Quick start guide for the **JEntigrator**.

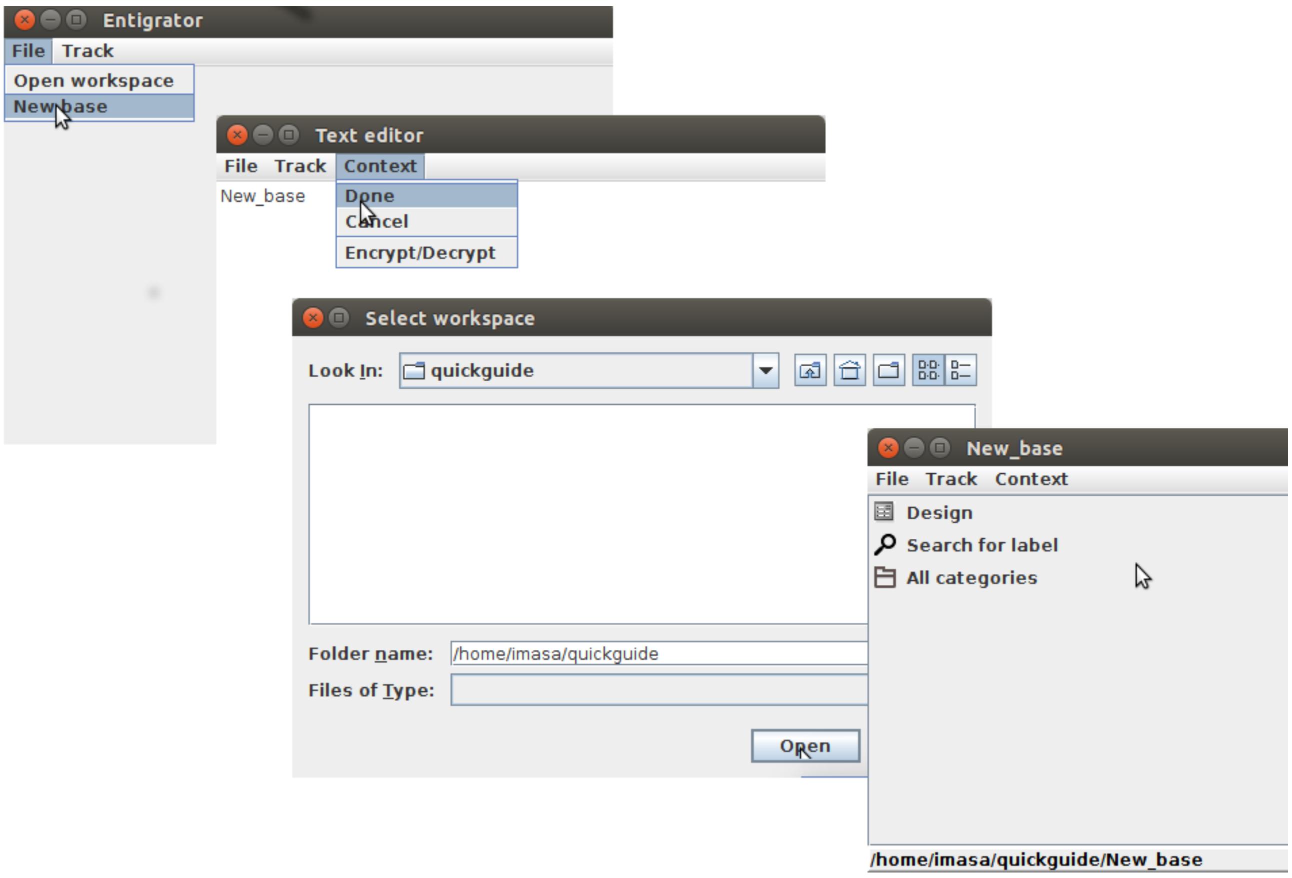
Alexander Imas, 2016, Darmstadt.

1.Requirements.

The **JEntigrator** is the single-user local file-based data management application implemented as a **java** program. It is distributed as the exectable **entigrator.jar** file**.** You need the **Java7** runtime environment to start and use the application or the **Java7 JDK** to maintaine and adapt the application. The **JEntigrator** is the open source program published under **GNU GPL** licence on the **GitHub** platform. You can start the application from the terminal/DOS window using the command line: **java -jar entigrator.jar.**

2.First start.

The main application's window appears after the start. Use the menu item **File -> New base** to create a new database. The application proposes the name of the new database **New_base** in the text editor windiw. Accept the name and click the **Done** menu item in the **Context** menu. Select the directory to save the database in the **FileChooser** dialog. The applications creates the new database and displays the **BaseNavigator** in the context panel.

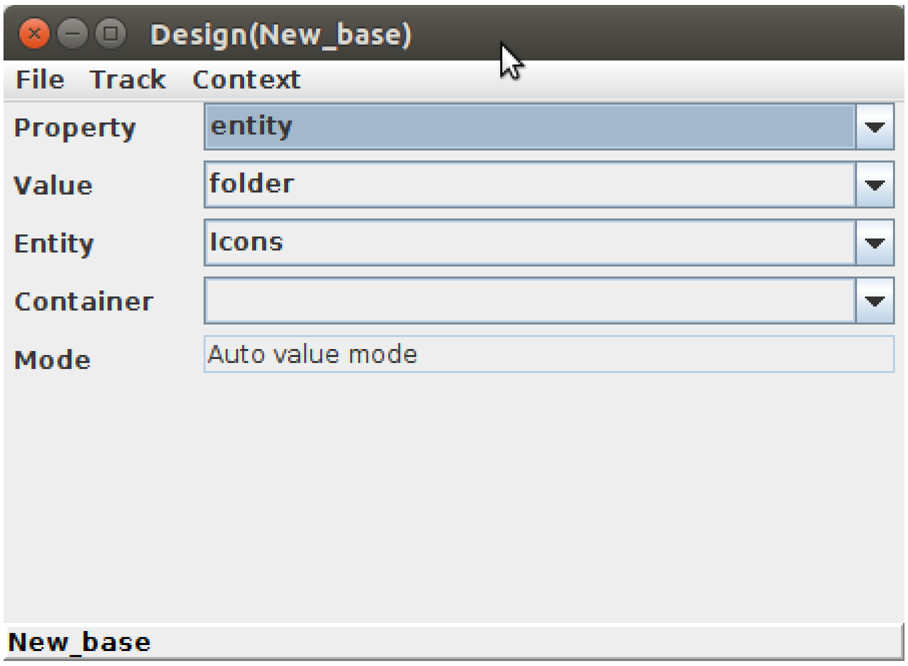


Now we found out that the main application window is a frame containing a menu bar and the context panel. The menu bar contains two permanent menus **File** and **Track** and the changeable menu **Context.** The context panel is a placeholder. It's content depends on the user's activity. The **Context** menu depends on the current context. The user interacts with the application through the menus and contorls in the context panel.

3.Explore the new database.

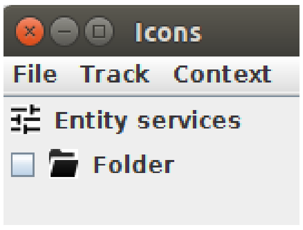
The **JEntigrator** provides some ways to get access to the content of the database. The **BaseNavigator** context panel lists three of them: **Design, Search for label** and **All categories** items. Click on the items puts its context in the context panel.

The **Design** context displays the multifunctional contol panel for low-level management of the database.

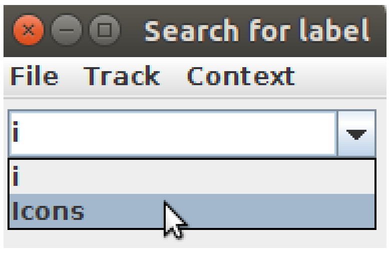


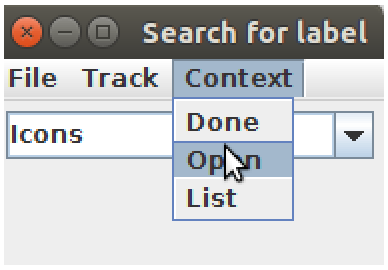
Property: entity
Value: folder
Entity: Icons
Container:
Mode: Auto value mode

We know, that each database in the **JEntigrator** contains some unified data objects called entitties. Each entity has two mandatory properties: **entity,** containing the type of the entity, and the **label,** containing the unique name of the entity. The **Property** selector keeps a list of all properties of all entities in the database. The **Value** selector shows all values of the selected property. The **Entity** selector contains a list of all entities having the selected value assigned. The **Container** selector shows the list of all containers of the selected entity if exist. In our case the entity **Icons** of the type **folder** is selected. Selection of entities is an important but not sole feature of the **Design** control. We don't want to consider them here to keep explanations short. Now we can click on the **Entity** label and display the **FacetsPanel** context for the selected entity.

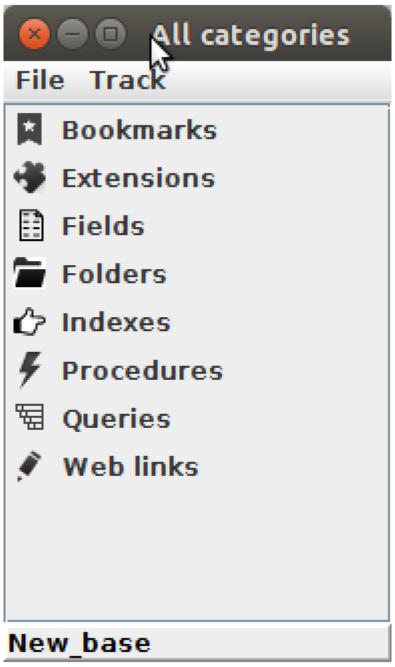


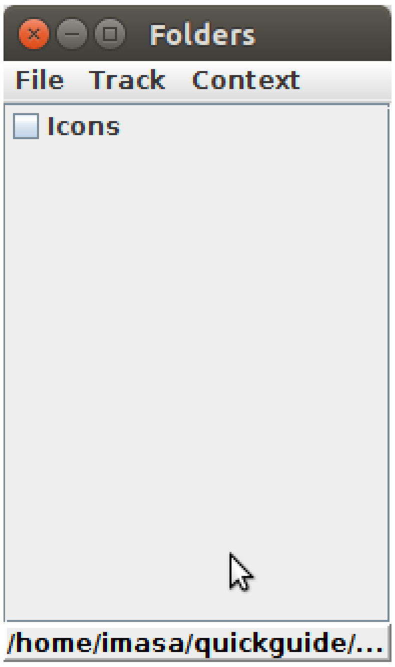
The **Search for label** context exposes the autocomplete textbox. The user types first letters of the label and the textbox exposes the drop-down list of suggested labels to select the entity. The click on the **Open** menu item shows the **FacetsPanel** context for the selected entity.





