Serveur

Sommaire

I. Généralités

Le serveur LightBird est un programme qui fonctionne en fond de tâche sur l'ordinateur qui héberge les fichiers à partager d'un utilisateur. Son rôle est distribuer ces fichiers aux clients, en fonction des droits d'accès qui leurs sont accordés.

Pour l'utilisateur final son fonctionnement se veut le plus simple et transparent possible. Dans la plus part des cas il n'a pas à s'en soucier, son administration se faisant via des interfaces séparés.

Le serveur est composé de trois parties principales, à savoir la **base de données** qui stocke les données persistantes, les **plugins** qui constituent l'intelligence du serveur et qui définissent comment il interagit avec les utilisateurs, et enfin le **serveur** lui-même qui a pour rôle d'orchestrer le tout.

Les plugins et la base de données sont traités dans des documentations séparés. Celle-ci se concentre sur le serveur.

Le serveur est développé en utilisant le Framework Qt, en version 4.7.0, à l'heure ou ces lignes sont écrites (05/2011), et fonctionne sur tous les systèmes supportés par Qt (principalement Windows, Linux, et Mac OSX).

II. Arguments

L'exécutable du serveur peut prendre deux arguments optionnels :

Argument	Description
-c *configuration.xml*	L'argument **–c**, est suivi du chemin complet (absolu ou relatif) vers le fichier de configuration du serveur. Ce dernier contient toutes les informations permettant au serveur de démarrer selon les préférences de l'utilisateur. Si cet argument n'est pas défini, le chemin de configuration par défaut est « **configurations/Configuration.xml** ». Si la configuration spécifiée ou par défaut n'existe pas, elle sera créée à partir des ressources du serveur.
-noGui	Si cet argument est présent il indique au serveur de démarrer en mode **noGui**, c'est-à-dire qu'aucune interface graphique ne sera affichée. Cela permet de faire fonctionner le serveur sur un système ne disposant pas d'affichage graphique (tels qu'un serveur Linux sans serveur X).

III. Configuration

Le fichier de configuration du serveur est au format XML. Il contient toutes les informations permettant de lancer le serveur suivant les préférences de l'utilisateur.

Une version par défaut de ce fichier est présente dans les ressources du serveur (dans son exécutable), et est utilisée dans le cas ou aucune autre configuration n'est trouvée.

