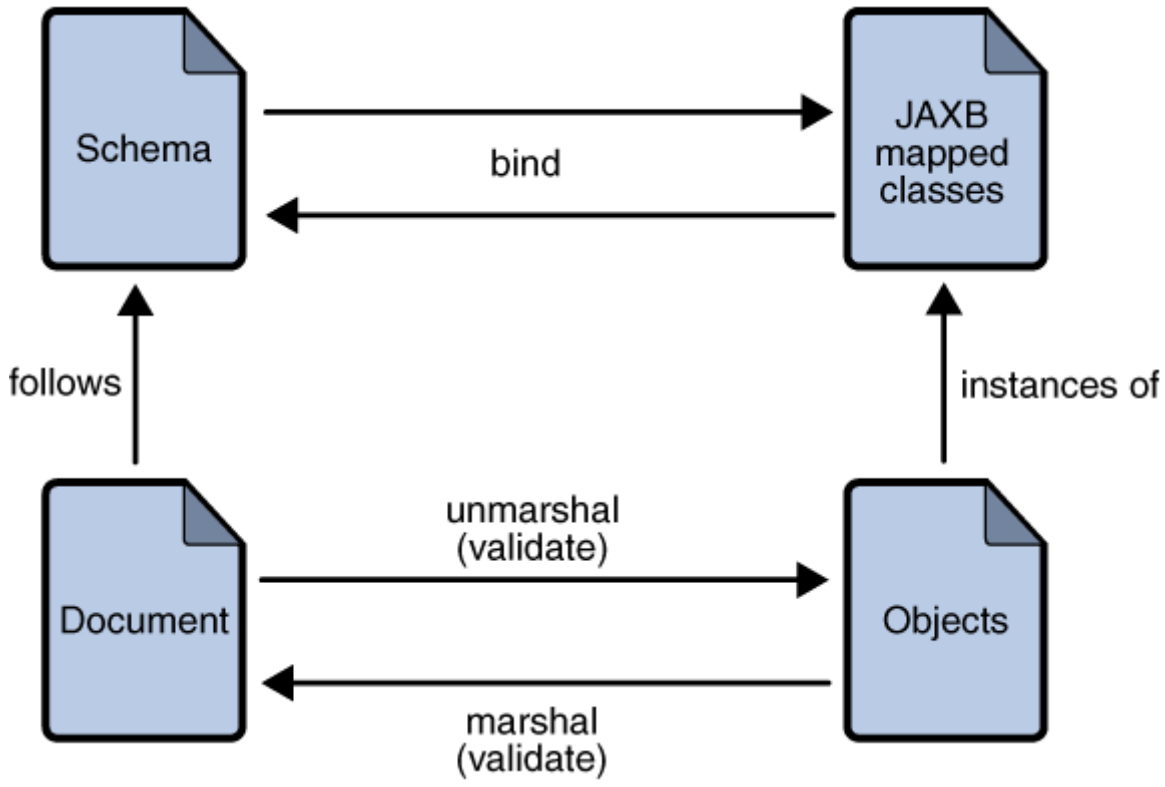
Terence Chao, 07/31/2007

Basic Examples for JAXB 筆記

Preface

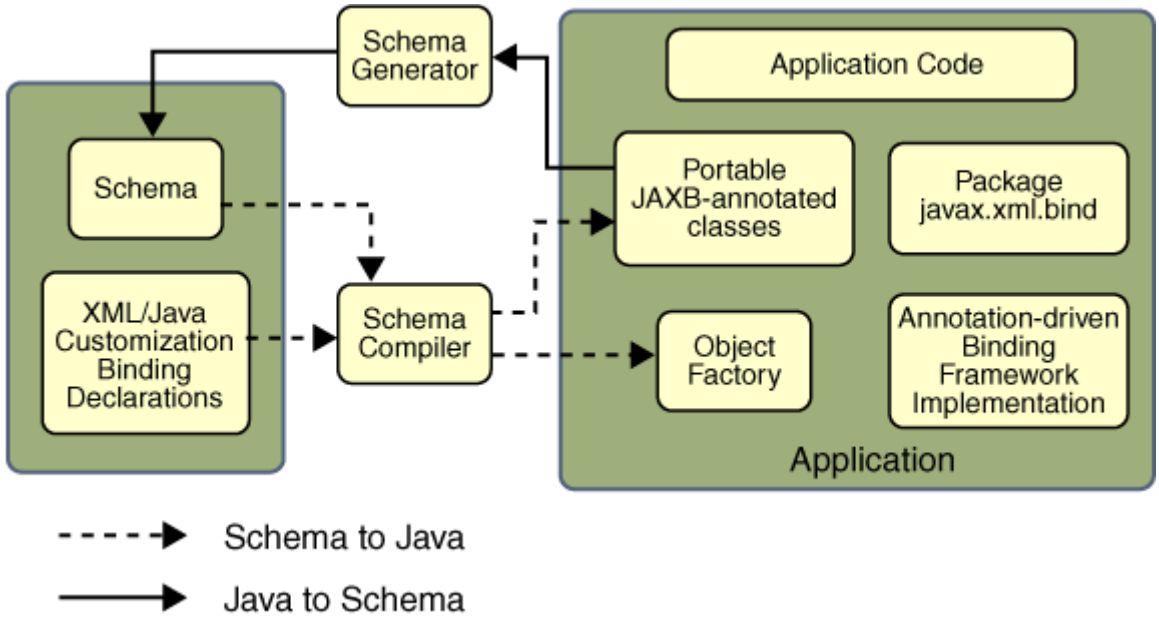
這篇主要是補充 Sun 的 JAXB 的文件有關 Basic Examples 的部份。下面是展示 JAXB 中有關 Marshal、Unmarshal 和 Validate 這三個部份的基本功能。
JAXB 就跟他的名字一樣，JAXB 協助我們可以由 XML 產生 Java 物件(文件寫 Java Content trees)，或由 Java 物件來產生 XML 文件。跟 RMI 那邊用詞一致，由 XML->Java instance 的動作稱爲 Unmarshalling；由 Java instance -> XML 的動作稱爲 Marshalling。在 JAXB 這些動作和名詞可以從 Sun 文件中的這個圖看出來



