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:

Det betyder att du i icke-kommersiella syften får:

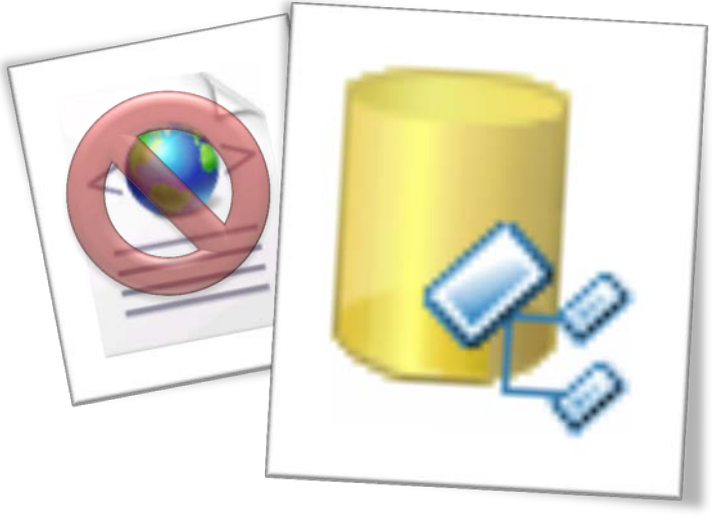
- kopiera hela eller delar av innehållet
- sprida hela eller delar av innehållet
- visa hela eller delar av innehållet offentligt och digitalt
- konvertera innehållet till annat format
- du får även göra om innehållet

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 https://coursepress.lnu.se/kurs/aspnet-mvc och till Creative Common-licensen här ovan.

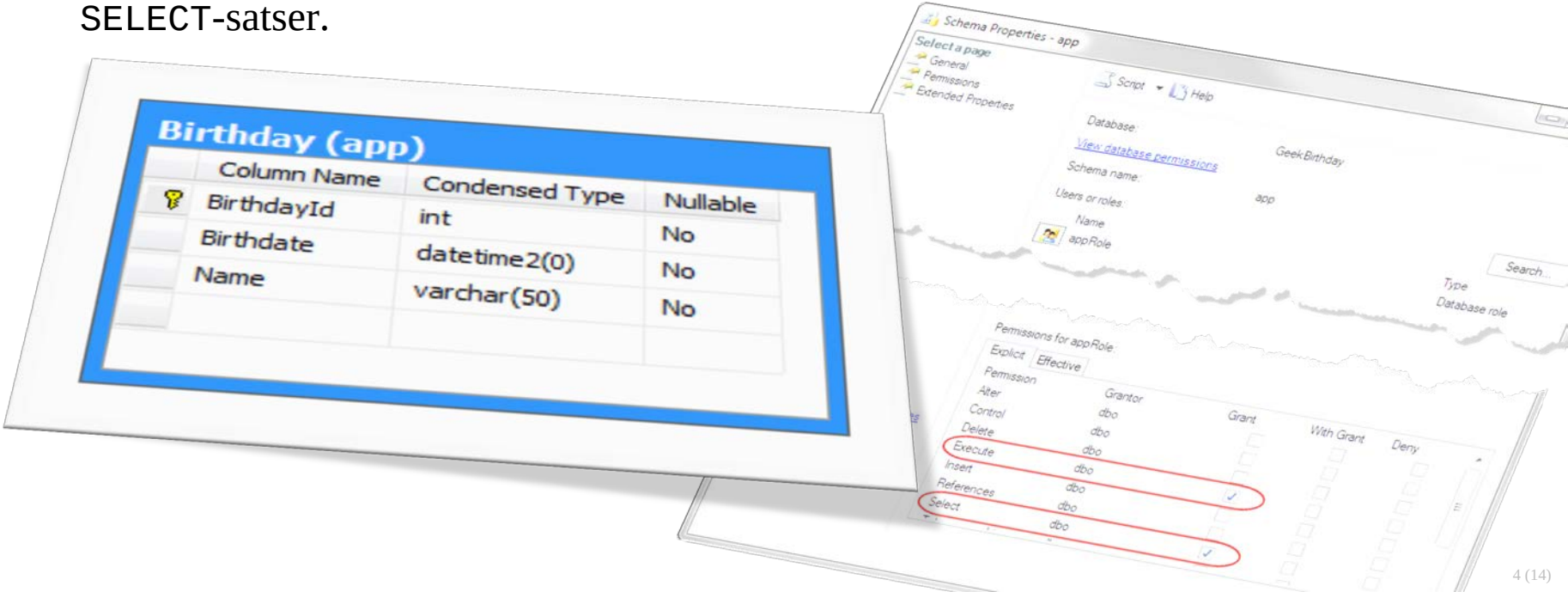
Lagra persistent data i en databas

- ✓ 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*.



