# 1. Thư viện

Example: http://websystique.com/java/json/jackson-json-annotations-example/

Muốn sử dụng Json Annotation phải thêm thư viện:

\_ jackson-annotations-2.6.3

**import** com.fasterxml.jackson.annotation.\*;

\_ jackson-databind-2.6.3

**import** com.fasterxml.jackson.databind.annotation.JsonSerialize;

\_ jackson-core-2.6.3.jar

**import** com.fasterxml.jackson.core.JsonGenerator;

**import** com.fasterxml.jackson.core.JsonProcessingException;

# 2. Demo

## Class Car:

**package** com.jsonannotations;

**import** java.util.\*;

**import** com.fasterxml.jackson.annotation.\*;

**import** com.fasterxml.jackson.databind.annotation.JsonSerialize;

@JsonIgnoreProperties({"ignoreme1","ignoreme2"})

**public** **class** Car {

@JsonProperty

**private** String name;

@JsonProperty("carModel")

**private** **int** model;

@JsonProperty

**private** String price;

**private** String ignoreme1;

**private** String ignoreme2;

@JsonProperty

**private** List<String> colors = **new** ArrayList<>();

@JsonProperty

@JsonSerialize(using = CustomDateSerializer.**class**)

**private** Date promoDate;

**private** Map<String, Object> otherProperties = **new** HashMap<>();

@JsonAnyGetter

**public** Map<String, Object> getOtherProperties() {

**return** otherProperties;

}

@JsonAnySetter

**public** **void** setOtherProperties(Map<String, Object> otherProperties) {

**this**.otherProperties = otherProperties;

}

@JsonAnySetter

**public** **void** set(String name, Object value){

otherProperties.put(name, value);

}

@Override

**public** String toString() {

**return** "Car [name=" + name + ", model=" + model + ", price=" + price

+ ", ignoreme1=" + ignoreme1 + ", ignoreme2=" + ignoreme2

+ ", colors=" + colors + ", promoDate=" + promoDate

+ ", otherProperties=" + otherProperties + "]";

}

}

## Class CustomDateSerializer:

**package** com.jsonannotations;

**import** java.io.IOException;

**import** java.text.SimpleDateFormat;

**import** java.util.\*;

**import** com.fasterxml.jackson.core.JsonGenerator;

**import** com.fasterxml.jackson.core.JsonProcessingException;

**import** com.fasterxml.jackson.databind.JsonSerializer;

**import** com.fasterxml.jackson.databind.SerializerProvider;

**public** **class** CustomDateSerializer **extends** JsonSerializer<Date>{

**private** **static** **final** SimpleDateFormat ***dateFormat*** = **new** SimpleDateFormat("dd/MM/yyyy");

@Override

**public** **void** serialize(Date date, JsonGenerator generator, SerializerProvider provider)

**throws** IOException, JsonProcessingException {

String formattedDate = ***dateFormat***.format(date);

generator.writeString(formattedDate);

}

}

## Class Demo:

**package** com.jsonannotations;

**import** java.io.File;

**import** java.io.IOException;

**import** com.fasterxml.jackson.core.JsonParseException;

**import** com.fasterxml.jackson.databind.JsonMappingException;

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

**public** **class** JsonJacksonAnnotationsDemo {

**public** **static** **void** main(String[] args) **throws** JsonParseException, JsonMappingException, IOException {

ObjectMapper mapper = **new** ObjectMapper();

Car car = mapper.readValue(**new** File("inputfile.json"), Car.**class**);

mapper.writeValue(**new** File("outputfile.json"), car);

System.***out***.println(car);

}

}

## Inputfile.json:

{

"name": "AUDI",

"carModel": 2014,

"price": 30000,

"foo": "bar",

"tom": "jerry",

"ignoreme1": "ig1",

"ignoreme2": "ig2",

"colors": [

"GRAY",

"BLACK",

"WHITE"

