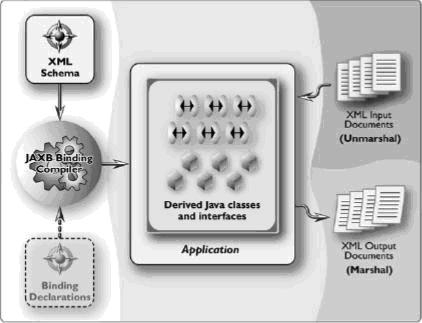
JAXB技术介绍  
    Java  Architecture for XML Binding (JAXB) 是一个业界的标准，是一项可以根据XML Schema产生Java类的技术。该过程中，JAXB也提供了将XML实例文档反向生成Java对象树的方法，并能将Java对象树的内容重新写到XML实例文档。从另一方面来讲，JAXB提供了快速而简便的方法将XML模式绑定到Java表示，从而使得Java开发者在Java应用程序中能方便地结合XML数据和处理函数。  
    这意味着你不需要处理甚至不需要知道XML编程技巧就能在Java应用程序中利用平台核心XML数据的灵活性。而且，可以充分利用XML的优势而不用依赖于复杂的XML处理模型如SAX或DOM。JAXB 隐藏了细节并且取消了SAX和DOM中没用的关系——生成的JAXB类仅描述原始模型中定义的关系。其结果是结合了高度可移植Java代码和高度可移植的XML数据。其中这些代码可用来创建灵活、轻便的应用程序和Web服务。

2.         JAXB的体系结构  
[](http://mengshj.blog.163.com/album/prevPhoto.do?photoId=_fks_h3eYVsf8FWzR3nXK5c9nZ4fTQtGY5UqD)  
  
JAXB的体系结构和应用过程如图所示，一般来说包含以下几个步骤：

1)        根据你的应用程序所要操作的XML数据格式，撰写相应的XML Schema，有关XML Schema的知识，请参阅“[参考资料](http://mengshj.blog.163.com/richTextEditor.do?bid=&r=0&uid=0#XMLSchema)”

2)        使用JAXB 所带的编译工具（Binding Compiler），将这个XML Schema文件作为输入，产生一系列相关的Java Class和Interface

3)        在使用JAXB编译工具的时候，可以有选择性的提供一个配置文件（图的虚线部分），来控制JAXB编译工具的一些高级属性。

4)        这些Java Class和Interface是你的应用程序操纵XML数据的主要接口和方法。

5)        通过JAXB对XML文档进行的操作主要包括：将符合XML Schema规定的XML文档解析生成一组相应的Java对象；对这些对象进行操作（修改、增加和删除对象的属性等等）；然后将这些对象的内容保存到这个XML文档中。

JDK中JAXB相关的重要Annotation：

@XmlType，将Java类或枚举类型映射到XML模式类型

@XmlAccessorType(XmlAccessType.FIELD) ，控制字段或属性的序列化。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。其他值还有XmlAccessType.PROPERTY和XmlAccessType.NONE。

@XmlAccessorOrder，控制JAXB 绑定类中属性和字段的排序。

@XmlJavaTypeAdapter，使用定制的适配器（即扩展抽象类XmlAdapter并覆盖marshal()和unmarshal()方法），以序列化Java类为XML。

@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）。

@XmlRootElement，将Java类或枚举类型映射到XML元素。

@XmlElement，将Java类的一个属性映射到与属性同名的一个XML元素。

@XmlAttribute，将Java类的一个属性映射到与属性同名的一个XML属性。

定义数据格式xml Schema ：DemoSchema.xsd

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<schema xmlns=*"http://www.w3.org/2001/XMLSchema"* targetNamespace=*"http://www.example.org/DemoSchema"*

xmlns:xs=*"http://www.example.org/DemoSchema"* elementFormDefault=*"qualified"*>

<element name=*"movie"* type=*"xs:Movie"*/>

<complexType name=*"ObjectInfo"*>

<sequence>

<element name=*"id"* type=*"string"* />

<element name=*"image-url"* type=*"string"* nillable=*"true"*/>

<element name=*"priority"* type=*"int"* nillable=*"true"*/>

<element name=*"localized"* type=*"xs:Localized"* nillable=*"true"*/>

</sequence>

</complexType>

<!--localized-->

<complexType name=*"Localized"*>

<sequence>

<element name=*"lang"* type=*"xs:Lang"* minOccurs=*"0"* maxOccurs=*"unbounded"* nillable=*"true"*/>

</sequence>

</complexType>

<!--lang-->

<complexType name=*"Lang"*>

<sequence>

<element name=*"code"* type=*"string"* />

<element name=*"name"* type=*"string"* nillable=*"true"*/>

<element name=*"attribute"* type=*"string"* nillable=*"true"*/>

<element name=*"description"* type=*"string"* nillable=*"true"*/>

</sequence>

</complexType>

<complexType name=*"Movie"*>

<sequence>

<element name=*"object-Infos"* type=*"xs:ObjectInfos"* maxOccurs=*"1"* minOccurs=*"0"*></element>

