using System;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

using System.Xml.Serialization;

using System.Runtime.Serialization.Json;

using System.Xml;

using System.Xml.Linq;

using System.Collections.Generic;

using System.Linq;

//Сериализация позволяет разработчику сохранять состояние объекта и воссоздавать его при необходимости.

//Это полезно для длительного хранения объектов или для обмена данными.

namespace lab14

{

class Program

{

static void Main(string[] args)

{

Point point = new Point(5, 9);

Point point\_2 = new Point(7, 23);

Point[] points = { point, point\_2 };

Console.WriteLine("\_\_\_\_\_\_\_\_\_Бинарная сериализация\_\_\_\_\_\_\_\_\_");

BinaryFormatter formatter = new BinaryFormatter();//Сериализует и десериализует объект или полный граф связанных объектов в двоичном формате

using (FileStream fs = new FileStream("point.dat", FileMode.OpenOrCreate))//FileStream - синхронные и асинхронные операции чтения и записи. FileMode.OpenOrCreate - Указывает, что операционная система должна открыть файл, если он существует, в противном случае должен быть создан новый файл

{

formatter.Serialize(fs, point);//Сериализация — это процесс преобразования объекта в поток байтов для сохранения или передачи в память, базу данных или файл.

Console.WriteLine("Объект сериализован");

formatter.Serialize(fs, points);//Сериализация — это процесс преобразования объекта в поток байтов для сохранения или передачи в память, базу данных или файл.

Console.WriteLine("Объект сериализован");

}

using (FileStream fs = new FileStream("point.dat", FileMode.OpenOrCreate))//FileStream - синхронные и асинхронные операции чтения и записи. FileMode.OpenOrCreate - Указывает, что операционная система должна открыть файл, если он существует, в противном случае должен быть создан новый файл

{

Point newPoint = (Point)formatter.Deserialize(fs);//получить из потока байтов ранее сохраненный объект

Console.WriteLine("Объект десериализован");

Console.WriteLine($"X: {newPoint.X}, Y: {newPoint.Y}");

Point[] newPoint2 = (Point[])formatter.Deserialize(fs);//получить из потока байтов ранее сохраненный объект

Console.WriteLine("Объект десериализован");

foreach (Point p in newPoint2)

{

Console.WriteLine($"X: {p.X}, Y: {p.Y}");

}

}

Console.WriteLine();

Console.WriteLine("\_\_\_\_\_\_\_\_\_Сериализация в XML\_\_\_\_\_\_\_\_\_");

XmlSerializer serializer = new XmlSerializer(typeof(Point));//Сериализует и десериализует объекты в XML-документы и из них. XmlSerializer позволяет контролировать способ кодирования объектов в XML

XmlSerializer serializer1 = new XmlSerializer(typeof(Point[]));//Сериализует и десериализует объекты в XML-документы и из них. XmlSerializer позволяет контролировать способ кодирования объектов в XML

using (FileStream fs = new FileStream("point2.xml", FileMode.OpenOrCreate))////FileStream - синхронные и асинхронные операции чтения и записи. FileMode.OpenOrCreate - Указывает, что операционная система должна открыть файл, если он существует, в противном случае должен быть создан новый файл

{

serializer.Serialize(fs, point);

Console.WriteLine("Объект сериализован");

}

using (FileStream fs = new FileStream("point2.xml", FileMode.OpenOrCreate))////FileStream - синхронные и асинхронные операции чтения и записи. FileMode.OpenOrCreate - Указывает, что операционная система должна открыть файл, если он существует, в противном случае должен быть создан новый файл

{

Point newPoint = (Point)serializer.Deserialize(fs);

Console.WriteLine("Объект десериализован");

Console.WriteLine($"X: {newPoint.X}, Y: {newPoint.Y}");

}

using (FileStream fs = new FileStream("point3.xml", FileMode.OpenOrCreate))////FileStream - синхронные и асинхронные операции чтения и записи. FileMode.OpenOrCreate - Указывает, что операционная система должна открыть файл, если он существует, в противном случае должен быть создан новый файл