],

"promoDate": 1399647200000

}

## Result:

Ta được file Outputfile.json như sau:

{

"name": "AUDI",

"price": "30000",

"colors": [

"GRAY",

"BLACK",

"WHITE"

],

"promoDate": "09/05/2014",

"carModel": 2014,

"tom": "jerry",

"foo": "bar"

}

Và kết quả xuất ra màn hình console sẽ khác:



## Giải thích:

### 1) @JsonProperty :

This annotation can be used on a property or method which will be used for Serialization and Deserialization of JSON. It takes an optional ‘name’ parameter which is useful in case the property name is different than ‘key’ name in json.

*When to use:*  
– If you just want to declare property and not getter/setters.  
– If you are using property and getter/setters BUT want to use a different getter/setter or property name than the one coming in JSON ['key' name]. Just set the name parameter in annotations with actual ‘key’ in JSON.

For example, above POJO [car.java] does not contains any getter/setter for individual property. Instead only the properties are declared and annotated with @JsonProperty. By default, if key name matches with property name, value is mapped to property value. But you can provide a name attribute in case the key name in input is different than property name in POJO [e.g. model property is mapped to carModel input]

@JsonProperty sử dụng cho property hoặc method, dùng cho serialization và deserialization. Mỗi property sẽ map với 1 phần tử trong file json. Nếu tên khác với trong file json thì có thể sử dụng thêm param. Sử dụng khi: muốn khai báo property nhưng k getter/setter, khai báo param khi key trong code và key trong file Json không giống nhau.

### **2) @JsonIgnoreProperties :**

This Class level annotation can be used to prevent certain properties to be serialized & deserialized. What it means is that they will not be mapped to JSON content.

*When to use:*  
– If you want to ignore serialization/deserialization of certain properties [For example some properties are valid in past but not anymore].

For example, In our case, we don’t want property “ignoreme1″ & “ignoreme2″ to be mapped. So we have included these properties in JsonIgnoreProperties. You can see from content of outputfile.json that it does not contain ignoreme1 & ignoreme2 (means they were not serialized). Similarly, from toString output it is clear that ignoreme1 & ignoreme2 were not deserailzed (they did not contain any value).

ignoreme1 và ignoreme2 là 2 key trong json, vì thế khi sử dụng @JsonIgnoreProperties cho class này thì ignoreme1 và ignoreme2 k được map.

### **3) @JsonAnySetter, @JsonAnyGetter :**

These annotations works as a Catch-All and are applied on Getters/Setter working with a Map. If there is any JSON value which is not mapped to a property in POJO, then that value can be caught by @JsonAnySetter, and stored (deserialized) into Map. Similarly the values which are stored into Map can be serialized back to JSON using @JsonAnyGetter.

*When to use:*  
– If you don’t want to declare a property or method for every possible ‘key’ in JSON, but still want to capture(serialize/deserialize) the data.

For example, in our case, wee don’t have properties ‘tom’ or ‘foo’ defined in pojo. But by using @JsonAnySetter & @JsonAnyGetter, we were able to store those values(deserialized) [otherProperties={tom=jerry, foo=bar}] and create them again(serialized)["tom":"jerry","foo":"bar"].

Sử dụng 2 annotation này để map với các giá trị còn lại.

### **4) @JsonSerialize :**

This annotation can be used to change[or customize] the default serialization (Java to JSON) process.

*When to use:*  
– When default serialization is not fitting your needs and you want a custom behavior to be applied.

For example, in our input JSON, the “promoDate” field {“promoDate”:1399647200000} contains date in milliseconds format. In case we want to write the date in some other format(dd/MM/yyyy e.g.), we can do so using a Custom Serializer. We wrote one CustomDateSerializer. So before writing JSON output, this serializer will be called for applicable field(the one annotated with @JsonSerialize(using = CustomDateSerializer.class), and as a result convert the output in desired format before writing that output.

Dùng để format lại giá trị theo ý của coder, ví dụ như format lại ngày tháng.

