Professor Hans Noodles

41 уровень

20.05.2019 15622 8

# Форматы сериализации в Java

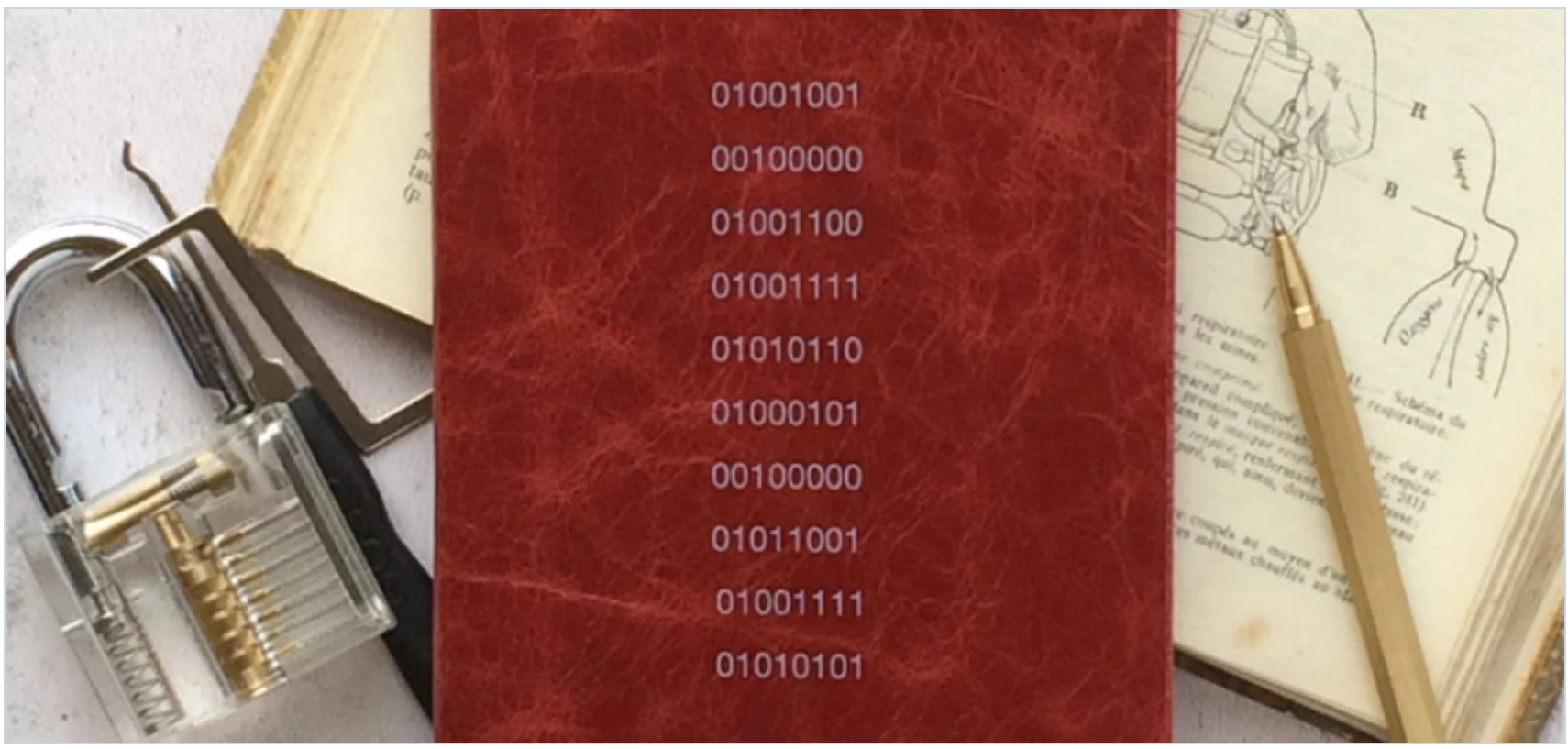
**Статья из группы Java Developer**

42757 участников

Вы в группе

Привет!

Давай поговорим о сериализации. Ты наверняка помнишь, что лекции по сериализации у нас уже были. Так и есть :)




Вот первая

А вот вторая

Если ты уже не очень хорошо помнишь, как работает сериализация, зачем она нужна, и какие в Java есть инструменты для нее, можешь пробежаться по этим лекциям.

Сегодняшняя же лекция будет теоретической, и в ней мы подробнее рассмотрим **форматы сериализации**.

