CS348 Project #4

Overview

In this project, you'll create a Cloud-based Web application to store course enrollment data for a university. This project will let you experiment with one of the latest industry practices for storing data.

Requirements

This project involves re-doing most of Project1, this time as a Web application using the Google App Engine (You should still read through this document, because the requirements are not exactly the same as those for Project1. Be sure to use the source files provided with the project, as they will make your job much easier).

As the database of your application, you will be using Google App Engine's Datastore, which provides a schema-less data storage mechanism. Your application will have a very simplistic interface, consisting of a text area for the user to enter commands and a "send command" button. You do not need to worry about the interface design, as the provided template already takes care of this. When users visit the application site in their browsers, they will see the following prompt:

Please enter a command in the text area below.

The application should support the following commands (parameters are separated by a single space, and every parameter consists of a single word/string with no spaces in it):

add_student SNUM SNAME SLEVEL AGE
add_faculty FID FNAME
add_class CNAME ROOM MEETS_AT FID
enroll SNUM CNAME
get_classes_for_student SNUM
get_instructors_for_student SNUM

You should implement each of the above commands. A detailed description of each command is provided below. For each of the above commands, you can assume that the input will always be formatted correctly, i.e. the number of parameters will be the same as that provided in the command list above and all parameters will be in the required domain. The only error cases you need to check are provided in the description for each command. You also do not need to check logical constraints such as a classroom being occupied by two different classes at the same time etc., as well as foreign key constraints. Those cases will not be tested. In the case that

something goes wrong with a command (which normally should not be the case, but you never know, life is full of surprises!), other than the errors listed below, please output your own descriptive error message. Note that each field of every table will be stored as a string this time, to make your job easier, and we do not have a "Department" table.

Description of commands

add_student:
SNUM – unique identifier for the student (string)
SNAME – name of the student (string)
SLEVEL – year of the student in the department ("Freshman", "Sophomore", "Junior" or "Senior") (string)
AGE – age of the student (string)

This command should add a student with the given parameters to the datastore. Addition of duplicate SNUM is not allowed. In the case of attempting duplicate SNUM addition, your program should print "Error: Student already exists!". If the insertion of the student succeeds, you should print "Command executed successfully!"

add_faculty:

FID – unique identifier for the faculty (string)
FNAME – full name of the faculty (string)

This command should add a faculty with the given parameters to the datastore. Addition of duplicate FID is not allowed. In the case of attempting duplicate FID addition, your program should print "Error: Faculty already exists!".

add_class:

CNAME – unique identifier for the class (string)
ROOM – the classroom for this class (string)
MEETS_AT – day and time this class meets at (string)
FID – identifier of the faculty teaching this class (string)

This command should add a class with the given parameters to the datastore. Addition of duplicate CNAME is not allowed. In the case of attempting duplicate CNAME addition, your program should print "Error: Class already exists!".

enroll:

SNUM – identifier of the student to enroll in the class
CNAME – identifier of the class the student will enroll in

This command should add an enrollment record with the given parameters to the datastore. Addition of duplicate SNUM-CNAME tuples is not allowed. In the case of attempting duplicate records, your program should print "Error: Enrollment already exists!".

get_classes_for_student:

This command should print the name, room, time and faculty identifier of each class the student identified by SNUM is taking, with fields of a record separated by commas and different records separated by semicolons.

Example:

get_classes_for_student 00178
CS34801, HAASG066, TR12:00pm, 101; CS30702, LWSN1106, MWF8:30am, 105

get_instructors_for_student:

This command should print the names of the instructors of all classes that the student identified by SNUM takes, with names separated by a semicolon (duplicate names are allowed)

Example:

get_instructors_for_student 00178
S.Layton; J. Lim

Getting Started

It is highly recommended that you do the entire project in the Eclipse IDE for Java EE (version 3.7 or below). If you prefer to work on the lab machines, you can download Eclipse (download link: http://wiki.eclipse.org/Older_Versions_Of_Eclipse) in any directory and run it from there, no additional setup/permission is needed. The Google App Engine Java tutorial at https://developers.google.com/appengine/docs/java/gettingstarted/introduction will provide all the necessary information for you to get started with the project if you

haven't used the Google App Engine before. The only parts of the tutorial you need to look at are "Installing the Java SDK", "Creating a Project", "Using JSPs" and "Using the Datastore".

