1С ПАБЛИШИНГ

Е.Ю. Хрусталева

Технологии интеграции «1С:Предприятия 8.3»

Издание 2,
стереотипное

Требуется доступ к Интернету

ПРОФЕССИОНАЛЬНАЯ РАЗРАБОТКА

Е.Ю. Хрусталева

Технологии интеграции «1С:Предприятия 8.3», 2-е стереотипное издание

Электронная книга в формате pdf; ISBN 978-5-9677-3308-6.

Электронный аналог печатного издания «Технологии интеграции "1С:Предприятия 8.3". Издание 2, стереотипное» (ISBN 978-5-9677-3303-1, М.: ООО «1С-Паблишинг», 2023;
артикул печатной книги по прайс-листу фирмы «1С»: 4601546147462);
по вопросам приобретения печатных изданий издательства «1С-Паблишинг» обращайтесь к партнеру «1С», обслуживающему вашу организацию, или к другим партнерам фирмы «1С», в магазины «1С Интерес», а также в книжные и интернет-магазины.

Книга адресована специалистам, имеющим опыт разработки на платформе «1С:Предприятие». С ее помощью можно освоить механизмы «1С:Предприятия 8.3», предназначенные для обмена данными в распределенных системах, а также механизмы и технологии, позволяющие интегрировать прикладные решения с другими информационными системами, не использующими «1С:Предприятие».
Книга включает описание интернет-технологий, которые появились в платформе 8.3 или не были описаны в издании по предыдущей версии платформы:
• JSON;
• HTTP-сервисы (REST);
• HTTP-запросы;
• автоматический REST-интерфейс (OData);
• Web-сервисы;
• FTP-соединение;
• электронная почта.

Для создания демонстрационных примеров использована версия 8.3.22.1709 платформы «1С:Предприятие 8». Учебную версию платформы «1С:Предприятие 8.3» можно бесплатно скачать по адресу https://online.1c.ru/catalog/free/learning.php#platform. Информационные базы с примерами, описанными в книге, опубликованы на портале 1С:ИТС. Вы можете скачать их по адресу https://its.1c.ru/bmk/integr83.

Данное 2-е издание является стереотипным, не отличается по содержанию от книги с аналогичным названием, выпущенной в 2020 году издательством «1С-Паблишинг» (ISBN 978-5-9677-2962-1, артикул «1С» 4601546143242).

Книга выпущена под редакцией Максима Радченко.

Оглавление

Введение

В книге собрана и систематизирована наиболее важная информация, которая может понадобиться разработчику прикладных решений «1С:Предприятия 8.3».

Несмотря на то что в одной книге невозможно рассмотреть все ситуации, возникающие при разработке прикладных решений, на большинство вопросов в книге можно найти ответы. Причем читать будет одинаково интересно как начинающим, так и продвинутым разработчикам.

При написании мы стремились к тому, чтобы книга стала серьезным инструментом для разработчиков: к ней всегда можно было бы обратиться в случае затруднений, узнать что-то новое о хорошо известной предметной области или познакомиться с новым взглядом на привычные вещи.

При подготовке материала были использованы различные источники информации:

- опыт преподавания на учебных курсах по платформе и прикладным решениям «1С:Предприятия 8»;
- опыт внедрения прикладных решений;
- опыт, накопленный разработчиками фирмы «1С»;

- материалы информационно-технологической поддержки (ИТС);
- материалы форума партнеров-разработчиков на сайте http://partners.v8.1c.ru;
- общение на партнерских семинарах, проводимых фирмой «1С».

Глава 1. Интернет-технологии

JSON

В настоящее время в веб-приложениях широко используется формат обмена данными JSON. JSON (JavaScript Object Notation) – это текстовый формат обмена данными, с которым могут работать все браузеры. Этот формат похож на XML, но по сравнению с XML он является более лаконичным и требует меньше места.

Общая информация

Есть несколько причин для широкого использования этого формата на уровне платформы. Во-первых, JSON – это современный формат, с помощью которого прикладные решения «1С:Предприятия» могут осуществлять интеграцию со сторонними приложениями. Во-вторых, JSON активно используется в HTTP-интерфейсах, а платформа «1С:Предприятие 8» как раз предоставляет два способа реализации таких интерфейсов – это REST-интерфейс, который автоматически формируется для всего прикладного решения, и HTTP-сервисы, которые можно создавать самостоятельно.

