

Scarab - Developer's Guide

Scarab for Scarab developers

The Scarab Development Team

Scarab - Developer's Guide: Scarab for Scarab developers

The Scarab Development Team

Table of Contents

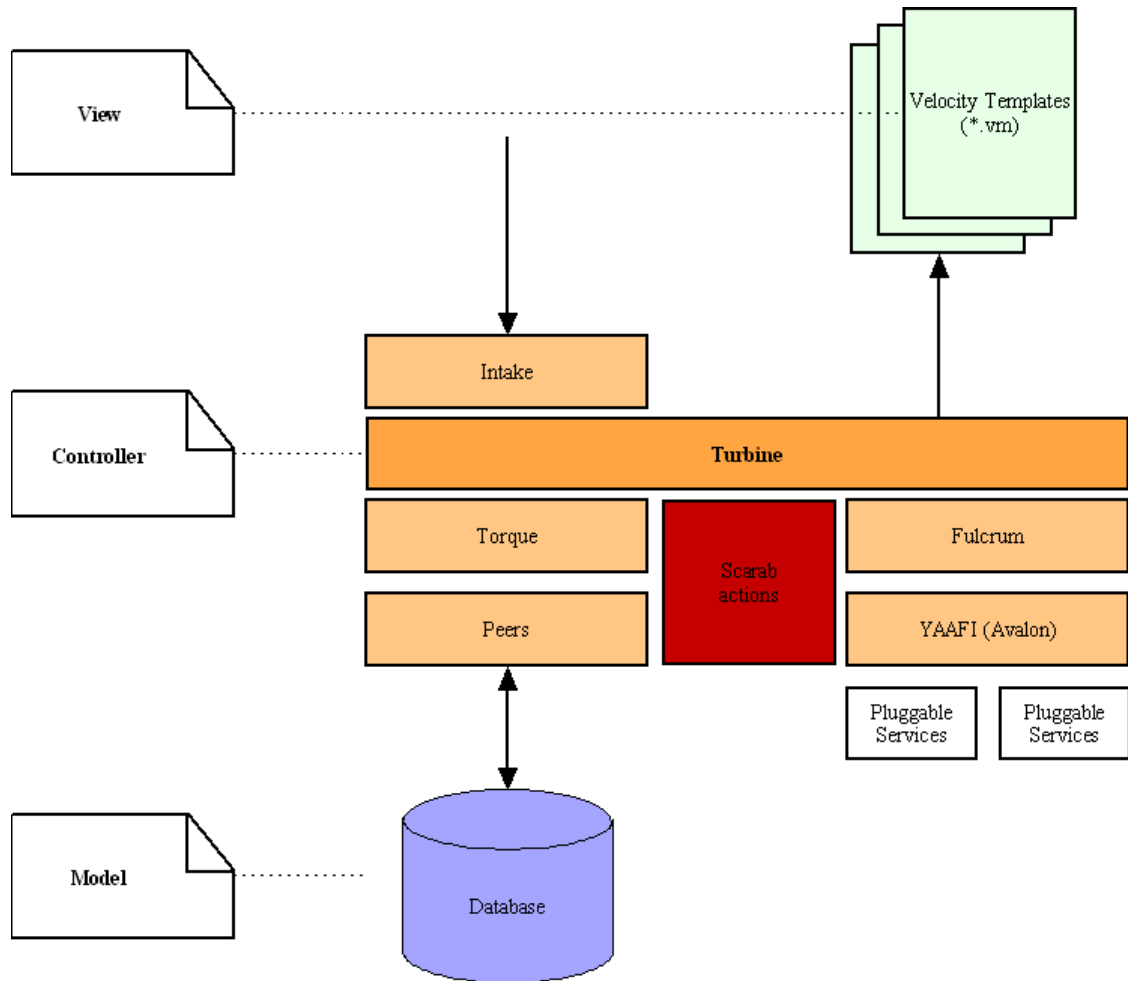
1. Scarab architecture	1
Scarab: a bird's eye view	1
An MVC web application based on the Turbine framework	1
The main Scarab components	2
Intake	2
Turbine (in the strict sense of the word)	2
Fulcrum: the services	3
YAAFI (Avalon): the microcontainer for Fulcrum services	3
O/R mapping: Torque and the <i>Peer</i> classes	3
The View: Velocity	4
2. Understanding the Ant build	5
\$SCARAB_ROOT/build/build.xml	5
\$SCARAB_ROOT/src/conf/conf/build.xml	5
3. Maven	7
Maven 1	7
Building Scarab with Maven	7
Normal (full) build	7
Quick build	7
Running the tests	7
4. Scarab development with Eclipse	9
Getting the Scarab sources	9
Getting the subclipse plugin	9
Connecting to the Scarab Subversion repository	9
Scarab in-place development	18
Installing the Sysdeo Tomcat Plugin	18
Generating the required files with Maven	19
Configuring and using the Tomcat plugin	19
5. Scarab development with Netbeans	21
Getting the Scarab sources	21
Getting the Mevenide plug-in (Maven-Netbeans integration)	21
Installing the Mevenide plug-in	21
Scarab in-place development	25
6. The Scarab data model	30
Introduction	30
Unique IDs generation: the <code>id_table</code> table	30
Reserved ID ranges	30
Turbine framework tables	31
Scarab tables	34
Entities related to users	34
Modules	36
Issue types	38
Attributes	40
Issues	42
Queries	48
Workflow	50
7. Workflow	51
Introduction	51
Workflows in Scarab	51
DefaultWorkflow	51
BasicWorkflow	52
Defining a new workflow	52
Using WfmOpen as a Scarab workflow engine	52
8. Accessing Scarab via XML-RPC	53

List of Tables

6.1. ID ranges	31
----------------------	----

Chapter 1. Scarab architecture

Scarab: a bird's eye view



An MVC web application based on the Turbine framework

A web application

Scarab is a web application that uses the Java servlet technology..

MVC Architecture

Regarding its global architecture, Scarab uses the MVC (Model-View-Controller) design pattern. The part that interacts with the user (View) is decoupled from the data defining the application state (Model) by a Controller that managing the navigation in Scarab.

The Turbine framework

A framework is a way of building applications, plus tools to help building them. The Turbine framework (<http://jakarta.apache.org/trubine> [<http://jakarta.apache.org/trubine/>]) is a doctrine to program web applications according to the MVC design pattern. Turbine also brings quite a set of tools (components, projects) to help code your application. Some of these components are detailed below, as they are used by Scarab.

The main Scarab components

Intake

The form data sent by the user may be submitted to a validation process (required fields, well-formed answers,...). This preliminary check is performed by a Turbine component called Intake.

Intake is now a part of Fulcrum (see below).

Note

Intake plays in Turbine the same role as the Struts Validator in Struts, if you're familiar with this well-known framework.

In the Scarab distribution

Intake validation rules are expressed in XML in the *src/conf/intake.xml* file.

On the Internet

The Intake documentation is available at this URL: <http://jakarta.apache.org/turbine/fulcrum/fulcrum-intake> [<http://jakarta.apache.org/turbine/fulcrum/fulcrum-intake/>].

Turbine (in the strict sense of the word)

It's the heart of the framework. As in others MVC2 frameworks, there is only one entry point in the application, the *scarab* servlet, that is an instance of the *org.apache.turbine.Turbine* class.

The behaviour of this servlet is ruled by five properties files, declared in the *TurbineConfiguration.xml* deployment descriptor :

1. `custom.properties`
2. `defaultCustom.properties`
3. `PermissionMapping.properties`
4. `TurbineResources.properties`
5. `Torque.properties`

Note

Which version of Turbine ?

If you know Turbine or if you have read the documentation on the Jakarta web site, you have probably real-

ized that the architecture of the different versions of Turbine (2.3 and 2.4 are currently available) is quite different.

In fact, Scarab has been developed initially by members of the Turbine project and it used a pre-version of Turbine 3 -- that will probably never be released. So the current development team spends some energy trying to bring Scarab back in the current Turbine development flow by "downgrading" the version of some of the components and integrating some of the Turbine 3 evolutions back in Turbine 2.4.

In the Scarab distribution

Configuration files mentioned above are in the `src/conf/conf/` directory of the Scarab distribution, and under `WEB-INF/conf` in the web application.

Turbine itself is a jar in `www/repository/turbine/jars` (and the corresponding source files in `www/repository/turbine/src`). As explained above, it is an intermediary release to which you probably cannot substitute another tagged release.

On the Internet

The Turbine project documentation is available at this URL: <http://jakarta.apache.org/turbine> [<http://jakarta.apache.org/turbine/>].

Fulcrum: the services

The Turbine architecture in 2.4= versions is based on the concept of services and uses a microcontainer of the Avalon family. Some of these services (among which Intake, mentioned earlier) are part of Fulcrum and used by Scarab.

In the Scarab distribution

The Fulcrum archives (a dozen of jars) are in `www/repository/fulcrum/jars`

On the Internet

The Fulcrum project documentation is available at this URL: <http://jakarta.apache.org/turbine/fulcrum/>

YAAFI (Avalon): the microcontainer for Fulcrum services

The services used by Scarab are implemented on top of a microcontainer, member of the Avalon family. (By the way, YAAFI is just an acronym for *Yet Another Avalon Framework Implementation*.)

In the Scarab distribution

YAAFI is in `www/repository/fulcrum/jars/fulcrum-yaafi-1.0.3.jar`

On the Internet

The YAAFI project documentation and its interface with Fulcrum are available at this URL: <http://jakarta.apache.org/turbine/fulcrum/fulcrum-yaafi> [<http://jakarta.apache.org/turbine/fulcrum/fulcrum-yaafi/>].

O/R mapping: Torque and the *Peer* classes

Scarab Java objects are populated from, and persisted to, a relational database (RDBMS) using a soft-

ware component that was once part of Turbine but became independent since: Torque (Torque now belongs to the Apache database project).

From an XML high-level description of the database schema, Torque can generate:

1. database schema creation scripts (SQL/DDDL), for various RDBMS;
2. Java classes that map the various database tables to JavaBeans, with finders and accessor methods.

If you are interested in further details about the database schema, read chapter 3.

In the Scarab distribution

The three XML files describing the database schema are in the `src/schema` of the Scarab distribution.

On the Internet

The documentation of the Torque project is available at this URL: <http://db.apache.org/torque> [<http://db.apache.org/torque/>]

The View: Velocity

Turbine can use JSP (JavaServer Pages) but the first Scarab developers chose to use the Velocity technology for the dynamic pages sent to the user.

Velocity is a template language: at runtime, the values of JavaBeans placed in the Velocity "Context" are inserted in a predefined (static) content.

There are two differences with the JSP technology that are worth a mention here, that may well have been the motives of the initial Scarab development team:

1. first, there is a point about architecture. There are control structures in Velocity (such as conditions, loops, etc.) but it is not possible to insert Java code in a template, there is no way to insert the so-called "scriptlets". This should make the maintenance easier in the long term because the view and the controller are completely decoupled from each other.
2. in practice, the JSP can only be used to generate web pages (HTML or XML); Velocity is a more general templating language and can be used to generate mails, SQL scripts, PostScript and quite a lot of other funny things.

In the Scarab distribution

The Velocity templates for the Scarab web pages (and emails, etc.) are under `src/webapp/WEB-INF/templates`, grouped by functionality.

On the Internet

The documentation of the Velocity project is available at this URL: <http://jakarta.apache.org/velocity> [<http://jakarta.apache.org/velocity/>].

Chapter 2. Understanding the Ant build

The Scarab application currently may be built by the end user using Ant. Ant is used to create and initialize the database.

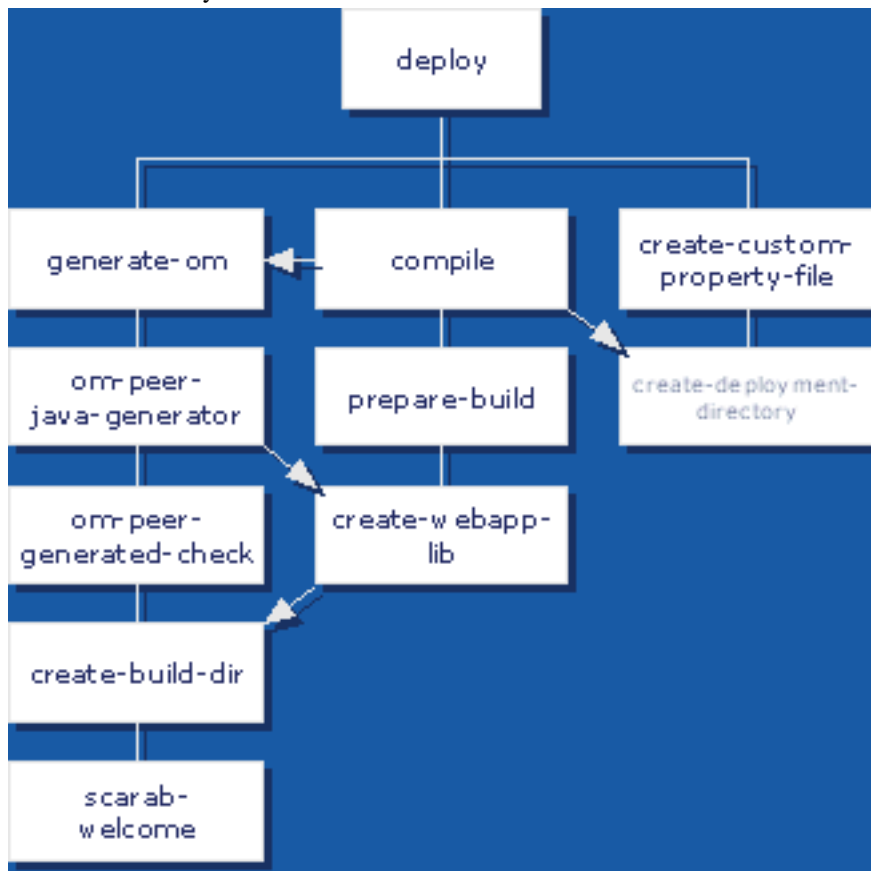
`$SCARAB_ROOT/build/build.xml`

This is the main build file.

The default target, `deploy`, is used to generate the code of the OM (Object Mapping) classes and to compile and build the Scarab application.

The second main target, `create-db`, is used to create and initialize the Scarab database. It does this by calling the second ant build file, described in the second section.

The target dependency graph below may help you visualize how targets are related to each other and the order in which they are called.

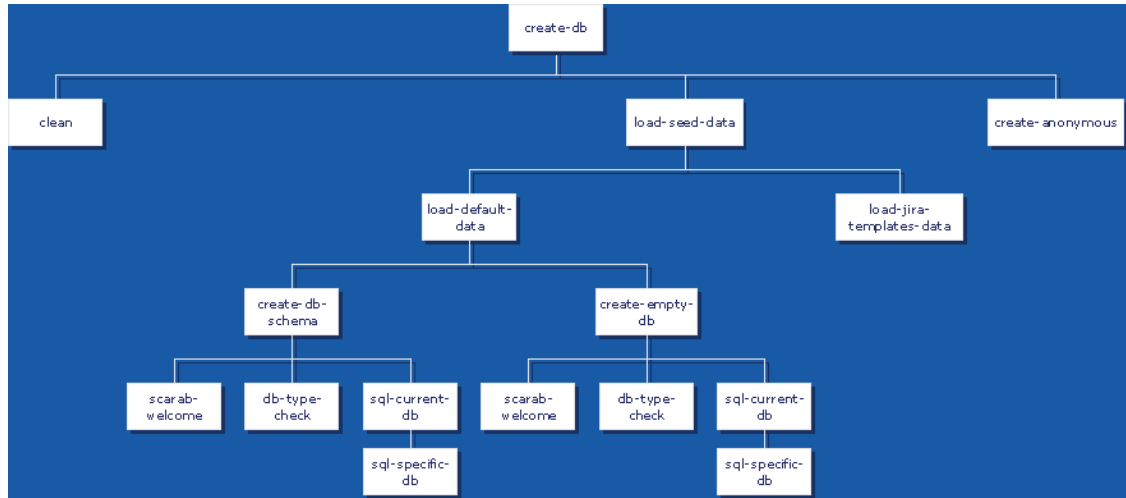


`$SCARAB_ROOT/src/conf/conf/build.xml`

This build file is used to create and populate the Scarab database.

The target dependency graph below may help you visualize how targets are related to each other and the order in which they are called.

Understanding the Ant build



Chapter 3. Maven

Though Scarab end-users are more likely to use Ant to build Scarab, as a developer you will need Maven to access all development functionalities, among which documentation generation.

Maven 1

Though Maven 2 is getting momentum and has officially become the mainstream version at the end of 2005, Scarab still uses Maven 1.

Maven 1 and Maven 2 are functionnally equivalent but they are not syntactically compatible and use different build files.

So you will need to install Maven 1 and refer to the Maven 1 documentation at this URL: <http://maven.apache.org/maven-1.x/>

Building Scarab with Maven

Normal (full) build

Go to the `$SCARAB_ROOT` directory.

Launch Maven:

```
maven war
```

Quick build

Go to the `$SCARAB_ROOT` directory.

Launch Maven:

```
maven war -Dmaven.test.skip
```

`-Dmaven.test.skip` skips the execution of unit tests. Scarab unit tests have been designed to be executed against a database; if yours is not (yet) configured, they will most certainly fail.

