12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Nachschlag

Entity Framework 4.0

Thomas Haug

MATHEMA Software GmbH

In eigener Sache

Herbstcampus

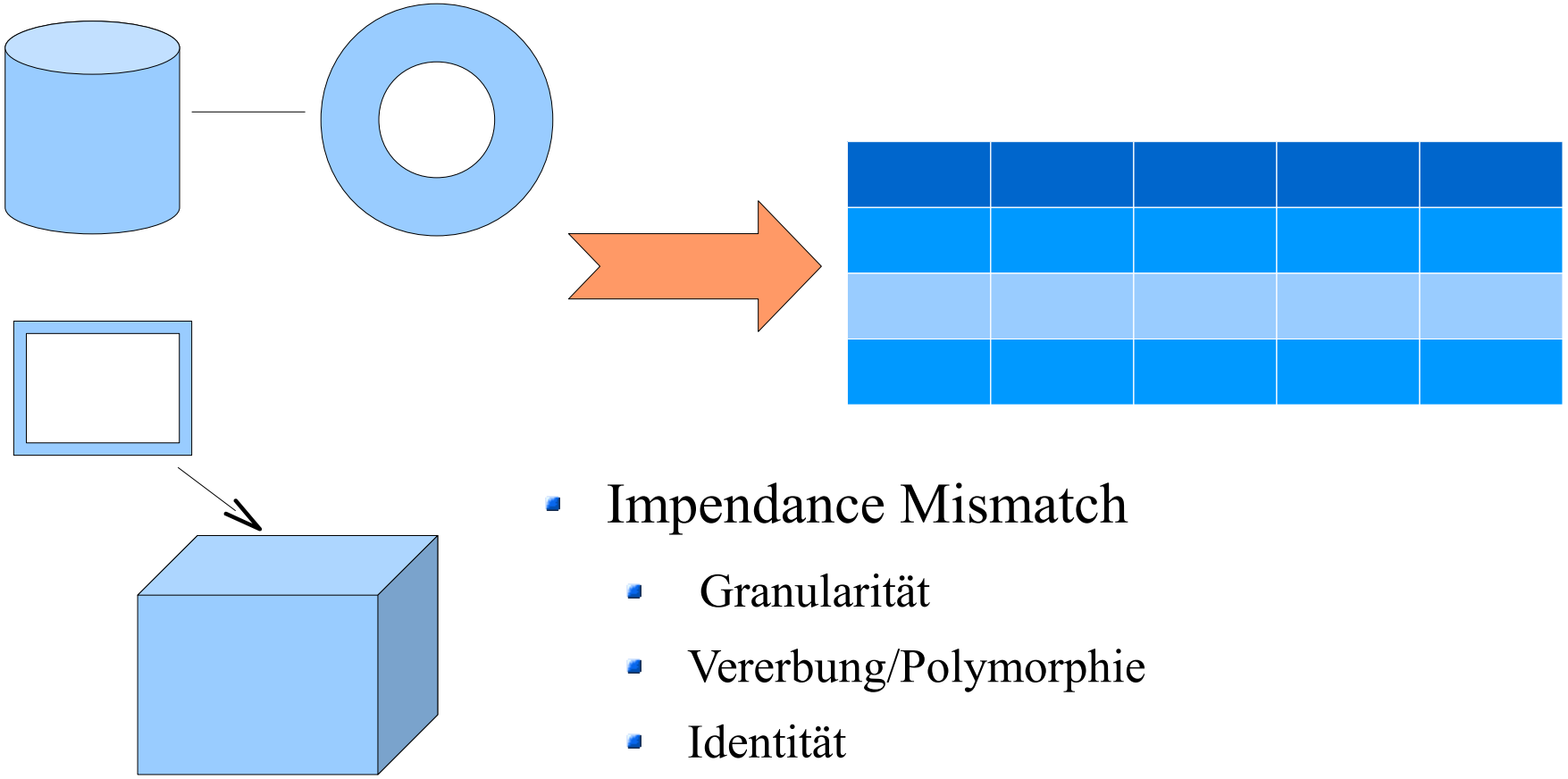
- Dipl.-Inf (Univ.)
- Senior Consultant, Architekt und Trainer
- Programmiererfahrung > 25 Jahre
- 12+ Jahre Java Enterprise Entwicklung
- 7+ Jahre .Net Entwicklung (C#)
- Schwerpunkte
  - Software-Architektur
  - Verteilte (heterogene) Systeme
  - Objekt-Relationales Mapping

- E-Mail: thomas.haug@mathema.de

Agenda

- Entity Framework 1.0
  - Überblick
  - Architektur
  - CRUD
  - Abfrage Möglichkeiten
  - Warum Update notwendig?
- *Model First*
- *Self-Tracking-Entities*
- *Code-First*
- Fazit

- Objekt Relationales Mapping



- Impendance Mismatch
  - Granularität
  - Vererbung/Polymorphie
  - Identität
  - Beziehungen

Überblick

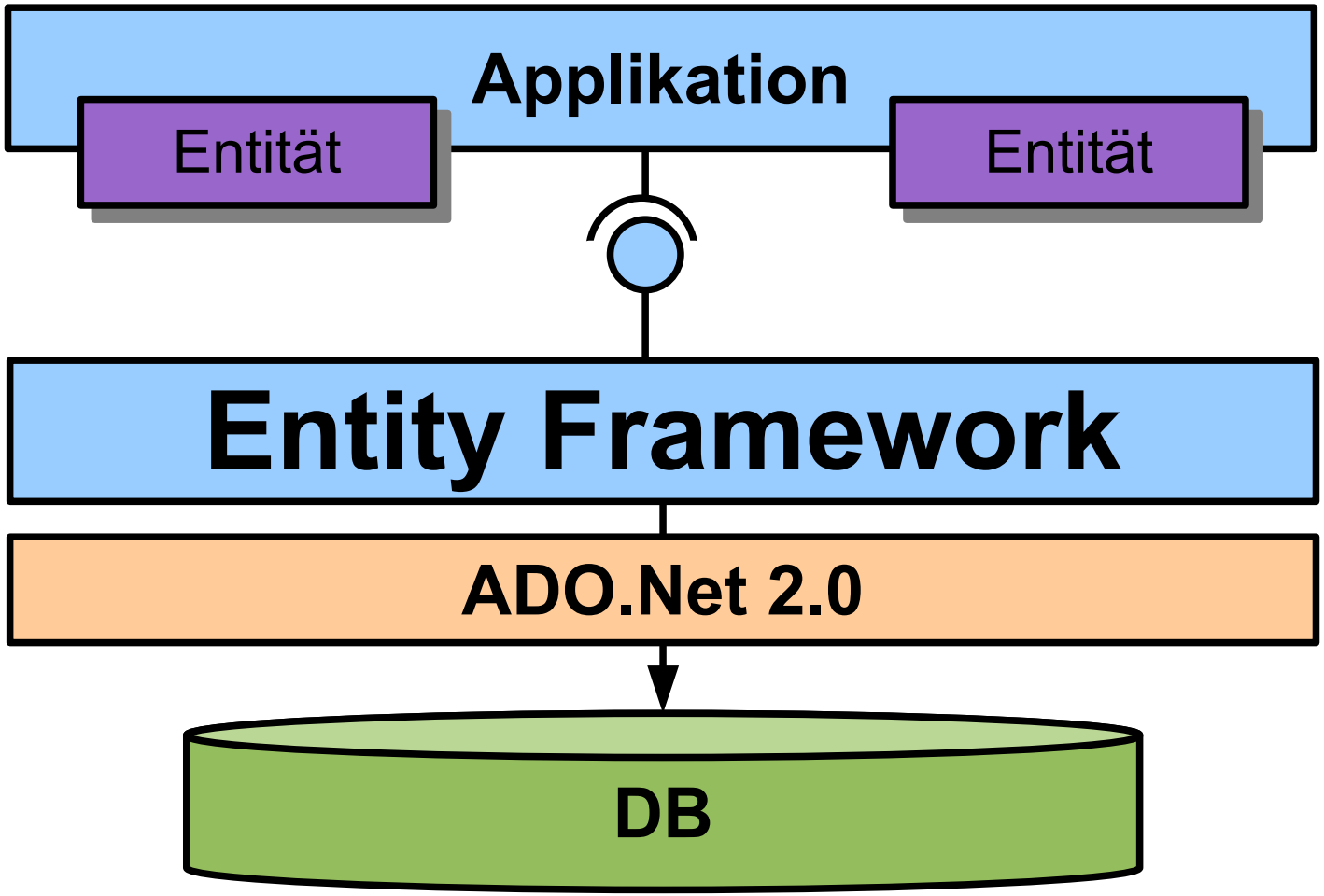
- Offizielle Version Entity Framework 1.0
  - Bestandteil .Net 3.5 SP1
- Entity Framework 4.0 Bestandteil des .Net 4.0 Release
- Community Technology Preview (CTP) 4 für VS2010
  - ermöglicht Code First
  - ermöglicht Annotations-basierte Persistenz

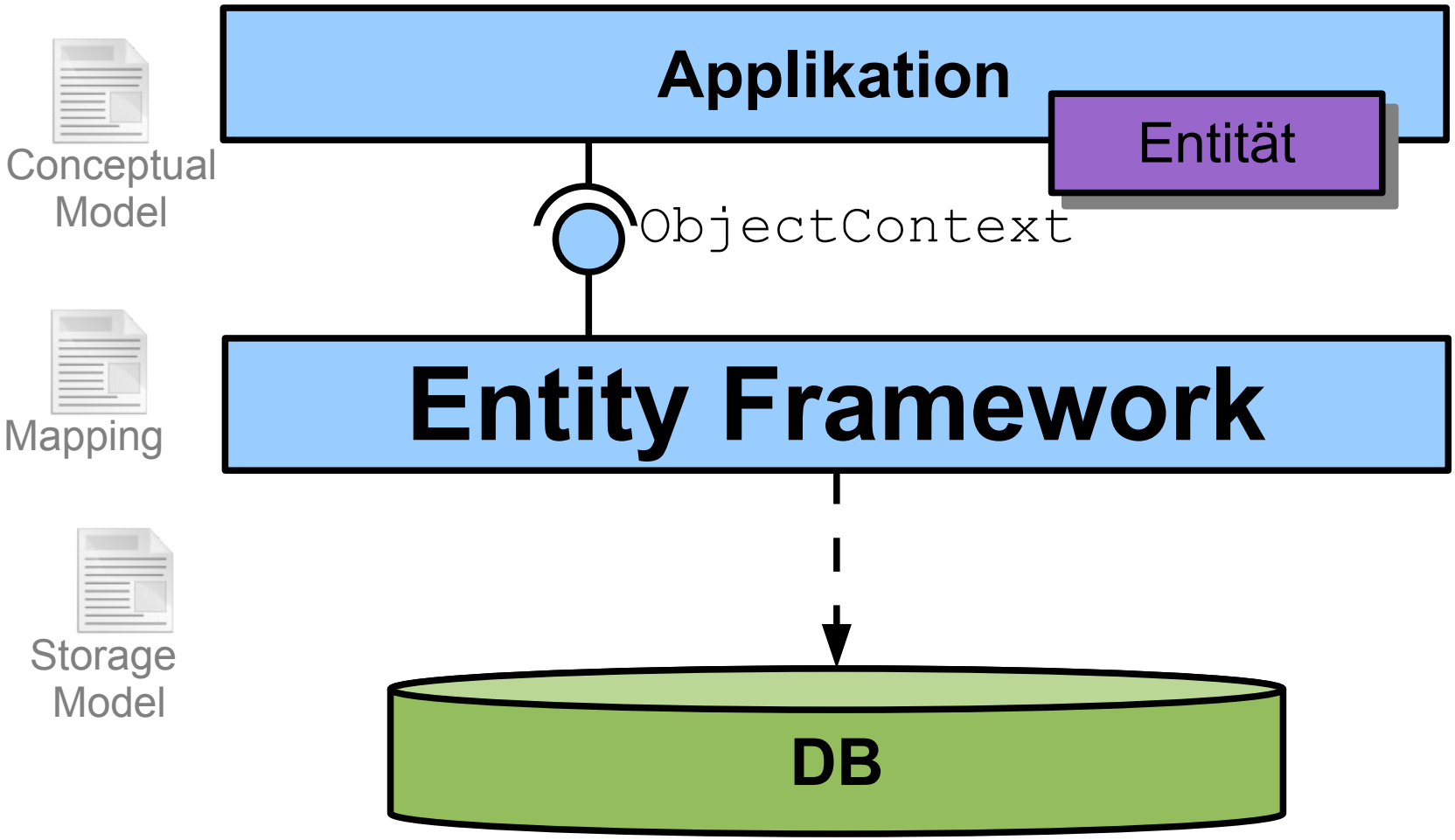
Überblick

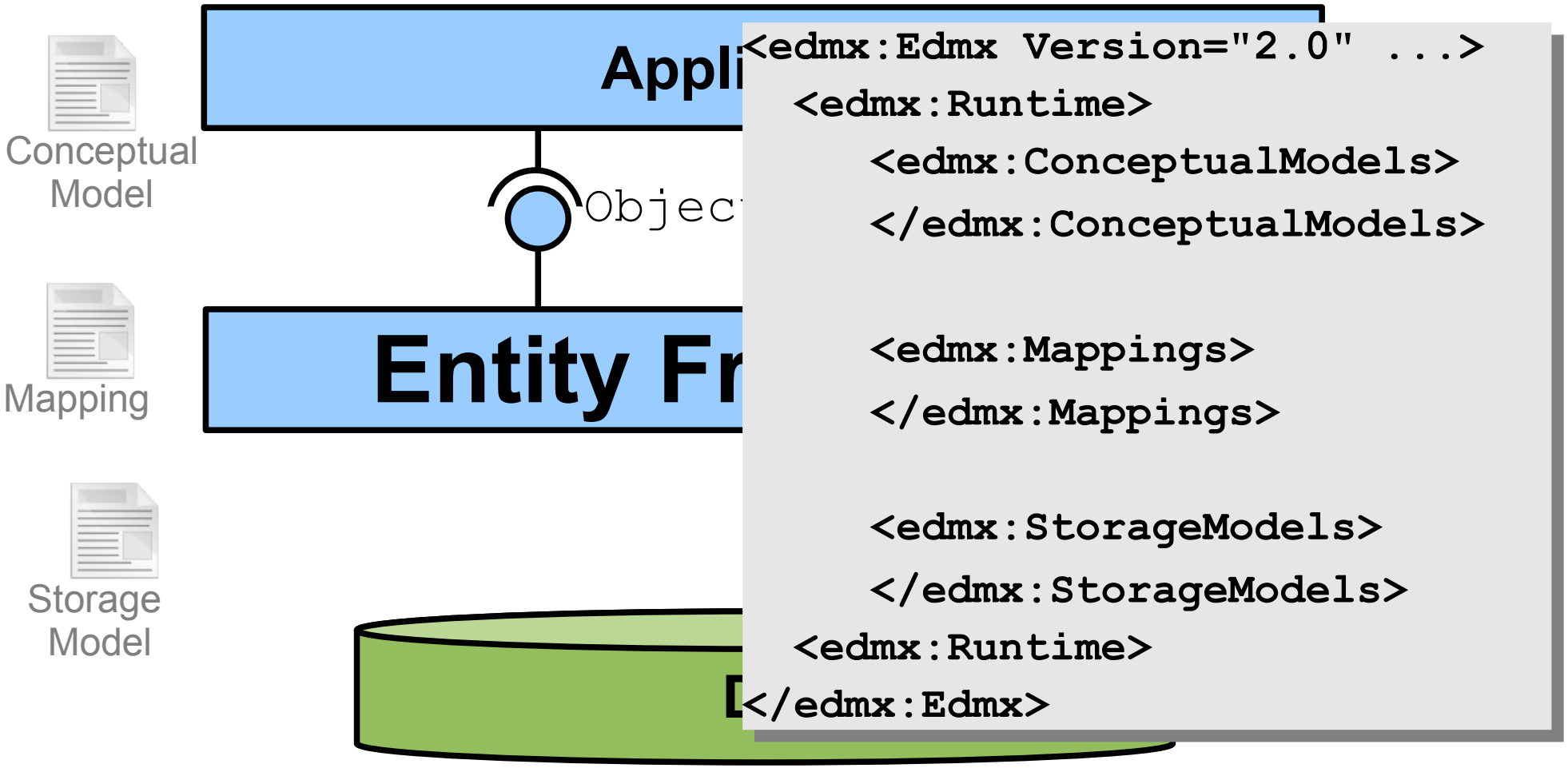
- Dient dem Speichern von Objekten in relationalen Datenbanken
- Kapselt ADO.Net und SQL vor Entwickler
- (Transparente) Persistenz für (POCO) Klassen
- Unterstützung von Vererbungshierarchien und polymorphen Abfragen
- Transitive Persistenz für Objektgraphen
- Unterstützung für
  - „Lazy Loading“ (implizit erst ab Version 4.0)
  - „Eager Fetching“

