Outline

- Introduction
- Database Interface Portion
- Java Programs
  - ChatServer - ServerSocket, ChatDealer
  - ChatClient
- Integration
- Last Viewgraphs Covering Creating ChatPlayer

Introduction

Objectives
Tasks
Versions

Objectives - Casino IM System

- Casino Database Key Points…
  - Table:Game name:Chat
  - Generate Queries, JavaBeans to Process Queries
- Jstudent0.chatAlpha
  - Supports 1 Client, String Messages, Echo Server,
  - Message oo is Over and Out
- Jstudent0.chatBeta
  - Supports Multiple Clients, ChatPacket Message
- Integration
  - JavaWebStart or Applet ChatServer

Chat Tasks

- Database Tasks…
  - Create Required Database Queries
  - Implement Derby on NetBeans – Glassfish 2.0
  - Create JavaBeans, Implement all DB, Junit Test
- Java Programs
  - ChatServer Web Start serverSocket accept
  - ChatDealer Client Socket Manages Clients
  - ChatClient Basic Client Application Interface
  - ChatPlayer Web Start or Applet GUI Client socket Interface
- Integration
  - Web Page or JSF Front End
  - JSP Files to Database Beans
  - ChatServer (Dealer) Calls JSP Files

Database Interface

Tasks
DB Upfront Concerns
Queries
NetBeans – Java Derby System
JavaBean

5

Database Tasks…

- Create SQL Queries
- Setup Derby on Netbeans/Glassfish V2
  - Create JavaBean w String Array to Create Tables/Insert Data
- JavaBean
  - Constructor Connect to Database
  - Method to Create & Drop Casino Database;
  - JUnit Testing clear gamePlayer, gameTable
  - Methods Send Chat Both Queries & Data Manipulation
  - Multiple Ways to Return Query Results, int, String and html
- JSP Program
  - Check Casino Table:gameTable for ActiveChat Server
  - Login chatClient add user to Casino Table:gamePlayer

Database Concerns

- Always Protect Against SQL Injection
- Reset Database (We have Creation Strings)
- Tests All Queries In PHPMyAdmin First
- Using Java DB Derby (Lightweight)
  - Java.sql.Types.

Database Goals

- Complexity inside Bean or Java Program
  - Keep JSP File Simple
- Loading Driver/URL Connection Processing Query Should be In One Spot
- Jsp File Loads Beans Query Calls
- Jsp File (Beans Connect) send Queries for Java Program

Desired Queries…

1) Get gameID from Table:game for name="Chat"
2) get gameTableID of active chat Server
3) get active chat Server IP
4) Insert chat Server w IP into gameTable
5) Update gameTable when chat Server exits

6) Login Email/password get playerID from Table:player
7) Insert player w IP, playerID, startTime into gamePlayer

8) List active gamePlayers

- JavaBean Method That Displays Casino Contents

Queries 1..4

```
# 1) gameID chat Server, currently 5
     int getchatID( );
Select gameID from game where name="Chat";

#2) active chat Server gameTableID
    int getchatTableID( );
    select gameTableID from gameTable where
    gameID = getchatID() and endTime is null;

#3) active chat Server IP
      String getchatServerIP( ); - may become java.net.Something later on
Select IP from gameTable  where gameTableID = getchatTableID();

# 4) Insert chat server into gameTable
      Boolean insertChat( String ip  );
INSERT INTO `gameTable` (`gameTableID`, `gameID`, `casinoID`,
`startTime`, `endTime`, `IP`) VALUES ( NULL, '5', '100', now(), NULL, ip );
```


Queries 5..7

```
 # 5) update gameTable when chatServer exits
   Boolean updateChat( );
UPDATE `gameTable` SET `endTime` = now()
WHERE `gameTableID` = number is From getchatTableID() ;

#6) Map from Email/password to playerID assume jstudent0
# int getplayerID( String email, String passwd )
 SELECT playerID FROM `player` WHERE `email` =
"jstudent0@cim.saddleback.edu" and password = sha1("cs4b" );

# 7) insert player into gamePlayer
#  insertPlayer( int playerID, String ip )
INSERT INTO `gamePlayer`
 (`gameTableID`,`playerID`,`netGain`, `rounds`,`gamesWon`,`IP`)
VALUES (gTIDabove, '6', '0', '0', '0', 'ipaddress');
```


