Faculty of Information Technology , Brno University of Technology

Project Documentation for IPP subject

Project documentation for project IJA 2009

Project no. 1

22. April 2009

Team members:

Tomáš Benčok	xbenco00@stud.fit.vutbr.cz	20
Ivan Homoliak	xhomol11@stud.fit.vutbr.cz	20
Matúš Kontra	xkontr00@stud.fit.vutbr.cz	20
Patrik Šebeň	xseben00@stud.fit.vutbr.cz	20

Contents

1 Introduction

The calendar has been there for many years (the first calendar is dated to 4,000 BC) and for many people, it‘s an essential and unthinkable part of their lives. Take for example students – nowadays, it would be almost impossible to remember all the possible dates of seminars, deadlines of the projects, parity of the week, and many other different things without a calendar – yes, it’s possible, but not so comfortable...

This documentation describes the layout and the implementation of the calendar in Java language for IJA subject, and then as the next thing it deals with potential problems, that might appear during the solving of the problem.

The objective of the project wasn’t so easy – it was necessary to create an application Calendar on the principle of the client-server architecture (communication using a safe SSL connection) with a graphical user interface (GUI). The application had to provide all the basic functions of a classical (paper) calendar plus something more – e.g. planning events, alerts on these events, their printout and so on...

The documentation is divided into several parts, chapter 2 talks about our proceeding during the implementation of the project, in chapter 3 you will find basic information about the server, chapter 4 on the other hand describes the client program, chapter 5 explains the stored data structure in the database, chapter 6 deals with the script, and finally - chapter 7 summarizes the pros and cons of the project.

2 Proceeding during the implementation of program parts of the project

As the first thing, we’ve implemented classes, that represent the entities used in the application – e.g. user, group, calendar, event, notice, alert... At the same time, we’ve also implemented the methods for work with these types of objects (export and import from / to XML representation).

Next, we have put into operation the SSL communication between simple test applications, which we could use for further communication between semi-finished client and server applications. Then it was necessary to design the communication protocol. The Client-Server communication was developed as a synchronous one. Our protocol works on the following principle: During the communication, the size of data in bytes and 8-character representation of the request (or the answer) are send in the header. The body contains the mentioned data for transmission.

Than we started in parallel to develop and test the processing of requirements or the answers on the server. The ConnectionManager for the server was designed and it was subsequently connected with the RequestHandler. We’ve also created a ServerController and it’s ServerCache (for more details, see description below). The function of these parts was tested on the simulation of client‘s requirements and its answers.

Simultaneously we worked on the client application, particularly ClientController and the module for the cache management. These two were interconnected and then connected with the CommunicationHandler. The testing and debugging of the client was done directly within the communication with the server. It is expected, that only one client can connect to the server from the same computer.

Later we've designed all dialogs for GUI and the module representing the services of the actions was created too. Then we've connected it to the ClientController. After this, all parts of the calendar should be done right now - only one thing remains – testing.

Testing was probably the most difficult and the most problematic part of the development. It took us a lot of time, but we finally did it.

