2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Einheitsrahmenwerk

Entity Framework 6

Thomas Haug
MATHEMA Software GmbH

Agenda

- (Architektur) Überblick
- Entitäten beschreiben
- CRUD
- EF 6 Feature Beispiele
- Zusammenfassung

Agenda

- (Architektur) Überblick
- Entitäten beschreiben
- CRUD
- 6 Feature Beispiele
- Zusammenfassung

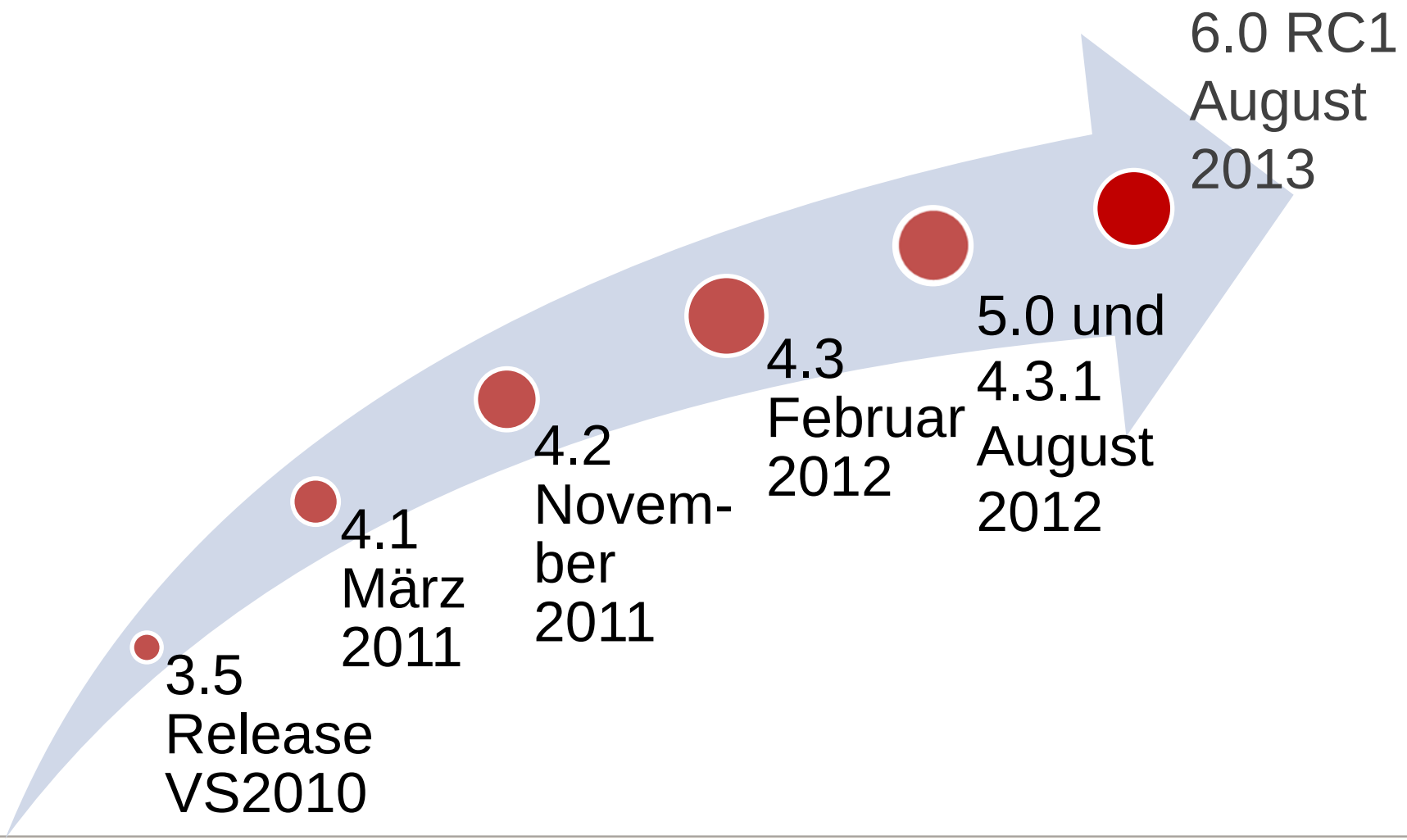
Überblick Entity Framework

- Aktuell Version 5.0 „released“, Version 6 als Release Candidate 1 verfügbar
- Kapselt ADO.Net und SQL vor Entwickler
- Unterstützt SQL Server „out-of-the-box“
- (Transparente) Persistenz für (POCO) Klassen
- Unterstützung von Vererbungshierarchien und polymorphen Abfragen
- Transitive Persistenz für Objektgraphen

Überblick Entity Framework

- 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
  - Ab EF 6 wird alles besser

Entity Framwork - Versionen



Entity Framework – Features 5.0

- Open Source (!)
  - Mono 3.0.x enthält das Entity Framework
- Mehrere Diagramme pro Entity Modell
- Enum Unterstützung
- Spatial Types (DBGeography Typ, DbGeometry Typ)
- Value-Typed Functions (nur bei Database-First)
- Performance Optimierungen unter Verwendung von .NET 4.5 (http://msdn.microsoft.com/en-us/data/hh949853.aspx)

Entity Framework – Features 6.0

- Async Query und Save Methoden (task-based async patterns)
- Custom Code First Conventions
- Code First Mapping für Stored Procedures
- Connection Resilience für temporäre Verbindungsausfälle
- Dependency Resolution zur Unterstützung des Service Locator Patterns
- Code-Based Configuration

Entity Framework – Features 6.0

- Interception/SQL logging
- Konfigurierbare Migrations History Tabelle
- Mehrere Code First Modelle pro Datenbank über mehrere Context Objekte
- Neue Methode DbModelBuilder.HasDefaultSchema im Code First API
- Enums, Spatial und verbesserte Performance für .NET 4.0
- DBContext kann bereits offene DbConnection nutzen