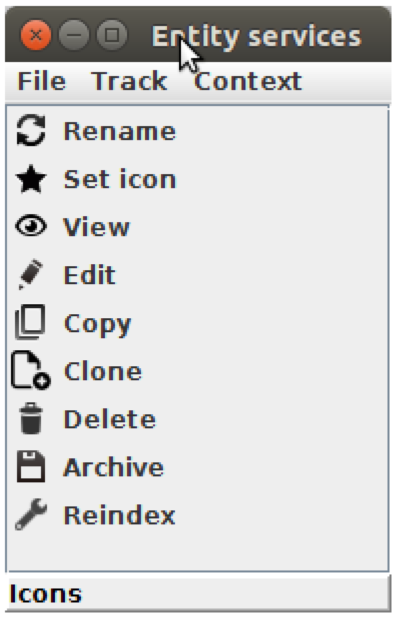
The **All categiries** context displays the list of all possible entity's types in the database. The **New_base** contains only one entity **Icons** of the **folder** type. Click on the **Folder** type to show the entity **Icons** in the **Category panel** context. Click on the item **Icons** exposes again the **FacetsPanel** context for the entity

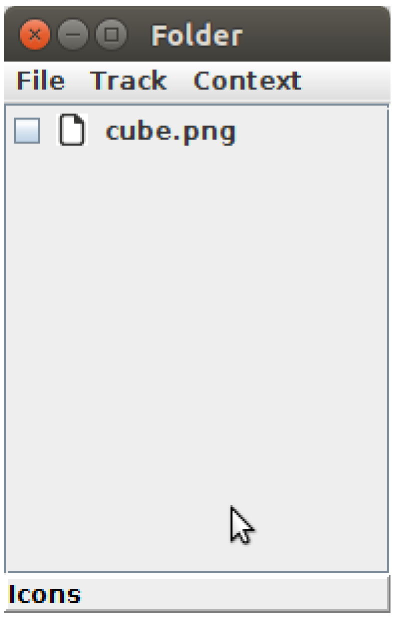




4.Explore an entity.

An entity in the **JEntigrator** is defined as a collection of **facets.** A **facet** is a composition of two components: data and program. The data component is a collection of some fields within entity and the program component (the facet handler) is a code within the **JEntigrator** processing this data. The **FacetsPanel** context lists all facets assigned to the entity. Each entity has at least one mandatory facet **Entity services** that collects standard operations independent on type or content of the entity. The click on the facet item displays its context panel. The picture below shows entity services context. The click om the service item displays its context. The **Done** menu item from the **Context** menu close the context.

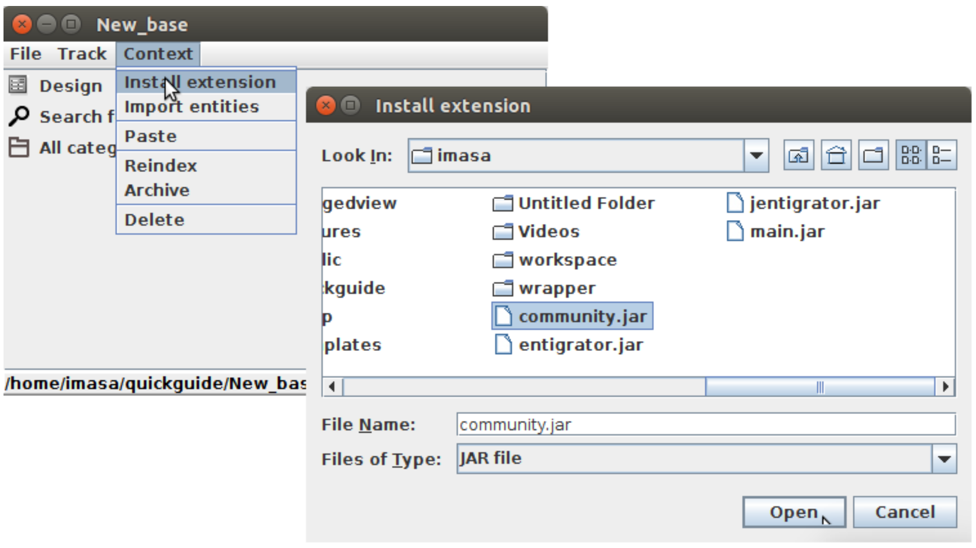


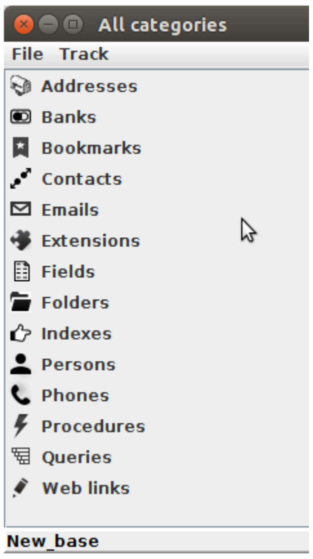


4.Extend the database.

The **JEntigrator** application originally includes only general-purpose facets. In order to be really useful the application should be adapted to the user's busines area. That can be done using extensions. An extension is a special entity of the **extension** type containing a library with with new facet handlers and descriptions of these facets. The extension can be imported as any other data object into the database or by starting of the executable jar file from the **JEntigrator.** The **community.jar** file extends the database with contact facets.

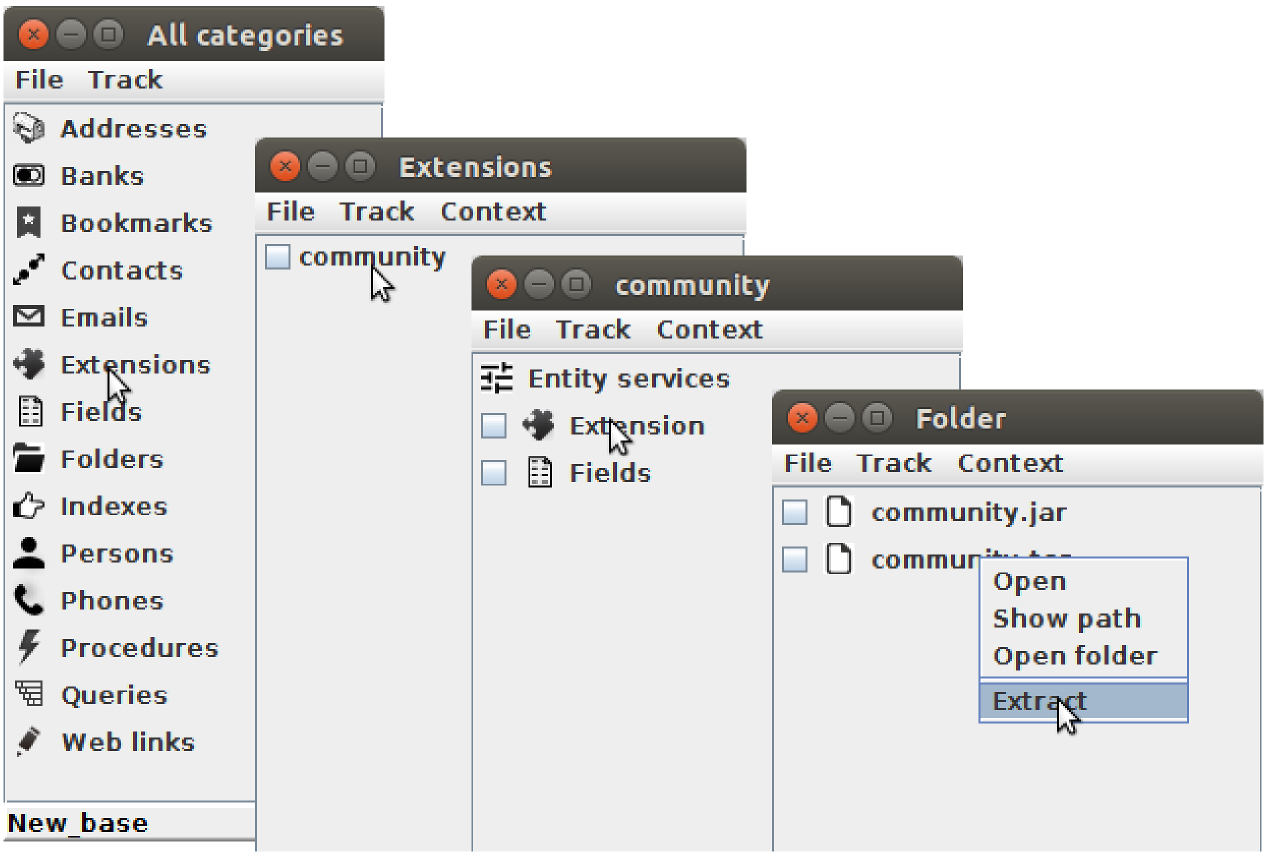
The installation of othe extension can be done from the **BaseNavigator** context. Click on the **Install extension** menu item in the **Context** menu. Select the **community.jar** file in the **File chooser** dialog and click the **Open** button. Close the application and start the **JEntigrator** again. In the **All categories** context you can see now some new entity types.



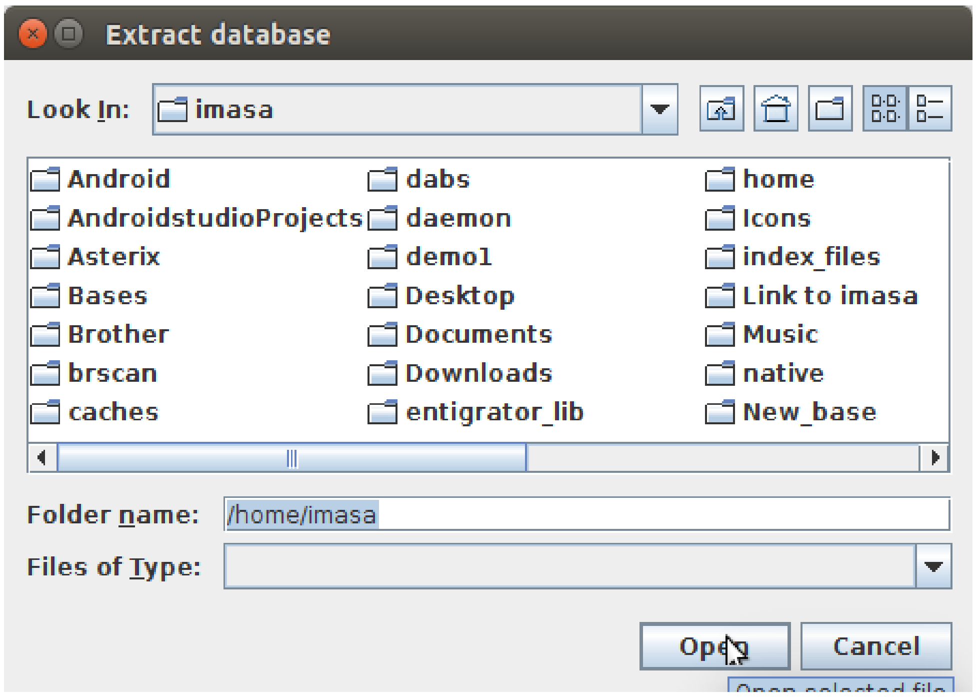


5. Install the sample database **community.**

The **community** extension contains the template database **communicty.** Its archive file **community.tar** is located in the extension folder. Navigate to the content of **community** extension and right click the **community.tar** item. Then the left click on the **Extract** menu item from the context menu.

