CZU: 004.43 DOI: 10.46727/c.steam-2023.p272-278

# ВАЖНОСТЬ ПРИМЕНЕНИЯ ФОРМАТОВ XML И JSON ПРИ СОЗДАНИИ ПРИЛОЖЕНИЙ, РЕАЛИЗОВАННЫХ НА JAVA

**Olga CERBU**, dr. conf. universitar
https://orcid.org/0000-0002-6278-7115
**Tatiana SESTACOVA**, dr. conf universitar
https://orcid.org/0000-0002-6995-4254
**Maryia ROTAR**
Государственный Университет Молдовы, Факультет Информатики
Технический университет Молдовы

**Резюме**. В этой статье рассматриваются универсальные форматы для сериализации, обмена данными и конфигураций в приложениях на Java, такие как `json` и `xml`. Также рассматривается проблематика, из-за которой возникла необходимость применять технологии `json` и `xml`, анализ их различий и особенностей. Форматы стандартизируют обмен данными между разными системами (`xml`) и в веб-приложениях и между клиентом и сервером (`json`).
**Ключевые слова**: xml, json, java, web-приложения.

**Summary**. This article discusses generic formats for serialization, communication, and configurations in Java applications, such as `json` and `xml`. It also discusses the problems that necessitated the use of `json` and `xml` technologies, as well as the analysis of their differences and features. Formats standardize the exchange of data between different systems (`xml`) and in web applications and between client and server (`json`).
**Key words**: xml, json, java, web applications.

## Введение

В настоящий момент для проектирования информационных систем является важным возможность быстро настроить систему и удобно и эффективно обмениваться данными между частями этой системы. То есть, на первое место при разработке ставится простота, быстрота и читаемость. Так, возникла необходимость применять особое структурирование данных, которое позволяло разработчикам максимально быстро разобраться в логике проекта. В этом контексте невозможно не уделить внимание двум мощным инструментам - JSON (JavaScript Object Notation) и XML (Extensible Markup Language). Эти форматы данных стали базой для современных приложений[1,2].

JSON имеет простой и понятный синтаксис, используется для веб-приложений и API как для обмена данными между клиентом и сервером (или микросервисами), так и для до хранения конфигурационных файлов[2].

XML имеет иерархическую структурую, возможность валидации данных, и является более старым способом представления информации. Этот формат нашел применение в областях, включая представление сложных структур данных, создание документов и обмен информацией между системами[3].
В данной работе будут рассмотрены примеры использования форматов в контексте java-приложений, проанализированы преимущества и недостатки, а также их применение при разработке.

**XML**

XML - это формат структурирования данных, аналогичный языку разметки HTML, который расшифровывается как "Расширяемый Язык Разметки" (англ. Extensible Markup Language). Этот формат является стандартом, рекомендованным сообществом W3C для использования в качестве общего инструмента разметки. В отличие от некоторых других языков разметки, XML не предопределен, что означает, что разработчику необходимо самостоятельно определять используемые теги и структуру.

Проблема, которая привела к развитию Xml, сформировалась еще в конце 1990-х годов и начале 2000-х годов. Эта проблема - необходимость в стандарте для обмена данными между различными системами, работающими на разных платформах и с использованием разных языков программирования. В итоге, XML был разработан как универсальный формат, способный описывать структурированные данные с использованием разметки. Он предоставил стандартизированный и расширяемый способ представления данных, что позволило разным системам и приложениям обмениваться информацией и работать вместе.

В данный момент основной задачей XML все еще является обеспечение передачи данных между различными системами, даже если эти системы концептуально различны. Это делает его идеальным для обмена информацией в интернете и между разными приложениями. Множество других языков и стандартов, таких как XHTML, MathML, SVG, XUL, XBL, RSS и RDF, базируются на XML.

