

# Avro1 入门概述

## Avro介绍

1. 数据串行化系统
2. 提供了丰富的数据结构，紧凑的快速的二进制格式，存储持久化数据的容器，远程过程调用
3. 动态语言的简单集成，代码生成不需要读写数据文件，也不需要实现RPC协议
4. 跨语言
5. 可压缩、可切分
6. 自描述语言：数据和数据结构都存在文本文件中，使用json的格式进行储存数据

## Avro数据类型

### 基本数据类型

数据类型	解释
null	Null is a type having no value.
int	32-bit signed integer.
long	64-bit signed integer.
float	single precision (32-bit) IEEE 754 floating-point number.
double	double precision (64-bit) IEEE 754 floating-point number.
bytes	sequence of 8-bit unsigned bytes.
string	Unicode character sequence.

### 复杂数据类型

1. Records，是以一种集合，类似于表结构，包括以下属性

```
name
namespace
type
fields
```

2. Enums

```
{
  "type": "enum",
  "name": "Numbers",
  "namespace": "data",
  "symbols": ["ONE", "TWO", "THREE", "FOUR"]
}
```

### 3. Arrays

```
{
  "type" : "array",
  "items" : "int"
}
```

### 4. Maps

```
{"type" : "map", "values" : "int"}
```

### 5. Unions

```
{
  "type" : "record",
  "namespace" : "tutorialspoint",
  "name" : "empdetails",
  "fields" : [
    { "name" : "experience", "type": ["int", "null"] },
    { "name" : "age", "type": "int" }
  ]
}
```

### 6. Fixed

```
{
  "type" : "fixed" ,
  "name" : "bdata",
  "size" : 1048576
}
```

## Avro的序列与反序列化

### 设计并创建schema模板

根据数据设计并创建schema模板, person.avsc (一张名为person的table,有两个字段name和age)

```
{
  "namespace": "com.example.avro",
  "type": "record",
  "name": "User",
  "fields": [
    { "name": "name", "type": "string" },
    { "name": "favorite_number", "type": ["int", "null"] },
    { "name": "favorite_color", "type": ["string", "null"] }
  ]
}
```

## 编译employee.avsc(不推荐直接使用命令)

```
$cmd> java -jar G:\Hadoop\hadoop配置\Avro\avro-tools-1.9.2.jar compile schema
person.avsc .
```

## 新建java项目，并将生成的类，导入到java项目中

### 添加Maven依赖

```
<dependency>
  <groupId>org.apache.avro</groupId>
  <artifactId>avro</artifactId>
  <version>1.9.2</version>
</dependency>
```

## 创建对象

```
Employee e1=new Employee();
    e1.setAge(11);
    e1.setName("tom1");

    //e2 and e3 ...
```

## 第一种办法：不生成schema Java类的方式

```
#直接上代码即可
package com.example.avro;

import org.apache.avro.Schema;
import org.apache.avro.file.DataFileReader;
import org.apache.avro.file.DataFileWriter;
import org.apache.avro.generic.GenericData;
import org.apache.avro.generic.GenericDatumReader;
import org.apache.avro.generic.GenericDatumWriter;
import org.apache.avro.generic.GenericRecord;
import org.apache.avro.io.DatumReader;
import org.apache.avro.io.DatumWriter;

import java.io.File;
import java.io.IOException;

/**
 * 不生成Schema代码的Avro示例.
 */
```

```

public class AvroDemowithoutCodeGeneration {
    public static void main(String[] args) throws IOException {
        // 读取Schema
        Schema schema = new Schema.Parser().parse(new
File("G:/Hadoop/mapreduce/src/main/data/person.avsc"));

        // 使用schema创建用户对象.
        GenericRecord user1 = new GenericData.Record(schema);
        user1.put("name", "Alyssa");
        user1.put("favorite_number", 256);

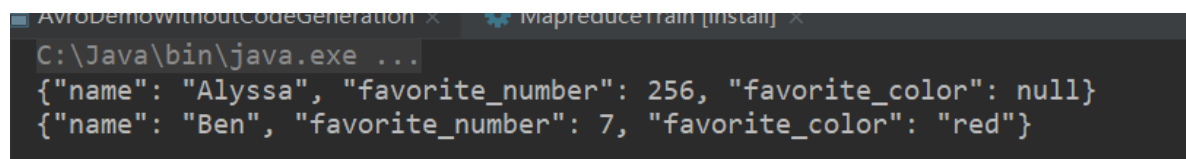
        GenericRecord user2 = new GenericData.Record(schema);
        user2.put("name", "Ben");
        user2.put("favorite_number", 7);
        user2.put("favorite_color", "red");

        // 序列化并写出到磁盘
        File file = new File("G:/Hadoop/mapreduce/src/main/output/users.avro");
        DatumWriter<GenericRecord> datumWriter = new
GenericDatumWriter<GenericRecord>(schema);
        DataFileWriter<GenericRecord> dataFileWriter = new
DataFileWriter<GenericRecord>(datumWriter);
        dataFileWriter.create(schema, file);
        dataFileWriter.append(user1);
        dataFileWriter.append(user2);
        dataFileWriter.close();

        // 从磁盘反序列化, 打印在控制台
        DatumReader<GenericRecord> datumReader = new
GenericDatumReader<GenericRecord>(schema);
        DataFileReader<GenericRecord> dataFileReader = new
DataFileReader<GenericRecord>(file, datumReader);
        GenericRecord user = null;
        while (dataFileReader.hasNext()) {
            user = dataFileReader.next(user);
            System.out.println(user);
        }
        // 运行结果:
        // {"name": "Alyssa", "favorite_number": 256, "favorite_color": null}
        // {"name": "Ben", "favorite_number": 7, "favorite_color": "red"}
    }
}

