Magentix2 Guide

INDEX

Introduction

In multi-agent system (MAS) field one of the goals is to build systems capable of making decisions in an autonomous and flexible way. Moreover, these systems must cooperate with other systems inside a "*society*". The term "society" needs to meet requirements such as: distribution, constant evolution, flexibility to allow members to enter or exit in the society, appropriate management of the organizational structure that defines the society, multi-device agent execution including devices with limited resources, and so on. On the other hand is necessary to build systems that enable complete interoperability for messaging middleware and also guarantee a very high efficiency. Additionally is necessary to provide developers methods, tools and appropiated architectures which support all the requirements in order to develop this kind of systems.
The main goal of the Magentix2 project is to develop a new open multi-agent system platform that is suitable for the development of systems with all these characteristics. The features of the Magentix2 platform can be structured in two basic levels, as follows:

- Low-level architecture: networking protocol and semantics required for messaging middleware services in Magentix2 are defined using the open standard *Advanced Message Queuing Protocol* (AMQP)[1] and specifically using *Apache Qpid*[2] Open Source implementation of this protocol. Magentix2 is FIPA-ACL compliant and is able to communicate with any platform that understands FIPA-ACL.
- High-level architecture: This architecture has been called THOMAS (MeTHods, Techniques and Tools for Open Multi-Agent Systems). The proposal tries to raise a total integration of two promising technologies, that is, multi-agent systems and service-oriented computing. In THOMAS architecture agents can offer and invoke services in a transparent way to other agents or entities, as well as external entities can interact with agents through the use of the offered services. This architecture is the first step in order to obtain true deployed virtual organizations.

Magentix2 is distributed under the terms of the LGPL (Lesser General Public license) http://www.gnu.org/copyleft/lesser.html

This document briefly describes how to install, develop and run some examples of the Magentix2 platform.

---

[1] Networking protocol and semantics required for messaging middleware services are defined using the open standard *Advanced Message Queuing Protocol* (AMQP).
http://jira.amqp.org/confluence/display/AMQP/Advanced+Message+Queuing+Protocol

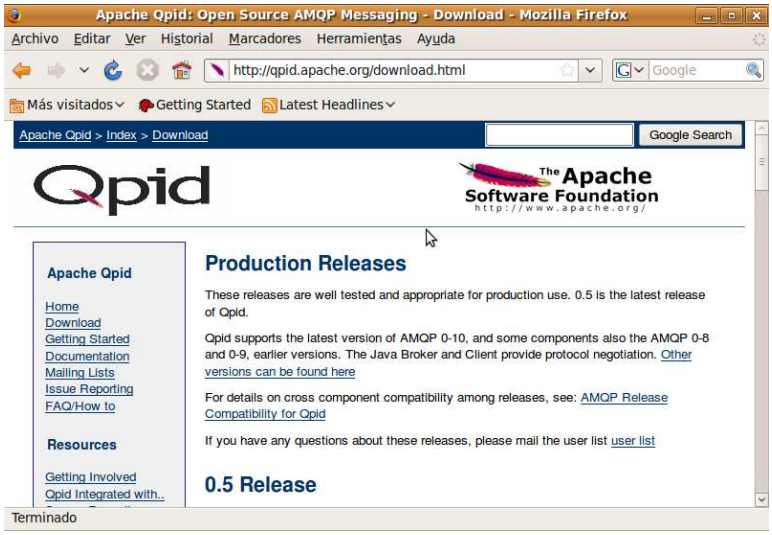
[2] *Apache Qpid* implements the latest AMQP specification and provides two AMQP messaging brokers implemented in C++ and Java and Client APIs in C++, Java, JMS, Ruby, Python, and C#.
http://qpid.apache.org/

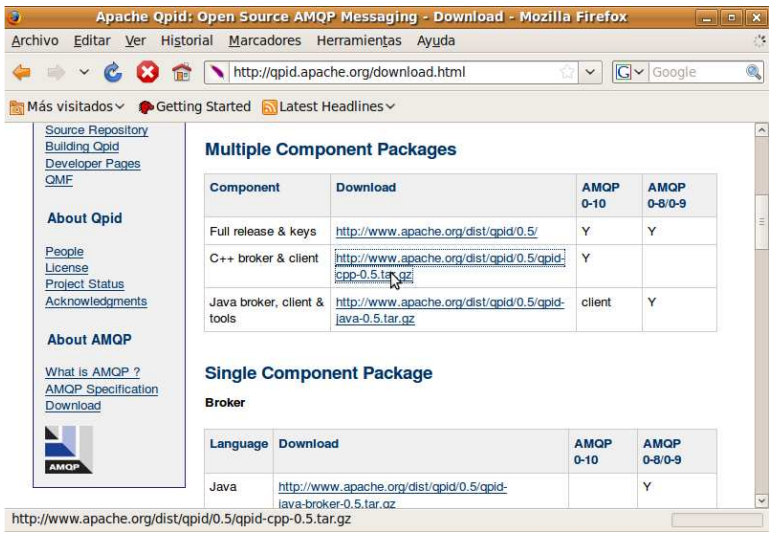
Install Qpid C++ Broker

This section[3] describes how to install **Qpid/C++**[4] messaging broker from the source distribution http://qpid.apache.org/ on Linux/Unix operating system, particularised to **Ubuntu 9.04 32-bit** version http://www.ubuntu.com/. In particular, this section explains how to install the some libraries (pre-requisites) required to install Qpid/C++ and also the modifications on the source code of Qpid/C++ required to compile and install the broker [5].

Download QPID

http://qpid.apache.org/download.html





3 This section is based on the INSTALL file included on the Qpid distribution

4 Qpid has been built using the GNU C++ compiler: gcc <http://gcc.gnu.org/> (3.4.6).

5 A patch has been developed to facilitate these modifications.

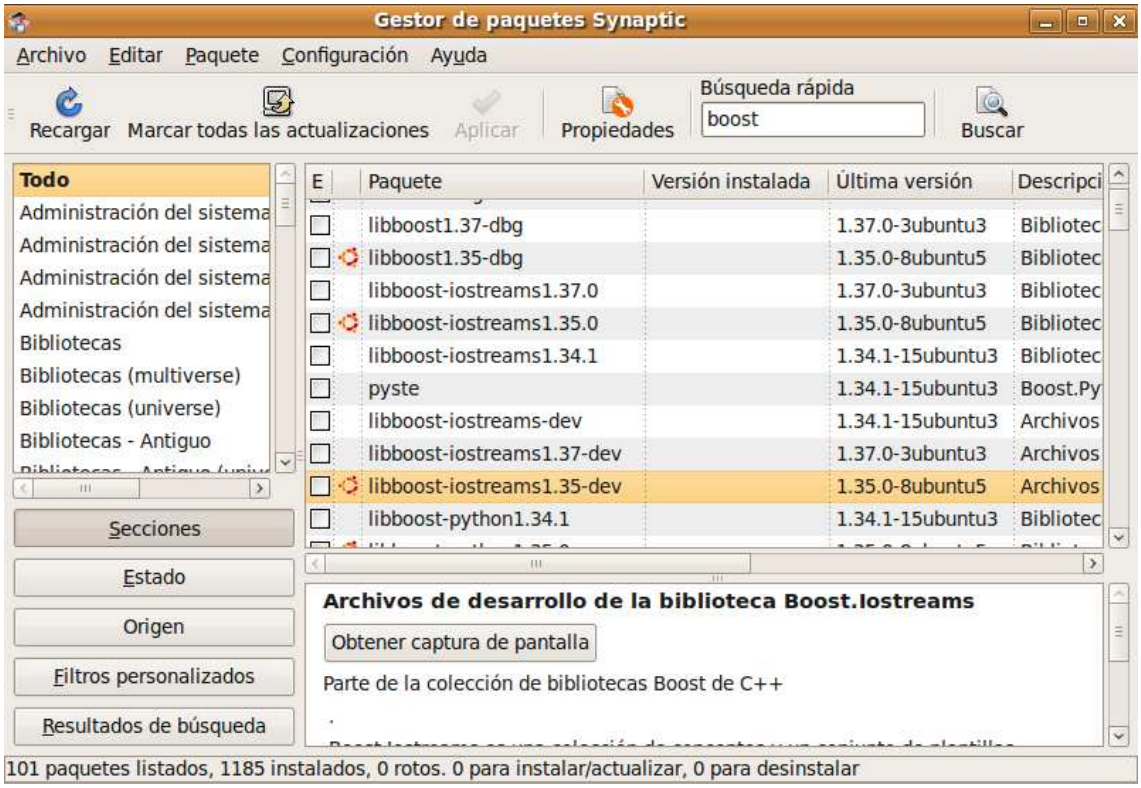
Pre-Requisites

The following libraries must be installed to build a source distribution of Qpid:

- **libboost-iostreams 1.35-dev**: <http://www.boost.org> (1.35)[6]
- **e2fsprogs**: <http://e2fsprogs.sourceforge.net/> (1.39)
- **pkgconfig**: <http://pkgconfig.freedesktop.org/wiki/> (0.21)
- **uuid 1.2-1.41.4**
- **ruby 4.2**
- **ruby 1.8**

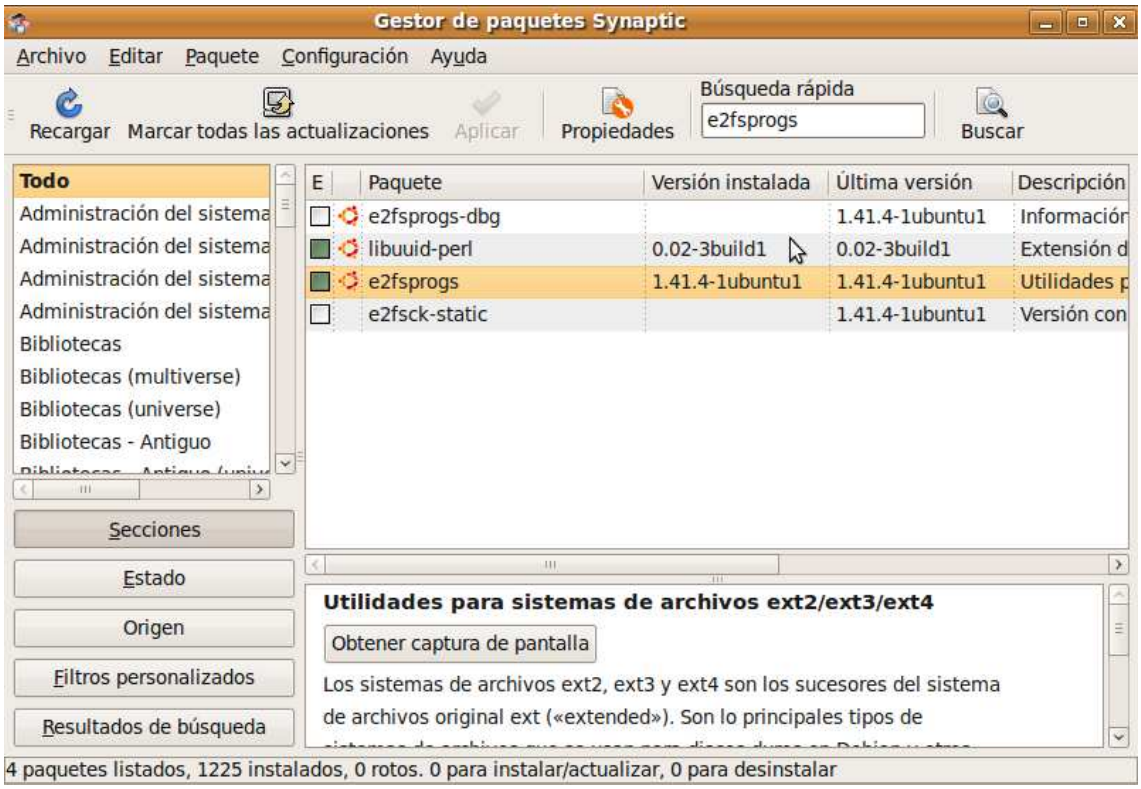
On *Ubuntu* operating system these packages can be installed using the distribution's package management tool *Synaptic*.

libboost-iostreams 1.35-dev

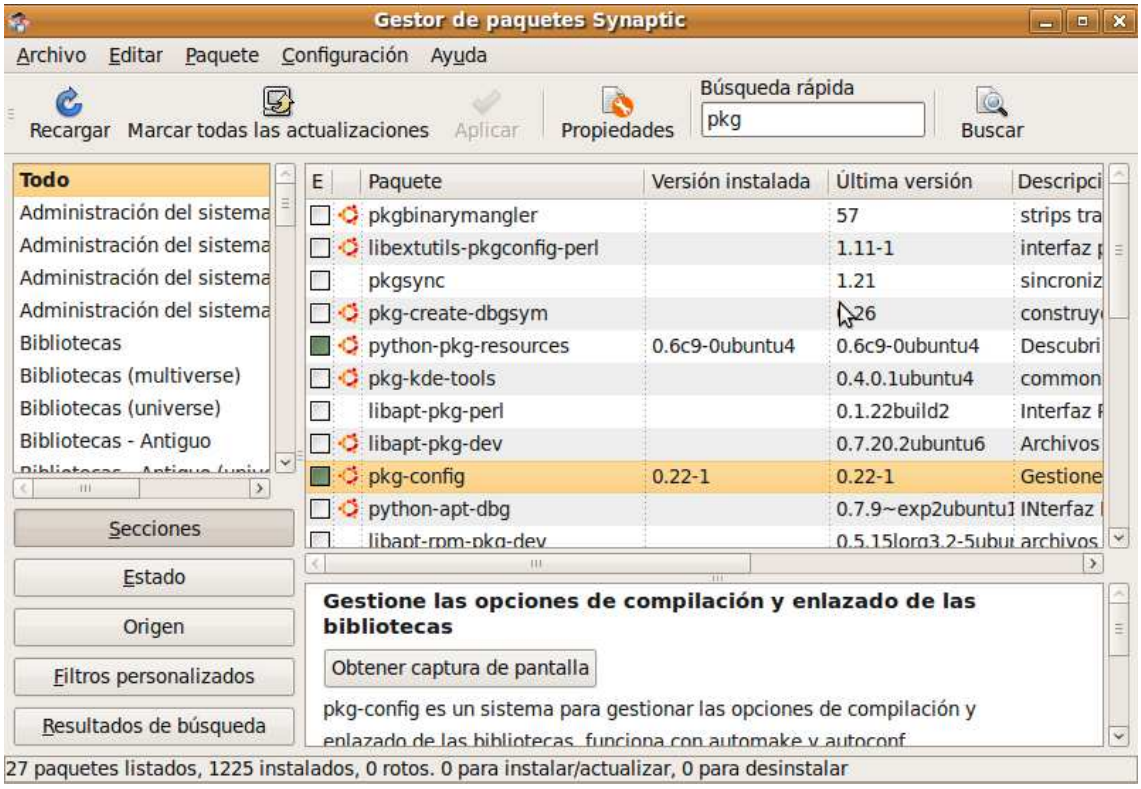


[6] Earlier versions of boost e.g. 1.33 also work and there is a patch to get 1.32 working in the svn tree though that is only recommended as a last resort.

