# 配置文件配置

## 指定程序集版本

`App.config` 文件配置 `configuration` 节点下添加 `runtime` 节点

参考链接https://docs.microsoft.com/zh-cn/dotnet/framework/configure-apps/file-schema/runtime/assemblybinding-element-for-runtime

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="mscorlib" publicKeyToken="b77a5c561934e089"
culture="neutral" />
        <bindingRedirect oldVersion="4.0.0.0" newVersion="2.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

## 配置私有目录(可以将`dll`分类放在根目录的子目录)

参考链接https://docs.microsoft.com/zh-cn/dotnet/standard/assembly/location

https://docs.microsoft.com/zh-cn/dotnet/framework/configure-apps/file-schema/runtime/probing-element#example

`App.config` 文件配置 `configuration` 节点下添加 `runtime` 节点下增加 `probing` 标签,文件夹之间用分号隔开

运行时默认搜索 `DLL` 顺序

- 应用程序的目录或子目录。

  这是部署程序集最常用的位置。 应用程序根目录的子目录可以基于语言或区域性。 如果程序集具有 culture 特性中的信息，则它必须位于带有该区域性名称的应用程序目录下的子目录中。

- 全局程序集缓存。(`GAC`)

  这是安装于公共语言运行时安装位置的计算机范围内的代码缓存。 大多数情况下，如果要与多个应用程序共享程序集，应将程序集部署到全局程序集缓存中。

- 在 HTTP 服务器上。

  部署在 HTTP 服务器上的程序集必须具有强名称，请在应用程序配置文件的基本代码节中指向此程序集

```
<runtime>
    <assemblyBinding
        xmlns="urn:schemas-microsoft-com:asm.v1">
        <probing
privatePath="CodeMng;ControlParamMng;OrganizationMng;PermissionMng;AutoUpdate;Schedu
ling;I18N;License;NumGen;Core;DawnReport;ApplySheet;Interface;Oncology;Order;OrderTe
rm;WorkStationNew;Plan;TaskList;BeyondMDC;Customer;Patient;TreatmentHistory;RealoneW
orkStationInpatient;RealOneWorkStation;EmrQC;
Vte;VteReport;Cdss;CDSS_App;EmrCore2.0;"/>
        <dependentAssembly>
            <assemblyIdentity name="Newtonsoft.Json" culture="neutral"
publicKeyToken="30ad4fe6b2a6aeed" />
            <bindingRedirect oldVersion="0.0.0.0-12.0.0.0" newVersion="11.0.0.0" />
        </dependentAssembly>
    </assemblyBinding>
</runtime>
```

# 配置文件读取

参考链接https://www.cnblogs.com/kissdodog/archive/2013/04/11/3014227.html

## App.config文件配置

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="he_zhw" value="he_zhw@neusoft.com"/>
  </appSettings>
  <connectionStrings>
    <add name="connectStr" providerName="MySql.Data.MySqlClient"
connectionString="User
Id=root;Password=123;server=192.168.72.128;port=3306;user=root;password=246;database
=bookstore"></add>
  </connectionStrings>
</configuration>
```

## 读取appSettings

引入`System.Configuration.dll`通过属性
`System.Configuration.ConfigurationManager.AppSettings`用key值获取value

```
static void Main(string[] args)
{
    string valie =
System.Configuration.ConfigurationManager.AppSettings["he_zhw"].ToString();//根据key
值获取value值
    Console.WriteLine(valie);
    Console.ReadLine();
}
//输出结果
/*
he_zhw@neusoft.com
*/
```

## 读取`connectionStrings`节点