另外，JAXB 也提供不同的路讓我們選擇

1. 由 Java Class 透過 Schema Generator 來產生 XML Scheama
   %JAVA_HOME%/bin/schemagen.exe
2. 透過 XML Schema 產生 Java Class 和 Object Factory
   %JAVA_HOME%/bin/xjc.exe

Note:
JAXB 是在 JDK 6 以後才放在 J2SE 裡面，如果使用 JDK 5 或更早以前的請另外下載。



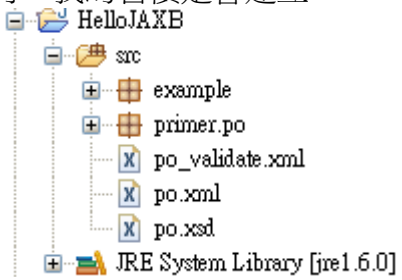
在這邊我是使用 J2EE 5 裡面的範例程式來踹 Sun 文件的範例，因此我走的路線是從 XML Schema 透過 Schema Compiler 產生 Java Class 和 Object Factory 的方式。

Environment

JDK 1.6
Eclipse 3.2

Project Hierarchy

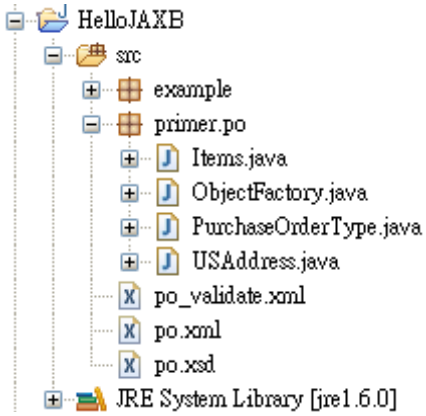
只要建立一般的 Java Project 即可。我的習慣是會建立 source folder，因此看起來如下



透過 xjc 產生的 java class 和 Java Tutorial 一樣放在 primer.po 下，po.xml 也是從 Java EE 5 範例裡面偷過來用的；po_validate.xml 則是用於 Validate 那邊使用和 po.xml 只有在 quantity 的地方違反 po.xsd 的規定。

Generate Java Class from XML Schema

xjc.exe 提供很多方法(本地、遠端、透過 HTTP 等)讓我們由 XML Schema 產生 Java Class。在這裡我已經從 Java EE 5 的範例中拿到 po.xsd，所以很簡單直接用-p 的參數設定產生的 Java Class 要放在那個 package 下即可，如果沒有對應的 folder xjc.exe 會自己 create。



po.xsd vs Generated Java Classes

如上節圖中所示，xjc.exe 會由 XML Schema 產生許多類別和一個 ObjectFactory.java

USAddress in po.xsd

```
<xsd:complexType name="USAddress">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="street" type="xsd:string"/>
    <xsd:element name="city" type="xsd:string"/>
    <xsd:element name="state" type="xsd:string"/>
    <xsd:element name="zip" type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>
</xsd:complexType>
```