NetBeans/Glassfish V2 Derby

- In NetBeans Tools → Servers
  - Make Sure Tab Options (DB Enabled)
- Add derby.jar to ClassPath for Compile
  - Right^Click Project, Properties
  - For JSP Add…
    Libraries
    glassfishV2installFolder/javadb/lib/derby.jar
  - For Junit/Main JavaBeans Add…
    glassfishV2InstallFolder/javadb/lib/derbyrun.jar

Java Derby

- String driver = "org.apache.derby.jdbc.ClientDriver"
- String connectionURL = "jdbc:derby://localhost:1527/casino;create=true"
- Window → Services → Databases
  - Right^Click JavaDB to Manually Start Server
    - You may Have Problems with Auto Start
  - Right Click to Create DB w username/password

Create Tables

- Derby Programmers Lightweight Java DB
- Write JavaBean, w Create Tables Strings
- PHPMyAdmin Export as Plan as Possible
- Derby REF: http://db.apache.org/derby/docs/dev/ref/
- Use statement.execute("For create/insert");
  - Output Windows List Column of SQL Syntax Error
- Limited Types, Modifiers, Refer to java.sql.types
- Partial Database Creation String at http://cim.saddleback.edu/casino

Derby Kludges

- Create table if not exist – just use create table
- Don't Use Backquotes `,
- But use Single Quotes especially for 'tom@saddleback' or '1.2.2'
- No auto_increment
  use generated always as identity (START WITH n, INCREMENT BY 1)
  - On insert Do not give values for identity
  - note n is table dependent
- No int(11) or unsigned just use int
- Use TIMESTAMP instead of DATETIME,
- use CURRENT_TIMESTAMP instead of now() (cim supports both)
- No sha1, password straight text or try java.security.MessageDigest
  - I could get sha1 to match the DB encryption so I used straight text
  - Byte, Unicode, Char Inconsistency

Create Strings Java Derby

```
public static final String tables[] = { "agent", "casino", "casinoGame", "game", "gamePlayer", "gameTable", "player" };

  public static  final String create[] =    {
" CREATE TABLE casino (   casinoID int NOT NULL " +
" generated always as identity (START WITH 100, INCREMENT BY 1), " +
"  name varchar(32) NOT NULL ,   url varchar(64) NOT NULL, " +
" playerID int NOT NULL,  description varchar(64) NOT NULL," +
"  PRIMARY KEY  (casinoID) ) ",

" INSERT INTO casino ( name, url, playerID, description) VALUES" +
"( 'Greatest Ever', 'http://cim.saddleback.edu/~jstudent0/casino', 3, 'Joe Students Great
     Casino'), " +
"( 'Disney', 'http://cim.saddleback.edu/~mmouse0', 4, 'Disneylands only Casino - Except for
     Cruises') " ,

"CREATE TABLE  game (   gameID int NOT NULL,  name varchar(16) NOT NULL, " +
 "description varchar(64)  NOT NULL,   rules varchar(64) ,  PRIMARY KEY  (gameID)  ) " ,

 "INSERT INTO game (gameID, name, description, rules) VALUES " +
 "(1, 'Blackjack21', 'Blackjack 21', 'Push on Ties, no 5 Card Charlie, pays 2:1 on 21 (two cards)'), "
     +
"(2, 'VideoPoker', 'Casino Video Poker', 'Standard Vegas Ruls'), " +
"(3, 'Craps', 'Craps 7 - 11, 2 - 3 or 12', 'Standard Rules'), "+
"(5, 'Chat', 'IM Chat using Direct Play Ports', 'IP of Client should be in gamePlayer '), " ,

 "CREATE TABLE gameTable ( gameTableID int NOT NULL " +
" generated always  as identity (START WITH 2000, INCREMENT BY 1), "+
"gameID int NOT NULL,   casinoID int NOT NULL,   startTime TIMESTAMP NOT NULL, "  +
"endTime TIMESTAMP default NULL,  IP varchar(32) NOT NULL, PRIMARY KEY  (gameTableID) )",

 "INSERT INTO gameTable ( gameID, casinoID, startTime, endTime, IP) VALUES " +
"( 1, 100, '2009-03-18 22:07:18', '2009-03-19 22:07:22 ",
```

