Debugging pa1, HTTP

Last time: transactions

Using transactions in a database-backed program (pizza3, music3, etc. Also via JPA in pizza2/music2):

- Do DB actions, individually in DAO methods
- Commit or Rollback, in service layer

- Note we can rollback based on data we see, not just DB problems. For example, we figure out the user is not authorized to do something.
- Note that the database doesn't have to be relational to have transactions: mongodb supports transactions. See Transaction support in Mongodb 4.2+. It's more complicated to use, however.
- Cloud notes: AWS Aurora DB uses a special cloud-based storage engine under mysql or PostgresSql, i.e., it's a cloud-based relational DB with transactions (of course).
  - An AWS whitepaper emphasize the importance of short transactions or just leaving auto-commit on.

Eclipse Debugging Demo (on Windows 10)

Use pizza1 for debugging demo, since like current project.

Run SystemTest from its source file in Project Explorer: right-click SystemTest.java, select Run>Java application. This works for any program that can run without arguments.

If you need to specify arguments, set up a "Run Configuration" and use it to launch the app.

1. Break it by changing insertOrder to insertOrderx in PizzaOrderDAO

See red stars in eclipse, but suppose we're not using that...

Need to rebuild to get fresh jar: mvn clean package

Fails with compilation error: StudentService can't find insertOrder

Harder case: it compiles but doesn’t run…

Project Compiles but Unit test fails

2. Break it by putting sizeName = null at start of insertOrder

String sizeName = null; //order.getPizzaSize().getSizeName();

No red stars in eclipse, so this compiled

mvn clean package: see compilation, but JUnit test fails, so jar never built

java.sql.SQLException: no such pizza size available

    at cs636.pizza.dao.PizzaOrderDAO.insertOrder(PizzaOrderDAO.java:44) ←close to error

    at cs636.pizza.dao.PizzaOrderDAOTest.testMakeOrder(PizzaOrderDAOTest.java:47)

    ...

Run Junit tests in eclipse: right-click project, Run As> JUnit test

Failing Unit test: in eclipse

Try it in eclipse: right-click project, Run As> JUnit test

See nice display showing the one test that fails because of the defective insertOrder, and the backtrace as above

Moral: run the JUnit tests before building the jar. Maven does this with “mvn package”

In music: keep the JUnit tests current by uncommenting parts

Bug in service layer crashes DAO code

3. Break it by putting "order = null;" just before call to insertOrder, in StudentService.makeOrder.

Try JUnit tests, the badMakeOrder test succeeds, because it's testing something else wrong. It should be called badInsertOrder test, because it doesn't use makeOrder, only insertOrder.

But the DeleteToppingTest (of the service layer) fails because it expects a Service Exception but sees a NullPointerException at insertOrder line 42

line 42 String sizeName = order.getPizzaSize().getSizeName();

so order = null here, a clue, but let's build and run the jar and see how that goes.

mvn clean package won't work because of the JUnit test failure

mvn clean package -DskipTests skips JUnit tests

Run with "runOnH2 SystemTest", see backtrace and NullPointerException reported from line 42 in insertOrder in DAO, same as JUnit test failure

Note that the bug is in StudentService, not in the DAO, but our original clues come from the DAO.

Bug in service layer crashes DAO code

Run SystemTest with debugger in eclipse: right-click SystemTest.java in project explorer, select Debug>Java application

See crash report, same as running with runOnH2, except link to PizzaOrderDAO.java, site of crash (line of code) and its caller, etc.

**********so 5**********

Error in run of SystemTest:

java.lang.NullPointerException

Stack Trace: java.lang.NullPointerException

at cs636.pizza.dao.PizzaOrderDAO.insertOrder(PizzaOrderDAO.java:42) <-- click to see code in editor

at cs636.pizza.service.StudentService.makeOrder(StudentService.java:82)

at cs636.pizza.presentation.SystemTest.handleStudentOrder(SystemTest.java:129)

at cs636.pizza.presentation.SystemTest.run(SystemTest.java:98)

at cs636.pizza.presentation.SystemTest.main(SystemTest.java:64)

Bug in service layer crashes DAO code: using a breakpoint

Set breakpoint in PizzaOrderDAO: double-click along blue left-hand edge of editor window at desired line.