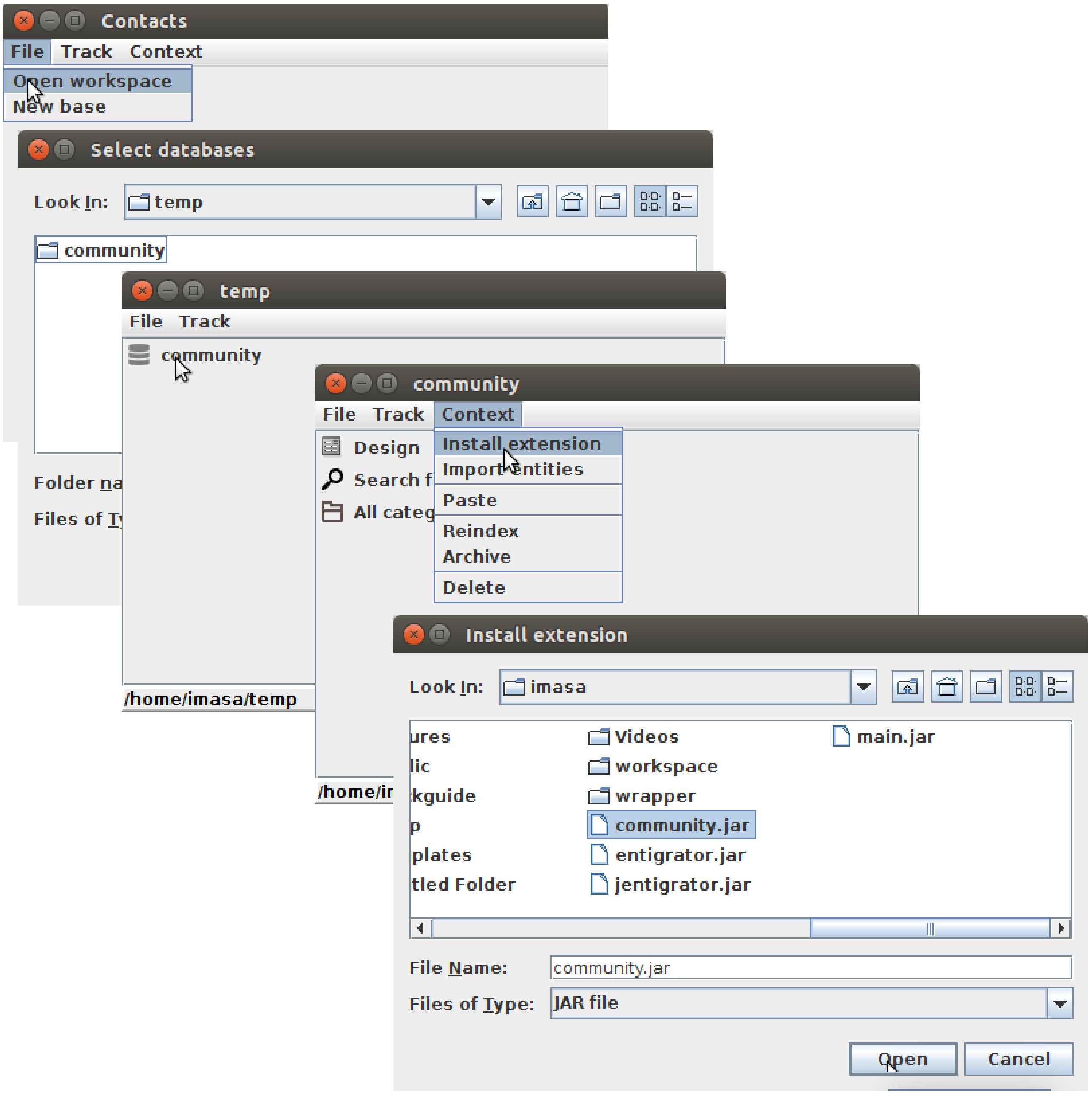


Select the installation directory in the **FileChooser** dialog and click the **Open** button.



Now we shall open the **community** database and install the extension there.

- **File** → **Open workspace** → **Open:** select the directory containing the **community** database in the **FileChooser**
- Click the **community** item in the **DatabasesList** context: open the **BaseNavigator** context.
- **Context** → **Install extension** → **Open:** select the **community.jar** extension in the **FileChooser** and install it in the **community** database.

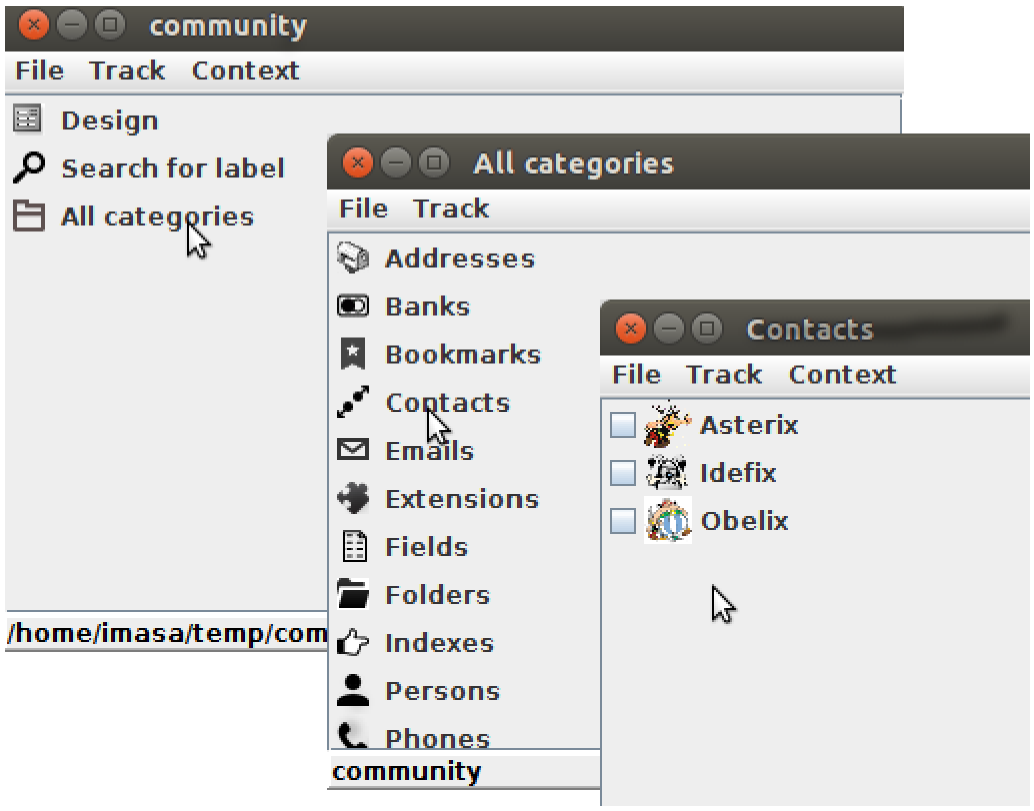


6. Explore the content of the database.

Now it's time to do an overview tour on the content of the **community** database. The basic entity type there is **contact.** A contact must have the mandatory unique name (the **label** of the entity) and may have some other facets reflecting its properties. These facets may be embedded into the contact entity as properties or tied to the contact as components.

6.1. List contacts in the **CategoryPanel** context.

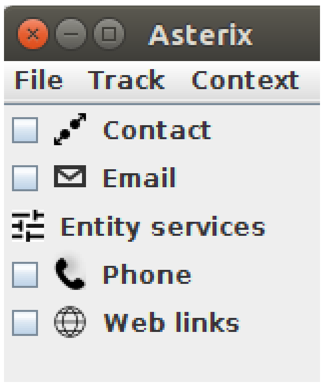
Open the **BaseNavigator** context of the **community** database as described above. Click the **All categories** item. Click the **Contacts** category item.



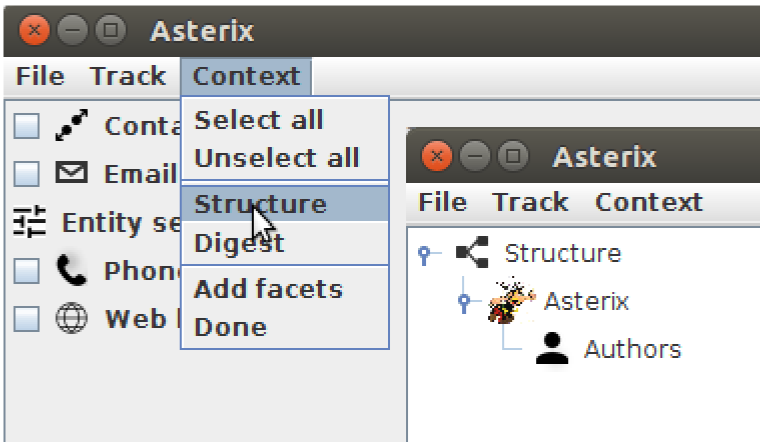
The database contains three sample contacts **Asterix**, **Idefix** and **Obelix.**

6.2. Explore the contact.

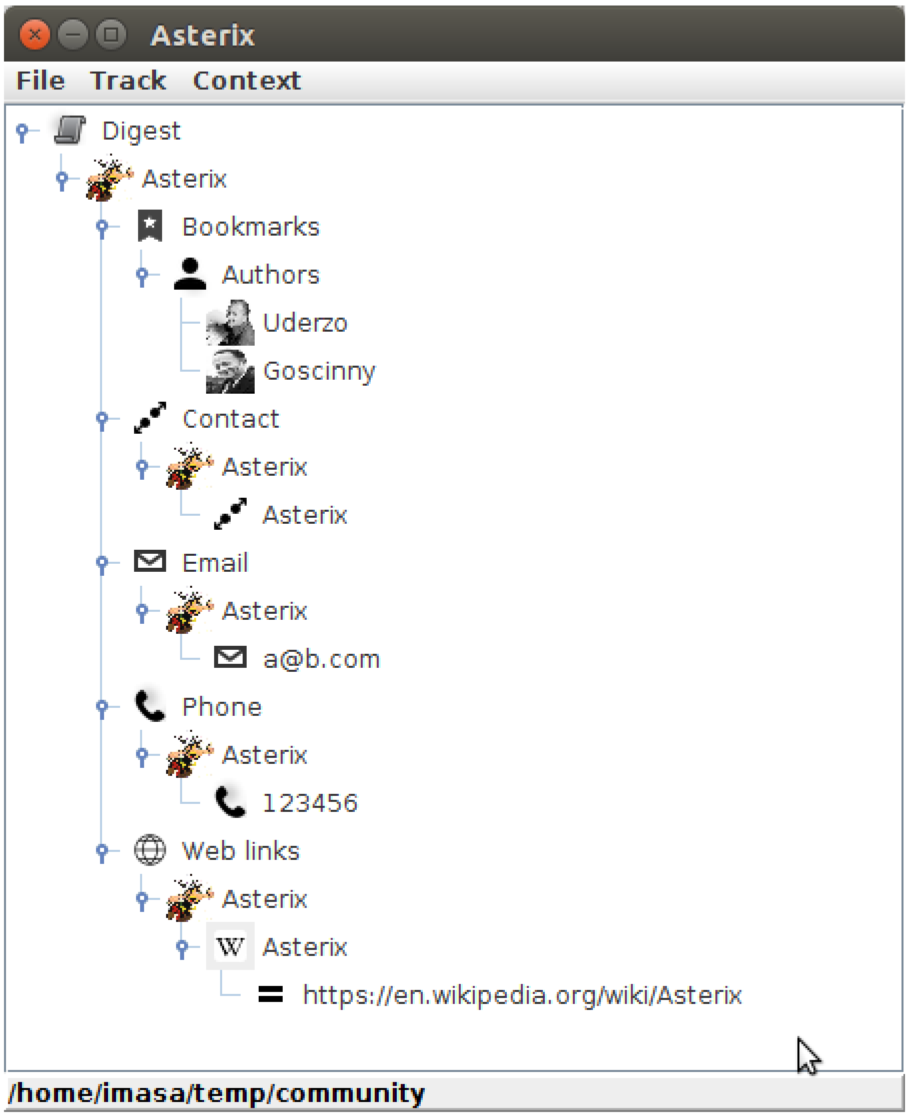
An entity in the application is defined as a set of facets. So if you click on the contact item in the list you switch into the **FacetsPanel** context . The **FacetsPanel** exposes the list of all facets assigned to the entity.