{

serializer1.Serialize(fs, points);

Console.WriteLine("Объект сериализован");

}

using (FileStream fs = new FileStream("point3.xml", FileMode.OpenOrCreate))////FileStream - синхронные и асинхронные операции чтения и записи. FileMode.OpenOrCreate - Указывает, что операционная система должна открыть файл, если он существует, в противном случае должен быть создан новый файл

{

Point[] newPoint2 = (Point[])serializer1.Deserialize(fs);

Console.WriteLine("Объект десериализован");

foreach (Point p in newPoint2)

{

Console.WriteLine($"X: {p.X}, Y: {p.Y}");

}

}

Console.WriteLine();

Console.WriteLine("\_\_\_\_\_\_\_\_\_Сериализация в JSON\_\_\_\_\_\_\_\_\_");

Person person1 = new Person("Tom", 29);

Person person2 = new Person("Will", 25);

Person[] people = { person1, person2 };

DataContractJsonSerializer jsonFormatter = new DataContractJsonSerializer(typeof(Person[]));//Сериализует объекты в нотацию объектов JavaScript (JSON) и десериализует данные JSON в объекты. Этот класс не наследуется

using (FileStream fs = new FileStream("people.json", FileMode.OpenOrCreate))//FileStream - синхронные и асинхронные операции чтения и записи. FileMode.OpenOrCreate - Указывает, что операционная система должна открыть файл, если он существует, в противном случае должен быть создан новый файл

{

jsonFormatter.WriteObject(fs, people);//Записывает все данные объекта (начальный XML-элемент, содержимое и закрывающий элемент) в XML-документ или поток

}

using (FileStream fs = new FileStream("people.json", FileMode.OpenOrCreate))//FileStream - синхронные и асинхронные операции чтения и записи. FileMode.OpenOrCreate - Указывает, что операционная система должна открыть файл, если он существует, в противном случае должен быть создан новый файл

{

Person[] newpeople = (Person[])jsonFormatter.ReadObject(fs);//Считывает XML-поток и возвращает десериализованный объект

foreach (Person p in newpeople)

{

Console.WriteLine("Имя: {0} --- Возраст: {1}", p.Name, p.Age);

}

}

Console.WriteLine();

Console.WriteLine("\_\_\_\_\_\_\_\_\_Linq to XML\_\_\_\_\_\_\_\_\_");

//создание документа

XDocument xdoc = new XDocument(new XElement("users",//создается документ, а затем в него добавляется комментарий и элемент

new XElement("user",

new XAttribute("name", "Bill Gates"),

new XElement("company", "Microsoft"),

new XElement("age", "48")),

new XElement("user",

new XAttribute("name", "Larry Page"),

new XElement("company", "Google"),

new XElement("price", "42"))));

xdoc.Save("users.xml");

IEnumerable<XElement> elements = xdoc.Descendants("user"); //Предоставляет перечислитель, который поддерживает простой перебор элементов неуниверсальной коллекции(kollectiya)//получаем отдельный эл-т

foreach (XElement e in elements)

Console.WriteLine("Элемент {0} : значение = {1}", e.Name, e.Value);

IEnumerable<XElement> elements2 = xdoc.Descendants("user").Where(e => ((string)e.Element("company")) == "Microsoft");

foreach (XElement e in elements2)

Console.WriteLine("Элемент {0} : значение = {1}", e.Name, e.Value);

Console.WriteLine();

//Используя XPath напишите два селектора для вашего XML документа

Console.WriteLine("\_\_\_\_\_\_\_\_\_XPath\_\_\_\_\_\_\_\_\_");

XmlDocument xml = new XmlDocument();

xml.Load("users.xml");

XmlElement xRoot = xml.DocumentElement;//только компании, все узлы корневого элемента//Возвращает корень XmlElement для документа

