Advanced Software Engineering

Homework #1

Patrick O’Brien

ORB:

For testing and development I used JacORB ([www.jacorb.org/index.html](http://www.jacorb.org/index.html)) version 3.2 (<http://www.jacorb.org/releases/3.2/jacorb-3.2-binary.zip> )

DEVELOPMENT ENVIORNMENT:

I am using Eclipse IDE Juno Service Release 2. Dependency management is being handled by IVY.

SYSTEM ARCHITECTURE/DEPENDENCIES

I am using Spring to instantiate the Client and Server code. This is allowing me to abstract the core business logic (i.e. the auction house requirements) from the communication requirements of CORBA.

I have also abstracted out the ‘data’ layer.

I am using the built in java TreeMap data structures, this is due to no requirements for persistence another data source can easily be plugged in, including a database/cache, no-sql or map-reduce implementation. All variables accessible via multi-threading are using the java concurrency package.

Logging is handled via slf and logback. The logback.xml file included in the project is very standard for a basic setup to write to the console.

Initialization of the auction house is done via xtreme library; parsing a json file. Invalid data is included as a testing of the loading software validation to ensure that the auction house does not go into an invalid state. I. E.. Auctions for items that do not exist.

The ORBs connect via the NameSpace file that is generated upon server startup. The customer object then use that file to connect to the Server. The reason for this is described in the Jacorb documentation.

The name server does *not* use a well known port for its service and there is no way to direct it to a specific port. Since clients cannot (and need not) know in advance where the name service will be provided, we use a bootstrap file in which the name server records an opaque object reference to itself (its *Interoperable Object Reference* or IOR). The name of this bootstrap file has to be given as an argument to the ns command. This bootstrap file has to be available to clients networkwide, so we demand that it be reachable via a URL -- that is, there must be an appropriately configured HTTP server in your network domain which allows read access to the bootstrap file over a HTTP connection. (This implies that the file must have its read permissions set appropriately. If the binding to the name service fails, please check that this is the case.) After locating the name service through this mechanism, clients will connect to the name server directly, so the only HTTP overhead is in the first lookup of the server.

The name bindings in the server's database are mirrored in the bootstrap file, so you have a network wide readable log of which names are already in use.

The Server ran in the standard server mode, while the client used the bidirectional mode. Other then access to the NS file the only other dependency the customer objects have on the server is the IDL.

PROBLEMS:

I did not have any problems setting this up at all.