**JSON**

JSON - это формат данных, который подобен объектам JavaScript. Это сокращение означает "JavaScript Object Notation" (Запись Объектов JavaScript), и он широко используется для обмена данными в веб-приложениях и API и при этом не зависит от языка, так как базируется на стандартных соглашениях программирования. В отличие от XML, JSON предоставляет более простой и

легковесный синтаксис. В JSON данные представляются в виде пар ключ-значение, что делает их легкими для чтения и понимания как человеком, так и машиной.

Проблемой, связанной с необходимостью развития json, являлась потребность обмениваться данными между клиентской и серверной сторонами веб-приложений. Так, JSON был создан как формат данных, который мог быть легко интерпретирован и создан средствами JavaScript. Он предоставил простой способ представления данных, который был совместим с языком JavaScript и идеально подходил для обмена данными в веб-приложениях. JSON также стал популярным для использования в RESTful API, что способствовало его проникновению в java-разработку для обработки данных, полученных из таких API, и передачи данных обратно.

JSON примечателен тем, что его синтаксис напрямую отражает структуры данных в объектно-ориентированных языках, что делает его естественным выбором для взаимодействия с веб-технологиями. Кроме того, JSON обладает относительно небольшим размером данных, что делает его эффективным в сетевых приложениях.

Итак, какие-то из корневых причин создания XML и JSON включают стандартизацию обмена данными между разными системами (XML) и предоставление удобного формата для обмена данными в веб-приложениях и между клиентом и сервером (JSON). Эти форматы решают разные задачи и имеют разные преимущества в зависимости от контекста и требований приложений.

**Отличия xml и json**

Далее описываются основные отличия технологий xml и json:

JSON (JavaScript Object Notation):

- Простота чтения и записи: JSON имеет чистый и простой синтаксис, что облегчает чтение и запись данных в формате JSON. Это особенно важно при разработке приложений, где необходима человекочитаемость данных.
- Легковесность: JSON формат более компактный и легковесный, чем XML, что делает его эффективным для передачи данных по сети.

XML (Extensible Markup Language):

- Структура и гибкость: XML предоставляет богатую структуру данных и поддерживает сложные иерархии. Это особенно полезно для представления документов, конфигураций и данных с разнообразными свойствами.
- Валидация данных: XML позволяет определять схемы данных (XSD) для валидации данных. Это важно, когда требуется обеспечить строгую согласованность данных.

**Сходства xml и json**

- Оба JSON и XML обладают характеристикой самоописания (способность данных или формата описывать свою структуру и содержание внутри себя). Самоописывающиеся форматы спроектированы таким образом, что их легко читать и записывать как человеку, так и машине. Это позволяет обеспечивать человекопонимаемость и машинную интерпретацию данных.
- JSON и XML обладают встроенной поддержкой определения и проверки структуры данных. Это означает, что можно легко создавать схемы, которые описывают, какие данные допустимы, и проверять, соответствуют ли данные этим схемам.
- JSON и XML поддерживаются множеством языков программирования. Например, JSON активно поддерживается в JavaScript, Python, Perl и Ruby, в то время как XML имеет поддержку в языках, таких как JavaScript, PHP и C#. Это демонстрирует их универсальность и пригодность для разных целей.
- Оба формата также могут использоваться для сериализации данных, то есть преобразования структуры данных в формат, подходящий для хранения или передачи. Это делает их ценными инструментами для обмена информацией между различными приложениями и системами через различные коммуникационные каналы, такие как HTTP или SOAP.
- Оба формата являются текстовыми, что облегчает их использование и чтение. Хотя JSON считается более простой альтернативой XML.
- Форматы имеют иерархическую структуру, в которой каждое поле имеет свое имя и значение, что облегчает понимание сложных структур.

**Поддержка форматов в Java**

Поддержка Json в Java:

В Java существуют библиотеки, такие как Jackson, Gson и org.json, которые упрощают работу с JSON. Эти библиотеки позволяют легко преобразовывать в объекты Java и обратно, преобразовывать в JSON данные.