Entity Framework – Features 6.0

- Vereinfachte Transaktionsaktionshandhabung mit DbContext.Database.UseTransaction und DbContext.Database.BeginTransaction
- Standard Transaktionsisolation für Code First auf READ_COMMITTED_SNAPSHOT umgestellt
- Performance Optimierungen für Enumerable.Contains in LINQ Queries
- Verbessertes Startverhalten, insbesondere für große Modelle

Entity Framework 6 - Namespaces

- DbModelBuilder.Configurations.AddFromAssembly für Fluent API mit Configuration Klassen
- Custom Migrations Operations
- Pluggable Pluralization & Singularization Service
- Verbesserte POCO Unterstützung
- System.Data.* wurde nach System.Data.Entity.Core.* „bewegt“, z. B.:

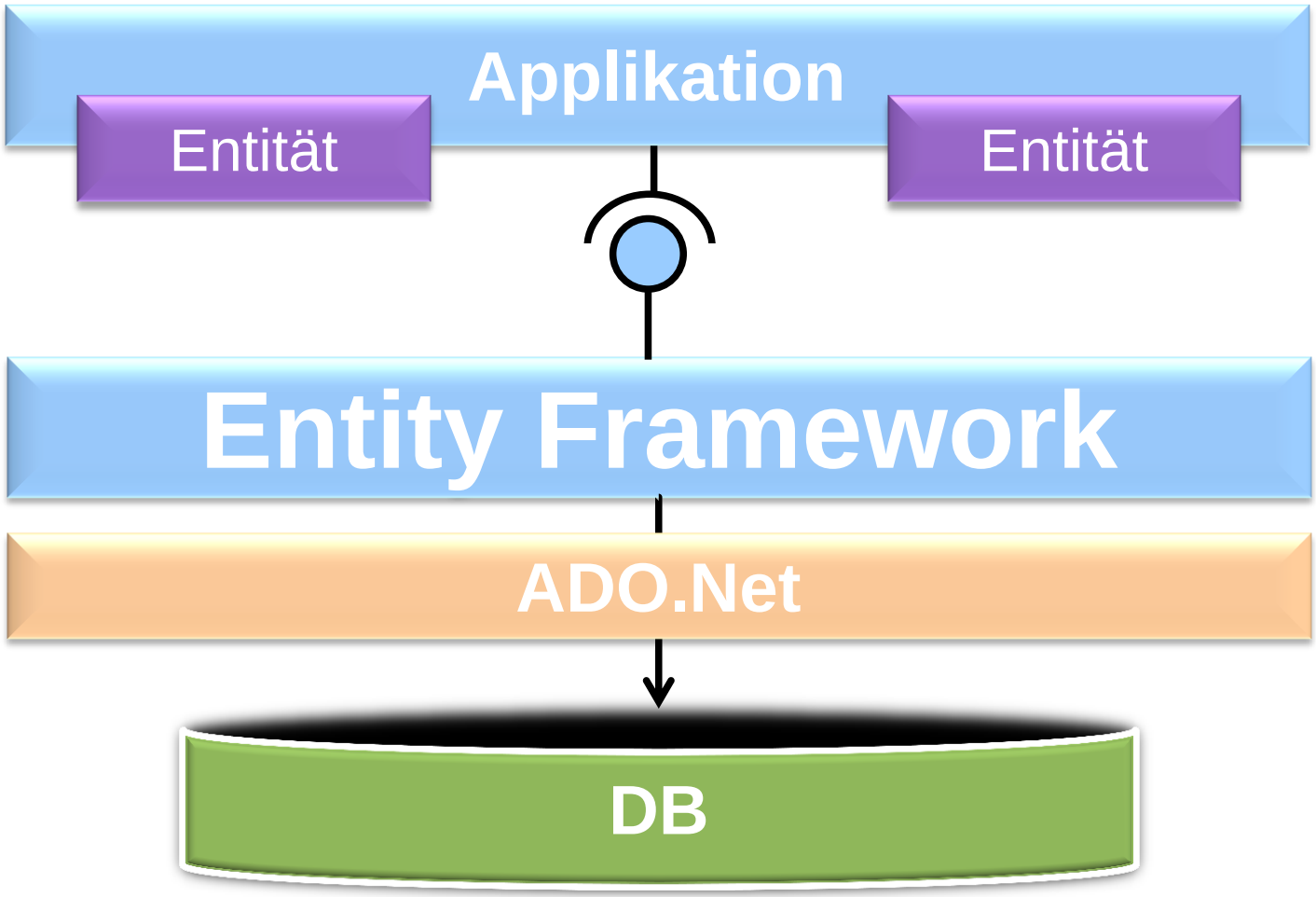
  System.Data.EntityException → System.Data.Entity.Core.EntityException

  System.Data.Objects.ObjectContext → System.Data.Entity.Core.Objects.ObjectContext

