The Matrex Server

Table of Contents

Introduction

Before version 2.0, Matrex has been a standalone application. Matrex loaded and saved projects on the PC in which it was running, without any need of using the network.

From version 2.0, I wanted the possibility to **share** a project among different users, so that they could work on it together and simultaneously . To do this I have chosen a client/server architecture.

In the same time I did not want to **force** users to work in a client/server fashion.

So, from version 2.0, the user can:

- continue to work exactly as before, with projects located on his own PC and without connecting to any server (**standalone**)
- working with projects located on **servers** *and*, if he wants, with projects on his own PC. In this case we say that Matrex Desktop becomes a **client** in the Matrex client/server system.

Clients can connect to servers only inside a **LAN** (local area network). It is therefore not possible for a client to connect to a server on the Internet. This is because:

- Matrex 2.0 uses heavily **asynchronous messaging** to keep clients and servers synchronized. This does not work well on the internet.
- For a system that is built to calculate big amounts of data, **time is important**. And the time must be used to calculate the data, not to send data through the network.

Remote projects

From version 2.0 Matrex allows to connect to a Matrex Server, create/open a project located in a server and work with this project (display/add/update/remove items in the server).

We call the projects located in a server **remote projects**.

But what is exactly the meaning of remote project?

It means that it is the Matrex Server that:

- loads/saves the **project's files**
- **calculates** functions
- **updates** the content of matrices, presentations, charts
- **fires** the timers events

In this modality the only duty of the Matrex Desktop application is to:

- **display** the content of matrices, presentations, charts
- keep the visual representation of the project **updated** with the project content in the server
- allow **adding, editing, removing** items in the remote project.

It is easy to see that, working with remote projects, the activity of the Matrix Desktop is only graphical and not CPU intensive. Better said, Matrex acts as a remote GUI for the server.

Reasons to work with remote projects

The main reasons to work with remote projects are:

- A group of users wants to work **together** and possibly **simultaneously** on a project.
- A user wants to **delegate** the project calculation to a server and therefore reduce the use of resources on his own PC. He can potentially run Matrex Desktop on a netbook and leave the calculations to a more powerful PC.

RMI Network Protocol

A Matrex Desktop connects and interacts with a Matrex Server using the **RMI** network protocol, which is currently used for J2EE systems (http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp).

RMI is not a new protocol but is widely used and well tested.

To use RMI, the server must register itself in the **RMI registry**, which is an application included in the JRE (Java Runtime Environment).

The Matrex Desktop client will then:

- connect to the **registry** to find out which servers are available
- **choose** one of the servers
- **connect** to the server and start to work with it.

More than one server can run in the same computer at the same time. The only constraint is that they need to be registered in the RMI registry with **different names**.

You can have more information about this, looking at the Matrex Desktop help file (the RemoteProjects.html document), included in the Matrex setup.

Requirements

To run a Matrex Server application you need to have installed a **JRE** with version >= 5 (Java 5, Java 6...).

Install Matrex Server

After you installed Matrex 2.0, you will see that in its directory there is a file called matrex_2_0_server.jar.

Execute this jar file with java in the computer where you want to install the Matrex Server. This will start an IzPack (http://izpack.org/) installer, the same used for the installation of the Matrex Desktop application. Follow the instructions in the installer to install the server.

Start Matrex Server

Before you start a Matrex Server you need to have the RMI registry running on the same computer.

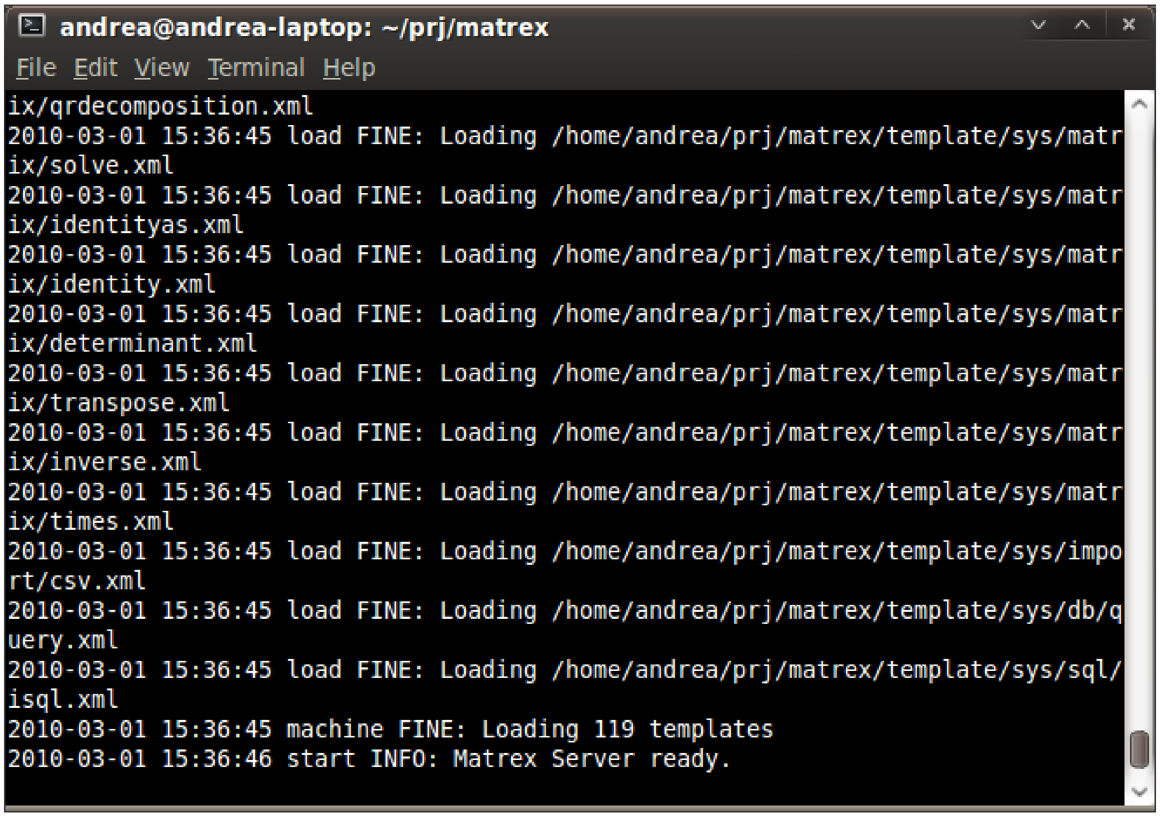
The RMI registry can be started using the **rmiregistry** command that is included with the JRE.

