Тип «UEObject» в Mobile SMARTS

Последние изменения: 2024-03-26

UEObject – тип, предназначенный для обработки данных, сохраненных в формате XML или JSON. Доступен в клиентском и серверном контексте выполнения. С помощью данного класса можно разбирать полученные данные, изменять структуру данных и собирать обратно в строку. Сериализация/ десериализация проводится без использования схемы/класса с метадатой.

Доступные поля:

Имя поля
Описание
Childs
Коллекция узлов и атрибутов объекта (UEObject)
HasValue
Есть ли значение
IsArray
Это массив
IsAttribute
Это атрибут
OName
Имя узла
OValue
Значение узла. Если строка загружена извне, тип по умолчанию — строка.
Type
Тип

Доступные методы:

Имя метода
Описание
UEObject()
Создаёт новый пустой UEObject (в случае XML создаёт атрибут <root>)
UEObject(string)
Создаёт новый UEObject с узлом внутри. Атрибут – название узла (для JSON нужно присвоить узлу значение)
FromXml(string)
Получить XML строку

FromJson(string, boolean)
Получить JSON строку. В качестве аргумента также передаётся наличие префикса «@» у атрибутов

GetAttribute(string)
Получить атрибут
GetNode(string)
Получить узел
GetNodes(string)
Получить несколько узлов
AddNode(string, object)
Добавить узел (строку). Аргументы – название узла, его значение (для JSON нужно добавлять значение, либо другой узел)
AddNode(UEObject)
Добавить узел (UEObject)
SetAttribute(string, object)
Добавить/заменить атрибут. Аргументы – название атрибута, его значение
Contains(string)
Содержит ли объект строку
AsBoolean()
Сохранить значение в булевой переменной
AsDateTime()
Сохранить значение в виде даты
AsDecimal()
Сохранить значение в виде десятичной дроби
AsDouble()
Сохранить значение в виде числа с плавающей запятой
AsInt32()
Сохранить значение в виде целого числа
AsString()
Сохранить значение в виде строки
ToXml(bool)
Сохранить объект в виде XML Аргумент – добавлять ли XML декларацию
ToJson(bool)
Сохранить объект в виде JSON Аргумент – добавлять ли префикс «@» к атрибутам

Примеры использования

Создание XML строки

К примеру, если мы хотим создать пустую XML строку, то это можно сделать так:

```
Объект = new Cleverence.Connectivity.UEObject();
   Строка = Объект.ToXml(true);
```

В этом коде мы создаём объект UEObject, конвертируем его в XML строку с помощью метода ToXml (с добавлением декларации), и заносим строку в переменную «Строка».

Если же нужно создать XML строку с начальным узлом, то нужно написать следующее:

```
Объект = new Cleverence.Connectivity.UEObject(«НовыйУзел»);
   Строка = Объект.ToXml(true);
```

В данном случае будет создан узел с именем «НовыйУзел» и пустым значением.

Результаты:

Без начального узла

С начальным узлом

Добавление узлов

К примеру, нам необходимо создать подобную XML строку:

```
<?xml version="1.0" encoding="UTF-8"?>
   <ObjectKey1><ObjectKey2>ObjectValue2</ObjectKey2></ObjectKey1>
```

Это можно сделать так:

```
Объект = new Cleverence.Connectivity.UEObject();
   Объект.AddNode("ObjectKey1", null).AddNode("ObjectKey2", "ObjectValue2");
   Строка = Объект.ToXml(true);
```

В объекте создаётся узел «ObjectKey1», который содержит в себе узел «ObjectKey2» со значением «ObjectValue2».

Результат:

Добавление атрибута

Атрибуты добавляются практически также, как узлы. Если в узел нужно добавить атрибут, то необходимо обратиться к узлу и написать метод «SetAttribute». К примеру:

```
Объект = new Cleverence.Connectivity.UEObject();
   Объект.AddNode("ObjectKey1", null).SetAttribute("ObjectAttribute", 15);
   Строка = Объект.ToXml(true);
```

В данном случае мы добавляем в узел «ObjectKey1» атрибут «ObjectAttribute» со значением 15:

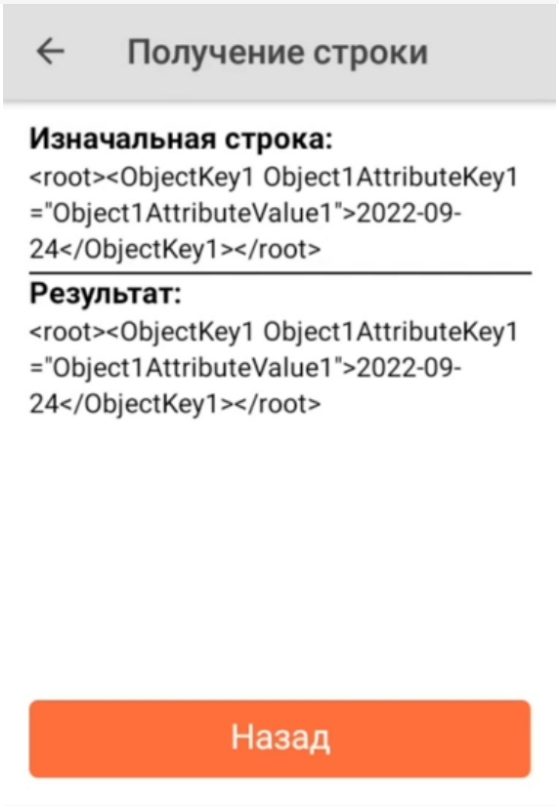
Получение строки XML

UEObject позволяет не только создавать свои строки, но и также получать их. К примеру, если у нас есть таблица, где хранятся XML строки, то можно вытащить оттуда строку и начать с ней работать. Сделать это можно таким образом:

```
Объект = new Cleverence.Connectivity.UEObject();
   Строка = select first (*) from XML where Item.Ид == "1";
   Объект.FromXml(Строка.Значение);
   Результат = Объект.ToXml(false);
```

Здесь мы вытаскиваем из таблицы XML строку с «Ид» равной 1 и записываем её в объект. В конце конвертируем объект в XML строку и записываем в переменную.

Результат:



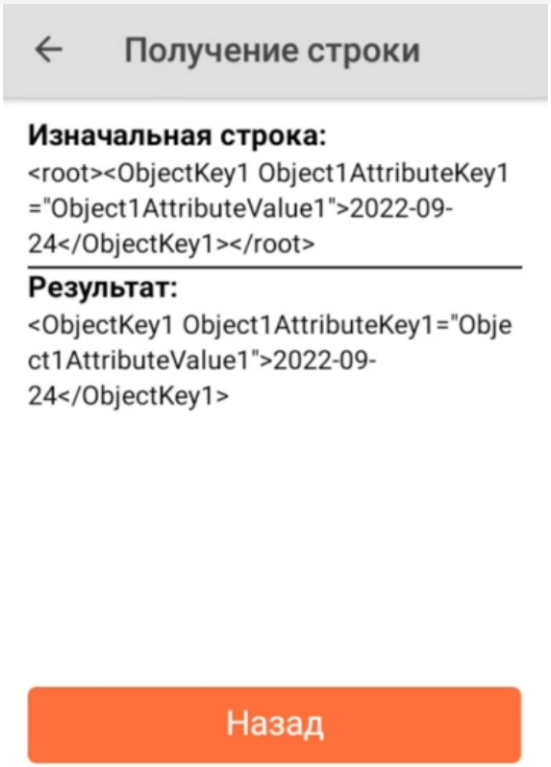
Результат работы кода приведён в нижней части.

Получение узла

Чтобы получить узел с его содержимым нужно написать следующий код:

```
Объект = new Cleverence.Connectivity.UEObject();
   Строка = select first (*) from XML where Item.Ид == "1";
   Объект.FromXml(Строка.Значение);
   Результат = Объект.GetNode("ObjectKey1").ToXml(false);
```

В метод GetNode нужно добавить название узла, который необходимо получить. Нам нужен узел «ObjectKey1». Если в узле, где ведётся поиск (в нашем случае «Объект») не будет найден узел с таким именем, то метод вернёт null.



В верхней части выводится вся строка

Получение полей узла

Если нужно получить какое-либо поле узла, то это можно сделать следующими способами:

```
Результат = Объект.GetNode("ObjectKey1").OValue;
```

```
Результат = Объект.ObjectKey1.AsDateTime();
```

```
Результат = Объект.GetNode("ObjectKey1").OName;
```

