Tiny
3 avril 2006

Tiny
Refonte de la gestion des droits

v1.1

Sommaire

1. Introduction

Jusqu'à la version 1.5.x, Tiny présentait une gestion très simple des droits et des utilisateurs. Les fonctionnalités offertes permettaient de s'adapter simplement à la plupart des organisations métiers.

Deux choses étaient pourtant regrettables :
- l'absence de la notion de groupes d'utilisateurs
- le manque de finesse de la gestion des droits applicatifs

On se propose donc dans ce document de décrire les solutions mises en place pour pallier à ces deux défauts.

Le chapitre 2 décrit l'implémentation des groupes.

De façon à permettre une gestion fine des droits des utilisateurs / groupes sur les modules Tiny, il s'est avéré nécessaire de référencer en base les modules d'une instance, et pour chaque modules la liste des actions. L'implémentation de cette bibliothèque des modules est décrite au chapitre 3.

La dernière partie du document détaille la gestion des droits applicatifs dans Tiny. Ce paragraphe traitera principalement trois points :
- la gestion des droits des utilisateurs / groupes sur les sites : chapitre 4
- la gestion des droits éditoriaux des utilisateurs / groupes (i.e. droits sur les rubriques et les articles) : chapitre 5
- la gestion des droits d'utilisation des utilisateurs / groupes (i.e. droits sur les modules) : chapitre 6

2. Groupes d'utilisateurs

2.1. Définitions

On intègre ici la notion de groupe d'utilisateurs.
Un **utilisateur** est défini par un enregistrement dans la table *personne*.
Un **groupe** est un ensemble d'utilisateurs. Dans un souci de simplicité, nous partons du principe que :

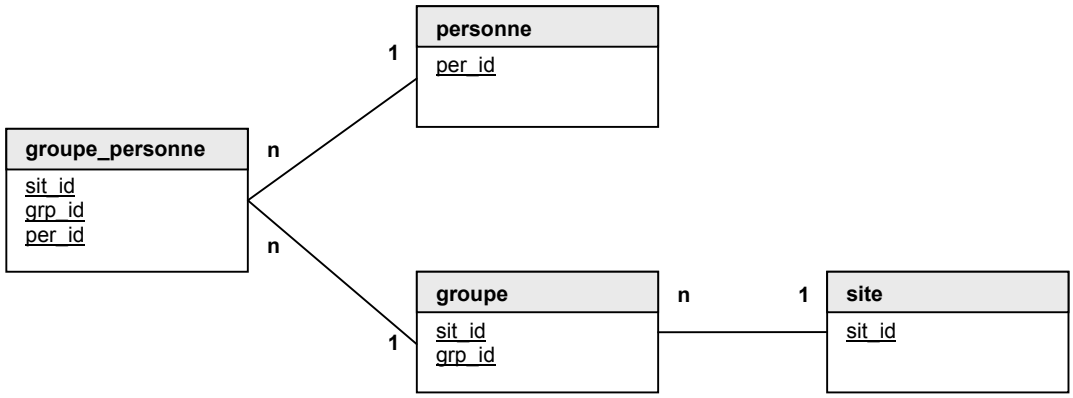
- un groupe est défini pour un site donné
- un groupe ne contient que des utilisateurs : on ne définira pas d'ensembles de groupes (pas de récursivité)

C'est dans un souci de simplicité du code et des interfaces que l'on prend le parti de ne pas gérer de groupes transversaux (groupes disponibles pour l'ensemble des sites et affectés à la discrétion de l'administrateur) : la gestion des utilisateurs et des groupes sera laissée à la discrétion de l'administrateur de chaque site ; les groupes créés seront donc des groupes locaux.