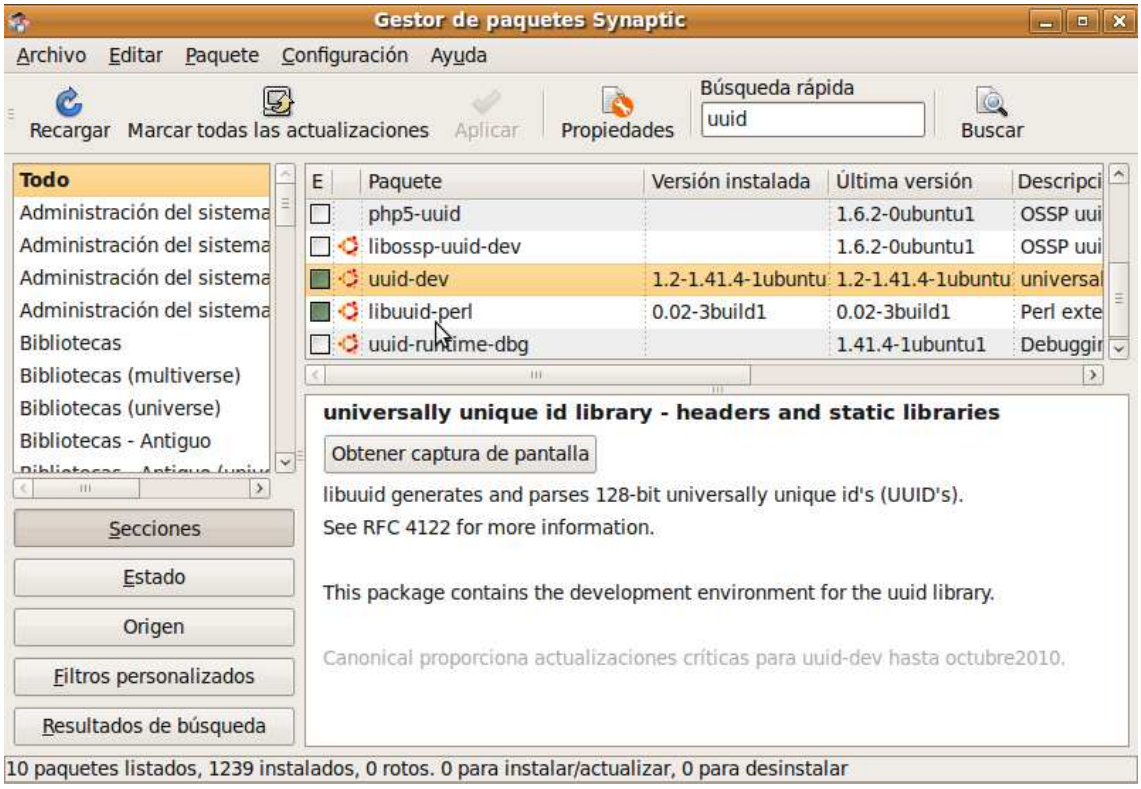
e2fsprogs 1.41.4



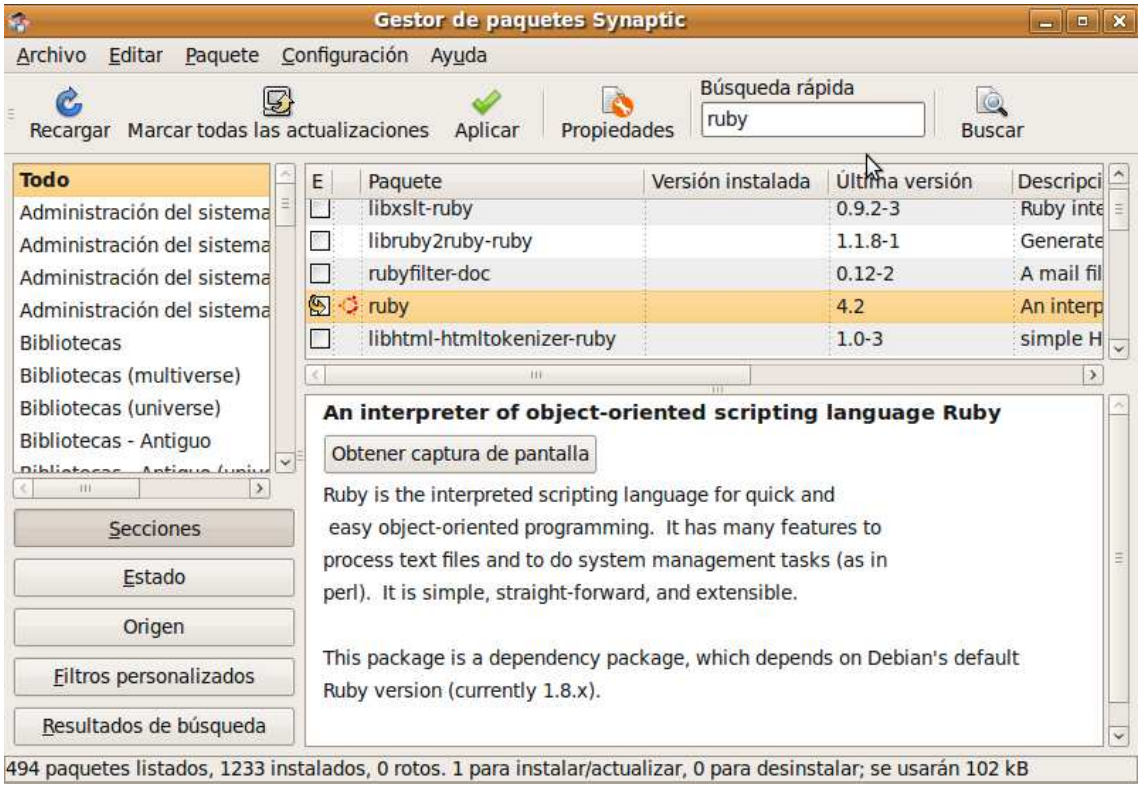
pkg-config 022-1



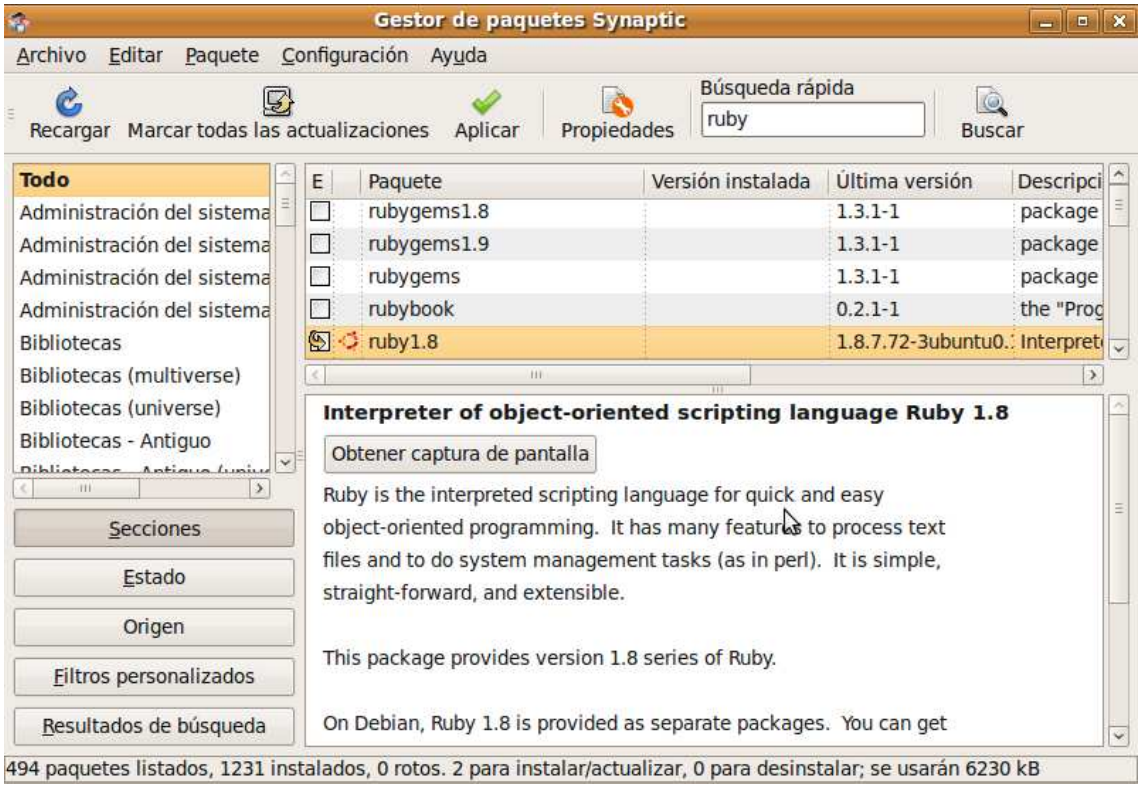
uuid 1.2-1.41.4



ruby 4.2

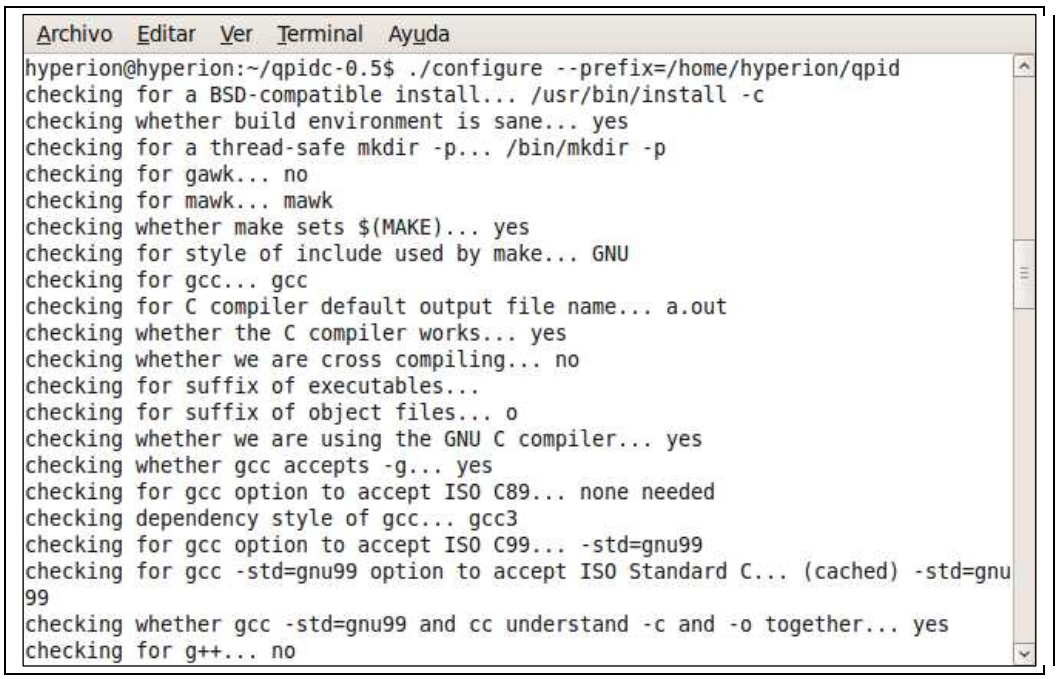


ruby 1.8 (required for install ruby 4.2)



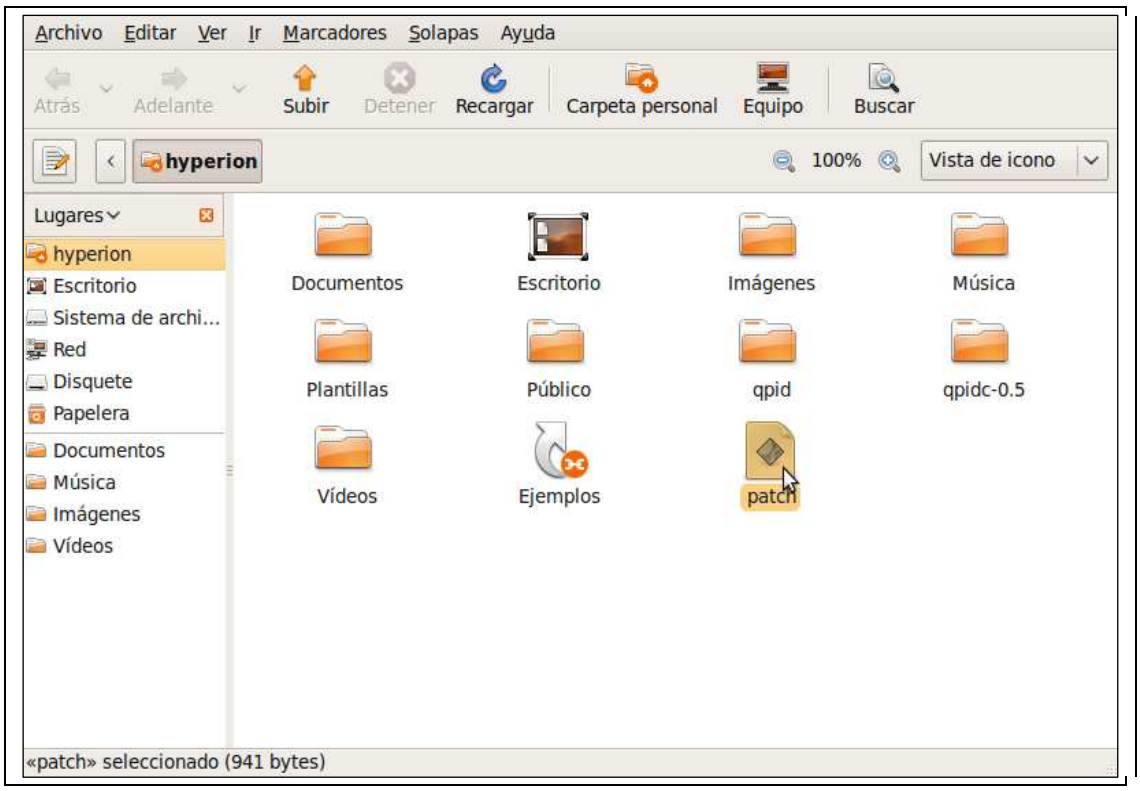
Configure QPID

./configure –prefix=/home/hyperion/qpid

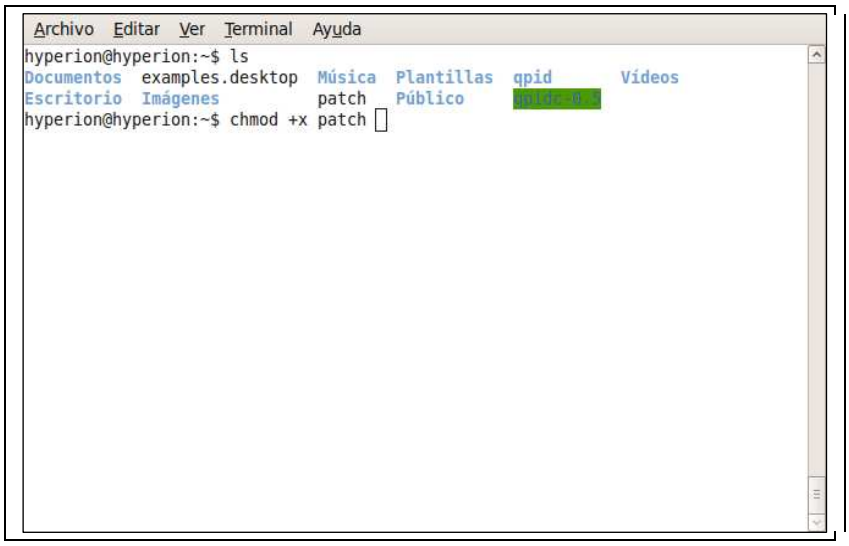


Modify Qpid Source Code (patch)

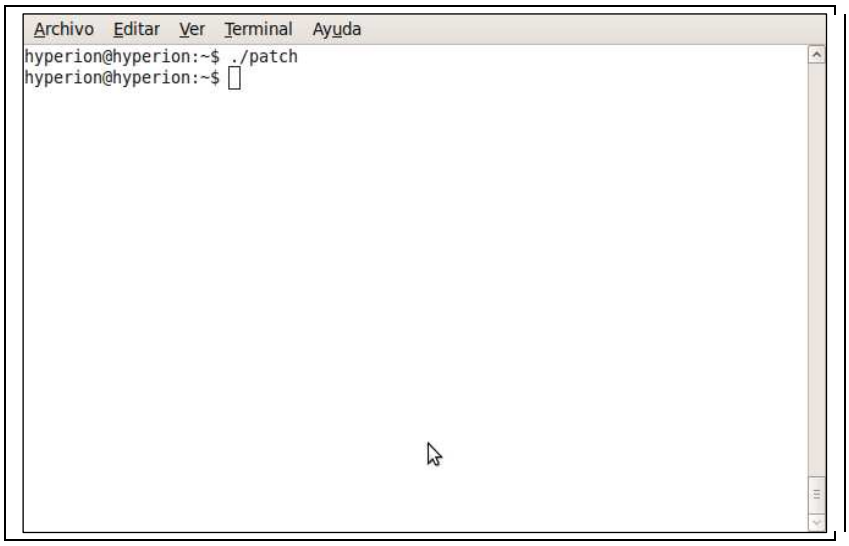
In order to compile and install the Qpid C++ broker in Ubuntu 9.04 some modifications of the source code are required. A detailed explanation of these modifications is included in the *Apendix I* of this document. A patch has been developed in order to facilitate this process and is included in the Magentix2 distribution.



