Database Management Systems

Term Project Instructions

This document lists the steps that every team must complete to start working on the term project. **WHEN FOLLOWING THESE INSTRUCTIONS OR LATER DURING THE COURSE, DON'T GIVE ANY CREDIT CARD INFORMATION TO ANY OF THE SERVICES WE ARE USING.**

Accounts

Every team member has to get a GitHub account and an IBM account.

1. **GitHub**: Visit https://github.com/ and use the "Sign up for GitHub" button to register.
2. **IBM**: Visit https://hub.jazz.net/ and press on the "SIGN UP" button. **GIVE YOUR ITU E-MAIL ADDRESS AS THE E-MAIL ADDRESS** and select "Student" for affiliation. You can use this IBM id both on JazzHub and on Bluemix.
   Although every member will have a separate account on JazzHub, only one of these accounts will be the team account on Bluemix. The team members have to agree on which member's account to use as the team account on Bluemix.

Log in to the GitHub, JazzHub, and Bluemix sites (https://bluemix.net/) and familiarize yourself with the interface.

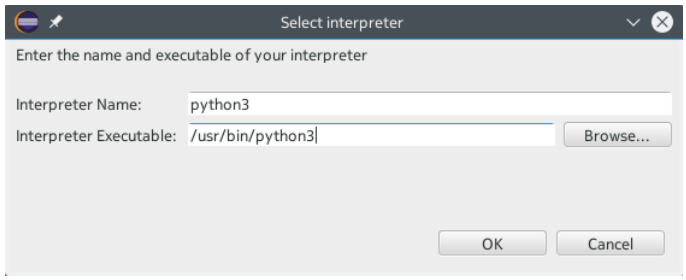
Installation

Every team member has to install the following tools on their computers:

1. **Git**: Most Linux distributions already include "git" as a package, so you can install it through the package manager. For other systems, visit https://git-scm.com/downloads.
2. **Python** (version 3.4): Most Linux distributions already include "python3" as a package, so you can install it through the package manager. For other systems, visit https://www.python.org/downloads/.
   1. **Flask**: On Linux distributions, look for a package named like "python3-flask". For other systems, visit http://flask.pocoo.org/.
   2. **Psycopg2**: On Linux distributions, look for a package named like "python3-psycopg2". For other systems, visit http://initd.org/psycopg/.
   3. **Sphinx**: On Linux distributions, look for a package named like "python3-sphinx". For other systems, visit http://sphinx-doc.org/.
3. **Eclipse** (Luna edition): Download it from https://www.eclipse.org/downloads/. Prefer the Luna edition over newer editions for compatibility with some plugins and choose the "Eclipse IDE for Java Developers" bundle. On Linux, it's safer **NOT** to use packages that might be provided by the distribution.

   Install the following plugins through the Eclipse Marketplace: **PyDev**, **AnyEdit Tools**, **HTML Editor**, **ReST Editor**.

Go to “Window → Preferences → PyDev → Interpreters → Python Interpreter” and add the executable of the Python interpreter you have installed. Make sure to give it the name “python3”.



Project Setup

The steps marked as “individually” have to be completed by every member for himself/herself. The steps marked as “as a team” have to be completed in a team session with every member present. Before following the steps on any of the videos, make sure to watch all of the video to understand what you are trying to accomplish.

1. (as a team) Follow the steps in the “How to create an organization for your team on GitHub” video at the address https://www.youtube.com/watch?v=bu24X2caphk
2. (as a team) Follow the steps in the “How to fork the skeleton project on GitHub” video at the address https://www.youtube.com/watch?v=Ndm29njmhj8
3. (as a team) Follow the steps in the “How to create the application on IBM Bluemix” video at the address https://www.youtube.com/watch?v=RemX1np9w3E.
4. (individually) Follow the steps in the “How to setup the project in Eclipse” video at the address https://www.youtube.com/watch?v=_iqxJ4MC58w. Every member should make a different change in the home page of application, such as including a hello message containing his/her own name. Also make sure that your name shows up correctly on the git log page of your project's GitHub repository.

Note that the videos are edited to leave out some parts that might take a long time such as building and deploying projects. The actual times it will take when you follow the steps might be much longer than in the video.

Database Setup

You can install PostgreSQL locally if you like but configuring it correctly might be difficult. Instead, you can use a virtual machine that's already setup for this course.

1. **VirtualBox**: Download it from https://www.virtualbox.org/wiki/Downloads. On Linux, it's safer **NOT** to use packages that might be provided by the distribution.
2. **Vagrant**: Download it from https://www.vagrantup.com/downloads.html. On Linux, it's safer **NOT** to use packages that might be provided by the distribution.
3. (only one member in the team) Open a “Git Bash” on your project folder and run the command:
   ```
   git pull https://github.com/uyar/itucsdb15.git master
   ```
   Next, do a git push.
4. On Git Bash, run the command:
   ```
   git pull -u
   ```
5. On Git Bash, run the command:
   ```
   vagrant up
   ```
   On the first run, this operation will download a very large disk image from the Internet and take a long time. You might prefer to do this through the university network, if your home connection is charged by the amount of your traffic. On subsequent runs, the disk image will not be downloaded again but the command will still take some time.

6. Open a web browser and visit the address http://localhost:50080/phppgadmin to manage the database.
7. On your computer, you can connect to the database with any PostgreSQL client -and also from your Python code- using the following credentials:
   ```
   host=localhost port=54321 dbname=itucsdb
   username=vagrant password=vagrant
   ```
8. If phppgadmin is not working on your setup, you can install pgadmin (http://www.pgadmin.org/) and use the credentials given above to connect to the database.
9. You can log in to the virtual machine through VirtualBox, or by running the following command using Git Bash:
   ```
   vagrant ssh
   ```

Important Notes

- Always do a pull before you do a push.
- Before pushing any changes to the master, test your code locally and make sure that it works. Don't push changes that break the project. If you do, you will also make life more difficult for your teammates.
- Conflicts are the most difficult issues that you will have to deal with. Look for lines starting with the prefix “>>>” to spot conflicts.
- Do not use any paths that only exist on your computer, such as “C:\...” because those paths won't be available on your teammates' computers and on Bluemix, where the application is supposed to run. Always use paths relative to the top of the project or to the environment, such as the home directory of the current user.
- If some team members are not doing their parts, try to isolate their responsibilities from the rest of the project.
- Try to divide the work of various team members into separate Python source files so that you can minimize the amount of files that multiple members might have to edit. This way, you will get fewer commit conflicts.
- Your application should work on only one database. Do not create different databases -and therefore database connections- for different parts of the project.
- Include a link or a button in your project that, when clicked, will initialize the database. It should create the necessary tables and fill them with some data that you can use in the demo.
- The Flask examples given in the exercise sessions are meant to get you started with the project. You are expected to improve on that information.