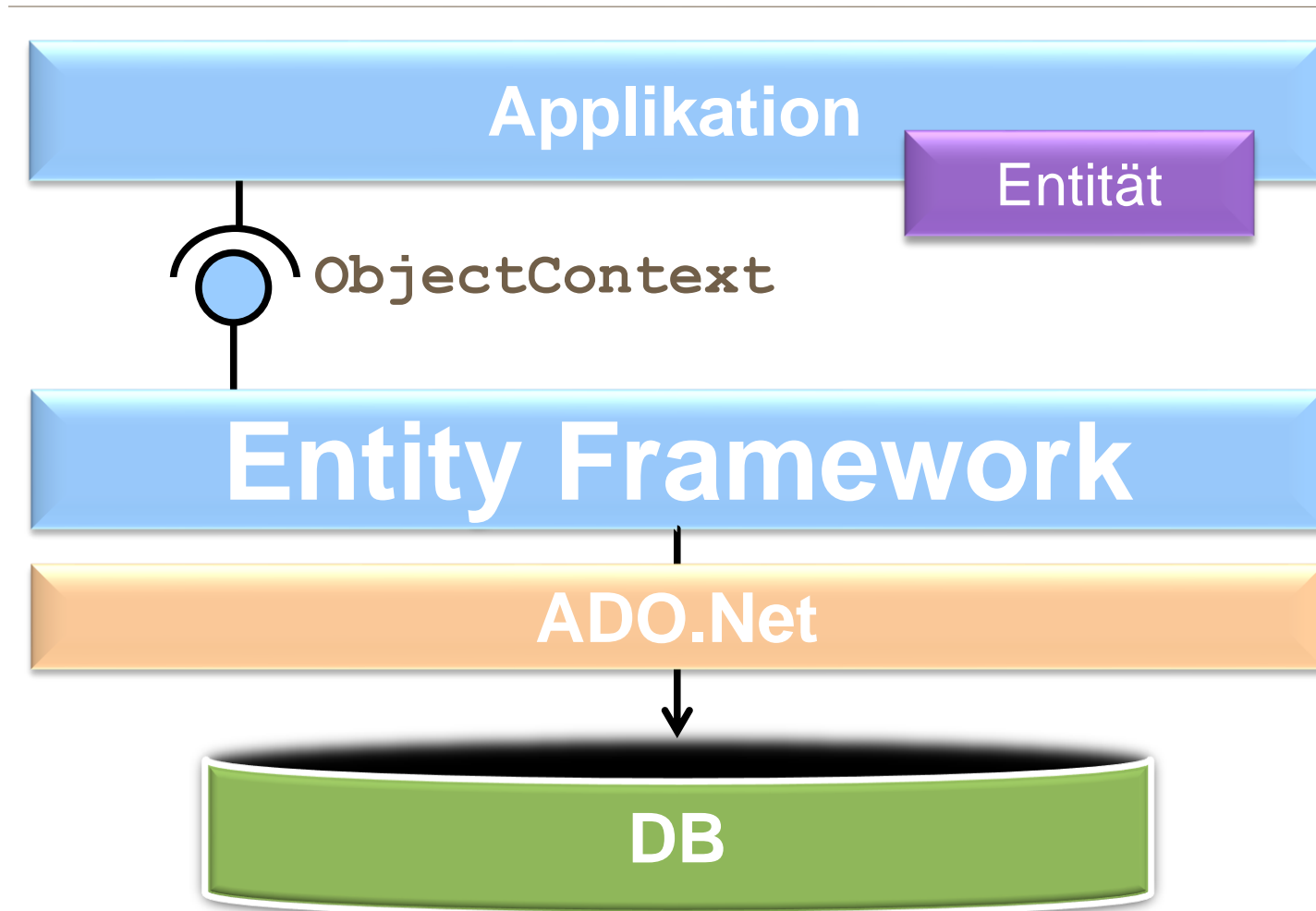
Entity Framework Architektur



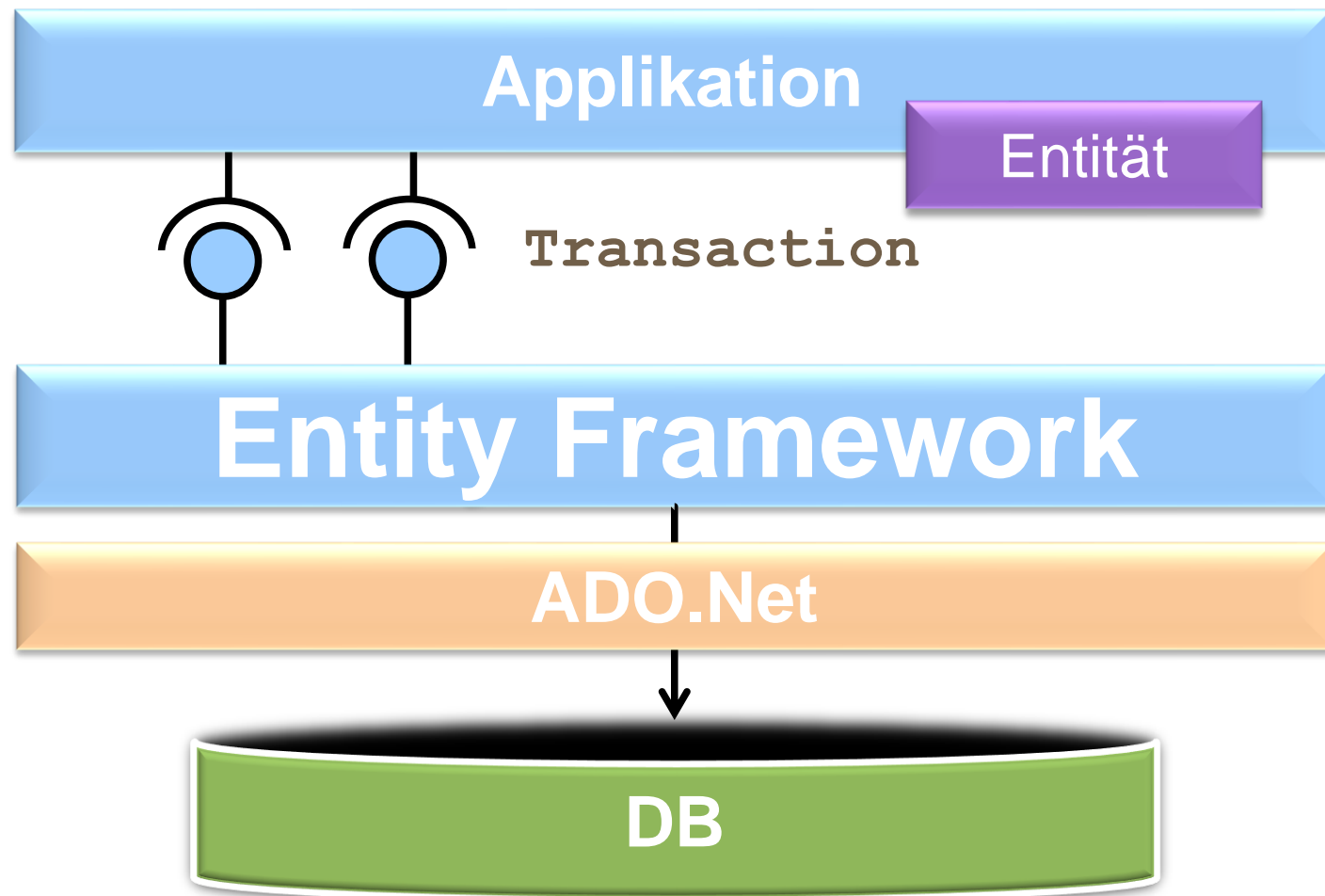
Entity Framework Architektur



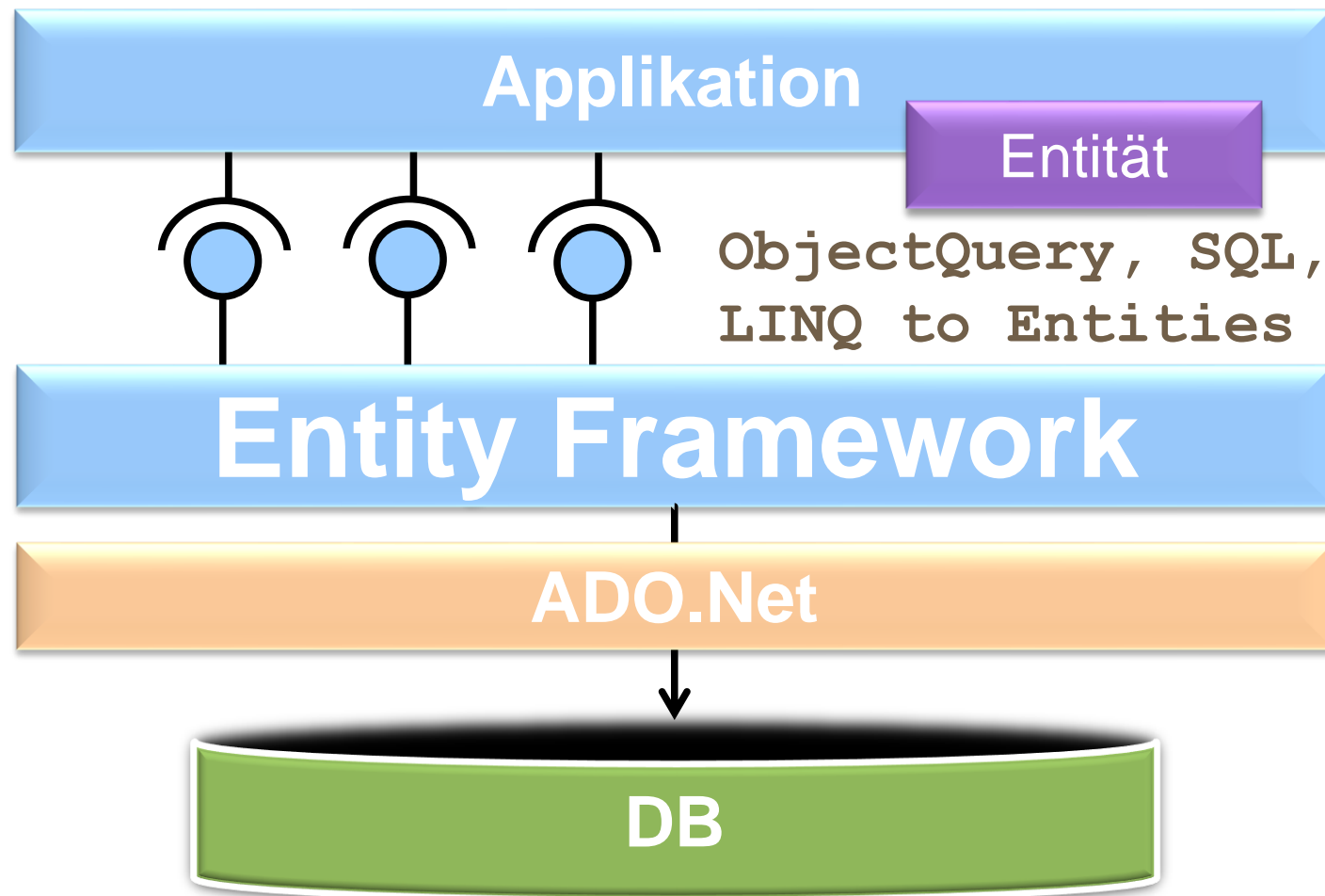
Entity Framework Architektur



Entity Framework Architektur



Entity Framework Architektur



Vergleich

NHibernate

- ▶ Configuration
- ▶ ISessionFactory
- ▶ ISession
- ▶ ITransaction
- ▶ Abfrage
 - ▶ ICriteria, QueryOver
 - ▶ IQuery
 - ▶ SQL
 - ▶ LINQ to NHibernate

Entity Framework

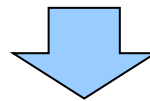
- ▶ Entity Data Model (EDMx)
- ▶ [ObjectContext]
- ▶ ObjectContext
- ▶ Transaction
- ▶ Abfrage
 - ▶ ObjectQuery
 - ▶ SQL
 - ▶ LINQ to Entities


Agenda

- ▶ (Architektur) Überblick
- ▶ Entitäten beschreiben
- ▶ CRUD
- ▶ Beziehungen
- ▶ Zusammenfassung



Beispiel



Simple_Categories				
	Column Name	Data Type	Nullable	Identity
	CategoryID	int	No	<input checked="" type="checkbox"/>
	name	nvarchar(50)	No	<input type="checkbox"/>
	description	nvarchar(200)	Yes	<input type="checkbox"/>
				<input type="checkbox"/>

Entitäts-Persistenz beschreiben

NHibernate

```
<hibernate-mapping
  xmlns="urn:nhibernate-mapping-2.2">

  <class name="Entities.Category,
    NHBeispielMitXML"
    table="Simple_Categories" >

    <id name="CategoryID"
      column="CategoryID"
      type="integer">

      <generator class="native" />
    </id>

    <property name="Name"
      column="name"
      type="String"
      length="50"/>

    <!-- weitere Properties -->

  </class>

</hibernate-mapping>
```

Entity Framework

```
<edmx:Edmx Version="2.0" ...>

  <edmx:ConceptualModels>

    <Schema Namespace="EFvsNHModel" Alias="Self"
      xmlns:store="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator"
      xmlns="http://schemas.microsoft.com/ado/2008/09/edm">

      <EntityContainer Name="EFvsNHEntities">

        <EntitySet Name="Simple_Categories" EntityType="EFvsNHModel.Category" />

      </EntityContainer>

      <EntityType Name="Category">

        <Key>

          <PropertyRef Name="CategoryID" />

        </Key>

        <Property Name="CategoryID" Type="Int32" Nullable="false"
          store:StoreGeneratedPattern="Identity" />

        <Property Name="Name" Type="String" Nullable="false" MaxLength="50" Unicode="true"
          FixedLength="false" />

        <Property Name="Description" Type="String" MaxLength="200" Unicode="true"
          FixedLength="false" />

      </EntityType>

    </Schema>

  </edmx:ConceptualModels>
```

