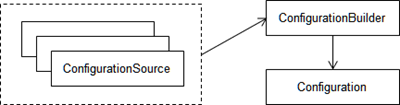
Configuration代表一个配置对象

Configuration由ConfigurationBuilder生成

我们可以在ConfigurationBuilder中注册多个ConfigurationSource，ConfigurationBuilder利用这些ConfigurationSource构建最终的Configuration



上述三个对象的接口定义在nuget”Microsoft.Extensions.Configuration.Abstractions”中

其实现在“Microsoft.Extensions.Configuration”中

**创建并读取IConfiguration**

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

    ["longDatePattern"] = "dddd, MMMM d, yyyy",

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

    ["shortDatePattern"] = "M/d/yyyy",

    ["shortTimePattern"] = "h:mm tt"

};

IConfiguration config =

    // 新建ConfigurationBuilder

    new ConfigurationBuilder ()

    // 使用内存配置源

    .Add (new MemoryConfigurationSource { InitialData = source })

    // 生成Configuration

.Build ();

// 读取Configuration数据

longDatePattern = config["LongDatePattern"];

**读取结构化数据**

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"];

**自动绑定配置数据**

如果你觉得手动绑定数据到对象太麻烦，那么你可以使用包“Microsoft.Extensions.Options.ConfigurationExtensions”

使用自动绑定后，我们的DateTimeFormatOptions和FormatOptions的构造函数可以删除了

FormatOptions options = new ServiceCollection ()

    // 注册 option

    .AddOptions ()

    // 配置 option，传入我们的 IConfiguration

    .Configure<FormatOptions> (config.GetSection ("Format"))

    .BuildServiceProvider ()

    // 获取 option

    .GetService<IOptions<FormatOptions>> ()

    .Value;