DB JavaBean

- createCasinoDB.java Two Static String Arrays
  - String tables[] Array of Casino Table Names
    - Can Only drop one table at a time in Derby
  - String create[] = { String Array to Create and Insert… }
  - Incomplete Strings at http://cim.saddleback.edu/casino
- ChatDatabase.java
  - Bean with both Drivers for CIM and Derby
    - Cim are Commented Out
- Use Main/JSP File For Testing But JUnit Testing is Better

Table: gamePlayer

- Players Be Both in chat and a game
- For Games set credit to -1,
  - Prohibit a Starting a Second Game
- Login.jsp
  - Get username(email) and password
  - Use Query to Validate Login (Three Attempts)
  - Compare username/password to Table:Player
  - Generate code for applet/Java Web Start
  - Include Player info as Applet/Web Start Parameter

Java Executables

Client Message oo is Over and Out

Identify Objectives/Concerns - Schematic

Version jstudent0.chatAlpha
ChatServer.java (has ChatDealer) &
ChatClient.Java

Review Java I/O

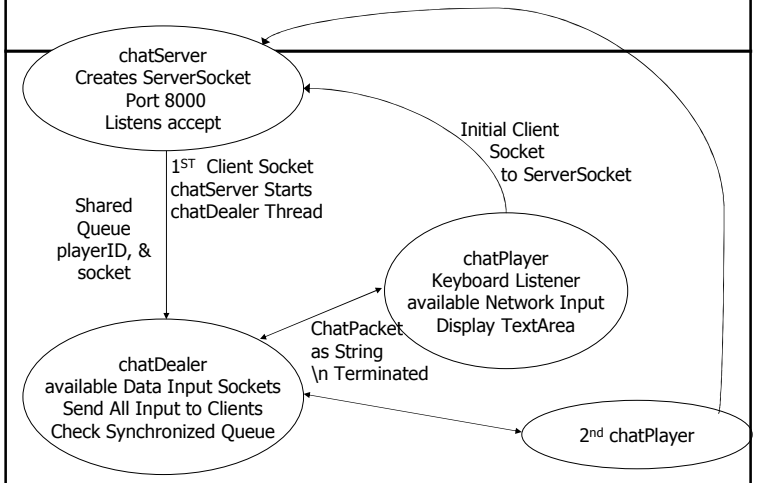
- Package java.io.*
- All IO Works Off a Buffer in Memory
- Stream Connects Buffer to Data Source
- Socket.getInputStream( )
  - Connects Buffer to a Socket

Schematic

chatServer
Creates ServerSocket
Port 8000
Listens accept

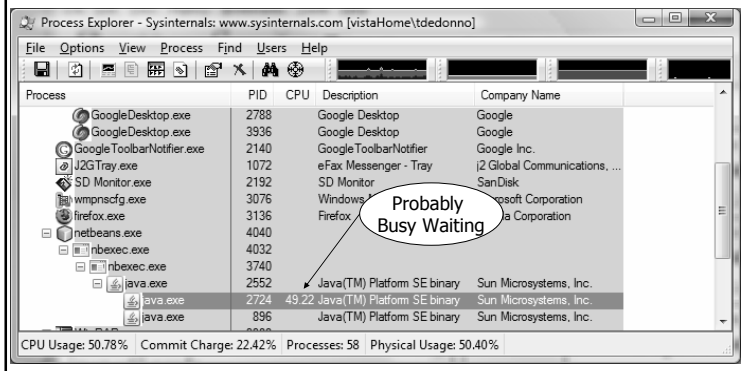
- Only One Chat Server Active at a Time
  - Responsible for All DB Inserts
  - IP of ChatServer Store in Table:gameTable
  - ChatServer Waits for Client Socket Startup Request
- ChatServer
  - ChatDealer inner class or ChatServer
  - chatDealer updates gameTable
  - One ChatDealer Mutliple chatClient
  - ChatServer Starts ChatDealer, if no ChatDealer Thread Active
  - P/C ChatServer Add Client Socket to Queue, ChatDealer Removes
- Using Class ChatPacket Protocol

Schematic



Controlling Threads

- Download Microsoft Process Explorer



Busy-Waiting Concerns

- Prohibit Busy Waiting
  - Input.readObject( )….
    - Block until you have Data, causes busy Waiting
    - No input Data, Thread Eats Up CPU Checking
- Deadlock Avoidance
  - Two Threads Waiting on Event that will never
  - Dealer expect Player to talk first
  - Player expect Dealer to talk first
  - Waiting on Incomplete Packet (need \n)