Entitäts-Persistenz beschreiben

NHibernate

```
<hibernate-mapping
  xmlns="urn:nhibernate-mapping-2.2">

  <class name="Entities.Category,
    NHBeispielMitXML"
    table="Simple_Categories" >

    <id name="CategoryID"
      column="CategoryID"
      type="integer">

      <generator class="native" />
    </id>

    <property name="Name"
      column="name"
      type="String"
      length="50"/>

    <!-- weitere Properties -->

  </class>
</hibernate-mapping>
```

Entity Framework

```
<edmx:Edmx Version="2.0" ...>
  <edmx:Runtime>
    <edmx:StorageModels>

      <Schema Namespace="EFvsNHModel.Store" Alias="Self"
        Provider="System.Data.SqlClient" ProviderManifestToken="2008"

        <EntityContainer Name="EFvsNHModelStoreContainer">

          <EntitySet Name="Simple_Categories"
            EntityType="EFvsNHModel.Store.Simple_Categories" store:Type="Tables" Schema="dbo" />

        </EntityContainer>

        <EntityType Name="Simple_Categories">

          <Key>

            <PropertyRef Name="CategoryID" />

          </Key>

          <Property Name="CategoryID" Type="int" Nullable="false"
            StoreGeneratedPattern="Identity" />

          <Property Name="name" Type="nvarchar" Nullable="false" MaxLength="50" />

          <Property Name="description" Type="nvarchar" MaxLength="200" />

        </EntityType>

      </Schema>

    </edmx:StorageModels>
```


Entitäts-Persistenz beschreiben

NHibernate

```
<hibernate-mapping
  xmlns="urn:nhibernate-mapping-2.2">

  <class name="Entities.Category,
    NHBeispielMitXML"
    table="Simple_Categories" >

    <id name="CategoryID"
      column="CategoryID"
      type="integer">

      <generator class="native" />
    </id>

    <property name="Name"
      column="name"
      type="String"
      length="50"/>

    <!-- weitere Properties -->

  </class>
</hibernate-mapping>
```

Entity Framework

```
edmx:Edmx Version="2.0" ...>

  <edmx:Mappings>

    <Mapping Space="C-S"
      xmlns="http://schemas.microsoft.com/ado/2008/09/mapping/cs">

      <EntityContainerMapping
        StorageEntityContainer="EFvsNHModelStoreContainer"
        CdmEntityContainer="EFvsNHEntities">

        <EntitySetMapping Name="Categories"><EntityTypeMapping
          TypeName="EFvsNHModel.Category"><MappingFragment
            StoreEntitySet="Simple_Categories">

          />

          <ScalarProperty Name="CategoryID" ColumnName="CategoryID"

          <ScalarProperty Name="Name" ColumnName="name" />

          <ScalarProperty Name="Description"
            ColumnName="description" />

        </MappingFragment></EntityTypeMapping></EntitySetMapping>

      </EntityContainerMapping>

    </Mapping>

  </edmx:Mappings>
```

Entitäts-Persistenz beschreiben

NHibernate (App.config)

```
<configuration>
  <configSections>
    <section name="hibernate-
      configuration" type="Nhibernate.Cfg.ConfigurationSectionHandler,
        NHibernate"/>
  </configSections>

  <hibernate-configuration xmlns=".">
    <session-factory>
      <property name="connection.provider">
        NHibernate.Connection.DriverConnectionProvider
      </property>

      <property name="connection.driver_class">
        NHibernate.Driver.SqlClientDriver
      </property>

      <property name="show_sql">true</property>
    </session-factory>
  </hibernate-configuration>
</configuration>
```

...

Entitäts-Persistenz beschreiben

NHibernate (App.config, Fortsetzung)

...

```
<property name="connection.connection_string">
    server=localhost\SQLEXPRESS;
    Integrated Security=SSPI;
    database=EFvsNH
</property>

<property name="proxyfactory.factory_class">
    NHibernate.ByteCode.Castle.ProxyFactoryFactory,
    NHibernate.ByteCode.Castle
</property>

</session-factory>
</hibernate-configuration>
</configuration>
```

Entitäts-Persistenz Annotationen

NHibernate

```
[Class(Table="NH31_Category")]
public class Category
{
    [Id(Name = "CategoryId",
        Column="ID",
        Access = "nosetter.camelcase",
        UnsavedValue="-1")]
    [Generator(1,Class="native")]
    public virtual int Id {
        get { return id; }
    }

    [Property(Column = "NAME",
        Length=50,
        NotNull=true,
        Index="NAME_IDX")]
    public virtual string Name
        { get;set; }
}
```

Entity Framework

```
[Table("EF41_Category")]
public class Category
{
    [Key]
    [Column("CAT_ID")]
    [DatabaseGenerated(
        DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }

    [StringLength(20)]
    [Column("NAME")]
    [Required]
    public string Name { get; set; }
}
```

Entitäts-Persistenz Fluent API

NHibernate

```
public class CategoryMap :
    ClassMap<Category> {
    public CategoryMap()
    {
        Id(c => c.Category_Id)
            .Column("Cat_Id")
            .GeneratedBy.Identity();
        Map(c => c.tName)
            .Not.Nullable()
            .Length(50)
            .Column("Cat_Name")
    }
}
```

Entity Framework

```
public class BugShopContext : DbContext {
    protected override void
        OnModelCreating(DbModelBuilder builder)
    {
        EntityTypeConfiguration<Category>
            catConfig = builder.Entity<Category>();
        catConfig.HasKey<int>(c =>
            c.Category_Id);
        catConfig.Property(c => c.Category_Id)
            .HasDatabaseGeneratedOption(
                DatabaseGeneratedOption.Identity)
            .HasColumnName("Cat_Id");
        catConfig.Property(c => c.Name)
            .IsRequired()
            .HasMaxLength(50)
            .HasColumnName("Cat_Name");
    }
}
```

Entitäts-Persistenz Fluent API

NHibernate

```
public class Program {  
    private ISessionFactory  
        BuildSessionFactory() {  
        string dbUrl= "...";  
  
        FluentConfiguration cfg =  
            Fluently.Configure()  
                .Database(FluentNHibernate.Cfg.  
                    Db.MsSqlConfiguration.MsSql2008  
                    .ConnectionString(dbUrl)  
                    .Driver<NHibernate.Driver.  
                        SqlClientDriver>());  
  
        return cfg.Mappings(m =>  
            m.FluentMappings.  
                Add(typeof(CategoryMap))  
                .BuildSessionFactory());  
    }  
}
```