Running the tests

Just launch:

```
maven test
```

The results of the tests will be in `$(SCARAB_HOME)/target/test-results/`

If you'd like to make your tests with another db environment, follow these steps:

1. `maven clean` (you better start from scratch if you gonna test)
2. Configure your `build.properties` to contain at least the correct `scarab.database.type` value.
3. `maven war`

4. `maven scarab:create-db` (make sure you choose the same database than before!!)
5. `maven test`

Chapter 4. Scarab development with Eclipse

Getting the Scarab sources

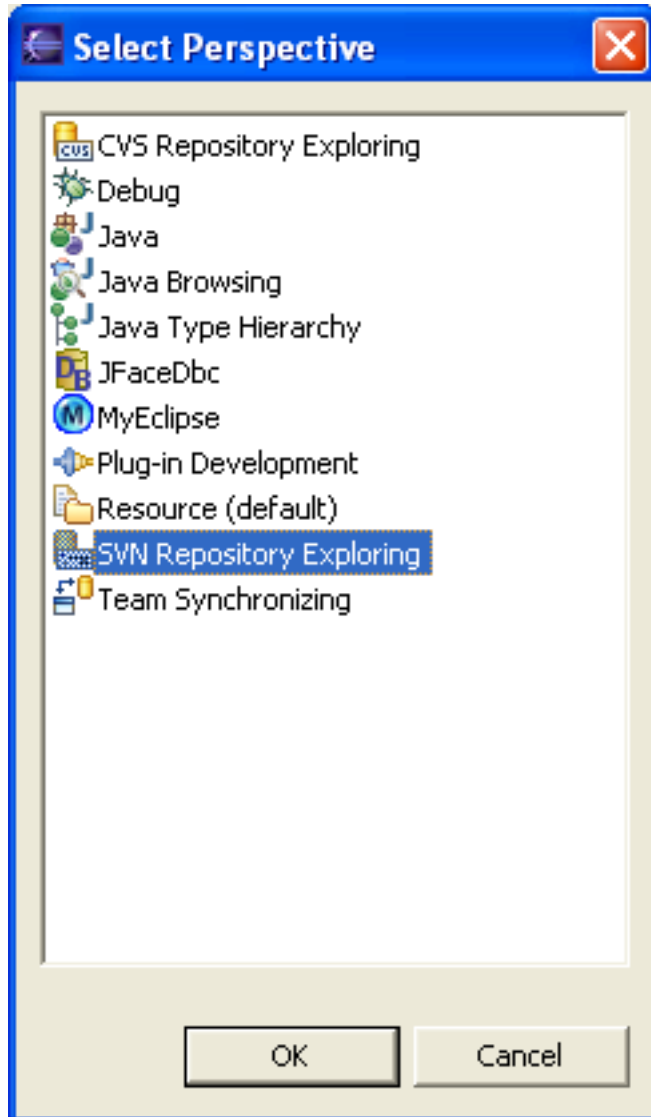
Getting the subclipse plugin

You will need the Subclipse plugin to connect to the Subversion repository that hosts and manages the Scarab source code base.

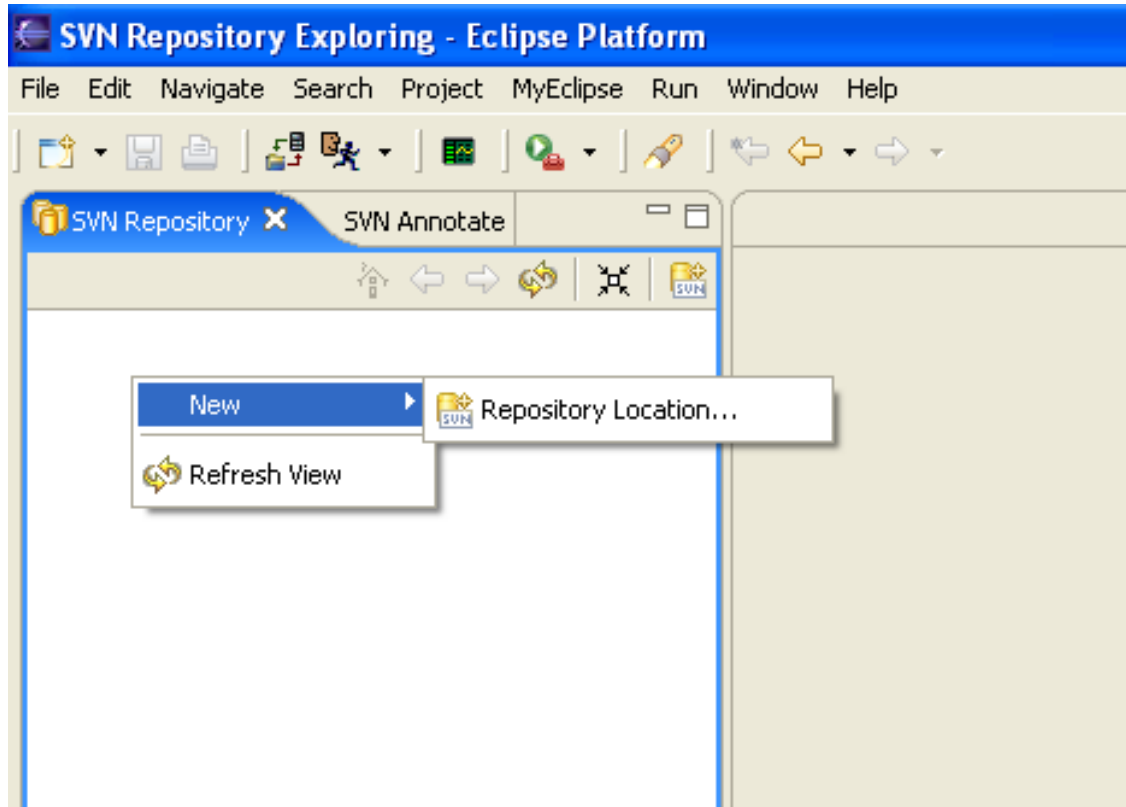
If you do not have this plugin already installed, you will find detailed installation instructions at this URL : <http://subclipse.tigris.org/install.html>

Connecting to the Scarab Subversion repository

Select the 'SVN Repository Exploring' perspective :



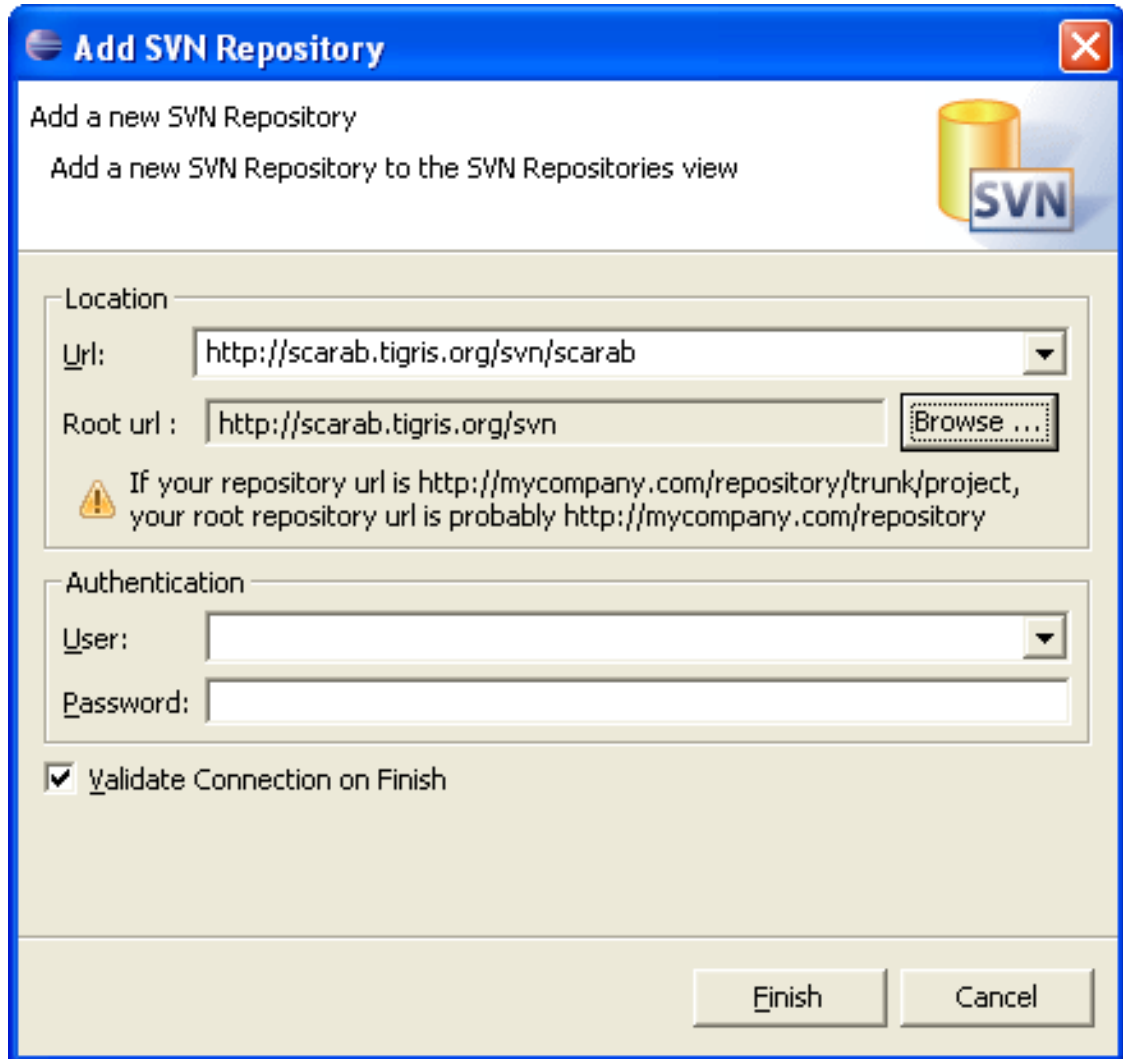
Right-click in the SVN Repository view to connect to a new Subversion repository :



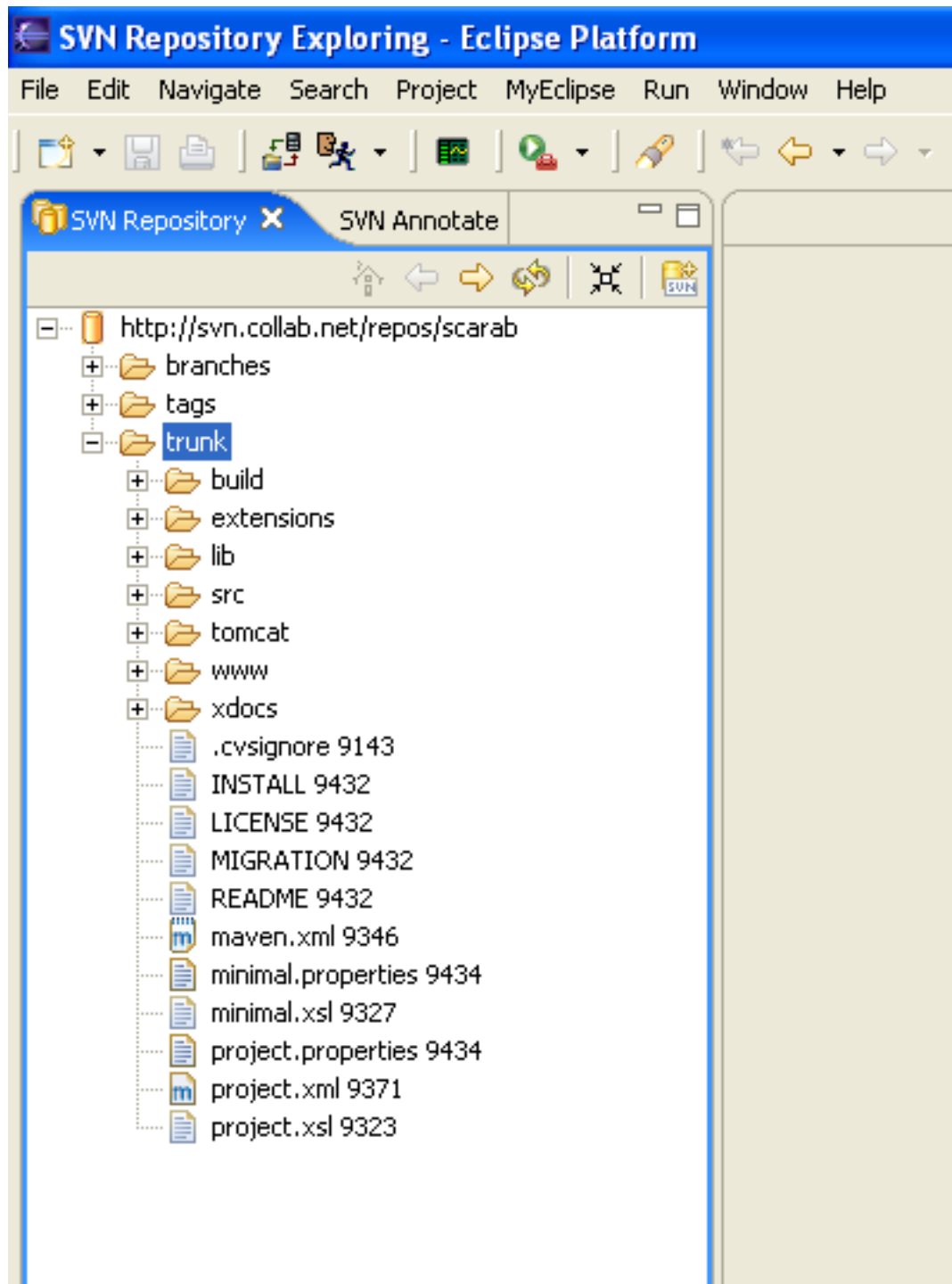
The connection URL is :: <http://scarab.tigris.org/svn/scarab>

and the Root URL is : <http://scarab.tigris.org/svn>

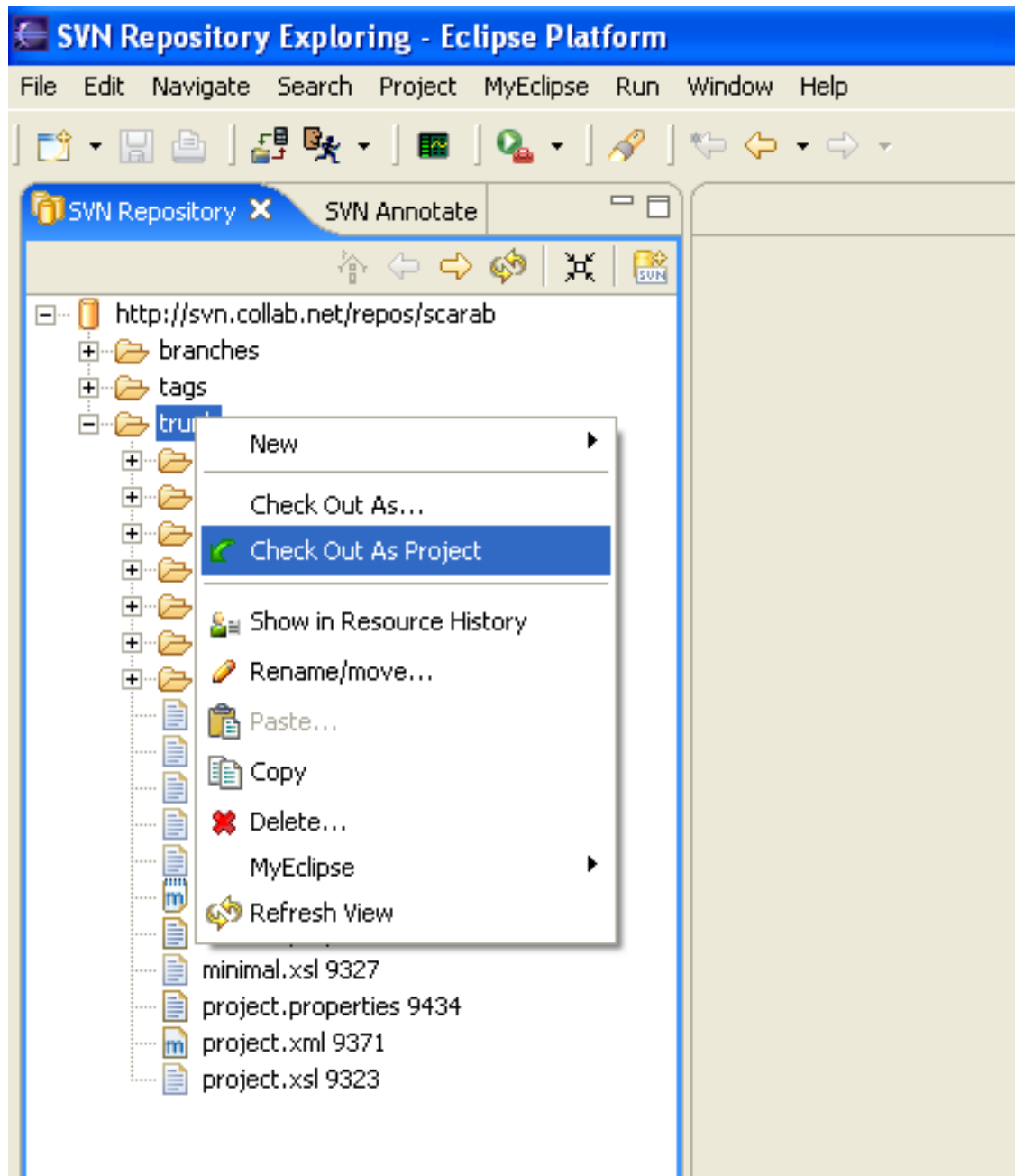
If you are a Scarab committer, use your tigris.org login and password to connect; otherwise you do not need any particular identification and you may leave the *User* and *Password* fields blank (though it seems that any *User/Password* will work here).



