Serveur

* **Introduction**

Streamit server is a program running as a background task on the computer hosting files desired to be shared by the user. The server job is to distribute those files to clients according to their "clearance".

The server is made to be as easy as possible to use, the user don't even have to know it's running on its computer because the server is managed by an administration interface which is a separated program.

The server is made of 3 principals parts, the **database** which contains persistent data, the  **plugins** which represents the server mind and define how it interacts with users, and the **server** itself which

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’orchestrate everything.

Database plugins are explained in separated documentions, this one is much more about the server core.

The server is developed with the QT Framework, 4.6.2 version and works on every operating systems tolerated by QT (Mainly Windows, Linux and Mac OSX).

* **Options**

|  |  |
| --- | --- |
| **Options** | **Description** |
| **-c *configuration.xml*** | **–c**, is followed by the complet path(relativ or absolute) to the server configuration file. This file contains every data needed by the server to startup with user preferences. If this option isn't mentioned the default configuration path is « **configurations/Configuration.xml** ». If the mentioned configuration doesn't exist it will be make using the server resources. |
| **-noGui** | Tells the server to start without graphical interface, not even the splash screen will be displayed. Allows to run the server on a system which doesn't use graphical display (a linux server without X server). |

* **Configuration**

It is an XML configuration file containing every data needed by the server to be run with user specific parameters.

A default file is given in the server resources and is used when no configuration is found.

Example:

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<name>Home</name>

<pluginsPath>Example</pluginsPath>

<filesPath>Example</filesPath>

<temporaryPath>tmp</temporaryPath>

<cleanTemporaryPath>true</cleanTemporaryPath>

<languagesPath>languages</languagesPath>

<language>fr</language>

<maxTimers>3</maxTimers>

<splashScreen>:splashScreen</splashScreen>

<database>

<name/>

<file>server.db</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>

</permissions>

<log>

<level>Trace</level>

<display>true</display>

<file>server.log</file>

<path>logs</path>

<maxNbOfFile>10</maxNbOfFile>

<maxSize>1M</maxSize>

<validityPeriod>30</validityPeriod>

</log>

<ports>

<port protocol="HTTP SiTP" transport="TCP" maxClients="10">4242</port>

<port>80</port>

</ports>

<plugins>

<plugin>Example</plugin>

<plugin>SiTP</plugin>

<plugin>Client/Web</plugin>

</plugins>

</configuration>

* **Divers**

|  |  |  |
| --- | --- | --- |
| **Node** | **Description** | **Default value** |
| **name** | The server name given by the user during install. | Empty |
| **pluginsPath** | The repository name containing plugins folders. | plugins |
| **filesPath** | The folder name contianing files to be shared. It is possible to share files outside the folder but files sent by the client are gathered here. | 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** | If the node value is **true**, The temporary folder defined by temporaryPath is cleaned after each server startup. | False |
| **languagesPath** | The folder path containing traductions. The files are hightlighten by the extension "**.qm"**, they are generated by QT tools. | languages |
| **language** | The language file name to load for the server traduction without its final extention. If this file doesn't exist in the languages folder, it will be looked after in the server resources, in the folder« languages ». If this node isn't filled up the local language is used. | local language |
| **maxTimers** | Gives the maximum set of timers which can be simultaneously activated by a plugin, each timers requires its own thread. The value 0 can be set for desactivating every timers, but plugins can be troubled by having few timers. | 3 |
| **splashScreen** | The path of the server splash screen. It is a picture that shows up indicating the user the server is loading, if this node isn't filled up or the server is in noGui mode, no screen splash <ill be displayed. | Empty |

* **Database**

This node contains data needed by the server to connect to the database. The configuration above shows how to access an SQLite database. But it is still possible to configure the server for other databases thanks to the abstration allowed by QT.

|  |  |  |
| --- | --- | --- |
| **Node** | **Description** | **Default value** |
| **name** | If not defined, the database is considered as a server based database(Oracle, MySQL), if it is, the database is considered as file based(SQLite), and nodes **path**/**file** are used. | Empty |
| **file** | If the database is file based, this node contains this name file. | server.db |
| **path** | If the database is file based, this node contains the file path defined by the node **file**. | databases |
| **type** | The QT driver name for accesing the database, by example the driver for SQLite is named QSQLITE. | QSQLITE |
| **user** | User name to access the database. | Empty |
| **password** | User password to access the database. | Empty |
| **host** | Database adresse. | Empty |
| **port** | Database port. | Empty |
| **options** | Specific database options. Check out the QT documentation regarding the method **QSqlDatabase::setConnectOptions.** Each option has to be in a node named **option**, son of node **options**. | Empty |
| **pragmas** | It is a non standard SQL query, specific to each database, used to configured. Configuration pragmas are run right after the database connexion. In the example above, the first pragma activates SQLite recursive triggers, the second desactivates writting synchronisation on the disk. | Empty |

* **Permissions**

|  |  |  |
| --- | --- | --- |
| **Node** | **Description** | **Default value** |
| **activate** | If true, the permissions system is activated. If false every user has acces to every file on the server set to reading/writting mode. Administration functions are usable only by administrator, no matter what. | False |
| **default** | Set the objects default rights which doesn't have inherited or explicit permission. | False |
| **inheritance** | Desactivates object inheritance permissions through the folders and collections arboreous. If inheritance is desactivated, every object which doens't have explicit rights set are tied to the default permission(**default** node). | True |
| **ownherInheritance** | 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 |

* **Log**

The server allows basic log display on the stndard output as a level system to display only important log. Advanced functionalities are managed by plugins. So, only the nodes **level** and **display** are directly used by the server. The other nodes are used y the plugin LogFile (which rights logs on a file) and others. The nodes list below varies with installed plugins.