Entity Framework

```
public class BugShopContext : DbContext {  
    protected override void  
        OnModelCreating(DbModelBuilder builder)  
    {  
        EntityTypeConfiguration<Category>  
            catConfig = builder.Entity<Category>();  
  
        catConfig.HasKey<int>(c =>  
            c.Category_Id);  
  
        catConfig.Property(c => c.Category_Id)  
            .HasDatabaseGeneratedOption(  
                DatabaseGeneratedOption.Identity)  
            .HasColumnName("Cat_Id");  
  
        catConfig.Property(c => c.Name)  
            .IsRequired()  
            .HasMaxLength(50)  
            .HasColumnName("Cat_Name");  
    }  
}
```

Agenda

- ▶ (Architektur) Überblick
- ▶ Entitäten beschreiben
- ▶ CRUD
- ▶ Beziehungen
- ▶ Zusammenfassung



Create RUD

NHibernate

```
ISessionFactory sF =
    CreateSessionFactory();

using (ISession session =
    sF.OpenSession()) {
    using (ITransaction tx =
        session.BeginTransaction()) {
        Category myCat = new
            Category("Motorteile");
        myCat.Description = "Alles";
        session.Save(myCat);
        [session.Flush();]
        tx.Commit();
    }
}
```

Entity Framework

```
using (EFvsNHEntities ctx =
    new EfvsNHEntities()) {
    Category myCat = new
        Category();
    myCat.Name = "Motorteile";
    ctx.Categories.
        AddObject(myCat);
    ctx.SaveChanges();
}
```


C Read UD

NHibernate

```
using (ISession session =
    sF.OpenSession()) {
    using (ITransaction tx =
        session.BeginTransaction()) {

        Category gefCat =

        //erste Alternative
        session.Load<Category>(myCat.←
            CategoryID);

        //zweite Alternative
        session.Get<Category>(myCat.←
            CategoryID);

    }
}
```

Entity Framework

```
using (EFvsNHEntities ctx =
    new EfvsNHEntities()) {

    EntityKey key =
    new EntityKey("EFvsNH.Categories"
        , "CategoryID"
        , myCat.CategoryID);

    Category gefCat = (Category)
        ctx.GetObjectByKey(key) ;

    bool gefunden =
        context.TryGetObjectByKey(key,
            out gefundeneCategory) ;

}
```

CR Update D

NHibernate

```
using (ISession s =
    sF.OpenSession()) {
    using (ITransaction tx =
        s.BeginTransaction()) {
        // bereits existierende, aber
        // nicht 'attached' Entität

        myCat.Name = "neuer Name";

        // erste Alternative
        session.Update(myCat);

        // zweite Alternative
        session.SaveOrUpdate(myCat);
    }
}
```

Entity Framework

```
using (EFvsNHEntities ctx =
    new EfvsNHEntities()) {

    // bereits existierende, aber
    // nicht 'attached' Entität

    ctx.Attach(myCat);

    myCat.Name = "neuer Name";

    ctx.SaveChanges();
}
```

CR Update D

NHibernate

```
using (ISession s =
    sF.OpenSession()) {
    using (ITransaction tx =
        s.BeginTransaction()) {

        //dritte Alternative
        Category myCat =
            session.Load<Category>
                (myCat.CategoryID) ;

        myCat.Name = "neuer Name";
        tx.Commit();
    }
}
```

Entity Framework

```
using (EFvsNHEntities ctx =
    new EfvsNHEntities()) {

    // bereits existierende, aber
    // nicht 'attached' Entität

    ctx.Attach(myCat) ;

    myCat.Name = "neuer Name";

    ctx.SaveChanges() ;
}
```

CRU Delete

NHibernate

```
using (ISession session =
    sF.OpenSession()) {
    using (ITransaction tx =
        session.BeginTransaction()) {

        session.Delete(myCat);

        tx.Commit();
    }
}
```

Entity Framework

```
using (EFvsNHEntities ctx =
    new EfvsNHEntities()) {

    //Object in Context laden ...

    ctx.DeleteObject(myCat);

    ctx.SaveChanges();
}
```

Agenda

- ▶ (Architektur) Überblick
- ▶ Entitäten beschreiben
- ▶ CRUD
- ▶ Beziehungen
- ▶ Zusammenfassung