## Link thêm khảo nhiều hơn các annotation:

http://fasterxml.github.io/jackson-annotations/javadoc/2.5/

# 3. Annotations trong java

<http://o7planning.org/vi/10197/huong-dan-su-dung-java-annotation>

<http://www.vanthuong.com/2016/03/java-annotation.html>

<http://laptrinh.vn/d/4710-annotation-trong-spring-mvc.html>

**Annotation** (Chú thích) được sử dụng để cung cấp thông tin dữ liệu cho mã Java của bạn. Là thông tin dữ liệu, các **Annotation** không trực tiếp ảnh hưởng đến việc thực hiện các mã của bạn, mặc dù một số loại chú thích thực sự có thể được sử dụng cho mục đích đó. **Annotation** đã được thêm vào Java từ Java 5.

## **1. @Deprecated**

Thông báo tự động của Eclipse hoặc của trình biên dịch, thường là thông báo cái method mình sử dụng đã lỗi thời, nên dùng method khác. Eclipse sẽ gạch ngang mehod lỗi thời.

**package** com.jsonannotations;

**import** java.util.Date;

**public** **class** DeprecatedDemo {

/\*\*

\* **@deprecated** replaced by {@link #todo(String,Date)}

\*/

@Deprecated

**public** **void** ~~todoJob~~(String jobName) {

System.***out***.println("Todo " + jobName);

}

**public** **void** todo(String jobName, Date atTime) {

System.***out***.println("Todo " + jobName + " at " + atTime);

}

**public** **void** todoNothing() {

System.***out***.println("Todo Nothing");

}

**public** **static** **void** main(String[] args) {

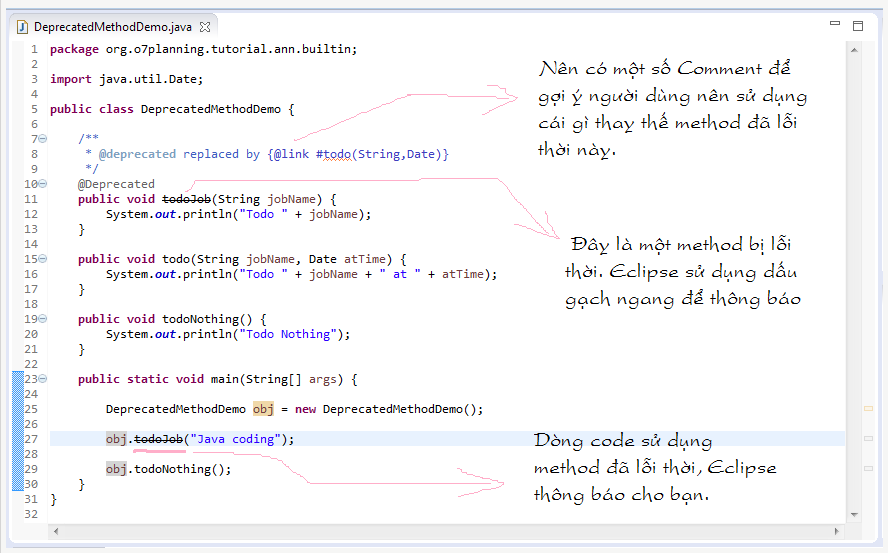
DeprecatedDemo obj = **new** DeprecatedDemo();

obj.~~todoJob~~("Java coding");

obj.todoNothing();

}

}



## 2. @Override

Phương thức ghi đè quen thuộc. Nếu nó là method k phải ghi đè mà để @Override thì sẽ báo lỗi ngay.

## **3. @SuppressWarnings**

Để tắt cảnh báo của trình biên dịch như: thông báo lỗi thời, ép kiểu k an toàn, khai báo biến k sử dụng.

