МЕТОД СРАВНЕНИЯ СТРУКТУР ДАННЫХ, ОПИСАННЫХ С ПОМОЩЬЮ JSONSCHEMAS

METHODS FOR COMPARING DATA STRUCTURES DESCRIBED WITH JSONSCHEMAS



УДК 004.42

Решетова Вера Сергеевна, студент 2 курс, ФГБОУ ВО «Рязанский государственный университет имени С.А. Есенина», Россия, г. Рязань

Reshetova Vera Sergeevna, reshetova_vs@mail.ru

Аннотация

Современные программные продукты в большинстве крупных компаний построены на принципе интеграции распределенных модулей. При организации взаимодействия между ними: передаче данных, хранении и обработке информации, – в последние годы все большую популярность приобретает формат JSON. Предложен метод сравнения JSON-схем, разработан алгоритм анализа структур данных, описанных с помощью JSON-схем. Представлены критерии сравнения элементов, приведен перечень анализируемых атрибутов.

Annotation

Modern software products in most large companies are built on the principle of distributed modules integration. When organizing interaction between them: transferring data, storing and processing information, the JSON format has become increasingly popular in recent years. A method for comparing JSON-schemas is proposed, an algorithm for analyzing data structures described using JSON-schemas is developed. Criteria for comparing elements are presented, a list of analyzed attributes is given.

Ключевые слова: Формат JSON, JSON Schema, валидация, сравнение JSONS chemas, визуализация изменений.

Keywords: JSON format, JSON Schema, validation, JSONS chemas comparison, visualization of changes.

Современные программные продукты в большинстве крупных компаний построены на принципе интеграции распределенных модулей. При организации взаимодействия между ними: передаче данных, хранении и обработке информации, – в последние годы все большую популярность приобретает формат JSON.

В JSON нет правил, согласно которым некоторые объекты должны использовать определенный тип данных или даже содержать одинаковые поля. Им даже не нужно содержать одинаковое количество полей. Кроме того, нет правила, согласно которому данные должны быть в заданном формате. Гибкость – одна из вещей, которая делает JSON простым в использовании, но она же одновременно может вызвать проблемы искажения данных.

Массовое использование JSON для представления данных привело к необходимости вручную проверять содержимое документов. Для валидации был разработан стандарт JSON Schema [1], созданный на основе XSD (XML Schemadefinition). Стандарт удобен для использования при валидации и документировании структур данных, состоящих из чисел, строк, массивов и структур типа ключ-значение (которые, в зависимости от языка программирования, могут называться: объект, словарь, хэш-таблица, ассоциативный массив или карта). С помощью ключевых слов в схеме создаются правила проверки структуры объекта и типов его полей.

В большинстве проектов применение данной спецификации выполняет следующие задачи: упрощение поддержки интеграции ПО, осуществление проверки документов в документо-ориентированных и объектно-ориентированных БД, облегчение поддержки обратной совместимости API(applicationprogramminginterface), генерация DTO (DataTransferObject).

Валидация по JSONSchema представляет собой проверку соответствия экземпляра заявленным ограничениям. Схема является JSON-объектом, свойства которого не являются обязательными, каждое их них представляет собой инструкцию определенного правила проверки. JSONSchema описывает критерии, которым должен удовлетворять экземпляр JSON. Критерии представляют собой набор ключевых слов, описанный в спецификации.

Проверка начинается с применения корневой схемы ко всему документу экземпляра, поэтому различные ключевые слова используются для определения того, какие дополнительные подсхемы применяются к текущему или дочернему элементу. Эти ключевые слова также определяют, будут ли и, если да, то как, результаты проверки подсхемы изменяться и/или комбинироваться. Большинство ключевых слов рассматривается независимо, не оказывая влияния на результаты друг друга.

Каждое утверждение, определяемое ключевым словом, добавляет ограничения, которым должен удовлетворять экземпляр для успешной проверки. Отсутствующие утверждения ни в каком случае не ограничивают проверку. На этом основан процесс валидации.

Прежде всего, схема может ограничивать тип данных. В зависимости от типа проверяемых данных, применяются специализированные правила. Помимо специфичных для типа правил, есть дополнительные обобщенные правила, такие как required и format, а так же описательные правила, такие как id, title, description, $schema. Спецификация определяет несколько микроформатов, таких как: date-time (ISO 8601), date, time, utc-millisec, regex, color (W3C.CR-CSS21-20070719), style (W3C.CR-CSS21-20070719), phone, uri, email, ip-address (V4), ipv6, host-name, которые могут дополнительно проверяться, если определены и поддерживаются конкретной реализацией.

Одной из самых мощных и привлекательных функций JSON Schema является возможность из одной схемы ссылаться на другие, а так же наследовать (расширять) схемы (с помощью ссылок JSON-Ref). Делается это с помощью id, extends и $ref. При расширении схемы нельзя переопределять правила, только

дополнять их. При работе валидатора к проверяемым данным должны применяться все правила из родительской и дочерней схемы.