Beziehungskiste

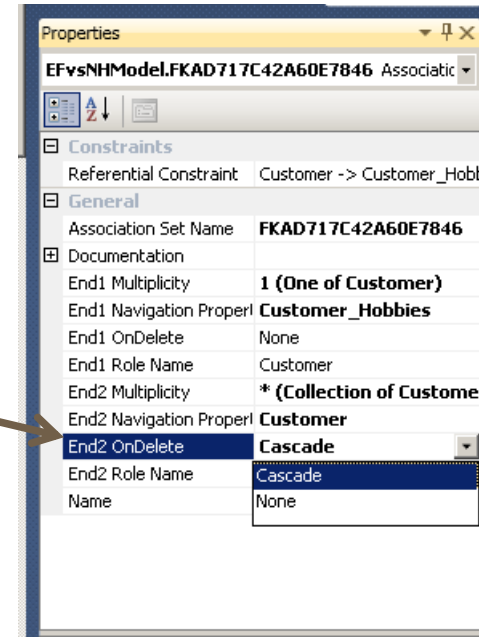
- ▶ Persistence-by-Reachability

Beziehungskiste

▶ Persistence-by-Reachability

▶ Entity Framework

- ▶ „save-update“ | delete



Beziehungskiste

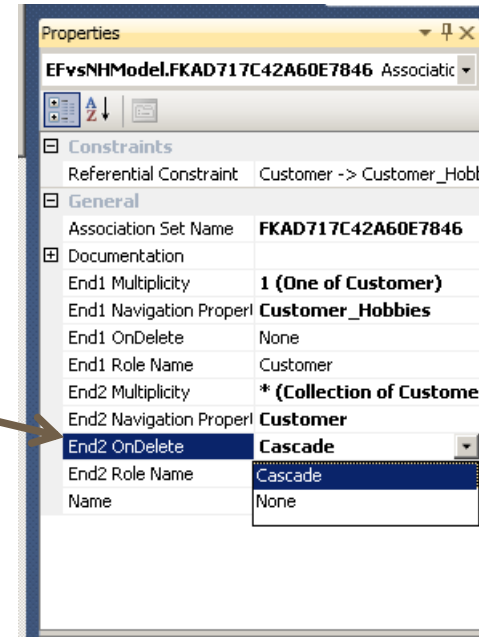
- ▶ Persistence-by-Reachability

- ▶ Entity Framework

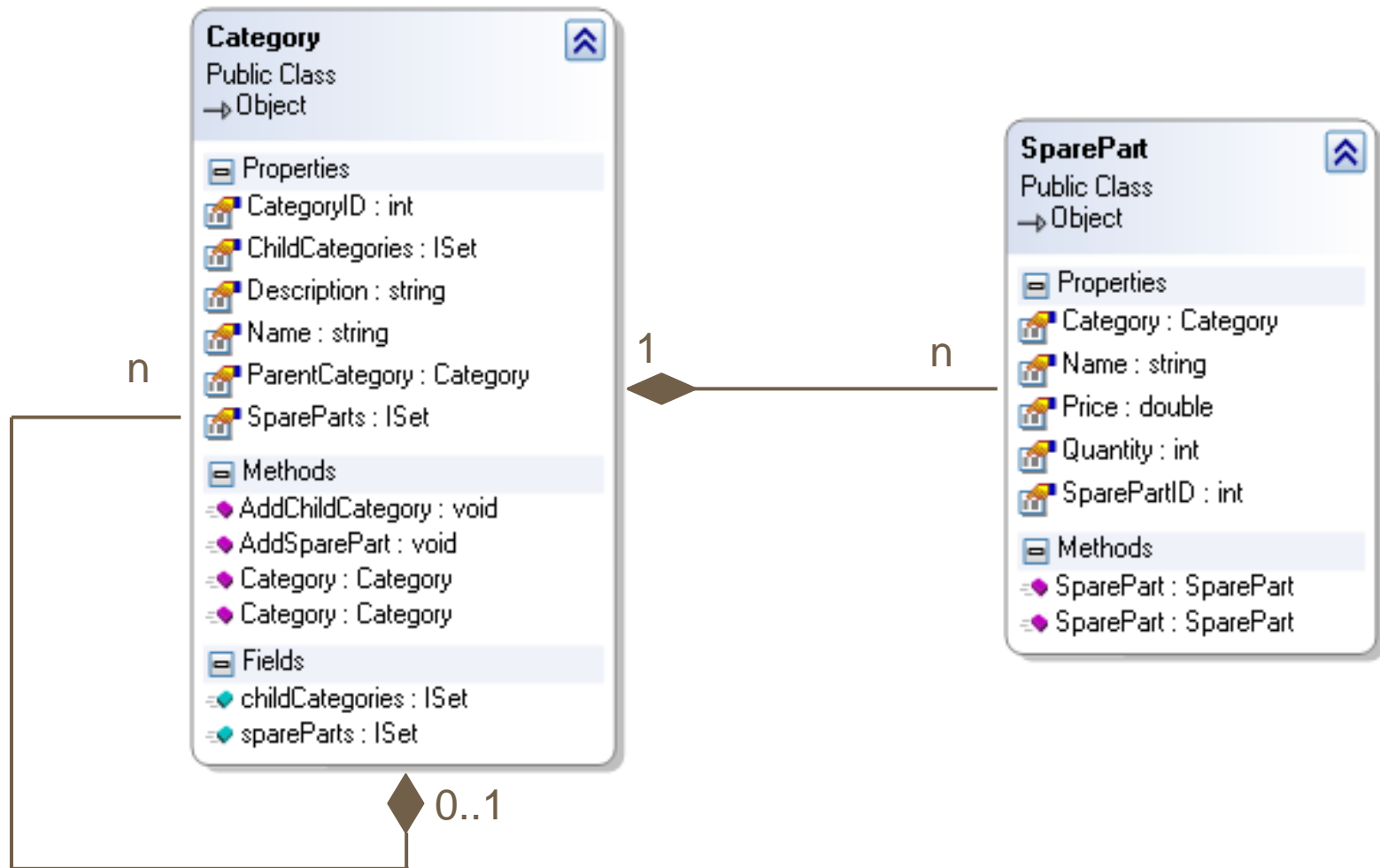
- ▶ „save-update“ | delete

- ▶ NHibernate

- ▶ cascade = none | save-update | delete | all



Beziehungskiste – Beispiel



Beispiel - Konfiguration

NHibernate

```

<hibernate-mapping
  xmlns="urn:nhibernate-mapping-2.2">

  <class name="Entities.Category,
    NHBeispielMitXML"
    table="Categories" >

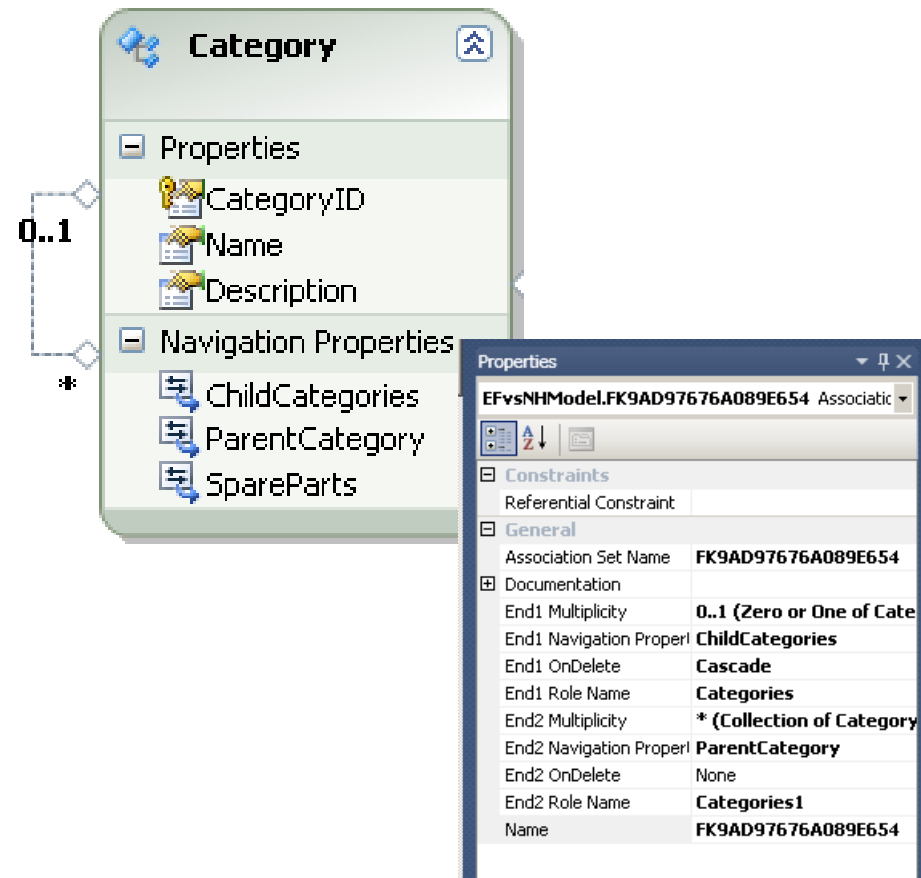
    <set name="ChildCategories"
      table="Categories"
      cascade="all"
      inverse="true">

      <key
        column="PARENT_CATEGORY_ID"/>
      <one-to-many
        class="Entities.Category" />
    </set>

    <many-to-one
      name="ParentCategory"
      class="Entities.Category"
      column="PARENT_CATEGORY_ID"
      cascade="none"/>
  </class>
</hibernate-mapping>

