

Functional Programming in Java

胡长城 （银狐999）

Workflow Preacher

Email: james-fly@vip.sina.com

Blog : <http://blog.csdn.net/james999>

Web : <http://www.javafox.org>

FP in Java

抛砖引？

玉
金
砖头
碎石

Functor, 俺也刚学.....

- **命令式编成** (Imperative programming)

a methodology that describes computation in terms of program states

- **函数式编成** (Functional Programming)

a style of programming that emphasizes the evaluation of expressions

Characteristic

- 支持闭包 (closure) (Functor)
- 支持高阶函数 (higher order function)
- 支持懒惰计算 (lazy evaluation) : 不进行非必要计算的延迟称为lazy evaluation (懒惰计算法)
- 使用递归作为控制流程的机制
- 加强了引用透明性
- 没有副作用

Functor

A functor is a function that can be manipulated as an object, or an object representing a single, generic function.

Functor Type

predicate

return a boolean value

function

return an Object value

procedure

don't return anything

Several Implements

Apache Commons Functor

<http://jakarta.apache.org/commons/sandbox/functor>

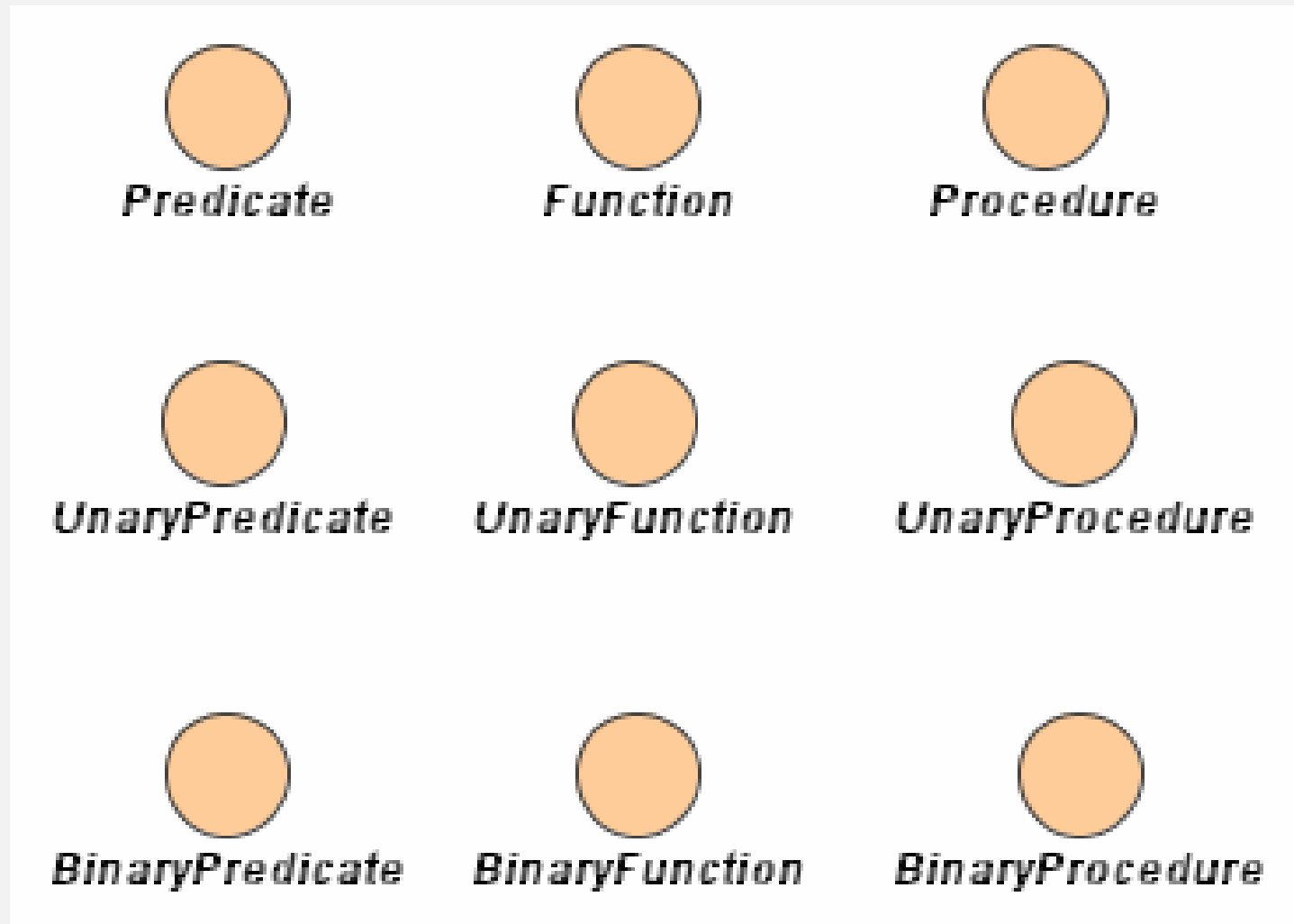
Generic Algorithms for Java (JGA)

