Exemple de serialisation XML d’un bon de commande

Vous pouvez couper et coller l'exemple de code suivant dans un fichier texte renommé avec une extension de nom de fichier en .cs ou .vb. Utilisez le compilateur C# ou Visual Basic pour compiler le fichier. Puis exécutez-le à l'aide du nom du fichier exécutable.

Cet exemple utilise un scénario simple pour illustrer comment l'instance d'un objet est créée et sérialisée dans un flux de données de fichier à l'aide de la méthode Serialize. Le flux de données XML est enregistré dans un fichier, qui est ensuite relu et reconstruit dans une copie de l'objet d'origine à l'aide de la méthode Deserialize.

Dans cet exemple, une classe nommée `PurchaseOrder` est sérialisée puis désérialisée. Une deuxième classe nommée `Address` est également incluse car le champ public nommé `ShipTo` doit avoir la valeur `Address`. De la même façon, une classe `OrderedItem` est incluse car un tableau d'objets `OrderedItem` doit avoir pour valeur le champ `OrderedItems`. Enfin, une classe nommée `Test` contient le code qui sérialise et désérialise les classes.

La méthode `CreatePO` crée les objets de classe `PurchaseOrder`, `Address`et `OrderedItem` et définit les valeurs de champs publics. La méthode construit également une instance de la classe **XmlSerializer** utilisée pour sérialiser et désérialiser `PurchaseOrder`. Notez que le code passe au constructeur le type de la classe qui sera sérialisée. Le code crée également un FileStream utilisé pour écrire le flux de données XML dans un document XML.

La méthode `ReadPo` est un peu plus simple. Elle crée juste des objets à désérialiser et lit leurs valeurs. Comme avec la méthode `CreatePo`, vous devez tout d'abord construire un **XmlSerializer**, en passant au constructeur le type de la classe à désérialiser. De même, un FileStream est requis pour lire le document XML. Pour désérialiser les objets, appelez la méthode **Deserialize** avec le **FileStream** en tant qu'argument. L'objet désérialisé doit être converti en une variable d'objet de type `PurchaseOrder`. Le code lit ensuite les valeurs du `PurchaseOrder` désérialisé. Notez que vous pouvez également lire le fichier PO.xml créé pour consulter le résultat XML réel.

C#

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

// The XmlRootAttribute allows you to set an alternate name
// (PurchaseOrder) for the XML element and its namespace. By
// default, the XmlSerializer uses the class name. The attribute
// also allows you to set the XML namespace for the element. Lastly,
// the attribute sets the IsNullable property, which specifies whether
// the xsi:null attribute appears if the class instance is set to
// a null reference.
[XmlRootAttribute("PurchaseOrder", Namespace="http://www.cpandl.com",
IsNullable = false)]
public class PurchaseOrder
{
    public Address ShipTo;
    public string OrderDate;
    // The XmlArrayAttribute changes the XML element name
    // from the default of "OrderedItems" to "Items".
```

```
    [XmlArrayAttribute("Items")]
    public OrderedItem[] OrderedItems;
    public decimal SubTotal;
    public decimal ShipCost;
    public decimal TotalCost;
}

public class Address
{
    // The XmlAttribute instructs the XmlSerializer to serialize the
    // Name field as an XML attribute instead of an XML element (the
    // default behavior).
    [XmlAttribute]
    public string Name;
    public string Line1;

    // Setting the IsNullable property to false instructs the
    // XmlSerializer that the XML attribute will not appear if
    // the City field is set to a null reference.
    [XmlElementAttribute(IsNullable = false)]
    public string City;
    public string State;
    public string Zip;
}

public class OrderedItem
{
    public string ItemName;
    public string Description;
    public decimal UnitPrice;
    public int Quantity;
    public decimal LineTotal;

    // Calculate is a custom method that calculates the price per item
    // and stores the value in a field.
    public void Calculate()
    {
        LineTotal = UnitPrice * Quantity;
    }
}

public class Test
{
    public static void Main()
    {
        // Read and write purchase orders.
        Test t = new Test();
        t.CreatePO("po.xml");
        t.ReadPO("po.xml");
    }

    private void CreatePO(string filename)
    {
        // Creates an instance of the XmlSerializer class;
        // specifies the type of object to serialize.
        XmlSerializer serializer =
        new XmlSerializer(typeof(PurchaseOrder));
        TextWriter writer = new StreamWriter(filename);
        PurchaseOrder po=new PurchaseOrder();

