# En databas, en tabell, lagrade procedurer och Entity Framework



#### Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen ASP.NET MVC vid Linnéuniversitetet.

#### Du får använda detta verk så här:

Allt innehåll i detta verk av Mats Loock, förutom Linnéuniversitetets logotyp och symbol, är licensierad under:



Creative Commons Erkännande-IckeKommersiell-DelaLika 2.5 Sverige licens. http://creativecommons.org/licenses/by-nc-sa/2.5/se/

#### Det betyder att du i icke-kommersiella syften får:

- kopiera hela eller delar av innehållet
- sprida hela eller delar av innehållet
- visa hela eller delar av innehållet offentligt och digitalt
- konvertera innehållet till annat format
- du får även göra om innehållet

Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp och symbol i din nya version!

Vid all användning måste du ange källan: "Linnéuniversitetet – ASP.NET MVC" och en länk till <a href="https://coursepress.lnu.se/kurs/aspnet-mvc">https://coursepress.lnu.se/kurs/aspnet-mvc</a> och till Creative Common-licensen här ovan.

#### Lagra persistent data i en databas

- ✓ Vad måste göras för att lagra Birthday-data i en databas istället för en XML-fil?
  - Skapa databas med tabell innehållande fält för namn och födelsedatum.
  - Använda *Entity Framework* för kommunikation med databasen.
  - Modifiera klassen Birthday så den fungerar med Entity Framework.
  - Skapa ett centrallager (*repository*) som använder *Entity Framework* istället för *LINQ to XML*.

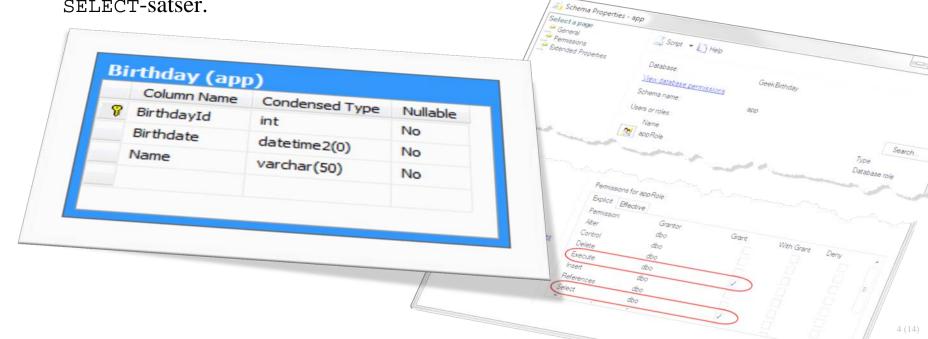


Linnéuniversitetet Kalmar ASP.NET MVC (1DV409)

#### Databasen

- ✓ Databasen GeekBirthday innehåller endast tabellen app. Birthday med fält för primärnyckel, namn och födelsedatum.
- ✓ Användaren appuser tillhör rollen appRole vars rättigheter bestäms av schemat app.

✓ Schemat app ger appUser rättigheter att exekvera lagrade procedurer och SELECT-satser.



Linnéuniversitetet Kalmar ASP.NET MVC (1DV409)

#### De lagrade procedurerna

✓ Lagrade procedurer finns för att lägga till, uppdatera och ta bort poster ur tabellen app.Birthday.

Den lagrade proceduren app. InsertBirthday returnerar den nya postens primärnyckel med en SELECT-sats då *Entity Framework* inte, på ett enkelt sätt, kan hantera OUTPUT-parametrar.

```
CREATE PROCEDURE [app].[UpdateBirthday]
                                                   CREATE PROCEDURE [app].[DeleteBirthday]
                                                       @BirthdayId int
    @BirthdayId int,
    @Name varchar(50),
                                                  BEGIN
    @Birthdate datetime2(0)
                                                      SET NOCOUNT ON:
                                                     IF EXISTS (SELECT BirthdayId FROM app. Birthday WHERE (Birthday
    IF EXISTS (SELECT BirthdayId FROM app.Bir
                                                     BEGIN
BEGIN
                                                         DELETE FROM app.Birthday
                                                                           yId = @BirthdayId)
                  app.Birthday
                             CREATE PROCEDURE [app].[InsertBirthday]
    BEGIN
        UPDATE
                                 @Name varchar(50),
                   (Birth
                                 @Birthdate datetime2(0)
        WHERE
                             AS
    END
                             BEGIN
                                SET NOCOUNT ON:
END
                                INSERT INTO app.Birthday
                                                    (Name, Birthdate)
                                            (@Name, @Birthdate
                                VALUES
                                SELECT SCOPE_IDENTITY() AS NewBirthdayId
```

