PageResult转json

# PageResult的toJson方法

|  |
| --- |
| //PageResult toJson方法, 过滤某些参数, 和转换日期格式  **public** JSONObject toJson(String[] attributes) {  JSONArray array = **null**;  **if** (items != **null && attributes.length > 0**) {  array = JSONArray.*fromObject*(items, BeanAttrFilter.*getConfig*(attributes, **null**)); //把list转化为jsonArray, 并在转化的时候过滤掉某些参数  } **else** {  array = JSONArray.fromObject(items);  }  JSONObject json = **new** JSONObject();  json.put("total", total);  json.put("items", array);  **return** json;  } |

# BeanAttrFilter类

|  |
| --- |
| **import** java.util.Date;  **import** net.sf.json.JsonConfig;  **import** net.sf.json.util.CycleDetectionStrategy;  **import** com.ufida.health.util.json.processor.DateJsonValueProcessor;  **import** com.ufida.health.util.json.processor.HibernateJsonBeanProcessor;  **import** com.ufida.health.util.json.processor.HibernateJsonBeanProcessorMatcher;  /\*\*  \* **TODO** class purpose description  \*  \* **@author** zhangweidong  \* **@version** 3.0 Revise History:  \*  \*/  **public** **class** BeanAttrFilter {  /\*\*  \* 去掉不需要过滤的数据  \*  \* **@param** attributes  \* **@return**  \*/  **public** **static** JsonConfig getConfig(String[] attributes, String datePattern) {  JsonConfig config = **new** JsonConfig();  config.setIgnoreDefaultExcludes(**false**); //设置默认过滤数据  config.setCycleDetectionStrategy(CycleDetectionStrategy.*LENIENT*); //循环引用问题  config.registerJsonValueProcessor(Date.**class**, **new** DateJsonValueProcessor(datePattern));//设置日期解析器  config.registerJsonBeanProcessor(org.hibernate.proxy.HibernateProxy.**class**, **new** HibernateJsonBeanProcessor()); //设置hibernate延迟加载问题  config.setJsonBeanProcessorMatcher(**new** HibernateJsonBeanProcessorMatcher()); //  **if** (attributes != **null && attributes.length > 0**) {  config.setExcludes(attributes); //设置要过滤的参数  }  **return** config;  }  } |

# JSON日期解析器(DateJsonValueProcessor类)

|  |
| --- |
| **import** java.text.DateFormat;  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **import** net.sf.json.JsonConfig;  **import** net.sf.json.processors.JsonValueProcessor;  /\*\*  \* JSON日期解析器  \*  \* **@author** zhangweidong  \*  \*/  **public** **class** DateJsonValueProcessor **implements** JsonValueProcessor {  **public** **static** **final** String *DEFAULT\_DATE\_PATTERN* = "yyyy-MM-dd";  **private** DateFormat dateFormat;  **public** DateJsonValueProcessor() {  dateFormat = **new** SimpleDateFormat(*DEFAULT\_DATE\_PATTERN*);  }  /\*\*  \* 构造方法.  \*  \* **@param** datePattern  \* 日期格式  \*/  **public** DateJsonValueProcessor(String datePattern) {  **try** {  **if** (datePattern != **null**) {  dateFormat = **new** SimpleDateFormat(datePattern);  } **else** {  dateFormat = **new** SimpleDateFormat(*DEFAULT\_DATE\_PATTERN*);  }  } **catch** (Exception ex) {  dateFormat = **new** SimpleDateFormat(*DEFAULT\_DATE\_PATTERN*);  }  }  **public** Object processArrayValue(Object value, JsonConfig jsonConfig) {  **if** (value != **null**) {  **return** process(value);  }  **return** **null**;  }  **public** Object processObjectValue(String key, Object value, JsonConfig jsonConfig) {  **if** (value != **null**) {  **return** process(value);  }  **return** **null**;  }  **private** Object process(Object value) {  **if** (value != **null**) {  **return** dateFormat.format((Date) value);  }  **return** **null**;  }  } |