To start a matrex server you just run:

java -jar matrex_server.jar

from the top Matrex directory.

Matrex Server is a console application, which should look like this:


andrea@andrea-laptop: ~/prj/matrex
File Edit View Terminal Help

```
ix/qrdecomposition.xml
2010-03-01 15:36:45 load FINE: Loading /home/andrea/prj/matrex/template/sys/matr
ix/solve.xml
2010-03-01 15:36:45 load FINE: Loading /home/andrea/prj/matrex/template/sys/matr
ix/identityas.xml
2010-03-01 15:36:45 load FINE: Loading /home/andrea/prj/matrex/template/sys/matr
ix/identity.xml
2010-03-01 15:36:45 load FINE: Loading /home/andrea/prj/matrex/template/sys/matr
ix/determinant.xml
2010-03-01 15:36:45 load FINE: Loading /home/andrea/prj/matrex/template/sys/matr
ix/transpose.xml
2010-03-01 15:36:45 load FINE: Loading /home/andrea/prj/matrex/template/sys/matr
ix/inverse.xml
2010-03-01 15:36:45 load FINE: Loading /home/andrea/prj/matrex/template/sys/matr
ix/times.xml
2010-03-01 15:36:45 load FINE: Loading /home/andrea/prj/matrex/template/sys/impo
rt/csv.xml
2010-03-01 15:36:45 load FINE: Loading /home/andrea/prj/matrex/template/sys/db/q
uery.xml
2010-03-01 15:36:45 load FINE: Loading /home/andrea/prj/matrex/template/sys/sql/
isql.xml
2010-03-01 15:36:45 machine FINE: Loading 119 templates
2010-03-01 15:36:46 start INFO: Matrex Server ready.
```

Configuration

The configuration files of the Matrex Server are by default in the **srvconfig** directory.

Like the Matrex Desktop, the Matrex Server depends by the **srvconfig/main.properties** file, which maps all the other configuration files, templates and scripts used by the server.

One if these files is **server.properties**. which contains the following properties:

- **servername**: the id that identifies the server when a client connects to it (it is actually the name of the server in the RMI registry).

- **projectsdir**: the directory containing all projects in the server (*srvprj* by default).
- **codebase**: needed for the RMI protocol, is the path where the server classes are located, by default matrex_server.jar

Accounts

The accounts file, **account.xml**, contains all the **accounts** of the users who can connect to the server.

The account file has the following structure:

```
<accounts>
        <account userid="<userId>" password="<password>"
encrypted="<encrypted>" />
        …
        ...
</accounts>
```

which is an XML list of all available accounts. When the server is installed, the accounts.xml file contains only the **guest** account with password **guest**.

To add an account so that a new user can login to the server, just add an <account> row, which contains the userid of the user and its password (in clear). The encrypted value must be **false**.

For example:

```
<accounts>
        …
        …
        <account userId="test" password="test2010" encrypted="false"/>
</accounts>
```

When the server starts, it rewrites the file so that all the passwords are encrypted.

It is a good practice, when some users accounts have been added, to remove the original guest account.

Troubleshooting

If you get this error:

```
java.rmi.ConnectException: Connection refused to host: andrea-laptop; nested exception
is:
        java.net.ConnectException: Connection refused
        at sun.rmi.transport.tcp.TCPEndpoint.newSocket(TCPEndpoint.java:601)
        at sun.rmi.transport.tcp.TCPChannel.createConnection(TCPChannel.java:198)
        at sun.rmi.transport.tcp.TCPChannel.newConnection(TCPChannel.java:184)
        at sun.rmi.server.UnicastRef.newCall(UnicastRef.java:322)
        at sun.rmi.registry.RegistryImpl_Stub.rebind(Unknown Source)
        at java.rmi.Naming.rebind(Naming.java:160)
```

it just means that you forgot to run the **RMI registry** (rmiregistry).

If you get this error:

```
java.security.AccessControlException: access denied (java.net.SocketPermission andrea-
laptop resolve)
        at
java.security.AccessControlContext.checkPermission(AccessControlContext.java:323)
        at java.security.AccessController.checkPermission(AccessController.java:546)
        at java.lang.SecurityManager.checkPermission(SecurityManager.java:532)
        at java.lang.SecurityManager.checkConnect(SecurityManager.java:1031)
        at java.net.InetAddress.getAllByName0
```

the server is probably not finding the file *policy.all* (in the top server directory), which contains the **security policies** needed for RMI to work.

The *codebase* property in the **server.properties** file is used to build the value for the system property *java.rmi.server.codebase*, which points to the path of the jar file containing the server classes and is needed for the RMI protocol to work.

The reason why we don't use directly the *java.rmi.server.codebase* system property is because this system property does not cope well with relative paths.

But if something goes wrong you can set instead the *java.rmi.server.codebase* system property directly.

To check the content and behaviour of a server if the server is having problems, check **Matrex Server JMX** document.