Überblick

- Transaktionales ‚Write-Behind'
- Optimistische Locking
- Unterstützung von abgekoppelten Objekten
- Unterstützt SQL Server „out-off-the-box"
- SQL Tracing ist Bestandteil, aber nur für Abfragen
  - Abhilfe schafft EFProviderWrapper
    - Caching provider (Velocity)
    - EFTracing Wrapper
    - Microsoft Public License (Ms-PL)


Applikation
Entität
Entität
Entity Framework
ADO.Net 2.0
DB


Applikation
Entität
ObjectContext
Entity Framework
DB
Conceptual Model
Mapping
Storage Model


Conceptual Model
Mapping
Storage Model
Appli
Objec
Entity Fr
<edmx:Edmx Version="2.0" ...>
<edmx:Runtime>
<edmx:ConceptualModels>
</edmx:ConceptualModels>
<edmx:Mappings>
</edmx:Mappings>
<edmx:StorageModels>
</edmx:StorageModels>
<edmx:Runtime>
</edmx:Edmx>

Conceptual Model

Mapping

Storage Model

```
<edmx:Edmx Version="2.0" ...>
  <edmx:Runtime>
    <edmx:ConceptualModels>
      <Schema Namespace="EFvsNHModel"..>
        <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" />
        </EntityType>
      </Schema>
    </edmx:ConceptualModels>
</edmx:Edmx>
```

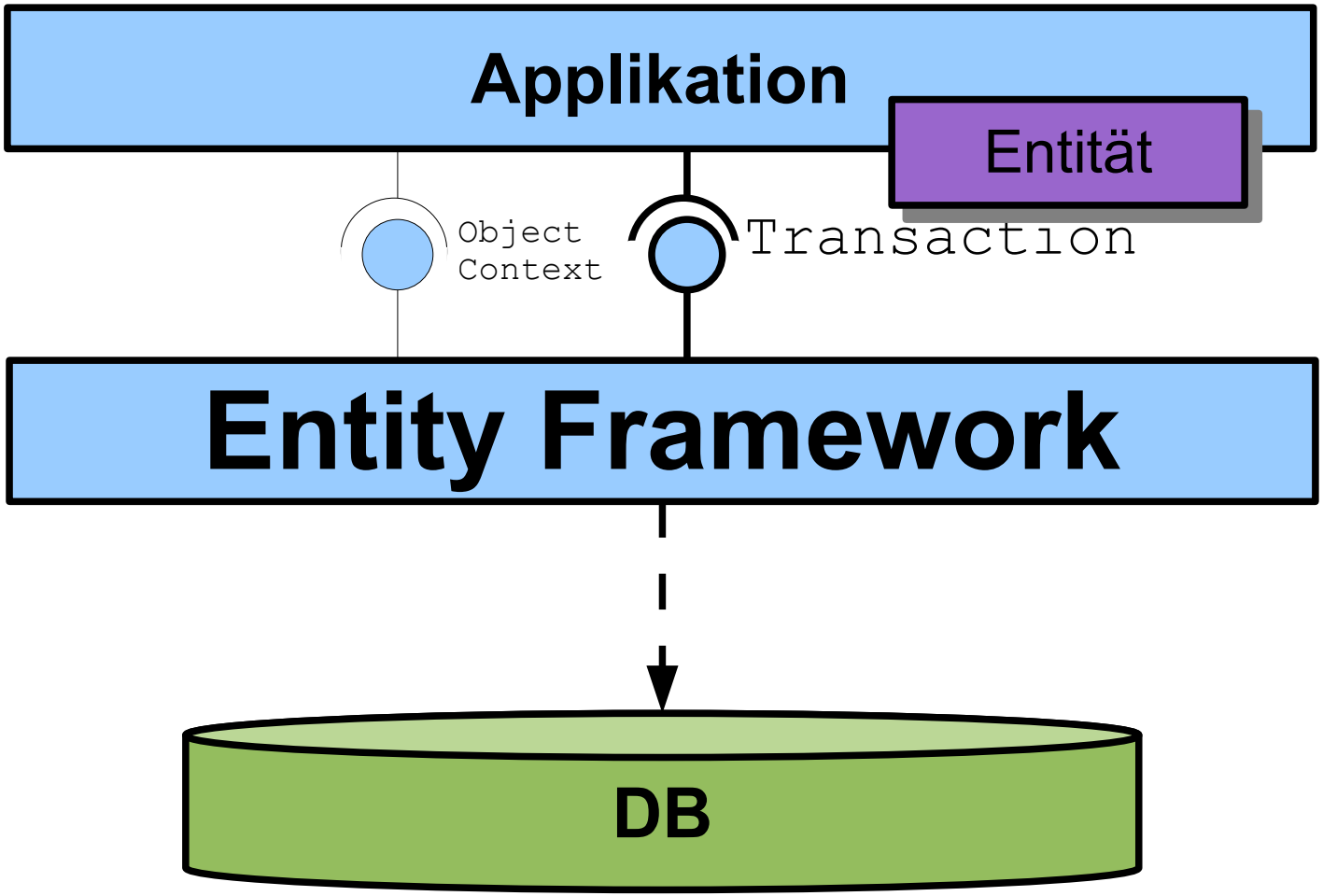

Conceptual Model

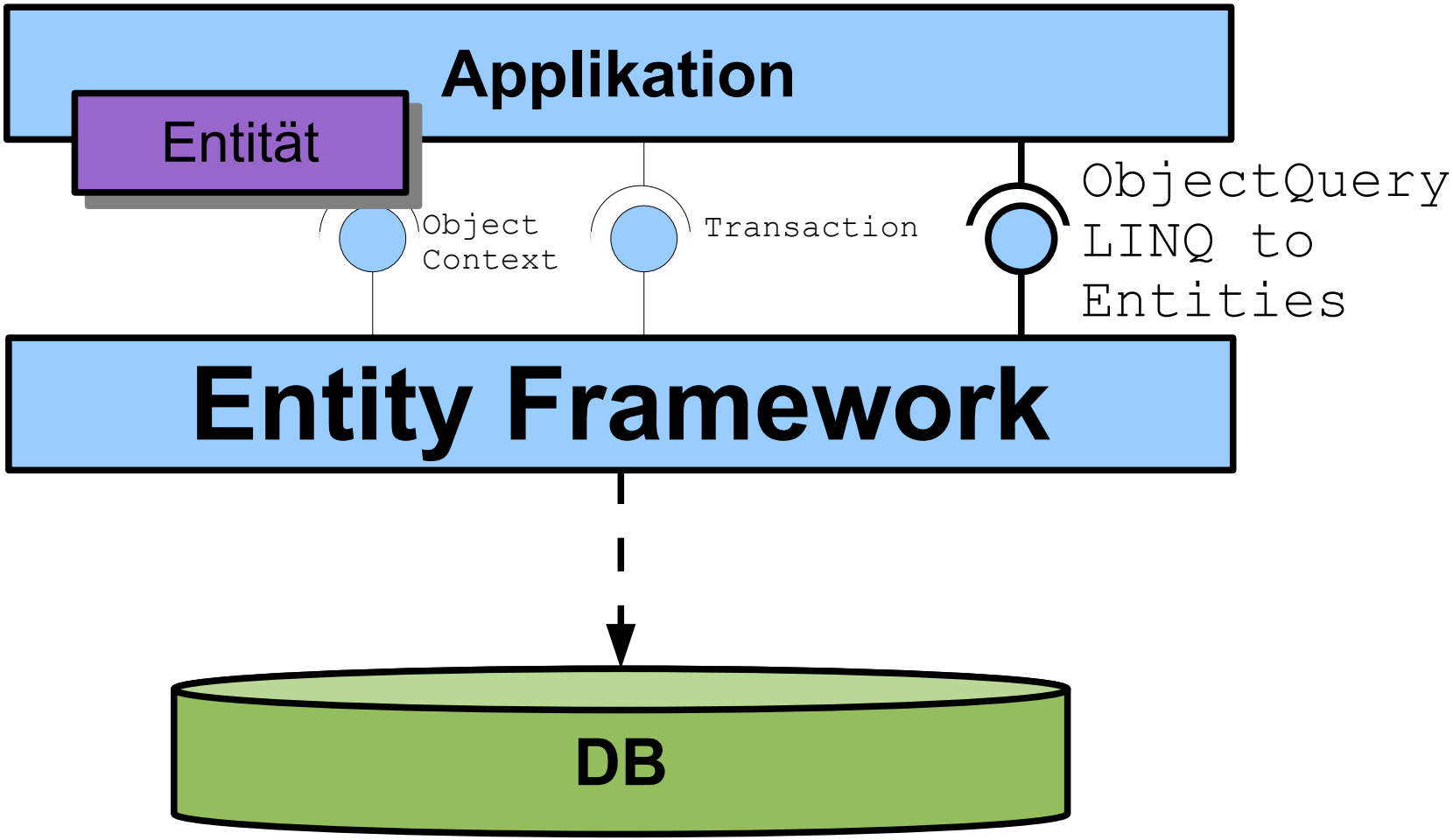

Mapping


Storage Model

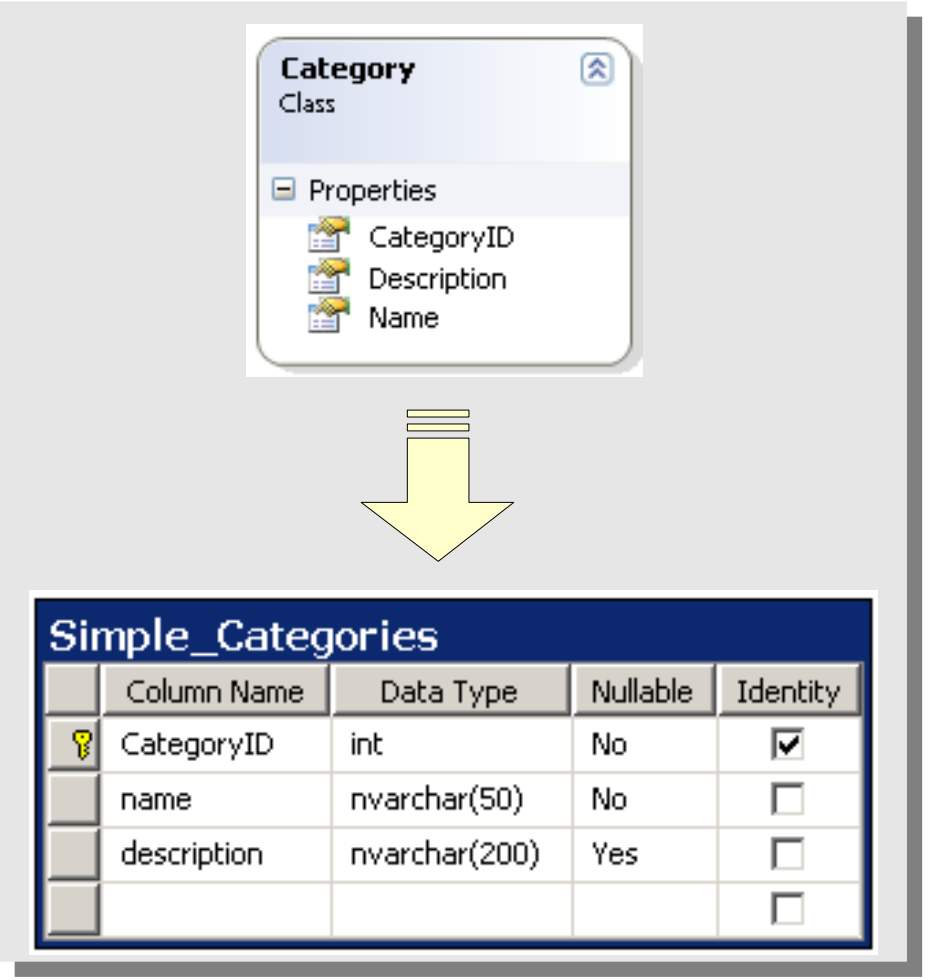
```
<edmx:Edmx Version="2.0" ...>
  <edmx:Runtime>
    <edmx:StorageModels>
      <EntityContainer Name="EFStoreContainer">
        <EntitySet Name="Simple_Categories"
    EntityType="EFvsNHModel.Store.Simple_Categories"
     store:Type="Tables" Schema="dbo" />
      </EntityContainer>
      <EntityType 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>
</edmx:Edmx>
```

```
<edmx:Edmx Version="2.0" ...>
  <edmx:Runtime>
     <edmx:Mappings>
        <Mapping Space="C-S"
xmlns="http://schemas.microsoft.com...">
        <EntityContainerMapping
           StorageEntityContainer="EFStoreContainer"
  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>
</edmx:Edmx>
```


Applikation
Entität
Object Context
Transaction
Entity Framework
DB


Applikation
Entität
Object Context
Transaction
ObjectQuery
LINQ to Entities
Entity Framework
DB

- **C)reate**
  Erzeugen und Persistieren einer Category Instanz
- **(R)ead**
  Lesen eines Category Objekts aus der Datenbank
- **(U)pdate**
  Modifizieren des Category Objekts in der Datenbank
- **(D)elete**
  Löschen der Category Instanz aus der Datenbank



- **Create**
  Erzeugen und Persistieren einer `Category`-Instanz

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

   // EF 1.0 Variante, deprecated
   ctx.AddToCategories(myCat);

   ctx.Categories.AddObject(myCat);

   ctx.SaveChanges();
}
```

- **Read**
  Lesen eines `Category`-Objekts aus der Datenbank

```
using (EFvsNHEntities ctx =
        new EfvsNHEntities()) {
   EntityKey key = new EntityKey(
                              "EFvsNH.Categories",
                              "CategoryID",
                              myCat.CategoryID);

   // Variante 1
   Category gefCat = (Category)
                        ctx.GetObjectByKey(key);

