

# **Scarab - Guide du développeur**

## **Scarab pour les développeurs de Scarab**

**The Scarab Development Team**

---

# Scarab - Guide du développeur: Scarab pour les développeurs de Scarab

The Scarab Development Team

---

---

---

---

## Table of Contents

|   |    |
|---|----|
| 1. Architecture de Scarab .....                                     | 1  |
| Scarab à vol d'oiseau .....   | 1  |
| Une application web MVC construite sur le framework Turbine .....   | 1  |
| Les principaux composants de Scarab .....                           | 2  |
| Intake .....  | 2  |
| Turbine (au sens strict) .....                                      | 2  |
| Les services : Fulcrum .....  | 3  |
| Le microconteneur : YAAFI (Avalon) .....                            | 3  |
| Mapping objet/relationnel : Torque et les classes <i>Peer</i> ..... | 4  |
| La vue : Velocity .....   | 4  |
| 2. Comprendre le build avec Ant .....                               | 6  |
| \$SCARAB_ROOT/build/build.xml .....                                 | 6  |
| \$SCARAB_ROOT/src/conf/conf/build.xml .....                         | 6  |
| 3. Maven .....  | 8  |
| Maven 1 .....   | 8  |
| Construire Scarab avec Maven .....                                  | 8  |
| Build normal .....  | 8  |
| Build abrégé .....  | 8  |
| Exécuter les tests .....  | 8  |
| 4. Développement Scarab avec Eclipse .....                          | 10 |
| Obtenir les sources de Scarab .....                                 | 10 |
| Installer le plug-in Subclipse .....                                | 10 |
| Se connecter au repository Subversion de Scarab .....               | 10 |
| Développement de Scarab dans Eclipse .....                          | 19 |
| Installer le plug-in Tomcat de Sysdeo .....                         | 19 |
| Générer les fichiers nécessaires avec Maven .....                   | 20 |
| Configurer et utiliser le plug-in Tomcat .....                      | 20 |
| 5. Développement Scarab avec Netbeans .....                         | 22 |
| Récupérer les sources de Scarab .....                               | 22 |
| Récupérer le plug-in Mevenide (intégrationMaven-Netbeans) .....     | 22 |
| Installer le plug-in Mevenide .....                                 | 22 |
| Développement de Scarab dans Netbeans .....                         | 26 |
| 6. Le modèle de données de Scarab .....                             | 31 |
| Introduction .....  | 31 |
| Générer des identifiants uniques : la table <b>id_table</b> .....   | 31 |
| Intervalles d'identifiants réservés .....                           | 32 |
| Les tables du framework Turbine .....                               | 32 |
| Les tables de l'application Scarab .....                            | 35 |
| Entités liées à l'utilisateur .....                                 | 35 |
| Modules .....   | 37 |
| Types de fiches .....   | 39 |
| Attributs .....   | 41 |
| Fiches .....  | 43 |
| Requêtes .....  | 49 |
| Workflow .....  | 51 |
| 7. Workflow .....   | 52 |
| Introduction .....  | 52 |
| Les workflows dans Scarab .....                                     | 52 |
| DefaultWorkflow .....   | 52 |
| BasicWorkflow .....   | 53 |
| Définir un nouveau workflow .....                                   | 53 |
| Utilisation de WfmOpen comme moteur de workflow pour Scarab .....   | 53 |
| 8. Accéder à Scarab par XML-RPC .....                               | 54 |

---

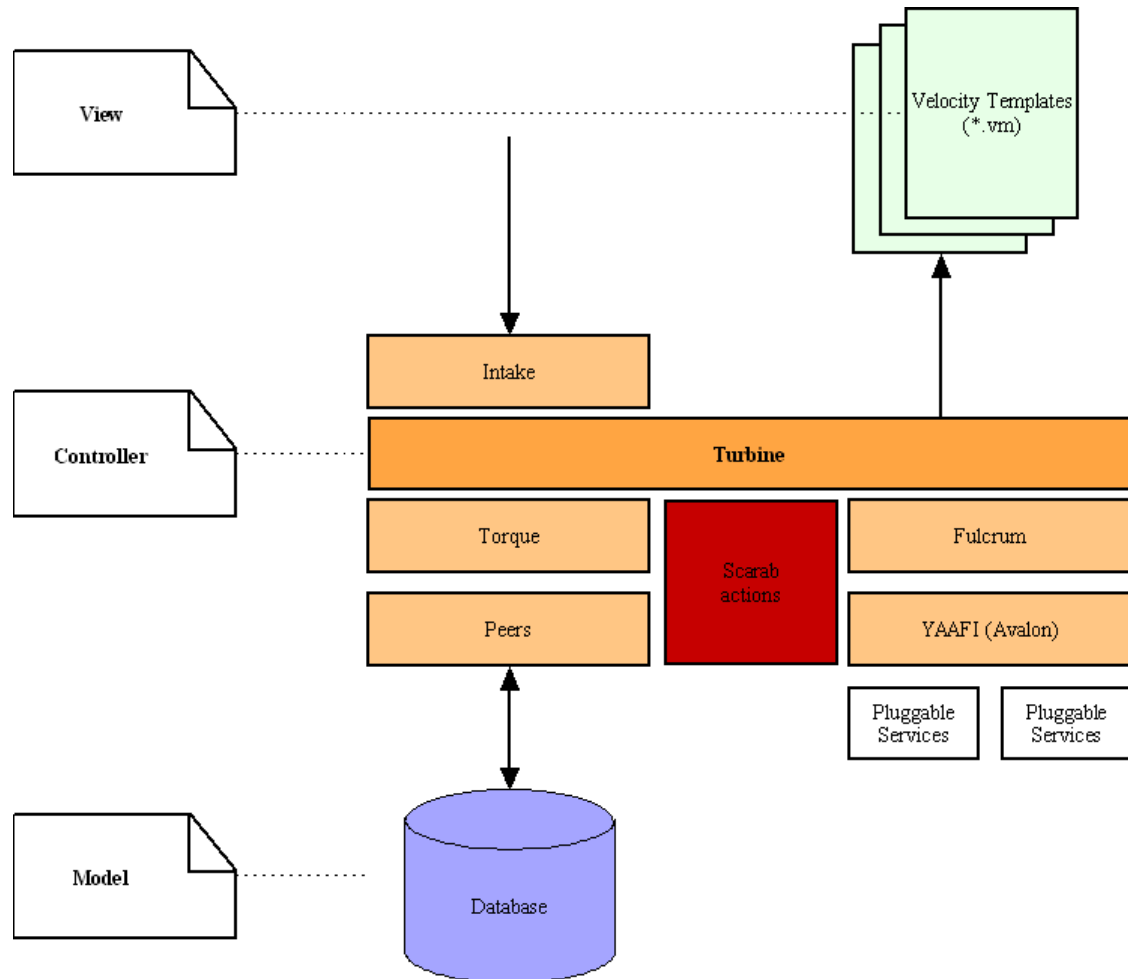
## List of Tables

|                             |    |
|-----------------------------|----|
| 6.1. Intervalles d'ID ..... | 32 |
|-----------------------------|----|

---

# Chapter 1. Architecture de Scarab

## Scarab à vol d'oiseau



## Une application web MVC construite sur le framework Turbine

### Une application web

Scarab est une application web utilisant la technologie Java servlet.

### Architecture MVC

Dans son architecture globale, Scarab utilise le *design pattern* MVC (Modèle-Vue-Contrôleur). La partie qui interagit avec l'utilisateur (Vue) est découplée des données qui caractérisent l'état de l'application (Modèle) par un Contrôleur qui gère de manière déclarative la navigation dans Scarab.

### Le framework Turbine

Un framework, c'est une certaine manière de faire des applications plus des outils pour les faire. Le framework Turbine (<http://jakarta.apache.org/turbine> [<http://jakarta.apache.org/turbine/>]) c'est donc une doctrine pour réaliser des applications web suivant le *design pattern* MVC et des outils - notamment un certain nombre de composants et de projets - pour les faire. Certains de ces composants, utilisés par Scarab, sont détaillés ci-dessous.

## Les principaux composants de Scarab

### Intake

Les données de formulaire soumises par l'utilisateur peuvent être d'abord soumises à un processus de validation (champs obligatoires, réponses bien formées, ...). Cette vérification est effectuée à l'aide d'un composant de Turbine appelé Intake.

Intake est aujourd'hui intégré à Fulcrum.

#### Note

Intake joue dans Turbine le même rôle que Struts Validator dans Struts, si vous êtes familier de ce framework.

### Dans la distribution de Scarab

Les règles de validation d'Intake sont décrites en XML dans le fichier *src/conf/intake.xml*

### Sur l'Internet

La documentation d'Intake est disponible sur l'Internet à cette URL : <http://jakarta.apache.org/turbine/fulcrum/fulcrum-intake> [<http://jakarta.apache.org/turbine/fulcrum/fulcrum-intake/>].

### Turbine (au sens strict)

C'est le coeur du framework. Comme dans tous les frameworks de type MVC2, il n'y a (pour l'essentiel) qu'un seul point d'entrée dans l'application, la servlet *scarab* qui est une instance de la classe *org.apache.turbine.Turbine*

Le comportement de cette servlet est paramétré par cinq fichiers de propriétés recensés dans le descripteur de déploiement *TurbineConfiguration.xml* :

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

#### Note

Quelle version de Turbine ?

Si vous connaissez Turbine ou si vous avez consulté la documentation sur le site de Jakarta, vous avez dû réaliser que l'architecture des différentes versions de Turbine, notamment 2.3 et 2.4 qui sont les deux versions d'usage courant, est assez différente.

En fait, l'équipe de développement de Scarab tente de faire converger la version utilisée de Turbine avec 2.4. Scarab a en effet été développé avec une pré-version 3 de Turbine -- qui ne verra probablement jamais le jour; nous tentons donc de ramener Scarab dans le flux normal du développement de Turbine en "baisant" de version.

## Dans la distribution de Scarab

Les fichiers de configuration mentionnés ci-dessus sont situés dans la distribution de Scarab sous *src/conf/conf/* et dans la web application sous *WEB-INF/conf* .

Turbine lui-même se trouve dans un jar sous *www/repository/turbine/jars* et les sources correspondantes sous *www/repository/turbine/src*. Comme expliqué dans la note plus haut, il s'agit d'une version intermédiaire à laquelle il n'est pas nécessairement possible de substituer une version identifiée.

## Sur l'Internet

La documentation du projet Turbine est disponible à cette URL : <http://jakarta.apache.org/turbine> [<http://jakarta.apache.org/turbine/>].

## Les services : Fulcrum

L'architecture de Turbine 2.4 est basée sur la notion de services et l'utilisation d'un microconteneur de la famille Avalon. Un certain nombre de ces services (notamment Intake mentionné plus haut) sont intégrés à Fulcrum et utilisés par Scarab.

## Dans la distribution de Scarab

Les jars de Fulcrum (une bonne dizaine d'archives) sont sous *www/repository/fulcrum/jars*

## Sur l'Internet

La documentation de Fulcrum est disponible à cette URL : <http://jakarta.apache.org/turbine/fulcrum/>

## Le microconteneur : YAAFI (Avalon)

Les services utilisés par Scarab sont implémentés au-dessus d'un microconteneur de la famille du (défunt) projet Avalon. YAAFI est d'ailleurs l'acronyme de *Yet Another Avalon Framework Implementation*.

## Dans la distribution de Scarab

YAAFI est fourni dans *www/repository/fulcrum/jars/fulcrum-yaafi-1.0.3.jar*

## Sur l'Internet

La documentation de YAAFI et de son interfaçage avec Fulcrum est disponible à cette URL : <http://jakarta.apache.org/turbine/fulcrum/fulcrum-yaafi> [<http://jakarta.apache.org/turbine/fulcrum/fulcrum-yaafi/>].

## Mapping objet/relationnel : Torque et les classes *Peer*

Les objets Java de l'application Scarab sont peuplés à partir d'une base de données relationnelle et persistés dans cette base en utilisant un composant logiciel autrefois partie intégrante de Turbine mais qui a depuis acquis son autonomie : Torque.

A partir d'une description de haut niveau du schéma de la base en XML, Torque va générer :

1. des scripts de création du schéma de la base, pour les différents SGBDR;
2. des classes Java qui mappent les différentes tables de la base de données sur des objets Java, avec un certain nombre de méthodes de recherches et d'accesses.

Si vous êtes intéressé(e) à en savoir davantage, vous pouvez vous référer au chapitre 3.

### Dans la distribution de Scarab

Les trois fichiers XML décrivant le schéma de la base sont dans le répertoire `src/schema` de la distribution Scarab.

### Sur l'Internet

La documentation du projet Torque est disponible à cette URL : <http://db.apache.org/torque> [<http://db.apache.org/torque/>]

## La vue : Velocity

Turbine peut fort bien s'accomoder des JSP (JavaServer Pages) mais le choix des premiers développeurs de Scarab était d'utiliser plutôt la technologie Velocity pour les pages dynamiques présentées à l'utilisateur.

Velocity est un langage de gabarits (*templates*); à l'exécution, les valeurs de JavaBeans placés dans le "contexte" du gabarit peuvent être insérées dans un contenu (statique) prédéfini.

Deux différences avec la technologie JSP (qui ont sans doute motivé le choix des initiateurs du projet Scarab) :

1. s'il y a bien dans Velocity des structures de contrôle tels que conditions, boucles ou parcours de collection, contrairement aux JSP, il n'est pas possible d'insérer du code Java dans un gabarit; il n'y a donc pas d'équivalent des *scriptlets*, ce qui peut rendre la maintenance plus facile à long terme car la vue est vraiment découplée du contrôleur et du modèle;
2. les JSP ne peuvent en pratique être utilisées que comme gabarits de pages web (c'est-à-dire pour la génération de HTML ou de XML); Velocity est un langage plus général et on peut l'utiliser pour générer des mails, des scripts SQL, du PostScript, etc.

### Dans la distribution de Scarab

Les gabarits Velocity des pages de Scarab (et des courriels, et de bien d'autres choses) sont sous le répertoire `src/webapp/WEB-INF/templates`.

### Sur l'Internet

La documentation du projet Velocity est disponible à cette URL : <http://jakarta.apache.org/velocity> [<http://jakarta.apache.org/velocity/>].

## **Note**

Notez que le manuel de l'utilisateur de Velocity est disponible en français : [http://jakarta.apache.org/velocity/translations/user-guide\\_fr.html](http://jakarta.apache.org/velocity/translations/user-guide_fr.html)

---

# Chapter 2. Comprendre le build avec Ant

L'application Scarab peut être construite par l'utilisateur en utilisant Apache Ant. C'est aussi avec Ant qu'est créée et initialisée la base de données.

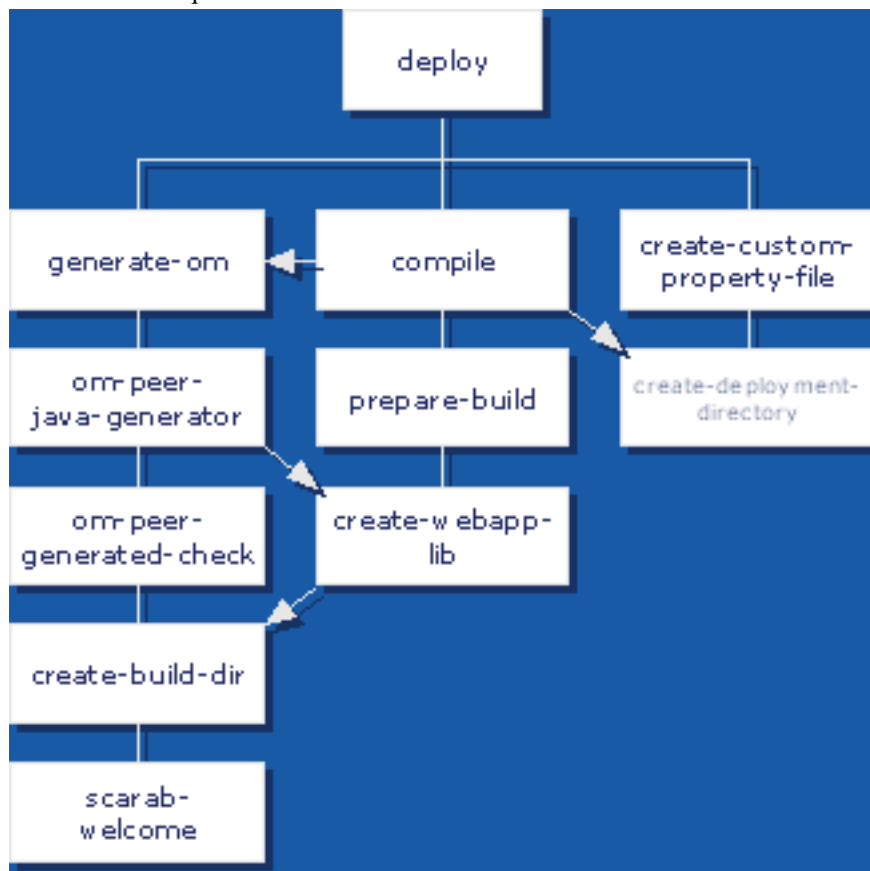
## `$SCARAB_ROOT/build/build.xml`

C'est le fichier utilisé à titre principal pour le build.

La cible par défaut, `deploy`, génère le code des classes de mapping objet-relationnel (OM) puis compile et assemble l'application Scarab.

La seconde cible intéressante, `create-db`, est utilisée pour créer et initialiser la base de données de Scarab. Elle réalise ces opérations en utilisant le second fichier de build Ant, décrit dans la section suivante.

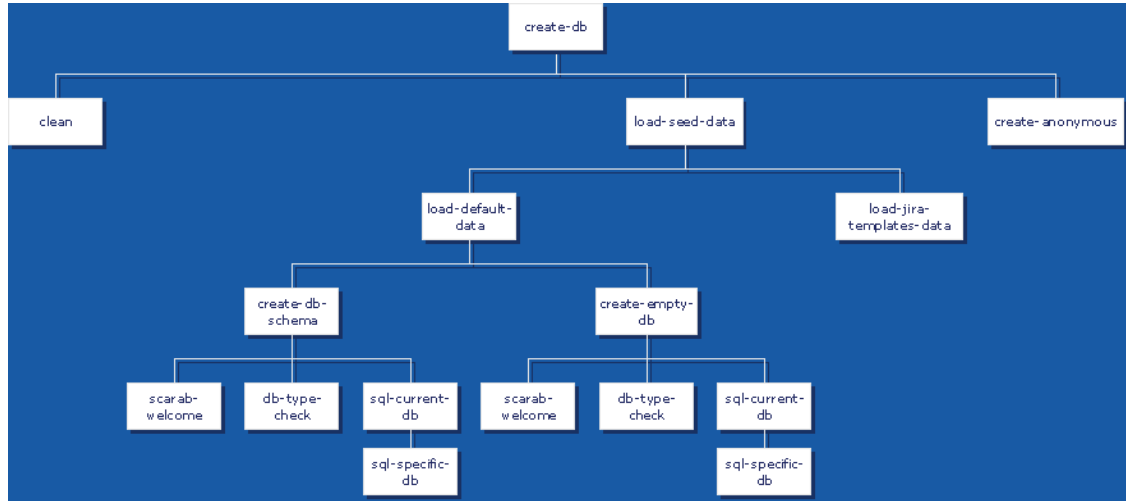
Le graphe de dépendance des cibles Ant devrait vous aider à visualiser la relation des cibles entre elles et l'ordre dans lequel elles sont exécutées.



