1. Что такое сериализация, десериализация?

**Сериализация** представляет процесс преобразования какого-либо объекта в поток байтов. После преобразования мы можем этот поток байтов или записать на диск или сохранить его временно в памяти. А при необходимости можно выполнить обратный процесс - **десериализацию**, то есть получить из потока байтов ранее сохраненный объект.

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

Бинарный, SOAP, xml, JSON

Для каждого формата предусмотрен свой класс: для сериализации в бинарный формат - класс BinaryFormatter, для формата SOAP - класс SoapFormatter, для xml - XmlSerializer, для json - DataContractJsonSerializer.

3. Какие классы существуют в пространстве имен System.Xml?

* **XmlNode**: представляет узел xml. В качестве узла может использоваться весь документ, так и отдельный элемент
* **XmlDocument**: представляет весь xml-документ
* **XmlElement**: представляет отдельный элемент. Наследуется от класса XmlNode
* **XmlAttribute**: представляет атрибут элемента
* **XmlText**: представляет значение элемента в виде текста, то есть тот текст, который находится в элементе между его открывающим и закрывающим тегами
* **XmlComment**: представляет комментарий в xml
* **XmlNodeList**: используется для работы со списком узлов

4. Какие атрибуты используются для настройки XML сериализации?

| **Атрибут** | **Применение** | **Что определяет** |
| --- | --- | --- |
| [XmlAnyAttributeAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlanyattributeattribute) | Открытое поле, свойство, параметр или возвращаемое значение, которое возвращает массив объектов [XmlAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.xmlattribute). | Во время десериализации массив будет заполняться объектами [XmlAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.xmlattribute), представляющими все атрибуты XML, не известные для схемы. |
| [XmlAnyElementAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlanyelementattribute) | Открытое поле, свойство, параметр или возвращаемое значение, которое возвращает массив объектов [XmlElement](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.xmlelement). | Во время десериализации массив заполняется объектами [XmlElement](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.xmlelement), представляющими все элементы XML, не известные для схемы. |
| [XmlArrayAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlarrayattribute) | Открытое поле, свойство, параметр или возвращаемое значение, которое возвращает массив сложных объектов. | Члены массива создаются как члены массива XML. |
| [XmlArrayItemAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlarrayitemattribute) | Открытое поле, свойство, параметр или возвращаемое значение, которое возвращает массив сложных объектов. | Производные типы, которые могут быть размещены в массиве. Обычно используется совместно с [XmlArrayAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlarrayattribute). |
| [XmlAttributeAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlattributeattribute) | Открытое поле, свойство, параметр или возвращаемое значение. | Член должен быть сериализован как атрибут XML. |
| [XmlChoiceIdentifierAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlchoiceidentifierattribute) | Открытое поле, свойство, параметр или возвращаемое значение. | Член может быть однозначно определен перечислением. |
| [XmlElementAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlelementattribute) | Открытое поле, свойство, параметр или возвращаемое значение. | Поле или свойство сериализуется как элемент XML. |
| [XmlEnumAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlenumattribute) | Открытое поле, являющееся идентификатором перечисления. | Имя элемента члена перечисления. |
| [XmlIgnoreAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlignoreattribute) | Открытые свойства и поля. | Свойство или поле должно игнорироваться при сериализации содержащего его класса. |
| [XmlIncludeAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlincludeattribute) | Объявления открытых производных классов и возвращаемые значения открытых методов для документов WSDL. | При создании схем класс должен быть включен (чтобы его можно было распознать во время сериализации). |
| [XmlRootAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmlrootattribute) | Объявления открытых классов. | Управляет XML-сериализацией конечного объекта атрибута как корневого XML-элемента. Используйте атрибут, чтобы подробнее указать пространство имен и имя элемента. |
| [XmlTextAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmltextattribute) | Открытые свойства и поля. | Свойство или поле сериализуется как текст XML. |
| [XmlTypeAttribute](https://docs.microsoft.com/ru-ru/dotnet/api/system.xml.serialization.xmltypeattribute) | Объявления открытых классов. | Имя и пространство имен типа XML. |

5. В чем отличие BinaryFormatter или SoapFormatter?

Класс форматирования BinaryFormatter реализует двоичный закрытый метод сериализации, класс SoapFormatter – текстовый и открытый, основанный на спецификации кодирования SOAP‑RPC

6. Что такое сериализация контрактов данных, контракт данных?

*Контракт данных* – это тип (класс или структура), описывающий информационный фрагмент. Если в качестве контракта данных используется обычный класс, информационный фрагмент образуют открытые поля и свойства. Можно пометить тип атрибутом [DataContract].

Атрибут [DataContract] имеет свойства Name и Namespace для указания имени и пространства имён корневого XML-элемента. У атрибута [DataMember] есть свойство Name, а также свойства Order (порядок сериализации членов контракта), IsRequired (обязательный элемент в сериализованном потоке), EmitDefaultValue (запись в поток значений элемента по умолчанию).

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

|  |  |
| --- | --- |
| **[OnDeserialized]** | Этот атрибут позволяет указать метод, который будет вызван немедленно после десериализации объекта |
| **[OnDeserializing]** | Этот атрибут позволяет указать метод, который будет вызван перед процессом десериализации |
| **[OnSerialized]** | Этот атрибут позволяет указать метод, который будет вызван немедленно после того, как объект сериализован |
| **[OnSerializing]** | Этот атрибут позволяет указать метод, который будет вызван перед процессом сериализации |

8. Что такое XPath? Приведите пример.

Язык запросов к элементам XML-документа. В XPath существует семь видов узлов: элемент, атрибут, текст, пространство имён, инструкции обработки, комментарии и узлы документа. XML-документы обрабатываются в виде деревьев узлов. Верхний элемент дерева называется корневым элементом. Посмотрите на следующий документ XML:

9. Какие возможности дает LINQ to Xml. Приведите примеры

предназначенный для создания, модификации, чтения и обхода XML-документов, а также для выполнения запросов LINQ на отдельном XML-объекте

Еще один подход к работе с Xml представляет технология LINQ to XML. Вся функциональность LINQ to XML содержится в пространстве именSystem.Xml.Linq. Рассмотрим основные классы этого пространства имен:

* *XAttribute: представляет атрибут xml-элемента*
* *XComment: представляет комментарий*
* *XDocument: представляет весь xml-документ*
* *XElement: представляет отдельный xml-элемент*

Ключевым классом является XElement, который позволяет получать вложенные элементы и управлять ими. Среди его методов можно отметить следующие:

* *Add(): добавляет новый атрибут или элемент*
* *Attributes(): возвращает коллекцию атрибутов для данного элемента*
* *Elements(): возвращает все дочерние элементы данного элемента*
* *Remove(): удаляет данный элемент из родительского объекта*
* *RemoveAll(): удаляет все дочерние элементы и атрибуты у данного элемента*

