

第一次Maven使用

使用准备

下载并配置好maven

首先进行一个文件的下载

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/sisu/sisu-inject-bean/1.4.2/sisu-inject-bean-1.4.2.pom (5.5 kB at 6.3 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/sisu/sisu-guice/2.1.7/sisu-guice-2.1.7.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/sisu/sisu-guice/2.1.7/sisu-guice-2.1.7.pom (11 kB at 10 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-model-builder/3.0/maven-model-builder-3.0.pom (2.2 kB at 4.2 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-aether-provider/3.0/maven-aether-provider-3.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-aether-provider/3.0/maven-aether-provider-3.0.pom (2.5 kB at 5.1 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/aether/aether-api/1.7/aether-api-1.7.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/aether/aether-api/1.7/aether-api-1.7.pom (1.7 kB at 3.8 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/aether/aether-parent/1.7/aether-parent-1.7.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/aether/aether-parent/1.7/aether-parent-1.7.pom (7.7 kB at 16 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/aether/aether-util/1.7/aether-util-1.7.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/aether/aether-util/1.7/aether-util-1.7.pom (2.1 kB at 3.9 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/aether/aether-impl/1.7/aether-impl-1.7.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/aether/aether-impl/1.7/aether-impl-1.7.pom (3.7 kB at 3.1 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/aether/aether-spi/1.7/aether-spi-1.7.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/aether/aether-spi/1.7/aether-spi-1.7.pom (1.7 kB at 2.3 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-invoker/3.0.1/maven-invoker-3.0.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-invoker/3.0.1/maven-invoker-3.0.1.pom (4.9 kB at 8.3 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-components/31/maven-shared-components-31.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-components/31/maven-shared-components-31.pom (5.1 kB at 6.9 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/31/maven-parent-31.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/31/maven-parent-31.pom (43 kB at 60 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/19/apache-19.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/19/apache-19.pom (15 kB at 21 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.2.1/maven-shared-utils-3.2.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.2.1/maven-shared-utils-3.2.1.pom (5.6 kB at 7.3 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-components/30/maven-shared-components-30.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-components/30/maven-shared-components-30.pom (4.6 kB at 9.5 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/30/maven-parent-30.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/30/maven-parent-30.pom (41 kB at 82 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/18/apache-18.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/18/apache-18.pom (16 kB at 33 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons-io/commons-io/2.2/commons-io-2.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons-io/commons-io/2.2/commons-io-2.2.pom (11 kB at 23 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-parent/24/commons-parent-24.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-parent/24/commons-parent-24.pom (47 kB at 98 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/9/apache-9.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/9/apache-9.pom (15 kB at 33 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-artifact-transfer/0.11.0/maven-artifact-transfer-0.11.0.pom
```

进行一个对pom.xml的观察

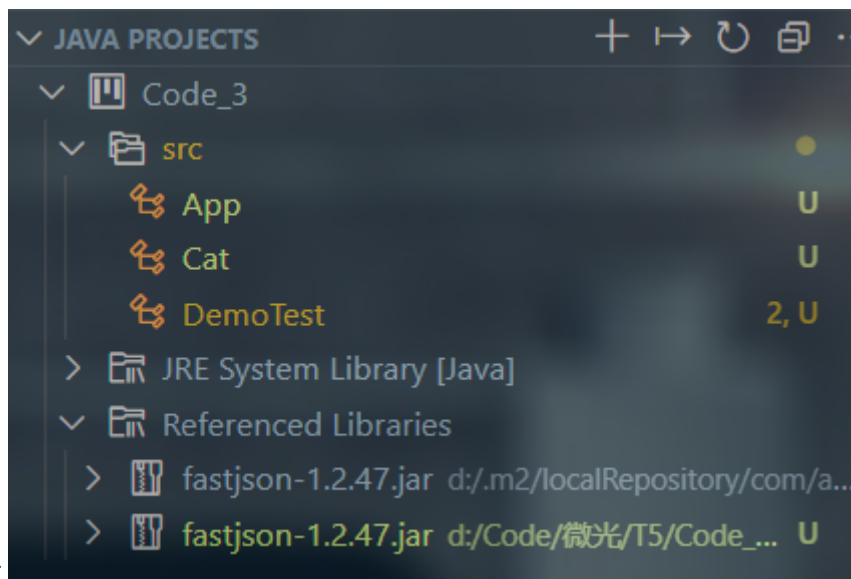
```

