Simple Blood Bank

Web Project

Assignment 02

CST8288 2021 Winter

Table of Contents

Important Notes

SQL

1) Optionally you can run the SQL script attached to the Assignment in your MySQL workbench.
    a. This is also done automatically by your skeleton code.
2) You must use MySQL server 8+.

Skeleton

1) Read all the comments in the skeleton, I have explained many things in the comments.
2) Do not use any methods that are ~~@Deprecated~~.
3) Project is made for NetBeans 12 however, if you desire you can use other IDE's, but you must handle the conversion yourself.

Output

Simply run your project and you can see all example tables. The pages that say 404 are to be completed by you. The output of the "Donate Blood Form" page and other are given as a video on BS assignment page. You can use the HTML code and the CSS provided freely in your code.

Bright Space

Keep an eye out for updates that will be posted on bright space assignment page and announcements. I will always update this document again, but you do not need to reprint everything. I will mention the pages that I have updated. You simply update those pages or simply write corrections in your prints.

Due Date Sunday April 18th Midnight

Sunday April 18th Midnight. There will **not** be any **extension** as it is the **end** of the **term**.

Penalty

-10% penalty for every day late, zero on the 5th day.

Do not Touch (unless needed for common issues)

1) Do not touch **persistence.xml** unless you know you can fix it yourself. However, be aware of its content.
2) Do not touch **context.xml** unless you know you can fix it yourself. However, be aware of its content. You can change the URL of your webpage through **context.xml**.
3) **GenericDAL** and **GenericLogic** can be expanded for more functionality but do not reduce or change the existing code.
4) Do not recreate your Entity classes. Use the ones provided in skeleton.

Attachments

1) SQL Script for DB just in case the script built in the project not working.
2) Download the skeleton code, unzip, and then open it using NetBeans 12 LTS.
3) Class and Sequence diagrams.
4) ERR diagram for DB.

Submission

Submit one zip file of your project.

[firstName]-[lastName]-[labSection#].zip

ex: shawn-emami-11.zip

There is no demo for this assignment. Just submit to Bright Space.

Export Instruction Slides

In NetBeans go to File/Export Project/To Zip. Make sure you have the correct project selected.

Common Issues

1) Make sure you checked you have JDK 11.
2) You need MySQL server 8 to run the SQL script correctly, it will not run properly in MySQL 5.6 version. 5.7 sometimes works. You can use mariadb in xampp as well, but it is up to you to set it up. Using MySQL 8 is much easier.
3) If you start tomcat in NetBeans and the password screen ask for username and password with XDB message, it means you have a conflict with your Oracle services. Search for Services in start menu and run it. Scroll down till you find oracle services. There should be 2 of them running, can be more. For each running oracle service, right click and go to properties. Then from the drop-down menu and choose disabled, then click stop. When done click apply and finish. Go back to NetBeans and start tomcat server again. Now instead of XDB it should say Tomcat Manager.
4) Make sure you have your output tab open to see the logs. If you do not see it click on window in toolbar and choose output. When you run your application for the first time it should open two new windows in output called tomcat and tomcat log. If they are not there click on window again and choose services. In the opened tab expand servers and right click your tomcat server. Choose view server log and view server output or just restart, they should both popup.
5) The content of output is in log form. Meaning they record everything that has happened. So, to find your error check the timestamp as you might have old errors that are irrelevant.
6) If in your log you are getting an error with access denied, it means you do not have the user: cst8288 and pass: 8288 in DB. open workbench, double click connection, choose file and then run SQL script attached. Navigate to the location which you have downloaded the newest attached SQL script and run it again.
7) If you are getting Database not found it means you did not load the SQL script attached to the assignment. Open workbench, double click connection, choose file, and then run SQL script attached. Navigate to the location which you downloaded the newest SQL script attached and run it.
8) If you are getting an error saying "zeroDateTimeBehavior must be one of the", navigate to "web pages/meta-inf/context.xml" in your NetBeans. Look for zeroDateTimeBehavior=convertToNull and change it to zeroDateTimeBehavior=CONVERT_TO_NULL.
9) If you are getting an error regarding your time zone especially if your base windows language is not English, navigate to "web pages/meta-inf/context.xml".
   Look for **?zeroDateTimeBehavior=convertToNull** and change it to:
   ?zeroDateTimeBehavior=convertToNull&amp;useUnicode=true&amp;useJDBCCompliantTimezoneShift=true&amp; useLegacyDatetimeCode=false&amp; serverTimezone=UTC
   all in one line.
10) If you try to run your code and nothing happens with no errors, try changing your default browser from NetBeans. Go to tools/options/general then change web browser to chrome or Firefox. If you have already opened your project also right click on project and go to properties/build/run and change browser to chrome, Firefox, edge, opera, etc.
11) If you are on mac and you run windows on parallel; when you open a project it might suddenly disappear from project list with no errors or a NullPointerException. To fix this move your project to drive c:\ and try opening it again. If it opens but you cannot save you will need to change permissions on your project directory for write permission. If you cannot do it come and see me.
12) I will add more if anything new comes up.