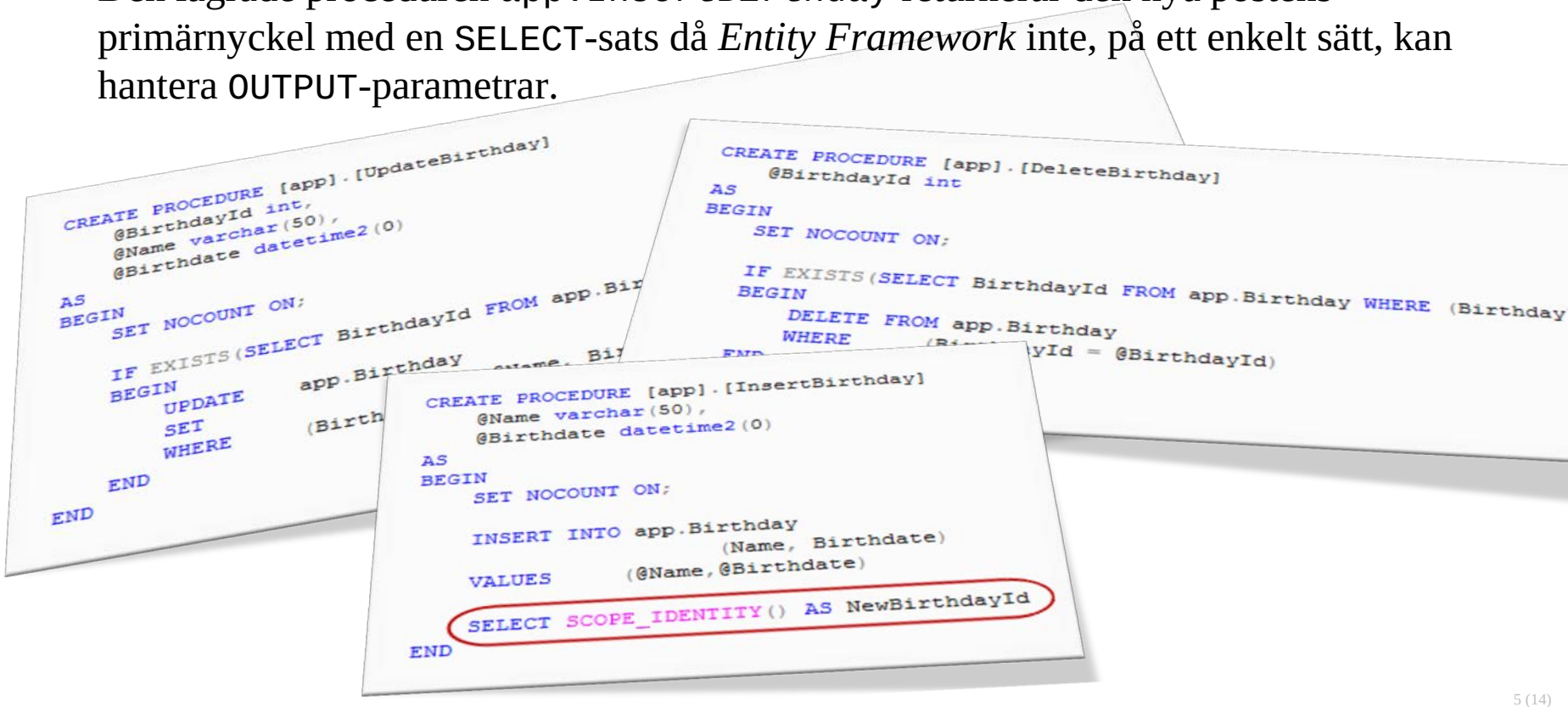
Databasen

- ✓ 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.



