جافا في □□□□ معالجة

التي جاكسون، مكتبة من الاستفادة يمكنك ،□□□□ لمعالجة □□□□ مشروع في com.fasterxml.jackson حزمة لاستخدام مع البدء، على للحصول بخطوة خطوة دليل أدناه العكس. و □□□□ إلى □□□□ الكائنات لتحويل واسع نطاق على ثستخدم عالية وظائف توفر التي ،jackson-databind وحدة على الدليل هذا يركز .□□□□□ على قائم مشروع مع تعمل أنك افتراض المستوى لترابط البيانات □□□□.

---

1. مشروعك إلى جاكسون التبعية إضافة

التبعية أضف ،□□□□□ تستخدم كنت إذا مشروعك. إلى جاكسون مكتبة إضافة عليك ،com.fasterxml.jackson حزمة لاستخدام بك: الخاص pom.xml ملف إلى التالية

```xml
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.12.5</version> <!--                  -->
</dependency>
```

- □ **ملاحظة**: تحقق من مستودع □□□□□ المركزي للحصول على أحدث نسخة، قد تكون أحدث من 2.12.5.
- □ تعتمد وحدة jackson-databind على jackson-core و jackson-annotations، فلا تحتاج إلى إضافتهما بشكل منفصل إلا إذا كان لديك متطلبات محددة.

المكتبة. لتنزيل □□□□□ المتكاملة التطوير بيئة في مشروعك تحديث أو mvn install بتشغيل قم التبعية، إضافة بعد

---

2. ObjectMapper مثيل إنشاء

للخطوط آمنة إنها .□□□□ لعمليات الرئيسية الأداة هي com.fasterxml.jackson.databind حزمة من ObjectMapper فئة استخدامه: إعادة يمكن واحد مثيل إنشاء الأفضل من لذا إنشاءها، عند كبير بشكل الموارد وتستهلك المتعددة

```java
import com.fasterxml.jackson.databind.ObjectMapper;

public class JsonExample {
    private static final ObjectMapper mapper = new ObjectMapper();
}
```

ضع هذا في فئة حيث ستقوم بعمليات □□□□.

---

3. تحويل كائن إلى [illegible] إلى [illegible] [illegible]التسلسل

تحويل كائن إلى [illegible] سلسلة إلى [illegible]، استخدم طريقة writeValueAsString. على سبيل المثال:

[illegible] فئة تعريف [illegible] إنشاء فئة مع الحقول التي تريد تسلسلها. تأكد من أن هيكل طرق الوصول و [illegible] الإعداد [illegible]، الخاصة: الحقول إلى للوصول افتراضي بشكل يستخدمها الكود، أن

```
public class MyClass {
    private String field1;
    private int field2;

    public MyClass(String field1, int field2) {
        this.field1 = field1;
        this.field2 = field2;
    }

    public String getField1() {
        return field1;
    }

    public void setField1(String field1) {
        this.field1 = field1;
    }

    public int getField2() {
        return field2;
    }

    public void setField2(int field2) {
        this.field2 = field2;
    }
}
```

تسلسل إلى [illegible] استخدم ObjectMapper لتحويل الكائن إلى [illegible]:

```
import com.fasterxml.jackson.databind.ObjectMapper;

public class JsonExample {
    private static final ObjectMapper mapper = new ObjectMapper();
```

```
    public static void main(String[] args) throws Exception {
        MyClass obj = new MyClass("value1", 123);
        String json = mapper.writeValueAsString(obj);
        System.out.println(json);
    }
}
```

الخرج:

```
{"field1":"value1","field2":123}
```

---

4. تحويل □□□□ إلى كائن □□□□ □الترميز□

لتحويل سلسلة □□□□ إلى كائن □□□□، استخدم طريقة readValue:

```
import com.fasterxml.jackson.databind.ObjectMapper;

public class JsonExample {
    private static final ObjectMapper mapper = new ObjectMapper();

    public static void main(String[] args) throws Exception {
        String json = "{\"field1\":\"value1\",\"field2\":123}";
        MyClass obj = mapper.readValue(json, MyClass.class);
        System.out.println(obj.getField1()); //     "value1"
    }
}
```

- □ **معالجة الأخطاء**: طريقة readValue تثير استثناء JsonProcessingException □استثناء □محقق إذا كان □□□□ غير صحيح أو لا يتطابق مع بنية الفئة. قم بمعالجته باستخدام كتلة □□□□□□□□□ أو إعلانها في توقيع الطريقة:

```
try {
    MyClass obj = mapper.readValue(json, MyClass.class);
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
```