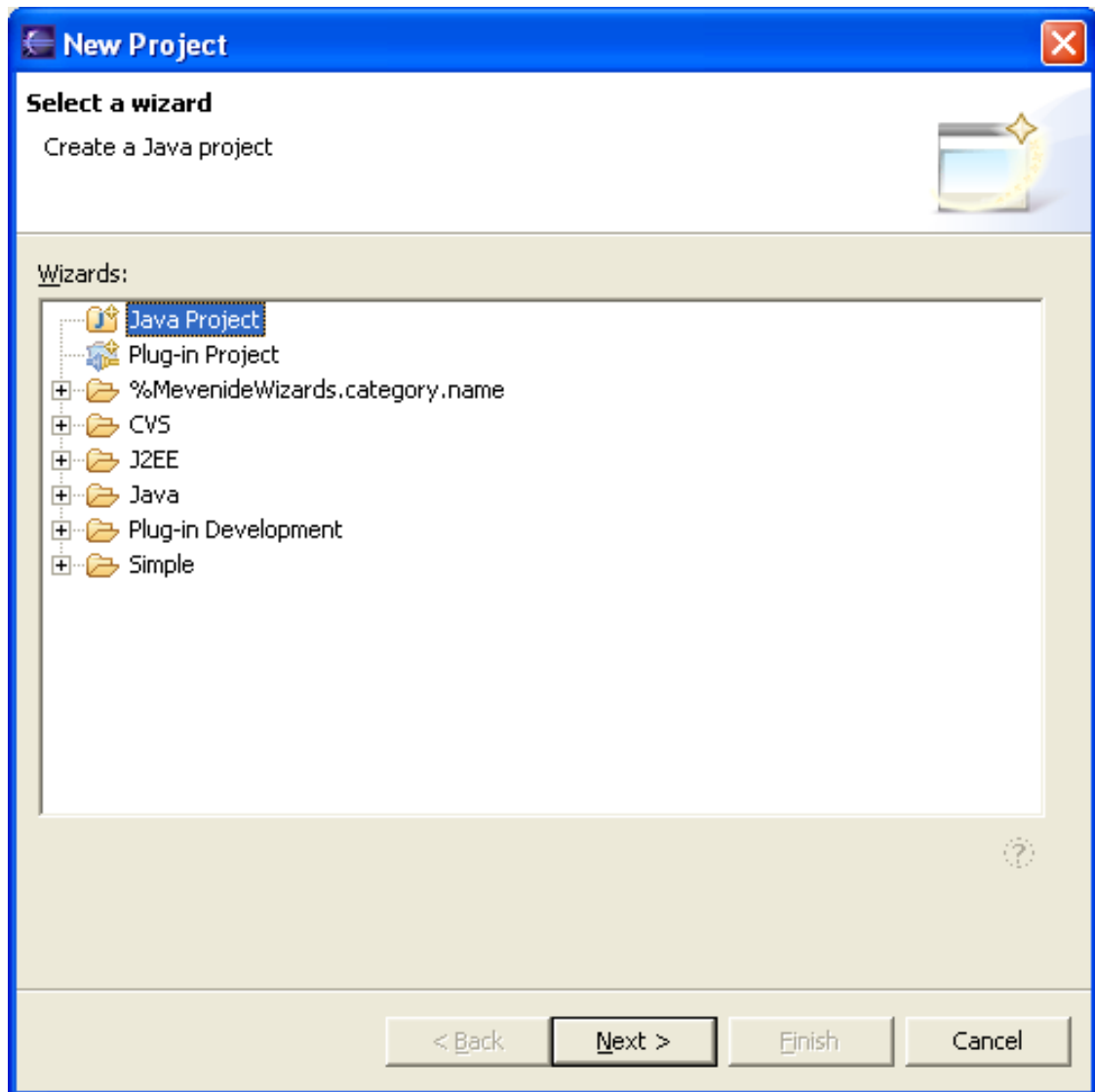
To get the current version, choose the 'trunk' branch (equivalent to CVS HEAD for those who have used this version control system before).



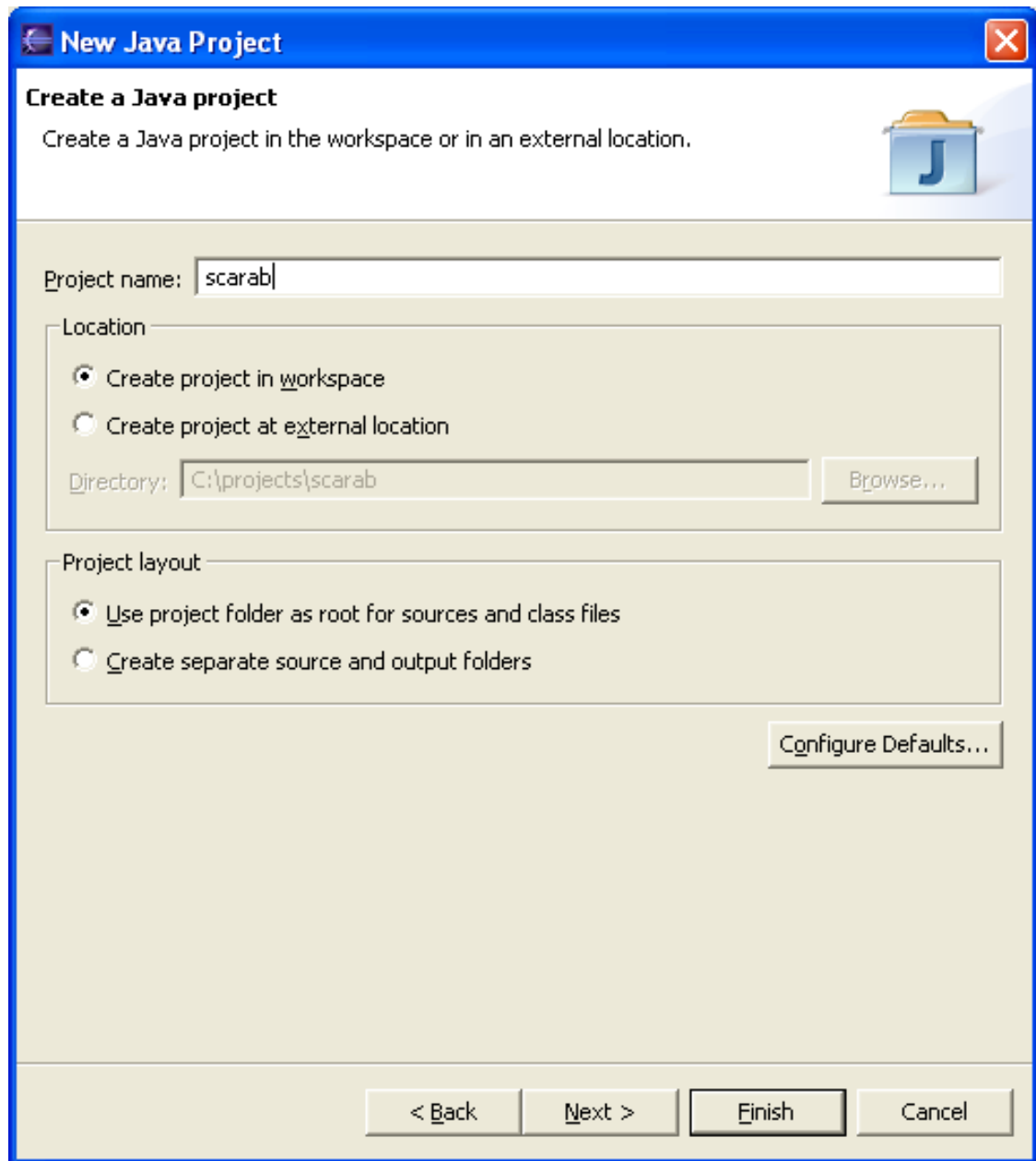
You can then use 'Checkout As Project', which allows you to rename the project (otherwise Eclipse would name it *trunk*, which is not that meaningful).



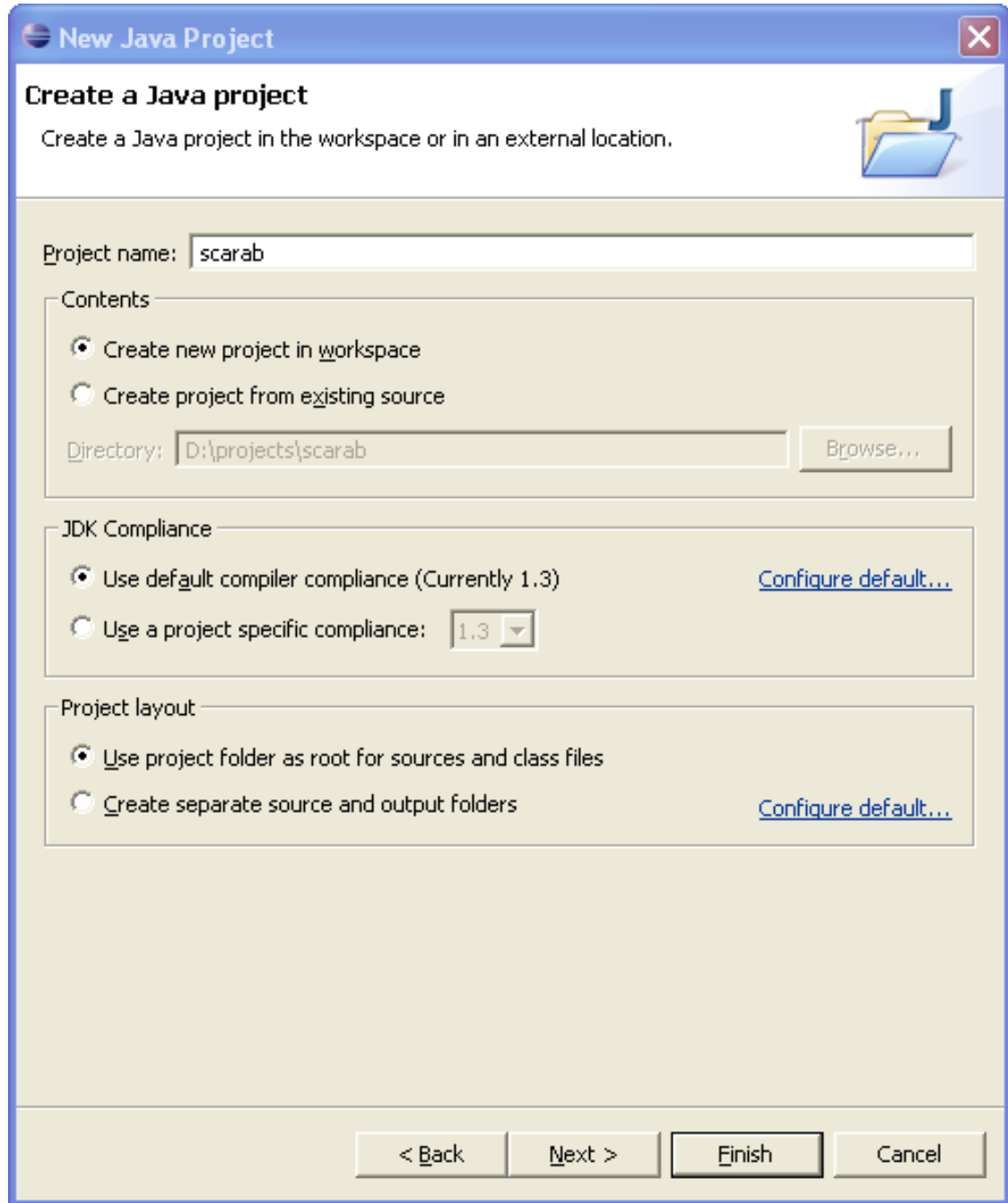
The 'New Project' wizard appears. Choose 'Java project'.



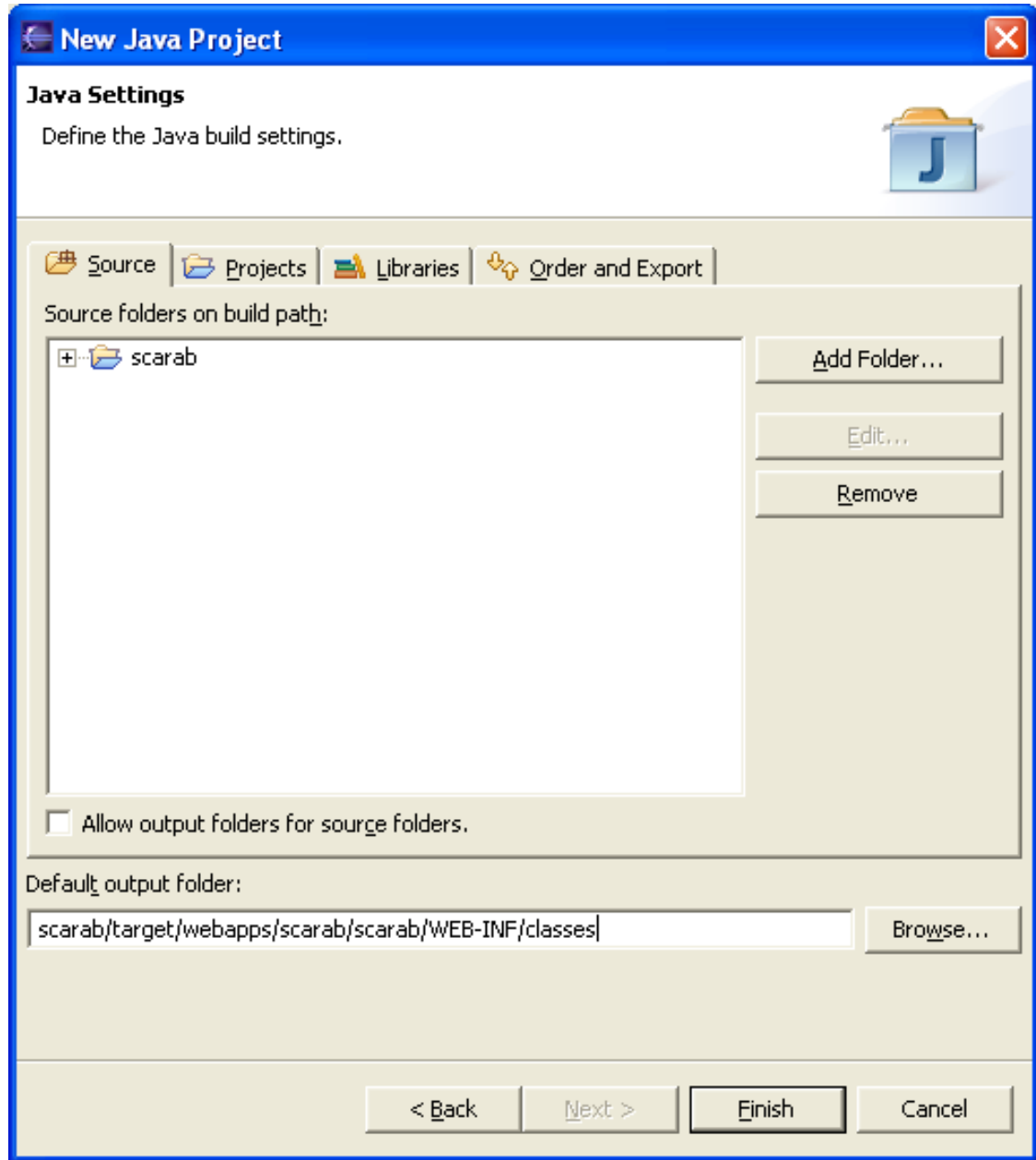
You now have to fill in the project name (you may choose *scarab* or any other name that suits you best).



If you are using Eclipse 3.1, you must choose J2SDK compatibility. At the moment, the development team has chosen to preserve JDK 1.3 compatibility.



Last step : it is best to change the *Default output folder* to `:scarab/target/webapps/scarab/scarab/WEB-INF/classes`



Scarab in-place development

Installing the Sysdeo Tomcat Plugin

The Sysdeo Tomcat plugin allows you to start and stop Tomcat from Eclipse and makes debugging the application under development easier.

Download the 3.1 beta version from this URL: <http://www.sysdeo.com/eclipse/tomcatplugin>.

To install this plugin, all you need is to uncompress the ZIP archive in the plugins directory of your Eclipse installation.

Generating the required files with Maven

At the root of your project, type the following commands :

```
maven war:inplace
```

```
maven eclipse
```

This command generates the .project file used by Eclipse.

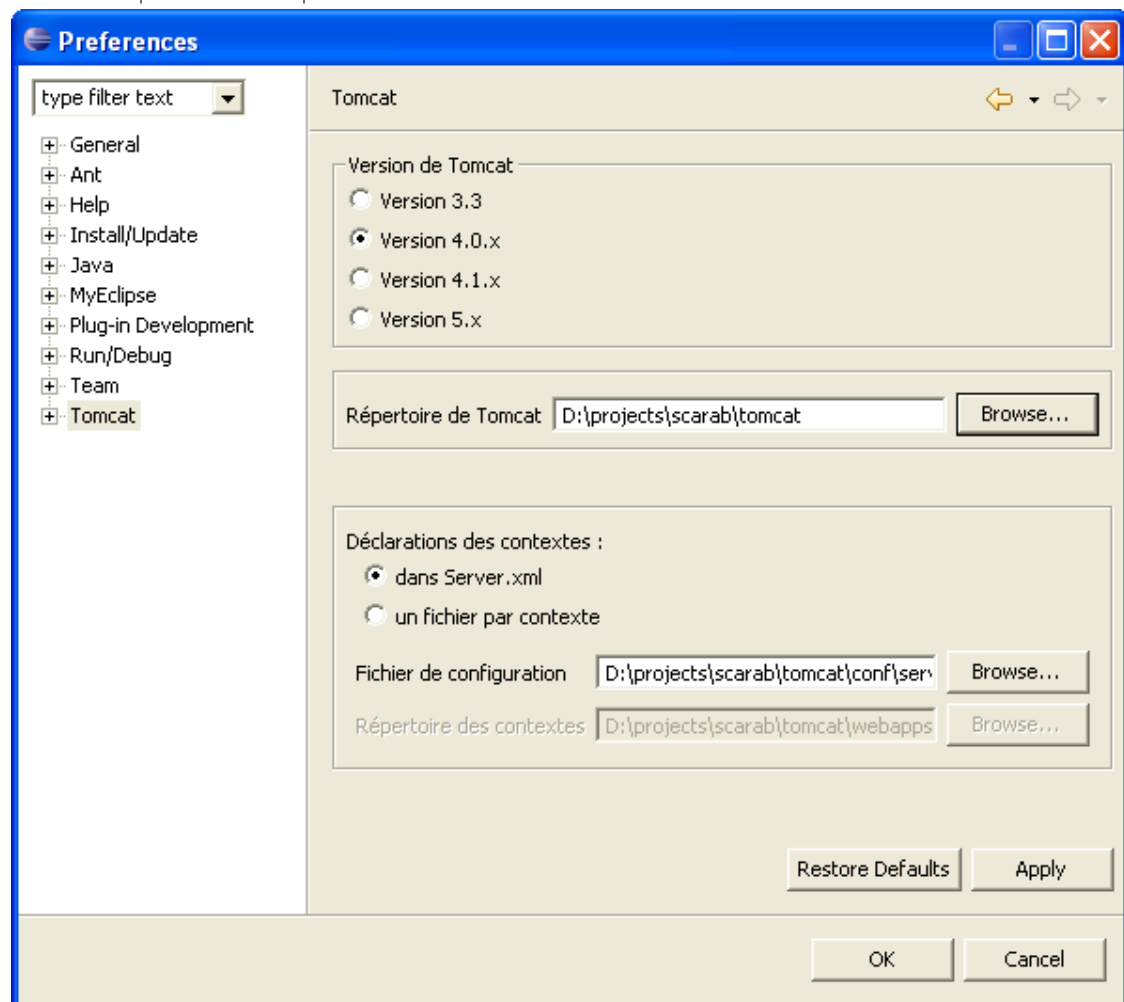
Configuring and using the Tomcat plugin

Rebuilding the project

You may now launch the build of the Scarab project in Eclipse.