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

Ce fichier de build est utilisé pour créer et peupler la base de données de Scarab.

Le graphe de dépendance des cibles Ant devrait vous aider à visualiser la relation des cibles entre elles et l'ordre dans lequel elles sont exécutées.



---

# Chapter 3. Maven

La plupart des utilisateurs de Scarab se servent probablement de ant pour construire Scarab. En tant que développeur, vous aurez besoin de Maven pour avoir accès à toutes les fonctionnalités de développement, notamment la génération de la documentation.

## Maven 1

Bien que Maven 2 prenne de la vitesse et soit devenu la version officielle depuis fin 2005, Scarab utilise toujours Maven 1.

Maven 1 et Maven 2 sont à peu près équivalents d'un point de vue fonctionnel mais ils ne sont pas syntaxiquement compatibles et utilisent des fichiers de build différents.

Il vous faudra donc installer Maven 1 et vous référer à la documentation de Maven 1 à cette URL: <http://maven.apache.org/maven-1.x/>

## Construire Scarab avec Maven

### Build normal

Allez dans le répertoire `$SCARAB_ROOT`

Lancez Maven de la manière suivante :

```
maven war
```

### Build abrégé

Allez dans le répertoire `$SCARAB_ROOT`

Lancez Maven de la manière suivante :

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

`-Dmaven.test.skip` évite d'exécuter les tests unitaires. Comme les tests sont conçus pour être exécutés avec une base de données, il est possible qu'ils échouent si la vôtre n'est pas encore configurée.

### Exécuter les tests

Si vous lancez les tests à présent, ils s'exécuteront sur la base de données que vous avez configurée. Tapez simplement

```
maven test
```

Le résultat des tests sera disponible dans `$(SCARAB_HOME)/target/test-results/`.

Si vous souhaitez exécuter les tests dans un autre environnement de base de données, procédez comme suit :

1. `maven clean` (il vaut mieux partir sur une base saine pour exécuter les tests)

2. Configurez votre fichier `build.properties` pour qu'il contienne au moins la valeur correcte de la propriété `scarab.database.type` value
3. `maven war`
4. `maven scarab:create-db` (assurez-vous de choisir la même base qu'auparavant!)
5. `maven test`

---

# Chapter 4. Développement Scarab avec Eclipse

## Obtenir les sources de Scarab

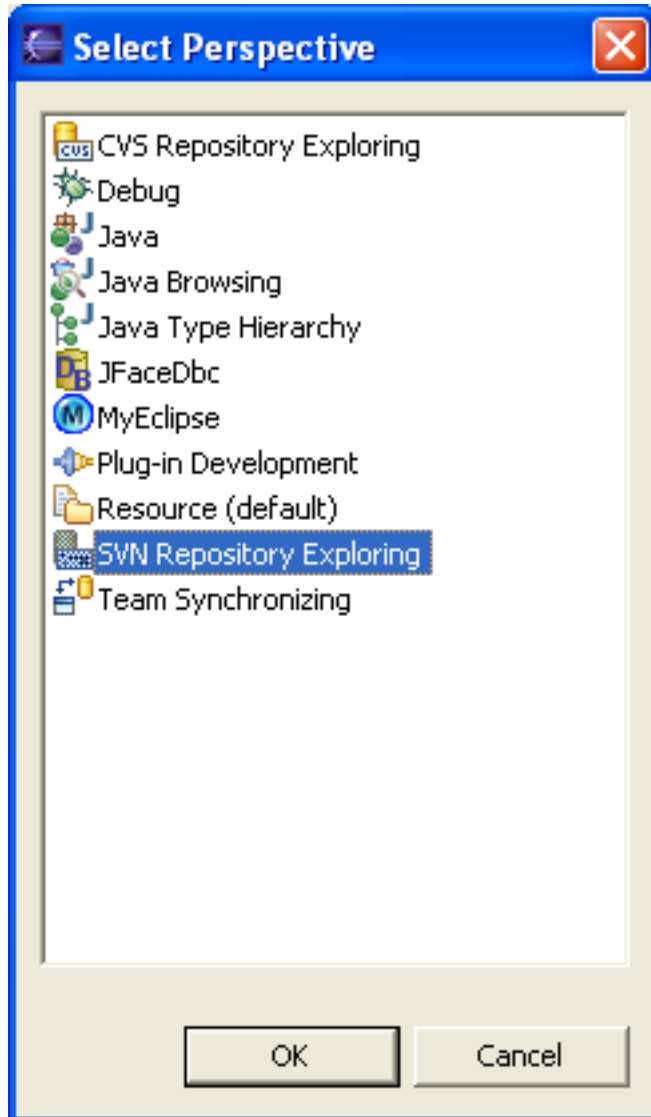
### Installer le plug-in Subclipse

Vous aurez besoin du plug-in subclipse pour vous connecter au repository Subversion qui contient et gère le code source de Scarab.

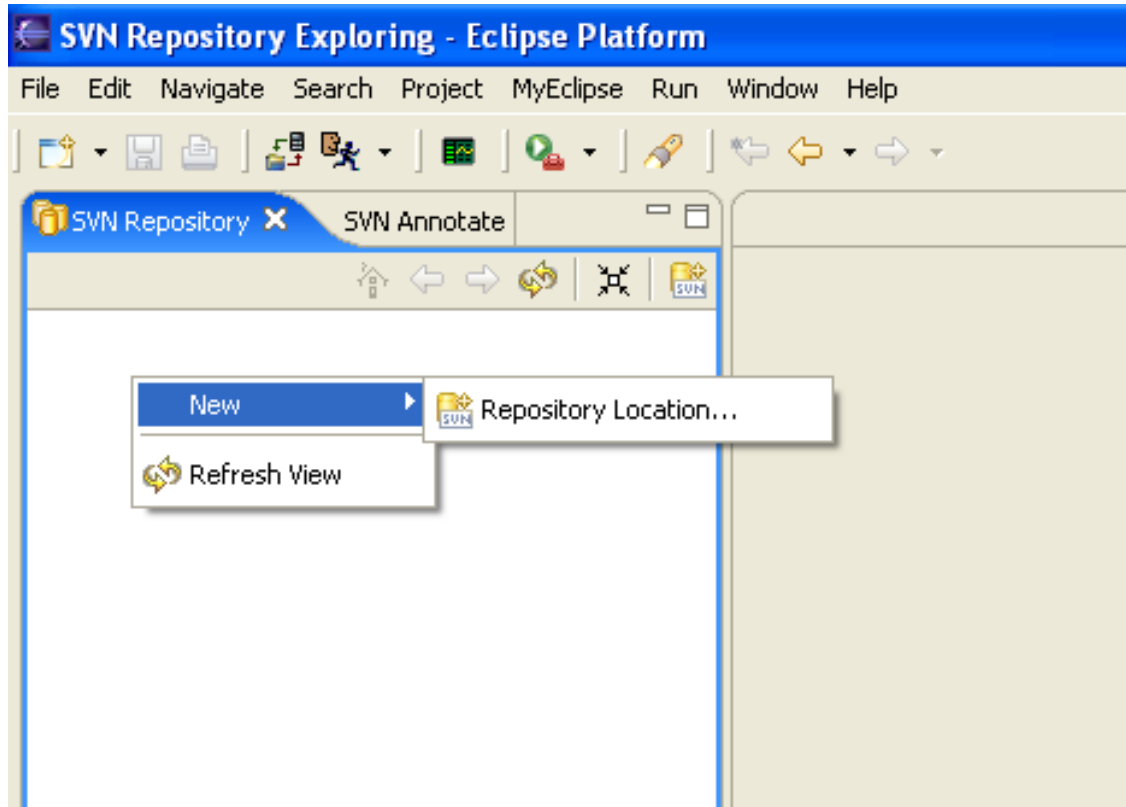
Si vous ne disposez pas de ce plug-in, vous pouvez consulter des instructions détaillées d'installation à cette URL: <http://subclipse.tigris.org/install.html>

### Se connecter au repository Subversion de Scarab

Sélectionnez la perspective SVN Repository Exploring :



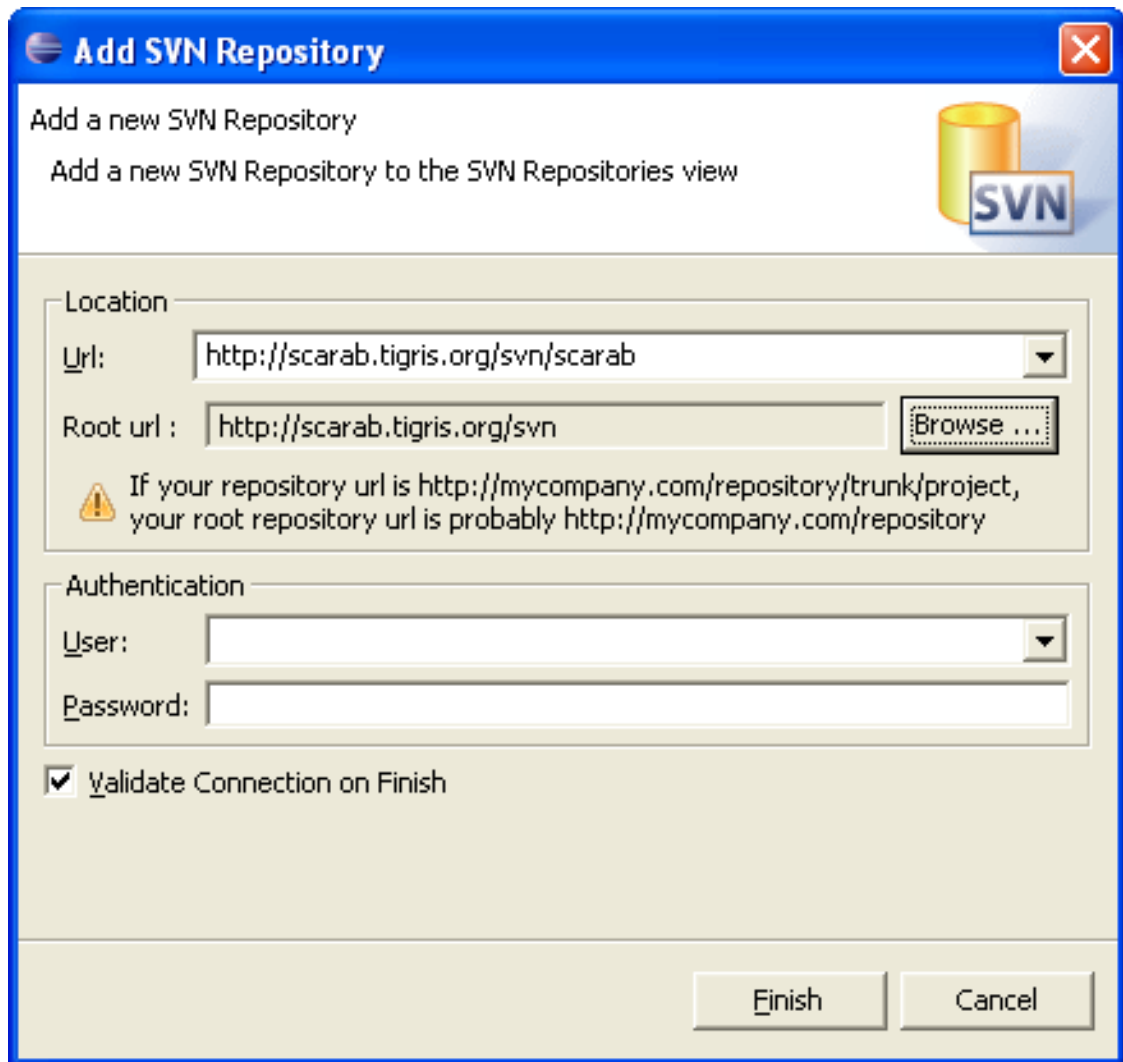
Un clic droit dans la vue SVN Repository vous permet de vous connecter à un nouveau repository Subversion :



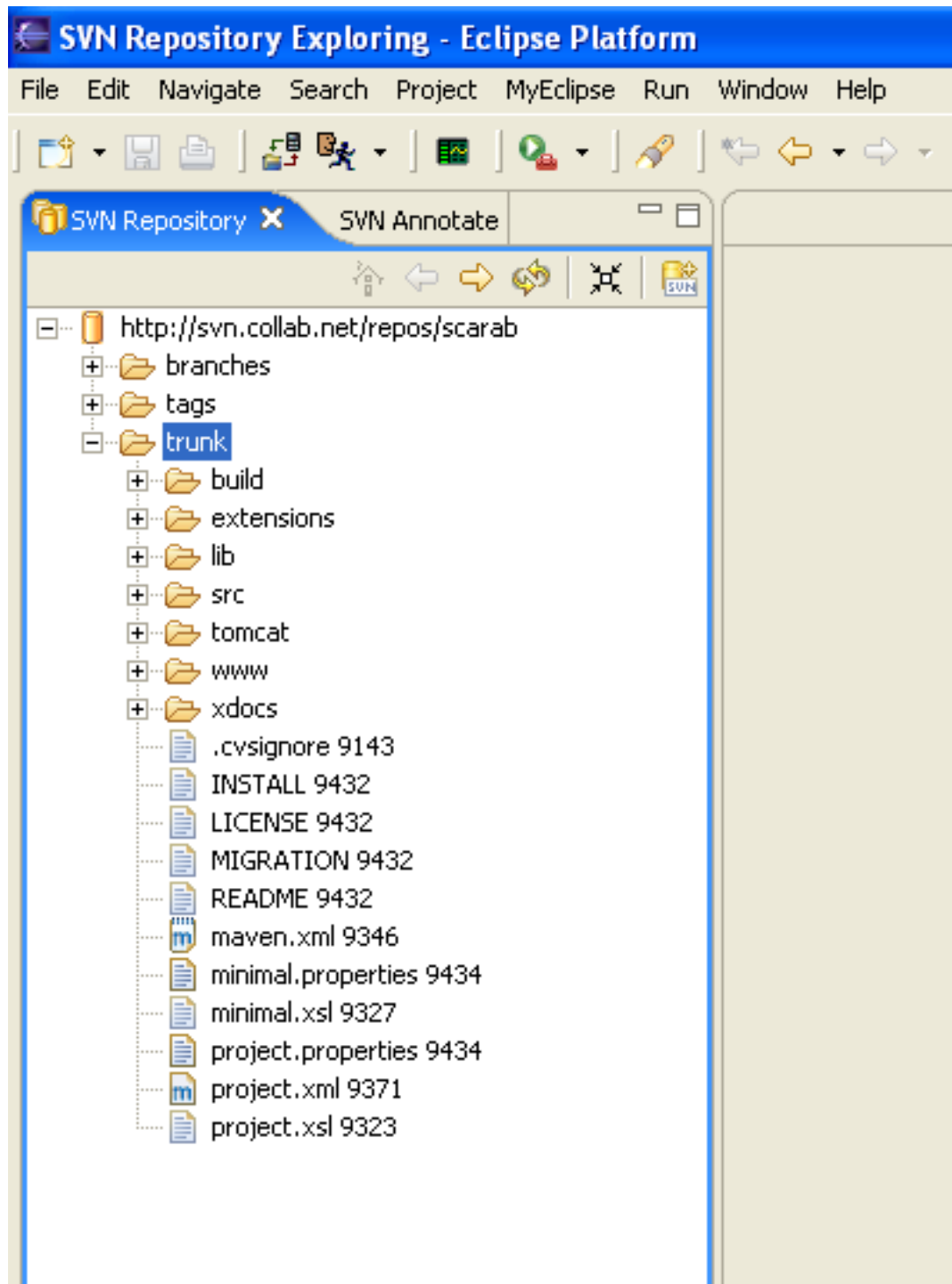
L'URL de connexion est: <http://scarab.tigris.org/svn/scarab>

et l'URL racine (Root URL): <http://scarab.tigris.org/svn>

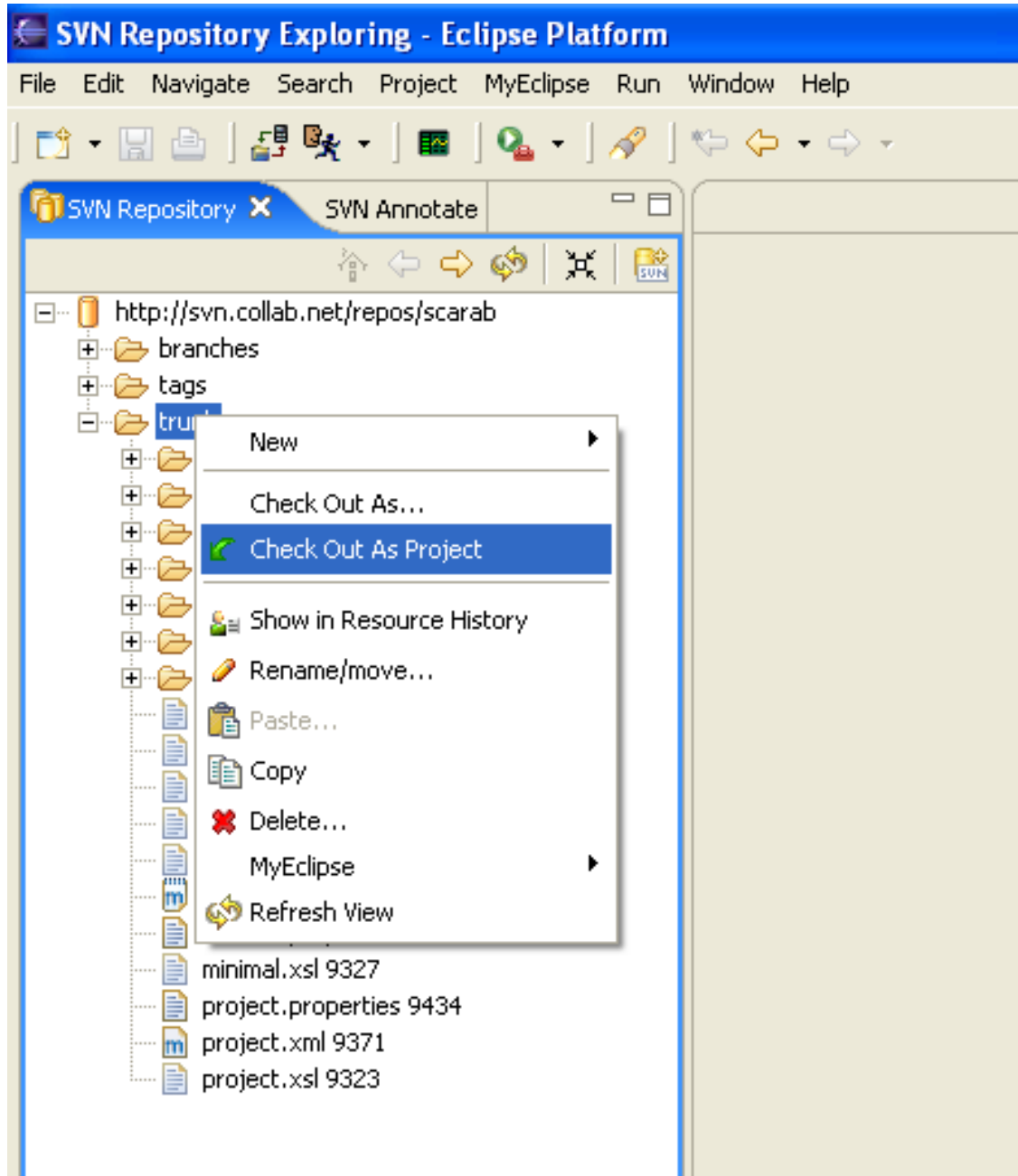
Si vous êtes un des *committers* de Scarab, utilisez vos identifiants sur tigris.org pour vous connecter; sinon, vous n'avez pas besoin d'authentification et il n'est donc pas nécessaire de remplir les champs *User* et *Password* (mais n'importe quelle paire *User/Password* fera aussi l'affaire).



