Import/Export Tutorial

0.1 (18/07/03)

by Dieter Wimberger

1. General

The import/export basically is a procedure that generates/serializes a *Contact* instance from/to a serialized form. This is often also called marshalling respectively unmarshalling. We will alternatetively use the terms import/unmarshalling and export/marshalling throughout the text.

Marshalling/unmarshalling is basically done via the generic *net.wimpi.pim.contact.io* package. This package contains two interfaces defining the contract for marshalling and unmarshalling:

1. **net.wimpi.pim.contact.io.ContactMarshaller**: For exporting contacts
2. **net.wimpi.pim.contact.io.ContactUnmarshaller**: For importing contacts

Available implementations reside in subpackages of the same package; for the moment there is only the vCard implementation (see *net.wimpi.pim.contact.io.vcard*).

The way can you import/export a *Contact* is always the same, equal which flavor you want to use:

The first step is to obtain the factory. At the moment you will only be able to obtain the default factory (the vCardIOFactory); however, it is likely that in the future you can state a flavor when obtaining the factory instance:

```
ContactIOFactory ciof = Pim.getContactIOFactory();
```

Then you can create marshaller/unmarshaller instances using the respective factory methods:

```
ContactUnmarshaller unmarshaller = ciof.createContactUnmarshaller();
```

or

```
ContactMarshaller marshaller = ciof.createContactMarshaller();
```

Optionally you can set an encoding for the I/O:

```
unmarshaller.setEncoding("UTF-8");
```

or

```
marshaller.setEncoding("UTF-8");
```

Last step to import the contact is to unmarshall it from an **InputStream** (in in the example):

```
Contact contact = unmarshaller.unmarshallContact(in);
```

or to marshall it to an **OutputStream** (out in the example):

```
marshaller.marshallContact(out, contact);
```

2. Importing/Exporting vCards

jpim contains a vCard implementation, that is capable of importing v2.1 and v3.0 (vCard Mime Directory Profile) vCards, as well as exporting v3.0 vCards.
See the specifications for more details.

The vCard implementation supports marshalling and unmarshalling of custom extensions. You should make yourself familiar with following part of the model:

- **net.wimpi.pim.contact.model.Extension**: The interface defining an extension so that it can be generically handled.
- **net.wimpi.pim.contact.model.Extensions**: The interface defining a collection of extensions, that can handle multiple extensions, as well as extensions with multiple items (same identifier).

The package *net.wimpi.pim.contact.basicimpl* provides basic implementations for the above mentioned interfaces, as well as an abstract **GenericExtension** that can be extended and comes already with an approbiate handler for the vCard entries.
To make it even simpler, there is a **SimpleExtension** class, which allows to add simple extensions very easily.

2.1. Marshalling a vCard with an Extension

First you will need to add an **Extension** collection instance to your **Contact** instance (*contact* in the example). As recommended, you should use the model factory (*cmf* in the example) to do so:

```
Extensions extensions = cmf.createExtensions();
contact.setExtensions(extensions);
```

Next you have to create the simple extension and add it to the *extensions* collection:

```
//create a simple extension
SimpleExtension ext = new SimpleExtension("X-KIDS");
//add some values
ext.addValue("Mary");
ext.addValue("Joe");
ext.addValue("Tom");
ext.addValue("Frank");
//add extension to the collection
```

```
extensions.add(ext);
```

The last step to make the marshalling work, is to register the extension with the vCard I/O implementation. This is done via the Singleton **ItemHandlerManager** instance:

```
ItemHandlerManager.getReference().addExtensionHandler(
    ext.getIdentifier(),
    new GenericExtensionItemHandler(ext)
);
```

That's it. When you marshall the *Contact* instance after this, the extension will be properly added to the vCard.

2.2. Unmarshalling a vCard with an Extension

Unmarshalling a vCard with extensions is as easy as marshalling it. All that needs to be done, is to register the approbiate extension with the vCard I/O implementation, via the Singleton **ItemHandlerManager** instance:

```
//the simple extension kids
SimpleExtension ext = new SimpleExtension("X-KIDS");
//add the handler
ItemHandlerManager.getReference().addExtensionHandler(
    ext.getIdentifier(),
    new GenericExtensionItemHandler(ext)
);
```

That's it. When you now unmarshall a vCard with the given extension, you can retrieve it via the *Extensions* collection of the *Contact* instance.