Configuring the Tomcat preferences

In Window | Preferences... | Tomcat :

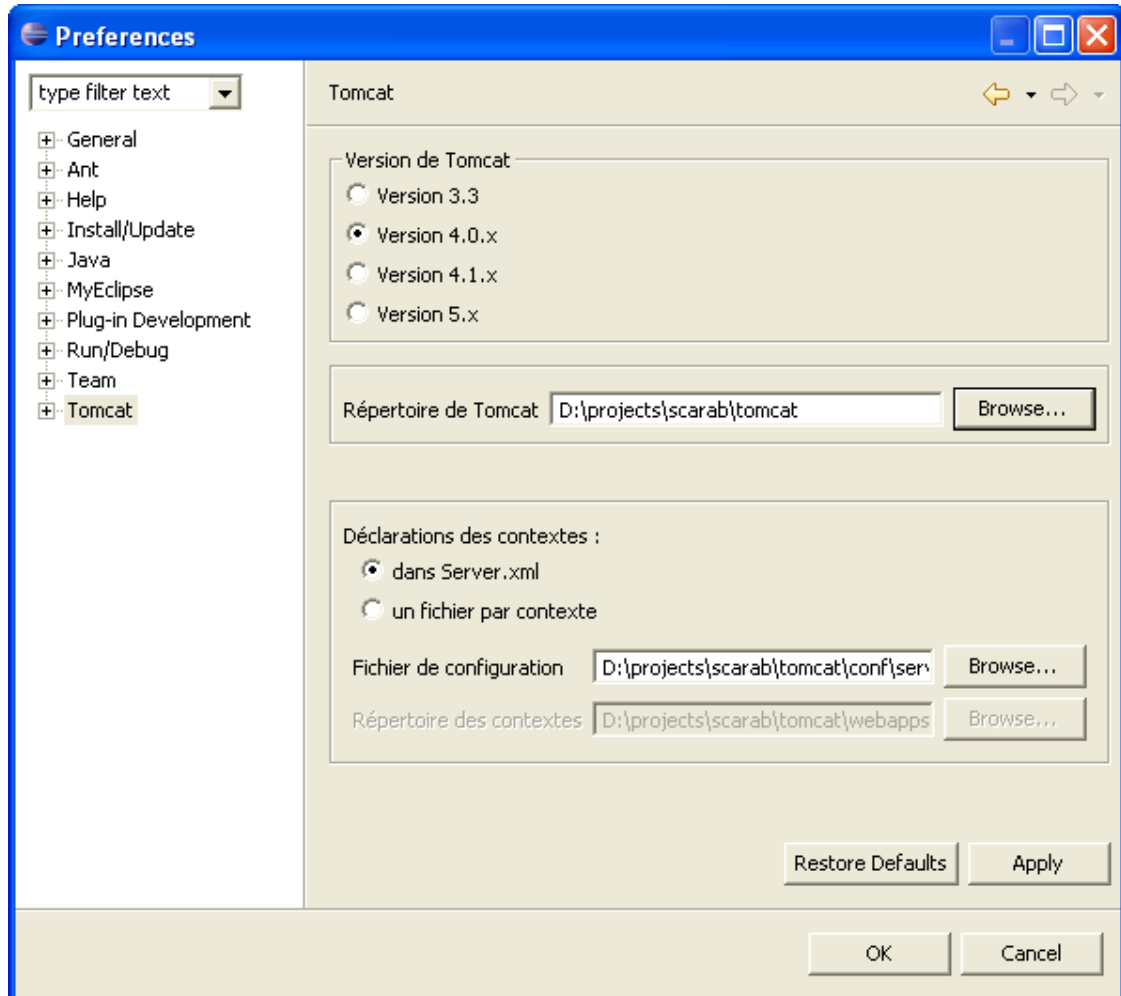


- Select the 4.0.x version

- Tomcat directory : the tomcat directory in your Scarab project
- Context declaration in server.xml

Setting the project Tomcat properties

In Project | Properties | Tomcat :



- Check the 'Is Tomcat project'
- Enter 'scarab' as context name
- Uncheck 'can update server.xml'
- Check 'Mark context as reloadable'
- Check 'Redirect context logger to Eclipse console...'

Chapter 5. Scarab development with Netbeans

Getting the Scarab sources

The current implementation of a Subversion client for Netbeans (code named "teepee", available from <http://subversion.netbeans.org/>) is only a prototype. So, at the moment, you need to use another client to check the sources out of the Scarab Subversion repository. If you use Windows as your development environment, you'll probably want to use TortoiseSVN [<http://tortoisesvn.tigris.org/>] - another Tigris project.

Getting the Mevenide plug-in (Maven-Netbeans integration)

Your best option to develop Scarab in Netbeans is certainly to use the Mevenide plug-in (which is, by the way, much more stable and functional in Netbeans than in Eclipse, not to mention JBuilder).

Download Mevenide for Netbeans at this URL: <http://mevenide.codehaus.org/mevenide-netbeans-project/index.html>

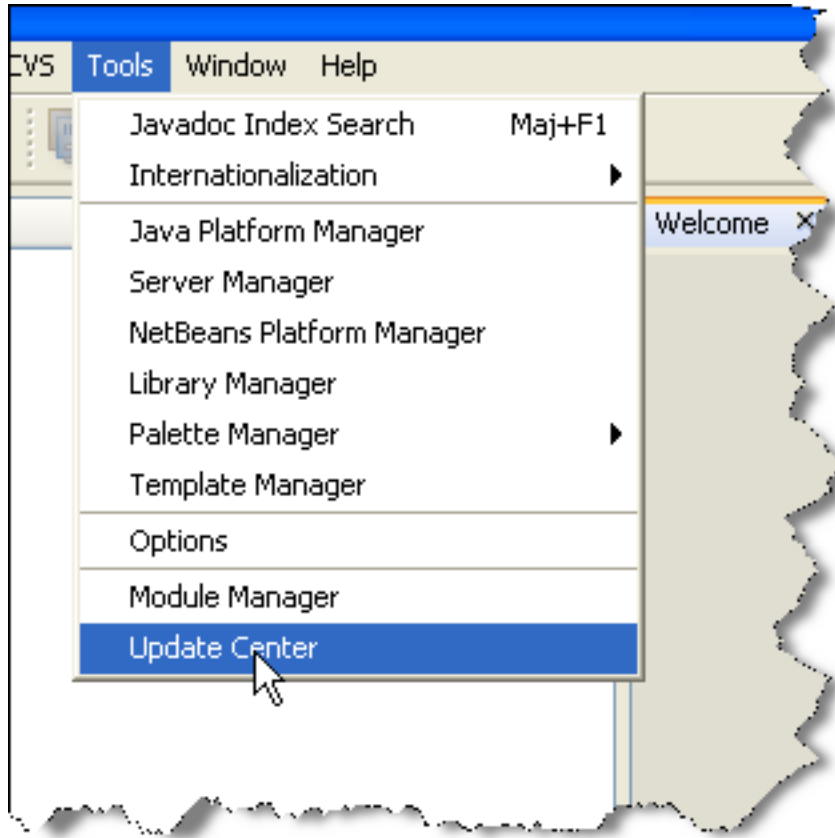
You will get a ZIP file with 16 nbm (Netbeans modules) inside. Unzip the file to a temporary directory.

Installing the Mevenide plug-in

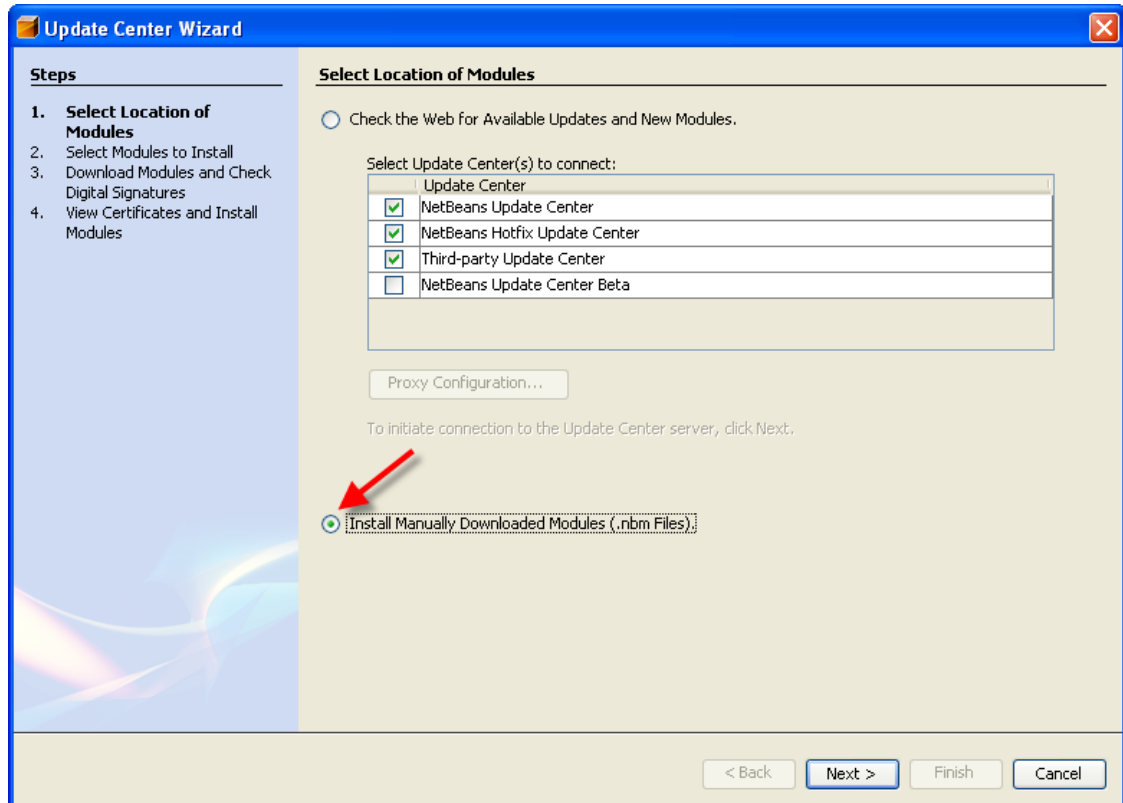
Tip

If you are familiar with Netbeans modules installation, you will probably want to skip this section.

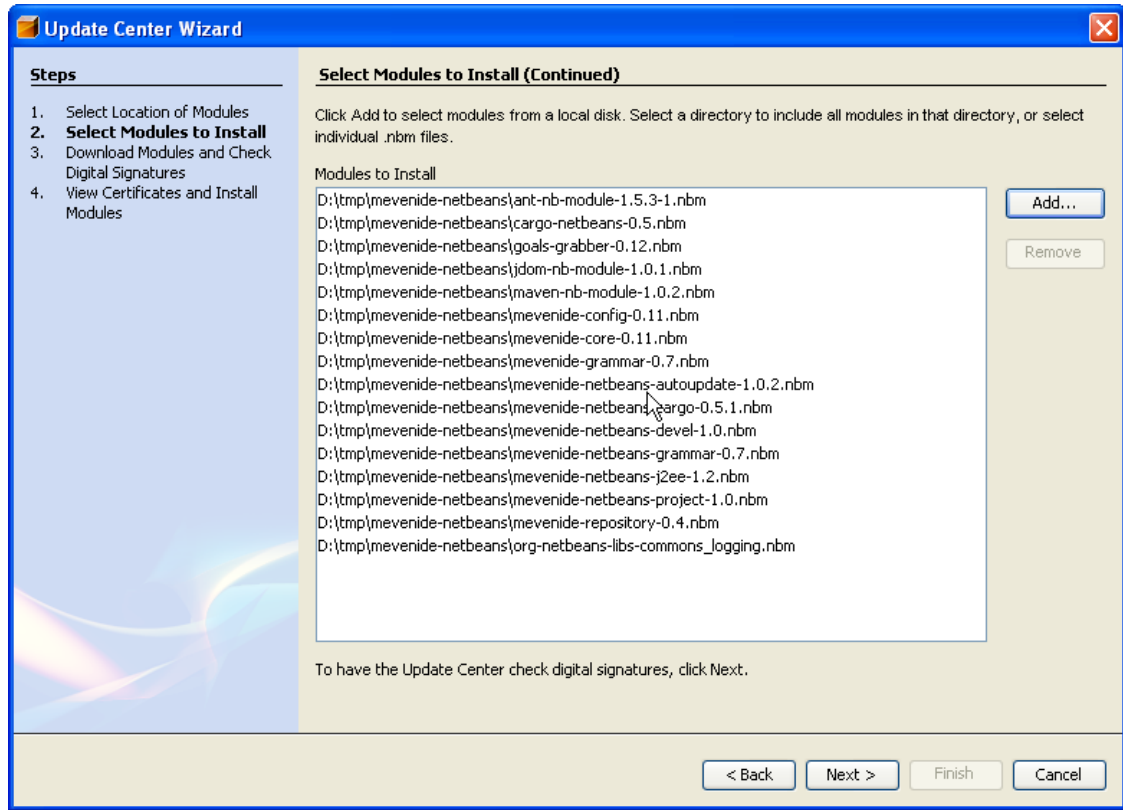
In the Tools menu, choose Update Center



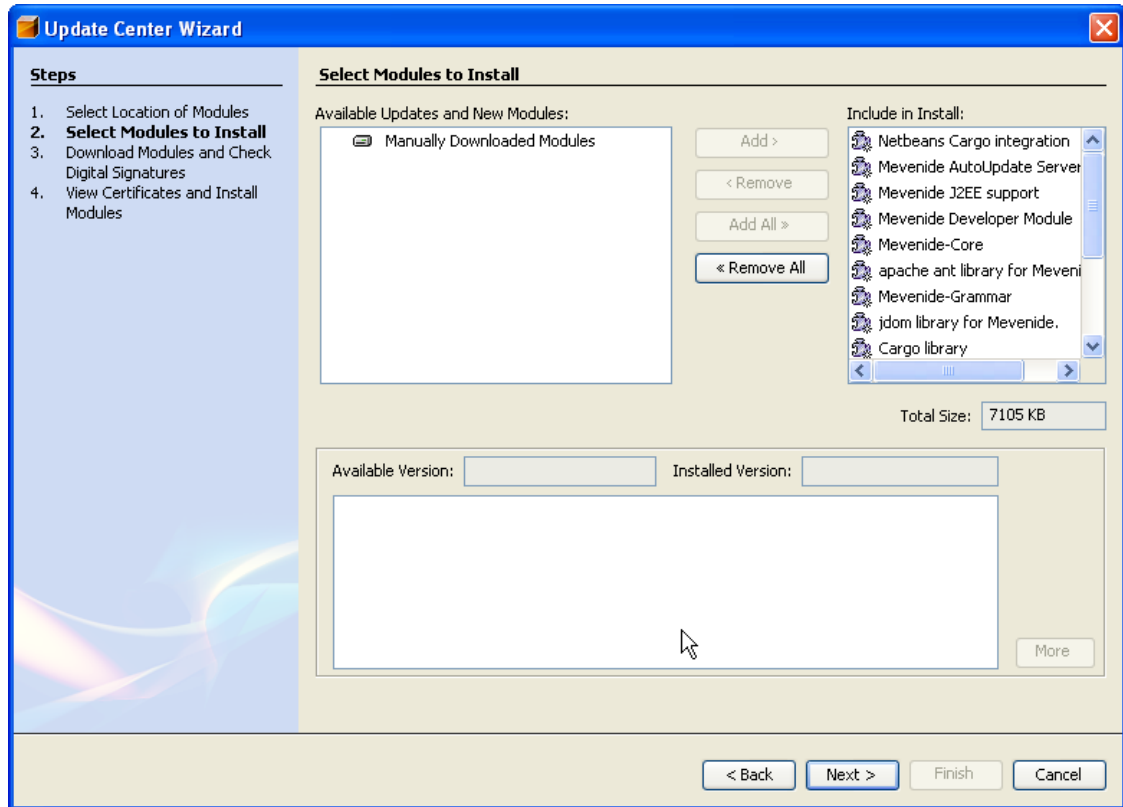
Check 'Install Manually Downloaded Modules (.nbm Files)'



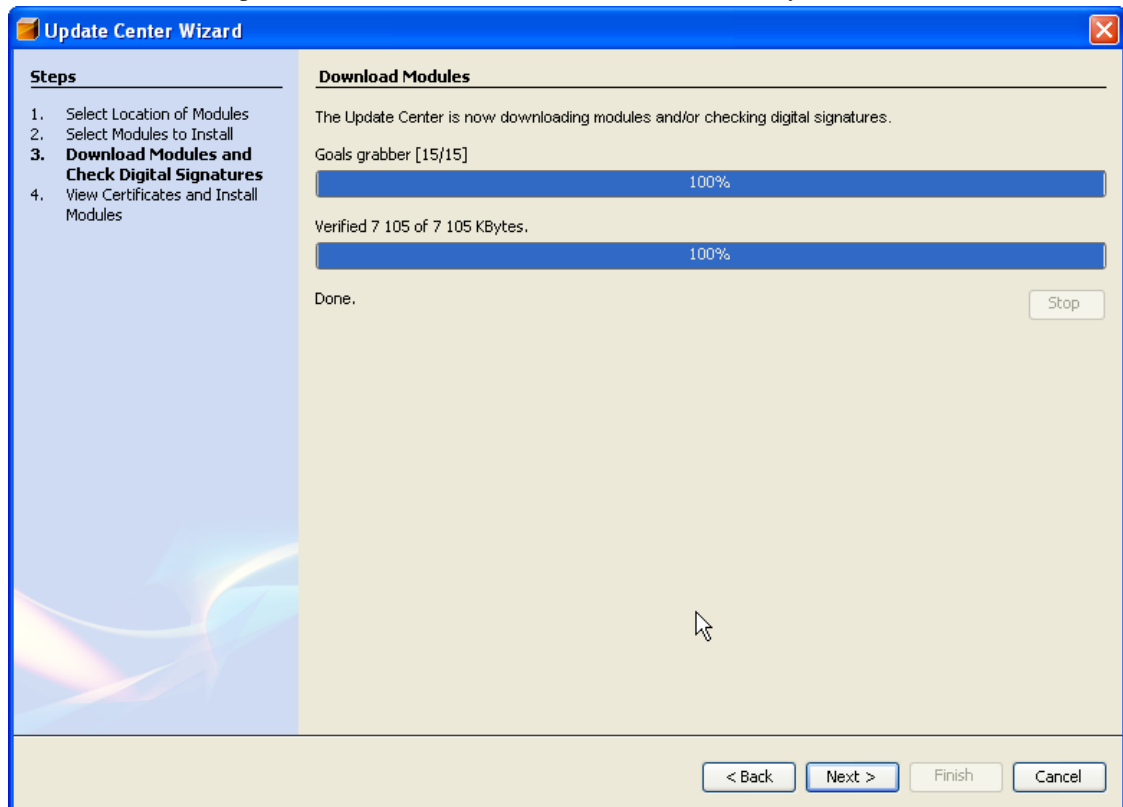
Use the 'Add...' button to add the 16 files you have downloaded.