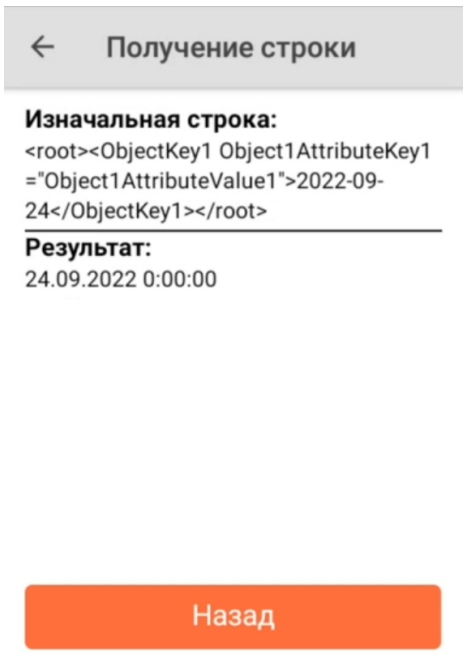
В первых двух случаях мы получаем значение узла. Отличие в том, что в первом случае мы получаем сырое значение узла (строку), а во втором в формате даты. И есть разница между тем, как идёт обращение к узлам: через метод GetNode или напрямую. В последнем же случае мы получаем имя узла.

Сырое значение

В формате даты

Имя узла


Получение строки
Изначальная строка:
<root><ObjectKey1 Object1AttributeKey1
="Object1AttributeValue1">2022-09-
24</ObjectKey1></root>
Результат:
2022-09-24
Назад


Получение строки
Изначальная строка:
<root><ObjectKey1 Object1AttributeKey1
="Object1AttributeValue1">2022-09-
24</ObjectKey1></root>
Результат:
24.09.2022 0:00:00
Назад


Получение строки
Изначальная строка:
<root><ObjectKey1 Object1AttributeKey1
="Object1AttributeValue1">2022-09-
24</ObjectKey1></root>
Результат:
ObjectKey1
Назад

Получение атрибута

Получить атрибут с его содержимым можно с помощью метода GetAttribute. Пример:

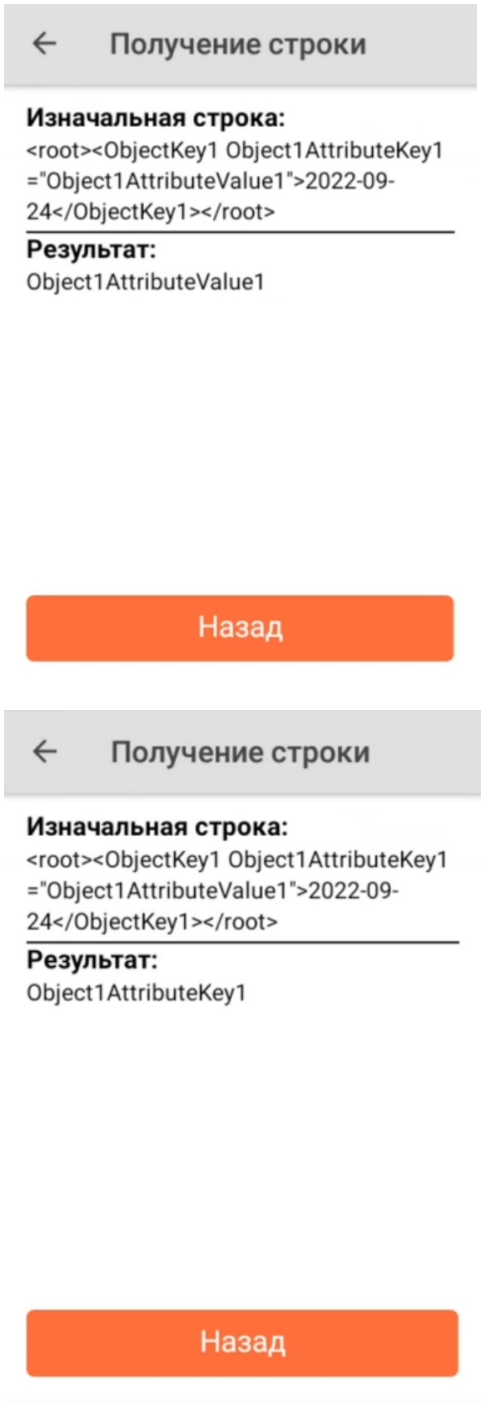
```
Результат = Объект.GetNode(«ObjectKey1».GetAttribute(«Object1AttributeKey1»).OValue;
```

```
Результат = Объект.GetNode(«ObjectKey1».GetAttribute(«Object1AttributeKey1»).OName;
```

В первом случае получаем значение атрибута, во втором его имя.