Для начала вспомним, что же такое сериализация.

**Сериализация — это процесс сохранения состояния объекта в последовательность байт.**

**Десериализация — это процесс восстановления объекта из этих байт.**

Java-объект можно сериализовать и передать по сети (например, на другой компьютер)

Так вот, эта самая последовательность байт может быть представлена в разных форматах. Тебе это знакомо из повседневного использования компьютера.

К примеру, электронная книга (или простой текстовый документ), которую ты читаешь, может быть записан в куче разных форматов:

- docx (формат Microsoft Word);
- pdf (формат Adobe);
- mobi (обычно используется в устройствах Amazon Kindle);
- и еще много всего (ePub, djvu, fb2...).

Казалось бы, задача одна и та же: представить текст в человеко-читаемом виде. Но люди изобрели целую россыпь форматов.

Даже не вдаваясь в подробности их работы, мы можем предположить, что сделано это не просто так. Вероятно, у каждого из них есть свои преимущества и недостатки по сравнению с остальными.

Может, и форматы сериализации были созданы по тому же принципу?

Что ж, хорошее предположение, студент! :) Так оно и есть.

Дело в том, что передача данных на расстояние — штука довольно тонкая, и в ней есть много факторов. Кто передает данные? Куда? Какой объем? В качестве принимающей стороны будет человек или машина (т.е. должны ли данные быть human-readable)? Что за устройство будет читать данные?

Очевидно, что ситуации бывают разные. Одно дело, когда нужно передать картинку размером 500Кб с одного смартфона на другой. И совсем другое, когда речь идет о 500 терабайтах бизнес-данных, которые нужно сжать максимально эффективно и при этом передать максимально быстро.

Давай же познакомимся с основными форматами сериализации и рассмотрим преимущества и недостатки каждого из них!

## JSON

JavaScript Object Notation. С ним ты уже немного знаком!

Мы говорили о нем вот в этой лекции, а сериализацию в JSON рассматривали вот тут.

Свое название он получил не просто так. Объекты Java, преобразованные в JSON, действительно выглядят точно так же, как объекты в языке JavaScript.

Тебе вовсе не нужно знать JavaScript, чтобы понять смысл нашего объекта:

```
{
   "title": "Война и мир",
   "author": "Лев Толстой",
   "year": 1869
}
```

Не обязательно передавать один объект. JSON может содержать и массив объектов:

```
[
 {
   "title": "Война и мир",
   "author": "Лев Толстой",
   "year": 1869
```

```
  {
    "title": "Бесы",
    "author": "Федор Достоевский",
    "year": 1872
  },

  {
    "title": "Чайка",
    "author": "Антон Чехов",
    "year": 1896
  }
]
```

Поскольку JSON — объект JavaScript, он поддерживает следующие форматы данных JavaScript:

- строки (string);
- числа (number);
- объекты (object);
- массивы (array);
- boolean-значения (true и false);
- null.

Какие же преимущества есть у JSON?

1. **Human-readable («человеко-читаемый») формат.** Это очевидное преимущество, если твой конечный пользователь — человек. К примеру, на твоем сервере хранится база данных с расписанием авиаперелетов. Клиент-человек запрашивает данные из этой базы с помощью веб-приложения, сидя дома за компьютером. Поскольку тебе нужно предоставить данные в формате, который он сможет понять, JSON будет отличным решением.

2. **Простота.** Можно сказать — элементарность :) Выше мы привели пример двух JSON-файлов. И даже если ты вообще не слышал о существовании JavaScript (и уж тем более о его объектах), ты легко поймешь, что за объекты там описаны. Вся документация JSON — это одна веб-страница с парой картинок.

3. **Широкая распространенность.** JavaScript — доминирующий язык фронтенда, и он диктует свои условия. Использование JSON — необходимость. Поэтому огромное число веб-сервисов используют JSON в качестве формата для обмена данными. Каждая современная IDE поддерживает JSON-формат (в том числе Intellij IDEA). Для работы с JSON написана куча библиотек для всех возможных языков программирования.

Например, ты уже работал с библиотекой Jackson в лекции, где мы учились сериализовывать Java-объекты в JSON.

Но помимо Jackson есть, например, GSON — очень удобная библиотека от Google.

## YAML

В начале своего существования расшифровывался как Yet Another Markup Language — «еще один язык разметки». В то время его позиционировали как конкурента XML.