   // Variante 2
   bool gefunden = ctx.TryGetObjectByKey(key,
                              out gefundeneCategory);
}
```

- **Update**
  Modifizieren eines `Category`-Objekts in der Datenbank

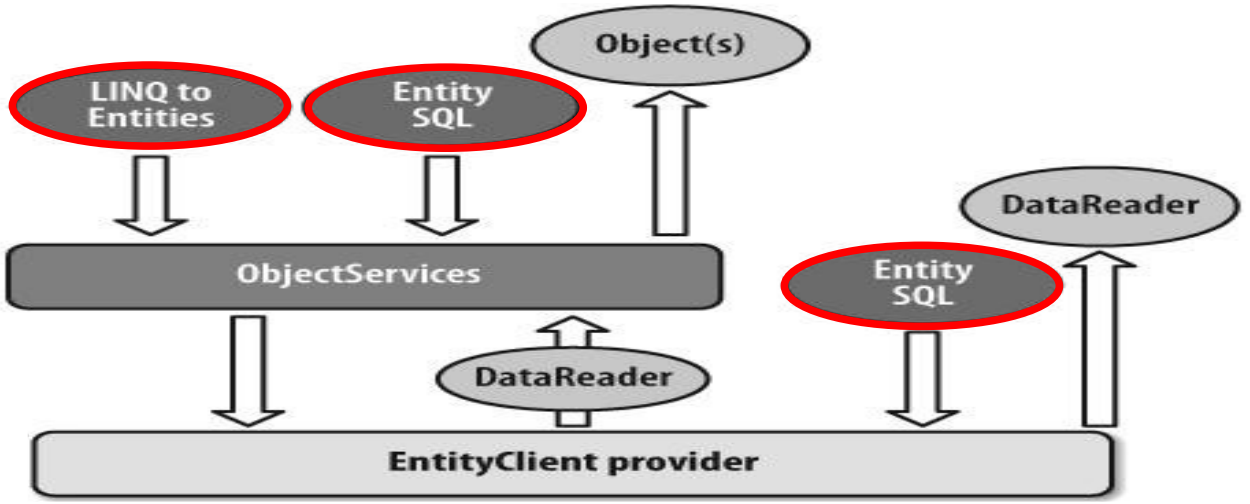
```
using (EFvsNHEntities ctx = new EfvsNHEntities()) {

   // bereits existierende, aber
   // nicht 'attached' Entität
   ctx.Attach(myCat);
   myCat.Name = "neuer Name";

   ctx.SaveChanges();
}
```

- **Delete**
  Löschen der `Category`-Instanz aus der Datenbank

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

   //Object muss im Context geladen sein!
   ctx.DeleteObject(myCat);

   ctx.SaveChanges();
}
```

Entity Abfragen

Herbstcampus



- LINQ to Entities
- Entity SQL mittels Object Services
- Entity SQL mittels Entity Client

Entity Abfragen


```
var cat =
  from categ in
    ctx.Categories
  where
    categ.Name.StartsWith("A")
  orderby categ.Name ascending
  select categ;


  // Projection
  select new {
     name = categ.Name,
     desc = categ.Description
  };
```

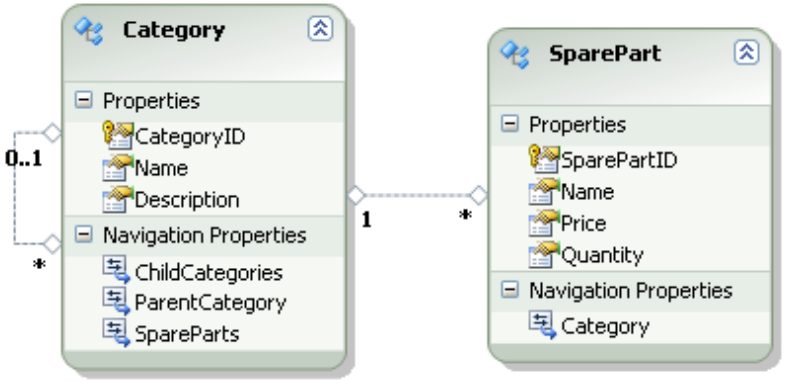
Entity Abfragen

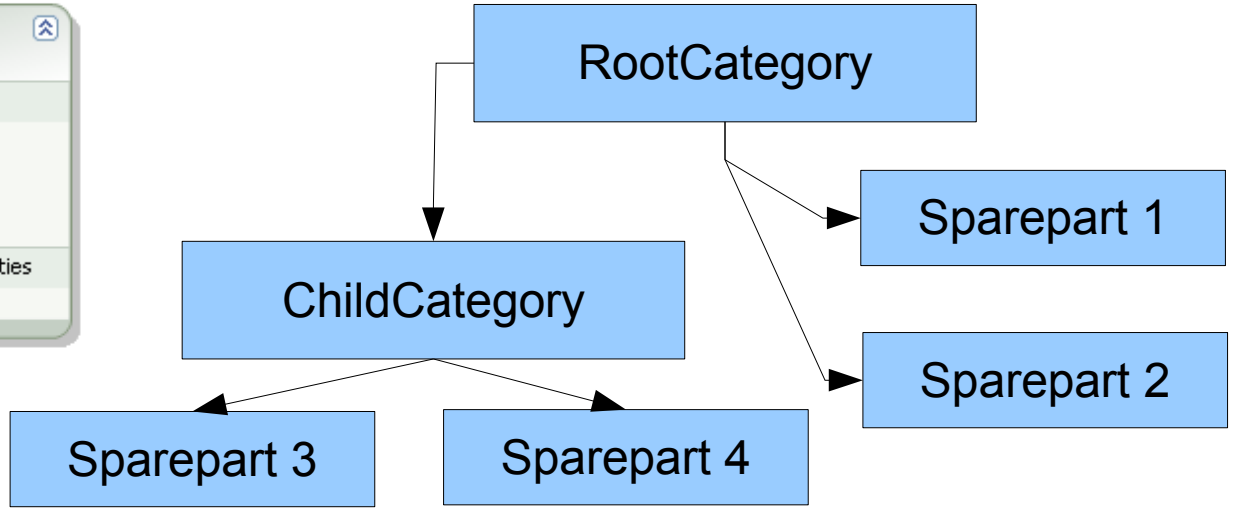

```
var qStr =
 @"SELECT VALUE c
   FROM Categories
   AS c WHERE c.Name='Motor'";

var entities = ctx.
  CreateQuery<Category>(qStr);
```

Entity Abfragen


```
using (var con = new
  EntityConnection("name=EF4")) {

  con.Open();
  var qStr =
    "SELECT VALUE c
     FROM Categories AS c ";
  var cmd = conn.CreateCommand();
  cmd.CommandText = qStr;
  using (var rdr = cmd.ExecuteReader(
    CommandBehavior.SequentialAccess)) {
    while (rdr.Read()) {
      Console.WriteLine(rdr.GetString(1));
    }
  }
}
```

Entity Abfragen – Eager Fetching

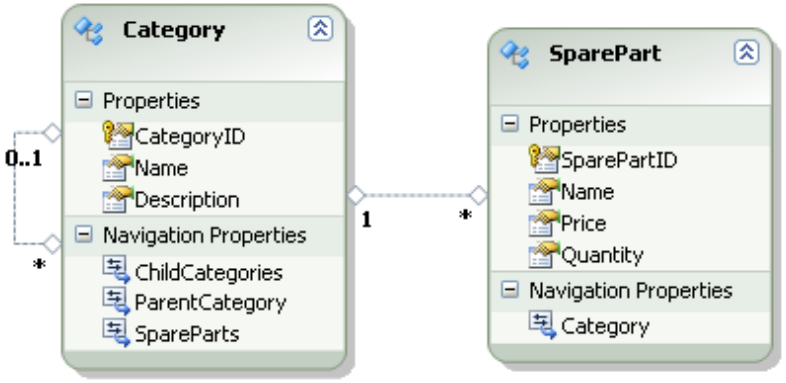


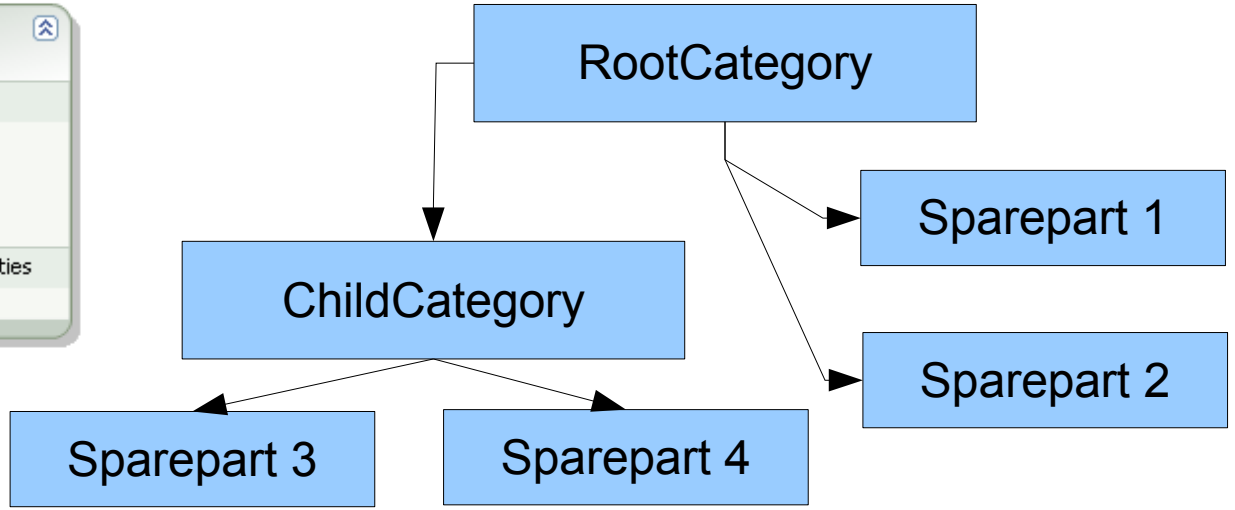


- Fetching

```
Category tempRoot =
      (from cat1 in
          context.Categories)

       where cat1.CategoryID ==
                 rootCategory.CategoryID
       select cat1)
      .Single<Category>();
```

Entity Abfragen – Eager Fetching





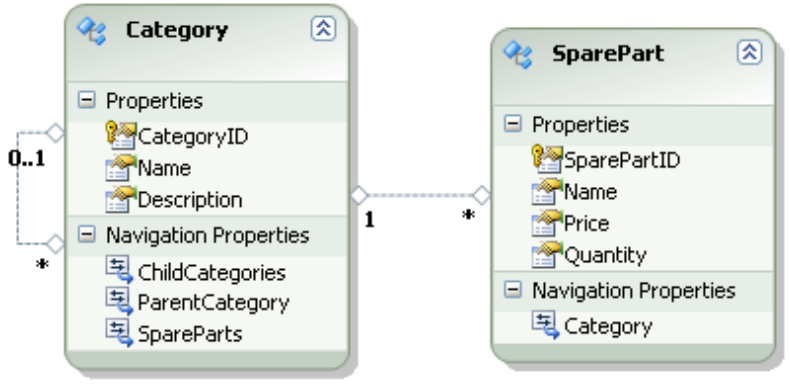
- Eager fetching

```
Category tempRoot =
      (from cat1 in
          context.Categories
          .Include("ChildCategories")
          .Include("SpareParts")

       where cat1.CategoryID ==
                 rootCategory.CategoryID
       select cat1)
      .Single<Category>();
```

Warum Update notwendig?

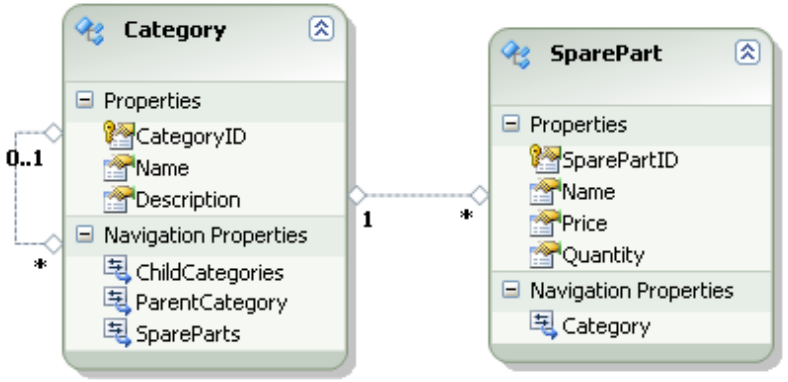
- Implizites Lazy Loading wurde nicht unterstützt
- Der EDM Designer ist fehlerhaft und unvollständig
- Model First wird nicht unterstützt
- Keine Unterstützung für verteilte Anwendungen (N-Tier)
- Keine vollständige Unterstützung für Plain Old CLR Objects (POCO)