XmlNodeList list = xRoot.SelectNodes("//user/company");//Выбирает список узлов в соответствии с выражением XPath

foreach (XmlNode nodes in list)

{

Console.WriteLine(nodes.InnerText);//Возвращает или задает связанные значения узла и всех его дочерних узлов

}

XmlNode childnode = xRoot.SelectSingleNode("user[company='Microsoft']");//узел, у которого вложенный элемент "company" имеет значение "Microsoft"

if (childnode != null)

Console.WriteLine(childnode.OuterXml);

}

}

}

**1. Что такое сериализация, дисериализация?**

**Сериализация** – процесс преобразования объектов в поток байт  
**Дисериализация** – получение из потока байт сохраненного объекта

**2. Какие существуют форматы сериализации? Поясните структуру для каждого формата. Какие классы для работы с ними сущ. в .NET?**

Объекты серил. типов м. сохр. в поток в различ. форматах  
\* бинарный  
\* SOAP  
\* xml  
\* JSON

**класс BinaryFormatter**  
(функционал определен в инт. IFormatter)  
SerializationBinder Binder { get; set; }  
StreamingContext Context { get; set; }  
ISurrogateSelector SurrogateSelector { get; set; }  
object Deserialize( Stream serializationStream);  
void Serialize( Stream serializationStream, object graph); //поток сер-ции и сер. объект

**класс SoapFormatter**

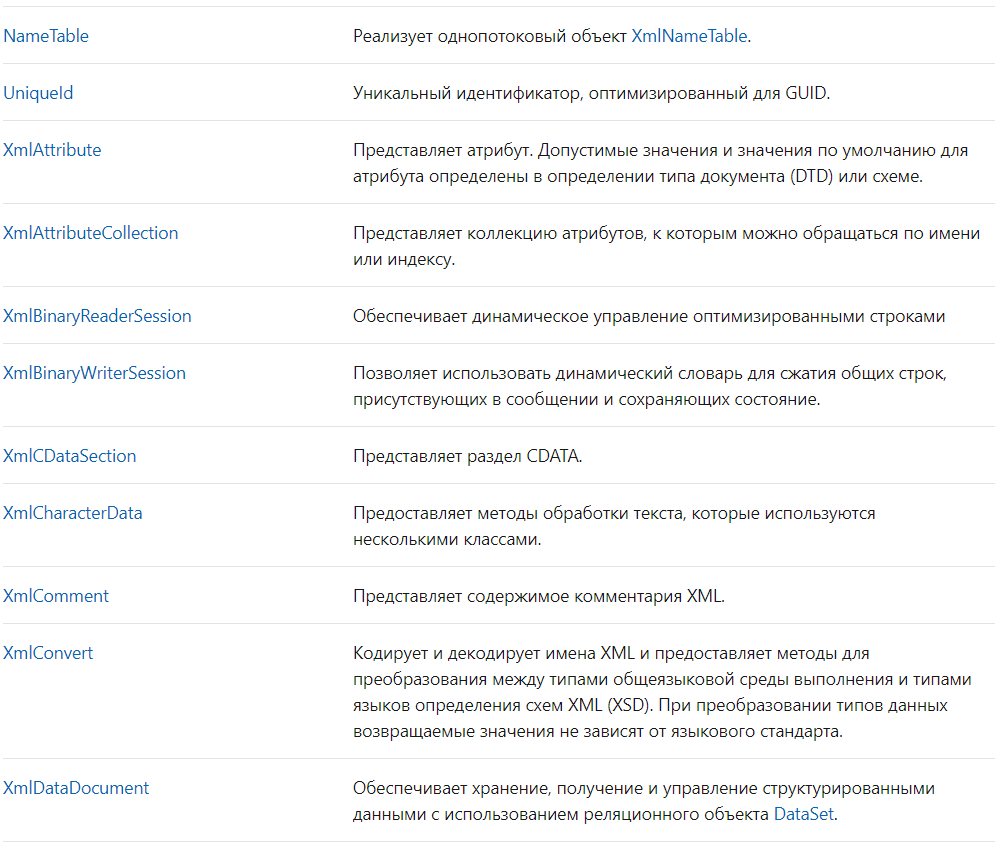
SOAP (Simple Object Access Protocol) -   
станд. процесс вызова методов независ. от платформы и ОС