        // Creates an address to ship and bill to.
```

```
        Address billAddress = new Address();
        billAddress.Name = "Teresa Atkinson";
        billAddress.Line1 = "1 Main St.";
        billAddress.City = "AnyTown";
        billAddress.State = "WA";
        billAddress.Zip = "00000";
        // Sets ShipTo and BillTo to the same addressee.
        po.ShipTo = billAddress;
        po.OrderDate = System.DateTime.Now.ToLongDateString();

        // Creates an OrderedItem.
        OrderedItem i1 = new OrderedItem();
        i1.ItemName = "Widget S";
        i1.Description = "Small widget";
        i1.UnitPrice = (decimal) 5.23;
        i1.Quantity = 3;
        i1.Calculate();

        // Inserts the item into the array.
        OrderedItem [] items = {i1};
        po.OrderedItems = items;
        // Calculate the total cost.
        decimal subTotal = new decimal();
        foreach(OrderedItem oi in items)
        {
            subTotal += oi.LineTotal;
        }
        po.SubTotal = subTotal;
        po.ShipCost = (decimal) 12.51;
        po.TotalCost = po.SubTotal + po.ShipCost;
        // Serializes the purchase order, and closes the TextWriter.
        serializer.Serialize(writer, po);
        writer.Close();
    }

    protected void ReadPO(string filename)
    {
        // Creates an instance of the XmlSerializer class;
        // specifies the type of object to be deserialized.
        XmlSerializer serializer = new
XmlSerializer(typeof(PurchaseOrder));
        // If the XML document has been altered with unknown
        // nodes or attributes, handles them with the
        // UnknownNode and UnknownAttribute events.
        serializer.UnknownNode+= new
        XmlNodeEventHandler(serializer_UnknownNode);
        serializer.UnknownAttribute+= new
        XmlAttributeEventHandler(serializer_UnknownAttribute);

        // A FileStream is needed to read the XML document.
        FileStream fs = new FileStream(filename, FileMode.Open);
        // Declares an object variable of the type to be deserialized.
        PurchaseOrder po;
        // Uses the Deserialize method to restore the object's state
        // with data from the XML document. */
        po = (PurchaseOrder) serializer.Deserialize(fs);
        // Reads the order date.
        Console.WriteLine ("OrderDate: " + po.OrderDate);

        // Reads the shipping address.
        Address shipTo = po.ShipTo;
```

```
        ReadAddress(shipTo, "Ship To:");
        // Reads the list of ordered items.
        OrderedItem [] items = po.OrderedItems;
        Console.WriteLine("Items to be shipped:");
        foreach(OrderedItem oi in items)
        {
            Console.WriteLine("\t"+
            oi.ItemName + "\t" +
            oi.Description + "\t" +
            oi.UnitPrice + "\t" +
            oi.Quantity + "\t" +
            oi.LineTotal);
        }
        // Reads the subtotal, shipping cost, and total cost.
        Console.WriteLine(
        "\n\t\t\t\t\t Subtotal\t" + po.SubTotal +
        "\n\t\t\t\t\t Shipping\t" + po.ShipCost +
        "\n\t\t\t\t\t Total\t\t" + po.TotalCost
        );
    }

    protected void ReadAddress(Address a, string label)
    {
        // Reads the fields of the Address.
        Console.WriteLine(label);
        Console.Write("\t"+
        a.Name +"\n\t" +
        a.Line1 +"\n\t" +
        a.City +"\t" +
        a.State +"\n\t" +
        a.Zip +"\n");
    }

    protected void serializer_UnknownNode
    (object sender, XmlNodeEventArgs e)
    {
        Console.WriteLine("Unknown Node:" +   e.Name + "\t" + e.Text);
    }

    protected void serializer_UnknownAttribute
    (object sender, XmlAttributeEventArgs e)
    {
        System.Xml.XmlAttribute attr = e.Attr;
        Console.WriteLine("Unknown attribute " +
        attr.Name + "='" + attr.Value + "'");
    }
}
```

Le résultat XML peut se présenter comme suit.

```
<?xml version="1.0" encoding="utf-8"?>
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.cpandl.com">
    <ShipTo Name="Teresa Atkinson">
        <Line1>1 Main St.</Line1>
        <City>AnyTown</City>
        <State>WA</State>
        <Zip>00000</Zip>
    </ShipTo>
    <OrderDate>Wednesday, June 27, 2001</OrderDate>
    <Items>
```

```
        <OrderedItem>
            <ItemName>Widget S</ItemName>
            <Description>Small widget</Description>
            <UnitPrice>5.23</UnitPrice>
            <Quantity>3</Quantity>
            <LineTotal>15.69</LineTotal>
        </OrderedItem>
    </Items>
    <SubTotal>15.69</SubTotal>
    <ShipCost>12.51</ShipCost>
    <TotalCost>28.2</TotalCost>
</PurchaseOrder>
```