

Запомните

XSD схема — это документ, который определяет структуру и правила валидации для XML документов. Каждый XSD документ начинается с корневого элемента `<xs:schema>`, который объявляет пространство имен XSD с помощью атрибута `xmlns:xs="http://www.w3.org/2001/XMLSchema"`. Это означает, что вы используете стандартные определения и типы данных(стандартные теги), которые определены в спецификации XML Schema от W3C.

Нужно ли писать `<xs:schema>` в XSD схеме?

Да, это обязательно. Элемент `<xs:schema>` является корневым элементом для любой XSD схемы и задает контекст, в котором будут определены все элементы и типы данных схемы. Без него схема не будет считаться валидной, и парсеры XML не смогут корректно интерпретировать содержимое схемы.

Какую схему в `xs:schema` нужно указывать обязательно?

Вы должны указать URL пространства имен XSD, которое является стандартным и должно быть `http://www.w3.org/2001/XMLSchema` для элемента `<xs:schema>`. Это гарантирует, что вы используете стандартные определения типов и элементов, признанные и поддерживаемые всеми XML парсерами.

Можно ли писать несколько `<xs:schema>`?

В одном XSD документе может быть только один корневой элемент `<xs:schema>`.

Однако, вы можете иметь несколько XSD файлов, каждый с собственным корневым элементом `<xs:schema>`, и эти файлы могут быть связаны через импорты и включения (`import` и `include`).

Импорт (`import`) схемы (когда нужно включить определения из другой XSD схемы с другим пространством имен)

Когда вы используете `import` в XSD, вы обычно ссылаетесь на внешнюю схему, которая может быть определена в другом пространстве имен. При этом указывается `namespace`, который является идентификатором пространства имен внешней схемы (это не должен быть реальный URL адрес к которому должен быть интернет-доступ! это, как и стандартный <http://www.w3.org/2001/XMLSchema> - просто текст по которому парсер XSD определяет пространства имён).

Также указывается `schemaLocation`, который является путём к самому XSD файлу. `schemaLocation` может быть как URL, так и путём к файлу в файловой системе:

- Если указана ссылка на URL, то парсер попытается загрузить XSD с этого URL, что потребует доступа к интернету.
- Однако, чаще всего `schemaLocation` указывает на локальный файл, и парсер просто проверяет наличие файла в указанном месте файловой системы. Для работы с локальным файлом не нужно подключение к интернету.

```
<xs:import namespace="http://www.example.com/schema" schemaLocation="externalSchema.xsd"/>
```

В этом случае `externalSchema.xsd` должен быть доступен локально, в том же каталоге, что и основная XSD схема, или по пути, указанному в `schemaLocation`.

Включение (`include`) схемы (чтобы добавить определения из другой схемы XSD без изменения пространства имен) `include` используется для расширения текущей схемы определениями из другой схемы, которая находится в том же пространстве имен. `schemaLocation` в директиве `include` всегда указывает на локальный файл:

```
<xs:include schemaLocation="additionalTypes.xsd"/>
```

Здесь `additionalTypes.xsd` должен быть доступен в локальной файловой системе, и парсер будет искать его в том же каталоге, где находится основная XSD схема, либо по пути, указанному в `schemaLocation`.

Префикс **xs:** в схемах XSD случае используется для указания пространства имен (`namespace`) "<http://www.w3.org/2001/XMLSchema>", которое определяет стандартные типы данных XML Schema. При использовании элементов, типов данных и других конструкций из схемы XML, необходимо указывать префикс `xs:` для того, чтобы указать, что эти элементы относятся к стандартным типам данных XML Schema. Это позволяет избежать конфликтов с другими пространствами имен(при использовании своих придуманных типов данных) и упрощает понимание структуры документа.

Можно ли писать XSD без `xs`?

Нет, без указания пространства имен программа, которая разбирает XML не сможет правильно понять элементы и типы данных, используемые в схеме XSD. Поэтому каждый элемент, тип данных или атрибут в XML-схеме должен быть определен с помощью префикса, указывающего пространство имен, к которому он относится. И для обозначения пространства имен XSD используется префикс `xs`.

Можно ли переименовать xs?

Нет, используется именно префикс `xs`, который обозначает пространство имен XML Schema (XSD). Это означает, что элементы и атрибуты, определенные в схеме, будут относиться к пространству имен XML Schema, что позволяет использовать различные схемы в одном документе.

Конечно, в принципе, вы можете попробовать использовать другой префикс вместо `xs`, если он был объявлен в XML документе как префикс для XSD пространства имен, но это редко используется и не рекомендуется, так как это может сбивать с толку других разработчиков, которые будут работать с вашим кодом.

А какие префиксы можно сделать самому, для своих типов данных?

Можно использовать свои префиксы для нестандартных типов данных, но для этого нужно объявить соответствующий пространство имен (namespace) и связать его с префиксом. Например, для объявления типов данных из пространства имен "<http://example.com/mytypes>" и связывания их с префиксом "my", можно использовать следующую инструкцию:

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:my="http://example.com/mytypes">
  <!-- Определения типов данных из пространства имен "http://example.com/mytypes" -->
  <xs:complexType name="MyType">
    <xs:sequence>
      <xs:element name="myfield" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <!-- Использование типа данных "MyType" с префиксом "my" -->
  <xs:element name="myelement" type="my:MyType"/>
</xs:schema>
```

Здесь элементы и типы данных, определенные в пространстве имен "<http://example.com/mytypes>", используются с префиксом "my".

Задание

Найдите ошибки в XSD, исправьте их (помните, что XSD схема пишется на языке разметки XML, поэтому можете использовать как подсказку валидатор, такой же как для XML сообщений - <https://codebeautify.org/xmlvalidator>)

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
<xs:element name="book">
```

```
<xs:complexType>
```

```
<xs:sequence>
```

```
<xs:element name="title" type="xs:string"/>
```

```
<xs:name="author" type="xs:string"/>
```

```
<xs:element name="price" type="xs:decimal">
```

```
</xs:sequence>
```

```
</xs:complexType>
```

```
</xs:element>
```

```
</xs:schema>
```