Solutions – Busy Waiting

- Option 1 Protocol, Thread.sleep w available();
  - Pause Between Checking Available
  - Available()
    - Read String Representing ChatPacket Object
    - Must Send ChatPacket then flush()
    - Well Behaved Client Avoid Deadlock
    - Could Use Separate Thread for Each Client
- Option 2 Java Non-Blocking I/O
  - java.nio.Buffer, java.nio.channels.SocketChannetl
  - Extend Buffer ChatPacket Object → Bytes
  - Superior (Bad Pack in 1) can cause Problems

Version Summary

- Jstudent0.chatAlpha
  - Support Only One Client
  - Java Applications Text Based Interfaces
  - Echo Server – Echo Back the Client String
- Jstudent0.chatBeta
  - Support Multiple Clients
  - Send Strings ChatPacket Format
- Integration Phase

jstudent0.chatAlpha
Proof of Principle - Prototype

- ChatServer.java ServerSocket(Port:8000)
  - Prints IP Address - Waits for Accepts
  - On Accept Start ChatDealer Thread, returns to Wait
- ChatDealer Thread – Echo Server
  - Get Socket from Chatserver, Socket → I/O Buffered Streams
  - Checks Available(), No Input sleept
- ChatClient
  - System.in.read at Keyboard (available)
  - No System.in.available or network available Sleep main Thread
- Create String Message w '\n' terminator then Flush

Public class ChatServer {

```
public static void main( String args[] )
   {
    ChatServer chatServer = new ChatServer( );
    ServerSocket server = (ServerSocket)null;

        try {
            server = new ServerSocket(8000);
            System.out.println( "InetAddress:"  + server.getInetAddress() );
        } catch (IOException ex) { logger… System.exit( 0 );   }
        ChatDealer chatDealer = (ChatDealer)null;

            while( true ) {
                  Socket clientSocket;
                   try {
                       clientSocket = server.accept();
                   } catch (IOException ex) { logger… continue; }

                   if( chatDealer == (ChatDealer)null )
                      //start chatDealer, chatDealer is a thread.
                      chatDealer = new ChatDealer( clientSocket  );
                   } //end While True
    }
```

private static class ChatDealer extends Thread {

```
private ObjectOutputStream out;
private ObjectInputStream in;

   public ChatDealer( Socket s )
   {
         try {
             out = new ObjectOutputStream(s.getOutputStream());
             in = new ObjectInputStream( s.getInputStream() );
         } catch (IOException ex) {
         Logger.getLogger(ChatDealer.class.getName()).log(Level.SEVERE,
          null, ex);  }

      System.out.println(
      "Chat Dealer Constructor I/O Buffer Created starting thread");
      start( );  //thread Start and call public void run
   }

//ChatDealer inner class of ChatServer
```


public void run( ) { //ChatDealer

```
StringBuffer str = new StringBuffer( );  char c;
while (true) {
        try {
                if (in.available() > 0 ) {
                   System.out.print("Bytes Available:" + in.available());

                   while(  (c = in.readChar()) != '\n'  )  str.append(  c  );
                   System.out.println( "Got : " + str.toString() );

                   if( str.toString().equalsIgnoreCase("oo") ) break;

                   out.writeChars( str.toString() );
                   out.writeChar(  '\n'  );    out.flush( );
                   str.delete( 0, str.length() );
                   }

                if( in.available() < 1 ) { System.out.println( "Sleeping" );  sleep( 2000 ); }
             } catch (Exception ex)
             { Logger.getLogger(ChatDealer.class.getName()).log(Level.SEVERE, null, ex); }
          } //end while True

System.out.println( "Closing Table" );
try {  in.close();  out.close(); socket.close();  }
catch (IOException ex) { logger…  }
```


public class ChatClient {

```
public static void main( String args[] ) throws IOException
{
    Socket socket = (Socket)null;
    ObjectOutputStream out = (ObjectOutputStream)null;
    ObjectInputStream in = (ObjectInputStream)null;

    try {
        socket = new Socket(  (args.length>0?args[0]:"0.0.0.0"), 8000);
    } catch (UnknownHostException ex) {
        Logger.getLogger(ChatClient.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(ChatClient.class.getName()).log(Level.SEVERE, null, ex);
    }

    System.out.println( "Client has Soekct Connection " );

   try {
        out = new ObjectOutputStream( socket.getOutputStream() );
        in = new ObjectInputStream( socket.getInputStream() );
    }
    catch (IOException ex) {
     System.out.println( "Exception : " + ex.getMessage() ); System.exit( 0 );  }
```


