직렬화(Serialization)

우선, serialization 이 무엇인지 알아야 하겠죠?

컴퓨터 용어에서 serialization 은 종종 하나의 오브젝트를 바이트의 배열로 전환(converting)시키는 것을 의미합니다. 하지만

WCF 에서의 serialization 은 이러한 의미로 사용되지는 않구요, 하나의 오브젝트(or 닷넷 클래스)를 XML Information Set(XML Infoset) 으로 전환시킨다는 의미로 사용됩니다.

WCF 에서는 아래와 같은 네 가지의 serialization 을 위한 클래스를 제공합니다.

- DataContractSerializer
  : WCF 에서 사용하는 기본 serializer 를 제공한다.
- NetDataContractSerializer
  : 타입에 대한 추가적인 정보를 제공하는 serializer
- XmlSerializer
  : .NET 2.0 에서 제공되었던 serialization 을 수행한다.
- DataContractJsonSerializer
  : serialization 포맷으로 JSON 을 제공한다.

이번 포스팅에서는 이러한 클래스들이 직렬화를 했을 때 어떤 결과물을 보여주는지에 대해 써볼까 합니다. 이러한 내용들이 WCF 서비스를 만들때 직접적인 도움이 되진 않겠지만, WCF 서비스를 이해하고, 목적에 맞는 서비스를 만들기 위한 커스터마이징을 하기 위해선 도움이 될 수 있을 것입니다.

DataContractSerializer

DataContractSerializer 는 WCF 에서 사용하는 기본 serialization 을 수행합니다. (이후 부터는 serialization 대신 "직렬화"를 사용하겠습니다.) 이 serializer 를 사용하기 위해서는 직렬화가 되는 클래스에 [DataContract] 특성을 지정하여 주면 됩니다. 이 방법은 WCF 서비스를 만들때 많이 사용했던 방법이기에 다들 익숙하리라 생각됩니다.

DataContractSerializer 클래스가 어떤 형태로 직렬화를 수행하는지 알아보도록 하겠습니다. 우선, 다음과 같은 클래스를 생성합니다.

```
[DataContract]
public class Employee
{
            public Employee(int employeeID, string firstName, string lastName
            {
            this.EmployeeID = employeeID;
            this.FirstName = firstName;
            this.LastName = lastName;
            }
      [DataMember]
      public int EmployeeID { get; set; }
      [DataMember]
      public string FirstName { get; set; }
      [DataMember]
      public string LastName { get; set; }
}
```

그리고, 콘솔 어플리케이션에서 다음과 같은 코드를 작성합니다.

```
using System.Runtime.Serialization;
using System.IO;

namespace ConsoleApplication1
{
      class Program
      {
            static void Main(string[] args)
            {
                  Employee e = new Employee(101, "Tae kyeom", "Oh");
                  FileStream writer = new FileStream("sample.xml", FileMode.Create);
                  DataContractSerializer ser = new DataContractSerializer(typeof(Employee));
                  ser.WriteObject(writer, e);
                  writer.Close();
            }
      }
}
```

이 코드를 실행을 시켜보면~ 프로젝트의 bin₩Debug 폴더안에 sample.xml 파일이 생성됩니다.

코드는 간단하기에 이해하는데 큰 어려움은 없을 것 같지만, 간단히 설명하자면,,, Employee 클래스 인스턴스를 생성하고, 이 오브젝트를 XML 형태로 직렬화를 시킵니다. 그리고 이에 대한 내용을 sample.xml 파일에 썼구요. 그리고, 직렬화 할 때는, DataContractSerializer 클래스를 사용했습니다

그럼, sample.xml 파일의 내용을 확인 해보겠습니다.

네~ 다음과 같은 내용을 보여주고 있네요, 딱 봐도 Employee 인스턴스가 가지고 있는 데이터를 XML 로 표현해주고 있다는 것을 알 수 있습니다.

```
<Employee xmlns="http://schemas.datacontract.org/2004/07/" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <EmployeeID>101</EmployeeID>
  <FirstName>Tae kyeom</FirstName>
  <LastName>Oh</LastName>
</Employee>
```

여기서 다시 한번 더 되새겨봐야 할 것은 WCF 에서 [DataContract] 특성이 적용된 클래스의 경우 서비스에서 클라이언트로 전달될 때, 위 모습과 같은 형태로 전달된다는 것입니다.