## 4. Viết Annotation của bạn:

Sử dụng ***@interface*** là từ khóa khai báo một Annotation, annotation khá giống một interface. Annotation có hoặc không có các phần tử (element) trong nó. Đặc điểm phần tử của annotation:

* Không có thân hàm
* Không có tham số hàm
* Khai báo trả về phải là một kiểu cụ thể:
  + Các kiểu nguyên thủy (boolean, int, float, ...)
  + Enum
  + Annotation
  + Class (Ví dụ String.class)
* Có thể có giá trị mặc định

### 4.1 Annotation đầu tiên:

**package** com.jsonannotations;

**public** **@interface** MyFirstAnnotation {

// Phan tu name

**public** String name();

// Phan tu description, co gia tri mac dinh ""

**public** String description() **default** "";

}

Annotation có thể được gắn trên:

1. TYPE - Gắn trên khai báo Class, interface, enum, annotation.
2. FIELD - Gắn trên khai báo trường (field), bao gồm cả các hằng số enum.
3. METHOD - Gắn trên khai báo method.
4. PARAMETER - Gắn trên khai báo parameter
5. CONSTRUCTOR - Gắn trên khai báo cấu tử
6. LOCAL\_VARIABLE - Gắn trên biến địa phương.
7. ANNOTATION\_TYPE - Gắn trên khai báo Annotation
8. PACKAGE - Gắn trên khai báo package.

**package** com.jsonannotations;

**public** **class** UsingMyFirstAnnotation {

// Annotation duoc gan tren cau tu

// Voi gia tri cua phan tu name la "John"

// Gia tri phan tu description la "Write by John"

@MyFirstAnnotation(name = "John", description = "Write by John")

**public** UsingMyFirstAnnotation(){

}

//Annotation duoc gan tren method

// Voi gia tri phan tu name la "Tom"

// Phan tu description khong khoi tao, no se lay theo gia tri mac dinh

@MyFirstAnnotation(name = "Tom")

**public** **void** someMethod(){

}

// Annotation gan tren tham so cua method

**public** **void** todo(@MyFirstAnnotation(name = "none") String job){

// Annotation duoc gan len bien dia phuong

@MyFirstAnnotation(name = "Som name")

**int** localVariable = 0;

}

}

### 4.2 Annotation với phần tử value (có sự đặc biệt)

**package** com.jsonannotations;

**public** **@interface** AnnWithValue {

// Phan tu co ten value, cua Annotation

// Co mot chut dac biet khi su dung phan tu nay

**public** **int** value();

// Phan tu name cua Annotation

**public** String name() **default** "";

}

Sử dụng:

**package** com.jsonannotations;

**public** **class** UsingAnnWithValue {

// Khoi tao cac phan tu cua Annotation theo cach thong thuong

@AnnWithValue(name = "Name1", value = 100)

**public** **void** someMethod1(){

}

// Khoi tao cac phan tu cua Annotation theo cach thong thuong

// Phan tu name cua Annotation lay mac dinh

@AnnWithValue(value = 100)

**public** **void** someMethod2(){

}

// Voi phan tu value, ban co the chi can ghi ra gia tri cua no

@AnnWithValue(100)

**public** **void** someMethod3(){

}

}

### 4.3 @Retention & @Target

@Retention & @Target là 2 annotation sẵn có của Java.   
  
**@Retention**: Dùng để chú thích mức độ tồn tại của một annotation nào đó.   
Cụ thể có 3 mức nhận thức tồn tại của vật được chú thích:

1. **RetentionPolicy.SOURCE**: Tồn tại trên code nguồn, và không được bộ dịch (compiler) nhận ra.
2. **RetentionPolicy.CLASS**: Mức tồn tại được bộ dịch nhận ra, nhưng không được nhận biết bởi máy ảo tại thời điểm chạy (Runtime).
3. **RetentionPolicyRUNTIME**: Mức tồn tại lớn nhất, được bộ dịch (compiler) nhận biết, và máy ảo thời điểm chạy cũng nhận ra sự tồn tại của nó.