```
static void Main(string[] args)
{
    string providerName =
System.Configuration.ConfigurationManager.ConnectionStrings["connectStr"].ProviderNa
me;//根据name值获取ProviderName的值
    string connectionString =
System.Configuration.ConfigurationManager.ConnectionStrings["connectStr"].Connection
String;//根据name值获取ConnectionString的值
    Console.WriteLine($"providerName={providerName}\r\nconnectionString=
{connectionString}");
    Console.ReadLine();
}
//输出结果
/*
providerName=MySql.Data.MySqlClient
connectionString=User
Id=root;Password=123;server=192.168.72.128;port=3306;user=root;password=246;database
=bookstore
*/
```

## 自带Handler获取配置

需引用命名空间`using System.Collections.Specialized;`

- `System.Configuration.NameValueSectionHandler`--以`NameValue`键值对的形式返回配置节中的信息
- `System.Configuration.DictionarySectionHandler`--以Dictionary字典键值对的形式返回配置节中的信息
- `System.Configuration.SingleTagSectionHandler`--基础结构。处理`.config`文件中由单个 XML 标记所表示的各配置节。

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
```

```
  <configSections>
    <section name="Person" type="System.Configuration.NameValueSectionHandler"/> <!--以NameValue键值/对的形式返回配置节中的信息-->
    <section name="Man" type="System.Configuration.DictionarySectionHandler"/>  <!--以Dictionary字典键值对的形式返回配置节中的信息-->
    <section name="Name" type="System.Configuration.SingleTagSectionHandler" /> <!--基础结构。处理 .config 文件中由单个 XML 标记所表示的各配置节。-->
  </configSections>
  <Person>
    <add key="老大" value="刘备" />
    <add key="老二" value="关羽" />
    <add key="老三" value="张飞" />
  </Person>

  <Man>
    <add key="老大" value="曹操" />
    <add key="老二" value="典韦" />
    <add key="老三" value="郭嘉" />
  </Man>

  <Name one="1" two="2" three="3" four="4" five="5" /> <!--注意是要单个节SingleTagSectionHandler才能处理，但是无论你索性有多少个也能处理-->
</configuration>
```

## System.Configuration.NameValueSectionHandler获取

```
static void Main(string[] args)
{
    NameValueCollection nameValueCollection = (NameValueCollection)System.Configuration.ConfigurationManager.GetSection("Person");
    foreach (var key in nameValueCollection.AllKeys)
    {
        Console.WriteLine($"key={key} value={nameValueCollection[key]}");
    }
    Console.ReadLine();
}
//输出结果
/*
key=老大 value=刘备
key=老二 value=关羽
key=老三 value=张飞
*/
```

## System.Configuration.DictionarySectionHandler获取

```
static void Main(string[] args)
{
    var dictionary = (System.Collections.IDictionary)ConfigurationManager.GetSection("Man");
    foreach (string key in dictionary.Keys)
```

```
    {
        Console.WriteLine($"key={key} value={dictionary[key]}");
    }
    Console.ReadLine();
}
//输出结果
/*
key=老二 value=典韦
key=老三 value=郭嘉
key=老大 value=曹操
*/
```

## `System.Configuration.SingleTagSectionHandler`获取

```
static void Main(string[] args)
{
    var dictionary =
(System.Collections.IDictionary)ConfigurationManager.GetSection("Name");
    foreach (string key in dictionary.Keys)
    {
        Console.WriteLine($"key={key} value={dictionary[key]}");
    }
    Console.ReadLine();
}
```

## property属性的方式读取

- 自定义一个类继承`System.Configuration.ConfigurationSection`

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

namespace ConfigurationDemo
{
    public class PersonSection : ConfigurationSection
    {
        [ConfigurationProperty("age", IsRequired = false, DefaultValue = 0)]
        public int Age
        {
            get { return (int)base["age"]; }
            set { base["age"] = value; }
        }

        [ConfigurationProperty("name", IsRequired = false, DefaultValue = "")]
        public string Name
        {
            get { return (string)base["name"]; }
```

```
            set { base["name"] = value; }
        }
    }
}
```

- 配置文件格式

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <!--后面的type是处理处理节点PersonSection所在的位置第二个参数是程序集，你可以不要
Version开始之后的-->
    <section name="Person"
type="ConfigurationDemo.PersonSection,ConfigurationDemo,Version=1.0.0.0,Culture=
neutral,PublicKeyToken=null" allowLocation="true" allowDefinition="Everywhere"
/>
  </configSections>
  <Person age="23" name="刘备" />
</configuration>
```

- 代码调用

```
static void Main(string[] args)
{
    PersonSection personSection = ConfigurationManager.GetSection("Person") as
PersonSection;
    Console.WriteLine($"age={personSection.Age} name={personSection.Name}");
    Console.ReadLine();
}
```