pom.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5
6
7    <groupId>com.example</groupId>
8    <artifactId>demo</artifactId>
9    <version>1.0-SNAPSHOT</version>
10
11    <name>demo</name>
12    <!-- FIXME change it to the project's website -->
13    <url>http://www.example.com</url>
14
15    <properties>
16      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
17      <maven.compiler.source>1.7</maven.compiler.source>
18      <maven.compiler.target>1.7</maven.compiler.target>
19    </properties>
20
21    <dependencies>
22      <dependency>
23        <groupId>junit</groupId>
24        <artifactId>junit</artifactId>
25        <version>4.11</version>
26        <scope>test</scope>
27      </dependency>
28    </dependencies>
29
30    <build>
31      <pluginManagement><!-- Lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
32      <plugins>

```

然后修改本地仓库位置，将镜像更换为阿里云(其实没感觉到快多少)。

-----学了一周maven后放弃治疗了-----



然后尝试将下载的jar包导入本地仓库

```

DemoTest.java:1: 错误: 程序包com.alibaba.fastjson不存在
import com.alibaba.fastjson.*;
^
DemoTest.java:5: 错误: 找不到符号
    System.out.println(JSON.toJSONString(cat));
                        ^
  符号:   变量 JSON
  位置: 类 DemoTest
2 个错误

```

结果不知道为什么路径设置好了还一直出错，于是选择开做t2

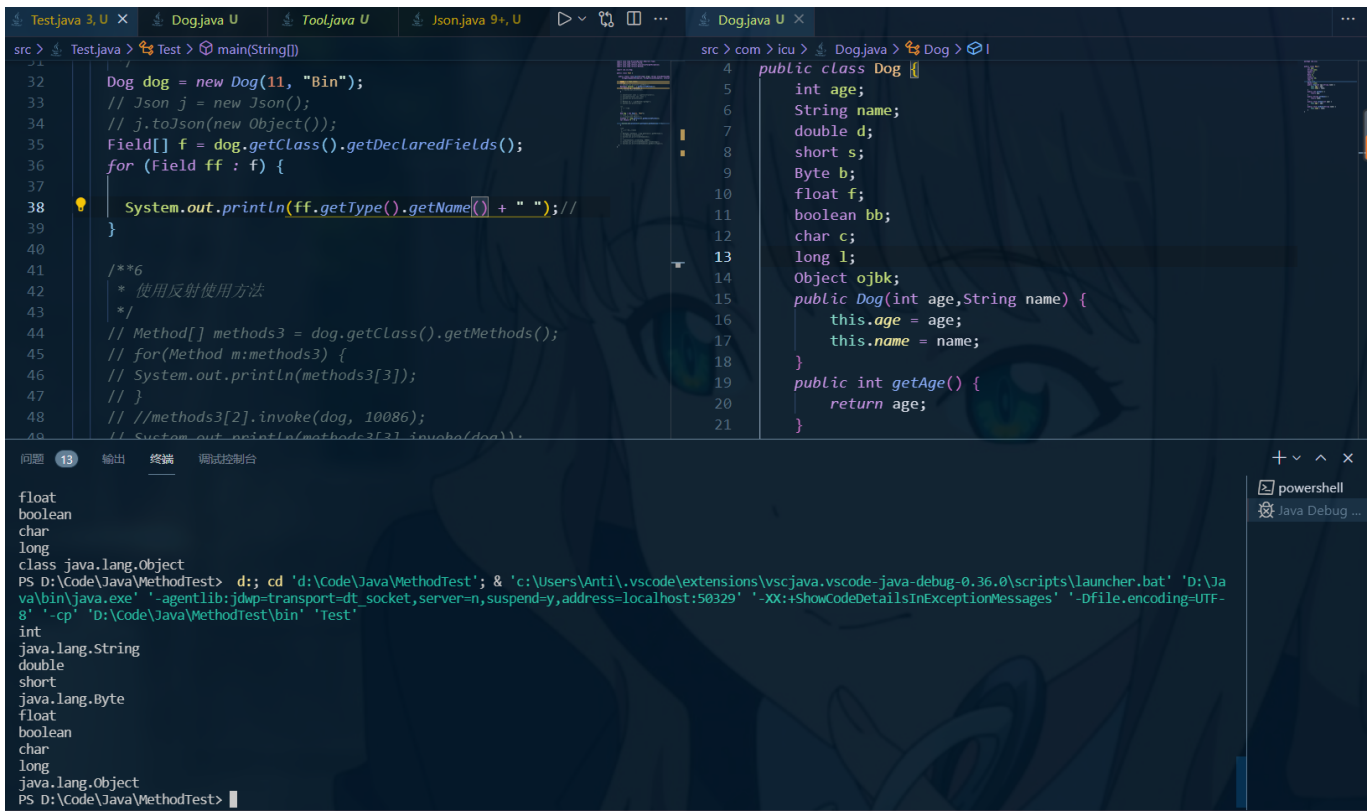
Task2

这道题前置知识有点多啊 ==，而且网上还有点难找想要的资料。

又花了几天学了一点反射

