【专题二：工业软件的通讯协议】1.Modbus和NModbus样码

一起来读开源工业软件系列

爱科技勤折腾的苹果二 2023-6-20

讨论内容

1 前导知识

- 1.1 客户端和服务器
- 1.2 设计模式：Factory、Facade和Adapter
- 1.3 通讯协议：Modbus

2 NModbus

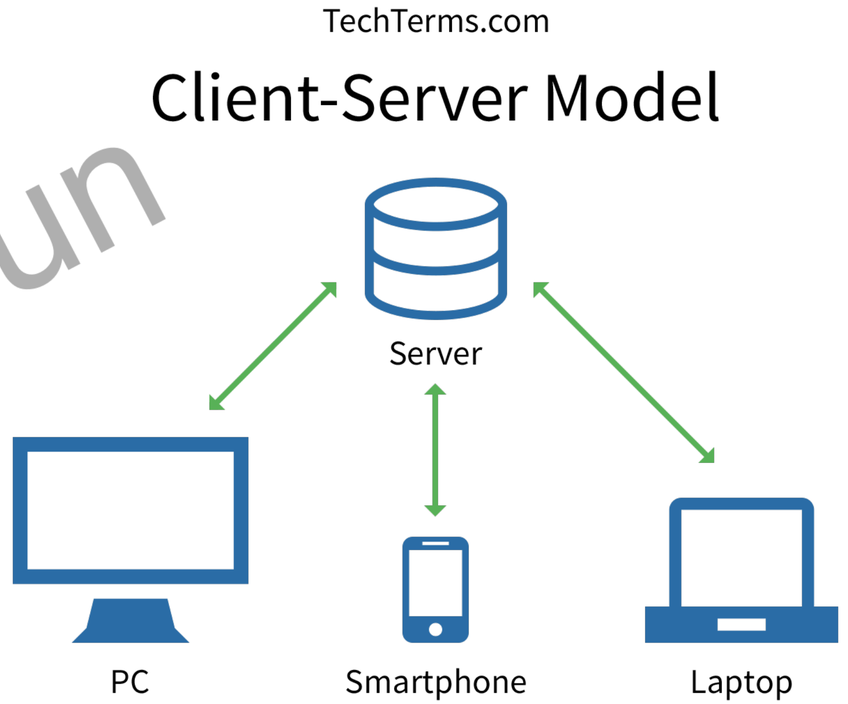
3 阅读NModbus样码

4 参考资料

1 前导知识

1.1 客户端和服务器架构

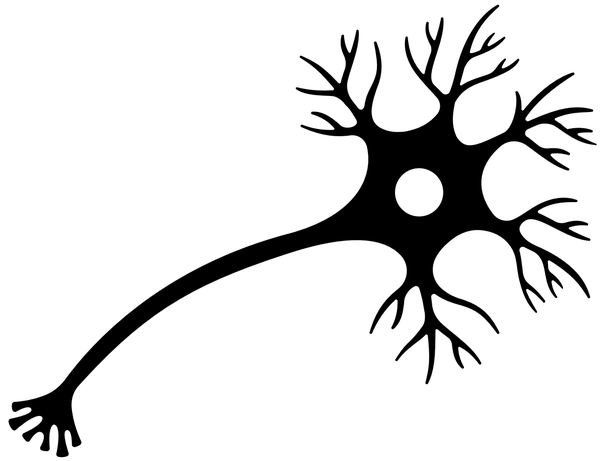
- 常用通讯协议：TCP/IP协议，面向连接的协议
- 应用情景：面向一个中心的多个请求
- 优点：
  - 中心化结构，易通过安全策略施加的访问限制保护数据。
  - 客户端和服务器的通讯使用平台无关的协议。
- 缺点：当有太多的客户同时向服务器请求数据时，服务器会不堪重负。会造成网络拥堵，并可能导致服务被拒绝。



图片来源：https://techterms.com/definition/client-server_model

想一想，代码有这些特点吗?