2.2. Implémentation en base

L'implémentation est très naturelle :

2.2.1. Schéma global



2.2.2. Table GROUPE

Cette table contient les groupes locaux créés pour chaque site de l'instance Tiny

Champ	Type	Description
sit_id	varchar(10)	Identifiant du site
grp_id	integer	Identifiant du groupe
grp_libelle	varchar(128)	Libellé du groupe
grp_description	text	Description du groupe

2.2.3. Table GROUPE_PERSONNE

Cette table matérialise l'appartenance de personnes à un groupe :

Champ	Type	Description
sit_id	varchar(10)	Identifiant du site
grp_id	integer	Identifiant du groupe
per_id	varchar(30)	Identifiant de la personne

3. Bibliothèque de modules

3.1. Définitions

3.1.1. Module

Techniquement, un module est un objet PHP qui hérite de la classe Container. En pratique, dans Tiny, tout ce qui n'est pas CMS est un module. Par exemple :
- RubManager sur le back-office : module de gestion des rubriques
- FaqFront sur le front-office : module d'affichage des F.A.Q.
- FaqManager sur le back-office : module de gestion des F.A.Q. d'un site

En pratique, il n'y a pas de différence entre un module front et un module back. En revanche, dans le code, on doit pouvoir différencier les deux. C'est pourquoi dans la base de données, les modules seront typés.

Certains modules sont génériques, dans le sens où ils peuvent être utilisés sur tous les sites d'une instance : c'est par exemple le cas de :
- FaqFront : utilisable sur tous les sites (front) d'une instance
- RubManager : utilisé sur le back de tous les sites d'une instance
- … etc.

En revanche, d'autres modules peuvent être spécifiques à un site, dans le sens où le développeur peut ne pas avoir pris la peine de code son module de façon générique. On doit donc pouvoir définir, pour chaque module, s'il est spécifique à un site ou s'il est générique.

3.1.2. Action

Une action est une action (!) que l'utilisateur peut effectuer sur un module, et que le développeur/concepteur veut réserver à des personnes habilitées. Par exemple, dans le module de gestion des rubriques, on pourra définir les actions suivantes :
- Créer une sous rubrique
- Modifier une rubrique
- Supprimer la rubrique
- … etc.

3.1.3. Rôle

Un rôle est un regroupement d'actions : un module pouvant contenir un grand nombre d'actions, il paraît judicieux de rassembler ces actions en rôles génériques. Les rôles seront octroyés aux utilisateurs (ou aux groupes) : ainsi si le rôle évolue, il suffit de modifier les actions associées au rôle pour impacter les droits de tous les utilisateurs ayant ce rôle.

