# 注解

@JsonProperty(value="",index=)：指定Bean的属性转化成JSON字符串时的名称，如果没有指定，则默认使用定义的属性名。

@JsonIgnore：忽略Bean中的单个属性

@JsonIgnoreProperties({""})：以集合的方式忽略bean中的属性。

@JsonIgnoreType：作用于类，表示被注解该类型的属性将不会被序列化和反序列化。

@JsonInclude(Include.NON\_NULL)：空值（null）不参与序列化。

@JsonView：

**案例：**现在有一个用户类，包括用户名和密码，在我们序列化用户

类时，我们不希望密码被序列化，然而，在特殊情况下，我们有希

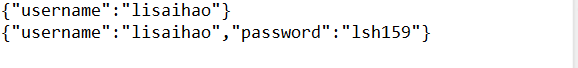
望序列化密码。

@JsonView使用步骤：1.使用视图声明多个视图。2.在值的get方

法上指明视图。

|  |
| --- |
| **public** **class** User  {  //不显示密码的视图  **public** **interface** WithoutPasswordView {};  //显示密码的视图，并继承于不显示密码的视图  **public** **interface** WithPasswordView **extends** WithoutPasswordView {};  **private** String username;  **private** String password;  **public** User()  {  }  **public** User(String username, String password)  {  **this**.username = username;  **this**.password = password;  }    @JsonView(WithoutPasswordView.**class**)  **public** String getUsername()  {  **return** **this**.username;  }  @JsonView(WithPasswordView.**class**)  **public** String getPassword()  {  **return** **this**.password;  }  } |

|  |
| --- |
| **public** **static** **void** main(String[] args) **throws** JsonGenerationException, JsonMappingException, IOException  {  ObjectMapper mapper = **new** ObjectMapper() ;  User user = **new** User("lisaihao","lsh159");  ByteArrayOutputStream out = **new** ByteArrayOutputStream();  //没有密码的JSON字符串  ObjectWriter writer = mapper.writerWithView(User.WithoutPasswordView.**class**);  writer.writeValue(out, user);  System.***out***.println(out.toString());    //有密码的JSON字符串  ObjectWriter writer2 = mapper.writerWithView(User.WithPasswordView.**class**);  writer2.writeValue(out, user);  System.***out***.println(out.toString());  } |



@JsonFormat：

|  |
| --- |
| **public** **class** Login  {  @JsonFormat(pattern="yy-MM-dd HH:mm:ss",timezone="GMT+8")  **private** Date currentLoginTime ;    **private** Date lastLoginTime ;  **public** Date getCurrentLoginTime() {  **return** currentLoginTime;  }  **public** **void** setCurrentLoginTime(Date currentLoginTime) {  **this**.currentLoginTime = currentLoginTime;  }  @JsonFormat(pattern="yy-MM-dd HH:mm:SS")  **public** Date getLastLoginTime() {  **return** lastLoginTime;  }  **public** **void** setLastLoginTime(Date lastLoginTime) {  **this**.lastLoginTime = lastLoginTime;  }  } |

|  |
| --- |
| **public** **static** **void** main(String[] args) **throws** JsonGenerationException, JsonMappingException, IOException  {  ObjectMapper mapper = **new** ObjectMapper() ;  Login login = **new** Login();  login.setCurrentLoginTime(**new** Date());  login.setLastLoginTime(**new** Date());  System.***out***.println(mapper.writeValueAsString(login));  } |



**注意：**如果直接使用@JsonFormat，则显示的结果会相差8个小时，

所以我们必须指定timezone=”GMT+8”