Voici un exemple de configuration :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <name>Home</name>
  <pluginsPath>plugins</pluginsPath>
  <QtPluginsPath>./QtPlugins</QtPluginsPath>
  <filesPath>files</filesPath>
  <temporaryPath>tmp</temporaryPath>
  <cleanTemporaryPath>true</cleanTemporaryPath>
  <languagesPath>languages</languagesPath>
  <language>fr</language>
  <maxTimers>3</maxTimers>
  <database>
    <name/>
    <file>database.sqlite</file>
    <path>databases</path>
    <resource>:database</resource>
    <type>QSQLITE</type>
    <user/>
    <password/>
    <host/>
    <port/>
    <options/>
    <pragmas>
      <pragma>PRAGMA recursive_triggers=true</pragma>
      <pragma>PRAGMA synchronous=false</pragma>
    </pragmas>
  </database>
  <permissions>
    <activate>false</active>
    <default>false</default>
    <inheritance>true</inheritance>
    <ownerInheritance>true</ownerInheritance>
    <groupInheritance>true</groupInheritance>
  </permissions>
  <log>
    <level>Trace</level>
    <display>true</display>
    <file>server.log</file>
    <path>logs</path>
    <maxNbOfFile>10</maxNbOfFile>
    <maxSize>1M</maxSize>
    <expires>30</expires>
  </log>
  <ports>
    <port protocol="HTTP UPnP" transport="TCP" maxClients="100">80</port>
  </ports>
  <plugins>
    <plugin>Example/Basic</plugin>
  </plugins>
  <configurations>
    <plugin id="Example/Basic">
      <!—Configuration of the plugin -->
    </plugin>
  </configurations>
</configuration>
```

1. Divers

Noeud	Description	Valeur par défaut
name	Le nom du serveur, donné par l'utilisateur lors de son installation.	Vide
pluginsPath	Contient le nom du répertoire dans lequel se trouvent les plugins.	plugins
QtPluginsPath	Le chemin vers les plugins Qt utilisés dans les versions distribués du serveur.	./QtPlugins
filesPath	Nom du dossier dans lequel sont stockés les fichiers à partager. Il est possible de partager des fichiers en dehors de ce dossier, mais les fichiers envoyés par les clients sont stockés ici.	files
temporaryPath	Ce dossier stocke les fichiers temporaires créés lors de l'exécution du serveur. Ces fichiers sont en général supprimés lorsque le serveur ou ses plugins n'en ont plus besoin. Ce dossier est créé s'il n'existe pas.	tmp
cleanTemporaryPath	Si la valeur de ce nœud est **true**, le dossier temporaire défini par temporaryPath est vidé à chaque démarrage du serveur.	true
languagesPath	Le chemin du dossier qui contient les fichiers de traductions. Ces fichiers ont l'extension **.qm**, et sont générés via les outils de Qt.	languages
language	Le nom du fichier de langage à charger afin de traduire le serveur, sans son extension finale. Si ce fichier n'existe pas dans le dossier des langages, il sera cherché dans les resources du serveur, sous le dossier « languages ». Si ce nœud est vide ou non défini, la langue locale du système est utilisée (en, fr…).	Langue locale
maxTimers	Défini le nombre maximum de timers qu'un plugin peut activer en même temps, chaque timer requérant son propre thread. La valeur 0 peut être utilisée pour désactiver tout les timers, mais un nombre de timers trop bas peut empêcher certains plugins de fonctionner correctement.	3

2. Database

Ce nœud contient toutes les informations permettant au serveur de se connecter à la base de données. La configuration ci-dessus montre comment accéder à une base de données SQLite. Mais il est possible de configurer le serveur pour accéder à d'autres types de base de données, le serveur utilisant l'abstraction fournie par Qt.

Nœud	Description	Valeur par défaut
name	Si ce nœud est défini, la base de données est considérée comme étant basée sur un serveur (Oracle, MySQL). Dans le cas contraire, elle est basée sur un fichier (SQLite), et les nœuds **path**/**file** sont utilisés.	Vide
file	Si la base de données est basée sur un fichier, ce nœud contient le nom de ce fichier.	database.sqlite
path	Si la base de données est basée sur un fichier, représente le chemin vers le fichier défini par le nœud **file**.	databases
type	Le nom du driver Qt utilisé pour accéder à la base de données. Par exemple, le driver SQLite se nomme QSQLITE.	QSQLITE
user	Nom de l'utilisateur pour se connecter à la base de données.	Vide
password	Mot de passe pour se connecter à la base de données.	Vide
host	Adresse de la base de données.	Vide
port	Port de la base de données.	Vide
options	Options spécifiques à la base de données. Consultez la documentation Qt sur la méthode **QSqlDatabase::setConnectOptions** pour plus d'informations. Chaque option doit être dans un nœud **option**, fils du nœud **options**.	Vide
pragmas	Un pragma est une requête SQL non standard, donc spécifique à chaque type base de données, utilisée pour la configurer. Les pragmas de la configuration sont exécutés juste après la connexion à la base de données. Dans l'exemple au-dessus, le premier pragma active les triggers récursifs de SQLite, et le second désactive la synchronisation de l'écriture sur le disque, ce qui accélère nettement les requêtes.	Vide