Значение

Имя


Получение строки
Изначальная строка:
<root><ObjectKey1 Object1AttributeKey1
="Object1AttributeValue1">2022-09-
24</ObjectKey1></root>
Результат:
Object1AttributeValue1
Назад

Получение строки
Изначальная строка:
<root><ObjectKey1 Object1AttributeKey1
="Object1AttributeValue1">2022-09-
24</ObjectKey1></root>
Результат:
Object1AttributeKey1
Назад

Использование select

UEObject позволяет использовать select. Возьмём в качестве примера предыдущий случай. Можно получить значение атрибута с помощью оператора select:

```
Значение = select (*) from Объект.ObjectKey1.Childs where Item.OName ==
"Object1AttributeKey1";
   Результат = Значение[0].OValue;
```

В данном случае мы обращаемся к коллекции Childs у узла ObjectKey1 и ищем в ней строку, у которой имя является именем атрибута «ObjectAttributeKey1». В полученной коллекции мы обращаемся к элементу с индексом 0 и получаем значение нашего атрибута.

Работа с JSON

Логика работы с JSON строками идентична с XML, но есть несколько важных моментов:

1. **При создании пустого объекта не добавляется узел «root»**
2. **Нельзя создать пустой объект UEObject с начальным узлом**
3. **В методах FromJson и ToJson есть булева переменная, отвечающая за наличие префикса «@» у атрибутов. Нужно для обозначения атрибутов в случае конвертации в/из XML.**
4. **Если у узла есть и значение и атрибут, то значение записывается в «#text»**

Примеры:

```
Объект = new Cleverence.Connectivity.UEObject();
   Объект.AddNode("ObjectKey1", null).SetAttribute("ObjectAttribute", 15);
```

Создаём объект с узлом «ObjectKey1» и атрибутом «ObjectAttribute» со значением 15.

ToJson(true)

ToJson(false)


```
Объект = new UEObject();
   Объект.AddNode("ObjectKey1", "ObjectValue1").SetAttribute("ObjectAttribute", "Attribute");
   Строка = Объект.ToJson(true);
```

Создаём объект с узлом «ObjectKey1» у которого есть и значение и атрибут. Добавляем префикс для атрибута.

Взаимные конвертации

XML и JSON строки можно конвертировать между собой. Самое главное – при конвертации из JSON в XML задавать атрибутам префиксы.

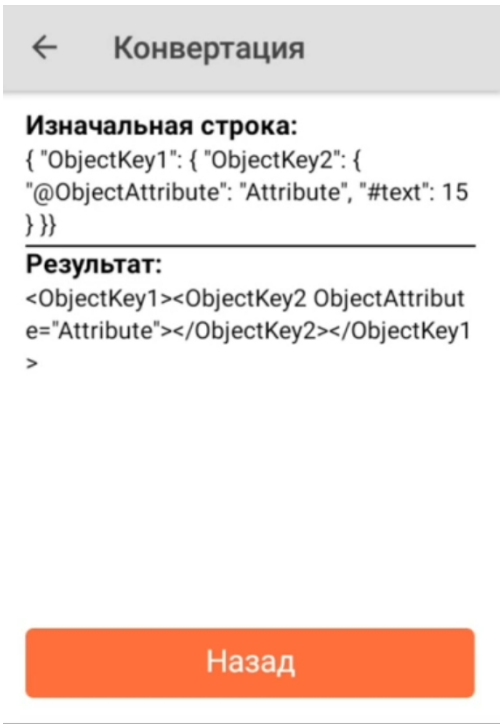
```
Объект = new UEObject();
   Объект.AddNode("ObjectKey1", "ObjectValue1").AddNode("ObjectKey2",
15).SetAttribute("ObjectAttribute", "Attribute");
   Строка = Объект.ToXml(false);
   Объект2 = new Cleverence.Connectivity.UEObject();
   Объект2.FromXml(Строка);
   Строка2 = Объект2.ToJson(true);
```

Здесь мы создаём объект с 2-я узлами и атрибутом, в переменную «Строка» записываем XML строку, создаём новый пустой объект и помещаем туда нашу XML строку. В конце, конвертируем 2-ой объект в JSON строку и записываем в «Строка2».

Результаты:

Аналогично поступаем в случае конвертации из JSON в XML (не забывая обозначать атрибуты префиксами).

```
Объект = new Cleverence.Connectivity.UEObject();
   Объект.AddNode("ObjectKey1", "ObjectValue1").AddNode("ObjectKey2",
15).SetAttribute("ObjectAttribute", "Attribute");
   Строка = Объект.ToJson(true);
   Объект2 = new Cleverence.Connectivity.UEObject();
   Объект2.FromJson(Строка, true);
   Строка2 = Объект2.ToXml(false);
```



Не нашли что искали?

Задать вопрос в техническую поддержку