

Chat Room LLD

Terminology

Chat Room

A virtual environment in which users can post their messages and read the messages written by other users.

User

A person who interacts with the system.

Nickname

A familiar or humorous name the user uses to identify himself.

Registration

The act of recording user details.

Login

The act of signing into the system by the user.

Message

The text which the user delivers. Message content is limited to 150 characters.

Message Frame

A written communication sent between the users of the system.
A wrapper for a message.

Business Logic

ChatRoom class

Functionality

Operate the chat room and provides the user with the requested functionalities from the requirements document. This class is static.

Fields :

HOME_URL - holds a static field with the url when running the server locally.

BGU_URL - holds a static field with the url when connecting to the server at the university.

_location - holds the client's location.

_loggedinUser - holds the current user info.

enums

Place - can get the values of Home or University. helps to easily connect to a server that is running locally. (public)

Properties

URL - return the server's url according to the client location.

LoggedInUser - get and set the lodged in user.

Functions

Start(Place location) - initialize the ram with the registers users and messages, and set the location of the server according to location.

exit() - close the program.

isLoggedIn() - return if there is a logged in user

register(string nickname, int GroupID, string password) - register with the given nickname, group, and password.

our's groupID.

login(string nickname, int GroupID, string password) – login an existing with the given nickname, group, and password.

logout() - logout the user from the server

send(string body) - send a new message with the given body

DrawLastMessages() – draw the last messages since last draw.

EditMessage(Guid ID, string newBody) – Edit. message by guid and save it.

requestAllMessagesfromUser(string nickname, int GroupID) - receive all the messages that was sent by the user with the given nickname on the requested group ID

sort(ICollection<IMessage> messages, MessageService.Sort SortBy, bool descending)- Sort a message List by the time

requestAllMessagesfromUser(string nickName, int GroupID)- Receive all the messages from a certain user

requestMessagesfromUser(ICollection<IMessage> messages, string nickName, int GroupID)- Receive all the messages from a certain user from a certain collection

requestAllMessagesfromGroup(int GroupID)- Receive all the messages from a certain group

requestMessagesfromGroup(ICollection<IMessage> messages, int GroupID)- Receive all the messages from a certain group from a certain collection

requestMessages(int number) - receive the last n stored messages (private method)

MessageService class

enums

sort- Sort options

Functionality

handle the stored on RAM messages. This class is static

Properties

RamMessages - get and set the stored messages

Functions

FilterByUser(IUser user)- receive all the messages from a certain user

FilterByGroup(int groupId)- receive all the messages from a certain group

FilterByUser(IUser user, ICollection<IMessage> toFilter)- receive all the messages from a certain user and from a certain collection of messages

FilterByGroup(int groupId, ICollection<IMessage> toFilter)- receive all the messages from a certain group and from a certain collection of messages

sort(ICollection<IMessage> messages, Sort SortBy, bool descending)- Sort a message List by the time

deserialize()-deserializes data.

serialize(ICollection<IMessage> Data)-serializes data

DefaultSort(ICollection<IMessage> Data)- Sorting by time

SaveMessage(IMessage msg)- save a single message to the RAM

EditMessage(Guid ID, string newBody) - edit the message with the given guid to the new content

lastNMessages(int amount) - receive the last n stored messages

SaveLast10FromServer(string url) - request the last 10 messages from the server

UserSrvice class

Functionality

handle the stored on RAM users. This class is static

Properties

RamUsers - get and set the stored users

Functions

register(IUser user, string password) - save the given user to the ram after registration
CanRegister(IUser user) - returns if it is possible to register to the given user (not taken already)

CanLogin(IUser user, string password) - returns if it is possible to login to the given user (was registered)

deserialize()-erializes data.

serialize(ICollection<IUser> Data)- serializes data.

DefaultSort(ICollection<IUser> Data)-returns the massege list as is

IUser interface

Functionality

This is an interface that representing a user

Properties

NickName - get this user's nickname

Group_ID - get this user's nickname

Functions

Send(string msg, string url) - send a new message to the server with the requested content to the given url

logout() - logout this user

User class

Functionality

Implements IUser interface. represent a user in the chat room

Fields :

_groupID - the group ID of this user.

_nickName - the nickname of this user.

GROUP_ID – the group ID of our team from the registration sheet.

Properties

Implements the IUser's properties :

NickName - get this user's nickname

Group_ID - get this user's nickname

Functions

Send(string msg, string url) - send a new message to the server with the requested content to the given url

logout() - logout this user

Equals(object obj) - return if two users are equals (same group ID and nickname) - override object's Equals method

ToString() - return a string that represent this user - override object's ToString method

Message class

Functionality

Implements IMessage interface. represent a message in the chatroom

Fields :

- _guid** - the unique identifier of this message, the guid.
- _recivingTime** - the time this message was received by the server
- _sender** - the user that sent this message
- _body** - the content of this message

Properties

Implements the properties of IMessage :

- Date - get the receiving time
- GroupID - get the group ID of the sender
- Id - get the guid go this message
- MessageContent - get and set (private setter) the contest of this message
- UserName - get the sender's user name
- Sender - get the sender of this message (private, not from IMessage)

Functions

- IsValid(string body)** -return if the body is valid. A static method.
- Equals(object obj)** - return if two messages are equals (same guid) -
override object's Equals method
- ToString()** - return a string that represent this message - override object's
ToString method

GeneralHandler class

Fields :

_ramData- Stores a copy of the data in the ram for a quick access

Functions:

Start()- initiates the ram's saves from users stored in the disk.

