

SEVCHAT

Vsevolod Geraskin

Table of Contents

Introduction.....	3
Purpose.....	3
Technology.....	3
Functionality.....	3
Preliminary Setup.....	4
PostgreSQL.....	4
Erlang.....	5
Source code files	5
Java files.....	5
WEB files and libraries.....	6
Deployment Manual.....	8
How to build war file.....	8
Steps to access the program.....	9
User Manual.....	9
Modifying Erlang server settings.....	9
Creating chat users.....	10
Logging in and chatting.....	10
Testing.....	11
Validation.....	11
Communication.....	11
Sample Session.....	11
Discussion.....	14
How it all works.....	14
Limitations.....	15
Suggested improvements.....	15
Appendix.....	16
System Diagram.....	16

Introduction

Purpose

The purpose of the final project is to create a scalable web-based multi-user chat application. The project combines Erlang OTP with stripes framework.

Technology

The chat application uses the following components:

- to handle chat messages and send them to users, Erlang sevchat_server,
- to send and receive messages to/from Erlang server, java classes with OTP Erlang library,
- to handle clients webpage functionality, stripes framework and javascript/AJAX,
- to create and store clients logins, Erlang node settings, etc; postgresSQL database.

Functionality

The user are able to do the following:

- register themselves with their username, password, and name,
- log in with their username and password,
- view all connected users,
- chat with all connected users,
- and send private messages to selected users.

Furthermore, all messages are logged in PostgreSQL database.

Preliminary Setup

PostgreSQL

The following PostgreSQL tables and records were created by queries located in /final/queries/ folder:

chatsettings (/final/queries/createchatsettings)

```
CREATE TABLE chatsettings
(
  id serial NOT NULL,
  password character varying(28) NOT NULL,
  serverip character varying(15) NOT NULL,
  servername character varying(10) NOT NULL,
  servercookie character varying(10) NOT NULL,
  CONSTRAINT pk_chatsettings PRIMARY KEY (id )
)
```

Chatsettings table controls Erlang server settings.

Default chatsettings record with admin password 'admin1' (/final/queries/insertchatsettings)

```
INSERT INTO chatsettings(password, serverip, servername, servercookie) VALUES
('bHyjRfY/g1yzU/8VvWxeBS7Ajno=', '192.168.0.101', 'chatserver', 'sev11');
```

Important Note: chatsettings must always contain only one record with id = 1 (default password is admin1).

chatuser (/final/queries/createchatuser)

```
CREATE TABLE chatuser
(
  id serial NOT NULL,
  username character varying(10) NOT NULL,
  password character varying(28) NOT NULL,
  firstname character varying(10) NOT NULL,
  lastname character varying(10) NOT NULL,
  CONSTRAINT pk_chatuser PRIMARY KEY (id)
)
```

Chatuser table contains usernames, passwords, and names of chat users.

chatmessage (/final/queries/createchatmessage)

```
CREATE TABLE chatmessage
(
```

```

id serial NOT NULL,
usernamefrom character varying(10) NOT NULL,
usernameeto character varying(10),
chatmessage character varying(256) NOT NULL,
messagetime timestamp with time zone NOT NULL DEFAULT now(),
CONSTRAINT pk_chatmessage PRIMARY KEY (id )
)

```

Chatmessage table stores all chat messages.

Erlang

Erlang chat server and client are located in /final/erlang/ folder.

Starting erlang chat server (/final/erlang/sevchat_server.erl)

```

vgeraskin@haskell:~/erlang$ erl -name 'chatserver@142.232.17.17' -setcookie sev11
(chatserver@142.232.17.17)1> c(sevchat_server).
{ok,sevchat_server}
(chatserver@142.232.17.17)2> sevchat_server:start().
{ok,<0.48.0>}

```

Starting erlang chat client (/final/erlang/sevchat_client.erl)

```

vgeraskin@haskell:~/erlang$ erl -name 'chatclient@142.232.17.17' -setcookie sev11
(chatclient@142.232.17.17)1> c(sevchat_client).
{ok,sevchat_client}
(chatclient@142.232.17.17)2> net_adm:ping('chatserver@142.232.17.17').
pong
(chatclient@142.232.17.17)3> sevchat_client:start().
<0.54.0> received [{<0.54.0>,"sevchat"},], "sevchat"
<0.54.0>

```

Important Note: both erlang server and client name, and server cookie must be a maximum of 10 characters because of the programmer's design decision.