Run again with debugger: can use little bug icon at top of Debug window.

When breakpoint is reached, whole IDE window format changes over from “Java Perspective” to "Debug Perspective"

See before-crashed state in insertOrder, with back trace in window, editor on PizzaOrderDAO.java showing breakpoint in green highlight

Click levels in back trace, see editor windows on various files.

Click makeOrder level, see its code, obvious problem user = null, but we'll pretend we don't see it yet.

Set breakpoint in caller, makeOrder, to find the cause.

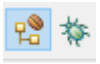
Run again with debugger, stops in makeOrder, use step-over button to advance line by line, seeing variable values change.

Finally see user become null, that's the bug.

Using the debugger

Can also step-into: rerun with breakpoint in SystemTest, step into makeOrder, etc.

Note debugger perspective, different from Java perspective, switch back and forth with perspective buttons:



You don't have to live with what is shown in the Debugger perspective:
Example: add Project Explorer view to Debugger display.

Sometimes you see undeserved red stars on some source files in Project Explorer. This can be fixed by "cleaning the project" by top-of-window Project> Clean. in one case I had to do Maven>Update Project

If you forget to clean out a crashed debugging session by using the red square button and double-X button, you may see another kind of crash on rerun that mentions H2. This is caused by H2's single-session rule: the crashed process still has a session with H2 until you kill it completely.

Web servers used in hw3

- Departmental web server: Apache on www.cs.umb.edu, at port 80
  - URL http://www.cs.umb.edu/~username/test.html for hw3
    - this file found by special ~ syntax handling supported by Apache
  - URL http://www.cs.umb.edu/cs636 class home page
  - doc root /data/htdocs, so class home at /data/htdocs/cs636 in the filesystem
    - The page that shows is /data/htdocs/cs636/index.html
    - Apache and other web servers convert a directory URL into request for its index.html or index.htm or index.php
- Single-app web server tomcat on pe07.cs.umb.edu, at port 9002 (needs tunnel or lynx on pe07)
  - URL http://pe07.cs.umb.edu:9002/welcome.html
  - doc root /home/eoneil/636/pizza3/src/main/webapp by standard Maven setup
  - So would expect file at /home/eoneil/636/pizza3/src/main/webapp/welcome.html
  - But with a webapp, not so simple…

The pizza3 webapp

In hw3, you explored the pizza3 webapp by firing requests at it, for example, HTTP GETs to

http://pe07.cs.umb.edu:9002/welcome.html showed home page

http://pe07.cs.umb.edu:9002/studentWelcome.html showed student welcome page

http://pe07.cs.umb.edu:9002/adminController/adminWelcome.html

...

Although these URLs end with .html, they do not access HTML files, but instead are interpreted by a Java servlet running on pe07, and the servlet uses **JSP (Java Server Pages)** files to generate the responses.

So for example, access to welcome.html ends up executing welcome.jsp. As we will see, a JSP file looks like HTML, with some additional elements.

The pizza3 webapp

We see that this app (pizza3) owns even the top-level URLs--it's not pizza3/welcome.html, but just welcome.html.

This is because this tomcat web server is serving only one app, not sharing the server with other apps as we would normally see with apache servers.

- It is using Spring Boot, which knows how to run tomcat as an **embedded server**, i.e., running *inside* the pizza3 Java app.

Tomcat can also run as a **shared server**, with multiple apps using URLs like pizza3/welcome.html and music3/index.html for two apps, pizza3 and music3.

A shared-server tomcat is running at port 8080 on pe07, with doc root at /var/cs636/tomcat-8.5/webapps, and subdirs for each student for experiments. Put test.html there and see it at http://pe07.cs.umb.edu:8080/username/test.html

HTTP: read Chap. 18 to pg. 555. Now look at slides for Chap. 18.