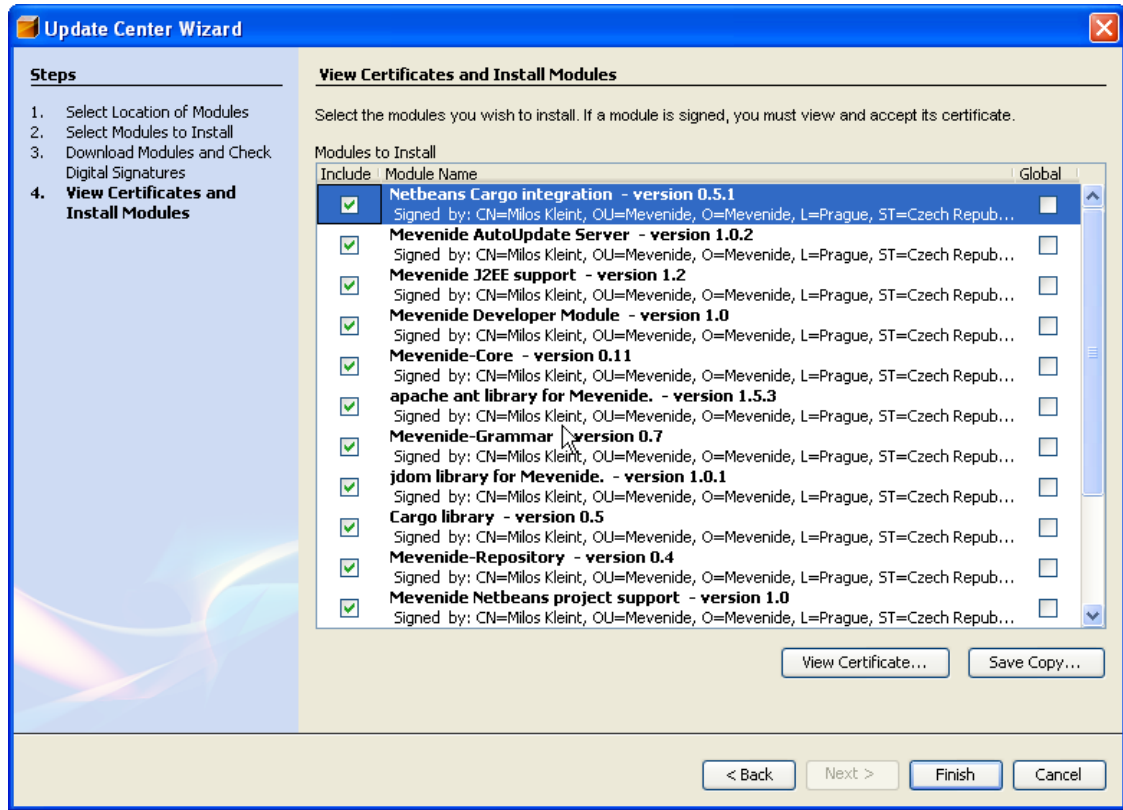
Just click 'Next >'.



You will have to accept the different licences. The modules are then ready for installation.



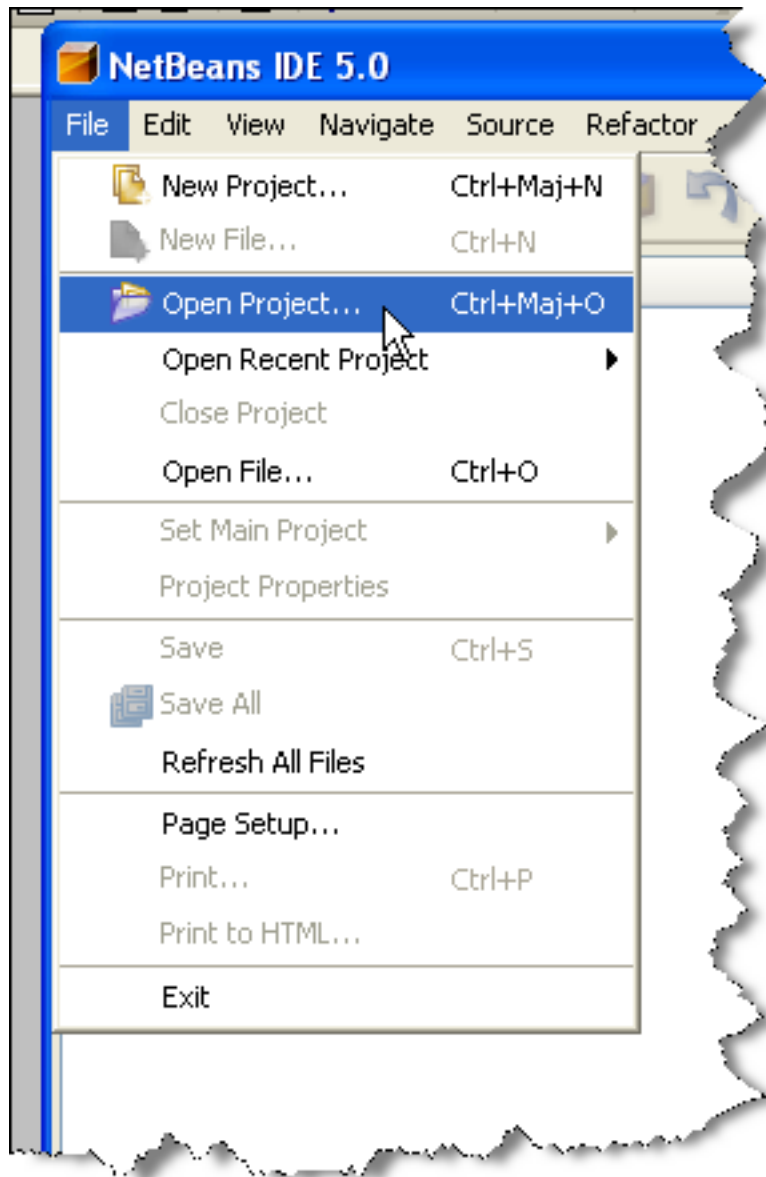
View the certificate (there is only one here) and click 'Finish' to install the modules.



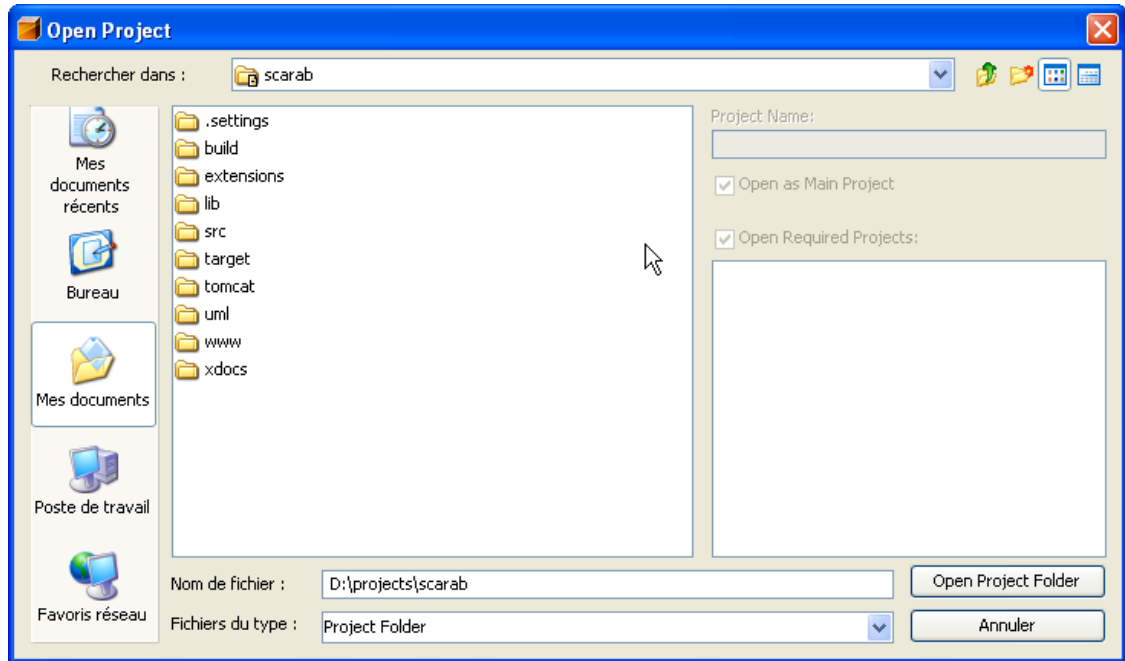
You need now to restart Netbeans to use the Mevenide plug-in you have just installed.

Scarab in-place development

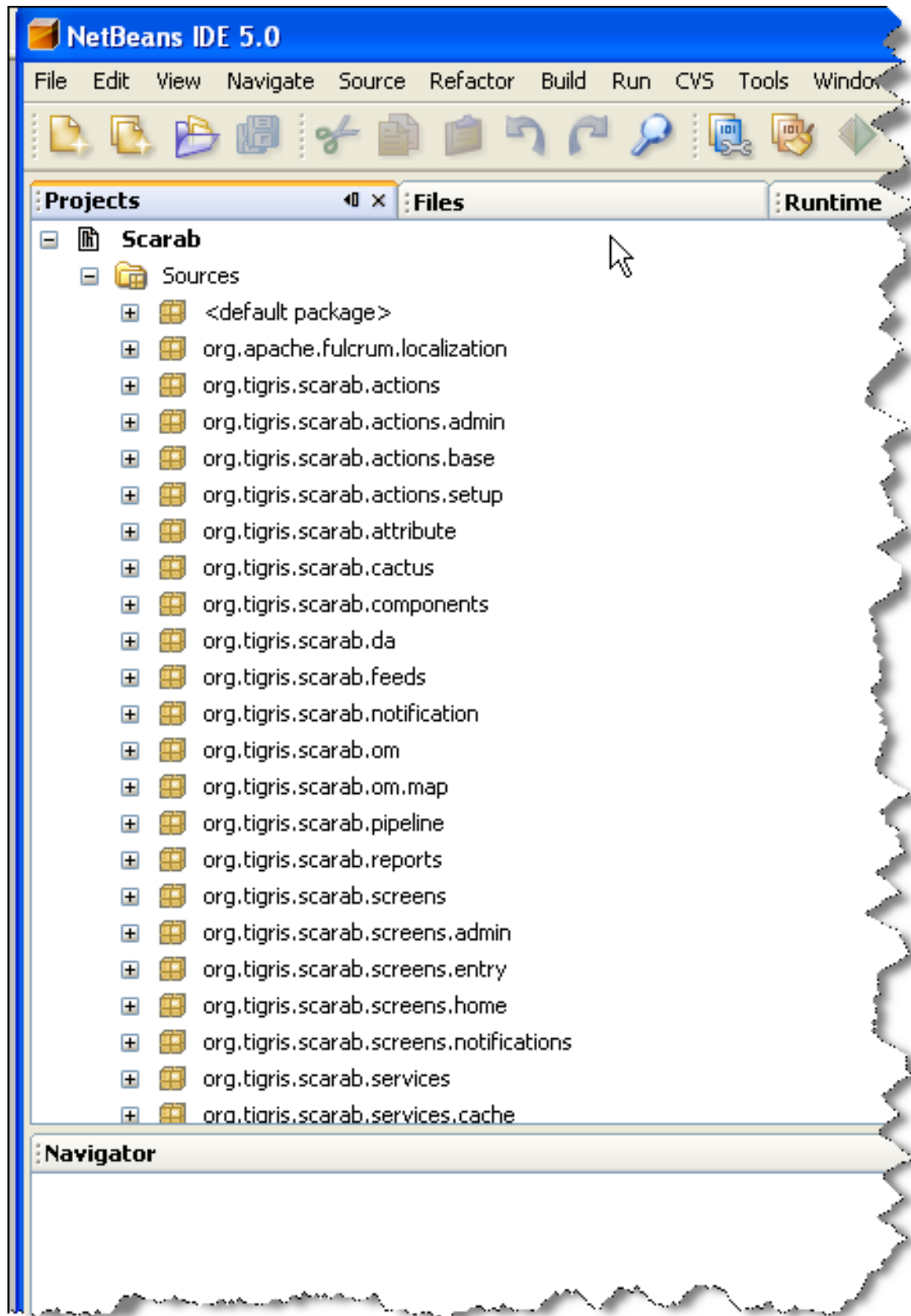
Accessing the Scarab project is now as easy as it can be. In the File menu, choose 'Open Project...'



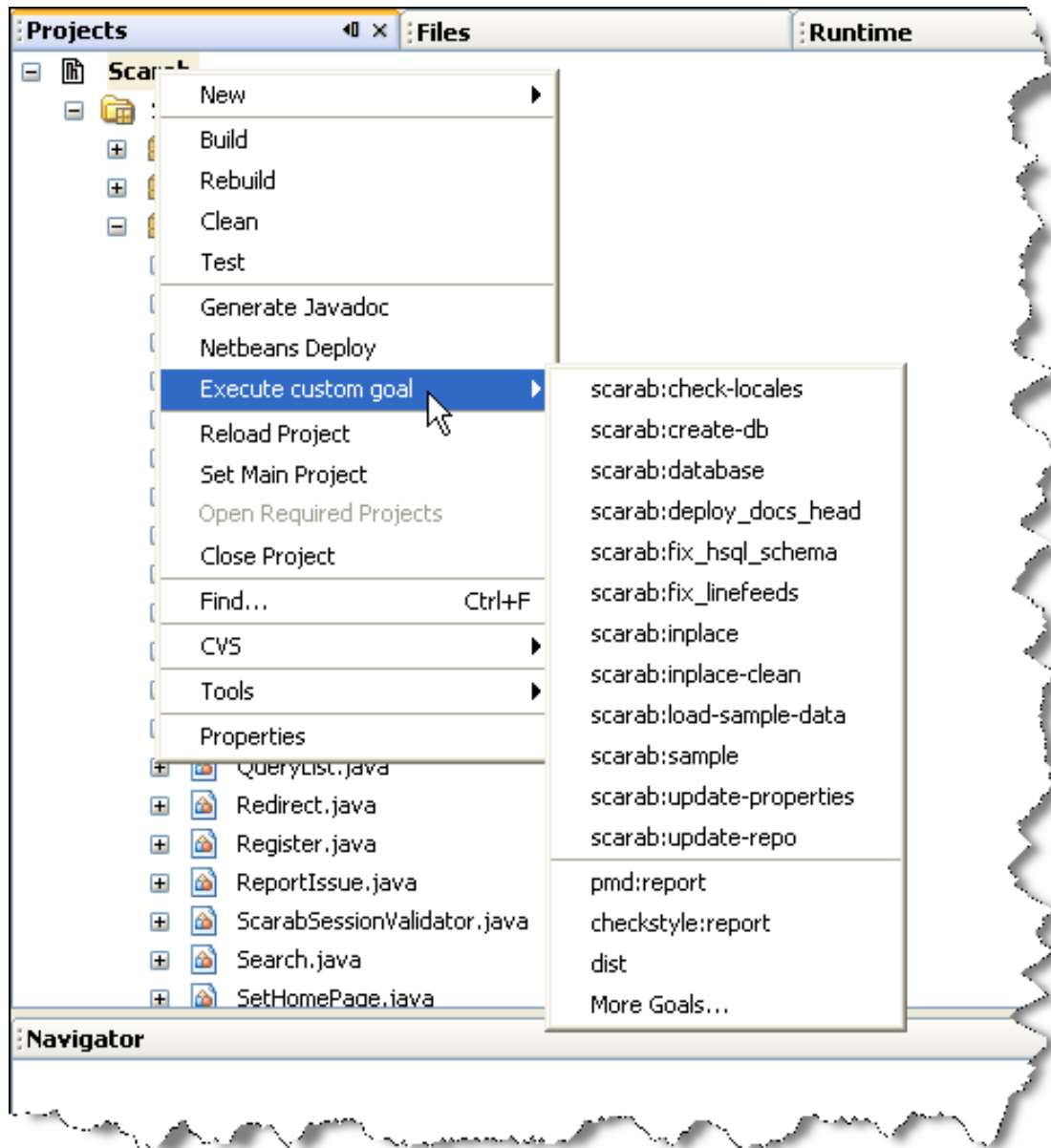
Choose the directory in which you checked out Scarab from the Subversion repository.



There it is !



You have direct access to Scarab custom Maven goals by right-clicking the Scarab project.



Chapter 6. The Scarab data model

Introduction

Scarab data are accessed via Torque, the OR mapping developed for and with the Turbine framework (but Torque is today independent in its way).

The Scarab database schema is formally described in XML. This schema will be explained in three parts, corresponding to the three XML files under `./src/schema`.

- the table associated to the primary key broker (`id-table-schema.xml`);
- the tables used to manage the Scarab users (for authentication and other purposes) and the permissions in the Turbine framework (`turbine-schema.xml`);
- the tables of the Scarab application itself (`scarab-schema.xml`).

From this formal definition of the database schema, Torque generates:

- SQL (DDL) scripts to create tables in the various RDBMS;
- a set of java classes that map the database entities as objects (the so called "peers").

Unique IDs generation: the `id_table` table

Most tables in the Scarab schema have meaningless (large) integers as primary keys. Many RDBMS offer a native mechanism to generate such primary keys (integers, not necessarily in sequence): Oracle SEQUENCE's, auto-incremented columns in MySQL or MSSQL, etc.

One of the Scarab design goals was to be portable across many RDBMS; thus, the generation of primary keys could hardly rely on such native mechanisms. So Scarab uses for this a Torque functionality ("id broker").

The `id_table` table is used to generate primary keys for the various tables in the Scarab schema.

<i>id_table</i>		
PK	ID_TABLE_ID	INTEGER
N	TABLE_NAME	varchar-255
A	NEXT_ID	INTEGER
A	QUANTITY	<input type="text"/> -UnNamed-

Reserved ID ranges

As a rule, everything under 10000 is reserved for development use (default data, sample data etc.)

