به نام یگانه برنامه نویس عالم

با سلام در این مقاله قصد داریم به چگونگی نوشتن و خواندن از و در فایلهای XML بپردازم. اما قبل از هر چیز توضیح مختصری از فایلهای XML می دهم، به طور کلی XML مخفف Extensible Markup Language یا زبان نشانه گذاری قابل گسترش است در واقع XML خود یک زبان برنامه نویسی نیست بلکه فقط زبانی است برای توصیف داده ها و از آنجا دارای اهمیت می شود که می توان از آن برای انتقال اطلاعات به صورت فایل های متنی با خوانایی بالا و میان پلتفرم های مختلف استفاده کرد.

برای یادگیری اصول اولیه XML می توانید با جستجوی در وب منابع متعددی را به زبان فارسی و انگلیسی پیدا کنید البته برای کار با XML در C# نیاز به دانستن جزئیات زیادی درباره XML ندارید. فقط باید بدانید یک تگ چیست و مباحثی از این قبیل که با حدود ۱ ساعت مطالعه می توانید به راحتی این مباحث را فرا بگیرید. اما استفاده از XML در C# و اینکه اصلا چه دلیلی دارد که می از این تکنولوژی استفاده نماییم؟ فرض کنید شما نرم افزاری نوشته اید و این نرم افزار می خواهد اطلاعات خود را به یک کمپانی بفرستد اما باید فرمت اطلاعات شما حالتی باشد که نرم افزار گیرنده بتواند آن را درک نموده و بخواند و احتیاج به دانستن ساختار فایل برنامه شما نباشد در ضمن برای یک فرد هم قابل درک باشد در چنین موردی می توانید اطلاعات خود را در قالب یک فایل XML به کمپانی مورد نظر(از طریق شبکه) ارسال نمایید و برنامه مقصد هم می تواند به راحتی فقط با داشتن اطلاعات کمی درباره ساختار داده ای فایل XML شما که آن را هم می توان در قالب یک فایل DTD یا XML Schema فرستاد آن را بخواند و تجزیه نماید. همان طور که دیدید XML می تواند کار ما را در چنین مواردی حیلی راحتتر کند البته کاربرد این زبان فقط به این مورد محدود نمی شود و کاربردهای بسیار دیگری نیز دارد که در این بحث نمیگنجد.

برای استفاده از XML در C# باید با یک دستور using در ابتدا فضای نام System.Xml.Serialization را به برنامه خود اضافه نمایید همان طور که از نام این فضای نام پیدا است برای کارکردن با فایلهای XML به کار می رود و دارای کلاسهای بسیار زیادی در این زمینه است در ضمن همان طور که میدانید برای کار با هر نوع ورودی و خروجی اعم از فایل های باینری و یا XML باید ابتدا یک جریان فایل ایجاد نموده سپس آن را به کلاس مورد نظر خود مربوط نمایید برای این امر هم باید فضای نام System.IO را هم به برنامه خود اضافه نمایید. پس از انجام این مراحل باید اطلاعاتی را که میخواهید در فایل XML خود ذخیره نمایید در قالب یک کلاس کپسوله سازی نمایید. به عنوان کلاس صفحه بعد را در نظر می گیریم این کلاس همان طوریکه از نام آن پیدا است برای ذخیره سازی اطلاعات یک شخص طراحی شده است البته فیلدهای بسیار دیگیری را نیز می تواند در قالب این کلاس قرار داد اما برای سادگی کار من فقط از همین چند فیلد استفاده کرده ام. همان طوریکه مشاهده می نمایید علاوه بر تعدادی متغیر خصوصی تعدادی خاصیت(Property) هم به منظور دسترسی به متغیرهای عضو در نظر گرفته شده است که کاربر می تواند با استفاده از آنها با متغیر های عضو مقداردهی نماید و یا مقداری را از آنها بخواند. خوب به سراغ ذخیره در فایل XML می رویم برای ذخیره سازی اطلاعات و خواندن اطلاعات از و در یک فایل XML از کلاسی به نام XMLSerializer استفاده می نماییم که جزو فضای نام System.Xml است طریقه کلی کار این کلاس بدین صورت است که این کلاس در میان کلاس شروع به گشتن می نامید و سپس خصوصیت هایی را که هم دارای متد get و هم دارای متد set هستند پیدا کرده و مقادیر آنها را در فایل XML قرار می دهد طرز کار این کلاس بسیار ساده بوده و توانمندی قدرتمندی را در ذخیره و بازیابی فایل های XML که فرمت آنها را می دانیم به ما ارائه می دهد خوب به مثال زیر و توضیحات آن توجه نمایید:

```
class PersonInfo
{
    private string _Name;
    private string _LastName;
    private byte _Age;
    public string Name
    {
        get
        {
            return _Name;
        }
        set
        {
            _Name = value;
        }
    }
    public string LastName
    {
        get
        {
            return _LastName;
        }
        set
        {
            _LastName = value;
        }
    }
    public byte Age
    {
        get
        {
            return _Age;
        }
        set
        {
            _Age = value;
        }
    }

}
```