- 将需求直接翻译成代码，if-else，switch case一大堆
- 没有设计，没有运用设计模式



设计模式的类型

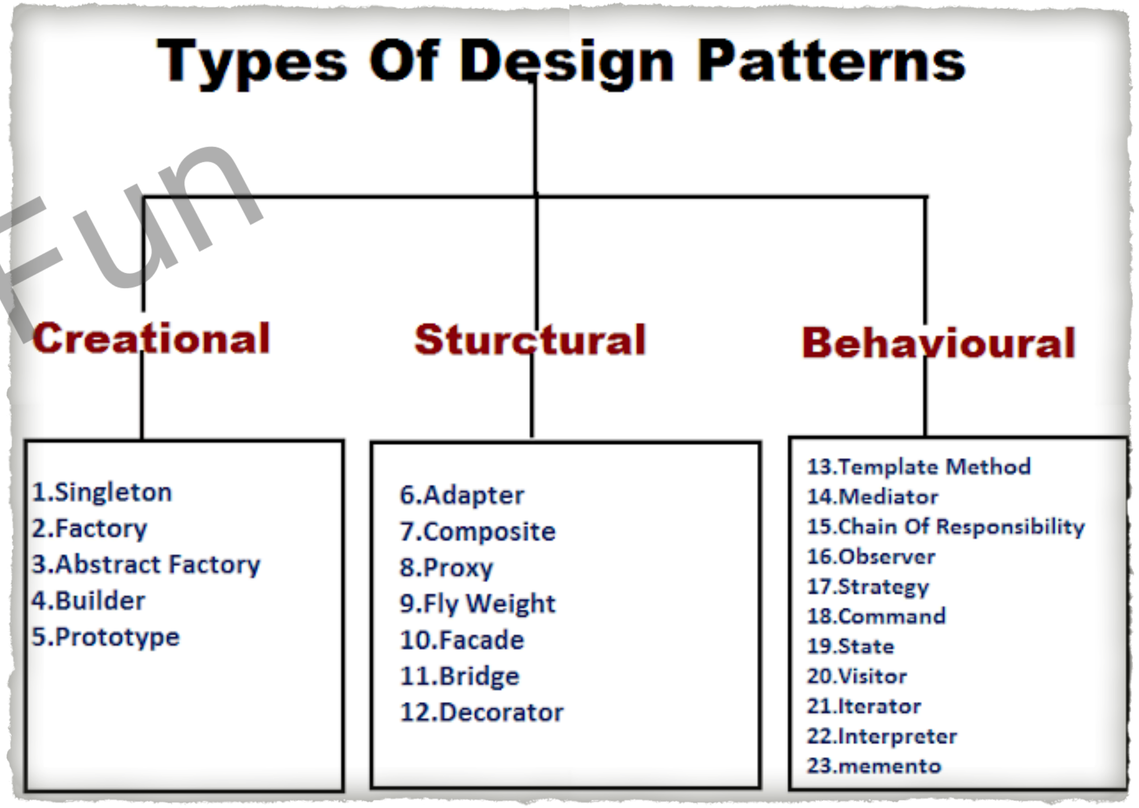
1 处理对象创建

- 将系统使用的具体类封装起来；
- 隐藏这些具体类的实例创建和结合的方式。

2 处理结构设计

- 通过识别一个简单的方法来实现实体之间的关系以简化设计；