Существует несколько основных сценариев использования JSON:

- Интеграция с внешними системами через их HTTP-интерфейсы: Google Calendar, Salesforce.com, REST-интерфейс «1С:Предприятия», SharePoint и т. д.
- Организация собственного HTTP-интерфейса прикладного решения.
- Обмен файлами JSON с внешними системами. Формирование конфигурационных, настроечных файлов. Использование их в процедурах обмена данными, например с интернет-магазинами.
- Использование файлов JSON для обмена данными между разными приложениями «1С:Предприятия».

JSON – это текстовый формат, поэтому данные в формате JSON могут содержать:

- *Объект* – неупорядоченное множество пар <имя свойства>:<значение>, заключенный в фигурные скобки ({}). Пары <имя свойства>:<значение> разделяются запятыми (,).
- *Массив* – множество значений. Массив заключается в квадратные скобки ([]). Значения разделяются запятыми (,).
- *Значение* – может быть строкой, числом, объектом, массивом или литералом `true`, `false`, `null`.
  - `Строка` – набор символов, заключенный в двойные кавычки («»).
  - `Число` – сериализуется с разделителем точка (.). Точность числа не ограничена.

Таким образом, с помощью вышеперечисленных элементов допускается описание объектов любой сложности для представления в формате JSON.

В платформе реализовано несколько слоев работы с JSON. Самые универсальные и гибкие – это низкоуровневые средства потоковой записи и чтения. Более высокоуровневые и не такие универсальные – средства сериализации в JSON примитивных типов и коллекций «1С:Предприятия». И, наконец, третий слой – это средства, позволяющие сериализовать/десериализовать прикладные типы «1С:Предприятия»: ссылки, объекты, наборы записей и вообще любые типы, для которых поддерживается XDTO-сериализация (про XDTO-сериализацию рассказывается в разделе «XDTO-сериализация»).

ПОДРОБНЕЕ

Более подробно про работу с форматом данных JSON рассказано в главе 1 «Интернет-технологии».

Подробнее познакомиться с примерами сериализации/десериализации данных в формате JSON можно в демонстрационной конфигурации «Примеры работы», которая прилагается к книге.

Потоковая работа

Для потоковой работы предназначены объекты `ЧтениеJSON` и `ЗаписьJSON`. Они последовательно, от значения к значению, читают JSON из файла или строки или последовательно записывают JSON в файл или строку. Таким образом, чтение и запись JSON происходят без формирования всего документа в памяти. В результате потоковая техника может дать существенный выигрыш, особенно если объем документа потенциально большой или же если этот объем заранее неизвестен.

Запись и чтение

Для того чтобы выполнить потоковую запись JSON-документа, необходимы записываемые данные и объект `ЗаписьJSON`.

Допустим, нам нужно записать в JSON-документ некоторый объект, обладающий свойствами Код, Наименование, Телефоны (массив строк), ОбъемПродаж, Поставщик?.

Один JSON-файл содержит одно значение. Это то самое значение, о котором говорилось выше: объект, массив, строка, число или один из литералов. Поэтому записываем в JSON наш объект следующим образом (листинг 1.1).

Листинг 1.1. Пример потоковой записи JSON-документа

```
&НаСервереБезКонтекста
Процедура ПотоковаяЗаписьНаСервере()

    // Создать объект записи и открыть файл, в который будет выполняться запись.
    Запись = Новый ЗаписьJSON;
    Запись.ОткрытьФайл("c:\temp\streamWrite.json", , , Новый ПараметрыЗаписиJSON(, Символы.Таб));

    // Выполнить запись значений с помощью объекта записи (Запись).
    // Записать начало нашего объекта.
    Запись.ЗаписатьНачалоОбъекта();

    // Заполнить свойство Код типа Строка.
    Запись.ЗаписатьИмяСвойства("Код");
    Запись.ЗаписатьЗначение("000000017");

    // Заполнить свойство Наименование типа Строка.
    Запись.ЗаписатьИмяСвойства("Наименование");
    Запись.ЗаписатьЗначение("ОАО Топаз");

    // Заполнить свойство Телефоны типа Массив.
```

```
        // Поэтому после имени свойства записываем массив, состоящий из значений – строк.
        Запись.ЗаписатьИмяСвойства("Телефоны");
        Запись.ЗаписатьНачалоМассива();
        Запись.ЗаписатьЗначение("8-999-777-55-33");
        Запись.ЗаписатьЗначение("+71112223344");
        Запись.ЗаписатьКонецМассива();

        // Заполнить свойство ОбъемПродаж типа Число.
        Запись.ЗаписатьИмяСвойства("ОбъемПродаж");
        Запись.ЗаписатьЗначение(5000000);

        // Заполнить свойство Поставщик? типа Булево.
        Запись.ЗаписатьИмяСвойства("Поставщик?");
        Запись.ЗаписатьЗначение(Ложь);

        // Записать конец нашего объекта.
        Запись.ЗаписатьКонецОбъекта();

        // Завершить работу с файлом.
        Запись.Закрыть();

КонецПроцедуры
```