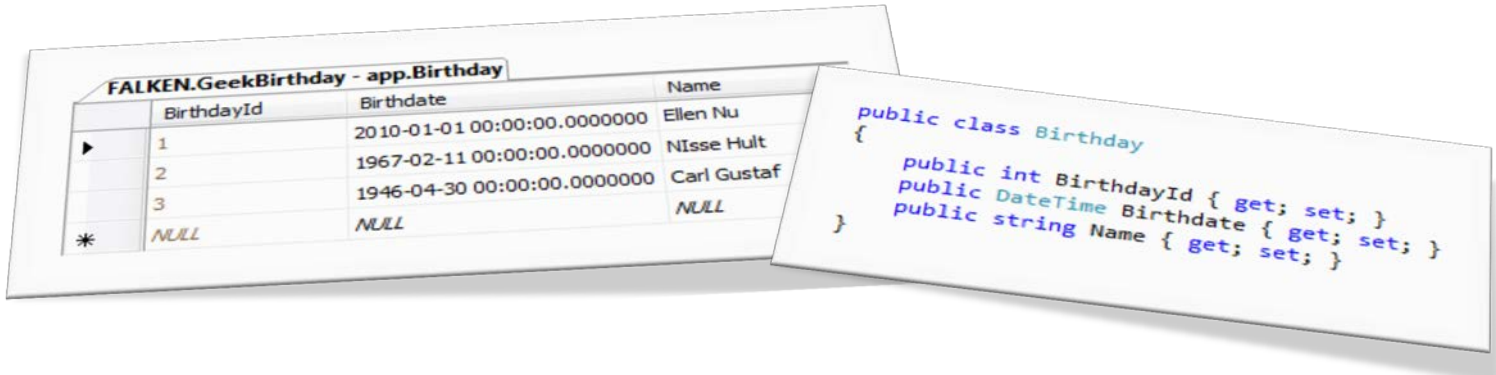
De lagrade procedurerna

- 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.