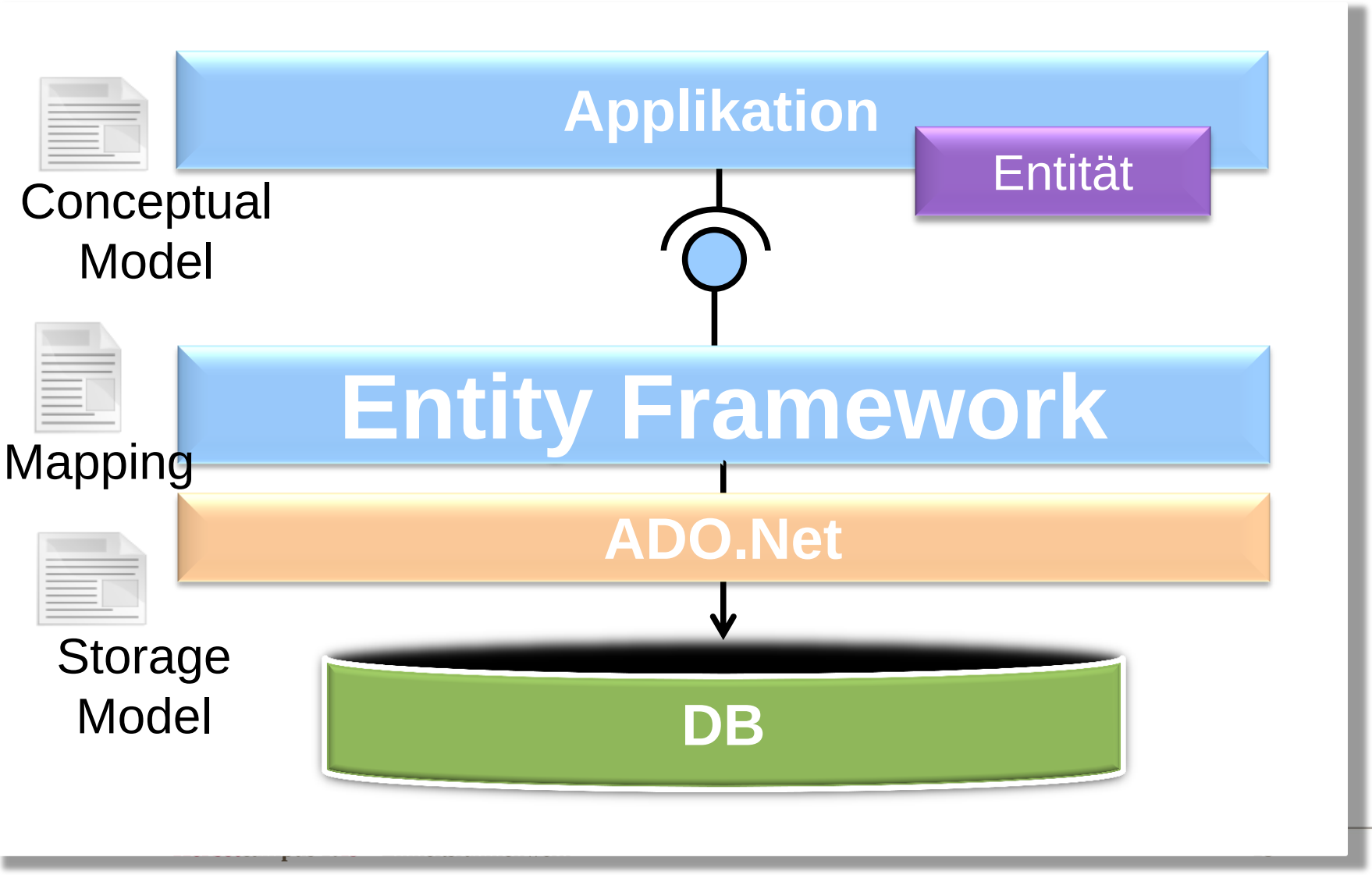
Vergleich

NHibernate	Entity Framework
• Open Source	• Open Source
• Version 3.3.3 GA	• Version 5.0 (6.0 RC1)
• Unterstützt verschiedene Datenbanken	• SQL Server, andere über Drittanbieter
• NHibernate Attributes, Fluently NHibernate	• Code First Annotations, Code First Fluent API
• SQL Tracing leicht möglich	• SQL Tracing ab EF 6.0 leicht möglich, zuvor eingeschränkt