There are another two contexts displaying an entity: **Structure** and **Digest.** The **Structure** shows the container-components tree for the entity. The **Digest** shows all properties including components. We can switch to another context using the **Context** menu.

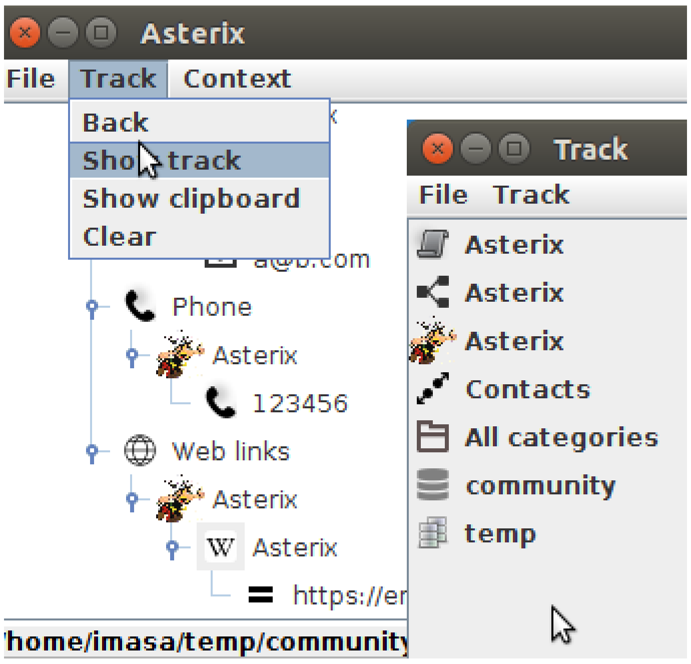


We can see that the **Asterix** has one component **Authors.** An entity may have many containers. So you can check that the **Authors** entity is a component of **Idefix** and **Obelix** too. Click the **Context** menu item **Digest** to switch into the **DigestPanel** context. The **Digest** collects all facet properties of all components and provides the overview of whole information about the **Asterix.**



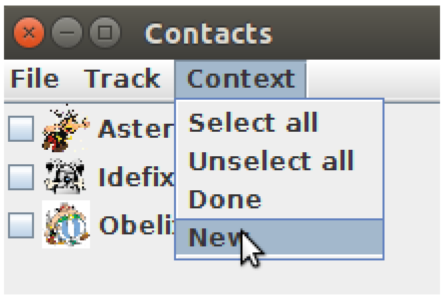
7. Browsing the database .

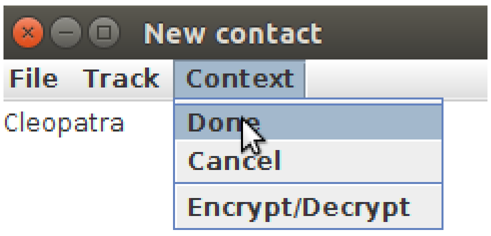
The **GUI** of the **JEntigrator** is built as a set of context panels. A user switches between contexts in order to get access to contols needed for the current task. Practical experience shows that is often necessary to return to already shown contexts. The **Track** menu makes this work easier. The **Back** menu item displays the previous context. The **Show track** menu item lists the serie of previous contexts in the hierarchical order. The click on the list item removes items above the selection and displays the selected context. This policy keeps the track short and clear. The **Clear** menu item clears the track. Alternatively you can any time open the database from the File menu and navigate using the **BaseNavigator** context. The **JEntigrator** saves track on exit and shows the last used context on start.



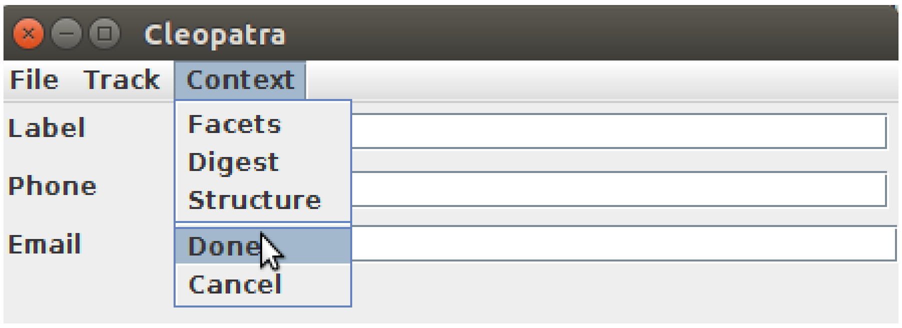
8. Facet management.

An entity evolves in the way that facets can be added or removed from it. To illustrate the procedure we create a new contact and assign facets to it. Navigate to the **Contacts** category context as described above and click the **New** menu item from the **Context** menu. Edit the name in the **NewContact** context and click **Done.**

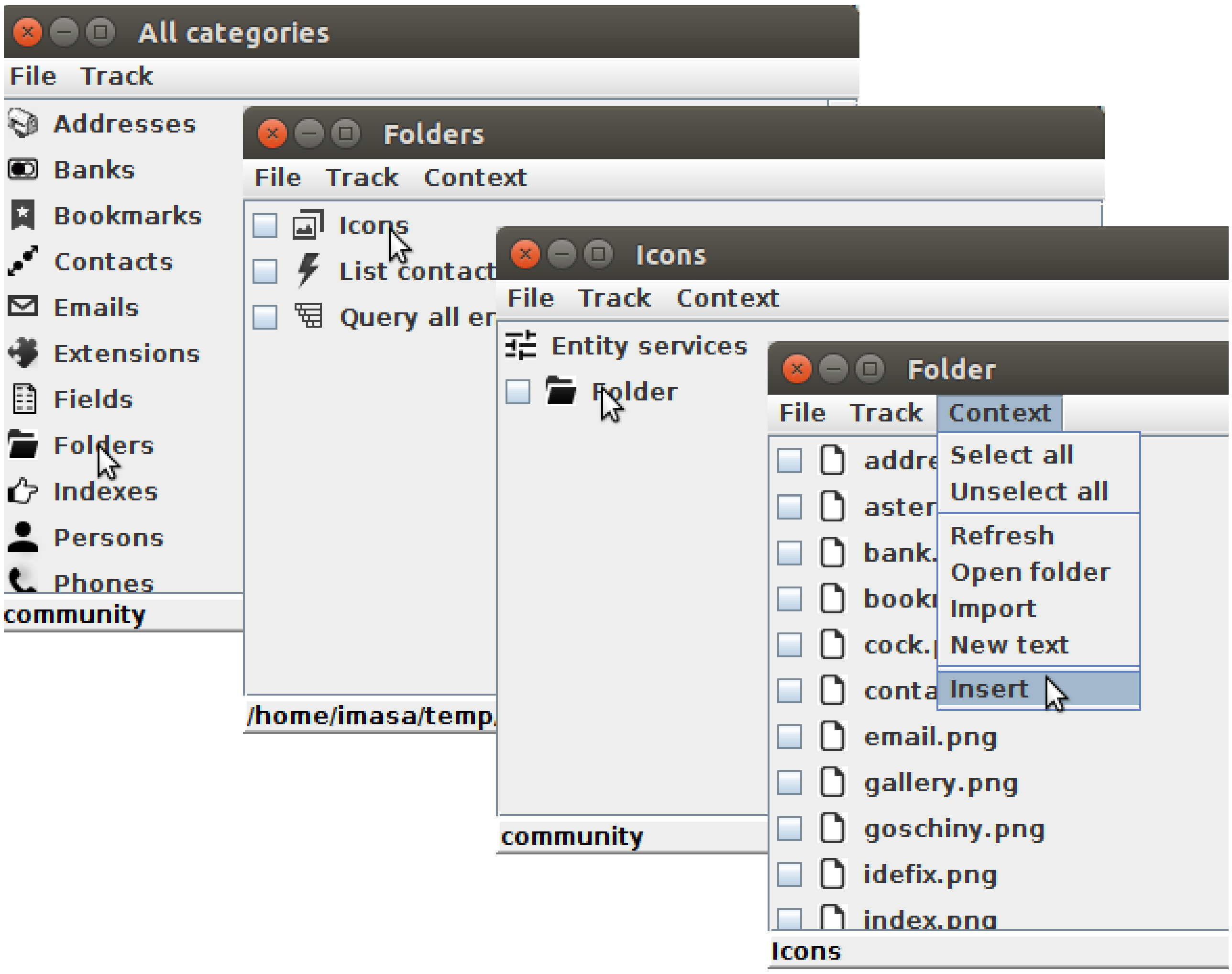




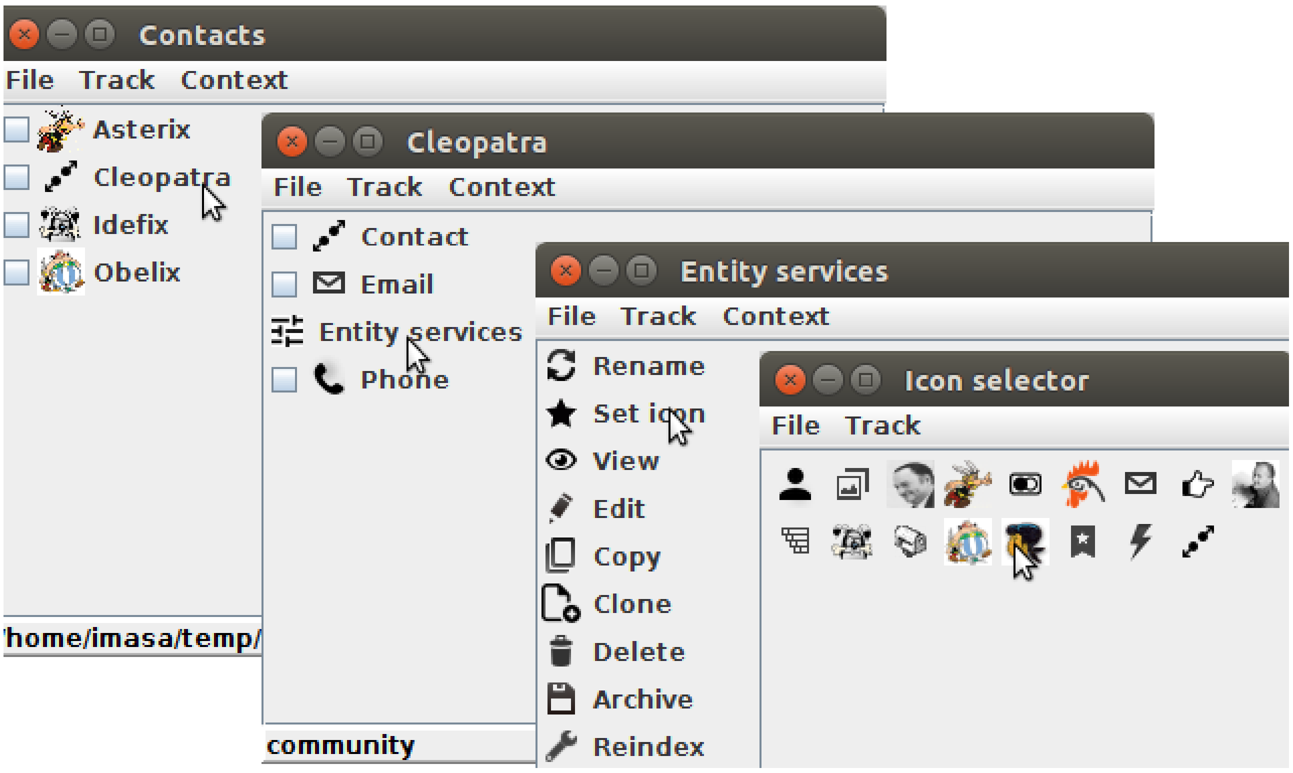
The **JEntigrator** assigns the **phone** and **email** facets to the new contact on default. Edit these values and the **Done** menu item.