Deserialize()-deserializes data.

Serialize(ICollection<T> Data)-serializes data

DefaultSort(ICollection<T> Data)-sorts the data

UpdateDisk()- updates the data stored in the disk after changes in the ram

SetRam()- setting the ram, if null.

MergeIntoFirst(ICollection<T> list2)- merge two collections to the first one without duplications.

DisplayMessage class, implements IMessage

Fields:

- MAX_LENGTH = 100 - message maximum length
- _guid - The unique identifier of the message
- _receivingTime - The time the server received the message
- _nickName - The sender name user
- _groupID - The group of the sender
- _body - The message's content

Functions:

- isValid(String body) – checks if the message's content is valid.
- ToString() – Handles how the message will be displayed.

Persistent Layer

SerializationService class

Functionality

Enable to serialize and deserialize any object to any file. Static class.

Functions

serialize(object toSerialize, string fileName) - serialize the given object to fileName

deserialize(string fileName) - deserialize an object from fileName

UserSerializationService class

Functionality

Enable to serialize and deserialize the users. This class is static.

Properties

USERS_LIST - hold the location to serialize and deserialize users from

Functions

serialize(ArrayList users) - serialized the users

deserialize() -desterilize the uses

MessageSerializationService class

Functionality

Enable to serialize and deserialize the messages. This class is static.

Properties

Messages_LIST - hold the location to serialize and deserialize messages from

Functions

serialize(ArrayList messages) - serialize the messages

deserialize() - deserialize the messages

Logging Layer

Logger class

Functionality

Operate the lodger. This class is static.

Fields :

log - holds a static field with the logger

Properties

Log - get the logger

Functions

Developer(string logMessage) - can be used to put a reference at the begetting of the message - “Developer : “ to inform that this logged message is for the developer

Maintenance(string logMessage) - can be used to put a reference at the begetting of the message - “Maintenance : “ to inform that this logged message is for the maintenance

Server(string logMessage)- can be used to put a referece at the begennig of the message – Server

MethodStart(MethodBase method) - can be used to put a state when a method was entered for debugging

MethodStart(string methodName, string className) - can be used to put a state when a method was entered for debugging

Presentation Layer

OvesrvanleObject class

Properties:

Messages-holdes the messages

messageContent- holds the message content

errorText- holds the exeption error text

mainWindowLoginRadio-is it a login or register

username- holds the username of the logged in user

groupid- holds the groupid of the logged in user

usernameBox- a text box that holds the username filter text

groupidBox- a text box that holds the groupid filter text

passwordBox – a password box that holds the user's password until the login/register action is completed

filterUsername- Boolean that holdes if the filterusername is checked

filterGroupid- Boolean that holdes if the filtergroupid is checked

filterNone- true if none of the filters are used

filterGroupString-holds the group name the messages sorted by

filterNameString- holds the name the messages sorted by

sortAscending-true if the sort type is ascending

sortDescending- true if the sort type is descending

sortOption- holds the number of the sort type

CLI class

Functionality

Operate the common-line interface. This class use the singleton design

Functions

CLI()- implements the CLI as a singleton.

initialize()-initialize(): used in order to initialize the CLI. introducing the user to the chat room instructions and allows him to interact with the relevant menu.

entranceManager()-This menu handles a client which isn't logged-in in the chat room.

selectionMenu()-This menu handles a client that has logged-in.

retrieveMessages()-Trying to retrieve last 10 messages from server, responds accordingly if the attempt was successful or not.

display20Messages()-Trying to display last 20 messages from server, responds accordingly if the attempt was successful or not.

displayUserMessages()-Trying to display all retrieved messages of a certain user, responds accordingly if the attempt was successful or not.

writeMessage()-This function allows a user to send a new message, only if under 150 chars.

logoutFunction()-Handles login-out.

menuNotification()-A message that pops-up when a user is pressing irrelevant keys (Instructions for menus).

boldingText(string text, ConsoleColor color)- Displays the text in the requested color.

login()-Handles login-in.

register()-Handles registration.

exitFunction()-Handles exit request.