# property属性的方式读取-配置子元素

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="complex"
type="ConfigurationDemo.ComplexSection,ConfigurationDemo"/>
  </configSections>
  <complex height="182">
    <child firstName="张" lastName="飞"/>
  </complex>
</configuration>
```

子元素的类型 `ChildSection` 需要继承 `System.Configuration.ConfigurationElement`

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

```
using System.Text;

namespace ConfigurationDemo
{
    public class ComplexSection : ConfigurationSection
    {
        [ConfigurationProperty("height", IsRequired = true)]
        public int Height
        {
            get { return (int)base["height"]; }
            set { base["height"] = value; }
        }

        [ConfigurationProperty("child", IsDefaultCollection = false)]
        public ChildSection Child
        {
            get { return (ChildSection)base["child"]; }
            set { base["child"] = value; }
        }
    }

    public class ChildSection : ConfigurationElement
    {
        [ConfigurationProperty("firstName", IsRequired = true, IsKey = true)]
        public string FirstName
        {
            get { return (string)base["firstName"]; }
            set { base["firstName"] = value; }
        }

        [ConfigurationProperty("lastName", IsRequired = true)]
        public string LastName
        {
            get { return (string)base["lastName"]; }
            set { base["lastName"] = value; }
        }
    }
}
```

调用例子

```
static void Main(string[] args)
{
            ComplexSection complexSection =
ConfigurationManager.GetSection("complex") as ComplexSection;
            Console.WriteLine($"Height={complexSection.Height} FirstName=
{complexSection.Child.FirstName} LastName={complexSection.Child.LastName}");
            Console.ReadLine();
}
```

# 读取配置文件中CDATA里的内容

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="MySection" type="ConfigurationDemo.MySection, ConfigurationDemo"
/>
  </configSections>
  <MySection>
    <HTML>
      <![CDATA[
          <div style="#background-color:#000; font-size:24px">加粗显示</div>
      ]]>
    </HTML>
    <SQL>
      <![CDATA[
          SELECT TOP 10 * FROM Person
      ]]>
    </SQL>
  </MySection>
</configuration>
```

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

namespace ConfigurationDemo
{
    public class MySection : System.Configuration.ConfigurationSection
    {
        [ConfigurationProperty("HTML", IsRequired = false)]
        public MyTextElement HTML
        {
            get { return (MyTextElement)base["HTML"]; }
            set { base["HTML"] = value; }
        }

        [ConfigurationProperty("SQL", IsRequired = false)]
        public MyTextElement SQL
        {
            get { return (MyTextElement)base["SQL"]; }
            set { base["SQL"] = value; }
        }
    }

    public class MyTextElement : ConfigurationElement
    {
```

```
        protected override void DeserializeElement(System.Xml.XmlReader reader, bool
serializeCollectionKey)
        {
            CommandText = reader.ReadElementContentAs(typeof(string), null) as
string;
        }
        protected override bool SerializeElement(System.Xml.XmlWriter writer, bool
serializeCollectionKey)
        {
            if (writer != null)
            {
                writer.WriteCData(CommandText);
            }
            return true;
        }

        [ConfigurationProperty("data", IsRequired = false)]
        public string CommandText
        {
            get { return this["data"].ToString(); }
            set { this["data"] = value; }
        }
    }
}
```

调用例子

```
static void Main(string[] args)
{
    MySection mySection = ConfigurationManager.GetSection("MySection") as MySection;
    Console.WriteLine($"HTML={mySection.HTML.CommandText} SQL=
{mySection.SQL.CommandText}");
    Console.ReadLine();
}
```

# 自定义`Section`中获取key/value值

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="MyCollectionSection" type="ConfigurationDemo.MyCollectionSection,
ConfigurationDemo" />
  </configSections>
  <MyCollectionSection>
    <add key="a" value="刘备"></add>
    <add key="b" value="关羽"></add>
    <add key="c" value="张飞"></add>
  </MyCollectionSection>
</configuration>
```