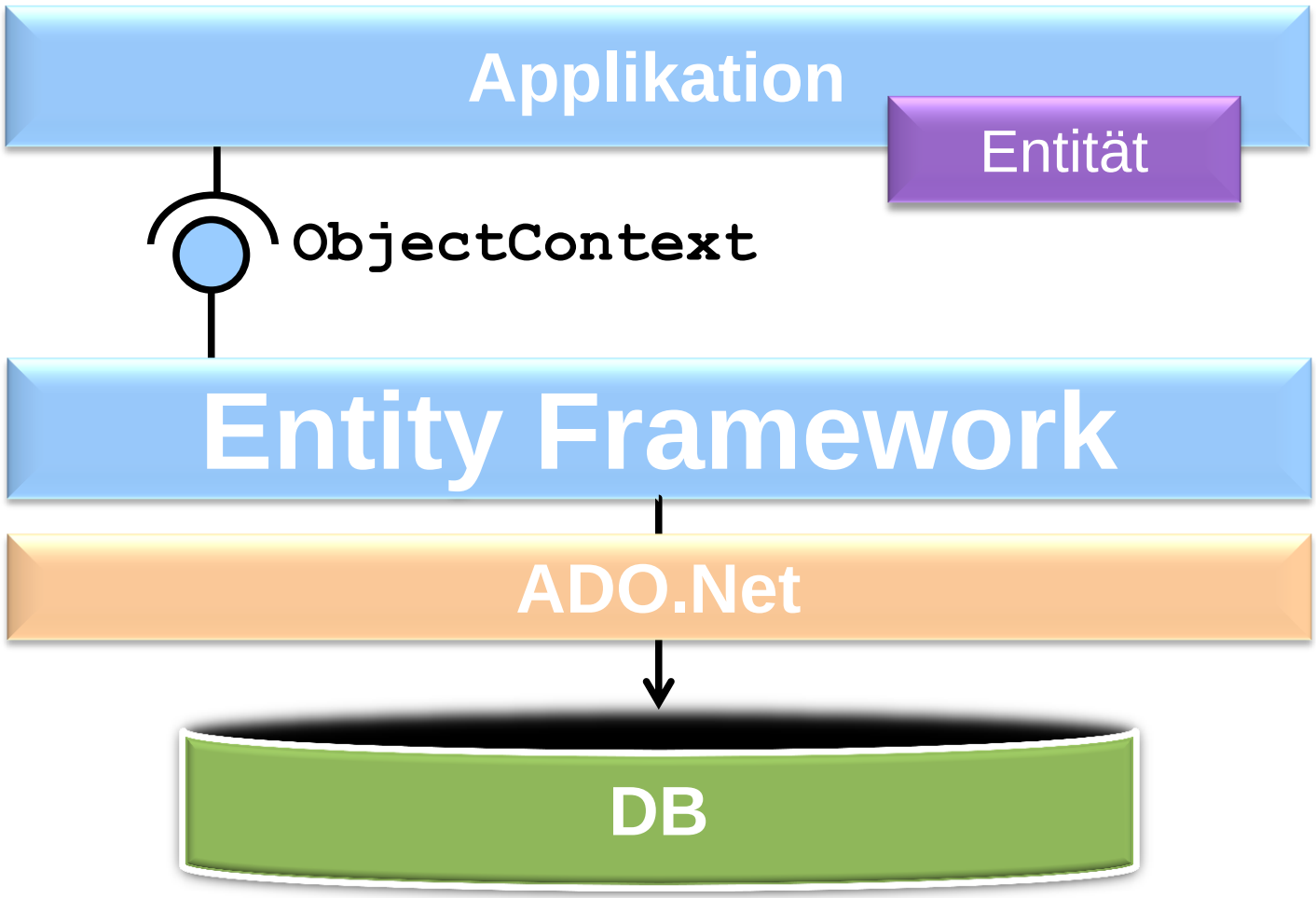
Entity Framework Architektur



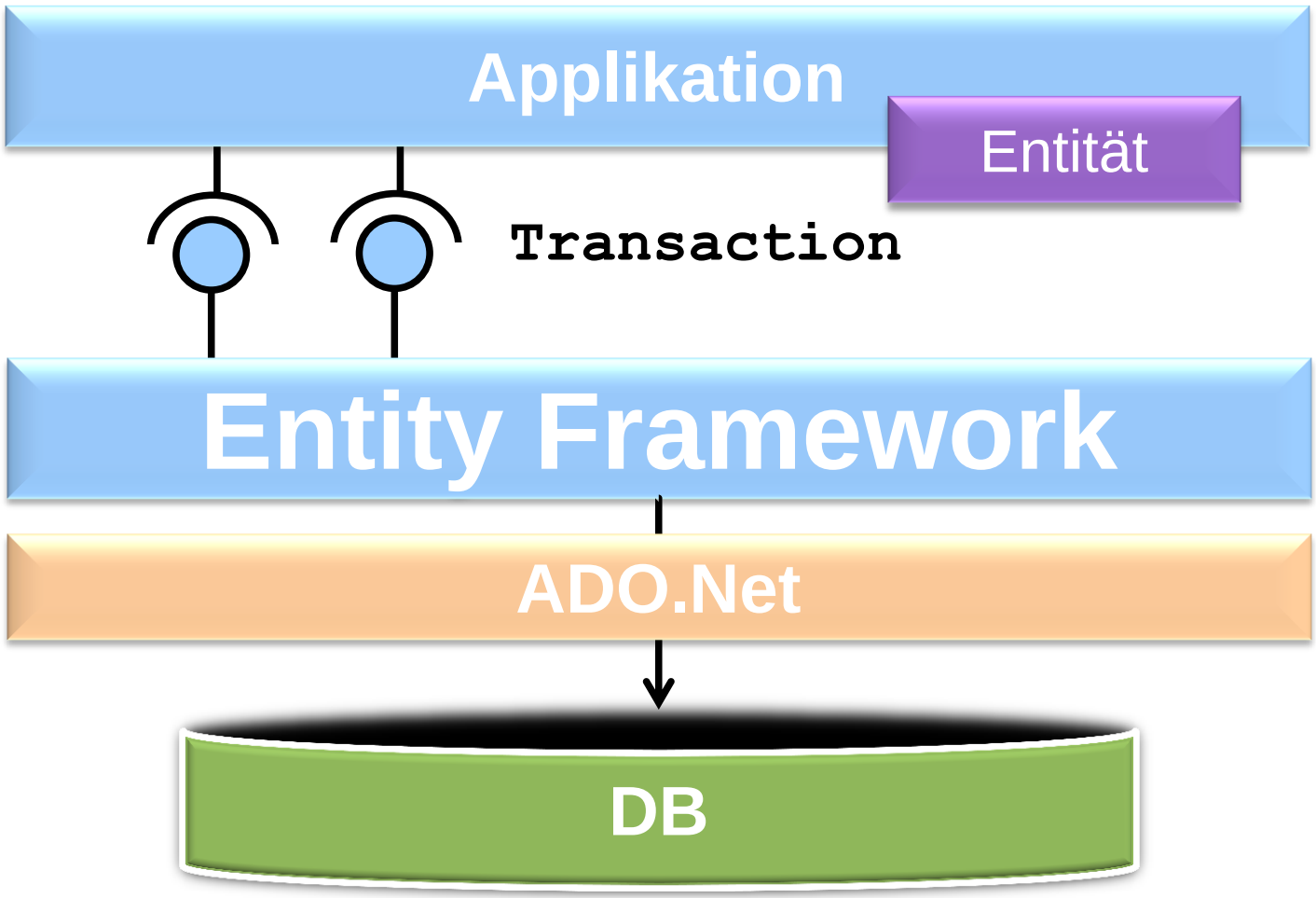
Entity Framework Architektur



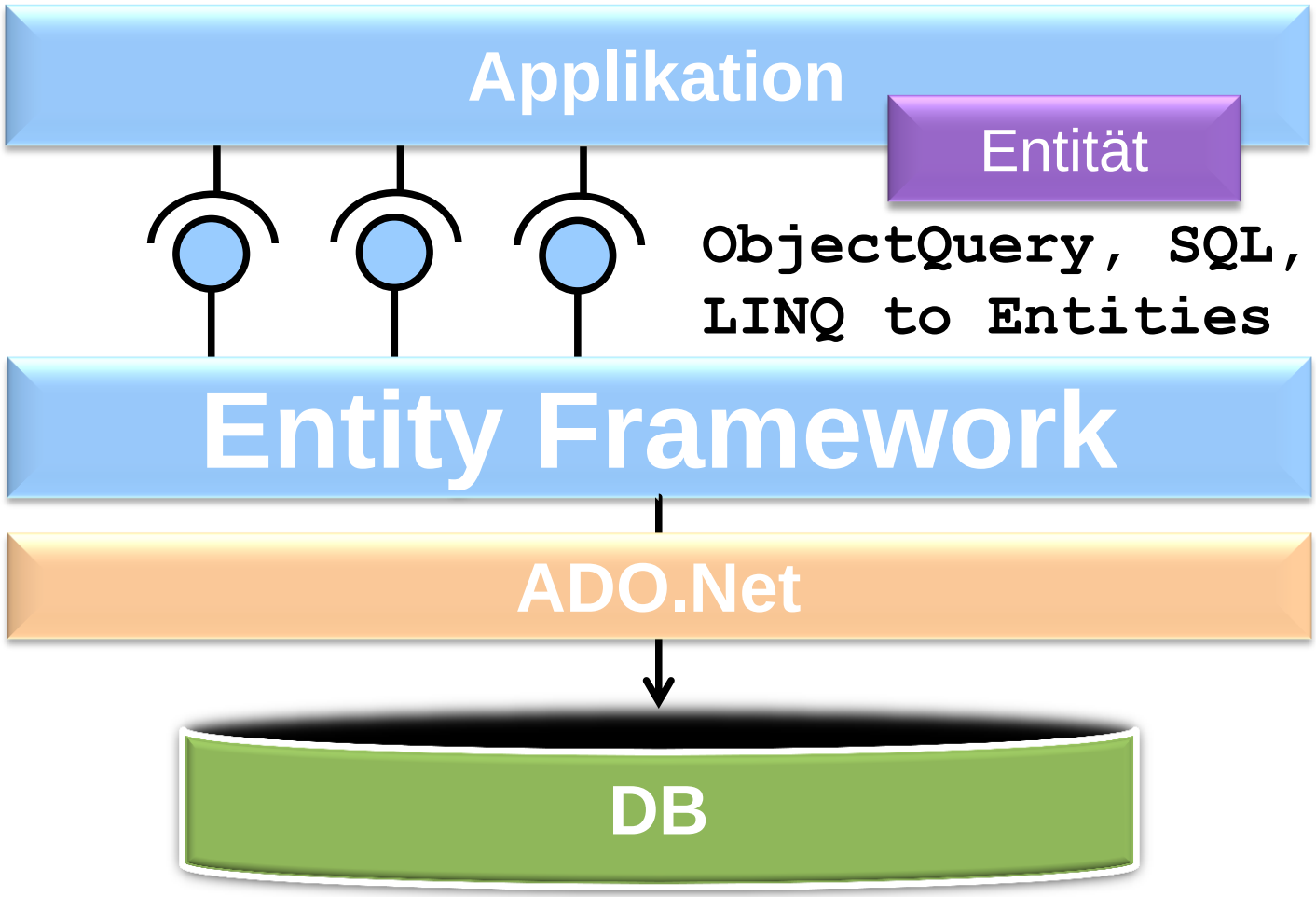
Entity Framework Architektur



Entity Framework Architektur



Entity Framework Architektur



Vergleich

NHibernate

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

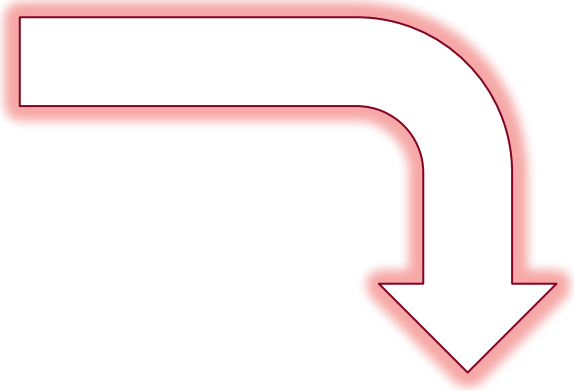
Entity Framework

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

Agenda

- (Architektur) Überblick
- Entitäten beschreiben
- CRUD
- EF 6 Feature Beispiele
- Zusammenfassung

Beispiel mit Code First



Simple_Categories

	Column Name	Data Type	Nullable	Identity
🔑	CategoryID	int	No	☑
	name	nvarchar(50)	No	☐
	description	nvarchar(200)	Yes	☐
				☐

Beispiel mit Code First

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("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; }
}
```

Fluent API 1

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");

  }

}
```

Fluent API 2

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

```
n/a
```

Agenda

- (Architektur) Überblick
- Entitäten beschreiben
- CRUD
- 6 Feature Beispiele
- Zusammenfassung

Create RUD

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

   ctx.Categories.Add(myCat);

   ctx.SaveChanges();
}
```

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.
                   Add(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);


     ctx.Categories.Find(myCat.Id)

     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


   // ObjectContext
   ctx.Attach(myCat);

   //DbContext
   ctx.Categories.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


   // ObjectContext
   ctx.Attach(myCat);
   //DbContext
   ctx.Categories.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 ...


   //ObjectContext
   ctx.DeleteObject(myCat);


   // DbContext
   ctx.Categories.Remove(myCat)


   ctx.SaveChanges();

}
```

