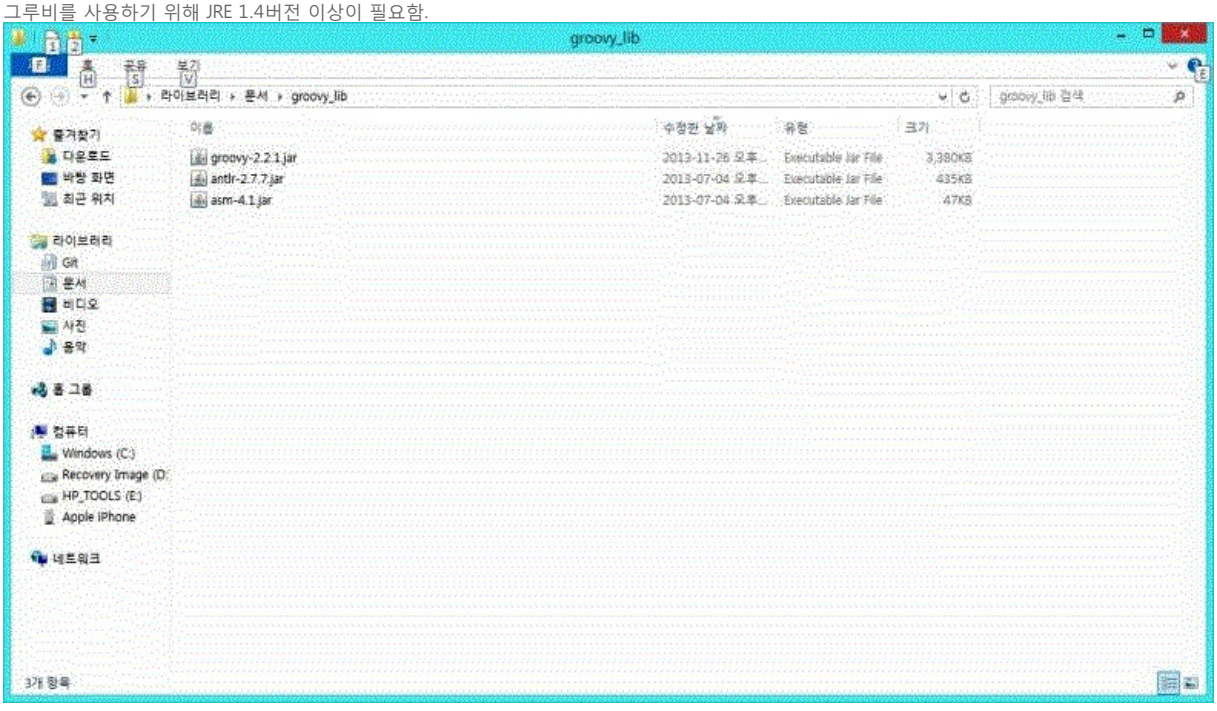
08.3주차 그루비 통합

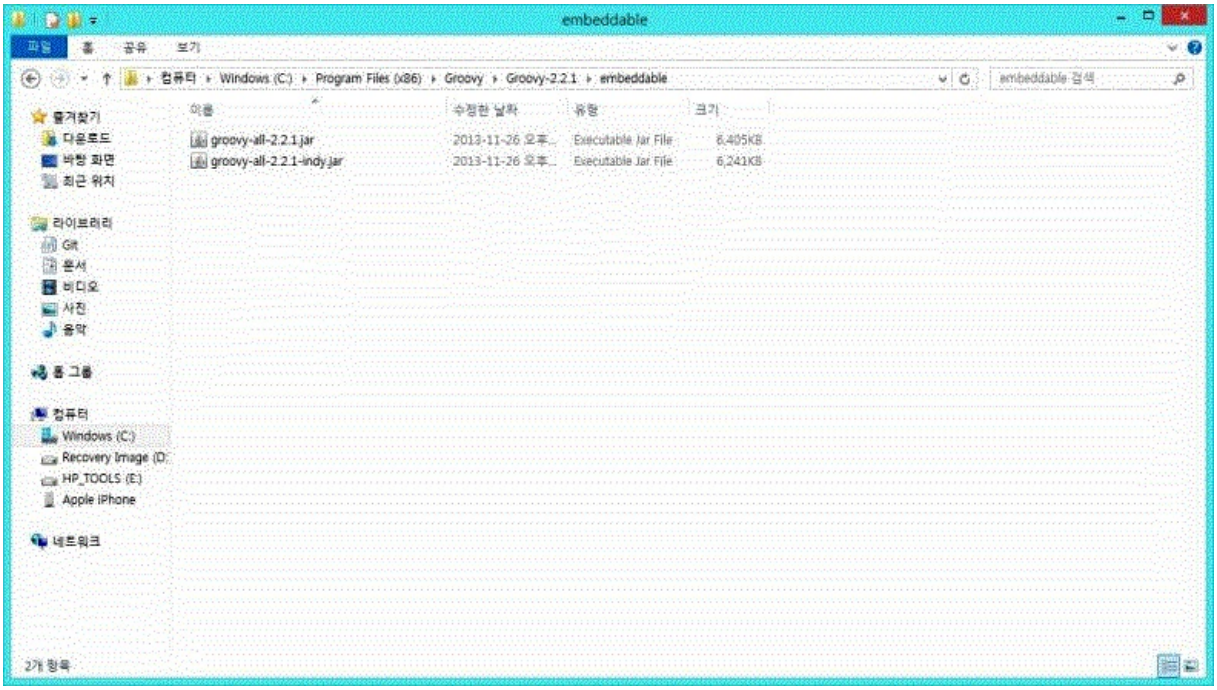
그루비 통합

Table of Contents

의존관계 설정

프로젝트에서 그루비 사용시 그루비를 사용할수 있게 설정하기위해 필요한 라이브러리.
기존 라이브러리와 충돌이 발생하기도 함.
특히 하이버네이트나 스피링의 경우 ASM(http://linuxism.tistory.com/448)을 프록시 생성기로 사용하기 때문에 같은 버전의 ASM을 사용하는 하이버네이트나 스프링이 아니면 그루비를 사용할 수 없음.
통합용 라이브러리인 groovy-all-2.2.1.jar을 이용하면 다른 라이브러리와 충돌을 일으키자 않고 그루비 사용가능함.
그루비를 사용하기 위해 JRE 1.4버전 이상이 필요함.





GroovyShell

표현식이나 스크립트를 해석하는 도구
자바스크립트의 eval과 기능이 비슷함.

```
def script = '''
def x = 20
def y = 10
return x * y
'''

def shell = new GroovyShell()
def result = shell.evaluate(script)

// 그루비 스크립트로 구현시에는 그루시 스크립트의 상위 클래스가 Script이기 때문에 아래와 같이도 작성 가능
// http://groovy.codehaus.org/api/groovy/lang/Script.html
// def result = evaluate(script)

println result
```

http://groovy.codehaus.org/api/groovy/lang/GroovyShell.html

GroovyShell생성의 반복적 코드를 생략할시 groovy.util.Eval을 사용.

http://groovy.codehaus.org/api/groovy/util/Eval.html

