1.基本说明

DirectoryInfo类和Directory类之间的关系与FileInfo类和File类之间的关系十分类似。下面介绍一下DirectoryInfo类的常用属性。

|  |  |
| --- | --- |
| 属性 | 说明 |
| Attributes | 设置当前 FileSystemInfo 的 FileAttributes |
| CreationTime | 设置当前 FileSystemInfo 对象的创建时间 |
| Exists | 获取指示目录是否存在的值 |
| FullName | 获取目录或文件的完整目录 |
| Parent | 获取指定子目录的父目录 |
| Name | 获取此 DirectoryInfo 实例的名称 |

（1）CreationTime 属性

设置当前 FileSystemInfo 对象的创建时间。

语法：

public DateTime CreationTime { get; set; }

属性值：当前 FileSystemInfo 对象的创建日期和时间。

例如，获得F:\test\0000\目录下文件夹AA的创建日期和时间，首先将DirectoryInfo实例化，然后通过CreationTime属性得到此文件夹的创建日期和时间。代码如下所示：

[复制代码](javascript:void(0);)

string Paths = @"F:\test\0000\AA";

DirectoryInfo di = new DirectoryInfo(Paths);// 将DirectoryInfo实例化

string name = di.CreationTime.ToString();//通过CreationTime属性得到此文件夹的创建日期和时间

Label1.Text = "文件夹创建时间："+name;

[复制代码](javascript:void(0);)  
（2）Exists属性

判断指定的文件夹是否存在。

语法：

public override bool Exists { get; }

属性值：如果目录存在，则为 True；否则为 False。

例如，判断F:\test\0000\目录下是否存在名为ls的文件夹，首先要将DirectoryInfo实例化，然后通过Exists属性判断此文件夹是否存在，如果文件夹存在，返回值为True，否则为False。代码如下所示：

string Paths = @"F:\test\0000\AA";

DirectoryInfo di = new DirectoryInfo(Paths);

if (di.Exists){

Page.RegisterStartupScript("","<script>alert('该文件夹已经存在')</script>");

}

## 2、DirectoryInfo类的常用方法及说明

|  |  |
| --- | --- |
| 方法 | 说明 |
| Create | 创建目录 |
| CreateObjRef | 创建一个对象，该对象包含生成用于与远程对象进行通信的代理所需的全部相关信息 |
| CreateSubdirectory | 在指定路径中创建一个或多个子目录。指定路径可以是相对于 DirectoryInfo 类的此实例的路径 |
| Delete | 从路径中删除 DirectoryInfo 及其内容 |
| Equals | 确定两个 Object 实例是否相等 |
| GetAccessControl | 获取当前目录的访问控制列表 (ACL) 项 |
| GetDirectories | 返回当前目录的子目录 |
| GetFiles | 返回当前目录的文件列表 |
| GetFileSystemInfos | 检索表示当前目录的文件和子目录的强类型 FileSystemInfo 对象的数组 |
| GetHashCode | 用作特定类型的哈希函数。GetHashCode 适合在哈希算法和数据结构（如哈希表）中使用 |
| GetLifetimeService | 检索控制此实例的生存期策略的当前生存期服务对象 |
| GetObjectData | 设置带有文件名和附加异常信息的 SerializationInfo 对象 |
| GetType | 获取当前实例的 Type |
| InitializeLifetimeService | 获取控制此实例的生存期策略的生存期服务对象 |
| MoveTo | 将 DirectoryInfo 实例及其内容移动到新路径 |
| ReferenceEquals | 确定指定的 Object 实例是否是相同的实例 |
| Refresh | 刷新对象的状态 |
| SetAccessControl | 将 DirectorySecurity 对象所描述的访问控制列表 (ACL) 项应用于当前 DirectoryInfo 对象所描述的目录 |
| ToString | 返回用户所传递的原始路径 |

下面对DirectoryInfo类中比较重要的方法进行详细介绍。

（1）Create方法

此方法可以方便的创建目录。

语法：

public void Create ()

例如，在D盘下创建名为AA的文件夹，首先要将DirectoryInfo类实例化，然后判断是否存在同名的文件夹，如果不存在，则使用Create方法创建目录。代码如下所示：

string path = "D:\\AA";

DirectoryInfo di = new DirectoryInfo(path);

if (!di.Exists){

di.Create();}

else{Response.Write("此文件夹已经存在！");}

（2）GetFileSystemInfos方法

返回表示某个目录中所有文件和子目录的强类型FileSystemInfo项的数组。

语法：

public FileSystemInfo[] GetFileSystemInfos ()  
返回值：

强类型 FileSystemInfo 项的数组。

（3）MoveTo方法

将指定的内容移动到新路径。

语法：

public void MoveTo (string destDirName)  
参数：

destDirName ：要将此目录移动到的目标位置的名称和路径，目标不能是另一个具有相同名称的磁盘卷或目录。

（4）Delete方法

删除指定的文件夹。

语法：

public override void Delete ()

例如，删除D盘下名为AA的文件夹，首先将DirectoryInfo类实例化，然后判断是否存在此文件夹，如果存在则使用Delete方法删除此文件夹。

string path = "D:\\AA";

DirectoryInfo di = new DirectoryInfo(path);

if (di.Exists){

di. Delete();}

else{Response.Write("不存在此文件夹！");

}

Directory 与 DirectoryInfo 这两个能够实现大多数相同的功能。  
  
区别可分为两点：  
 一、静态与实例调用 前者必须被实例化后才能使用，而后者则只提 供了静态的方法。如果多次使用某个对象一般使用前者； 如果仅执行某一个操作则使用后者提供的静态方法效率更 高一些。   
Directory 可直接调用，不需要实例化，比如 Directory.Delete(path)。一般比DirectoryInfo的函数要多出一个path参数。 DirectoryInfo 必须实例化才能调用其方法，在频繁地操作某一目录时，这种方式比较方便。  
 二、功能多少 DirectoryInfo 的功能要稍微比 Directory 多些、强些，比如 DirectoryInfo 的 GetFiles 返回的是 FileInfo[]，而 Directory.GetFiles 返回的是 string[]。