Creational Patterns

These design patterns provide a way to create objects while hiding the creation logic, rather than instantiating objects directly using new operator. This gives program more flexibility in deciding which objects need to be created for a given use case.

Factory

namespace Factory

{

public interface ISavingAccount

{

void Balance();

}

public class CITISavingAccount : ISavingAccount

{

public void Balance()

{

Console.WriteLine("Citi saving account has $200.");

}

}

public class NationalSavingAccount : ISavingAccount

{

public void Balance()

{

Console.WriteLine("National saving account has $300.");

}

}

public interface ICreditUnionFatory

{

ISavingAccount Create(string AccountNumber);

}

public class CreaditUnionFatcory : ICreditUnionFatory

{

public ISavingAccount Create(string AccountNumber)

{

if (AccountNumber.Contains("CITI"))

{

return new CITISavingAccount();

}

else if (AccountNumber.Contains("National"))

{

return new NationalSavingAccount();

}

else

throw new InvalidOperationException();

}

}

class Program

{

static void Main(string[] args)

{

ISavingAccount savingAccount = null;

CreaditUnionFatcory obj = new CreaditUnionFatcory();

savingAccount = obj.Create("CITI-100");

savingAccount.Balance();

savingAccount = obj.Create("National-100");

savingAccount.Balance();

Console.ReadLine();

}

}

}

Abstract Factory

namespace AbstractFactory

{

public interface ISavingAccount

{

}

public interface ILoanAccount

{

}

public class CITISavingAccount : ISavingAccount

{

public CITISavingAccount()

{

Console.WriteLine("This is CITI Saving Account.");

}

}

public class CITILoanAccount : ILoanAccount

{

public CITILoanAccount()

{

Console.WriteLine("This is CITI Loan Account.");

}

}

public class NationalSavingAccount : ISavingAccount

{

public NationalSavingAccount()

{

Console.WriteLine("This is National Saving Account.");

}

}

public class NationalLoanAccount : ILoanAccount

{

public NationalLoanAccount()

{

Console.WriteLine("This is National Loan Account.");

}

}

public interface ICreditUnionFactory

{

ILoanAccount CreateLoanAccount();

ISavingAccount CreateSavingAccount();

}

public class CITICreditUnionFactory : ICreditUnionFactory

{

public ILoanAccount CreateLoanAccount()

{

return new CITILoanAccount(); }

public ISavingAccount CreateSavingAccount()

{

return new CITISavingAccount();

}

}

public class NationalCreditUnionFactory : ICreditUnionFactory

{

public ILoanAccount CreateLoanAccount()

{

return new NationalLoanAccount();

}

public ISavingAccount CreateSavingAccount()

{

return new NationalSavingAccount();

}

}

public class CreditUnionFactoryProvider

{

public static ICreditUnionFactory GetCreditUnionFactroy(string accountNo)

{

if (accountNo.Contains("CITI"))

{

return new CITICreditUnionFactory();

}

else if (accountNo.Contains("NATIONAL"))

{

return new NationalCreditUnionFactory();

}

else

{

return null;

}

}

}

class Program

{

static void Main(string[] args)

{

List<string> accounts = new List<string>();

accounts.Add("CITI - 100");

accounts.Add("NATIONAL - 200");

foreach (var account in accounts)

{

ICreditUnionFactory factory = CreditUnionFactoryProvider.GetCreditUnionFactroy(account);

factory.CreateLoanAccount();

factory.CreateSavingAccount();

}

Console.ReadLine();

}

}

}

Singleton

namespace Singleton

{

public class Policy

{

private int policyNo { get; set; } = 12345;

private string Insured { get; set; } = "Peter Parker";

public string GetInsuredName() => Insured;

//private static Policy \_instance;

// Not Thread safe

//public static Policy Instance

//{

// get

// {

// if (\_instance == null)

// {

// \_instance = new Policy();

// }

// return \_instance;

// }

//}

// Thread Safe

//private static readonly object \_lock = new object();

//public static Policy Instance

//{

// get

// {

// lock (\_lock)

// {

// if (\_instance == null)

// {

// \_instance = new Policy();

// }

// return \_instance;

// }

// }

//}

// Even Better Approch

private static readonly Policy \_instance = new Policy();

private static readonly object \_lock = new object();

public static Policy Instance

{

get

{

return \_instance;

}

}

}

class Program

{

static void Main(string[] args)

{

Policy policy = new Policy();

string insured = policy.GetInsuredName();

Console.WriteLine(insured);

Console.ReadLine();

}

}

}

Structural Patterns

These design patterns concern class and object composition. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities.

