**Вопросы к лабораторной работе №13:**

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

**Сериализация**— процесс перевода какой-либо структуры данных в последовательность битов. Обратной к операции сериализации является операция…

**Десериализации**— восстановление начального состояния структуры данных из битовой последовательности.

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

В .NET можно использовать следующие форматы:

* **бинарный** (**Двоичный** **файл** — в широком смысле: последовательность произвольных байтов.)
* **SOAP** (SOAP-сообщение представляет собой XML-документ; сообщение состоит из трех основных элементов: конверт (SOAP Envelope), заголовок (SOAP Header) и тело (SOAP Body).)
* **xml** — расширяемый язык разметки, он описывает документ и частично описывает поведение программ, читающих **XML**-документы. **Формат** языка разрабатывался с расчетом применения широкого его в Интернете.
* **JSON** (JSON-текст представляет собой (в закодированном виде) одну из двух структур: Набор пар ключ: значение. В различных языках это реализовано как запись, структура, словарь, хеш-таблица, список с ключом или ассоциативный массив. Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.)

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

1. **Какие классы существуют в пространстве имен System.Xml?**

В пространстве имен System.Xml содержатся классы для работы с XML. Основные из них:

* XmlDocument

Позволяет создавать, загружать и изменять XML-документы.

* XmlElement

Представляет элемент XML-документа.

* XmlAttribute

Представляет атрибут элемента XML.

* XmlNode

Базовый класс для всех узлов XML-документа.

* XmlText

Представляет текстовый узел в XML.

* XmlReader

Позволяет читать XML-данные в потоковом режиме.

* XmlWriter

Позволяет записывать XML-данные в потоковом режиме.

* XmlSchema

Для работы со схемами XML (XSD).

* XmlNamespaceManager

Управляет пространствами имен XML.

* XmlSerializer

Для сериализации объектов в XML и десериализации из XML.

Эти классы поддерживают чтение, запись, анализ и изменение данных в формате XML.

1. **Какие атрибуты используются для настройки XML сериализации?**

Для настройки XML сериализации в .NET используются атрибуты из пространства имен System.Xml.Serialization:

[XmlElement] — указывает, что поле или свойство должно быть сериализовано как элемент XML.

[XmlAttribute] — указывает, что поле или свойство должно быть сериализовано как атрибут XML.

[XmlArray] — указывает, что поле или свойство является массивом или коллекцией и должно быть сериализовано как XML-массив.

[XmlArrayItem] — указывает, какие элементы должны быть сериализованы в массив XML.

[XmlIgnore] — указывает, что поле или свойство не должно быть сериализовано.

[XmlRoot] — задаёт корневой элемент XML-документа.

[XmlType] — позволяет настроить тип сериализации.

Эти атрибуты позволяют контролировать, как объекты будут преобразовываться в XML и обратно.

1. **В чем отличие BinaryFormatter или SoapFormatter?**

BinaryFormatter: сериализует в двоичный формат, быстро, компактно, но зависим от платформы .NET и небезопасен. Устарел.

SoapFormatter: сериализует в текстовый XML (SOAP), читабельно и межплатформенно, но медленно и занимает больше места. Тоже устарел.

Протокол **SOAP** (Simple Object Access Protocol — простой протокол доступа к объектам) определяет стандартный процесс вызова методов в независящей от платформы и операционной системы манере.

1. **Что такое сериализация контрактов данных, контракт данных?**

**Сериализация контрактов данных** — это процесс преобразования объектов в XML или JSON для обмена данными между системами, в соответствии с определенными правилами (контрактами).

**Контракт данных** — это соглашение (описание), которое определяет структуру данных, подлежащих сериализации, включая поля, типы данных и их представление. Используется в технологиях, таких как WCF.

1. **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 |

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

**XPath** (XML Path Language) — язык запросов к элементам XML-документа.

XDocument xdoc = XDocument.Load(@"..\CharactersList\_xml.xml");

var items = from i in xdoc.Element("ArrayOfCharacters").Elements("Characters")

where Convert.ToInt32(i.Element("Attack").Value) > 100

select i;

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

Чтобы начать работу с имеющимся **xml**-файлом, надо сначала загрузить его с помощью статического метода **XDocument.Load(),** в который передается путь к файлу.

Поскольку **xml** хранит иерархически выстроенные элементы, то и для доступа к элементам надо идти начиная с высшего уровня в этой иерархии и далее вниз.

Значение простых элементов, которые содержат один текст, можно получить с помощью свойства **Value**

Сочетая операторы Linq и LINQ to XML можно довольно просто извлечь из документа данные и затем обработать их. Например, имеется следующий класс:

|  |  |
| --- | --- |
| 1  2  3  4  5 | class Phone  {  public string Name { get; set; }  public string Price { get; set; }  } |

Создадим на основании данных в xml объекты этого класса:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | 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  };  foreach (var item in items)  Console.WriteLine($"{item.Name} - {item.Price}"); |