Model Tutorial

0.1 (18/07/03)

by Dieter Wimberger

1. General

The basic idea of jpim is to provide a library with the **M**odel element of an **MVC** architecture.

In case of the contact package, the model is fairly complex and has been modeled after the "vCard Mime Directory Profile" (see the specifications for more details).
The idea is a strong typed model (versus a property based implementation), but fairly complete, so many different types of applications can make use of it.

The model itself is only defined as interfaces, that specify the contract with the actual implementation. Best way to understand the different elements, is to take a closer look at the above mentioned vCard MIME Directory Profile.
When creating model instances, it is recommended to use the model factory. At the moment you will only be able to obtain the default factory (the BasicContactModelFactory); however, it is likely that in the future you can state a flavor when obtaining the factory instance:

```
ContactModelFactory cmf = Pim.getContactModelFactory();
```

> **Note:**
> You should be able to use your own factory and model implementations without problems. I/O implementations which are properly implemented against the model interfaces will be able to import/export your own implementations (like the vCard implementation for example).

Model instances should be obtained from the respective factory methods e.g.:

```
Contact contact = cmf.createContact();
```

2. An Example

The following example will create *Contact* instance with some data:

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


Copyright © 2001-2003 jpim Development Team All rights reserved.

```
//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");

//and the organization
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);
```

2.1. HELP! This seems complicated...

We agree that the expressivity of the model makes it *fairly complex*. A first thought would be to simplify it, but on the second thought we realize that this is probably not the right way to go (as we want the model to be applicable in many different applications, not some specific one).

Copyright © 2001-2003 jpim Development Team All rights reserved.

However, we are also aware that you would like to save yourself writing more code than necessary, so we thought that the best solution for the problem is the so called *Facade* pattern. Basically this is an implementation/interface that hides the complexity of the model, being taylored to the needs of a specific application.
If you have a specific need, you probably might need to implement a facade once, however, we consider this worth the work, as you will most likely be able to use other facilities provided by the library (like importing and exporting to vCards for example).

jpim comes with an example facade, the **import net.wimpi.pim.contact.facades.SimpleContact** class.
The simplest way to see what it does for you is to wrap the *Contact* instance from above and retrieve data more easily:

```
SimpleContact sct = new SimpleContact(contact);
String email = sct.getFullEmail();
String company = sct.getCompany()
```


Copyright © 2001-2003 jpim Development Team All rights reserved.