CSE3OAD/CSE4OAD Assignment 2

Due Date: 10 AM, Wednesday October 24th, 2018

Assessment: This assignment 2 is worth 25% of the final mark for CSE3OAD and CSE4OAD.

This is an individual assignment.

Copying, Plagiarism: Plagiarism is the submission of somebody else's work in a manner that gives the impression that the work is your own. The Department of Computer Science and Information Technology treats plagiarism very seriously. When it is detected, penalties are strictly imposed. Students are referred to the Department of Computer Science and Information Technology's Handbook and policy documents with regard to plagiarism.

No extensions will be given: Penalties are applied to late assignments (5% of your total assignment mark given is deducted per day, accepted up to 5 days after the due date only). If there are circumstances that prevent the assignment being submitted on time, an application for special consideration may be made. See the departmental Student Handbook for details. Note that delays caused by computer downtime cannot be accepted as a valid reason for a late submission without penalty. Students must plan their work to allow for both scheduled and unscheduled downtime. Assignments submitted more than 5 days late (i.e. after 10 AM, Wednesday October 31st, 2018) will receive the mark of 0.

Return of Assignments: Students are referred to the departmental Student Handbook for details.

Objectives: To design and implement a RESTful API that will provide an access endpoint to a data source controller similar to the one you implemented using JDBC in Assignment 1.

Introduction

The aims of the assignment are:

- To implement web services for making available resources regarding your fridge groceries and items. The groceries and items are those that you have worked with in Assignment 1.
    - Using Java Reflection and Servlet API
- To apply the relevant validation strategies to the grocery, using the Validation Framework we built Lab 07.

Files Provided:

The following are provided in the **`fridge`** directory (zip file)

- The servlet descriptor file **`web.xml`** (directory **`fridge/WEB-INF)`**
- The jar files for JSON conversion and JDBC (MySQL) driver (directory **`fridge/WEB-INF/lib`**)
- The following in the directory **`fridge/WEB-INF/classes`**
    - The Java files for the *Validation framework* (complete files):
        - **`Min.java`**, **`MinValidator.java`**
        - **`Max.java`**, **`MaValidator.java`**
        - **`NotNull.java`**, **`NotNullValidator.java`**
        - **`CharCount.java`**, **`CharCountValidator.java`**
        - **`Validator.java`**, **`ValidationException.java`**
    - The Data Source Controller, **`FridgeDSC.java`**
        - Complete (NOTE: *do not use your solution from Assignment 1*)
    - The Models (complete)
        - **`Grocery.java`** (excluding the validation annotations required for **Task 1**)
        - **`Item.java`**
    - The Controllers (incomplete)
        - **`GroceryController.java`**
        - **`ItemController.java`**
    - The Router Servlet (incomplete)
        - **`FridgeRouterServlet.java`**
    - A few custom ***`Exception`*** *sub-classes* to be used throughout this Assignment 2 (complete)
        - **`MissingArgumentException.java`**
        - **`ResourceNotFoundException.java`**
        - **`UpdateNotAllowedException.java`**
    - A separate unrelated Model for CSE4OAD students (Task 4)
        - **`UserProfile.java`**
- The SQL Script to create and populate your database
    - **`CreateDatabaseScript.sql`** (MySQL on your own computer)
    - **`CreateDatabaseScript-latcs7.sql`** (MySQL on **`latcs7.cs.latrobe.du.au`**)

Task 1

NOTE: you are strongly advised to complete this task before the end of Week 9

1. Study the *Validation Framework* provided (complete code provided) as well as the lecture and lab materials that covered Java Reflection, Annotations and the Validation Framework.
2. **TODO:** Clearly annotate the relevant fields of the **`Grocery.java`** model.
    - You may want to test if your validation on the model is working (see how we did that in the labs) – a **`static main`** has been provided in **`Grocery.java`** for you to add your testing code.

Task 2

NOTE: you are strongly advised to complete this task before the end of Week 10

The controllers are the bridge between the **`FridgeDSC`** class and the **`FridgeRouterServlet`** class.

- Study the complete **`FridgeDSC.java`** file provided, paying attention to its constructor (some changes has been made, which differs from Assignment 1)
    - NOTE: the constructor requires ***database host***, ***username*** and ***password*** as arguments
    - NOTE: **CSE4OAD** students, see **Task 4a.** before attempting Task 2.

For each controller (**`GroceryController.java`** and **`ItemController.java`**)

- **TODO:** Complete each of the method stubs provided. Each method stubs needs to make a call to a relevant **`FridgeDSC.java`** method. Identify which one and code it in the controllers.
    - You may want to test if the controllers are responding properly when calling each **`FridgeDSC`** method – a **`static main`** method has been provided in each class for you to add your testing code.