Changes the permission of the PATCH file to run it as a program:
chmod +x patch

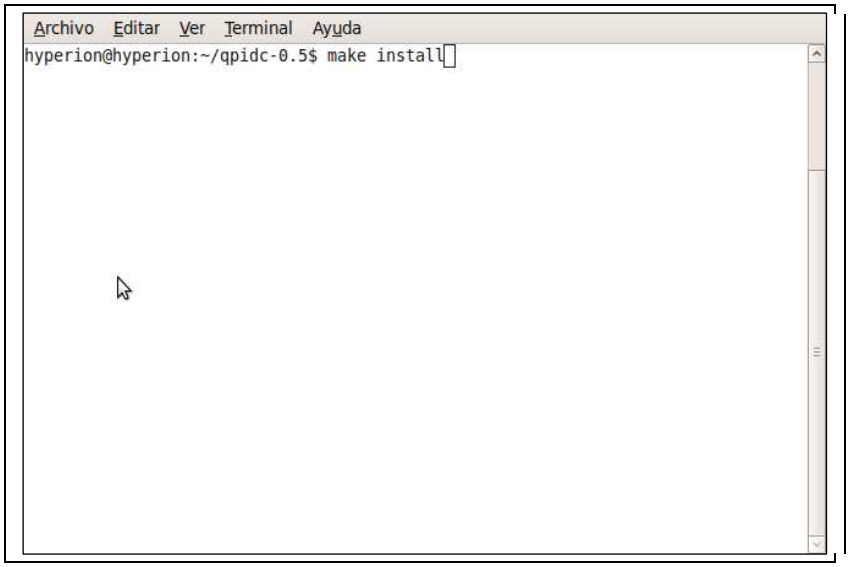


Run the patch file:
./patch



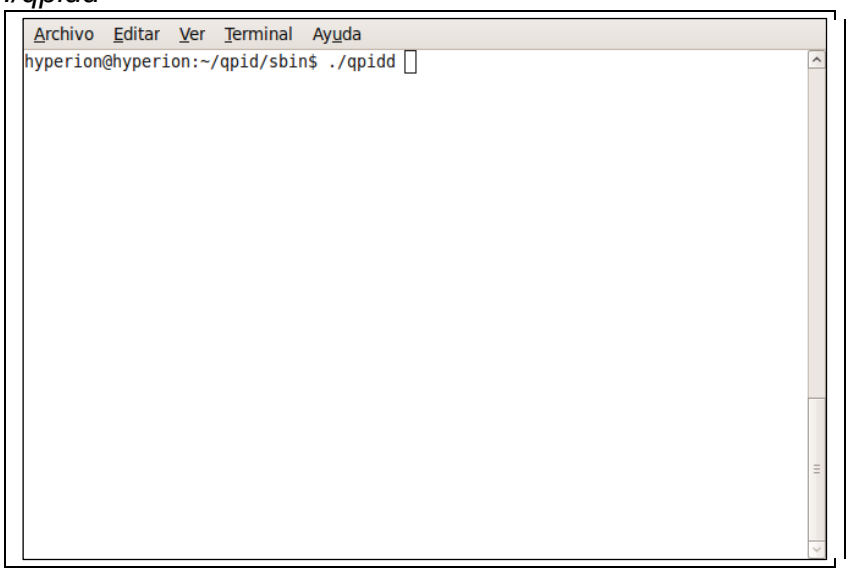
Install Qpid

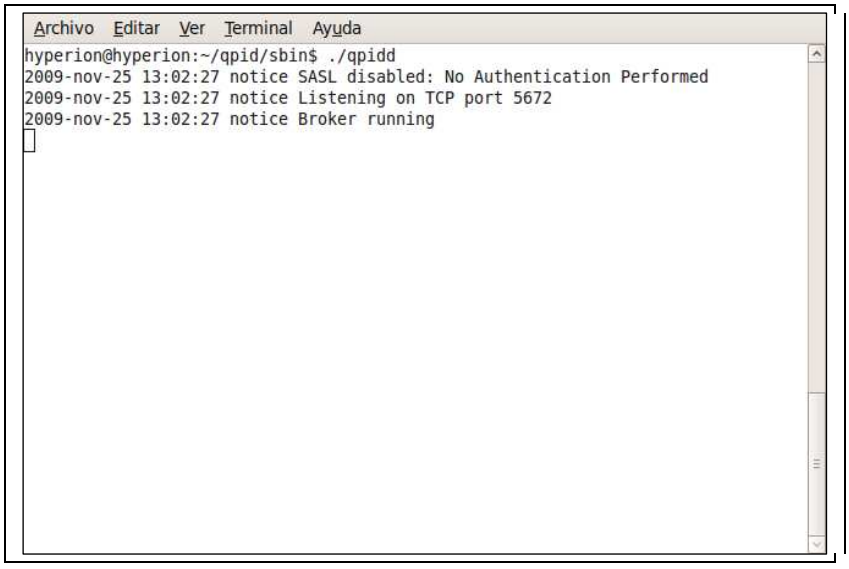
Make install



Start Qpid Broker

./qpidd

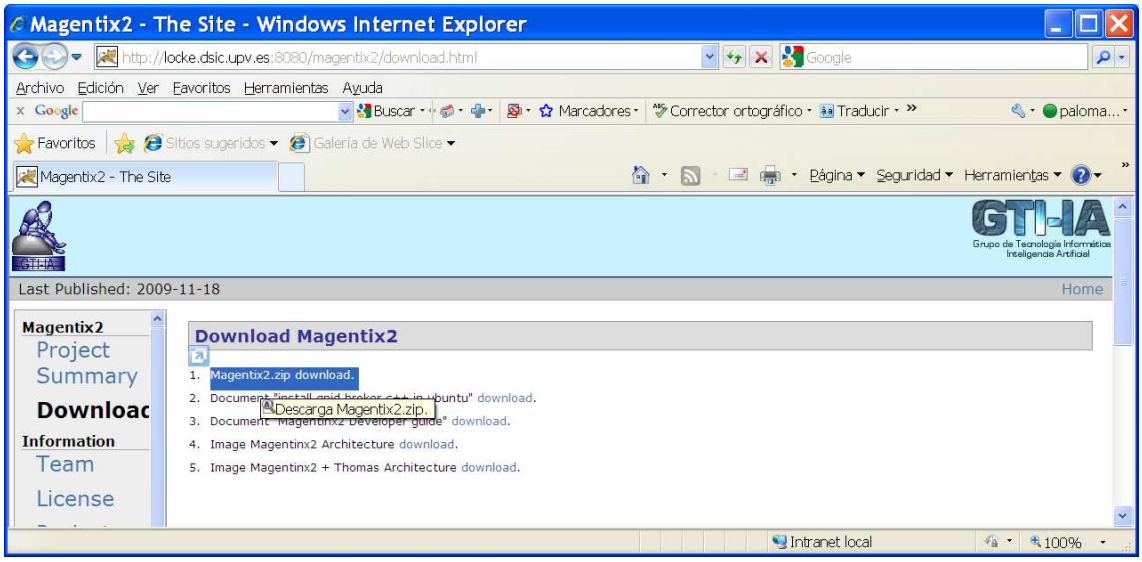




Magentix2

Download Magentix 2

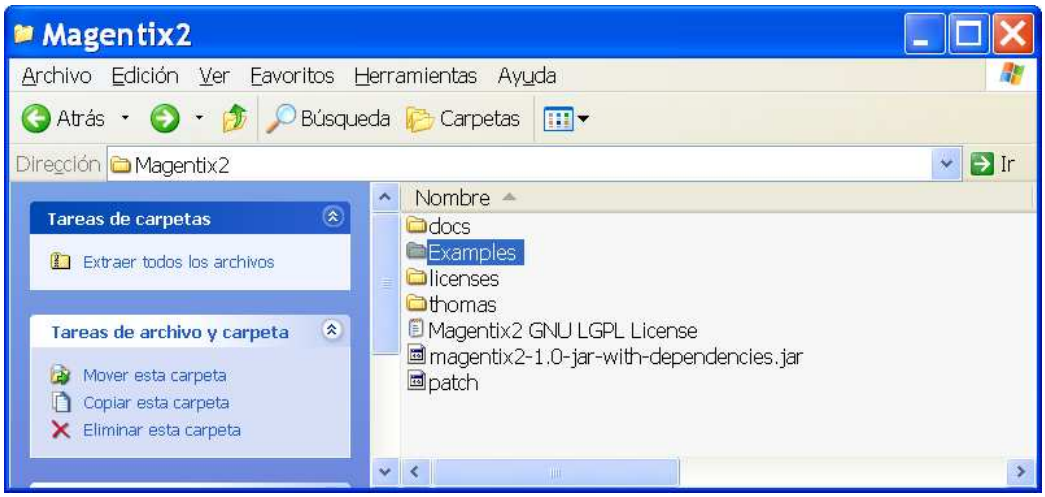
http://locke.dsic.upv.es:8080/magentix2/



Unzip Magentix 2

Unzipped Magentix2 directory includes:

- ***Magentix2-1.0-jar-with-dependencies.jar***: includes all additional libraries required by Magentix2.
- ***Magentix2 GNU PGGL License file***: license statement of Magentix2.
- ***Configuration sub-directory***: includes the *settings.xml* and *loggin.xml* configuration files of Magentix2.
- ***Examples sub-directory***: includes some examples of Magentix2 agent's implementation.
- ***Licenses sub-directory***: includes license statement terms of all software libraries required by Magentix2.
- ***Docs sub-directory***: includes guides of Magentix2.
- ***Patch file:*** executable that corrects Qpid source code problems.



Pre-requisites

QPID (required)

The document "**INSTALL QPID IN UBUNTU 9.04 32 BITS.pdf"** describes how to install Qpid/C++ broker from the source distribution http://qpid.apache.org/ on Linux/Unix operating system. Suppose that QPID c++ broker is running in a host called "broker.host.name" and listen messages on port 5672[7]. Other Broker configuration parameters as "virtualhost"[8], "user" "password" and "ssl" can be configured on the *settings.xml* file of Magentix2 (*configuration directory*):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
……….
        <entry key="host"> broker.host.name</entry>
        <entry key="port">port</entry>
        <entry key="vhost">virtualhost</entry>
        <entry key="user">username</entry>
        <entry key="password">password</entry>
        <entry key="ssl">false</entry>
……….
</properties>
```

JDK 6 (required)

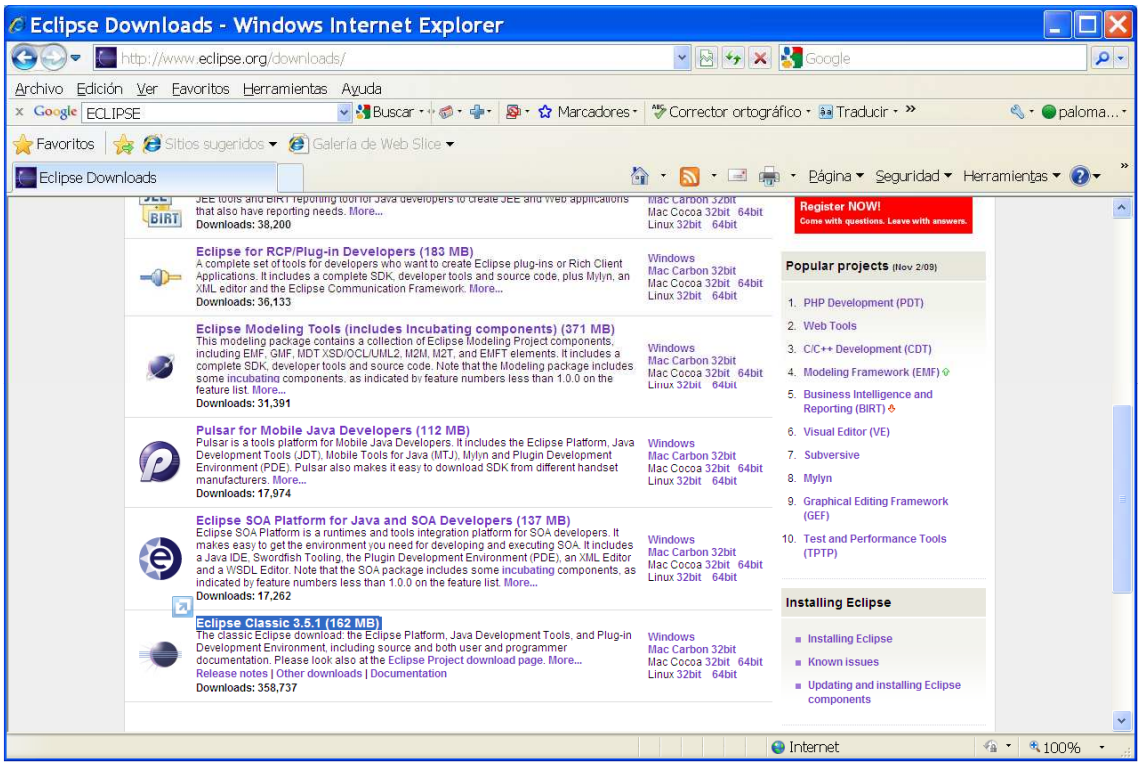
http://java.sun.com/javase/downloads/index.jsp#jdk

[7] The QPID C++ broker is pre-configured by default to listen messages on port 5672 but also it is possible configure to listen on other ports (http://qpid.apache.org/rasc.html).

[8] The term Virtual Host refers to the practice of running more than one web site on a single machine. The fact that they are running on the same physical server is not apparent to the end user.

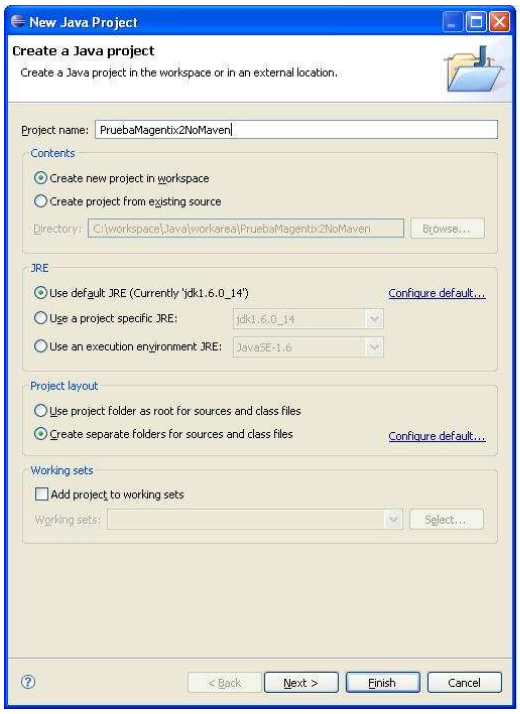
ECLIPSE (recommended IDE)

http://www.eclipse.org/

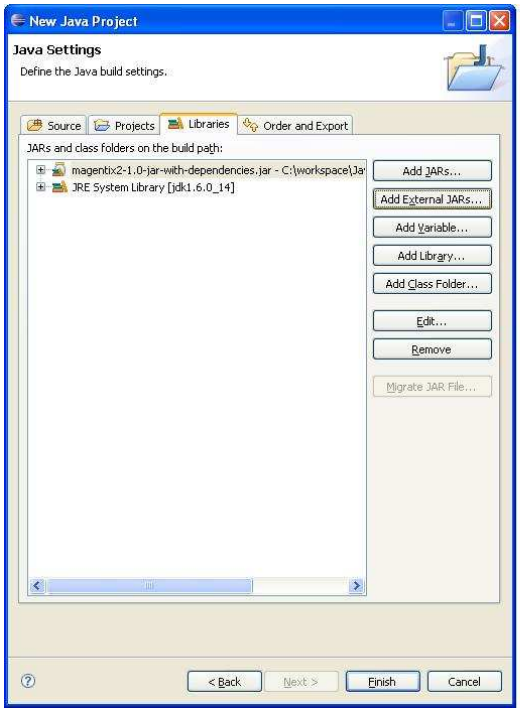


Configure Eclipse IDE

Create a New Java project:



Add Magentix2.jar dependence



Basic Examples

Description

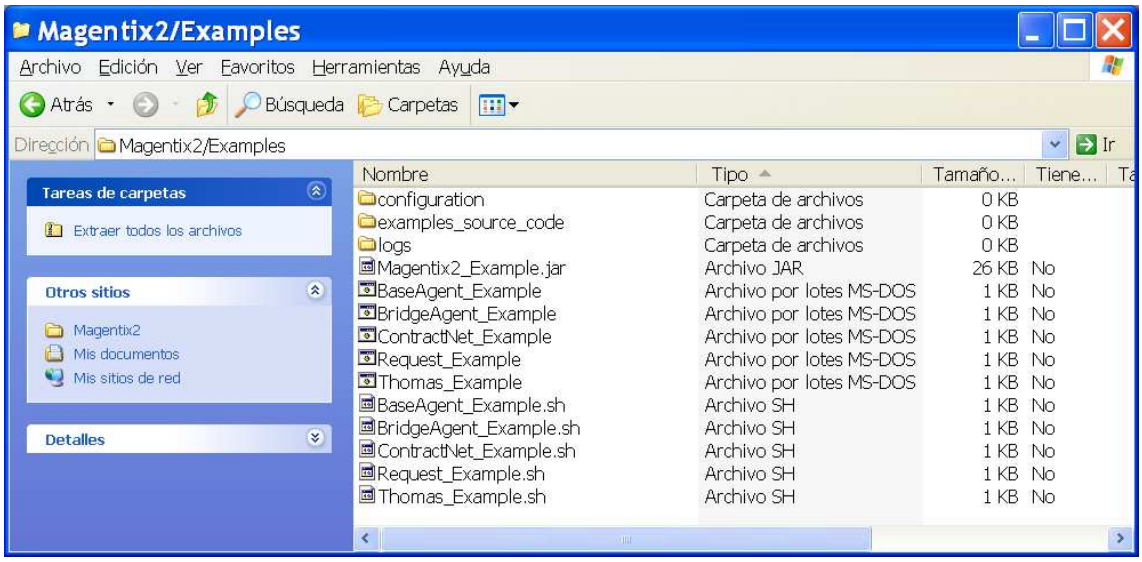
Subdirectory "*examples*" of Magentix2 distribution includes some basic examples of agents developed for this platform:

- *BaseAgent*: example of issuer/consumer messages. The sender agent builds and sends an ACLMessage to the consumer. When the ACLMessage arrives, the consumer displays the message on screen.
- *BridgeAgent*: example in which an internal agent send a message to an external agent of our platform.
- *Request*: example of an agent that implements the FIPA Request protocol. In this example, create two types of agents, one with the role responding to simulate a hospital attending emergency calls and the other a witness to an accident. When the witness see an accident sends a message of support to the hospital, hospital staff will tell whether the accident is within its area of action and where it can be the result of the operation.
- *ContractNet*: example of an agent that implements the FIPA ContractNet protocol. In this example, create two types of agents, one with the role responding to pretend to be a dealer and another with the role inciador to pretend to be a buyer, the initiator send a purchase request to each dealer, the dealer will return your bid and wait for the initiator decides, the client will choose one and this will return the contract, the rest are sent a rejection
- *Thomas*: example of an agent that access to Thomas SF and OMS web services. In this example development of a new agent client and a provider agent, is modelled as a unit (travelagency) within which are provided search services tourist information and booking of hotels and flights. Two types of roles within the unit interact travelagency: the role client (customer) and the role of service provider (provider).

Content

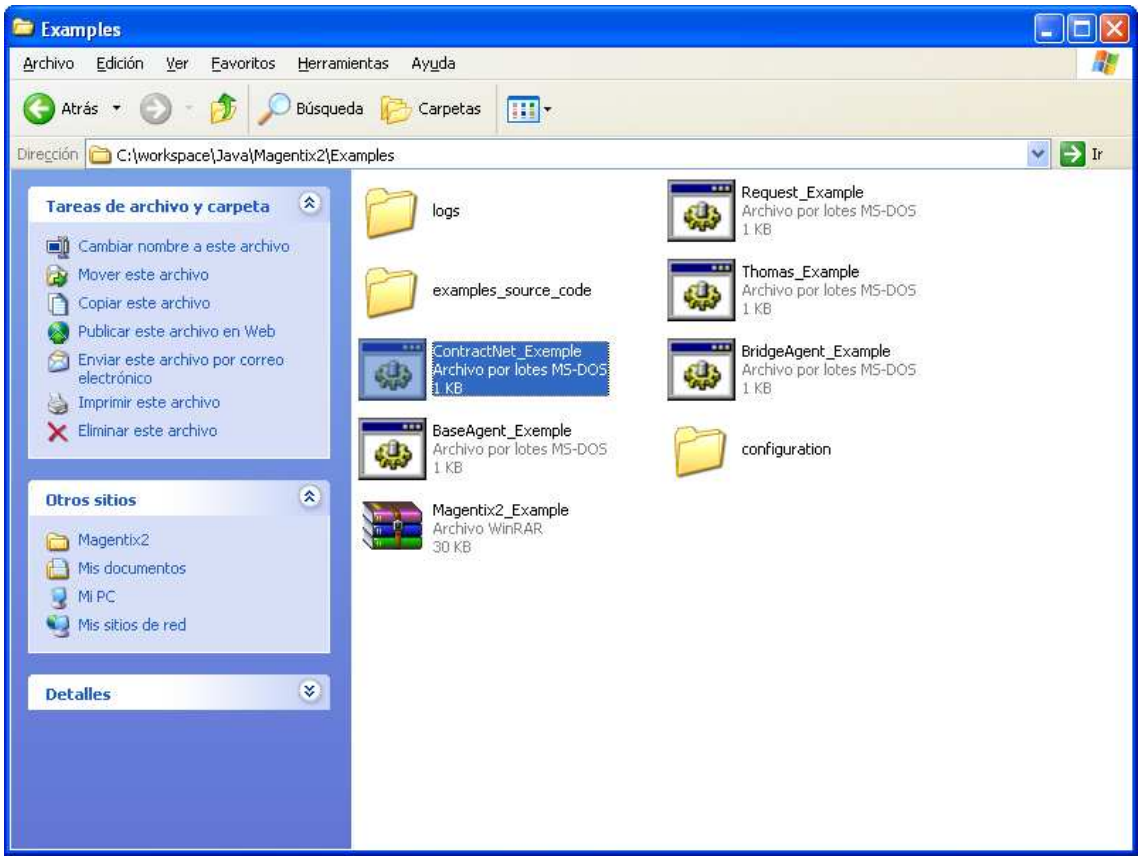
The examples subdirectory includes:

- "*Magentix2_Example.jar":* compiled code required to run examples.
- "*/examples_source_code*" subdirectory: the source code of examples.
- *"*.bat files*": run examples directly (windows)[9].
- *"*.sh files*": run examples directly (linux)
- *"/logs" subdirectory*: default log4j exit.
- *"/configuration"*: includes "*settings.xml*" and "*logs.xml*" configuration files of Magentix2 platform.



[9] For UNIX Systems the corresponding "**.sh*" files are required

Run Magentix2 examples (windows)



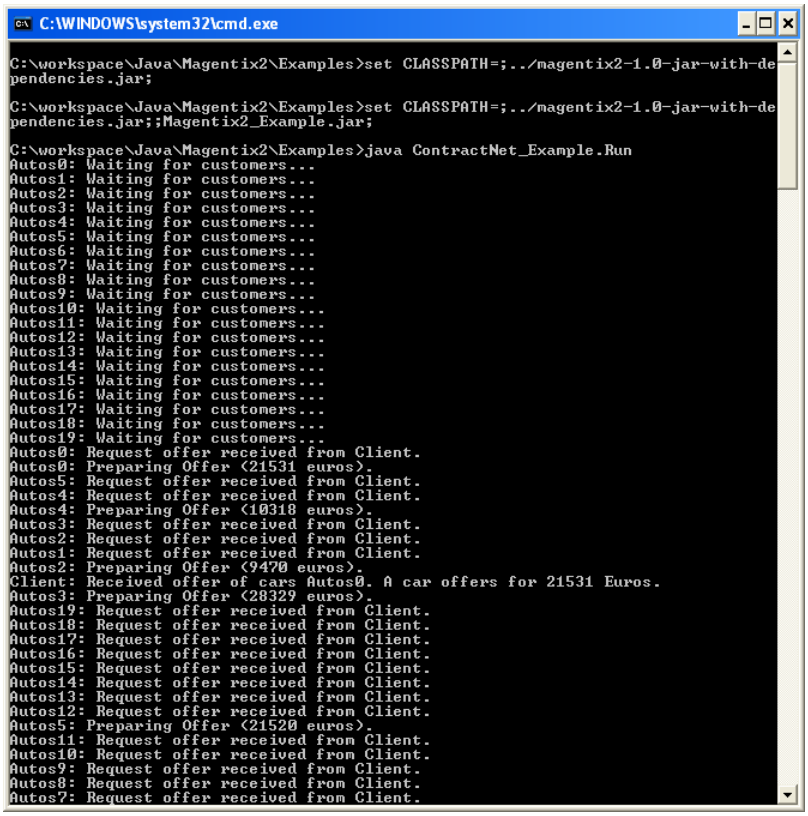
Required jars Classpath and run commad are included in *.bat files. For example:

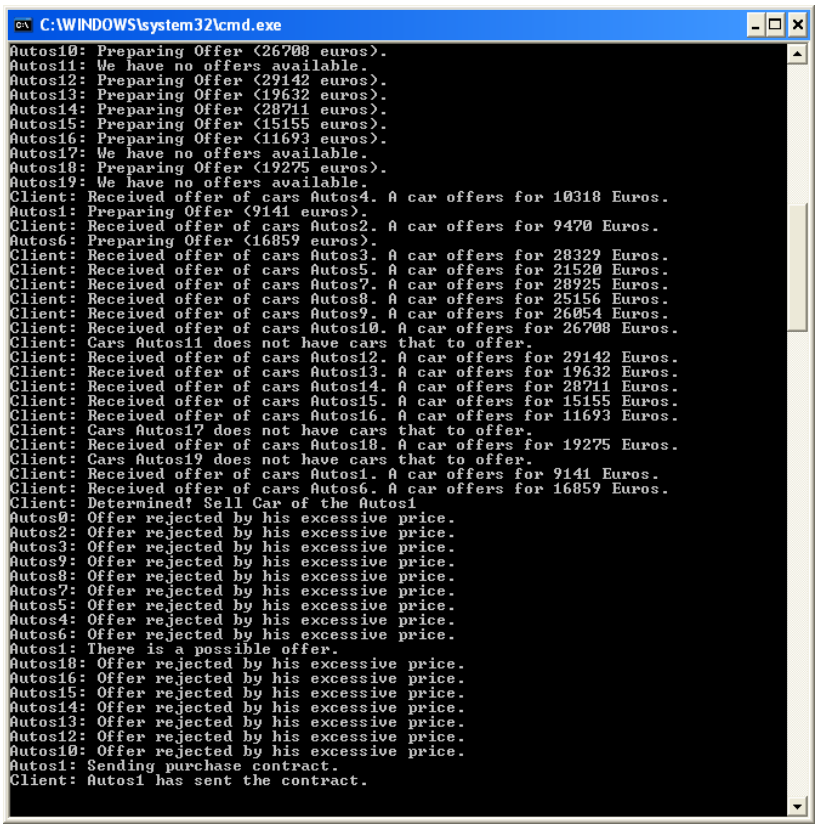
ContractNet_Example.bat

```
set CLASSPATH=%CLASSPATH%;../magentix2-1.0-jar-with-dependencies.jar;

set CLASSPATH=%CLASSPATH%;Magentix2_Example.jar;

java ContractNet_Example.Run
```


C:\WINDOWS\system32\cmd.exe
C:\workspace\Java\Magentix2\Examples>set CLASSPATH=;../magentix2-1.0-jar-with-dependencies.jar;
C:\workspace\Java\Magentix2\Examples>set CLASSPATH=;../magentix2-1.0-jar-with-dependencies.jar;;Magentix2_Example.jar;
C:\workspace\Java\Magentix2\Examples>java ContractNet_Example.Run
Autos0: Waiting for customers...
Autos1: Waiting for customers...
Autos2: Waiting for customers...
Autos3: Waiting for customers...
Autos4: Waiting for customers...
Autos5: Waiting for customers...
Autos6: Waiting for customers...
Autos7: Waiting for customers...
Autos8: Waiting for customers...
Autos9: Waiting for customers...
Autos10: Waiting for customers...
Autos11: Waiting for customers...
Autos12: Waiting for customers...
Autos13: Waiting for customers...
Autos14: Waiting for customers...
Autos15: Waiting for customers...
Autos16: Waiting for customers...
Autos17: Waiting for customers...
Autos18: Waiting for customers...
Autos19: Waiting for customers...
Autos0: Request offer received from Client.
Autos0: Preparing Offer (21531 euros).
Autos5: Request offer received from Client.
Autos4: Request offer received from Client.
Autos4: Preparing Offer (10318 euros).
Autos3: Request offer received from Client.
Autos2: Request offer received from Client.
Autos1: Request offer received from Client.
Autos2: Preparing Offer (9470 euros).
Client: Received offer of cars Autos0. A car offers for 21531 Euros.
Autos3: Preparing Offer (28329 euros).
Autos19: Request offer received from Client.
Autos18: Request offer received from Client.
Autos17: Request offer received from Client.
Autos16: Request offer received from Client.
Autos15: Request offer received from Client.
Autos14: Request offer received from Client.
Autos13: Request offer received from Client.
Autos12: Request offer received from Client.
Autos5: Preparing Offer (21520 euros).
Autos11: Request offer received from Client.
Autos10: Request offer received from Client.
Autos9: Request offer received from Client.
Autos8: Request offer received from Client.
Autos7: Request offer received from Client.


C:\WINDOWS\system32\cmd.exe
Autos10: Preparing Offer (26708 euros).
Autos11: We have no offers available.
Autos12: Preparing Offer (29142 euros).
Autos13: Preparing Offer (19632 euros).
Autos14: Preparing Offer (28711 euros).
Autos15: Preparing Offer (15155 euros).
Autos16: Preparing Offer (11693 euros).
Autos17: We have no offers available.
Autos18: Preparing Offer (19275 euros).
Autos19: We have no offers available.
Client: Received offer of cars Autos4. A car offers for 10318 Euros.
Autos1: Preparing Offer (9141 euros).
Client: Received offer of cars Autos2. A car offers for 9470 Euros.
Autos6: Preparing Offer (16859 euros).
Client: Received offer of cars Autos3. A car offers for 28329 Euros.
Client: Received offer of cars Autos5. A car offers for 21520 Euros.
Client: Received offer of cars Autos7. A car offers for 28925 Euros.
Client: Received offer of cars Autos8. A car offers for 25156 Euros.
Client: Received offer of cars Autos9. A car offers for 26054 Euros.
Client: Received offer of cars Autos10. A car offers for 26708 Euros.
Client: Cars Autos11 does not have cars that to offer.
Client: Received offer of cars Autos12. A car offers for 29142 Euros.
Client: Received offer of cars Autos13. A car offers for 19632 Euros.
Client: Received offer of cars Autos14. A car offers for 28711 Euros.
Client: Received offer of cars Autos15. A car offers for 15155 Euros.
Client: Received offer of cars Autos16. A car offers for 11693 Euros.
Client: Cars Autos17 does not have cars that to offer.
Client: Received offer of cars Autos18. A car offers for 19275 Euros.
Client: Cars Autos19 does not have cars that to offer.
Client: Received offer of cars Autos1. A car offers for 9141 Euros.
Client: Received offer of cars Autos6. A car offers for 16859 Euros.
Client: Determined! Sell Car of the Autos1
Autos0: Offer rejected by his excessive price.
Autos2: Offer rejected by his excessive price.
Autos3: Offer rejected by his excessive price.
Autos9: Offer rejected by his excessive price.
Autos8: Offer rejected by his excessive price.
Autos7: Offer rejected by his excessive price.
Autos5: Offer rejected by his excessive price.
Autos4: Offer rejected by his excessive price.
Autos6: Offer rejected by his excessive price.
Autos1: There is a possible offer.
Autos18: Offer rejected by his excessive price.
Autos16: Offer rejected by his excessive price.
Autos15: Offer rejected by his excessive price.
Autos14: Offer rejected by his excessive price.
Autos13: Offer rejected by his excessive price.
Autos12: Offer rejected by his excessive price.
Autos10: Offer rejected by his excessive price.
Autos1: Sending purchase contract.
Client: Autos1 has sent the contract.

Thomas

Architecture

Recently several works have appeared that try to solve the problem of integrating the multi-agent system paradigm and the service-oriented computing paradigm. It is obvious, that there are many similarities among them. Both paradigms try to offer solutions for the development of complex and adaptive systems in distributed open environments. In this line, integrating these technologies is possible to model autonomous and heterogeneous computational entities in dynamic and open environments. Such entities may be reactive, proactive and with the ability to communicate in a flexible way with other entities. One of the existing proposals works in the line to create links, as a gateway, between the two directions. The proposed solution tries to communicate agents and web services in a transparent, but independent, way. This is the line of the Agent and Web Services Interoperability (AWSI) IEEE FIPA Working Group (http://www.fipa.org/subgroups/AWSI-WG.html). Although interesting, our proposal tries to go beyond, raising a total integration of both technologies. So agents can offer and invoke services in a transparent way to other agents or entities, as well as external entities can interact with our agents through the use of the offered services. THOMAS architecture consists basically of a set of modular services. THOMAS feeds initially of the FIPA architecture expanding its capabilities. The agents have access to the infrastructure offered by THOMAS through a range of services including on different modules or components. The main components of THOMAS are the following:

- Service Facilitator (SF), this component offers simple and complex services to the active agents and organizations. Basically, its functionality is like a yellow page service and a service descriptor in charge of providing a green page service.
- Organization Manager Service (OMS), it is mainly responsible of the management of the organizations and their entities. Thus, it allows the creation and the management of any organization.

Service Facilitator

The SF is a mechanism and support by which the organization and agents can, at the same time, offer and discover services. The SF provides a place in which the autonomous entities can register service descriptions as directory entries.
The SF acts as a gateway to access the THOMAS platform. It manages this access transparently, by means of security techniques and access rights management. The SF can find services searching for a given service profile or searching by the goals that can be fulfilled executing the service. This is done using the matchmaking and service composition mechanisms that are provided by the SF.
The SF acts also as a yellow pages manager and in this way it can also find which entities provide a given service. A service offers some capacities, each of which to fulfil a given goal. The service may have some pre-conditions that have to be true before the service can be executed. It exchange one or more input and output messages. Before a successful service execution it has some effects on its environment. Moreover, there could be additional parameters,

which are independent of the service functionality (non-functional parameters), such as quality of service, deadlines, and security protocols among other. And finally, the service results can be enhanced using automatic service composition mechanisms (for example, partial matchmaking). To do this the SF maintains the description of the internal processes that are executed when the service is running.

A service represents an interaction of two entities, which are modelled as communications among independent processes. In our case, the Multi-agent Technology provides us with FIPA communication protocols which are well established mechanisms in order to standardize the interactions. In this way, every service has an associated protocol. In those cases in which the service requires the execution of a chain of protocols, the service is marked as "complex". Taking into account that we are working with semantic services, another important data is the ontology used in the service. In this way, when the service description is accessed, any entity will has all the needed information in order to interact with the service and how to make an application that can use the service. Such a description can also be used for pre-compiled services, in which the process model of the service will be, instead of the internal processes of the service, the sequence of the elementary services that will be executed.

Organization Manager Service

This component is in charge of organizations life cycle management, including specification and administration of both their structural components (roles, units and norms) and their execution components (participant agents and the roles they play; active units in each moment). OMS offers all services needed for a suitable organization performance. These services are classified as:

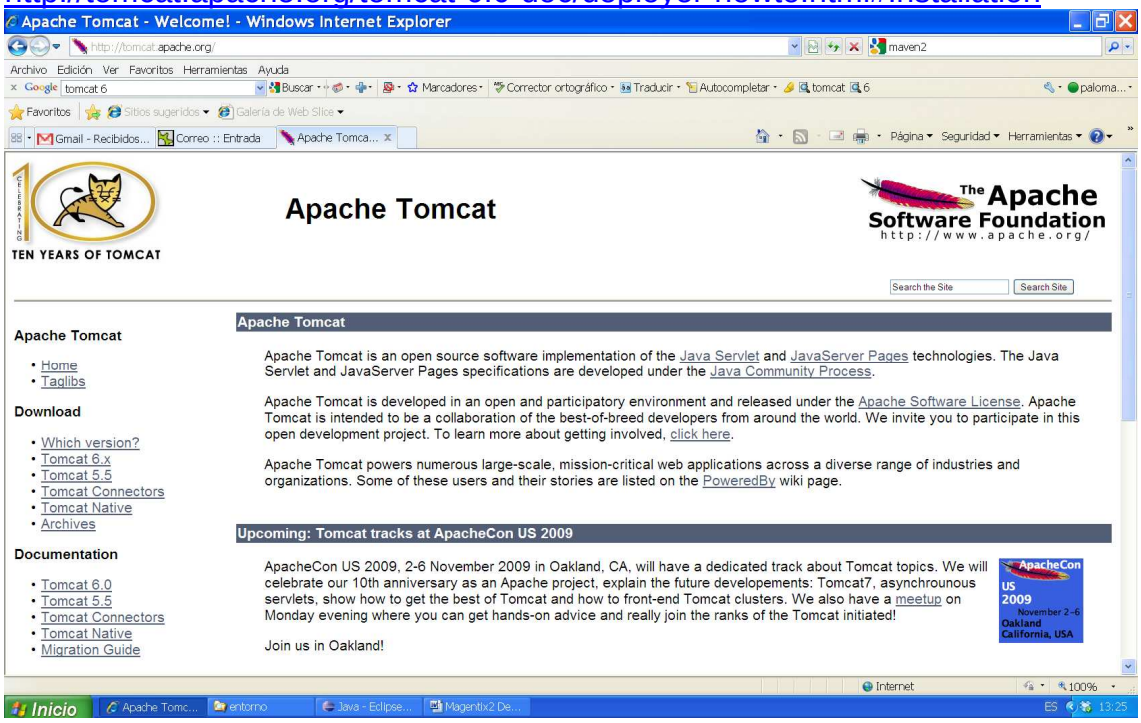
- Structural services, that modify the structural and normative organization specification;
- Dynamical services, that allow agents to entry or leave the organization dynamically, as well as role adoption.

By means of the publication of the structural services, OMS allows modifying, in execution time, some aspects related to the organization structure, functionality or normativity. For example, a specific agent of the organization could be allowed to add new norms, roles or units. This type of services should be restricted to internal roles of the system, which have enough permission for doing this kind of operations (i.e. supervisor role). Moreover, in some concrete applications those services might not be published in the SF, so then agents cannot dynamically modify the structural components. Dynamical services manage creation of new agents in the organization, entry or exit of unit members and role adoption. These services are always published in the SF.

Additional requisites

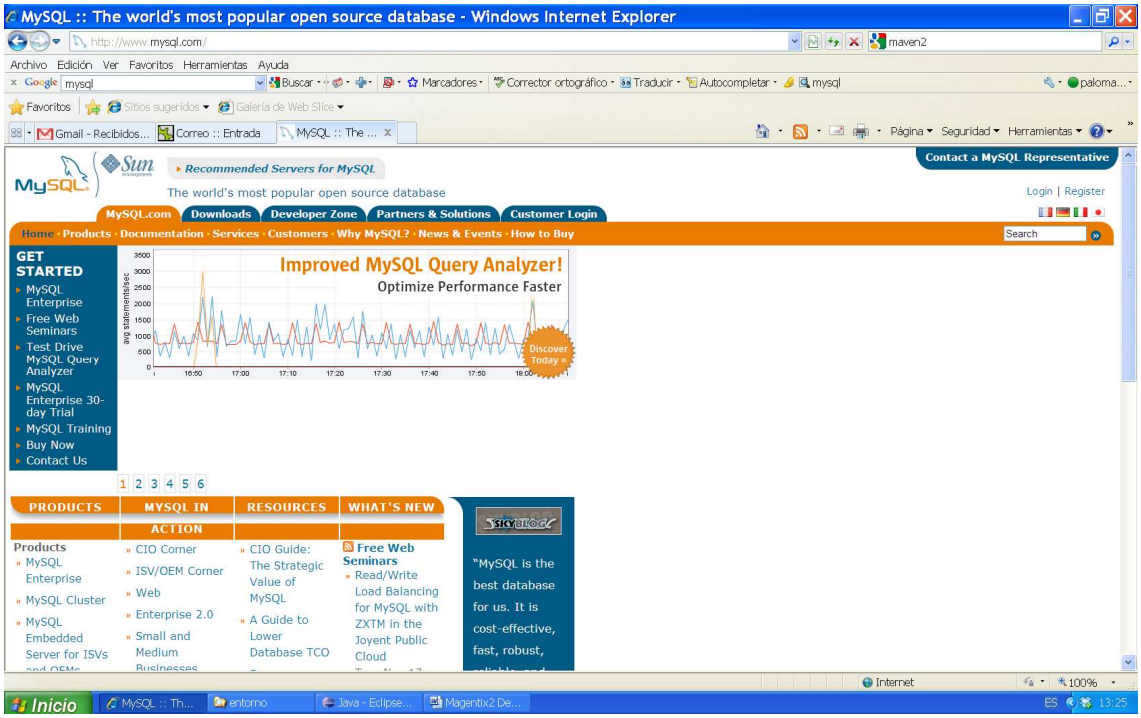
TOMCAT 6.0 (recommended server)

http://tomcat.apache.org/tomcat-6.0-doc/deployer-howto.html#Installation



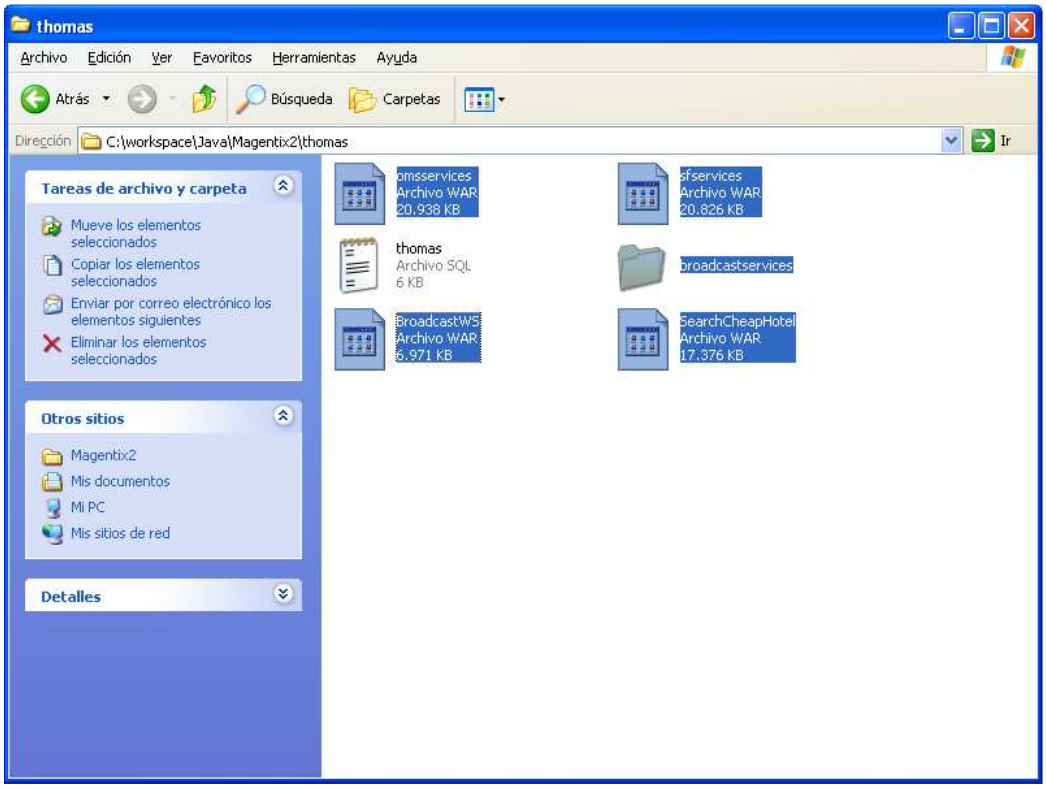
MySQL (recommended DDBB)

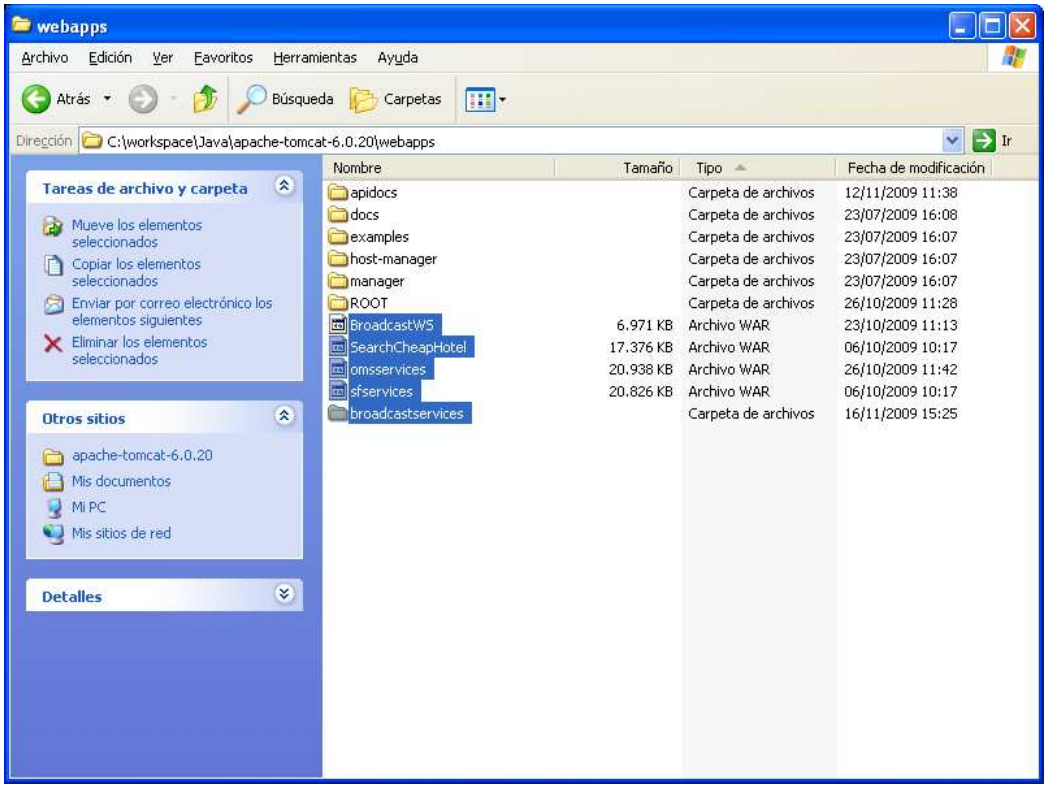
http://www.mysql.com/



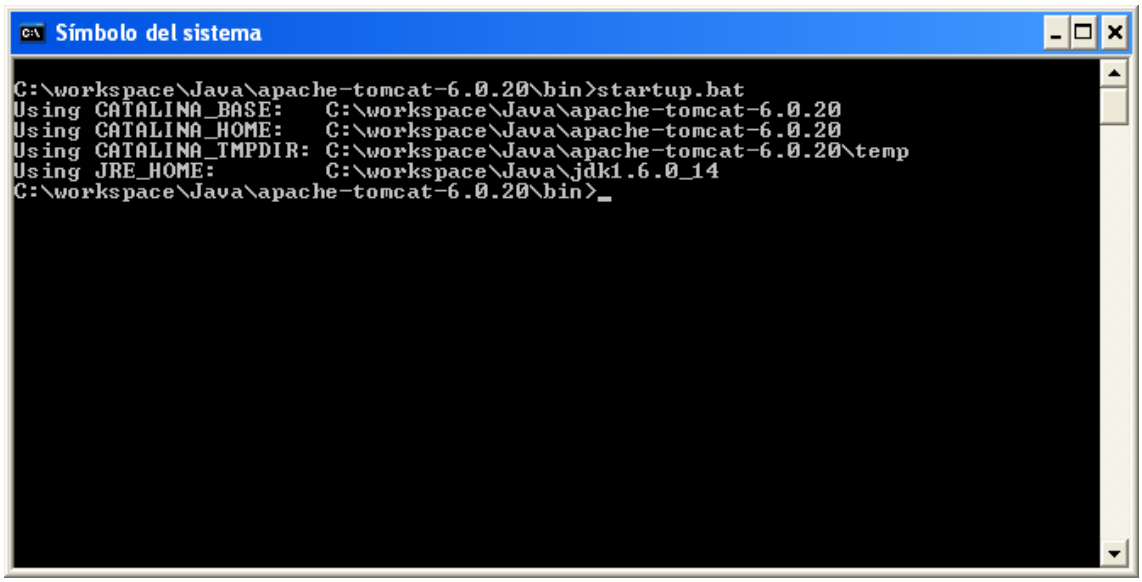
Deploy Thomas in TOMCAT

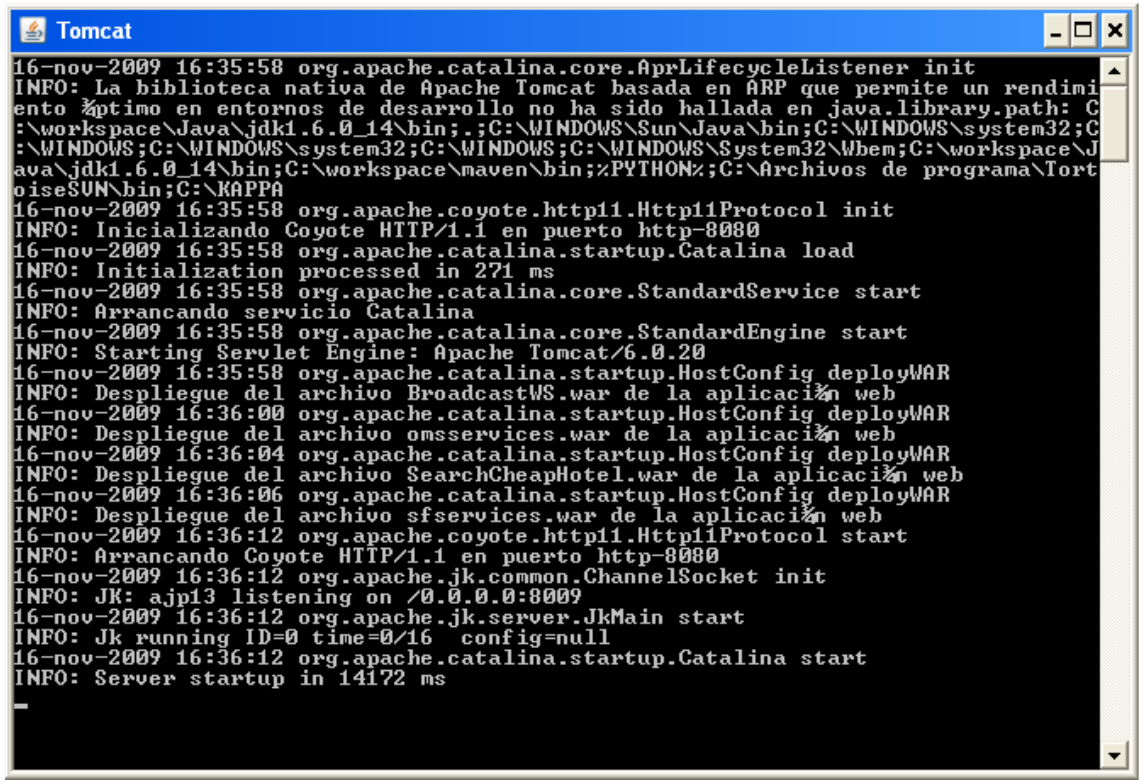
Copy packaged libraries of Thomas (*omsservices.war, sfservices.war*) and also the Thomas examples (*SearchCheapHotel.war, BroadcastWS.war and the broadcastservices directory*) from Magentix2 distribution to the subdirectory *webapss* of Tomcat:





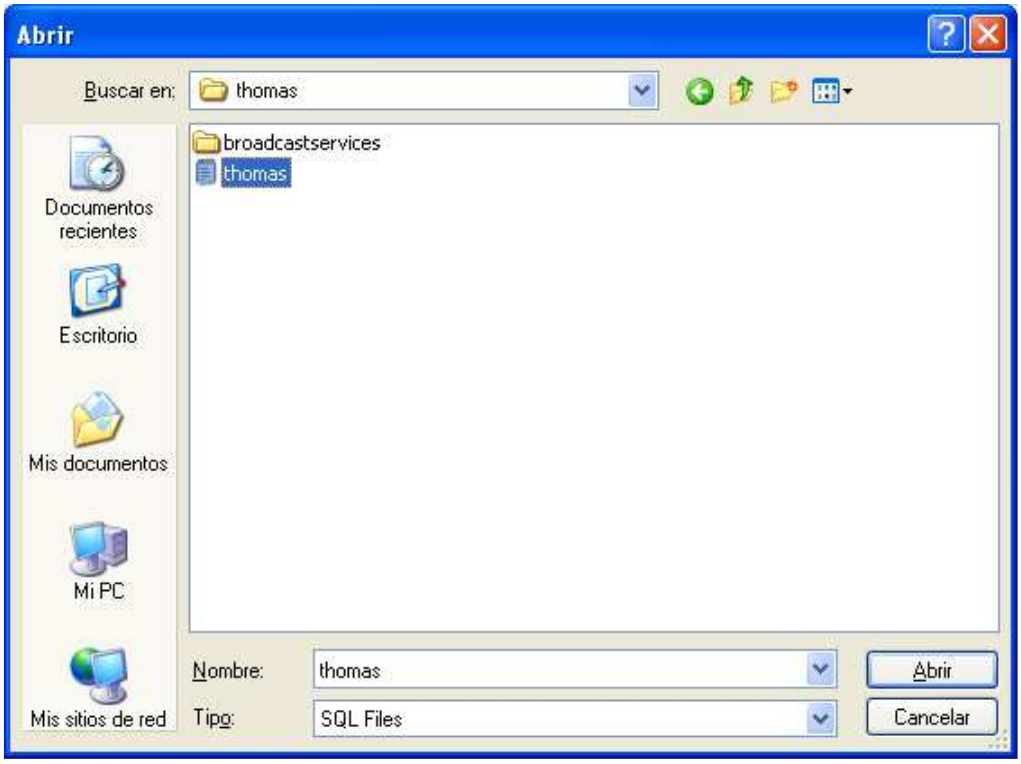
Start Tomcat by running the *startup.bat* file on the */bin* subdirectory of Tomcat. You must have the JDK 1.6 installed before you can start Tomcat.



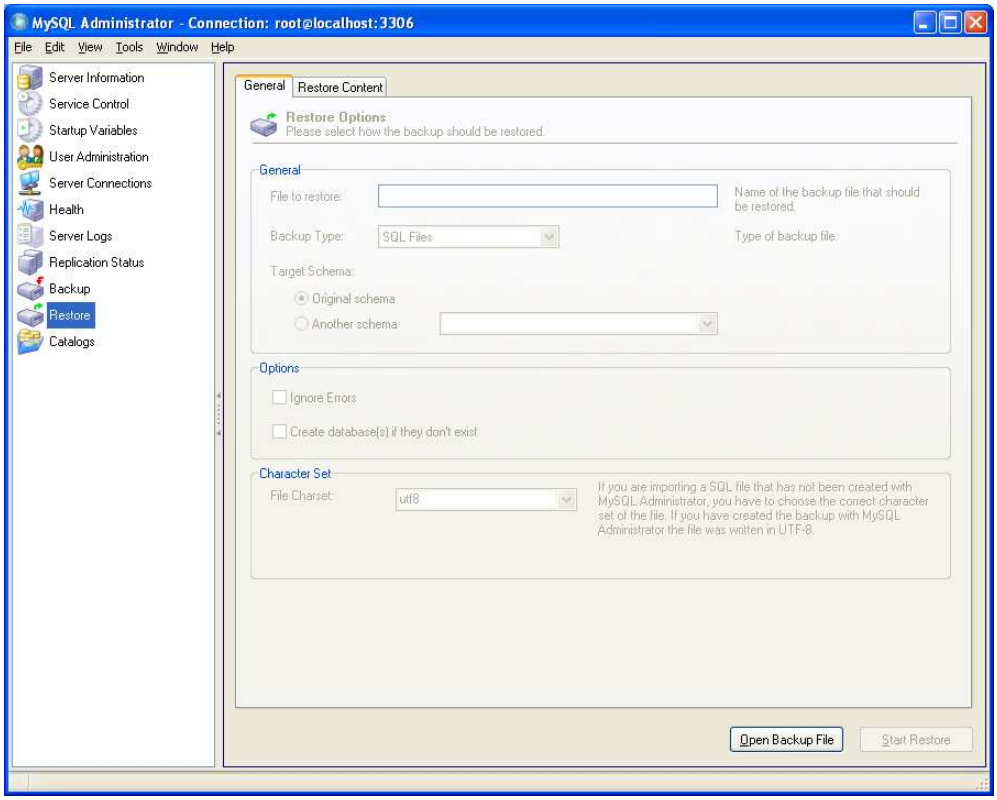


Install Thomas DDBB in MySQL

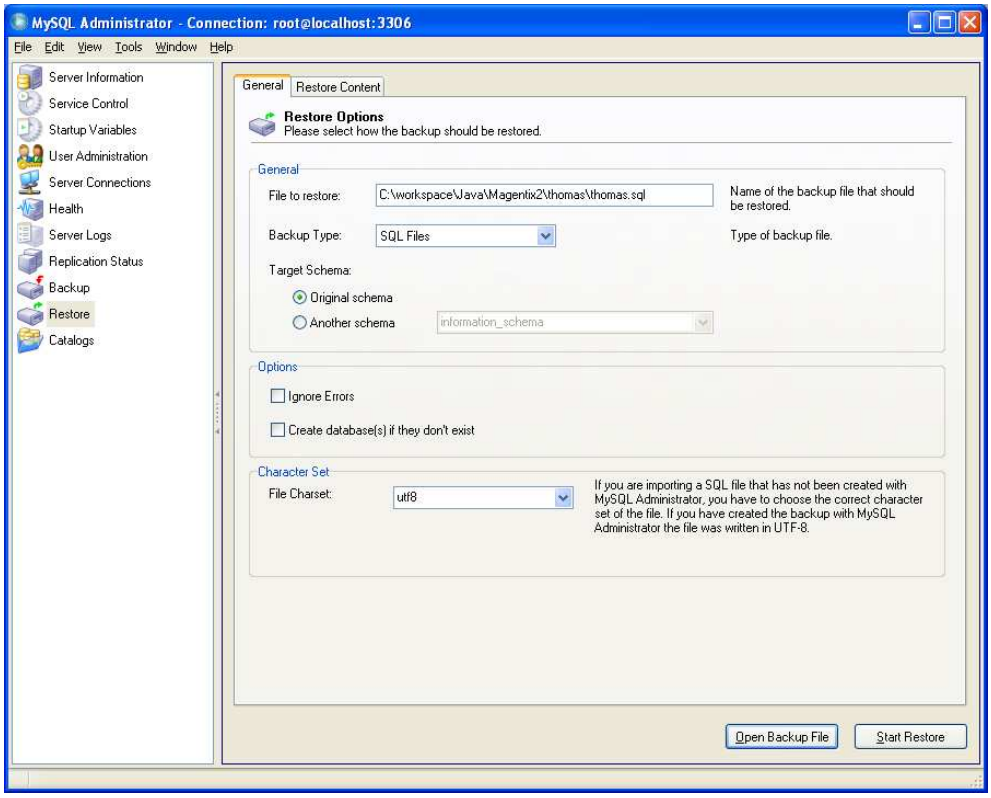
"*Thomas.sql*" file allow restore Thomas Data Base in MySQL:



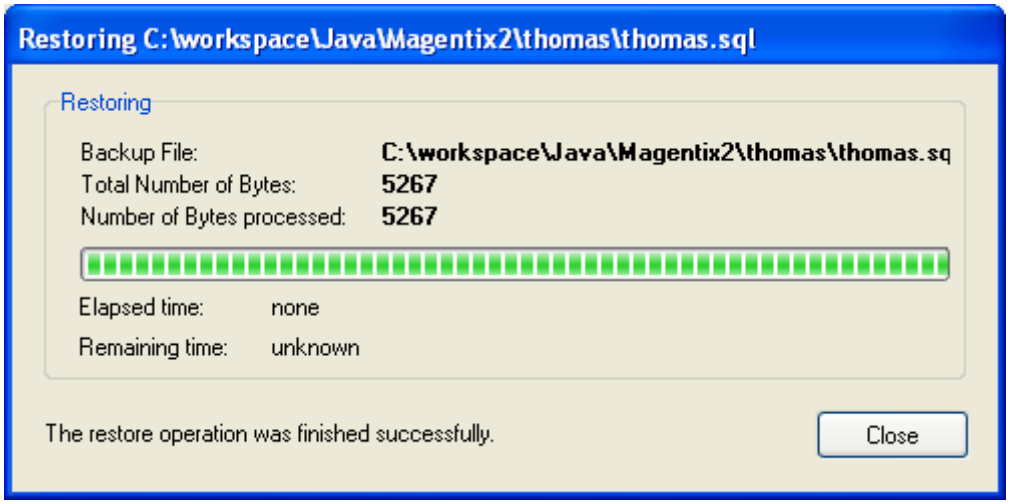
Open the MySQL Administrator and select *restore* option:



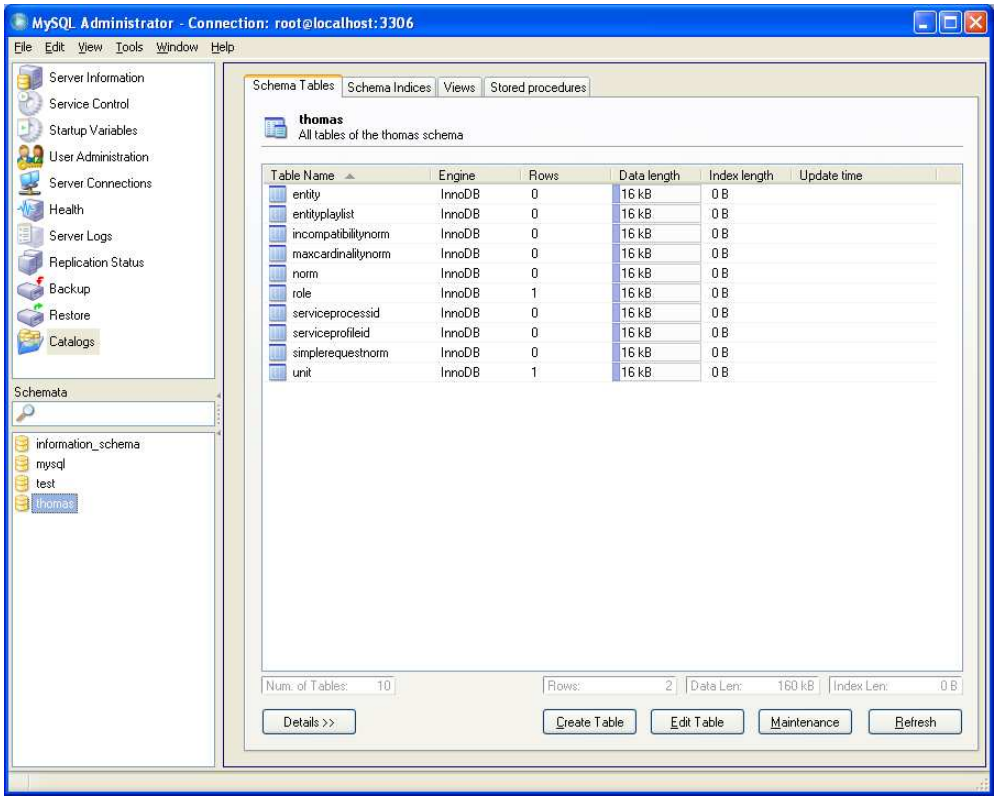
Select *Thomas.sql* file:



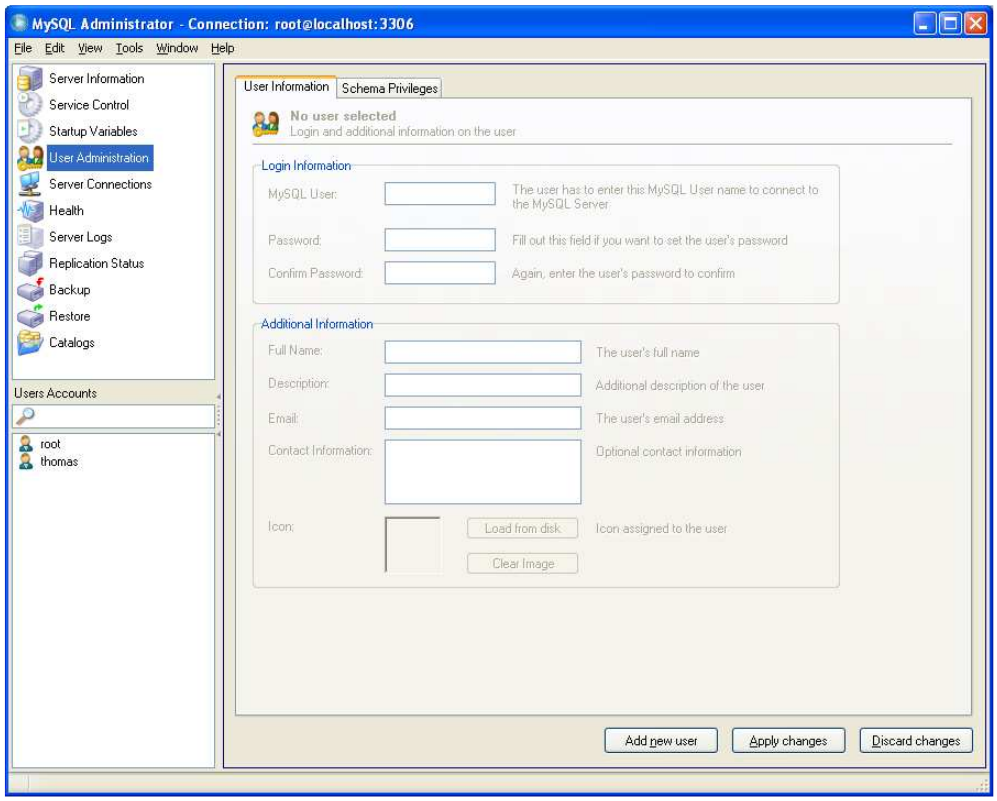
Init restore process:



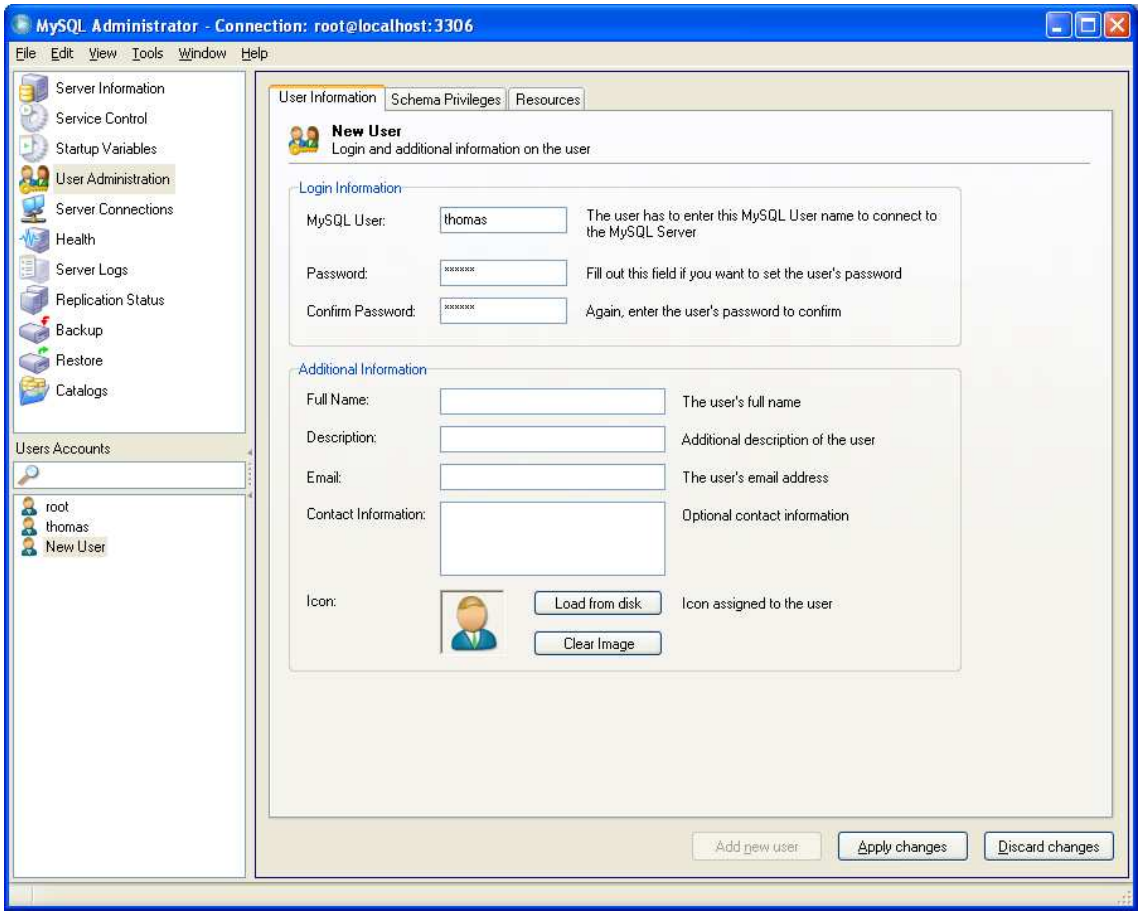
Now Thomas DDBB could be installed in MySQL



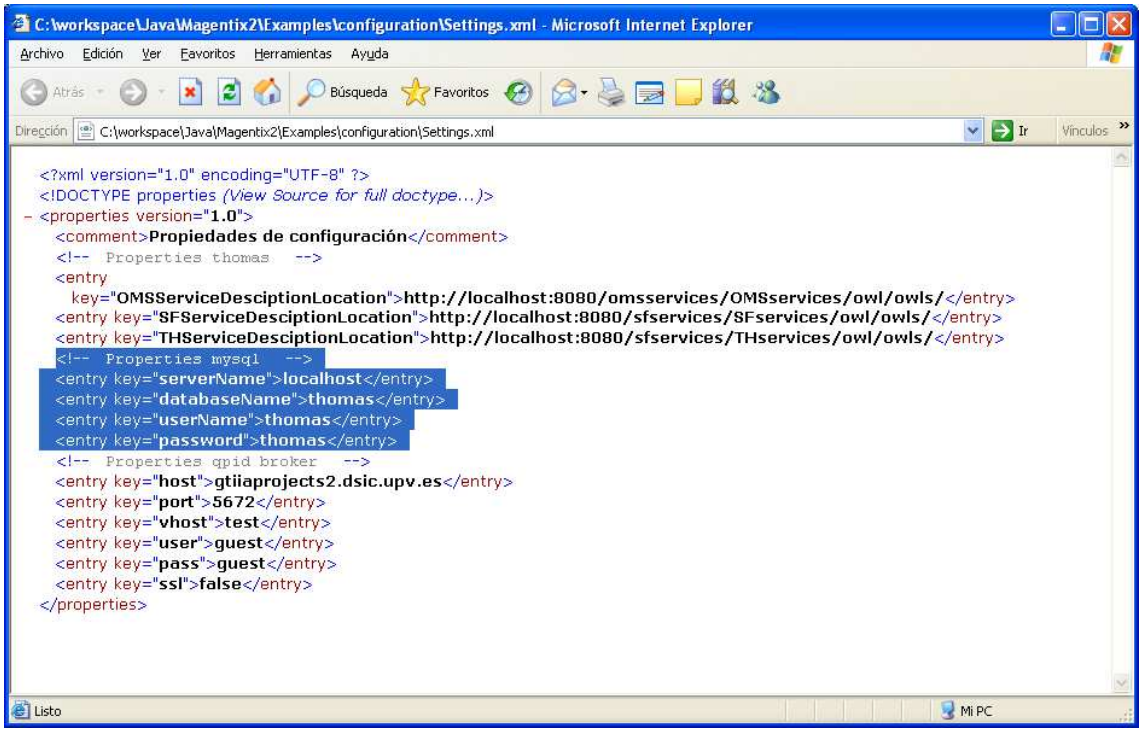
Add a new user to Thomas Schema:



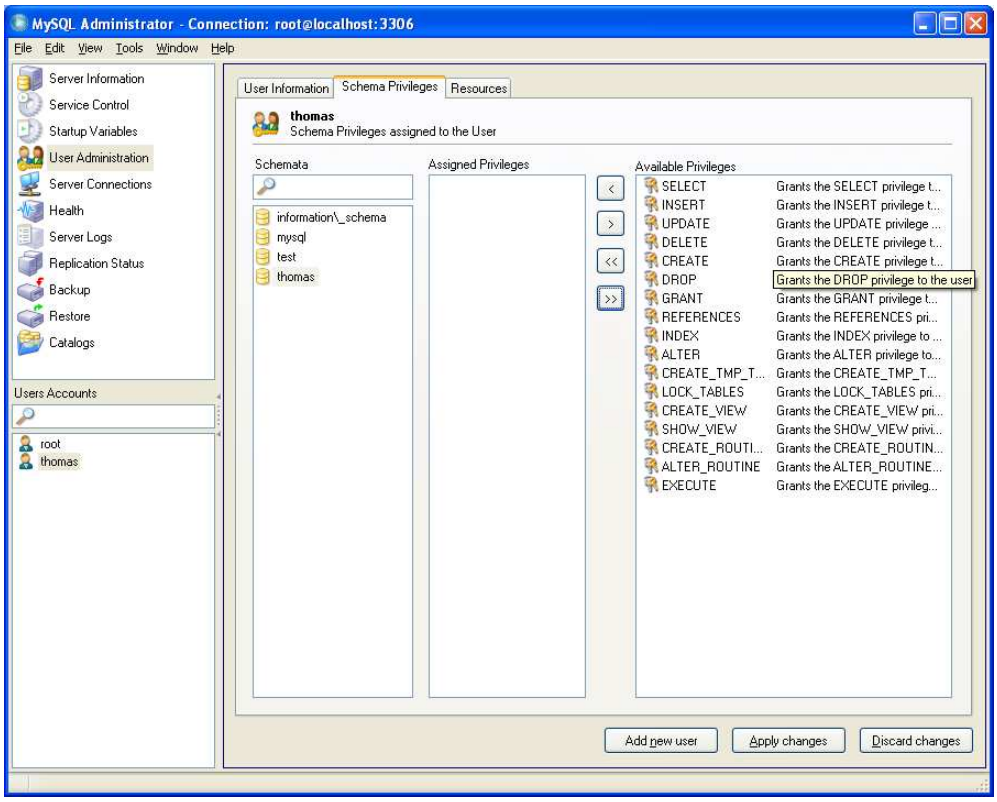
For example we add the user=”thomas” with password= “thomas”

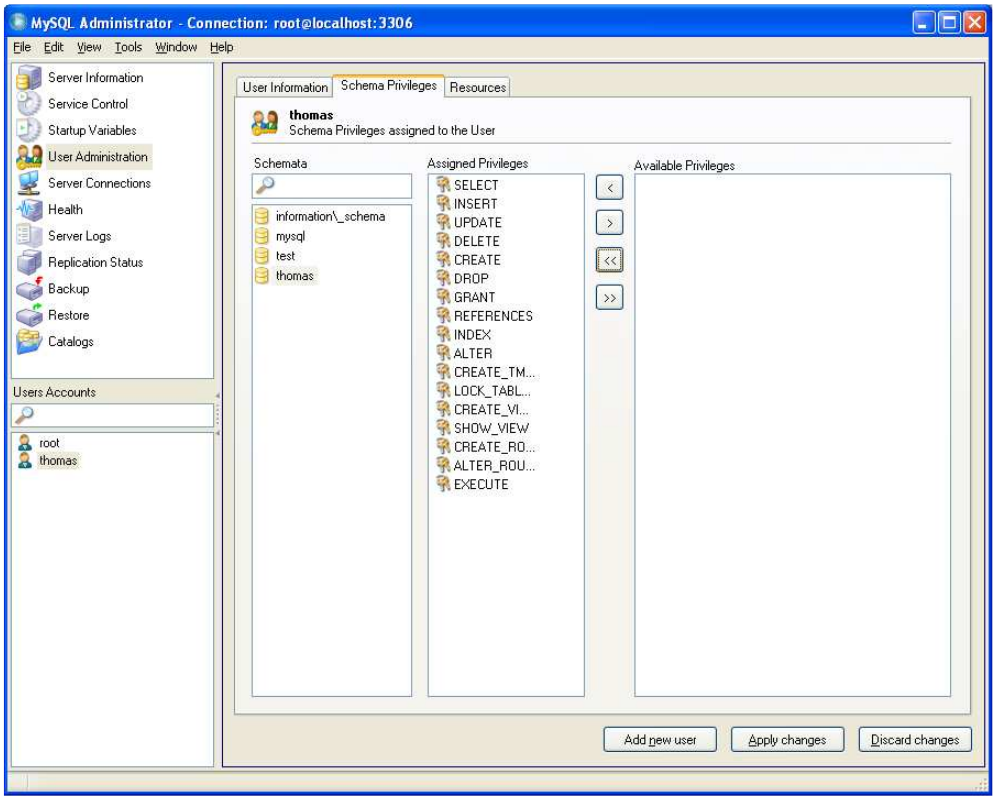


Also these values must be configured on the *settings.xml* file of Magentix2:

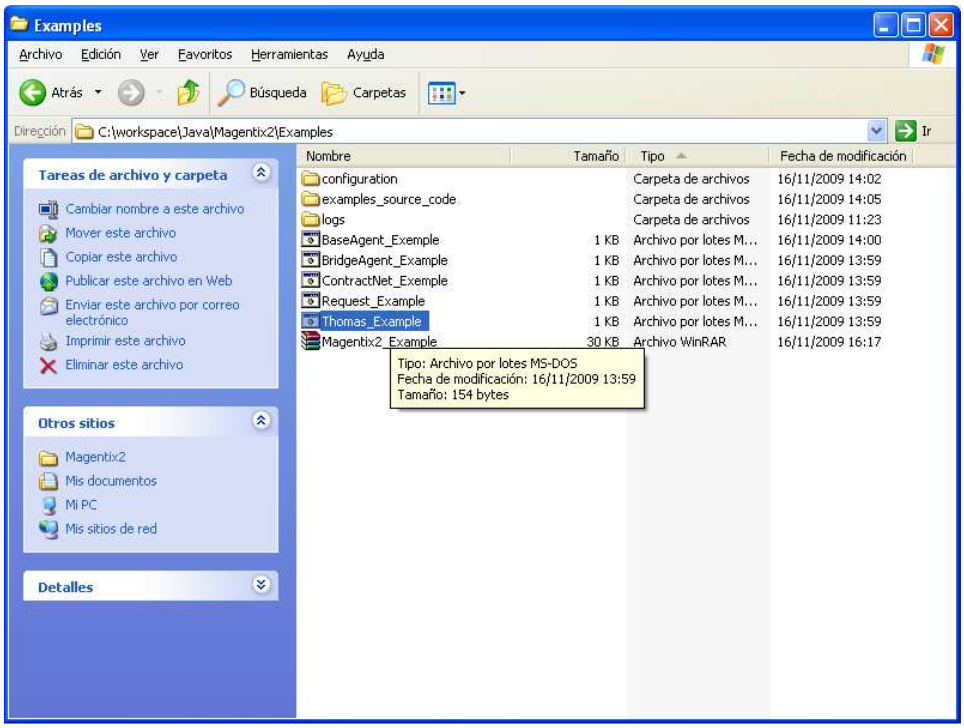


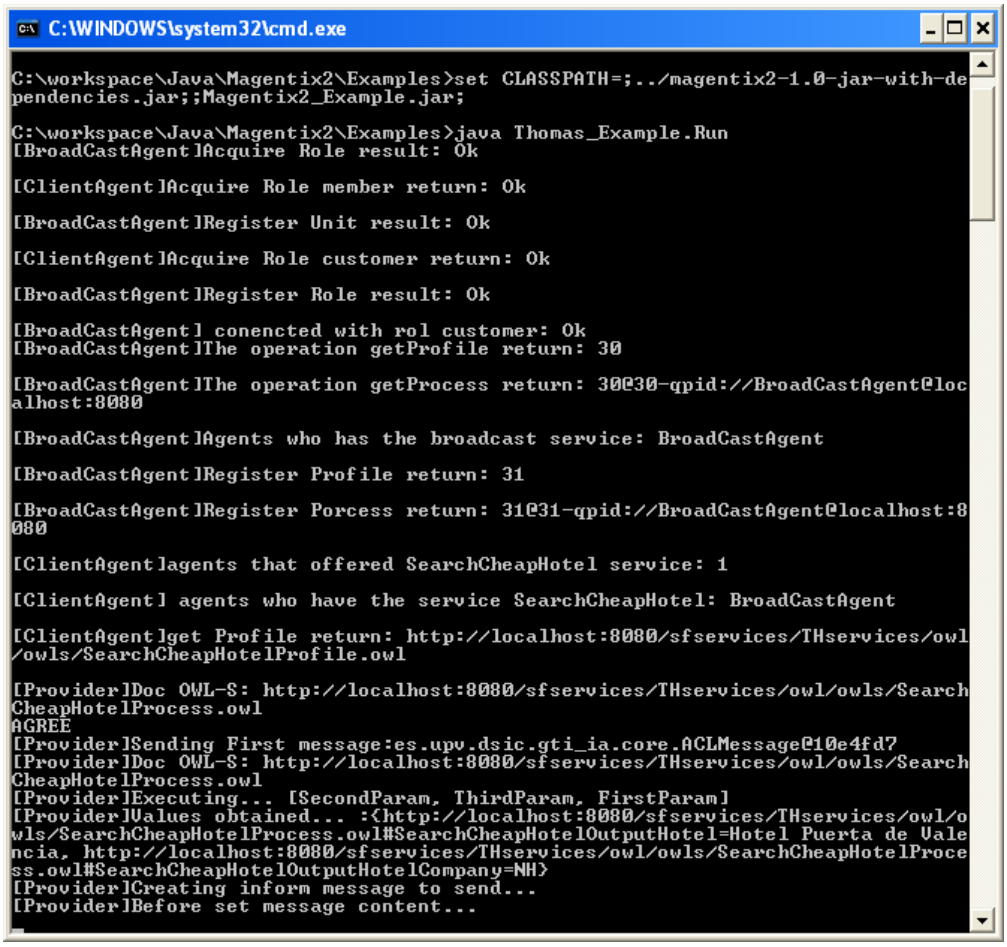
Finally available privileges for all tables of Thomas must be assigned to this user:





Run Thomas Example





Developing Agents for Magentix2

Sender Agent

```
public class SenderAgent extends BaseAgent {

	public SenderAgent(AgentID aid) throws Exception {
		super(aid);
	}

	public void execute() {
		logger.info("Executing, I'm " + getName());
		AgentID receiver = new AgentID("consumer");

		/**
		 * Building a ACLMessage
		 */
		ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
		msg.setReceiver(receiver);
		msg.setSender(this.getAid());
		msg.setLanguage("ACL");
		msg.setContent("Hello, I'm " + getName());
		/**
		 * Sending a ACLMessage
		 */
		send(msg);
	}
```

Consumer Agent

```
public class ConsumerAgent extends BaseAgent {

	LinkedBlockingQueue<MessageTransfer> internalQueue;

	public ConsumerAgent(AgentID aid) throws Exception {
		super(aid);
	}

	public void execute() {
		logger.info("Executing, I'm " + getName());
		/**
		 * This agent has no definite work. Wait infinitely the arrival of new
		 * messages.
		 */
		while (true) {

		}
	}

	public void onMessage(ACLMessage msg) {
		/**
		 * When a message arrives, its shows on screen
		 */
		logger.info("Mensaje received in " + this.getName()
				+ " agent, by onMessage: " + msg.getContent());
	}
```

Client Agent

```
acquired the member role at the organization

result = OMSservices.acquireRole(this, "member", "virtual");

result = OMSservices.acquireRole(this,"customer", "travelagency");

waiting that the agentBroadcast registered service SearchCheapHotel

do{
                results = SFservices.searchService(this, "SearchCheapHotel");
}while(results.size()==0);


agents = SFservices.getProcess(this, results.get(0));



for (AgentID agent : agents)
                        System.out
                                        .println("[ClientAgent] agents who have the service
SearchCheapHotel: "
                                                        + agent.name+"\n");


String profile = SFservices.getProfile(this,results.get(0));

                ArrayList<String> arg = new ArrayList<String>();
                arg.add("FirstParam");
                arg.add("SecondParam");
                arg.add("ThirdParam");

call the service SearchCheapHotel

SFservices.genericService(this,agents.get(0),profile,"http://localhost:8080/sfservices/THservice
s/owl/owls/SearchCheapHotelProcess.owl", arg);
```

PROVIDER AGENT

```
//We create the class that will make us the agent proxy oms,facilitates access to the methods of the OMS
                OMSProxy OMSservices = new OMSProxy();

                //We create the class that will make us the agent proxy sf,facilitates access to the methods of the SF
                SFProxy SFservices = new SFProxy();

//We create a SFServiceDescription, one for service that we have
SFServiceDescription serviceOne = new SFServiceDescription("http://localhost:8080/broadcastservices/owl/owls/","http://localhost:8080/broadcastservices/owl/owls/");
acquired the member role at the organization
try
                {
result = OMSservices.acquireRole(this, "member", "virtual");

result = OMSservices.acquireRole(this,"customer", "travelagency");




serviceONe.setServiceGoal("SearchCheapHotel");


                        SFservices.registerProfile(this,serviceOne);
                        System.out.println("[ProviderAgent]The operation register Profile return: "+  serviceOne.getID()+"\n");

                    SFservices.registerProcess(this, serviceOne);

                //Rol responder
                        Responder responder = new Responder(this);

                        this.addTask(responder);

                        //when we do not have to create more roles we await the expiration of the other roles


                    Monitor m = new Monitor();
                    m.waiting();

                }catch(Exception e){
                        logger.error(e.getMessage());

                }
```

Launching agents:

Initialization tasks:

Our platform have been developed with log4j, for this reason is neccesary it will be initialized as follows:

i. DOMConfigurator.configure("configuration/loggin.xml");
ii. Logger logger = Logger.getLogger(Run.class);

Connecting to Qpid broker

Always before launch any agent, a connection to the Qpid broker must have been established. Thus any agent in our platform will use this communication. There are three different ways to establish a connection with Qpid broker:

- Connect()
  Take the input connection parameters from settings.xml file.
- connect(qpidHost, qpidPort, qpidVhost, qpdidUser, qpidPassword, qpidSSL)
  Take into account all the parameters specefied as input
- Connect(qpidHost)

Take into account the qpidhost parameter and considering the rest as default parameters.

For instance:

- AgentsConnection.connect();
- AgentsConnection.connect(localhost, 5672,"test","guest","guest",false);
- AgentsConnection.connect("localhost ");

Instantiating agents

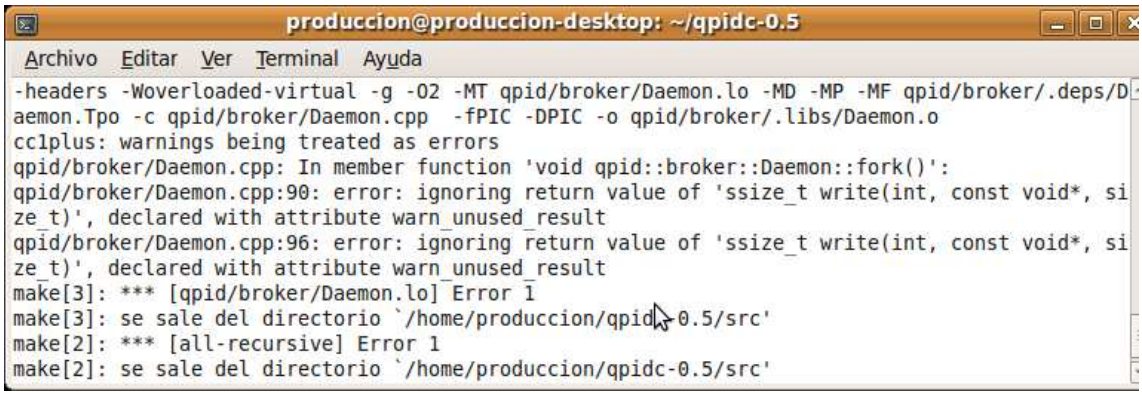
Once created agents, we can now instantiate it. Please note that the platform can not allow several agents with the same name.

i. SenderAgent agent1 = new SenderAgent(new AgentID(
"qpid://emisor@localhost:8080"));
ii. ConsumerAgent agent2 = new ConsumerAgent(new AgentID("consumer"));

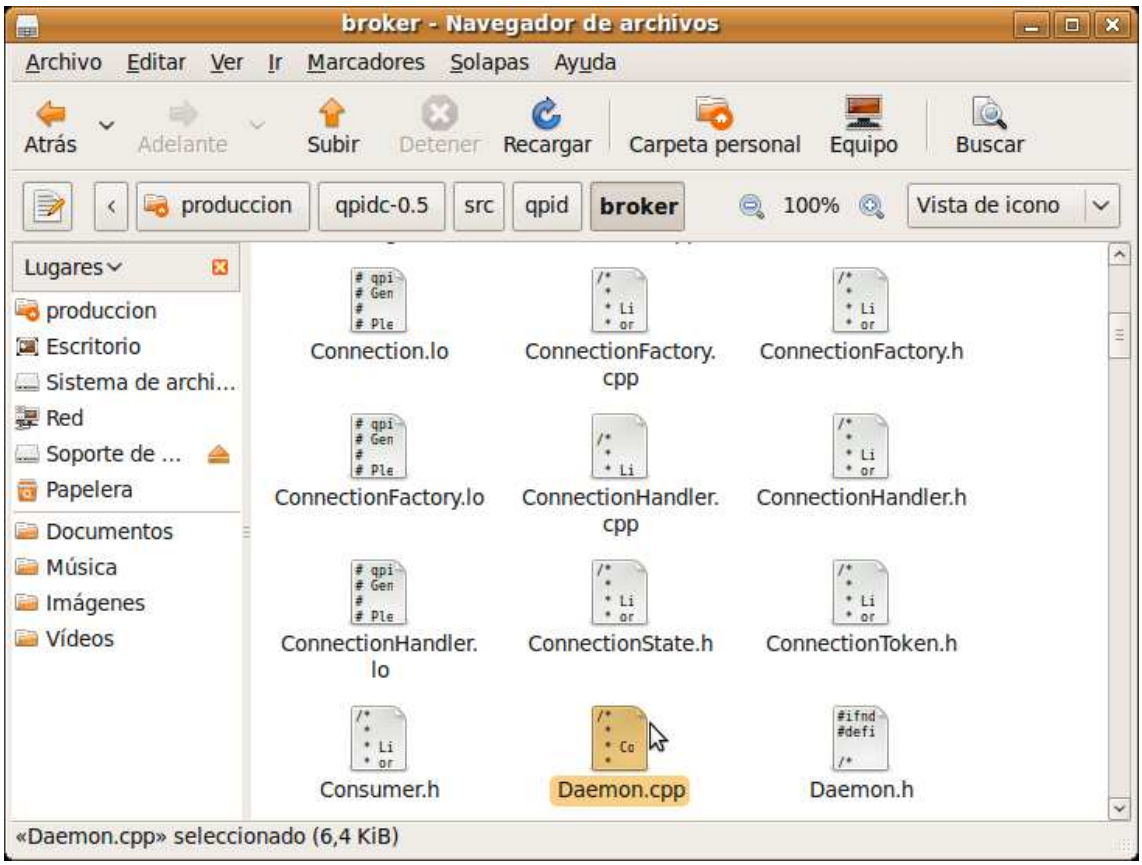
1. Launching agents
   i. agent2.start();
   ii. agent.start();

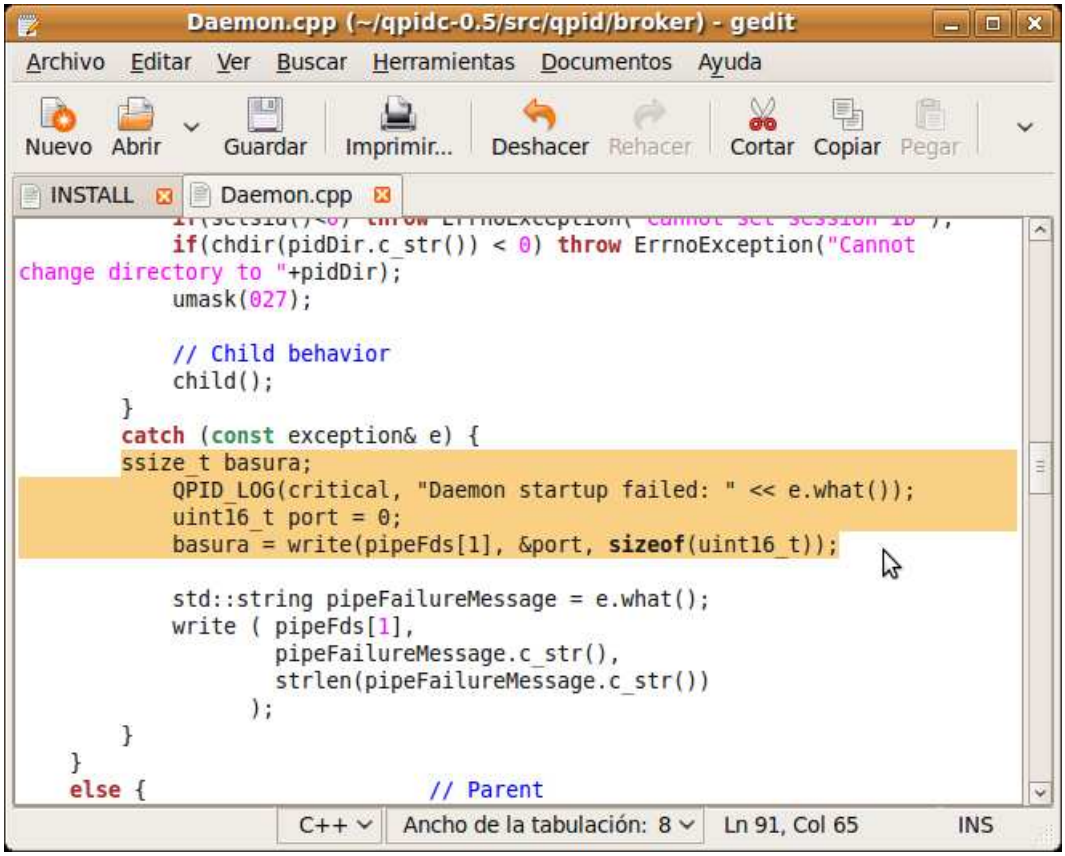
APENDIX I Qpid Source Code modifications

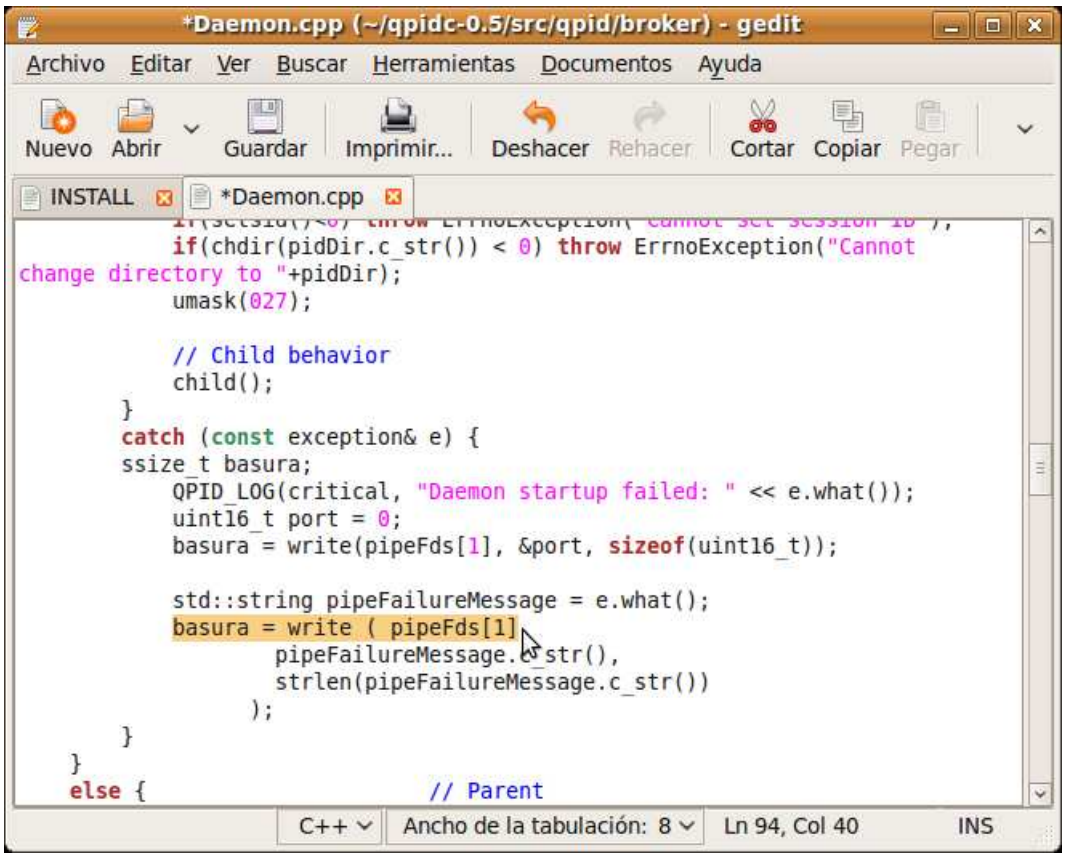
Problem: ignoring the return value of '**ssize_t**' in the '**write'** function causes a compilation error. This error was a warning in previous versions of the c++ compiler but must be corrected in order to compile and install the broker in Ubuntu 9. 04

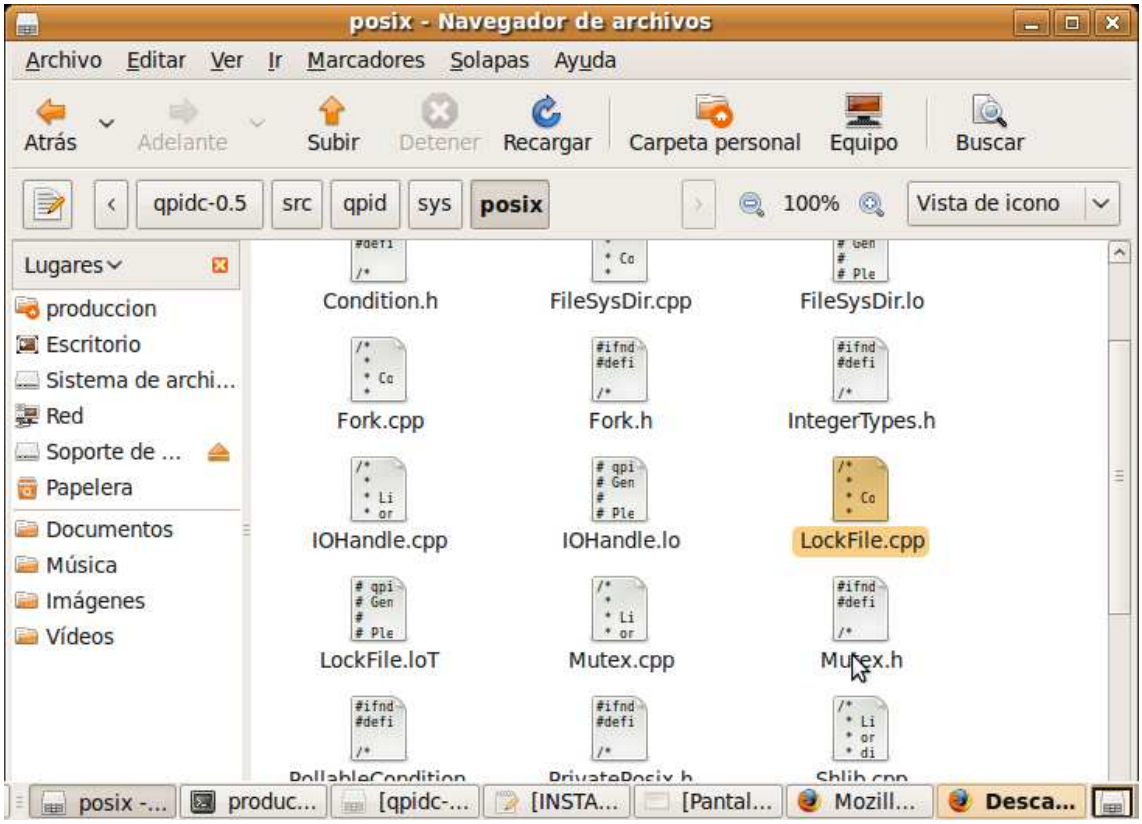


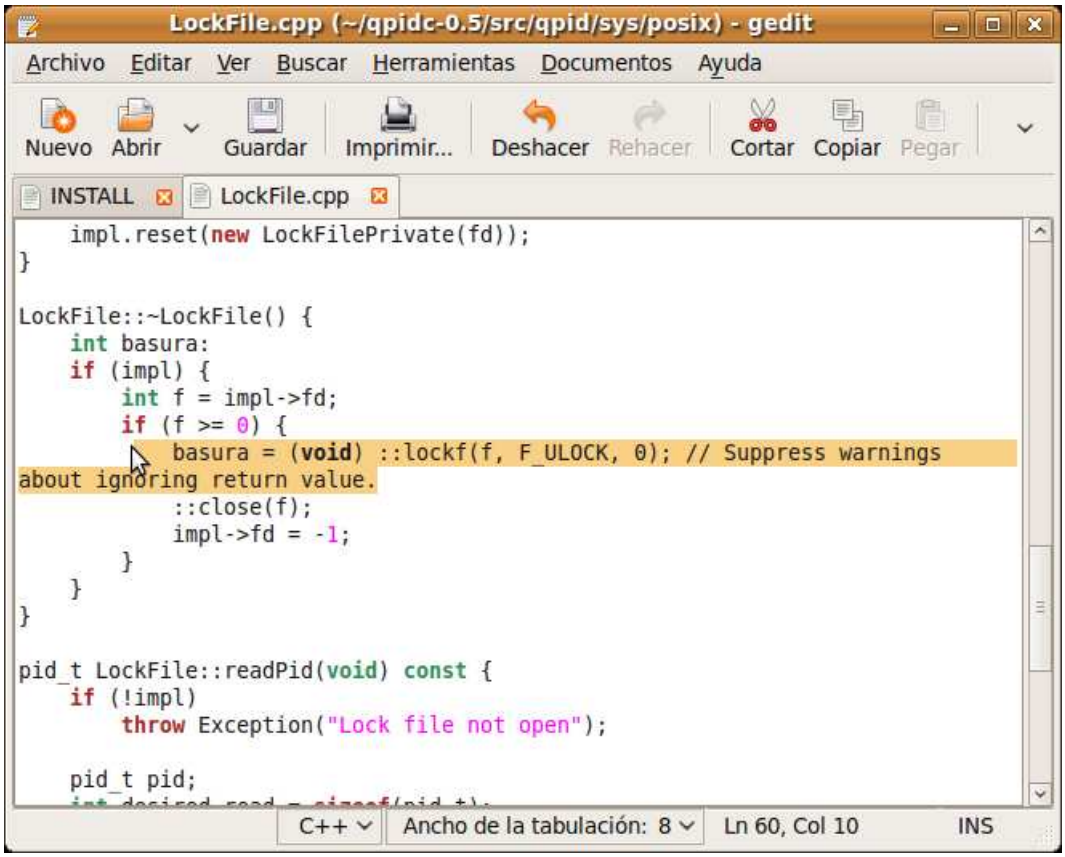
Different ssize_t type variables "basura" have been added in the classes Daemon.cpp, LockFile.cpp and ManagementAgentImpl.cpp in order to compile Qpid broker in UBUNTU.

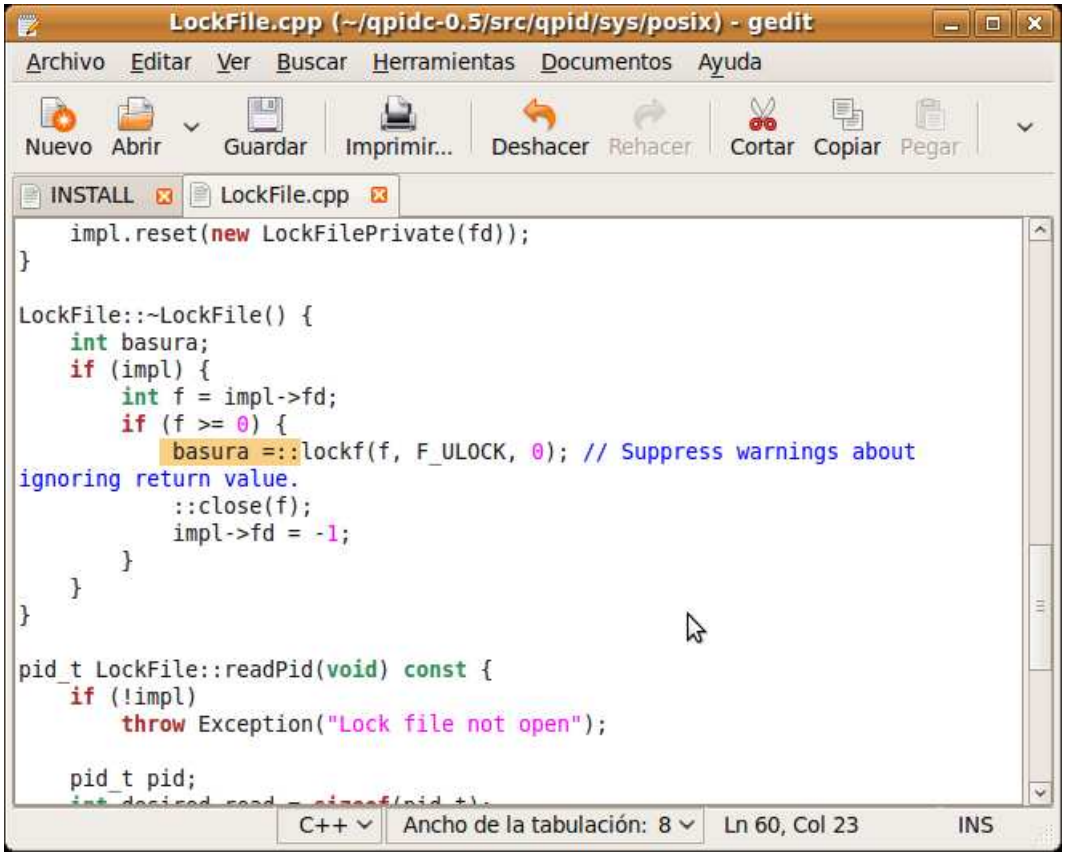



Daemon.cpp (~/qpidc-0.5/src/qpid/broker) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Nuevo Abrir Guardar Imprimir... Deshacer Rehacer Cortar Copiar Pegar
INSTALL
Daemon.cpp
if(chdir(pidDir.c_str()) < 0) throw ErrnoException("Cannot change directory to "+pidDir);
umask(027);
// Child behavior
child();
}
catch (const exception& e) {
ssize_t basura;
QPID_LOG(critical, "Daemon startup failed: " << e.what());
uint16_t port = 0;
basura = write(pipeFds[1], &port, sizeof(uint16_t));
std::string pipeFailureMessage = e.what();
write (pipeFds[1],
pipeFailureMessage.c_str(),
strlen(pipeFailureMessage.c_str())
);
}
}
else { // Parent
C++ Ancho de la tabulación: 8 Ln 91, Col 65 INS

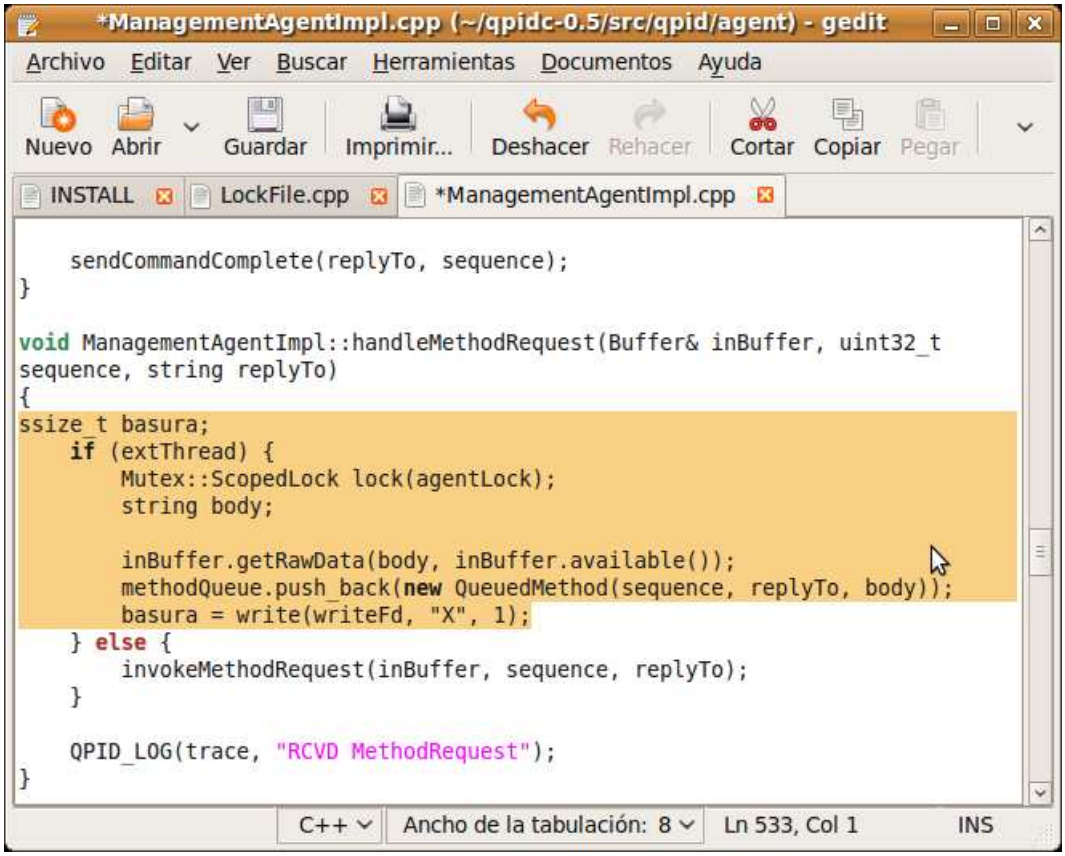

*Daemon.cpp (~/qpidc-0.5/src/qpid/broker) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Nuevo Abrir Guardar Imprimir... Deshacer Rehacer Cortar Copiar Pegar
INSTALL
*Daemon.cpp
if(chdir(pidDir.c_str()) < 0) throw ErrnoException("Cannot change directory to "+pidDir);
umask(027);
// Child behavior
child();
}
catch (const exception& e) {
ssize_t basura;
QPID_LOG(critical, "Daemon startup failed: " << e.what());
uint16_t port = 0;
basura = write(pipeFds[1], &port, sizeof(uint16_t));
std::string pipeFailureMessage = e.what();
basura = write (pipeFds[1],
pipeFailureMessage.c_str(),
strlen(pipeFailureMessage.c_str())
);
}
}
else { // Parent
C++ Ancho de la tabulación: 8 Ln 94, Col 40 INS


posix - Navegador de archivos
Archivo Editar Ver Ir Marcadores Solapas Ayuda
Atrás Adelante Subir Detener Recargar Carpeta personal Equipo Buscar
qpidc-0.5 src qpid sys posix
100%
Vista de icono
Lugares
produccion
Escritorio
Sistema de archi...
Red
Soporte de ...
Papelera
Documentos
Música
Imágenes
Vídeos
Condition.h
FileSysDir.cpp
FileSysDir.lo
Fork.cpp
Fork.h
IntegerTypes.h
IOHandle.cpp
IOHandle.lo
LockFile.cpp
LockFile.loT
Mutex.cpp
Mutex.h
posix -...
produc...
[qpidc-...
[INSTA...
[Pantal...
Mozill...
Desca...


LockFile.cpp (~/qpidc-0.5/src/qpid/sys/posix) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Nuevo Abrir Guardar Imprimir... Deshacer Rehacer Cortar Copiar Pegar
INSTALL
LockFile.cpp
impl.reset(new LockFilePrivate(fd));
}
LockFile::~LockFile() {
int basura:
if (impl) {
int f = impl->fd;
if (f >= 0) {
basura = (void) ::lockf(f, F_ULOCK, 0); // Suppress warnings about ignoring return value.
::close(f);
impl->fd = -1;
}
}
}
pid_t LockFile::readPid(void) const {
if (!impl)
throw Exception("Lock file not open");
pid_t pid;
C++
Ancho de la tabulación: 8
Ln 60, Col 10
INS


LockFile.cpp (~/qpidc-0.5/src/qpid/sys/posix) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Nuevo Abrir Guardar Imprimir... Deshacer Rehacer Cortar Copiar Pegar
INSTALL
LockFile.cpp
impl.reset(new LockFilePrivate(fd));
}
LockFile::~LockFile() {
int basura;
if (impl) {
int f = impl->fd;
if (f >= 0) {
basura =::lockf(f, F_ULOCK, 0); // Suppress warnings about
ignoring return value.
::close(f);
impl->fd = -1;
}
}
}
pid_t LockFile::readPid(void) const {
if (!impl)
throw Exception("Lock file not open");
pid_t pid;
C++
Ancho de la tabulación: 8
Ln 60, Col 23
INS


agent - Navegador de archivos
Archivo Editar Ver Ir Marcadores Solapas Ayuda
Atrás Adelante Subir Detener Recargar Carpeta personal Equipo Buscar
produccion qpidc-0.5 src qpid agent
100%
Vista de icono
Lugares
produccion
Escritorio
Sistema de archi...
Red
Soporte de ...
Papelera
Documentos
Música
Imágenes
ManagementAgent.h
ManagementAgentImpl.cpp
ManagementAgentImpl.h
ManagementAgentImpl.lotT
agent ...
produc...
[qpidc-...
[LockF...
[Mozill...
[Desc...
[sirad ...



The modified version of QPID source code can be compiled for UBUNTU 9. 04