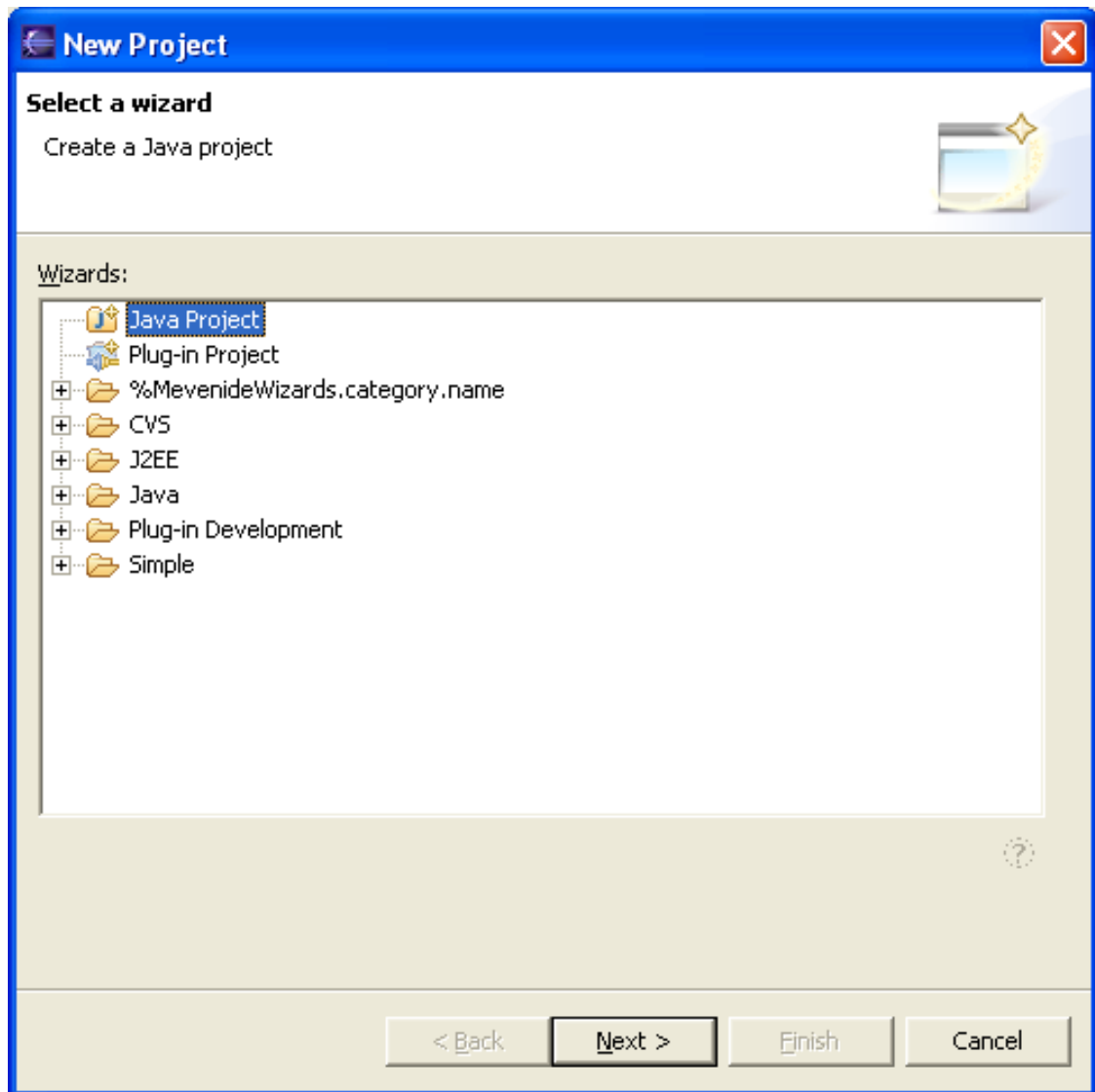
Pour extraire la version courante, choisissez la branche 'trunk' (l'équivalent de CVS HEAD pour ceux qui ont pratiqué ce système de contrôle de versions auparavant).



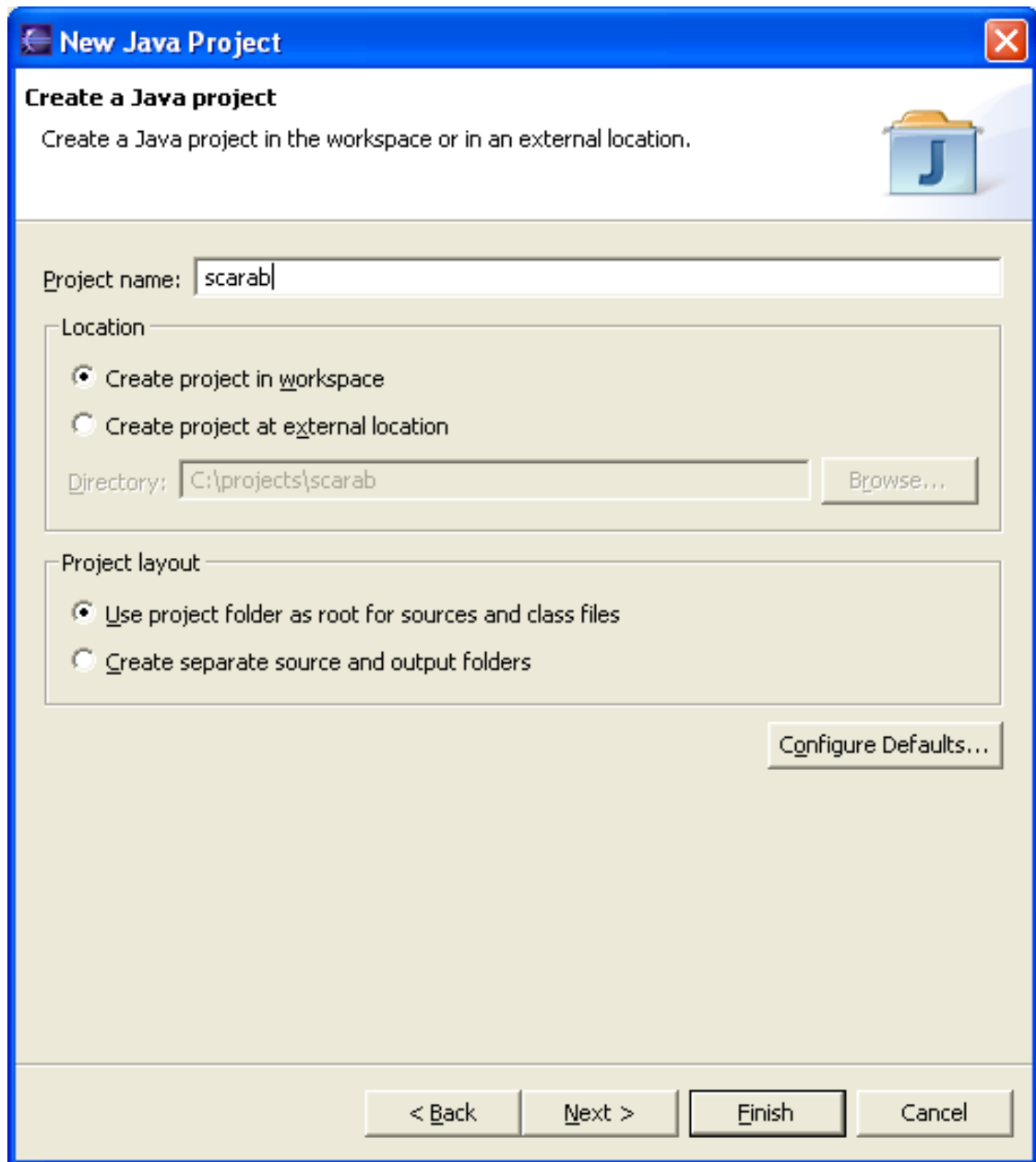
Il vous suffit ensuite de l'extraire comme projet (Checkout As Project), ce qui vous permet de le renommer au passage car sinon il s'appellerait *trunk*, ce qui n'est pas très expressif.



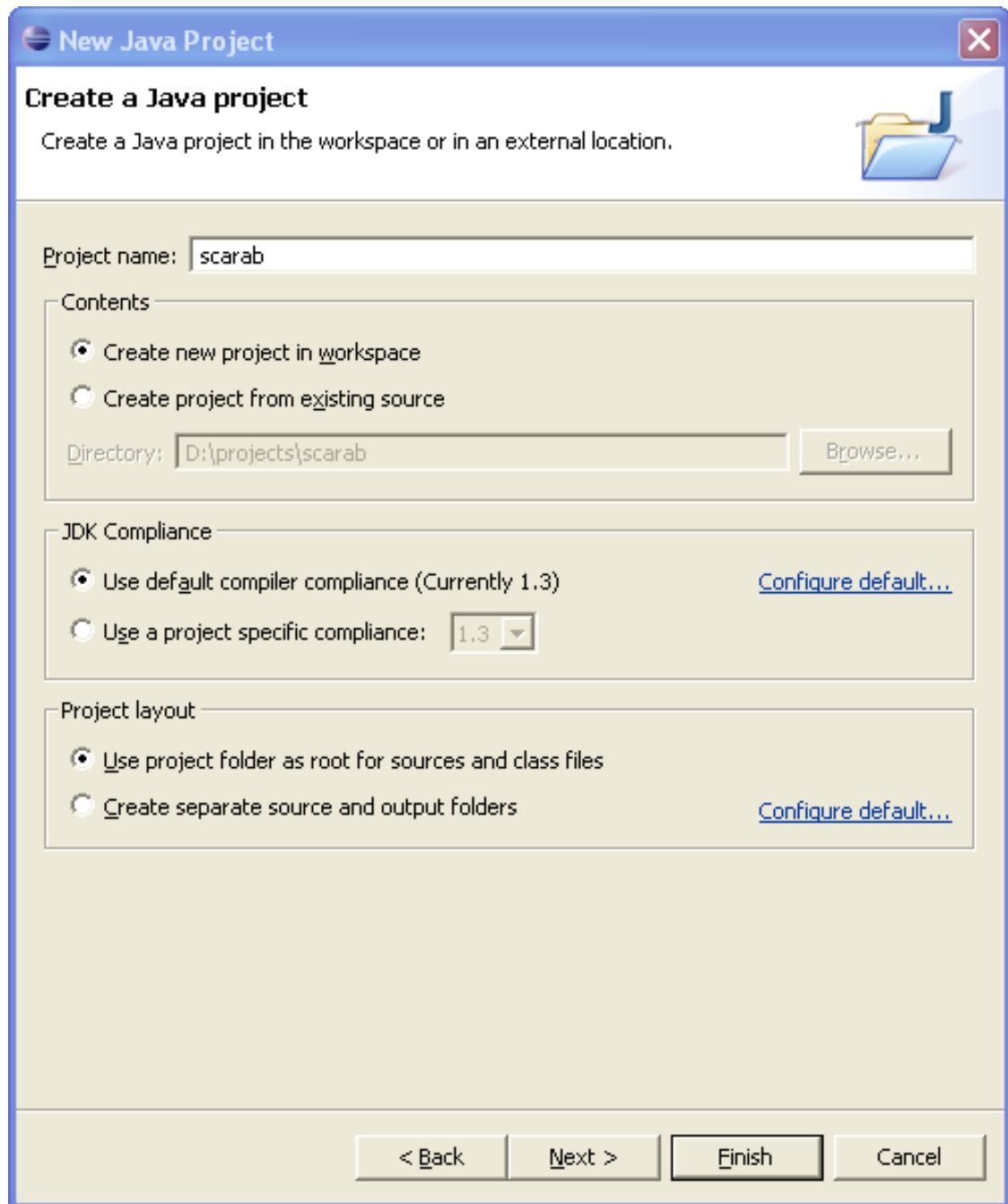
L'assistant Nouveau Projet (New Project) apparaît. Déclarez qu'il s'agit d'un projet Java.



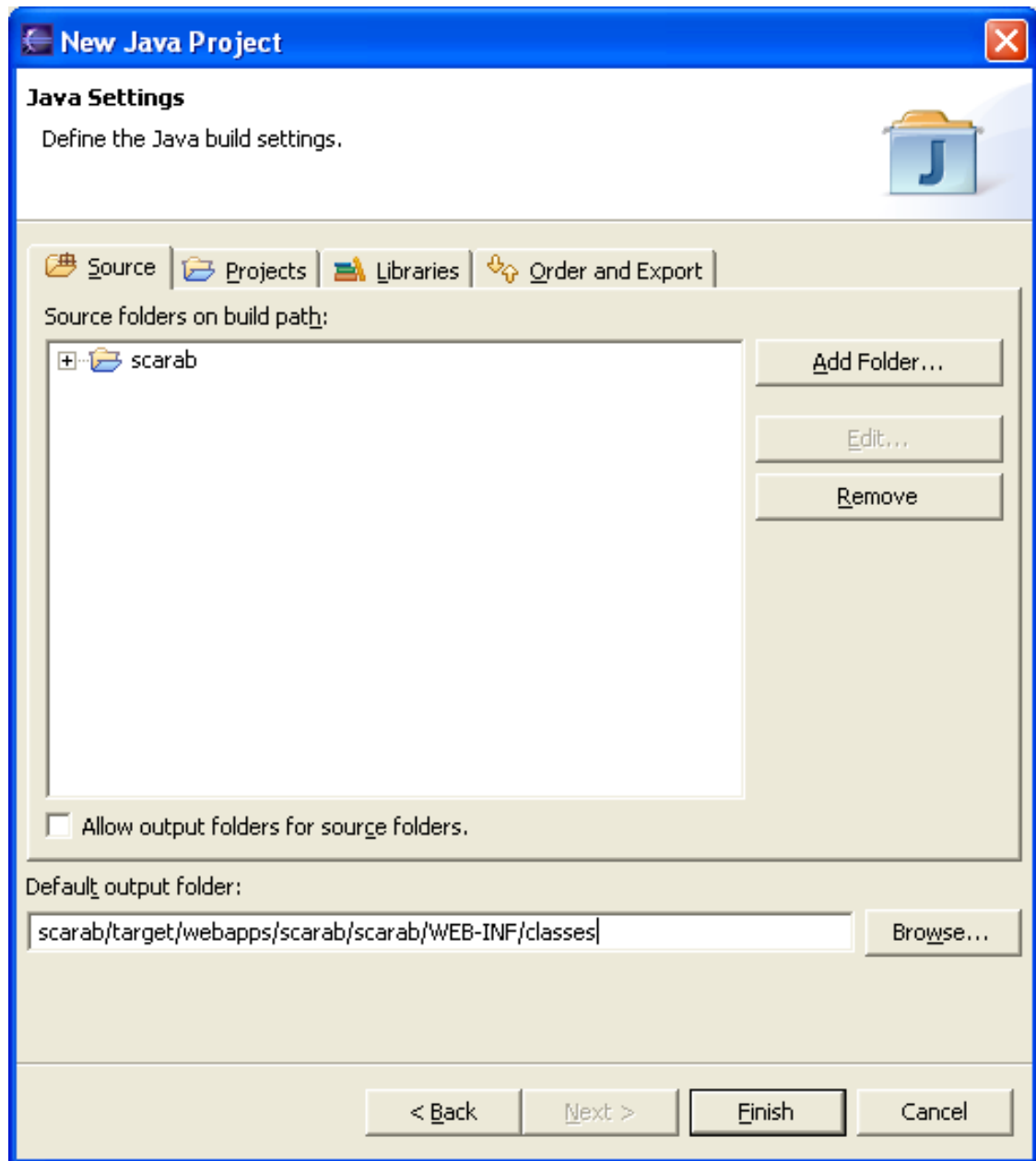
Le nom du projet vous est demandé (vous pouvez choisir *scarab* ou un autre nom à votre convenance).



Si vous utilisez Eclipse 3.1, il vous est demandé en plus de préciser avec quel J2SDK scarab doit être compatible. Pour l'instant, l'équipe de développement s'est astreinte à conserver la compatibilité avec le JDK 1.3.



A la dernière étape de l'assistant, il est préférable de modifier le chemin de sortie (*Default output folder*) pour lui donner la valeur suivante: scarab/target/webapps/scarab/scarab/WEB-INF/classes



## Développement de Scarab dans Eclipse

### Installer le plug-in Tomcat de Sysdeo

Le plug-in Tomcat permet de lancer et arrêter Tomcat à partir d'Eclipse et facilite le débogage de l'application en cours de développement.

Téléchargez la version 3.1 beta du plugin Tomcat de Sysdeo à cette URL: <http://www.sysdeo.com/eclipse/tomcatplugin>.

Pour installer ce plug-in, il suffit de décompresser l'archive ZIP dans le répertoire plugins de votre installation Eclipse.