- 处理类与对象的组合。

3 处理对象行为



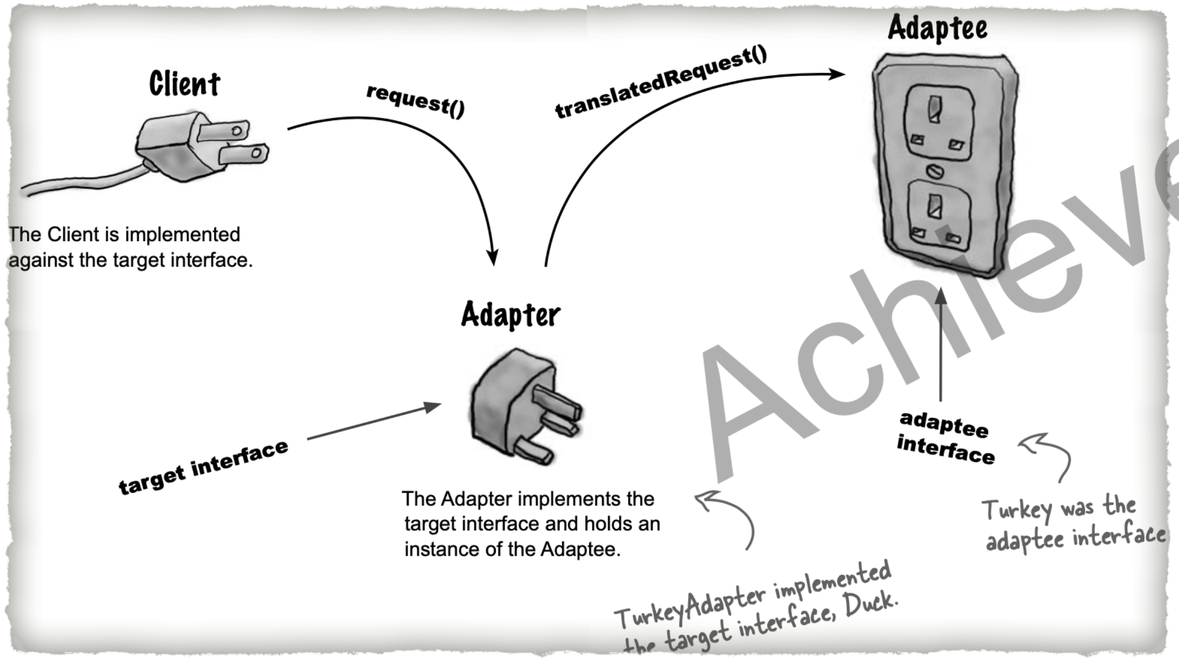
创建型模式：工厂方法模式

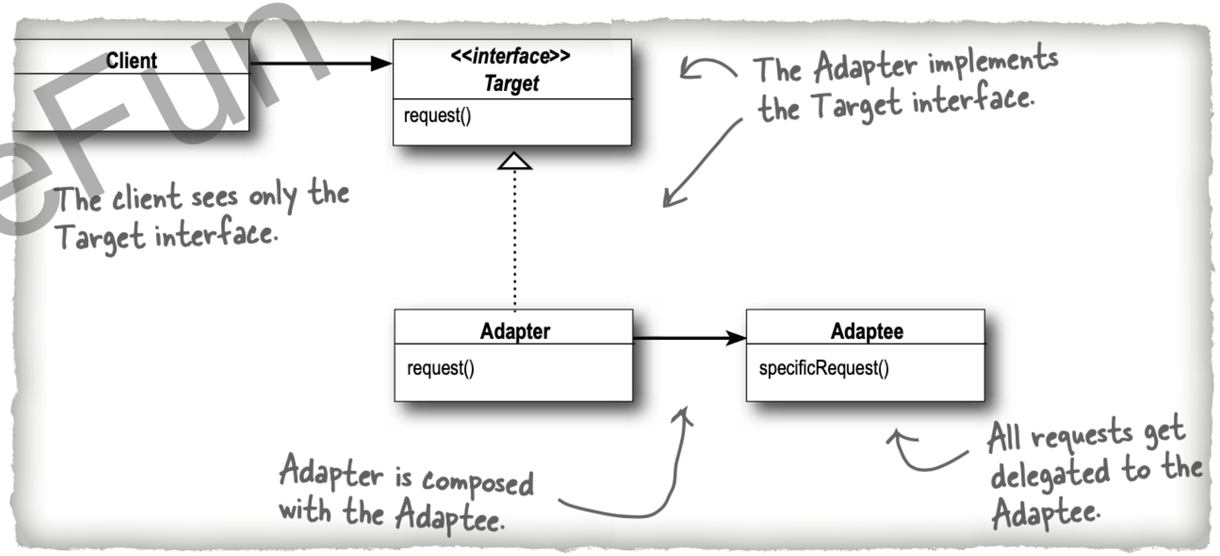
- 定义了一个创建对象的接口，但由子类决定要实例化的类是哪一个。工厂方法让类被实例化推迟到子类。

