01.1주차1-3클로저사용하기

클로저

Table of Contents

클로저란?

변수처럼 사용할 수 있는 익명의 코드 블록이라고 한다.

클로저 정의(선언) 및 호출 방법

기본적인 클로저 사용

클로저를 인자로 받는 기본 메서드 형태에 중괄호으로 둘러싸인 문장을 만드는 가장많이 활용할수 있는 문장이다.
인자가 있는 경우 먼저 인자들이 나오고 화살표(->) 뒤에 문장들이 나오기도 한다. 그리고 인자가 하나일경우에는 생략하고 it 으로 사용 가능하다.
클로저를 인자로 받는 기본 메서드 API

```
// 1. 반복형태의 클로저 사용
log = ''
(1..10).each{ counter -> log += counter }
// 1-1 클로저 결과값 확인
assert log == '12345678910'

log = ''
(1..10).each{ log += it }
assert log == '12345678910'

log = ''
def closure = { log += it  }
(1..10).each closure
assert log == '12345678910'
```

할당 형태의 클로저 사용

클로저를 할당 형태로 사용 가능한 형태로 만들어서 사용가능하다.
이때 클로저를 사용시 마치 메소드처럼 () 을 이용해 인자값을 넣어주거나, call() 메소드를 이용해서 클로저를 호출 가능하다.

```
// 2.할당 형태의 클로저 사용
def c = { num -> return num + 1 }
// 2-1. 기본적인 클로저 호출 방법
assert 3 == c(2)
assert 3 == c.call(2)
```

메서드를 클로저 형태로 사용

기존의 메서드를 이용해서 클로저 형태로 사용이 가능한데 이때 "reference.&" 연산자를 사용하면 가능하다.
여기에서 reference 란 클래스 혹은 this 를 말한다.

```groovy
// 3. 메서드를 클로저 형태로 "reference.&" 연산자를 사용하면 가능하다.
// 3-1 일단 테스트를 위한 클래스와 메소드를 하나 생성후 호출
class MethodClosureSample {
    int sum(int num1, int num2){
        return num1 + num2
    }
}
MethodClosureSample sample = new MethodClosureSample();
Closure closure = sample.∑
assert 3 == closure(2,1)
// 3-2 자체 메소드를 생성후 호출
def sumNum(num1, num2){
    return num1 + num2
}
def closure2 =  this.&sumNum
assert 3 == closure2(1,2)
```

기타 클로저의 메서드 활용

클로저에는 자체 내장되어있는 기능이 다양하게 있는데 클로저 api 에서 확인 가능하다. 여기에서는 일부 유용하게 쓸수있는 2가지 기능을 소개한다.
인자 수에 반응하는 기능을 가진 getParameterTypes

```groovy
// 인자 수에 반응하는 기능을 가진 getParameterTypes
def caller(Closure closure){
    closure.getParameterTypes().size()
}
assert caller{ one -> } == 1
assert caller{ one, two -> } == 2
```

클로저 인자수를 고정시키는 커리(curry) 사용하기
커리(curry)란 인자를 여러 개 받는 함수에서 인자 몇 개를 고정시킨 또 다른 함수를 만들어 낸다는 것이다.
클로저의 curry 메서드를 이용해 한 개 혹은 그 이상의 인자가 특정 값으로 거정된 복사본 클로저를 만들 수 있다.인자를 고정시킬때는 왼쪽부터 고정된다.

```groovy
// 클로저 인자수를 고정시키는 커리(curry) 사용하기
def addr = { x, y, t -> return x+y+t }
def addOne = addr.curry(1).curry(2)
assert addOne(5) == 8
```

기타 클로저를 활용한 예제

직원 클래스(이름, 급여, 직책-매니져 여부) 를 가지고 있고 4명의 직원이 있다.(다음은 마티 파울러의 그루비 클로저 예제입니다.)

```groovy
class Employee {
    def name, salary
    boolean manager
    String toString() { return name }
}

def emps = [new Employee(name:'Guillaume', manager:true, salary:200),
    new Employee(name:'Graeme', manager:true, salary:200),
    new Employee(name:'Dierk', manager:false, salary:151),
    new Employee(name:'Bernd', manager:false, salary:50)]
```

1. 직책인 매니저 인지 확인
2. 급여가 150이상 인 직원을 확인
3. 급여를 특정 금액을 입력해서 해당 금액 보다 큰 직원을 확인

위 내용을 클로저를 활용해서 작성해보자

1. 매니져 여부

```groovy
def managers(emps) {
    emps.findAll { e -> e.isManager() }
}
assert emps[0..1] == managers(emps) // [Guillaume, Graeme]

list = []
emps.each { if( it.manager ) list += it  }
assert emps[0..1] == list
```

2. 급여가 150 이상인 직원 확인

```groovy
def highPaid(emps) {
```

```
    threshold = 150
    emps.findAll { e -> e.salary > threshold }
}
assert emps[0..2] == highPaid(emps) // [Guillaume, Graeme, Dierk]

def methodHighPaid(emps){
    return emps.salary > 150 ? true : false
}
def closulre = this.&methodHighPaid
assert emps[0..2] ==  emps.findAll(closulre)
```

3. 급여를 직접 입력해서 그 금액보다 큰직원을 확인

```
def paidMore(amount) {
    { e -> e.salary > amount}
}
def highPaid = paidMore(150)

assert highPaid instanceof Closure
assert highPaid(emps[0]) // true
assert emps[0..2] == emps.findAll(highPaid)
```

링크 목록

- 클로저를 인자로 받는 기본 메서드 API - http://groovy.codehaus.org/groovy-jdk/java/lang/Object.html
- 클로저 api - http://groovy.codehaus.org/gapi/groovy/lang/Closure.html
- 다음은 마틴 파울러의 그루비 클로저 예제입니다. - https://www.google.com/url?q=http://groovy.codehaus.org/Martin%2BFowler%27s%2Bclosure%2Bexamples%2Bin%2BGroovy&sa=U&ei=qtP-UtKSD4mCogS_qoCQBw&ved=0CAkQFjAE&client=internal-uds-cse&usg=AFQjCNGDwL4DIyvu_tgMoaVRtO3nDnxjAQ