## Générer les fichiers nécessaires avec Maven

A la racine de votre projet Scarab, lancez les commandes:

```
maven war:inplace
```

```
maven eclipse
```

Cette dernière commande génère ou régénère un fichier .project nécessaire à Eclipse

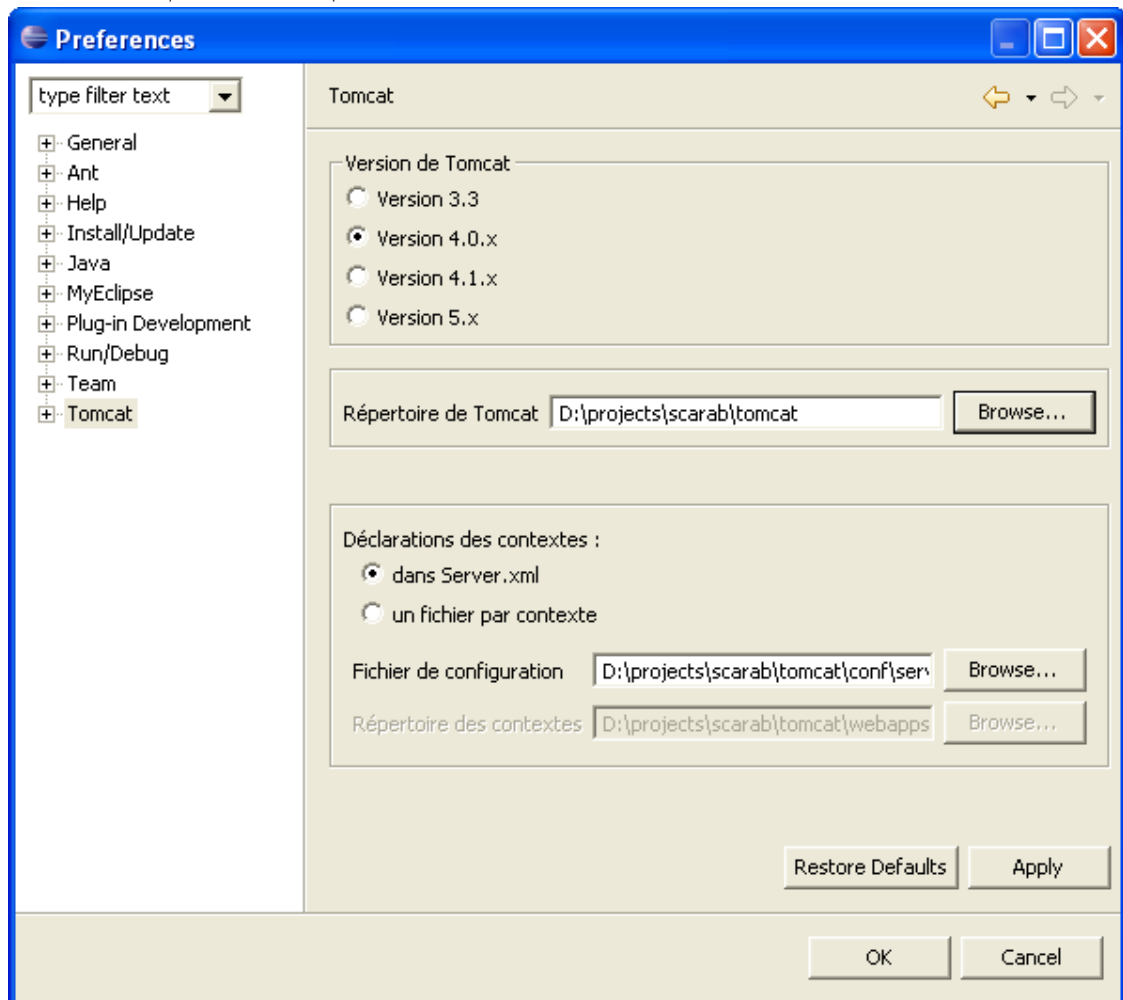
## Configurer et utiliser le plug-in Tomcat

### Reconstruire le projet

Relancez le build du projet Scarab dans Eclipse.

### Configurer les préférences de Tomcat

Dans Window | Preferences... | Tomcat :

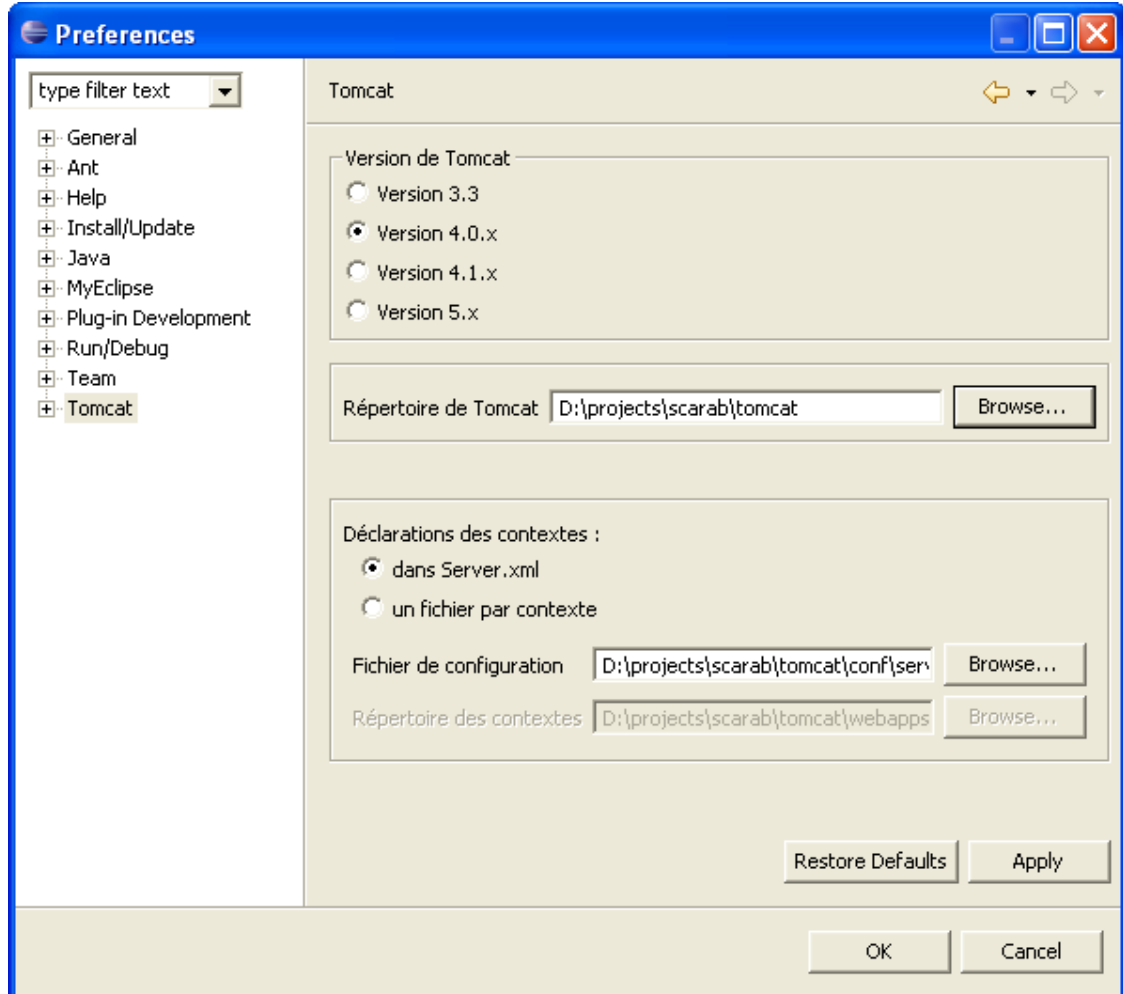


- Sélectionnez la version 4.0.x

- Répertoire de Tomcat: le répertoire tomcat dans votre projet Scarab
- Déclaration des contextes: dans server.xml

## Configurer les propriétés Tomcat du projet

Dans Project | Properties | Tomcat :



- Cochez l'option 'Est un projet Tomcat'
- Nom du contexte: scarab
- Décochez l'option 'Autoriser la mise à jour de la définition du contexte'
- Cochez l'option 'Autoriser le rechargement automatique de ce contexte' (cette option est normalement cochée par défaut)
- Cochez l'option 'Rediriger les logs de ce contexte vers la console Eclipse'

---

# Chapter 5. Développement Scarab avec Netbeans

## Récupérer les sources de Scarab

L'implémentation actuelle d'un client Subversion pour Netbeans est à l'état de prototype (son nom de code est "teepee", on peut suivre son développement à cette adresse ;<http://subversion.netbeans.org/>). Pour le moment, il vous faudra donc utiliser un autre client Subversion pour récupérer les sources depuis le repository de Scarab. Si vous développez sur Windows, un client agréable et facile à utiliser est TortoiseSVN [<http://tortoisesvn.tigris.org/>] - qui est hébergé par Tigris.org, comme Scarab.

## Récupérer le plug-in Mevenide (intégrationMaven-Netbeans)

Le meilleur moyen de développer Scarab dans Netbeans est très certainement d'utiliser le plug-in Mevenide (qui est d'ailleurs beaucoup plus stable et fonctionnel dans Netbeans qu'il ne l'est dans Eclipse, sans parler de JBuilder).

Téléchargez Mevenide for Netbeans depuis le site de Codehaus à cette URL : <http://mevenide.codehaus.org/mevenide-netbeans-project/index.html>

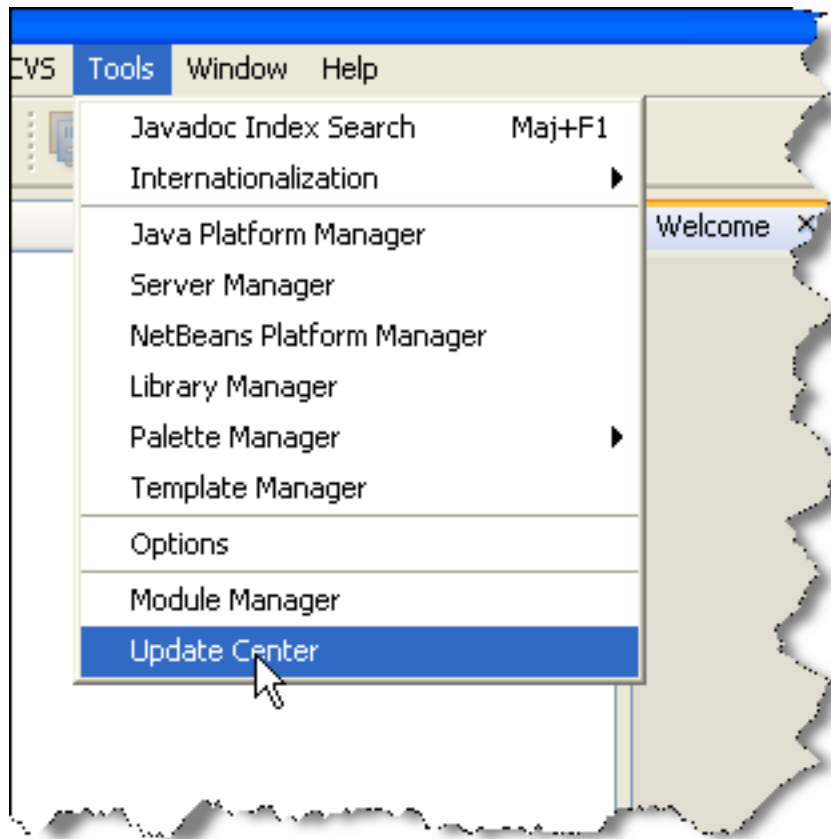
Vous obtiendrez ainsi un fichier ZIP contenant 16 nbm (Netbeans modules). Décompressez ce fichier dans un répertoire temporaire.

## Installer le plug-in Mevenide

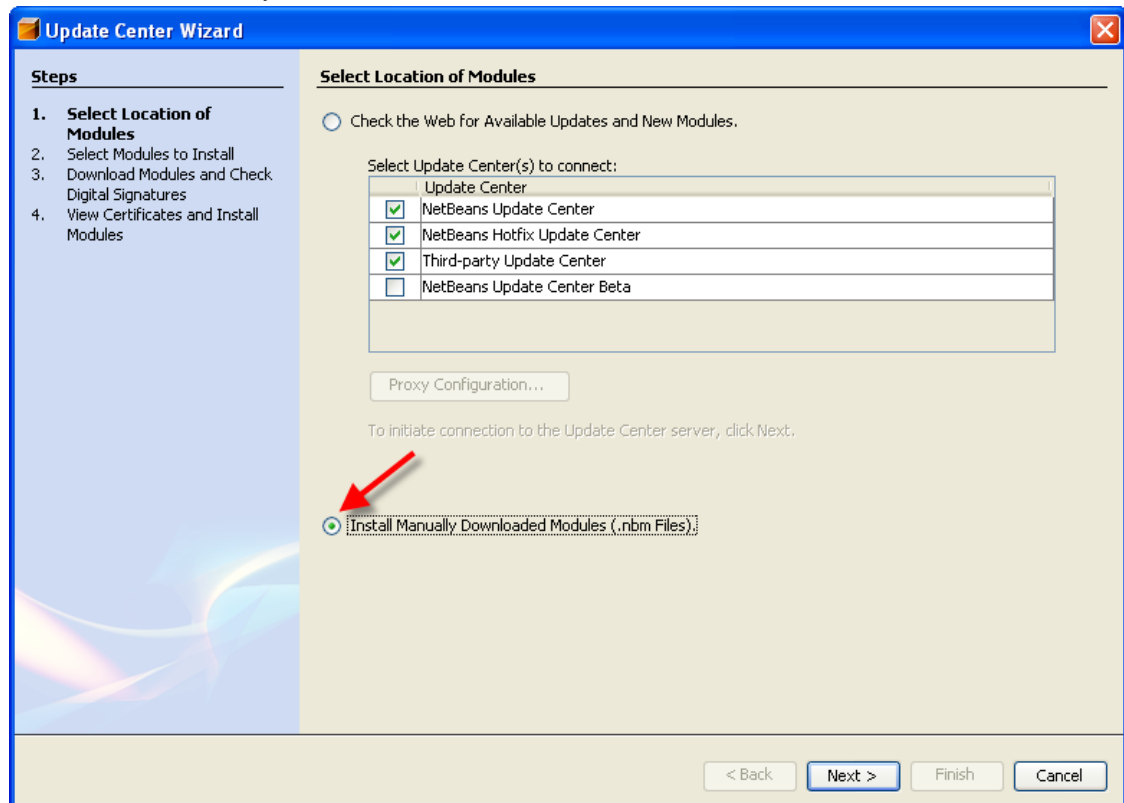
### Tip

Si l'installation de modules dans Netbeans vous est familière, vous pouvez hardiment sauter cette section.

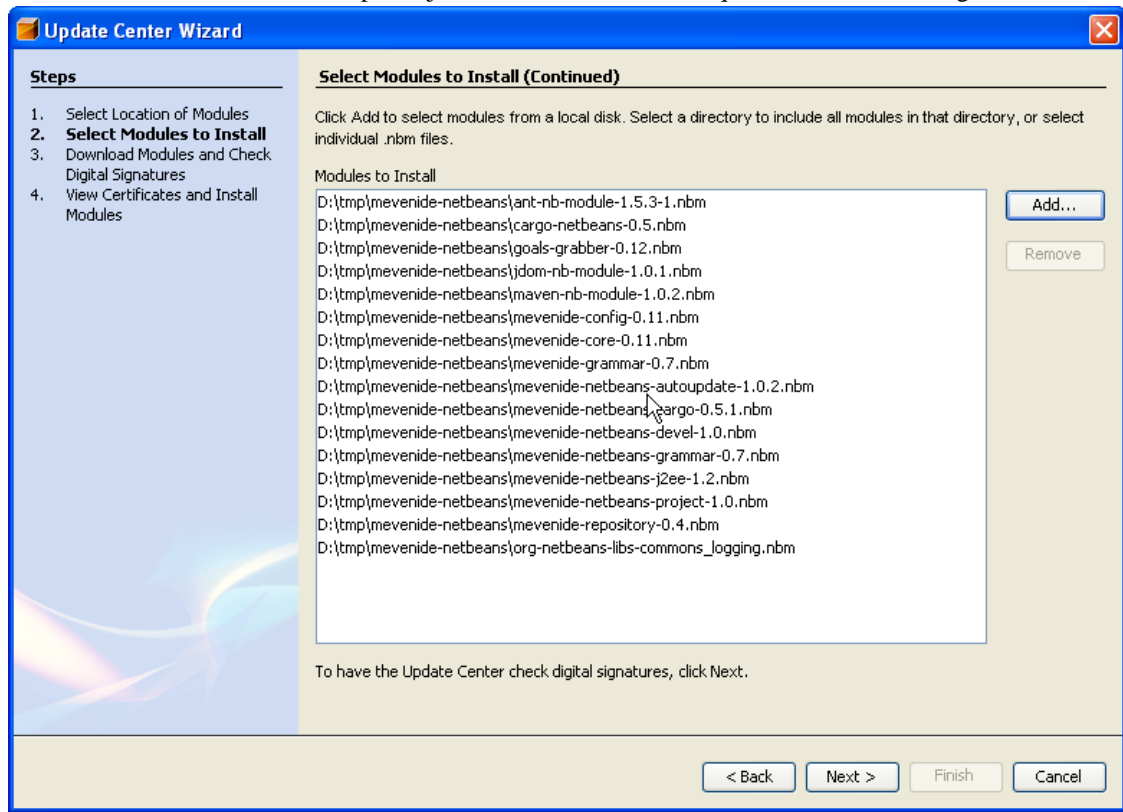
Dans le menu Tools, cliquez Update Center



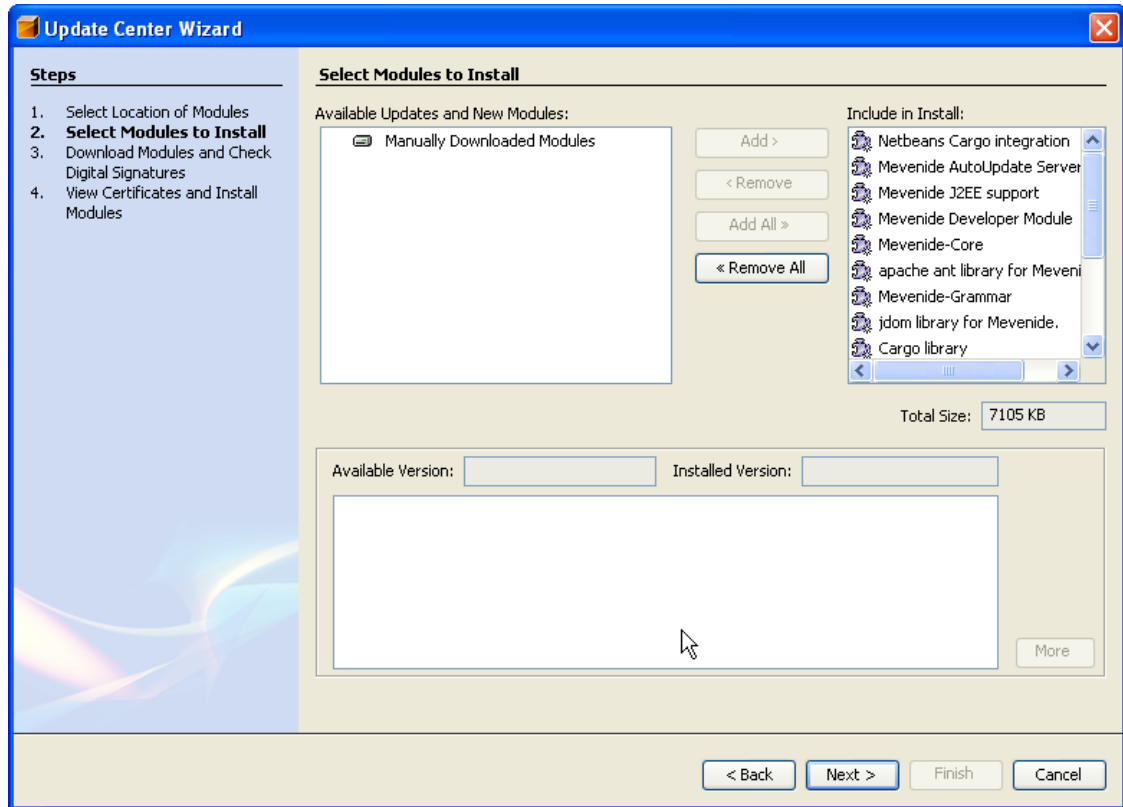
Cochez 'Install Manually Downloaded Modules (.nbm Files)'



