参考文章：https://www.cnblogs.com/artech/p/inside-asp-net-core-1.html

配置作为系统初始化的数据，在系统中比较常见，我们一般使用的多是json文件配置

**简单示例**

Dictionary<string, string> source = new Dictionary<string, string> {

    // IConfiguration使用 : 来分隔层次结构，如下配置结构为

    // {

    //     format:{

    //         dateTime:{

    //             longDatePattern:"dddd, MMMM d, yyyy",

    //             ...

    //         }

    //     }

    // }

    ["format:dateTime:longDatePattern"] = "dddd, MMMM d, yyyy",

    ["format:dateTime:longTimePattern"] = "h:mm:ss tt",

    ["format:dateTime:shortDatePattern"] = "M/d/yyyy",

    ["format:dateTime:shortTimePattern"] = "h:mm tt",

};

public class FormatOptions {

public DateTimeFormatOptions DateTime { get; set; }

    public FormatOptions (IConfiguration config) {

        // 使用 GetSection 获取 dateTime 配置结构

        this.DateTime = new DateTimeFormatOptions (config.GetSection ("DateTime"));

    }

}

public class DateTimeFormatOptions {

    ...

    //其他成员

    public DateTimeFormatOptions (IConfiguration config) {

        // 键值配置配置

        this.LongDatePattern = config["LongDatePattern"];

        this.LongTimePattern = config["LongTimePattern"];

        this.ShortDatePattern = config["ShortDatePattern"];

        this.ShortTimePattern = config["ShortTimePattern"];

    }

}

IConfiguration config =

    // 新建ConfigurationBuilder

    new ConfigurationBuilder ()

    // 使用内存配置源

    .Add (new MemoryConfigurationSource { InitialData = source })

    // 生成Configuration

    .Build ();

// 使用 Format 配置结构生成 FormatOptions

FormatOptions formatOptions = new FormatOptions(config.GetSection("Format"));

// 当然，你也可以直接获取数据， : 代表层次结构

var longDatePattern = config["format:dateTime:longDatePattern"];

**配置设计**

**IConfiguration配置**

IConfiguration可以看作配置树的某一个节点，可以是根节点，子节点，或叶节点

public interface IConfiguration {

    // 获取当前配置下的所有子配置

    IEnumerable<IConfigurationSection> GetChildren ();

    // 获取某一配置

    IConfigurationSection GetSection (string key);

    IChangeToken GetReloadToken ();

    // 根据键取得配置值

    string this [string key] { get; set; }

}

IConfiguration有2个继承

IConfigurationRoot表示根配置节点

public interface IConfigurationRoot : IConfiguration

{

    // 重新加载配置

    void Reload();

}

IConfigurationSection表示非根子节点

public interface IConfigurationSection : IConfiguration {

    // 当前配置的路径

    string Path { get; }

    // 当前配置路径下的Key

    string Key { get; }

    // 如果当前节点为叶子节点，那么 Value 保存其值，否则为 NULL

    string Value { get; set; }

}

**IConfiguration从何而来**

IConfiguration由IConfigurationBulder生成

public interface IConfigurationBuilder

{

    IEnumerable<IConfigurationSource>  Sources { get; }

    Dictionary<string, object>         Properties { get; }

    // 添加配置源

    IConfigurationBuilder     Add(IConfigurationSource source);

    // 生成根配置

    IConfigurationRoot        Build();

}

在生成根配置前，我们可以添加多个配置源IConfigurationSource

**IConfigurationSource配置源**

IConfigurationSource表示配置源，其提供了一个方法Build用于生成配置提供者

public interface IConfigurationSource {

    // 生成配置提供者

    IConfigurationProvider Build (IConfigurationBuilder builder);

}

从这里我们可以意识到，IConfigurationSource并不提供数据访问，而提供数据访问的是IConfigurationProvider

**IConfigurationProvider**

IConfigurationProvider代表配置数据提供者，提供数据访问

public interface IConfigurationProvider {

    // 加载配置数据

    void Load ();

    // 试图获取数据

    bool TryGet (string key, out string value);

    // 设置配置数据（这个配置一般不持久化到物理数据中）

    void Set (string key, string value);