Requirements

1) Finish the DAO design pattern. Review the Class Diagrams Posted. You can add extra methods if needed. Do not reduce anything.
2) Create a Junit 5 for all logic classes (except for account). It must be in the test folder.
3) Complete the LogicFactory using the sequence diagram provided.
4) Make Servlets for view and create of each table/entity.
5) Follow proper coding and naming conventions.
6) Provided proper documentation on methods that need it, use your judgment. You do not need to use any fancy JavaDoc keywords. You simply need to provide meaningful definition and purpose.
7) Create a Sequence diagrams for DonateBloodFrom::doPost. The main in DonateBloodFrom which creates all he entities and add them to DB.

Bonuses

1) Add update
2) Add delete
3) Add functional search for all tables, search should be able to at least search two columns.=
4) Add user verification for login. to do so you need:
   a) Sessions, cookies, or URL editing. Session is the best way.
   b) @WebFilter which will allow you to filter incoming URL requests. It also allows redirecting to other page, convenient for login page.
5) Beautify the web pages as you see fit. Creativity effects bonus award.
6) Use JSP, **do not** use scriptlets (<% %>). Use JSP in combination of Servlet using Expression Language.

Instructions

Styling and Conventions

Follow proper coding and naming conventions.

- Everything must be properly indented.
- All class names must start with capital letter.
- Package names must all be small letters.
- All instance variables must be private.
- Static variables should be accessed from Class not instance.
- Variable names must make sense and be meaningful.
- If you are unsure about anything ask, do not assume anything.

Documentation

Provided proper documentation on methods that need it, use your own judgment. You do not need to use any fancy JavaDoc keywords, like @link, @see, <Code>, and <P>. You simply need to provide meaningful definition and purpose.

UML Diagrams

Create a Sequence diagrams for LoadDataView::doGet.

Finish DAO design pattern

Review the Class Diagrams Posted. Class diagrams for all packages are attached. You can add extra methods if needed. Take advantage of Account code that is already given to you.

1) Create the rest of logic classes. Take advantage of Class diagrams attached.
    a) Pay close attention to accessors on Class diagrams for logic.
    b) Logic should do proper error checking for input data, such as string is not too large, what can and cannot be null. This information can be found in ERR diagram.
    c) Use the current Date if the conversion of your String to Data has failed. Use the conversion methods provided in the super class of your logic.
    d) Inside of createEntity do not create logics, dals, and/or entities of other types.
        i. **Dependencies should be set outside of the createEntity method.**
    e) Inside of updateEntity which is part of update bonus, you can create other logics and entities.
2) Create the rest of dal classes using the GenericDAL provided. Take advantage of Class diagrams attached. When using findResult and/or findResults you need to use the name of @NamesQuery of appropriate Entity. Look at Account and AccountDAL for examples.
3) You are required to provide get for every column of every table.

Servlets

Servlets to view each Table

Every table should be displayed in a web page. You can put the code for it in one servlet which updates the table based on the URL or you can create a different servlet for each table. A servlet for each table is a bit more coding but the easiest way.

1) Look at the example servlets provided for Account. JPS is bonus.
2) When creating the html output use getColumnNames, getColumnCodes and extractDataAsList. do not hard code or type the data one at a time.

Servlets to create Entities

Create input servlets. **Remember auto generated IDs are filled by the DB**, no need to ask for them.

1) Use CreateAccount as an example.
2) Remember when creating Entities which have dependencies, you should set those before adding it to DB.