```
# Reflect笔记
## 基础
### 结构信息
- Class c = Class.forName("类的名称"); 从类的名称创建类对象。
- class.getName(); 获得类的名称 会带上包名，例如 com.java.test.App
- class.getSimpleName(); 获得类的简单名称，例如 App
- class.getInterfaces(); 获得类的接口，是一个数组，可以通过 Arrays.toString() 输出;
- class.getModifiers(); 获得修饰符，得到一个代表修饰符的数字;
- 可以通过 Modifier.toString(数字) 得到字符串，如 public final
### 成员变量
- Filed[] f = class.getDeclaredFields(); 获取所有 public 属性
- class.getDeclaredFields() 获取所有权限的属性
- f.getName 获得属性名字
- f.getType 获得属性类型
### 构造方法
- class.getConstructors() 所有构造方法
- class.getDeclaredConstructors() 所有权限的构造
- class.getConstructor() 指定的构造方法
- class.getDeclaredConstructor() 所有指定构造方法
- 通过 constructor.newInstance() 构造新对象
### 成员方法
- getMethods() 获得所有方法
- getName(); 获得名字
- getModifiers(); 获得修饰符
- getReturnTypes(); 获得返回类型
## 操作
### 使用反射操作属性(不安全)
1. 获取类的完整路径字符串
2. 根据类的完整路径获得类对象
3. 使用 set(obj,值) 进行对相应属性的修改
4. f.getXxx(obj) 获得obj该属性的值
### 使用Method执行方法
- m.invoke(obj) 具有返回值
- invoke 简单点来说，就是将一个方法 (method) 运用到具体的类上
比如我有一个类 Person，中间有属性 age
得到 Person 的 getAge 方法后
```

但是不知道怎么判断一个属性的类型，于是决定用了一个有点低级的方法，获得类型的名称再比较字符串。



- 1.先把基本的数据类型试出来。
- 2.再把数组试出来 可以发现数组的名称是由[]开头的，比如整形数组[]。要注意，数组可以存储数组（二维数组）或者对象。
- 3.再把对象试出来 对象有点复杂，比如我在icu.lzn.Cat得到的名称是icu.lzn.Cat，不好判断，于是我默认不是基本数据或者数组的都算作类（好像有点问题）

部分代码

- 大力出奇迹进行一个类型的判断

```
private boolean isBase(Field f) { // 判断可以直接输出的
    boolean ans = false;
    String s = f.getType().toString();
    if (s.equals("int") || s.equals("java.lang.String") || s.equals("double")
        || s.equals("short")
        || s.equals("java.lang.Byte") || s.equals("float") ||
        s.equals("boolean") || s.equals("char")
        || s.equals("long"))
        ans = true;
    return ans;
}
```

- 发现直接field.get(Object)好像会出问题，决定使用invoke。先得匹配对应的getter

ps.Boolean的getter格式是isXXX(),所以我进行了两次判断 分别判断和getXXX或isXxx匹配

- 获得对应get方法的代码