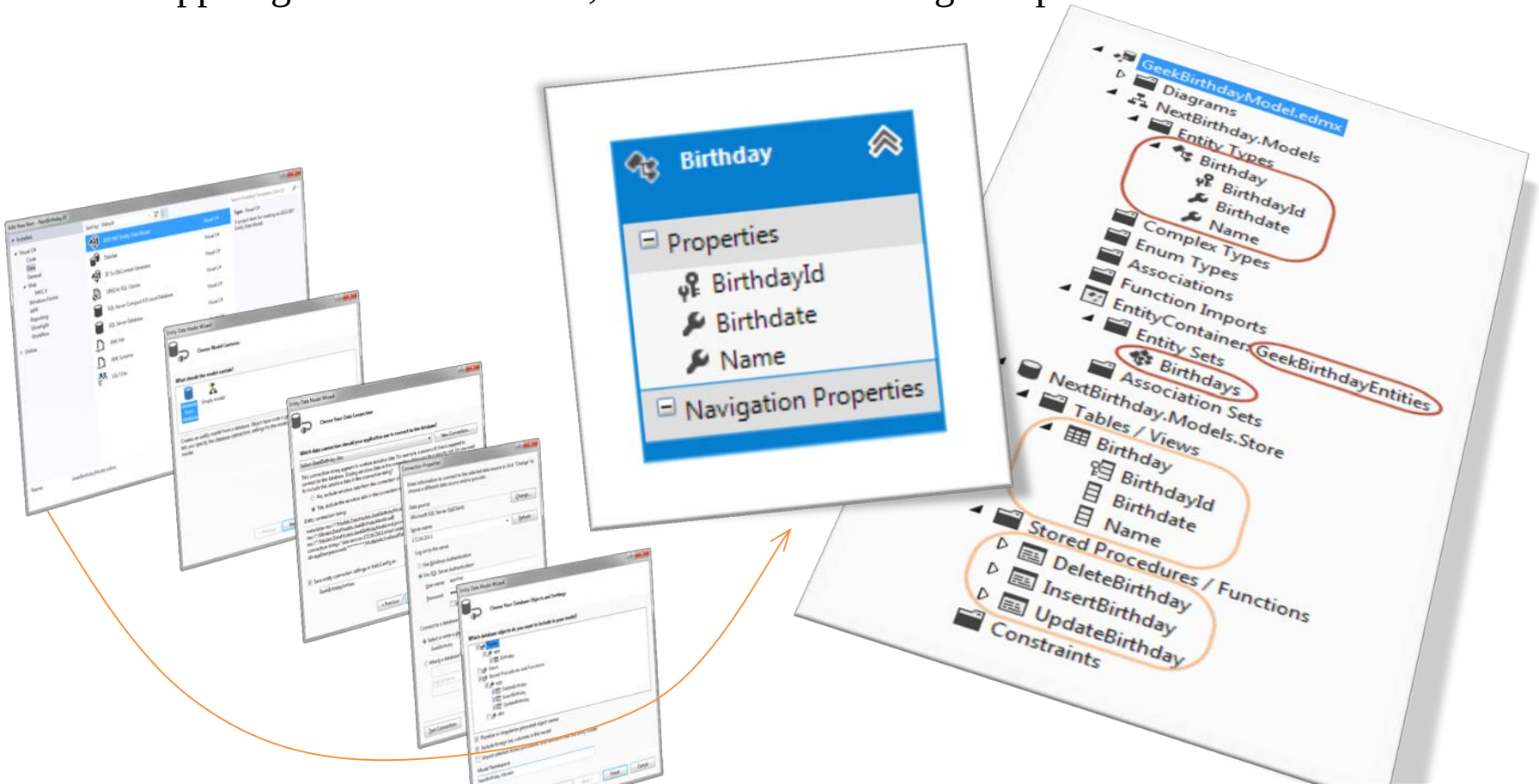
Entity Framework

- 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.