Client Socket R/W Loop

```
StringBuffer keysTyped = new StringBuffer();      char  c = 'b';
while( true ) {
       try {
       while( System.in.available() > 0  ) {
            if( (c = (char) System.in.read()) == '\n' )   {
               out.writeChars( keysTyped.toString() );
               out.writeChar( c );      out.flush( );

               if(  (keysTyped.toString()).equalsIgnoreCase( "oo")  )  {
                    System.out.println( "Closing up Connection and I/O ");
                    in.close(); out.close(); socket.close(); System.exit( 0 );
                    }
               keysTyped.delete( 0, keysTyped.length() );
               break; //check for input we have a delay on the return
               }
            else keysTyped.append(  (char)c );
         } //end while( System.ini.available()

        // without sleep this will cause a busy wait
        if( in.available() >= 2 ) {
           System.out.print( "Getting Message: " );
           while( (c = in.readChar()) != '\n'  ) System.out.print(  (char)c );
           System.out.println(  );
        }
         if( in.available() < 5 && System.in.available() == 0 ) Thread.sleep( 1000 );
    } catch( Exception ex ) { Logger.getLogger(ChatClient.class.getName()).log(Level.SEVERE, null, ex); }
} //end while true
```

Jstudent0.chatBeta

Client Message oo is Over and Out

Implement a ChatPacket

Support Multiple Clients

Concurrent Queue

33

chatPacket.java Bean

- R/W Objects Were Failing in Alpha
- Server was not Updating?
- ChatPacket w toString, fromString

- public class ChatPacket implements Serializable {

- private int playerID; //casino databse playerID
- private long gmt; //msec since epoch
- private String IP; //ip of message source
- private String message;
- //Hello to initialize; oo over and out terminate

ChatDealer w Multiple Clients

- Option 1:
  - Producer/Consumer Queue of New Sockets
    - ChatServer Producer, ChatDealer is Consumer
  - ChatDealer ClientInfo Vector
    - Step Through Vector to Check Input
    - Enumerate Vector Send Output to Each Client
- Option 2: java.net.MulticastSocket
  - Uses DatagramSocket and DatagramPacket
    - Datagram is a packet routed each time
  - More Advance Version could consider mutlicastSocket and nio.channels.Datagram

public class ClientInfo {

```
Socket socket;
 ObjectOutputStream out;
 ObjectInputStream in;  //add PlayerInfo later on

 public ClientInfo ( Socket s )   {
    socket = s;
    try {
    out = new ObjectOutputStream( s.getOutputStream() );
    in = new ObjectInputStream( s.getInputStream() );
    }
    catch( Exception ex )
   { Logger.getLogger(ClientInfo.class.getName()).log(Level.SEVERE, null, ex); }
    }

public void close( )  {
      try {
         out.close();  in.close(); socket.close();
       } catch (IOException ex)
      {Logger.getLogger(ChatServer.class.getName()).log(Level.SEVERE, null, ex); }
   }
} //end class ClientInfo
```

Jstudent0.chatBeta.ChatServer

- Modifier Synchronous & Thread Locks
- ConcurrentLinkedQueue<Socket> socketQueue;
  - Package java.util.concurrent
  - ChatServer Produces sockets
  - ChatDealer Consumes sockets
- Client Connects to ChatServer
  - Checks chatDealer null or ! chatDealer.isAlive()
    - Starts Up chatDealer Thread
  - Else Add Client Socket to socketQueue

Thread ChatDealer

- Used Vector<ClientInfo> c
  - Keeps Track of all Clients
- Reads Vector One a a Time Check for Input
  - Reminder: Stick with ChatPacket Protocol
  - Send flush right after ChatPacket.toString()
- If it Finds Input, Echos to All Clients
- After Round it Consumes socketQueue

Casino Tasks…

- Write a ChatPlayer
  - ChatClient with a Web GUI
- Swing GUI
  - Each Element is a Thread
  - JTextArea/JTextFields Throws Events
  - Listener Instead Worrying About Blocking
- ChatServer is Available (Change Package)
  - CIM/Tomcat Single jstudent0.ChatServer