Decorator

Component

namespace Decorator.Component

{

public abstract class Car

{

public string Description { get; set; }

public abstract string GetDescription();

public abstract double GetPrice();

}

}

Concrete Component

namespace Decorator.ConcreteComponent

{

public class CompactCar:Car

{

public CompactCar()

{

Description = "Compact Car";

}

public override string GetDescription() => Description;

public override double GetPrice() => 29000.0;

}

}

namespace Decorator.ConcreteComponent

{

public class FullSizeCar : Car

{

public FullSizeCar()

{

Description = "Full Size Car";

}

public override string GetDescription() => Description;

public override double GetPrice()=> 29000.0;

}

}

Decorator

namespace Decorator.Decorator

{

public class CarDecorator: Car

{

protected Car \_car;

public CarDecorator(Car car)

{

\_car = car;

}

public override string GetDescription() => \_car.GetDescription();

public override double GetPrice() => \_car.GetPrice();

}

}

Concrete Decorator

namespace Decorator.ConcreteDecorator

{

public class LeatherSeat: CarDecorator

{

public LeatherSeat(Car car):base(car)

{

Description = "Leather Seat";

}

public override string GetDescription() => $"{\_car.GetDescription()}, {Description}";

public override double GetPrice() => \_car.GetPrice() + 2500;

}

}

namespace Decorator.ConcreteDecorator

{

public class Navigation : CarDecorator

{

public Navigation(Car car) : base(car)

{

Description = "Navigation";

}

public override string GetDescription() => $"{\_car.GetDescription()}, {Description}";

public override double GetPrice() => \_car.GetPrice() + 5000;

}

}

namespace Decorator.ConcreteDecorator

{

public class Sunroof : CarDecorator

{

public Sunroof(Car car) : base(car)

{

Description = "Sunroof";

}

public override string GetDescription() => $"{\_car.GetDescription()}, {Description}";

public override double GetPrice() => \_car.GetPrice() + 7000;

}

}

class Program

{

static void Main(string[] args)

{

Car compactCar = new CompactCar();

compactCar = new Navigation(compactCar);

compactCar = new Sunroof(compactCar);

Console.WriteLine(compactCar.GetDescription());

Console.WriteLine(compactCar.GetPrice());

Console.ReadLine();

}

}

Behavioral Patterns

These design patterns are specifically concerned with communication between objects.

Observer

Observer

public interface IFan

{

void Update(ICelebrity celebrity);

}

ConcreteObserver

public class Fan : IFan

{

public void Update(ICelebrity celebrity)

{

Console.WriteLine($"Fan notified. Tweet of {celebrity.FullName} : {celebrity.Tweet}");

}

}

Subject

namespace Observer.Subject

{

public interface ICelebrity

{

string FullName { get; }

string Tweet { get; set; }

void Notify(string tweet);

void AddFollower(IFan fan);

void RemoveFollower(IFan fan);

}

}

ConcreteSubject

namespace Observer.ConcreteSubject

{

public class GClooney : ICelebrity

{

private readonly List<IFan> \_fans = new List<IFan>();

private string \_tweet;

public GClooney(string tweet)

{

\_tweet = tweet;

}

public string FullName => "George Clooney";

public string Tweet { get { return \_tweet; } set { Notify(value); } }

public void AddFollower(IFan fan)

{

\_fans.Add(fan);

}

public void Notify(string tweet)

{

\_tweet = tweet;

foreach (var fan in \_fans)

{

fan.Update(this);

}

}

public void RemoveFollower(IFan fan)

{

\_fans.Remove(fan);

}

}

}

namespace Observer.ConcreteSubject

{

public class TSwift : ICelebrity

{

private readonly List<IFan> \_fans = new List<IFan>();

private string \_tweet;

public TSwift(string tweet)

{

\_tweet = tweet;

}

public string FullName => "T Swift";

public string Tweet { get { return \_tweet; } set { Notify(value); } }

public void AddFollower(IFan fan)

{

\_fans.Add(fan);

}

public void Notify(string tweet)

{

\_tweet = tweet;

foreach (var fan in \_fans)

{

fan.Update(this);