**@Target**: Dùng để chú thích cho một annotation khác, và annotation đó sẽ được sử dụng trong phạm vi nào.

1. **ElementType.TYPE** - Gắn trên khai báo Class, interface, enum, annotation.
2. **ElementType.FIELD** - Gắn trên khai báo trường (field), bao gồm cả các hằng số enum.
3. **ElementType.METHOD** - Gắn trên khai báo method.
4. **ElementType.PARAMETER** - Gắn trên khai báo parameter
5. **ElementType.CONSTRUCTOR** - Gắn trên khai báo cấu tử
6. **ElementType.LOCAL\_VARIABLE** - Gắn trên biến địa phương.
7. **ElementType.ANNOTATION\_TYPE** - Gắn trên khai báo Annotation
8. **ElementType.PACKAGE** - Gắn trên khai báo package.

**package** com.jsonannotations;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

//Chu thich nay noi rang AnnFM chi duoc nhan biet tren code nguon

// Se khong duoc bo dich biet

// va trong thoi gian chay may ao cung khong biet su ton tai cua no

@Retention(value = RetentionPolicy.***SOURCE***)

// AnnFM se chi duoc gan tren FIELD hoac METHOD

@Target(value = {ElementType.***FIELD***, ElementType.***METHOD***})

**public** **@interface** AnnFM {

}

Class sử dụng:

**package** com.jsonannotations;

**public** **class** UsingAnnFM {

// AnnFM gan tren FIELD

@AnnFM

**protected** **int** someField = 100;

// AnnFM gan tren METHOD

@AnnFM

**public** **void** someMethod(){

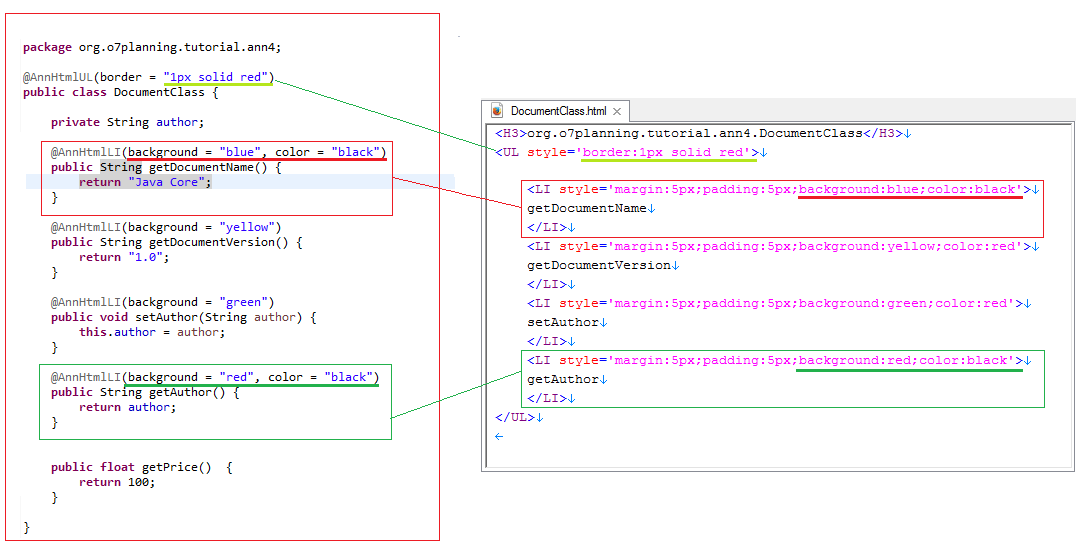
}

}

### 4.4 Annotation & Reflection

**Java Reflection** có thể nhận biết được những thứ (Class, field, method, ..) được chú thích bởi một Annotation nào đó. Và tất nhiên nó chỉ nhận biết được các Annotation có ***@Retention(RetentionPolicy.RUNTIME)***

Ví dụ dưới tiếp theo mô phỏng một chương trình đọc các chú thích trên các file nguồn Java và tạo ra các file Html. Mỗi class tương ứng với một file html.