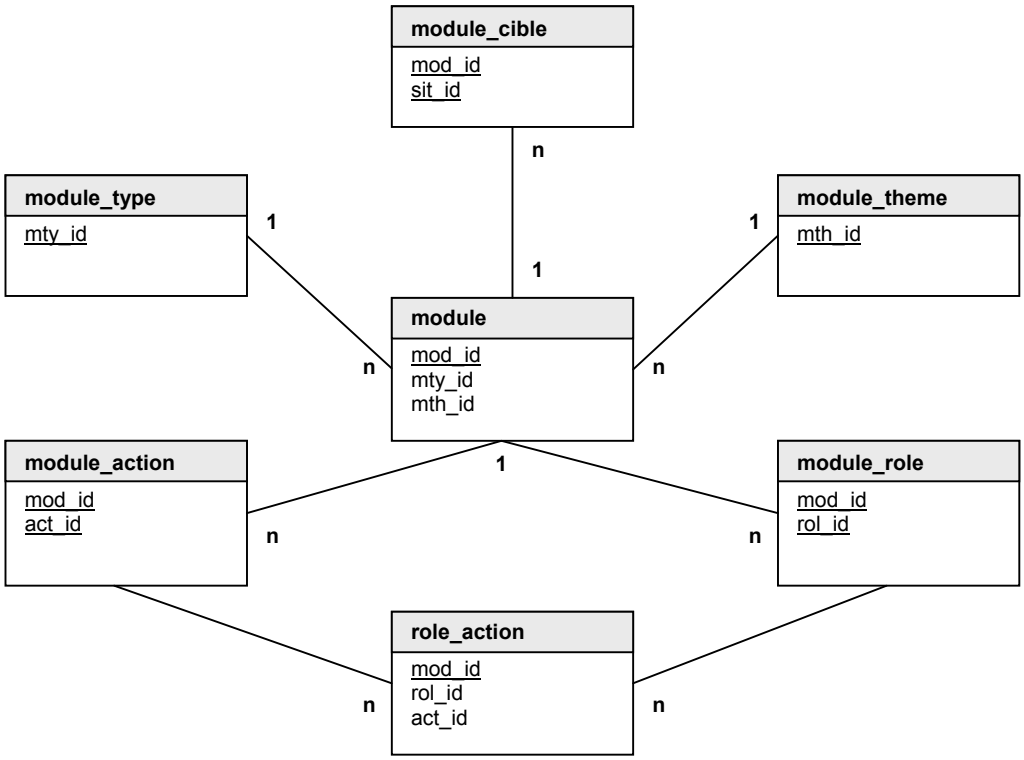
3.2. Description

L'objectif est de référencer en base l'intégralité des modules utilisés dans l'instance Tiny. Pour chaque module, on spécifiera :
- un type : module back-office ou front-office
- un thème : cet attribut sera utilisé dans le menu gauche du back-office, pour regrouper les modules par thématique (Gestion de contenu, Administration, …etc.)

- des actions : l'ensemble des actions qu'un utilisateur peut effectuer dans un module, et pour lesquelles on veut positionner des droits d'accès (ajouter, modifier, supprimer … etc.)

3.3. Implémentation en base

3.3.1. Schéma global



3.3.2. Table MODULE_TYPE

Cette table contient les deux types de modules tiny : back-office et front-office

Champ	Type	Description
mty_id	char(1)	Identifiant du type (« **B** », « **F** »)
mty_libelle	varchar(50)	Libellé du type

3.3.3. Table MODULE_THEME

Cette table contient les thèmes de modules :

Champ	Type	Description
mth_id	smallint	Identifiant du thème
mth_libelle	varchar(128)	Libellé du thème

3.3.4. Table MODULE_CIBLE

Cette table permet de spécifier si un module est générique ou spécifique à un site

Champ	Type	Description
mod_id	varchar(50)	Identifiant du module
sit_id	varchar(10)	Identifiant de site : - si « back » : module générique - si « *sid* » : module spécifique au site *sid*

Remarques :
- pour rendre générique le module spécifique *mod* du site *sid*, il suffit de modifier son enregistrement dans la table module_cible : [mod, sid] ⇒ [mod, back]
- le fait que la clé primaire soit composée (mod_id et sit_id) permet de gérer le cas où un module serait spécifique à une partie des sites de l'instance, sans pour autant être générique …

3.3.5. Table MODULE

Cette table contient la liste des modules de l'instance :

Champ	Type	Description
mod_id	varchar(50)	Identifiant du module (nom de la classe PHP)
mty_id	char(1)	Type de module
mth_id	smallint	Thème
mod_libelle	varchar(128)	Libellé
mod_description	text	Description

3.3.6. Table MODULE_ACTION

Cette table contient les actions liées à un module :

Champ	Type	Description
mod_id	varchar(50)	Identifiant du module
act_id	varchar(50)	Identifiant de l'action (nom naturel : « **ajouter** », « **supprimer** » …)
act_libelle	varchar(128)	Libellé
act_description	text	Description

3.3.7. Table MODULE_ROLE

Cette table contient les rôles associés à un module :

Champ	Type	Description
mod_id	varchar(50)	Identifiant du module
rol_id	int	Identifiant du rôle (autoincrément)
rol_libelle	varchar(128)	Libellé
rol_description	text	Description