3. Permissions

Ces nœuds permettent de configurer le système de permission du serveur.

Nœud	Description	Défaut
activate	Si ce nœud est à true, le système de permission est activé. Dans le cas contraire, tous les utilisateurs ont accès à tous les fichiers du serveur en lecture et écriture. Les fonctions d'administrations restent cependant uniquement accessibles par les administrateurs.	false
default	Ce nœud défini les droits par défaut des objets n'ayant pas de permissions hérités ou explicites.	false
inheritance	Permet de désactiver l'héritage des permissions des objets à travers l'arborescence des dossiers et des collections. Si l'héritage est désactivé, tous les objets sur lesquels des droits n'ont pas été explicitement appliqués se retrouvent avec la permission par défaut (définie par le nœud **default**).	true
ownerInheritance	Si cette option est activée, le propriétaire d'un dossier a tous les droits sur les objets qu'il contient, même s'il n'a pas de permission dessus. Pour les collections, il a tous les droits sur ses sous collections, pas sur les fichiers.	true
groupInheritance	En cas de conflit entre groupes sur une permission, c'est l'autorisation ou la restriction la plus basse dans l'arborescence des groupes qui l'emporte.	true

4. Log

Le serveur fourni un affichage des logs basique sur la sortie standard, ainsi qu'un système de niveaux, afin de n'afficher que les logs important. Les fonctionnalités avancées des logs sont gérés par les plugins. Ainsi, la seule les nœuds **level** et **display** sont utilisés directement par le serveur. Les autres nœuds sont utilisés par le plugin Log/File qui écrit les logs dans un fichier, et d'autres. La liste des nœuds ci-dessous n'est donc pas exhaustive, et dépend des plugins installés.

Nœud	Description	Valeur par défaut
level	Niveau à partir duquel les logs sont affichés. Les logs inférieurs à ce niveau ne sont pas traités. Les niveaux par ordre décroissant d'importance sont Fatal, Error, Warning, Info, Debug, et Trace.	Info
display	Si ce nœud vaut **true**, les logs seront affichés sur la sortie standard.	False
file	Nom du fichier dans lequel les logs sont écrits. Ce nœud et les suivants dépendent du plugin **Log/File**.	server.log
path	Chemin vers le fichier de log.	logs
maxNbOfFile	Nombre maximum de fichier de log. Lorsque ce nombre est dépassé, les fichiers les plus anciens sont supprimés du dossier de log.	10 fichiers
maxSize	Taille maximale d'un fichier de log en octet. Lorsque cette taille est dépassée, le fichier est renommé avec le paterne « **fichier.yyyy-MM-dd hh-mm-ss.extension** ». Il est possible de mettre les lettres **K**, **M**, ou **G**, pour indiquer que la taille est en **Kilo**, **Mega**, ou **Gigaoctets**. Par exemple 42K est l'équivalent de 43008 (42 * 1024) octets.	1M
expires	Durée maximal de conservation des fichiers de logs en jour.	30 jours

Voici la liste des niveaux de logs possible :

Niveau	Description
Fatal	Sévère erreur qui a causée un arrêt prématuré.
Error	Autre erreur de fonctionnement, ou une condition inattendue.
Warning	Utilisation d'une API dépréciée, presque une erreur, ou une situation indésirable ou inattendue, mais qui n'est pas forcément fausse.
Info	Evénement intéressant du fonctionnement (démarrage/arrêt). Doit être gardé au minimum.
Debug	Informations détaillées sur le fonctionnement du système.
Trace	Informations plus détaillées.

5. Ports

Les ports permettent au serveur d'écouter sur le réseau, et de traiter les requêtes des clients. La liste des ports se trouve dans le nœud **ports**, et la valeur de chaque nœud **port** est le numéro du port à ouvrir. Chaque port peut prendre différents attributs, qui sont détaillés dans le tableau ci-dessous. Notez qu'il est possible que le server ne puisse pas ouvrir un port, si celui-ci est déjà utilisé par un autre programme.

Attribut	Description	Valeur par défaut
transport	Nom du protocole de transport utilisé par les clients pour se connecter au serveur. La valeur peut être TCP ou UDP.	TCP
protocol	Noms des protocoles que les clients doivent utiliser pour communiquer avec le serveur via ce port. Les noms des protocoles sont séparés par des espaces. La valeur spéciale ***All*** indique que tous les protocoles peuvent être utilisés sur le port. Dans la configuration d'exemple ci-dessus, le port **80** est ouvert aux protocoles **HTTP** et **UPnP**. Une même connexion peut utiliser plusieurs protocoles. Le protocole utilisé par un client est défini pour chaque nouvelle requête, et est utilisé pour la réponse associée. Cet attribut est essentiellement utilisé par les plugins, puisque ce sont eux qui communiquent avec les clients. Le serveur lui même du serveur ne maîtrise aucun protocole.	Aucun protocole