Servlets for DonateBloodForm

Create a new servlet called DonateBloodForm. This servlet will be a location for Entering all your data at once, except for blood bank. By the time you click add, there should be a new person, blood record, and donation record in your db.

doPost

In this method you will combine all the codes you have written in your previous create servlets to have one unified method which creates a person, blood record, and donation record.

doGet

simply call the processRequest (if you have the method) then create your html page and send it as response. For each input use the proper names (static final fields) from each logic. This way you can use your createEntities without issue. If you are going to use the same input for multiple logics make sure they have the same name. you can change the names given to you in the UML diagrams. For example, created in donation record and donated blood is the same. So, they can use the same name.

processRequest

All the html examples you need will be in the AccountTableView and SampleForm.

Junit

Create Junit for Each logic. All Methods with their edge, normal, and invalid states must be tested. You can use the example provided for AccountTest as starting point.

Also, the starting point of BloodDonation is also provided. It shows how to add dependencies specifically in JUnit.

Approach

Break down your work in smaller chunks and assign them to each member.

Set deadlines for yourselves so you all can be on the same page.

If you need help regarding your group coordination come and see me in one of the labs.

DO NOT LEAVE THESE FOR THE LAST WEEK, YOU WILL NOT FINISH.

Suggested Weekly Schedule

Week 10 – Read the Assignment and Class Diagrams

Make sure to read everything. Assign group works and start on DAL.

Week 11 – Complete Logic and DAL

All you need for this is inside of skeleton already provided to you. Look at Account classes.

Week 12 – Complete Junit and View Servlets

Use AccountTest as example to complete the junit for all logics. Create the remaining view servlets.

Week 13 – Complete Create Servlets

Finish the servlets that display the tables. Finish the simple ones before you try the bonus.

Week 14 – Complete LoadDataView and Diagram

It should be very small amount of work, good time to catch up on any work you are behind.

Finish the sequence diagram. Double check your work. Try bonuses or just relax.
Do not forget to submit.

Work Break Down Suggestion

There are 4 tables in the project. Meaning each student should be able to do at least one of dal, logic, view, create, and junit.

The final diagram and them form should be group effort.

Hibernate Notes

Adding, updating, and/or removing

Hibernate is designed to be OOP, meaning when you are trying to add an entity to a table you need to make sure your object does not have null dependency/s. For example, when creating a new Entity to be added to DB, its dependences must also be set on the Entity object. These are the steps to have in mind, look at doPost method in createAccount.java for an example:

1) Create all needed Logics. Primary logic (logic of entity to be added to DB) and logics for dependency entities.
2) Using methods from primary logic to check for duplicates at this point.
    a) You can use **request.getParameter(key:String):String** to get a specific key.
3) If duplicate does exist (might not be needed for every table)
    a) Complete errorMessage and continue.
4) If duplicate does not exist
    a) Call createEntity on primary logic and pass request.getParameterMap():Map<String, String[]> to it. The return is an entity.
        i) If your entity has dependencies call on relevant logics and use their get methods to find dependent entities.
        ii) Set the entities returned from getters to the entity created using createEntity.
            (1) Do not create entities of other types in createEntity.
    b) Finally call add/update on primary logic and pass the entity created using createEntity.

URL Patterns

These patterns already match the "href" in **index.html**. If you do not follow these, you must fix **index.html** and other references.

Viewing Tables

- /AccountTable
- /PersonTable
- /DonationRecordTable
- /BloodDonationTable
- /BloodBankTable

Creating Entities

- /CreateAccount
- /CreatePerson
- /CreateDonationRecord
- /CreateBloodDonation
- /CreateBloodBank

Load Data Page

- /DonateBloodFrom

DB Details

- All String are by default minimum 1 character long if not null.
- Lengths can be found on the EER diagram.
- Columns with a rhombus can be null.

Account

- id is auto generated.
- nickname can be null.
- name is unique.
- username is unique.

Person

- id is auto generated.
- Nothing can be null.

BloodBank

- id is auto generated.
- Owner can be null.
- Name is unique.

BloodDonation

- id is auto generated.
- Bank_id can be null.

DonationRecord

- id is auto generated.
- Person_id can be null.
- Donation_id can be null.