Jstudent0.chatGM

GM Gold Medal
Code Released to Best Clients

Bullet-Proof Code
Avoid Spaghetti Code
GUI Interface

40

Bullet-Proof Code

- Solid Code
  - Will not Crash, Performs at 99+%
  - Work Under All Conditions
- Handle All Possibilities?
  - Beta Doesn't Handle Misbehaved Clients
    Forget to send oo or send incomplete packets
- Code must Handle all Possibilities
- Each Module – Method Verify Integrity
  - JUnit Testing
  - Examine & Optmize Code during JavaDoc
  - Thead.UncaughtExceptionHandler

Spaghetti and Meatball Code

- Spaghetti - Complex Tangled Code
  - Some Attempted BJ Dealer, GUI, Player were same
  - Worse Dealer, Player and GUI Inter-tangled
- Meatball – Toss in Objects Here and There
- Problems
  - Difficult to Update, Modify and Maintain
  - House are Build Modular Standards, Why?
- Solutions…
  - (BJPlayer,Dealer,Shoe Standalone Models), GUI Separate
  - Design Patterns MVC (Model View Controller)

MVC Design Pattern Approach

- Model Jstudent0.chatGM
  - Make it BulletProof, ChatServer, ClientInfo, ChatPacket
- Viewer
  - JSP Pages
  - JFrame GUI Controller Display
  - Pass chatViewer Object to ChatServer
- Controller
  - Start, Stop, Thread
  - Start Chat Server
- MVC harder to think through,
  than tossing together

ChatViewer ChatController

- Netbeans
  - New Java File → JFrame
- Source and Design Mode
- Design Mode Drag and Drop
  - Right Click to Set Properties
  - Add ActionListener by Clicking

GUI – Java Web Start App

- Status
  - JLabel w IP Servers Address
  - JTextArea or JEditorPane w gamePlayer Info
  - JTextArea w Last ~10 Messages
  - ChatDealer, ChatServer Thread Status
- Control
  - Update DB (# messages sent/received)
  - ChatDealer: stop, start, status and restart
  - Update Status Screen
- Save DB Tasks to Integration Phase

Java Web Start

- JFrame Executed From Web Browser
  - Has More Permission That JApplet
- Creates a JNLP File Includes Jar
- Enable Netbeans for Java Web Start
  - Chat Server
- Lots of Web Links For Java Web Start

JNLP - Java Web Start

- Test Locally First
  - Right^Click Properties (Local Execution)
- Project Folder/dist (Distribution)
  - jar File Java Archive
  - launch.html (Edit this for Remote Execution)
  - launch.jnlp
    - XML File Defines Launch Environment
    - Check File May Need lib Folder
  - Readme File Additional Help
  - Codebase URL site Root for the Java Code
    - Should be JSP Home Page

launch.jnlp

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<jnlp codebase="file:/C:/Saddleback/cs4b/Jstudent0/chat/dist/" href="launch.jnlp" spec="1.0+">
    <information>
        <title>Basic Application Example</title>
        <vendor>Sun Microsystems Inc.</vendor>
        <homepage href="http://appframework.dev.java.net"/>
        <description>A simple java desktop application based on Swing Application Framework</description>
        <description kind="short">Basic Application Example</description>


    </information>

    <resources>
<j2se version="1.5+"/>
<jar eager="true" href="chat.jar" main="true"/>


    <jar href="lib/appframework-1.0.3.jar"/>
<jar href="lib/swing-worker-1.1.jar"/>
</resources>
    <application-desc main-class="chat.ChatController">

    </application-desc>
</jnlp>
```

Applets

- User Java Web Start w JFrame
- extend JApplet

