Views

1

index.jsp

Login.jsp POST

**Controller** -

viewticket.jsp GET & POST

Error.jsp (

notfound.jsp/ GET

loginFail.jsp POST

ticketConfirmation.jsp POST

workerTicketList.jsp POST-after logging in GET-if returning to list from ticket

adminTicketList.jsp “ “

Workerticketlist

<><a href=”conteroller?ticketID=2”>Printer on Fire</a>

${ticket.id} ${ticket.title}

ticket is an object and id is the attribute of the object

CAN ONLY DO POST WITH A BUTTON AND A FORM WITH A METHOD OF POST

link above will do a “doGet”

Login takes you to the ticketList (worker or admin version)

TIKET VIEW

Title

Ticket Number

bullet List for notations

Add Annotation Field

BOX

submit annotation button >>> keeps you on the ticket view >>>POST Command

finish button >>> take you back to list >>> GET

return to list >>> take you back to list >>> GET

annotations needs the ticket number to submit to the right record

TICKET LIST VIEW

needs ID to pull tickets assigned to user

employeeID - token - stored in session

DTO

sometimes have an add

TICKET.java

ticketID

ticketTitle

ticketEntered

ticketDescription

ticketAnnotations - arrayList<Annotation>

ticketStatus string - open finished closed

ticketAssignment - assigned unassigned

METHODS

constructor()

getterMethods for all above attributes

CONSTRUCTOR

ticket (int,Date,String,String,ArrayList<Annotation>)

JUNIT TEST

NO ANNOTATIONS

WITH ANNOTATIONS

ArrayList<Annotation>anList = newArrayList<Annotation>();

anList.add(new Annotation(?,”Fred”, “i’m on the case”,22));

anList.add(new Annotation(?,”Fred”, “i’m on the case”,22));

Ticket instance = new Ticket(22, ?, “Printer on fire”, “Smoke, FLames, cough cough”, anList);

assertEquals(“id”, 22, instance.getID());

assertEquals(“title”, “Printer on FIre”, instance.getTItle());

assertEquals(“date”, 22, instance.getID());

GregorianCalendar ticketCal =

new GregorianCAlendat (instanceDatePosted());

assertEquals(“Year Posted”, 2013, ticketCal.get(Calendar.Year));

assertEquals(“month Posted”, XXX, ticketCal.get(Calendar.Month));

assertEquals(“day Posted”, XXX, ticketCal.get(Calendar.Day));

assertEquals(“Number of annotation”, 2, instance.getTicketAnnotations.size());

assertEquals(“description”, 22, instance.getID());

Annotation.java CLASS

annoDate

annoAuthor (authorName ) (UserID that wrote)

ticketID (Ticket Associated with)

annoText (Notation)

JUNIT

WorkTicket

DTO

autoGen Getters

remove at returnds

Source - Generate constructor with all fiels

\*\*\*\*GENERATE JAVADOCS - JAVA/BIN/JAVADOC - settings to \*\*\*\*

dto

viewTicket,jsp > ticket.jsp

list<ticket> workerTicketList.jsp

Ticket DAO

handle DB access for the work ticket application

PreparedStatement getTixForWorkerStmt;

PreparedStatement getAllTixStmt;

PreparedStatement getTicketByIDStmt;

PreparedStatement getTicketAnnotationsForTicktStmt;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DONT NEED AN ARRAYLIST IN DAO BC IN DB\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Connect to DB and create PreparedStatements

Public TicketDAO(){

//TODO Constructor

}

//returns the list of tickets assigned to worker

@param userID is of the worker to whom the tickets are assigned

public ArrayList<WorkerTickets>getTicketsAssignedToWorker(int userID){

ArrayList<WorkerTickets> ticketList = new ArrayList<WorkerTickets>();

//TODO retrieve tickets from DB, add them to the list

// where clause

return ticketList;

}

/returns the list of all tickets

@param

public ArrayList<WorkerTickets>getAllTickets(){

ArrayList<WorkerTickets> allTicketList = new ArrayList<WorkerTickets>();

//TODO retrieve tickets from DB, add them to the list

//no where clause

return ticketList;

}

ALL ASSIGNED TICKETS - Attribute

ALL UNASSIGNED TICKETS

ALL OPEN TICKETS

ALL CLOSED TICKETS

LOOK AT JUST ONE PULL FOR ALL TICKETS

/return the single ticket with id and all associated annotation

/@param ticketID DB id of the ticket

public WorkTicket getSingleTicket (int ticketID){

return null;

}

METHOD

Change Status of Ticket

//Changes the status of a ticket with ticketID to new Status

public void updateTicketStatus (in TIcketID, String newStatus){

//

}

assignTicket

//Change status of ticket with ticketID

//to Assign ticket to assigned to worker user ID

public void assignTicket (in TIcketID, int UserID){

//

}

//adds new ticket to the DB the ticket will be unassigned and have no annotation

public void createTicket(String title, String description){

}

//adds an annotation to a WorkTicket

public void addAnnotation(int TicketID, String annoText, int UserID){

}

// Check to see if the user has access and log them in

//returns true if username and password match an entry in userTable

public boolean isAuthenticatedUser(string login,string password){

return false;

}

public int getUserID (string login){

}

public

//CHecks user supplied login credits against DB

@param login login typed in by user

@param password password typed inby user

return user object represnting this user if login is sucessful otherwise return null

public User authenticatedUser (String Login, String Password){

public string getUserRole

}

CREATE User CLASS

//represents a user of the workTicket system

----- update passwords

------ just hand code people into the DB

int USerID

String userName

String role

\*\*\*\*\*3 TABLES - TICKETS ANNOTATIONS USERS\*\*\*\*\*\*\*\*

CONTROLLER ----only two methods

DOGET  
DOPOST

give all buttons the name command

----------------------------------

Make sure one loop works and then divy up all the SHIT!!!!!!!!!!

--------------------------------------------

MIST 7570 Code Commentary

Work Ticket Team Project

Team Joey: Hugh, Jason, Craig

Servlet Workflow:

--METHOD is either GET or POST. Use GET to view resources, POST to send data that

will change the system state (e.g. by adding a new ticket or assigning a ticket).

--View names (JSP page names) are conventionally in lower camel case.

--Use case 1: we would not \_send\_ the error message as a request parameter. Any

error message will be a DTO sent by the controller back to the JSP page.

--Use case 3: "View open tickets" could be a value of the parameter named "command".

Otherwise this one looks good. Remember to use the auth tokens in the session

when the controller processes this one.

--Use case 5: "View Open Tickets" would be a value for request parameter "command."

--Use case 7: You don't need the time that a ticket is submitted as a request

parameter. Your DAO can find the current time when it stores the ticket in the DB.

Schema:

--User: needs the password or a hashed version of the password (for logins)

--Request: What is the difference between a request and a ticket? You may be able

to combine these two tables.

--Request: "Office notes" will comprise multiple annotations by different workers

at different times. Suggest a new Annotation table linked to Ticket/Request via

foreign key.

JavaDocs:

-- please provide Javadocs for your classes