maxClients	Nombre maximum de clients pouvant se connecter en même temps à un port.	Pas de limite

6. Plugins

Ce nœud liste les plugins à charger lors du démarrage du serveur. Les plugins sont chargés dans l'ordre de leur apparition, et sont également appelés dans cet ordre (mais cela ne doit pas avoir de conséquence sur leur fonctionnement). Un plugin doit être installé pour être chargé.

La valeur de chaque nœud **plugin** est l'identifiant du plugin à charger, c'est-à-dire le nom du ou des dossiers dans lequel il se trouve, sous le dossier défini par le nœud **pluginsPath**.

La valeur spéciale ***All*** chargera tous les plugins installés sur le serveur.

7. Plugins configurations

Le nœud **configurations** contient les configurations des plugins installés. Chaque configuration est stockée dans un nœud **plugin** dont l'attribut **id** est le nom du plugin concerné. Si la configuration d'un plugin n'est pas présente dans le nœud configurations, il n'est pas installé et ne pourra pas être chargé.

Chaque plugin possède une configuration qu'il peut structurer comme bon lui semble, à l'exception de quelques nœuds qui ont une signification particulière. Consultez la documentation sur les plugins pour plus d'informations.

IV. Fonctionnement

Cette partie, plus technique que les autres, explique le fonctionnement interne du serveur.

1. Droits

Le serveur offre une gestion complète des droits. La première chose à savoir est qu'il y a deux types d'utilisateurs, les administrateurs et les autres. Les administrateurs ont accès à la totalité des fonctionnalités du serveur, ainsi qu'à tous les fichiers qui y sont stockés. Ils ne sont donc pas soumis au système de droit.

Par défaut, le système de permissions du serveur est désactivé, afin de simplifier son utilisation pour le grand publique. Tous les utilisateurs enregistrés ont ainsi accès à la totalité des fichiers du serveur, et peuvent les modifier comme bon leur semble. Bien entendu, les fonctions d'administration sont toujours réservées aux administrateurs. Le nœud **permissions/activate** de la configuration du serveur permet de modifier ce comportement.

Lorsque le système de permissions est activé, les choses se compliquent :

Types de droits

Il existe quatre types de droits par défaut, chacun ayant conséquences variés pour les objets sur lesquels ils sont appliqués. Les plugins peuvent ajouter leurs propres droits à ce système.

Le tableau qui suit lie les permissions et leurs conséquences sur les objets :

Objets / Droits	Add	Read	Modify	Delete
Files	-	Lecture du fichier et de ses informations.	Modification du fichier et de ses informations.	Suppression.
Directories	Ajout de fichiers ou de dossiers.	Affichage du contenu et de ses informations.	Modification de ses informations.	Suppression, si les éléments qu'il contient peuvent eux aussi être supprimés.
Collections	Ajout de fichiers ou de collections.	Affichage du contenu et de ses informations.	Modification de ses informations.	Suppression, si les collections qu'elle contient peuvent elles aussi être supprimées.

Droits implicites

Il est important de noter que certains droits en impliquent d'autres. Ainsi, lorsque le droit *delete* est accordé à un utilisateur, les droits *modify* et *read* lui sont implicitement attribué. De même, si la permission *modify* est accordée, *read* l'est aussi implicitement. Pour résumer, *delete* implique *modify*, qui implique lui-même *read*. Le droit *Add* n'est pas concerné. Il est possible d'accorder ou de refuser explicitement un droit.

Héritage

Les droits sont héréditaires, c'est-à-dire que lorsqu'un droit est appliqué à un dossier, tous les objets qu'il contient et qui ne redéfinissent pas ce droit en héritent. Ainsi c'est l'autorisation ou la restriction la plus basse dans l'arborescence des dossiers qui est prise en compte. Contrairement aux dossiers pour lesquels l'héritage concerne les fichiers et les dossiers qu'ils contiennent, les droits sur les collections ne sont héréditaires que pour les collections qu'elles contiennent, les fichiers gardant les mêmes droits que dans l'arborescence des dossiers.

Lorsqu'un objet n'hérite d'aucun droit, le nœud **permissions/default** de la configuration défini le comportement à adopter. L'héritage des droits peut être désactivé par le nœud **permissions/inheritance**.

Exemple

Voici un exemple qui illustre le fonctionnement des droits :

- Directory1
  - Directory2
    - Directory3
      - File1.txt
      - File2.txt
    - Directory4
      - File3.txt
    - File4.txt
  - Directory5
  - File5.txt
  - File6.txt
- Directory6
  - File7.txt
- Directory7
  - Directory8
    - File8.txt
  - File9.txt

Les permissions accordées sont en **vert**, et celles refusées sont en **rouge**. Par défaut les droits sont refusés. Voici les permissions qui permettent de définir ces droits (nom du fichier, droit accordé ou refusé) :