3.3.8. Table ROLE_ACTION

Cette table décrit la liste des actions associées à chaque rôle

Champ	Type	Description
mod_id	varchar(50)	Identifiant du module
rol_id	int	Identifiant du rôle
act_id	varchar(50)	Identifiant de l'action

4. Droits des utilisateurs / groupes sur les sites

4.1. Niveaux d'utilisateurs

On distinguera trois niveaux d'utilisateurs :
- **utilisateur** : il s'agit de l'utilisateur lambda ayant accès au back-office
- **administrateur de site** : ce niveau donne tous les droits à l'utilisateur sur le site qu'il administre (droits éditoriaux, et droits applicatifs)
- **super-administrateur** : ce niveau donne tous les droits à l'utilisateur sur tous les sites de l'instance (droits éditoriaux, et droits applicatifs)

Pour définir ces niveaux, on utilisera la table « droit » existante :
- pour être d'un niveau *utilisateur*, la personne devra s'être vue attribuer un privilège sur ce site
- pour être d'un niveau *administrateur*, la personne devra s'être vue attribuer le privilège « administrateur » sur ce site
- pour être d'un niveau *super-administrateur*, la personne devra s'être vue attribuer le privilège « administrateur » sur le site « back / Outil d'administration »

Ce choix d'implémentation s'appuie intégralement sur l'existant : cela facilitera la migration des sites déjà en place.

4.2. Les privilèges

4.2.1. Description

La notion de privilège n'est pas triviale. En effet, il y a deux types de privilèges :
- les privilèges éditoriaux
- les privilèges d'utilisation

L'octroi d'un *privilège éditorial* à une personne sur un site ne donne pas forcément un *droit* à cette personne sur ce site : le privilège éditorial n'est utilisable sur une rubrique que si, en plus de posséder le privilège, l'utilisateur est acteur de la rubrique.

L'octroi d'un *privilège d'utilisation*, en revanche, donne de facto les droits équivalents sur le site. L'utilisateur n'a donc pas besoin d'être acteur d'une rubrique pour exercer ses privilèges d'utilisation.

4.2.2. Evolution des privilèges

Les privilèges définis dans Tiny v1.5.x étaient les suivants :

0 : Aucun
1 : Utilisateur
2 : Tâches basiques
4 : Rédacteur
8 : Valideur
16 : Tâches sensibles
32 : Administrateur

Les privilèges 2 et 16 sont obsolètes : les droits d'utilisation des modules seront gérés comme décrit dans le chapitre suivant. Restent donc les privilèges éditoriaux (0, 2, 4, et 8), et les privilèges d'utilisation (1 et 32).

Désormais, nous en considèrerons plus que les privilèges suivants :

0 : Aucun
1 : Utilisateur
4 : Rédacteur
8 : Valideur
32 : Administrateur

4.2.3. Le super-administrateur

Pour implémenter la notion de *super-administrateur*, nous aurions pu créer un nouveau privilège (64 : super admin), et statuer qu'un super administrateur est une personne possédant le privilège *super admin* sur un site.

Toujours dans une optique d'économie de moyens et de compatibilité avec l'existant, nous préférons statuer qu'un *super-administrateur* est en fait un administrateur de l'outil d'administration : un *super-administrateur* est donc une personne possédant le privilège *administrateur* sur le site *back*.

4.2.4. Le privilège « Aucun »

Ce privilège permet de définir des utilisateurs comme acteurs de rubriques, sans pour autant leur donner de droit d'accès au back-office. C'est utile pour donner l'accès aux rubriques privées du site à des personnes qui ne sont pas censées accéder à l'outil d'administration.

5. Droits éditoriaux des utilisateurs / groupes

5.1. Description

En ce qui concerne les droits éditoriaux (création d'articles, validation, … etc.) nous nous appuierons encore sur l'existant. Seules quelques modifications seront apportées au modèle pour l'adapter à la gestion de groupes.