- Explizites Lazy Loading

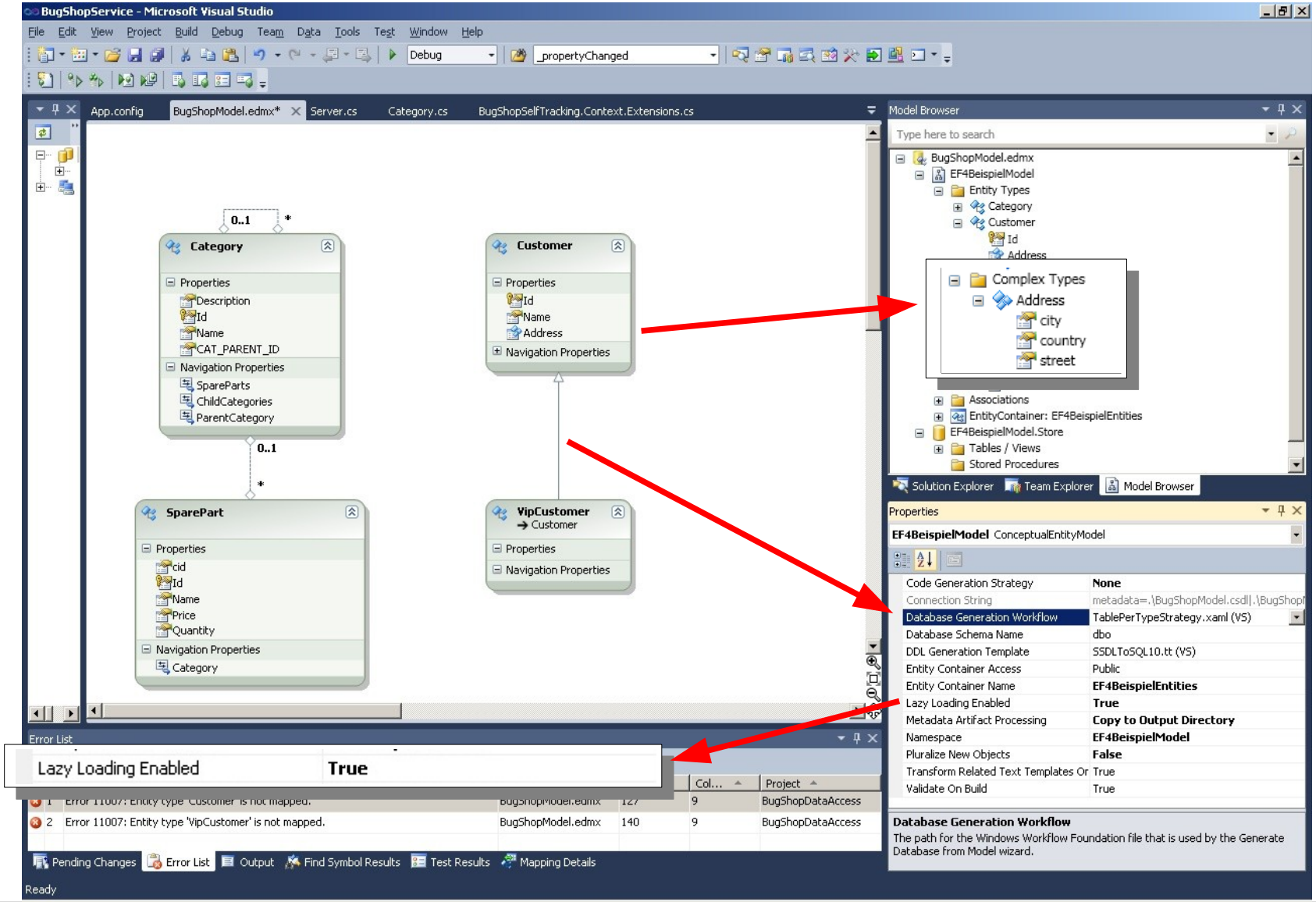


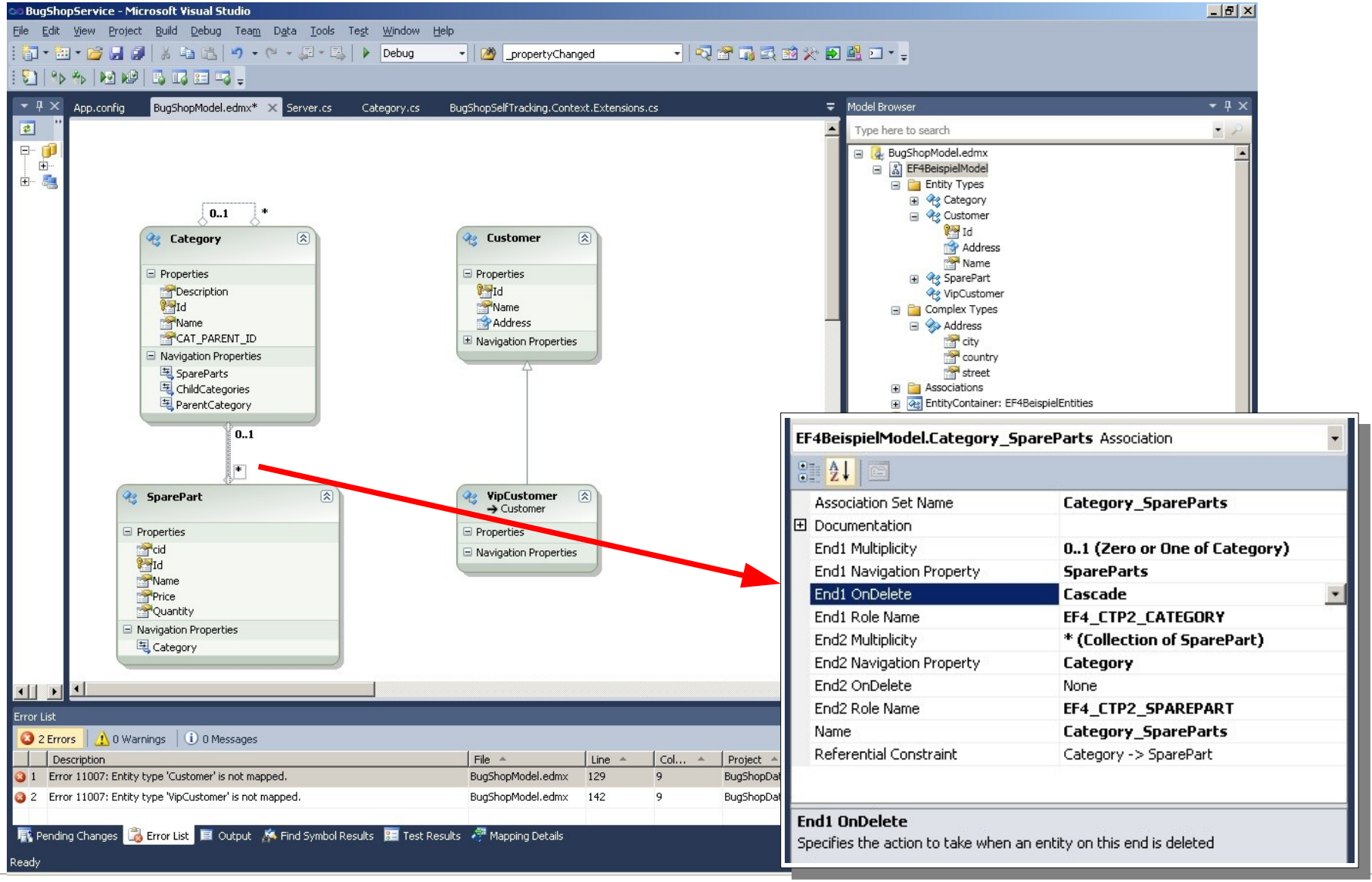

```
using (EFvsNHEntities ctx =
       new EFvsNHEntities()) {

  Category tempRoot= (Category)
   ctx.GetObjectByKey(rootCategory.EntityKey);

tempRoot.ChildCategories.Load();

 foreach (var category
          in tempRoot.ChildCategories) {
   category.SpareParts.Load();

   Console.WriteLine("Ersatzteile {0}",
            category.SpareParts.Count());
 }
}
```

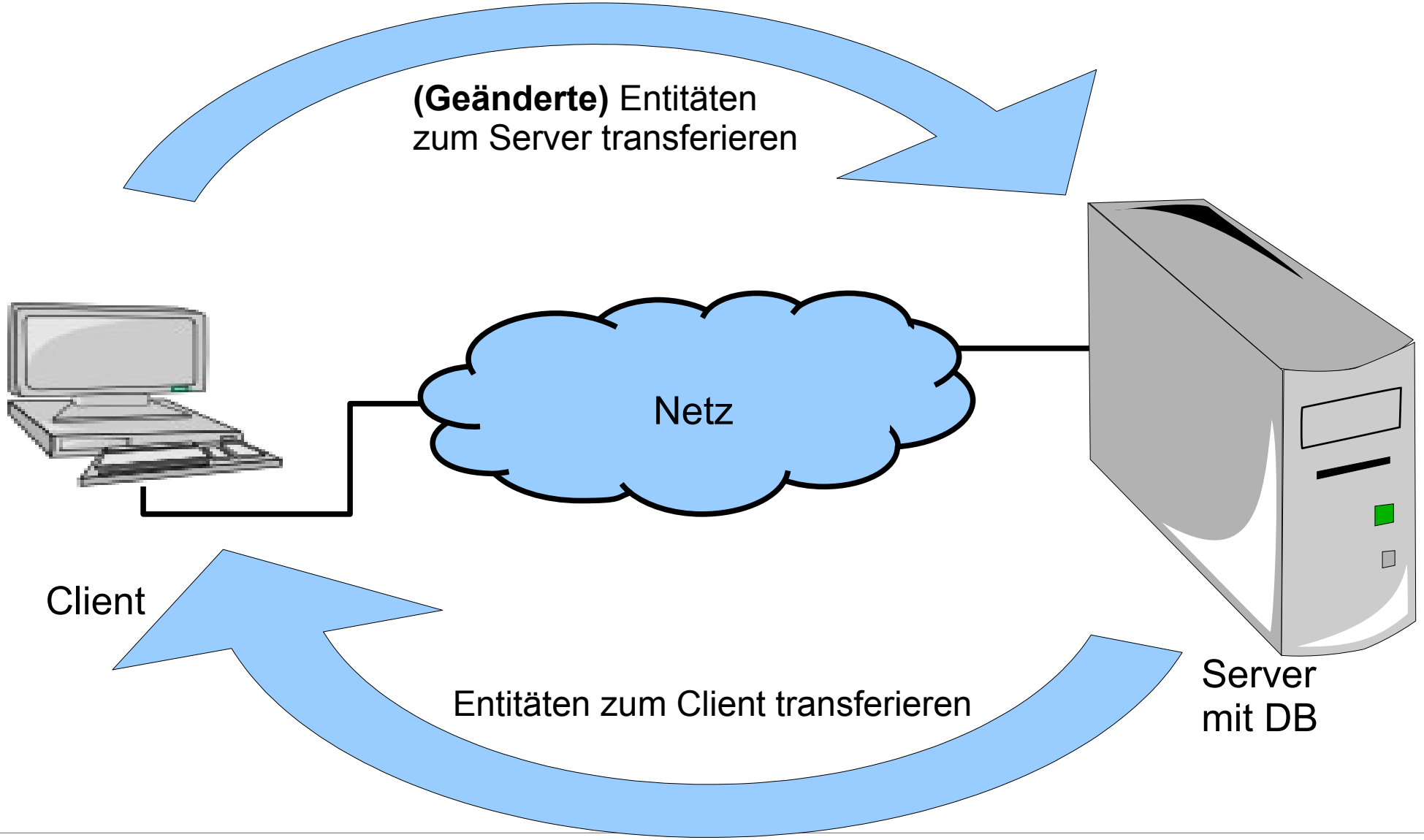
- Implizites Lazy Loading



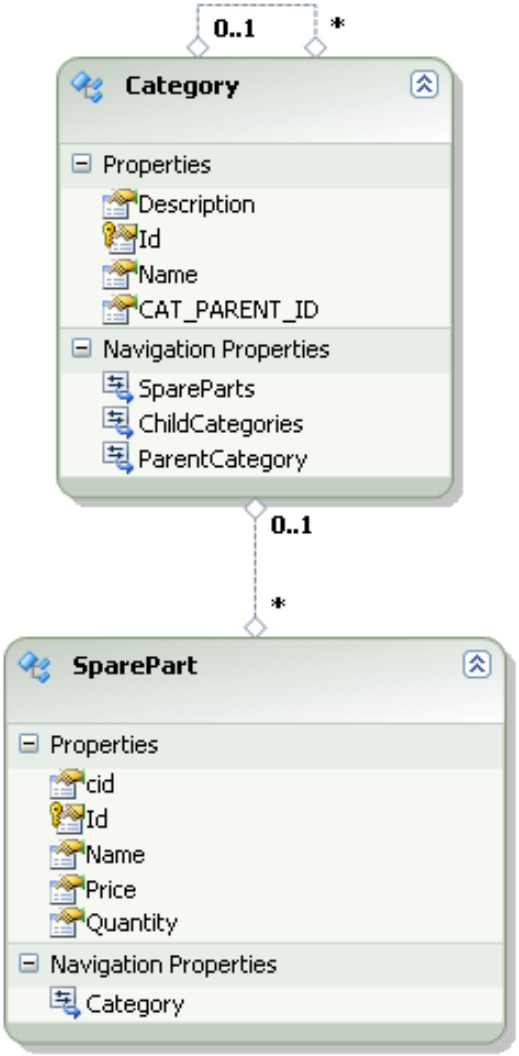

```
using (EFvsNHEntities ctx =
       new EFvsNHEntities()) {

  ctx.ContextOptions
     .LazyLoadingEnabled = true;

  Category tempRoot= (Category)
   ctx.GetObjectByKey(rootCategory.EntityKey);

 foreach (var category
          in tempRoot.ChildCategories) {
   Console.WriteLine("Ersatzteile {0}",
            category.SpareParts.Count());
 }
}
```


BugShopService - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
_propertyChanged
App.config
BugShopModel.edmx*
Server.cs
Category.cs
BugShopSelfTracking.Context.Extensions.cs
0..1
*
Category
Properties
Description
Id
Name
CAT_PARENT_ID
Navigation Properties
SpareParts
ChildCategories
ParentCategory
Customer
Properties
Id
Name
Address
Navigation Properties
SparePart
Properties
cid
Id
Name
Price
Quantity
Navigation Properties
Category
VipCustomer
Customer
Properties
Navigation Properties
Model Browser
Type here to search
BugShopModel.edmx
EF4BeispielModel
Entity Types
Category
Customer
Id
Address
Complex Types
Address
city
country
street
Associations
EntityContainer: EF4BeispielEntities
EF4BeispielModel.Store
Tables / Views
Stored Procedures
Solution Explorer
Team Explorer
Model Browser
Properties
EF4BeispielModel ConceptualEntityModel
Code Generation Strategy None
Connection String metadata=.\BugShopModel.csdl|.\BugShop
Database Generation Workflow TablePerTypeStrategy.xaml (VS)
Database Schema Name dbo
DDL Generation Template SSDLToSQL10.tt (VS)
Entity Container Access Public
Entity Container Name EF4BeispielEntities
Lazy Loading Enabled True
Metadata Artifact Processing Copy to Output Directory
Namespace EF4BeispielModel
Pluralize New Objects False
Transform Related Text Templates Or True
Validate On Build True
Database Generation Workflow
The path for the Windows Workflow Foundation file that is used by the Generate Database from Model wizard.
Lazy Loading Enabled True
Error List
Col...
Project
2 Error 11007: Entity type 'VipCustomer' is not mapped. BugShopModel.edmx 140 9 BugShopDataAccess
Pending Changes
Error List
Output
Find Symbol Results
Test Results
Mapping Details
Ready


BugShopService - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
_propertyChanged
App.config
BugShopModel.edmx*
Server.cs
Category.cs
BugShopSelfTracking.Context.Extensions.cs
Model Browser
Type here to search
BugShopModel.edmx
EF4BeispielModel
Entity Types
Category
Customer
Id
Address
Name
SparePart
VipCustomer
Complex Types
Address
city
country
street
Associations
EntityContainer: EF4BeispielEntities
0..1
*
Category
Properties
Description
Id
Name
CAT_PARENT_ID
Navigation Properties
SpareParts
ChildCategories
ParentCategory
Customer
Properties
Id
Name
Address
Navigation Properties
SparePart
Properties
cid
Id
Name
Price
Quantity
Navigation Properties
Category
VipCustomer
Customer
Properties
Navigation Properties
EF4BeispielModel.Category_SpareParts Association
Association Set Name
Category_SpareParts
Documentation
End1 Multiplicity
0..1 (Zero or One of Category)
End1 Navigation Property
SpareParts
End1 OnDelete
Cascade
End1 Role Name
EF4_CTP2_CATEGORY
End2 Multiplicity
* (Collection of SparePart)
End2 Navigation Property
Category
End2 OnDelete
None
End2 Role Name
EF4_CTP2_SPAREPART
Name
Category_SpareParts
Referential Constraint
Category -> SparePart
End1 OnDelete
Specifies the action to take when an entity on this end is deleted
Error List
2 Errors
0 Warnings
0 Messages
Description
File
Line
Col...
Project
1 Error 11007: Entity type 'Customer' is not mapped.
BugShopModel.edmx
129
9
2 Error 11007: Entity type 'VipCustomer' is not mapped.
BugShopModel.edmx
142
9
Pending Changes
Error List
Output
Find Symbol Results
Test Results
Mapping Details
Ready

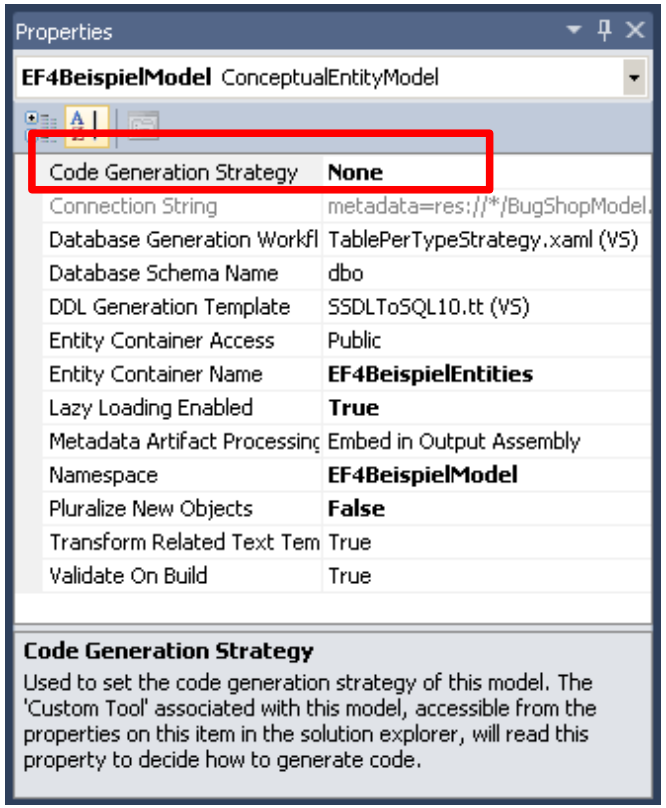

(Geänderte) Entitäten
zum Server transferieren
Netz
Client
Server
mit DB
Entitäten zum Client transferieren

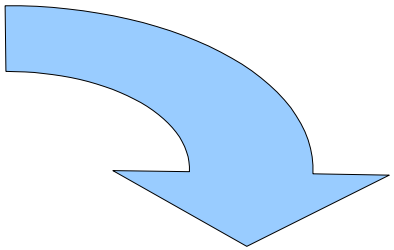
- Veränderungen an abgekoppelten (detached) Entitäten protokollieren
- Anstelle des *Standard Code Generators* wird der *Self-Tracking-Entity Generator* genutzt
- Generiert zwei T4 Templates