Содержимое процедуры понятно из комментариев в тексте листинга. Поэтому не будем еще раз на этом останавливаться. Про параметры записи JSON, переданные четвертым параметром в метод `ОткрытьФайл()`, будет рассказано ниже, в разделе «Параметры записи JSON».

В результате сформированный JSON-документ будет иметь следующий вид (листинг 1.2).

Листинг 1.2. Содержимое JSON-документа

```
{
        "Код": "000000017",
        "Наименование": "ОАО Топаз",
        "Телефоны": [
                "8-999-777-55-33",
                "+71112223344"
        ],
        "ОбъемПродаж": 5000000,
        "Поставщик?": false
}
```

Теперь представим, что мы хотим записать не какой-то абстрактный объект, а объект, обладающий собственным именем – Контрагент1. Возможно, вместе с ним мы захотим записать и другой аналогичный объект с именем Контрагент2. Как поместить это в один JSON-файл?

Поскольку, как мы говорили, JSON-файл может содержать только одно значение, мы запишем в него корневой объект с единственным свойством.

Имя этого свойства – это имя нашего объекта, а значение этого свойства – это совокупность свойств, которыми обладает наш объект (листинг 1.3).

Листинг 1.3. Пример потоковой записи JSON-документа

```
&НаСервереБезКонтекста
Процедура ПотоковаяЗаписьНаСервере()

	// Создать объект записи и открыть файл, в который будет выполняться запись.
	Запись = Новый ЗаписьJSON;
	Запись.ОткрытьФайл("c:\temp\streamWrite_2.json", , , Новый ПараметрыЗаписиJSON(, Символы.Таб));

	// Выполнить запись значений с помощью объекта записи (Запись).
	// Записать начало корневого объекта.
	Запись.ЗаписатьНачалоОбъекта();

	// Записать имя свойства корневого объекта.
	Запись.ЗаписатьИмяСвойства("Контрагент1");

	// Записать начало нашего объекта.
	Запись.ЗаписатьНачалоОбъекта();

	// Заполнить свойство Код типа Строка.
	Запись.ЗаписатьИмяСвойства("Код");
	Запись.ЗаписатьЗначение("000000017");

	// Заполнить свойство Наименование типа Строка.
	Запись.ЗаписатьИмяСвойства("Наименование");
	Запись.ЗаписатьЗначение("ОАО Топаз");

	// Заполнить свойство Телефоны типа Массив.
	// Поэтому после имени свойтва записываем массив, состоящий из значений – строк.
	Запись.ЗаписатьИмяСвойства("Телефоны");
	Запись.ЗаписатьНачалоМассива();
	Запись.ЗаписатьЗначение("8-999-777-55-33");
	Запись.ЗаписатьЗначение("+71112223344");
	Запись.ЗаписатьКонецМассива();

	// Заполнить свойство ОбъемПродаж типа Число.
	Запись.ЗаписатьИмяСвойства("ОбъемПродаж");
	Запись.ЗаписатьЗначение(5000000);

	// Заполнить свойство Поставщик? типа Булево.
	Запись.ЗаписатьИмяСвойства("Поставщик?");
	Запись.ЗаписатьЗначение(Ложь);

	// Записать конец нашего объекта.
	Запись.ЗаписатьКонецОбъекта();

	// Записать конец корневого объекта.
	Запись.ЗаписатьКонецОбъекта();

	// Завершить работу с файлом.
	Запись.Закрыть();

КонецПроцедуры
```

В результате сформированный JSON-документ будет иметь следующий вид (листинг 1.4).

