JAXB

# 介绍JAXB

JAXB - Java Architecture for XML Binding

JAXB提供了一种把Java object转成XML，或者把XML转成Java object的机制。

JAXB有两个过程，一个是unmarshalling(解组;数据编出;)，另一个是marshalling(编组)。

unmarshalling：reading。从XML instance转成Java content。

marshalling：writing。从Java content转成XML instance。

（***jdk1.6以后可以包含JAXB***）

# 常用API

1. JAXBContext类，是应用的入口，通过该类创建序列化和反序列化对象，也即编组对象和解组对象；
2. Marshaller 编组接口，将Java对象序列化为XML数据；
3. Unmarshaller 解组接口，将XML数据反序列化为Java对象。

# 常用注解

1. @XmlRootElement，将Java类或枚举映射成XML元素根节点，是唯一一个必须注解，name属性指定根节点名称，不指定默认为类名的小写；
2. @XmlElement，将Java类的一个属性映射为XML节点元素，name属性可自定义元素名；
3. @XmlAttribute，将Java类的一个属性映射为XML节点元素的属性，name属性可自定义属性名；
4. @XmlType，将Java类或枚举类型映射到XML模式类型，常与@XmlRootElement、@XmlAccessorType共用，propOrder属性定义字段生成的XML节点顺序；
5. @XmlAccessorType，控制字段或属性的序列化。属性XmlAccessType有4个常量值:
6. FIELD表示JAXB将自动绑定Java类中的每个非静态的（static）、非瞬态的（由@XmlTransient标注）字段到XML；
7. PROPERTY表示java对象中所有通过getter/setter方式绑定成属性到XML；
8. PUBLIC\_MEMBER表示Java对象中所有的public访问权限的成员变量和通过getter/setter方式访问的成员变量，该值为默认值；
9. NONE表示Java对象的所有属性都不映射为XML的元素；
10. @XmlAccessorOrder，控制JAXB 绑定类中属性和字段的排序，有两个属性，AccessorOrder.ALPHABETICAL——对生成的XML元素按字母书序排序，XmlAccessOrder.UNDEFINED——不排序，默认为该值；
11. @XmlJavaTypeAdapter，自定义适配器（即扩展抽象类XmlAdapter并覆盖marshal()和unmarshal()方法），解决日期（Date）,数字（Number）格式化问题；
12. @XmlElementWrapper ，对于数组或集合（即包含多个元素的成员变量），生成一个包装该数组或集合的XML元素（称为包装器），该注解只能用在集合上；
13. @XmlTransient ，用于标示在由Java对象映射XML时，忽略此属性，在生成的XML文件中将不出现此元素。

# 实际应用中注意的问题

1. 如果JavaBean中定义了有参的构造器，那么必须同时定义无参构造器，否则转XML会抛无默认构造函数的异常;
2. 成员变量值为NULL时，将不会映射成对应的XML元素——由于基本数据类型默认值不为空，所以基本数据类型不设值也会映射成XML元素，值为默认值,所以如果模型需要基本数据，在属性定义的时候尽量使用包装类型；
3. @XmlAccessorType 注解中如果属性值为XmlAccessType.FIELD，则表示通过成员变量来映射，set/get方法上的映射注解就是多余的，所以如果此时set/get方法上再标注元素或者属性映射注解，将抛属性重复性异常；属性值为XmlAccessType.NONE不映射为XML元素的前提是Java字段或set/get方法上都没有映射注解；
4. @XmlType propOrder属性能够自定义字段的排序，该属性如果设置，要么写成{}的形式，否则在就必须将所有@XmlElement标注或者没有@XmlElement标注的但实际上会被映射为XML节点的字段添加到排序列表，不然会抛异常；如果propOrder属性设置有值，@XmlAccessorOrder注解的元素排序规则将失效；

# 编写JAXB类

|  |
| --- |
| **package** com.fwx.pojo;  **import** javax.xml.bind.annotation.\*;  **/\*\*  \* 账号实体类  \* @author Fwx  \* @param *null* \* @return  \*/  /\*\*  \* @XmlAccessorType(XmlAccessType.FIELD)指定映射为XML文档的类中要映射成XML格式中元素标签的成员属性，为这个类中的所有成员属性变量  \* @XmlRootElement 用在class类的注解，常与@XmlRootElement，@XmlAccessorType一起使用.也可以单独使用,如果单独使用,需要在成员属性的get方法上加@XmlElement等注解.  \* @XmlType,在使用@XmlType的propOrder 属性时，必须列出JavaBean对象中的所有XmlElement，否则会报错。  \* @author  \*/** @XmlAccessorType(XmlAccessType.***FIELD***) @XmlRootElement(name = **"Account"**) @XmlType(propOrder = {**"accountName"**,**"comment"**,**"passWord"**}) **//可以设置生成字段的顺序 public class** Account {  @XmlElement(name = **"AccountName"**,required = **true**)  **private** String **accountName**;  @XmlAttribute(name = **"CreationTime"**)  **private** String **creationTime**;  @XmlElement(name = **"PassWord"**,required = **true**)  **private** String **passWord**;  @XmlElement(name = **"Comment"**)  **private** String **comment**;   **public** Account() {  }   **public** Account(String accountName, String creationTime, String passWord, String comment) {  **this**.**accountName** = accountName;  **this**.**creationTime** = creationTime;  **this**.**passWord** = passWord;  **this**.**comment** = comment;  }   **public** String getAccountName() {  **return accountName**;  }   **public void** setAccountName(String accountName) {  **this**.**accountName** = accountName;  }   **public** String getPassWord() {  **return passWord**;  }   **public void** setPassWord(String passWord) {  **this**.**passWord** = passWord;  }   **public** String getComment() {  **return comment**;  }   **public void** setComment(String comment) {  **this**.**comment** = comment;  }   **public** String getCreationTime() {  **return creationTime**;  }   **public void** setCreationTime(String creationTime) {  **this**.**creationTime** = creationTime;  }   @Override  **public** String toString() {  **return "Account{"** +  **"accountName='"** + **accountName** + **'\''** +  **", creationTime='"** + **creationTime** + **'\''** +  **", passWord='"** + **passWord** + **'\''** +  **", comment='"** + **comment** + **'\''** +  **'}'**;  } } |