Сейчас же, по прошествии времени, он расшифровывается как «YAML Ain’t Markup Language» («YAML — не язык разметки»).

Что же он из себя представляет?

Давай представим, что нам нужно создать 3 класса персонажей для нашей компьютерной игры: Воин, Маг и Вор.

У них буду следующие характеристики: сила, ловкость, выносливость, набор оружия.

Вот как будет выглядеть наш YAML-файл с описанием классов:

```yaml
classes:
 class-1:
   title: Warrior
   power: 8
   agility: 4
   stamina: 7
   weapons:
     - sword
     - spear

 class-2:
   title: Mage
   power: 5
   agility: 7
   stamina: 5
   weapons:
     - magic staff

 class-3:
   title: Thief
   power: 6
   agility: 6
   stamina: 5
   weapons:
     - dagger
     - poison
```

YAML-файл имеет древовидную структуру: одни элементы вложены в другие. Вложенностью мы можем управлять при помощи некоторого количества пробелов, которым обозначаем каждый уровень.

Какими же преимуществами обладает YAML-формат?

1. **Human-readable.** Опять же, даже увидев yaml-файл без описания, ты легко поймешь, какие объекты там описаны. YAML насколько хорошо читается человеком, что главная страница yaml.org — это обычный yaml-файл :)

2. **Компактность.** Структура файла формируется за счет пробелов: нет необходимости использовать скобки или кавычки.

3. **Поддержка структур данных, «родных» для языков программирования.** Огромное преимущество YAML перед JSON и многими другими форматами заключается в том, что он поддерживает разные структуры данных. В их числе:

   - !!map
     Неупорядоченный набор пар ключ:значение без возможности дубликатов;

   - !!omap
     Упорядоченная последовательность пар ключ:значение без возможности дубликатов;

   - !!pairs:
     Упорядоченная последовательность пар ключ:значение с возможностью дубликатов;

   - !!set
     Неупорядоченная последовательность значений, которые не равны друг другу;

   - !!seq
     Последовательность произвольных значений;

   Некоторые из этих структур знакомы тебе по Java! :) За счет этой фичи в формат YAML можно сериализовать разные структуры данных из языков программирования.

4. **Возможность использования anchor и alias**

Перевод слов «anchor» и «alias» — «якорь» и «псевдоним». В принципе, он довольно точно описывает суть этих терминов в YAML.

Они позволяют тебе идентифицировать какой-то элемент в yaml-файле, и ссылаться на него в остальных частях этого файла, если он встречается повторно. Anchor создается с помощью символа `&`, а alias — с помощью `*`.

Допустим, у нас есть файл с описанием книг Льва Толстого. Чтобы не писать имя автора каждый раз вручную, мы просто создадим якорь «leo» и будем ссылаться на него с помощью алиаса, когда нам это будет нужно:

```yaml
books:
 book-1:
   title: War and Peace
   author: &leo Leo Tolstoy
   year: 1869

 book-2:
   title: Anna Karenina
   author: *leo
   year: 1873

 book-3:
   title: Family Happiness
   author: *leo
   year: 1859
```

Когда мы будем считывать этот файл каким-то парсером, на месте нашего алиаса в нужных местах будет подставляться значение «Leo Tolstoy».

5. **В YAML можно встроить данные в других форматах.** Например, JSON:

```yaml
books: [
        {
          "title": "War and Peace",
          "author": "Leo Tolstoy",
          "year": 1869
        },

        {
          "title": "Anna Karenina",
          "author": "Leo Tolstoy",
          "year": 1873
        },

        {
          "title": "Family Happiness",
          "author": "Leo Tolstoy",
          "year": 1859
        }
      ]
```

## Другие форматы сериализации

### XML

Этот формат основан на так называемом дереве тегов.

```
<book>
   <title>Harry Potter and the Philosopher’s Stone</title>
   <author>J. K. Rowling</author>
   <year>1997</year>
</book>
```

Каждый элемент состоит из открывающего и закрывающего тега (<> и </>). У каждого элемента могут быть вложенные элементы.

XML — распространенный формат, не уступающий JSON и YAML (если говорить об использовании в реальных проектах). Об XML у нас есть отдельная лекция.

## BSON (binary JSON)

Как и следует из его названия, очень похож на JSON, но **не является human-readable** и оперирует данными в двоичном формате.

За счет этого он очень удобен при хранении и передаче изображений и других вложений. Кроме того, BSON поддерживает некоторые типы данных, недоступные в JSON. Например, в BSON-файл можно записать дату (в формате миллисекунд) или даже кусок JavaScript кода.