To install the Google App Engine plugin for Eclipse, go to "Install New Software" under "Help" in Eclipse, put the installation location (e.g. https://dl.google.com/eclipse/plugin/3.7) for your version of Eclipse in the "Work With" textbox and hit Enter. This will bring up a list of plugins in the window below and you only need to select "Google Plugin for Eclipse" and "SDKs" in that list and click "Next". You should then proceed with the next steps to complete the plugin installation.

Your web application should be named "University" and the package for your source code should be named "university". Once you are done with the setup of the project (according to the tutorial), you should add the provided "ProcessCommandServlet.java" file to your package (university), add the provided "university.jsp" file to your "war" folder and replace the "web.xml" file in the folder "war/WEB-INF" with the web.xml we provide (You will notice ProcessCommandServlet.java corresponds to SignGuestbookServlet and university.jsp corresponds to guestbook.jsp in the tutorial).

The only file you will need to modify is ProcessCommandServlet.java. The parts you will need to implement in that file are clearly marked, with comments to guide you. For additional information on querying the datastore of the Google App Engine, you can check https://developers.google.com/appengine/docs/java/datastore/queries.

Once you are done with the project and have tested it offline, you should upload your application to the Google App Engine, following the instructions at https://developers.google.com/appengine/docs/java/gettingstarted/uploading. Make sure you do not exceed the free storage and access limits when testing your application online, so we can test it online too. You might also want to delete the data in the datastore generated by your application, but the results of the tests we perform should not be affected even if you do not (unless you coincidentally insert the same data that we will use in our tests).

Evaluation

Your project will be evaluated based on how well it meets the specification, i.e. the correctness of the output.

Submitting Your Work

Please create a README file that contains identifying information. For example:

CS348 - Project 4

Author: John Doe
Login: jdoe
Email: jdoe@cs.purdue.edu
Application location: http://*johns_app_id*.appspot.com/

Include here anything you might want us to know when grading your project.

All commands should be implemented in the template file provided (ProcessCommandServlet.java). This file takes care of posting the results of a command to the application website, so you don't need to worry about that.

To turn in your project, ssh to *lore.cs.purdue.edu*, create a folder named *project4* in your home directory and copy ProcessCommandServlet.java and README.txt to that folder. We should be able to run your application by opening your application location in a Web browser. If there happens to be an error with the online version, we should be able to run it in Eclipse, using the university.jsp and web.xml provided with the project.

After copying your files in the folder project4, execute the following command in your home directory:

turnin -c cs348 -p proj4 project4

To verify the contents of your submission, execute the following command:

turnin -c cs348 -p proj4 –v

Notes:

- Note that order doesn't matter when printing out the results of a query.
- Be careful about not letting duplicate records in your datastore. Remember duplicate records are identified using the primary key.
- Make sure the application doesn't throw errors for any commands.

- You do not need to do any input error checking (on the number of fields in a command, validity of a command etc.) for this project. The project will be tested with correctly formatted input.
- Note that you do not need to implement a “save” or “load” method this time, as the datastore takes care of that. For simplicity, we’re using a single database here, but the application can be modified to use multiple databases.
- To test your application, you can use the sample test provided with Project 1 (of course only with the commands and input parameters required in this project). We will be using a completely different dataset than that of Project 1 to test your application, so you should not have to worry about possible overlap of data even if you do not delete the data after your own tests.
- Including your application address in the README.txt is **VERY** important, so please do not forget to do that.