پس از قرار دادن تعریف خود در یک فایل کلاس در برنامه فضاهای نام مربوط را هم به برنامه خود اضافه نمایید حال وقت آن است که یک متد به کلاس PersonInfo اضافه نمکاییم که کلاس بتواند محتویات خود را در یک فایل XML بنویسد برای این کار کد زیر را بنویسید:

```
public void Write_Into_XML_File(string FilePath)
        {
FileStream XmlStream = new
FileStream(FilePath,FileMode.Create,FileAccess.Write);
XmlSerializer XmlSerializer = new XmlSerializer(this.GetType());
XmlSerializer.Serialize(XmlStream, this);
XmlStream.Close();
}
```

اگر تعریف صفحه قبل را به کلاس خود اضاف فه نمایید و متغیری از نوع کلاس تعریف نموده و مقادیر پارامتر نام را به Gholamreza نام فامیلی را به Sabery و سن را به ۱۹ ست نمایید فایل XML با خروجی زیر برایتان ایجاد می شود فراموش نکنید مسیر فایل خود را به تابع بدهید مثلا "C:\\MyFirstXMLDocument.xml" در مسیر از دو علامت // استفاده شده چرا که یک علامت / برای سی شارپ معنای دیگری دارد و برای اینکه به سی شارپ بفهمانیم منظور ما از علامت / خود آن است باید از دو علامت پشت سر هم استفاده نماییم:

```
<?xml version="1.0" ?>
- <PersonInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Name>Gholamreza</Name>
  <LastName>Sabery</LastName>
  <Age>19</Age>
   </PersonInfo>
```

همان گونه که مشاهده می نمایید اطلاعات ما به درستی در قالب یک فایل XML ذخیره شد همان طور که گفتم کلاس XMLSerializer در بین اعضای کلاس خصوصیت هایی را که دارای متدهای set و get هستند پیدا کرده و در فایل می نویسد. اما نکاتی که باید در کد بالا به آنها توجه نمایید این است که همان طور که مشاهده می نمایید در ابتدا یا جریان فایل به نام XmlStream تعریف کرده ام و سپس مقدار پارامتر FilePath را که نام و مسیر فایل XML است به آن ارسال کرده ام سپس بخش های دیگر را نیز که برای ایجاد یک فایل لازم است به تابع سازنده کلاس FileStream ارسال کرده ام سپس یک شی از نوع XmlSerializer تعریف کرده ام که برای سریالایز کردن داده ها در فایل XML به کار می رود(به عمل نوشتن اطلاعات در یک فایل XML اصطلاحا سریالایز کردن و به عمل خواندن آن دی سریالایز می گویند) تابع سازنده کلاس XmlSerializer به عنوان آرگومان تابع سازنده خود احتیاج به نوع شیء دارد که می خواهد آن را سریالایز نماید به همین دلیل من در اینجا نوع کلاس را با استفاده از تابع gettype() و اشاره گر this به سازنده فرستاده ام همان طور که می دانید اشاره گر this در هر کلاس به خود آن اشاره می کند در ضمن همان طور که می دانید هر کلاسی به طور مستقیم یا غیر مستقیم از کلاس Object به ارث می برد و تابع gettype هم یکی از توابع کلاس Object است که نوع شی فراخواننده خود را باز می گرداند. پس از آن با استفاده از متد Serialize کلاس XmlSerializer که احتیاج به دو آرگومان دارد یکی نام جریان که از نوع FileStream است و دیگری نام شی، شی مورد نظر را در فایل ذخیره کرده ام و سپس با استفاده از تابع عضو close() مر بوط به کلاس FileStream ارتباط برنامه را با فایل قطع نموده ام.