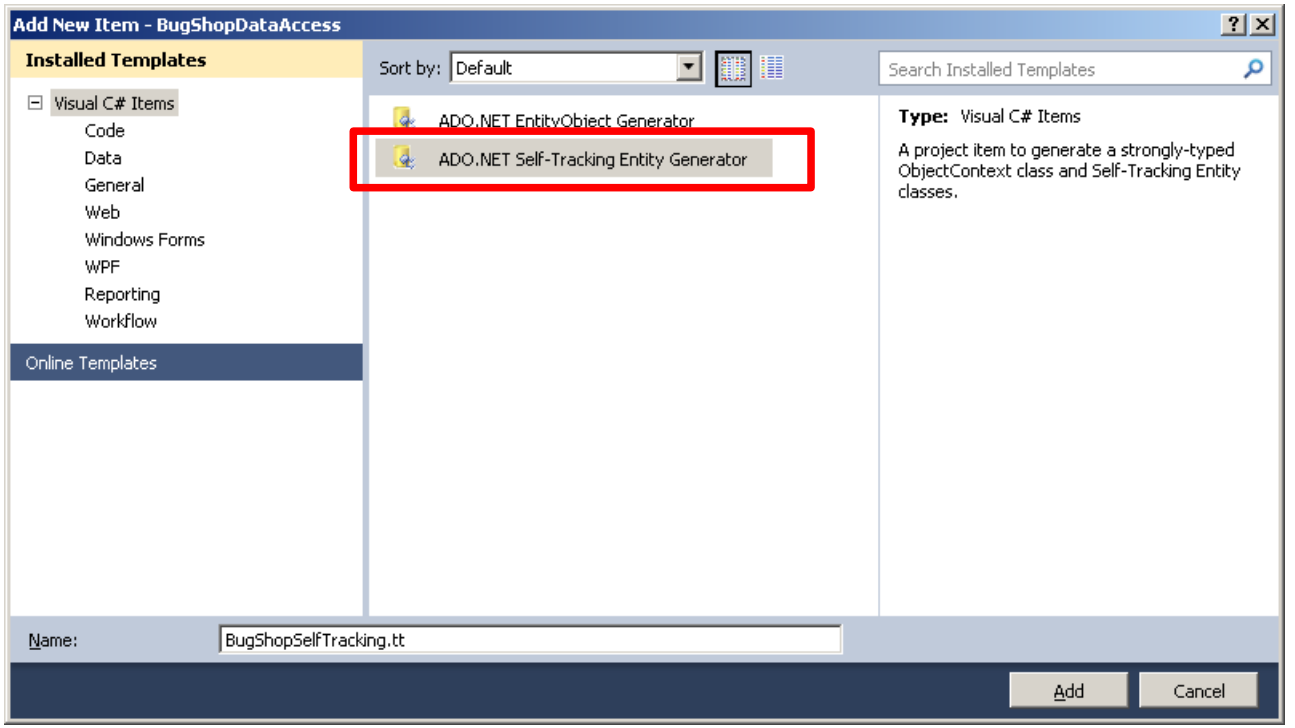


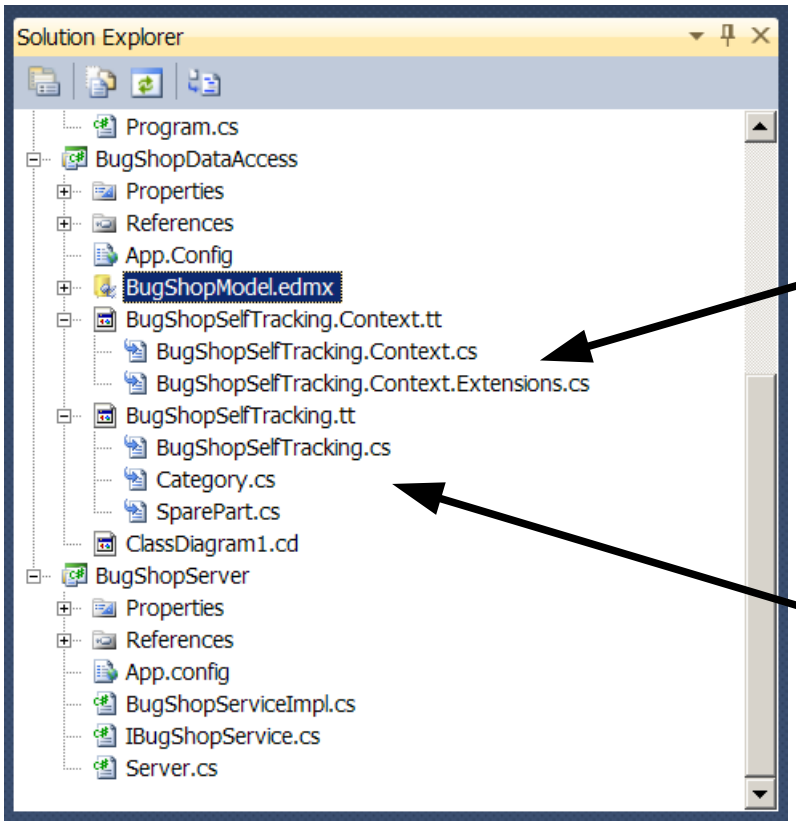
Self-Tracking

Herbstcampus









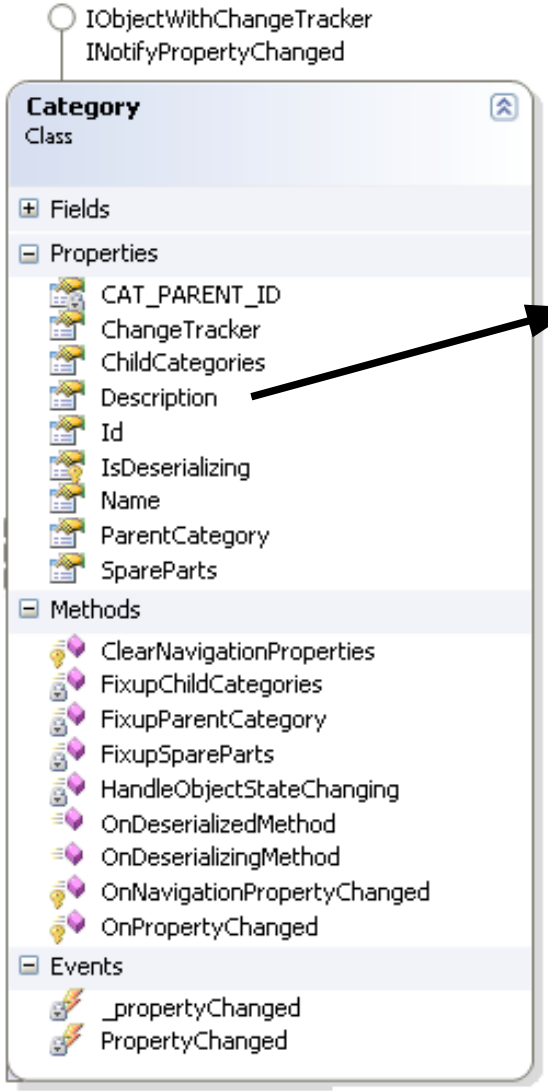
Generierte Context Klassen

- Standard ObjectContext Klasse
- Erweiterungen für Self-Tracking

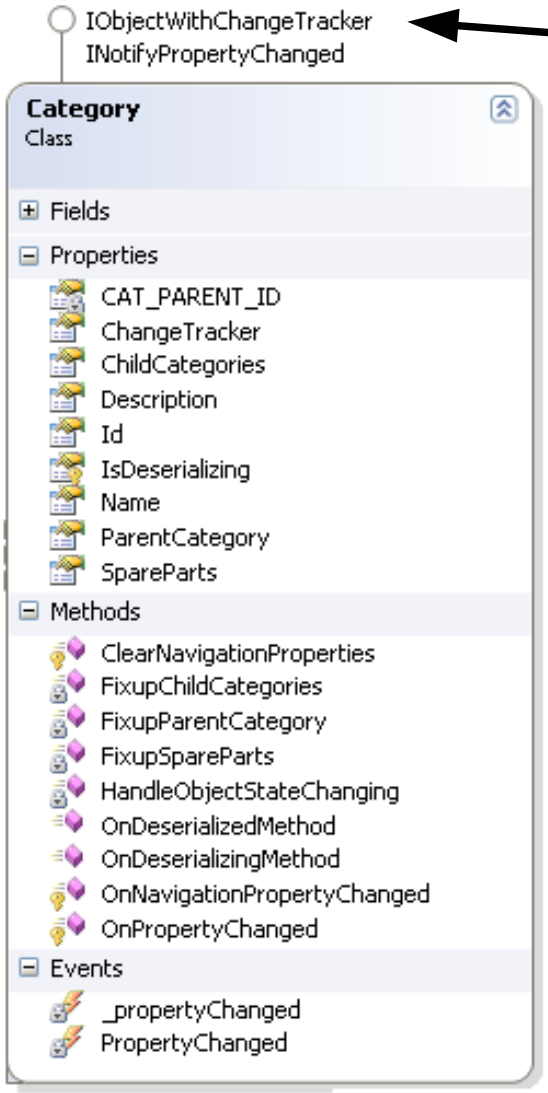
Generierte Entitäts Klassen

- Entitäten
- Erweiterungen für Self-Tracking innerhalb der Entitäten

Self-Tracking - Details




```
[DataMember]
public string Description
{
   get { return _description; }
   set
   {
      if (_description != value)
      {
         _description = value;
         OnPropertyChanged("Description");
      }
   }
}
private string _description;
```



- Schnittstelle findet sich in `BugShopSelfTracking.cs`

```
protected virtual void
    OnPropertyChanged(String propertyName) {
    if (ChangeTracker.State !=
        ObjectState.Added &&
        ChangeTracker.State !=
        ObjectState.Deleted) {
          ChangeTracker.State =
              ObjectState.Modified;
    }
    if (_propertyChanged != null) {
      _propertyChanged(this,
          new
    PropertyChangedEventArgs(propertyName));
    }
}
```

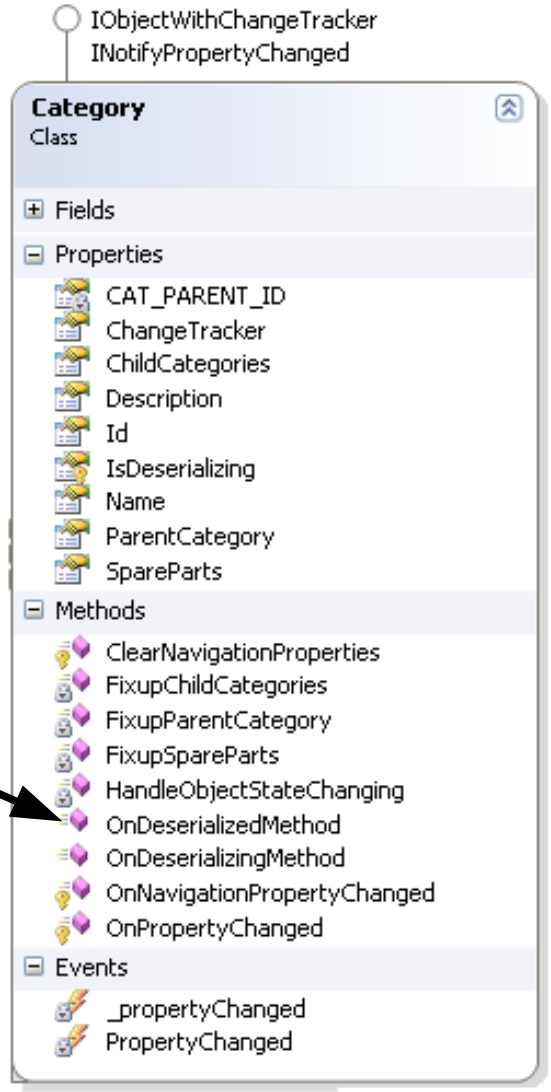
Self-Tracking

- Handhabung eines ID Felds

```
[DataMember]
public long Id {
  get { return _id; }
  set {
    if (_id != value) {
      if (ChangeTracker.ChangeTrackingEnabled &&
          ChangeTracker.State != ObjectState.Added) {
        throw new InvalidOperationException("The
              property 'Id' is part of the object's key and
              cannot be changed. Changes to key properties can
              only be made when the object is not being tracked
              or is in the Added state.");
        }
        _id = value;
        OnPropertyChanged("Id");
      }
    }
  }
```

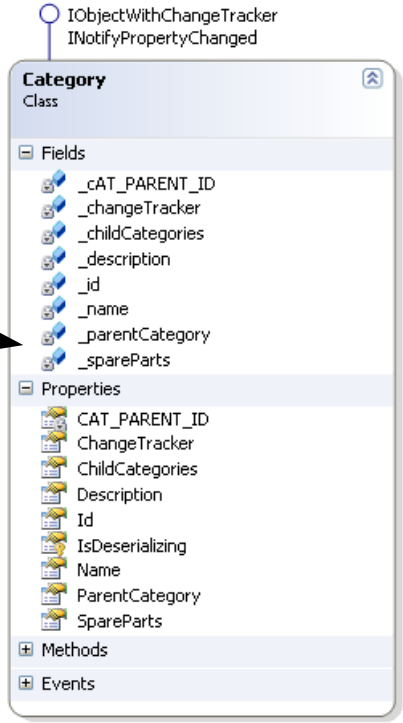
- Self-Tracking wird bei der Deserialisierung aktiviert

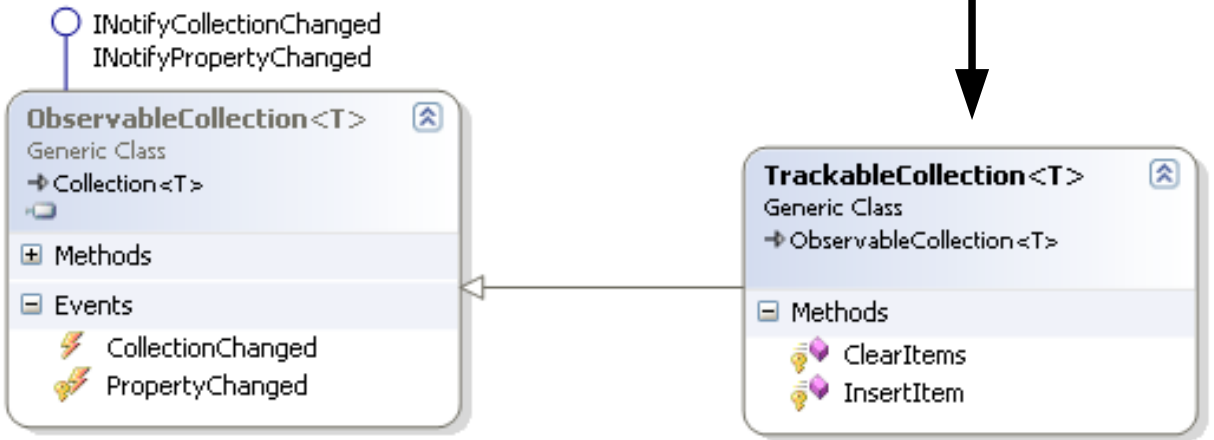
```
[OnDeserialized]
public void OnDeserializedMethod(
        StreamingContext context) {
   IsDeserializing = false;
 ChangeTracker.ChangeTrackingEnabled
    = true;
}
```



- Self-Tracking von Beziehungen

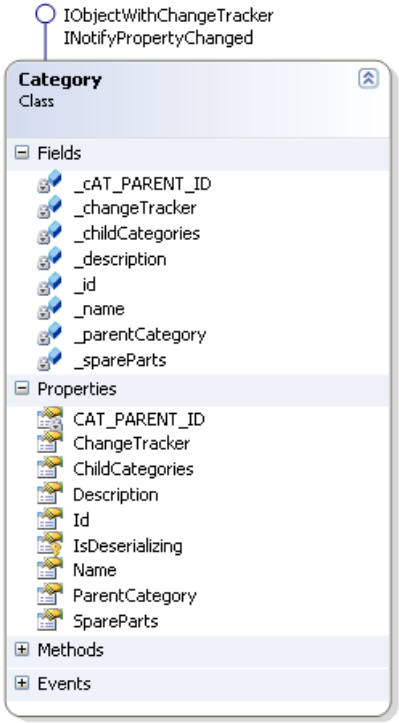
```
private TrackableCollection<SparePart>
    _spareParts;

[DataMember]
public TrackableCollection<SparePart>
        SpareParts {
    get ;  // siehe folgende Folien
    set; // siehe folgende Folien
}
```