5.2. Problématique des privilèges éditoriaux dans Tiny 1.5.x

Dans la version 1.5.x et antérieur, un problème se pose dans le cas des sites protégés (c'est-à-dire dont l'accès au front-office de la rubrique est possible uniquement si on est acteur de la rubrique). On peut placer deux personnes en tant que rédacteur, chacune sur une rubrique différente, sans qu'elles aient la possibilité d'être rédacteur sur la rubrique de l'autre. Pour cela, il faut rendre ces personnes uniquement acteur de la rubrique où elles doivent exercer leur privilège de rédacteur. Cependant, dans ce cas, elles ne peuvent plus accéder au front-office de la rubrique dont elles ne sont pas acteur.

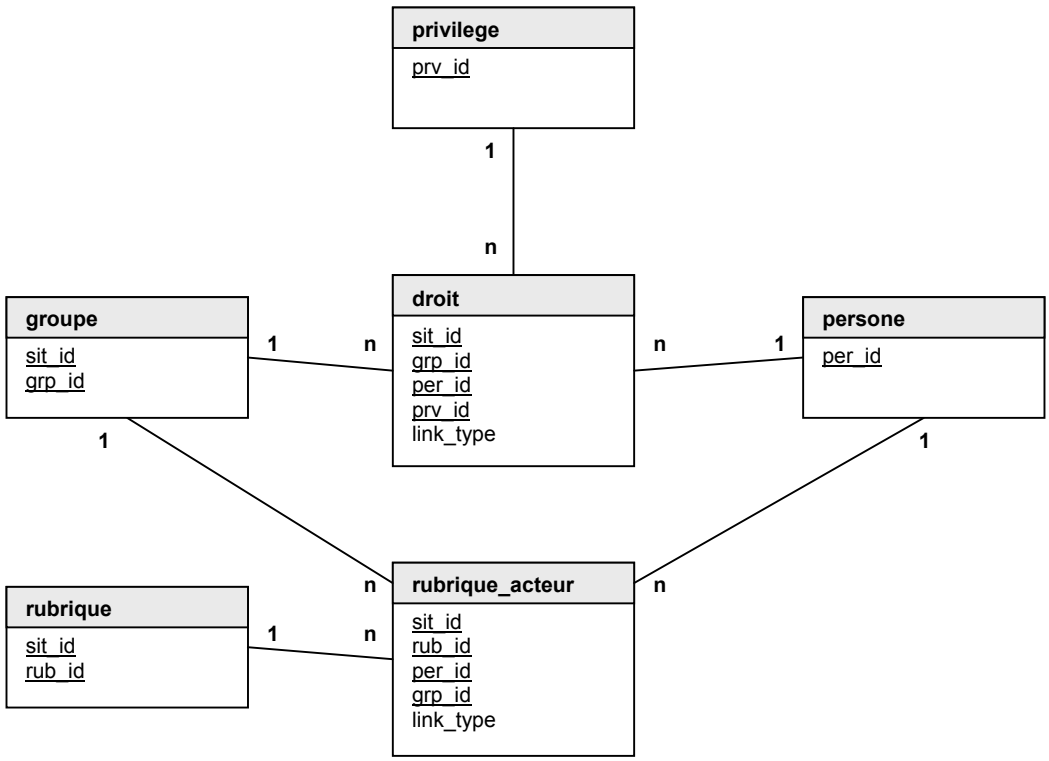
C'est la gestion de groupe qui permet de régler ce problème : il suffit de procéder comme suit :

- créer un groupe G1, lui donner le privilège de Rédacteur sur le site, et le rendre acteur de la rubrique R1
- créer un groupe G2, lui donner le privilège de Rédacteur sur le site, et le rendre acteur de la rubrique R2
- donner aux deux personnes le privilège Utilisateur sur le site, et les rendre acteur des deux rubriques : les deux personnes pourront donc consulter tout le front-office
- affecter une personne au groupe G1 : cette personne pourra rédiger des articles dans R1 (mais pas dans R2)
- affecter l'autre personne au groupe G2 : cette personne pourra rédiger des articles dans R2 (mais pas dans R1)

5.3. Implémentation en base

Comme cela a été dit dans ce qui précède, on s'appuie sur le modèle existant, légèrement modifié pour qu'il fonctionne aussi avec les groupes.

