DotNetFramework.CAP 使用手册

## Step1 基础库引用。

* 引用DotNetFramework.CAP基础库。
* 引用DotNetFramework.CAP.SqlServer
* 引用DotNetFramework.CAP.SqlServer

## Step2 启动配置

public class CapConfig

{

public static IServiceCollection Services { get; set; }

public static void RegisterCap()

{

Services = new ServiceCollection();

Services.AddCap(stetup =>

{

// 注册节点到 Consul

stetup.UseSqlServer("Data Source=localhost;database=EFCodeFirst;Uid=sa;pwd=sa;");

stetup.UseRabbitMQ(option =>

{

option.VirtualHost = "HengQueue";

option.HostName = "localhost";

option.Port = 5672;

option.UserName = "zhangheng";

option.Password = "123456";

});

});

Services.AddScopedMuti<ICapSubscribe, OpService>();

Services.AddScopedMuti<ICapSubscribe, Op2Service>();

Services.BeginRegister();

Services.ServiceProvider.GetService<IBootstrapper>().BootstrapAsync(new CancellationToken());

}

}

protected void Application\_Start()

{

CapConfig.RegisterCap();

}

## Step3 消息发布

* Ado.net

使用消息和执行事务控制的（消息持久化库必须和当前连接执行业务同一库）

using (var connection = new SqlConnection(""))

using (var transaction = connection.BeginTransaction(\_capBus, autoCommit: false))

{

//业务代码

//这里的事务是指 发送消息和执行当前SQL一致性

connection.Execute(sql, transaction: transaction);

\_capBus.Publish("xxx.services.update.username", "12", "callback-show-execute-time");

transaction.Commit(\_capBus);

}

}

使用消息无事务控制的（不能确保消息失败执行会回滚）

using (var connection = new SqlConnection(""))

connection.Execute(sql, transaction: transaction);

\_capBus.Publish("xxx.services.update.username", "12", "callback-show-execute-time");

}

}

* EntityFramework

使用消息和执行事务控制的（消息持久化库必须和当前连接执行业务同一库）

using (var context = new MyContext())

{ using (var trans = context.Database.BeginTransaction(\_capBus, autoCommit: false))

{

//业务代码

\_capBus.Publish("xxx.services.update.username", "12", "callback-show-execute-time");

trans.Commit(\_capBus);

}

}

不使用消息和业务事务控制

using (var context = new MyContext())

{

//业务代码

\_capBus.Publish("xxx.services.update.username", "12", "callback-show-execute-time");

context.SaveChanges();

}

黄色为标记项： 确保使用扩展提交方法。

## Step3 消息订阅

* 1位于controller下边的方法（直接添加attribute及可）

XXController

[CapSubscribe("xxx.services.update.username")]

public string CheckReceivedMessage(string username)

{

Console.WriteLine(username);

return username;

}

* 位于其他类（例如OPService），

1. 需要实现接口ICapSubscribe
2. 需要在程序启动时注册到Ioc容器里面。且必须使用如下方法。

Services.AddScopedMuti<ICapSubscribe, OpService>();

Services.AddScopedMuti<ICapSubscribe, Op2Service>();

public class OpService : ICapSubscribe

{

[CapSubscribe("callback-show-execute-time")]

public void Callback(string username)

{

//异常后会多次尝试

if (string.IsNullOrEmpty(username))

throw new Exception();

Console.WriteLine(username);

}

[CapSubscribe("xxx.services.update.username")]

public string CheckReceivedMessage(string username)

{

Console.WriteLine(username);

return username;

}

}

## Step4比较

1. 修改后启动项，启动方式变化。
2. 手动提交事务变化。（trans.Commit(\_capBus);
3. ICapSubscribe 注入方式变化，

必须使用Services.AddScopedMuti<ICapSubscribe, Op2Service>()此方法;