```

结果如图:

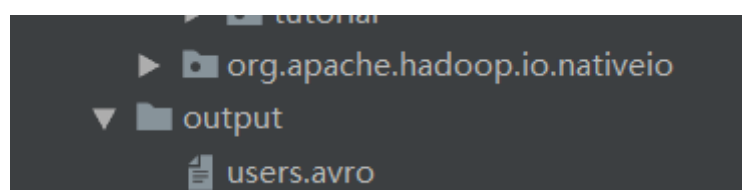


```

C:\Java\bin\java.exe ...
{"name": "Alyssa", "favorite_number": 256, "favorite_color": null}
{"name": "Ben", "favorite_number": 7, "favorite_color": "red"}

```

在output目录下有已经插入好数据的avro文件



## 第二种办法：生成Schema Java类的方式

- 需要添加maven插件

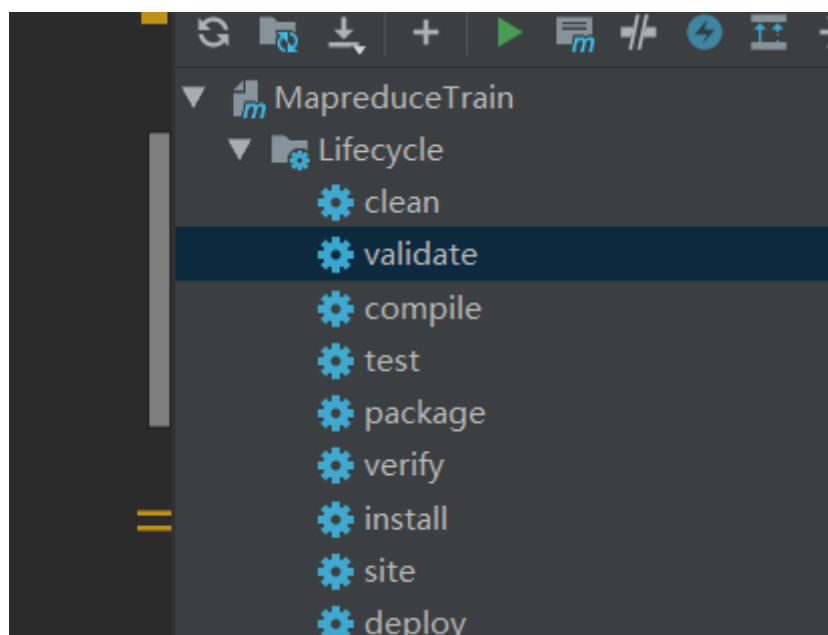
```
<plugin>
  <groupId>org.apache.avro</groupId>
  <artifactId>avro-maven-plugin</artifactId>
  <version>1.7.7</version>
  <executions>
    <execution>
      <phase>generate-sources</phase>
      <goals>
        <goal>schema</goal>
      </goals>
      <configuration>

<sourceDirectory>${project.basedir}/src/main/data/</sourceDirectory>  //源路径

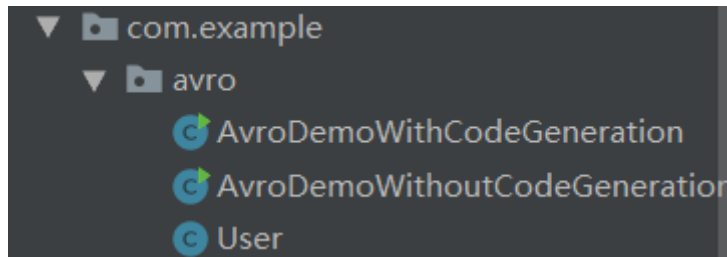
<outputDirectory>${project.basedir}/src/main/java/</outputDirectory>  //生成java文件路径

      </configuration>
    </execution>
  </executions>
</plugin>
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <encoding>utf-8</encoding>
  </configuration>
</plugin>
```