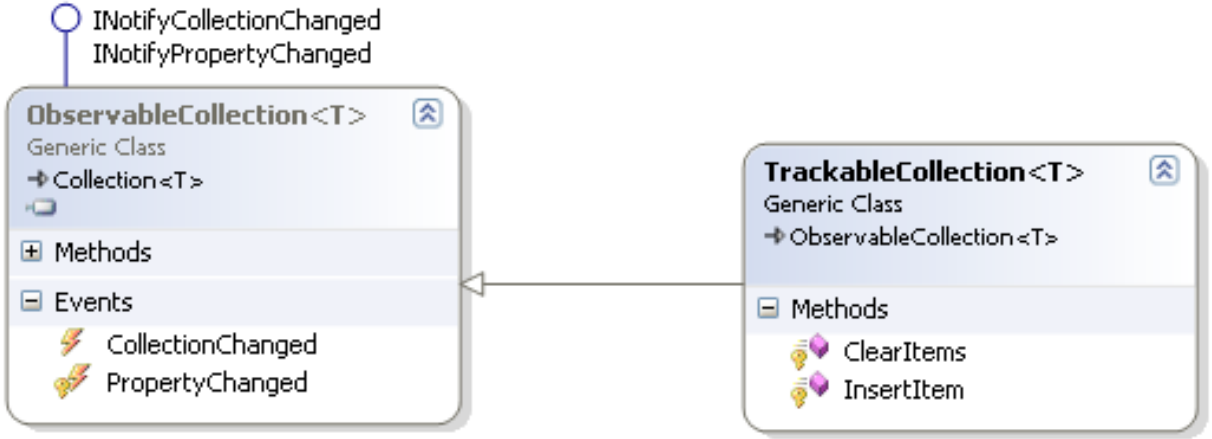




- Self-Tracking von Beziehungen

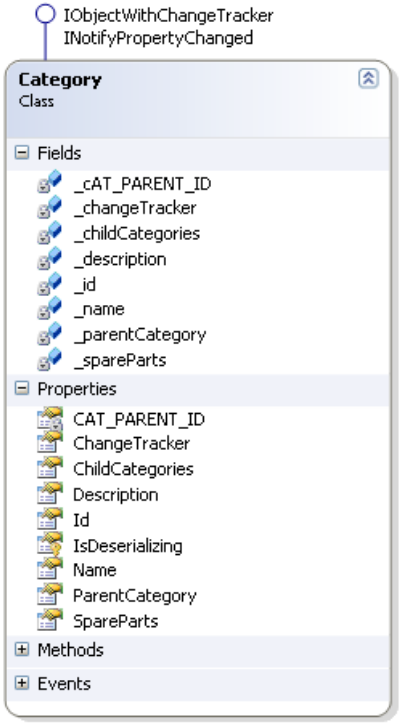
```
get {
    if (_spareParts == null) {
        _spareParts = new
          TrackableCollection<SparePart>();
        _spareParts.CollectionChanged +=
            FixupSpareParts;
    }

    return _spareParts;
  }
```

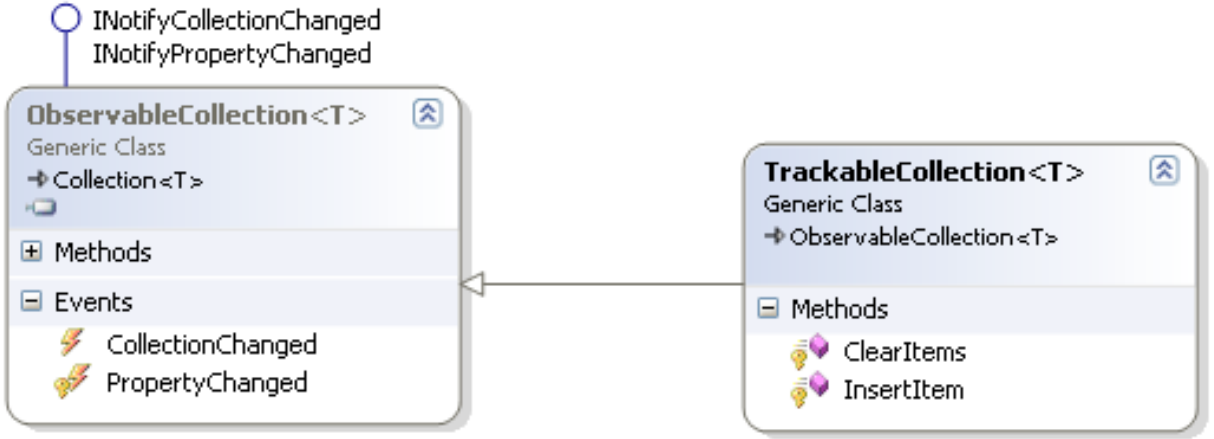




- Was passiert im Client ??

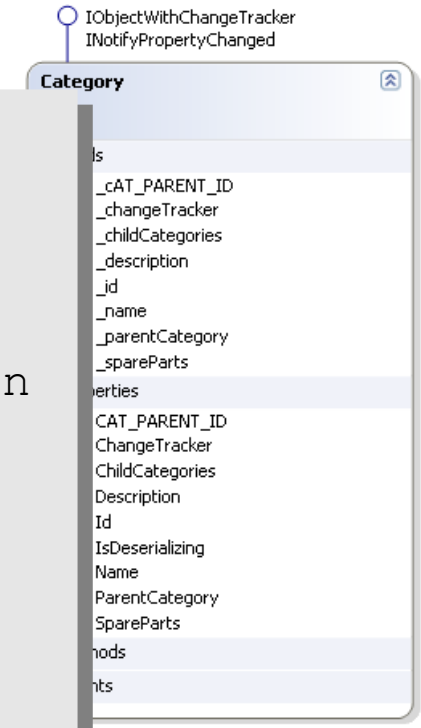
```
Category transferCategory =
    service.GetCategory(meineID);

transferCategory.SpareParts =
  new TrackableCollection<SparePart>();

transferCategory.SpareParts.Add(new ...);
```

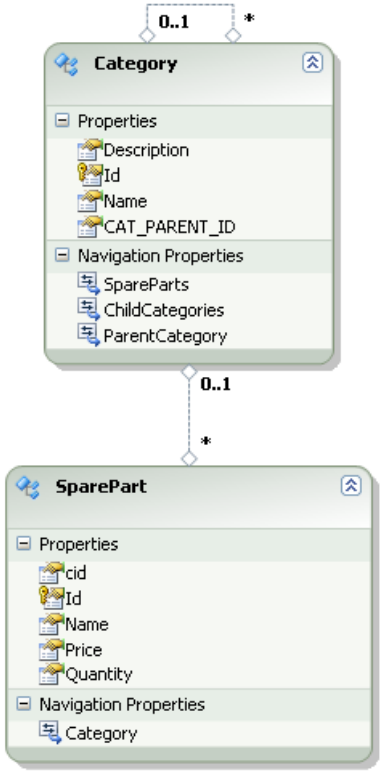




- Self-Tracking von Beziehungen

```
set {
  if (!ReferenceEquals(_spareParts,
                       value)) {
    if (ChangeTracker.ChangeTrackingEnabled) {
      throw new InvalidOperationException(
       "Cannot set the FixupChangeTrackingCollection
        when ChangeTracking is enabled");
    }
    if (_spareParts != null) {
      _spareParts.CollectionChanged -=
                          FixupSpareParts;
    }
    _spareParts = value;
    if (_spareParts != null) {
      _spareParts.CollectionChanged +=
                          FixupSpareParts;
    }
    OnNavigationPropertyChanged("SpareParts");
  }
}
```

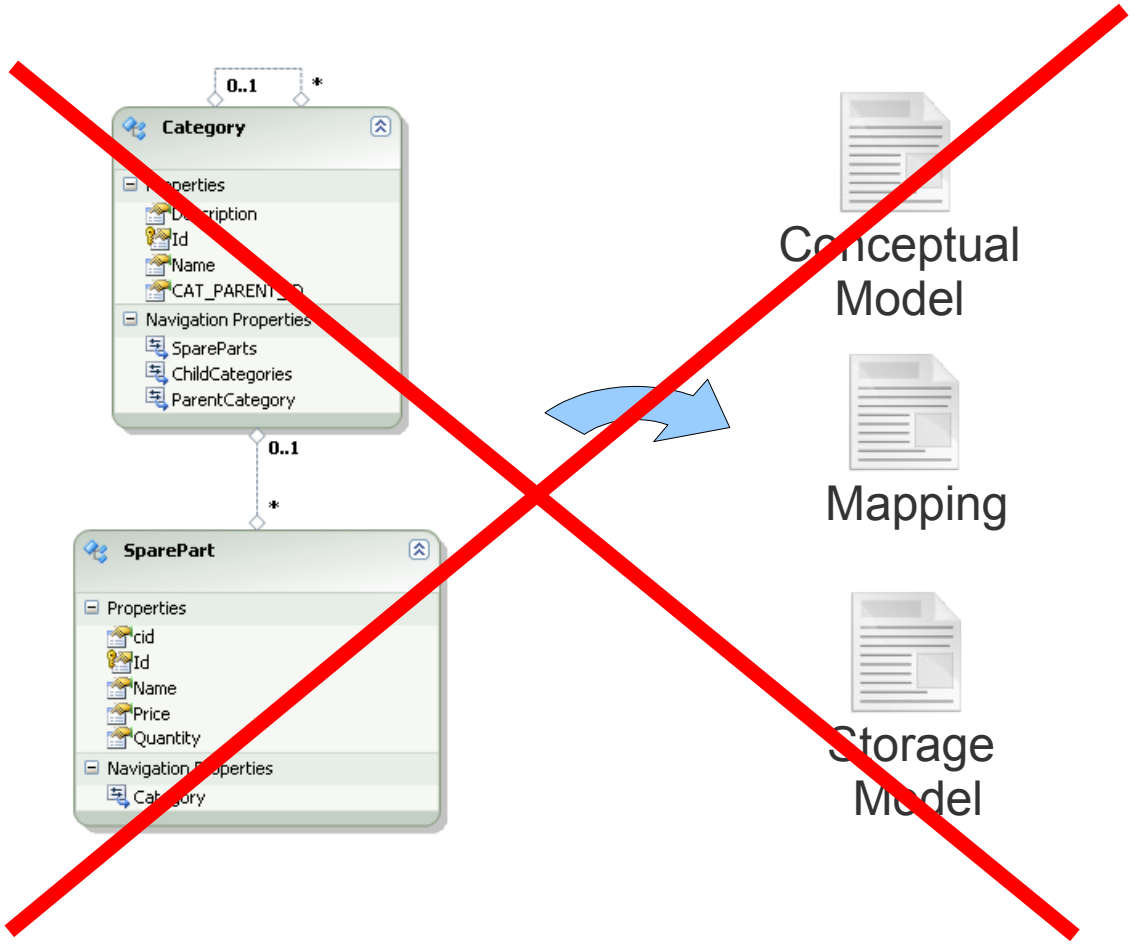




Conceptual Model

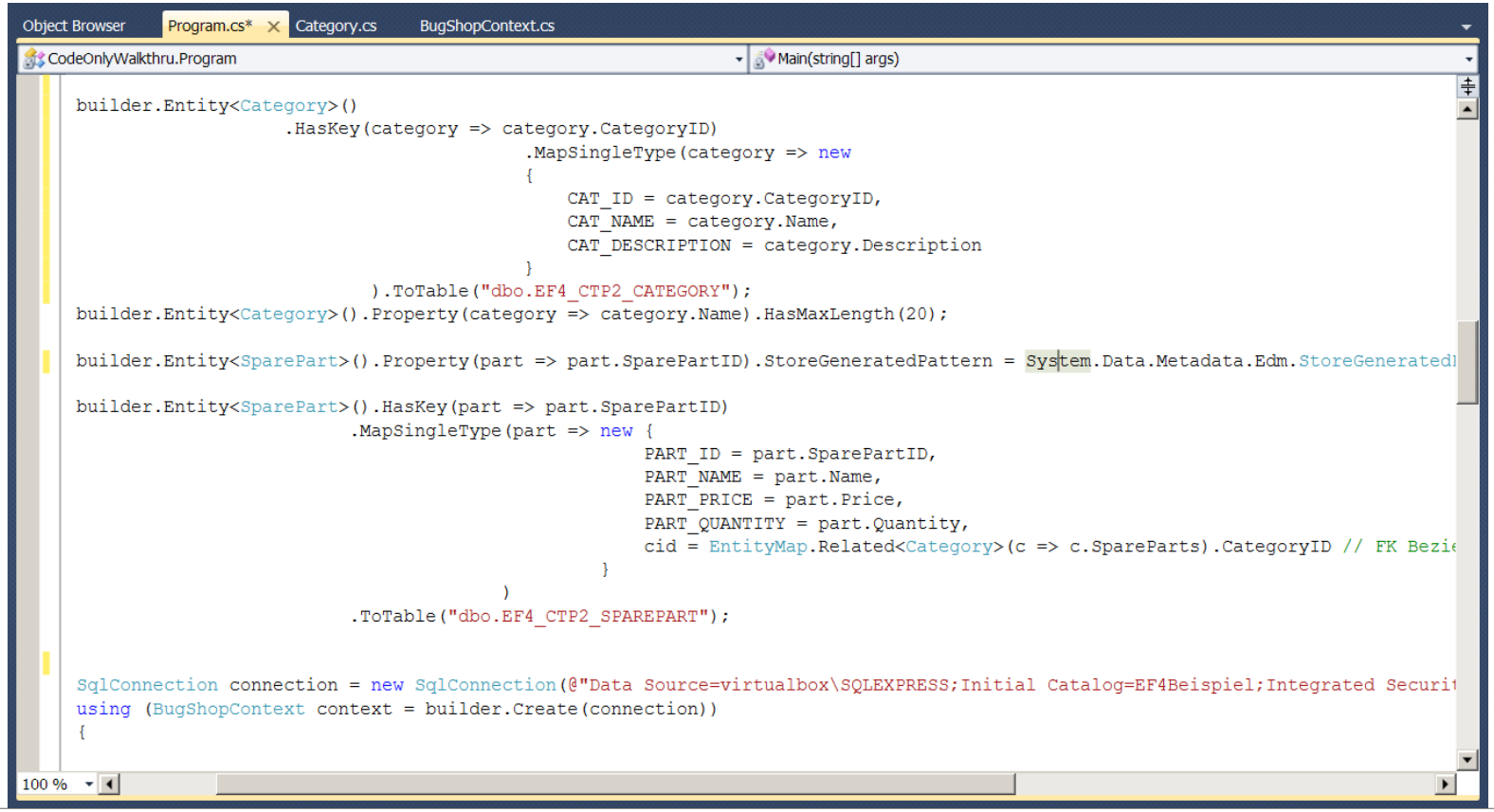
Mapping

Storage Model

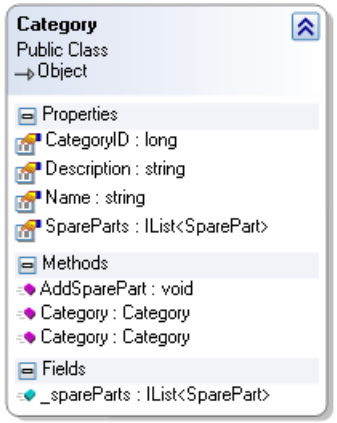


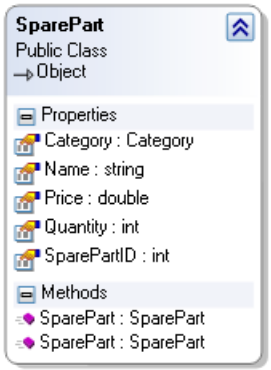
- Kein integraler Bestandteil des Entity Frameworks 4.0