Task 3

Implement the servlet (**`FridgeRouterServlet`** class) - The purpose of this servlet class is to:

1. Identify which resource the URL is requesting, using *`HttpServletRequest`*
    - example, the resource from a browser call to http://localhost:8080/fridge/api/**grocery** will then be "**grocery**" (the same applies for a call to http://localhost:8080/fridge/api/**GroCerY**)
    - the HTTP method can also be found here (in this example, a browser call means using HTTP **GET** method)
2. Using the Servlet API,
    - retrieve your database *host*, *username* and *password* from the servlet descriptor file **`web.xml`**. (add your MySQL username & password where needed in file **`web.xml`**)
3. Using the Reflection API,
    a. Find if the resource *controller* exists - from the above example, resource "**grocery**" should have a matching *controller* class **`GroceryController`** *[hint: using `Class.forName(…)`]*
    b. Create an instance of such controller class
    NOTE:
        - HTTP **GET** (with **no** parameters identified) maps to *controller* method **`get()`**
        - HTTP **GET** (with parameter identified) maps to *controller* method **`get(id)`**
        - HTTP **POST** (with parameter identified) maps to *controller* method **`add(…)`**
        - HTTP **PUT** (with parameter identified) maps to *controller* method **`update(id)`**
        - HTTP **DELETE** (with parameter identified) maps to *controller* method **`delete(id)`**
    c. If we have an HTTP GET method, with no identifiable parameters (from the above example), find if **`GroceryController`** class has a no-argument method named **`get()`**
        - then, make a call to this method, and store the returned data
    d. Do the same (a, b, c above) if the resource identified is "**item**"
        - NOTE: clearly distinguish the difference between *controller* classes GroceryController and ItemController
4. Using the *Gson* package provided and *`HttpServletResponse`*
    a. Convert any returned data (from *controller* method call) to JSON data (using *Gson*)
    b. Convert any valid case messages (if any) to JSON data (using *Gson*)
    c. Send the any JSON data back to the browser
    d. Making sure any potential errors (throughout steps 1-4 described for this Task 3) are caught, identified and send back to the browser (using *GSON* to convert messages to JSON) and using appropriate ***HTTP Status Codes*** and adequate (descriptive) error messages.

You will use Postman to test your API (https://www.getpostman.com/) – More on Postman will be covered in your lab sessions.

Task 4 (only for CSE4OAD)

a. TODO: Update **`FridgeDSC`** to be a *Singleton*, that is, not more than 1 instance of **`FridgeDSC`** can ever be created. (you would want to do this part of task 4 before attempting coding the controllers in task 2, as the controller calls to the DSC will be different once it has been converted to a *Singleton*)

b. Study the **`UserProfile.java`** class provided.

   TODO: Analyse the **`UserProfile.java`** class provided and carefully decide which of the provided validation annotations should be used for each of the attributes

   - An attribute may have none, one or more annotations
   - You are to come up with the reasons for each attribute validation reason
   - For each annotation added to **`UserProfile.java`**, briefly comment (code comment) why you are using such validation annotation and their parameters (if any parameters used)
     - Look at attribute **`id`** as a given example for annotation and comments.
   - Create a new **`Email.java`** annotation (should your implementation have any parameters?)
   - Create its matching **`EmailValidator.java`** *`Validator`* sub-class (as we did in lab 7), using *Regular Expression* as checking measure for the validation rule.
   - Annotate the email attribute in the **`UserProfile.java`** class with the newly created Email Validation annotation.
   - Follow the directions (TODO) provided in the **`UserProfile.java`** class ***`main(…)`*** method to test **Task 4b**.

Consultation and other resources

1. Consultation sessions will be announced (LMS) prior to the submission date of this Assignment 2.
2. You are strongly advised to regularly check the Assignment 2 LMS section for any hints, clarification or corrections regarding Assignment 2.

What to submit

Electronic copy of all of the classes required to run the application, including those that are provided, are to be submitted to latcs8 using the **`submit OAD <directory or filename>`** command

Note that the submission is not through LMS.

All of your classes must be able to be compiled from their current directory.

This means, they must not be contained in any package.

As for the database, you can use the one on latcs7 or on your local machine. The only requirement is that your program should work on the database tables **grocery** and **item** which must have the same structure as the one in the provided MySQL script (**`CreateDatabaseScript.sql`**).

For each class that you submit, you must include, as part of the comments, at the beginning of each file,

- your full name (with surname in capital),
- your student ID,
- your user name (if different from student ID), and
- the subject code (CSE3OAD or CSE4OAD)

Assignment submissions without any of these will have 5% of the mark deducted.