|  |  |  |
| --- | --- | --- |
| **Node** | **Description** | **Default value** |
| **level** | Minimum log level to display logs. Logs under this level are not traited. (list: Fatal, Error, Warning, Info, Debug, and Trace). | Info |
| **display** | If **true**, logs will be displayed on the standard output. | False |
| **file** | File name on which logs are written. This node and the following depend on **LogFile plugin**. | server.log |
| **path** | Path to the log file. | Current folder |
| **maxNbOfFile** | Maximum number of log files. When this number is reached, the older files are removed from the log folder. | 128 files |
| **maxSize** | Maximum size of a log file in octet. When this size is reached, the file is renamed (**file.yyyy-MM-dd hh-mm-ss.extension**).  It is possible to put letters **K**, **M** and **G** to set size in **Kilo**, **Mega**, or **Gigaoctets**. By example 42K is eaquel to 43008 (42 \* 1024) octets. | 1024 octets |
| **validityPeriod** | Maximum log file keeping time in days. | 30 days |

|  |  |
| --- | --- |
| **Niveau** | **Description** |
| **Fatal** | Severe error which stops not as excepted. |
| **Error** | Another other or an unexpected condition. |
| **Warning** | Use of a deprecated API, close to an error but isn't necessarily wrong. |
| **Info** | Interesting event on on it works(start/stop). Must be kept by the minimum. |
| **Debug** | Detailled data on how the system works. |
| **Trace** | Informations plus détaillées. |

* **Ports**

Ports allow the server to listen the network and to process client queries.

Ports list is in the node **ports**, the value of a port is the number to open. Each port can take severals attributes (below). If a port is used by another program the server may not be able to open it.

|  |  |  |
| --- | --- | --- |
| **Attribut** | **Description** | **Default value** |
| **transport** | Transport protocol name used by the clients for connecting to the server, it can be TCP or UDP. | TCP |
| **protocol** | Protocol names the clients must used to communicate with the server through this port. Protocol names are separated by spaces.  ***All*** means every protocol can be used on this port. In the example above, port **4242** is open to protocol **HTTP** and **SiTP**. The same connexion can use severals protocols. The protocol is defined to each new queries and is used for the associated answer. This attribute is mainly used by the plugins because they are the ones communicating with the clients. The server itself knows no protocol. | No protocol |
| **maxClients** | Maximum number of clients that can connect to the same port. | No limit |

* **Plugins**

This node lists plugins to load when the server starts. They are loaded as they appear.

The value of each **plugin** node is the plugin ID to be load, it means the name of the folder(s) in which it isdefined by the node **pluginsPath**.

The value **All** will load every plugins on this folder.

* **How it works**

Explains the server internal way to work.

* **Rights**

There is two kind of users, the administror and the others. Administrator can access to all the functionalities as well as all the files. They aren't tied to the rights logic.

Bt default the persmission system is desactivated to sumplify its utilisation for every users. Every users recored have a total access to hte server files and cam modify as they want. When the permission system is activated, things are getting harder.

***Rights***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objets / rights** | **Add** | **Read** | **Modify** | **Delete** |
| **Files** | - | Dowload, streaming, display information. | File modification and its information. | Suppression. |
| **Directories** | Files/folders adding. | Display the content and its information. | Its information modification. | Suppression, if the elements it contains can also be removed. |
| **Collections** | Add files or collections | Display the content and its information | Its information modification. | Suppression, if the collections it contains can also be removed. |

***Implicit rights***

Some rights imply others. So when the right *delete* is given to a user, modify and read are imply implicitely. If *modify* **is given,** *read* is also there.

***Inritance***

Rights are hereditary it means when a right is aplpied to a folder, every object in it are tied to the parent rights if they don't redefined a right for them.

When an object inherites from no right, the node **permissions/default** from the configuration defines the behavior. Inheritance can be desactivatedby the ndoe **permissions/inheritance**.

***Example***

|  |
| --- |
| * Dossier1 * Dossier2 * Dossier3 * Fichier1.txt * Fichier2.txt * Dossier4 * Fichier3.txt * Fichier4.txt * Dossier5 * Fichier5.txt * Fichier6.txt * Dossier6 * Fichier7.txt * Dossier7 * Dossier8 * Fichier8.txt * Fichier9.txt |

Permissions granted are in **green**, permissions refused are red.

|  |
| --- |
| {"Dossier2", "true"}  {"Dossier3", "false"}  {"Fichier2.txt", "false"}  {"Fichier3.txt", "false"}  {"Fichier6.txt", "false"}  {"Fichier6.txt", "true"}  {"Dossier6", "true"}  {"Dossier8", "true"} |

***Owner***

Objects can be tied to owners which are most of the time the object creator. The owner has every right on it no matter permissions set on it. Owner rights are heriditary. An owner folder has every right on what is i inside, the behavior can be changed by the property **permissions/ownerInheritance from the configuration**.

***Groups and accounts***

Permission on objects can be attributed to groups or accounts. Groups are lists of accounts, and an account can contains several groups.

If an account doesn't have access to an object but at least one of its group has, the lower persission is used. In the exemple above, if the user can modify **Dossier2**but a group which it is part of isn't granted to modify **Dossier3**, the user won't be able to modify **Fichier1.txt** and **Fichier2.txt**.

An account permission on the same level as a group permission in the arboreous, the account will win.

***Generalization***

When setting a right, setting to whom the right is tied with isn't mandatory. It will be applied on everyone. As if the object on whitch the right is applied isn't given, it is like the server trunk. If no right name is set to an object tied to an accessor, it measn every rights.

Thoses generics permissions don't have priority on specific rights. So if everyone is granted to watch a video but in the same time we don't give grant to someone it means everyone except this person.