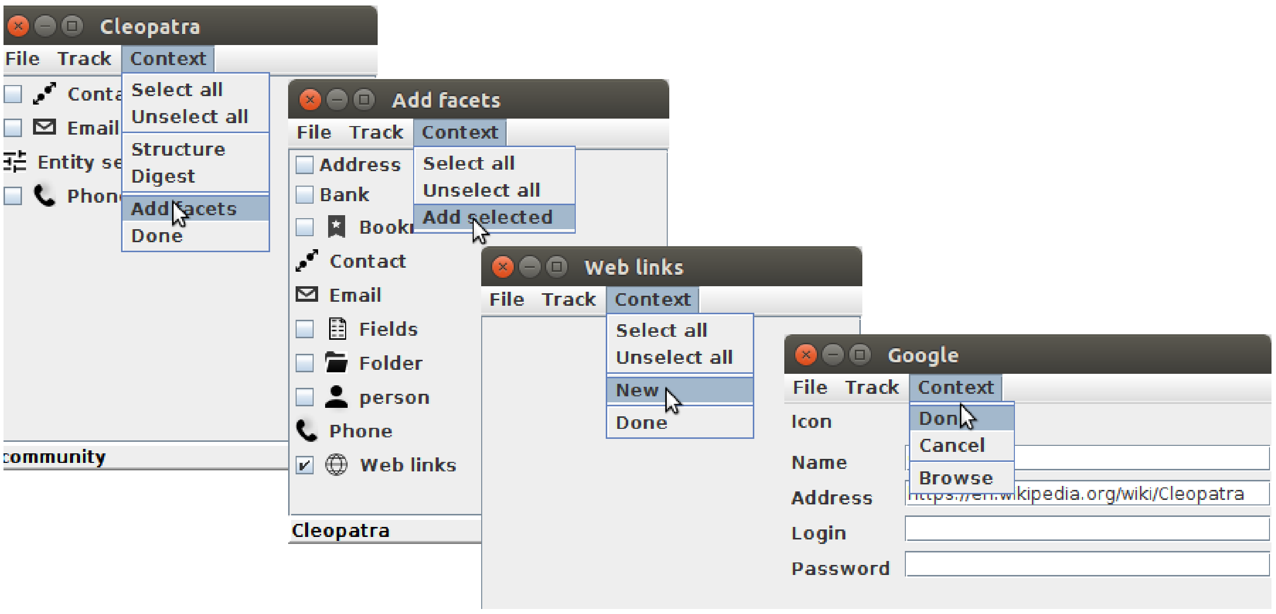
You may want to assign an icon to the **Cleopatra**. Assume the icon is already copied into the system clipboard. You should paste it into the folder **Icons** in the database. The picture below shows how to open the folder in the system file manager and paste the file there.



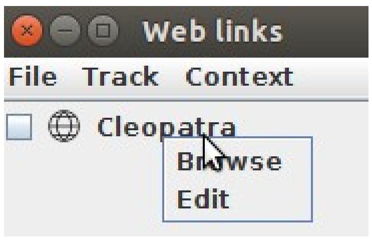
Alternatively you can use the **Open folder** menu item to open the system file manager and paste the icon there. Now it is possible to assign the item to the **Cleopatra.** Navigate to **CategoryPanel** or contacts. The next actions are shown on the picture below.



The next step is to add the **WebLinks** facet to the **Cleopatra.** Navigate to the **FacetsPanel** of the **Cleopatra.** The next actions are shown on the picture below.

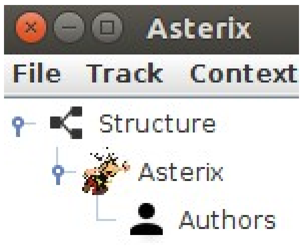


You have to type the name **Cleopatra** and the address **https://en.wikipedia.org/wiki/Cleopatra** in the form fields before **Done.** Any number other links can be added to the entity. To access the web link actions use the right click on the item.

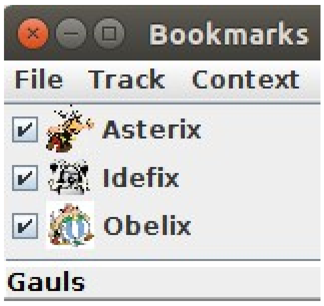


9. Associations.

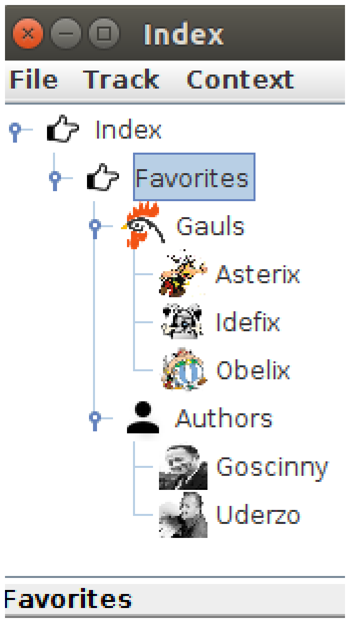
The **JEntigrator** provides some ways to unite data in groups. The container-component relation gives the possibility to create complex entities. For example **Authors** is the component of **Asterix.**



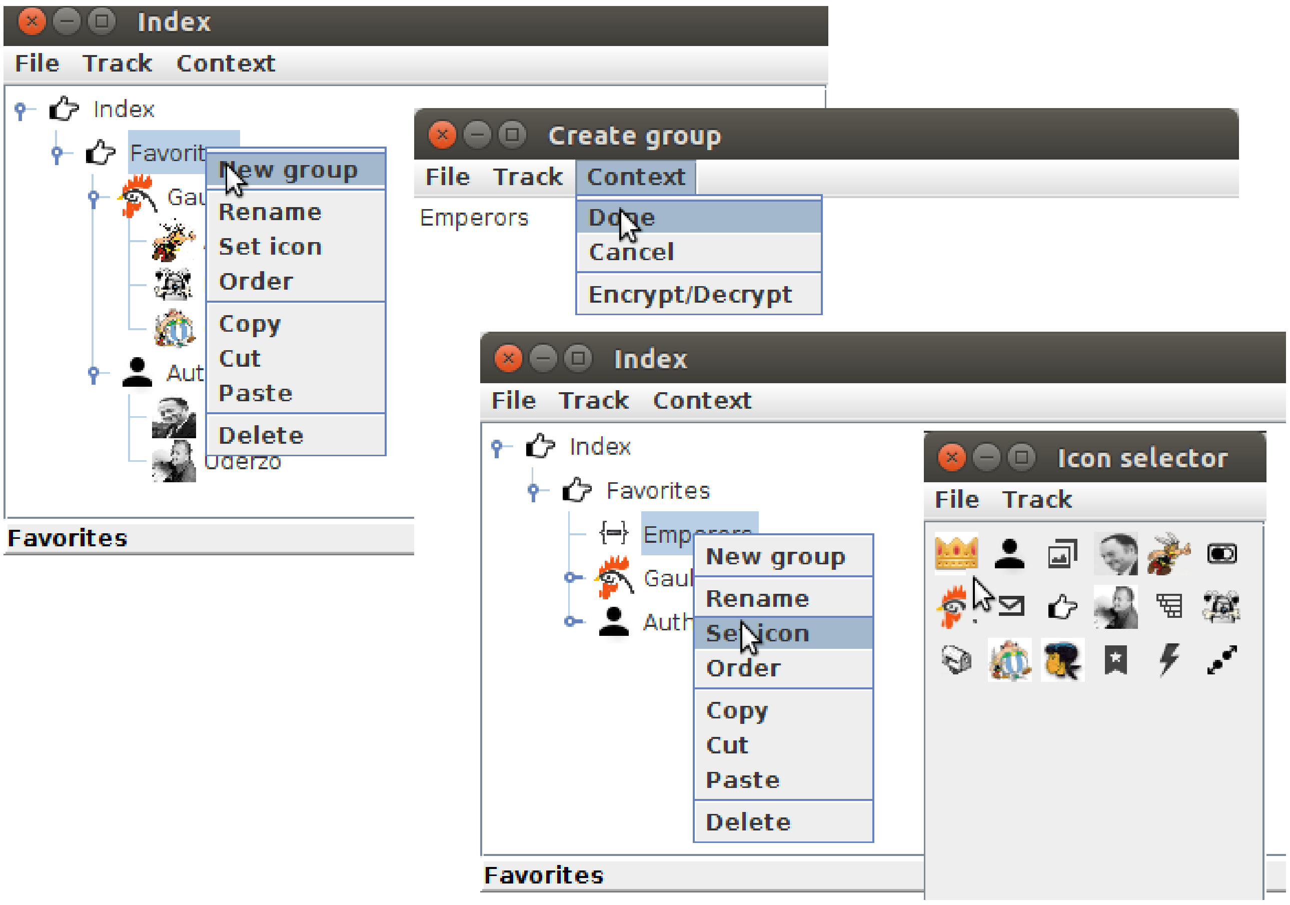
A **bookmark** facet makes an entity to be a collection of links to entities or/and files within the database. It provides quick access to the certain data selection.



An **index** is a tree structure of groups containing another groups and links to entities or/and files within the database. A user create indexes according his/her business logic. There can be multiple indexes in the database.

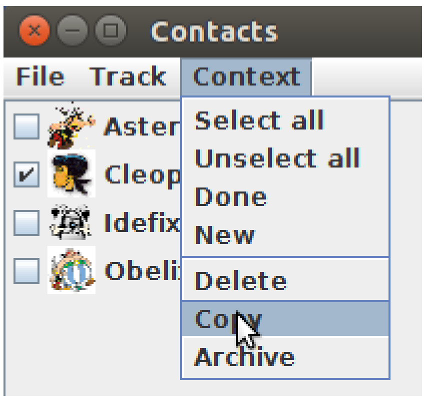


To create a new group use the right click menu.