5.3.1. Schéma global



5.3.2. Description des modifications

Les modifications portent sur les tables **droit** et **rubrique_acteur** : deux champs ont été ajoutés à ces tables :

- link_type, char(1) : type de lien ; **P** : personne, **G** : groupe
- grp_id, integer : identifiant du groupe

Lorsqu'un enregistrement référence une personne, on a donc :

- per_id : identifiant de la personne
- grp_id : 0
- link_type : P

Lorsqu'un enregistrement référence un groupe, on a :

- per_id : « none »
- grp_id : identifiant du groupe
- link_type : G

Note : deux identifiants séparés (per_id et grp_id) sont utilisés ici car per_id est de type varchar : utiliser le même identifiant nous aurait contraint à utiliser ce même type pour grp_id, ce qui n'était pas souhaitable.

6. Droits d'utilisation des utilisateurs / groupes

6.1. Principe

La bibliothèque des modules recense les modules de l'instance. Or, chaque module est réutilisable :

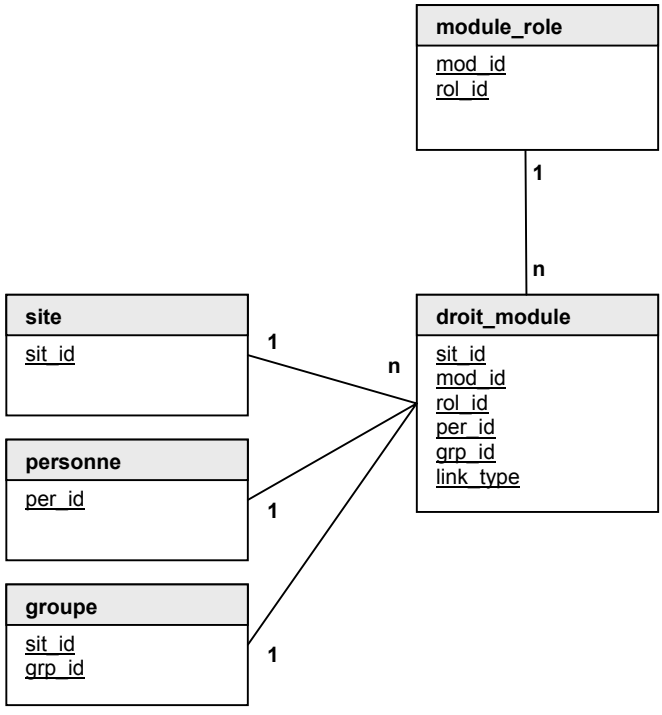
- chaque module de type front-office peut être implémenté sur plusieurs sites de l'instance
- chaque module du back-office peut être utilisé quelque soit le site courant (par exemple, le module de gestion des rubriques sert pour administrer les rubriques de tous les sites de l'instance)

Or quelque soit le type de module (front / back), il semble pertinent de définir site par site les droits d'utilisation de ce module, et ce action par action.

La modélisation en base des droits d'utilisation des modules consiste donc en une table décrivant qui – personne ou groupe – peut effectuer telle action de tel module sur tel site.

6.2. Implémentation en base

6.2.1. Schéma global



6.2.2. Table DROIT_MODULE

Champ	Type	Description
sit_id	varchar(10)	Identifiant du site
mod_id	varchar(50)	Identifiant du module
rol_id	int	Identifiant du rôle sur le module
per_id	varchar(30)	Identifiant de la personne (si link_type = P), « none » sinon
grp_id	integer	Identifiant du groupe (si link_type = G), « 0 » sinon
link_type	char(1)	Type de lien : **G** pour « groupe », **P** pour « personne »