Задание 1. Implementing a multi-client echo server

Начало: 16 января 2017 года  
Конец: 22 января 2017 года (23:59:59)

# Цель задания

The goal of this assignment is to get up to speed on the C# programming language and to help remind you about the basics of TCP socket programming. In this assignment you will implement a simple multi-client echo server in C#: every message sent by a client should be echoed to all connected clients.

# Описание задания

#### Характеристики сервера

Your multi-client echo server must have the following characteristics:

1. The server must manage and interact with its clients concurrently using TCP channels. Multiple clients should be able to connect/disconnect to the server simultaneously.
2. The server should assume that all messages are line-oriented, separated by newline (\n) characters. When the server reads a newline-terminated message from a client, it must respond by writing that exact message (up to and including the newline character) to all connected clients, including the client that sent the message.
3. The server must be responsive to slow-reading clients. To better understand what this means, consider a scenario in which a client does not call Read for an extended period of time. If during this time the server continues to write messages to the client's TCP connection, eventually the TCP connection's output buffer will reach maximum capacity and subsequent calls to Write made by the server will block.

To handle these cases, your server should keep a queue of at most 100 outgoing messages to be written to the client at a later time. Messages sent to a slow-reading client whose outgoing message buffer has reached the maximum capacity of 100 should simply be dropped. If the slow-reading client starts reading again later on, the server should make sure to write any buffered messages in its queue back to the client.(Hint: use a buffered channel to implement this property).

#### Требования к программе

This project is intentionally open-ended and there are many possible solutions. That said, your implementation must meet the following four requirements:

* The project must be done individually. You are not allowed to use any code that you have not written yourself.
* All synchronization must be done using TCP Socket channels.
* You may use the following components –
* System.Net.Sockets.TcpListener,
* System.Net.Sockets.TcpClient.
* You must format your code using Code Rules Document.
* You must put Client and Server projects in one Visual Studio solution. If you have other projects and/or libraries source codes, please add it to solution as projects. All of your source code must be in ONE solution.

We don't expect your solutions to be overly-complicated. As a reference, solution is a little over 200-250 lines including sparse comments and whitespace. We do, however, highly recommend that you familiarize yourself with C# concurrency model before beginning the assignment. For additional resources, check out online resources and books (see. Additional Material).

## Оценка

|  |  |
| --- | --- |
| Реализация клиент-серверной модели | **10 баллов** |

## Оформление

Вашу программу следует размести в Git (Желательно создать учетную Visual Studio Online (VSTS). Должно быть минимум два модуля правильно оформленных (Client и Server). Логика должна находится в отдельных классах и быть тестируемой. На вход программе должна быть возможность подавать данные с консоли. Клиенты должны писать все сообщения в лог файл (должно быт можно указать в командной строке), в том виде котором они приходят от сервера.

Весь код должен быть покрыт модульными тестами. Ссылку на доступ к репозиторию можно прислать вместо архива.

Структура:

**src** – исходники, без каких либо бинарных файлов

**lib** – дополнительные библиотеки, которые используются в вашем решении (кроме стандартных). Если пользуетесь **.nuget**, то заливать библиотеки не надо

**bin** – в этой директории должен хранится zip архив с вашими исполняемыми файлами и библиотеками. После распаковки архива, программа должна запускаться без дополнительных действий. Если есть особенности, указывайте их в readme.txt

Не забудьте положить в архив файл **readme.txt**. В файле описать интерфейс программы (алгоритм работы с программой, пункты меню, управляющие клавиши)

!Внимание: обратите внимание на правильное именование и структуру работы, а также на содержание файла readme.txt. Если ваша работа не будет соответствовать требованиям, баллы могут быть снижены. Время сдачи работы оценивается по последнему комиту. Если у вас есть модификации к работе после последнего срока – делайте branch и заливайте изменения туда.

# Результаты работы

Результаты смотрите в интернете. Совместная работа или обмен кусками кода, при условии что факт командной работы не был указан в readme.txt, считается плагиатом и оценивается в -5 баллов всем участникам.

# Дополнительный материал

* <https://msdn.microsoft.com/en-us/library/system.net.sockets%28v=vs.110%29.aspx>
* <http://www.java2s.com/Code/CSharp/Network/SimpleTcpServer.htm>
* <http://www.codeproject.com/Articles/13232/A-very-basic-TCP-server-written-in-C>
* <http://www.codeproject.com/Articles/488668/Csharp-TCP-Server>