- Kein integraler Bestandteil des Entity Frameworks 4.0
  - Installation des CTP4 notwendig



- Ausgangssituation
  - Entitätsklassen ohne Entity Framework Abhängigkeiten



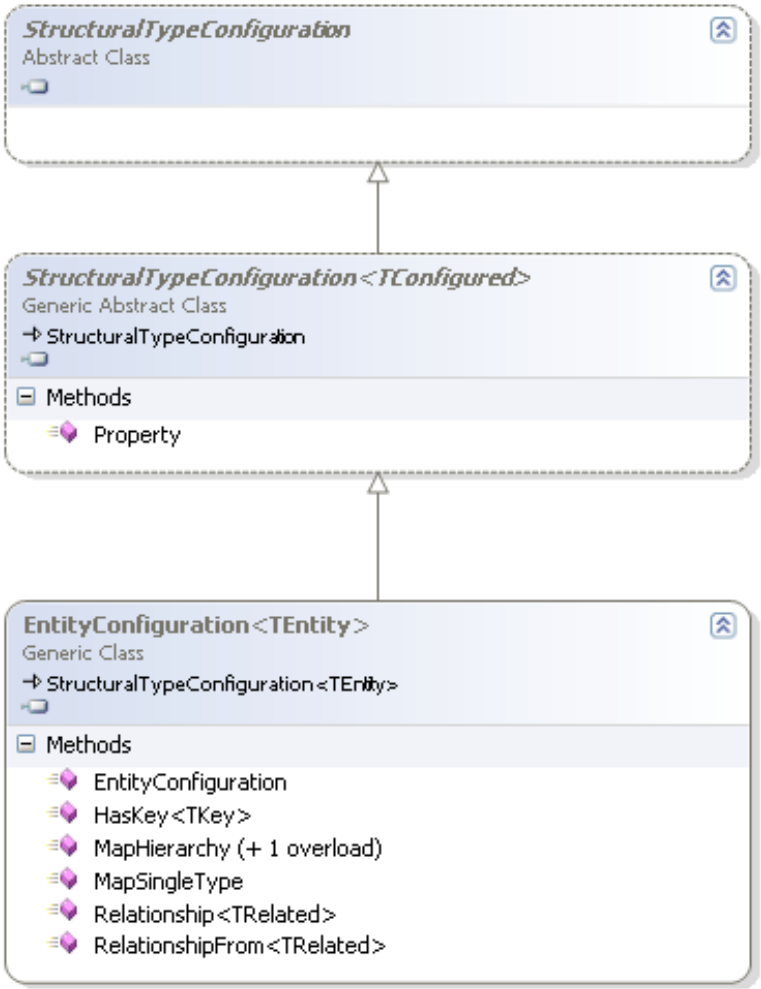



```
public class Category
{
    private IList<SparePart> _spareParts= new List<SparePart>();

    public virtual long CategoryID { get; private set; }
    public virtual string Name { get; set; }
    public virtual string Description { get; set; }

    public virtual IList<SparePart> SpareParts
    {
        get {
            return _spareParts;
        }
        set {
            _spareParts = value;
        }
    }

    public Category()
    {
    }

    public Category(string name) {
        if (name == null) {
            throw new System.ArgumentNullException("name", "parameter must not be null");
        }
        this.Name = name;
    }

    public void AddSparePart(SparePart part)
    {
        if (part == null)
        {
            return;
        }
        part.Category = this;
        _spareParts.Add(part);
    }
}
```

- ObjectContext

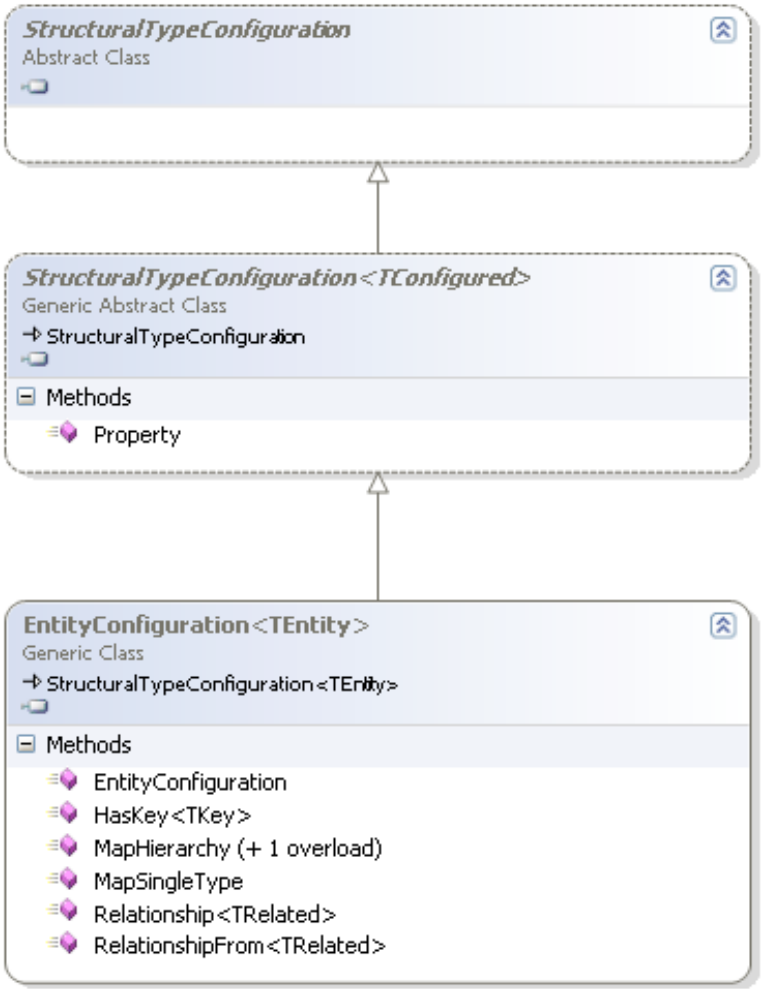
```
public class BugShopContext: ObjectContext {
   public BugShopContext(EntityConnection connection)
            : base(connection, "BugShopContext") {
      ContextOptions.LazyLoadingEnabled= true;
   }

   IObjectSet<Category> _Categories;
   public IObjectSet<Category> Categories {
      get {
         if (_Categories == null)
         {
            _Categories =
               base.CreateObjectSet<Category>("Categories");
         }
         return _Categories;
      }
   }

   // weitere Properties …
}
```

- Entitäten auf Tabellen abbilden
  - Mittels `EntityConfiguration`
    - Fluent Interface
    - Identifikatoren (Primary Key)
    - Vererbungshierarchien oder Singuläre Entitäten
    - Beziehungen zwischen Entitäten
  - Per Attribute
    - noch nicht verfügbar



- Entitäten auf Tabellen abbilden
  - Identifikatoren (Primary Key)

```
ContextBuilder<BSContext> builder =
  new ContextBuilder<BSContext>();

builder.Entity<Category>()
 .HasKey(category =>
         category.CategoryID);
```



- Entitäten auf Tabellen abbilden
  - Properties auf DB Spalten abbilden
  - Tabellennamen festlegen

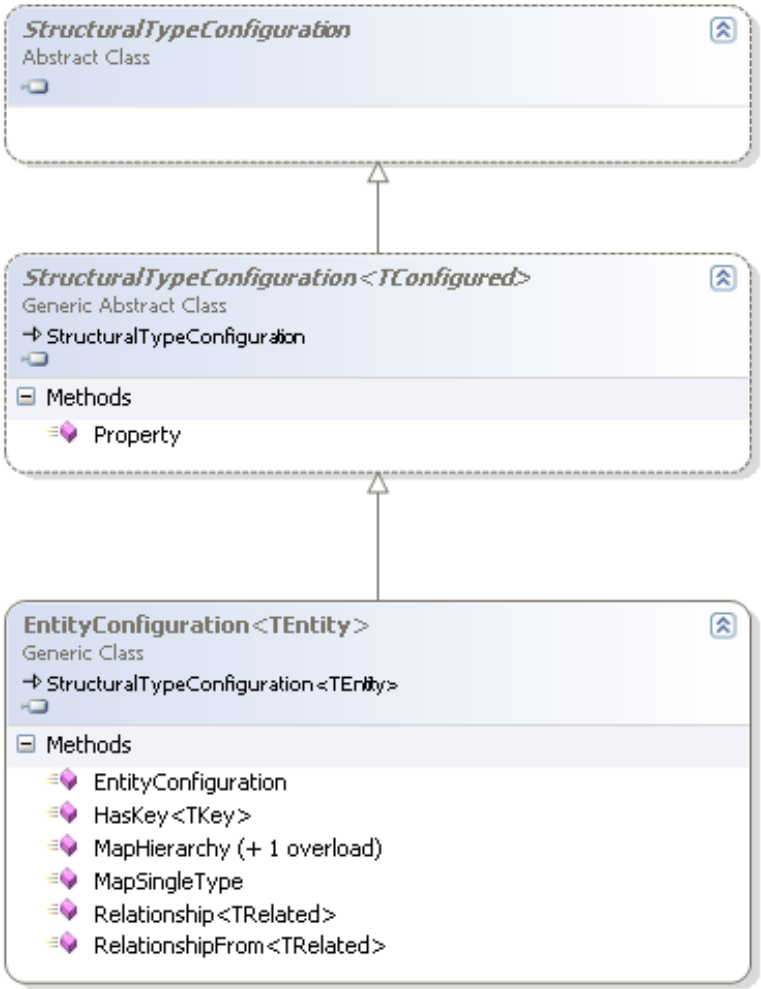
```
builder.Entity<Category>()
  .MapSingleType(
     cat => new {
       CAT_ID = cat.CategoryID,
       CAT_NAME = cat.Name,
       CAT_DESC = cat.Description
     }
   );

builder.Entity<Category>()
  .ToTable("EF4_CTP2_CATEGORY");
```

EF4_CTP2_CATEGORY

Column Name	Condensed Type	Nullable	Identity
CAT_ID	bigint	No	☐
CAT_NAME	nvarchar(4000)	Yes	☐
CAT_DESC	nvarchar(4000)	Yes	☐
			☐

- Entitäten auf Tabellen abbilden
  - Fluent Interface

```
ContextBuilder<BSContext> builder =
  new ContextBuilder<BSContext>();

builder.Entity<Category>()
 .HasKey(category =>
         category.CategoryID)
 .MapSingleType(
   cat => new {
     CAT_ID = cat.CategoryID,
     CAT_NAME = cat.Name,
     CAT_DESC = cat.Description
   }
  )
 .ToTable("EF4_CTP2_CATEGORY");
```



- Persistente Properties 'einschränken'

EF4_CTP2_CATEGORY

	Column Name	Condensed Type	Nullable	Identity
🔑	CAT_ID	bigint	No	☐
	CAT_NAME	nvarchar(4000)	Yes	☐
	CAT_DESC	nvarchar(4000)	Yes	☐
				☐

```
EntityConfiguration<Category>
  catConf =
        builder.Entity<Category>();

catConf.Property(cat => cat.CategoryID)
      .IsIdentity();
```

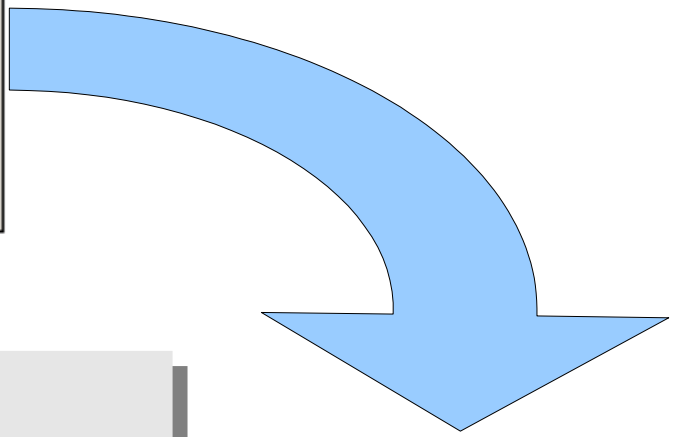
EF4_CTP2_CATEGORY

	Column Name	Condensed Type	Nullable	Identity
🔑	CAT_ID	bigint	No	☑
	CAT_DESC	nvarchar(200)	Yes	☐
	CAT_NAME	nvarchar(20)	No	☐
				☐

- Persistente Properties 'einschränken'

EF4_CTP2_CATEGORY

	Column Name	Condensed Type	Nullable	Identity
🔑	CAT_ID	bigint	No	☐
	CAT_NAME	nvarchar(4000)	Yes	☐
	CAT_DESC	nvarchar(4000)	Yes	☐
				☐

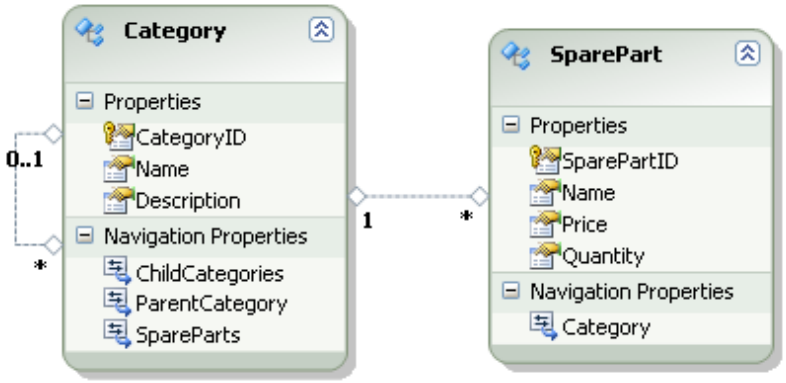


```
EntityConfiguration<Category>
  catConf =
        builder.Entity<Category>();

catConf.Property(cat => cat.Name)
   .HasMaxLength(20);
catConf.Property(cat => cat.Name)
   .IsRequired();  // not null
```

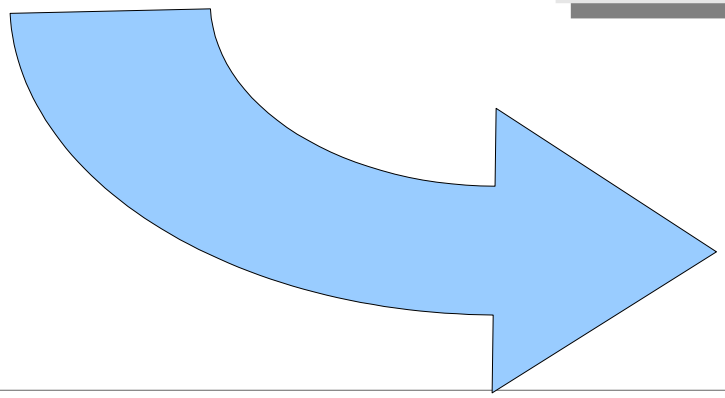
EF4_CTP2_CATEGORY

	Column Name	Condensed Type	Nullable	Identity
🔑	CAT_ID	bigint	No	☑
	CAT_DESC	nvarchar(200)	Yes	☐
	CAT_NAME	nvarchar(20)	No	☐
				☐

- Beziehungen




```
builder.Entity<SparePart>()
   .HasKey(part => part.SparePartID)
   .MapSingleType(part => new  {
            PART_ID = part.SparePartID,
            PART_NAME = part.Name,
            PART_PRICE = part.Price,
            PART_QUANT = part.Quantity,
            cid =
         EntityMap.Related<Category>
       (c => c.SpareParts).CategoryID
    }
 ).ToTable("EF4_CTP2_SPAREPART");
```



