Inheritance Strategy in Entity Framework 6

We have seen in the Code-First Conventions section that EF creates database tables for each concrete domain class. However, you can design your domain classes using inheritance. Object-oriented techniques include "has a" and "is a" relationships, whereas SQL-based relational model has only a "has a" relationship between tables. SQL database management systems don't support type inheritance. So, how would you map object-oriented domain classes with the relational database?

Below are three different approaches to represent an inheritance hierarchy in Code-First:

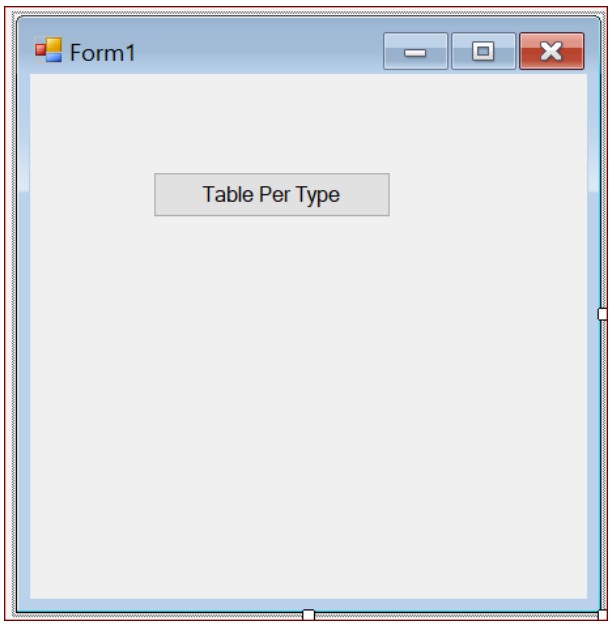
- **Table per Hierarchy (TPH):** This approach suggests one table for the entire class inheritance hierarchy. The table includes a discriminator column which distinguishes between inheritance classes. This is a **default** inheritance mapping strategy in Entity Framework.
- **Table per Type (TPT):** This approach suggests a separate table for each domain class.
- **Table per Concrete Class (TPC):** This approach suggests one table for one concrete class, but not for the abstract class. So, if you inherit the abstract class in multiple concrete classes, then the properties of the abstract class will be part of each table of the concrete class.

We are not going into detail here. Visit the following reference links for more detailed information:

1. Inheritance with EF Code First: Table per Hierarchy (TPH)
2. Inheritance with EF Code First: Table per Type (TPT)
3. Inheritance with EF Code First: Table per Concrete class (TPC)

Type 3

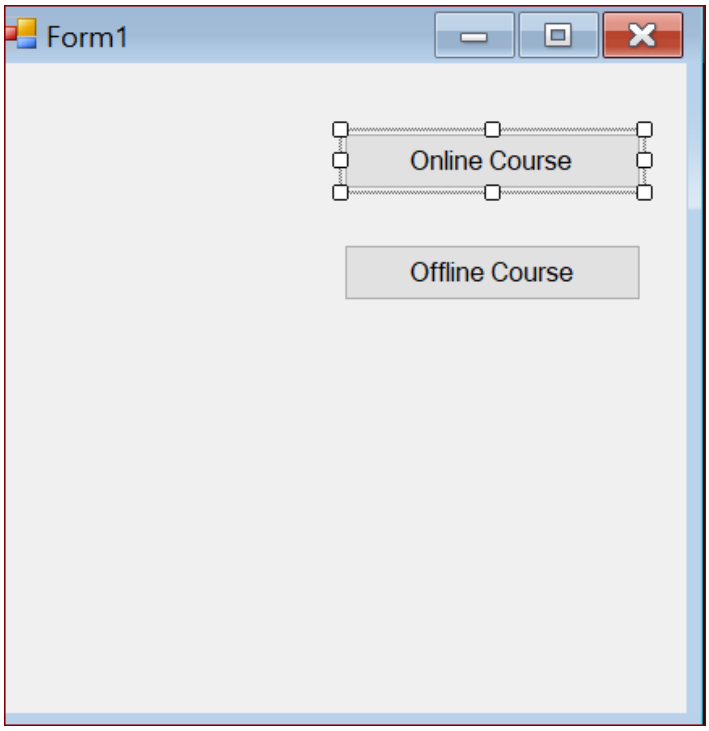
```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Type3
{
    public class Course
    {
        [Key]
        public int CourseId { get; set; }
        public string Name { get; set; }
        public string Details { get; set; }
        public string Trainer { get; set; }
    }
    [Table("OnlineCourse")]
    public class OnlineCourse : Course
    {
        public string URL { get; set; }
    }
    [Table("OfflineCourse")]
    public class OfflineCourse : Course
    {
        public string Address { get; set; }
    }
    public class TPTContext:DbContext
    {
        public DbSet<Course> Courses { get; set; }
        public TPTContext() : base("name=TPT")
        {
            Database.SetInitializer<TPTContext>(new
DropCreateDatabaseIfModelChanges<TPTContext>());
        }
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Course>().ToTable("Course");
            modelBuilder.Entity<OnlineCourse>().ToTable("OnlineCourse");
            modelBuilder.Entity<OfflineCourse>().ToTable("OfflineCourse");
        }
    }
}
```




```
private void button1_Click(object sender, EventArgs e)
        {
            using (var context = new TPTContext())
            {
                OnlineCourse oc = new OnlineCourse();
                oc.Details = "New course";
                oc.Name = "New name";
                oc.Trainer = "New trainer";
                oc.URL = "New URL";

                OfflineCourse offc = new OfflineCourse();
                offc.Details = "New course";
                offc.Name = "New name";
                offc.Trainer = "New trainer";
                offc.Address = "New Address";

                context.Courses.Add(offc);
                context.Courses.Add(oc);

                context.SaveChanges();
                MessageBox.Show("Saved");
            }
        }
```