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

namespace ConfigurationDemo
{
    public class MyCollectionSection: ConfigurationSection
    {
        private static readonly ConfigurationProperty s_property = new
ConfigurationProperty(string.Empty, typeof(MyKeyValueCollection), null,
ConfigurationPropertyOptions.IsDefaultCollection);

        [ConfigurationProperty("", Options =
ConfigurationPropertyOptions.IsDefaultCollection)]
        public MyKeyValueCollection KeyValues
        {
            get
            {
                return (MyKeyValueCollection)base[s_property];
            }
        }
    }

    [ConfigurationCollection(typeof(MyKeyValueSetting))]
    public class MyKeyValueCollection : ConfigurationElementCollection        // 自定
义一个集合
    {
        // 基本上，所有的方法都只要简单地调用基类的实现就可以了。
        public MyKeyValueCollection() : base(StringComparer.OrdinalIgnoreCase)    //
忽略大小写
        {

        }

        // 其实关键就是这个索引器。但它也是调用基类的实现，只是做下类型转就行了。
        new public MyKeyValueSetting this[string name]
        {
            get { return (MyKeyValueSetting)base.BaseGet(name); }
        }

        // 下面二个方法中抽象类中必须要实现的。
        protected override ConfigurationElement CreateNewElement()
        {
            return new MyKeyValueSetting();
        }

        protected override object GetElementKey(ConfigurationElement element)
        {
            return ((MyKeyValueSetting)element).Key;
        }
```

```
        // 说明：如果不需要在代码中修改集合，可以不实现Add, Clear, Remove
        public void Add(MyKeyValueSetting setting)
        {
            this.BaseAdd(setting);
        }

        public void Clear()
        {
            base.BaseClear();
        }

        public void Remove(string name)
        {
            base.BaseRemove(name);
        }
    }

    public class MyKeyValueSetting : ConfigurationElement    // 集合中的每个元素
    {
        [ConfigurationProperty("key", IsRequired = true)]
        public string Key
        {
            get { return this["key"].ToString(); }
            set { this["key"] = value; }
        }

        [ConfigurationProperty("value", IsRequired = true)]
        public string Value
        {
            get { return this["value"].ToString(); }
            set { this["value"] = value; }
        }
    }
}
```

调用示例

```
static void Main(string[] args)
{
     MyCollectionSection myCollectionSection =
ConfigurationManager.GetSection("MyCollectionSection") as MyCollectionSection;
     foreach (MyKeyValueSetting keyValues in myCollectionSection.KeyValues)
     {
       Console.WriteLine($"key={keyValues.Key} value={keyValues.Value}");
     }
}
```

# 读取`SectionGroup`里的配置

```
```

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <sectionGroup name="MySection">
      <section name="MyCollectionSection"
type="ConfigurationDemo.MyCollectionSection, ConfigurationDemo" />
    </sectionGroup>
  </configSections>
  <MySection>
    <MyCollectionSection>
      <add key="a" value="刘备"></add>
      <add key="b" value="关羽"></add>
      <add key="c" value="张飞"></add>
    </MyCollectionSection>
  </MySection>
</configuration>
```

```
static void Main(string[] args)
{
    /*System.Configuration.Configuration config =
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
    ConfigurationSectionCollection sections =
config.GetSectionGroup("MySection").Sections;
    MyCollectionSection myCollectionSection = sections["MyCollectionSection"] as
MyCollectionSection;
    foreach (MyKeyValueSetting keyValues in myCollectionSection.KeyValues)
    {
        Console.WriteLine($"key={keyValues.Key} value={keyValues.Value}");
    }*/
    MyCollectionSection myCollectionSection =
ConfigurationManager.GetSection("MySection/MyCollectionSection") as
MyCollectionSection;
    foreach (MyKeyValueSetting keyValues in myCollectionSection.KeyValues)
    {
        Console.WriteLine($"key={keyValues.Key} value={keyValues.Value}");
    }
    Console.ReadLine();
}
```