```
println Eval.me("'aaaa'")
println Eval.x(1,"x")
println Eval.xy(1,2,"x*y")
println Eval.xyz(1,2,3,"x*y+z")
```

Binding

http://groovy.codehaus.org/api/groovy/lang/Binding.html
스크립트로 작성시 재사용을 할 수 없다는 단점을 보완하기 위해 사용.
변수를 설정할 수 있음.

```
def binding = new Binding()
binding.x = 20
binding.y = 10

def shell = new GroovyShell(binding)
def ex = "x * y"
println shell.evaluate(ex)
```

```
binding.setVariable("x", 30)
println shell.evaluate(ex)

println binding.getVariable("v")
```

스크립트 내부에서 Binding에 변수 추가가 가능함.
하지만 지역변수는 접근을할 수 없음.

```
def binding = new Binding(x:20,y:10)
def shell = new GroovyShell(binding)
shell.evaluate('''
sum = x + y
min = x - y
''')

println binding.getVariable("sum")
println binding.min
```

```
def binding = new Binding(x:20,y:10)
def shell = new GroovyShell(binding)
shell.evaluate('''
def sum = x + y
min = x - y
''')

println binding.getVariable("sum")
println binding.min
```

Binding는 스크립트의 마지막 문장까지 해석한 후 만들어진 값을 가지고 옴.