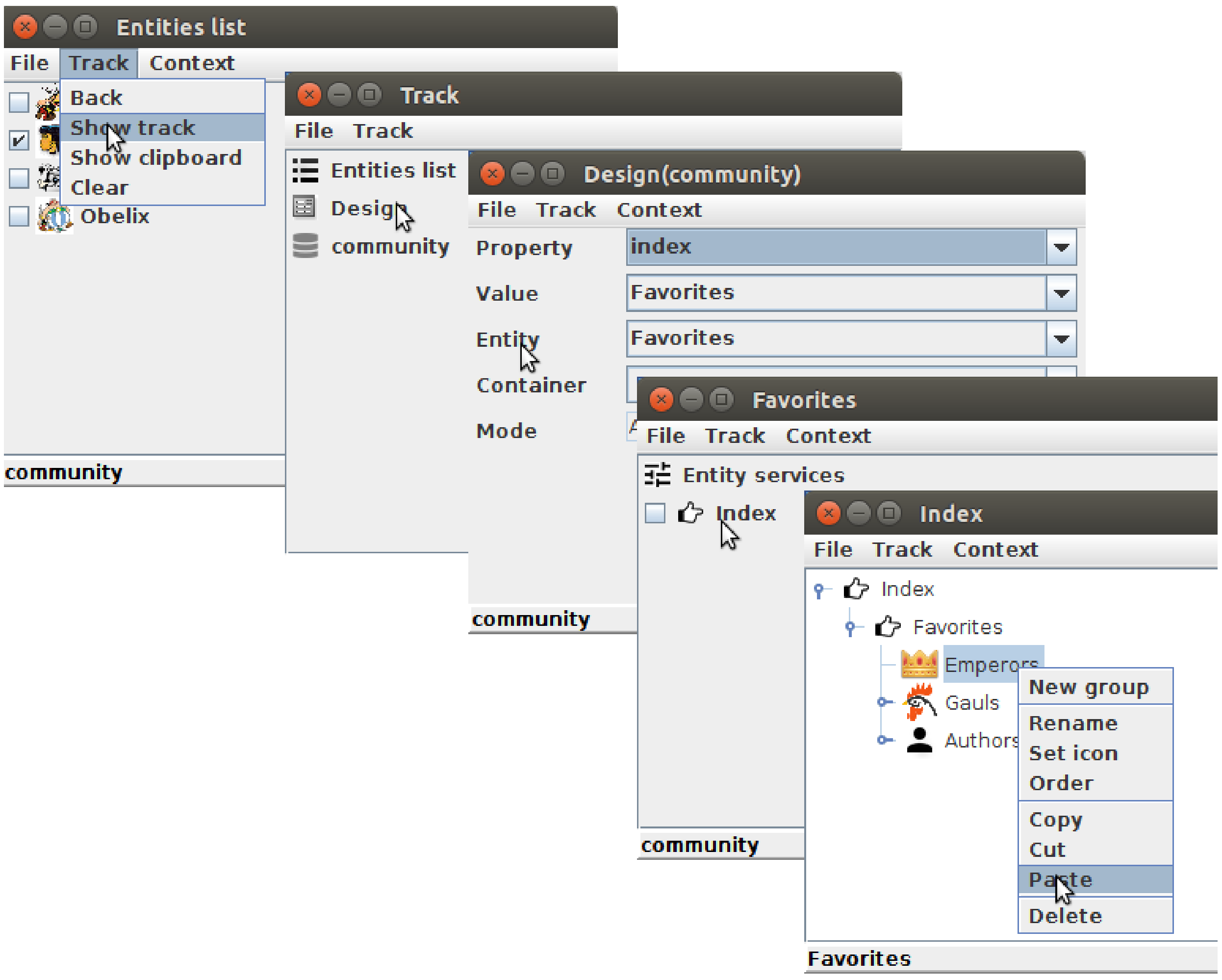


To put a new member into the group use the copy → paste procedure.

- Navigate to the contacts list and copy **Cleopatra** into the application clipboard.



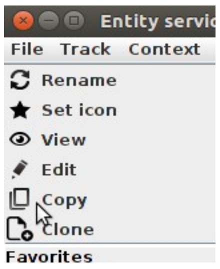
- Navigate to **Favorites** and paste the entity.



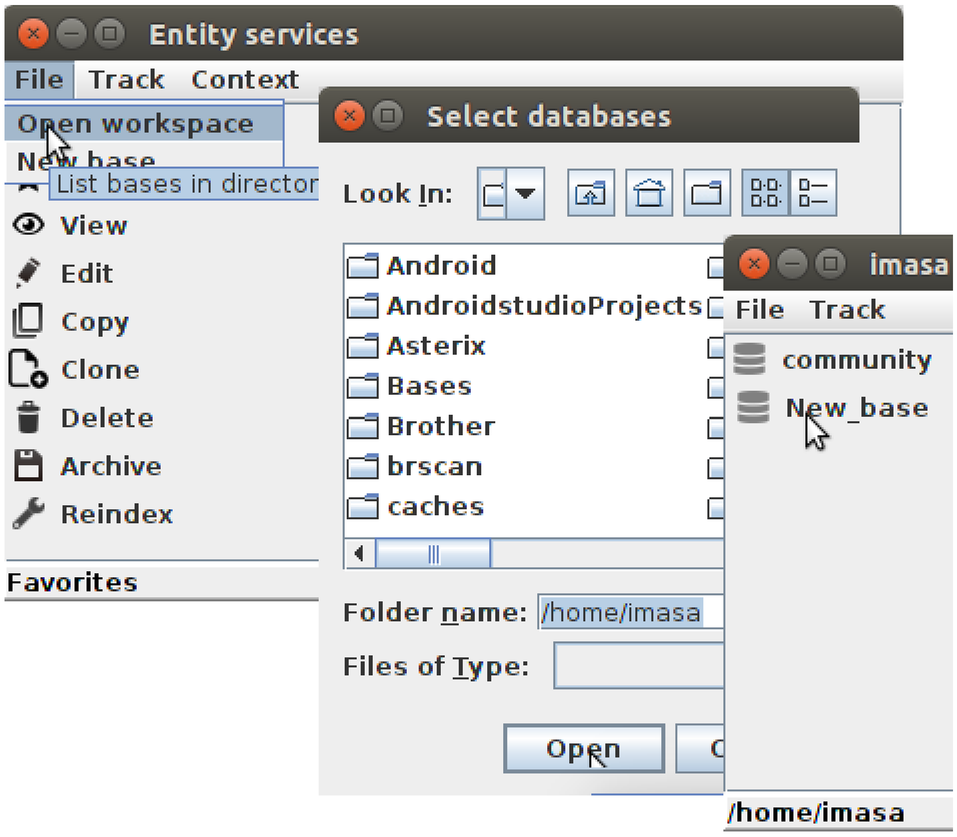
At least a user can create a special property (the tag) and assign it to all entities in the association. Later this tag can be used to retrieve a list of association members.

10. Data transfer.

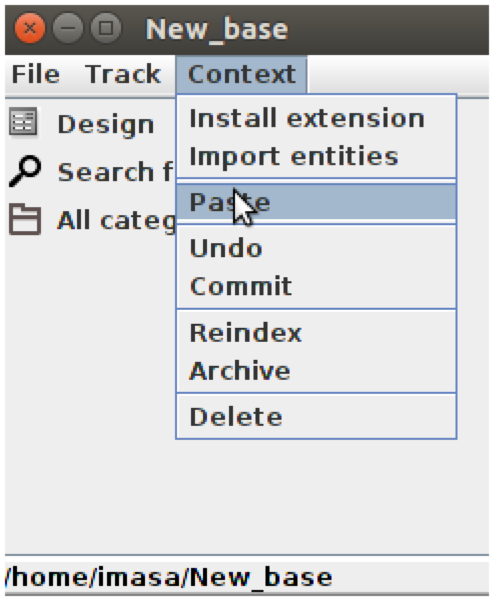
The data model in the **JEntigrator** has no schema and data objects are presented as files within the database directory. That means that data objects can be transferred in the same way as any other files. The difference is that together with data objects all referenced entities must be transferred too to keep the target database consistent. The property index of the target database must alse be modified accordingly. The   **JEntigrator** makes this work automatically when pasting entites from another database. The example of copy → paste the **Favortes** from the **community** to the **New_base** is shown below. The **Favortes** is an index which references some other entities. First we should copy selected entities into the application clipboard. It can be done for example from the entity services context.



Then we should navigate to the **BaseNavigator** context of the **New_base.**

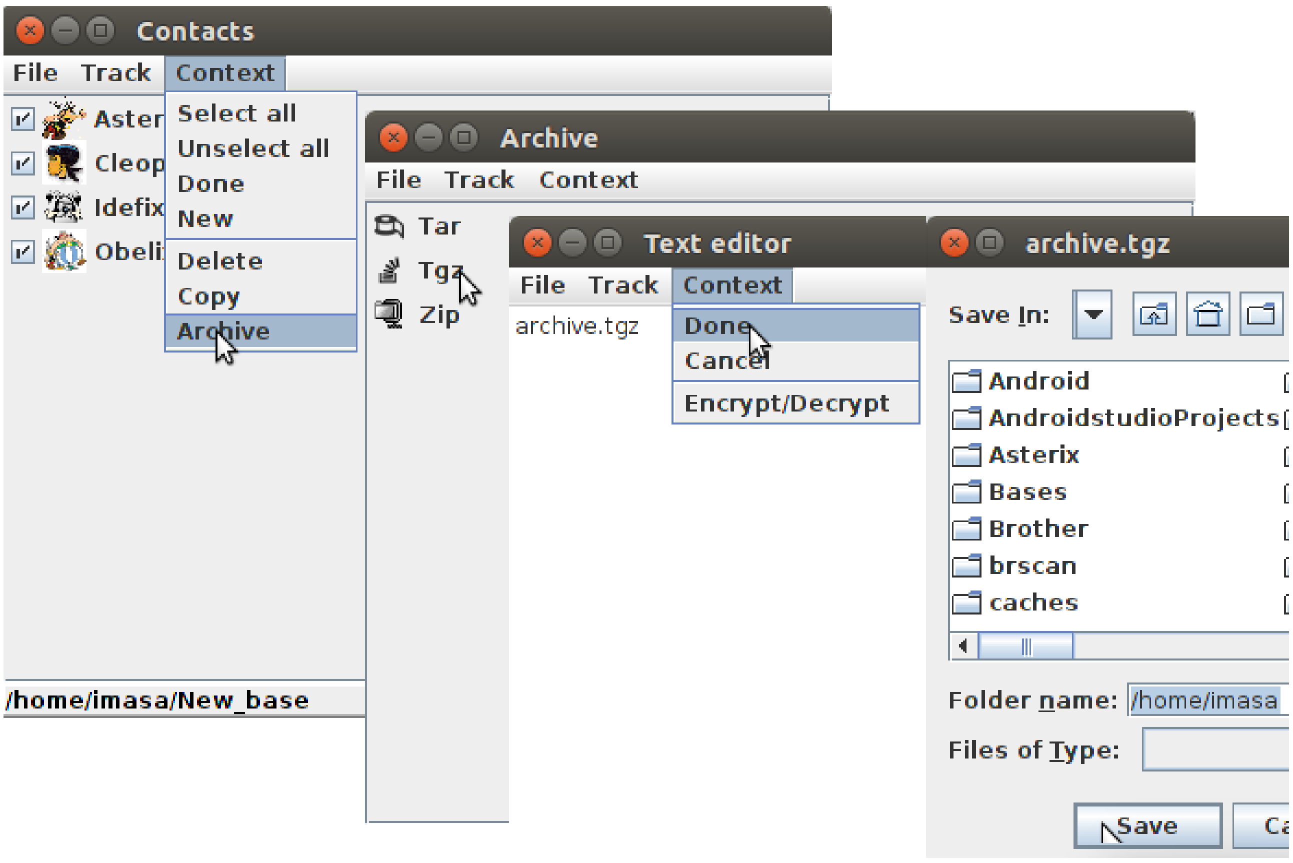


The last step is to paste entites from the clipboard.



The **JEntigrator** resolves references and paste all entities into the target database.

It is possible to archive some selected entites into an archive file and import them later into another database. To check it delete the **New_base** and create a new one. Then archive all contacts into the archive file. Navigate to the **CategoryPanel** context for **Contacts** in the **community** database. Select all items and click the **Acrhive** menu item. Click the desirable archive type item. Accept the suggestedfile name or type a new one. Select the target directory and save the archive. The archive can be imported from the **BaseNavigator** context.