}

}

public void RemoveFollower(IFan fan)

{

\_fans.Remove(fan);

}

}

}

namespace Observer

{

class Program

{

static void Main(string[] args)

{

var gClooney = new GClooney("I Love MY New Wife.");

var tSwift = new TSwift("I started a joke is my favorite number.");

var firstFan = new Fan();

var secondFan = new Fan();

gClooney.AddFollower(firstFan);

tSwift.AddFollower(secondFan);

gClooney.Tweet= "I Love Joker";

tSwift.Tweet= "I am looser";

Console.ReadLine();

}

}

}

Iterator

namespace Iterator.Iterator

{

public interface IIterator

{

void First();

string Next();

bool Done();

string CurrentItem();

}

}

namespace Iterator.Iterator

{

public class LANewspaperIterator: IIterator

{

private string[] \_reporters;

private int \_current;

public LANewspaperIterator(string[] reporters)

{

\_reporters = reporters;

\_current = 0;

}

public string CurrentItem()

{

return \_reporters[\_current];

}

public bool Done()

{

return \_current >= \_reporters.Length;

}

public void First()

{

\_current = 0;

}

public string Next()

{

return \_reporters[\_current++];

}

}

}

namespace Iterator.Iterator

{

public class NYNewspaperIterator : IIterator

{

private List<string> \_reporters;

private int \_current;

public NYNewspaperIterator(List<string> reporters)

{

\_reporters = reporters;

\_current = 0;

}

public string CurrentItem()

{

return \_reporters[\_current];

}

public bool Done()

{

return \_current >= \_reporters.Count;

}

public void First()

{

\_current = 0;

}

public string Next()

{

return \_reporters[\_current++];

}

}

}

Aggregate

namespace Iterator.Aggregate

{

public interface INewspaper

{

IIterator CreateIterator();

}

}

namespace Iterator.Aggregate

{

public class LANewspaper : INewspaper

{

private string[] \_reporters;

public LANewspaper()

{

\_reporters = new[] {

"Bruce Wyan -LA",

"Kent Clark-LA"

};

}

public IIterator CreateIterator()

{

return new LANewspaperIterator(\_reporters);

}

}

}

namespace Iterator.Aggregate

{

public class NYNewspaper : INewspaper

{

private List<string> \_reporters;

public NYNewspaper()

{

\_reporters = new List<string>

{

"Peter Parker - NY",

"Marry Jane - NY"

};

}

public IIterator CreateIterator()

{

return new NYNewspaperIterator(\_reporters);

}

}

}

class Program

{

static void Main(string[] args)

{

INewspaper ny = new NYNewspaper();

INewspaper la = new LANewspaper();

IIterator nyItr = ny.CreateIterator();

IIterator laItr = la.CreateIterator();

Console.WriteLine("----- NY ");

PrintReporter(nyItr);

Console.WriteLine("----- LA ");

PrintReporter(laItr);

Console.ReadLine();

}

public static void PrintReporter(IIterator iterator)

{

iterator.First();

while (!iterator.Done())

{

Console.WriteLine(iterator.Next());

}

}

}

Repository Pattern

namespace StudentDemo.Models

{

public class Faculity

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

public int Year { get; set; }

}

}

namespace StudentDemo.Models

{

public class StudentDemoContext : DbContext

{

// You can add custom code to this file. Changes will not be overwritten.

//

// If you want Entity Framework to drop and regenerate your database

// automatically whenever you change your model schema, please use data migrations.

// For more information refer to the documentation:

// http://msdn.microsoft.com/en-us/data/jj591621.aspx

public StudentDemoContext() : base("name=connectionString")

{

}

public System.Data.Entity.DbSet<StudentDemo.Models.Student> Students { get; set; }

public System.Data.Entity.DbSet<StudentDemo.Models.Faculity> Faculities { get; set; }

}

}

public class RepositoryContext : DbContext

{

// You can add custom code to this file. Changes will not be overwritten.

//

// If you want Entity Framework to drop and regenerate your database

// automatically whenever you change your model schema, please use data migrations.