Листинг 1.4. Содержимое JSON-документа

```
{
	"Контрагент1": {
		"Код": "000000017",
		"Наименование": "ОАО Топаз",
		"Телефоны": [
			"8-999-777-55-33",
			"+71112223344"
		],
		"ОбъемПродаж": 5000000,
		"Поставщик?": false
	}
}
```

Если требуется записать в этот же файл второй объект Контрагент2, то нам нужно просто добавить к корневому объекту еще одно свойство (Контрагент2) и его значение (листинг 1.5).

Листинг 1.5. Пример потоковой записи JSON-документа

```
&НаСервереБезКонтекста
Процедура ПотоковаяЗаписьНаСервере()

	// Создать объект записи и открыть файл, в который будет выполняться запись.
	Запись = Новый ЗаписьJSON;
	Запись.ОткрытьФайл("c:\temp\streamWrite_3.json", , , Новый ПараметрыЗаписиJSON(, Символы.Таб));

	// Выполнить запись значений с помощью объекта записи (Запись).
	// Записать начало корневого объекта.
	Запись.ЗаписатьНачалоОбъекта();

	// Записать имя первого свойства корневого объекта.
	Запись.ЗаписатьИмяСвойства("Контрагент1");

	// Записать начало первого объекта.
	Запись.ЗаписатьНачалоОбъекта();

	...

	// Записать конец первого объекта.
	Запись.ЗаписатьКонецОбъекта();

	// Записать имя второго свойства корневого объекта.
	Запись.ЗаписатьИмяСвойства("Контрагент2");

	// Записать начало второго объекта.
	Запись.ЗаписатьНачалоОбъекта();

	...
```

```
    // Записать конец второго объекта.
    Запись.ЗаписатьКонецОбъекта();

    // Записать конец корневого объекта.
    Запись.ЗаписатьКонецОбъекта();

    // Завершить работу с файлом.
    Запись.Закрыть();

КонецПроцедуры
```

В результате сформированный JSON-документ будет иметь следующий вид (листинг 1.6).

Листинг 1.6. Содержимое JSON-документа

```
{
    "Контрагент1": {
        "Код": "000000017",
        "Наименование": "ОАО Топаз",
        "Телефоны": [
            "8-999-777-55-33",
            "+71112223344"
        ],
        "ОбъемПродаж": 5000000,
        "Поставщик?": false
    },
    "Контрагент2": {
        "Код": "000000018",
        "Наименование": "ОАО Алмаз",
        "Телефоны": [
            "8-111-222-33-44",
            "+78886664422"
        ],
        "ОбъемПродаж": 6000000,
        "Поставщик?": true
    }
}
```

Потоковое чтение JSON-документа выполняется с помощью объекта `ЧтениеJSON`. Документ открывается для чтения и поэлементно считывается в цикле. При этом разработчик определяет, что считано, и соответствующим образом обрабатывает считываемые данные.

Простейший вариант чтения JSON-документа может быть выполнен с помощью следующей процедуры (листинг 1.7).

Листинг 1.7. Пример потокового чтения JSON-документа

```
&НаСервереБезКонтекста
Процедура ПотоковоеЧтениеНаСервере()

	Сообщение = Новый СообщениеПользователю;

	// Создать объект чтения и открыть файл, из которого будет выполняться чтение.
	Чтение = Новый ЧтениеJSON;
	Чтение.ОткрытьФайл("c:\temp\streamWrite_2.json");

	// Выполнить чтение поэлементно в цикле.
	Пока Чтение.Прочитать() Цикл

		Если Чтение.ТипТекущегоЗначения = ТипЗначенияJSON.ИмяСвойства Тогда
			Сообщение.Текст = "Имя = " + Чтение.ТекущееЗначение;
			Сообщение.Сообщить();
		КонецЕсли;

		Если Чтение.ТипТекущегоЗначения = ТипЗначенияJSON.Булево Или
			Чтение.ТипТекущегоЗначения = ТипЗначенияJSON.Строка Или
			Чтение.ТипТекущегоЗначения = ТипЗначенияJSON.Число Или
			Чтение.ТипТекущегоЗначения = ТипЗначенияJSON.Комментарий Тогда
			Сообщение.Текст = "Значение = " + Чтение.ТекущееЗначение;
			Сообщение.Сообщить();
		КонецЕсли;
	КонецЦикла;

	// Завершить работу с файлом.
	Чтение.Закрыть();

КонецПроцедуры
```

Результат чтения JSON-документа, содержимое которого показано в листинге 1.4, будет выглядеть следующим образом (рис. 1.1).