3. A Working Example

To complete the import/export tutorial part, here some example classes that show how to marshall a *Contact* instance and how to unmarshall a vCard and marshall it again to *System.out*:

Create a new *Contact* instance and marshall it to *System.out*:

```
import net.wimpi.pim.Pim;
import net.wimpi.pim.contact.basicimpl.SimpleExtension;
import net.wimpi.pim.contact.io.ContactMarshaller;
import net.wimpi.pim.contact.io.vcard.GenericExtensionItemHandler;
import net.wimpi.pim.contact.io.vcard.ItemHandlerManager;
import net.wimpi.pim.contact.model.Address;
import net.wimpi.pim.contact.model.Communications;
import net.wimpi.pim.contact.model.Contact;
import net.wimpi.pim.contact.model.EmailAddress;
import net.wimpi.pim.contact.model.Extensions;
```

```
import net.wimpi.pim.contact.model.Organization;
import net.wimpi.pim.contact.model.OrganizationalIdentity;
import net.wimpi.pim.contact.model.PersonalIdentity;
import net.wimpi.pim.contact.model.PhoneNumber;
import net.wimpi.pim.factory.ContactIOFactory;
import net.wimpi.pim.factory.ContactModelFactory;

public class createvCard {

  public static void main(String[] args) {
    try {
      ContactModelFactory cmf = Pim.getContactModelFactory();

      Contact contact = cmf.createContact();
      contact.addCategory("Business");

      //a personal identity
      PersonalIdentity pid = cmf.createPersonalIdentity();
      pid.setFormattedName("Frank Miller");
      pid.setFirstname("Frank");
      pid.setLastname("Miller");
      contact.setPersonalIdentity(pid);

      //an address
      Address addr = cmf.createAddress();
      addr.setStreet("Somestreet");
      addr.setCity("Somecity");
      addr.setPostalCode("55555");
      addr.setCountry("United States");
      addr.setWork(true);
      contact.addAddress(addr);

      //some communications
      Communications comm = cmf.createCommunications();
      contact.setCommunications(comm);

      //a phone number
      PhoneNumber number = cmf.createPhoneNumber();
      number.setNumber("+1(55)555-4575");
      number.setWork(true);
      comm.addPhoneNumber(number);
      //an email address
      EmailAddress email = cmf.createEmailAddress();
      email.setAddress("somebody@aol.com");
      comm.addEmailAddress(email);

      //an organizational identity
      OrganizationalIdentity orgid = cmf.createOrganizationalIdentity();
      orgid.setRole("Unit Manager");
      orgid.setTitle("Boss");
      Organization org = cmf.createOrganization();
      org.setURL("http://www.wimpi.net");
      org.setName("Wimpi Inc.");
      org.addUnit("IT");
```

```
      org.addUnit("Software Development");
      orgid.setOrganization(org);
      contact.setOrganizationalIdentity(orgid);

      //some simple extension
      Extensions extensions = cmf.createExtensions();
      SimpleExtension ext = new SimpleExtension("X-KIDS");
      ext.addValue("Mary");
      ext.addValue("Joe");
      ext.addValue("Tom");
      ext.addValue("Frank");
      extensions.add(ext);

      //add the handler, so the marshalling will work
      ItemHandlerManager.getReference().addExtensionHandler(
          ext.getIdentifier(),
          new GenericExtensionItemHandler(ext)
      );

      //add the extension to the contact
      contact.setExtensions(extensions);


      //and now write the vCard to standard out
      ContactIOFactory ciof = Pim.getContactIOFactory();
      ContactMarshaller marshaller = ciof.createContactMarshaller();
      marshaller.marshallContact(System.out, contact);

    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }//main

}//class createvCard
```

This will produce the following vCard:

```
BEGIN:VCARD
VERSION:3.0
N:Miller;Frank;;;
FN:Frank Miller
ADR;TYPE=WORK:;;Somestreet;Somecity;;55555;United States
TEL;TYPE=WORK:+1(55)555-4575
EMAIL;TYPE=INTERNET:somebody@aol.com
TITLE:Boss
ROLE:Unit Manager
ORG:Wimpi Inc.;IT;Software Development
CATEGORIES:Business
X-KIDS:Mary;Joe;Tom;Frank
X-ORG-URL:http://www.wimpi.net
END:VCARD
```

Read a vCard from a file and marshall it again to *System.out*:

```
import java.io.FileInputStream;

import net.wimpi.pim.Pim;
import net.wimpi.pim.contact.basicimpl.SimpleExtension;
import net.wimpi.pim.contact.io.ContactMarshaller;
import net.wimpi.pim.contact.io.ContactUnmarshaller;
import net.wimpi.pim.contact.io.vcard.GenericExtensionItemHandler;
import net.wimpi.pim.contact.io.vcard.ItemHandlerManager;
import net.wimpi.pim.contact.model.Contact;
import net.wimpi.pim.factory.ContactIOFactory;

public class readvCard {

  public static void main(String[] args) {
    try {
      ContactIOFactory ciof = Pim.getContactIOFactory();
      ContactUnmarshaller unmarshaller = ciof.createContactUnmarshaller();
      if((args.length == 2 && args[1] != null && args[1].length() == 0)) {
        unmarshaller.setEncoding(args[1]);
      }

      //Add handler for the simple extension kids
      SimpleExtension ext = new SimpleExtension("X-KIDS");
      //add the handler, so the marshalling will work
      ItemHandlerManager.getReference().addExtensionHandler(
          ext.getIdentifier(),
          new GenericExtensionItemHandler(ext)
      );

      //unmarshall contact
      Contact contact = unmarshaller.unmarshallContact(new FileInputStream(args[0]));

      //marshall it again
      ContactMarshaller marshaller = ciof.createContactMarshaller();
      marshaller.marshallContact(System.out, contact);

    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }//main

}//class readvCard
```