동적 클래스 생성하기

```
def shell = new GroovyShell()
def cls = shell.evaluate('''
    class TestCls {
        def testMethod(){
            "test"
        }
    }

    return TestCls
''')

println cls.name
def instance = cls.newInstance()
println instance.testMethod()
```

스크립트 파싱

GroovyShell 의 parse를 이용하여 스크립트의 인스턴스를 만든다.
이를 이용하여 다시 컴파일 하지 않고 스크립트를 재사용 가능함.
evaluate 메서드와 중복정의 되어 있음.
parse는 코드를 실행하지 않고 Script 클래스의 인스턴스만 생성함.

```
def sum = "x + y + z"
def shell = new GroovyShell()
def script = shell.parse(sum)

script.binding.x = 10
script.y = 20
script.z = 30

println script.run()

script.binding = new Binding(x:40,y:50,z:60)
println script.run()
```

```
//GroovyShell run
class GTest2 {

    static main(args) {
        def seventyfive = new GroovyShell().run("args[0] + args[1]", "StringSummerScript", ['7', '5'])
```

```
        println seventyfive

    }
}
```

```
class Car {
    String state
    Long distance = 0
}



import groovy.lang.Script

abstract class CarScript extends Script {

    def start() {
        this.binding.car.state = 'started'
    }

    def stop() {
        this.binding.car.state = 'stopped'
    }

    def drive(distance) {
        this.binding.car.distance += distance
    }

}


def compilerConfiguration = new CompilerConfiguration()
compilerConfiguration.setScriptBaseClass(CarScript.class.name)
//compilerConfiguration.scriptBaseClass = CarScript.class.name

def car = new Car()
def binding = new Binding(car: car)

def shell = new GroovyShell(this.class.classLoader, binding, compilerConfiguration)

def carDsl = '''
start()
drive 20
stop()
'''

shell.evaluate carDsl

println car.state
println car.distance
```

GroovyScriptEngine

GroovyShell은 독립적인 하나의 스크립트를 다룰때는 좋지만 여러개의 스크립트가 필요한 경우 적용하기 쉽지 않음.
GroovyScriptEngine은 스크립트가 수정되면 다시 로딩하는 기능이 있어서 실행중 업무로직을 변경하게 해줌.

http://groovy.codehaus.org/api/groovy/util/GroovyScriptEngine.html

```
HelloGroovy2.groovy
output = 'hello' + input


String[] roots = ['./src/coma']
def engine = new GroovyScriptEngine(roots)

def binding = [input:'world'] as Binding

engine.run('HelloGroovy2.groovy', binding)

println binding.getVariable('output')
```

```
class HelloGroovy {
    def hello() {
        'Hello Groovy'
    }
}

String[] roots = ['./src/coma']
engine = new GroovyScriptEngine(roots)

helloClass = engine.loadScriptByName('HelloGroovy.groovy')
hello = helloClass.newInstance()

println hello.hello()
```

GroovyClassLoader

그루비에서 구현한 클래스로더로 그루비 클래스와 스크립트를 일반적인 클래스로 정의하고 파싱.
GroovyClassLoader를 이용하여 그루비나 자바에서 클래스를 사용할 수 있음.
요청받은 클래스를 컴파일하고,의존관례에 있는 클래스도 컴파일함.