В больших, часто модифицируемых промышленных системах регулярно требуется внесение правок, актуализация JSON-схем. При этом возникает задача сравнения нескольких схем для определения и анализа различий между ними с целью обнаружения ошибок. В случае работы со сложными структурами и большими объемами данных проводить такую операцию вручную представляется практически невозможным. Вследствие этого в крупных проектах при обработке и хранении данных в формате JSONсуществует необходимость применения средства, которое позволит автоматизировать процесс сравнения JSONSchema.

Для решения задачи сравнения JSON-схем предлагается алгоритм, основанный на обходе с помощью поиска в глубину структуры данных, описанной схемой. Процесс сравнения элементов представляет собой итеративное исследование структур данных, описанных двумя схемами. При сравнении для каждого из элементов первой схемы ищется соответствующий ему элемент второй схемы. Элементы, которые присутствуют в первой схеме, но отсутствуют во второй считаются добавленными. Элементы, которые присутствуют во второй схеме, но отсутствуют в первой считаются удаленными. Блок-схема алгоритма сравнения JSON-схем представлена на рисунке 1.



Рисунок 1. Алгоритм сравнения JSON-схем

Важной частью алгоритма сравнения JSON-схем является процедура сравнения дочерних элементов.

Каждый элемент может быть отнесен к одной из следующих групп (для определения группы используется служебный атрибут type):

- Элементы, описывающие объектные типы. Группа включает в себя тип object.
- Элементы, описывающие массив элементов. Группа включает в себя тип array.
- Элементы-контейнеры, позволяющие описать специфичные правила валидации JSON. Группа включает в себя конструкции anyOf, oneOf, if-then-elseи т.д.
- Элементы примитивных типов. Группа включает в себя типы boolean, integer, null, number, string.

Элементы, отличные от примитивных, принято называть композитными. Композитные элементы могут иметь дочерние элементы, относящиеся к любой из описанных выше групп.

Для сравнения элементов JSON-схемы требуется определить отношение эквивалентности для всех групп элементов.

1. Два композитных элемента эквивалентны, если эквивалентны все их дочерние элементы. При этом для массивов и контейнеров дочерние элементы могут содержаться внутри служебных конструкций, определенных спецификацией JSONSchema.Например, структура данных, описывающая элементы массива (тип array), находится внутри служебного дочернего элемента items[2].

2. Два примитивных элемента эквивалентны, если значения всех их служебных атрибутов совпадают. Перечень служебных атрибутов для сравнения приведен в таблице 1.

Блок-схема алгоритма сравнения элементов JSON-схем представлена на рисунке 2.

Таблица 1. Служебные атрибуты примитивных типов

Наименование типа	Наименование атрибута	Описание
Все примитивные типы	name	Имя элемента
	description	Описание элемента
	type	Тип элемента. Для примитивных типов один из boolean, integer, null, number, string
boolean	Не содержит дополнительных атрибутов	
integer	multipleOf	Делитель значение элемента. Например, если значение multipleOf=2, то значение элемента 4 является корректным, а 3 – нет [3]
	maximum	Максимальное значение элемента (включительно)
	exclusiveMaximum	Максимальное значение элемента (не включительно)
	minimum	Минимальное значение элемента (включительно)
	exclusiveMinimum	Минимальное значение элемента (не включительно)
null	Не содержит дополнительных атрибутов	

number	Набор атрибутов аналогичен типу integer(integeropисывает целые числа, a number –вещественные)	
string	pattern	Регулярное выражение, которому должно соответствовать значение элемента
	minLength	Минимальная длина значенияэлемента
	maxLength	Максимальная длина значения элемента
	enum	Перечень допустимых значений элемента