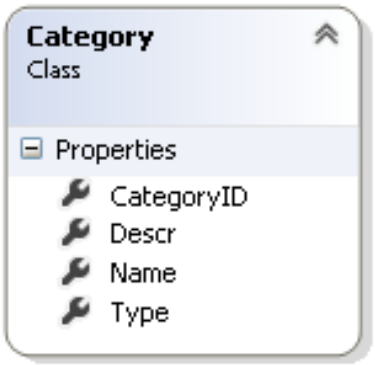
Agenda

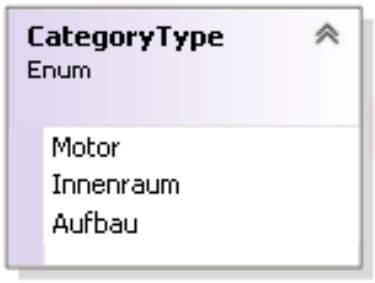
- (Architektur) Überblick
- Entitäten beschreiben
- CRUD
- EF 6 Feature Beispiele
- Zusammenfassung

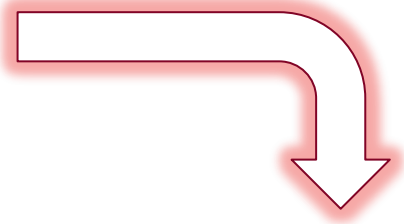
Enum Unterstützung (EF 5.0)

Enum Unterstützung (EF 5.0)

- Enums können in Entity und ComplexType Klassen genutzt werden
- Gespeichert werden diese als Byte, Int16, Int32, Int64, oder SByte.







Categories

Column Name	Data Type	Allow Nulls
CategoryID	bigint	☐
Type	int	☐
Name	nvarchar(MAX)	☑
Descr	nvarchar(MAX)	☑
		☐

Logging (EF 6.0)

Logging (EF 6.0)

- Ermöglicht (endlich!) die Ausgabe der Ausgeführten SQL Anweisungen
- Beispiel (einfach)

```
Ctx.Database.Log = Console.Write
```

- Was wird geloggt (SQL)
  - Queries, einschließlich LINQ Queries, eSQL queries und “reine” Queries von SqlQuery
  - Inserts, updates, and deletes durch SaveChanges()
  - Relationship loading queries durch lazy loading
  - Parameter
  - synchron vs. asynchron
  - Zeitstempel des Starts und Ausführungszeit
  - wurde Command erfolgreich, mit Fehler oder „gecancelled“ (bei Async) ausgeführt

Logging (EF 6.0)

- Beispiel (erweitert – über Code-based Configuration)

```
public class MyDbConfiguration : DbConfiguration {
   public MyDbConfiguration() {
      SetCommandLogger(
      (ctx, sink) => new DemoCommandLogger(ctx, sink));
   }
}
```

```
public class DemoCommandLogger : DbCommandLogger {
   public DemoCommandLogger(DbContext context,
                Action<string> sink): base(context, sink){ }

   public override void LogCommand(DbCommand cmd,
                     DbCommandInterceptionContext intCont) {
         Sink(string.Format(
             "Cmd'{0}' wird ausgeführt \n",
             cmd.CommandText.Replace(Environment.NewLine,
                                     "")));
     }
}
```

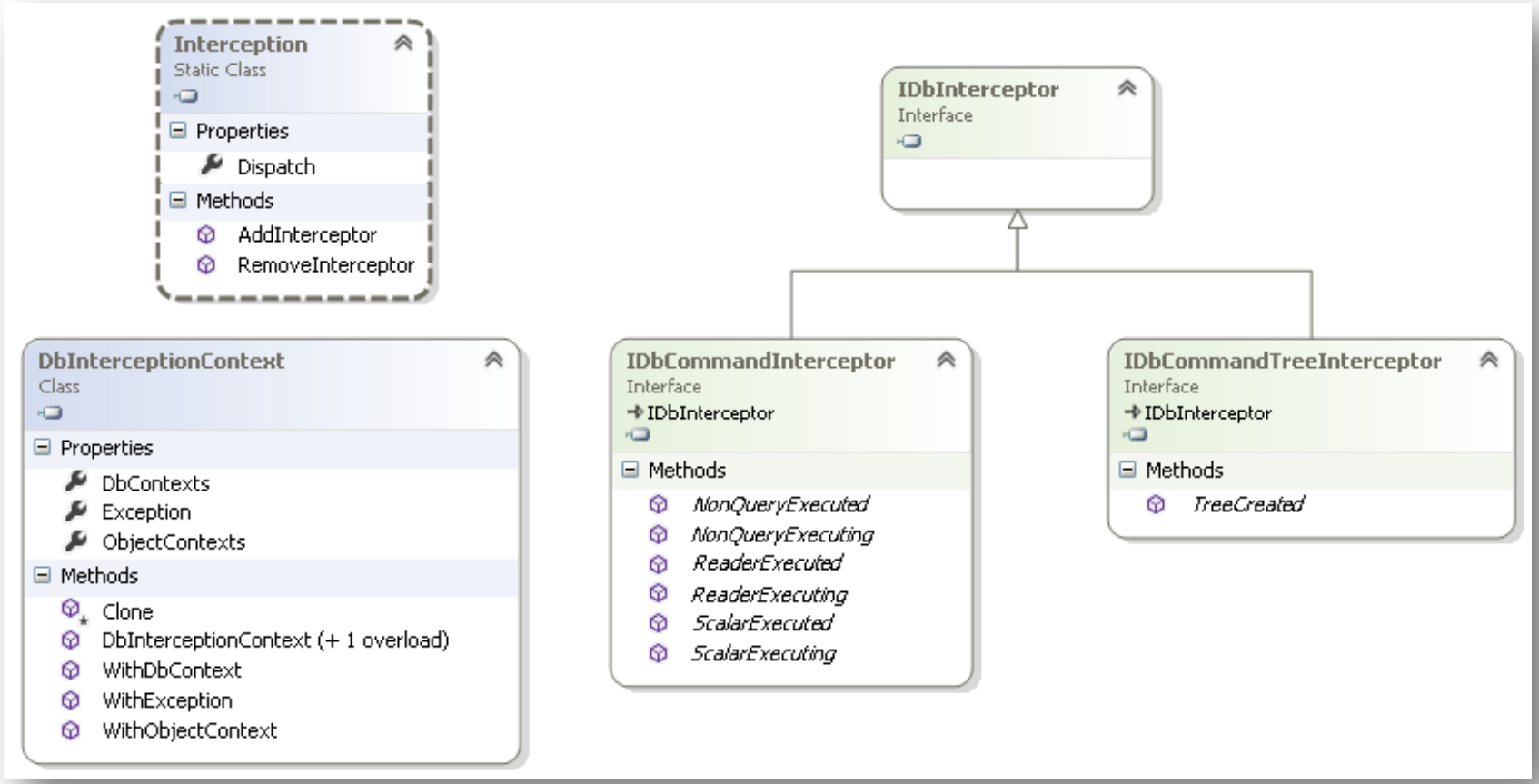
Interception (EF 6.0)

