Implement Your AppServer and AppSession

English (United States) v1.6

Keywords: AppServer, AppSession

What is AppSession?

AppSession represents a logic socket connection, connection based operations should be defined in this class. You can use the instance of this class to send data to tcp clients, receive data from connection or close the connection.

What is AppServer?

AppServer stands for the server instance which listens all clients' connections, hosts all tcp connections. Ideally, we can get any session which we want to find from the AppServer. Application level operations and logics should be defined in it.

Create your AppSession

1. You can override base AppSession's operations

```
public class TelnetSession : AppSession<TelnetSession>
{
    protected override void OnSessionStarted()
    {
        this.Send("Welcome to SuperSocket Telnet Server");
    }

    protected override void HandleUnknownRequest(StringRequestInfo requestInfo)
    {
        this.Send("Unknow request");
    }

    protected override void HandleException(Exception e)
    {
        this.Send("Application error: {0}", e.Message);
    }

    protected override void OnSessionClosed(CloseReason reason)
    {
        //add you logics which will be executed after the session is closed
        base.OnSessionClosed(reason);
    }
}
```

   In above code, the server send the welcome message to the client immediately after the session is connected. The code also overrided the other methods of AppSession to process it's own logic.

2. You can add new properties for your session according your business requirement Let me create a AppSession which would be used in a game server:

```
public class PlayerSession : AppSession<PlayerSession>
{
    public int GameHallId { get; internal set; }

    public int RoomId { get; internal set; }
}
```

3. Relationship with Commands

   In the first document, we talked about commands, now we revise this point here:

```
public class ECHO : CommandBase<AppSession, StringRequestInfo>
{
    public override void ExecuteCommand(AppSession session, StringRequestInfo requestInfo)
    {
        session.Send(requestInfo.Body);
    }
}
```

   In the command's code, you should have found the parent class pf ECHO is CommandBase<AppSession, StringRequestInfo>, which has a generic type parameter AppSession. Yes, it's the default AppSession. If you want to use your new AppSession, please pass your AppSession type as the parameter, or the server cannot discover the command:

```
public class ECHO : CommandBase<PlayerSession, StringRequestInfo>
{
    public override void ExecuteCommand(PlayerSession session, StringRequestInfo requestInfo)
    {
        session.Send(requestInfo.Body);
    }
}
```

Create your AppServer

1. Work with Session If you want to make available your AppSession, you must alter your AppServer to use it:

   ```
   public class TelnetServer : AppServer<TelnetSession>
   {

   }
   ```

   Then the session TelnetSession will can be used in TelnetServer.

2. There are also many protected methods you can override

   ```
   public class TelnetServer : AppServer<TelnetSession>
   {
       protected override bool Setup(IRootConfig rootConfig, IServerConfig config)
       {
           return base.Setup(rootConfig, config);
       }

       protected override void OnStartup()
       {
           base.OnStartup();
       }

       protected override void OnStopped()
       {
           base.OnStopped();
       }
   }
   ```

Benifits

Imeplement your AppSession and AppServer allow you extend SuperSocket as your business requirement, you can hook session's connected and closed event, server instance's startup and stopped event. You can read your own customized configuration in AppServer's Setup() method. In summary it give you lots of abilities to build a socket server which is exact what you want very easily.

- Prev: A Telnet Example (/v1-6/en-US/A-Telnet-Example)
- Next: Start SuperSocket by Configuration (/v1-6/en-US/Start-SuperSocket-by-Configuration)