產生的 USAddress.java 的程式碼片段和 Annotation

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "USAddress", propOrder = {
    "name",
    "street",
    "city",
    "state",
    "zip"
})
public class USAddress {
    @XmlElement(required = true)
    protected String name;
    @XmlElement(required = true)
    protected String street;
    @XmlElement(required = true)
    protected String city;
    @XmlElement(required = true)
    protected String state;
    @XmlElement(required = true)
    protected BigDecimal zip;
    @XmlAttribute
    @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
    protected String country;
```

To be Continue...

Unmarshal Read Example

這個範例在 Java EE 裡面有完整的程式，就不貼完整程式。
要使用 JAXB 的 API 我們要先取得 JAXBContext 物件

```
JAXBContext jc = JAXBContext.newInstance( "primer.po" );
```

我們可以很簡單的透過 JAXBContext 的 static factory method 來取得 JAXBContext inatance，接著就可以如下列範例透過 JAXBContext 取得 Marsheller、Unmarshaller 等物件。

```
Unmarshaller u = jc.createUnmarshaller();
```

取得 Unmarshaller 物件之後，就可以將 XML docuement unmarshal 成 Java instance

```
JAXBElement<?> poElement = (JAXBElement<?>)u.unmarshal(new File("bin\\po.xml"));
PurchaseOrderType po = (PurchaseOrderType)poElement.getValue();
```

Unmarshaller 有其他的 method 可以讓我們設定 unmarshal 之後的 type，就可以拿到 JAXVElement<Type>的物件。

Marshal Modify Example

接著就是從 XML unmarshal 成 Java 物件之後，我們改變物件內容再透過 Marshaller 來產生對應的 XML 文件顯示在 System.out。
需要注意的事，如果產生的 Java Class 的 annotation 沒有@XmlRootElement 這個 annotation 會在 marshaling 的時候發生 Exception，JAXB 會提醒說需要加這個 annotation。

```
package example;

import java.io.File;
import java.math.BigDecimal;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBElement;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
import primer.po.PurchaseOrderType;
import primer.po.USAddress;

/**
 * 要在生出來的PurchaseOrderType.java
 * 加上@XmlRootElement的Annotation, 才可以正常Marshal<br>
 *
 * @author Terry
 *
 */
public class MarshalModify {
  public static void main(String[] args) {
    try {
      JAXBContext context = JAXBContext.newInstance("primer.po");

      Unmarshaller u = context.createUnmarshaller();

      JAXBElement<?> poElement = (JAXBElement<?>) u.unmarshal(new File(
                              "bin\\po.xml"));
      PurchaseOrderType po = (PurchaseOrderType) poElement.getValue();

      USAddress address = po.getBillTo();
      address.setName("John Bob");
      address.setStreet("242 Main Street");
      address.setCity("Beverly Hills");
```

```
      address.setState("CA");
      address.setZip(new BigDecimal("90210"));

      // Marshal to XML
      Marshaller m = context.createMarshaller();
      m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
      m.marshal(po, System.out);
    } catch (JAXBException e) {
      e.printStackTrace();
    }
  }
}
```

Marshal Create Example

這個範例就是我們自己產生 PurchaseOrderType 的物件透過 JAXB 的 Marshaller 幫我們產生 XML Document
主要是透過 ObjectFactory 來產生 primer.po 裡如 PurchaseOrderType、USAddress 等物件，然後使用這些物件，最後還是透過 Marshaller 來將 Java 物件產生 XML Document。
在這個範例中，有一個小地方就是 po.xsd 產生的 Class 的日期類別是 javax.xml.datatype.XMLGregorianCalendar。在 datatype 這個 package 也可以看到有一個 DataTypeFactory 的 Class，所以我們可以用下列程式產生 XMLGregorianCalendar 物件。

```
XMLGregorianCalendar theDate =
DatatypeFactory.newInstance().newXMLGregorianCalendarDate(2007, 7, 30,
TimeZone.LONG);
```

當然 newXMLGregorianCalendarDate 有很多可以選擇，這邊是隨便找一個最簡單的來做展示用。

完整程式碼如下

```
package example;

import java.math.BigDecimal;
import java.util.List;
import java.util.TimeZone;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.datatype.DatatypeConfigurationException;
import javax.xml.datatype.DatatypeFactory;
import javax.xml.datatype.XMLGregorianCalendar;
import primer.po.Items;
import primer.po.ObjectFactory;
import primer.po.PurchaseOrderType;
import primer.po.USAddress;

public class MarshalCreate {
  public static void main(String[] args) {
    try {
      JAXBContext ctx = JAXBContext.newInstance("primer.po");

      ObjectFactory objFactory = new ObjectFactory();
      PurchaseOrderType po = objFactory.createPurchaseOrderType();
```

```
      XMLGregorianCalendar theDate =
DatatypeFactory.newInstance().newXMLGregorianCalendarDate(2007, 7, 30,
TimeZone.LONG);
      po.setOrderDate(theDate);