// For more information refer to the documentation:

// http://msdn.microsoft.com/en-us/data/jj591621.aspx

public RepositoryContext() : base("name=connectionString")

{

}

public System.Data.Entity.DbSet<StudentDemo.Models.Student> Students { get; set; }

public System.Data.Entity.DbSet<StudentDemo.Models.Faculity> Faculities { get; set; }

}

public interface IStudentRepository :IDisposable

{

Task<List<Student>> GetStudentList();

Task<Student> GetStudentById(int? Id);

void InsertStudent(Student student);

void UpdateStudent(Student student);

void Delete(int Id);

void DeleteStudent(Student student);

Task<int> Save();

}

public class StudentRepository : IStudentRepository , IDisposable

{

// Part 1

private RepositoryContext \_context;

public StudentRepository(RepositoryContext context)

{

\_context = context;

}

// Part 2

public void Delete(int Id)

{

Student student = \_context.Students.Find(Id);

\_context.Students.Remove(student);

}

public void DeleteStudent(Student student)

{

\_context.Students.Remove(student);

}

public Task<Student> GetStudentById(int? Id)

{

return \_context.Students.FindAsync(Id);

}

public Task<List<Student>> GetStudentList()

{

return \_context.Students.ToListAsync();

}

public void InsertStudent(Student student)

{

\_context.Students.Add(student);

}

public Task<int> Save()

{

return \_context.SaveChangesAsync();

}

public void UpdateStudent(Student student)

{

\_context.Entry(student).State = EntityState.Modified;

}

// Part 3

private bool disposed = false;

protected virtual void Dispose(bool disposing)

{

if (!this.disposed)

{

if (disposing)

{

\_context.Dispose();

}

this.disposed = true;

}

}

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

}

public class GenericRepository<TEntity> where TEntity : class

{

// Part 1

internal RepositoryContext context;

internal DbSet<TEntity> dbSet;

public GenericRepository(RepositoryContext context)

{

this.context = context;

this.dbSet = context.Set<TEntity>();

}

// Part 2

public virtual IEnumerable<TEntity> Get(

Expression<Func<TEntity, bool>> filter = null,

Func<IQueryable<TEntity>, IOrderedQueryable<TEntity>> orderBy = null,

string includeProperties = "")

{

IQueryable<TEntity> query = dbSet;

if (filter != null)

{

query = query.Where(filter);

}

foreach (var includeProperty in includeProperties.Split(

new char[] { ','}, StringSplitOptions.RemoveEmptyEntries))

{

query = query.Include(includeProperty);

}

if (orderBy != null)

{

return orderBy(query).ToList();

}

else

{

return query.ToList();

}

}

public virtual List<TEntity> Get()

{

return dbSet.ToList();

}

public virtual TEntity GetById(Object Id)

{

return dbSet.Find(Id);

}

public virtual void Insert(TEntity entity)

{

dbSet.Add(entity);

}

public virtual void Save()

{

}

public virtual void Update(TEntity entityToUpdate)

{

dbSet.Attach(entityToUpdate);

context.Entry(entityToUpdate).State = EntityState.Modified;

}

public virtual void Delete(object Id)

{

TEntity entityToDelete = dbSet.Find(Id);

Delete(entityToDelete);

}

public virtual void Delete(TEntity entityToDelete)

{

if (context.Entry(entityToDelete).State == EntityState.Detached)

{

dbSet.Attach(entityToDelete);

}

dbSet.Remove(entityToDelete);

}

}

public class UnitOfWork : IDisposable

{

//Part 1

private RepositoryContext context = new RepositoryContext();

private GenericRepository<Faculity> faculityRepository;

private GenericRepository<Student> studentRepository;

public GenericRepository<Faculity> FaculityRepository

{

get

{

if (this.faculityRepository == null)

{

this.faculityRepository = new GenericRepository<Faculity>(context);

}

return this.faculityRepository;

}

}

public GenericRepository<Student> StudentRepository

{

get

{

if (this.faculityRepository == null)

{

this.studentRepository = new GenericRepository<Student>(context);

}

return this.studentRepository;

}

}

// Part 2

public void Save()

{

context.SaveChanges();

}

private bool disposed = false;

protected virtual void Dispose(bool disposing)

{

if (!this.disposed)

{

if (disposing)

{

context.Dispose();

}

}

this.disposed = true;

}

public void Dispose()

{

Dispose(true);

}

}

public class StudentController : Controller

{

//private StudentDemoContext db = new StudentDemoContext();

private IStudentRepository studentRepository;

private GenericRepository<Student> genericRepository;

public StudentController()

{

this.studentRepository = new StudentRepository(new RepositoryContext());

this.genericRepository = new GenericRepository<Student>(new RepositoryContext());