Printer<T>(ICollection<T> list)- An easy way to print the relevant lists received from the ChatRoom class

Chatwindow.xaml

Fields :

- ❑ **ObservableObject bindObject**- Responsible for all the binding in the project
- ❑ **DispatcherTimer dispatcherTimer**- a timer that handles refreshing the chat
- ❑ **ICollection<IMessage> last20**- holds the last 20 messages
- ❑ **bool sortChanged**- holds a boolean true if the sort method changed
- ❑ **bool filterApplied**- holds a boolean true if a filter is applied

Functions

- ❑ **dispatcherTimer_Tick(object sender, EventArgs e)**- updates the messages showing on the screen every 2 seconds
- ❑ **sendButton_Click(object sender, RoutedEventArgs e)**- sends the message typed in the textbox
- ❑ **logoutButton_Click(object sender, RoutedEventArgs e)**- logs out
- ❑ **Printer(ICollection<IMessage> list)**- prints a message on the screen **UpdateScreen()**- updates the screen
- ❑ **RadioButton_Click(object sender, RoutedEventArgs e)**
- ❑ **ComboBox_SelectionChanged(object sender, SelectionChangedEventArgs e)**- takes care of the sort type to apply
- ❑ **Filter_Click(object sender, RoutedEventArgs e)**- takes care to clicks on the filter button
- ❑ **Apply_Click(object sender, RoutedEventArgs e)**- takes care to clicks on the apply button

Mainwindow.xaml

Functions:

- ❑ **userHandlerButton_Click(object sender, RoutedEventArgs e)**- takes care of logging in or registering a user
- ❑ **exitButton_Click(object sender, RoutedEventArgs e)**- exits the window

Hasing.cs

Fields :

- MIN_PASSWORD_LENGTH – the minimum password length
- MAX_PASSWORD_LENGTH – the maximum password length
- SALT – the addition to the string in order for the password to be encrypted.

Functions:

- GetHashCode(string inputString1) – computes hash as bytes from inputString.
- GetHashCodeString(string inputString) – convert string to hash string.
- IsValidPassword(string password) – checks if the password is valid.

Data Access Layer

Connect class

Fields:

- `_conn` – the class's `SqlConnection`.

Functions:

- `getCommand(string query)` – get `SqlCommand` object.
- `ExecuteReader(SqlCommand command)` – Execute a query that returns a table.
- `ExecuteNonQuery(SqlCommand command)` – Execute a query that does not return a value, and return amount of rows that were changed.
- `ConnectionString()` – create the connection string

MessageExcuteor class

Fields:

- `_lastMessageTime` – The `DateTime` of the last recived message.
- `Query` – an object to create the queries.

Functions:

- `AddTimeFilter(DateTime time)` – adds time filter
- `clearFilter()` – clears the query filters
- `FilterByUser(IUser user)` – receive all the messages from a certain user.
- `FilterByGroup(int groupId)` – receive all the messages from a certain group.
- `INSERT(IMessage Data, int UserID)` – add message to the DB.
- `UPDATE(IMessage msg)` – update a message.
- `getFilteredMessages()` – get filtered messages.
- `Execute(SqlCommand queryCommand)` – execute the query and return the messages that were drown from the DB.
- `prepareGroupFilter(int group)` – clear old filters and add filter by the group and by the time.

MessageQueryCreator class

Functions:

- ☐ clearFilters() – clear filter options
- ☐ addGroupFilter(int group) – add a group filter
- ☐ addUserID(int id) – add a user id to the query
- ☐ addNicknameFilter(string nickname) – add a nickname filter
- ☐ addTimeFilter(DateTime from) – add a time filter
- ☐ getQueryString() – get the query string command
- ☐ clear() – reset selections
- ☐ where() – get a “WHERE” property to the filtering options
- ☐ whereNoTime() – return the “WHERE” property without time filter
- ☐ BuildTimeFilter(string currentWhere) – add the time filter to the “WHERE” property
- ☐ JOIN() – return INNER JOIN command
- ☐ Values() – get the VALUES statements for the INSERT query
- ☐ Set() – get the SET statement for UPDATE queries

QueryCreator class

Functions:

- ☐ SETtoSELECT() – set this query type to SELECT.
- ☐ SETtoINSERT(T item) – set this query type to INSERT, and get the item to insert.
- ☐ SETtoUPDATE(T item) – set this query type to UPDATE, and get the item to update.
- ☐ clearFilters() – clear all the filter.
- ☐ getQuary() – get the query.
- ☐ addQuaryItem(T item) – add an item to the query.
- ☐ Clear() – reset the query
- ☐ addType(string type) – add a type to the query.

UserExcuteor class

Functions:

- ☐ Excute(SqlCommand query) – execute the query and return the users the WHERE statement drawn from the DB.
- ☐ INSERT(IUser user, string ebcryptedPassword) – insert a new user to the data base.
- ☐ Loginable(IUser user, string encryptedPassword) – check if a user can login, if som return it's id (return -1 if it can't login).
- ☐ canRegister(IUser user) – return if a user can register

UserQueryCreator class

Functions:

- ☐ addEncryptedPassword(string password) – add password to query.
- ☐ clearFilters() – reset filters
- ☐ getQueryString() – get the query command string.
- ☐ setToLogisterQuery() – sets the current query to check regiserable\loginable.
- ☐ canLogisterQuery() – get the query to check regiserable\loginable.
- ☐ Value() – get the VALUE part of the query.
- ☐ Where() – get the WHERE part of the query.
- ☐ Clear() – clear inputs.