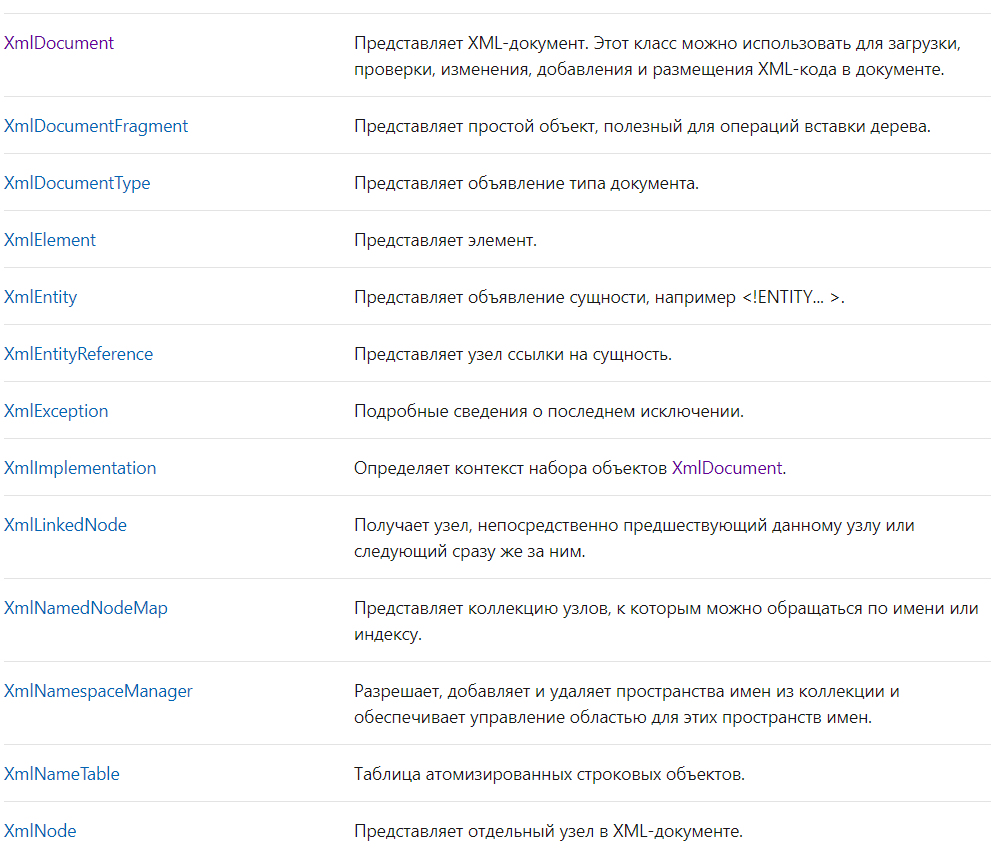
**класс XmlSerializer**

Ограничения:  
\* класс д. иметь к-р без парам  
\* сериализации подлежат только открытые члены  
\* приватные д-е, не предст. св-вами, игнорируются  
\* требует указания типа

