# OGNL

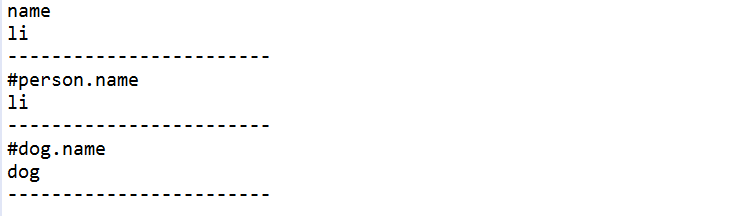
OGNL（Object Graph Navigation Language）：对象图导航语言。

OgnlContext（上下文对象）：存在唯一的叫做根的对象，可以通过程序设定上下文当中的哪个对象作为根对象。

所需要的jar包：



|  |
| --- |
| **import** ognl.Ognl;  **import** ognl.OgnlContext;  **import** ognl.OgnlException;  **public** **class** OgnlDemo1  {  **public** **static** **void** main(String[] args) **throws** OgnlException  {  Person person = **new** Person();  person.setName("li");  Dog dog = **new** Dog();  dog.setName("dog");  //创建上下文对象  OgnlContext context = **new** OgnlContext();  //在context添加所需要管理的对象  context.put("person",person);  context.put("dog", dog);    //person为根  context.setRoot(person);    Object object = Ognl.*parseExpression*("name") ;  Object object1 = Ognl.*getValue*(object, context,context.getRoot()) ;  System.***out***.println(object);  System.***out***.println(object1);  System.***out***.println("------------------------");    Object object2 = Ognl.*parseExpression*("#person.name") ;  Object object13 = Ognl.*getValue*(object2, context,context.getRoot()) ;  System.***out***.println(object2);  System.***out***.println(object13);  System.***out***.println("------------------------");    Object object4 = Ognl.*parseExpression*("#dog.name") ;  Object object15 = Ognl.*getValue*(object4, context,context.getRoot()) ;  System.***out***.println(object4);  System.***out***.println(object15);  System.***out***.println("------------------------");  }  } |



**调用方法：**

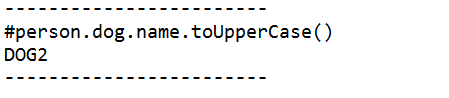
Object object8 = Ognl.*parseExpression*("#person.dog.name.toUpperCase()") ;

Object object19 = Ognl.*getValue*(object8, context,context.getRoot()) ;

System.***out***.println(object8);

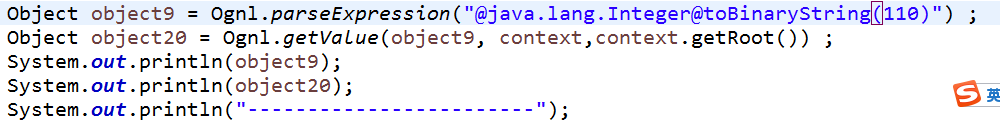
System.***out***.println(object19);

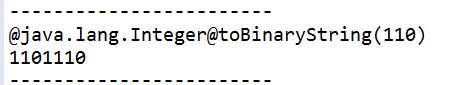
System.***out***.println("------------------------");



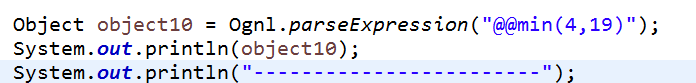
**OGNL调用静态方法：**

@packagename.className@methodName(param1,param2,…..)



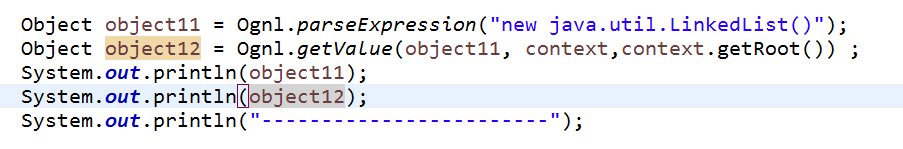


**特殊的：对于OGNL来说java.lang.Math使其默认的类，无需指定类名**



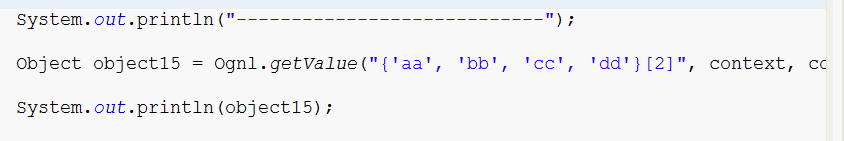


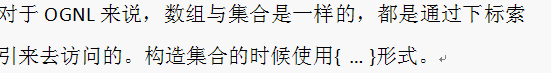
**生成对象：**



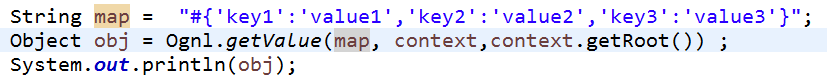


**处理集合和数组：**



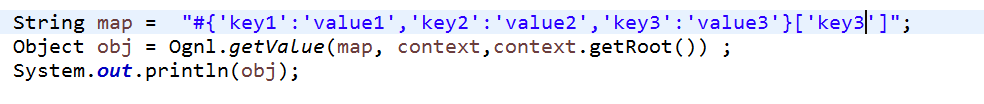


**处理Map:**





**获取Map值：**

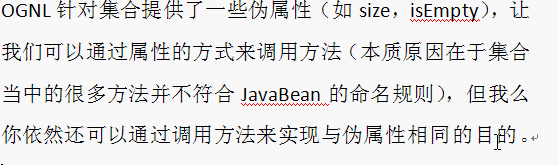




**过滤：操作集合对象，返回一个集合的子集和。**







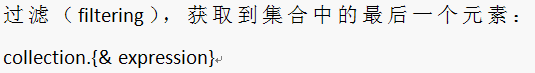
**获取集合中的第一个元素：**

Collection.{^ expression}





**获取集合中的最后一个元素：**



**投影：collection.{expression**