1. Сериализация и десериализация:

- Сериализация (Serialization): Процесс преобразования объекта в поток байт.

- Десериализация (Deserialization): Процесс восстановления объекта из потока байт.

2. Форматы сериализации в .NET:

* + бинарный -BinaryFormatter
  + SOAP -SoapFormatter
  + xml - XmlSerializer
  + JSON - DataContractJsonSerializer

3. Классы в пространстве имен System.Xml:

- Пространство имен System.Xml содержит классы для работы с XML в .NET, такие как XmlDocument, XmlElement, XmlAttribute и другие.

4. Атрибуты для настройки XML сериализации:

XmlRoot - применяется к типу и задаёт корневой элемент в XML-файле

XmlElement - настраивается имя и пространство имён XML-элемента

XmlText - Свойство или поле сериализуется как текст XML.

XmlIgnore - Поля и свойства, которые не должны сохраняться

5. Отличие BinaryFormatter от SoapFormatter:

- BinaryFormatter использует бинарный формат для сериализации, в то время как SoapFormatter создает XML-представление объектов в формате SOAP.

6. Сериализация контрактов данных и контракт данных:

Разница между DataContract и Serializable связана с тем, как эти атрибуты взаимодействуют с механизмами сериализации в .NET.

DataContract:

DataContract — это атрибут из пространства имен System.Runtime.Serialization, предназначенный для работы с Data Contract Serialization, которая является частью Windows Communication Foundation (WCF).

Используется для явного определения контрактов данных для сериализации и десериализации объектов.

Позволяет более тонко настроить процесс сериализации с использованием атрибута DataMember, где можно указать имя элемента в сериализованном виде, порядок, обязательность и другие характеристики.

Часто используется при создании служб WCF для передачи данных между клиентом и службой.

Serializable:

Serializable — это атрибут из пространства имен System, используемый для общей сериализации объектов в .NET.

Используется для обозначения класса, который может быть сериализован с использованием стандартных механизмов сериализации, таких как BinaryFormatter или XML-сериализаторы.

Не предоставляет тонкой настройки процесса сериализации, но обеспечивает базовую поддержку сериализации для объектов.

Коротко говоря, DataContract предоставляет более гибкий и контролируемый способ сериализации данных, особенно в контексте служб WCF, в то время как Serializable предоставляет более простой способ для общего использования. В большинстве современных приложений рекомендуется использовать DataContract, особенно при работе с WCF и при необходимости более точного управления процессом сериализации.

7. Атрибуты [OnSerializing], [OnSerialized], [OnDeserializing], [OnDeserialized]:

Атрибуты [OnSerializing], [OnSerialized], [OnDeserializing], и [OnDeserialized] в C# используются для указания методов, которые вызываются в различных этапах процесса сериализации и десериализации объектов с использованием BinaryFormatter. [OnSerializing] вызывается перед сериализацией, [OnSerialized] - после сериализации, [OnDeserializing] - перед десериализацией, и [OnDeserialized] - после десериализации. Эти методы позволяют выполнять дополнительные действия перед или после процессов сериализации и десериализации.

8. XPath:

- XPath - это язык запросов для выбора узлов из XML-документа. Пример:

9. LINQ to XML:

- LINQ to XML предоставляет средства для работы с XML в стиле LINQ. Некоторые возможности:

- Создание XML: XElement, XAttribute.

- Запросы к XML: использование LINQ-запросов.

- Манипуляции с XML: добавление, удаление, изменение элементов.

XML

* + Класс должен иметь конструктор без параметров
  + Сериализации подлежат только открытые члены
  + Приватные данные, не представленные свойствами, игнорируются.
  + Требует указания типа

JSON

* Объект, который подвергается десериализации, должен иметь либо конструктор без параметров, либо конструктор, для всех параметров которого в десериализуемом json-объекте есть значения
* Сериализации подлежат только публичные свойства объекта

Для классов BinaryFormatter и SoapFormatter сам функционал сериализации определен в интерфейсе **IFormatter**:

Хотя классы BinaryFormatter и SoapFormatter по-разному реализуют данный интерфейс, но общий функционал будет тот же: для сериализации будет использоваться метод Serialize, который в качестве параметров принимает поток, куда помещает сериализованные данные (например, бинарный файл), и объект, который надо сериализовать. А для десериализации будет применяться метод Deserialize, который в качестве параметра принимает поток с сериализованными данными.

Класс XmlSerializer не реализует интерфейс IFormatter и по функциональности в целом несколько отличается от BinaryFormatter и SoapFormatter, но и он также предоставляет для сериализации метод Serialize, а для десериализации Deserialize. И в этом плане очень легко при необходимости перейти от одного способа сериализации к другому.