اما برای خواندن اطلاعات از فایل باید چه کاری انجام دهیم؟ باید به شما مژده بدهم خواندن اطلاعات از فایل های XML به را حتی نوشتن آنها است البته مورد زیر فقط در مورد کلاس هایی صدق می کند که فرمت آنها را بدانیم یعنی در واقع خودمان شیعی را از نوع کلاسی که در فایل XML است در دسترس داشته باشیم وگرنه باید از روش دیگری برای خواندن فایل استفاده نماییم که در ادامه به آن خواهیم پرداخت خوب برای خواندن از فایل تابعی دیگر به نام Read_From_XML_File به کلاس به شکل زیر اضافه می نماییم:

```
 public ArrayList Read_From_XML_File(string FilePath)
 {
FileStream XmlStream = new FileStream(FilePath, FileMode.Open,
FileAccess.Read);
XmlSerializer XmlReader = new XmlSerializer(this.GetType);
 ArrayList AllItems = new ArrayList();
PersonInfo Info=new PersonInfo();
  try
            {
                while (true)
```

```
                {
Info=(PersonInfo)XmlReader.Deserialize(XmlStream);
AllItems.Add(Info);
                }
            }
            catch
            {
                XmlStream.Close();
            }
            return AllItems;
        }
```

همان طور که در کد بالا مشاهده نمودید خواندن از یک فایل XML هم دقیقا مانند نوشتن در آن است فقط چند تفاوت جزئی وجود دارد در ابتدا یک جریان فایل و سپس شی برای خواندن XML ایجاد نمودیم که در بخش قبلی طریقه کار با این بخش ها را آموختیم سپس یک ArrayList تعریف کرده ام برای اینکه بتوان تمامی داده های موجود در فایل XML را از آن خواند همین طور یک شی از کلاس PersonInfo برای اینکه داده های تک تک خوانده شده را در آن قرار دهیم سپس هر یک را به ArrayList اضافه نمایم، سپس در یک دستور try-catch عمل خواندن از فایل را انجام داده ام. در یک حلقه while همان طور که مشاهده می نمایید یک شرط همیشه درست قرار داده ام این باعث می شود حلقه به صورت مدام اجرا شود سپس شی تعریف شده از PersonInfo را به شکلی که مشاهده می کنید استفاده کرده ام:

```
Info=(PersonInfo)XmlReader.Deserialize(XmlStream);
```

به علت اینکه نوع بازگشتی متد Deserialize کلاس XmlSerializer یک نوع Object است به همین علت باید نوع بازگشتی را برای تخصیص به شیء موردنظرمان تبدیل نماییم عمل تبدیل همان طور که میدانید به صورت:

TargetTypeObject=(TargetType)SourceType;

انجام می شود که در آن TargetTypeObject شئی از نوع مورد نظر است که می خواهید پس از تبدیل مقدار به آن اختصاص یابد و (TargetType) نوعی است که می خواهید داده شما به آن تبدیل شود و SourceType هم نوعی است که می خواهید تبدیل نمایید. پس از تخصیص هم کلاس را در ArrayList اضافه کرده ام و در پایان تابع آنرا با دستور Return بازگردانده ام. اما حلقه while و دستورات try-catch این شکل به کارگیری دستور باعث می شود داده ها از فایل خوانده شود و هر زمان که دیگر داده ای برای خواندن از فایل نباشد خطایی ایجاد می گردد و سپس برنامه به بخش catch رفته و جریان فایل بسته میشود.