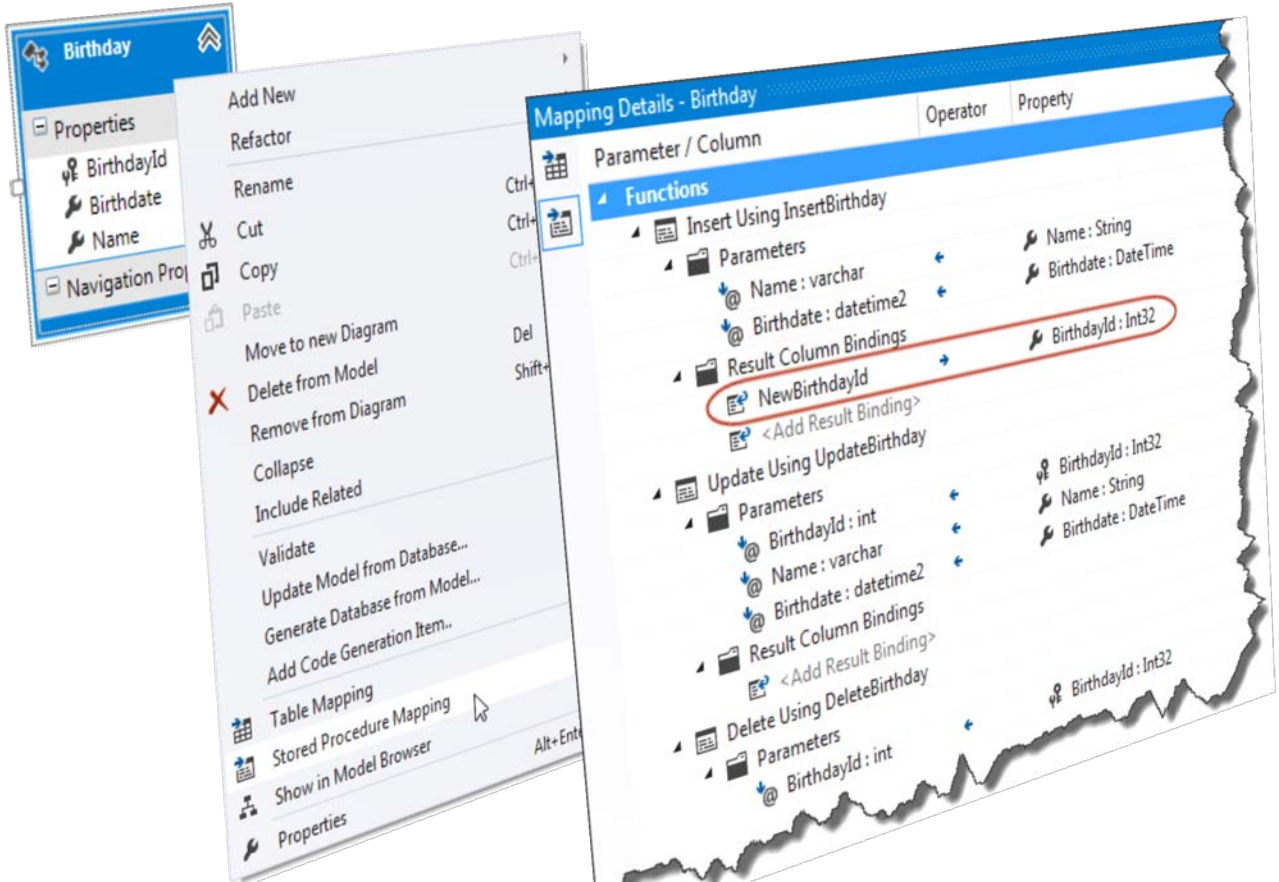
Lägga till ”Entity Framework”-klasser

- 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`

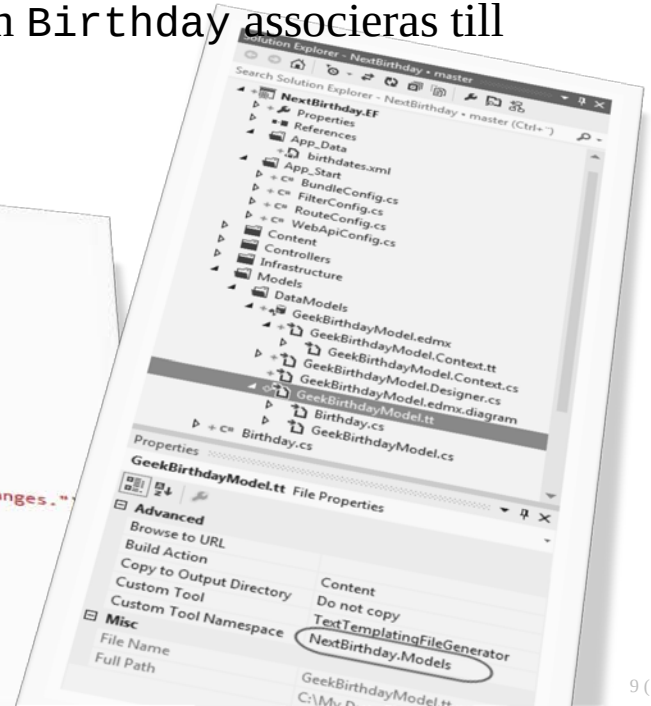
- För att *Entity Framework* ska använda de lagrade procedurerna för att lägga till, uppdatera och ta bort poster måste de kopplas till entitetsklassen `Birthday`.



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 med hjälp av attributet `MetadataType`.

```
[MetadataType(typeof(Birthday_Metadata))]
public partial class Birthday
{
    [ScaffoldColumn(false)]
    public int Age { ... }

    [ScaffoldColumn(false)]
    public int DaysUntilNextBirthday { ... }

    [ScaffoldColumn(false)]
    public DateTime NextBirthdayDate { ... }

    public class Birthday_Metadata
    {
        [Required(ErrorMessage = "Födelsedatum måste anges.")]
        [BeforeToday(ErrorMessage = "Ett födelsedatum tidigare än dagens datum måste anges.")]
        [DataType(DataType.Date)]
        [DisplayName("Födelsedatum")]
        public DateTime Birthdate { get; set; }

        [Required(ErrorMessage = "Namn måste anges.")]
        [DisplayName("Namn")]
        public string Name { get; set; }
    }
}
```