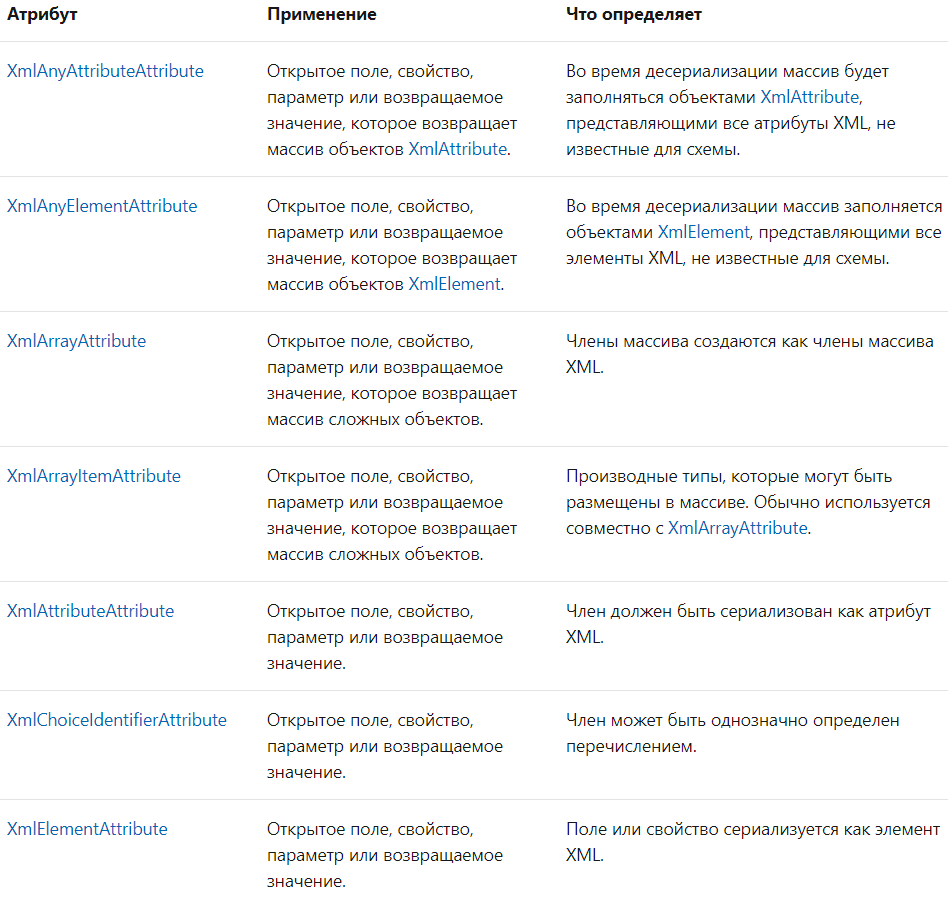
класс DataContractJsonSerializer

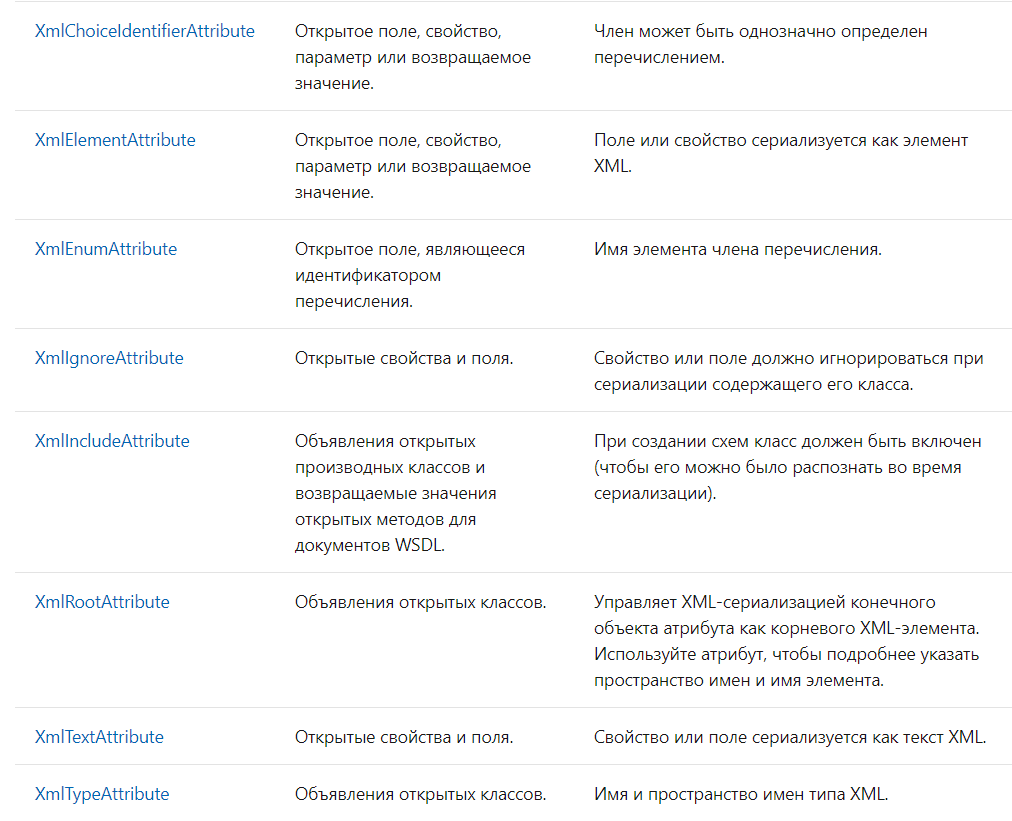
**3. Какие классы сущ. в пр-ве имен System.Xml?**