خواندن اطلاعات از فایل XML که فرمت آنرا نمی دانیم:

برای این کار باید از کلاس XmlTextReader استفاده نمایید، این کلاس یک دسترسی یکطرفه (Forward-Only) و فقط خواندنی را به روی فایل XML شما ایجاد می نماید این کلاس یکی از قدرتمند ترین کلاس های NET. برای کار با فایل های XML است و توسط آن می توانید کنترل تقریبا کاملی را روی تک تک بخش های فایل های XML داشته باشید، طریقه استفاده از آن به صورت زیر است:

XmlTextReader textreader=new XmlTextReader(filename);

که در آن filename نام و مسیر فایل XML شما است که می خواهید آنرا بخوانید. پس از آن می توانید داده ها را به صورت زیر از فایل بخوانید

```
while(textreader.Read())
{
   doing someting
}
```

متد Read این کلاس یک مقدار از نوع Boolean باز می گردان که مشخص کننده این است که آیا هنوز داده ای برای خوانده شدن وجود دارد و یا خیر. مثال زیر نمونه کاملی است از استفاده از این کلاس که در زیر آمده است:

```
public class Sample {

  private const String filename = "items.xml";

  public static void Main() {

     XmlTextReader reader = null;

     try {

        // Load the reader with the data file and ignore all white
space nodes.
        reader = new XmlTextReader(filename);
        reader.WhitespaceHandling = WhitespaceHandling.None;

        // Parse the file and display each of the nodes.
        while (reader.Read()) {
           switch (reader.NodeType) {
             case XmlNodeType.Element:
               Console.Write("<{0}>", reader.Name);
               break;
             case XmlNodeType.Text:
               Console.Write(reader.Value);
               break;
             case XmlNodeType.CDATA:
               Console.Write("<![CDATA[{0}]]>", reader.Value);
               break;
             case XmlNodeType.ProcessingInstruction:
               Console.Write("<?{0} {1}?>", reader.Name,
reader.Value);
               break;
             case XmlNodeType.Comment:
               Console.Write("<!--{0}-->", reader.Value);
               break;
             case XmlNodeType.XmlDeclaration:
               Console.Write("<?xml version='1.0'?>");
               break;
             case XmlNodeType.Document:
               break;
             case XmlNodeType.DocumentType:
               Console.Write("<!DOCTYPE {0} [{1}]", reader.Name,
reader.Value);
               break;
             case XmlNodeType.EntityReference:
               Console.Write(reader.Name);
               break;
             case XmlNodeType.EndElement:
               Console.Write("</{0}>", reader.Name);
               break;
           }
        }
     }

     finally {
        if (reader!=null)
          reader.Close();
     }
```

```
    }
} // End class
```

همان طور که مشاهده می نمایید در یک دستور switch می توانید مقدا فعلی المان را با خاصیت NodeType چک نمایید که مثلا اگر XmlNodeType.Text باشد متن میان دو تگ باز و بسته است و به همین ترتیب.

درباره خودم: اسم بنده را که دربالای تمامی صفحات مشاهده می نمایید و احتیاجی به ذکر دوباره آن نمیبینم اما من دانشجوی کاردانی پیوسته رشته کامپیوتر هستم و تخصص و علاقه بنده در برنامه نویسی با زبانهای C/C++/C#.NET است به علاوه به Linux هم علاقه وافری دارم و آشنایی تقریبا متوسط(روبه بالا) با آن داشته و در برنامه نویسی با Shell هم اندک تجربه ای دارم امیدوارم مقالات نوشته شده بتواند اندکی در برنامه نویسی با #C به شما کمک کند.

در پایان امیدوارم استفاده لازم را از این مقاله با وجود نقص های بسیار آن برده باشید.
در صورت تمایل می توانید نظرات خود و یا اشکالاتتن را به ایمیل:
Sephiroth_isnot_bad@yahoo.com
ارسال نمایید.