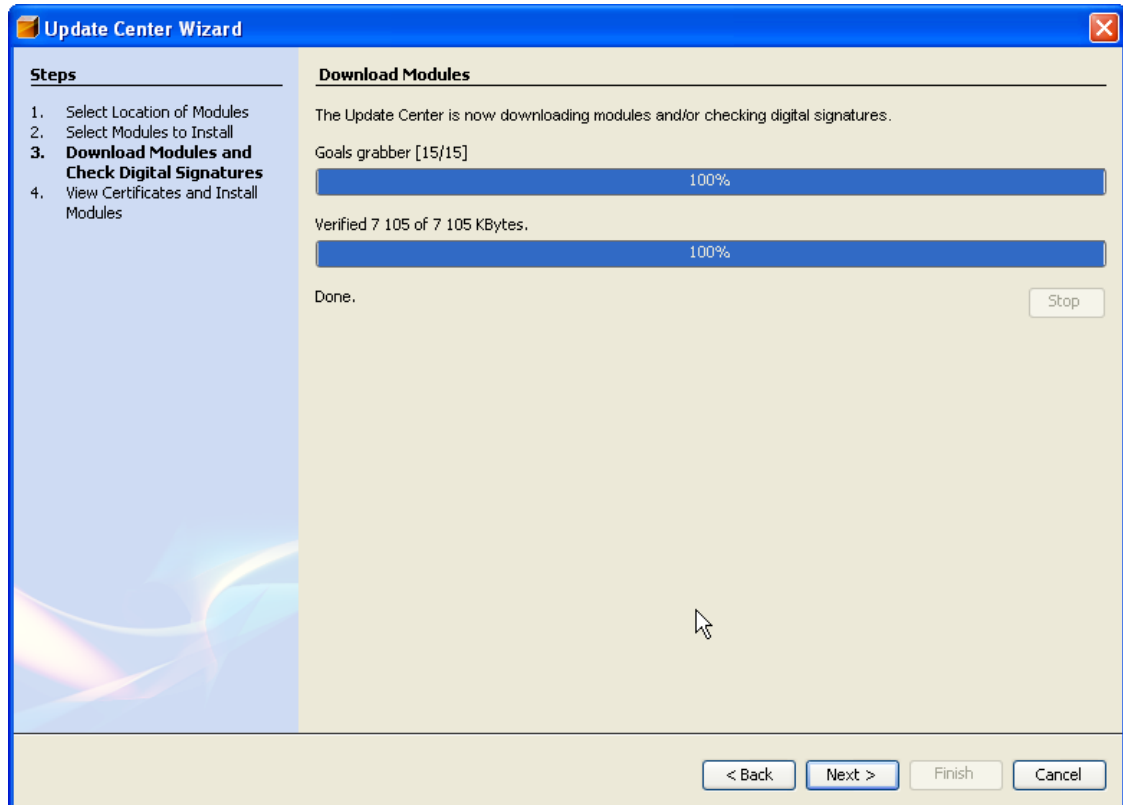
Utilisez le bouton 'Add...' button pour ajouter les 16 fichiers .nbm que vous avez téléchargés.



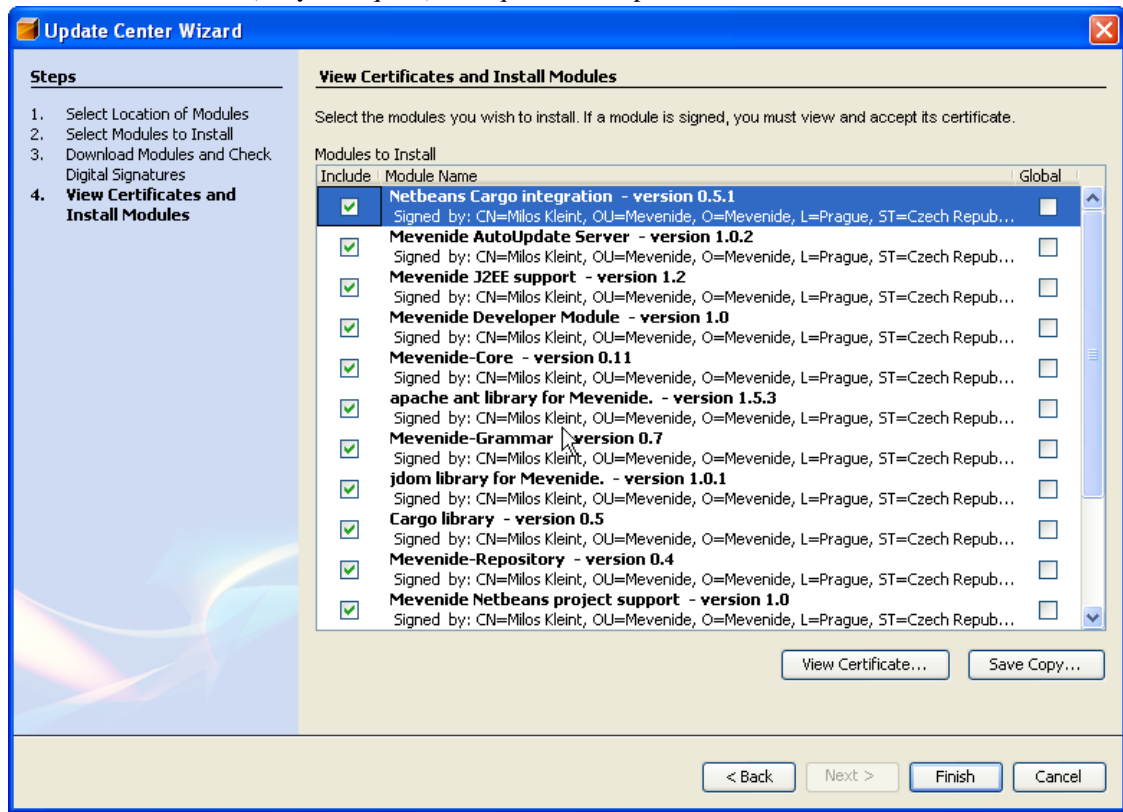
Cliquez 'Next >', ça suffira.



Il vous faut à présent accepter les différentes licences. Les modules sont alors prêts pour l'installation.



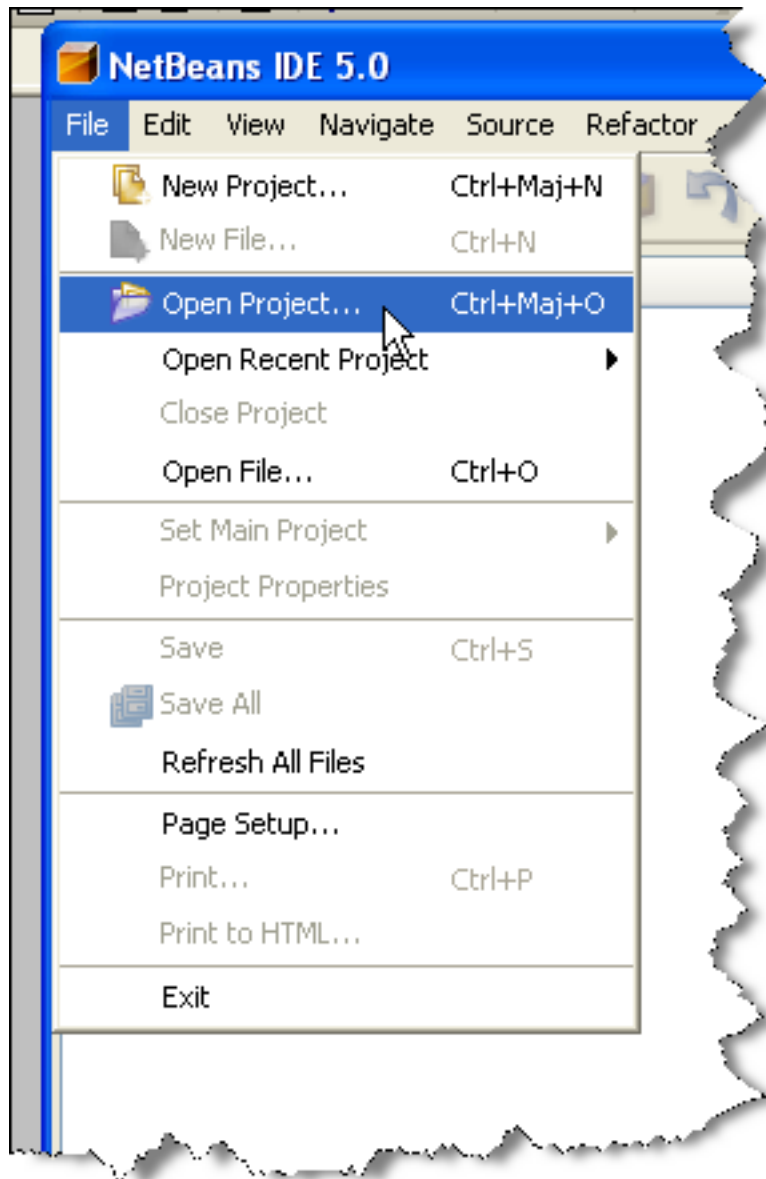
Consultez le certificat (il n'y en a qu'un) et cliquez 'Finish' pour finir d'installer les modules.



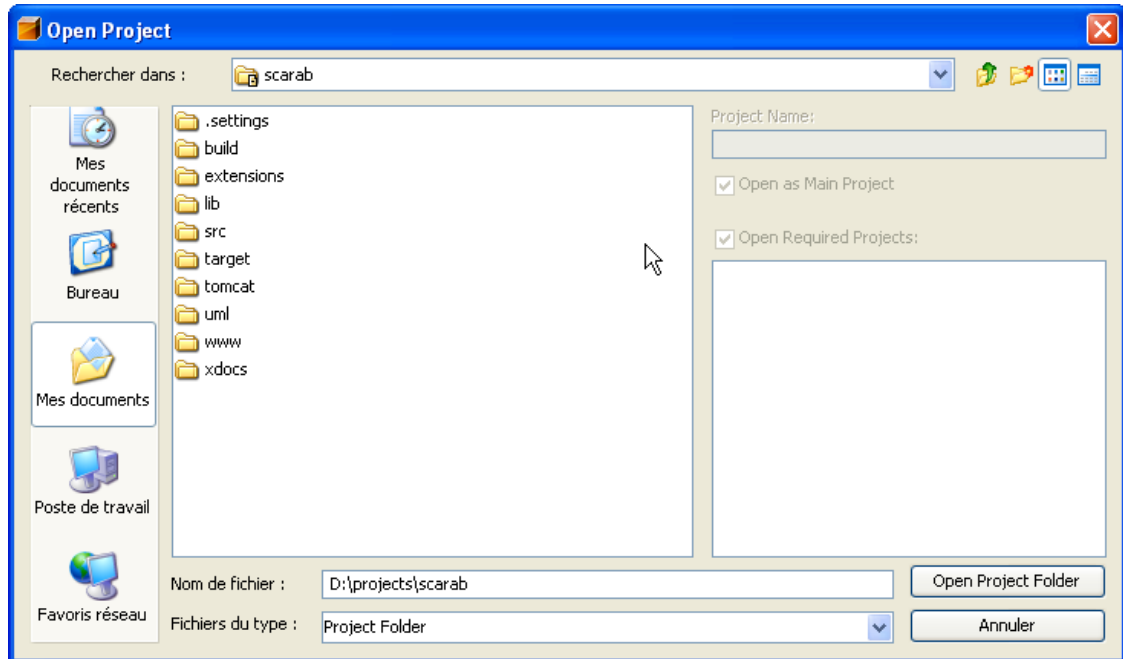
Vous devrez redémarrer Netbeans pour l'utiliser avec le plug-in Mevenide que vous venez d'installer.

## Développement de Scarab dans Netbeans

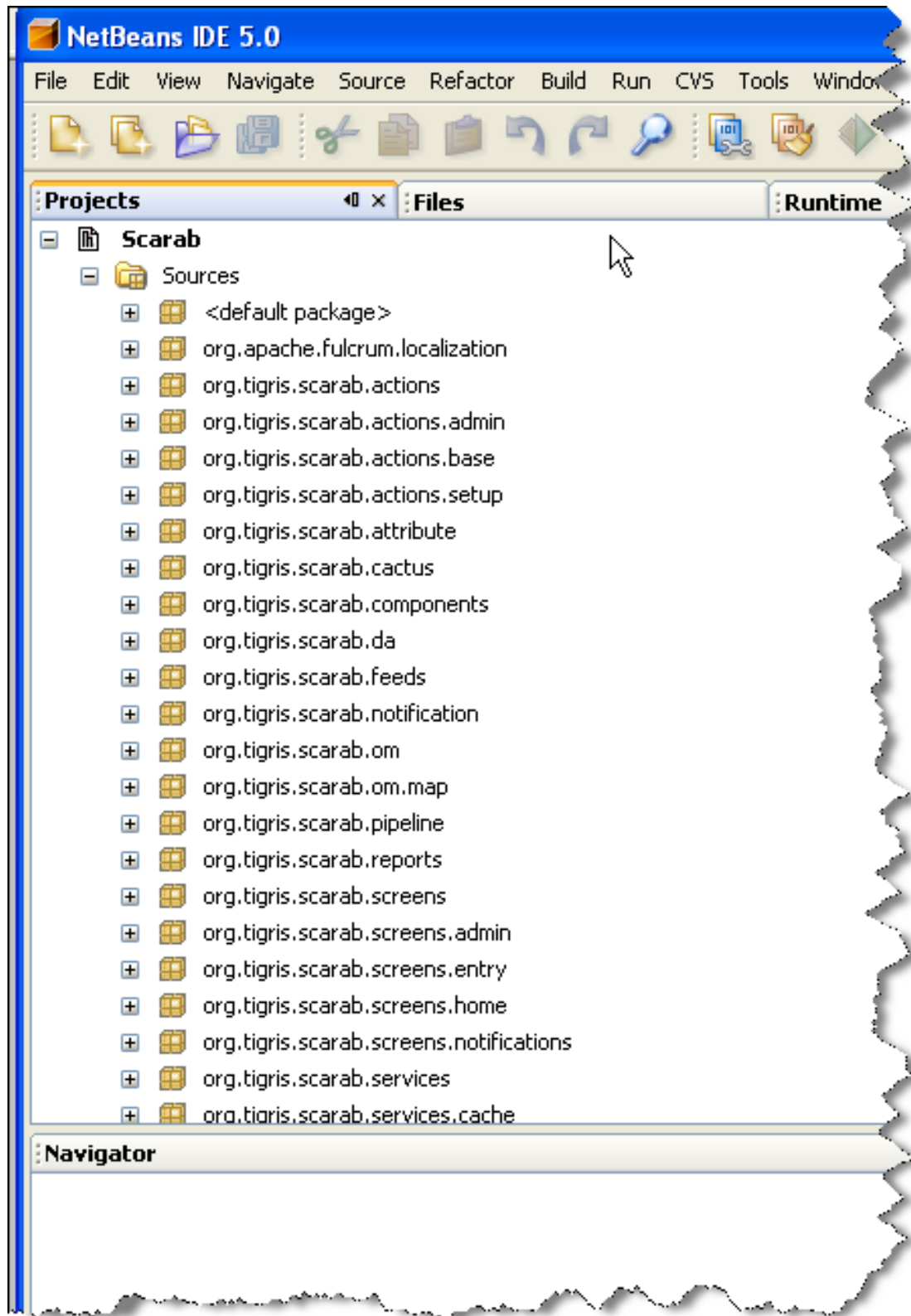
Accéder au projet Scarab est maintenant aussi simple que possible. Dans le menu 'File', sélectionnez 'Open Project...'



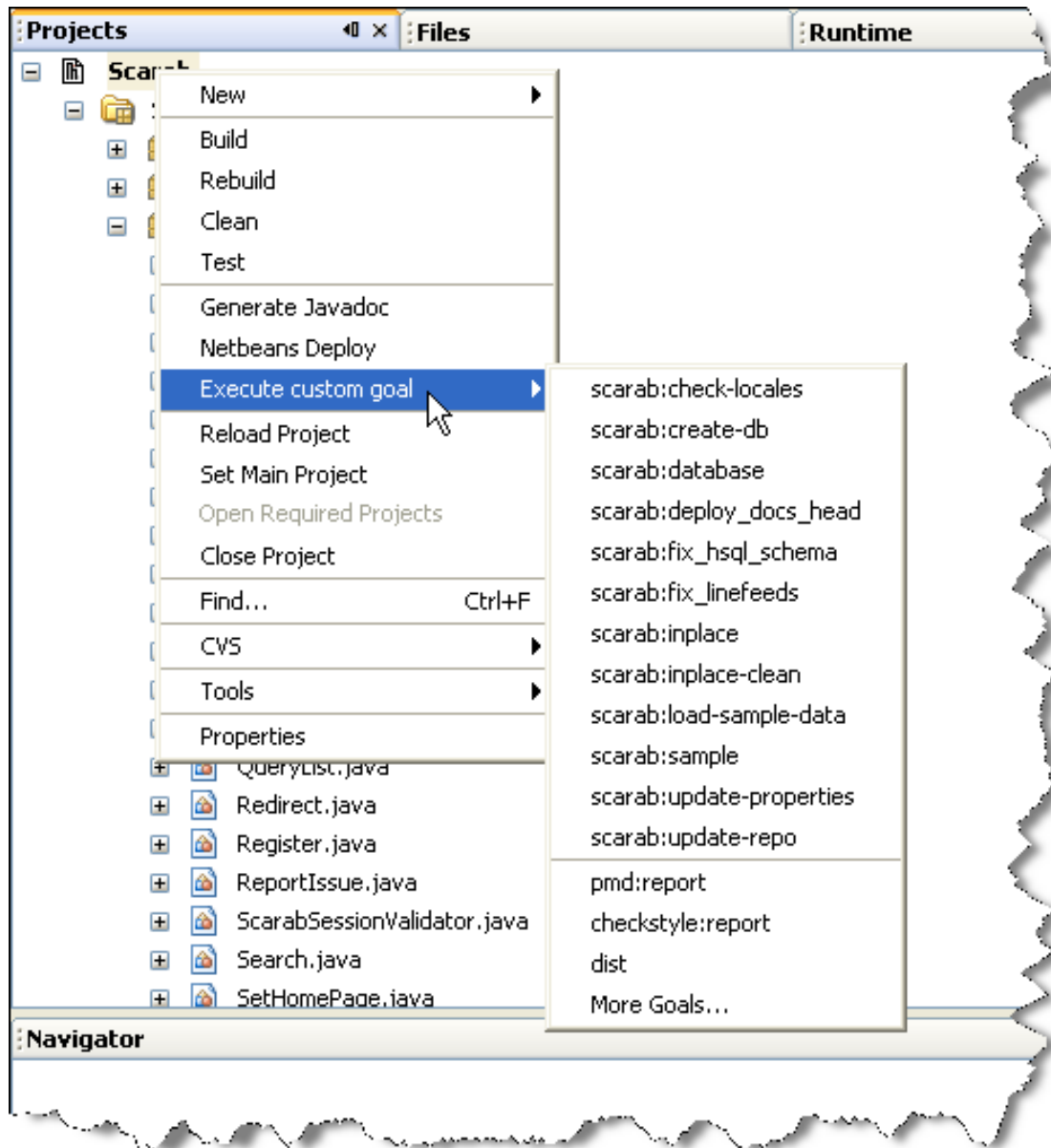
Choisissez le répertoire dans lequel vous avez récupéré Scarab depuis le repository Subversion.