```

Entity Framework



Beispiel - Konfiguration

NHibernate

```
<hibernate-mapping
  xmlns="urn:nhibernate-mapping-2.2">
  <class name="Entities.Category,
    NHBeispielMitXML"
    table="Categories" >
    <set name="ChildCategories"
      table="Categories"
      cascade="all"
      inverse="true">
      <key
        column="PARENT_CATEGORY_ID"/>
      <one-to-many
        class="Entities.Category" />
    </set>
    <many-to-one
      name="ParentCategory"
      class="Entities.Category"
      column="PARENT_CATEGORY_ID"
      cascade="none"/>
  </class>
</hibernate-mapping>
```

Entity Framework

```
<edmx:Edmx Version="2.0" ...>
  <edmx:ConceptualModels>
    <AssociationSet Name="FK9..654"
      Association="EFvsNHModel.FK9...654">
      <End Role="ChildCategories"
        EntitySet="Categories" />
      <End Role="ParentCategory"
        EntitySet="Categories" />
    </AssociationSet>
    <Association Name="FK9...654">
      <End Role="ChildCategories"
        Type="EFvsNHModel.Category"
        Multiplicity="0..1" >
      <OnDelete Action="Cascade"/>
    </End>
      <End Role="ParentCategory"
        Type="EFvsNHModel.Category"
        Multiplicity="*" />
    </Association>
  </edmx:ConceptualModels>
```

Beispiel - Konfiguration

NHibernate

```

<hibernate-mapping
  xmlns="urn:nhibernate-mapping-2.2">

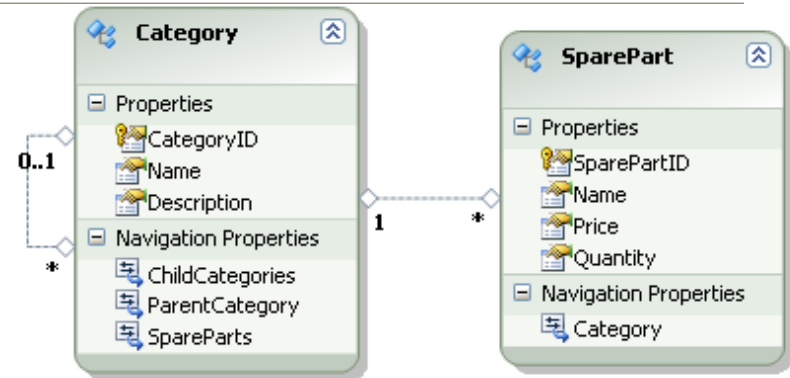
  <class name="Entities.Category" >

    <set name="SpareParts"
      table="SPARE_PART"
      cascade="all"
      inverse="true">
      <key column="CATEGORY_ID"/>
      <one-to-many
        class="Entities.SparePart" />
    </set>

  <class name="Entities.SparePart" >

    <many-to-one name="Category"
      class="Entities.Category"
      column="CATEGORY_ID"
      not-null="true" />
  </class>
</hibernate-mapping>
  
```

Entity Framework



Properties	
EFvsNHModel.FKA14325FA4DFEDC56 Associati...	
<input type="checkbox"/> Constraints Referential Constraint	
<input type="checkbox"/> General	
Association Set Name	FKA14325FA4DFEDC56
<input type="checkbox"/> Documentation	
End1 Multiplicity	1 (One of Category)
End1 Navigation Properl	SpareParts
End1 OnDelete	Cascade
End1 Role Name	Categories
End2 Multiplicity	*(Collection of SparePar
End2 Navigation Properl	Category
End2 OnDelete	None
End2 Role Name	SpareParts
Name	FKA14325FA4DFEDC56

Beispiel - Konfiguration

NHibernate

```
<hibernate-mapping
  xmlns="urn:nhibernate-mapping-2.2">
  <class name="Entities.Category" >
    <set name="SpareParts"
      table="SPARE_PART"
      cascade="all"
      inverse="true">
      <key column="CATEGORY_ID"/>
      <one-to-many
        class="Entities.SparePart" />
    </set>

    <class name="Entities.SparePart" >
      <many-to-one name="Category"
        class="Entities.Category"
        column="CATEGORY_ID"
        not-null="true"/>
    </class>
  </class>
</hibernate-mapping>
```

Entity Framework

```
<edmx:Edmx Version="2.0" ...>
  <edmx:ConceptualModels>
    <AssociationSet Name="FKA...C56"
      Association="EFvsNHModel.FKA...C56">
      <End Role="Categories"
        EntitySet="Categories" />

      <End Role="SpareParts"
        EntitySet="SpareParts" />
    </AssociationSet>

    <Association Name="FKA...C56">
      <End Role="Categories"
        Type="EFvsNHModel.Category"
        Multiplicity="1" >
      <OnDelete Action="Cascade" />
    </End>
    <End Role="SpareParts"
      Type="EFvsNHModel.SparePart"
      Multiplicity="*" >
    </End>
  </edmx:ConceptualModels>
```

Beispiel – Speichern von Entitäten

NHibernate

Entity Framework

```
Category motorCategory = new Category("Motorteile");
motorCategory.Description = "Alles für den Käfer Motor";

Category category1300er = new Category("Motorteile - 1300cc");
category1300er.Description = "Standard Teile für 1300cc Motor";

motorCategory.AddChildCategory(category1300er);

SparePart vergaser = new SparePart("Solex Vergaser 34PCI", 300.0, 1);
category1300er.AddSparePart(vergaser);

SparePart zuendverteiler = new SparePart("Zündverteiler", 200.0, 5);
category1300er.AddSparePart(zuendverteiler);

SparePart benzinPumpe = new SparePart("Benzinpumpe", 50.0, 3);
category1300er.AddSparePart(benzinPumpe);
```

```
using (ITransaction tx =
    session.BeginTransaction()) {

    session.Save(rootCategory);
    tx.Commit();

}
```

```
using (EFvsNHEntities ctx =
    new EfvsNHEntities()) {

    ctx.Categories.AddObject(rootCategory);
    context.SaveChanges();

}
```

Beispiel – Speichern von Entitäten

NHibernate

Entity Framework

```
select * from Categories
select * from SpareParts
```

Results		Messages			
	CategoryID	name	description	PARENT_CATEGORY_ID	
1	1	Motorteile	Alles für den Käfer Motor	NULL	
2	2	Motorteile - 1300cc	Standard Teile für 1300cc Motor	1	

	SparePartID	NAME	PRICE	QUANTITY	CATEGORY_ID
1	1	Benzinpumpe	50	3	2
2	2	Solex Vergaser 34PCI	300	1	2
3	3	Zündverteiler Bosch 009	200	5	2

Beispiel – Eager Fetching

NHibernate

ICriteria

```
ICriteria criteria =
    session.CreateCriteria<Category>();

criteria.Add(Expression.Eq("Name",
    "Motorteile - 1300cc"));

IList<Category> list =
    criteria.list();
```