Table 6.1. ID ranges

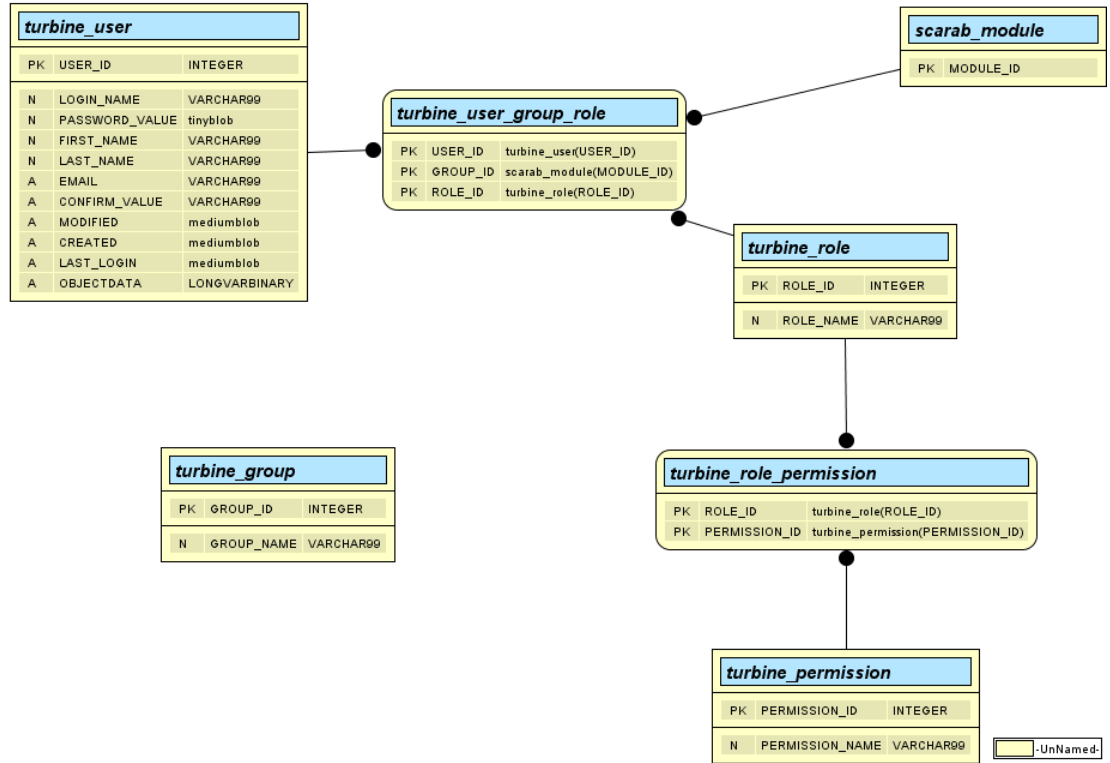
ID range	Use
0-99	Required data.
100-189	Default data.
190-199	Anonymous user data.
200-299	JIRA templates.
300-399	Bugzilla templates.
1000-1999	Sample data.
2000-8999	(Reserved for future use.)
9000-9999	User custom (custom user templates, etc.)

Turbine framework tables

These tables are used to authenticate and manage the Scarab users, their roles and permissions in the different modules.

The tables in this part of the model are:

- **turbine_user** : stores data about the Scarab users, their logins, passwords, etc..
- **turbine_role** : stores the list of roles. Roles are common to all Scarab modules.
- **turbine_permission** : stores the different permissions defined in the Scarab application.
- **turbine_group** : this table may be required by Turbine at runtime but it is not used by Scarab. It is always empty.
- The **turbine_user_group_role** and **turbine_role_permission** tables are many-to-many relations. Their role is explained in the ERD below:

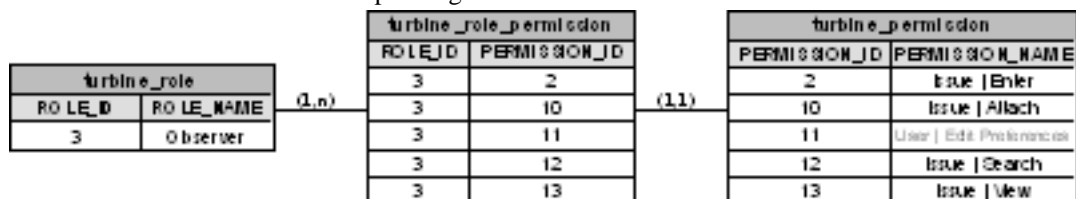


The following screenshot shows the definition of roles in Scarab (with the example data):

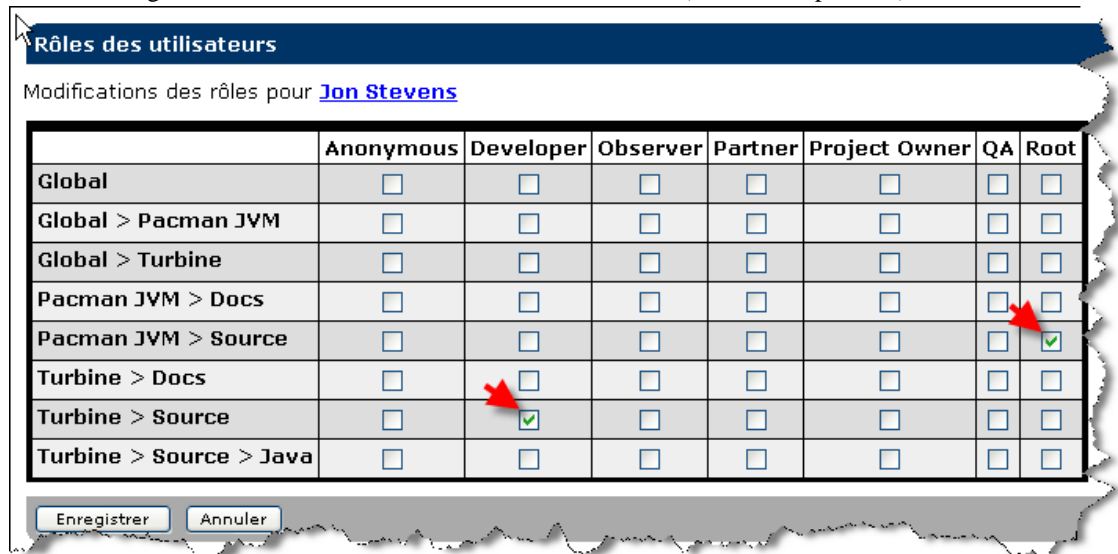
Éditer les droits pour le rôle [Observer]

Permettre	Droit
<input type="checkbox"/>	Domain Admin
<input type="checkbox"/>	Domain Edit
<input type="checkbox"/>	Issue Assign
<input checked="" type="checkbox"/>	Issue Attach
<input type="checkbox"/>	Issue Edit
<input checked="" type="checkbox"/>	Issue Enter
<input type="checkbox"/>	Issue Move
<input checked="" type="checkbox"/>	Issue Search
<input checked="" type="checkbox"/>	Issue View
<input type="checkbox"/>	Item Approve
<input type="checkbox"/>	Item Delete
<input type="checkbox"/>	Module Add
<input type="checkbox"/>	Module Configure
<input type="checkbox"/>	Module Edit
<input type="checkbox"/>	User Approve Roles
<input checked="" type="checkbox"/>	User Edit Preferences
<input type="checkbox"/>	Vote Manage

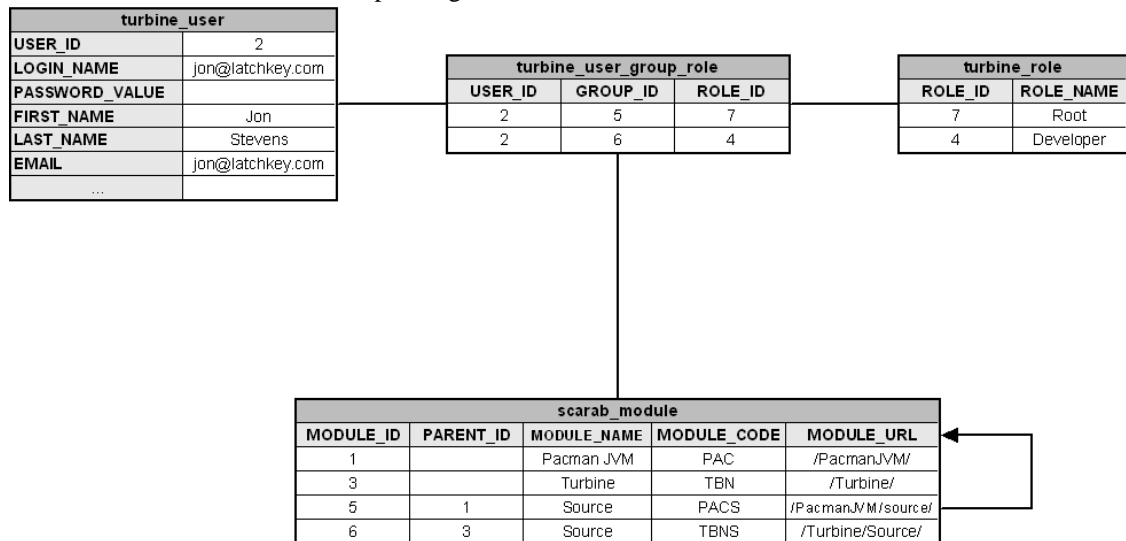
This is an illustration of the corresponding data model:



The following screenshot shows the definition of a user's roles (in the example data):



This is an illustration of the corresponding data model:

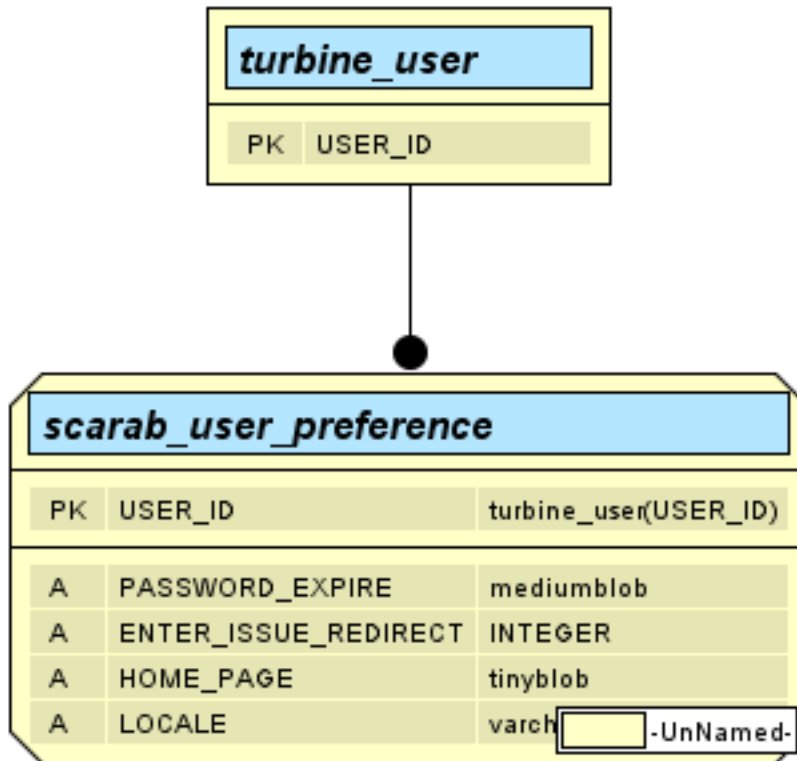


Scarab tables

Entities related to users

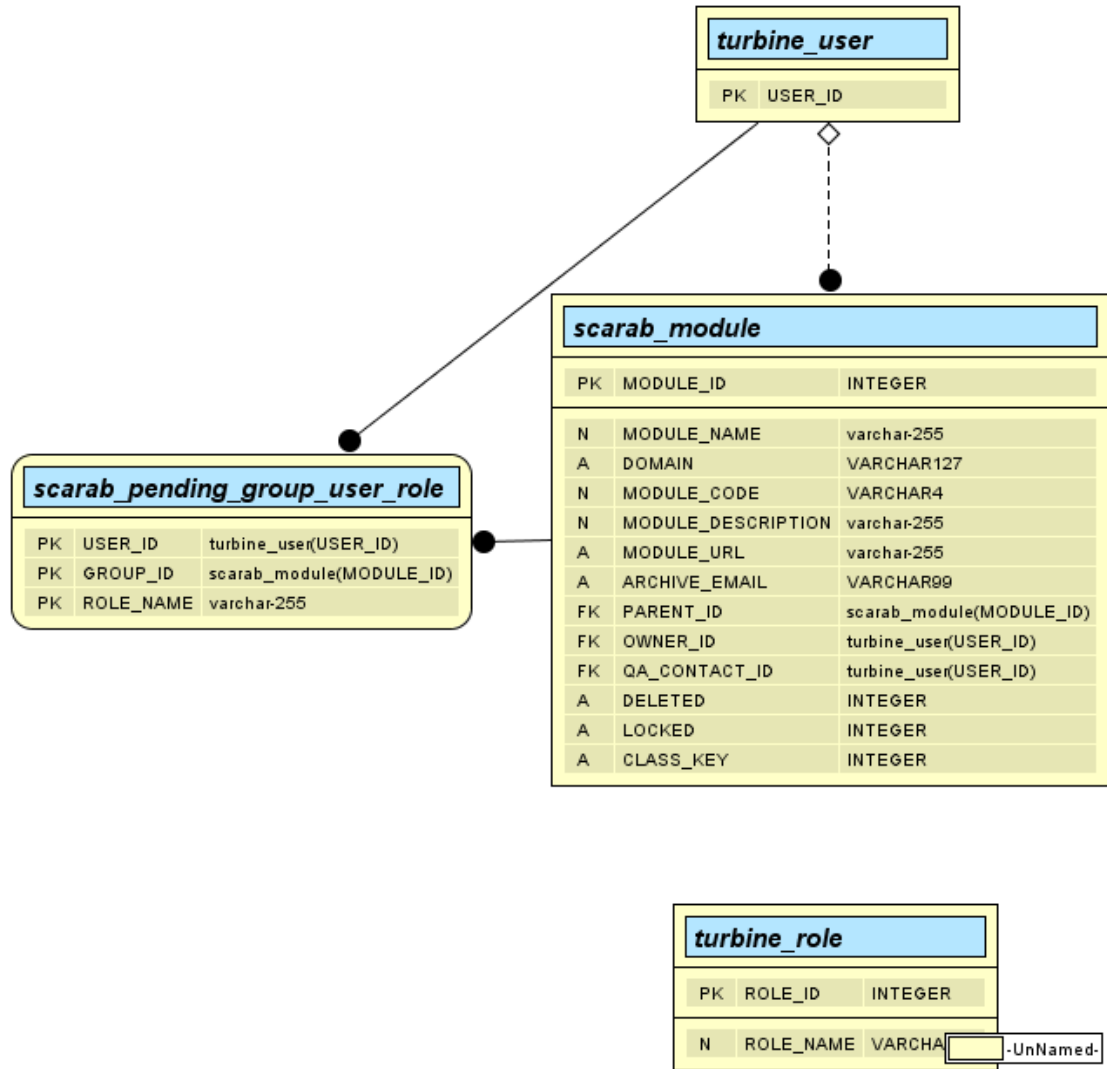
Before entering the heart of the Scarab data model, let us mention two series of simple entities related to the users as defined just above.

Preferences



Pending role demands

The `scarab_pending_group_user_role` table stores temporarily the role demands as supplied by users. These demands must be approved by the module administrator.



Modules

The list of modules and the corresponding data are stored in the `scarab_module` table.

scarab_module		
PK	MODULE_ID	INTEGER
N	MODULE_NAME	varchar-255
A	DOMAIN	VARCHAR127
N	MODULE_CODE	VARCHAR4
N	MODULE_DESCRIPTION	varchar-255
A	MODULE_URL	varchar-255
A	ARCHIVE_EMAIL	VARCHAR99
FK	PARENT_ID	scarab_module(MODULE_ID)
FK	OWNER_ID	turbine_user(USER_ID)
FK	QA_CONTACT_ID	turbine_user(USER_ID)
A	DELETED	INTEGER
A	LOCKED	INTEGER
A	CLASS_KEY	INTEGER <input type="text" value="-UnNamed-"/>

Modules	
Nom	Description
Global	Built-in root module
Global > Pacman JVM	Sample project
Global > Turbine	The Turbine Project
Pacman JVM > Docs	Documentation
Pacman JVM > Source	Source
Turbine > Docs	Documentation
Turbine > Source	Source
Turbine > Source > Java	Java

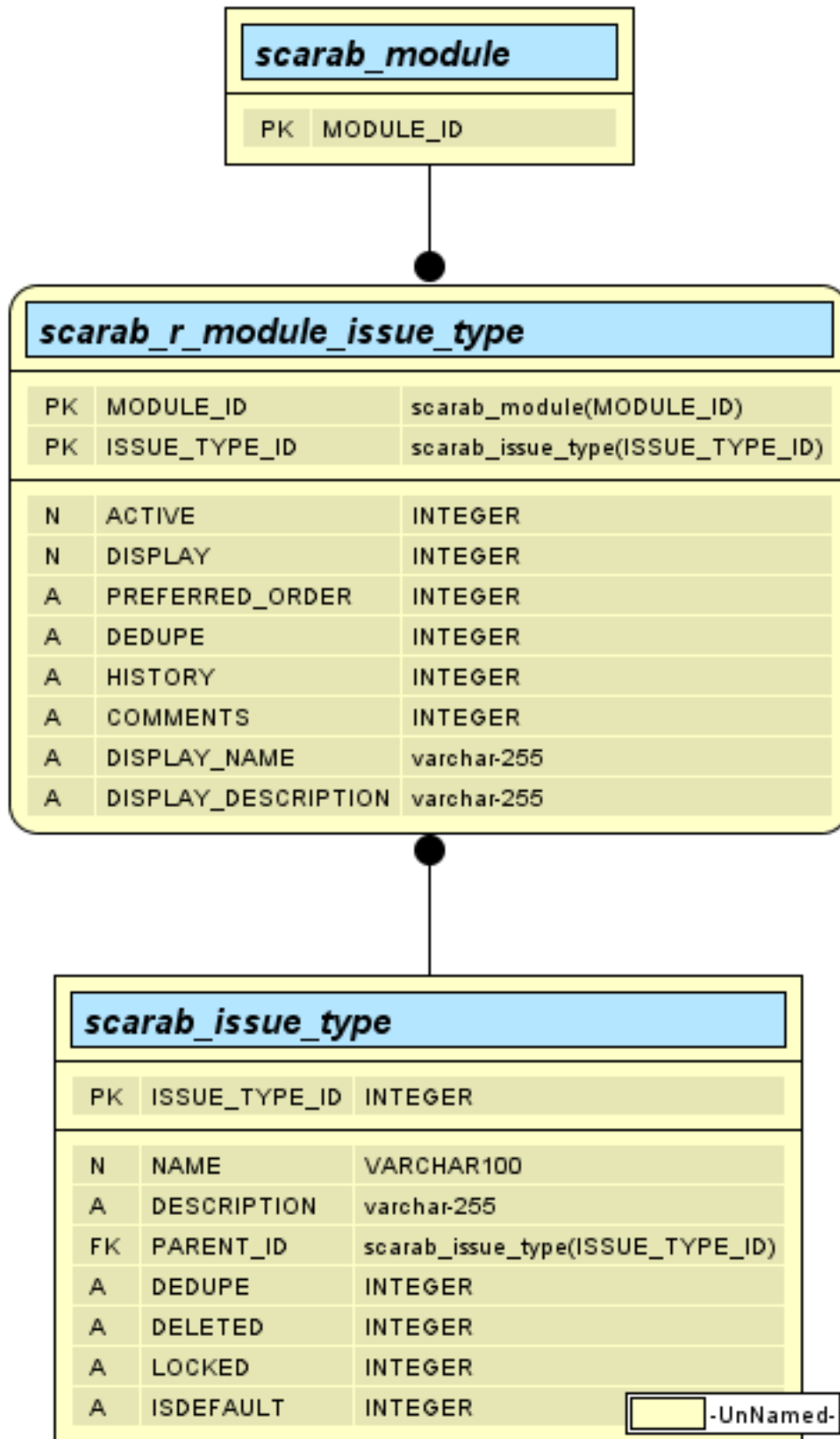
Tip

Our QA manager asked one day for a table that would give a correspondance between the module codes and their names. The slight difficulty here is that the module name is often meaningful with the name of the parent module. It is easy to obtain this kind of table from SCARAB_MODULE with the following SQL request:

```
select M1.MODULE_CODE, M2.MODULE_NAME, M1.MODULE_NAME from SCARAB_MODULE
M1, SCARAB_MODULE M2 where M1.PARENT_ID = M2.MODULE_ID order by
M1.MODULE_CODE;
```

There it is!