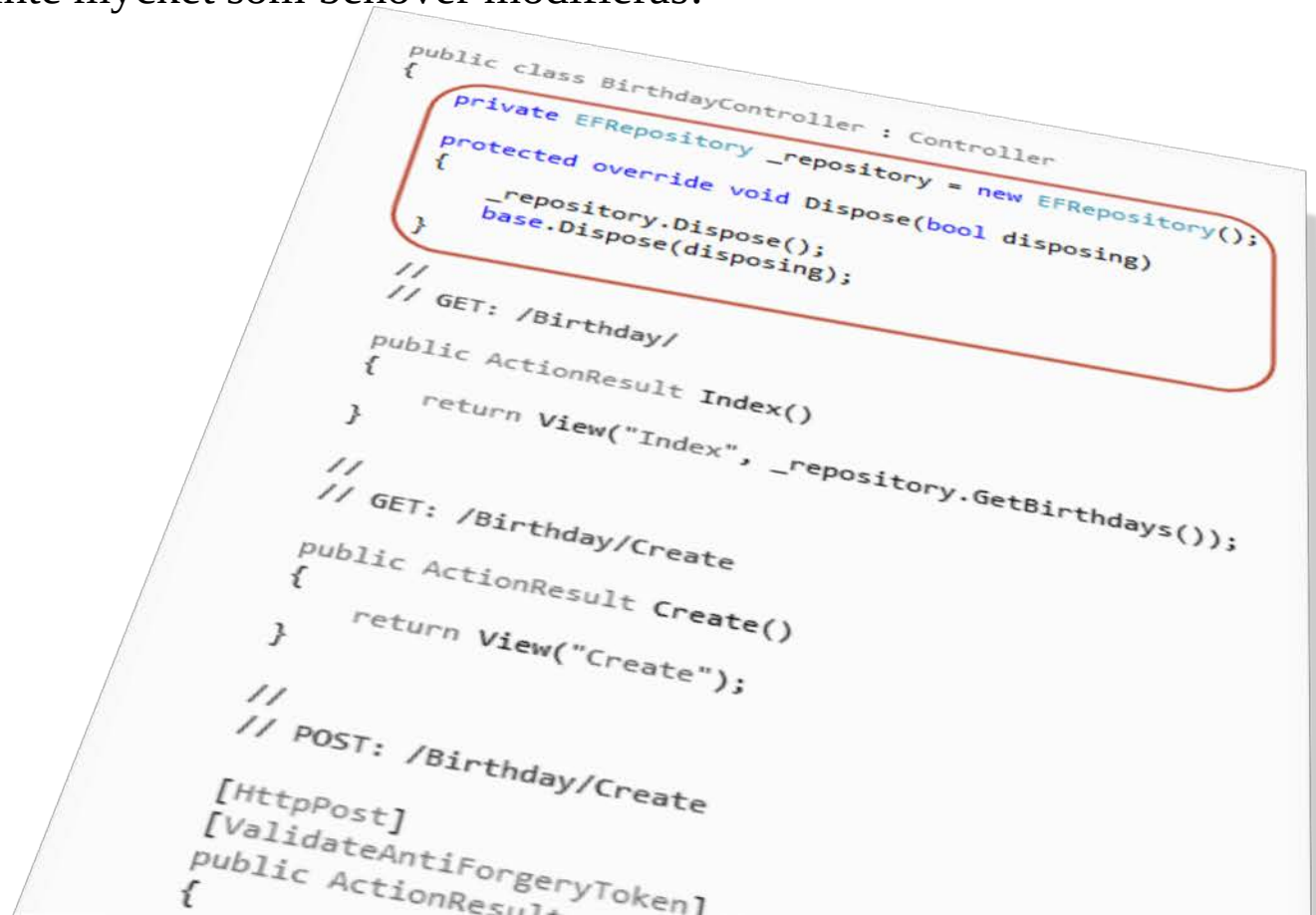
”*Repository*”-klass som använder *Entity Framework*

- 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`

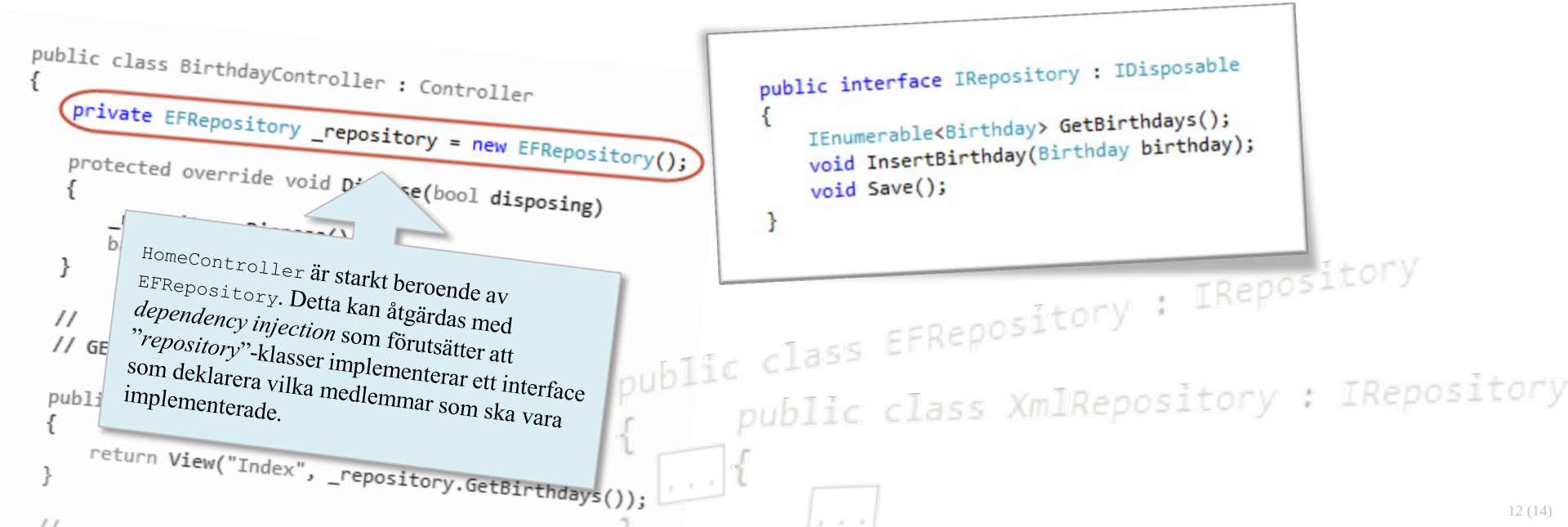
- I och med att klassen `HomeController` använder ett centrallager (*repository*) är det inte mycket som behöver modifieras!



```
public class BirthdayController : Controller
{
    private EFRepository _repository = new EFRepository();

    protected override void Dispose(bool disposing)
    {
        _repository.Dispose();
        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]
    [ValidateAntiForgeryToken]
    public ActionResul
```