---

5. التعليقات باستخدام [illegible] مع الجة تخصيص

com.fasterxml.jackson.annotation من التعليقات هذه أضف ترميزها. أو الحقول تسلسل كيفية لتخصيص تعليقات جاكسون يوفر فيئة: إلى

الحقل اسم تغيير استخدم `@JsonProperty` حقل لتخزين [illegible] إلى اسم حقل [illegible] مختلف:

```
import com.fasterxml.jackson.annotation.JsonProperty;

public class MyClass {
    @JsonProperty("name")
    private String field1;
    private int field2;
    //
}
```

الخرج:

```
{"name":"value1","field2":123}
```

تجاهل حقل استخدم `@JsonIgnore` التسلسل: من حقل لتفادي

```
import com.fasterxml.jackson.annotation.JsonIgnore;

public class MyClass {
    private String field1;
    @JsonIgnore
    private int field2;
    //
}
```

الخرج:

```
{"field1":"value1"}
```

التواريخ تنسيق استخدم `@JsonFormat` التواريخ: تسلسل كيفية لتحديد

```
import com.fasterxml.jackson.annotation.JsonFormat;
import java.util.Date;

public class MyClass {
```

```java
    private String field1;
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")
    private Date date;
    //
}
```

□مثال□: الخرج

```json
{"field1":"value1","date":"2023-10-25"}
```

---

6. معالجة السيناريوهات المتقدمة

مفيدة: تجدها قد التي الإضافية الميزات بعض هنا

writerWithDefaultPrettyPrinter: استخدم للقراءة، قابل □□□□ للخرج **جميلة □□□□ طباعة**

```java
String prettyJson = mapper.writerWithDefaultPrettyPrinter().writeValueAsString(obj);
```

الخرج:

```json
{
  "field1" : "value1",
  "field2" : 123
}
```

لتجاهل: ObjectMapper بتكوين قم، □□□□ فئة في موجودة غير حقول على يحتوي □□□□ كان إذا **المجهولة الخصائص تجاهل**

```java
mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false);
```

ملف: إلى كتابة أو من قراءة **الملفات مع العمل**

```java
//
mapper.writeValue(new File("output.json"), obj);

//
MyClass obj = mapper.readValue(new File("input.json"), MyClass.class);
```

العام: النوع معلومات على للحفاظ TypeReference استخدم للمجموعات،   **العامة الأنواع أو القوائم مع التعامل**

```java
import com.fasterxml.jackson.core.type.TypeReference;
import java.util.List;

List<MyClass> list = mapper.readValue(json, new TypeReference<List<MyClass>>() {});
```

---

7. معالجة □□□□ الديناميكي باستخدام JsonNode

الشجرة: لمقاربة JsonNode استخدم محددة، فئة إلى □□□ خريطة في ترغب لم إذا

```java
import com.fasterxml.jackson.databind.JsonNode;

JsonNode node = mapper.readTree("{\"field1\":\"value1\",\"field2\":123}");
String value = node.get("field1").asText(); //     "value1"
```

ديناميكي: بشكل □□□ بناء أيضًا يمكنك

```java
import com.fasterxml.jackson.databind.node.ObjectNode;

ObjectNode node = mapper.createObjectNode();
node.put("field1", "value1");
node.put("field2", 123);
String json = mapper.writeValueAsString(node);
```

ملخص

من واحد مثيل إنشاء .2 .pom.xml ملف إلى jackson-databind التبعية أضف .1 :com.fasterxml.jackson حزمة لاستخدام ObjectMapper لإعادة الاستخدام. 3. استخدم writeValueAsString لتحويل الكائنات إلى □□□□ و readValue لتحويل □□□□ إلى الكائنات. 4. تخصيص السلوك باستخدام التعليقات مثل @JsonProperty و @JsonIgnore و @JsonFormat. 5. معالجة الاستثناءات مثل □JsonProcessingException□ وتكوين ObjectMapper حسب الحاجة مثل□ الخصائص تجاهل□ الديناميكي. □□□□ لمعالجة JsonNode أو الملفات، إدخال□إخراج جميلة، طباعة مثل المتقدمة الميزات استكشاف 6. المجهولة□.

فعال. بشكل جاكسون مكتبة باستخدام بك الخاص □□□□ تطبيق في □□□□ معالجة يمكنك الخطوات، بهذه