**Поддержка Xml в Java:**

Для работы с этим форматом в Java существует библиотека Java API for XML Processing (JAXP), которая предоставляет инструменты для чтения и записи XML данных. JAXB (Java Architecture for XML Binding) позволяет маршалировать и демаршалировать XML данные в объекты Java.

**Примеры форматов в приложениях Java**

**Пример применения Json в Java**:

```
import org.json.JSONObject;
public class JSONExample {

    public static void main(String[] args) {

        JSONObject jsonObject = new JSONObject();
        jsonObject.put("name", "John");
        jsonObject.put("age", 30);
        jsonObject.put("city", "New York");
        String jsonString = jsonObject.toString();

        System.out.println(jsonString);

    }
}
```

Здесь рассмотрен простой пример создания json-объекта на java с выводом в консоль.

Результат вывода: `{"name":"John","age":30,"city":"New York"}`

Так как в основном json объекты пересылаются как ответ на запрос с фронтенда в Restful приложениях, то чаще всего применение json-а будет выглядить следующим образом:

```
@RestController
public class ApiController {

    @GetMapping("/api/example")
    public DataObject getData() {
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("name", "John");
        jsonObject.put("age", 30);
        jsonObject.put("city", "New York");
        return jsonObject;
    }
}
```

Таким образом, клиент получит легко преобразуемые данные, которые очень просто обработать и прочитать разработчику.

**Пример применения Xml в Java**:

Что касается применения xml для передачи объектов в java, то он менее удобный и объемный, чем json. Простой пример для данного формата:

```
public class SimpleXMLExample {
    public static void main(String[] args) {
        try {
```

```
        DocumentBuilderFactory docFactory =
    DocumentBuilderFactory.newInstance();
                DocumentBuilder docBuilder =
    docFactory.newDocumentBuilder();
                // Создание коневого элемента:
                Document doc = docBuilder.newDocument();
                Element rootElement = doc.createElement("person");
                doc.appendChild(rootElement);
                // Добавление дочерних элементов
                Element name = doc.createElement("name");
                name.appendChild(doc.createTextNode("John"));
                rootElement.appendChild(name);
                Element age = doc.createElement("age");
                age.appendChild(doc.createTextNode("30"));
                rootElement.appendChild(age);
                Element city = doc.createElement("city");
                city.appendChild(doc.createTextNode("New York"));
                rootElement.appendChild(city);
                String xmlString = docToString(doc);
                System.out.println(xmlString);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

Результат вывода будет представлять собой структуру xml:

```
<person>
    <name>John</name>
    <age>30</age>
    <city>New York</city>
</person>
```

Видно, что создание `xml`-объекта очень долгое. Поэтому, для передачи данных предпочтительнее применять `json`. Однако, структура в виде xml нашла себе место в файлах конфигураций. Например, pom.xml (файл настройки проекта в системе управления проектами Apache Maven, который используется для определения структуры и зависимостей проекта).

Чтение конфигураций xml-файла максимально простое:

```
public static void main(String[] args) {
        try {
            Configurations configs = new Configurations();
            XMLConfiguration config = configs.xml("config.xml");
            String name = config.getString("person.name");
            String age = config.getString("person.age ");
            String city = config.getString("person.city");}
}
```

**Вывод о важности применения xml и json в Java приложениях**

Все современные приложения представляют собой очень сложные системы, поэтому для их понимания и облегчения настройки требуются средства структурирования, такие как `xml` и `json`. В данных момент, они четко разделили роли для применения в java-приложениях: `xml` – в конфигурационных файлах, а `json` – для передачи информации между информационными системами.

**Библиография**


1. HERBERT, S. Java: The Complete Reference. Second edition, 2021.
2. DOUGLAS, C. JSON: Up and Running. Second edition, 2015.
3. SANDOVAL, J. XML and JSON Processing in Java. First edition, 2019.