## Entity Framework

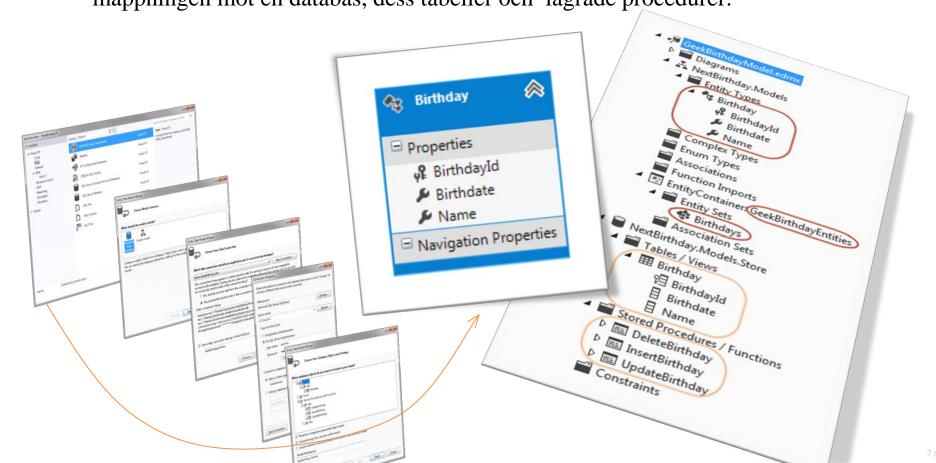
- ASP.NET MVC har stöd för flera olika teknologier för hantering av data i databaser.
  - Entity Framework, LINQ to SQL, NHibernate, LLBLGen Pro, SubSonic, WilsonORM eller "råa" ADO.NET-klasser som SqlDataReader och DataSet.
- ✓ Entity Framework är ett ORM (Object Relational Mapper) vilket gör det enkelt att koppla tabeller i en databas till C#-klasser.
  - Tabeller motsvaras av C#-klasser.
  - Fält i en tabell motsvaras av egenskaper i en C#-klass.
  - En rad i en tabell motsvaras av en instans av C#-klassen.



Linneuniversitetet Kalmar ASP.NET MVC (1DV409)

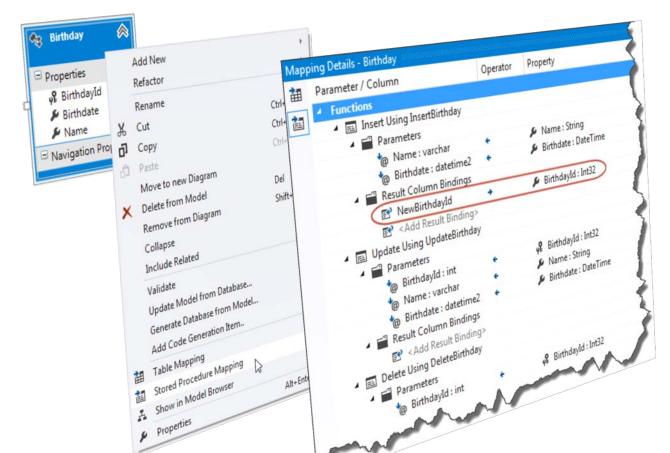
#### Lägga till "Entity Framework"-klasser

Med *Entity Data Model Wizard* är det enkelt att generera de klasser som behövs för mappningen mot en databas, dess tabeller och lagrade procedurer.



#### Koppla lagrade procedurer till klassen Birthday

För att *Entity Framework* ska använda de lagrade procedurerna för att lägga till, uppdatera och ta bort poster måste de kopplas till entitetsklassen Birthday.



Linneuniversitetet Kalmar

# Modifiering av klassen Birthday

Då *Entity Data Model Wizard* genererat en mall för en partiell klass med namnet Birthday måste den sedan tidigare existerande klassen Birthday modifieras. Se även till att den nya partiella klassen Birthday tillhör samma namnrymd som den ursprungliga klassen Birthday.

Klassen Birthday har gjorts partiell och egenskaperna Birthdate och Name, inklusive attribut, har flyttats till en metadataklass som Birthday associeras till 6.200 B LD &

med hjälp av attributet MetadataType.

```
阿 NextBirthday.EF
  [MetadataType(typeof(Birthday_Metadata))]
 public partial class Birthday
     [ScaffoldColumn(false)]
    public int Age...
                                                                                                                               + C* RouteConfig.cs
                                                                                                                                 WebApiConfig.cs
   [ScaffoldColumn(false)]
   public int DaysUntilNextBirthday...
                                                                                                                            Infrastructure
  [ScaffoldColumn(false)]
                                                                                                                             DataModels
 public DateTime NextBirthdayDate...
                                                                                                                             GeekBirthdayModel.edmx
                                                                                                                               GeekBirthdayModel.Context.tt
public class Birthday_Metadata
                                                                                                                                  D GeekBirthdayModel.Context.cs
                                                                                                                               GeekBirthdayModel,Designer.cs
                                                                                                                                GeekBirthdayModel.edmx.diagram
    [Required(ErrorMessage = "Födelsedatum måste anges.")]
   [Required(trrormessage = rodeisedatum maste anges.)]
[BeforeToday(ErrorMessage = "Ett födelsedatum tidigare än dagens datum måste anges."
                                                                                                                                 D Birthday.cs
                                                                                                                      P + C≈ Birthday.cs
                                                                                                                                 D GeekBirthdayModel.cs
  [DisplayName("Födelsedatum")]
                                                                                                                Properties
  public DateTime Birthdate { get; set; }
                                                                                                                GeekBirthdayModel.tt File Properties
 [Required(ErrorMessage = "Namn måste anges.")]
                                                                                                              E Advanced
[DisplayName("Namn")]
                                                                                                                Browse to URL
public string Name { get; set; }
                                                                                                               Build Action
                                                                                                               Copy to Output Directory
                                                                                                               Custom Tool
                                                                                                             Custom Tool Namespace NextBirthday. Models
                                                                                                                                   TextTemplatingFileGenerator
                                                                                                             File Name
                                                                                                            Full Path
                                                                                                                                   GeekBirthdayModel th
                                                                                                                                  C:\M.D
```