# 修改`App.config`下`appSettings`节点的配置

- 注意,修改后的配置只有在`Debug`文件夹下对应的`Config`文件`ConfigurationDemo.exe.Config`能看到修改后的内容，vs调试界面看不到

```
<configuration>
  <appSettings>
    <add key="he_zhw" value="he_zhw@neusoft.com"/>
  </appSettings>
  <connectionStrings>
    <add name="connectStr" providerName="MySql.Data.MySqlClient"
connectionString="User
Id=root;Password=123;server=192.168.72.128;port=3306;user=root;password=246;database
=bookstore"></add>
  </connectionStrings>
</configuration>
```

```
static void Main(string[] args)
{
try
{
Configuration config =
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
config.AppSettings.Settings.Remove("he_zhw");//先移除再添加
config.AppSettings.Settings.Add("he_zhw", "he_zhw@neusoft.com1");
config.Save(ConfigurationSaveMode.Modified);
ConfigurationManager.RefreshSection("appSettings");  //让修改之后的结果生效
}
catch (Exception ex)
{

Console.WriteLine(ex.Message);
}

Console.ReadLine();
}
```

# 自定义类型写入配置文件

- 注意,修改后的配置只有在 `Debug` 文件夹下对应的 `Config` 文件 `ConfigurationDemo.exe.Config` 能看到修改后的内容，vs调试界面看不到

添加一个自定义类型继承 `ConfigurationSection` 类

参考链接https://docs.microsoft.com/zh-cn/dotnet/api/system.configuration.configurationsectioncollection.add?view=netframework-4.8

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

namespace ConfigurationDemo
{
```

```
    public class PersonSection : ConfigurationSection
    {
        [ConfigurationProperty("age", IsRequired = false, DefaultValue = 0)]
        public int Age
        {
            get
            {
                return (int)this["age"];
            }
            set
            {
                this["age"] = value;
            }
        }

        [ConfigurationProperty("name", IsRequired = false, DefaultValue = "")]
        public string Name
        {
            get { return (string)this["name"]; }
            set
            {
                this["name"] = value;
            }
        }

    }
}
```

**代码示例**

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

namespace ConfigurationDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Configuration config =
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
                PersonSection personSection = new PersonSection();
                personSection.Age = 18;
```

```
                personSection.Name = "he_zhw";
                if (config.GetSection("Person")==null)
                {
                    config.Sections.Add("Person", personSection);
                }
                else
                {
                    config.Sections.Remove("Person");
                    config.Sections.Add("Person", personSection);
                }
                personSection.SectionInformation.ForceSave = true;
                config.Save(ConfigurationSaveMode.Modified);
                ConfigurationManager.RefreshSection("Person");  //让修改之后的结果生效
            }
            catch (Exception ex)
            {

                Console.WriteLine(ex.Message);
            }

            Console.ReadLine();
        }
    }
}
```

**原配置**

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>

  </configSections>
</configuration>
```

**修改后配置**

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>

    <section name="Person" type="ConfigurationDemo.PersonSection, ConfigurationDemo,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=null" />
  </configSections>
  <Person age="18" name="he_zhw" />
</configuration>
```

# ODP.NET配置

参考链接https://docs.oracle.com/en/database/oracle/oracle-data-access-components/19.3.2/odpnt/InstallConfig.html#GUID-ECDA2778-4835-417C-B81A-E0E1103B5B52

## `Oracle.ManagedDataAccess.dll`连接oracle配置

需引用`Oracle.ManagedDataAccess.dll`

### 未安装客户端的`App.config`中连接串配置

连接串中的`Data Source`配置成与`tns`文件的一样

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <connectionStrings>
  <add name="connectStr" connectionString="User Id=zdlyhiscs;Password=test1;Data
Source=(DESCRIPTION =(ADDRESS_LIST =(ADDRESS = (PROTOCOL = TCP)(HOST =
192.168.72.138)(PORT = 1521)))(CONNECT_DATA =(SERVER = DEDICATED)(SERVICE_NAME =
orcl.localdomain)))"></add>
  </connectionStrings>
    <startup>

    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" /></startup>
</configuration>
```

### 使用客户端连接配置`App.config`中的配置

安装了客户端`Data Source`可以`TNS`文件配置的别名连接(有些数据会连不上)

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <connectionStrings>
  <add name="connectStr" connectionString="User Id=zdlyhiscs;Password=test1;Data
Source=MYORACLE"></add>
  </connectionStrings>
    <startup>

    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" /></startup>
</configuration>
```