****

****

**4. Какие атрибуты исп. для настройки XML сериализации?**

****

****

**5. В чем отличие BinaryFormater или SoapFormater?**

**BinaryFormatter** – сериализация состояния объекта в поток, используя 2сс формат  
**SoapFormatter** – сохр. сост. объекта в виде сообщений SOAP

**6. Что такое сериализация контрактов д-х, контракт д-х?**

**Контракт д-х** – это тип (класс или структура), объект которого описывает инф. фрагмент (открытые поля и св-ва) – один из механизмов сериализации

Осн. формат хранения контрактов – xml  
в кач. контракта д-х исп. класс, инф.фрагмент образуют открытые поля и св-ва  
видимость не им. значения

**7. Где и для чего исп. атрибуты [OnSerializing], [OnSerialized],   
[OnDeserializing], [OnDeserialized]?**

конструкторы не вызываются  
Если нужна особая инициализация объекта и восстановление несохраненных полей [OnSerializing], [OnSerialized], [OnDeserializing], [OnDeserialized], вызываются CLR автоматически до и после сериализации или десериализации.

Метод, обозначенный атрибутом, должен принимать объект класса StreamingContext и не возвр. значений

[Serializable]  
 public class ParkIT {  
 [Onserializing]  
 private void Beforeserialization( StreamingContext context) {  
 CalculateSmt(); }

[OnDeserialized]  
 private void AfterDeserialization( StreamingContext context {  
 FindSmt(); }  
 }  
 [Serializable]  
 class Student {  
 private String name;  
 private int age;  
 [NonSerialized]  
 private String address;

**8. Что такое XPath? Приведите пример.**

XPath – язык запросов в XML  
Позв. выбир. эл-ты, соотв. опр. селектору:  
. – выбор тек. узла  
.. - выбор род. узла  
\* - выбор всех дочерних узлов  
user - выбор всех узлов с опр. именем (“user”)  
@name - выбор атрибута тек. узла. после @ назв. атрибута  
@+ - выбор всех атриб. тек. узла

Д-вие XPath осн. на 2 методах:  
\* SelectSingleNode() – выбор ед. узла из выборки  
\* SelectNodes()

**9. Какие возможности дает LINQ to Xml. Приведите примеры**

XDocument xdoc = XDocument.Load(“phones.xml”);  
var items = from xe in xdoc.Element(“phones”).Elements(“phone”)  
 where xe.Element(“company”).Value==”Samsung”  
 select new Phone  
 Name = xe.Attribute(“name”).Value,  
 Price = xe.Element(“price”).Value