图片来源：HeadFirst设计模式

结构型模式：适配器模式

- 将类的接口转换为客户端期望的另一个接口。适配器让那些因为接口不兼容而不能一起工作的类一起工作。





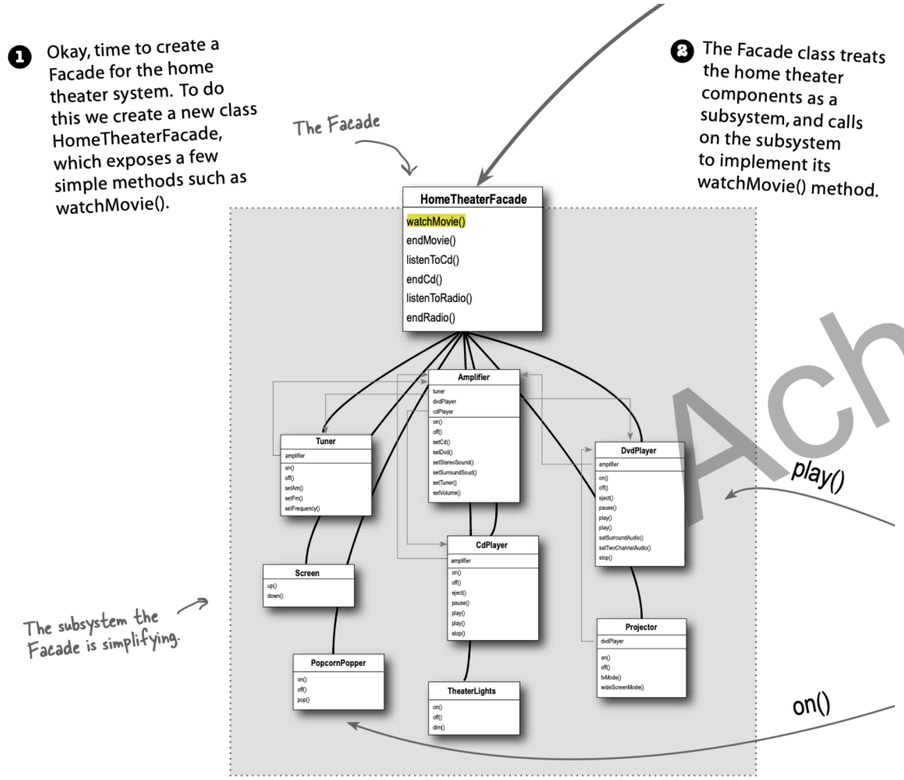
图片来源：HeadFirst设计模式

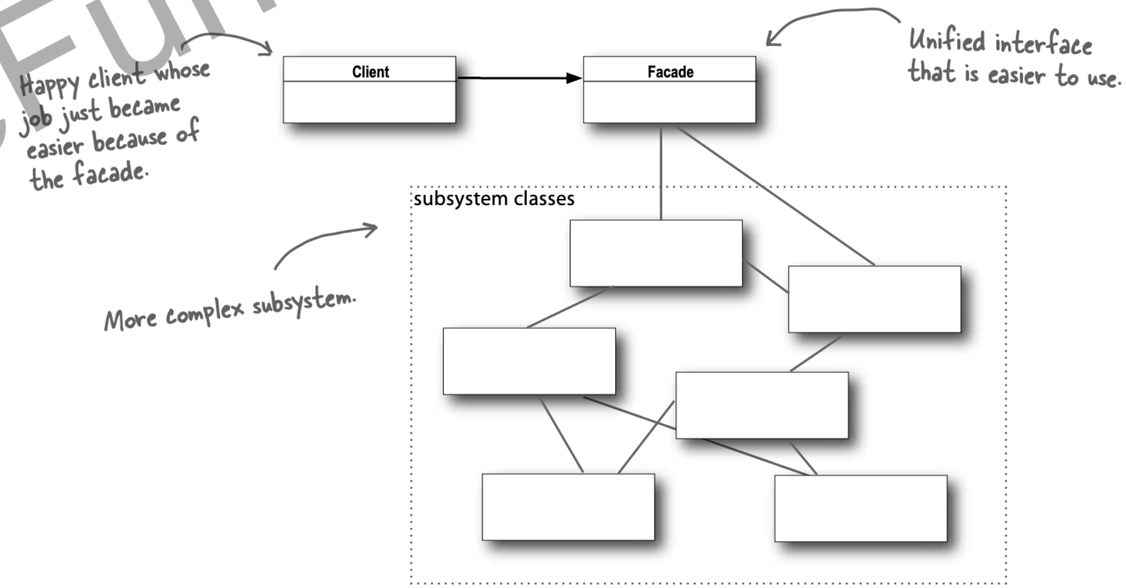
适配器模式在工业软件中的应用

- 采用对象适配器模式以便对接不同的仪器对象。适配器模式可以将一个类的接口转换成客户期望的另一个接口。
- 适配器模式包括适配器和被适配器。一种智能仪器的适配器收到方法调用时，会委托给其对应的被适配器，被适配器会接收到适配器所应用的接口上的调用。
- 优点：有着良好的面向对象的设计原则，其使用对象组合，以修改的接口包装被适配者。客户只看到目标接口，适配器实现目标接口。所有的请求都委托给被适配者，适配器与被适配者组合。
- 在系统中，适配器将从系统发出的请求转换成不同智能设备可以理解的请求，以便匹配不同的智能设备。在设计中，可使用多个适配器，每一个都负责转换不同组的后台类。
- 体现了良好的面向对象设计原则，即使用对象组合。并且，被适配者的任何子类，都可以搭配着适配器使用。

结构型模式：Facade模式

- Facade模式：为子系统中的一组接口提供了一个统一的接口。Facade定义了一个更高层次的接口，使子系统更容易使用





图片来源：HeadFirst设计模式

设计模式总结

设计模式名称	类型	特点
Factory	创建型	将实现从使用中解耦，减少应用程序和具体类之间的依赖
Adapter	结构型	将一个接口转成另一个接口
Facade	结构型	让接口更简单，也将客户端Client从组件的子系统中解耦

1.3 Modbus通讯协议

- https://modbus.org/tech.php
- https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf
- 结构和功能
- 协议知识：寄存器、串行消息帧格式、差错校验、TCP消息帧格式、功能码