Et voilà!



Pour accéder aux 'buts' (goals) Maven, il suffit de faire un clic droit sur le projet Scarab.



---

# Chapter 6. Le modèle de données de Scarab

## Introduction

L'accès aux données de Scarab se fait par le biais de Torque, le mécanisme de mapping objet-relationnel développé pour et avec le framework Turbine -- mais qui a acquis aujourd'hui son indépendance.

Le schéma de la base de Scarab est décrit de manière formelle en XML. Ce schéma est explicité en trois parties et autant de fichiers, sous `./src/schema`.

On y distingue :

- la table associée au mécanisme de fabrication de clés primaires (`id-table-schema.xml`);
- les tables de gestion des utilisateurs (notamment authentification) et des droits du framework Turbine (`turbine-schema.xml`);
- les tables de l'application Scarab elle-même (`scarab-schema.xml`).

A partir de cette définition formelle du schéma de la base, Torque va générer :

- les scripts SQL (DDL) de création de tables pour différents SGBDR;
- un ensemble de classes java qui mappent sous forme d'objets les entités de la base (les "peers").

## Générer des identifiants uniques : la table `id_table`

La plupart des tables du modèle de données de Scarab ont des entiers comme clés primaires. La plupart des SGBDR ont un mécanisme natif pour générer des clés primaires (entiers pas nécessairement consécutifs) : séquences Oracle, colonnes auto-incrémentées dans MySQL ou MSSQL, etc.

Comme Scarab se veut portable, la génération de ces clés primaires ne pouvait facilement se reposer sur ces mécanismes natifs. Scarab utilise donc pour cela une fonctionnalité de Torque ("id broker").

La table `id_table` est donc utilisée pour attribuer les clés primaires aux différentes tables.

| <b>id_table</b> |             |             |
|-----------------|-------------|-------------|
| PK              | ID_TABLE_ID | INTEGER     |
| N               | TABLE_NAME  | varchar-255 |
| A               | NEXT_ID     | INTEGER     |
| A               | QUANTITY    | -UnNamed-   |

## Intervalles d'identifiants réservés

Par convention, tous les identifiants inférieurs à 10.000 sont réservés au développement (données par défaut, données d'exemple, etc.)

**Table 6.1. Intervalles d'ID**

| Intervalle d'ID | Usage   |
|-----------------|---|
| 0-99            | Données nécessaires au fonctionnement de Scarab ( <i>required data</i> ). |
| 100-189         | Données par défaut.   |
| 190-199         | Données relatives à l'utilisateur anonyme.                                |
| 200-299         | Gabarits JIRA.  |
| 300-399         | Gabarits Bugzilla.  |
| 1000-1999       | Données d'exemple ( <i>sample data</i> ).                                 |
| 2000-8999       | (Réservé.)  |
| 9000-9999       | Réservé à l'utilisateur (gabarits ou types de fiches personnalisés, etc.) |

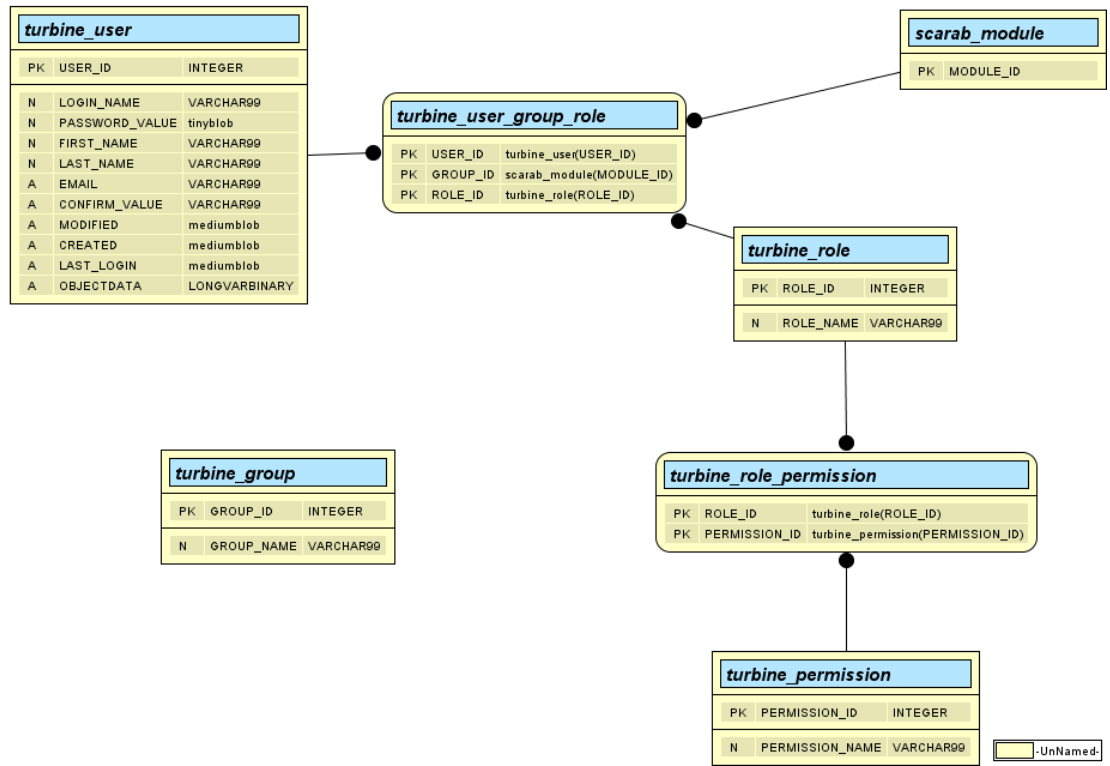
## Les tables du framework Turbine

Ces tables servent à authentifier et à gérer les utilisateurs, leur rôles et leurs droits dans les différents modules.

Les tables de cette partie du modèle sont :

- **turbine\_user** : contient les utilisateurs, leurs identifiants et authentifiants.
- **turbine\_role** : définit la liste des rôles. Les rôles sont communs à tous les modules de Scarab.
- **turbine\_permission** : définit les différentes permissions associées à l'application.
- **turbine\_group** : cette table, peut-être nécessaire au bon fonctionnement de Turbine à l'exécution, n'est pas utilisée dans Scarab. Elle est toujours vide.
- Les tables **turbine\_user\_group\_role** et **turbine\_role\_permission** sont des tables de

relation m-à-n dont le fonctionnement est explicité par le diagramme ci-dessous :

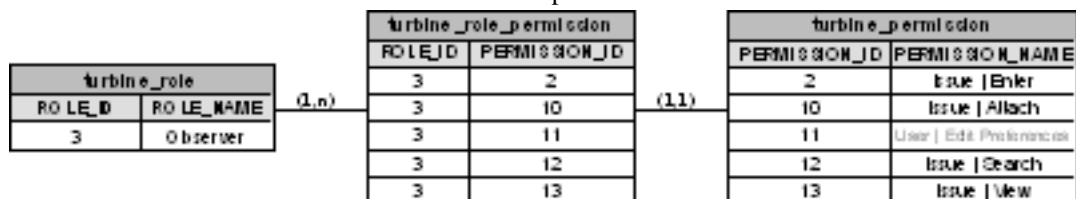


La capture d'écran montre la définition des rôles dans Scarab (avec les données d'exemple) :

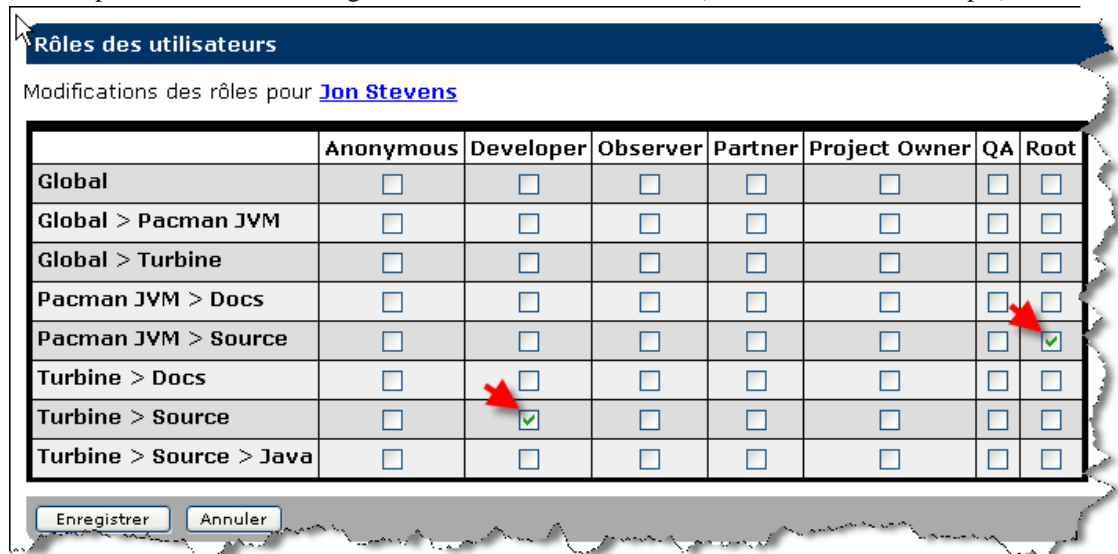
### Éditer les droits pour le rôle [Observer]

| 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           |

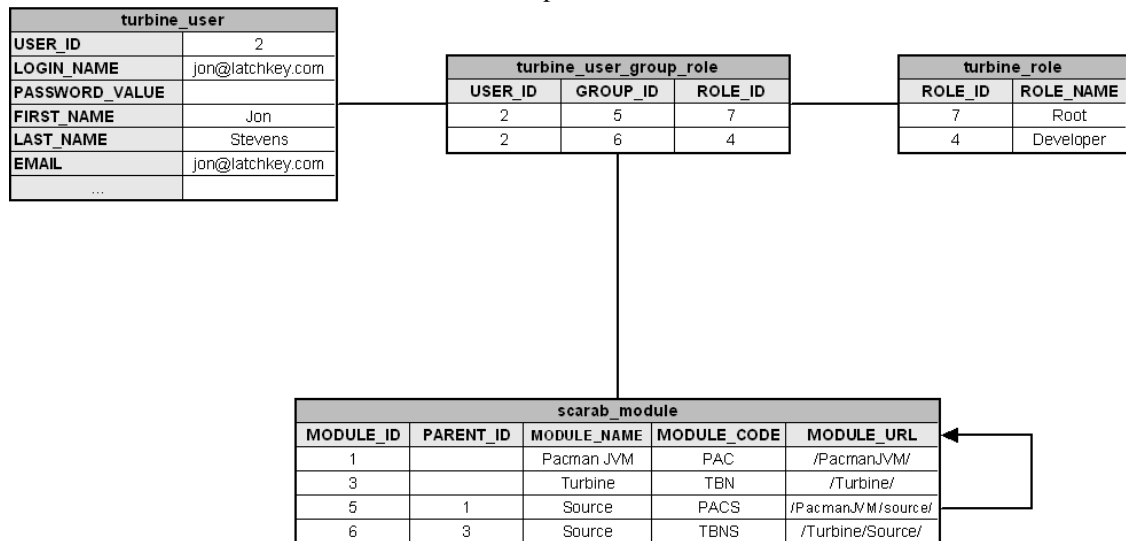
Voici une illustration du modèle de données correspondant :



Cette capture d'écran montre la gestion des rôles d'un utilisateur (avec les données d'exemple) :



Et voici l'illustration du modèle de données correspondant :

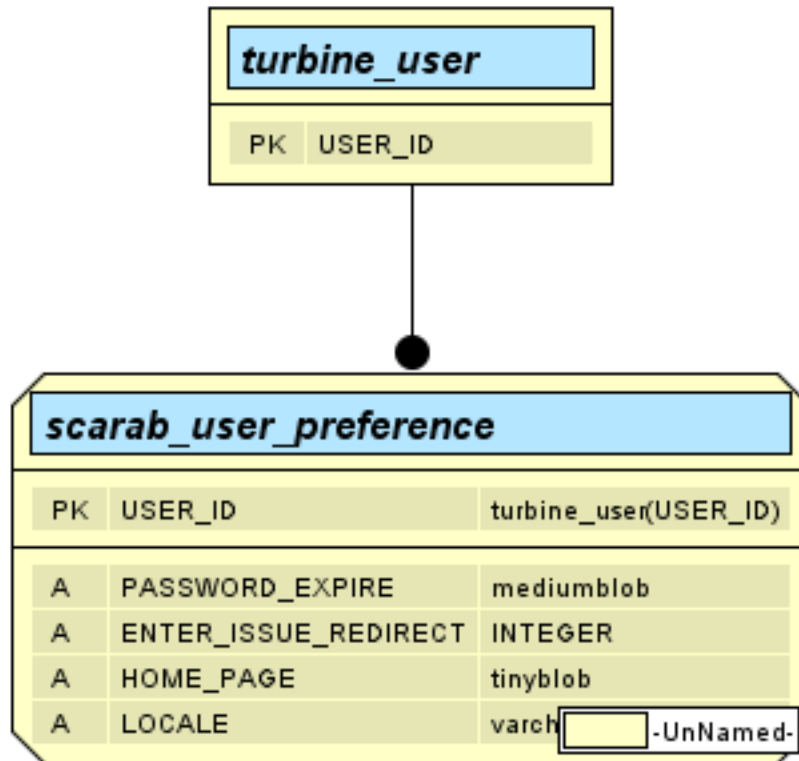


## Les tables de l'application Scarab

### Entités liées à l'utilisateur

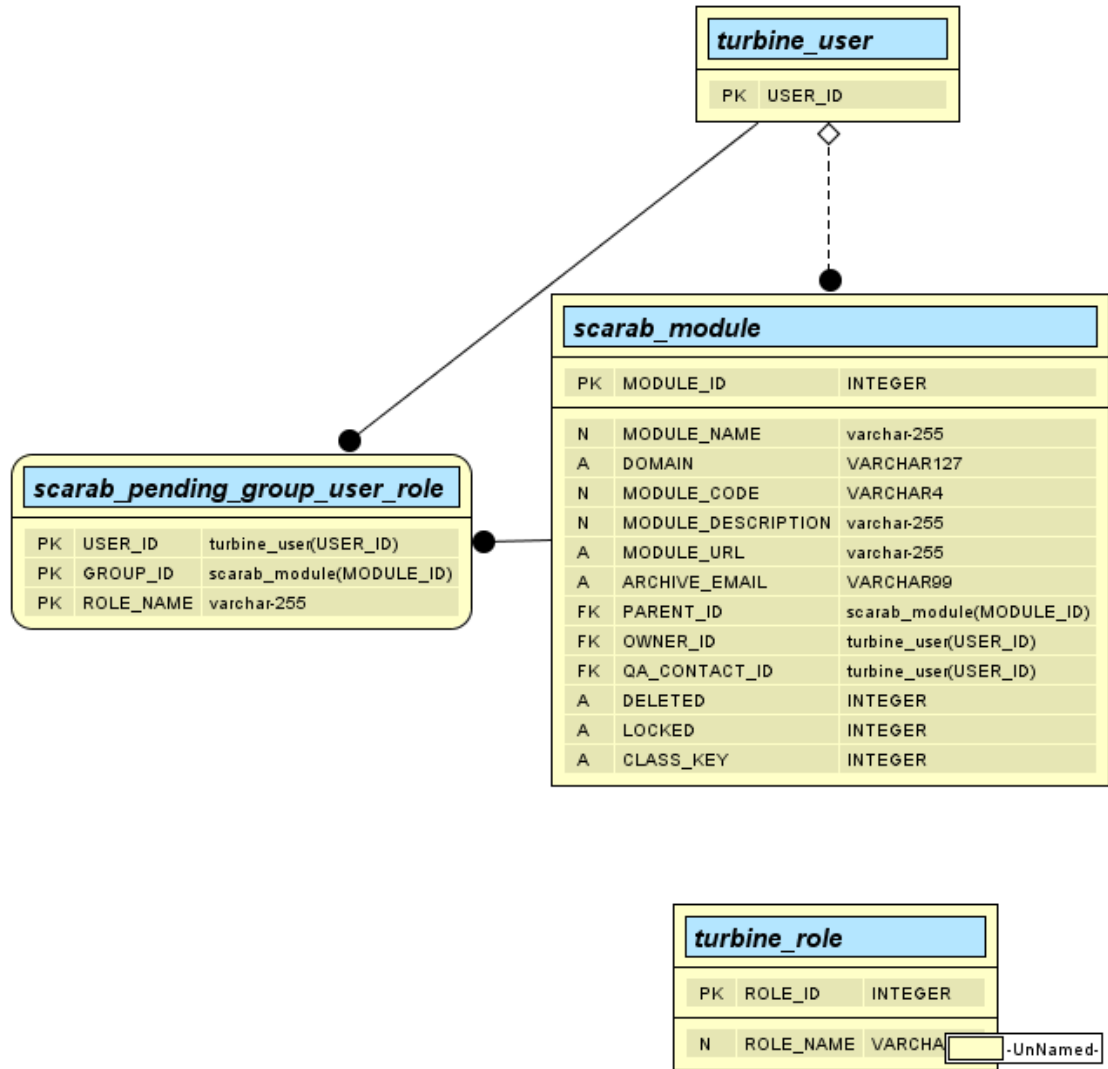
Avant de rentrer dans le coeur de l'application, mentionnons deux séries d'entités simples liées à l'utilisateur.