</sequence>

</complexType>

<complexType name=*"ObjectInfos"*>

<sequence>

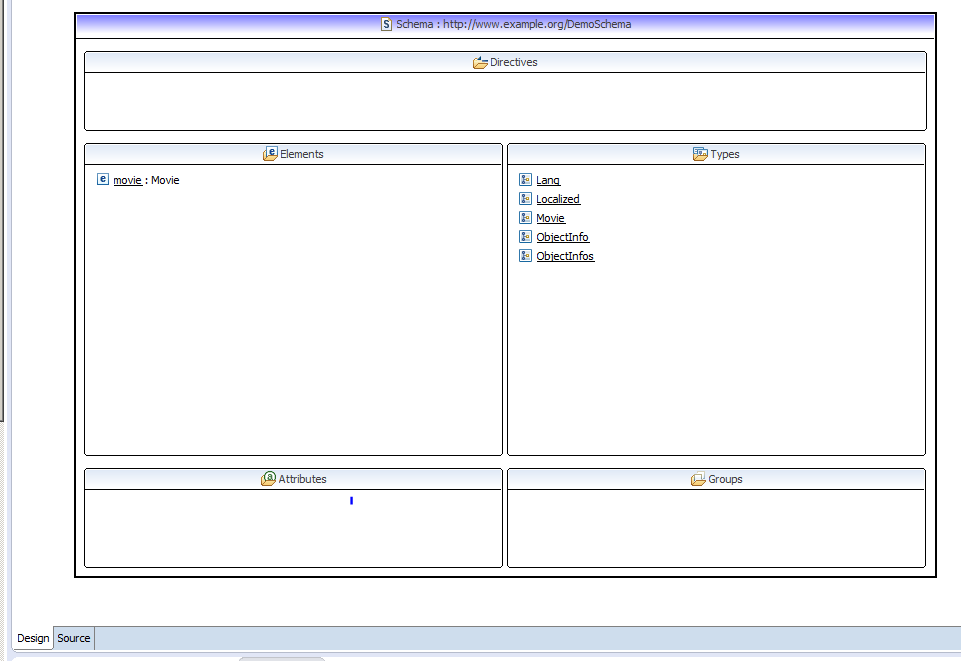
<element name=*"object-info"* type=*"xs:ObjectInfo"* maxOccurs=*"unbounded"* minOccurs=*"0"*></element>

</sequence>

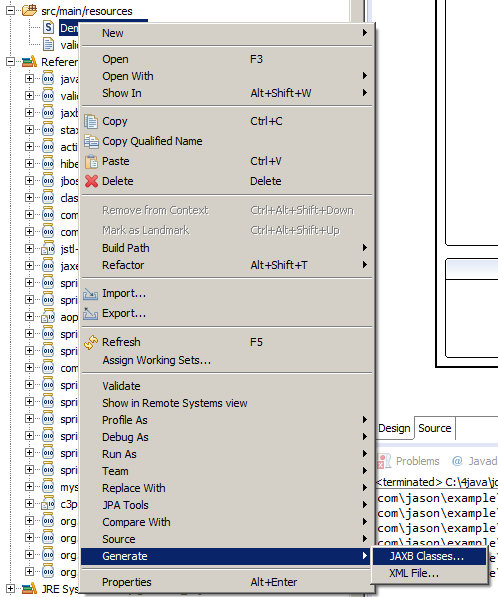
</complexType>

</schema>

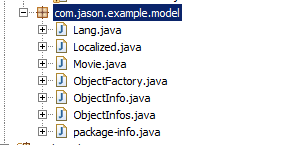
可以使用图形界面来编辑 Schema



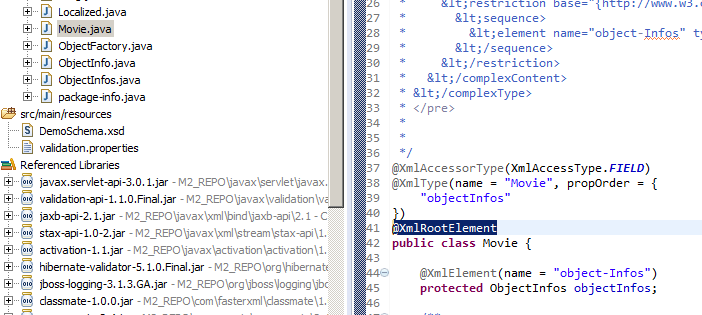
Schema 编辑完成以后，导出 Schema 的model 类



导出的 model 类的list



需要在 Movie 的类中配置XML 根节点 @XmlRootElement



2 数据导入DemoFileUploadController

DemoCache 缓存导入的数据

3 数据导出DemoSeekController

4 Demo 导入导出 Url

http://127.0.0.1:8080/data-ingestion-demo-war/fileUpload.html

<http://127.0.0.1:8080/data-ingestion-demo-war/getInfoDemo.htm>

导入的 demo.xml数据的格式：

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<movie xmlns="http://www.example.org/DemoSchema">

<object-Infos>

<object-info>

<id>bf435848-a165-4361-a2ad-13c23fa0b3ff</id>

<image-url>20141211064023715.png</image-url>

<priority>666</priority>

<localized>

<lang>

<code>en</code>

<name>3333</name>

<attribute>555</attribute>

<description>555</description>

</lang>

</localized>

</object-info>

<object-info>

<id>bf435848-a165-4361-a2ad-13c23fa0b3</id>

<image-url>20141211064023715.png</image-url>

<priority>140</priority>

<localized>

<lang>

<code>en</code>

<name>Sintel\_en</name>

<attribute>Hero\_en</attribute>

<description>Hero\_en</description>

</lang>

<lang>

<code>zh</code>

<name>Sintel\_zh</name>

<attribute>Hero\_zh</attribute>

<description>Hero\_zh</description>

</lang>

</localized>

</object-info>

</object-Infos>

</movie>