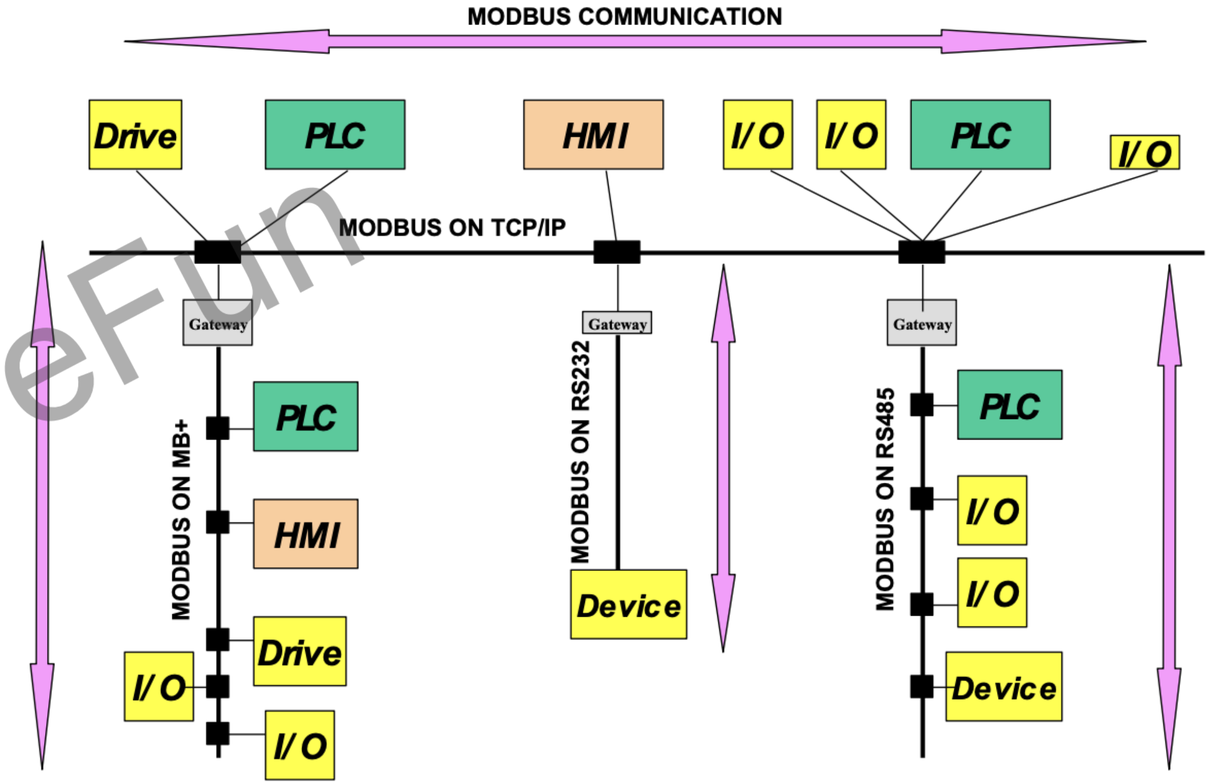


Figure 2: Example of MODBUS Network Architecture

Modbus RTU协议 vs Modbus TCP协议

- 通讯方式：一种串行通讯协议，数据通过串口进行传输。
- 编码方式：使用二进制编码。
- 通讯速率：较快，适用于需要高速数据传输和实时控制的应用场景。
- 通讯距离：通讯距离较短，一般不超过1000米。
- 使用场景
  - 工业自动化领域中的**串行通讯**应用，例如PLC、传感器、变频器等设备之间进行数据交换和通讯。
  - 通讯距离较短的应用场景，一般不超过1000米。
  - 需要高速数据传输和实时控制的应用场景

- 通讯方式：一种基于TCP/IP协议的网络通讯协议，数据通过以太网进行传输。
- 编码方式：使用ASCII码或者二进制编码。
- 通讯速率：通讯速率较慢，适用于需要长距离通讯和网络通讯的应用场景。
- 通讯距离：通讯距离较远，可以通过互联网进行通讯。
- 使用场景
  - 工业自动化领域中的**网络通讯**应用，例如智能制造、智能物流等领域。
  - 通讯距离较远的应用场景，可以通过互联网进行通讯。
  - 需要长距离通讯和网络通讯的应用场景。

2 NModbus

NModbus主要模块

- https://nmodbus.github.io/api/NModbus.html
- https://github.com/NModbus/NModbus
- https://github.com/NModbus/NModbus/tree/develop/NModbus