### Préférences



## Demandes en attente d'approbation

La table `scarab_pending_group_user_role` stocke temporairement les demandes de rôles effectuées par les utilisateurs, en attente d'approbation par l'administrateur.



## Modules

La liste des modules et les données correspondantes sont stockées dans la table scarab\_module.

| <b>scarab_module</b> |                    |  |
|----------------------|--------------------|--|
| 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-"/> |

| <b>Modules</b>                                |                      |
|---|----------------------|
| Nom   | Description          |
| <a href="#">Global</a>                        | Built-in root module |
| <a href="#">Global &gt; Pacman JVM</a>        | Sample project       |
| <a href="#">Global &gt; Turbine</a>           | The Turbine Project  |
| <a href="#">Pacman JVM &gt; Docs</a>          | Documentation        |
| <a href="#">Pacman JVM &gt; Source</a>        | Source               |
| <a href="#">Turbine &gt; Docs</a>             | Documentation        |
| <a href="#">Turbine &gt; Source</a>           | Source               |
| <a href="#">Turbine &gt; Source &gt; Java</a> | Java                 |

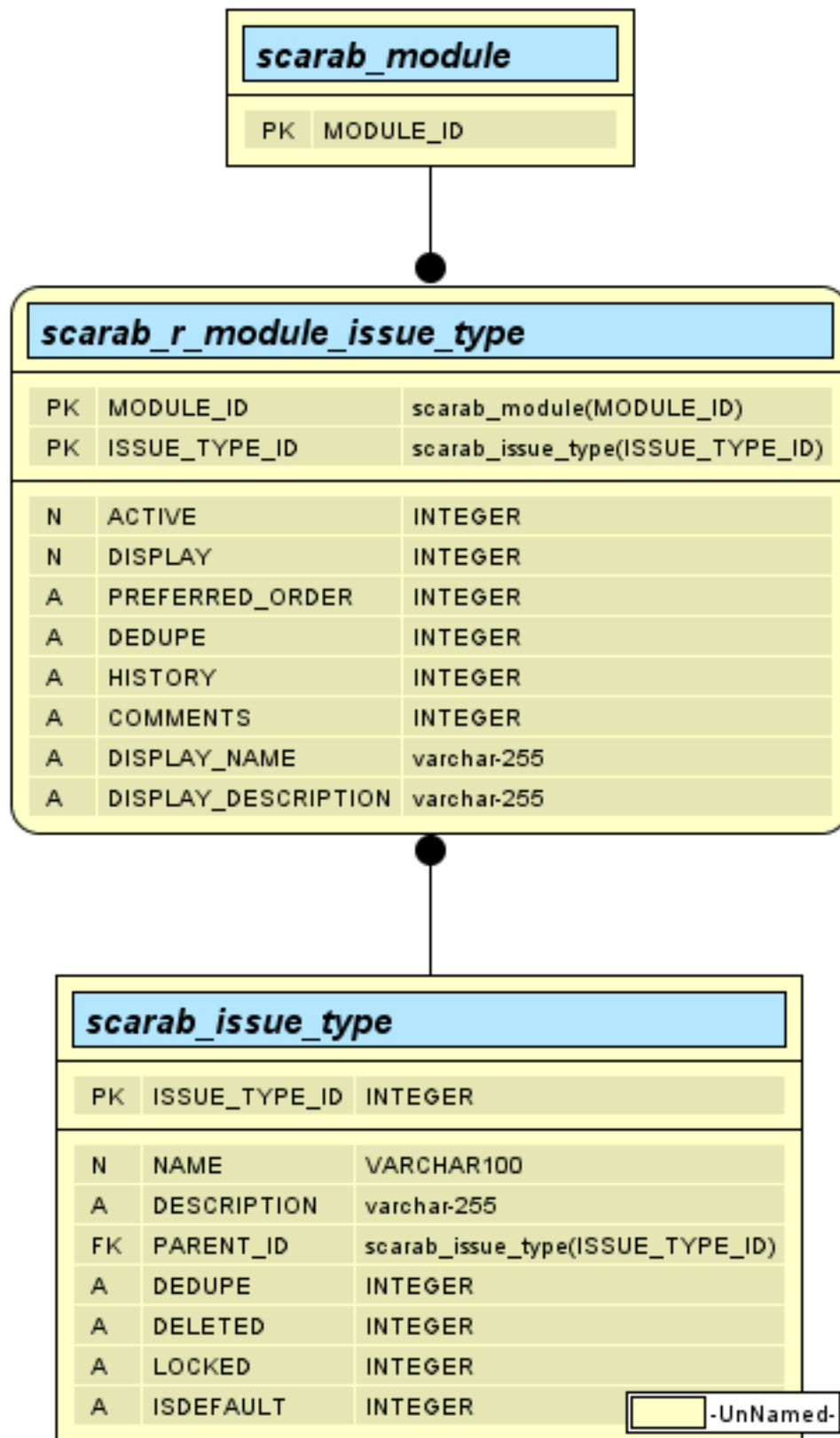
**Tip**

Notre responsable Qualité m'a demandé un jour une table de correspondance entre les codes modules et leur nom. Souvent, ce nom ne fait sens que précédé du libellé du module parent. Ce genre de tableau peut s'obtenir assez simplement à partir de la table SCARAB\_MODULE en exécutant la requête suivante :

```
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;
```

Et voilà !

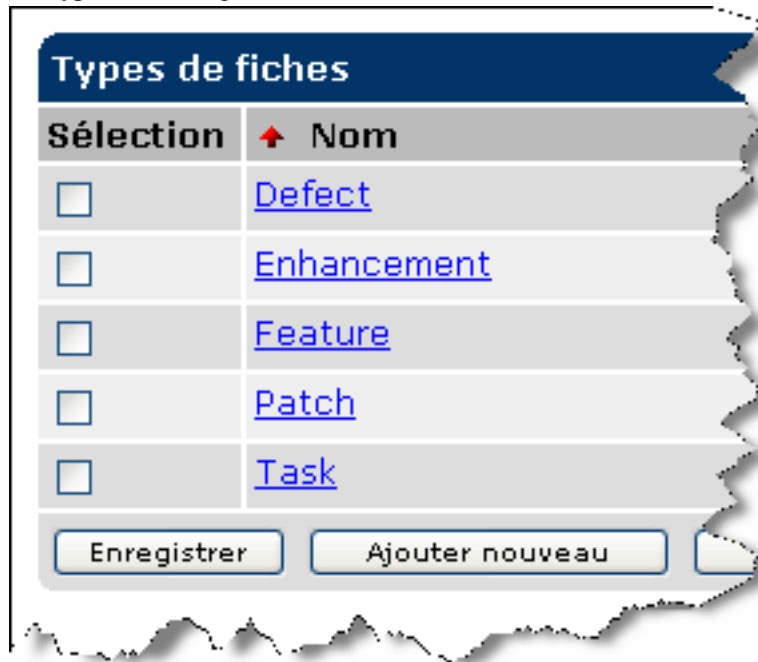
## Types de fiches



Les types de fiches sont stockés dans la table `scarab_issue_type`. Les modules et les types de

fiche sont associés les uns aux autres dans une relation de plusieurs à plusieurs (m-n) par le biais de la table de relation scarab\_r\_module\_issue\_type.

Les types de fiches globaux, illustrés ci-dessous sont associés au module "Global" (dont l'ID est 0).



The screenshot shows a web interface with a dark blue header containing the text "Types de fiches". 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".

| Sélection                | Nom                         |
|--------------------------|-----------------------------|
| <input type="checkbox"/> | <a href="#">Defect</a>      |
| <input type="checkbox"/> | <a href="#">Enhancement</a> |
| <input type="checkbox"/> | <a href="#">Feature</a>     |
| <input type="checkbox"/> | <a href="#">Patch</a>       |
| <input type="checkbox"/> | <a href="#">Task</a>        |

Enregistrer    Ajouter nouveau

## Attributs

| <b>scarab_attribute</b> |                    |  |
|-------------------------|--------------------|--|
| 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                                  |

| <b>scarab_attribute_type</b> |                     |  |
|------------------------------|---------------------|--|
| 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                                  |

| <b>scarab_attribute_class</b> |                      |             |
|-------------------------------|----------------------|-------------|
| 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

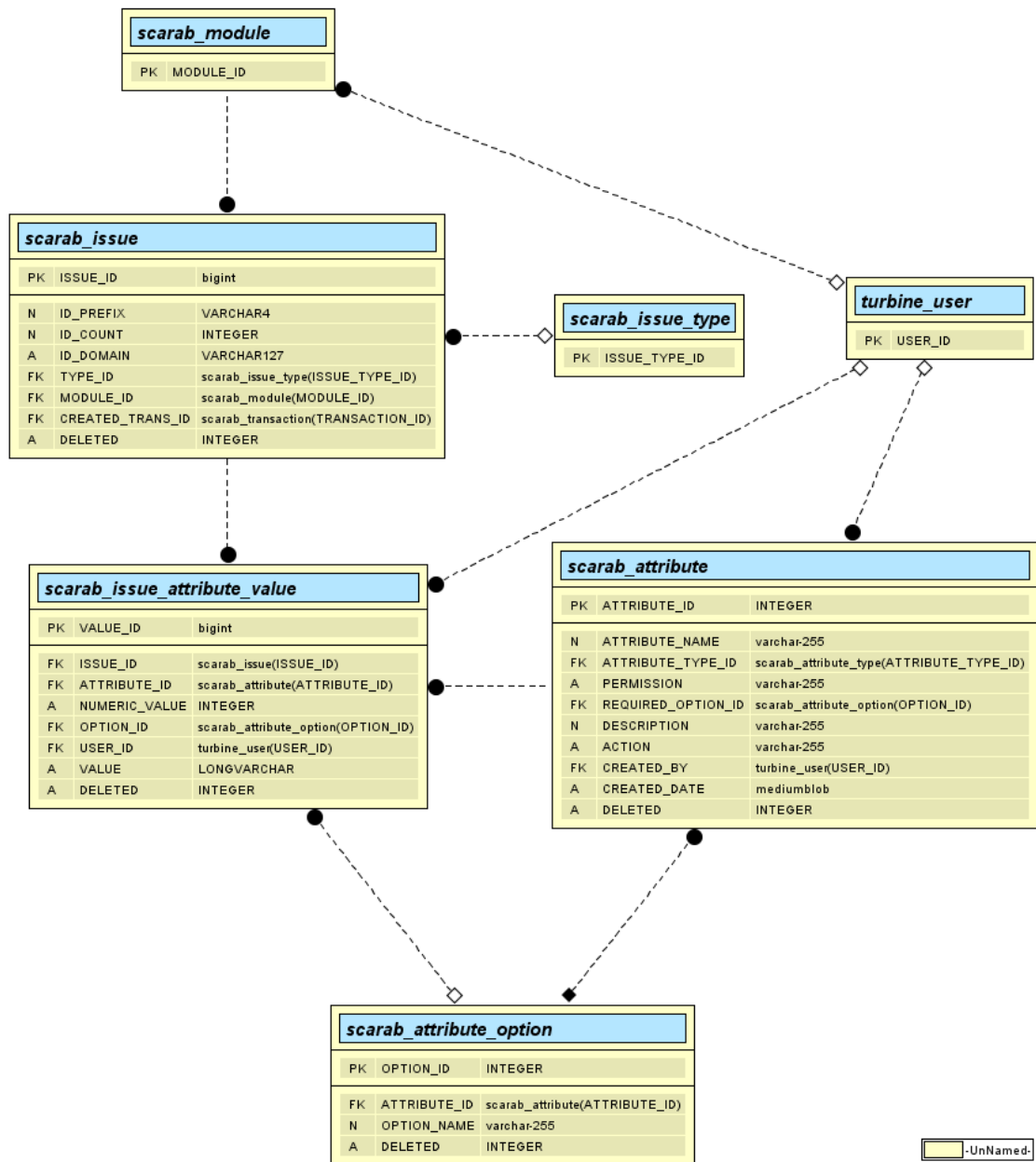
| Sélection                | Nom                             | Description     | Type de |
|--------------------------|---------------------------------|-----------------|---------|
| <input type="checkbox"/> | <a href="#">Description</a>     | Description     | long    |
| <input type="checkbox"/> |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

Attributs globaux | Attributs utilisateur globaux

**Attributs utilisateur globaux**

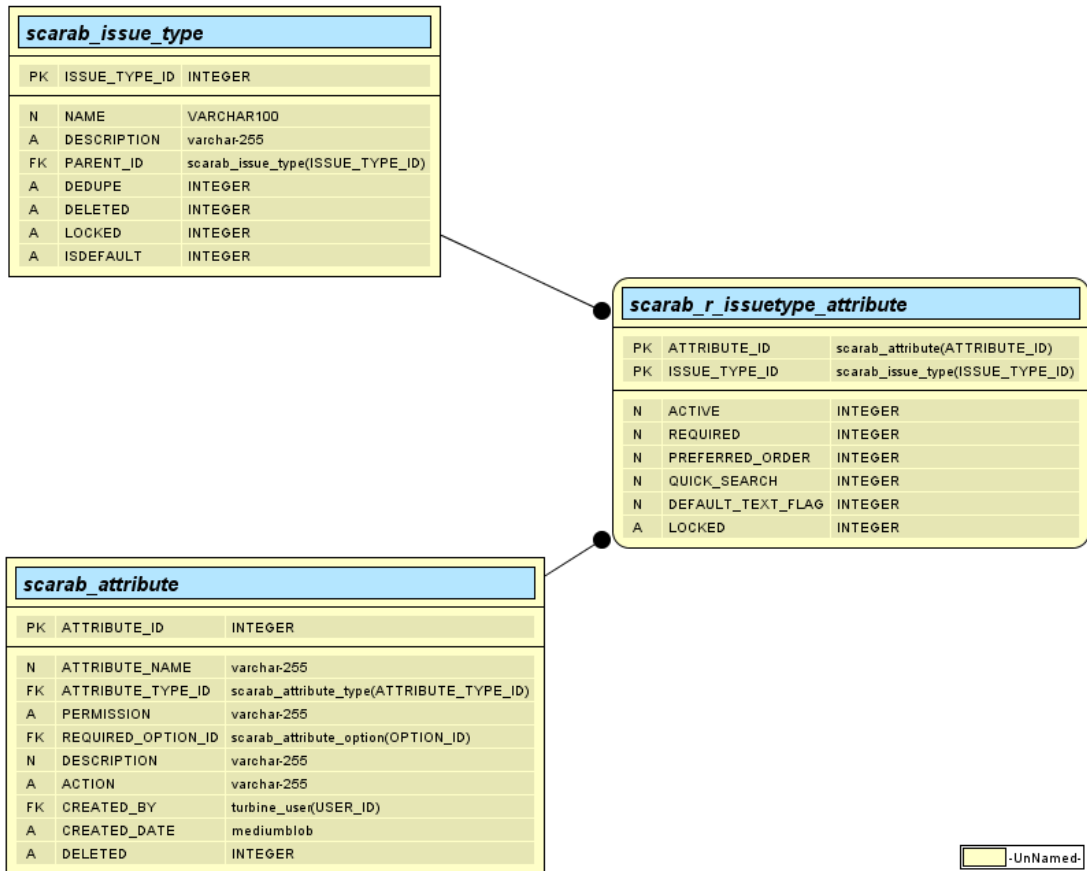
| Sélection                | Nom                        | Description | Action mél                            | Droit de base                                 |
|--------------------------|----------------------------|-------------|---------------------------------------|---|
| <input type="checkbox"/>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