# 使用JAXB生成xml文件

|  |
| --- |
| **/\*\*  \* 1.使用Java对象生成xml文件(只有account对象)  \*  \* @param  \* @return void  \*/** @Test **public void** javaToXml() {  **try** {  **//1.创建Jaxb对象** JAXBContext jaxbContext = JAXBContext.*newInstance*(Account.**class**);  **//2.创建marshaller** Marshaller marshaller = jaxbContext.createMarshaller();  **//3.1.设置输出样式(不设置生成的xml文件没有换行和缩进)** marshaller.setProperty(Marshaller.***JAXB\_FORMATTED\_OUTPUT***, **true**);  **//3.2.设置编码格式** marshaller.setProperty(Marshaller.***JAXB\_ENCODING***,**"UTF-8"**);  **//4.生成数据** Account account = **new** Account(**"1234AN"**, **"2020-02-18"**, **"123PW"**, **"qq"**);  **//5.生成xml文件** marshaller.marshal(account, **new** FileOutputStream(**"D:\\jaxbTest.xml"**));  } **catch** (JAXBException e) {  e.printStackTrace();  } **catch** (FileNotFoundException e) {  e.printStackTrace();  } } |

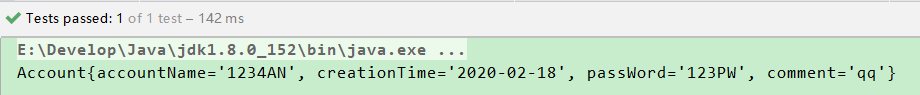
生成的文件jaxbTest.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" standalone="yes"?>  <Account CreationTime="2020-02-18">  <AccountName>1234AN</AccountName>  <Comment>qq</Comment>  <PassWord>123PW</PassWord>  </Account> |

# 读取xml文件生成java对象

|  |
| --- |
| **/\*\*  \* 2.使用xml文件生成Java对象(只有account对象)  \*  \* @param  \* @return void  \*/** @Test **public void** xmlToJava() {  **try** {  **//1.创建Jaxb对象** JAXBContext jaxbContext = JAXBContext.*newInstance*(Account.**class**);  **//2.创建unmarshaller** Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();  **//3.读取xml文件，生成对象** Account account = (Account) unmarshaller.unmarshal(**new** File(**"D:\\jaxbTest.xml"**));  **//4·打印对象** System.***out***.println(account);  } **catch** (JAXBException e) {  e.printStackTrace();  } } |