Source code files

All source files are located in /final/ (src and web) folders.

Java files

```
[root@linusaur src]# ls -LR
```

```
..
```

```
action dao ext META-INF model nonext StripesResources.properties
```

./action:

BaseActionBean.java LoginActionBean.java NotifyListActionBean.java
chat LogoutActionBean.java RegisterActionBean.java
ChatActionBean.java NotifyActionBean.java SettingsActionBean.java

./action/chat:

ChatClient.java ChatNotify.java ChatNotifyList.java NotifyThread.java

./dao:

ChatMessageDao.java ChatSettingsDao.java ChatUserDao.java Dao.java impl

./dao/impl:

BaseDaoImpl.java ChatSettingsDaoImpl.java
ChatMessageDaoImpl.java ChatUserDaoImpl.java

./ext:

LoginInterceptor.java MyActionBeanContext.java

./META-INF:

MANIFEST.MF persistence.xml

./model:

ChatMessage.java ChatSettings.java ChatUser.java

./nonext:

PasswordTypeConverter.java

WEB files and libraries

[root@linusaur web]# ls -LR

..

css images index.jsp js jsp WEB-INF

./css:

style.css style_ie.css

./images:

sevchat.jpg

./js:

jquery.js prototype.js

./jsp:

chat.jsp login.jsp reg_complete.jsp set_complete.jsp

common notify.jsp register.jsp settings.jsp

errors.jsp notifylist.jsp result.jsp

./jsp/common:

taglibs.jsp

./WEB-INF:

lib web.xml

./WEB-INF/lib:

antlr-2.7.6.jar hibernate-entitymanager.jar

asm-attrs.jar hibernate-validator.jar

asm.jar javassist.jar

c3p0-0.9.1.jar jboss-archive-browsing.jar

cglib-2.1.3.jar jstl.jar

commons-collections-2.1.1.jar jta.jar

commons-logging.jar OtpErlang.jar

dom4j-1.6.1.jar postgresql-9.1-901.jdbc4.jar

ehcache-1.2.3.jar servlet-api-2.5-6.1.5.jar

ejb3-persistence.jar standard.jar

hibernate3.jar stripersist.jar
hibernate-annotations.jar stripes.jar
hibernate-commons-annotations.jar

Deployment Manual

How to build war file

Folder /final/ contains build.xml for ant build. Please use ant command as follows:

- ant clean: deletes built files
- ant: builds classes
- ant war: generates war file

Ant examples

```
[root@linusaur final]# ant clean
```

```
Buildfile: /home/sev/bcit/8061/final/build.xml
```

```
clean:
```

```
[delete] Deleting directory /home/sev/bcit/8061/final/build/classes
```

```
[delete] Deleting directory /home/sev/bcit/8061/final/dist
```

```
BUILD SUCCESSFUL
```

```
Total time: 0 seconds
```

```
[root@linusaur final]# ant
```

```
Buildfile: /home/sev/bcit/8061/final/build.xml
```

```
build-subprojects:
```

```
init:
```

```
[mkdir] Created dir: /home/sev/bcit/8061/final/build/classes
```

```
[copy] Copying 3 files to /home/sev/bcit/8061/final/build/classes
```

```
build-project:
```

```
[echo] final: /home/sev/bcit/8061/final/build.xml
```

```
[javac] /home/sev/bcit/8061/final/build.xml:63: warning: 'includeantruntime' was not set, defaulting  
to build.sysclasspath=last; set to false for repeatable builds
```

```
[javac] Compiling 26 source files to /home/sev/bcit/8061/final/build/classes
```

```
build:
```

```
BUILD SUCCESSFUL
```

```
Total time: 2 seconds
```



```
[root@linusaur final]# ant war
Buildfile: /home/sev/bcit/8061/final/build.xml

war:
    [war] Building war: /home/sev/bcit/8061/final/dist/final.war

BUILD SUCCESSFUL
Total time: 2 seconds
```

Steps to access the program

The program can be built in the following steps:

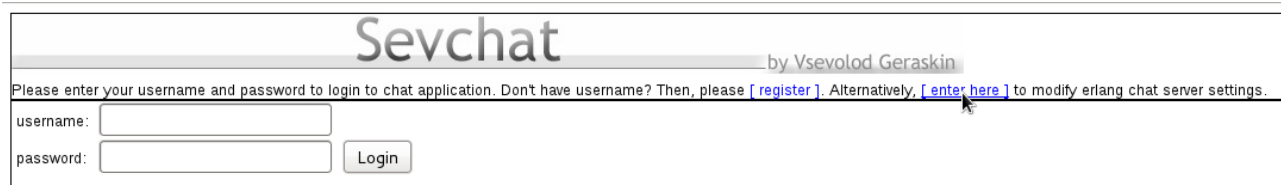
- unzip vgeraskin_final.zip,
- cd to ~/final folder and run ant clean, ant, and ant war commands,
- start Tomcat from ~/apachetomcat7.0.23/bin folder (bash catalina.sh start),
- copy the WAR file to the tomcat webapps directory,
- in web browser, go to <http://142.232.17.17:8811/final/>

User Manual

Prior to using the program, please ensure that all tables are created, insertchatsettings query is ran, and erlang sevchat_server is compiled and running.

Modifying Erlang server settings

1. Click on the link to go to Erlang server settings:



2. Enter Erlang chat server information (use admin1 as password):



3. Save the settings and go back to the main screen:

Sevchat

by Vsevolod Geraskin

Erlang settings update complete. You may now [\[login \]](#).

Creating chat users

1. Click on register link to go to register new chat user screen:

Sevchat

by Vsevolod Geraskin

Please enter your username and password to login to chat application. Don't have username? Then, please [\[register \]](#). Alternatively, [\[enter here \]](#) to modify erlang chat server settings.

username:

password:

2. Register yourself:

Sevchat

by Vsevolod Geraskin

Please enter your registration details.

First Name:

Last Name:

User Name:

Password:

Confirm password:

3. Save the user and go back to the main screen:

Sevchat

by Vsevolod Geraskin

Registration complete. You may now [\[login \]](#).

Logging in and chatting

1. Login with your username and password:

Sevchat

by Vsevolod Geraskin

Please enter your username and password to login to chat application. Don't have username? Then, please [\[register \]](#). Alternatively, [\[enter here \]](#) to modify erlang chat server settings.

username:

password:

2. Connect to the running Erlang server (instructions for starting Erlang server are on page 5):

<div style="text-align: center;"> <h1>Sevchat</h1> <p>by Vsevolod Geraskin</p> </div>	
Welcome, sev! [logout] <input type="button" value="Connect"/> <input type="button" value="Disconnect"/> say to everyone: <input type="text"/> <input type="button" value="Chat"/>	Connected clients:

3. Type a message in the checkbox to chat, click on the connected user to send a private message (whisper), or disconnect from the server:

<div style="text-align: center;"> <h1>Sevchat</h1> <p>by Vsevolod Geraskin</p> </div>	
Welcome, sev! [logout] <input type="button" value="Connect"/> <input type="button" value="Disconnect"/> say to everyone: <input type="text" value="hello, world!"/> <input type="button" value="Chat"/> you say hello, world!	Connected clients:
sev says at (2012-03-22 13:10:55.758) hello, world! status says at (2012-03-22 13:10:02.302) sev joins the chat status whispers to sev at (2012-03-22 13:10:02.29) connection established with chatserver@142.232.17.17 status whispers to sev at (2012-03-22 13:10:02.243) please wait, connecting to chatserver@142.232.17.17...	
	[sev] [sevchat]

Testing

Validation

All input forms have basic validation for non-blank input and a certain number of characters.

Communication

The test is setup as follows:

- sevchat_server.erl running on 142.232.17.17 (chatserver)
- sevchat_client.erl running on 142.232.17.17 (chatclient)
- Fedora Firefox chat client at 174.6.236.28 (192.168.0.101) (username: sev password: sev11)
- Windows XP Firefox chatclient at 174.6.236.28 (192.168.0.103) (username: melinda password: melinda11)

Important Note: Please see page 5 on starting erlang server and client and page 9 for erlang web settings.

Sample Session

This session demonstrates the sample three-way conversation between sev, melinda, and erlang chatclient.