```

private static Method gMethod(Field f, Object obj) {
    Method tmp = null;
    Method[] ms = obj.getClass().getDeclaredMethods(); // 获取方法
    for (Method m : ms) {
        // System.out.println(m);
        if (m.getName().toLowerCase().equals(("get" +
f.getName()).toLowerCase()))// 通过字符串匹配, 得到对应的get
            tmp = m;
        if (m.getName().toLowerCase().equals(("is" +
f.getName()).toLowerCase()))// 通过字符串匹配, 得到对应的get
            tmp = m;
    }
    return tmp;
}

```

实例代码

说在前面的一些话

- 1.这个toJson的方法还有很多都暂时没实现, 比如数组是二维数组, 对象数组, 这份代码应该是不能转换的。数组暂时只写了`int[]` 和 `String[]`,但是`int[]`其他基本类型的数组是一个道理, 所以就不再增加代码长度了
2. 如果属性指向了`null`是会报错的, 所以在测试时要保证不是空指针。
3. 如果一个类里面没有属性, 或者数组长度为0, 也没有进行特别判断。
4. 数组或对象的get方法不应该直接`return int[]`否则会导致两个引用变量的值相同, 导致不安全。应当像我 `Cat.getAbilities()`那样, 新建一个数组, 再返回。其他地方同理, 但是我没修改= =

- 首先是toJson

```

import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

public class Json {
    public static Object obj;

    public static void toJson(Object obj)
        throws IllegalArgumentException, IllegalAccessException,
        InvocationTargetException {
        StringBuffer json = new StringBuffer();
        Work(obj, json);
        System.out.println(json);
    }

    // 主要的工作函数, 可以进行递归操作
    private static void Work(Object obj, StringBuffer s)
        throws IllegalArgumentException, IllegalAccessException,
        InvocationTargetException {

```



```
s.append("{\n");
// 获取所有属性
Field[] fs = obj.getClass().getDeclaredFields();
for (Field f : fs) {
    /**
     * 如果是基础类型的情况
     */
    if (isBase(f)) {
        jsonBase(f, s, obj);
        continue;
    }
    /**
     * 如果是数组的情况
     */
    if (isArr(f)) {
        Method tmp = gMethod(f, obj);
        s.append "\"" + f.getName() + "\": [" );
        if (tmp.invoke(obj).getClass() == int[].class) {
            int[] is = (int[]) tmp.invoke(obj);
            for (int i = 0; i < is.length; i++) {
                s.append "\"" + is[i] + "\"" + "," );
            }
            s.deleteCharAt(s.length() - 1);
            s.append("],\n");
            continue;
        }
        Object[] as = (Object[]) tmp.invoke(obj);

        if (as.length == 0) {
            s.append("],\n");
            continue;
        }
        for (Object ob : as) {
            s.append "\"" + ob + "\"" + "," );
        }
        s.deleteCharAt(s.length() - 1);
        s.append("],\n");
        continue;
    }
    /**
     * 若属性为对象, 进行递归
     */
    s.append "\"" + f.getName() + "\": ";
    Method tmp = gMethod(f, obj);
    Work(tmp.invoke(obj), s);
    s.append(",\n");
}
s.deleteCharAt(s.length() - 2);
s.append("}");
return;
}
```

```
/**
 * 判断是否为基础类型
 */
private static boolean isBase(Field f) {
    boolean ans = false;
    String s = f.getType().getName();
    if (s.equals("int") || s.equals("java.lang.String") || s.equals("double")
|| s.equals("short")
        || s.equals("java.lang.Byte") || s.equals("float") ||
s.equals("boolean") || s.equals("char")
        || s.equals("long"))
        ans = true;
    return ans;
}

// 判断是否是数组
private static boolean isArr(Field f) {
    boolean ans = false;
    String s = f.getType().getName();
    if (s.startsWith "["))
        ans = true;
    return ans;
}

