**Flask\_RESTful返回标准化参数强化：**

**1.重命名属性和默认值：**

需求1：

有的时候你对外给出的属性名和模型内部的属性名不相同时，

可使用 attribute可以配置这种映射。

比如想要返回模型对象user.username的值，但是在返回给外面的时候，想以uname返回去。

需求2：

在返回某些字段的时候，有时候可能没有值，但想给一个值用以提示，

那么这时候可以在指定fields的时候使用default指定默认值

|  |
| --- |
| **from** flask **import** Flask,url\_for,render\_template  **from** flask\_restful **import** Api,Resource,reqparse,inputs,fields,marshal\_with  app = Flask(\_\_name\_\_)  api = Api(app)  #flask\_restful返回标准化参数强化  **class** User():  **def** \_\_init\_\_(self,username,age):  self.username=username  self.age=age  self.signature=**None**  **class** UserView(Resource):  resource\_fields={  #1.重命名属性  **'uname'**:fields.String(attribute=**'username'**),  **'age'**:fields.Integer,  #2.默认值  **'signature'**:fields.String(default=**'此人很懒，什么也没写'**)  }  @marshal\_with(resource\_fields)  **def** get(self):  user = User(**'莫莫'**,18)  **return** user  api.add\_resource(UserView,**'/user/'**)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  app.run(debug=**True**) |

**2.复杂的参数结构：**

大型的互联网项目中，返回的数据格式，是比较复杂的结构。

如：小米商城，

<https://www.mi.com/>

<https://a.huodong.mi.com/flashsale/getslideshow>

总结，无非就是key对应的value又是一个json，或者key对应的是一个列表，列表中的每项都是json

那么可以使用一些特殊的字段来实现。

如在一个字段中放置一个列表，那么可以使用fields.List，

如在一个字段下面又是一个字典，那么可以使用fields.Nested。

|  |
| --- |
| **from** flask **import** Flask,url\_for,render\_template  **from** flask\_restful **import** Api,Resource,reqparse,inputs,fields,marshal\_with  app = Flask(\_\_name\_\_)  api = Api(app)  #flask\_restful返回标准化参数强化  #复杂的参数结构  #实体间关系有 1：1 1:n n:n(转为2个1:n)  #新闻系统后台 用户 新闻 新闻标签  **class** User():  **def** \_\_init\_\_(self,id,uname,age):  self.id = id  self.uname = uname  self.age = age  **def** \_\_repr\_\_(self):  **return "<User:{id},{uname},{age}>"**.format(id=self.id, uname=self.uname,age=self.age)  **class** News():  **def** \_\_init\_\_(self,id,title,content):  self.id=id  self.title=title  self.content=content  #关系映射  self.author=**None**  self.tags=[]  **def** \_\_repr\_\_(self):  **return "<News:{id},{title},{content},{author},{tags}>"**\  .format(id=self.id, title=self.title,content=self.content,author=self.author,tags=self.tags)  **class** NewsTag():  **def** \_\_init\_\_(self,id,name):  self.id=id  self.name=name  **def** \_\_repr\_\_(self):  **return '<NewsTage:{id},{name}>'**.format(id =self.id , name=self.name)  **def** createData():  user = User(110,**'莫莫'**,30)  tag1 = NewsTag(200,**"要闻"**)  tag2 = NewsTag(210,**"娱乐"**)  news =News(300,**'吴京征服了世界上海拔最高的派出所'**,**'4月23日中午11点，吴京发了一条微博，配文“世界上海拔最高的派出所”，并@了另外一名演员张译。微博中有两张图片，第一张是吴京和张译两人坐在地上的合照，背后几个大字“中国边防”。第二张则是两人与派出所民警们的合照。 '**)  news.author = user #绑定新闻作者  news.tags.append(tag1) #绑定新闻标签1  news.tags.append(tag2) #绑定新闻标签2  print(news)  **return** news  **class** NewsView2(Resource):  resource\_fields={  **'id'**:fields.Integer,  **'title'**: fields.String,  **'content'**: fields.String,  #如在一个字段下面又是一个字典，那么可以使用fields.Nested({...})  **'author'**: fields.Nested({  **'id'**: fields.Integer,  **'uname'**: fields.String,  **'age'**:fields.Integer  }),  #如要在一个字段中放置一个列表，那么可以使用fields.List(fields.Nested({...}))  **'tags'**: fields.List(fields.Nested({  **'id'**: fields.Integer,  **'name'**: fields.String  }))  }  @marshal\_with(resource\_fields)  **def** get(self):  news = createData()  **return** news  api.add\_resource(NewsView2,**'/news2/'**)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  app.run(debug=**True**) |