# "Repository"-klass som använder Entity Framework

- Klassen EFRepository implementerar de tre metoderna GetBirthdays, InsertBirthday och Save, d.v.s. exakt samma publika metoder som klassen XmlRepository.
- Klassen implementerar även ett interfacet, IDisposal. Vilket gör det möjligt för användaren av klassen att återlämna de resurser som tagits i anspråk vid kommunikation med den underliggande databasen.

```
public class EFRepository : IRepository
    private GeekBirthdayEntities _entities = new GeekBirthdayEntities();
    public IEnumerable<Birthday> GetBirthdays()
       return _entities.Birthdays.ToList();
   public void InsertBirthday(Birthday birthday)
       _entities.Birthdays.Add(birthday);
   public void Save()
       _entities.SaveChanges();
  #region IDisposable
  private bool disposed = false;
 protected virtual void Dispose(bool disposing)
     if (!this.disposed)
         if (disposing)
             entities.Dispose();
    this.disposed = true;
public void Dispose()
    Dispose(true);
    GC.SuppressFinalize(this);
#endregion
```

# Modifiering av klassen HomeController

✓ I och med att klassen HomeController använder ett centrallager (*repository*) är det inte mycket som behöver modifieras!

```
public class BirthdayController : Controller
               private EFRepository _repository = new EFRepository();
              protected override void Dispose(bool disposing)
                base.Dispose(disposing);
           // GET: /Birthday/
          public ActionResult Index()
             return View("Index", _repository.GetBirthdays());
       // GET: /Birthday/Create
      public ActionResult Create()
         return View("Create");
  // POST: /Birthday/Create
 [HttpPost]
[ValidateAntiForgervToken]
public ActionParile
```

Linneuniversitetet Kalmar

## Ett interface är förutsättningen för DI

- Klassen HomeController kan kopplas bort från det hårdkodade beroendet av en specifik "repository"-klass med hjälp av ett designmönster som kallas dependency injection (DI). Detta gör det möjligt i en förlängning att skriva tester för controllerklassen.
- Metoder som är gemensamma för klasserna XmlRepository och EFRepository extraheras till ett interface, IRepository, som klasserna sedan implementerar.

```
public class BirthdayController : Controller
                                                            public interface IRepository : IDisposable
   private EFRepository _repository = new EFRepository()
                                                                IEnumerable<Birthday> GetBirthdays();
                                                                void InsertBirthday(Birthday birthday);
   protected override void p: se(bool disposing)
                                                                void Save();
                                                public class EFRepository : IRepository
         HomeController \ddot{a}r starkt beroende av
        EFRepository. Detta kan åtgärdas med
        dependency injection som förutsätter att
        "repository"-klasser implementerar ett interface
       som deklarera vilka medlemmar som ska vara
                                                          public class XmlRepository : IRepository
    return View("Index", _repository.GetBirthdays());
```

# Oberoende med dependency injection

- Det privata fältet \_repository är av typen IRepository och kan referera till vilket objekt som helst som är instansierat från en klass som implementerar interfacet IRepository.
- ✓ ASP.NET MVC använder standardkonstruktorn för att instansiera objekt av typen BirthdayController varför ett EFRepository-objekt då kommer att utgöra centrallagret (repository).
- Då tester skrivs, eller om t.ex. Ninject används, kan den andra konstruktorn användas som gör det möjligt att skicka med vilket objekt som helst så länge det är instansierat från en klass som implementerar interfacet IRepository.

```
public class BirthdayController : Controller
    private IRepository _repository;
    public BirthdayController()
        : this(new EFRepository())
       // Empty!
   public BirthdayController(IRepository repository)
       _repository = repository;
  protected override void Dispose(bool disposing)...
  // GET: /Birthday/
 public ActionResult Index()...
 // GET: /Birthday/Create
public ActionResult Create()...
// POST: /Birthday/Create
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(Birthday birthday)...
```

#### **Mer information**

- Entity Framework
  - http://msdn.microsoft.com/en-us/data/aa937723
- ✓ Getting Started with EF 5 using MVC 4
  - http://www.asp.net/mvc/tutorials/getting-started-with-ef-5-using-mvc-4