NJIS.ConfigurationCenter.Client 使用文档

目录

启动器(ConfigurationCenterStarter)

1.ConfigurationCenterStarter.SetLog

```
//功能: 更改日志存放方式
//使用方式:
public class Log : ILog
{
    public void Error(string errorText)
    {
        throw new NotImplementedException();
    }
    public void Error(Exception e)
    {
        throw new NotImplementedException();
    }
    public void Info(string InfoText)
    {
        throw new NotImplementedException();
    }
}
ConfigurationCenterStarter.SetLog(new Log()) //在调用 Setup.Start()之前
```

2.ConfigurationCenterStarter.SetLocalCofig

```
//功能:更改配置存放方式
//使用方式:
//Tuple<string, string, T> item1:配置路径 item2:版本号 item3:配置实体对象
public class ExtendLocalConfig : ILocalConfig
{
    public Tuple<string, string, T> Get<T>(string configAddress)
    {
        throw new NotImplementedException();
    }
```

```
        public void Save<T>(string configAddress, string version, T configObject)
        {
            throw new NotImplementedException();
        }
    }
    ConfigurationCenterStarter.SetLocalCofig(new ExtendLocalConfig());
```

3.ConfigurationCenterStarter.Start()

```
    //1.通过调用Start() 开启远程连接
    //功能：启动入口 启动远程socket连接 全局只能调用一次
    //方法重载
    //1.Start() 使用配置启动
    //配置:
    <appSettings>
        <add key="ConfigServerIpAddress" value="10.10.14.55"/>
        <add key="ConfigServerPort" value="9999"/>
        <add key="ConfigServerAppName" value="例子演示"/>
    </appSettings>
    //当第一次连接成功之后获取该远程应用的Tcp连接配置并保存到本地appSetting中可以把这些配置删除
    //Start(string serviceUrl, int port) 不使用配置文件启动

    //2.给定配置\Configs\RemoteConfigCenterTcpUrl.json 或则 AppSetting  配置
    // 可以不需要调用Start()  ConfigurationClient会自动连接到远程电脑
    // 退出的时候也可以不用调用Stop() ConfigurationClient内部进行轮询主进程是否已经死掉
    // 发现主进程死掉那么就会释放所有资源
```

4. ConfigurationCenterStarter.Stop()

```
    //功能：关闭入口 关闭远程socket连接 全局只能调用一次
    //方法重载
    //1.Stop()
```

配置获取入口(ConfigEntry)

1.Get 通过配置路径获取配置

```
    //configAddress:配置路径
    //timeout:一次请求超时时间
    //isSave:是否持久化本地  这里调用ILocalConfig接口
    //isMonitor:是否监听 当isMonitor== true时，必须注册ConfigEntry.Changed事件
    //return:配置实体对象
```

```
    T Get<T>(string configAddress,
            int timeOut = 10 * 1000,
            bool isSave = true,
            bool isMonitor = false);
```

2.GetAllApplicationConfigAddress 应用下配置黄页

```
    //applicationName:应用程序名称
    //timeout:一次请求超时时间
    //return:该应用下的所有配置路径
    List<string> GetAllApplicationConfigAddress(string applicationName, int
timeOut = 10 * 1000)
```

3.ConfigMonitor 配置监控

```
    //configAddress:配置路径
    // Get 配置的时候 isMonitor 必须设置为true
    //使用配置监听的时候必须注册ConfigEntry.Changed事件
    void ConfigMonitor(string configAddress)
```

4.ConfigEntry.Changed事件 远程配置更改后触发该事件

```
    //configAddress:配置路径
    //使用配置监听的时候必须注册ConfigEntry.Changed事件
    ConfigEntry.Changed += ConfigEntry_Changed;
    //sender:配置路径
    //e:版本号(目前为guid)
    private void ConfigEntry_Changed(object sender, string e)
    {
    }
```

配置单例化

1.非数组，基于**ConfigBase<T>**

```
    类名:class ConfigBase<T> 必须加上特性ConfigCenterAttribute
```

2.非数组的使用方式

2.1.这里支持两种配置方式:

第一种需要远程配置中心获取配置

```
        // 这里有远程监控属性 默认为IsMonitor=false  设置为IsMonitor=true的时候远程更改数据
        // 会通知本地，并保存下来
        [ConfigCenter("测试应用/20180426/测试")]
        public class Class1 : ConfigBase<Class1>
        {
            public Int16 Art1 { get; set; }
            ///IgnoreAutoUpdate特性就是在IsMonitor=true的情况下
            ///远程配置更改通知不对该字段进行内存级别更新(但是也会持久化到本地)
            [IgnoreAutoUpdate]
            public UInt32 Art2 { get; set; }
            public Int64 Art11 { get; set; }
        }
        //[ConfigCenter("")] 中的地址为空或则为null 那么必须重写ConfigAddress
        [ConfigCenter("")]
        public class Class1 : ConfigBase<Class1>
        {
            public Int16 Art1 { get; set; }
            ///IgnoreAutoUpdate特性就是在IsMonitor=true的情况下
            ///远程配置更改通知不对该字段进行内存级别更新(但是也会持久化到本地)
            [IgnoreAutoUpdate]
            public UInt32 Art2 { get; set; }
            public Int64 Art11 { get; set; }
            public override string Address
            {
                get{return "测试应用/20180426/测试"; }
            }
        }
        public void TestMethod2()
        {

            Client.ConfigurationCenterStarter.Start();

            ConfigEntry.Changed += ConfigEntry_Changed;

            while (true)
            {

                Class1 c = Class1.Current;//可以拿到相应的配置  属于配置单例化
                Thread.Sleep(1000);
            }
        }
```

第二种本地配置文件获取配置(配置中心不参与，完全离线状态)

```
        [ConfigCenter("本地文件配置地址",true)] // true代表使用要使用离线配置管理
        public class Class1 : ConfigBase<Class1>
```

```
{
    public Int16 Art1 { get; set; }
    public UInt32 Art2 { get; set; }
    public Int64 Art11 { get; set; }
}

public void TestMethod2()
{
    while (true)
    {

        Class1 c = Class1.Current;//可以拿到相应的配置  属于配置单例化
        Thread.Sleep(1000);
    }
}
```

3.基于数组,基类:ConfigsBase<T>

```
///
/// 不支持使用IsLocal = true的功能
///
[ConfigCenter("例子演示/ListTest", IsMonitor = true)]
public class ListClass2 : ConfigsBase<ListClass2>
{
    public Int32 Pro1 { get; set; }
    public Int32 Pro2 { get; set; }
}
```

4. 数组的使用方式

```
///
/// 通过数组下标来获取Item
///
var c = ListClass2.Current[0].Pro1;
///
///   配置中心的例子请看 配置中心的应用<<例子演示>>中的ListTest节点
///
///
```

实战

```
//启动Socket连接
ConfigurationCenterStarter.Start();

//获取远程配置
```

```
    //只有在 isMonitor:true 的时候，该配置路径才会启动监听
    AppSetting obj = ConfigEntry.Get<AppSetting>("例子演示/newnode1", isMonitor:
true);

    //注册事件
    ConfigEntry.Changed += ConfigEntry_Changed;
    private void ConfigEntry_Changed(string sender, string e)
    {
    }

    //断开连接
    ConfigurationCenterStarter.Stop();

    //实体类
    public class AppSetting
    {
        public string Name { get; set; }
        public string Sex { get; set; }
        public Class1 Attr391 { get; set; }
    }

    public class Class1
    {
        public Int64 Tetete { get; set; }
    }

    //远程配置示例:与实体类的映射关系
    // http://10.10.14.55:8889/ConfigDetail?
configAddress=%E4%BE%8B%E5%AD%90%E6%BC%94%E7%A4%BA/newnode1
```

链接

1.nuget 包(宁基智能信息化内部nuget地址:http://10.10.14.46:8070/nuget)

Nuget包名称:NJIS.ConfigurationCenter.Client

2. 配置中心Web端地址:

```
http://10.10.14.55:8889/
```

3.配置中心Socket服务器地址:

```
Ip地址:10.10.14.55
端口号: 9999
主要由配置中心客户端使用
```

维护

1. svn目录

https://nji-svn.sogal.org:2016/svn/05_Information_Technology_Department/IT_Workshop/trunk/NJIS.ConfigurationCenter

2.bug、用户体验提交**(jira**路径**)**

http://10.10.14.46:8900　项目名称**:MES-Factory4.0**

3.联系人

> 宁基智能信息技术部
> 11002390 庄名杰
> 邮箱:mingjie.zhuang@sfygroup.com