<http://jga.sourceforge.net>

Generic Libraries for Java (JGL)

<http://www.recursionsw.com/jgl.htm>

Apache Common Functor



Unary Functor

```
public interface UnaryPredicate {  
    boolean test(Object obj);  
}
```

```
public interface UnaryFunction {  
    Object evaluate(Object obj);  
}
```

```
public interface UnaryProcedure {  
    void run(Object obj);  
}
```

Simple Sample

```
item1.getPrice() > item2.getPrice());
```

可以用Comparable去实现
在list的时候可以用Comparator实现
还可以用Functor去实现

Simple Sample

```
public class PriceComparator implements Comparator
{
    public int compare (Object o1, Object o2)
    {
        return (((SETLItem)o1).getPrice()-((SETLItem)o2).getPrice());
    }
}
```

Simple Sample

```
Comparator pc = new PriceComparator();  
BinaryPredicate bp = new IsGreaterThanOrEqualTo(pc);  
SETLItem item1 = new SETLItem();  
item1.setPrice(100);  
SETLItem item2 = new SETLItem();  
item2.setPrice(99);  
bp.test(item1, item2)
```

Generic and Functor

Apache Commons Functor 项目是一个正在开发中的函数式编程库，但目前看来并不是类型安全的；J2SE 5.0提供了有限的generic能力，除了用于Collection之外，类型安全的functor也是其用武之地，已有一个开源项目Generic Algorithms for Java (JGA) 开始了这方面的工作

My colleague

Chelsea ,

<http://blog.csdn.net/chelsea>

Unary Functor

```
public interface UnaryFunction<R, P> {  
    R evaluate(P obj);  
}  
  
public interface UnaryPredicate<T> {  
    boolean test(T obj);  
}  
  
public interface UnaryProcedure<T> {  
    void run(T obj);  
}
```

Binary Functor

```
public interface BinaryFunction<R, T, S> {  
    R evaluate(T left, S right);  
}  
  
public interface BinaryPredicate<T, S> {  
    boolean test(T left, S right);  
}  
  
public interface BinaryProcedure<T, S> {  
    void run(T left, S right);  
}
```

UnaryPredicate Sample

```
public class SETLItemPredicate
    implements UnaryPredicate<SETLItem> {
    private SETLItem item;
    public SETLItemPredicate(SETLItem item){
        this.item = item;
    }
    public boolean test(SETLItem obj) {
        return (item.getPrice() - obj.getPrice())<0;
    }
}
```

计算 是否大于 item

UnaryPredicate Sample : Select

定义了一个选择算法，放在类 Algorithms 中：

```
public static <T> List<T> select(
    Collection<T> source,
    UnaryPredicate<T> selector) {
    List<T> result = new ArrayList<T>();
    for(T item : source){
        if( selector.test( item ) ){
            result.add(item);
        }
    }
    return result;
}
```

UnaryPredicate Sample

```
item1.setPrice(100);  
item2.setPrice(99);  
item3.setPrice(101);
```

来看看测试例子：

```
List<SETLItem> tl = new ArrayList<SETLItem>();  
tl.add(item1); tl.add(item2); tl.add(item3);
```

```
for (SETLItem item: Algorithms.select(tl,new SETLItemPredicate(item1)) )  
{  
    System.out.print("=== "+item.getPrice());  
}
```

BinaryPredicate Sample

```
public class SETLItem2Predicate
    implements BinaryPredicate<SETLItem,SETLItem> {
    public boolean test(SETLItem T,SETLItem S) {
        return (T.getPrice() - S.getPrice())>0;
    }
}
```

BinaryPredicate Sample : Select

```
public static <T> List<T> select(
    Collection<T> source, BinaryPredicate<T,T>
    selector, T t){
    List<T> result = new ArrayList<T>();
    for(T item : source){
        if( selector.test( item , t ) ){
            result.add(item);
        }
    }
    return result;
}
```

UnaryPredicate Sample

```
item1.setPrice(100);  
item2.setPrice(99);  
item3.setPrice(101);
```

来看看测试例子：

```
List<SETLItem> tl = new ArrayList<SETLItem>();  
tl.add(item1); tl.add(item2); tl.add(item3);
```

```
for (SETLItem item: select( tl , new SETLItem2Predicate() , item1 ) ) {  
    System.out.print("=== "+item.getPrice());  
}
```

FP in Java Development

- Strong in Algorithm and Logic
And, Translator, Filter, Select
- Support Rule Application
Rule Engine
- Supportde by Generic Programming
Safety

有兴趣，大家应该去学学，特别是结合 Generic

wfchina,致力workflow在国内的推广和服务



胡长城（银狐999）

手机: 13911168779（北京）

Email: james-fly@vip.sina.com

Blog : <http://blog.csdn.net/james999>

Web : <http://www.javafox.org>