Erlang chatclient@142.232.17.17

```
(chatclient@142.232.17.17)2> sevchat_client:start().
<0.49.0> received [{<0.49.0>,"sevchat"},"sevchat"]
<0.49.0>
<0.49.0> received [{<6666.1.0>,"sev"},{<0.49.0>,"sevchat"},"sev"]
<0.49.0> received [{<6667.1.0>,"melinda",
    {<0.49.0>,"sevchat"},
    {<6666.1.0>,"sev"}],
    "melinda"}
<0.49.0> received {"Hello, world!","sev",[]}
<0.49.0> received {"Hi!!!!","melinda",[]}
<0.49.0> received {"hey erlang client","sev","sevchat"}
```

Web user melinda:

The screenshot shows a Mozilla Firefox browser window with the title "Vsevolod Geraskin - COMP 8061 Final Project". The address bar shows the URL "142.232.17.17:8811/final/Chat.action". The page content includes a header with the "Sevchat" logo and "by Vsevolod Geraskin". Below the header, there is a welcome message "Welcome, melinda!" with a "logout" link. There are "Connect" and "Disconnect" buttons. A text input field contains "Hi!!!!" and a "Chat" button. A message "disconnect button clicked" is displayed. The main chat area shows a log of messages with timestamps and usernames: "status" joins and leaves the chat, "sev" joins and leaves, "melinda" leaves, "sev" says "hello, world!", "melinda" says "Hi!!!!", "sev" says "Hello, world!", "status" reports connection establishment and disconnection, and "sev" joins the chat.

Vsevolod Geraskin - COMP 8061 Final Project - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Vsevolod Geraskin - COMP 8061 Final Project

142.232.17.17:8811/final/Chat.action

Sevchat

by Vsevolod Geraskin

Welcome, **melinda**! [logout] [Connect] [Disconnect] say to everyone: [Chat] disconnect button clicked

status says at (2012-03-22 15:18:55.49) sev joins the chat

status says at (2012-03-22 15:17:58.638) sev leaves the chat

status whispers to **melinda** at (2012-03-22 15:17:48.006) successfully disconnected

status says at (2012-03-22 15:17:47.997) melinda leaves the chat

status whispers to **melinda** at (2012-03-22 15:17:47.995) disconnecting...

sev says at (2012-03-22 15:12:53.278) hello, world!

sev whispers to **melinda** at (2012-03-22 15:10:44.68) hey melinda

melinda says at (2012-03-22 15:10:20.622) Hi!!!!

sev says at (2012-03-22 15:10:09.897) Hello, world!

status says at (2012-03-22 15:09:58.944) melinda joins the chat

status whispers to **melinda** at (2012-03-22 15:09:58.935) connection established with chatserver@142.232.17.17

status whispers to **melinda** at (2012-03-22 15:09:58.925) please wait, connecting to chatserver@142.232.17.17...

status says at (2012-03-22 15:09:39.68) sev joins the chat

Web user sev:

Activities

Firefox

Thu 15:20

Vsevolod Geraskin

Vsevolod Geraskin - COMP 8061 Final Project - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Vsevolod Geraskin - COMP 80...

142.232.17.17:8811/final/Chat.action?cancel=

Google

Sevchat

by Vsevolod Geraskin

Welcome, **sev** [[logout](#)] say to everyone: connect button clicked

status says at (2012-03-22 15:18:55.49) sev joins the chat

status whispers to **sev** at (2012-03-22 15:18:55.474) connection established with chatserver@142.232.17.17

status whispers to **sev** at (2012-03-22 15:18:55.452) please wait, connecting to chatserver@142.232.17.17...

status whispers to **sev** at (2012-03-22 15:17:58.647) successfully disconnected

status says at (2012-03-22 15:17:58.638) sev leaves the chat

status whispers to **sev** at (2012-03-22 15:17:58.637) disconnecting...

status says at (2012-03-22 15:17:47.997) melinda leaves the chat

sev says at (2012-03-22 15:12:53.278) hello, world!

sev whispers to **sevchat** at (2012-03-22 15:10:57.062) hey erlang client

sev whispers to **melinda** at (2012-03-22 15:10:44.68) hey melinda

melinda says at (2012-03-22 15:10:20.622) H!!!!

sev says at (2012-03-22 15:10:09.897) Hello, world!

status says at (2012-03-22 15:09:58.944) melinda joins the chat

status says at (2012-03-22 15:09:39.68) sev joins the chat

status whispers to **sev** at (2012-03-22 15:09:39.671) connection established with chatserver@142.232.17.17

status whispers to **sev** at (2012-03-22 15:09:39.631) please wait, connecting to chatserver@142.232.17.17...