## Fiches



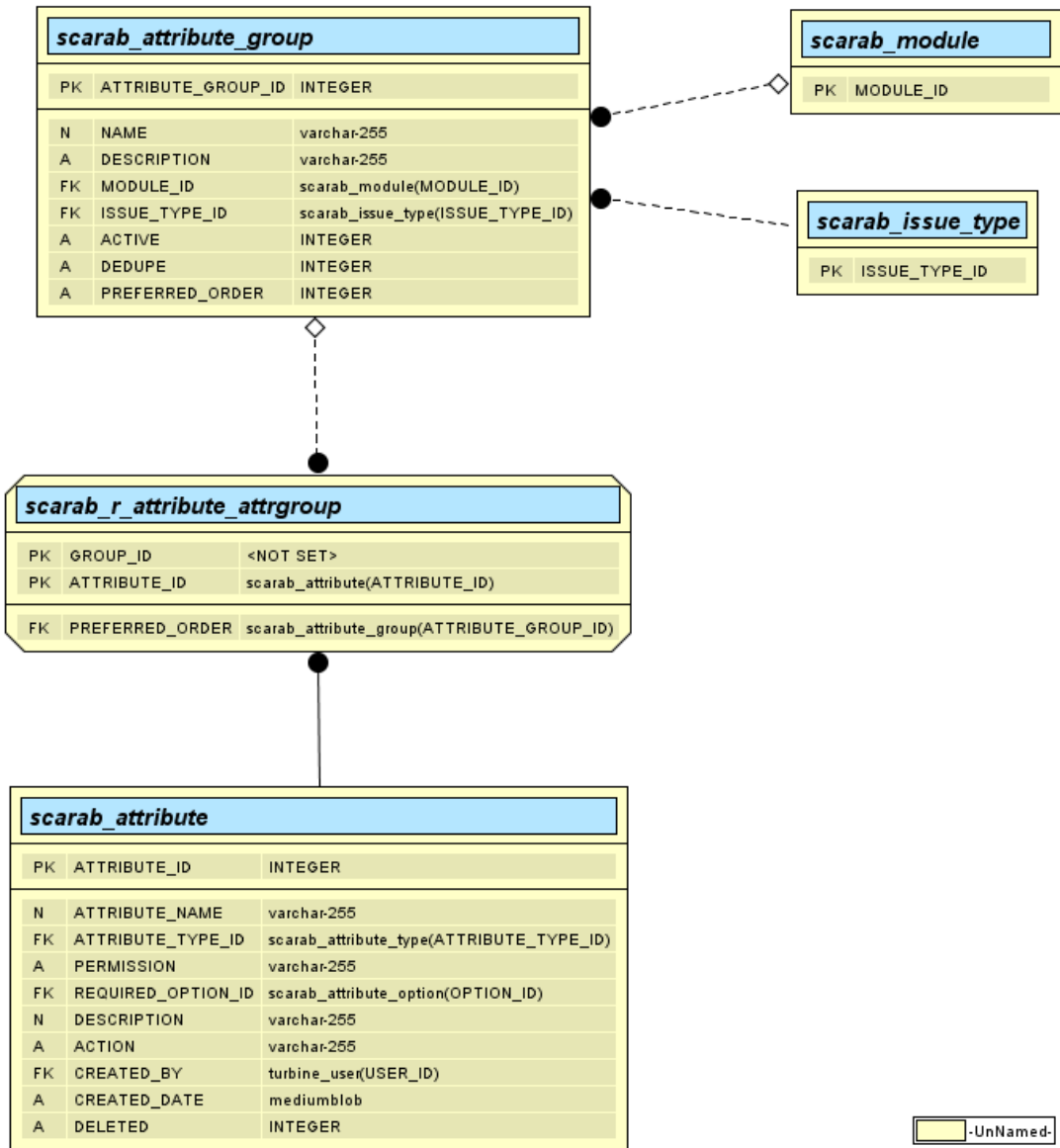
## Attributs

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



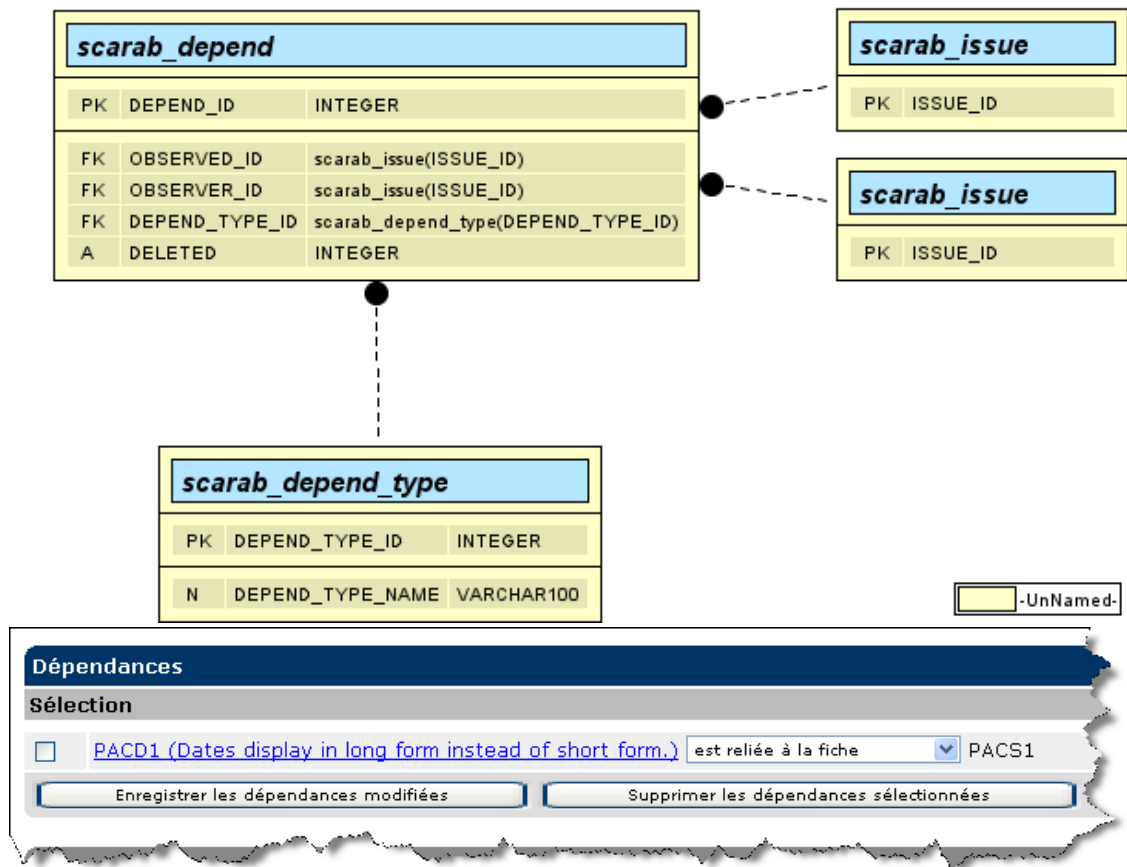
## Groupes d'attributs

Pour un module et un type de fiches donnéLe regroupement des attributs est utilisé

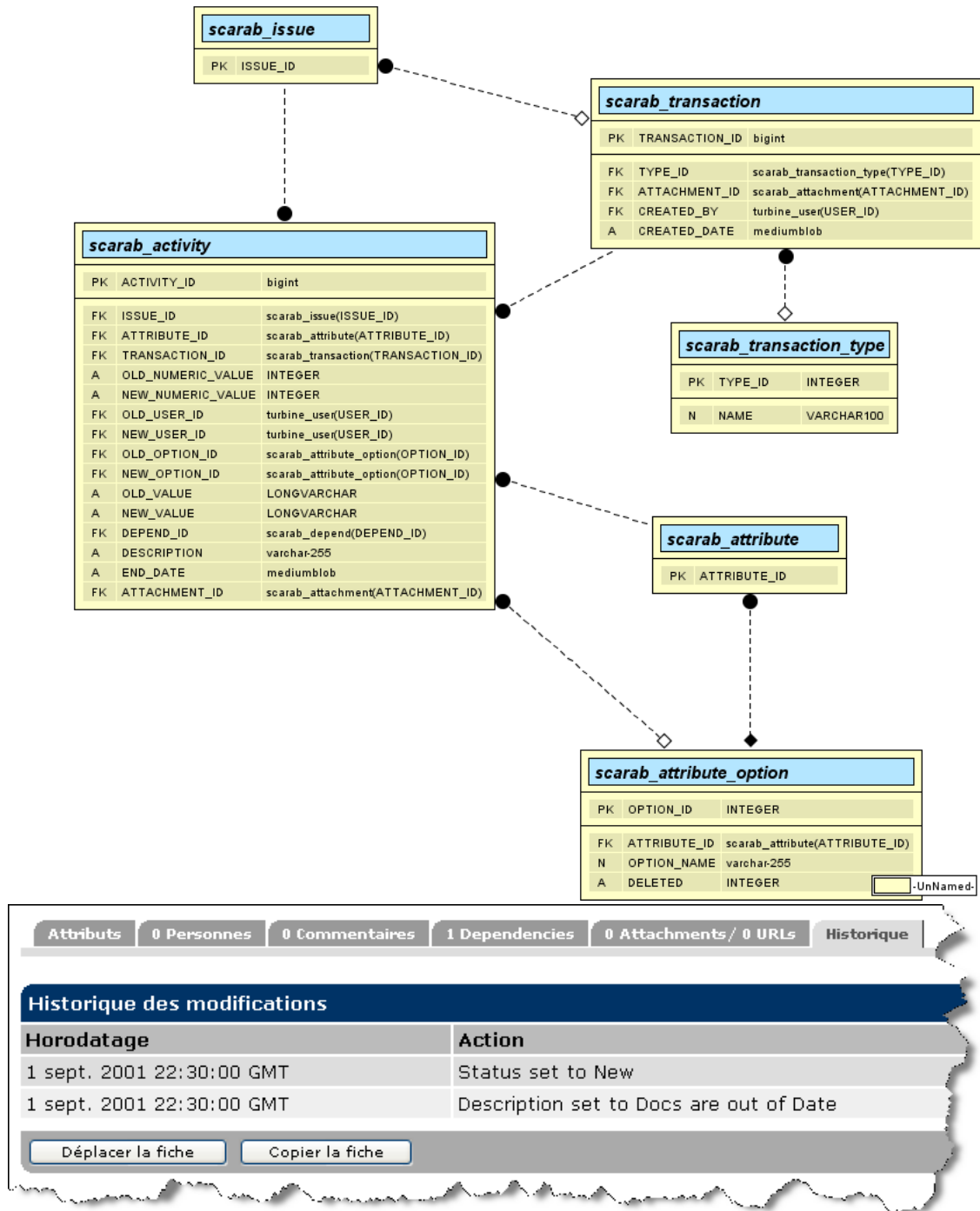


## Pièces jointes (*attachments*)

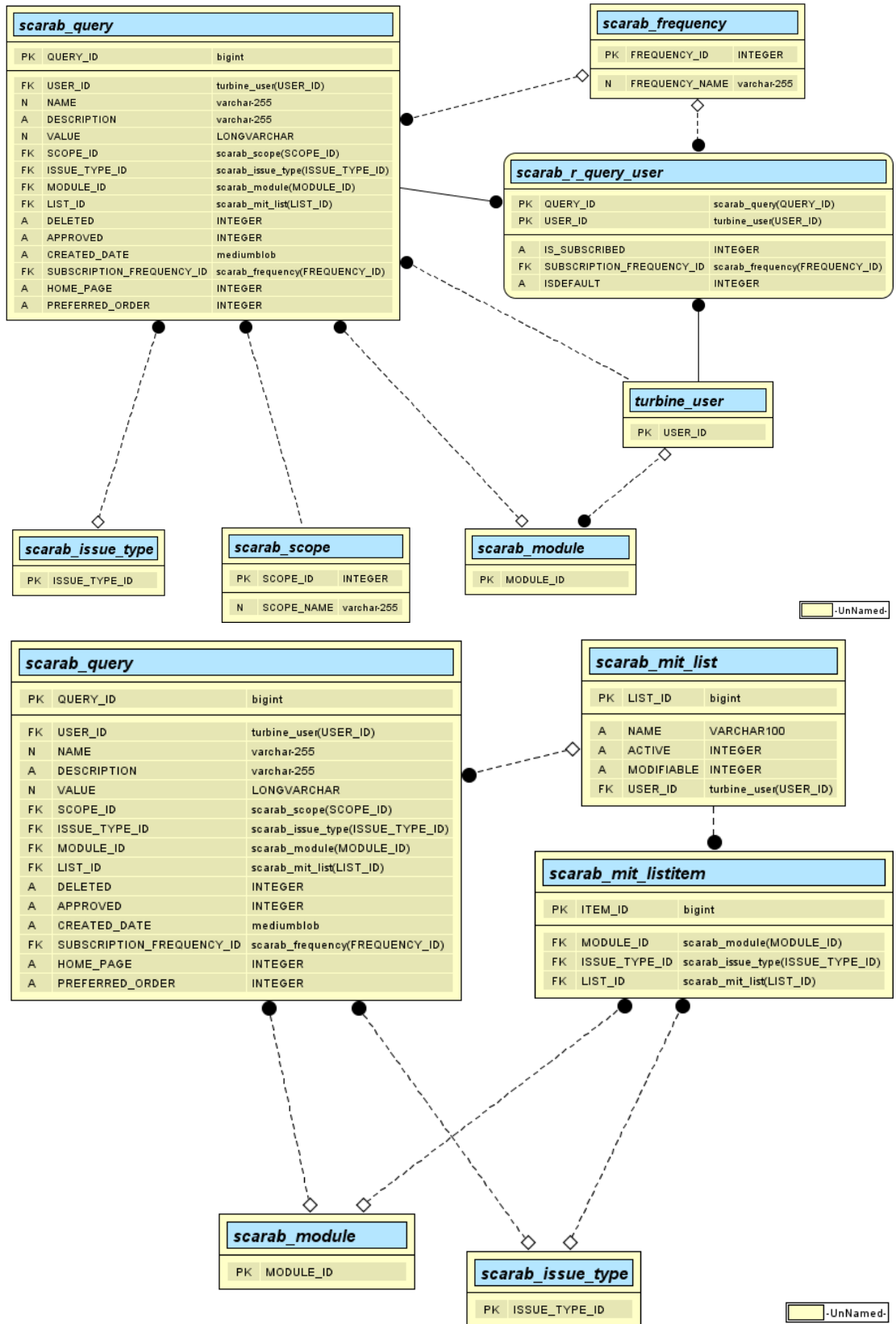
## Dépendances



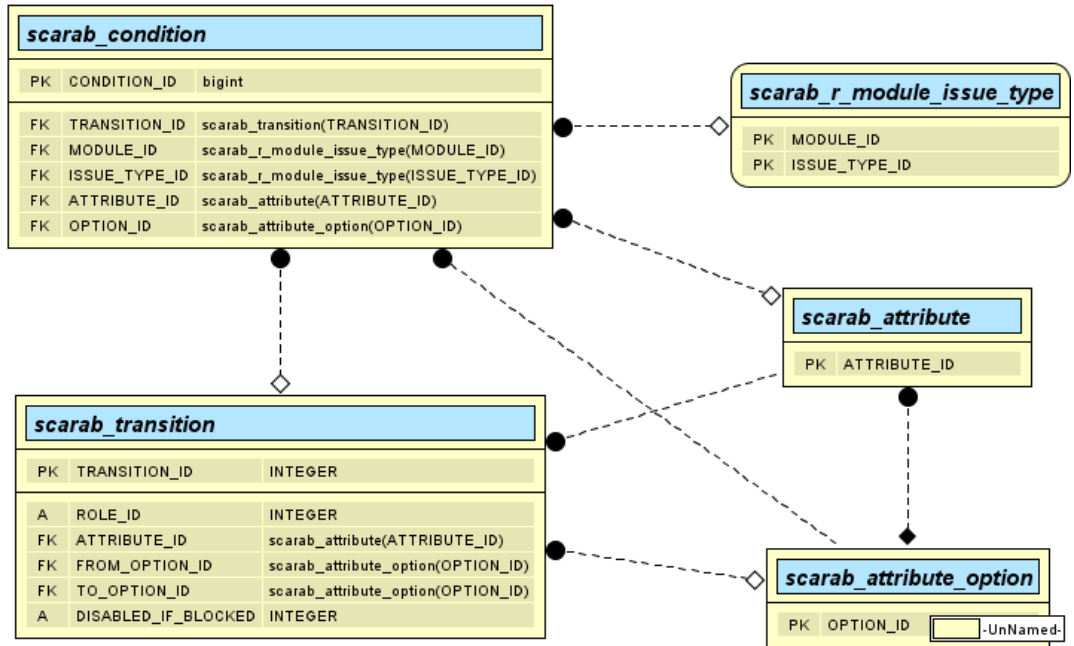
## Historique



## Requêtes



## Workflow



# Chapter 7. Workflow

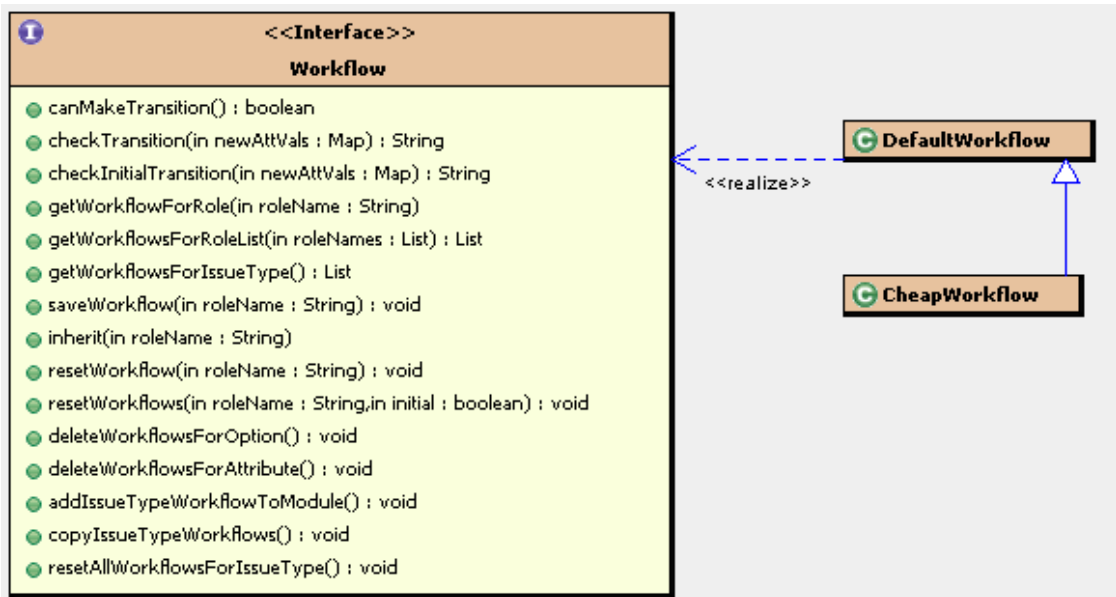
## Introduction

Comme il a été dit brièvement dans le chapitre *Workflow* du *Guide de l'utilisateur*, Scarab n'a pas de workflow "codé en dur". Les transitions d'état lors de l'enregistrement des fiches sont toutes permises, et cela n'entraîne pas non plus d'interaction avec quelque système externe que ce soit.

Examinons brièvement comment ceci est implémenté.

## Les workflows dans Scarab

Techniquement, un workflow est toute classe java qui implémente l'interface `org.tigris.scarab.workflow.Workflow` (le code source de cette interface se trouve sous `src/java`).



Le workflow est instancié dynamiquement par Turbine sur base d'une déclaration dans `Scarab.properties` (ce fichier se trouve à l'exécution dans l'arborescence de l'application web sous `WEB-INF/conf`).

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

## DefaultWorkflow

C'est le workflow par défaut pour toutes les versions de Scarab jusqu'à la *milestone* 19.

Ce workflow est implémenté dans la classe

`org.tigris.scarab.workflow.DefaultWorkflow`. C'est un bouchon (*stub*) qui ne fait rien et accepte toutes les transitions.

## BasicWorkflow

C'est le workflow par défaut pour les versions de Scarab à partir de la *milestone* 20. Son fonctionnement est expliqué dans le *Guide de l'utilisateur*.

Ce workflow est implémenté dans la classe `org.tigris.scarab.workflow.CheapWorkflow` (malgré les apparences). Il implémente l'interface *Workflow* indirectement en héritant du workflow par défaut, `DefaultWorkflow` sans doute pour ne pas devoir redéfinir certains comportements par défaut.

## Définir un nouveau workflow

Pour définir un nouveau workflow, utilisant d'autres mécanismes, communiquant avec un ou plusieurs systèmes externes ou simplement adapté aux besoins et aux processus de votre organisation, il vous suffit donc :

- d'écrire une classe java qui implémente l'interface `org.tigris.scarab.workflow.Workflow` (soit directement, soit en héritant de `org.tigris.scarab.workflow.DefaultWorkflow`);
- de définir le nom de cette classe dans le fichier `Scarab.properties` mentionné plus haut.

## Utilisation de WfmOpen comme moteur de workflow pour Scarab

Quelqu'un a commencé à implémenter une passerelle entre Scarab et le moteur de workflow open-source WfmOpen. Il reste de cette implémentation une entrée sur le *tracker* de SourceForge à cette 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) [http://sourceforge.net/tracker/index.php?func=detail&aid=961620&group\_id=76143&atid=546206].

L'état de ce développement n'est pas connu.

---

# Chapter 8. Accéder à Scarab par XML-RPC

Il est possible d'utiliser à distance les fonctionnalités de Scarab via XML-RPC.

Un exemple d'utilisation se trouve dans l'arborescence des sources (`src/java`):

- la classe `org.tigris.scarab.util.SimpleHandler.java` (serveur, s'exécute côté Scarab) est utilisée pour l'intégration Scarab-Subversion (se référer au chapitre correspondant du *Guide de l'utilisateur*);
- la classe `org.tigris.scarab.util.SimpleHandlerClient.java` (client, s'exécute dans une autre application) est un exemple simple d'utilisation du serveur XML-RPC.