/**
 * 生成基本类型的json
 *
 * @throws IllegalAccessException
 * @throws IllegalArgumentException
 * @throws InvocationTargetException
 *
 */
private static void jsonBase(Field f, StringBuffer s, Object obj)
    throws IllegalArgumentException, IllegalAccessException,
InvocationTargetException {
    s.append("\"" + f.getName() + "\": "); // 获得变量名

    Method tmp = gMethod(f, obj);
    s.append "\"" + tmp.invoke(obj) + "\",\n");
}

// 获取一个属性对应的getXxx
private static Method gMethod(Field f, Object obj) {
    Method tmp = null;
    Method[] ms = obj.getClass().getDeclaredMethods(); // 获取方法
    for (Method m : ms) {
        // System.out.println(m);
        if (m.getName().toLowerCase().equals(("get" +
f.getName()).toLowerCase())) // 通过字符串匹配, 得到对应的get
            tmp = m;
        if (m.getName().toLowerCase().equals(("is" +
f.getName()).toLowerCase())) // 通过字符串匹配, 得到对应的get
            tmp = m;
    }
}
```

```
        return tmp;
    }
}
```

- Cat类(放在了包lizinuo.icu下)

```
package lizinuo.icu;

public class Cat {
    private String name;
    private int age;
    private boolean hasOwner;
    private String breed;
    private String[] abilities = null;
    private Owner owner;
    Cat() {
    };

    public Cat(String name, int age, boolean hasOwner, String breed, String[]
abilities, Owner owner) {
        this.name = name;
        this.age = age;
        this.hasOwner = hasOwner;
        this.breed = breed;
        this.abilities = abilities;
        this.owner = owner;
    }
    public Owner getOwner() {
        return owner;
    }
    public String[] getAbilities() {
        return abilities;
    }
    public int getAge() {
        return age;
    }
    public String getBreed() {
        return breed;
    }
    public String getName() {
        return name;
    }
    public boolean isHasOwner() {
        return hasOwner;
    }
}
```

- 然后是 Owner类


```
package lizينو.icu;

public class Owner {
    int[] memberAge;
    String name;

    public Owner(String name, int[] i) {
        this.name = name;
        this.memberAge = i;
    }

    public int[] getMemberAge() {
        return memberAge;
    }

    public String getName() {
        return name;
    }
}
```

- 最后是测试代码

```
import lizينو.icu.Cat;
import lizينو.icu.Owner;

public class App {
    public static void main(String[] args) throws Exception {
        Cat myCat = new Cat("Pickle", 2, true, "American Shorthair", new String[]
{ "ear", "sleep", "run" },
            new Owner("Glimmer", new int[] { 24, 19, 18 }));
        Json.toJson(myCat);
    }
}
```

- 进行检查

输出了

```
{
  "name": "Pickle",
  "age": "2",
  "hasOwner": "true",
  "breed": "American Shorthair",
  "abilities": ["ear", "sleep", "run"],
  "owner": {
    "memberAge": ["24", "19", "18"],
    "name": "Glimmer"
  }
}
```

PS D:\Code\Java\MethodTest>

再随便在网上找个json格式转化网站试一下,发现能够成功转化,说明格式对了

JSON数据

复制 格式化 删除空格 删除空格并转义 去除转义

```
{
  "name": "Pickle",
  "age": "2",
  "hasOwner": "true",
  "breed": "American Shorthair",
  "abilities": [
    "ear",
    "sleep",
    "run"
  ],
  "owner": {
    "memberAge": [
      "24",
      "19",
      "18"
    ],
    "name": "Glimmer"
  }
}
```

视图

查找: GO! 下一个 上一个 全部展开 全部收缩

JSON

- name : "Pickle"
- age : "2"
- hasOwner : "true"
- breed : "American Shorthair"
- abilities
 - 0 : "ear"
 - 1 : "sleep"
 - 2 : "run"
- owner
 - memberAge
 - 0 : "24"
 - 1 : "19"
 - 2 : "18"
 - name : "Glimmer"

Task3

还没学注解, 我先交了^ ^