- 对person.avsc执行idea的compile指令（注意文件生成路径）



结果如下图所示：



- 编写代码

```
package com.example.avro;

import org.apache.avro.file.DataFileReader;
import org.apache.avro.file.DataFileWriter;
import org.apache.avro.io.DatumReader;
import org.apache.avro.io.DatumWriter;
import org.apache.avro.specific.SpecificDatumReader;
import org.apache.avro.specific.SpecificDatumWriter;

import java.io.File;
import java.io.IOException;

/**
 * Avro示例（使用CodeGeneration）。
 *
 */
public class AvroDemoWithCodeGeneration {
    public static void main(String[] args) throws IOException {
        // 使用三种不同的方式实例化三个user
        User user1 = new User();
        user1.setName("Alyssa");
        user1.setFavoriteNumber(256);

        User user2 = new User("Ben", 7, "red");

        User user3 = User.newBuilder()
            .setName("Charlie")
            .setFavoriteColor("blue")
            .setFavoriteNumber(null)
            .build();

        // 序列化user1, user2 and user3 到磁盘
        DatumWriter<User> userDatumWriter = new SpecificDatumWriter<User>
(User.class);
        DataFileWriter<User> dataFileWriter = new DataFileWriter<User>
(userDatumWriter);
        File file = new File("G:/Hadoop/mapreduce/src/main/output/users1.avro");
        dataFileWriter.create(user1.getSchema(), file);
        dataFileWriter.append(user1);
        dataFileWriter.append(user2);
        dataFileWriter.append(user3);
        dataFileWriter.close();

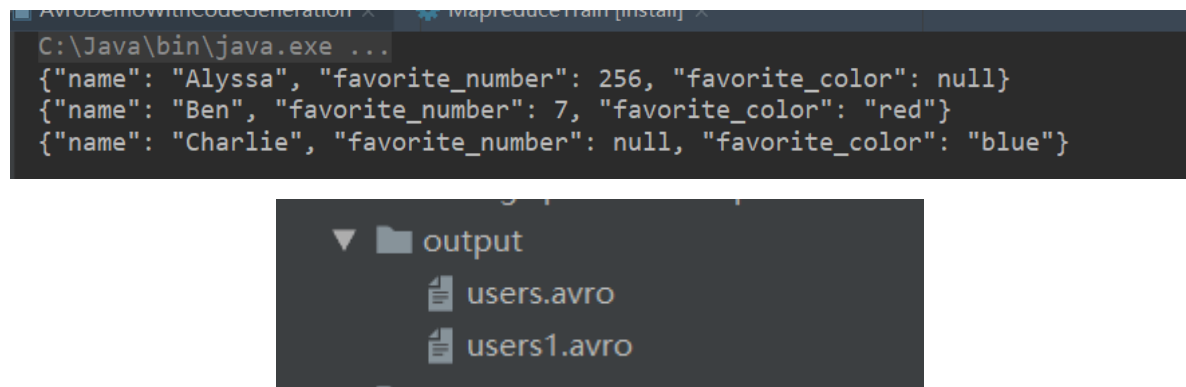
        // 从磁盘反序列化
        DatumReader<User> userDatumReader = new SpecificDatumReader<User>
(User.class);
        DataFileReader<User> dataFileReader = new DataFileReader<User>(file,
userDatumReader);
```

```
User user = null;
while (dataFileReader.hasNext()) {
    user = dataFileReader.next(user);
    System.out.println(user);
}

// 运行结果
/* {"name": "Alyssa", "favorite_number": 256, "favorite_color": null}
   {"name": "Ben", "favorite_number": 7, "favorite_color": "red"}
   {"name": "Charlie", "favorite_number": null, "favorite_color": "blue"}*/

}
```

结果如下:



```
C:\Java\bin\java.exe ...
{"name": "Alyssa", "favorite_number": 256, "favorite_color": null}
{"name": "Ben", "favorite_number": 7, "favorite_color": "red"}
{"name": "Charlie", "favorite_number": null, "favorite_color": "blue"}

output
  users.avro
  users1.avro
```