```
{"Directory2", "true"}
{"Directory3", "false"}
{"File2.txt", "false"}
{"File3.txt", "false"}
{"File6.txt", "false"}
{"File6.txt", "true"}
{"Directory6", "true"}
{"Directory8", "true"}
```

Propriétaire

Les objets peuvent être associés à un propriétaire, qui est généralement l'utilisateur qui a créé l'objet. Le propriétaire d'un objet a tous les droits sur ce dernier, quelque soit les permissions qui y sont appliquées. De plus les droits des propriétaires sont héréditaires. Ainsi le propriétaire d'un dossier aura tous les droits sur les objets qu'il contient, même s'il n'est pas propriétaire de ces éléments et qu'il n'a pas de droits dessus. Ce comportement peut être modifié par la propriété **permissions/ownerInheritance** de la configuration.

Groupes et comptes

Les permissions sur des objets peuvent être attribuées à des groupes ou des comptes. Les groupes sont des listes de comptes, et un compte peut appartenir à plusieurs groupes. Les groupes peuvent contenir d'autres groupes.

Si un compte n'a pas accès à un objet, mais qu'au moins un de ses groupes y a accès, c'est la permission la plus basse dans l'arborescence qui accordée. Dans l'exemple du dessus, si l'utilisateur a le droit de modifier le **Directory2** mais qu'un groupe auquel il appartient n'a explicitement pas le droit de modifier le **Directory3**, l'utilisateur ne pourra pas modifier **File1.txt** et **File2.txt**.

Si la permission d'un groupe et d'un compte sont au même niveau dans l'arborescence, c'est celle du compte qui l'emporte. Si ce conflit oppose plusieurs groupes, c'est l'autorisation qui l'emporte.

Si **permissions/groupInheritance** est activé dans la configuration, c'est le groupe dont l'autorisation ou le refus est le plus bas dans l'arborescence des groupes qui l'emporte.

Généralisation

Lorsqu'on applique un droit, on n'est pas obligé de préciser à qui se droit est destiné. Il sera donc appliqué à tous le monde. De même si on ne précise pas d'objet sur lequel un droit s'applique, cela revient à désigner la racine du serveur. Enfin si on ne précise pas le nom du droit qui lie un objet à un accesseur, cela désigne tous les droits.

Ces permissions génériques ne sont pas prioritaires sur les droits spécifiques. Donc si on autorise tout le monde à regarder une vidéo, mais qu'en même temps on interdit quelqu'un de la voir, ça revient à dire *tout le monde sauf cette personne*.