名称	说明	类型
ModbusMaster	Modbus主站模块，支持读写多种类型的Modbus寄存器，如线圈、离散输入、保持寄存器和输入寄存器等。	设备
ModbusSlave	Modbus从站模块，支持响应主站的读写请求，并且可以设置从站地址和数据存储区域	设备
ModbusFactory	Modbus工厂模块，用于创建Modbus主站和从站实例。	基础类
ModbusMessage	Modbus消息模块，用于封装和解析Modbus消息，包括Modbus请求和响应消息。	消息类
ModbusTransport	Modbus传输模块，用于处理Modbus消息的传输和接收，包括串口、TCP、UDP等多种传输方式。	输入输出类

NModbus

- https://nmodbus.github.io/api/NModbus.html
- 支持Modbus协议的扩展功能，包括Modbus TCP的Keep-Alive机制和Modbus RTU的帧校验功能
- 支持在TCP连接上启用Keep-Alive机制，以确保连接的稳定性和可靠性。可以通过设置TcpClientAdapter的KeepAlive属性来启用Keep-Alive机制。
- 支持在Modbus RTU通信中启用帧校验功能，以确保数据的正确性和可靠性。可以通过设置SerialPortAdapter的Parity属性来启用帧校验功能。

```
using NModbus;
using NModbus.IO;
using NModbus.Serial;

// 创建TCP传输
IModbusTransport transport = new TcpClientAdapter(ipAddress, port);

// 启用Keep-Alive机制
((TcpClientAdapter)transport).KeepAlive = true;

// 创建Modbus主站
IModbusMaster master = ModbusFactory.CreateMaster(transport);
```

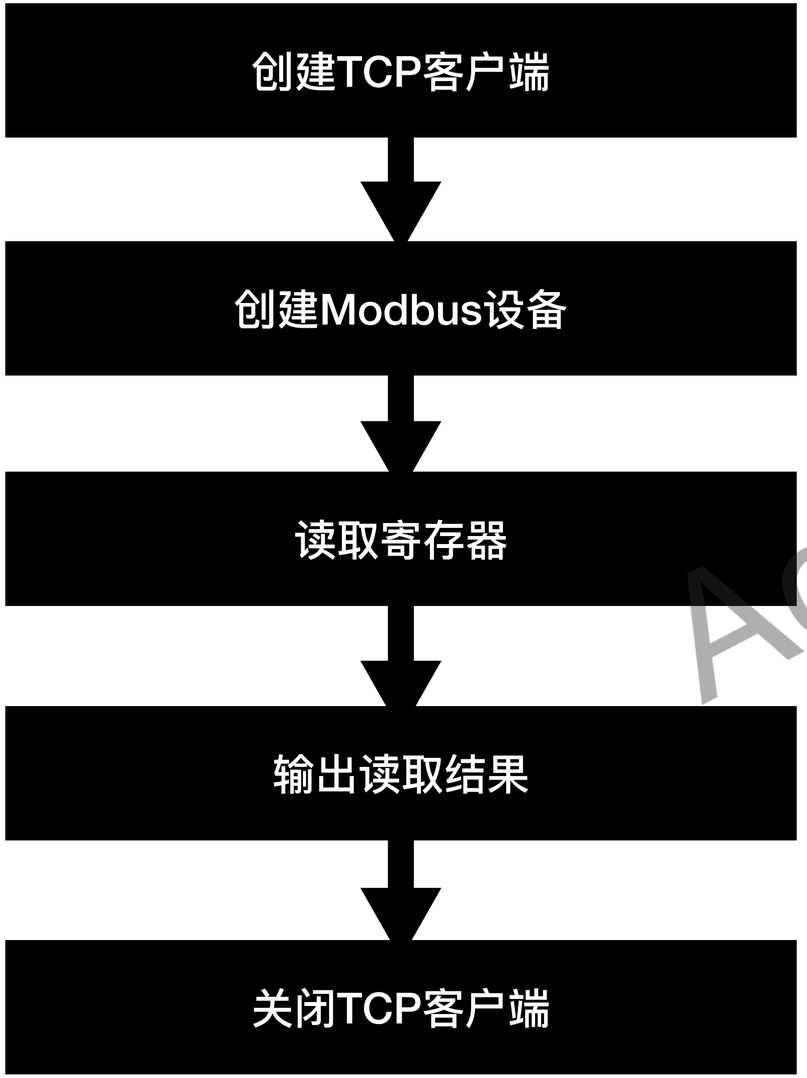
```
using NModbus;
using NModbus.IO;
using NModbus.Serial;

// 创建串口传输
IModbusSerialTransport transport = new SerialPortAdapter(port);

// 启用帧校验功能
((SerialPortAdapter)transport).Parity = Parity.Even;

// 创建Modbus主站
IModbusMaster master = ModbusFactory.CreateMaster(transport);
```

使用NModbus库的样码




```
using System;
using System.Net;
using System.Net.Sockets;
using Modbus.Data;
using Modbus.Device;

namespace ModbusExample
{
    class Program
    {
        static void Main(string[] args)
        {
            // 创建TCP客户端
            TcpClient client = new TcpClient("192.168.1.1", 502);

            // 创建Modbus设备
            ModbusIpMaster modbusDevice = ModbusIpMaster.CreateIp(client);

            // 读取寄存器
            ushort startAddress = 0;
            ushort numRegisters = 10;
            ushort[] registers = modbusDevice.ReadHoldingRegisters(1, startAddress,
numRegisters);

            // 输出读取结果
            for (int i = 0; i < numRegisters; i++)
            {
                Console.WriteLine("Register {0}: {1}", startAddress + i, registers[i]);
            }

            // 关闭TCP客户端
            client.Close();
        }
    }
}
```

3 阅读NModbus样码

NModbus样码

代码名称	主要功能	过程
ModbusSerialRtuMasterWriteRegisters	创建一个RTU主站	配置串行口，适配通讯协议，创建工厂类，创建Modbus主站，写寄存器
StartModbusSerialRtuSlaveNetwork	创建一个从属RTU网络，多个Modbus从属设备可以利用同一个物理网络适配器	配置串行口，创建工厂类实例，适配通讯协议，创建工厂类，创建从属设备，将从属设备加入到网络中，异步监听，等待网络响应
IModbusMasterLogging	如何使用一个简单的facade为IModbusMaster记录日志	该类实现了IModbusMaster接口，并提供了一些基础的Modbus主站操作方法，同时在这些方法的基础上添加了日志记录的功能。
ModbusRegisterScanner	扫描查找哪些寄存器对一个给定的Modbus设备有效，从站寄存器扫描器扫描Modbus从站中的寄存器，读取其中的数据并进行处理	获取串口，创建工厂类实例，打开串口，适配通讯协议，创建主站实例，读取指令地址的寄存器，用捕获异常的方式判断该地址是否存在寄存器，输出所有存在寄存器的地址

阅读代码注意事项

- 通信方式和通信参数：串口、TCP、以太网和通信速率、超时时间、重试次数
- 数据类型：整型、浮点型、布尔型。
- 读写操作：寄存器地址
- 使用的模块：Data/Device/Extensions/IO/Interfaces/Logging/Message/Utility
- 使用的设计模式：Factory，Facade，Adapter
- 如何处理异常：捕获异常、设置超时、重试机制、日志记录。
- 如何处理错误：地址越界、数据类型不匹配，使用错误码提示用户或操作。

4 参考资源

- https://modbus.org/tech.php
- https://www.se.com/us/en/faqs/FA168406/ What is Modbus and How does it work?
- Headfirst Design Pattern 英文中文版
- https://github.com/NModbus/NModbus

总结

1 前导知识

- 1.1 客户端和服务器
- 1.2 设计模式：Factory、Facade和Adapter
- 1.3 通讯协议：Modbus

2 NModbus

3 阅读NModbus样码

4 参考资料

下一次，
Modbus的安全性和可靠性