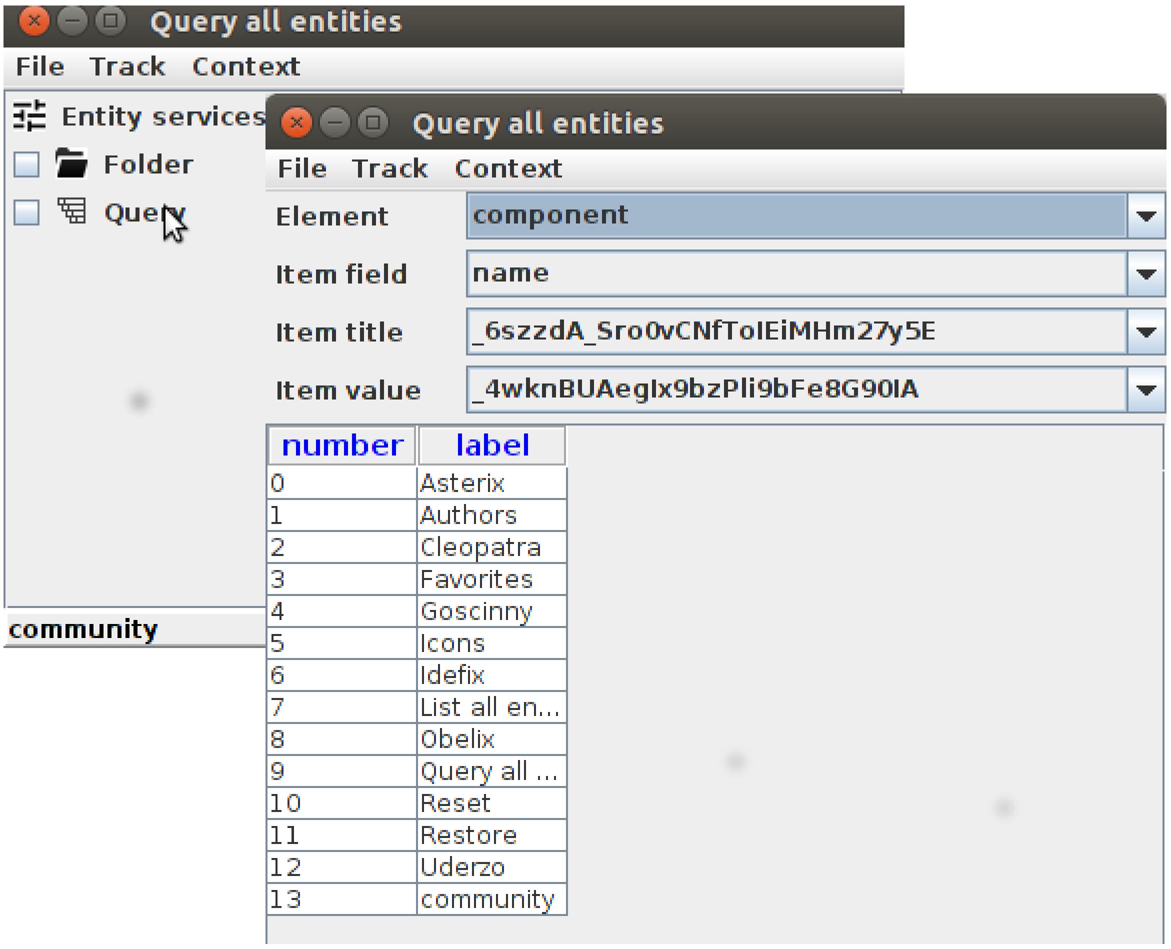
11. Adaptable contexts **Query** and **Procedure.**

The **JEntigrator** includes two facets **query** and **procedure** that let users create adaptable controls to fulfil current requirements. These facets relate to corresponding entities of type **query** or **procedure.** The entity's folder contains a class file performing the program part of work, the source java file and **eclipse** project files. A user can modify and compile this file in order to adapt the control . That material is outside the scope of this guide.   The facet provides GUI to control the service.

The **query** facet provides a table view of information . Take a look to the sample **Query all entities.** Navigate to the **FacetsPanel** for the **Query all entities** query. Click the **Query** item. The **QueryPanel** context appears. This context comprises two parts: the item selector and the table view. The entity selector of the query is implemented in the query class file **_2DOtCo5e_S6ARWU0H_MtyFnnPThE.class** located in the entitiy's folder. This class must have the public method **select.** You may want to check the query source file **_2DOtCo5e_S6ARWU0H_MtyFnnPThE.java** there:

```
import java.util.logging.Logger;
import gdt.data.store.Entigrator;
import gdt.jgui.console.JMainConsole;
import gdt.jgui.entity.query.Query;

public class _2DOtCo5e_S6ARWU0H_MtyFnnPThE  implements Query {
private final static String ENTITY_KEY="_2DOtCo5e_S6ARWU0H_MtyFnnPThE";
@Override
public String[] select(JMainConsole console,String entihome$){
try{
//Do NOT change this section of the code
Entigrator entigrator=console.getEntigrator(entihome$);
String [] sa;

// Put query code here

sa=entigrator.indx_listEntitiesAtPropertyName("entity");

//Do NOT change this section of the code

return sa;

}catch(Exception e){
Logger.getLogger(getClass().getName());
return null;
}}}
```

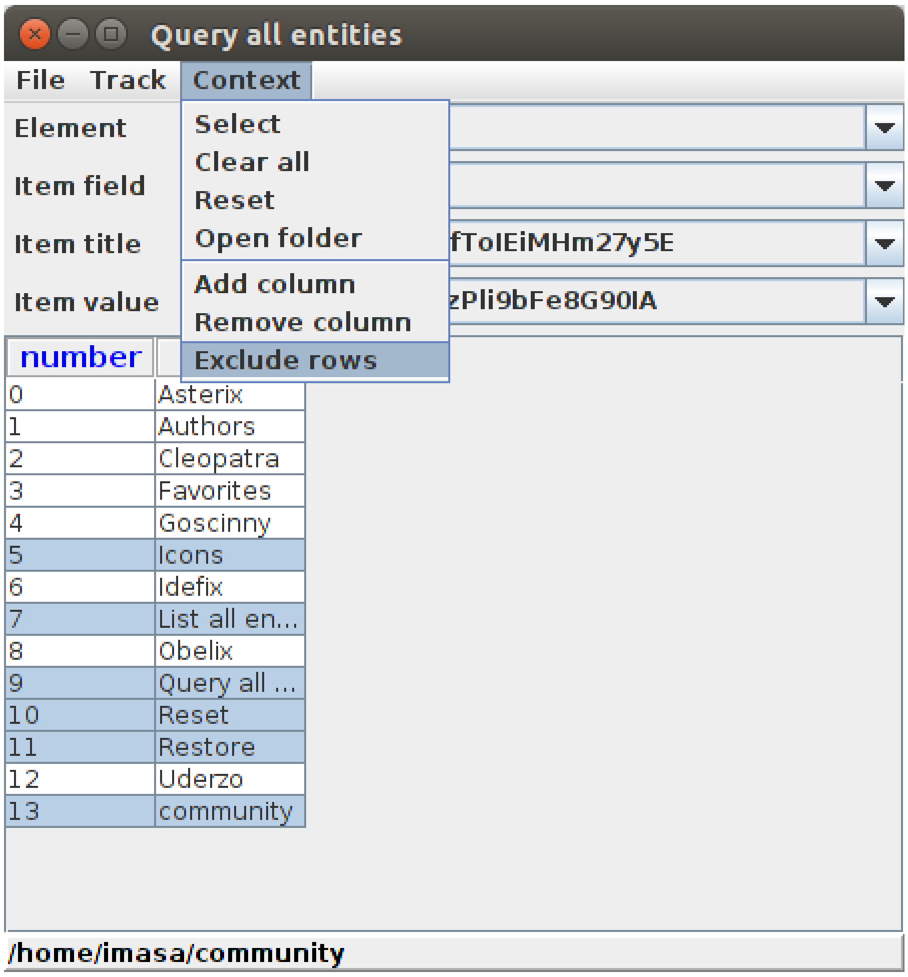
The table view initially lists all entities in the database. A user can modify the table using the context menu and item selector.



Element: component
Item field: name
Item title: _6szzdA_Sro0vCNfToIEiMHm27y5E
Item value: _4wknBUAeglx9bzPli9bFe8G90IA

number	label
0	Asterix
1	Authors
2	Cleopatra
3	Favorites
4	Goscinny
5	Icons
6	Idefix
7	List all en...
8	Obelix
9	Query all ...
10	Reset
11	Restore
12	Uderzo
13	community

To remove some rows from the table select the numbers and click the context menu item **Exclude rows.** The selected rows will be removed from the list.



Element

Item field

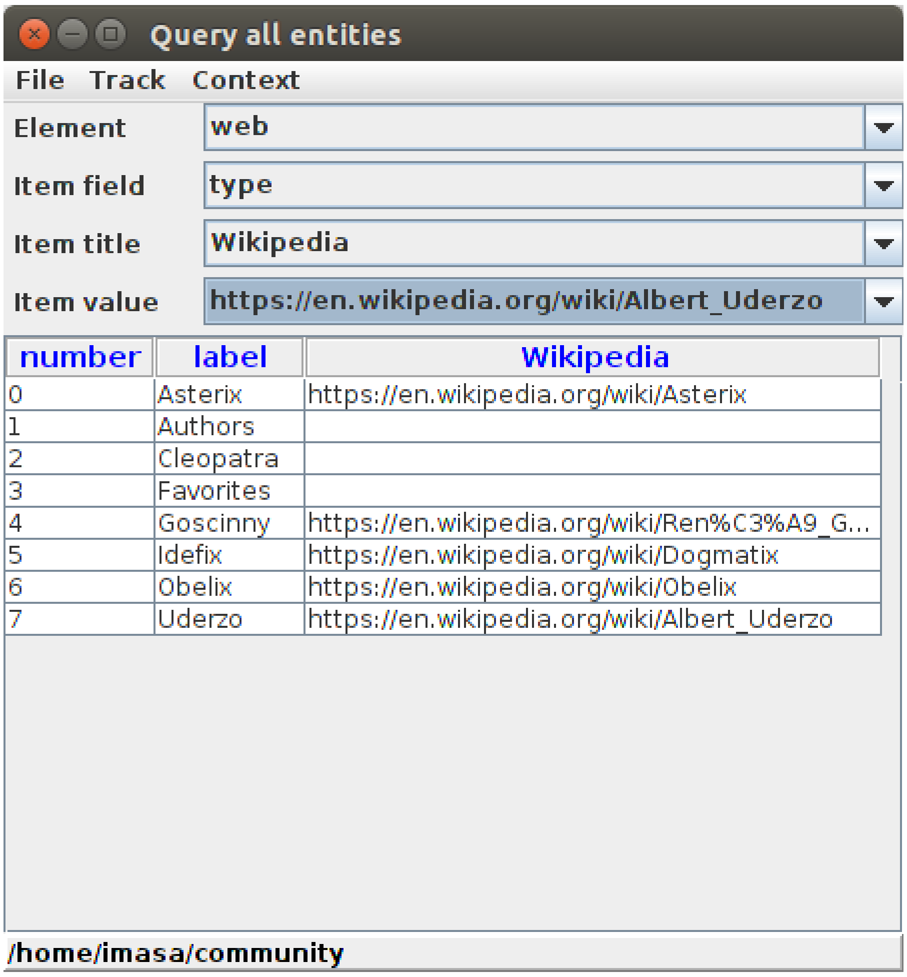
Item title fTolEiMHm27y5E

Item value zPli9bFe8G90IA

number	
0	Asterix
1	Authors
2	Cleopatra
3	Favorites
4	Goscinny
5	Icons
6	Idefix
7	List all en...
8	Obelix
9	Query all ...
10	Reset
11	Restore
12	Uderzo
13	community

/home/imasa/community

To add a column to the table view you have to select an element of the entity and the field of the core to select. The **Item title** shows the name of the column and The **Item value** all values (for all entities). After that click the context menu item **Add column.** To delete a column select it in the item selector and click the **Remove column** menu item. The click on the item in the **label** column displays the **FacetsPanel** for the entity. The **Clear all** menu item clears all changes and resets the query in the initial state. The **Reset** item replaces the query source file with the default default content. All changes will be lost ! Add the element **parameter** with the core {null, **noreset**,null} into the query to hide the **Reset** item.



