Atitit cfg gradle manage maen and xml seriz配置管理xml序列化

目录

[1. Xml序列化 1](#_Toc9854)

[1.1. XStream 1](#_Toc28509)

[1.2. JAXB java.beans.XMLEncoder 1](#_Toc2714)

[2. gradle 1](#_Toc30920)

# Xml序列化

## XStream

**public** **static** **void** main(String[] args) {

Map m=**new** HashMap ();

m.put("k1",11);

//dom4j is trouble

XStream xStream=**new** XStream();

String xmocontent=xStream.toXML(m);

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

}

在Java对象的XML序列化和反序列化(1)和Java对象的XML序列化和反序列化(2)介绍了利用java.beans.XMLEncoder/XMLDecoder实现Java对象的XML序列化和反序列化。从XMLEncoder的输出结果来看，产生的XML序列包含了太多的描述信息，看上去很不直观。而且XMLEncoder在对纯数据对象进行序列化的时候，生成的XML代码简直就是一团乱麻。

## JAXB java.beans.XMLEncoder

在实际应用中，往往希望根节点名对应类名、节点名对应类成员名，节点值对应类成员值，有什么工具类可以实现这一点呢？答案就是JAXB（Java Architecture for XML Binding），使用非常简单，生成的XML代码简洁明了，与被序列化的Java对象对应关系非常好。

下面是使用JAXB对纯数据对象进行XML序列化和反序列化的例子：

# gradle

Insatll gradle 6.5 99M

Gradle plugin already in eckpse

New gradle prj

/gradlePrj1/build.gradle

dependencies {

// https://mvnrepository.com/artifact/com.thoughtworks.xstream/xstream

compile group: 'com.thoughtworks.xstream', name: 'xstream', version: '1.4.2'

// https://mvnrepository.com/artifact/org.dom4j/dom4j

compile group: 'org.dom4j', name: 'dom4j', version: '2.1.0'

// This dependency is exported to consumers, that is to say found on their compile classpath.

api 'org.apache.commons:commons-math3:3.6.1'