Persistency Tutorial

0.1 (18/07/03)

by Dieter Wimberger

1. General

Persistency should basically provide you with means to persist collections of *Contact* and *ContactGroup* instances.

> **Note:**
> This database might also be called an "Address Book", so if you are interested in something alike, don't miss the following. Read on!

Persistency is modeled by the generic *net.wimpi.pim.contact.db* package. This package contains six interfaces defining the contract for implementations:

1. **net.wimpi.pim.contact.db.ContactDatabase**: For the contact database
2. **net.wimpi.pim.contact.db.ContactCollection**: For a collection of *Contact* instances
3. **net.wimpi.pim.contact.db.ContactGroup**: For groups of contacts
4. **net.wimpi.pim.contact.db.ContactGroupCollection**: For a collection of *ContactGroup* instances
5. **net.wimpi.pim.contact.db.ContactFilter**: For a filter that is capable of filtering *Contact* instances
6. **net.wimpi.pim.contact.db.ContactGroupFilter**: For a filter that is capable of filtering *ContactGroup* instances

Available implementations reside in subpackages of the same package; for the moment there is only a Serialization based implementation (see *net.wimpi.pim.contact.db.serializable*).

The basic way you can utilize a *ContactDatabase* should be fairly the same for any implementation:

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

```
ContactDBFactory cdbf = Pim.getContactDBFactory();
```


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

Then you can create *ContactDatabase* instance using the respective factory method:

```
ContactDatabase ctdb = cdbf.createContactDatabase();
```

Now you can add *Contact* and *ContactGroup* instances to the *ContactDatabase*, list them (with and without filters etc.). Please see the API Documentation for more information on the operations.

Group instances can also be obtained from the **ContactDBFactory**:

```
ContactGroup group = ctdb.createContactGroup();
```

> **Note:**
> Implementations will differ most in the way you store and load the database instances. Please see the corresponding sections for more detail on implementations.

2. The Serializable Database

jpim contains a *ContactDatabase* implementation, that is serializable. It can be stored and retrieved (from and to streams) using the standard Java serialization mechanism.

2.1. Storing a Database Instance

You can serialize the database to any type of *OutputStream* instance:

```
FileOutputStream fout = new FileOutputStream(ctdb.getUID() + ".ser");
ObjectOutputStream out = new ObjectOutputStream(fout);
out.writeObject(ctdb);
```

2.2. Loading a Database Instance

You can de-serialize the database from any type of *InputStream* instance:

```
FileInputStream fin = new FileInputStream(filename);
ObjectInputStream in = new ObjectInputStream(fin);
ContactDatabase ctdb = (ContactDatabase) in.readObject();
```

> **Note:**
> Extensions will be properly serialized and de-serialized, if they are *Serializable*. The **SimpleExtension** implementation is an example.

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