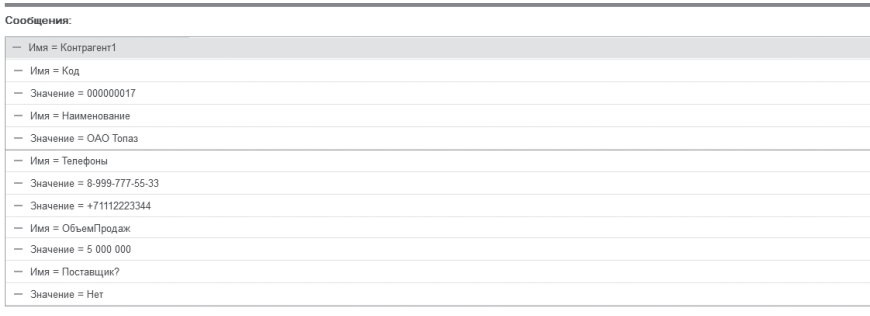

Рис. 1.1. Потоковое чтение JSON-документа

Работа со строкой JSON

Данные в формате JSON можно получать не только в виде файла, но и в виде строки. Например, при работе с HTTP-сервисами очень часто нужно работать с телом HTTP-запроса, которое представляет собой строку JSON. В таких случаях можно писать данные не в файл, а в строку. И потом подставлять эту строку в качестве тела HTTP-запроса. И, наоборот, брать строку, которая является телом HTTP-запроса, и читать из нее данные с помощью объекта `ЧтениеJSON`.

Для этого нужно использовать метод `УстановитьСтроку()` объектов `ЗаписьJSON`/`ЧтениеJSON`. После вызова этого метода для получения строки со сформированными данными JSON достаточно просто завершить запись документа методом `Закрыть()` объекта `ЗаписьJSON` (листинг 1.8).

Листинг 1.8. Запись и чтение содержимого JSON в/из строки

```
&НаСервереБезКонтекста
Процедура РаботаСоСтрокойНаСервере()

	Сообщение = Новый СообщениеПользователю;

	// Создать объект записи и записать строковое значение в строку JSON.
	Запись = Новый ЗаписьJSON;
	Запись.УстановитьСтроку();
	Запись.ЗаписатьЗначение("строковое значение");
	СтрокаJSON = Запись.Закрыть();

	// Показать результат.
	Сообщение.Текст = СтрокаJSON;
	Сообщение.Сообщить();

	// Создать объект чтения и прочитать JSON из строки.
	Чтение = Новый ЧтениеJSON;
	Чтение.УстановитьСтроку(СтрокаJSON);
	Пока Чтение.Прочитать() Цикл

		Если Чтение.ТипТекущегоЗначения = ТипЗначенияJSON.ИмяСвойства Тогда
			Сообщение.Текст = "Имя = " + Чтение.ТекущееЗначение;
			Сообщение.Сообщить();
		КонецЕсли;

		Если Чтение.ТипТекущегоЗначения = ТипЗначенияJSON.Булево Или
			Чтение.ТипТекущегоЗначения = ТипЗначенияJSON.Строка Или
			Чтение.ТипТекущегоЗначения = ТипЗначенияJSON.Число Или
			Чтение.ТипТекущегоЗначения = ТипЗначенияJSON.Комментарий Тогда
			Сообщение.Текст = "Значение = " + Чтение.ТекущееЗначение;
			Сообщение.Сообщить();
		КонецЕсли;
	КонецЦикла;

	// Завершить чтение.
	Чтение.Закрыть();

КонецПроцедуры
```

Проверка структуры записываемого JSON-документа

Следует иметь в виду, что при потоковой записи разработчик самостоятельно формирует структуру JSON-документа. При записи данных объект `ЗаписьJSON` автоматически проверяет правильность записываемой структуры. За это отвечает свойство `ПроверятьСтруктуру`. По умолчанию оно истинно, поэтому если при записи нарушена целостность структуры JSON, то будет сгенерировано исключение и запись выполнена не будет. Но для увеличения скорости работы эту проверку можно отключить (листинг 1.9).