Популярная NoSQL база данных MongoDB хранит информацию именно в BSON формате.

## Position based protocol

В некоторых ситуациях нам необходимо резко снизить количество передаваемых данных (например, если данных очень много и нужно уменьшить нагрузку).

В этой ситуации мы можем использовать **position based protocol**, то есть передавать значения параметров без названий самих параметров.

```
"Leo Tolstoy" | "Anna Karenina" | 1873
```

Данные в таком формате занимают в разы меньше места, чем полноценный JSON файл.

Конечно, существуют и другие форматы сериализации, но тебе сейчас не нужно знать их все :) Хорошо, если ты будешь знаком с теми форматами, которые сейчас являются промышленным стандартом при разработке приложений, и будешь помнить их преимущества и отличия друг от друга.

А наша лекция на этом подошла к концу :)

Ответить 0

**Aleksandr Alekseenko** Network engineer 18 июля 2021, 02:24

Статья из мира практичности.
Надеюсь позволит пересмотреть мнение к стандартным библиотекам разработки.

Ответить +1

**Николай Т.** Уровень 40, Рязань, Россия 7 июля 2021, 11:00

Про характеристику интеллект для персонажей забыли, исправьте :)
И почему у вас вор менее ловкий чем маг? Увольте такого вора :)

Ответить 0

**Эли Гутман** Уровень 37 5 января 2021, 22:44

Сразу понял и запомнил принципы anchor-alias в YAML, поскольку играл в Морровинд, и это мне напомнило связку заклинаний "пометка-возврат".

Ответить 0

**Ярослав** Java Developer MASTER 21 мая 2019, 08:13

Даю совет всем, кто первый раз услышал про YAML: формат довольно популярный и приятный, разберитесь с ним и если уже со спрингом работали, попробуйте даже конфиги на .yaml писать, читабельность выше :)

Ответить +19

**MartyMcAir** Уровень 41, Россия 9 марта 2020, 19:17

Оо.. Вопрос.., в статье: JSON и XML. Что лучше?
Пишут, что JSON хоть и короче, но.. цитата: "Но я готов поспорить, что возможность отладки и исправления ошибок гораздо важнее, чем удобочитаемость."
Что это за отладка такая там?
Т.е. да по краткости, JSON и YAML не сильно отличаются, но все же.. От сюда вопрос т.е. в JSON - нет какой-то отладк,и, что есть в XML.. И тогда интересно... а в YAML - есть эта некая отладка?
(пытался гуглить ничего внятного не нашёл..)

Так же смотрел:
Tree — убийца JSON, XML, YAML , INI
языки в стиле XML и языки в стиле YAML
Как мы перевели конфигурирование наших сервисов с XML на YAML - от сюда узнал еще несколько минусов XML

Ответить +4

**Ярослав** Java Developer MASTER 10 марта 2020, 01:01

Сам не знаю, о какой такой возможности отладки и исправления ошибок рассказывается в xml.
Максимум о каких-нибудь xml схемах, но вряд ли это кто-то будет делать.
YAML сильно похож на JSON, у него мало общего с XML и вряд ли у YAML есть какой-нибудь аналог схем из xml.
Лично мне ни разу не пришлось дебажить xml, со стороны клиента что так, что так обычно дебажится в целом запрос и ответ от сервера, а не какой-то конкретный формат данных. И учитывая то, что лично в Java везде запихнуты уже готовые механизмы сериализации/десериализации любого формата данных, xml схемы вряд ли сильно нужны, ведь вручную сочинять xml вряд ли нужно будет.

Почитал статью про Tree, выглядит-то неплохо, но неплохо выглядят по сути все кандидаты в списке по-своему. Сейчас сильнее всего распространены JSON, YAML, они удобны, они подходят для большинства задач, сам JSON обычно юзается при работы сервера и клиента, на YAML я видел пишутся конфигурации. JSON удобно юзать, так как это JavaScript Object Notation, то есть его очень легко создавать для отправки на сервер клиенту.

Ответить +9

**MartyMcAir** Уровень 41, Россия 19 июня 2020, 23:03

спс, за разъяснения),
а то уже думал что там за отладка такая..

Ответить 0

RUSH

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

**ПОДПИСЫВАЙТЕСЬ**

**ЯЗЫК ИНТЕРФЕЙСА**

Русский