مثال هایی از کاربرد Snake YAML برای خواندن اطلاعات کانفیگ برنامه در جاوا

# خواندن YAML در جاوا با استفاده از Jackson

**تاریخ مقاله:** 30 می 2016

**تاریخ ترجمه:** 15 دی 1395

**آدرس:** <https://dzone.com/articles/read-yaml-in-java-with-jackson>

Jackson یکی از بهترین کتابخانه های JSON در جاواست. حالا با استفاده از افزونه[[1]](#footnote-1) YAML آن می توانید فرمت YAML را پردازش کنید.

در این مقاله خواهیم دید که چگونه می توانیم با استفاده از کلاس YamlFactory  کتابخانه Jackson فرمت YAML را به Bean جاوایی بخوانیم.

**YAML چیست؟**

YAML یک فرمت بسیار کاربردی برای نوشتن فایل های پیکربندی و تنظیمات است. در سایت [Yaml.org](http://yaml.org/) می توانید کتابخانه های مختلفی از این فرمت برای زبان های برنامه نویسی مختلف را مشاهده نمایید. نظیر PHP, Java, Python, Perl و غیره.

نکته: در پشت صحنه[[2]](#footnote-2) Jackson از [SnakeYAML](https://bitbucket.org/asomov/snakeyaml/" \t "_blank)  برای خواندن فرمت YAML استفاده می کند.

**افزودن وابستگی های Maven یی**

<dependencies>

<dependency>

<groupId>com.fasterxml.jackson.dataformat</groupId>

<artifactId>jackson-dataformat-yaml</artifactId>

<version>2.3.0</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.2.3</version>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>3.4</version>

</dependency>

</dependencies>

[ به افزونه Jackson به منظور پشتیبانی از فرمت غیر JSON یی در اینجا YAML توجه کنید. ]

[نکته: وابستگی commons-lang3 صرفا برای toString کردن یک آبجکت بدون نیاز به این که متد مربوطه را صریحا پیاده سازی کنیم، استفاده شده است. ]

**خواندن فایل های YAML به آبجکت های Java**

مثال زیر را در نظر بگیرید:

# Details of a user

---

name: Test User

age: 30

address:

line1: My Address Line 1

line2: Address line 2

city: Washington D.C.

zip: 20000

roles:

- User

- Editor

اجازه دهید یک Bean جاوایی به نام User.java درست کنیم که این اطلاعات را در خود نگاه خواهد داشت:

package com.mms.mja.blog.demo.yaml;

import java.util.Map;

public class User {

private String name;

private int age;

private Map<String, String> address;

private String[] roles;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public Map<String, String> getAddress() {

return address;

}

public void setAddress(Map<String, String> address) {

this.address = address;

}

public String[] getRoles() {

return roles;

}

public void setRoles(String[] roles) {

this.roles = roles;

}

}

استفاده از Jackson برای خواندن این اطلاعات

package com.mms.mja.blog.demo.yaml;

import java.io.File;

import org.apache.commons.lang3.builder.ReflectionToStringBuilder;

import org.apache.commons.lang3.builder.ToStringStyle;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.fasterxml.jackson.dataformat.yaml.YAMLFactory;

public class YamlTesting {

public static void main(String[] args) {

ObjectMapper mapper = new ObjectMapper(new YAMLFactory());

try {

User user = mapper.readValue(new File("user.yaml"), User.class);

System.out.println(ReflectionToStringBuilder.toString(user,ToStringStyle.MULTI\_LINE\_STYLE));

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

[ همان طور که مشاهده می کنیم تنها نکته متمایز از خواندن فرمت YAML با استفاده از Jackson پاس دادن YAMLFactory() به کلاس ObjectMapper است. ]

در ادامه خروجی برنامه را مشاهده می کنید :

com.mms.mja.blog.demo.yaml.User@36d4b5c[

name=Test User

age=30

address={line1=My Address Line 1, line2=Address line 2, city=Washington D.C., zip=20000}

roles={User,Editor}

]

کلاس YamlFactory  آدرس را به MAP و رول ها را به آرایه رشته ای نگاشت کرده است. شما همچنین می توانید یک Address.java بسازید که آدرس را نگاه دارد به جای این که از MAP استفاده کنید.

[ انواع افزونه های[[3]](#footnote-3) Jackson برای پشتیبانی از فرمت های داده ای غیر JSON]

ابسترکت های Jackson (Jackson abstractions)

* streaming API
* data binding
* tree model

groupId: [com.fasterxml.jackson.dataformat](http://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat)

[jackson-dataformat-yaml](http://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-yaml) : برای خواندن و نوشتن فرمت YAML

[jackson-dataformat-xml](http://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-xml) : برای خواندن و نوشتن فرمت XML. این قابلیت بر روی Stax API (javax.xml.stream) پیاده سازی شده است. Jackson یک سری تایپ های استریمینگ دارد نظیر JsonGenerator، JsonParser و JsonFactory. که با پیاده سازی آنها و همچنین استفاده از Stax این امر محقق شده است. همچنین بعضی از تایپ های بایندینگ نظیر ObjectMapper پیاده سازی شده اند مثلا به XmlMapper.

[jackson-dataformat-smile](http://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-smile) : برای خواندن و نوشتن فرمت Smile  (JSON باینری)

[jackson-dataformat-csv](http://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-csv) : برای خواندن و نوشتن فرمت CSV

[jackson-dataformat-cbor](http://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-cbor) : برای خواندن و نوشتن [[4]](#footnote-4)GBOR (Concise Binary Object Representation)

[jackson-dataformat-properties](http://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-properties) : برای خواندن و نوشتن فرمت Java Property. ساختارهای تودرتو با یک جداساز نظیر نقطه از هم جدا شده اند.

[jackson-dataformat-avro](http://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-avro) : برای خواندن و نوشتن دیتاهای انکد شده به فرمت AVRO

[jackson-dataformat-protobuf](http://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-protobuf) : برای خواندن و نوشتن دیتای انکد شده به فرمت پروتکل protobuf گوگل.

1. Extension [↑](#footnote-ref-1)
2. Under the hood [↑](#footnote-ref-2)
3. Data format extension [↑](#footnote-ref-3)
4. https://www.rfc-editor.org/info/rfc7049 [↑](#footnote-ref-4)