QueryOver

```
IQueryOver<Category> queryOver =
    session.QueryOver<Category>();

queryOver.Where(c => c.Name ==
    "Motorteile - 1300cc");

IList<Category> list =
    queryOver.List();
```

SQL

```
SELECT
    this_.CategoryID as CategoryID0_0_,
    this_.name as name0_0_,
    this_.description as descript3_0_0_,
    this_.PARENT_CATEGORY_ID as PARENT4_0_0_
FROM
    Categories this_
WHERE
    this_.name = @p0; @p0 = 'Motorteile - 1300cc'

SELECT
    spareparts0_.CATEGORY_ID as CATEGORY5_1_,
    spareparts0_.SparePartID as SparePar1_1_,
    spareparts0_.SparePartID as SparePar1_1_0_,
    spareparts0_.NAME as NAME1_0_,
    spareparts0_.PRICE as PRICE1_0_,
    spareparts0_.QUANTITY as QUANTITY1_0_,
    spareparts0_.CATEGORY_ID as CATEGORY5_1_0_
FROM
    SpareParts spareparts0_
WHERE spareparts0_.CATEGORY_ID=@p0; @p0 = 2
```


Beispiel – Eager Fetching

Entity Framework

SQL

Entity SQL

```
string query =  
    @"select value c  
      from Categories as c  
     where c.Name =  
           'Motorteile - 1300cc';  
  
ctx.CreateQuery<Category>(query)  
    .Include("SpareParts");
```

LINQ to Entities

```
ctx.Categories.Include("SpareParts")  
    .Where(c => c.Name ==  
           "Motorteile - 1300cc");  
  
ctx.Categories.Include("SpareParts")  
    .Where("it.Name =  
          'Motorteile - 1300cc '");
```

SELECT

```
[Project1].[CategoryID] AS [CategoryID],  
[Project1].[name] AS [name],  
[Project1].[description] AS [description],  
[Project1].[PARENT_CATEGORY_ID] AS [PARENT_CATEGORY_ID],  
[Project1].[C1] AS [C1],  
[Project1].[SparePartID] AS [SparePartID],  
[Project1].[name1] AS [name1],  
[Project1].[PRICE] AS [PRICE],  
[Project1].[QUANTITY] AS [QUANTITY],  
[Project1].[CATEGORY_ID] AS [CATEGORY_ID]  
FROM ( SELECT  
  
        [Extent1].[CategoryID] AS [CategoryID],  
        [Extent1].[name] AS [name],  
        [Extent1].[description] AS [description],  
        [Extent1].[PARENT_CATEGORY_ID] AS [PARENT_CATEGORY_ID],  
        [Extent2].[SparePartID] AS [SparePartID],  
        [Extent2].[NAME] AS [name1],  
        [Extent2].[PRICE] AS [PRICE],  
        [Extent2].[QUANTITY] AS [QUANTITY],  
        [Extent2].[CATEGORY_ID] AS [CATEGORY_ID],  
        CASE WHEN ([Extent2].[SparePartID] IS NULL) THEN CAST(NULL AS int) ELSE  
1 END AS [C1]  
FROM    [dbo].[Categories] AS [Extent1]  
LEFT OUTER JOIN [dbo].[SpareParts] AS [Extent2] ON  
[Extent1].[CategoryID] = [Extent2].[CATEGORY_ID]  
        WHERE [Extent1].[name] =  
              'Motorteile - 1300cc'  
    ) AS [Project1]  
ORDER BY [Project1].[CategoryID] ASC, [Project1].[C1] ASC;
```

Beispiel – Eager Fetching

NHibernate

SQL

Konfiguration angepasst

```
<class name="Entities.Category" >
  <set name="SpareParts"
    table="SPARE_PART"
    cascade="all"
    ➔ fetch = "join" ←
    inverse="true">
    <key column="CATEGORY_ID"/>
    <one-to-many
      class="Entities.SparePart" />
    </set>
```

QueryOver

```
IQueryOver<Category> queryOver =
  session.QueryOver<Category>();
queryOver.Where(c => c.Name ==
  "Motorteile - 1300cc");
IList<Category> list =
  queryOver.List();
```

```
SELECT
  this_.CategoryID as CategoryID0_1_,
  this_.name as name0_1_,
  this_.description as descript3_0_1_,
  this_.PARENT_CATEGORY_ID as PARENT4_0_1_,
  spareparts2_.CATEGORY_ID as CATEGORY5_3_,
  spareparts2_.SparePartID as SparePar1_3_,
  spareparts2_.SparePartID as SparePar1_1_0_,
  spareparts2_.NAME as NAME1_0_,
  spareparts2_.PRICE as PRICE1_0_,
  spareparts2_.QUANTITY as QUANTITY1_0_,
  spareparts2_.CATEGORY_ID as CATEGORY5_1_0_
FROM Categories this_
left outer join SpareParts spareparts2_ ←
on
  this_.CategoryID=spareparts2_.CATEGORY_ID
WHERE
  this_.name = @p0; @p0 = 'Motorteile - 1300cc'
```

Beispiel – Eager Fetching

NHibernate

SQL

HQL

(Ignoriert fetch Einstellung
in der Konfiguration)

```
IQuery query = session.CreateQuery(
    "SELECT c FROM Category c
    left join fetch c.SpareParts part
    WHERE c.Name =
        'Motorteile - 1300cc'
");
IList<Category> list =
    query.List<Category>();
```

ABER...

```
select
    category0_.CategoryID as CategoryID0_0_,
    spareparts1_.SparePartID as SparePar1_1_1_,
    category0_.name as name0_0_,
    category0_.description as descript3_0_0_,
    category0_.PARENT_CATEGORY_ID as PARENT4_0_0_,
    spareparts1_.NAME as NAME1_1_,
    spareparts1_.PRICE as PRICE1_1_,
    spareparts1_.QUANTITY as QUANTITY1_1_,
    spareparts1_.CATEGORY_ID as CATEGORY5_1_1_,
    spareparts1_.CATEGORY_ID as CATEGORY5_0_,
    spareparts1_.SparePartID as SparePar1_0_
from
    Categories category0_
left outer join
    SpareParts spareparts1_
    on
    category0_.CategoryID=spareparts1_.CATEGORY_ID
where
    category0_.name='Motorteile - 1300cc'
```

Beispiel – Eager Fetching

NHibernate

ABER...