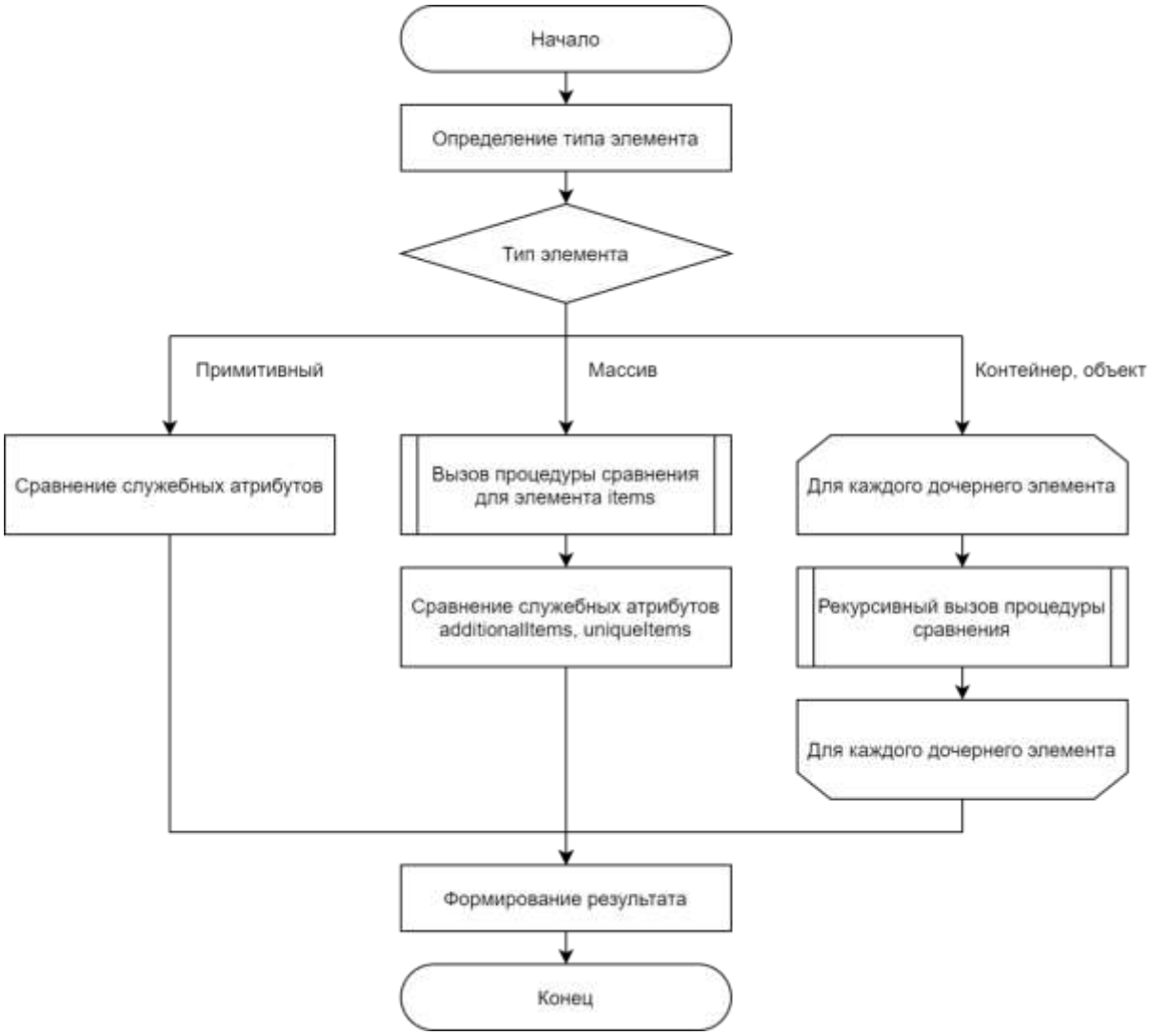


Рисунок 2. Алгоритм процедуры сравнения элемента JSON-схемы

Предложенный способ сравнения схем позволяет анализировать все конструкции спецификации JSONSchema(draft-07). К его достоинствам можно отнести то, что сравниваются не сами JSON-схемы, которые могут состоять из множества файлов, а описываемые с их помощью структуры данных, что облегчает интерпретацию и анализ как самих схем, так и результата сравнения.

К трудностям разработки приложения на основе представленного алгоритма стоит отнести:

- обработку вынесенных описаний типов (definitions);
- реализацию импорта по ссылкам других JSON-документов (атрибут $ref);
- длительное выполнение рекурсивных процедур;
- формирование дерева с результатом сравнения для представления пользователю.

Предполагается разработка программной системы, реализующей предложенный способ сравнения структур данных, описанных с помощью JSONSchema. Результат сравнения может быть предоставлен пользователю в графическом виде для интерактивного анализа или выгружен в виде отчета.

Проблемы с обработкой ссылочных атрибутов возможно решить с помощью использования существующих библиотек для обработки JSON-схем. Время сравнения будет сокращено, если сопоставление элементов одного уровня вложенности выполнять параллельно. При этом следует формировать результат в виде JSON-документа с указанием типа произошедших изменений. На стороне интерфейса полученные данные могут быть представлены в графическом виде, например, с выделением различным цветом измененных, удаленных и добавленных атрибутов.

Целевой аудиторией программной системы являются инженеры, занимающиеся проработкой интеграции автоматизированных систем в крупных технологических компаниях. Применение предложенного способа сравнения позволит ускорить аналитику доработок по интеграционным решениям, снизить риски, связанные с несовместимыми изменениями формата взаимодействия.

Литература

1. Specification | JSONSchema [Электронный ресурс]. –2019. – Режим доступа:https://json-schema.org/specification.html.

2. T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format (RFC 7159) [Электронный ресурс].–Google, 2014.– Режим доступа:https://tools.ietf.org/html/rfc7159.
3. H. Andrews. JSON Schema Validation: A Vocabulary for Structural Validation of JSON [Электронный ресурс].–Cloudflare, 2018. – Режим доступа: https://json-schema.org/draft-07/json-schema-validation.html#RFC7159.

Literature

1. Specification | JSONSchema [Electronic resource]. –2019. - Access mode: https: //json-schema.org/specification.html.
2. T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format (RFC 7159) [Electronic resource] .– Google, 2014.– Access mode: https: //tools.ietf.org/html/rfc7159.
3. H. Andrews. JSON Schema Validation: A Vocabulary for Structural Validation of JSON [Electronic resource]. - Cloudflare, 2018. - Access mode: https://json-schema.org/draft-07/json-schema-validation.html#RFC7159.