http://groovy.codehaus.org/api/groovy/lang/GroovyClassLoader.html

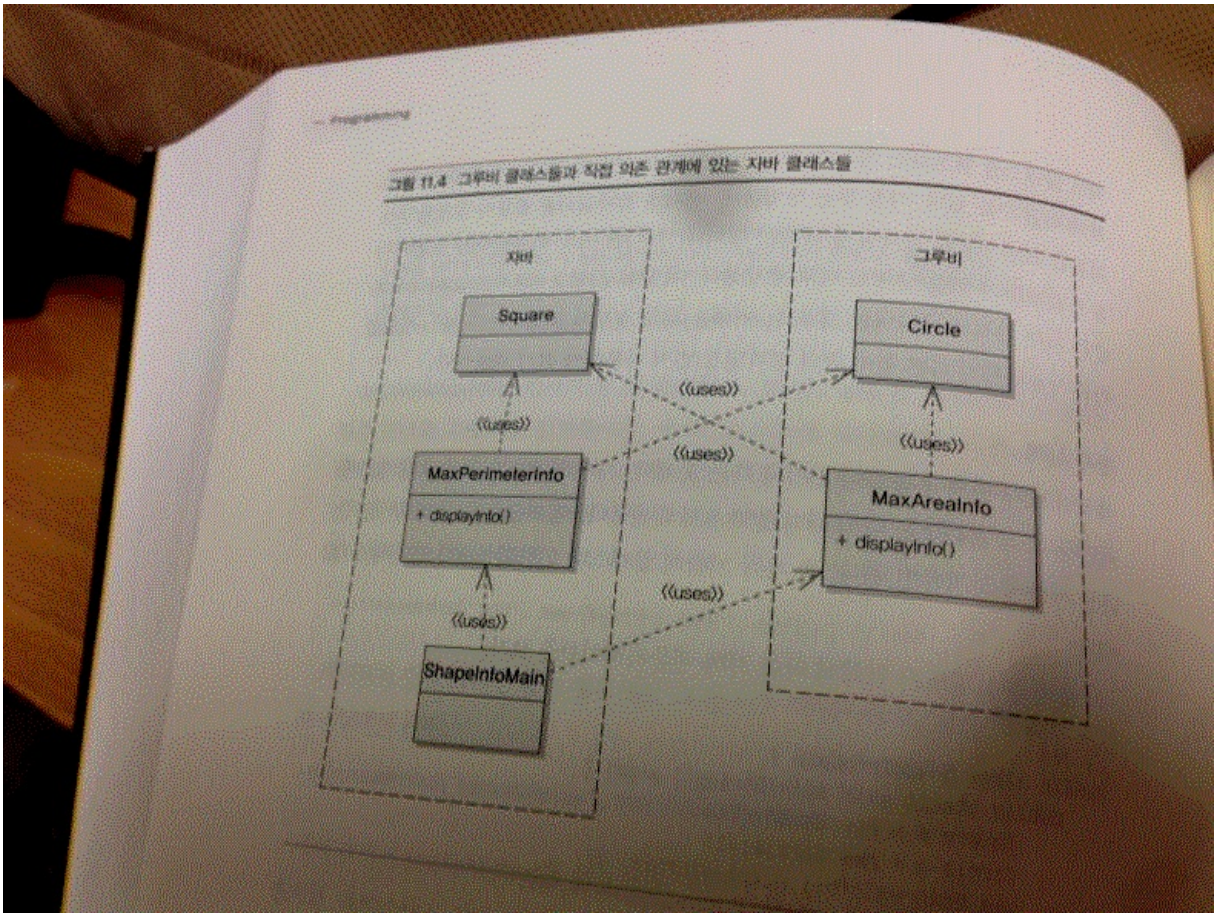
```
def gcl = new GroovyClassLoader()
Class helloClass = gcl.parseClass(new File("./src/coma/HelloGroovy.groovy"))
println helloClass.newInstance().hello()
```

duck type
어떤 객체에 특정 메서드들이 있고 다른 객체에도 동일한 메서드들이 있다면 상속구조와 상관없이 두 객체를 바꿔쓸 수 있음
http://ko.wikipedia.org/wiki/%EB%8D%95_%ED%83%80%EC%9D%B4%ED%95%91