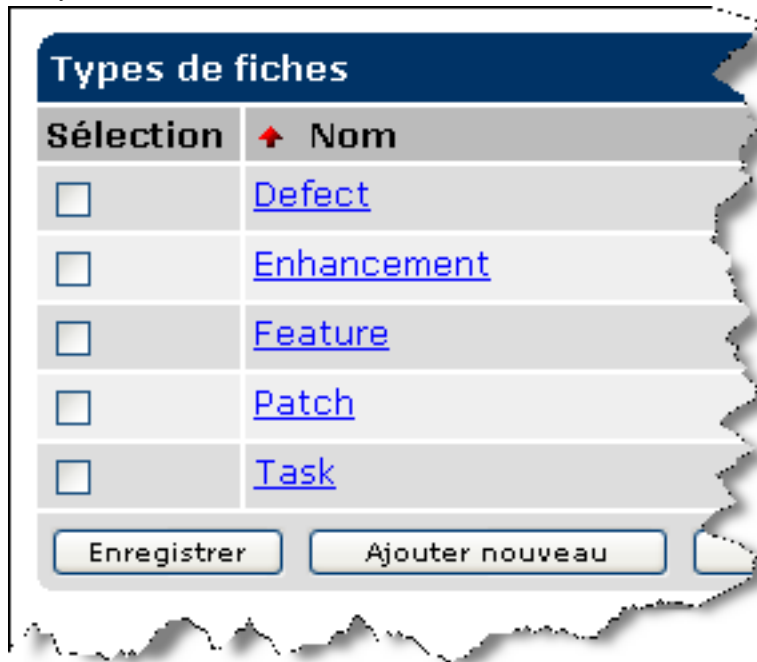
Issue types



Issue types are stored in the `scarab_issue_type` table. Modules and issue types are related to each

other in a many-to-many relationship through the `scarab_r_module_issue_type` relation table.

The global issue types, as illustrated below, are related to the "Global" module (the ID of this module is always 0).



The screenshot shows a web interface with a title bar "Types de fiches" in a dark blue header. Below the header is a table with two columns: "Sélection" and "Nom". The "Sélection" column contains five checkboxes, all of which are unchecked. The "Nom" column contains five entries: "Defect", "Enhancement", "Feature", "Patch", and "Task", each displayed as a blue hyperlink. Below the table are two buttons: "Enregistrer" and "Ajouter nouveau". The interface has a light gray background and a white border.

Sélection	Nom
<input type="checkbox"/>	Defect
<input type="checkbox"/>	Enhancement
<input type="checkbox"/>	Feature
<input type="checkbox"/>	Patch
<input type="checkbox"/>	Task

Enregistrer Ajouter nouveau

Attributes

scarab_attribute		
PK	ATTRIBUTE_ID	INTEGER
N	ATTRIBUTE_NAME	varchar-255
FK	ATTRIBUTE_TYPE_ID	scarab_attribute_type(ATTRIBUTE_TYPE_ID)
A	PERMISSION	varchar-255
FK	REQUIRED_OPTION_ID	scarab_attribute_option(OPTION_ID)
N	DESCRIPTION	varchar-255
A	ACTION	varchar-255
FK	CREATED_BY	turbine_user(USER_ID)
A	CREATED_DATE	mediumblob
A	DELETED	INTEGER



scarab_attribute_type		
PK	ATTRIBUTE_TYPE_ID	INTEGER
FK	ATTRIBUTE_CLASS_ID	scarab_attribute_class(ATTRIBUTE_CLASS_ID)
N	ATTRIBUTE_TYPE_NAME	varchar-255
A	JAVA_CLASS_NAME	varchar-255
A	VALIDATION_KEY	VARCHAR20



scarab_attribute_class		
PK	ATTRIBUTE_CLASS_ID	INTEGER
N	ATTRIBUTE_CLASS_NAME	varchar-255
N	ATTRIBUTE_CLASS_DESC	varchar-255
A	JAVA_CLASS_NAME	varchar-255

-Unnamed-

Attributs globaux | Attributs utilisateur globaux

Attributs globaux

Filtrer cette liste N'importe lequel

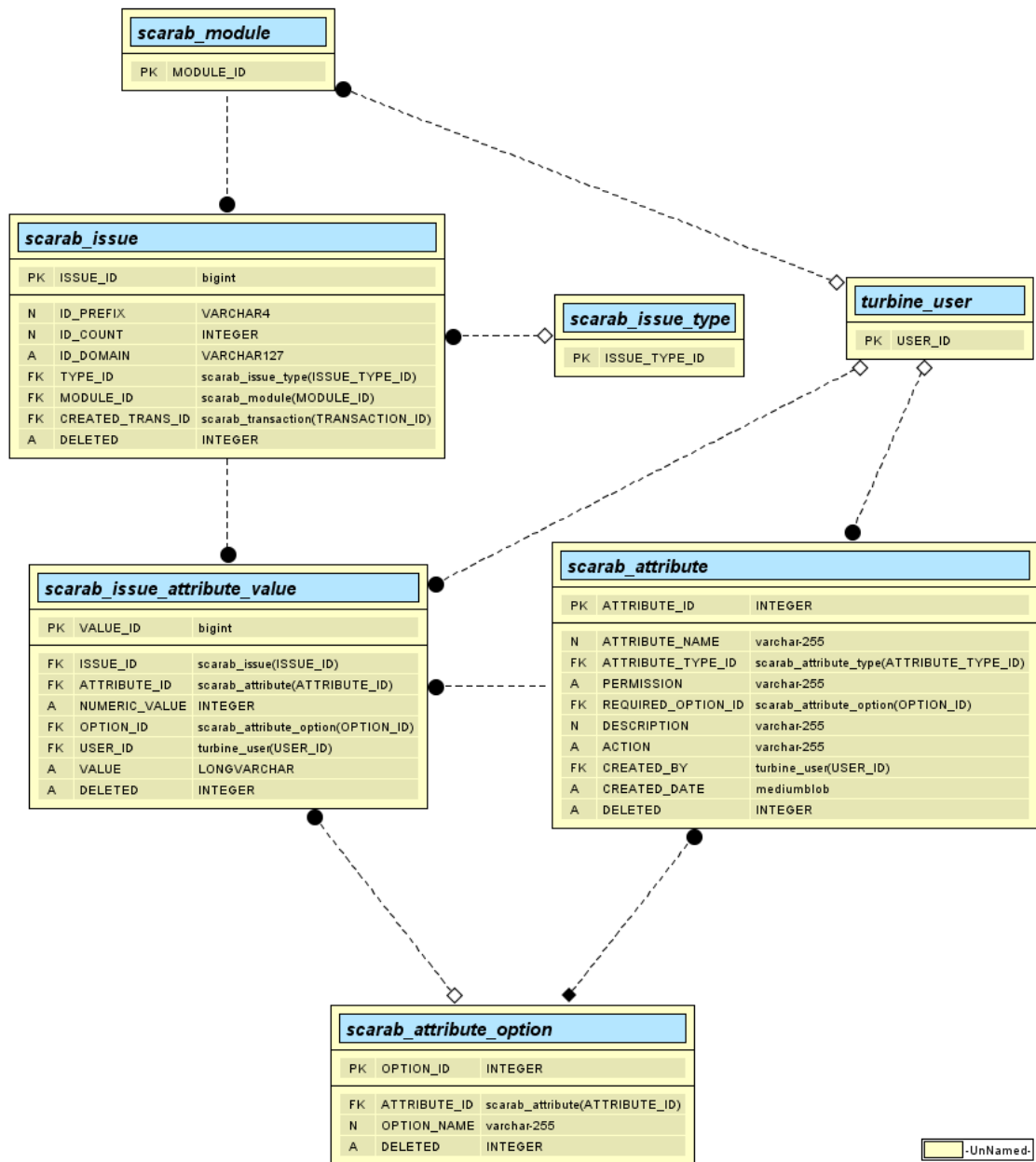
Sélection	Nom	Description	Type de
<input type="checkbox"/>	Description	Description	long
<input type="checkbox"/>	FunctionalArea	FunctionalArea	Dropd
<input type="checkbox"/>	OperatingSystem	OperatingSystem	Dropd
<input type="checkbox"/>	Platform	Platform	Dropd
<input type="checkbox"/>	Priority	Priority	Dropd
<input type="checkbox"/>	Resolution	Resolution	Dropd
<input type="checkbox"/>	Severity	Severity	Dropd
<input type="checkbox"/>	Status	Status	Dropd
<input type="checkbox"/>	Summary	Summary	string
<input type="checkbox"/>	Vote	Vote	Dropd

Attributs globaux | Attributs utilisateur globaux

Attributs utilisateur globaux

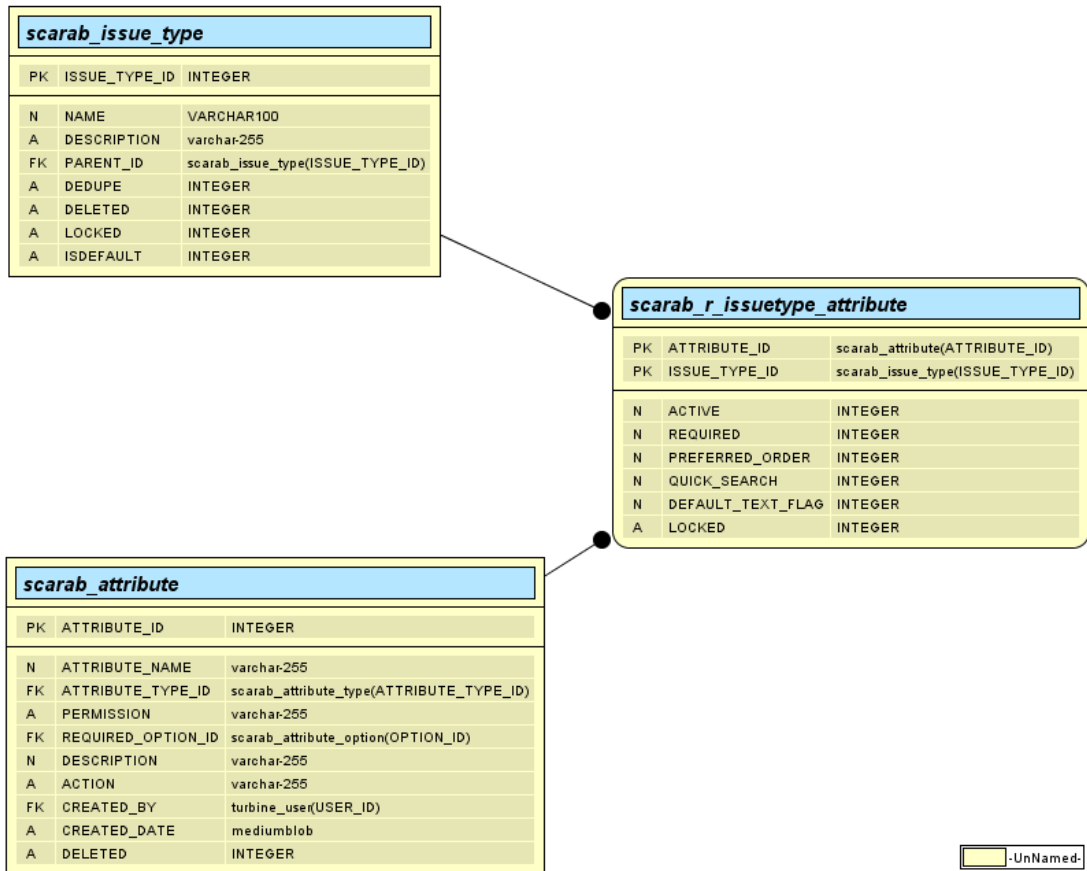
Sélection	Nom	Description	Action mél	Droit de base
<input type="checkbox"/>	AssignedCC	CcAttribute	CC:: <input type="button" value="v"/>	Issue View <input type="button" value="v"/>
<input type="checkbox"/>	AssignedTo	AssignedTo	à: <input type="button" value="v"/>	Issue Edit <input type="button" value="v"/>

Issues

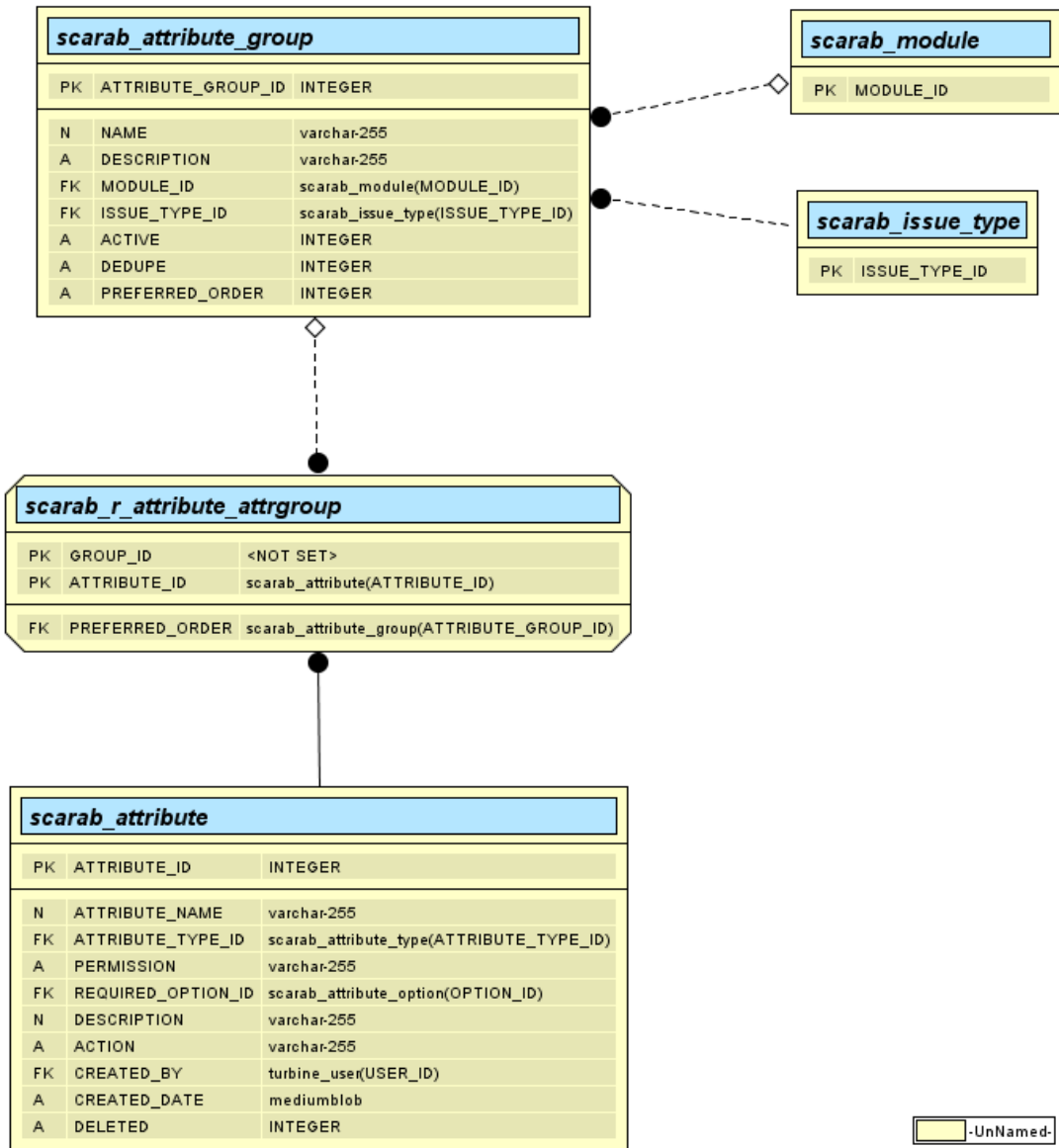


Attributes

Attributs	0 Personnes	0 Commentaires	1 Dependencies	0 A
Identifiant de la fiche	PACS1 (Defect)			
Créée par	Elicia David 1 sept. 2001 22:30:00 G			
Modifiée par	Elicia David 1 sept. 2001 22:30:00 G			
AttributeGroupOne (* indique un champ obligatoire)				
* Platform	<input type="text" value="SGI"/>			
* OperatingSystem	<input type="text" value="Linux"/>			
* Summary	<input type="text" value="Docs are out of date."/>			
AttributeGroupTwo (* indique un champ obligatoire)				
* Description	<input type="text" value="Documents are not as current a"/>			
Status	<input type="text" value="New"/>			
Resolution	<input type="text" value="Choisir..."/>			
Priority	<input type="text" value="Low"/>			
Vote	<input type="text" value="Choisir..."/>			
Severity	<input type="text" value="Minor"/>			
FunctionalArea	<input type="text" value="Setup"/>			