Connected clients:
[\[sev \]](#)
[\[sevchat \]](#)

Find:

< Previous > Next 🚩 Highlight all ☐ Match case

Furthermore, we can try sending messages on behalf of the webusers directly from the server to ensure that the server is involved in the messaging:

Erlang chatserver@142.232.17.17

```
(chatserver@142.232.17.17)7> sevchat_server:whisper("shhhh!", "melinda", "sev").
okwhisper
(chatserver@142.232.17.17)8> sevchat_server:whisper("shhhh!", "sev", "melinda").
okwhisper
(chatserver@142.232.17.17)9> sevchat_server:broadcast("Hello!!!", "sev").
okbroadcast
```

Important note: this will only work if the usernames specified (2nd and 3rd arguments) are valid, connected web usernames.

Web user sev:

Activities Firefox Thu 15:50

Vsevolod Geraskin - COMP 8061 Final Project - Moz

File Edit View History Bookmarks Tools Help

Vsevolod Geraskin - COMP ... Hotmail - sgeraskin@hotmail... +

142.232.17.17:8811/final/Chat.action

Sevchat

by Vsevolod Geraskin

Welcome, **sev**! [logout] say to everyone: connect button clicked

sev says at (2012-03-22 15:47:50.438) Hello!!!

sev whispers to **melinda** at (2012-03-22 15:47:20.168) shhhh!

melinda whispers to **sev** at (2012-03-22 15:47:05.378) shhhh!

status says at (2012-03-22 15:46:27.313) sev joins the chat

status whispers to **sev** at (2012-03-22 15:46:27.305) connection established with chatserver@142.232.17.17

status whispers to **sev** at (2012-03-22 15:46:27.295) please wait, connecting to chatserver@142.232.17.17...

Discussion

How it all works

Erlang sevchat_server keeps a list of all connected clients - their names and pids, and is able to look up clients by name for each messaging function.

When a chat user logs in the system, it creates a new ChatClient object in session memory for that specific user. ChatClient uses Java Erlang OTP library to connect to Erlang chat server using the

settings that are stored in chatsettings table. Erlang OTP library allows us to create an equivalent of erlang shell on the server for each web client, something like “erl -name 'chatusername@serverip' -setcookie 'servercookie'”. Furthermore, upon connecting, a listen NotifyThread is created for each client. Each thread is blocked waiting for RPC call from the Erlang server.

Ajax is used for partial refreshing of messages (notify.jsp) and client list (notifylist.jsp) components and echoing user actions on the screen. User actions are echoed to the right of the Chat button. The messages and client list components are constantly refreshed at certain intervals. Messages are stored in the chatmessage table, while a client list is a static list stored in the memory.

Messages (NotifyActionBean) and client list (NotifyListActionBean) action beans provide call back interfaces for ChatClient and NotifyThread objects. Upon receiving a response from the server, a call back is made to the appropriate action bean and the appropriate action is taken: either the message is written to chatmessage, or client list is updated.

Limitations

The project raised concerns regarding memory and multithreading relating to designing web-based applications. The design and implementation of the project requires each web client to create a listening thread that waits for an RPC call from the Erlang server. While this is not a problem in standalone software applications where only one thread is created per client program, this is a potential issue in web applications. All threads and objects are created on the web server, and, thus, a server could potentially run out of memory if enough clients are simultaneously connected.

The other discovered issue is lack of thread safety in web user sessions and action beans. Since action beans are destroyed and recreated upon every resolution, I originally intended to write message callbacks from the Erlang server directly to session queue for every user. However, this design proved impossible due to session returning nulls in multi-threaded environment. In fact, only stripesist with certain configuration was thread safe and, thus, I could only write the messages to the database upon receiving a callback. As of right now, to prevent duplicate messages, only a client thread that has username that corresponds to received sender's username in the message sends a callback to write a message to the database, while others send a nowrite callback. This limits a web chat application user to seeing messages sent only either from connected web users, or from an erlang server with Name or From parameter equalling to one of the connected web usernames.

Suggested improvements

The obvious improvements to increase scalability would be to use a single static multiplex, such as select, to handle RPC calls from the Erlang server. This multiplex would have to keep track of all the call back clients and be able to identify and send the message from the server to the appropriate client.

Furthermore, such single-threaded approach might allow us the original design, where a message queue would be stored in the user session rather than a database and written to the database in blocks. This might reduce the number of reads/writes to the database, potentially improving performance.

Appendix

System Diagram