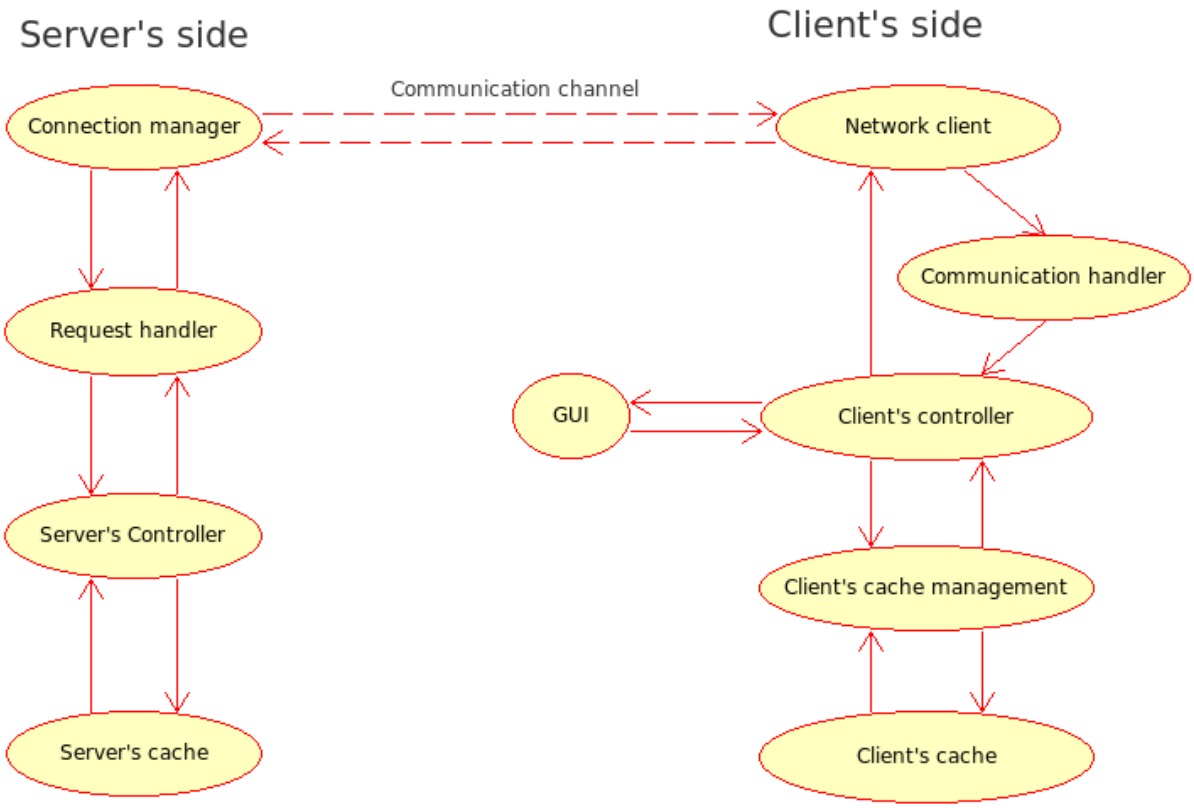

Diagram 1 – shows the relations between the individual parts of the client and server applications

3 Server

The server is composed of four main parts - ConnectionManager, RequesHandler, ServerController and DatabaseMemory. All parts are described bellow.

The ConnectionManager uses the SSL type of connection and it sends or receives packets for/from the client. It accepts the connection and for each connection, the new instance of private class is created and it represents this connection. It is always started as an individual thread and it will communicate with a corresponding client on a given port. It contains a private class ConnectionAdder, which manages adding or removing connections from the list of active connections.

A special packet is expected in the beginning of the communication – it should carry the request to log in (if the user exists), or to register a new user. If the requirements for the registration are met, the new user is created in the cache memory of the server, this user is automatically logged in and the acknowledgement (everything ok) is send back immediately. But if they aren‘t met, (e.g. the login already exists, wrong password) – the client will be alerted about this situation (a response with a negative acknowledgement is sent back).

The RequestHandler is intended for sending packets with operations, which have been done over the database on the side of the server (on the basis of requirements of other clients), or with responses for the client's requirements. It is designated to serve only to the registered users, that's why the registration (or log in) of the new users is done by the ConnectionManager. It utilizes methods implemented in the ServerController.

The group events such as a new event, a notice or an alert are handled in a way, that a new packet containing new or changed parts is sent to all members of the group, that are online. The offline members get a new version after they have logged in (because of the mechanism of synchronization).

The ServerController – It‘s main function is to load the content of the database stored on the physical disc to the cache memory of the server. After its initiation it‘ll work with this memory. There's a thread in it, that synchronises the content of the cache memory to the disc and back in regular intervals. In addition, it contains approximately 40 methods, which deal with adding, removing or editing the notices, alerts, groups, groups events, etc. – and there‘s also a mechanism for the authentication of the user. Passwords on the disk of the server aren't kept there in an open form, but the hash function is used for this purpose.

4 Client

It is composed of following parts: NetworkClientController, ClientController, CommunicationHandler, GUI and LocalCache. Just like the server, the client uses also the SSL type of connection.

The NetworkClientController – it sends or receives packets from the server. The packets are processed by the CommunicationHandler, but it does other things too, so the buffer above the packets is created. It contains a private class PacketReceiverThread, which instantly reads the packets from the buffer and starts their attendance.

The CommunicationHandler – reacts on the packets sent from the server and on the basis of their content it modifies the data stored in the LocalCache. When it's performing this action, it uses some of the methods implemented in the ClienConntroler. The received packets are parsed and prearranged for the ClientConntroler.