      USAddress shipTo = createUSAddress(objFactory,
                                         "Alice Smith",
                                         "123 Maple Street",
                                         "Cambridge",
                                         "MA",
                                         "US",
                                         "12345" );

      po.setShipTo( shipTo );

      USAddress billTo = createUSAddress(objFactory,
                                         "Robert Smith",
                                         "8 Oak Avenue",
                                         "Cambridge",
                                         "MA",
                                         "US",
                                         "12345" );

      po.setBillTo( billTo );

      Items items = objFactory.createItems();
      List<Items.Item> itemList = items.getItem();
      itemList.add(createItem(objFactory,
                              "Nosferatu - Special Edition (1929)",
                              5,
                              new BigDecimal("19.99"),
                              null,
                              null,
                              "242-NO"));
      itemList.add(createItem(objFactory,
                              "The Mummy (1959)",
                              3,
                              new BigDecimal("19.98"),
                              null,
                              null,
                              "242-MU"));
      itemList.add(createItem(objFactory,
                              "Godzilla and Mothra: Battle for Earth/Godzilla
vs. King Ghidora",
                              3,
                              new BigDecimal("27.95"),
                              null,
                              null,
                              "242-GZ"));

      po.setItems(items);

      Marshaller m = ctx.createMarshaller();
      m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
      m.marshal(po, System.out);
    } catch (JAXBException e) {
      e.printStackTrace();
    } catch (DatatypeConfigurationException e) {
```

```
      e.printStackTrace();
    }
  }

  private static USAddress createUSAddress(
                 ObjectFactory objFactory,
                 String name,
                 String street,
                 String city,
                 String state,
                 String contry,
                 String zip) {
    USAddress address = objFactory.createUSAddress();

    address.setName(name);
    address.setStreet(street);
    address.setCity(city);
    address.setState(state);
    address.setCountry(contry);
    address.setZip(new BigDecimal(zip));

    return address;
  }

  private static Items.Item createItem(
                 ObjectFactory objFactory,
                 String productName,
                 int quantity,
                 BigDecimal price,
                 String comment,
                 XMLGregorianCalendar shipDate,
                 String partNum) {
    Items.Item item = objFactory.createItemsItem();

    item.setProductName(productName);
    item.setQuantity(quantity);
    item.setUSPrice(price);
    item.setComment(comment);
    item.setShipDate(shipDate);
    item.setPartNum(partNum);

    return item;
  }
}
```

Unmarshal Validate Example

在 JAXB 1.0 是使用 Unmarshal 的 setValidate(boolean)的 method 來設定 validation；但是那個 method 在 JAXB 2.0 已經被 deprecated 了。在 JAXB 2.0 我們可以透過 javax.xml.validateion.Schema 來作 validate。根據 JavaDoc 可以看到那個 package 還有 SchemaFactory 這個 class，人家都明白表示我們當然是要用 Factory Class 來生產物件。因此最重要的程式片斷如下

```
Unmarshaller u = ctx.createUnmarshaller();
SchemaFactory schemaFactory =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
```

```
Schema schema = schemaFactory.newSchema(new File("bin\\po.xsd"));
u.setSchema(schema);
```

如果 Validate 沒有通過就會 catch 到 UnmarshalException。

完整程式碼如下

```
package example;

import java.io.File;
import javax.xml.XMLConstants;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBElement;
import javax.xml.bind.JAXBException;
import javax.xml.bind.UnmarshalException;
import javax.xml.bind.Unmarshaller;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import org.xml.sax.SAXException;
import primer.po.PurchaseOrderType;

/**
 * 在JAXB 2.0的Validate的方法<br>
 * 要透過Schema來做，因此我們透過SchemaFactory來取得Schema物件<br>
 *
 * @author Terry
 *
 */
public class UnmarshalValidate {
  public static void main(String[] args) {
    try {
      JAXBContext ctx = JAXBContext.newInstance("primer.po");

      Unmarshaller u = ctx.createUnmarshaller();
      SchemaFactory schemaFactory =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
      Schema schema = schemaFactory.newSchema(new File("bin\\po.xsd"));
      u.setSchema(schema);

      JAXBElement<?> poElement = (JAXBElement<?>) u.unmarshal(new
File("bin\\po_validate.xml"));
      PurchaseOrderType po = (PurchaseOrderType) poElement.getValue();

      System.out.println(po.getBillTo());
    catch(UnmarshalException e) {
      System.out.println("Caught UnmarshalException");
      e.printStackTrace();
    } catch(JAXBException e) {
      e.printStackTrace();
    } catch(SAXException e) {
      e.printStackTrace();
    }
  }
}
```

Reference

The Java EE 5 Tutorial from java.sun.com