ENSF 409 Winter 2018 - Project Help Document

This document will outline some of the more challenging parts of the project and will provide tips on their implementation.

Sending an Email in Java

Eclipse Setup:

1) Install & unzip JavaEE SDK which is available on the Oracle website
2) Right-Click Project -> Properties -> Java Build Path -> Add External JAR's
3) Add javaee.jar from /glassfish/lib directory
4) Add javax.mail.jar from /glassfish/modules directory

Sending the Email:

The easiest way to send an email is through a pre-existing SMTP Server (provided by Gmail, Yahoo Mail, Outlook, etc.). Below are steps to send email using a Gmail account. If you want to use another SMTP Server, these settings may change.

1) Create a `Properties` object with the following information:

```
Properties properties = new Properties();
properties.put("mail.smtp.starttls.enable", "true"); // Using TLS
properties.put("mail.smtp.auth", "true");            // Authenticate
properties.put("mail.smtp.host", "smtp.gmail.com");  // Using Gmail Account
properties.put("mail.smtp.port", "587");             // TLS uses port 587
```

2) Create a new email `Session` using your Gmail account Email Address and Password:

```
Session session = Session.getInstance(properties,
      new javax.mail.Authenticator(){
         protected PasswordAuthentication getPasswordAuthentication() {
             return new PasswordAuthentication(YOUR_ADDRESS, YOUR_PASSWORD);
         }
      });
```

3) Create and send the Email:

```
try {
      Message message = new MimeMessage(session);
      message.setFrom(new InternetAddress(YOUR_ADDRESS));
      message.addRecipient(Message.RecipientType.TO, RECIPIENT_ADDRESS);
      message.setSubject(("Your Message Subject");
      message.setText("Your Message Content");

      Transport.send(message); // Send the Email Message
} catch (MessagingException e) {
      e.printStackTrace();
}
```

Note: If you want to send the email to multiple people, just call `addRecipient()` for each additional person with a recipient type of:

`Message.RecipientType.`***`CC`*** **or** `Message.RecipientType.`***`BCC`***

(where ***BCC*** hides the recipient list from each recipient, and ***CC*** does not)

Sending Files Over a Socket

Note: You should add lots of error checking in this section of the program! For example, what if the file length is too large for an integer to store? What if ***PATH*** points to a directory instead of a file?

1) Get the `File` you want to send, either using (a.) the path or (b.) Java Swing components:

a.
```
File selectedFile = new File(PATH); // Add your file path here
```

b.
```
JFileChooser fileBrowser = new JFileChooser();
    if(fileBrowser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION)
        File selectedFile = fileBrowser.getSelectedFile();
```

2) Get the file content. To ensure PDF and image files are not corrupted, **both** a `FileInputStream` and `BufferedInputStream` should be used to read the file into an array of `Bytes`.

```
long length = selectedFile.length();
byte[] content = new byte[(int) length]; // Converting Long to Int
try {
      FileInputStream fis = new FileInputStream(selectedFile);
      BufferedInputStream bos = new BufferedInputStream(fis);
      bos.read(content, 0, (int)length);
} catch (FileNotFoundException e) {
      e.printStackTrace();
} catch(IOException e){
      e.printStackTrace();
}
```

3) Send the array of bytes over `Socket` via an `ObjectOutputStream`. (Assuming the socket is connected, and the Object Stream is already constructed)

```
// Sender (Client?)
try{
      objectOutputStream.writeObject(content);
      objectOutputStream.flush();
} catch(IOException e){
            e.printStackTrace();
}

// Receiver (Server?)
byte[] content = (byte[]) objectInputStream.readObject();
```

4) Save the received file the same way it was read:

```
File newFile = new File(STORAGE_PATH + FILE_NAME + FILE_EXTENSION);
try{
      if(! newFile.exists())
            newFile.createNewFile();
      FileOutputStream writer = new FileOutputStream(newFile);
      BufferedOutputStream bos = new BufferedOutputStream(writer);
      bos.write(content);
      bos.close();
} catch(IOException e){
      e.printStackTrace();
}
```

Note: You will need to send the file extension (.txt, .pdf, etc.) separately over socket to save the files format. Otherwise it will always be saved as a text file.

Setting Up the Database

Help installing MySQL and connecting to a database can be found in "***Some Help with MySQL***" posted on the D2L course page.

Example Database Structure:

User Table
id: int (8)
password: varchar (20)
email: varchar (50)
firstname: varchar (30)
lastname: varchar (30)
**type: char (1)

Assignment Table
id: int (8)
course_id: int (8)
title: varchar (50)
path: varchar (100)
active: bit (1)
due_date: char (16)

Grade Table
id: int (8)
assign_id: int (8)
student_id: int (8)
course_id: int (8)
*assignment_grade: int (3)

Course Table
id: int (8)
prof_id: int (8)
name: varchar (50)
active: bit (1)

StudentEnrollment Table
id: int (8)
student_id: int (8)
course_id: int (8)

Submission Table
id: int (8)
assign_id: int (8)
student_id: int (8)
path: varchar (100)
title: varchar (50)
*submission_grade: int (3)
comments: varchar (140)
timestamp: char (16)

Notes:

* In this model, every student submission can be graded (submission table), but only one may be chosen as the final assignment grade (in the grade table). These final assignment grades are used to calculate student averages and can be displayed on the course grade page.

** This attribute will identify if the user is a student or professor. A char is used instead of a boolean in case additional users are added in the future (staff, admin, TA, …)

Troubleshooting Tips:

Socket Communication: if your socket IO streams are getting blocked, make sure you are calling `flush()` each time you write an object to them. Also, make sure the input stream and output stream are declared in the opposite order in the client vs. the server code.

Receiving Objects Over Object Streams: When sending objects over object streams, it is important that each object implements `Serializable`, and has a unique `SerializiableUID` so the `ObjectInputStream` can deserialize the object.