بالتأكيد، سأقوم بشرح الأمر بشكل أكثر تفصيلاً:

1. تعريف واجهة المستودع (IRepository) باستخدام واجهة IRepositoryBase:  
   تبدأ عملية استخدام واجهة IRepositoryBase بتعريف واجهة المستودع (IRepository) باستخدام واجهة IRepositoryBase. في مثالنا، نحن نعرف واجهة المستودع IEmployeeRepository بناءً على واجهة IRepositoryBase على النحو التالي:

Copy

public interface IEmployeeRepository : IRepositoryBase<Employee>

{

// يمكن إضافة الأساليب الخاصة بالكائن Employee إلى هنا

}

1. تعريف مستودع الكائن Employee (EmployeeRepository) وتنفيذه بناءً على واجهة المستودع (IEmployeeRepository):  
   ثم، نقوم بتعريف مستودع الكائن Employee وتنفيذه بناءً على واجهة المستودع IEmployeeRepository. يمكننا استخدام الأساليب القياسية الموجودة في واجهة IRepositoryBase لتنفيذ العمليات الأساسية على كائنات Employee، مثل الإضافة (Create) والتعديل (Update) والحذف (Delete) والبحث (Get). يتم تنفيذ المستودع EmployeeRepository بناءً على IEmployeeRepository على النحو التالي:

Copy

public class EmployeeRepository : RepositoryBase<Employee>, IEmployeeRepository

{

public EmployeeRepository(ApplicationDbContext context) : base(context)

{

}

// يمكن إضافة الأساليب الخاصة بالكائن Employee إلى هنا

}

1. تعريف واجهة IRepositoryBase:  
   تعريف واجهة IRepositoryBase يتم استخدامها لتعريف الأساليب القياسية التي يجب تطبيقها في جميع المستودعات المرتبطة بالتطبيق. في مثالنا، يتم تعريف واجهة IRepositoryBase على النحو التالي:

arduino

Copy

public interface IRepositoryBase<T> where T : class

{

void Create(T entity);

void Update(T entity);

void Delete(T entity);

Task<T> GetById(int id);

Task<IEnumerable<T>> GetAll();

Task<IEnumerable<T>> GetByCondition(Expression<Func<T, bool>> expression);

}

1. تعريف مستودع RepositoryBase العام وتنفيذه بناءً على واجهة IRepositoryBase:  
   تعريف مستودع RepositoryBase العام وتنفيذه بناءً على واجهة IRepositoryBase. يتم استخدام المستودع العام RepositoryBase لتنفيذ الأساليب القياسية الموجودة في واجهة IRepositoryBase، والتي يمكن تطبيقها على جميع المستودعات في التطبيق. يتم تعريف المستودع RepositoryBase على النحو التالي:

typescript

Copy

public class RepositoryBase<T> : IRepositoryBase<T> where T : class

{

protected readonly ApplicationDbContext \_context;

public RepositoryBase(ApplicationDbContext context)

{

\_context = context;

}

public void Create(T entity)

{

\_context.Set<T>().Add(entity);

}

public void Update(T entity)

{

\_context.Set<T>().Update(entity);

}

public void Delete(T entity)

{

\_context.Set<T>().Remove(entity);

}

public async Task<T> GetById(int id)

{

return await \_context.Set<T>().FindAsync(id);

}

public async Task<IEnumerable<T>> GetAll()

{

return await \_context.Set<T>().ToListAsync();

}

public async Task<IEnumerable<T>> GetByCondition(Expression<Func<T, bool>> expression)

{

return await \_context.Set<T>().Where(expression).ToListAsync();

}

}

بشكل عام، يتم استخدام واجهة IRepositoryBase لتعريف الأساليب القياسية المشتركة التي يجب تطبيقها في جميع المستودعات المرتبطة بالتطبيق. عندما يتم تنفيذ مستودع معين - مثل EmployeeRepository في مثالنا - فإنه يجب تنفيذ واجهة المستودع المرتبطة بالواجهة IRepositoryBase - في هذه الحالة، IEmployeeRepository - وتنفيذ الأساليب الخاصة بكائن Employee في داخله. ويجب أن يستخدم RepositoryBase لتنفيذ الأساليب القياسية الموجودة في واجهة IRepositoryBase. هذا يجعل الكود أكثر قابلية لإعادة الاستخدام والصيانة، ويسهل عملية تطوير التطبيقات.