如果`Data Source`使用别名连不上使用以下配置

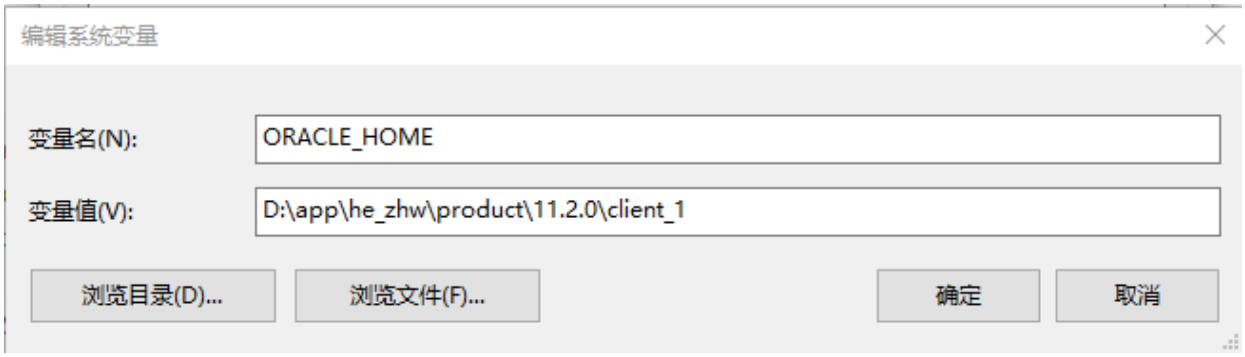
`TNS_ADMIN`为`tns`文件所在目录,`DataSource`

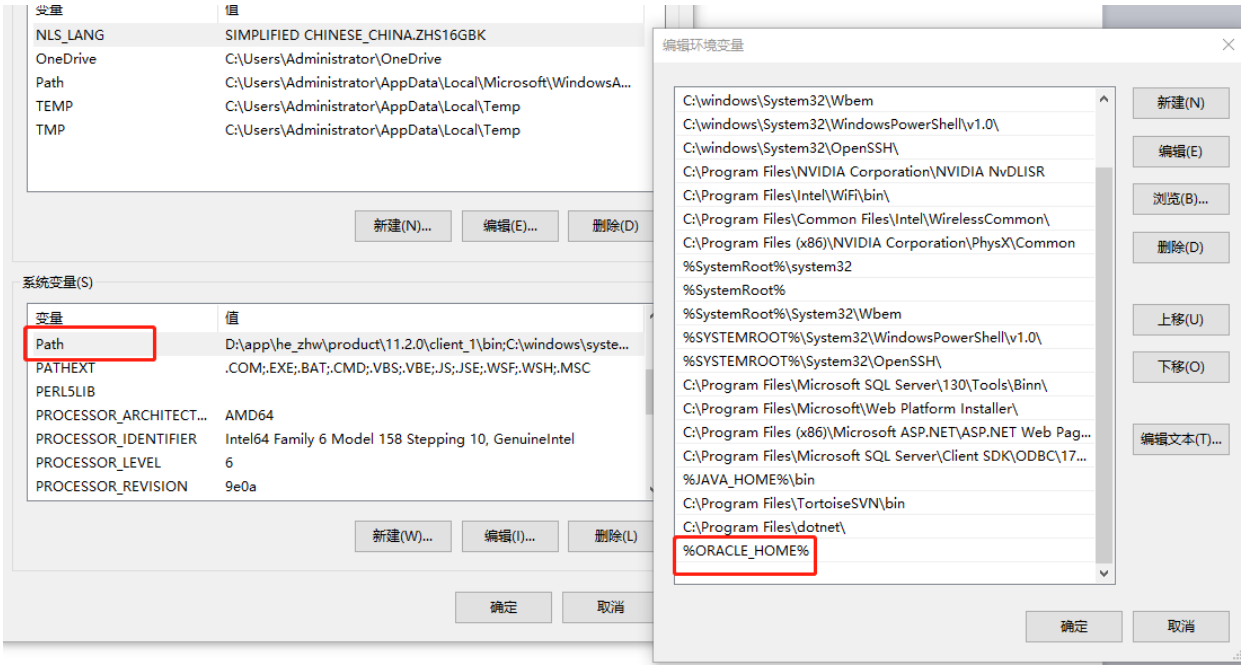
```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
```

```
    <section name="oracle.manageddataaccess.client"
type="OracleInternal.Common.ODPMSectionHandler, Oracle.ManagedDataAccess,
Version=4.122.19.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    </configSections>
  <connectionStrings>
  <add name="connectStr" connectionString="User Id=zdlyhiscs;Password=test1;Data
Source=MYORACLE"></add>
  </connectionStrings>
    <startup>

    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" /></startup>
    <oracle.manageddataaccess.client>
        <version number="*">
            <!--<dataSources>
                <dataSource alias="SampleDataSource" descriptor="(DESCRIPTION=
(ADDRESS=(PROTOCOL=tcp)(HOST=localhost)(PORT=1521))(CONNECT_DATA=
(SERVICE_NAME=ORCL))) " />
            </dataSources>-->
          <settings>
            <setting name="TNS_ADMIN"
value="D:\app\he_zhw\product\11.2.0\client_1\network\admin" />
          </settings>
        </version>
    </oracle.manageddataaccess.client>
</configuration>
```