**AnnHtmlUL.java**

**package** com.annotation;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Retention(RetentionPolicy.***RUNTIME***)

// Chi gan tren class, interface, annotation, enum

@Target(value = {ElementType.***TYPE***})

// Mo phong the <UL> trong HTML.

**public** **@interface** AnnHtmlUL {

**public** String border() **default** "border:1px solid blue;";

}

**AnnHtmlLI.java**

**package** com.annotation;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Retention(RetentionPolicy.***RUNTIME***)

//Chi gan tren Field, Method

@Target(value = {ElementType.***FIELD***, ElementType.***METHOD***})

// Mo phong the LI trong HTML

**public** **@interface** AnnHtmlLI {

**public** String background();

**public** String color() **default** "red";

}

**DocumentClass.java**

**package** com.annotation;

@AnnHtmlUL(border = "")

**public** **class** DocumentClass {

**private** String author;

@AnnHtmlLI(background = "blue", color = "black")

**public** String getDocumentName(){

**return** "Java Core";

}

@AnnHtmlLI(background = "yellow")

**public** String getDocumentVersion(){

**return** "1.0";

}

@AnnHtmlLI(background = "green")

**public** **void** setAuthor(String author){

**this**.author = author;

}

@AnnHtmlLI(background = "red", color = "black")

**public** String getAuthor(){

**return** author;

}

// Method nay khong chu thich bang Annotation nao

**public** **float** getPrice(){

**return** 100;

}

}

**HtmlGenerator.java**

**package** com.annotation;

**import** java.lang.reflect.Method;

**public** **class** HtmlGenerator {

**private** **static** **void** writeToFile(String filename, StringBuilder sb){

System.***out***.println(sb);

}

**public** **static** **void** main(String[] args) {

Class<?> clazz = DocumentClass.**class**;

// Kiem tra xem class nay co duoc gan boi Annotaion AnnHtmlUL khong

**boolean** isHtmlUL = clazz.isAnnotationPresent(AnnHtmlUL.**class**);

StringBuilder sb = **new** StringBuilder();

**if** (isHtmlUL) {

// Lay ra annotation AnnHtmlUL gan tren class nay

AnnHtmlUL annUL = clazz.getAnnotation(AnnHtmlUL.**class**);

sb.append("<H3>" + clazz.getName() + "</H3>");

sb.append("\n");

// Lay gia tri cua phan tu border cua AnnHtmlUL

String border = annUL.border();

sb.append("UL style='border:"+border+"'>");

sb.append("\n");

Method[] methods = clazz.getMethods();

**for** (Method method : methods) {

// Kiem tra xem method nay co duoc gan annotation AnnHtmlLI khong

**if** (method.isAnnotationPresent(AnnHtmlLI.**class**)) {

// Lay ra annotation do

AnnHtmlLI annLI = method.getAnnotation(AnnHtmlLI.**class**);

// Lay ra cac gia tri cac phan tu cua AnnHtmlLI.

String background = annLI.background();

String color = annLI.color();

sb.append("<LI style='margin:5px;padding:5px;background+"

+ background + ";color:"+color+"'>");

sb.append("\n");

sb.append(method.getName());

sb.append("\n");

sb.append("</LI>");

sb.append("\n");

}

}

sb.append("</UL>");

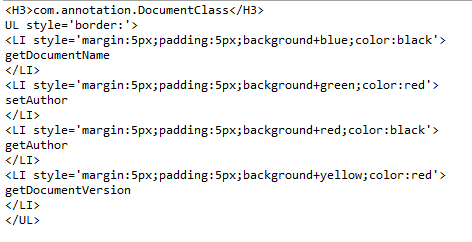
}

*writeToFile*(clazz.getSimpleName()+".html", sb);

}

}

***Result:***



Trường hợp ghi ra file html:

