eXtremeTolerance

Final Report

by Stefano Andreani

Computer Networks

Prof. Antonio Corradi

July 2001

Abstract

This work is a research on potential of applying spontaneous network concepts to a real enterprise computing problem. It uses an original approach to traditional load balancing and fault tolerance problems, adopting the metaphor of servers as autonomous agents aggregating each other and providing web services into a network.

Duties Specification

Target

The aim of this project is to realize a framework allowing distribution of web services within a network of server nodes providing support for spontaneous service join and discovery and for server nodes' fault tolerance.

Functional Requirements

The project must meet the following requirements:

- preserve client-server interaction state in case of server failure and reroute client requests to an available server which can access to that state
- use transparency from clients about server architecture, even in case of server failure (assume clients are simple web browsers)
- easy integration of industrial standard web services into the framework, requiring code changes a little as possible
- allow easy plugging in of different load balancing policies
- zero administration: services can be added to the cluster (or leave it) as performance needs change without requiring system reconfiguration

Analysis

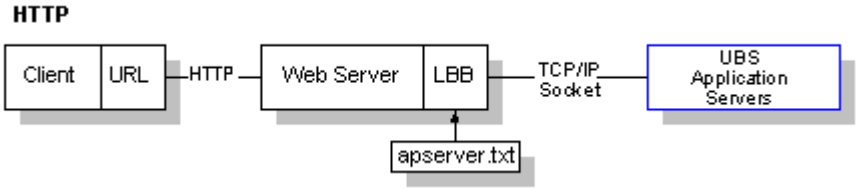
- clients are web browsers, so they have limited capabilities of state management . This implies that interaction state must be preserved by servers
- transparency means that if a server fails the client haven't to change its behavior to achieve fault tolerance (it can't be subject to an explicit redirection to another server, for example)
- in case of web services, interaction state can be represented simply by a session, added with a small amount of additional data
- easy integration of existent services means the framework must choose a component technology; requirements don't specify which industrial standard the framework have to support, making this a free design choice. A parameter to consider in this choice will be wide market availability of such components and easiness of integration.

Review of market solutions

Major application servers implement load balancing and fault tolerance. This short review describes these features with some implementation details, but doesn't provide performance or availability tests, and is limited to J2EE compliant servers, for their broad diffusion and functional homogeneity.

HP Bluestone Total-e-Server 7.3

HP Bluestone Total-e-Server provides some load balancing and fault tolerance features. A Load Balance Broker (LBB), a component that receives client requests through an attached web server and chooses the destination server with a nearly round-robin policy, handles load balancing. It supports a sort of Quality of Service reserving some servers to high priority clients. System administrators must specify available services and their priority level into a configuration file (apserver.txt) each time a new server is added to the cluster. If an application needs to keep user interaction state, then administrator must include a "session affinity" flag into config file to specify that following requests from the same client must be sent to the same server. This is achieved with a cookie stored in client's browser containing destination server address.
In case of server failure a new session cookie is assigned to the client and the request forwarded to another server. The application needs to be able to manage failover, since all session information will be lost. The most common way to implement failover in this environment is by using a persistent state server, i.e. a server that handles application requests to store objects into a database (using JDBC).



BEA WebLogic 6.1

BEA WebLogic uses the Cluster metaphor to manage server nodes. A WebLogic Server cluster is a group of servers that work together to provide a more scalable, more reliable application platform than a single server. A cluster appears to its clients as a single server but is in fact a group of servers acting as one. A cluster provides two key features above a single server: scalability and high availability. HTTP session state clustering and object clustering are the two primary cluster
services that WebLogic Server provides. We will focus on Http session state clustering since it is the main interest for this project. WebLogic Server instances in a cluster communicate using IP multicast for all-to-many messages and sockets for peer-to-peer communications. All server nodes use multicast to announce availability of new services and to advise with "heartbeat" messages their vitality.
To support automatic failover for servlet HTTP session states, WebLogic Server replicates the session state object in memory. This process creates a primary session state, which resides on the WebLogic Server to which the client first connects (A) and a secondary replica of the session state on another WebLogic Server instance in the cluster (B). As HP Bluestone, WebLogic Server also provides the ability to maintain the HTTP session state of a servlet using file-based or JDBC-based persistence.
WebLogic supports hardware and software load balancing. Software load balancing is handled by a proxy available as a plug-in of major web servers or as a standalone servlet. The load balancer interprets an identifier in the client's cookie to maintain the relationship between the client and the primary WebLogic Server hosting the HTTP session state.
After a failure on server A, WebLogic Server B becomes the primary server hosting the servlet session state, and a new secondary is created. In the HTTP response, the proxy updates the client's cookie to reflect the new primary and secondary servers, to account for the possibility of subsequent failovers.

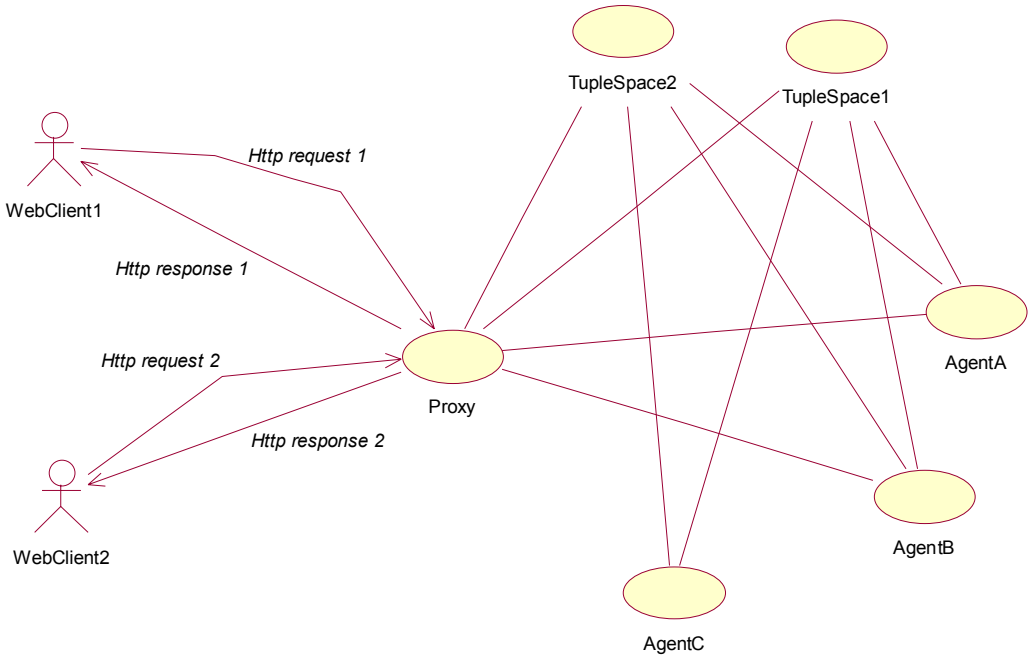
Design

System Architecture

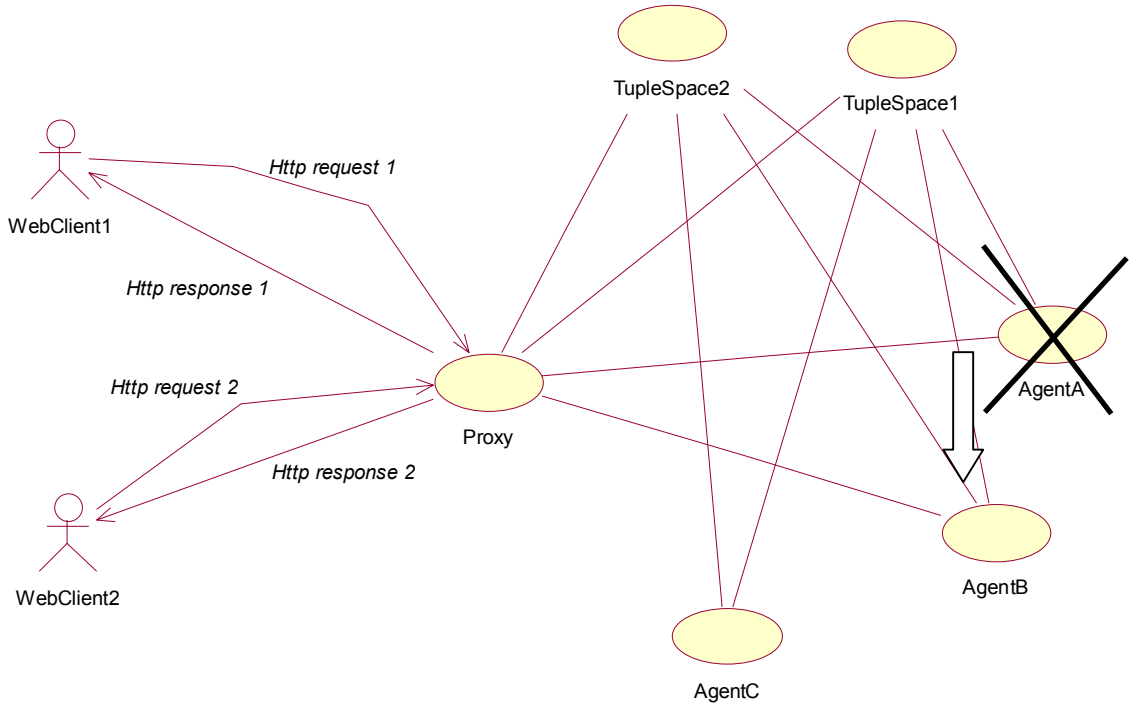
Our system needs a proxy to handle client requests and switch them to available servers. A first design choice is about where such a proxy could fit into the system architecture. As clearly explained by requirements, clients are simple browsers accessing to web services. Transparency requirement implies that we can't achieve load balancing or fault tolerance using a client side proxy (a plug-in for a specific browser, for example), so we need to manage client requests with a proxy server.

Project task could be accomplished with a farm of server agents accessing to a common session space. Each server maintains a partition of this state as a cache, to improve performances, but every change to the local cache must be propagated to the common space.

This architecture could avoid a single point of failure problem implementing the proxy as a simple cache and including all interaction states (sessions and client-server relationships) into one or more (replicated) persistent tuple spaces.



When a server node fails all sessions handled by this server are distributed to other available servers updating client-server bindings into the common session space and into the proxy cache, and copying each session into the selected server's cache.



Technological Choices

The first technological choice we have to make is about server architecture to support. We recall from project specifications that we have to build a framework for load balancing and fail over support of market widespread web services, so that we could reuse available code. A good solution could be to implement these features on top of the Servlet interfaces, to have at our disposal a wide number of developed services and a lot of application servers following those specifications. This is our (a bit arbitrary) choice.

A Servlet system provides a way to handle http requests and to store interaction state, since http is a stateless protocol. Servlets store user state into objects called `HttpSession`s. These objects act as hashtables, storing key/value pairs into their memory. Tuple spaces must deal with such kind of objects.

Requirements call for easy administration of services and scalability. This could be achieved with an infrastructure allowing spontaneous federations of services that expose their properties (service name and type, address, below hardware performances, etc.) to a network as they are started. Such a technology is Jini, from Sun Microsystems, a new middleware software based on Java and RMI, which enable service discovery and lookup through a network. In a Jini system there are always one or more lookup services. Both clients and services need to locate a lookup service in order to bootstrap into the Jini network. Services that don't know anything about network topology use a multicast discovery protocol to find a lookup service. After locating it they publish themselves by registering their proxies with that service using the Join protocol. A client can use the same multicast discovery protocol to locate a lookup service and then it uses that service to locate registered services it needs.

Using Jini technology server nodes act as proactive agents offering their services to the network and managing in autonomy their cycle of life. Clients of these services simply registers to a lookup server that alert them when a new service is available or an old one disappears.

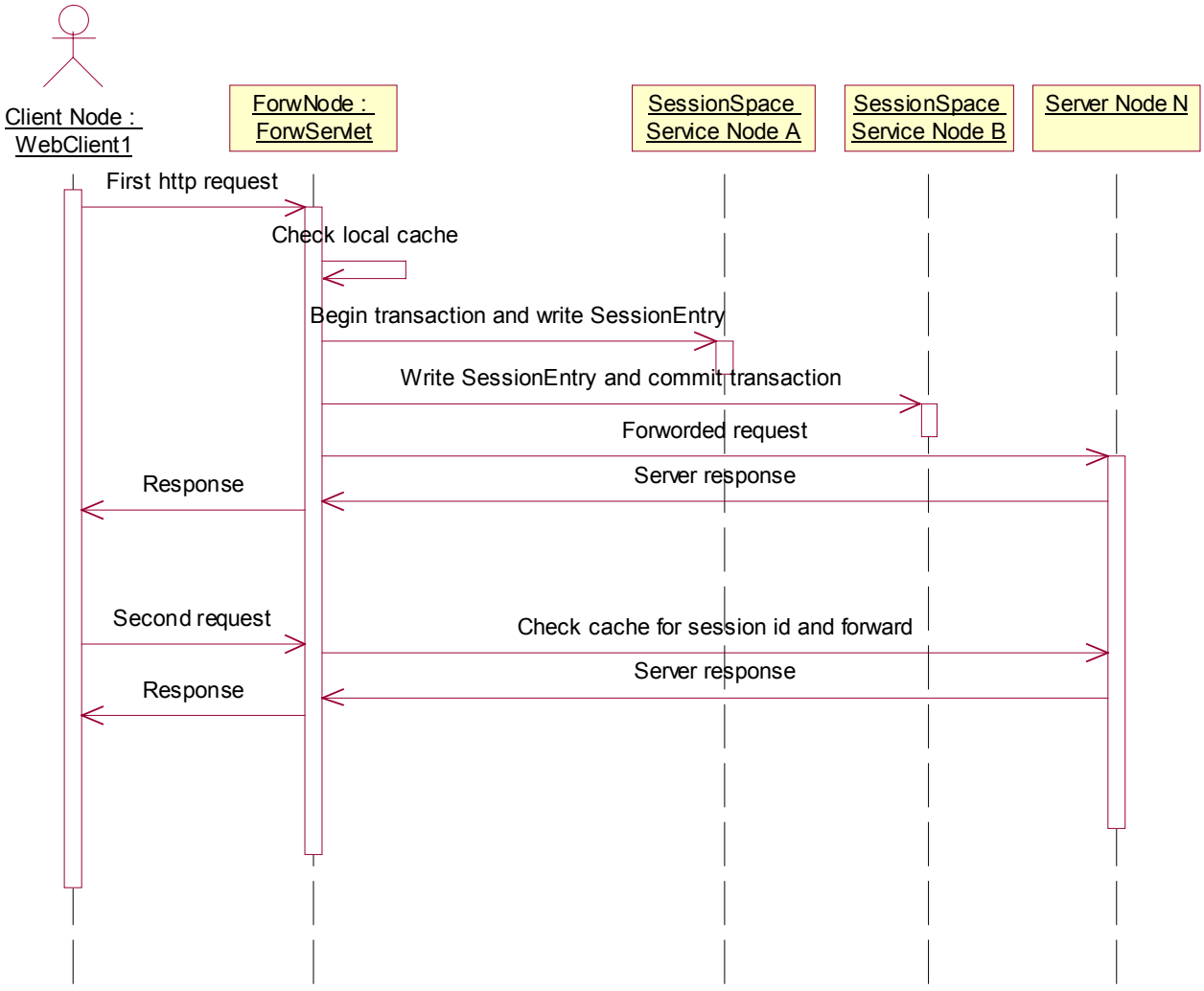
Moreover Jini offers as a service JavaSpace technology, an object oriented tuple space to manage and make persistent distributed objects. This seems to be the best choice for storing session objects. Other considered solutions were to use a DBMS or a custom common memory space, but were

rejected because they required an additional effort to develop those services JavaSpace provide by default.

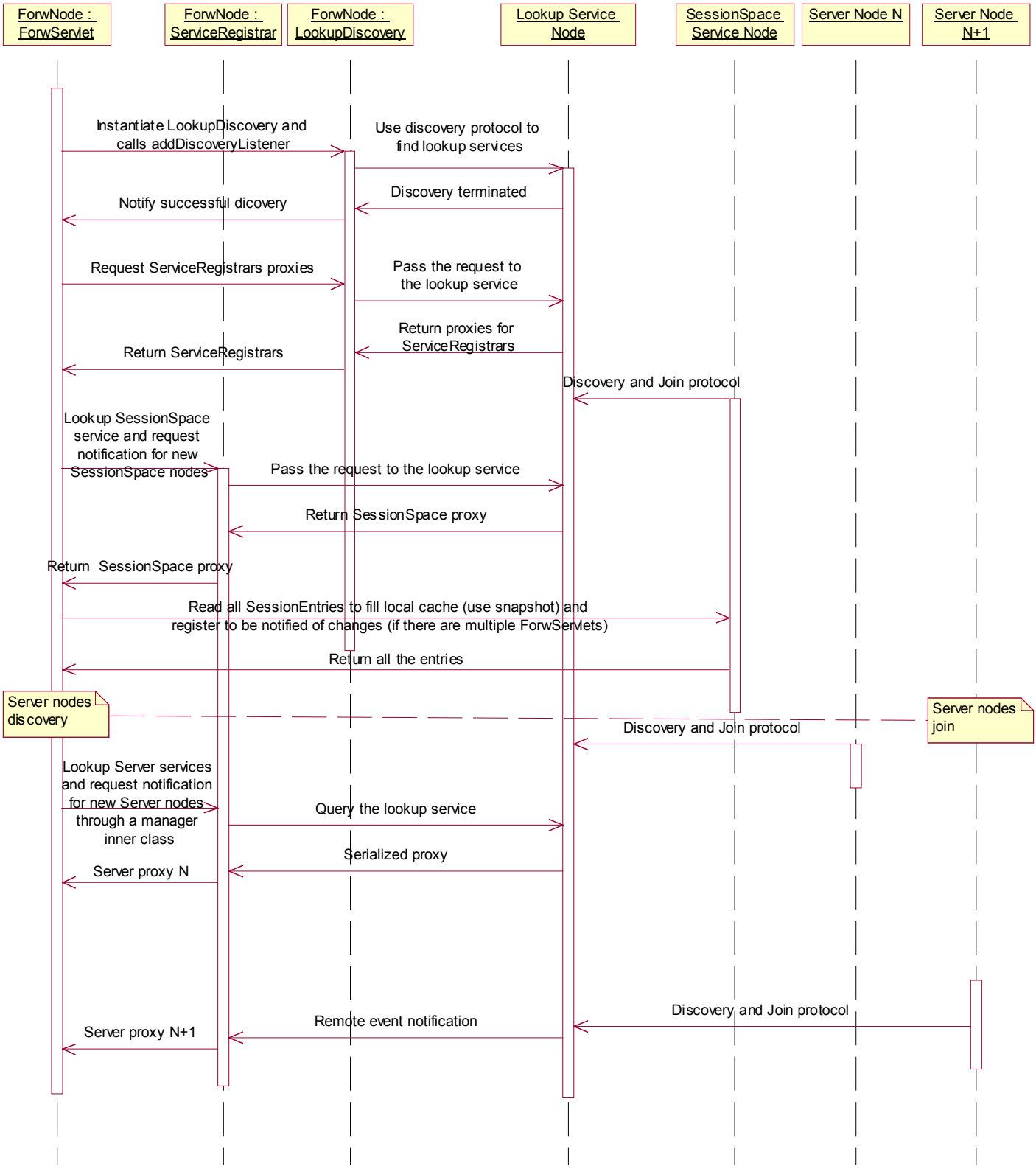
Interaction Protocols

In this section we present sequence diagrams to show various scenarios that can happen in our system as described above, delaying a detailed description to the test section.
High-level communication instruments are mainly cookies, because this is the preferred way to store and retrieve interaction state from a browser; internal protocols (between proxy and servers) use RMI below the Jini framework, even if cookies are used to pass client messages to servers.

Following diagram shows a simple interaction between a client and the system in case of multiple Session Spaces.



This diagram shows interaction between the proxy and server agents using the Jini framework when the proxy starts up (no client request is shown):



We will examine a case of failure handling in the test section of this report, because an example can help understanding that more complex situation.

Implementation

The design phase suggests the following package division:
`com.andreani.xtol.node:` contains classes needed by a server node
`com.andreani.xtol.proxy:` classes that execute request forwarding and server management
`com.andreani.xtol.test:` simple classes to test the application
`com.andreani.xtol.util:` common helper classes used by both proxy and node packages

A first implementation issue is about failures identification. A Jini system offers an automated way to identify failures: each service registers itself into the lookup service (it joins the network, using Jini terminology) for a limited amount of time, and obtains a Lease valid for that period. It is a sort of the heartbeat feature of WebLogic, because each server agent has to transmit its vital status to the other agents (through the lookup service). This is a good feature for our system, but it isn't enough to achieve a real time fail tolerance: a client can't wait for lease expiration, but needs an immediate substitute service. We can't reduce lease time to seconds (or milliseconds), because this will waste a lot of bandwidth and degrade network performances. Then we have to implement another mechanism to handle failures. This is rather simple: when the proxy connection to destination server fails, then that server is taken away from list of available servers and client request passed to another agent. However leases are useful to handle cases of failures of servers that aren't managing client requests for a while, and especially in case of servers that start to work again after a temporary failure (for example for network problems): after lease expiration the server will try to renew its lease till it will be accessible again, so our server manager doesn't need to monitor status of all servers: they act as agents joining and leaving the network autonomously.
To implement discovery and join protocols we can use new helper classes provided by Jini SDK 1.1 (they are part of the Jini specification), in particular JoinManager for agents who provide services to the network, and ServiceDiscoveryManager for agents' clients (SessionSpaceManager and ServerManager). JoinManager is used by server to register them into the lookup service and to manage Leases. ServiceDiscoveryManager is used by SessionSpaceManager to discover JavaSpaces as they appear into the network, and by ServerManager to take a cache of available servers and distribute requests to them.
When a client request reach a server, local session state is updated by `HttpSession`'s method `addAttribute()` (for example when a customer put a good into his shopping cart). To be sure that also global session is updated () we must handle sessions changes. Fortunately the last Servlet specification provides a method to handle session operations: it is sufficient to implement HttpSessionAttributeListener interface and listen for session changes. Class SessionEventHandler deals with this work[1], intercepting any change made to a Session and propagating it to all JavaSpaces (in an asynchronous way, so a client doesn't need to wait for state propagation).
We can take a look to the ServerNode class to show the usual structure of a Jini object: there are standard methods to allow easy lookup of services. For example, if the required service was "printing" and a client wants to print a document with the nearest printer, an intelligent lookup service could use location entries to provide requested service to the client. In our case the Server

[1] Since this is a very early feature it is implemented only by a beta version of the Sun's reference implementation (Tomcat 4.0 beta 5), and this makes impossible to execute tests on available Servlet applications (as I would), like the famous Java PetShop, because these rely on older stable version of servlet engines.

Manager could use specific server information (I/O and CPU performances) to distribute I/O bound and CPU bound requests to the best server.

```
public class ServerNode implements java.io.Serializable
{
   /**
    * IP address of the server
    */
   public String serverIP;
   /**
    * Port number associated to the service
    */
   public Integer serverPort;
   /**
    * Actual load of this server. It could be useful to realize intelligent
    * load balancing policies.
    */
   public Double serverLoad;
   final static String SERVICE_NAME = "Jini Service - Server Node";
   final static String SERVICE_MFR = "Stefano Andreani";
   final static String SERVICE_VENDOR = "DEIS";
   final static String SERVICE_VERSION = "1.0";
   final static String SERVICE_COMMENT = "This entity represent a server node.";
   final static String LOCATION_FLOOR = "second";
   final static String LOCATION_ROOM = "lab2";
   final static String LOCATION_BUILDING = "DEIS building";
   final static String CLASS_NAME = "serverNode";

   /**
    * Constructor for the ServerNode object
    *
    * @param ip    IP address
    * @param port  Port number
    * @param load  Actual load of the server
    */
   public ServerNode(String ip, int port, double load)
   {
      serverIP = ip;
      serverPort = new Integer(port);
      serverLoad = new Double(load);
   }

   /**
    * Gets the Entries attribute of the ServerNode object. This method returns
    * standard fields used by Jini services to lookup a service.
    *
    * @return   The Entries value
    */
   public Entry[] getEntries()
   {
      Entry myEntry[] = new Entry[4];
      myEntry[0] = new ServiceInfo(SERVICE_NAME,SERVICE_MFR,SERVICE_VENDOR,
        SERVICE_VERSION, "", "");
      myEntry[1] = new Name(CLASS_NAME);
      myEntry[2] = new Comment(SERVICE_COMMENT);
      myEntry[3] = new Location(LOCATION_FLOOR, LOCATION_ROOM, LOCATION_BUILDING);
      return myEntry;
   }
}
```

SessionEntry is the Entry object contained into Session Spaces that stores user state and his bound server address. Jini specs require each entry class must be serializable and must have public fields which values are assigned by the constructor.

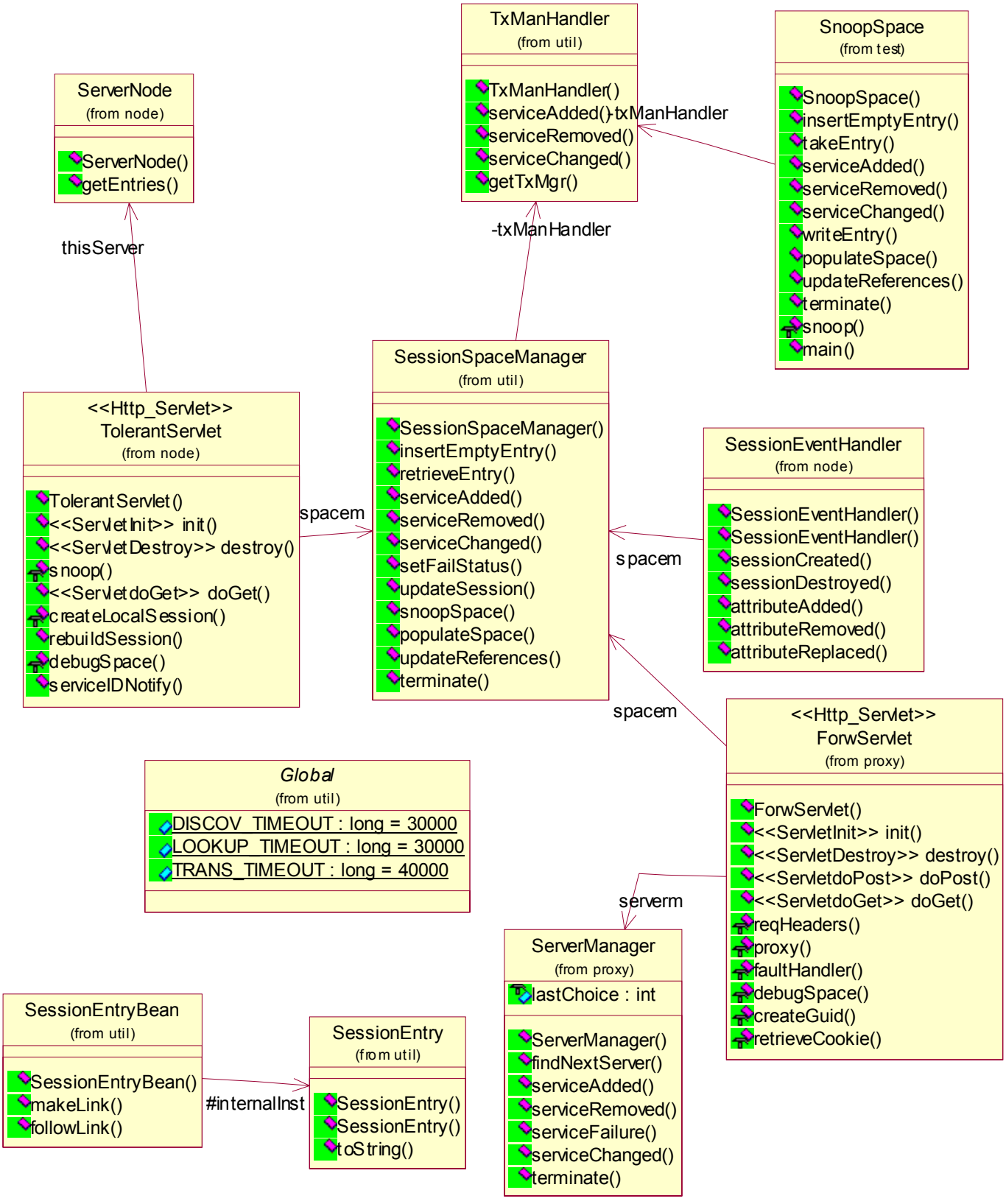
```
public class SessionEntry extends AbstractEntry
{
   /**
    * This is the Global Unique Id that identify user session.
    */
   public String id;
   /**
    * The address of destination server
    */
   public String serverAddr;
   /**
    * This Hashtable contains all objects representing user state.
    */
   public Hashtable session;
   /**
    * Fail is true if the server which was servicing has failed and the new one
    * assigned hasn't yet copied this session into its cache.
    */
   public Boolean fail;

   /**
    * Empty Constructor for the SessionEntry object
    */
   public SessionEntry()
   {
   }

   /**
    * Constructor for the SessionEntry object
    *
    * @param id_p          Session GUID
    * @param serverAddr_p  IP and port
    * @param obj           user state
    * @param fail_p        fail flag
    */
   public SessionEntry(String id_p, String serverAddr_p, Hashtable obj, Boolean fail_p)
   {
      id = id_p;
      serverAddr = serverAddr_p;
      session = obj;
      fail = fail_p;
   }
}
```

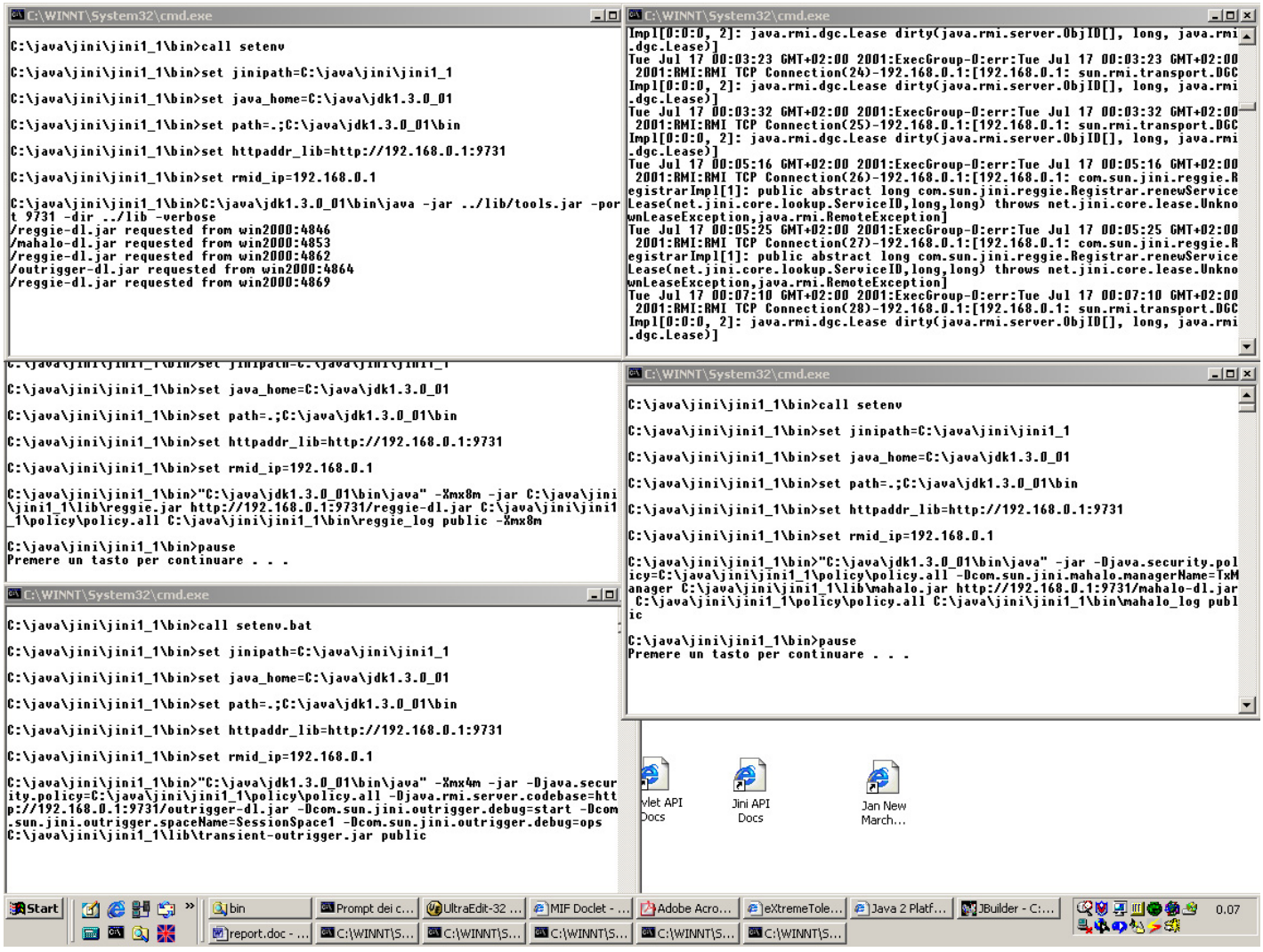
The `com.andreani.xtol.node` package contains the class `TolerantServlet`. This class realize the agent that join the network and offer to it its services. Is is a simple Servlet that uses its `init()` method to make it visible as service into the network. A developer who wants to load balance accesses to his server and make it fail tolerant must subclass this class and call `super.init()` and `super.doGet()` at the beginning of these local methods.
Easy plugging in of a sophisticated load balancing policy is assured by the structure of the framework: a developer who wants to change such policy needs to subclass the server manager and override only the `findNextServer()` method.

The UML diagram that follows shows most important classes involved in the project and their mutual relationships:



Test Results

Before testing the system we need to start services required by Jini: first of all we start an http server to satisfy requests of java classes not available in local classpaths (we can choose any IP address accessible from other nodes). The second server we need to start is the RMI daemon, then the lookup service and the transaction manager services, which register themselves both as activable objects through the RMI daemon. The last service needed to setup our Jini environment is the JavaSpace service. Sun's implementation provides both transient and persistent versions of JavaSpace, we can use the first during debugging and the second one in production, to increase fail tolerance of our system. Since services uses multicast to discover each other, we can start each service on a different node of a local network. It is also possible to use unicast to locate a server, allowing a variety of network configurations.



After start up of these basic services we are ready to test our application: we can start the proxy and observe how two server nodes are registered by the server manager as they appear on the network. We executed tests on a lot of configurations, but images printed in this paper refer to a network of two Pentium II 350 with 256 Mb of RAM connected through a 100 Mbit network. Numbers on the left column are milliseconds.

With the mentioned configuration a server takes about 6 seconds to start and registers into the

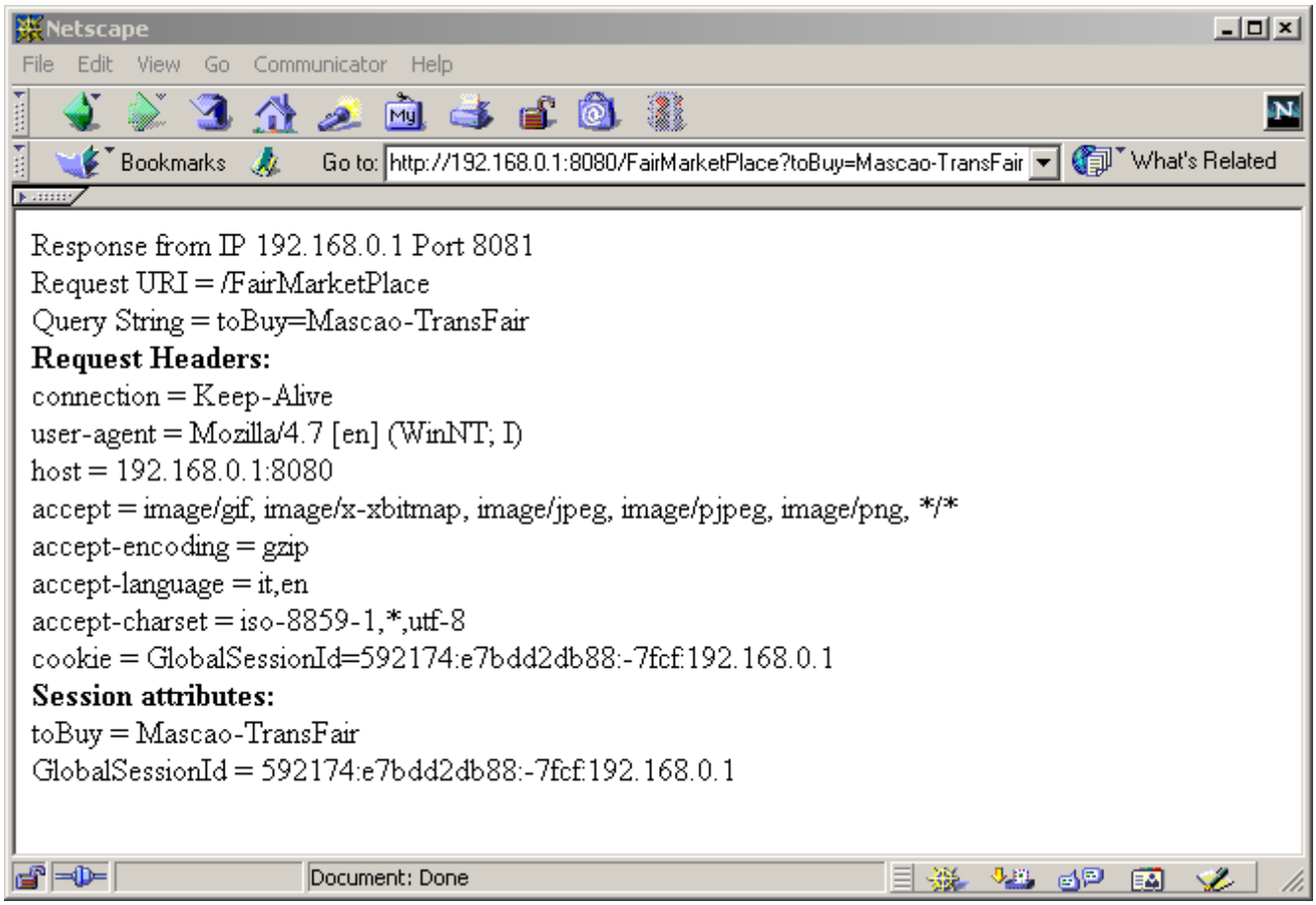
```
Apache Tomcat/4.0-b5
0 [main] DEBUG com.andreani.xtol.SessionEventHandler  - constructor
191 [main] DEBUG com.andreani.xtol.SessionSpaceManager  - contructor
241 [main] DEBUG com.andreani.xtol.TxManHandler  - contructor
241 [main] DEBUG com.andreani.xtol.TxManHandler  - new LookupDiscoveryManager
932 [main] DEBUG com.andreani.xtol.TxManHandler  - new ServiceDiscoveryManager
1032 [main] DEBUG com.andreani.xtol.TxManHandler  - new ServiceTemplate
2464 [main] DEBUG com.andreani.xtol.TxManHandler  - txItem.service=com.sun.jini.
mahalo.TxnManagerImpl_Stub[RemoteStub [ref: sun.rmi.server.UnicastRef2@5256fa]]
2504 [main] DEBUG com.andreani.xtol.SessionSpaceManager  - Creating LookupCache
2784 [main] DEBUG com.andreani.xtol.TolerantServlet  - contructor
2784 [main] DEBUG com.andreani.xtol.SessionSpaceManager  - contructor
2784 [main] DEBUG com.andreani.xtol.TxManHandler  - contructor
2784 [main] DEBUG com.andreani.xtol.TxManHandler  - new LookupDiscoveryManager
2794 [main] DEBUG com.andreani.xtol.TxManHandler  - new ServiceDiscoveryManager
2794 [main] DEBUG com.andreani.xtol.TxManHandler  - new ServiceTemplate
3225 [main] DEBUG com.andreani.xtol.TxManHandler  - txItem.service=com.sun.jini.
mahalo.TxnManagerImpl_Stub[RemoteStub [ref: sun.rmi.server.UnicastRef2@53f67e]]
3225 [main] DEBUG com.andreani.xtol.SessionSpaceManager  - Creating LookupCache
3235 [main] DEBUG com.andreani.xtol.TolerantServlet  - init()
Registering server with lookup service...
3255 [main] DEBUG com.andreani.xtol.TolerantServlet  - Local address=192.168.0.1

3255 [main] DEBUG com.andreani.xtol.TolerantServlet  - port...
3265 [main] DEBUG com.andreani.xtol.TolerantServlet  - port=8081
3275 [main] DEBUG com.andreani.xtol.TolerantServlet  - load=0.23400698297241174
3435 [main] DEBUG com.andreani.xtol.TolerantServlet  - new JoinManager
Registering this service...: jman=net.jini.lookup.JoinManager@b3364
3986 [task thread] DEBUG com.andreani.xtol.SessionSpaceManager  - serviceAdded
6409 [task thread] DEBUG com.andreani.xtol.SessionSpaceManager  - serviceAdded
6550 [taskThread] INFO com.andreani.xtol.TolerantServlet  - Registered with Serv
ice ID: 0190bfba-83ca-459b-988f-a1c380f7c81a
```

network

```
Starting service Tomcat-Standalone
Apache Tomcat/4.0-b5
0 [main] DEBUG com.andreani.xtol.ForwServlet  - contructor
20 [main] DEBUG com.andreani.xtol.ForwServlet  - ForwServlet inizialization
170 [main] DEBUG com.andreani.xtol.ServerManager  - constructor
2544 [main] DEBUG com.andreani.xtol.SessionSpaceManager  - contructor
2664 [main] DEBUG com.andreani.xtol.TxManHandler  - contructor
2664 [main] DEBUG com.andreani.xtol.TxManHandler  - new LookupDiscoveryManager
2664 [main] DEBUG com.andreani.xtol.TxManHandler  - new ServiceDiscoveryManager
2684 [main] DEBUG com.andreani.xtol.TxManHandler  - new ServiceTemplate
5087 [main] DEBUG com.andreani.xtol.TxManHandler  - txItem.service=com.sun.jini.
mahalo.TxnManagerImpl_Stub[RemoteStub [ref: sun.rmi.server.UnicastRef2@766cff]]
5118 [main] DEBUG com.andreani.xtol.SessionSpaceManager  - Creating LookupCache
8102 [task thread] DEBUG com.andreani.xtol.SessionSpaceManager  - serviceAdded
38075 [task thread] DEBUG com.andreani.xtol.ServerManager  - Service Added: Serv
erNodeAddr=192.168.0.1:8081
38075 [task thread] DEBUG com.andreani.xtol.ServerManager  - Registered services
: 1
77622 [task thread] DEBUG com.andreani.xtol.ServerManager  - Service Added: Serv
erNodeAddr=192.168.0.2:8081
77622 [task thread] DEBUG com.andreani.xtol.ServerManager  - Registered services
: 2
```

In our tests we simulate a market place scenario where a customer is buying some products. The first request is forwarded to the server A (192.168.0.1), that stores user choice (a chocolate bar) into local session and copies this item into the common Session Space.



After the first request we simulate a server failure pressing Control+C into the server A window. The Server Manager doesn't notice this problem till another client request bound to server A arrives or the Lease of server A expires. When the second client request comes the proxy tries to forward it to A, but the TCP layer signals a connection error. This error is correctly interpreted as a failure, so the Server Manager removes that server from the list of available ones and the Session Space Manager updates all the references to server A distributing sessions to other servers. Client's session is assigned to server B (IP 192.168.0.2), the only one available now in our test configuration, so the proxy connects to that server and passes to it client's request and cookies. B notices that it hasn't a local session corresponding to that client (because the JSESSION cookie passed from the client doesn't correspond to an existent local session), but the client owns a global session (shown by the GlobalSessionID cookie), so B tries to recover corresponding session from the Common Session Space and create a new local session. This protocol, as discussed above, assures transparency to server failures and is rather lightweight because session copies from the common space to each server happen only when a client request that was associated to a failed server arrives to a new server instead of handling recovery after failure verification.

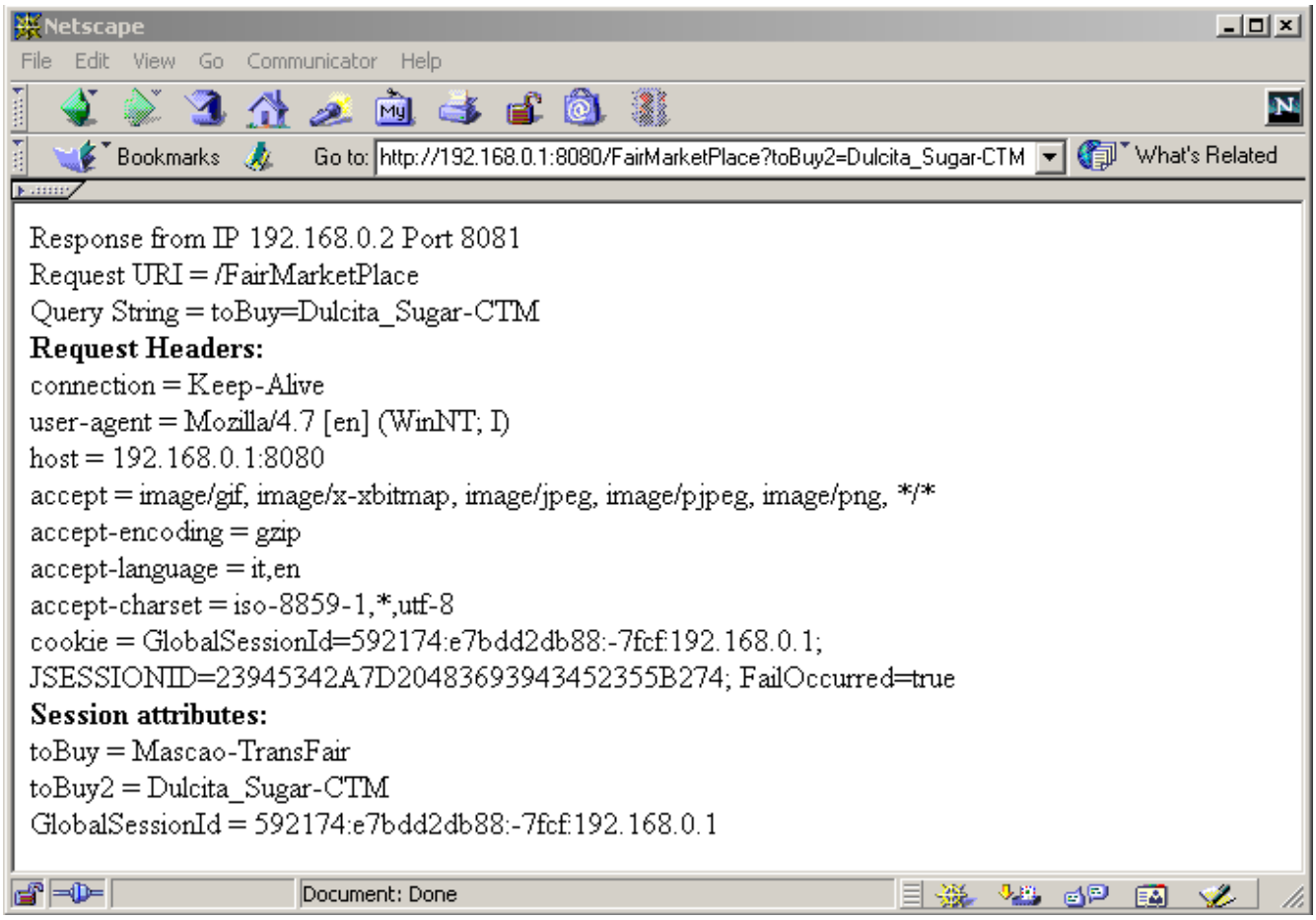
```
3626345 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ForwServlet  - proxy()
3626555 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ForwServlet  - clientCo
okies=GlobalSessionId=592174:e7bdd2db88:-7fcf:192.168.0.1; JSESSIONID=23945342A7
D20483693943452355B274
3626555 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ForwServlet  - url.open
Connection();
3626575 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ForwServlet  - con.conn
ect();
3627566 [HttpProcessor[8080][4]] INFO com.andreani.xtol.ForwServlet  - Connectio
n to 192.168.0.1:8082 failed: looking for an available server.
3627566 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ServerManager  - Sservi
ceFailure: failedAddr=192.168.0.1:8081
3627586 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ServerManager  - Regist
ered services: 1
3627586 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ServerManager  - findNe
xtServer
3627586 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ServerManager  - lastCh
oice=2 serverv.size()=1
3627727 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.SessionSpaceManager  -
updateReferences()
3627787 [HttpProcessor[8080][4]] INFO com.andreani.xtol.SessionSpaceManager  - O
ld Entry to update:

Session id=592174:e7bdd2db88:-7fcf:192.168.0.1#
serverAddr=192.168.0.1:8081#
fail flag=false
Session attributes:
GlobalSessionId=592174:e7bdd2db88:-7fcf:192.168.0.1
toBuy=Mascao-TransFair

3627847 [HttpProcessor[8080][4]] INFO com.andreani.xtol.SessionSpaceManager  - U
pdated Entry:

Session id=592174:e7bdd2db88:-7fcf:192.168.0.1#
serverAddr=192.168.0.2:8081#
fail flag=true
Session attributes:
GlobalSessionId=592174:e7bdd2db88:-7fcf:192.168.0.1
toBuy=Mascao-TransFair

3628027 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ForwServlet  - url.open
Connection();
3628027 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ForwServlet  - con.setR
equestProperty()
3628037 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ForwServlet  - con.conn
ect();
3628077 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ForwServlet  - done:con
=http://192.168.0.2:8081/FairMarketPlace?toBuy2=Dulcita_Sugar-CTM
3631141 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ForwServlet  - Set-Cook
ie: JSESSIONID=1719924522A3366522F35435D17294C5;Path=/
3876745 [HttpProcessor[8080][4]] DEBUG com.andreani.xtol.ForwServlet  - con.disc
onnect();
```



Performance tests are not available for this project because it would be an excessive effort to simulate the behavior of a lot of web browser (each one with its own session cookie) making requests and to make a significant measurement in case of partial failure.

Limits and Possible Enhancements

The implemented framework doesn't handle fail tolerance for Enterprise Beans: the project could be enhanced with methods to enable Session Beans migration as done for http sessions, but it is over the scope of this project.
Finally we can observe that this research shows as Jini technology is suited not only to appliances market, as the community initially supposed, by can be of great help in enterprise computing, providing new instruments to enhance system availability and scalability, without requiring great administration efforts.

References

K. Birman, Building Secure and Reliable Network Applications – [PDF available in his site], 1995
Sing Li, Professional Jini – Wrox, 2000
Allamaraju et al., Professional Java Server Programming J2EE Edition- Wrox, 2000
HP Bluestone site: http://www.bluestone.com
BEA documentation site: http://edocs.bea.com
Jan Newmarch's Guide to JINI Technologies:
http://pandonia.canberra.edu.au/java/jini/tutorial/Jini.html

eXtremeTolerance API

Version 1.0

Contents

Overview

Package Summary	
Packages	
com.andreani.xtol.node 5	The ***eXtremeTolerance*** node package.
com.andreani.xtol.proxy 15	The ***eXtremeTolerance*** proxy package.
com.andreani.xtol.test 23	The ***eXtremeTolerance*** test package.
com.andreani.xtol.util 27	The ***eXtremeTolerance*** helper classes package.

Package
com.andreani.xtol.node

Description
The ***eXtremeTolerance*** node package. This package contains classes necessary for server nodes.

Class Summary	
Classes	
ServerNode 6	This class represents a server and is used to register an available server into the lookup service.
SessionEventHandler 8	This class must be instantiated by the servlet container.
TolerantServlet 11	This class realize the agent that joins the network and offers to it its services.

com.andreani.xtol.node
ServerNode

Declaration

```
public class ServerNode implements java.io.Serializable
```

```
java.lang.Object
  |
  +--com.andreani.xtol.node.ServerNode
```

All Implemented Interfaces: java.io.Serializable

Description

This class represents a server and is used to register an available server into the lookup service. This compiled class must be accesible from the lookup service in order to use the browser utility from Sun's Jini reference implementation. Simply put the class into reggie-dl.jar for Jini 1.1

Member Summary

Fields

public	`serverIP` 7 IP address of the server
public	`serverLoad` 7 Actual load of this server.
public	`serverPort` 7 Port number associated to the service

Constructors

public	`ServerNode(String, int, double)` 7 Constructor for the ServerNode object

Methods

public Entry	`getEntries()` 7 Gets the Entries attribute of the ServerNode object.

Inherited Member Summary

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

serverIP

```
public java.lang.String serverIP
```

IP address of the server

serverLoad

```
public java.lang.Double serverLoad
```

Actual load of this server. It could be useful to realize intelligent load balancing policies.

serverPort

```
public java.lang.Integer serverPort
```

Port number associated to the service

Constructors

ServerNode(String, int, double)

```
public ServerNode(java.lang.String ip, int port, double load)
```

Constructor for the ServerNode object

Parameters:

`ip` - IP address

`port` - Port number

`load` - Actual load of the server

Methods

getEntries()

```
public net.jini.core.entry.Entry[] getEntries()
```

Gets the Entries attribute of the ServerNode object. This method returns standard fields used by Jini services to lookup a service.

Returns: The Entries value

com.andreani.xtol.node
SessionEventHandler

Declaration

```
public class SessionEventHandler implements javax.servlet.http.HttpSessionListener,
    javax.servlet.http.HttpSessionAttributeListener

java.lang.Object
  |
  +--com.andreani.xtol.node.SessionEventHandler
```

All Implemented Interfaces: java.util.EventListener, javax.servlet.http.HttpSessionAttributeListener, javax.servlet.http.HttpSessionListener

Description

This class must be instantiated by the servlet container. To do so you can include the following code in the deployment descriptor:

```
<web-app>
<display-name>MyApplication</display-name>
<listener>
<listener-class>com.andreani.xtol.node.SessionEventHandler</listener-class>
</listener>
<servlet>
<display-name>MyServlet</display-name>
...etc
</servlet>
</web-app>
```

The container is required to complete instantiation of the listener classes in a web application prior to the start of execution of the first request into the application. The container must reference each listener instance until the last request is serviced for the web application.

Member Summary

Fields

protected	cat$_9$ Log handler (Log4Java)

Constructors

public	SessionEventHandler()$_9$ Constructor for the SessionEventHandler object
public	SessionEventHandler(ServiceRegistrar)$_9$ The constructor searches SessionSpace services through the lookup service and registers itself for notifications of new SessionSpace services.

Methods

public void	attributeAdded(HttpSessionBindingEvent)$_{10}$ When an attribute is added to a session this method propagates modifications to the SessionSpaces

Member Summary	
public void	attributeRemoved(HttpSessionBindingEvent)[10] When an attribute is removed from a session this method propagates modifications to the SessionSpaces
public void	attributeReplaced(HttpSessionBindingEvent)[10] When an attribute is replaced into a session this method propagates modifications to the SessionSpaces
public void	sessionCreated(HttpSessionEvent)[10] Description of the Method
public void	sessionDestroyed(HttpSessionEvent)[10]

Inherited Member Summary
Methods inherited from class java.lang.Object clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

cat

```
protected org.apache.log4j.Category cat
```

Log handler (Log4Java)

Constructors

SessionEventHandler()

```
public SessionEventHandler()
```

Constructor for the SessionEventHandler object

SessionEventHandler(ServiceRegistrar)

```
public SessionEventHandler(net.jini.core.lookup.ServiceRegistrar registrar)
```

The constructor searches SessionSpace services through the lookup service and registers itself for notifications of new SessionSpace services.

Parameters:

`registrar` - Description of Parameter

Methods

attributeAdded(HttpSessionBindingEvent)

```
public void attributeAdded(javax.servlet.http.HttpSessionBindingEvent se)
```

When an attribute is added to a session this method propagates modifications to the SessionSpaces

Specified By: attributeAdded(HttpSessionBindingEvent)$_{10}$ in interface SessionEventHandler$_{8}$

attributeRemoved(HttpSessionBindingEvent)

```
public void attributeRemoved(javax.servlet.http.HttpSessionBindingEvent se)
```

When an attribute is removed from a session this method propagates modifications to the SessionSpaces

Specified By: attributeRemoved(HttpSessionBindingEvent)$_{10}$ in interface SessionEventHandler$_{8}$

attributeReplaced(HttpSessionBindingEvent)

```
public void attributeReplaced(javax.servlet.http.HttpSessionBindingEvent se)
```

When an attribute is replaced into a session this method propagates modifications to the SessionSpaces

Specified By: attributeReplaced(HttpSessionBindingEvent)$_{10}$ in interface SessionEventHandler$_{8}$

sessionCreated(HttpSessionEvent)

```
public void sessionCreated(javax.servlet.http.HttpSessionEvent se)
```

Description of the Method

Specified By: sessionCreated(HttpSessionEvent)$_{10}$ in interface SessionEventHandler$_{8}$

Parameters:
`se` - Description of Parameter

sessionDestroyed(HttpSessionEvent)

```
public void sessionDestroyed(javax.servlet.http.HttpSessionEvent se)
```

Specified By: sessionDestroyed(HttpSessionEvent)$_{10}$ in interface SessionEventHandler$_{8}$

com.andreani.xtol.node
TolerantServlet

Declaration

```
public class TolerantServlet extends javax.servlet.http.HttpServlet implements
    net.jini.lookup.ServiceIDListener

java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
              |
              +--com.andreani.xtol.node.TolerantServlet
```

All Implemented Interfaces: java.util.EventListener, java.io.Serializable, net.jini.lookup.ServiceIDListener, javax.servlet.Servlet, javax.servlet.ServletConfig

Description

This class realize the agent that joins the network and offers to it its services. Is is a simple Servlet that uses its init() method to make it visible as service into the network. A developer who wants to load balance accesses to his server and make it fail tolerant must subclass this class and call super.init() and super.doGet() at the beginning of these local methods.

Member Summary

Fields

protected	cat[12]

Constructors

public	TolerantServlet()[12]

Methods

public void	destroy()[12]
public void	doGet(HttpServletRequest, HttpServletResponse)[12] Always invoke super.doGet(req,resp) as first operation when you override this method.
public void	init(ServletConfig)[13] Registers itself into the lookup service.
public void	rebuildSession(String, HttpServletRequest)[13] This method create a new HttpSession retrieving session data from a SessionSpace.
public void	serviceIDNotify(ServiceID)[13] Debbugging purpose

cat

Inherited Member Summary

Methods inherited from class javax.servlet.GenericServlet
`getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, log, log`
Methods inherited from class javax.servlet.http.HttpServlet
`doDelete, doHead, doOptions, doPost, doPut, doTrace, getLastModified, service, service`
Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

cat

```
protected org.apache.log4j.Category cat
```

Constructors

TolerantServlet()

```
public TolerantServlet()
```

Methods

destroy()

```
public void destroy()
```

Specified By: `destroy()`[12] in interface `TolerantServlet`[11]

Overrides: javax.servlet.GenericServlet.destroy() in class javax.servlet.GenericServlet

doGet(HttpServletRequest, HttpServletResponse)

```
public void doGet(javax.servlet.http.HttpServletRequest req,
            javax.servlet.http.HttpServletResponse resp)
            throws ServletException, IOException
```

Always invoke super.doGet(req,resp) as first operation when you override this method.

Overrides: javax.servlet.http.HttpServlet.doGet(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse) in class javax.servlet.http.HttpServlet

Throws:

`IOException`, `ServletException`

init(ServletConfig)

```
public void init(javax.servlet.ServletConfig sconf)
```

Registers itself into the lookup service. Always invoke super.init(ServletConfig) as first operation when you override this method.

Specified By: `init(ServletConfig)`$_{13}$ in interface `TolerantServlet`$_{11}$

Overrides: javax.servlet.GenericServlet.init(javax.servlet.ServletConfig) in class javax.servlet.GenericServlet

rebuildSession(String, HttpServletRequest)

```
public void rebuildSession(java.lang.String globsessid,
            javax.servlet.http.HttpServletRequest req)
```

This method create a new HttpSession retrieving session data from a SessionSpace. It is usually used if the session server failed.

serviceIDNotify(ServiceID)

```
public void serviceIDNotify(net.jini.core.lookup.ServiceID id)
```

Debbugging purpose

Specified By: `serviceIDNotify(ServiceID)`$_{13}$ in interface `TolerantServlet`$_{11}$

serviceIDNotify(ServiceID)

Package
com.andreani.xtol.proxy

Description
The ***eXtremeTolerance*** proxy package. It contains classes that execute request forwarding and server management.

Class Summary	
Classes	
ForwServlet 16	This is
ServerManager 19	This class manages the list of available servers.

com.andreani.xtol.proxy
ForwServlet

Declaration

```
public class ForwServlet extends javax.servlet.http.HttpServlet implements
    javax.servlet.SingleThreadModel

java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
              |
              +--com.andreani.xtol.proxy.ForwServlet
```

All Implemented Interfaces: java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig, javax.servlet.SingleThreadModel

Description

This is

Member Summary

Fields

protected	cat[17] Log handler (Log4Java)
protected	serverAddr[17] This local cache contains the mapping between global session id and destination server address of each client.

Constructors

public	ForwServlet()[17] Constructor for the ForwServlet object

Methods

public void	destroy()[17]
public void	doGet(HttpServletRequest, HttpServletResponse)[17]
public void	doPost(HttpServletRequest, HttpServletResponse)[18]
public void	init()[18] Servlet initializzation

Inherited Member Summary

Methods inherited from class javax.servlet.GenericServlet

getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, log, log

Inherited Member Summary

Methods inherited from class javax.servlet.http.HttpServlet
`doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service`
Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

cat

`protected org.apache.log4j.Category` **`cat`**

Log handler (Log4Java)

serverAddr

`protected java.util.Hashtable` **`serverAddr`**

This local cache contains the mapping between global session id and destination server address of each client.

Constructors

ForwServlet()

`public` **`ForwServlet`**`()`

Constructor for the ForwServlet object

Methods

destroy()

`public void` **`destroy`**`()`

Specified By: `destroy()`[17] in interface `ForwServlet`[16]

Overrides: javax.servlet.GenericServlet.destroy() in class javax.servlet.GenericServlet

doGet(HttpServletRequest, HttpServletResponse)

```
public void doGet(javax.servlet.http.HttpServletRequest req,
            javax.servlet.http.HttpServletResponse resp)
            throws ServletException, IOException
```

Overrides: javax.servlet.http.HttpServlet.doGet(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse) in class javax.servlet.http.HttpServlet

doPost(HttpServletRequest, HttpServletResponse)

Throws:
`IOException`, `ServletException`

doPost(HttpServletRequest, HttpServletResponse)

```
public void doPost(javax.servlet.http.HttpServletRequest req,
            javax.servlet.http.HttpServletResponse resp)
            throws ServletException, IOException
```

Overrides: javax.servlet.http.HttpServlet.doPost(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse) in class javax.servlet.http.HttpServlet

Throws:
`IOException`, `ServletException`

init()

```
public void init()
            throws ServletException
```

Servlet initializzation

Overrides: javax.servlet.GenericServlet.init() in class javax.servlet.GenericServlet

Throws:
`ServletException` - Exception thrown by init

com.andreani.xtol.proxy
ServerManager

Declaration

```
public class ServerManager implements net.jini.lookup.ServiceDiscoveryListener

java.lang.Object
  |
  +--com.andreani.xtol.proxy.ServerManager
```

All Implemented Interfaces: net.jini.lookup.ServiceDiscoveryListener

Description

This class manages the list of available servers.

Member Summary

Fields

protected	`cat`[20] Log handler (Log4Java)

Constructors

public	`ServerManager()`[20] Constructor for the ServerManager object

Methods

public synchronized String	`findNextServer()`[20] This method returns the address of an available server.
public synchronized void	`serviceAdded(ServiceDiscoveryEvent)`[20] Method used by the ServiceDiscoveryListener interface to signal the presence of a new server.
public synchronized void	`serviceChanged(ServiceDiscoveryEvent)`[20] Method used by the ServiceDiscoveryListener interface to signal the change of a server.
public synchronized void	`serviceFailure(String)`[21] Method invoked by the proxy if it noticed a server is not accessible
public synchronized void	`serviceRemoved(ServiceDiscoveryEvent)`[21] Method used by the ServiceDiscoveryListener interface to signal the leaving of a server.
public void	`terminate()`[21] Clear termination of the manager

Inherited Member Summary

Methods inherited from class java.lang.Object

Inherited Member Summary
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

cat

```
protected org.apache.log4j.Category cat
```

Log handler (Log4Java)

Constructors

ServerManager()

```
public ServerManager()
```

Constructor for the ServerManager object

Methods

findNextServer()

```
public synchronized java.lang.String findNextServer()
```

This method returns the address of an available server. It follows a round robin rule. This is the method to override if you want to change manager policy.

Returns: address of the next server to invoke

serviceAdded(ServiceDiscoveryEvent)

```
public synchronized void serviceAdded(net.jini.lookup.ServiceDiscoveryEvent evt)
```

Method used by the ServiceDiscoveryListener interface to signal the presence of a new server.

Specified By: serviceAdded(ServiceDiscoveryEvent)$_{20}$ in interface ServerManager$_{19}$

Parameters:
evt - signaling event

serviceChanged(ServiceDiscoveryEvent)

```
public synchronized void serviceChanged(net.jini.lookup.ServiceDiscoveryEvent evt)
```

Method used by the ServiceDiscoveryListener interface to signal the change of a server.

Specified By: serviceChanged(ServiceDiscoveryEvent)$_{20}$ in interface ServerManager$_{19}$

Parameters:
evt - signaling event

serviceFailure(String)

```
public synchronized void serviceFailure(java.lang.String failedAddr)
```

Method invoked by the proxy if it noticed a server is not accessible

Parameters:
`failedAddr` - address of the failed server

serviceRemoved(ServiceDiscoveryEvent)

```
public synchronized void serviceRemoved(net.jini.lookup.ServiceDiscoveryEvent evt)
```

Method used by the ServiceDiscoveryListener interface to signal the leaving of a server. This method does nothing if the server has been already removed from the managed lists.

Specified By: `serviceRemoved(ServiceDiscoveryEvent)`[21] in interface `ServerManager`[19]

Parameters:
`evt` - signaling event

terminate()

```
public void terminate()
```

Clear termination of the manager

terminate()

Package
com.andreani.xtol.test

Description
The ***eXtremeTolerance*** test package. This package contains classes used for debugging and test.

Class Summary
Classes
SnoopSpace[24]

com.andreani.xtol.test
SnoopSpace

Declaration

```
public class SnoopSpace implements net.jini.lookup.ServiceDiscoveryListener

java.lang.Object
  |
  +--com.andreani.xtol.test.SnoopSpace
```

All Implemented Interfaces: net.jini.lookup.ServiceDiscoveryListener

Member Summary

Fields

protected	`cat`[25] — Log handler (Log4Java)

Constructors

public	`SnoopSpace()`[25] — Connects to the JavaSpaces that contain the SessionEntries and registers the listener for new spaces.

Methods

public void	`insertEmptyEntry(String, String)`[25] — Inserts an entry into the space with an empty hashtable.
public static void	`main(String[])`[25]
public void	`populateSpace(int)`[25] — This method populate a new JavaSpace with SessionEntries contained in the other spaces.
public void	`serviceAdded(ServiceDiscoveryEvent)`[25]
public void	`serviceChanged(ServiceDiscoveryEvent)`[26] — Description of the Method
public void	`serviceRemoved(ServiceDiscoveryEvent)`[26] — Description of the Method
public Entry	`takeEntry()`[26] — Takes an entry from the JavaSpace
public void	`terminate()`[26]
public void	`updateReferences(String, String)`[26]
public void	`writeEntry(Entry)`[26] — Description of the Method

Inherited Member Summary

Methods inherited from class java.lang.Object

Inherited Member Summary
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

cat

```
protected org.apache.log4j.Category cat
```

Log handler (Log4Java)

Constructors

SnoopSpace()

```
public SnoopSpace()
```

Connects to the JavaSpaces that contain the SessionEntries and registers the listener for new spaces.

Methods

insertEmptyEntry(String, String)

```
public void insertEmptyEntry(java.lang.String sessionid, java.lang.String server)
```

Inserts an entry into the space with an empty hashtable.

main(String[])

```
public static void main(java.lang.String[] args)
```

populateSpace(int)

```
public void populateSpace(int i)
```

This method populate a new JavaSpace with SessionEntries contained in the other spaces. ServiceDiscoveryListener docs suggests to simply note the occurrence of the ServiceDiscoveryEvent, and perform any time consuming event handling asynchronously, so this method needs to be fast, otherwise it needs to be invoked asynchronously.

Parameters:

`i` - Index of the space to populate

serviceAdded(ServiceDiscoveryEvent)

```
public void serviceAdded(net.jini.lookup.ServiceDiscoveryEvent evt)
```

Specified By: `serviceAdded(ServiceDiscoveryEvent)`[25] in interface `SnoopSpace`[24]

serviceChanged(ServiceDiscoveryEvent)

```
public void serviceChanged(net.jini.lookup.ServiceDiscoveryEvent event)
```

Description of the Method

Specified By: `serviceChanged(ServiceDiscoveryEvent)`[26] in interface `SnoopSpace`[24]

Parameters:

`event` - Description of Parameter

serviceRemoved(ServiceDiscoveryEvent)

```
public void serviceRemoved(net.jini.lookup.ServiceDiscoveryEvent event)
```

Description of the Method

Specified By: `serviceRemoved(ServiceDiscoveryEvent)`[26] in interface `SnoopSpace`[24]

Parameters:

`event` - Description of Parameter

takeEntry()

```
public net.jini.core.entry.Entry takeEntry()
```

Takes an entry from the JavaSpace

Parameters:

`sessionid` - Id of the entry to take

Returns: Found entry

terminate()

```
public void terminate()
```

updateReferences(String, String)

```
public void updateReferences(java.lang.String oldAddr, java.lang.String newAddr)
```

writeEntry(Entry)

```
public void writeEntry(net.jini.core.entry.Entry entry)
```

Description of the Method

Parameters:

`entry` - Description of Parameter

Package
com.andreani.xtol.util

Description
The ***eXtremeTolerance*** helper classes package. This package contains common classes used by both proxy and node packages for management purposes.

Class Summary	
Classes	
`Global` 28	This class provides some constants accessible from all the other classes.
`SessionEntry` 30	This class is the Entry object contained into Session Spaces that stores user state and his bound server address.
`SessionEntryBean` 33	This class is useful for browsing JavaSpace with Sun's space browser
`SessionSpaceManager` 35	This class manages available JavaSpaces putting and getting entries.
`TxManHandler` 38	Manager class for transactions.

com.andreani.xtol.util
Global

Declaration
```
public abstract class Global

java.lang.Object
  |
  +--com.andreani.xtol.util.Global
```

Description
This class provides some constants accessible from all the other classes.

Member Summary

Fields

public static final	COOKIE_FAIL_OVER[28] This cookie is used to signal a server node failure.
public static final	COOKIE_NAME_SESSIONID[29] Name of the cookie used to tie a client with a JavaSpace session object.
public static final	DISCOV_TIMEOUT[29] Timeout of the discovery phase.
public static final	LOOKUP_TIMEOUT[29] Timeout of the lookup phase.
public static final	TRANS_TIMEOUT[29] Transaction timeout: if a transaction delay more then this value, the transaction server rolls back.
public static final	TX_MANAGER_NAME[29] The name of the transaction manager.

Constructors

public	Global()[29]

Inherited Member Summary

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

COOKIE_FAIL_OVER
```
public static final java.lang.String COOKIE_FAIL_OVER
```

This cookie is used to signal a server node failure.

COOKIE_NAME_SESSIONID

```
public static final java.lang.String COOKIE_NAME_SESSIONID
```

Name of the cookie used to tie a client with a JavaSpace session object.

DISCOV_TIMEOUT

```
public static final long DISCOV_TIMEOUT
```

Timeout of the discovery phase.

LOOKUP_TIMEOUT

```
public static final long LOOKUP_TIMEOUT
```

Timeout of the lookup phase.

TRANS_TIMEOUT

```
public static final long TRANS_TIMEOUT
```

Transaction timeout: if a transaction delay more then this value, the transaction server rolls back.

TX_MANAGER_NAME

```
public static final java.lang.String TX_MANAGER_NAME
```

The name of the transaction manager.

Constructors

Global()

```
public Global()
```

com.andreani.xtol.util
SessionEntry

Declaration

```
public class SessionEntry extends net.jini.entry.AbstractEntry

java.lang.Object
  |
  +--net.jini.entry.AbstractEntry
        |
        +--com.andreani.xtol.util.SessionEntry
```

All Implemented Interfaces: net.jini.core.entry.Entry, java.io.Serializable

Description

This class is the Entry object contained into Session Spaces that stores user state and his bound server address.

Member Summary

Fields

public	`fail`[31] — Fail is true if the server which was servicing has failed and the new one assigned hasn't yet copied this session into its cache.
public	`id`[31] — This is the Global Unique Id that identify user session.
public	`serverAddr`[31] — The address of destination server
public	`session`[31] — This Hashtable contains all objects representing user state.

Constructors

public	`SessionEntry()`[31] — Empty Constructor for the SessionEntry object
public	`SessionEntry(String, String, Hashtable, Boolean)`[31] — Constructor for the SessionEntry object

Methods

public String	`toString()`[32] — Formatted output, useful for debugging

Inherited Member Summary

Methods inherited from class net.jini.entry.AbstractEntry

`equals, equals, hashCode, hashCode, toString`

Methods inherited from class java.lang.Object

Inherited Member Summary
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Fields

fail

`public java.lang.Boolean` **`fail`**

Fail is true if the server which was servicing has failed and the new one assigned hasn't yet copied this session into its cache.

id

`public java.lang.String` **`id`**

This is the Global Unique Id that identify user session.

serverAddr

`public java.lang.String` **`serverAddr`**

The address of destination server

session

`public java.util.Hashtable` **`session`**

This Hashtable contains all objects representing user state.

Constructors

SessionEntry()

`public` **`SessionEntry`**`()`

Empty Constructor for the SessionEntry object

SessionEntry(String, String, Hashtable, Boolean)

```
public SessionEntry(java.lang.String id_p, java.lang.String serverAddr_p,
            java.util.Hashtable obj, java.lang.Boolean fail_p)
```

Constructor for the SessionEntry object

Parameters:

`id_p` - Session GUID

`serverAddr_p` - IP and port

`obj` - user state

`fail_p` - fail flag

Methods

toString()

```
public java.lang.String toString()
```

Formatted output, useful for debugging

Overrides: net.jini.entry.AbstractEntry.toString() in class net.jini.entry.AbstractEntry

Returns: String representation of this object

com.andreani.xtol.util
SessionEntryBean

Declaration
```
public class SessionEntryBean implements net.jini.lookup.entry.EntryBean, java.io.Serializable

java.lang.Object
  |
  +--com.andreani.xtol.util.SessionEntryBean
```

All Implemented Interfaces: net.jini.lookup.entry.EntryBean, java.io.Serializable

Description
This class is useful for browsing JavaSpace with Sun's space browser

Member Summary	
Fields	
protected	internalInst[33]
Constructors	
public	SessionEntryBean()[34]
Methods	
public Entry	followLink()[34]
public void	makeLink(Entry)[34]

Inherited Member Summary
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

internalInst
```
protected SessionEntry[30] internalInst
```

Constructors

SessionEntryBean()

```
public SessionEntryBean()
```

Methods

followLink()

```
public net.jini.core.entry.Entry followLink()
```

Specified By: `followLink()`[34] in interface `SessionEntryBean`[33]

makeLink(Entry)

```
public void makeLink(net.jini.core.entry.Entry obj)
```

Specified By: `makeLink(Entry)`[34] in interface `SessionEntryBean`[33]

com.andreani.xtol.util
SessionSpaceManager

Declaration

```
public class SessionSpaceManager implements net.jini.lookup.ServiceDiscoveryListener

java.lang.Object
  |
  +--com.andreani.xtol.util.SessionSpaceManager
```

All Implemented Interfaces: net.jini.lookup.ServiceDiscoveryListener

Description

This class manages available JavaSpaces putting and getting entries.

Member Summary

Fields

protected	`cat`[36] — Log handler (Log4Java)

Constructors

public	`SessionSpaceManager()`[36] — Connects to the JavaSpaces that contain the SessionEntries and registers the listener for new spaces.

Methods

public void	`insertEmptyEntry(String, String)`[36] — Inserts an entry into the space with an empty hashtable.
public void	`populateSpace(int)`[36] — This method populate a new JavaSpace with SessionEntries contained in the other spaces.
public SessionEntry	`retrieveEntry(String)`[36] — Find an entry into the space.
public void	`serviceAdded(ServiceDiscoveryEvent)`[37]
public void	`serviceChanged(ServiceDiscoveryEvent)`[37]
public void	`serviceRemoved(ServiceDiscoveryEvent)`[37]
public void	`setFailStatus(String, boolean)`[37]
public void	`snoopSpace()`[37]
public void	`terminate()`[37]
public void	`updateReferences(String, String)`[37]
public void	`updateSession(SessionEntry)`[37]

Inherited Member Summary

Methods inherited from class java.lang.Object

Inherited Member Summary
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

cat

`protected org.apache.log4j.Category` **`cat`**

Log handler (Log4Java)

Constructors

SessionSpaceManager()

`public` **`SessionSpaceManager`**`()`

Connects to the JavaSpaces that contain the SessionEntries and registers the listener for new spaces.

Methods

insertEmptyEntry(String, String)

`public void` **`insertEmptyEntry`**`(java.lang.String sessionid, java.lang.String server)`

Inserts an entry into the space with an empty hashtable.

populateSpace(int)

`public void` **`populateSpace`**`(int i)`

This method populate a new JavaSpace with SessionEntries contained in the other spaces. ServiceDiscoveryListener docs suggests to simply note the occurrence of the ServiceDiscoveryEvent, and perform any time consuming event handling asynchronously, so this method needs to be fast, otherwise it needs to be invoked asynchronously.

Parameters:
`i` - Index of the space to populate

retrieveEntry(String)

`public` SessionEntry$_{30}$ **`retrieveEntry`**`(java.lang.String sessionid)`

Find an entry into the space. The caller must be sure the entry is into the space.

Parameters:
`sessionid` - id of the entry to find

Returns: returns the session entry

serviceAdded(ServiceDiscoveryEvent)

```
public void serviceAdded(net.jini.lookup.ServiceDiscoveryEvent evt)
```

Specified By: serviceAdded(ServiceDiscoveryEvent)[37] in interface SessionSpaceManager[35]

serviceChanged(ServiceDiscoveryEvent)

```
public void serviceChanged(net.jini.lookup.ServiceDiscoveryEvent event)
```

Specified By: serviceChanged(ServiceDiscoveryEvent)[37] in interface SessionSpaceManager[35]

serviceRemoved(ServiceDiscoveryEvent)

```
public void serviceRemoved(net.jini.lookup.ServiceDiscoveryEvent event)
```

Specified By: serviceRemoved(ServiceDiscoveryEvent)[37] in interface SessionSpaceManager[35]

setFailStatus(String, boolean)

```
public void setFailStatus(java.lang.String sessionid, boolean fail)
```

snoopSpace()

```
public void snoopSpace()
```

terminate()

```
public void terminate()
```

updateReferences(String, String)

```
public void updateReferences(java.lang.String oldAddr, java.lang.String newAddr)
```

updateSession(SessionEntry)

```
public void updateSession(SessionEntry[30] entry)
```

com.andreani.xtol.util
TxManHandler

Declaration
```
public class TxManHandler implements net.jini.lookup.ServiceDiscoveryListener

java.lang.Object
  |
  +--com.andreani.xtol.util.TxManHandler
```

All Implemented Interfaces: net.jini.lookup.ServiceDiscoveryListener

Description
Manager class for transactions.

Member Summary	
Fields	
protected	cat[38]
Constructors	
public	TxManHandler()[39] Constructor for the TxManHandler object
Methods	
public Transaction-Manager	getTxMgr()[39] Gets the TxMgr attribute of the TxManHandler object
public void	serviceAdded(ServiceDiscoveryEvent)[39]
public void	serviceChanged(ServiceDiscoveryEvent)[39]
public void	serviceRemoved(ServiceDiscoveryEvent)[39]

Inherited Member Summary
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

cat
```
protected org.apache.log4j.Category cat
```

Constructors

TxManHandler()

```
public TxManHandler()
```

Constructor for the TxManHandler object

Methods

getTxMgr()

```
public net.jini.core.transaction.server.TransactionManager getTxMgr()
```

Gets the TxMgr attribute of the TxManHandler object

serviceAdded(ServiceDiscoveryEvent)

```
public void serviceAdded(net.jini.lookup.ServiceDiscoveryEvent evt)
```

Specified By: `serviceAdded(ServiceDiscoveryEvent)`[39] in interface `TxManHandler`[38]

serviceChanged(ServiceDiscoveryEvent)

```
public void serviceChanged(net.jini.lookup.ServiceDiscoveryEvent event)
```

Specified By: `serviceChanged(ServiceDiscoveryEvent)`[39] in interface `TxManHandler`[38]

serviceRemoved(ServiceDiscoveryEvent)

```
public void serviceRemoved(net.jini.lookup.ServiceDiscoveryEvent event)
```

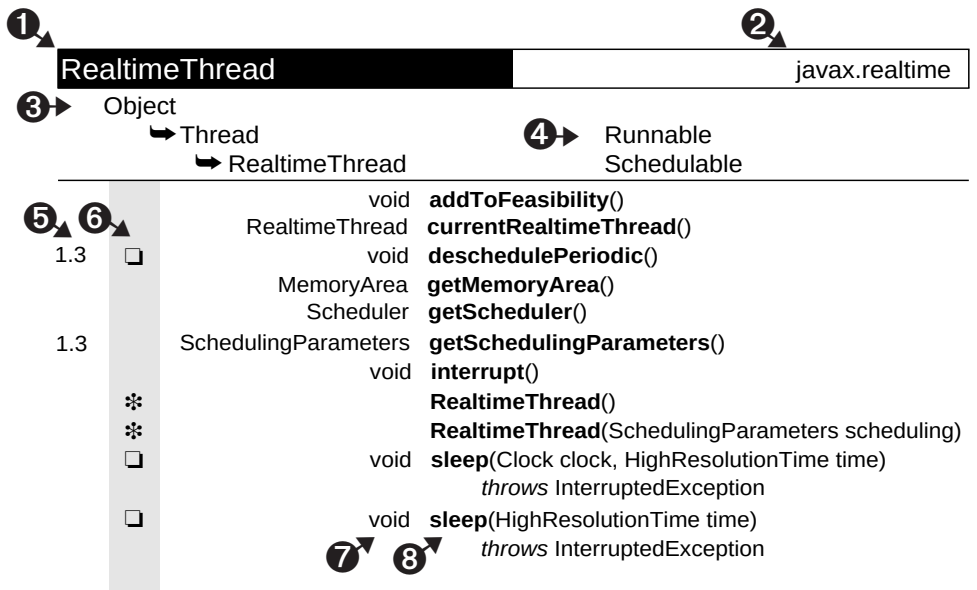
Specified By: `serviceRemoved(ServiceDiscoveryEvent)`[39] in interface `TxManHandler`[38]

serviceRemoved(ServiceDiscoveryEvent)

ALMANAC LEGEND

The almanac presents classes and intefaces in alphabetic order, regardless of their package. Each class displays a list of its members in alphabetic order – fields, methods and constructors are sorted together.

This almanac is modeled after the style introduced by Patrick Chan in his excellent book *Java Developers Almanac*.



1. The name of the class, interface, nested class or nested interface. All interfaces are shown in italic.
2. The name of the package containing the class.
3. The inheritance chain of superclasses. In this example, `RealtimeThread` extends `Thread`, which extends `Object`.
4. Implemented interfaces. The class and the interface it implements are on the same line. In this example, `Thread` implements `Runnable`, and `RealtimeThread` implements `Schedulable`.
5. The first column is for the value of the `@since` comment, which indicates the version in which the item was introduced.
6. The second column is for the following icons that indicate modifiers, constructors and fields. If the "protected" symbol does not appear, the member is public. (Private and package-private modifiers have no symbols.)

Modifiers	**Constructors and Fields**
❍ abstract	❇ constructor
● final	✍ field
❑ static	
■ static final	
♦ protected	

7. The return type of a method or the declared type of a field. It is blank for constructors.
8. The name of the constructor, field or method. Sorted alphabetically. Nested classes are not listed as members.

Almanac

ForwServlet	com.andreani.xtol.proxy

Object
➥javax.servlet.GenericServlet — javax.servlet.Servlet, javax.servlet.ServletConfig, java.io.Serializable
➥javax.servlet.http.HttpServlet — java.io.Serializable
➥ForwServlet — javax.servlet.SingleThreadModel

✍♦	org.apache.log4j.Category	**cat**
	void	**destroy()**
	void	**doGet(javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse resp) *throws* javax.servlet.ServletException, java.io.IOException**
	void	**doPost(javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse resp) *throws* javax.servlet.ServletException, java.io.IOException**
❊		**ForwServlet()**
	void	**init() *throws* javax.servlet.ServletException**
✍♦	java.util.Hashtable	**serverAddr**

Global	com.andreani.xtol.util

Object
➥Global

✍■	String	**COOKIE_FAIL_OVER**
✍■	String	**COOKIE_NAME_SESSIONID**
✍■	long	**DISCOV_TIMEOUT**
❊		**Global()**
✍■	long	**LOOKUP_TIMEOUT**
✍■	long	**TRANS_TIMEOUT**
✍■	String	**TX_MANAGER_NAME**

ServerManager	com.andreani.xtol.proxy

Object
➥ServerManager — net.jini.lookup.ServiceDiscoveryListener

✍♦	org.apache.log4j.Category	**cat**
	String	**findNextServer()**
❊		**ServerManager()**
	void	**serviceAdded(net.jini.lookup.ServiceDiscoveryEvent evt)**
	void	**serviceChanged(net.jini.lookup.ServiceDiscoveryEvent evt)**

	void	**serviceFailure(String failedAddr)**
	void	**serviceRemoved(net.jini.lookup.ServiceDiscoveryEvent evt)**
	void	**terminate()**

ServerNode — com.andreani.xtol.node

Object
➥ServerNode — java.io.Serializable

	net.jini.core.entry.Entry[]	**getEntries()**
✍	String	**serverIP**
✍	Double	**serverLoad**
❇		**ServerNode(String ip, int port, double load)**
✍	Integer	**serverPort**

SessionEntry — com.andreani.xtol.util

Object
➥net.jini.entry.AbstractEntry — net.jini.core.entry.Entry
➥SessionEntry

✍	Boolean	**fail**
✍	String	**id**
✍	String	**serverAddr**
✍	java.util.Hashtable	**session**
❇		**SessionEntry()**
❇		**SessionEntry(String id_p, String serverAddr_p, java.util.Hashtable obj, Boolean fail_p)**
	String	**toString()**

SessionEntryBean — com.andreani.xtol.util

Object
➥SessionEntryBean — net.jini.lookup.entry.EntryBean, java.io.Serializable

	net.jini.core.entry.Entry	**followLink()**
✍♦	SessionEntry	**internalInst**
	void	**makeLink(net.jini.core.entry.Entry obj)**
❇		**SessionEntryBean()**

SessionEventHandler — com.andreani.xtol.node

Object
➥SessionEventHandler — javax.servlet.http.HttpSessionListener, javax.servlet.http.HttpSessionAttributeListener

	void	**attributeAdded(javax.servlet.http.HttpSessionBindingEvent se)**
	void	**attributeRemoved(javax.servlet.http.HttpSessionBindingEvent se)**
	void	**attributeReplaced(javax.servlet.http.HttpSessionBindingEvent se)**
✍♦	org.apache.log4j.Category	**cat**
	void	**sessionCreated(javax.servlet.http.HttpSessionEvent se)**
	void	**sessionDestroyed(javax.servlet.http.HttpSessionEvent se)**
❇		**SessionEventHandler()**
❇		**SessionEventHandler(net.jini.core.lookup.ServiceRegistrar registrar)**

SessionSpaceManager — com.andreani.xtol.util

Object
➥SessionSpaceManager — net.jini.lookup.ServiceDiscoveryListener

✍✦	org.apache.log4j.Category	**cat**
	void	**insertEmptyEntry(String sessionid, String server)**
	void	**populateSpace(int i)**
	SessionEntry	**retrieveEntry(String sessionid)**
	void	**serviceAdded(net.jini.lookup.ServiceDiscoveryEvent evt)**
	void	**serviceChanged(net.jini.lookup.ServiceDiscoveryEvent event)**
	void	**serviceRemoved(net.jini.lookup.ServiceDiscoveryEvent event)**
❄		**SessionSpaceManager()**
	void	**setFailStatus(String sessionid, boolean fail)**
	void	**snoopSpace()**
	void	**terminate()**
	void	**updateReferences(String oldAddr, String newAddr)**
	void	**updateSession(SessionEntry entry)**

SnoopSpace — com.andreani.xtol.test

Object
➥SnoopSpace — net.jini.lookup.ServiceDiscoveryListener

✍✦	org.apache.log4j.Category	**cat**
	void	**insertEmptyEntry(String sessionid, String server)**
❑	void	**main(String[] args)**
	void	**populateSpace(int i)**
	void	**serviceAdded(net.jini.lookup.ServiceDiscoveryEvent evt)**
	void	**serviceChanged(net.jini.lookup.ServiceDiscoveryEvent event)**
	void	**serviceRemoved(net.jini.lookup.ServiceDiscoveryEvent event)**
❄		**SnoopSpace()**
	net.jini.core.entry.Entry	**takeEntry()**
	void	**terminate()**
	void	**updateReferences(String oldAddr, String newAddr)**
	void	**writeEntry(net.jini.core.entry.Entry entry)**

TolerantServlet — com.andreani.xtol.node

Object
➥javax.servlet.GenericServlet — javax.servlet.Servlet, javax.servlet.ServletConfig, java.io.Serializable
➥javax.servlet.http.HttpServlet — java.io.Serializable
➥TolerantServlet — net.jini.lookup.ServiceIDListener

✍✦	org.apache.log4j.Category	**cat**
	void	**destroy()**
	void	**doGet(javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse resp) *throws* javax.servlet.ServletException, java.io.IOException**
	void	**init(javax.servlet.ServletConfig sconf)**

	void	**rebuildSession(String globsessid, javax.servlet.http.HttpServletRequest req)**
	void	**serviceIDNotify(net.jini.core.lookup.ServiceID id)**
❇		**TolerantServlet()**

TxManHandler	**com.andreani.xtol.util**

Object
↳TxManHandler net.jini.lookup.ServiceDiscoveryListener

✍✦	org.apache.log4j.Category	**cat**
	net.jini.core.transaction.server.TransactionManager	**getTxMgr()**
	void	**serviceAdded(net.jini.lookup.ServiceDiscoveryEvent evt)**
	void	**serviceChanged(net.jini.lookup.ServiceDiscoveryEvent event)**
	void	**serviceRemoved(net.jini.lookup.ServiceDiscoveryEvent event)**
❇		**TxManHandler()**

Index

T

U

W