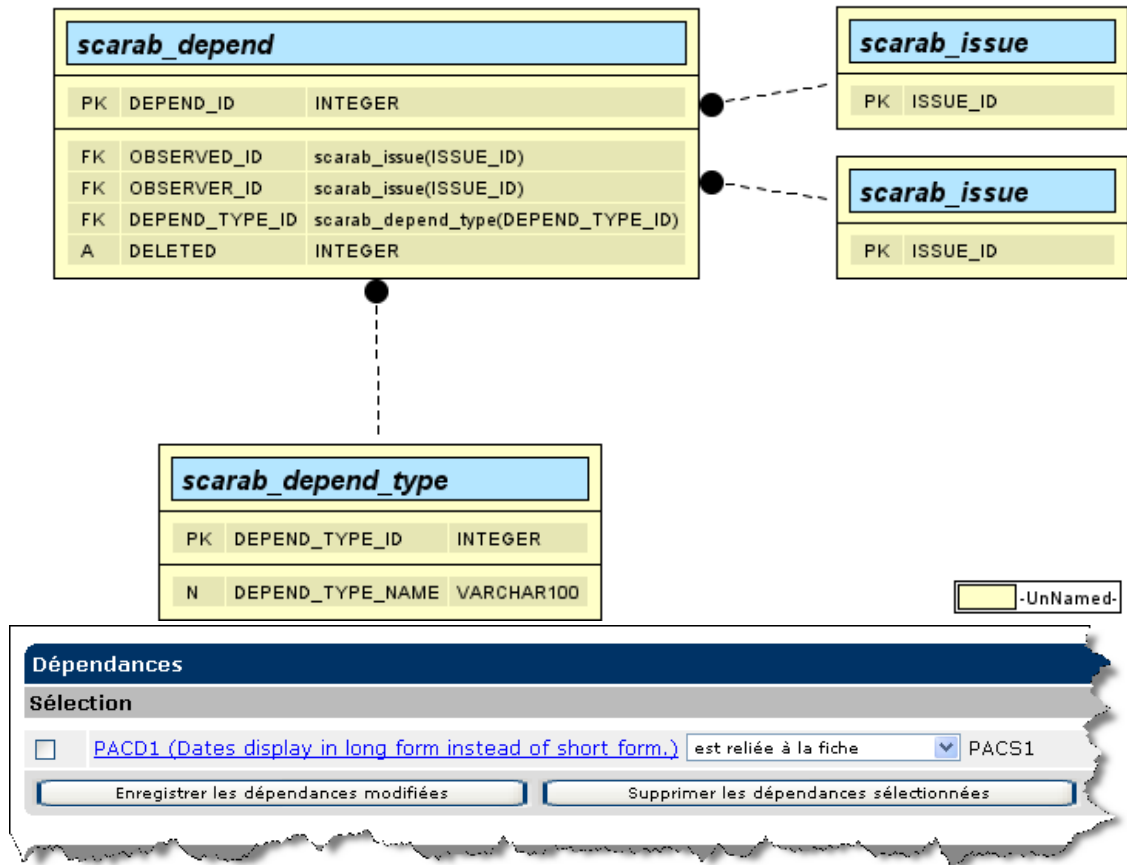


Attribute groups

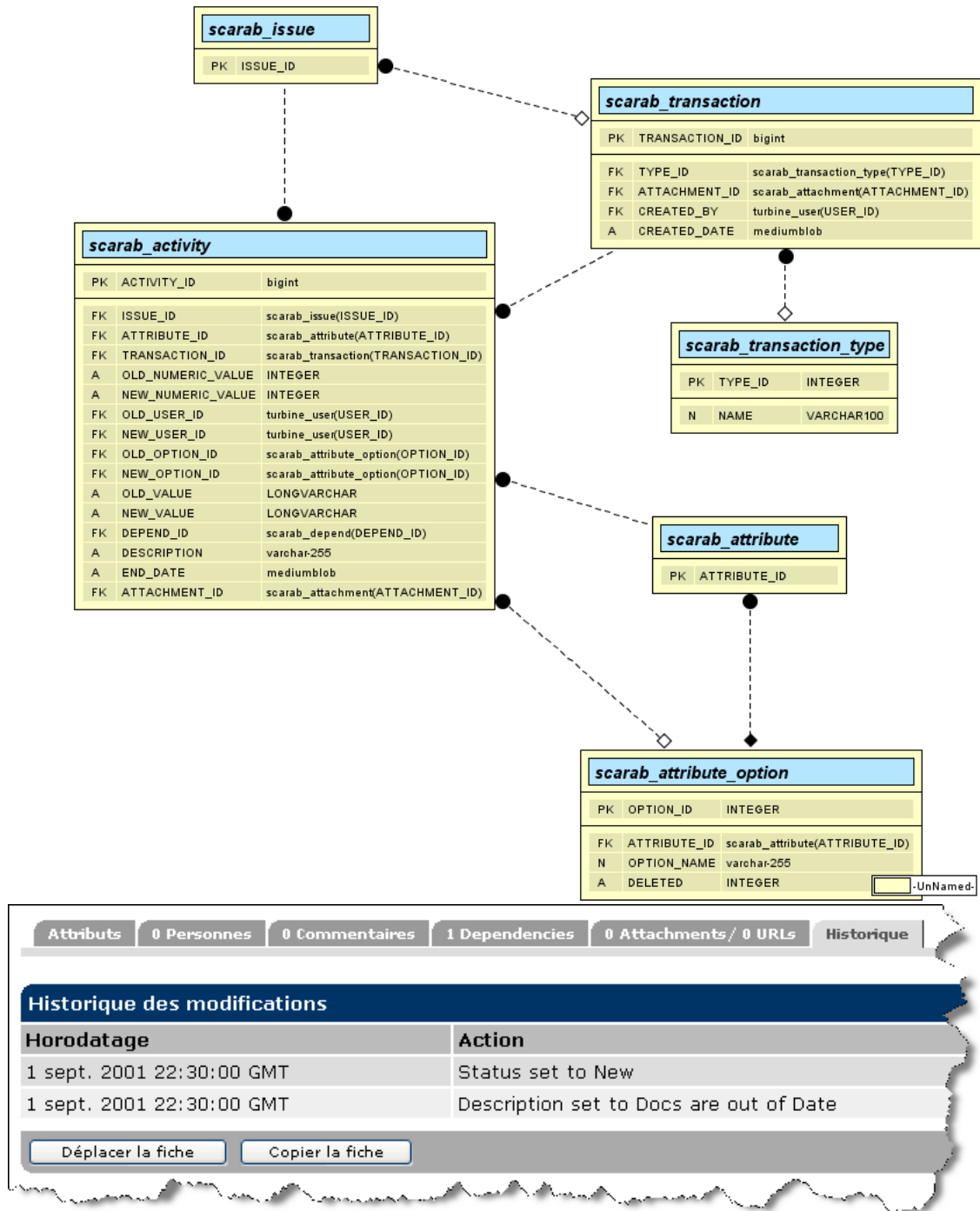


Attachments

Dependencies

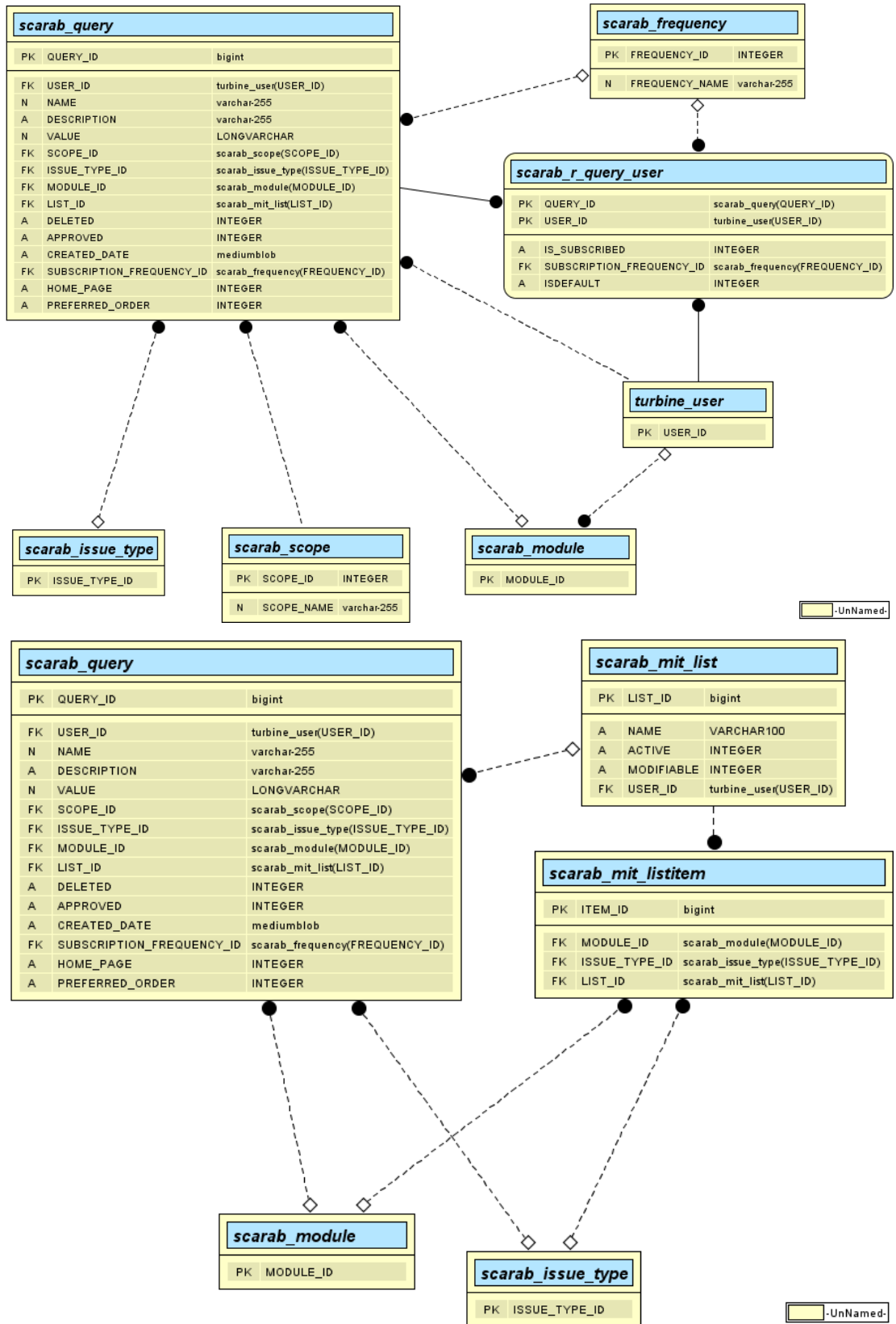


History

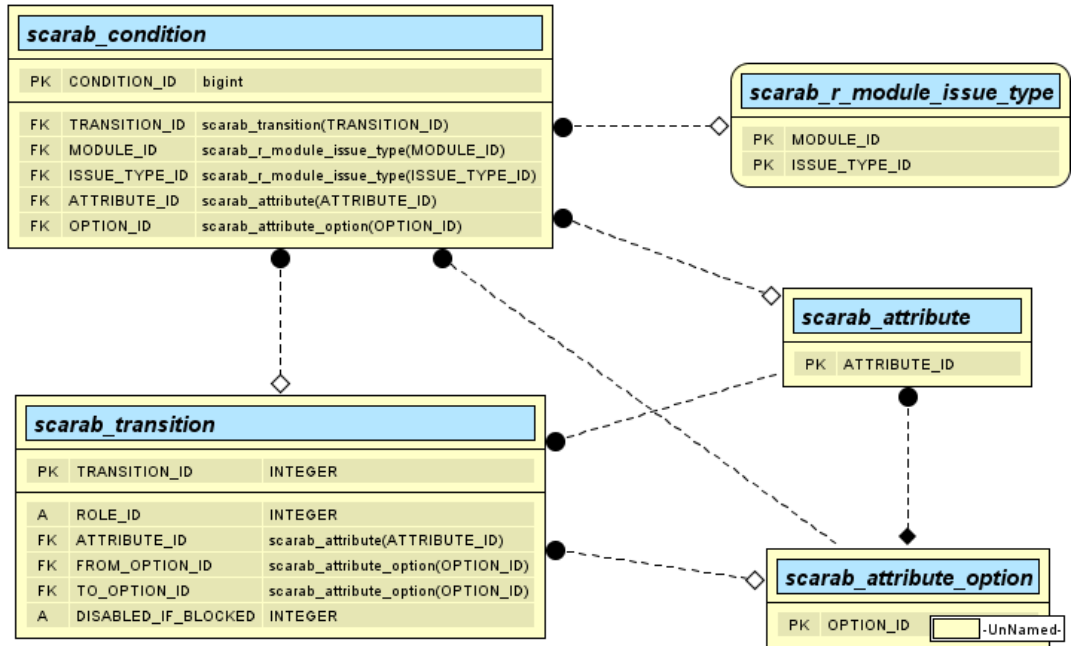


Queries

The Scarab data model



Workflow



Chapter 7. Workflow

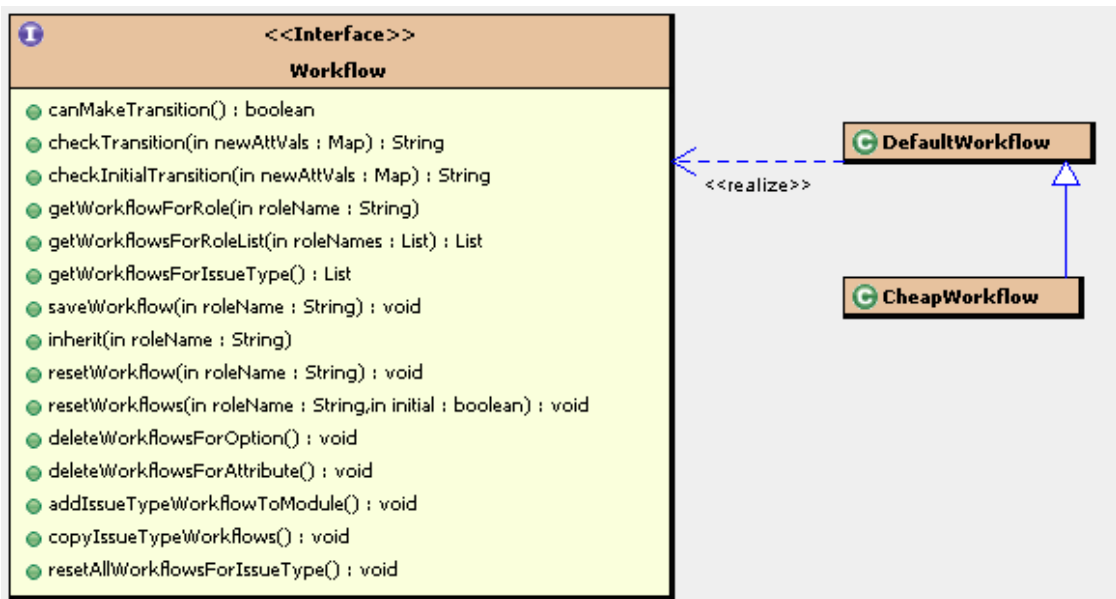
Introduction

As has been said in the *Workflow* chapter of the *Scarab User's Guide*, Scarab has no "hard-coded" workflow. When saving issues, all state transitions are allowed and there is no interaction with any external system ever.

Now let's have a look at how this is implemented.

Workflows in Scarab

At the programming level, a workflow is any java class that implements the `org.tigris.scarab.workflow.Workflow` interface (the source code of this interface is located under `src/java`).



The workflow is then instantiated dynamically at runtime by Turbine, according to a declaration in `Scarab.properties` (this file is located at runtime in the web application tree under `WEB-INF/conf`).

```
# The workflow tool
# scarab.workflow.classname=org.tigris.scarab.workflow.DefaultWorkflow
scarab.workflow.classname=org.tigris.scarab.workflow.CheapWorkflow
```

DefaultWorkflow

This was the default workflow for all versions of Scarab until milestone 19.

This workflow is implemented in the `org.tigris.scarab.workflow.DefaultWorkflow`

class. This class is a stub, that does nothing and allows all transitions.

BasicWorkflow

This is the default workflow for all Scarab versions from milestone 20. The way it works from the user point of view is explained in the *User's Guide*.

This workflow is implemented in the `org.tigris.scarab.workflow.CheapWorkflow` class (against the odds). It implements the *Workflow* interface indirectly by inheriting from the default workflow, `DefaultWorkflow` probably not to redefine certain default behaviours.

Defining a new workflow

To define a new workflow, using other mechanisms, communicating with one or several external systems or more adapted to the needs or processes of your organization, you just need to:

- write a java class that implements the `org.tigris.scarab.workflow.Workflow` interface (either directly or by inheriting from `org.tigris.scarab.workflow.DefaultWorkflow`);
- set the name of this class in the `Scarab.properties` mentioned above.

Using WfmOpen as a Scarab workflow engine

Someone wrote an interface between Scarab and the WfmOpen open-source workflow engine. This implementation, or the base of it, is currently stored in the SourceForge tracker at this URL: http://sourceforge.net/tracker/index.php?func=detail&aid=961620&group_id=76143&atid=546206 [http://sourceforge.net/tracker/index.php?func=detail&aid=961620&group_id=76143&atid=546206].

The status of this development is unknown.

Chapter 8. Accessing Scarab via XML-RPC

It is possible to access remotely the Scarab functionalities via XML-RPC.

An example is supplied in the Scarab sources (under `src/java`):

- la classe `org.tigris.scarab.util.SimpleHandler.java` (server, in Scarab) is used by the Scarab-Subversion integration (read the corresponding chapter in the *Scarab User's Guide*);
- la classe `org.tigris.scarab.util.SimpleHandlerClient.java` (client, in another application) is a simple example of using the server above.