或者配置`ORACLE_HOME`环境变量就可以不用配置`TNS_ADMIN`节点






```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="oracle.manageddataaccess.client"
type="OracleInternal.Common.ODPMSectionHandler, Oracle.ManagedDataAccess,
Version=4.122.19.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    </configSections>
  <connectionStrings>
  <add name="connectStr" connectionString="User Id=zdlyhiscs;Password=test1;Data
Source=MYORACLE"></add>
  </connectionStrings>
    <startup>

    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" /></startup>
    <oracle.manageddataaccess.client>
        <version number="*">
        </version>
    </oracle.manageddataaccess.client>
</configuration>
```

**oracle官方提供的tns搜索目录顺序**

1. `OracleConfiguration.OracleDataSources`
2. `<dataSources>` .NET配置文件中的配置
3. 目录在`OracleConnection.TnsAdmin`属性中的设置
4. 为`Tns_Admin`连接字符串属性 设置的目录
5. 目录在`OracleConfiguration.TnsAdmin`属性中的设置
6. `TNS_ADMIN` .NET配置文件中的目录设置

7. 当前工作目录
8. `TNS_ADMIN` Windows环境变量或容器环境变量的目录设置

**oracle官方推荐的完整配置**

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="oracle.manageddataaccess.client"
      type="OracleInternal.Common.ODPMSectionHandler, Oracle.ManagedDataAccess,
Version=4.122.19.1, Culture=neutral, PublicKeyToken=89b483f429c47342"/>
    </configSections>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.1"/>
    </startup>
    <system.data>
        <DbProviderFactories>
            <remove invariant="Oracle.ManagedDataAccess.Client"/>
            <add name="ODP.NET, Managed Driver"
invariant="Oracle.ManagedDataAccess.Client" description="Oracle Data Provider for
.NET, Managed Driver"
type="Oracle.ManagedDataAccess.Client.OracleClientFactory, Oracle.ManagedDataAccess,
Version=4.122.19.1, Culture=neutral, PublicKeyToken=89b483f429c47342"/>
        </DbProviderFactories>
    </system.data>
    <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
            <dependentAssembly>
                <publisherPolicy apply="no"/>
                <assemblyIdentity name="Oracle.ManagedDataAccess"
publicKeyToken="89b483f429c47342" culture="neutral"/>
                <bindingRedirect oldVersion="4.121.0.0 - 4.65535.65535.65535"
newVersion="4.122.19.1"/>
            </dependentAssembly>
        </assemblyBinding>
    </runtime>
    <oracle.manageddataaccess.client>
        <version number="*">
            <dataSources>
                <dataSource alias="SampleDataSource"
descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=localhost)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=ORCL))) "/>
            </dataSources>
        </version>
    </oracle.manageddataaccess.client>
</configuration>
```