

Logging in SuperSocket

English (United States)

v1.6

Keywords: Logging, log4net, Logging API, Logging Customization, LogFactory

customization : 주문에 따라 만들

The logging system in SuperSocket

The logging system is enabled automatically when the SuperSocket bootstrap is starting, so you needn't create your own logging tool by yourself. You'd better to use the logging function in SuperSocket.

By default, the SuperSocket uses log4net as it's logging framework. So if you are familiar with log4net, it will be very easy for you to use and customize the logging function in SuperSocket.

SuperSocket also provides the basic log4net configuration files log4net.config/log4net.unix.config, you should put the log configuration file into the sub directory "Config" of the root of the running application. The log4net config define the loggers and appenders to category all logs into the 4 rolling files in the folder named "Logs":

- info.log
- debug.log
- err.log
- perf.log

You also can customize the config according your logging requirements.

Because of the loose couple with the log4net, you need to reference the file log4net.dll manually by yourself (please use the one provided by SuperSocket).

The logging API

The logging in SuperSocket is very easy, you can log information in the most places of your code. Both of the basic class of AppServer and AppSession have the Logger property which can be used directly for logging.

The code below demonstrates the logging API:

A -

```

/// <summary>
/// PolicyServer base class
/// </summary>
public abstract class PolicyServer : AppServer<PolicySession, BinaryRequestInfo>
{
    .....

    /// <summary>
    /// Setups the specified root config.
    /// </summary>
    /// <param name="rootConfig">The root config.</param>
    /// <param name="config">The config.</param>
    /// <returns></returns>
    protected override bool Setup(IRootConfig rootConfig, IServerConfig config)
    {
        m_PolicyFile = config.Options.GetValue("policyFile");

        if (string.IsNullOrEmpty(m_PolicyFile))
        {
            if (Logger.IsErrorEnabled)
                Logger.Error("Configuration option policyFile is required!");
            return false;
        }

        return true;
    }

    .....
}

```

B -

```

public class RemoteProcessSession : AppSession<RemoteProcessSession>
{
    protected override void HandleUnknownRequest(StringRequestInfo requestInfo)
    {
        Logger.Error("Unknow request");
    }
}

```

Extend your logger

SuperSocket allow you to customize your logger by yourself. For example, if you want to save your business operations log into another file than the default SuperSocket logging files, then you can define a new logger in your log4net configuration (assume you are using log4net by default):

```
<appender name="myBusinessAppender">
  <!--Your appender details-->
</appender>
<logger name="MyBusiness" additivity="false">
  <level value="ALL" />
  <appender-ref ref="myBusinessAppender" />
</logger>
```

and then create this logger instance in your code:

```
var myLogger = server.LogFactory.GetLog("MyBusiness");
```

Use other logging framework than log4net

SuperSocket supports you implement your own log factory from the interface:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace SuperSocket.SocketBase.Logging
{
    /// <summary>
    /// LogFactory Interface
    /// </summary>
    public interface ILogFactory
    {
        /// <summary>
        /// Gets the log by name.
        /// </summary>
        /// <param name="name">The name.</param>
        /// <returns></returns>
        ILog GetLog(string name);
    }
}
```



The interfaces ILogFactory and ILog are defined in SuperSocket.

After you implement your own log factory, then you can enable it in the configuration:

```
<superSocket logFactory="ConsoleLogFactory">
  <servers>
    <server name="EchoServer" serverTypeName="EchoService">
      <listeners>
        <add ip="Any" port="80" />
      </listeners>
    </server>
  </servers>
  <serverTypes>
    <add name="EchoService"
      type="SuperSocket.QuickStart.EchoService.EchoServer, SuperSocket.QuickStart.EchoService" />
  </serverTypes>
  <logFactories>
    <add name="ConsoleLogFactory"
      type="SuperSocket.SocketBase.Logging.ConsoleLogFactory, SuperSocket.SocketBase" />
  </logFactories>
</superSocket>
```

There is a log factory implemented for **Enterprise Library Logging Application Block** in SuperSocket Extensions: <http://supersockettext.codeplex.com/> (<http://supersockettext.codeplex.com/>)

- Prev: Implement Your Commands by Dynamic Language ([/v1-6/en-US/Implement-Your-Commands-by-Dynamic-Language](#))
- Next: The Built in Flash Silverlight Policy Server in SuperSocket ([/v1-6/en-US/The-Built-in-Flash-Silverlight-Policy-Server-in-SuperSocket](#))

20개의 웹사이트를 직접 만들면서 배우는 웹 개발

실무중심의 웹 프론트엔드 코딩 인강