상호의존문제

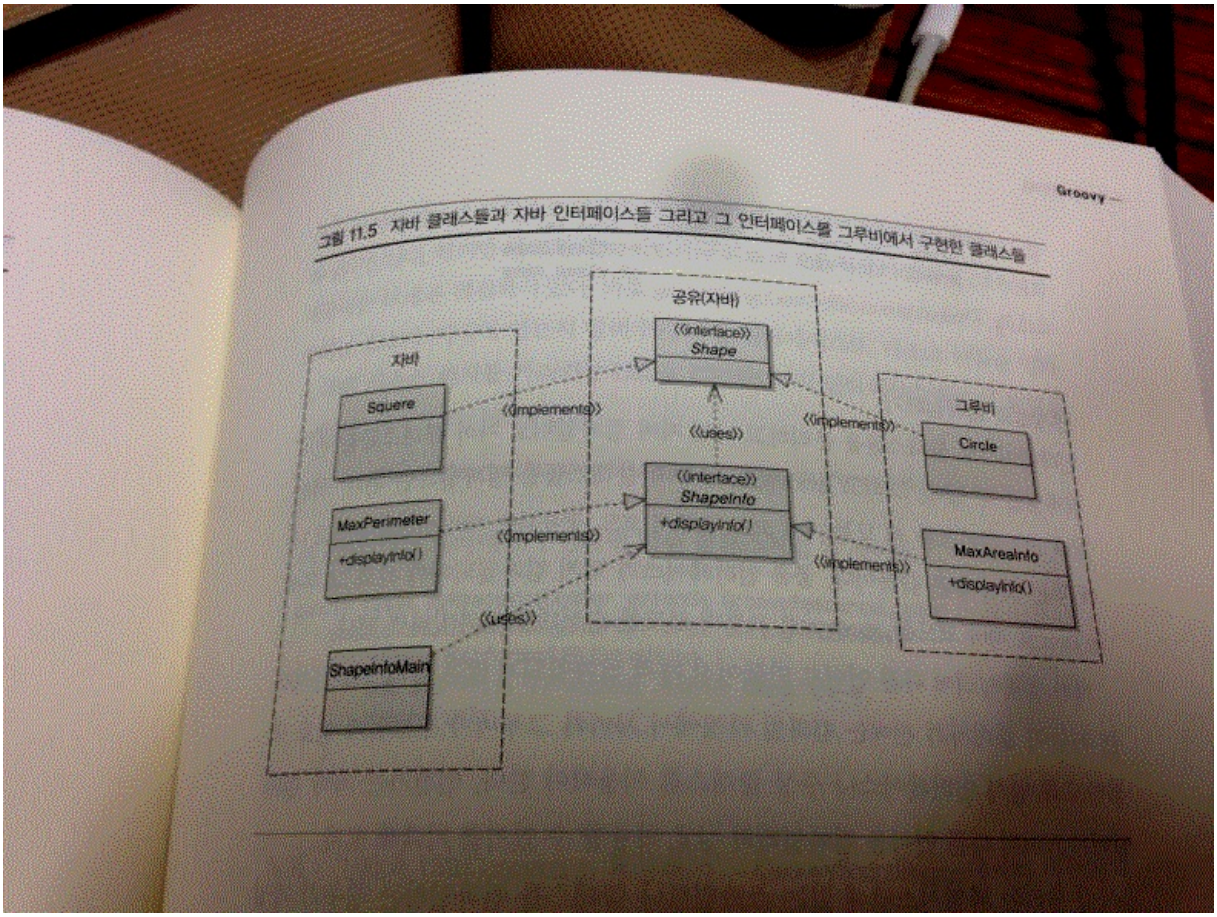
자바파일을 컴파일하는데 이 자바파일이 그루비를 참조하고 그 그루비소스파일에서 자바클래스를 참조하는 경우 의존관계가 복잡하게 꼬이는 상황이 발생함.
가장 좋은 방법은 두 언어의 컴파일러를 번갈아 호출하면서 필요한 클래스들을 컴파일 하는 방법이 좋음.
하지만 어느 컴파일러를 호출해야 할지 판단하기 힘든 상황이 발생하기도 하고 순환의존관계가 나타나는 경우.

```
public class ShapeInfoMain {
    public static void main(String[] args){
        Square square = new Square(7);
        Circle circle new Circle(4);
        new MaxAreaInfo().displayInfo(square,circle);
        new MaxPerimeterInfo().displayInfo(square,circle);
    }
}
```



가장 쉽게 해결하는 방법은 자바쪽에 인터페이스를 만들어 이를 구현하는 방법으로 분리를 하는 방법.