Interception (EF 6.0)

- Einfaches „Framework“ um Zugriffe auf die Datenbank abzufangen
- Vor Aufruf des Datenbank-Layers (XyzExecuting(…))
- Nach Aufruf des Datenbank-Layers (XyzExecuted(…))
- Interceptoren werden über die statische Klasse Interception registriert bzw. deregistriert
- Werden in der Reihenfolge ihrer Registrierung ausgeführt
- Gelten App-Domain weit

Interception (EF 6.0)

- Einfaches „Framework“



Interception (EF 6.0)

- Beispiel

```
using System.Data.Common;
using System.Data.Entity.Infrastructur.Interception;

public class DefaultInterceptor : IDbCommandInterceptor {
   public virtual void NonQueryExecuted(DbCommand command,
               DbCommandInterceptionContext<int> interceptionContext) {}

   public virtual void NonQueryExecuting(DbCommand command,
                DbCommandInterceptionContext<int> interceptionContext){}

   public virtual void ReaderExecuted(DbCommand command,
       DbCommandInterceptionContext<DbDataReader> interceptionContext){}

   public virtual void ReaderExecuting(DbCommand command,
       DbCommandInterceptionContext<DbDataReader> interceptionContext){}

   public virtual void ScalarExecuted(DbCommand command,
             DbCommandInterceptionContext<object> interceptionContext){}

   public virtual void ScalarExecuting(DbCommand command,
             DbCommandInterceptionContext<object> interceptionContext){}
}
```

Interception (EF 6.0)

- Beispiel

```
public class MeinInterceptor : DefaultInterceptor {
   public override void ReaderExecuted(DbCommand command,
       DbCommandInterceptionContext<DbDataReader> intCtx) {
     Console.Write("ReaderExecuted Command: {0}",
                   command.CommandText);
   }
}
```

```
public class Program {
   static void Main(string[] args) {
      using (DemoContext ctx = new DemoContext(con)) {

        //Beta1: Interception.AddInterceptor(new MeinInterceptor());
        DbInterception.Add(new MeinInterceptor());

        Category category = new Category() { Name = "TestCat"};
        ctx.Categories.Add(category);
        ctx.SaveChanges();
      }
   }
}
```

Async Unterstützung (EF 6.0)

Async Unterstützung (EF 6.0)

- Asynchrone Ausführung von EF Anweisungen
- Beispiel

```
static void Main(string[] args) {
      MachesAsync();
      Console.WriteLine("bitte <return>");
      Console.ReadLine();
}

static async void MachesAsync() {
   using (BugShopContext ctx = new BugShopContext()) {
      ctx.Database.Log = Console.Write;
      Category cat= new Category() { Name = "TestCategory"};

      ctx.Categories.Add(category);

      await ctx.SaveChangesAsync();
   }
}
```

Custom Code First Conventions

Custom Code First Conventions

- Eigene Code First Konventionen definieren

- Beispiel (einfach)

```
protected override void OnModelCreating(DbModelBuilder mbuilder)
{
   builder.Properties().Where(p => p.Name.EndsWith("Schluessel"))
                       .Configure(p => p.IsKey());
}
```

- Beispiel (erweitert)

```
public class DemoAttribute : Attribute {}


protected override void OnModelCreating(DbModelBuilder mbuilder)
{
   builder.Conventions.Add(new DemoAttributeConvention());
}
```

Custom Code First Conventions

- Beispiel (erweitert)

```
using System.Data.Entity.ModelConfiguration.Configuration;
using System.Data.Entity.ModelConfiguration.Configuration.Types;
using System.Data.Entity.ModelConfiguration.Conventions;

public class DemoAttributeConvention :

PropertyAttributeConfigurationConvention<DemoAttribute> {
    public override void Apply(PropertyInfo memberInfo,
                    /* Beta1: LightweightTypeConfiguration */
                                ConventionTypeConfiguration config,
                                DemoAttribute attribute) {

        var propertyConfig = config.Property(memberInfo);

        propertyConfig.HasColumnName(memberInfo.Name+"_Demo");
    }
}
```