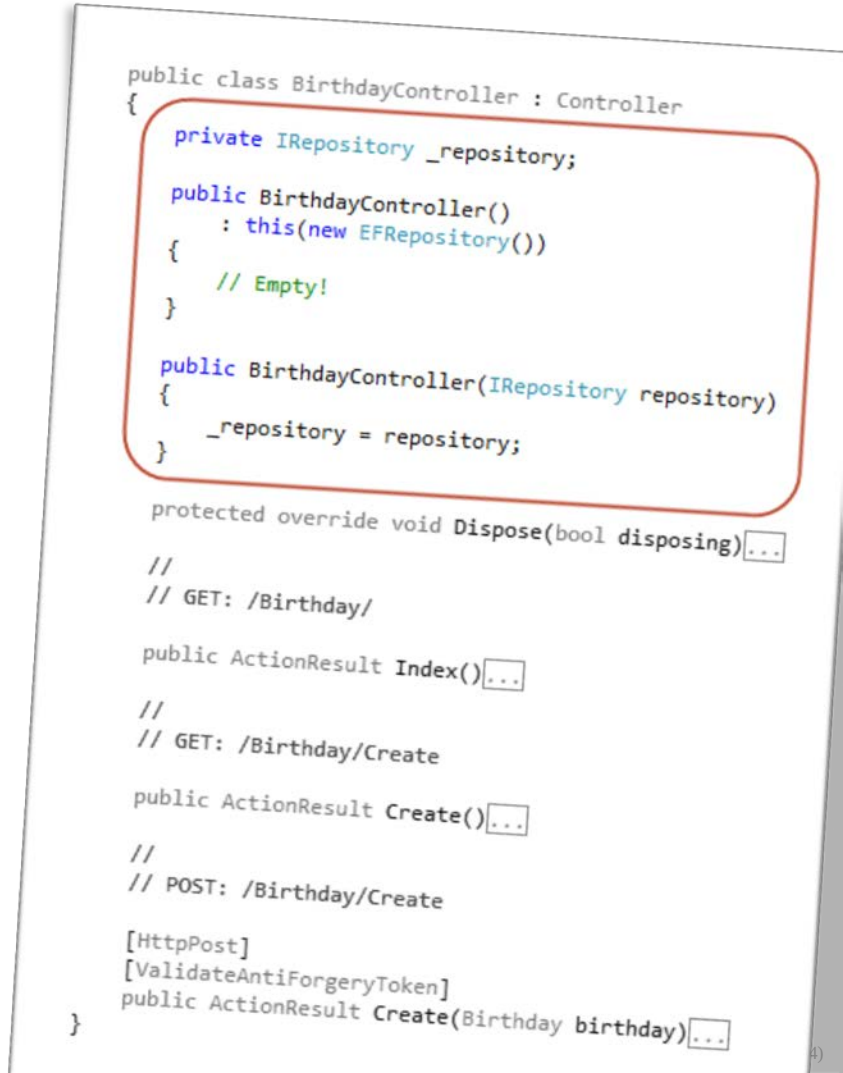
Ett interface är förutsättningen för DI

- 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.



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