인터페이스로 클래스들을 분리하고 자바 컴파일러로 자바 클래스들을 먼저 컴파일한 후 그루비 클래스를 컴파일한다.



스프링과 통합

그루비 빈 끼워넣기

```
<bean id="someBean" class="com.SomeClass">
    <property name="someProperty" ref="someOtherBean"/>
</bean>

def ctx = new ClassPathXmlApplicationContext('bean.xml')
def calc = ctx.getBean('someBean')

<lang:groovy id="someMain" refresh-check-delay="5000" script-source="classpath:spring/SomeMain.groovy">
    <lang:property name="someOne" value="1" />
    <lang:property name="someTwo" value="2" />
</lang:groovy>
```

lang:groovy를 이용하여 자동으로 그루비 팩토리가 동작.
refresh-check-delay 속성을 이용한 자동 갱신기능.

```
<lang:groovy id="someMain" refresh-check-delay="5000" script-source="classpath:spring/SomeMain.groovy">
    <lang:property name="someOne" value="1" />
    <lang:property name="someTwo" value="2" />
</lang:groovy>
```

인라인 스크립트를 이용한 그루비 클래스 사용가능.
스크립트의 내용이 하드코딩되기 때문에 자동갱신기능은 사용할 수 없음.
스크립트에 "<" 기호가 있으면 xml파서가 새로운 테스로 인식하기 때문에 CDATA로 해결해야 함.

```
<lang:groovy id="sorter">
<lang:inline-script>
    package spring
    class CountrySorter implements Sorter {
        String order
```

```
        List sort(Country[] items) {
            List result = items.toList().sort{ p1, p2 -> p1.population <=> p2.population }
            if (order == "reverse") return result.reverse() else return result
        }
    }
</lang:inline-script>
    <lang:property name="order" value="forward" />
</lang:groovy>
```

JSR-223

자바 플랫폼을 위한 스크립팅.
스크립트 엔진을 등록하고 사용할 수 있게 해준다.
키/값 쌍을 스크립트에 전달할 네임스페이스와 실행환경도 제공한다.

```
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;

public class CalcMain {

    public static void main(String[] args) throws ScriptException {
        ScriptEngineManager factory = new ScriptEngineManager();
        ScriptEngine engine = factory.getEngineByName("groovy");

        // basic example
        System.out.println(engine.eval("(1..10).sum()"));

        // example showing scripting variables
        engine.put("first", "HELLO");
        engine.put("second", "world");
        System.out.println(engine.eval("first.toLowerCase() + second.toUpperCase()"));

    }

}
```

링크 목록

- http://groovy.codehaus.org/api/groovy/lang/GroovyShell.html - http://groovy.codehaus.org/api/groovy/lang/GroovyShell.html
- http://groovy.codehaus.org/api/groovy/util/Eval.html - http://groovy.codehaus.org/api/groovy/util/Eval.html
- http://groovy.codehaus.org/api/groovy/lang/Binding.html - http://groovy.codehaus.org/api/groovy/lang/Binding.html
- http://groovy.codehaus.org/api/groovy/util/GroovyScriptEngine.html - http://groovy.codehaus.org/api/groovy/util/GroovyScriptEngine.html
- http://groovy.codehaus.org/api/groovy/lang/GroovyClassLoader.html - http://groovy.codehaus.org/api/groovy/lang/GroovyClassLoader.html
- http://ko.wikipedia.org/wiki/%EB%8D%95_%ED%83%80%EC%9D%B4%ED%95%91 - http://ko.wikipedia.org/wiki/%EB%8D%95_%ED%83%80%EC%9D%B4%ED%95%91