## 反序列化注解

@JacksonInject：作用于属性、方法、构造参数上，被用来反序

列化时标记已经被注入的属性。被该注解标注的属性不能通过JSON

字符串注入

|  |
| --- |
| **public** **class** Chuck  {  @JacksonInject("chuck\_name")  **private** String name ;  **private** **int** age ;  **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** **static** **void** main(String[] args) **throws** JsonGenerationException, JsonMappingException, IOException  {  ObjectMapper mapper = **new** ObjectMapper() ;  String chuckStr = "{\"age\":\"3\"}";  InjectableValues iv = **new** InjectableValues.Std().addValue("chuck\_name", "xiaoji") ;  Chuck chuck = mapper.reader(Chuck.**class**).with(iv).readValue(chuckStr) ;  System.***out***.println(chuck.getName());  System.***out***.println(chuck.getAge());  } |

@JsonAnySetter：

@JsonCreator：作用于方法，标注在构造方法或静态工厂方法上，

来创建实例。默认调用无参构造器，通常和@JsonProperty或者

@JacksonInject结合使用

|  |
| --- |
| **public** **class** Dog  {  **private** String name ;  **private** **int** age ;  @JsonCreator  **public** Dog(@JsonProperty("dog\_name")String name, @JsonProperty("dog\_age")**int** age) {  **this**.name = name;  **this**.age = age;  System.***out***.println("construct invoked");  }  **public** String getName() {  **return** name;  }  **public** **int** getAge() {  **return** age;  }  } |

**public** **class** DogMain

{

**public** **static** **void** main(String[] args) **throws** Exception

{

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

String dogStr = "{\"dog\_name\":\"taidi\",\"dog\_age\":\"10\"}" ;

Dog dog = mapper.readValue(dogStr, Dog.**class**);

System.***out***.println(dog.getName());

System.***out***.println(dog.getAge());

}

}



|  |
| --- |
| @JsonCreator  **public** **static** Dog getInstance(@JsonProperty("dog\_name")String name, @JsonProperty("dog\_age")**int** age)  {  Dog dog = **new** Dog();  dog.setAge(age);  dog.setName(name);  **return** dog;  } |

标注到静态工厂方法上：



@JsonSetter：

@JsonEnumDefaultValue：

## 序列化注解

@JsonAnyGetter

@JsonGetter：不建议使用

@JsonPropertyOrder

@JsonRawValue

@JsonValue

@JsonRootName

## 类型处理注解

@JsonSubTypes

@JsonTypeId

@JsonTypeInfo

@JsonTypeName

## 对象引用和标识注解

@JsonManagedReference，@JsonBackReference：

@JsonIdentityInfo：

## 元注解

@JacksonAnnotation：

@JacksonAnnotationInside：

ObjectMapper mapper = new ObjectMapper();

// 美化输出

mapper.enable(SerializationFeature.INDENT\_OUTPUT);

// 允许序列化空的POJO类，（否则会抛出异常）

// mapper.disable(SerializationFeature.FAIL\_ON\_EMPTY\_BEANS);

// 把java.util.Date, Calendar输出为数字（时间戳）

mapper.disable(SerializationFeature.WRITE\_DATES\_AS\_TIMESTAMPS);

// 在遇到未知属性的时候不抛出异常

mapper.disable(DeserializationFeature.FAIL\_ON\_UNKNOWN\_PROPERTIES);

// 强制JSON 空字符串("")转换为null对象值:

mapper.enable(DeserializationFeature.ACCEPT\_EMPTY\_STRING\_AS\_NULL\_OBJECT);

// 在JSON中允许C/C++ 样式的注释(非标准，默认禁用)

mapper.configure(JsonParser.Feature.ALLOW\_COMMENTS, true);

// 允许没有引号的字段名（非标准）

mapper.configure(JsonParser.Feature.ALLOW\_UNQUOTED\_FIELD\_NAMES, true);

// 允许单引号（非标准）

mapper.configure(JsonParser.Feature.ALLOW\_SINGLE\_QUOTES, true);

// 强制转义非ASCII字符

mapper.configure(JsonGenerator.Feature.ESCAPE\_NON\_ASCII, true);

// 将内容包裹为一个JSON属性，属性名由@JsonRootName注解指定

mapper.configure(SerializationFeature.WRAP\_ROOT\_VALUE, true);