jingtum-lib-csharp使用说明

V1.0

本文档介绍jingtum-lib在.net平台上的如何使用，另有接口的详细文档。

# 安装

通过nuget安装。

# 项目文件结构

jingtum-lib库基于websocket协议跟底层交互，其中websocket封装到Server类中，Server类是一个内部类；Server类封装在Remote类中，Remote类提供对外接口并可创建两类对象：Get方式请求的Request对象和Post方式请求的Transaction对象，这两类对象都通过submit()方法提交数据到底层。文件结构图如下：

Remote类

Request对象（Get请求）

WebSocket

通讯协议

Server

内部类

Transaction对象（Post请求）

# 创建钱包

首先添加jingtum-lib引用到项目中，代码中添加using

usingJingTum.Lib;

方法一：

public static WalletWallet.Generate()

方法二：

public static Wallet Wallet.FromSecret(string secret)

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | string | 井通钱包私钥 |

# Remote类

Remote是跟井通底层交互最主要的类，它可以组装交易发送到底层、订阅事件及从底层拉取数据。其主要方法如下。

## 创建Remote类

通过构造函数传入井通底层服务地址并制定是否本地签名来创建Remote类。

var remote= new Remote("ws://ts5.jingtum.com:5020", true);

## 创建连接

每个Remote对象都应该首先通过Connect方法手动连接底层，然后才可以请求底层数据。请求结果在回调函数的callback中。

代码示例：

var remote= new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if(result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

Console.Write(result.Message);

}

});

## 关闭连接

每个Remote对象可以手动通过Disconnect方法关闭连接。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if(result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

remote.Disconnect(); //关闭连接

}

});

## 请求底层服务器信息

首先通过RequestServerInfo方法返回一个Request对象，然后通过Submit方法获得井通底层的服务器信息，包含服务程序版本号Version、该服务器缓存的账本区间Ledgers、节点公钥Node、服务器当前状态State。其中服务器当前状态包含可提供服务状态full和验证节点状态proposing。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if(result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

var req = remote.RequestServerInfo();

req.Submit(reqResult =>

{

var info = reqResult.Result.Info;

Console.Write($"HostId:{info.HostId},Ledgers:{info.Ledgers},Node:{info.Node},State:{info.State},Version:{info.Version}");

});

}

});

## 获取最新账本信息

首先通过RequestLedgetClosed方法返回一个Request对象，然后通过Submit方法获得最新账本信息，包括区块高度(LedgerIndex)与区块hash(LedgerHash)。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if(result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

var req = remote.RequestLedgerClosed();

req.Submit(reqResult =>

{

var info = reqResult.Result;

Console.Write($"LedgerIndex:{info.LedgerIndex},LedgerHash:{info.LedgerHash}");

});

}

});

## 获取某一账本具体信息

首先通过RequestLedger方法返回一个Request对象，然后通过Submit方法获得某一账本的具体信息。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if(result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

var req = remote.RequestLedger(new LedgerOptions{ LedgerIndex = 8488670, Transactions = true});

req.Submit(reqResult =>

{

LedgerResponse info = reqResult.Result;

//账本信息在info变量中

});

}

});

## 查询某一交易具体信息

首先通过RequestTx方法返回一个Request对象，然后通过Submit方法获得某一交易的具体信息。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if (result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

var req = remote.RequestTx(new TxOptions { Hash = "084C7823C318B8921A362E39C67A6FB15ADA5BCCD0C7E9A3B13485B1EF2A4313" });

req.Submit(reqResult =>

{

TxResponse info = reqResult.Result;

//交易信息在info变量中

});

}

});

## 请求账号信息

首先通过RequestAccountInfo方法返回一个Request对象，然后通过Submit方法获得某一账号的交易信息。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if (result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

var req = remote.RequestAccountInfo(new AccountInfoOptions { Account = "jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" });

req.Submit(reqResult =>

{

AccountInfoResponse info = reqResult.Result;

//账号信息在info变量中

});

}

});

## 获得账号可接收和发送的货币

首先通过RequestAccountTums方法返回一个Request对象，然后通过Submit方法获得某一账号可发送和接收的货币种类。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if (result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

var req = remote.RequestAccountTums(new AccountTumsOptions { Account = "jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" });

req.Submit(reqResult =>

{

AccountTumsResponse info = reqResult.Result;

//可发送和接收的货币种类信息在info变量中

});

}

});

## 获得账号关系

井通账户之间会建立各种不同的关系。这些关系由井通后台的关系（relations）机制来处理，

目前支持以下关系：信任(trust)、授权(authorize)、冻结(freeze)。

首先通过RequestAccountRelations方法返回一个Request对象，然后通过Submit方法获得某一账号指定关系的信息。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if (result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

var req = remote.RequestAccountRelations(new AccountRelationsOptions

{

Account = "jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ",

Type = RelationType.Trust

});

req.Submit(reqResult =>

{

AccountRelationsResponse info = reqResult.Result;

//账号关系信息在info变量中

});

}

});

## 获得账号挂单

首先通过RequestAccountOffers方法返回一个Request对象，然后通过Submit方法获得某一账号的挂单信息。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if (result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

var req = remote.RequestAccountOffers(new AccountOffersOptions

{

Account = "jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ"

});

req.Submit(reqResult =>

{

AccountOffersResponse info = reqResult.Result;

//账号挂单信息在info变量中

});

}

});

## 获得账号交易列表

首先通过RequestAccountTx方法返回一个Request对象，然后通过Submit方法获得某一账号的交易列表信息。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if (result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

var req = remote.RequestAccountTx(new AccountTxOptions

{

Account = "jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ"

});

req.Submit(reqResult =>

{

AccountTxResponse info = reqResult.Result;

//账号交易列表信息在info变量中

});

}

});

## 获得市场挂单列表

首先通过RequestOrderBook方法返回一个Request对象，然后通过Submit方法获得市场挂单列表信息。

代码示例：

var remote = new Remote(@"ws://ts5.jingtum.com:5020", true);

remote.Connect(result =>

{

if (result.Exception != null)

{

Console.Write(result.Exception.Message);

}

else

{

var req = remote.RequestOrderBook(new OrderBookOptions

{

Gets = new AmountSettings("SWT", ""),

Pays = new AmountSettings("CNY", "jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS")

});

req.Submit(reqResult =>

{

OrderBookResponse info = reqResult.Result;

//市场挂单列表信息在info变量中

});

}

});

# Request类

# Transaction类