Type 2

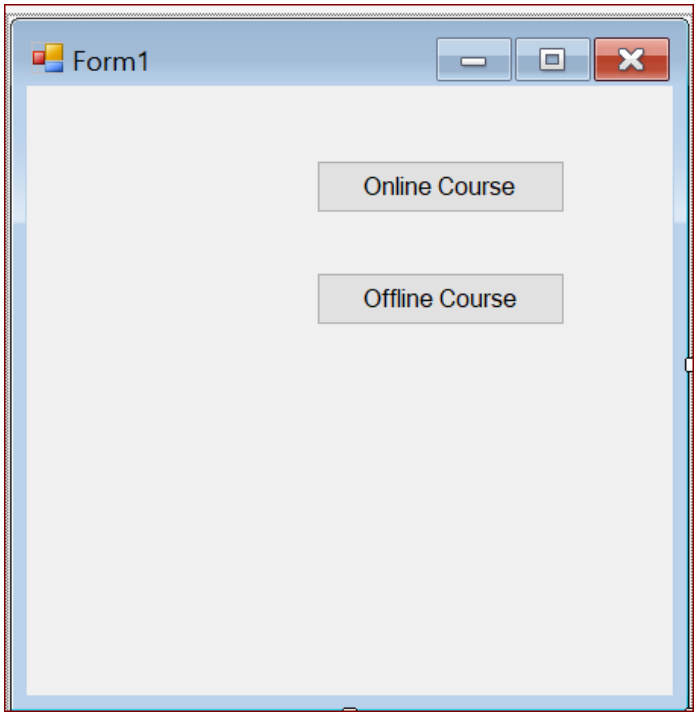
```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;

namespace Type2_TPC
{
    public abstract class Course //abstract class
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public string Name { get; set; }
        public string Details { get; set; }
        public string Trainer { get; set; }
    }
    public class OnlineCourse : Course //concrete class
    {
        public string URL { get; set; }
    }
    public class OfflineCourse : Course //concrete class
    {
        public string Address { get; set; }
    }
    public class TPCContext:DbContext
    {
        public DbSet<OfflineCourse> OfflineCourses { get; set; }
        public DbSet<OnlineCourse> OnlineCourses { get; set; }
        public TPCContext()
            : base("name=TPC")
        {
            Database.SetInitializer<TPCContext>(new
DropCreateDatabaseIfModelChanges<TPCContext>());
        }
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<OnlineCourse>().Map(m =>
            {
                m.MapInheritedProperties();
                m.ToTable("OnlineCourse ");
            });
            modelBuilder.Entity<OfflineCourse>().Map(m =>
            {
                m.MapInheritedProperties();
                m.ToTable("OfflineCourse ");
            });


        }
    }
}
```




```
private void button1_Click(object sender, EventArgs e)
        {
            using (var context = new TPCContext())
            {
                OnlineCourse oc = new OnlineCourse();
                oc.Details = "New course";
                oc.Name= "New name";
                oc.Trainer = "New trainer";
                oc.URL= "New URL";


                context.OnlineCourses.Add(oc);

                context.SaveChanges();
                MessageBox.Show("Saved");
            }
        }

        private void btnOfflineCourse_Click(object sender, EventArgs e)
        {
            using (var context = new TPCContext())
            {
                OfflineCourse offc = new OfflineCourse();
                offc.Details = "New course";
                offc.Name = "New name";
                offc.Trainer = "New trainer";
                offc.Address = "New Address";
                context.OfflineCourses.Add(offc);

                context.SaveChanges();
                MessageBox.Show("Saved");
            }
        }
```

Type 1

```
using System.ComponentModel.DataAnnotations;
using System.Data.Entity;

namespace Type1TPH
{
    public class Course
    {
        [Key]
        public int CourseId { get; set; }
        public string Name { get; set; }
        public string Details { get; set; }
        public string Trainer { get; set; }
    }

    public class OnlineCourse : Course
    {
        public string URL { get; set; }
    }

    public class OfflineCourse : Course
    {
        public string Address { get; set; }
    }
    public class TPHContext : DbContext
    {
        public DbSet<Course> Courses { get; set; }
        public TPHContext() : base("name=TPH")
        {
            Database.SetInitializer<TPHContext>(new
DropCreateDatabaseIfModelChanges<TPHContext>());
        }
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Course>()
            .Map<OnlineCourse>(m => m.Requires("Type").HasValue("OnlineCourse"))
            .Map<OfflineCourse>(m => m.Requires("Type").HasValue("OfflineCourse"));
        }
    }
}
```




```
private void button1_Click(object sender, EventArgs e)
        {
            TPHContext db = new TPHContext();
            OnlineCourse oc = new OnlineCourse();
            oc.Details = "Details";
            oc.Name = "new name";
            oc.Trainer = "new trainer";
            oc.URL = "new URL";
            db.Courses.Add(oc);
            db.SaveChanges();
            MessageBox.Show("Saved");
        }

        private void btnOfflineCourse_Click(object sender, EventArgs e)
        {
            TPHContext db = new TPHContext();
            OfflineCourse oc = new OfflineCourse();
            oc.Details = "Details";
            oc.Name = "new name";
            oc.Trainer = "new trainer";
            oc.Address = "02 An Hai";
            db.Courses.Add(oc);
            db.SaveChanges();
            MessageBox.Show("Saved");
        }
```