XmlSerializer

XmlSerializer 는 이 전 닷넷 버전에서도 지원해주었던 클래스이며, 이는 ASP.NET 웹 서비스에서 사용하던 직렬화 방법을 제공해줍니다.
이러한 직렬화 방법을 WCF 에서도 사용할 수 있으며, 이는 ASP.NET 웹 서비스와 호환이 가능하다는 장점이 있죠.
또한, 이것은 웹 서비스를 WCF 로의 전환하는 것이 그리 어려운 일이 아니라는 것을 의미하기도 합니다.^^

XmlSerializer 는 public 접근자의 기본 생성자, 필드, 그리고 프로퍼티를 직렬화 시켜줍니다.

Employee 클래스를 다음과 같이 조금 수정해보았습니다.
public 기본 생성자가 필요하기에 추가시켜주었구요, [DataContract]와 [DataMember] 특성을 뺐습니다.

```
public class Employee
{
    public int EmployeeID { get; set; }
      public string FirstName { get; set; }
      public string LastName { get; set; }
      public Employee()
      {
      }
      public Employee(int employeeID, string firstName, string lastName)
      {
          this.EmployeeID = employeeID;
                this.FirstName = firstName;
                this.LastName = lastName;
      }
}
```

그리고, 이 클래스를 XmlSerializer 를 이용하여 직렬화 시켜보겠습니다.

```
using System.IO;
using System.Xml.Serialization;

namespace ConsoleApplication1
{
      class Program
      {
            static void Main(string[] args)
            {
                  Employee e = new Employee(101, "Tae kyeom", "Oh");
                  FileStream writer = new FileStream("sample.xml", FileMode.Create);
                  XmlSerializer ser = new XmlSerializer(typeof(Employee));
                  ser.Serialize(writer, e);
                  writer.Close();
            }
      }
}
```

코드는 아주 간단하죠. DataContractSerializer 클래스를 이용했던 코드와 별반 다를건 없구요, 단지 직렬화 할때 XmlSerializer 클래스를 사용했습니다. 그리고 이 클래스엔 Serialize 메소드를 이용해 직렬화를 수행하죠.
sample.xml 파일을 확인해보면 아래와 같은 모습을 하고 있는 것을 알 수 있을 것입니다.

```
<?xml version="1.0" ?>
<Employee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EmployeeID>101</EmployeeID>
  <FirstName>Tae kyeom</FirstName>
  <LastName>Oh</LastName>
</Employee>
```

결과로 생성 된 xml 파일 역시 앞 예제에서의 결과 파일과 많이 다르진 않지만, 네임스페이스가 조금 바뀐 것을 확인할 수 있습니다.
참고로~ WCF 에서 XmlSerializer 를 사용할 수 있게끔 하기 위해선 서비스 계약(Service Contract)에서 [XmlSerializerFormat] 특성을 적용해주면 됩니다.

DataContractJsonSerializer

DataContractJsonSerializer 는 직렬화의 결과로 JSON 형태를 제공합니다.
JSON 형태의 경우 XML 보다 데이터의 양이 적은 장점이 있고, 이 직렬화를 사용하면, 자바 스크립트를 이용하여 서비스를 호출할 수 있어, 웹 어플리케이션에서 서비스를 쉽게 활용할 수 있다는 장점이 있습니다.

WCF 에서는 REST 서비스를 구현하는 경우 이러한 DataContractJsonSerializer 를 사용합니다.

앞 예제에서 사용한 Employee 클래스를 DataContractJsonSerializer 를 이용하여 직렬화 시켜보도록 하겠습니다.

```
using System.IO;
using System.Runtime.Serialization.Json;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Employee e = new Employee(101, "Tae kyeom", "Oh");
            FileStream writer = new FileStream("sample.txt", FileMode.Create);

            DataContractJsonSerializer ser = new DataContractJsonSerializer(typeof(Employee));
            ser.WriteObject(writer, e);
            writer.Close();
        }
    }
}
```

앞 예제와는 다르게 직렬화 결과를 xml 파일이 아닌 텍스트 파일로 썼습니다. 그리고 DataContractJsonSerializer 클래스를 사용하여 직렬화를 수행했습니다.

다음은 그 결과입니다.