}

// GET: Student

public async Task<ActionResult> Index()

{

//return View(await db.Students.ToListAsync());

//return View(await studentRepository.GetStudentList());

return View(genericRepository.Get());

}

// GET: Student/Details/5

public async Task<ActionResult> Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

//Student student = await db.Students.FindAsync(id);

//Student student = await studentRepository.GetStudentById(id);

Student student = genericRepository.GetById(id);

if (student == null)

{

return HttpNotFound();

}

return View(student);

}

// GET: Student/Create

public ActionResult Create()

{

return View();

}

// POST: Student/Create

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Create([Bind(Include = "Id,Name,Year")] Student student)

{

if (ModelState.IsValid)

{

//db.Students.Add(student);

//await db.SaveChangesAsync();

studentRepository.InsertStudent(student);

await studentRepository.Save();

return RedirectToAction("Index");

}

return View(student);

}

// GET: Student/Edit/5

public async Task<ActionResult> Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

//Student student = await db.Students.FindAsync(id);

Student student = await studentRepository.GetStudentById(id);

if (student == null)

{

return HttpNotFound();

}

return View(student);

}

// POST: Student/Edit/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Edit([Bind(Include = "Id,Name,Year")] Student student)

{

if (ModelState.IsValid)

{

//db.Entry(student).State = EntityState.Modified;

//await db.SaveChangesAsync();

studentRepository.UpdateStudent(student);

await studentRepository.Save();

return RedirectToAction("Index");

}

return View(student);

}

// GET: Student/Delete/5

public async Task<ActionResult> Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

//Student student = await db.Students.FindAsync(id);

Student student = await studentRepository.GetStudentById(id);

if (student == null)

{

return HttpNotFound();

}

return View(student);

}

// POST: Student/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DeleteConfirmed(int id)

{

//Student student = await db.Students.FindAsync(id);

//db.Students.Remove(student);

//await db.SaveChangesAsync();

Student student = await studentRepository.GetStudentById(id);

studentRepository.DeleteStudent(student);

await studentRepository.Save();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

//db.Dispose();

studentRepository.Dispose();

}

base.Dispose(disposing);

}

}

public class FaculityController : Controller

{

//private StudentDemoContext db = new StudentDemoContext();

private UnitOfWork unitofwork = new UnitOfWork();

// GET: Faculity

public ActionResult Index()

{

//return View(db.Faculities.ToList());

return View(unitofwork.FaculityRepository.Get());

}

// GET: Faculity/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

//Faculity faculity = db.Faculities.Find(id);

Faculity faculity = unitofwork.FaculityRepository.GetById(id);

if (faculity == null)

{

return HttpNotFound();

}

return View(faculity);

}

// GET: Faculity/Create

public ActionResult Create()

{

return View();

}

// POST: Faculity/Create

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Id,Name")] Faculity faculity)

{

if (ModelState.IsValid)

{

//db.Faculities.Add(faculity);

//db.SaveChanges();

unitofwork.FaculityRepository.Insert(faculity);

unitofwork.Save();

return RedirectToAction("Index");

}

return View(faculity);

}

// GET: Faculity/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

//Faculity faculity = db.Faculities.Find(id);

Faculity faculity = unitofwork.FaculityRepository.GetById(id);

if (faculity == null)

{

return HttpNotFound();

}

return View(faculity);

}

// POST: Faculity/Edit/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "Id,Name")] Faculity faculity)

{

if (ModelState.IsValid)

{

//db.Entry(faculity).State = EntityState.Modified;

//db.SaveChanges();

unitofwork.FaculityRepository.Update(faculity);

unitofwork.Save();

return RedirectToAction("Index");

}

return View(faculity);

}

// GET: Faculity/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

//Faculity faculity = db.Faculities.Find(id);

Faculity faculity = unitofwork.FaculityRepository.GetById(id);

if (faculity == null)

{

return HttpNotFound();

}

return View(faculity);

}

// POST: Faculity/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

//Faculity faculity = db.Faculities.Find(id);

//db.Faculities.Remove(faculity);

//db.SaveChanges();

Faculity faculity = unitofwork.FaculityRepository.GetById(id);

unitofwork.FaculityRepository.Update(faculity);

unitofwork.Save();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

//db.Dispose();

unitofwork.Dispose();

}

base.Dispose(disposing);

}

}