Code First & Stored Procedures

Code First & Stored Procedures

- Unterstützung von Stored Procedures

- Beispiel (einfach)

```
class DemoContextMitStoredProcedures : BugShopContext {
   protected override void
               OnModelCreating(DbModelBuilder builder) {
     builder.Entity<Category>().MapToStoredProcedures();
   }
}
```

- Beispiel (erweitert)

```
class DemoContextMitStoredProcedures : BugShopContext {
   protected override void
               OnModelCreating(DbModelBuilder builder) {
     builder.Entity<Category>().MapToStoredProcedures(
               s => s.Update(u => u.HasName("mod_cat"))
                     .Delete(d => d.HasName("del_cat"))
            .Insert(i => i.HasName("ins_cat")));
 }
}
```

Erweiterte Tx-Handhabung

```
using (var ctx = new DemoContext()) {

  using (var tx = ctx.Database.BeginTransaction()) {

     try {
         ctx.DemoEntities.Add(demoEntity);
         ctx.SaveChanges();

         tx.Commit();
     }
     catch (Exception) {
        dbCtxTxn.Rollback();
     }
  }
}
```

- Database.BeginTransaction([IsolationLevel]) liefert einen Wrapper ContextDbTransaction zurück
- Erwartet eine offene Verbindung bzw. öffnet diese bei Bedarf (und schließt sie dann auch wieder)

Erweiterte Tx-Handhabung

```
using (var conn = new SqlConnection(connString)) {
   conn.Open();
   using (var sqlTxn = conn.BeginTransaction()) {
     try {
        // SQL ausführen…

        using (var ctx =
          new DemoContext(conn,
                    contextOwnsConnection = false)){

          ctx.Database.UseTransaction(sqlTxn);

       ctx.DemoEntities.Add(demoEntity);
          ctx.SaveChanges();
        }
        sqlTxn.Commit();
     }
     catch (Exception e) {
       sqlTxn.Rollback();
      }
    }
}
```

Code Based Configuration

- Entity Framework Anwendungen Konfiguration über
  - Konfigurationsdatei (app.config/web.config) oder
  - Programmcode (Code based Configuration)

- Basisklasse ist System.Data.Entity.Config.DbConfiguration

```
public class DemoConfiguration : DbConfiguration {
   public DemoConfiguration() {
      SetDefaultConnectionFactory(
         new LocalDbConnectionFactory("v11.0"));
   }
}
```

- Abgeleitete Klassen
  - "gelten" pro App-Domain

Code Based Configuration

- Abgeleitete Klassen (Fortsetzung)
  - Müssen einen Default Konstruktor besitzen
  - werden automatisch beim Laden der Anwendung entdeckt
    - Sofern sich die Klasse in der gleichen Assembly wie die EF Context Klasse befindet
  - „Andernfalls“
    - Explizite Angabe per Annotation

```
[DbConfigurationType(typeof(DemoDbConfiguration))]
public class DemoContext : DbContext {
  …
}
```

Code Based Configuration

- Abgeleitete Klassen (Fortsetzung 2)
  - „Andernfalls“ (Fortsetzung)
    - Explizite Angabe in Konfiguration

```
<entityFramework
   codeConfigurationType="EF6.DemoDbConfiguration,
                          Ef6Demo">
    ... weitere Konfiguration ...
</entityFramework>
```

Dependency Resolution

- Ab EF 6.0 werden „wesentliche“ Dienste über einen Service Locator innerhalb des Frameworks identifiziert und genutzt
- Einsatz des Chain of Responsibilty Pattern
- IDbDepenendencyResolver
  - object GetService(Type type, object key)

```
public class DemoConfiguration : DbConfiguration {
  public DemoConfiguration() {
     var confactory = new MyConnectionFactory();
     var resolver = new
       SingletonDependencyResolver<IDbConnectionFactory>
       (confactory)

     AddDependencyResolver(resolver);
    }
}
```

Dependency Resolution

- Folgende Services sind definiert
  - System.Data.Entity.IDatabaseInitializer<TContext>
  - System.Data.Entity.Migrations.Sql.MigrationSqlGenerator
  - System.Data.Entity.Core.Common.DbProviderServices
  - System.Data.Entity.Infrastructure.IDbConnectionFactory
  - System.Data.Entity.Infrastructure.IManifestTokenService
  - System.Data.Entity.Infrastructure.IDbProviderFactoryService
  - System.Data.Entity.Infrastructure.IDbModelCacheKeyFactory
  - System.Data.Entity.Spatial.DbSpatialServices
  - System.Data.Entity.Infrastructure.IExecutionStrategy
  - System.Data.Entity.Migrations.History.HistoryContextFactory
  - System.Data.Common.DbProviderFactory
  - System.Data.Entity.Infrastructure.IProviderInvariantName
  - System.Data.Entity.Core.Mapping.ViewGeneration.IViewAssemblyCache
  - System.Data.Entity.Infrastructure.Pluralization.IPluralizationService

Agenda

- (Architektur) Überblick
- Entitäten beschreiben
- CRUD
- EF 6 Feature Beispiele
- Zusammenfassung

Zusammenfassung

- Entity Framework und NHibernate sind im Kern ähnlich
- Entity Framework 6 bringt wichtige / notwendige Neuerungen mit.
- Unterschiede
  - Handhabung von Beziehungen
  - Datenbank Support
  - CASCADE Funktionalität
  - Unterstützung von verschiedenen Collection Typen

Zusammenfassung

- Was fehlt mir:
  - Second level caching
  - Batch API

- ABER:
  Viele Aspekte wurden nicht beleuchtet: z. B. Einbetten von Funktionen, Vererbung, Performance, T4, Self-tracking, Migrations...

Fragen

Viel Spaß mit dem Entity Framework

thomas.haug@mathema.de
@tshaug
www.sharpmetrics.net

Vielen Dank!

Thomas Haug
MATHEMA Software GmbH