Первый основан на последовательной работе с документом. Например, документ последовательно читается, и каждый прочитанный элемент анализируется. Второй подход предполагает чтение документа в память целиком и представление документа в виде иерархии объектов определенных типов.

Для работы с XML обычно используются классы из пространства имен `System.Xml`. Для последовательного чтения XML-документов применяется класс `XmlReader` и его наследники: `XmlTextReader` (чтение на основе текстового потока), `XmlNodeReader` (разбор XML из объектов `XmlNode`) и `XmlValidatingReader` (при чтении производится проверка схемы документа). Конструктор класса перегружен и позволяет создать объект на основе указанного файла или текстового потока:

```
var reader = new XmlTextReader("Students.xml");
```

После создания объект `XmlTextReader` извлекает XML-конструкции из потока при помощи метода `Read()`. Тип текущей конструкции можно узнать, используя свойство `NodeType`, значениями которого являются элементы перечисления `XmlNodeType`. С конструкцией можно работать, используя различные свойства, такие как `Name` (возвращает имя элемента или атрибута), `Value` (возвращает данные элемента) и так далее. Набор методов класса `XmlTextReader`, начинающихся с префикса *`Move`* (`MoveToNextElement()`), может использоваться для перехода к следующей XML-конструкции в потоке. Вернуться к просмотренным конструкциям нельзя.

Класс `XmlWriter` – это абстрактный класс для создания XML-данных.

Класс `XmlDocument` представляет XML-документ в виде дерева объектов. Каждый объект является экземпляром класса, порождённого от `XmlNode`, который содержит методы и свойства для навигации по дереву, чтения и записи информации элемента и множество других.

Метод `Load()` считывает документ и разбирает его в памяти на элементы. Если документ не является хорошо оформленным, генерируется исключение `XmlException`.

За успешным вызовом метода `Load()` обычно следует обращение к свойству объекта `DocumentElement`. Свойство возвращает ссылку на объект класса `XmlNode`. Этот объект позволяет обнаружить у элемента дочерние элементы (свойство `HasChildNodes`) и получить к ним доступ, используя коллекцию `ChildNodes`. Комбинация свойств `HasChildNodes` и `ChildNodes` делает возможным применение при обработке XML-документов рекурсии.

У объекта `XmlNode` имеются свойства `NodeType`, `Name` и `Value`. Следует иметь в виду, что обращаться к `Name` и `Value` нужно в зависимости от значения `NodeType`. У элементов (`XmlNodeType.Element`) нет `Value`, но определено `Name`. У текста (`XmlNodeType.Text`) определено `Name`, но нет `Value`. У атрибутов определены и `Name`, и `Value`.

Для того чтобы получить определенный узел или набор узлов, нет необходимости итеративно просматривать весь XML-документ. Для этих целей можно использовать такие методы класса `XmlDocument`, как `GetElementsByTagName()`, `SelectNodes()`, `SelectSingleNode()`.

При помощи класса `XmlDocument` можно не только читать, но и создавать XML-документы. В следующем примере загружается файл `Students.xml`, удаляется его первый элемент, добавляется еще один элемент, описывающий студента, и результат сохраняется в файл `Students_new.xml`.

```
var doc = new XmlDocument();
doc.Load("Students.xml");
XmlNode root = doc.DocumentElement;
XmlElement student = doc.CreateElement("student");
XmlElement name = doc.CreateElement("name");
student.AppendChild(name);
doc.Save("Students_new.xml");
```

Другие методы класса `XmlDocument`, которые могут быть полезны при модификации XML-документа, это методы `PrependChild()`, `InsertBefore()`, `InsertAfter()`, `RemoveAll()`, `ReplaceChild()`.