Introduction to Databases
Web applications in Python to query a database
Practice n. 5

The goal of this practice is to develop a simple web application based on Python, capable of inserting and modifying the content of a database.

Preliminary steps

This practice makes use of the Flask web server and the MySQL database, offered respectively by Python and XAMPP. In order to carry out this practice, both services must be started.

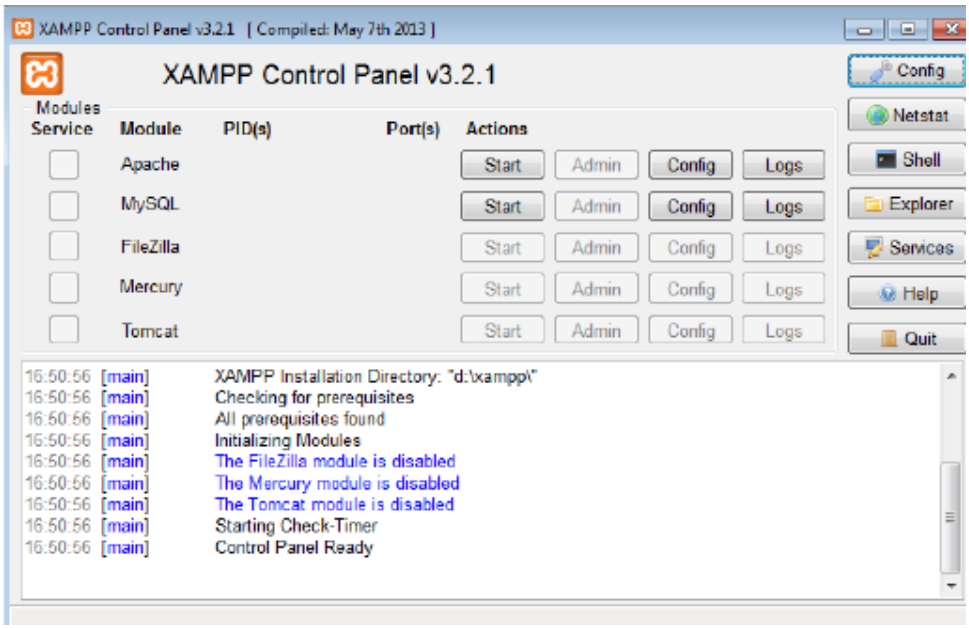
Boot MySQL server on localhost and start Apache

The execution of scripts with SQL commands for the creation and population of the database will be performed through the Web interface of MySQL. Before opening the Web interface of MySQL it is necessary to:

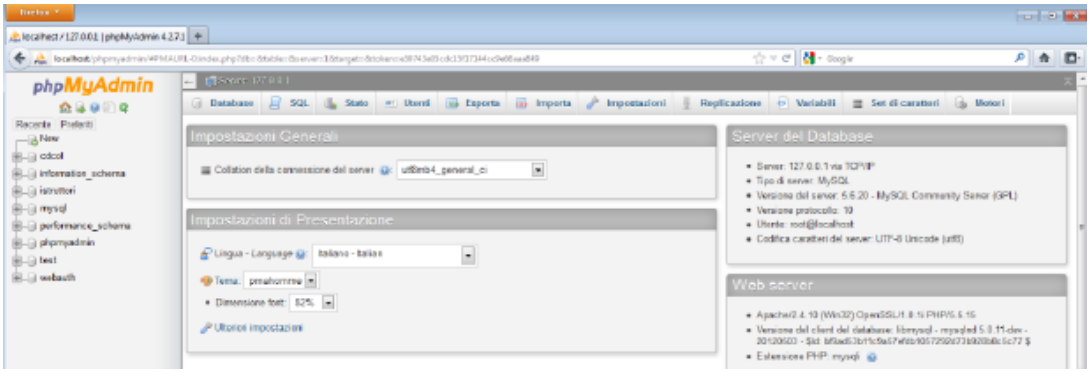
- Start the local Apache server;
- Start the local MySQL server.

Specifically, execute the following steps:

1. Start "XAMPP Control Panel".



2. Start Apache clicking the Start button in the row of "Apache" module.
3. Start MySQL clicking the Start button in the row of "MySQL" module.
4. Open the MySQL Web interface clicking the Admin button in the row of "MySQL" module (the browser will automatically open the URL associated to the page of administration and SQL querying, i.e., *phpMyAdmin*).