Element: web

Item field: type

Item title: Wikipedia

Item value: https://en.wikipedia.org/wiki/Albert_Uderzo

number	label	Wikipedia
0	Asterix	https://en.wikipedia.org/wiki/Asterix
1	Authors	
2	Cleopatra	
3	Favorites	
4	Goscinny	https://en.wikipedia.org/wiki/Ren%C3%A9_G...
5	Idefix	https://en.wikipedia.org/wiki/Dogmatix
6	Obelix	https://en.wikipedia.org/wiki/Obelix
7	Uderzo	https://en.wikipedia.org/wiki/Albert_Uderzo

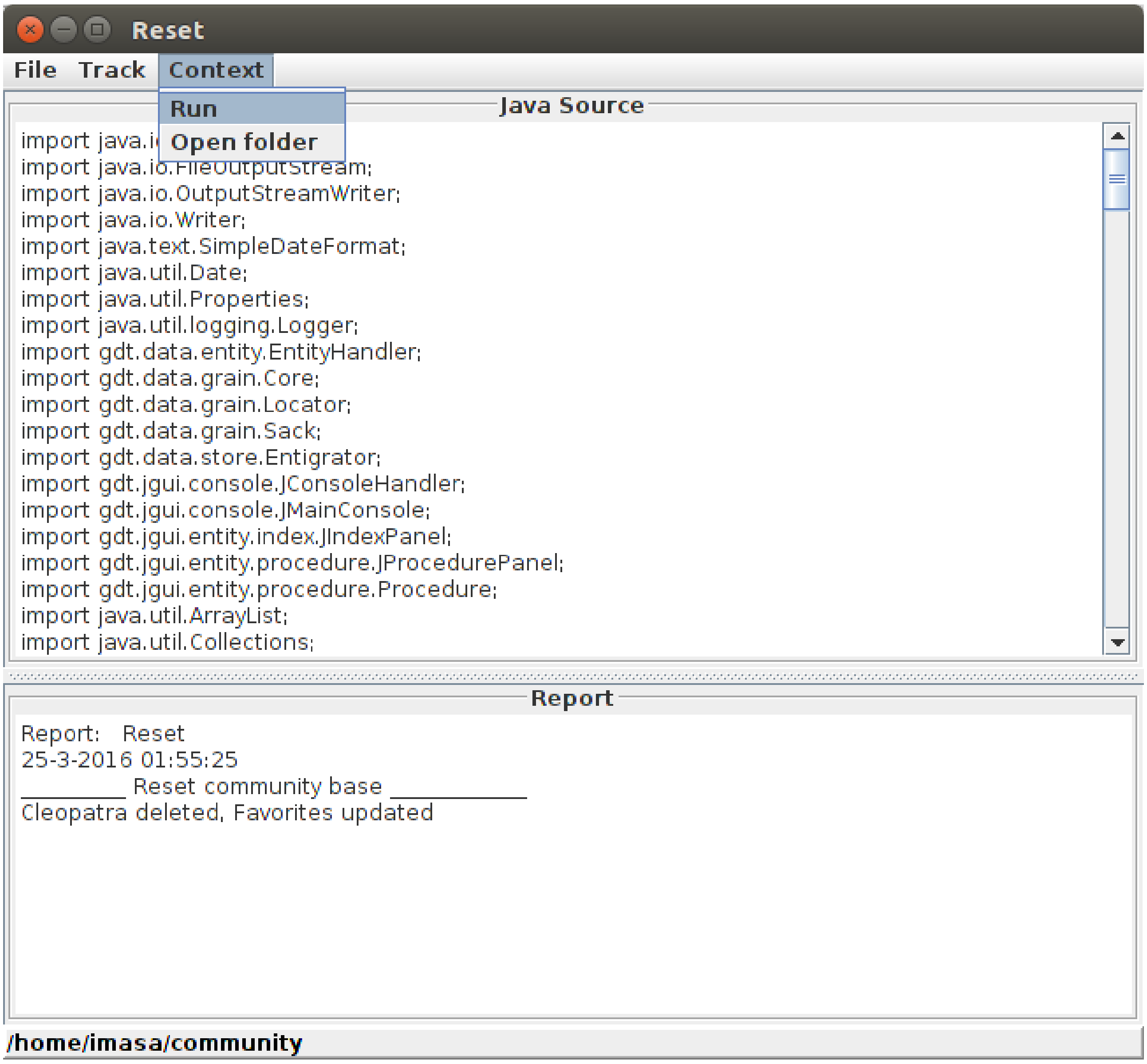
/home/imasa/community

The **procedure** facet provides the **run** method to perform certain actions in the database. For example the **Reset** procedure removes the **Cleopatra** contact and update the **Favorites** index. The source file **_Vo5ZNvv0Oc0LNE4PTUV36psyGCo.java** of the procedure is located in the entity's folder.

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Properties;
import java.util.logging.Logger;
import gdt.data.entity.EntityHandler;
import gdt.data.grain.Core;
import gdt.data.grain.Locator;
import gdt.data.grain.Sack;
import gdt.data.store.Entigrator;
import gdt.jgui.console.JConsoleHandler;
import gdt.jgui.console.JMainConsole;
import gdt.jgui.entity.index.JIndexPanel;
import gdt.jgui.entity.procedure.JProcedurePanel;
import gdt.jgui.entity.procedure.Procedure;
import java.util.ArrayList;
import java.util.Collections;
public class _Vo5ZNvv0Oc0LNE4PTUV36psyGCo  implements Procedure {
private final static String ENTITY_KEY="_Vo5ZNvv0Oc0LNE4PTUV36psyGCo";
@Override
public void run(JMainConsole console,String entihome$,Integer dividerLocation){
try{
//		System.out.println("Reset:run:BEGIN:entihome="+entihome$);
Entigrator entigrator=console.getEntigrator(entihome$);
String label$=entigrator.indx_getLabel(ENTITY_KEY);
//Delete Cleopatra from the database
String cleopatraKey$=entigrator.indx_keyAtLabel("Cleopatra");
System.out.println("Reset:run:Cleopatra key="+cleopatraKey$);
if(cleopatraKey$!=null){
	Sack cleopatra=entigrator.getEntityAtKey(cleopatraKey$);
	entigrator.deleteEntity(cleopatra);
}else
	System.out.println("Reset:run:cannot find Cleopatra");
//Delete Emperors from Favorites
String favoritesKey$=entigrator.indx_keyAtLabel("Favorites");
if(favoritesKey$!=null){
	Sack favorites=entigrator.getEntityAtKey(favoritesKey$);
	Core[] ca=favorites.elementGet("index.jlocator");
	if(ca!=null){
		Properties locator;
		String title$;
		String type$;
		String groupKey$;
		String nodeKey$;
		ArrayList <String>sl=new ArrayList<String>();
		//Find Emperors key
		String emperorsKey$=null;
		for(Core c:ca){
			try{
				locator=Locator.toProperties(c.value);
				title$=locator.getProperty(Locator.LOCATOR_TITLE);
				type$=locator.getProperty(JIndexPanel.NODE_TYPE);
				nodeKey$=locator.getProperty(JIndexPanel.NODE_KEY);
				groupKey$=locator.getProperty(JIndexPanel.NODE_GROUP_KEY);
				if("parent".equals(groupKey$)
					&&"Emperors".equals(title$)){
					emperorsKey$=nodeKey$;
					System.out.println("Reset:run:found Emperors key="+emperorsKey$);
					sl.add(c.name);
					break;
				}
			}catch(Exception e){
				System.out.println("Reset:run:"+e.toString());
			}
		}
		if(emperorsKey$==null){
			System.out.println("Reset:run:cannot find Emperors key");
			return;
		}
		//Find Emperors members
				for(Core c:ca){
					try{
						locator=Locator.toProperties(c.value);
						title$=locator.getProperty(Locator.LOCATOR_TITLE);
						type$=locator.getProperty(JIndexPanel.NODE_TYPE);
						nodeKey$=locator.getProperty(JIndexPanel.NODE_KEY);

	groupKey$=locator.getProperty(JIndexPanel.NODE_GROUP_KEY);
						if(emperorsKey$.equals(groupKey$)){
							sl.add(c.name);
							break;
						}
					}catch(Exception e){
						System.out.println("Reset:run:"+e.toString());
					}
```

```
                                        }
                                        if(sl.size()<1){
                                                System.out.println("Reset:run:empty emperors");
                                                return;
                                        }
                                        for(String s:sl)
                                                favorites.removeElementItem("index.jlocator", s);
                                        entigrator.save(favorites);
        }else
                System.out.println("Reset:run:empty Emperors");
}
else
        System.out.println("Reset:run:cannot find Favorites");
File report=new File(entihome$+"/"+ENTITY_KEY+"/report.txt");
if(!report.exists())
        report.createNewFile();
Date curDate = new Date();
SimpleDateFormat format = new SimpleDateFormat();
format = new SimpleDateFormat("dd-M-yyyy hh:mm:ss");
String date$= format.format(curDate);
FileOutputStream fos = new FileOutputStream(report, false);
Writer writer = new OutputStreamWriter(fos, "UTF-8");
writer.write("Report:   "+label$+"\n");
writer.write(date$+"\n");
writer.write("__________ Reset community base _____________\n");
writer.write("Cleopatra deleted, Favorites updated\n");
writer.close();
JProcedurePanel jpp=new JProcedurePanel();
String jppLocator$=jpp.getLocator();
jppLocator$=Locator.append(jppLocator$, Entigrator.ENTIHOME, entihome$);
jppLocator$=Locator.append(jppLocator$, EntityHandler.ENTITY_KEY,ENTITY_KEY);
jppLocator$=Locator.append(jppLocator$,
JProcedurePanel.DIVIDER_LOCATION,String.valueOf(dividerLocation));
JConsoleHandler.execute(console, jppLocator$);
}catch(Exception e){
Logger.getLogger(getClass().getName());
}}}
```

To run the procedure navigate to the **rocedurePanel** context and click the **Run** menu item.



```
import java.i
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Properties;
import java.util.logging.Logger;
import gdt.data.entity.EntityHandler;
import gdt.data.grain.Core;
import gdt.data.grain.Locator;
import gdt.data.grain.Sack;
import gdt.data.store.Entigrator;
import gdt.jgui.console.JConsoleHandler;
import gdt.jgui.console.JMainConsole;
import gdt.jgui.entity.index.JIndexPanel;
import gdt.jgui.entity.procedure.JProcedurePanel;
import gdt.jgui.entity.procedure.Procedure;
import java.util.ArrayList;
import java.util.Collections;
```

```
Report:   Reset
25-3-2016 01:55:25
_________ Reset community base ____________
Cleopatra deleted, Favorites updated
```

/home/imasa/community

The **Restore** procedure restores the **Cleopatra** and **Favorites** to the original state.