Листинг 1.9. Отключение проверки структуры записываемого JSON-документа

```
ЗаписьJSON.ПроверятьСтруктуру = Ложь;
```

Параметры записи JSON

При записи можно управлять некоторыми параметрами формируемого текста, например: использованием двойных кавычек, переносом строк, символами отступа и экранированием символов. Набор параметров записи задается в объекте `ПараметрыЗаписиJSON` и затем используется при открытии для записи JSON-документа.

Перенос строк

По умолчанию при записи JSON используется автоматический перенос строк. Это удобно для визуального контроля получившегося результата. Но иногда те или иные HTTP-интерфейсы требуют, чтобы JSON был записан в одну строку. В таких случаях можно в параметрах записи JSON отключить перенос строк (листинг 1.10). Кроме того, отключение переноса строк полезно в тех случаях, когда нужно экономить объем передаваемых данных: если не переносить строки, объем уменьшится в среднем на 20 %.

Листинг 1.10. Отключение переноса строк записываемого JSON-документа

```
...
    // Задать параметры записи JSON.
    ПараметрыJSON = Новый ПараметрыЗаписиJSON(ПереносСтрокJSON.Нет, " ", Истина);

    // Создать объект записи и открыть файл, в который будет выполняться запись.
    Запись = Новый ЗаписьJSON;
    Запись.ОткрытьФайл("c:\temp\streamWrite_4.json", , , ПараметрыJSON);`
...
```

В результате сформированный JSON-документ будет иметь следующий вид (листинг 1.11).

Листинг 1.11. Содержимое JSON-документа

```
{"Контрагент1":{"Код":"000000017","Наименование":"ОАО Топаз","Телефоны":
                                ["8-999-777-55-33","+71112223344"],"ОбъемПродаж":5000000,"Поставщик?":false}}
```

Символы отступа

По умолчанию перед строками JSON не вставляется синтаксический отступ. Но для визуального контроля удобно, когда строки форматируются с помощью синтаксического отступа. Поэтому в примерах выше мы использовали для такого отступа символ табуляции (листинг 1.12).

Листинг 1.12. Набор параметров записи JSON-документа с синтаксическим отступом

```
ПараметрыJSON = Новый ПараметрыЗаписиJSON(, Символы.Таб);
Запись = Новый ЗаписьJSON;
Запись.ОткрытьФайл("c:\temp\streamWrite.json", , , ПараметрыJSON);
```

Если бы мы этого не сделали (листинг 1.13), результат выглядел бы следующим образом (листинг 1.14).

Листинг 1.13. Запись JSON-документа без синтаксического отступа

```
...
        Запись.ОткрытьФайл("c:\temp\streamWrite_5.json");
...
```

Листинг 1.14. Содержимое JSON-документа

```
{
"Контрагент1": {
"Код": "000000017",
"Наименование": "ОАО Топаз",
"Телефоны": [
"8-999-777-55-33",
"+71112223344"
],
"ОбъемПродаж": 5000000,
"Поставщик?": false
}
}
```

Экранирование символов

Внешние системы могут использовать разные нотации JSON, могут считать те или иные символы недопустимыми и так далее. Для этих целей у конструктора `ПараметрыЗаписиJSON` и у самого объекта есть возможность управлять экранированием таких символов, как амперсанд, одинарные

кавычки, разделители строк, слеш и угловые скобки. Кроме того, есть возможность для строковых значений использовать одинарные кавычки вместо двойных.

Например, прямой слеш при записи в JSON не экранируется по умолчанию (листинг 1.15).

Листинг 1.15. Запись строки JSON без экранирования символов

```
...
	Сообщение = Новый СообщениеПользователю;

	// Записать строковое значение в JSON.
	Запись = Новый ЗаписьJSON;
	Запись.УстановитьСтроку();
	Запись.ЗаписатьЗначение("путь к файлу C:/файл.txt");
	СтрокаJSON = Запись.Закрыть();

	// Показать результат.
	Сообщение.Текст = СтрокаJSON;

...
```

В результате выполнения приведенного выше фрагмента кода будет получена следующая строка (листинг 1.16).

Листинг 1.16. Результирующая строка без экранирования слеша

```
"путь к файлу C:/файл.txt"
```

Однако эту возможность можно включить (листинг 1.17).