5. To execute a SQL script from the Web interface of MySQL:
   - Select the "Import" panel.
   - Select the file with the script you want to execute and click on "Go" button.
6. To execute the creation/population script more than once, you need to cancel any existing instance of the database, either directly from the "Database" panel or by including at the beginning of the script the commands for deleting the existing tables.

Creation and population of the database

The database used during this practice is the same as the one you created during the previous practice. The database is named GYM and it is about the activities in a gym. It is described by the following logical schema (primary keys are underlined, foreign keys are in italic, and optional attributed are denoted with *):

- TRAINER(SSN, Name, Surname, DateOfBirth, Email, PhoneNo*)
- COURSE(CId, Name, Type, Level)
- SCHEDULE (*SSN*, Day, StartTime, Duration, *CId*, GymRoom)

Create the GYM database and populate it using the *createDB.sql* and *populateDB.sql* scripts found on the course's website.

After the execution of the scripts, the tables will contain the following data:

TRAINER table

SSN	Name	Surname	DateOfBirth	Email	PhoneNo
SMTPLA80N31B791Z	Paul	Smith	31/12/1980	p.smith@gym.it	*NULL*
KHNJHN81E30C455Y	John	Johnson	30/5/1981	j.johnson@gym.it	+2300110303444
AAAGGG83E30C445A	Peter	Johnson	30/5/1981	p.johnson@gym.it	+2300110303444

COURSE table

CId	Name	Type	Level
CT100	Spinning for beginners	Spinning	1
CT101	Fitdancing	Music activity	2
CT104	Advanced spinning	Spinning	4

SCHEDULE table

SSN	Day	StartTime	Duration	CId	GymRoom
SMTPLA80N31B791Z	**Monday**	**10:00**	45	CT100	R1
SMTPLA80N31B791Z	**Tuesday**	**11:00**	45	CT100	R1
SMTPLA80N31B791Z	**Tuesday**	**15:00**	45	CT100	R2
KHNJHN81E30C455Y	**Monday**	**10:00**	30	CT101	R2
KHNJHN81E30C455Y	**Monday**	**11:30**	30	CT104	R2
KHNJHN81E30C455Y	**Wednesday**	**9:00**	60	CT104	R1

Publishing/loading a dynamic web page

In order to publish a dynamic web page connected to a database through Python, it's necessary to install Flask, SQLAlchemy and the necessary dependencies.

- install flask
- install sqlalchemy
- install mysqlclient

Provided you have `pip` installed and working (please verify how to use it on your specific operative system), you can install everything with the following command:

```
pip install Flask SQLAlchemy mysqlclient
```

Exercises

Develop a web application in Python capable of updating the GYM database through the web.

1. *Inserting new courses.* Create a web page containing a form which requires all the necessary data to insert a new course in the database (CId, Name, Type, Level).

   - The application must verify that all fields have been inserted, and the *Level* field is an integer number between 1 and 4.

   If a field is missing, or a key is duplicated, or the value of the *Level* field is outside the acceptable range, an error message must be displayed. Else, if all the fields are correct and the insert operation succeeds, a success message must be displayed.

   Figure 1 and Figure 2 show an example of the usage of the current functionality to insert new courses.

2. *Inserting a new weekly lesson in the schedule.* Create a dynamic web page in Python containing a form that allows to insert a new weekly lesson in the SCHEDULE table. The form must allow to insert all the necessary fields (SSN, Day, StartTime, Duration, CId, GymRoom) regarding the scheduling of a new lesson.

   - The selection of the trainer must happen using a dropdown menu containing Surname, Name, and SSN of the various instructors contained in the database.
   - Similarly, the selection of the course also has to happen using a dropdown menu populated with data contained in the database.
   - The other fields instead are simple textual field populated manually by the user.
   - The application must verify that the user isn't trying to insert a lesson longer than 60 minutes, and that the inserted day is between Monday and Friday (no weekend).
   - The insertion of a new lesson in the schedule has to be allowed and executed if, and only if, there are no other lessons of the same course in the same day of the week.