```
<applet
CODE="application.casino.videopoker.VideoPokerApplet"
  ARCHIVE = "URL to Jar File"
  WIDTH = "800" HEIGHT = "600" >
<param NAME = "draggable" VALUE = "true">
</applet>
```


Controller/Viewer – ChatController.java

- Java Web Start Program
- Controls ChatServer
  - List Threads, Start
- Database Interface
  - Update Button (Send Message to Dealer)
  - Display Buttons
  - Output sent to JEditorPane
  - Could also use JTable

Integration

Finalize Database Tasks
Web-Page Startup Interface
JSP Files Access ChatDatabaseBean
Put Complexity at JavaBean
Testing Server → CIM Deployment

JSP Integration Pages

- chatIndex.jsp
  - Login File, Check if Chat Server is Up
  - Calls chatValidate.jsp
- chatValidate.jspValidates Login Startup
  - Generates cim:8080/chat/chat.jnlp (w arguments PlayerID & IP)
    - Should Remove File After chatController Starts Up
    - File needs to be in a R/W Location for Tomcat
  - Failures Calls chatIndex.jsp
  - Should Maintain Session Count w Cookie And Validate request
- chatUpdate.jsp
  - DB Integrator Calls ChatDatabaseBean

Database Integration

1) Develop SQL – Database Schema
2) PHPMyAdmin Create Queries
3) Set up NetBean/Java Derby for Testing
4) JavaBean automate Query Task
   - Constructor Sets Up DB, Mnemonic Methods
   - Robust and BulletProof
5) **JSP File to Bean DB (chatUpdate.jsp)**
6) **Java Program Call JSP (JEditorPane)**
7) Prohibit SQL Injection – request Authenticate

DB Task #5 - Key Methods

- Jsp ChatDatabaseBean.chatDatabaseParser(HttpServletRequest req);

1) countActiveChatDealers
2) insertChatDealer&ip=ipaddress // set in JSP chatValidate.jsp
3) closeActiveChatDealer // done when chatController closes

4) insertChatPlayer&playerID=id&ip= //Managed by chatServer Thread
5) updateChatPlayer&playerID=id,received,sent,badPackets
6) closeChatPlayer&playerID=id, received, sent, badPackets

7) infoActiveChatPlayers // built in chatController Database Tab
8) infoUnknownChatPlayers
9) getchatDealerIP
10) infoChatPlayers (chat players with known names)
11) infoChatDealers
12) infoAllChatPlayers (includes Unknown Players)

chatUpdate.jsp

```
<body>
  <h1>Hello World! need to add
   fingerPrint or something later on</h1>

  <jsp:useBean id="cdb" scope="page"
  class="jstudent0.ChatDatabaseBean" />

  <%= cdb.chatDatabaseParser( request ) %>
Need to use urldecode inside of jsp file
</body>
```


Complexity is in JavaBean

- JSP Hard To Work With
- Put Complex Methods in JavaBean
  - Test Using JUnit Testing or Main Method
- Just Call Key Method
- Could Write Query Right at Java Code
  Pass Query instead of Calling Method
- Complex JSP Forms Move Towards JSF

Creating Front End Interface

- Web Page Advance Interface
  - Short Cut Tag + HTML IDE
    <%= "hello" %> === <% out.print("hello"); %>
  - Dreamweaver Supports JSP & Testing Server
- JSF
  - Java Server Faces
  - JSP w Forms & Templates
- Visual JSF inside of NetBeans
  - Very Advance GUI

ChatServer.java → chatUpdate.jsp Java.net.URL

```
URL url = new URL( "http://saddleback.edu" );
URLConnection  c  =url.openConnection();
InputStream in = c.getInputStream();

Or
 BufferedReader in = new BufferedReader(new
   InputStreamReader( url.openStream() ));
while( in.readLine()) != null ) …;
```


Java SE Full HTML Swing Support

- Listeners, HTML/XML Parsers, D
- From Java Web Start – JFrame
  - JEditorPane.setPage(String url); setText
    - Loads and Format the HTML Page
    - Not as Stable as java.net.URL
  - But Web Start You Do need to Get Resources. Gave up
- Javax.swing.event.HyperLinkListener
- From Applet…
  - getAppletContext().showDocument(URL)

Data Formats

- XML
  - SAX Content Handler
  - Class Tree Mimic Data Structure
- HTML
  - HTML Parsers
  - Java.swing.text.html.HTMLEditorKit
- Text Data (String like chatPacket)
- Mixture of Text (info) and HTML (Viewing)

Chat/chat.jnlp

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<jnlp codebase="http://cim.saddleback.edu:8080/jstudent0" href="/chat/chat.jnlp"
     spec="1.0+">
   <information>
      <title>Chat Server Launches Dealer</title>
      <vendor>Sun Microsystems Inc.</vendor>
      <homepage href="http://appframework.dev.java.net"/>
      <description>Launches the chat controller interface also passes the playerID launching
    app and remote IP</description>
      <description kind="short">Chat Interface Controller</description>
   </information>

   <resources>
<j2se version="1.5+"/>
<jar eager="true" href="chat.jar" main="true"/>
   <jar href="lib/appframework-1.0.3.jar"/>
<jar href="lib/swing-worker-1.1.jar"/>
</resources>
   <application-desc main-class="chat.ChatController">
      <argument> playerID </argument><argument> Value of PlayerID</argument>
      <argument>IP </argument><argument> Value of IP </argument>
   </application-desc>
</jnlp>
```