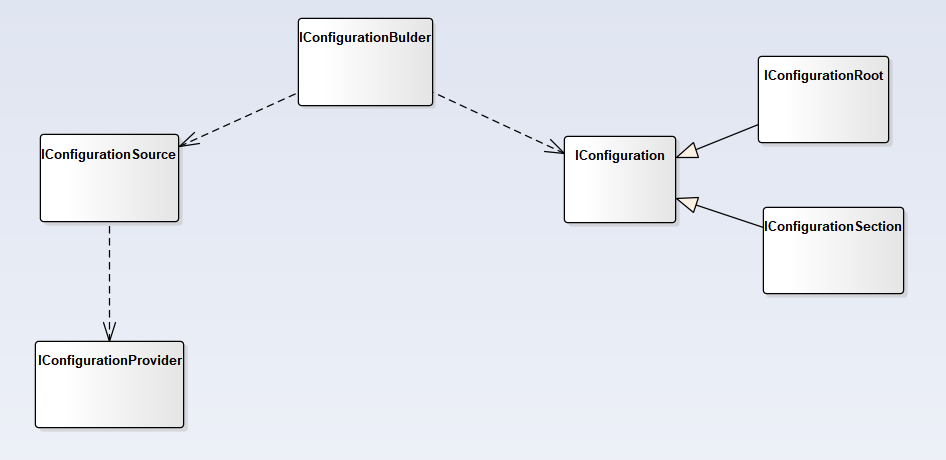
    // 获取某个配置节点下的子节点Key

    IEnumerable<string> GetChildKeys (IEnumerable<string> earlierKeys, string parentPath)

}

**配置的设计图**

如下是配置基于接口的设计



.net core 给我们提供了默认的IConfiguration和IConfigurationBulder，只要我们实现IConfigurationSource，我们就可以实现自定义的配置源

**.net core 提供的配置源**

**内存配置源**

内存配置源的实现为MemoryConfigurationSource

其提供了2个扩展方法将源快速添加到ConfigurationBuilder中，

public static IConfigurationBuilder AddInMemoryCollection(

    this IConfigurationBuilder configurationBuilder);

public static IConfigurationBuilder AddInMemoryCollection(

    this IConfigurationBuilder configurationBuilder,

    IEnumerable<KeyValuePair<string, string>> initialData);

**环境变量配置源**

环境变量配置源的实现为EnvironmentVariablesConfigurationSource

包：Microsoft.Extensions.Configuration.EnvironmentVariables

public static IConfigurationBuilder AddEnvironmentVariables(

    this IConfigurationBuilder configurationBuilder);

public static IConfigurationBuilder AddEnvironmentVariables(

    this IConfigurationBuilder configurationBuilder,

    string prefix);

**命令行参数**

命令行参数配置源的实现为CommandLineConfigurationSource

包：Microsoft.Extensions.Configuration.CommandLine

public static IConfigurationBuilder AddCommandLine(

    this IConfigurationBuilder configurationBuilder,

    string[] args);

public static IConfigurationBuilder AddCommandLine(

    this IConfigurationBuilder configurationBuilder,

    string[] args,

    IDictionary<string, string> switchMappings);

**Json文件**

其实现的配置源为JsonConfigurationSource

包：Microsoft.Extensions.Configuration.Json

包提供的添加Json配置源的扩展

public static class JsonConfigurationExtensions

{

    public static IConfigurationBuilder AddJsonFile(

        this IConfigurationBuilder builder,

        Action<JsonConfigurationSource> configureSource);

    public static IConfigurationBuilder AddJsonFile(

        this IConfigurationBuilder builder,

        string path);

    public static IConfigurationBuilder AddJsonFile(

        this IConfigurationBuilder builder,

        string path,

        bool optional);

    public static IConfigurationBuilder AddJsonFile(

        this IConfigurationBuilder builder,

        string path,            // 配置源路径

        bool optional,          // 是否可选配置源

        bool reloadOnChange);   // 改变是否重新载入

}

**Xml文件**

其实现为XmlConfiguationSource

包：Microsoft.Extensions.Configuration.Xml

包提供的XML配置源的扩展

public static class XmlConfigurationExtensions

{

    public static IConfigurationBuilder AddXmlFile(

        this IConfigurationBuilder builder,

        string path,

        bool optional,

        bool reloadOnChange);

    ...

}

XML文件示例

<Profiles>

  <Foo Gender="Male" Age="18">

    <ContactInfo EmailAddress ="foobar@outlook.com" PhoneNo="123"/>

  </Foo>

  <Bar Gender="Male" Age="25">

    <ContactInfo EmailAddress ="foobar@outlook.com" PhoneNo="123"/>

  </Bar>

  <Baz Gender="Male" Age="18">

    <ContactInfo EmailAddress ="baz@outlook.com" PhoneNo="789"/>

  </Baz>

</Profiles>

**ini文件**

其实现为IniConfigurationSource

包：Microsoft.Extensions.Configuration.Ini

包提供的扩展

public static class IniConfigurationExtensions

{

    public static IConfigurationBuilder AddIniFile(

        this IConfigurationBuilder builder,

        string path,

        bool optional,

        bool reloadOnChange);

    ...

}

INI文件示例

Gender                         = "Male"

Age                            = "18"

ContactInfo:EmailAddress       = "foobar@outlook.com"

ContactInfo:PhoneNo            = "123456789"