   If the request respects all constraints and the insert operation succeeds, a success message must be displayed, else an error message must be displayed (the error message must include the type of problem that caused the error)

   Warning: the SQL query that verifies if a lesson of a course isn't already scheduled for the same day has to be part of the same transaction as the insert operation (otherwise, problems might arise if different users try to insert multiple lessons for the same course on the same day concurrently).

   Figure 3 and Figure 4 show an example of the usage of the current functionality to insert new weekly lessons.

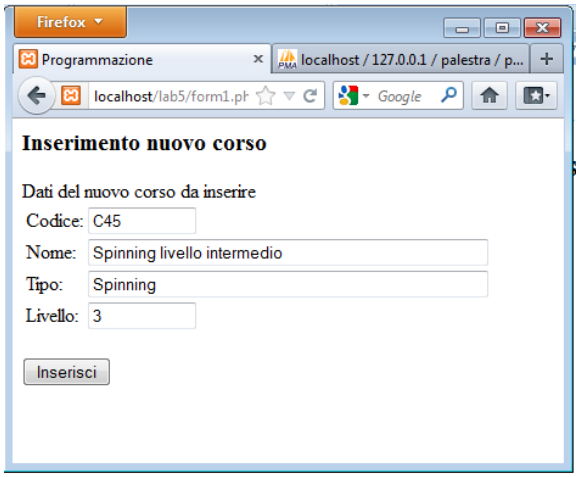


Figure 1: Form example for ex. 1

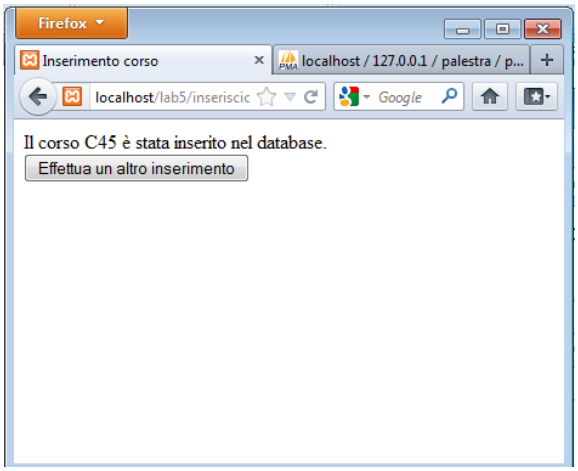


Figure 2: Outcome example for ex. 1

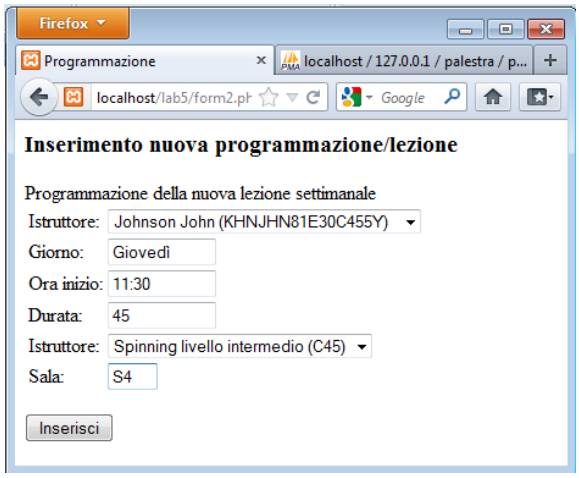


Figure 3: Form example for ex. 2

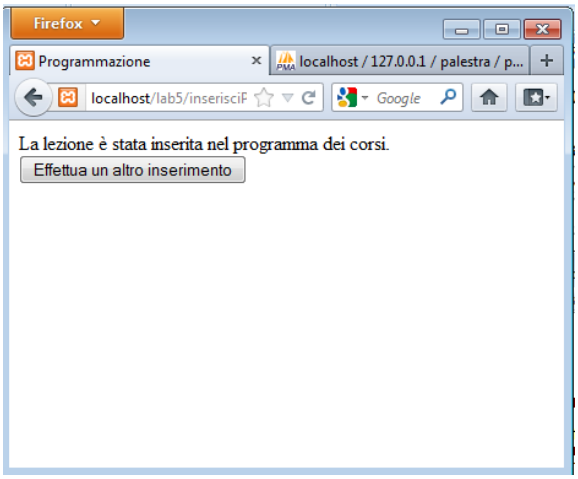


Figure 4: Outcome example for ex. 2