Листинг 1.17. Запись строки JSON с экранированием символов

```
...
	Сообщение = Новый СообщениеПользователю;
	ПараметрыJSON = Новый ПараметрыЗаписиJSON( , , , , , , , ,Истина);

	// Записать строковое значение в JSON.
	Запись = Новый ЗаписьJSON;
	Запись.УстановитьСтроку(ПараметрыJSON);
	Запись.ЗаписатьЗначение("путь к файлу C:/файл.txt");
	СтрокаJSON = Запись.Закрыть();

	// Показать результат.
	Сообщение.Текст = СтрокаJSON;
	Сообщение.Сообщить();
...
```

В результате выполнения приведенного выше фрагмента кода будет получена следующая строка (листинг 1.18).

Листинг 1.18. Результирующая строка с экранированием слеша

```
"путь к файлу C:\/файл.txt"
```

Сериализация коллекций значений (объектная техника)

Во многих случаях (например, при обмене информацией с внешними системами, чтении конфигурационных файлов в формате JSON и др.) проще и удобнее использовать так называемую объектную технику работы с JSON с помощью методов глобального контекста `ПрочитатьJSON()`, `ЗаписатьJSON()` и объектов `ЧтениеJSON`, `ЗаписьJSON`.

Эта техника позволяет избежать рутинной работы по чтению/записи каждого отдельного значения или свойства. При чтении документы JSON отображаются в фиксированный набор типов платформы: `Строка`, `Число`, `Булево`, `Неопределено`, `Массив`, `ФиксированныйМассив`, `Структура`, `ФиксированнаяСтруктура`, `Соответствие`, `Дата`. При записи можно сформировать в памяти и быстро записать структуру в файл JSON.

Таким образом, объектная техника предполагает достаточно простую работу с данными, однако платой за это является расход памяти, так как весь JSON-документ обрабатывается целиком в оперативной памяти. Кроме того, так можно сериализовать только примитивные типы данных и коллекции значений. Прикладные типы данных так сериализовать не получится (о сериализации прикладных типов «1С:Предприятия» рассказывается в разделе «Сериализация прикладных типов "1С:Предприятия"»).

Запись и чтение

Для того чтобы выполнить запись коллекции значений в формате JSON, нужен сам объект низкоуровневой записи `ЗаписьJSON`, собственно, записываемая структура данных. И затем все вышеперечисленные объекты нужно передать в качестве параметров в метод глобального контекста `ЗаписатьJSON()`.

Рассмотрим следующий пример записи в JSON-документ структуры, состоящей из примитивных типов данных (листинг 1.19).

Листинг 1.19. Пример сериализации структуры в JSON-документ

```
&НаСервереБезКонтекста
Процедура СериализацияПростыхТиповНаСервере()

    // Создать структуру с данными контрагента.
    Данные = Новый Структура;
    Данные.Вставить("Контрагент", "ОАО Топаз");
    Данные.Вставить("ОбъемПродаж", 5000000);
```

```
    // Добавить элемент структуры Телефоны типа Массив.
    Телефоны = Новый Массив;
    Телефоны.Добавить("+71112223344");
    Телефоны.Добавить("+79998887766");
    Данные.Вставить("Телефоны", Телефоны);
    Данные.Вставить("Поставщик", Ложь);

    // Создать объект записи и открыть файл, в который будет выполняться запись.
    Запись = Новый ЗаписьJSON;
    ПараметрыЗаписиJSON = Новый ПараметрыЗаписиJSON(ПереносСтрокJSON.Авто, Символы.Таб);
    Запись.ОткрытьФайл("c:\temp\Serialisation.json",,, ПараметрыЗаписиJSON);

    // Выполнить запись данных (Данные) с помощью объекта записи (Запись).
    ЗаписатьJSON(Запись, Данные);

    // Завершить работу с файлом.
    Запись.Закрыть();

КонецПроцедуры
```

В результате сформированный JSON-документ будет иметь следующий вид (листинг 1.20).

Листинг 1.20. Содержимое JSON-документа

```
{
    "Контрагент": "ОАО Топаз",
    "ОбъемПродаж": 5000000,
    "Телефоны": [
        "+71112223344",
        "+79998887766"
    ],
    "Поставщик": false
}
```

Чтение данных в объектной технике выглядит аналогично записи. Рассмотрим пример чтения JSON-документа, содержимое которого показано выше (листинг 1.21).