输出结果：



# 使用schema文件生成JAXB类（非必须）

## 编写schema文件

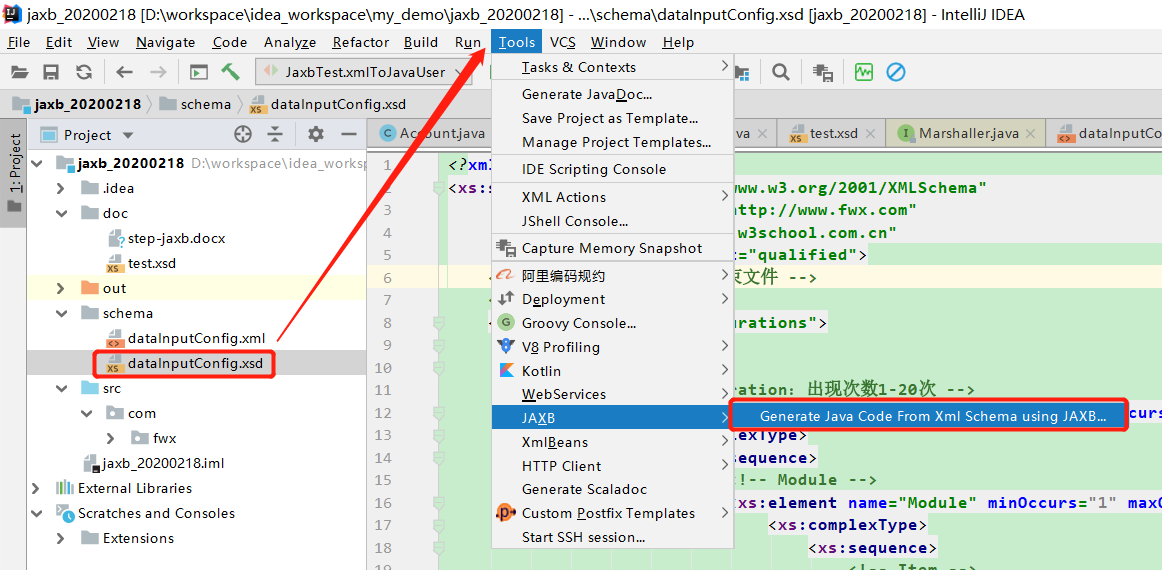
|  |
| --- |
| *<?***xml version="1.0"***?>* <**xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://www.fwx.com"  xmlns="http://www.w3school.com.cn"  elementFormDefault="qualified"**>  **<!-- DatainputConfig的约束文件 -->  <!-- Configurations -->** <**xs:element name="Configurations"**>  <**xs:complexType**>  <**xs:sequence**>  **<!-- Configuration：出现次数1-20次 -->** <**xs:element name="Configuration" minOccurs="1" maxOccurs="20"**>  <**xs:complexType**>  <**xs:sequence**>  **<!-- Module -->** <**xs:element name="Module" minOccurs="1" maxOccurs="20"**>  <**xs:complexType**>  <**xs:sequence**>  **<!-- Item -->** <**xs:element name="Item" minOccurs="1" maxOccurs="20"**>  <**xs:complexType**>  <**xs:attribute name="Name" type="xs:string" use="required"**></**xs:attribute**>  <**xs:attribute name="Value" type="xs:string" use="required"**></**xs:attribute**>  <**xs:attribute name="Description" type="xs:string" use="required"**></**xs:attribute**>  </**xs:complexType**>  </**xs:element**>  </**xs:sequence**>  <**xs:attribute name="Name" use="required"**></**xs:attribute**>  <**xs:attribute name="Type" use="required"**></**xs:attribute**>  <**xs:attribute name="Variable" use="required"**></**xs:attribute**>  <**xs:attribute name="Description" use="required"**></**xs:attribute**>  </**xs:complexType**>  </**xs:element**>  </**xs:sequence**>  <**xs:attribute name="Name" use="required"**></**xs:attribute**>  <**xs:attribute name="Description" use="required"**></**xs:attribute**>  </**xs:complexType**>  </**xs:element**>  </**xs:sequence**>  <**xs:attribute name="Name" type="xs:string" use="required"**/>  <**xs:attribute name="Descriptione" type="xs:string" use="required"**/>  </**xs:complexType**>  </**xs:element**>  </**xs:schema**> |

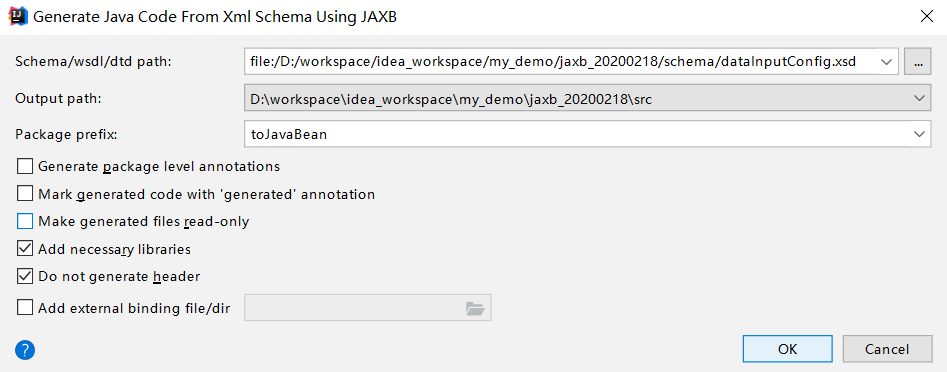
## 编写xml文件测试schema的正确性

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8" standalone="yes"***?>* **<!-- 测试schema文件的xml(DatainputConfig) -->** <**Configurations Name="kafka\_connection\_configuration" Descriptione="hdfs、Solr、HBase、Kafka等的配置信息" xmlns="http://www.fwx.com"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.w3school.com.cn dataInputConfig.xsd"**>  <**Configuration Name="kafka\_connection\_configuration" Description="hdfs、Solr、HBase、Kafka等的配置信息"**>  <**Module Name="kafka\_connect" Type="kafka" Variable=" " Description="接入kafka数据之后进行提取操作的配置 "**>  <**Item Name="kafka.topic" Value="mybcpdata" Description="topic，多个时以,分割"**/>  </**Module**>  </**Configuration**> </**Configurations**> |

## 生成操作

选中schema文件（datainputConfig.xsd）🡪Tools🡪JAXB🡪Generato Java Code From Xml Schema using JAXB…





测试时schema文件中element ref 属性引用暂时**没有成功**！入下图