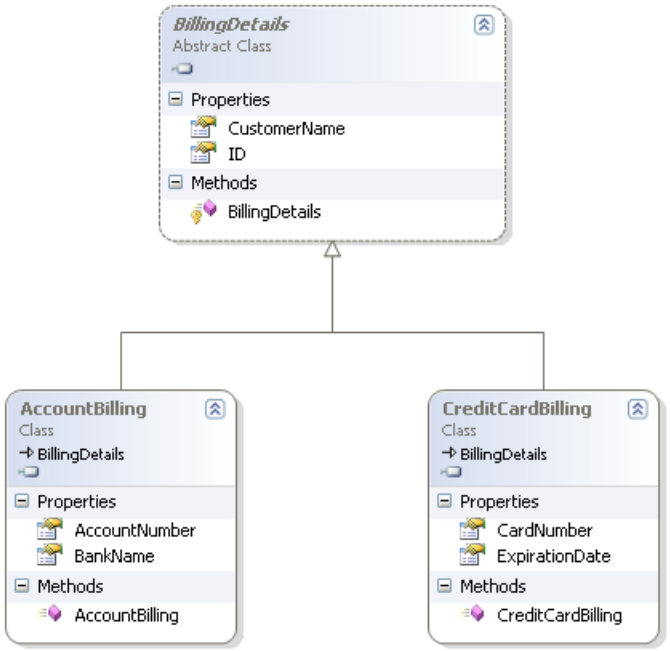
EF4_CTP2_CATEGORY

Column Name	Condensed Type	Nullable
CategoryID	bigint	No
Description	nvarchar(200)	Yes
Name	nvarchar(20)	No

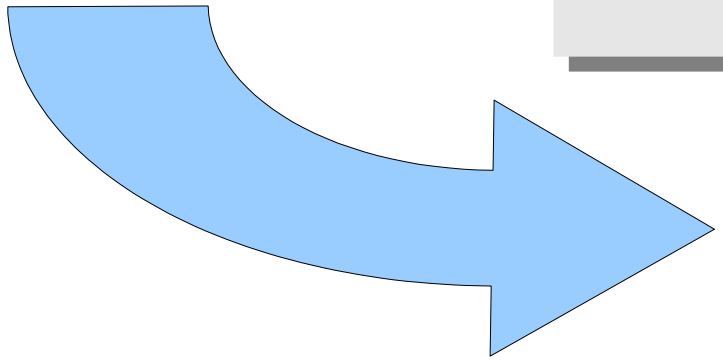
EF4_CTP2_SPAREPART

Column Name	Condensed Type	Nullable
cid	bigint	Yes
PART_ID	int	No
PART_NAME	nvarchar(4000)	Yes
PART_PRICE	float	No
PART_QUA...	int	No

- Vererbung: Table per Hierarchie



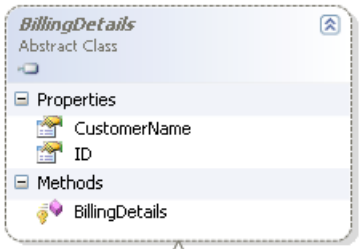

```
builder.Entity<BillingDetails>()
  .HasKey(billing => billing.ID)
  .MapHierarchy()
  .Case<BillingDetails>( details => new {
       details.ID,
       details.CustomerName
      }
   )
  .Case<AccountBilling>( account => new {
       account.AccountNumber,
       account.BankName,
       discriminator = "Account"
    }
   )
  .ToTable("EF4 CTP2 BILLING");
```



EF4_CTP2_BILLING

	Column Name	Data Type	Nullable	Identity
	AccountNumber	nvarchar(4000)	Yes	☐
	BankName	nvarchar(4000)	Yes	☐
	CardNumber	nvarchar(4000)	Yes	☐
	CustomerName	nvarchar(4000)	Yes	☐
	discriminator	nvarchar(4000)	No	☐
	ExpirationDate	datetime	Yes	☐
🔑	ID	bigint	No	☑
				☐

- Vererbung: Table per Concrete Type




```
builder.Entity<BillingDetails>()
   .MapHierarchy(details => new {
       details.ID,
       details.CustomerName
    })
.ToTable("EF4_CTP2_BILLING_DETAILS");
```

```
builder.Entity<AccountBilling>()
   .MapHierarchy(account => new {
       Account_ID =account.ID,
       account.AccountNumber,
       account.BankName
    })

.ToTable("EF4_CTP2_BILLING_ACCOUNT");
```

EF4_CTP2_BILLING_DETAILS

Column Name	Condensed Type	Nullable	Identity
CustomerName	nvarchar(4000)	Yes	☐
ID	bigint	No	☑
			☐

BILLING_ACCOUNT

Column Name	Condensed Type	Nullable	Identity
Account_ID	bigint	No	☐
AccountNumber	nvarchar(4000)	Yes	☐
BankName	nvarchar(4000)	Yes	☐
			☐

EF4_CTP2_BILLING_CARD

Column Name	Condensed Type	Nullable	Identity
Card_ID	bigint	No	☐
CardNumber	nvarchar(4000)	Yes	☐
ExpirationDate	datetime	No	☐
			☐

- ComplexTypes
  - Definition und Konfiguration

```
public class Customer {
  public string LastName { get; set; }
  public string FirstName { get; set; }
  public string Salutation { get; set; }
  public long CustomerNumber { get; set; }
}
```

```
ComplexTypeConfiguration<Customer>
   custConf = builder.ComplexType<Customer>();
custConf.Property(customer => customer.LastName)
   .IsRequired().HasMaxLength(20);
custConf.Property(customer => customer.FirstName)
   .IsRequired().HasMaxLength(30);
custConf.Property(customer => customer.Salutation)
   .HasMaxLength(10);
```

- ComplexTypes
  - Verwendung

```
builder.Entity<BillingDetails>()
   .HasKey(billing => billing.ID)
   .MapHierarchy()
   .Case<BillingDetails>( details => new {
       details.ID,
       C_LastName = details.Customer.LastName,
       C_FirstName = details.Customer.FirstName,
       C_Salutation = details.Customer.Salutation,
       C_Number = details.Customer.CustomerNumber
    } )
   .Case<AccountBilling> ( ...);
```

- Attribut-basiertes Code First
  - Data Annotations (teilweise von Silverlight)

```
public class Person
{
    [Key]
    public string SSN { get; set; }

    [StringLength(512)]
    public string Name { get; set; }

    [RelatedTo(RelatedProperty="Author")]
    public ICollection<Book> Books { get; set; }
}
```

- Attribut-basiertes Code First
  - Data Annotations (teilweise von Silverlight)
    - Key
    - StringLength
    - ConcurrencyCheck
    - Required
    - Timestamp
    - DataMember
    - RelatedTo
    - MaxLength
    - StoreGenerated

- Produktivitätsverbesserungen

```
public class BugshopContext : DbContext {
  public DbSet<Category> Categories { get;set; }
}

public class Category {
  public int CategoryId { get; set; }
  public string Name { get; set; }
  public string Description { get; set; }
}
```

- Attribut-basiertes Code First (eigene Lösung)

```
[Entity (Table="DB_TableName")]
public class IntegrationTestEntity {

  [Id]
  [Column(Name = "TEST_ID")]
  public long Id { get; set; }

  [Column(Name = "TEST_NAME", Length=20,
          Nullable=false)]
  public string Name { get; set; }

  [Column(Name = "TEST_VORNAME", Length=15,
          Nullable=true)]
  public string Vorname { get; set; }
}
```

- Attribut-basiertes Code First (eigene Lösung)
  - Demo

john_a_ward @ flickr

Fazit

Erweiterung des Entity Frameworks waren notwendig

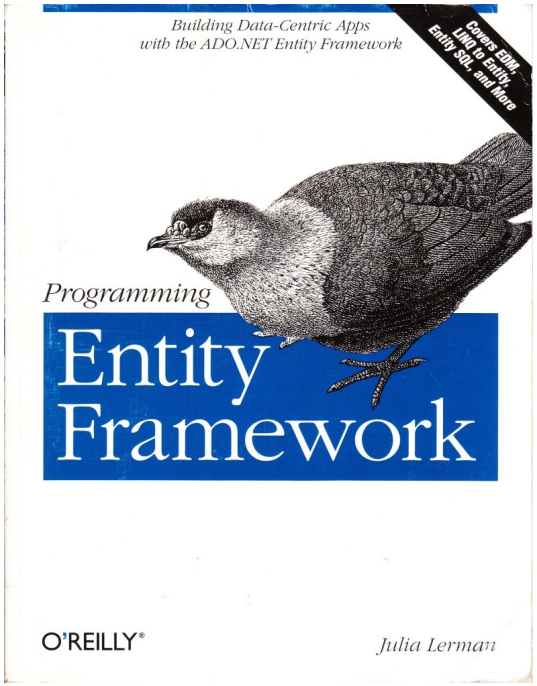
Wichtige und notwendige Erweiterungen

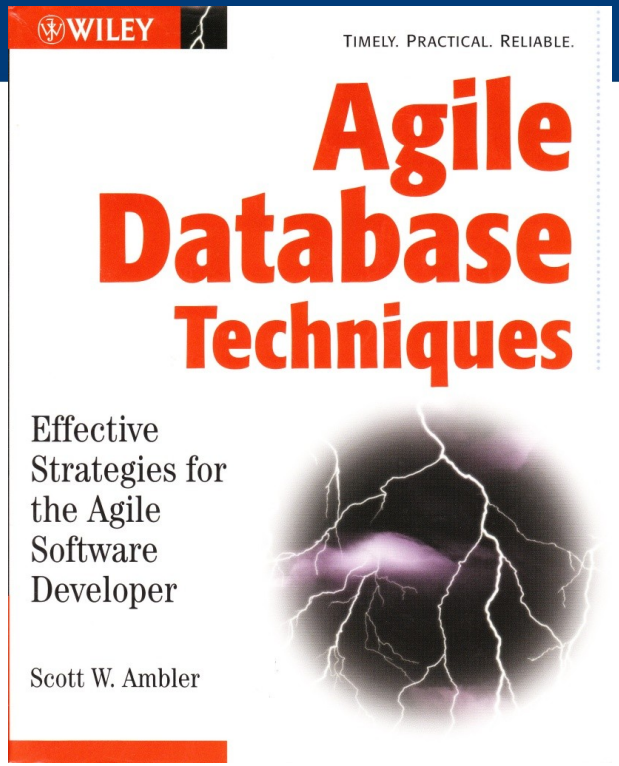
- Lazy Loading
- Model First

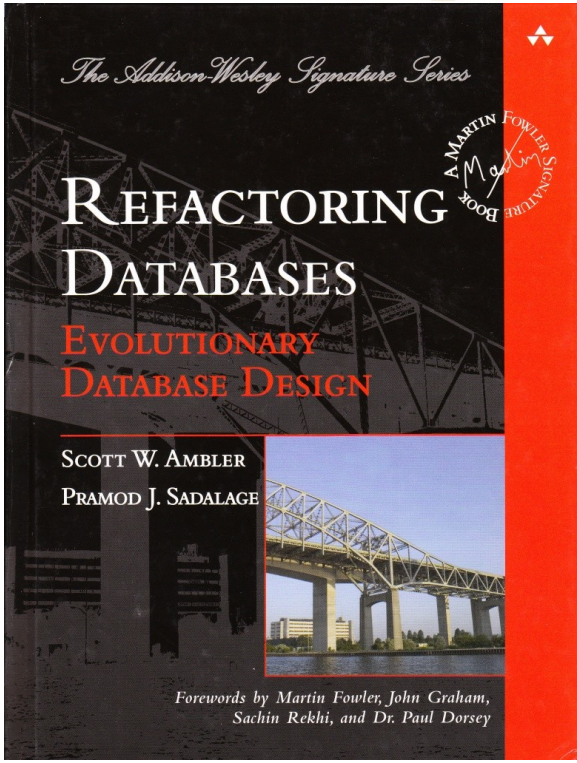
Schöne Erweiterungen

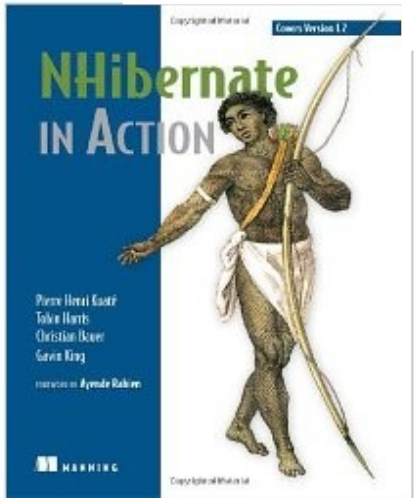
- Self-Tracking
- Code First (kein integraler Bestandteil, API instabil)

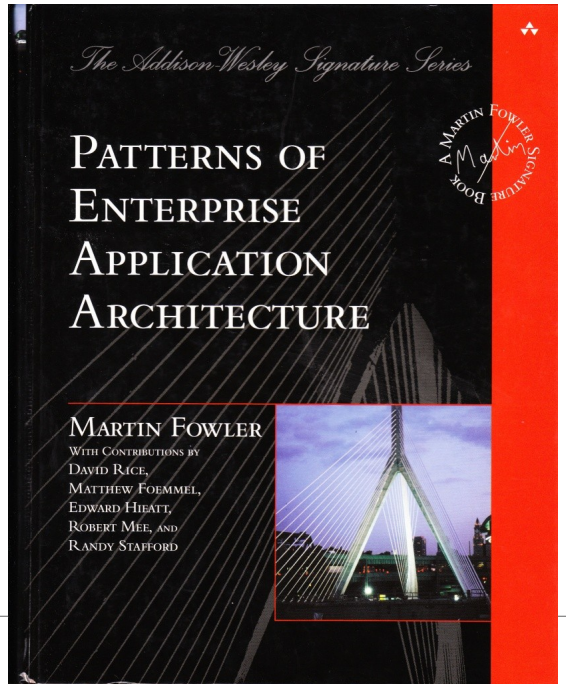
Wir haben nur einen Bruchteil gesehen


Building Data-Centric Apps
with the ADO.NET Entity Framework
Covers EDM, LINQ to Entity, Entity SQL, and More
Programming
Entity Framework
O'REILLY
Julia Lerman


WILEY
TIMELY. PRACTICAL. RELIABLE.
Agile Database Techniques
Effective Strategies for the Agile Software Developer
Scott W. Ambler


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Refactoring Databases
Evolutionary Database Design
Scott W. Ambler
Pramod J. Sadalage
Forewords by Martin Fowler, John Graham, Sachin Rekhi, and Dr. Paul Dorsey


NHibernate IN ACTION
Covers Version 1.2
MANNING


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Patterns of Enterprise Application Architecture
Martin Fowler
With Contributions by
David Rice,
Matthew Foemmel,
Edward Hieatt,
Robert Mee, and
Randy Stafford

Links

Herbstcampus

- POCO and Persistence Ignorance
- T4 Code Generation
- Lazy Loading
- Change-Tracking / N-Tier Support
- Model-First
- Code-First
- Pluralization
- Data Annotations
- Productivity Improvements

Vielen Dank!

Thomas Haug

MATHEMA Software GmbH

- **SQL mittels ObjectContext ausführen**

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

    // Parameter syntax with object values.
    foreach (string name in
                context.ExecuteStoreQuery<string>
        ("Select Name from Category where
          CategoryID < @p0", 5))
    {
        Console.WriteLine(name);
    }
}
```

- **SQL mittels ObjectContext ausführen**

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

    // Parameter syntax with object values.
    foreach (string name in
                context.ExecuteStoreQuery<string>
        ("Select Name from Department where
          DepartmentID < @p0",
                new SqlParameter {
                    ParameterName = "p0",
                    Value = 5
                }
         ))
    {
        Console.WriteLine(name);
    }
}
```

- **SQL mittels ObjectContext ausführen**

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

    // Parameter substitution
    foreach (string name in
       context.ExecuteStoreQuery<string>
        ("Select Name from Category where
          CategoryID < {0}", 5))
    {
        Console.WriteLine(name);
    }
}
```