HQL

```
IList<Category> list =  
    query.List<Category>();
```

QueryOver

```
IList<Category> list =  
    queryOver.List();
```

```
Console.WriteLine("Anzahl: {0}",  
    list.Count);
```

→ Ausgabe : 3

SQL

```
select  
    category0_.CategoryID as CategoryID0_0_,  
    spareparts1_.SparePartID as SparePar1_1_1_,  
    category0_.name as name0_0_,  
    category0_.description as descript3_0_0_,  
    category0_.PARENT_CATEGORY_ID as PARENT4_0_0_,  
    spareparts1_.NAME as NAME1_1_,  
    spareparts1_.PRICE as PRICE1_1_,  
    spareparts1_.QUANTITY as QUANTITY1_1_,  
    spareparts1_.CATEGORY_ID as CATEGORY5_1_1_,  
    spareparts1_.CATEGORY_ID as CATEGORY5_0_,  
    spareparts1_.SparePartID as SparePar1_0_  
from  
    Categories category0_  
left outer join  
    SpareParts spareparts1_  
    on  
    category0_.CategoryID=spareparts1_.CATEGORY_ID  
where  
    category0_.name='Motorteile - 1300cc'
```

Beispiel – Eager Fetching

SQL

```

select
  category0_.CategoryID as CategoryID0_0_,
  ...
  spareparts1_.SparePartID as SparePar1_0__
from
  Categories category0_
left outer join
  SpareParts spareparts1_
  on
  category0_.CategoryID=spareparts1_.CATEGORY_ID
where
  category0_.name='Motorteile - 1300cc'

```

	CategoryID0_0_	SpareP...	name0_0_	descript3_0_0_	PARENT4_0_0_	NAME1_1_	PRICE1_1_	QUANTITY1_1_	CATEGORY5_1_1_	CATEGORY5_0_0_	SparePar1_0_0_
1	2	1	Motorteile - 1300cc	Standard Teile für 1300cc Motor	1	Benzinpumpe	50	3	2	2	1
2	2	2	Motorteile - 1300cc	Standard Teile für 1300cc Motor	1	Solex Vergaser 34PCI	300	1	2	2	2
3	2	3	Motorteile - 1300cc	Standard Teile für 1300cc Motor	1	Zündverteiler Bosch 009	200	5	2	2	3
3	5	3	Motorteile - 1300cc	Standard Teile für 1300cc Motor	1	Zündverteiler Bosch 009	300	2	5	5	3

Beispiel – Eager fetching

NHibernate

HQL

```
IQuery query = session.CreateQuery("
SELECT DISTINCT c
FROM Category c
left join fetch c.SpareParts part
WHERE c.Name =
        'Motorteile - 1300cc');
```

ICriteria

```
ICriteria criteria =
    session.CreateCriteria<Category>()
        .Add(Expression.Like("Name",
            "Motorteile - 1300cc"));
```

```
IList<Category> list =
    criteria.SetResultTransformer(
```

```
CriteriaSpecification.DistinctRootEntity)
        .List<Category>();
```

In NHibernate müssen Duplikate
„manuell“ entfernt werden

Entity Framework

LINQ to Entities

```
ctx.Categories.Include("SpareParts")
    .Where(c => c.Name ==
        "Motorteile - 1300cc");
```

Das Entity Framework entfernt
Duplikate automatisch

Beispiel – Lazy Loading

NHibernate

SQL

Konfiguration angepasst

```
<class name="Entities.Category" >
  <set name="SpareParts"
    table="SPARE_PART"
    cascade="all"
    ➔ lazy = "true" ◀
    inverse="true">
    <key column="CATEGORY_ID"/>
    <one-to-many
      class="Entities.SparePart" />
  </set>
```

QueryOver

```
IQueryOver<Category> queryOver =
  session.QueryOver<Category>();

queryOver.Where(c => c.Name ==
  "Motorteile - 1300cc");

IList<Category> list =
  queryOver.List();
```

SELECT

```
this_.CategoryID as CategoryID0_0_,
this_.name as name0_0_,
this_.description as descript3_0_0_,
this_.PARENT_CATEGORY_ID as PARENT4_0_0_
```

FROM

```
Categories this_
```

WHERE

```
this_.name = @p0; @p0 = 'Motorteile - 1300cc'
```

Beispiel – Lazy Loading

NHibernate

QueryOver

```
IQueryOver<Category> queryOver =  
    session.QueryOver<Category>();  
  
queryOver.Where(c => c.Name ==  
    "Motorteile - 1300cc");  
  
IList<Category> list =  
    queryOver.List();  
  
foreach (Category category in list) {  
  
    Console.WriteLine("Category  
        name = {0} hat {1}  
        Ersatzteile",  
        category.Name,  
        category.SpareParts.Count);  
  
}
```

SQL

```
SELECT  
    this_.CategoryID as CategoryID0_0_,  
    this_.name as name0_0_,  
    this_.description as descript3_0_0_,  
    this_.PARENT_CATEGORY_ID as PARENT4_0_0_  
FROM  
    Categories this_  
WHERE  
    this_.name = @p0; @p0 = 'Motorteile - 1300cc'  
  
SELECT  
    spareparts0_.CATEGORY_ID as CATEGORY5_1_0_,  
    spareparts0_.SparePartID as SparePar1_1_0_,  
    spareparts0_.SparePartID as SparePar1_1_0_0_,  
    spareparts0_.NAME as NAME1_0_0_,  
    spareparts0_.PRICE as PRICE1_0_0_,  
    spareparts0_.QUANTITY as QUANTITY1_0_0_,  
    spareparts0_.CATEGORY_ID as CATEGORY5_1_0_0_  
FROM  
    SpareParts spareparts0_  
WHERE spareparts0_.CATEGORY_ID=@p0; @p0 = 2
```


Beispiel – Lazy Loading

Entity Framework

Entity Query

```
string query = "select value c
from Categories as c where c.name =
                'Motorteile - 1300cc'";

var cats =
    ctx.CreateQuery<Category>(query);

foreach (Category cat in cats) {

    cat.SpareParts.Load();
    Console.WriteLine("Name : {0}
                      #SpareParts {1}",
                      cat.Name,
                      cat.SpareParts.Count);

}
```

SQL

```
SQL SELECT
[Extent1].[CategoryID] AS [CategoryID],
[Extent1].[name] AS [name],
[Extent1].[description] AS [description],
[Extent1].[PARENT_CATEGORY_ID] AS
[PARENT_CATEGORY_ID]
FROM [dbo].[Categories] AS [Extent1]
WHERE [Extent1].[name] =
                'Motorteile - 1300cc'
```

Agenda

- ▶ (Architektur) Überblick
- ▶ Entitäten beschreiben
- ▶ CRUD
- ▶ Beziehungen
- ▶ Zusammenfassung



Zusammenfassung

- ▶ Entity Framework und NHibernate sind im Kern ähnlich
- ▶ Wesentliche Unterschiede
 - ▶ Handhabung von Beziehungen
 - ▶ Datenbank Support
 - ▶ CASCADE Funktionalität
 - ▶ *Unterstützung von verschiedenen Collection Typen*
 - ▶ *Logging*
- ▶ ABER:
Viele Aspekte wurden nicht beleuchtet: z. B. Einbetten von Funktionen, Vererbung, Performance, T4, Self-tracking, ...

5.– 8. September 2011
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Thomas Haug
MATHEMA Software GmbH