执行策略（重试策略）：EF提供了执行策略，在提交失败时可以自动重试

## **自定义执行策略**

**定义执行策略类**

public class MyExecutionStrategy : ExecutionStrategy

{

    // maxRetryCount：重试最大次数

    // maxRetryDelay：重试时间间隔

    public MyExecutionStrategy (DbContext context, int maxRetryCount, TimeSpan maxRetryDelay) :

        base (context, maxRetryCount, maxRetryDelay) { }

    // 是否应该重试

    // 如果返回false，将不再重试，否则重试至 maxRetryCount 次

    protected override bool ShouldRetryOn (Exception exception)

    {

        if (exception is SqlException) {

            return true;

        }

        return false;

    }

}

**指定EF执行策略**

// 注入上下文

services.AddDbContext<MyDbContext> (builder => {

    builder.UseSqlServer (

        @"connect string",

        options => {

            // 指定执行策略

            options.ExecutionStrategy (dependencies =>

                new MyExecutionStrategy (dependencies.CurrentDbContext.Context, 1, new TimeSpan (0))

            );

        });

});

**微软老大提供的执行策略**

微软为我们提供了失败重试策略，因此我们无需自定义策略

// 注入上下文

services.AddDbContext<MyDbContext> (builder => {

    builder.UseSqlServer (

        @"connect string",

        options => {

            // 启用失败重试策略

            // EnableRetryOnFailure 默认对一些sql错误码进行重试，我们也可以添加错误码

            // 如：4060 是无法登录，默认重试的sql错误码没有包含4060，所以这里添加了4060

            options.EnableRetryOnFailure(2, new TimeSpan(0), errorNumbersToAdd: new int[] { 4060 });

        });

});

## **自定义执行操作**

在 Context.SaveChanges(); 失败时默认重试策略是重新执行 Context.SaveChanges();

我们定义如下代码，当该段代码内执行失败时，重试策略重新执行该段代码

public IActionResult InsertEntity ()

{

    // 生成执行策略

    var strategy = \_myDbContext.Database.CreateExecutionStrategy ();

    // 指定执行策略的操作

    // 我们运行这段代码，传给 Execute 的 Action（执行策略的操作） 会立刻执行

    // 如果执行抛出异常，且 strategy.ShouldRetryOn() 方法返回 true，重新执行该 Action

    strategy.Execute (() =>

    {

        // 这里为什么要新建数据库上下文？

        // 因为上下文执行失败后，不应该使用就的上下文

        using (MyDbContext context = new MyDbContext ((DbContextOptions<MyDbContext>) HttpContext.RequestServices.GetService (typeof (DbContextOptions<MyDbContext>))))

        {

            context.MyEntities.Add (new MyEntity () { Name = "keke" });

            context.SaveChanges ();

        }

});

    return Redirect ("/My/MyAction");

}

## **提交期间的失败**

试想，如果我们将插入的语句提交到数据库，数据库还没返回是否执行成功，就出现了连接中断，那数据到底是成功保存到数据库了还是没保存，要不要重试

我们有一些几个选择

### **选项 1-不做任何操作**

如：你想添加一篇文章，但连接中断了，你不知道是否添加了，你可以查了文章列表来确定释放添加

### **选项 2-添加状态验证**

对于上述问题，我们需要自己验证是否保存成功再决定是否重试

public IActionResult InsertEntity ()

{

    var strategy = \_myDbContext.Database.CreateExecutionStrategy ();

    var myEntity = new MyEntity () { Id = 333, Name = "keke" };

\_myDbContext.MyEntities.Add (myEntity);

    // 自定义策略操作

    strategy.ExecuteInTransaction (

        // 传入的上下文

        \_myDbContext,

        // 要执行的操作

        operation : context => {

            // 保存到数据库，但不接收本地实体状态的改变（这里我们就可以一直使用SaveChanges了）

            context.SaveChanges (acceptAllChangesOnSuccess: false);

        },

        // 验证是否执行成功操作

        verifySucceeded : context => context.MyEntities.AsNoTracking ().Any (e => e.Id == myEntity.Id));

    // 接收本地实体状态的改变

\_myDbContext.ChangeTracker.AcceptAllChanges ();

    return Redirect ("/My/MyAction");

}