The ClientConntroler - part of the implemented methods is used by GUI, the remaining part is used by the CommunicationHandler. These methods register or log in the user, add/remove/modify events, notices, either in the LocalCache (it's expected, that he's logged in, but he works in the offline mode) or they send the requirement about the given action to the server and on the basis of the answer the LocalCache is updated or not (it concerns group events, for example creating a new group, new group events, alerts and notices).

The LocalCache – it's main function is to represent the local cache (the physical one) and to provide a great number of operations above it. After the user is logged in, the information from the disc are loaded to the local cache. It uses an additional thread, which performs syncing in regular intervals. The performed action depends on the state of the connection. If the user works online, the server sends the updated information to the client, otherwise the client sends new information to the server (but the synchronization must be done at the first place).

GUI – it is connected with the ClientController and all the control over the client's application is possible thanks to GUI. For more information, please see the user documentation.

5 Representation of the database on the side of the client and the server

The database of the groups and users is represented almost in the same way on the side of the server or client, but there is a small difference – the client maintains

information only about those users (and their groups), that were logged in from this client, whereby the server maintains the entire database of groups and users.

There are two items in the actual folders of the client and the server, called ***users*** and ***groups***, which contains the XML representation of information about users and groups. Each user (and also a group) has their own file. We've chosen an implementation provided by JDOM.

5.1 Structure of XML file representing the user

The root element User has attributes "Login", "Name", "Surname" and the hash of the "Password". Moreover it contains child elements Calendar & GroupMembership.

The Calendar has an attribute "Name" and several child elements of the type Event, of which every single element represents some other event. The element Event has attributes "Name", "Date", "Repeating", "HowOften", "Category". The child elements of Event are Notices and Alerts.

The element Notices contains an element named Notice with the attributes "Autor", "TimeEntered" and the body of this Notice of which every element represents notice of the parent Event.

The element Alerts contains child elements Alert with attributes "StartDate", "Repetable", and "Interval". Every type of this element represents a notification on some event of the parent's Event.

The GroupMembership – it contains several child elements of the type Group, whose attributes are "Name" & "Leader" and they represent the membership and the function of a specific user in the group.

5.2 Structure of XML file representing the group

The root element Group has the attributes "Name" – represents the name of the group and "Leader" - represents the leader of the group. The Group contains elements Event with the attributes "Name", "Date", "Repeating", "HowOften" and "Category". Each Event represents one group event, and it contains child elements called Notices and Alerts.

The element Notices contains child elements of the type Notice with the attributes "Author", "TimeEntered", and the body of this Notice.

The element Alerts contains several elements of the type Alert with the attributes "StartDate", "Repetable" and "Interval".

6 Script

We have chosen a python language for the implementation of the script, because we know it better than other script languages and it makes it possible to solve problems in a very quick and effective way. Here's the mechanism of how it works.

The function for processing of parameters of a command prompt is called in the beginning, and on the basis of the result, other functions are called next to solve the demanded tasks. Many of these functions are implemented as finite-state machines, because it would be much more harder to do this by a regular expression. The regular expression is used only in the function, that searches for some pattern.

At the start of each function, the list of java files is created. This list is made by a recursive function, that is diving to the depth of actual folder structure and searching for the java files. Then, the list of these files is examined in a cycle and every file is then processed by a finite-state machine (FSM). In many states of FSM are frequently used static lists declared globally and they contain the key words, operators and operators following the identifiers.

Ordering the files without a parameter, that indicates a display of a relative path is solved before the start of FSM – this way it allows to display the names of files with a number of their occurrences directly after passing the FSM. But if this parameter is used, the display of a relative path is not possible – that's why the list of two-somes was created (a two-some consists of the name of the file and the count of occurrations of the searched item). This list is ordered by a filename after passing through the structure of all files and finally is printed out in an ordered way with the found results.

7 Summary

Our effort was to fulfil all the objectives from the assignment in the most outstanding way. These objectives were, in our opinion, very difficult and regard other projects for other subjects - there wasn't enough time for the project realization. We are certain, that we would have improved our implementation, if there had been a little more time for it.

We were using the integrated NetBeans development environment as well as the PsPad Editor, because the NetBeans prohibits editing the generated code – this is its weak spot. But it was very good for example for designing dialog windows.

The biggest problem during our work was to create and troubleshoot the GUI, also its connecting to the ClientController wasn't so easy. Another tough thing was to synchronize the client with the server and their communication too.