Листинг 1.21. Пример десериализации структуры из JSON-документа

```
&НаСервереБезКонтекста
Процедура ДесериализацияПростыхТиповНаСервере()

    // Создать объект чтения и открыть файл, из которого будет выполняться чтение.
    Чтение = Новый ЧтениеJSON;
    Чтение.ОткрытьФайл("c:\temp\Serialisation.json");

    // Выполнить чтение данных в структуру Данные с помощью объекта чтения (Чтение).
    Данные = ПрочитатьJSON(Чтение);
```

```
        // Завершить работу с файлом.
        Чтение.Закрыть();

        // Вывести результат чтения в сообщение.
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "Контрагент: " + Данные.Контрагент + ", Объем продаж: " + Данные.ОбъемПродаж +
                          ", Поставщик?: "+ Данные.Поставщик + ", Телефоны: ";

        // Обойти в цикле элемент структуры данных Телефоны типа Массив.
        Для Каждого Телефон Из Данные.Телефоны Цикл
                Сообщение.Текст = Сообщение.Текст + Телефон + ", ";
        КонецЦикла;

        Сообщение.Сообщить();

КонецПроцедуры
```

Результат чтения JSON-документа, содержимое которого показано в листинге 1.20, будет выглядеть следующим образом (рис. 1.2).

Рис. 1.2. Чтение примитивных типов из JSON-документа

Чтение в соответствие

По умолчанию чтение данных методом `ПрочитатьJSON()` выполняется в структуру. Но некоторые внешние системы могут формировать имена свойств, которые, с точки зрения «1С:Предприятия», содержат недопустимые символы. Например, свойство может называться «user.first» или «user.last». С точки зрения платформы это недопустимое значение для строкового ключа (т. к. имя свойства не может содержать точку).

Например, JSON-документ содержит соответствие, имена свойств которого включают пробел и другие недопустимые символы (листинг 1.22).

Листинг 1.22. Содержимое JSON-документа

```
{
        "Контрагент №2": "ОАО Алмаз",
        "Объем Продаж": 6000000,
        "Мобильный тел.": "+79998887766",
        "Поставщик?": true
}
```

При стандартном чтении такого документа в структуру будет получена ошибка (рис. 1.3).

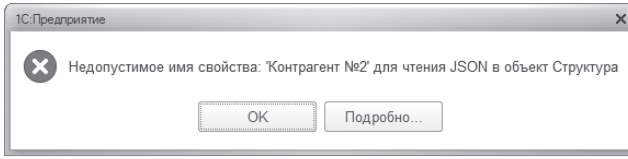

Рис. 1.3. Ошибка при чтении в структуру JSON-документа

В этом случае надо читать данные в соответствие, потому что ключи (имена свойств) соответствия могут быть любыми. Для этого вторым параметром в метод `ПрочитатьJSON(ЧтениеJSON, Истина)` надо передать истину (листинг 1.23).

Листинг 1.23. Пример десериализации соответствия из JSON-документа

```
&НаСервереБезКонтекста
Процедура РаботаССоответствиемНаСервере()

	// Создать объект чтения и открыть файл, из которого будет выполняться чтение.
	Чтение = Новый ЧтениеJSON;
	Чтение.ОткрытьФайл("c:\temp\Serialisation_8.json");

	// Выполнить чтение данных в соответствие Данные с помощью объекта чтения (Чтение).
	Данные = ПрочитатьJSON(Чтение, Истина);
	// Данные = ПрочитатьJSON(Чтение);

	// Завершить работу с файлом.
	Чтение.Закрыть();

	// Вывести результат чтения в сообщение.
	Сообщение = Новый СообщениеПользователю;
	Сообщение.Текст = "Контрагент: " + Данные["Контрагент №2"]
	                  + ", Объем продаж: " + Данные["Объем Продаж"] +
	                  ", Поставщик?: " + Данные["Поставщик?"] + ", Телефон моб.: " + Данные["Мобильный тел."];
	Сообщение.Сообщить();

КонецПроцедуры
```

Результат чтения JSON-документа, содержимое которого показано в листинге 1.22, будет выглядеть следующим образом (рис. 1.4).

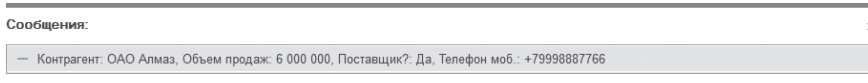

Рис. 1.4. Чтение данных из JSON-документа в соответствие