To Do List

- Make Initial Packet Validate Packet
  - Make Sure playID exists
  - Up to client to validate chatPlayers
- Passing variable to JNLP How?
- SQL Injection
  - Add Session Variable for Java Program and updateChat.jsp
  - Add Authentication to request Object getUserPrincipal, getrRemoteUser();
- Add Cookie Front Page, auto Login

MVC – Generic Design

Shoe, Dealer, BJPlayer – Models
BJPlayer.play() Thread or Socket to GUI
ChatServer/ChatDealer – Multiple Users
CIM Deployment
Steps on Creating a chatPlayer

BJPlayer (Model) → GUI

- BJPlayer.play( ) Socket to GUI
  - Different Systems (GUI is anywhere/anything)
  - Dealer Call Method Play on BJPlayer,
    - BUPlayer writes CardPacket to GUI Socket
    - Sleep,wait for Socket Response (HIT/Stand/Double)
- BJPlayer and GUI Threads on Same System
  - Dealer calls BJPlayer.play has GUI Object Pointer
    - GUI Display Cards, Enable Hit/Stand/DoubleDown
    - BJPlayer Thread.wait, may want a 30 second wakeup call
  - GUI End User Clicks Hit/Stand/DoubleDown
    - Disable Buttons, Set Some Variable, Thread.notifyAll

CIM Deployment

- Goal Minimize Changes From Derby to CIM
- Avoid Recompiling Attempt Switches In JSP
- ChatDatabaseBean.java
  - boolean onCIM = true; //when on CIM
  - Forces a Recompilation, null Constructor sets up DB
- Two Java Programs Call JSP File, Update Location
  - ChatServer.java method updateDB
  - ChatController.java method setPage ($codebase)
- Spaghetti Code May Not Be Deployable

Creating A Chat Player

- As Easy as 1, 2, 3
- 1) Create and Tests a Local Player
- 2) Add info on your Chat Player to DB:Casino
- 3) Test using Web ChatController Interface
  - http://cim.saddleback.edu:8080/chat
  - Redirect to http://cim:8080/jstudent0/chatIndex.jsp

1) Create A ChatPlayer

- Get Chat Code on S:\cs4b\chat
- Debugging/Developing Under ChatServer Beta
- Make Sure you Send all info using ChatPacket
  - Create a ChatPacket, use setMessage to Set Message
  - Send Data to Dealer using ChatPacket.toString()
  - Read Data into ChatPacket Object using fromString()
- On Local System you can Use Several IPs
  - Localhost, 0.0.0.0 (NetBeans), 127.0.0.1
  - Or your Local Area Network IP

2) Add Entry to Database

- Add entry to DB:Casino Table: casinoGame
- Record Must have your ChatPlayer CIM URL
- Make Sure gameID corresponds to ChatClient
- chatIndex.jsp Functionality
  - If no Active Dealer, Option to Start Server/Dealer
    - Validates User Startup ChatController Interface
  - Once Dealer is active Page Lists Available Chat Players
    - Note List is Generated Using Table:casinoGame
- On Validated User, Calls your ChatPlayer URL
  - Argument playerID=ValidatedID&IP=WebIPofServer&username=email Address
  - You may need to do a URLDecode on the Arguments

3) Test With Chat GM

- After Uploading your Java Web Start or Applet
- Use http://cim.saddleback.edu:8080/chat or
- http://cim.saddleback.edu:8080/jstudent0/chatIndex.jsp
- Start up Server, Then Test your Client
  - Note IP Address Stored in Database is the Web IP
  - You can Connect using localhost to a local chat Server
  - You May Need to Delete ChatServer That Were not Closed Properly
- To Connect to Web IP
  - You Must Configure Router w Port Forwarding
  - Search on directPlay or WoW For Port Forwarding
  - Saddleback SME Doesn't Have Port Forwarding
  - Note gameTable has a IPLocal Field – ChatServer Doesn't Update it

Testing Users

- Can Always Add Your Own
  - Make Sure sha1 Encryption on Password Field

Username (email)	Password
mmouse0@disney.com	two
gmouse@disney.com	two
goofy@disney.com	three
jloser@saddleback.edu	three