Datenbanken und Web-Techniken
Exercise 1:
DB Access 1: PHP and CGI

General Database Access

In general you follow these steps, to access a database:

1. Open the connection
   - Create a database-handle
   - Connect through this handle
2. Execute a query
   - Create a statement-handle
   - Bind the result-buffer to the statement-handle
   - Execute a query through this handle
3. Fetch the result
   - Request more result-tuple, while there is data
4. Close the connection
   - Free all handles

DBMS-specific Database Access using native API

For historical reasons, DBMS have their own APIs to provide access to their data. For PostgreSQL this is the C client library libpq. The provided C-CGI-example-script shows the usage of some basic functions.

A short abstract to illustrate the four steps, to access a PostgreSQL database with C using native API:

1. Open the connection

```
db_handle = PQsetdbLogin(host, port, NULL, NULL, databaseName, userName, password)
```

2. Execute a query

```
PGresult *result = PQexec(db_handle, query)
```

3. Fetch the result

```
PQgetvalue(result, i, j)
```

4. Close the connection

```
PQclear(result)
PQfinish(db_handle)
```

DBMS-specific Database Access using native API provided through PHP abstraction layer

This same client library is used and slightly abstracted by PHP to provide all functionality. The provided PHP-API-example-script again shows the usage of some basic functions.

A short abstract to illustrate the four steps, to access a PostgreSQL database with PHP using API:

1. Open the connection

```
$db_handle = pg_connect("host=" . $host . " port=" . $port . " dbname="
. $databaseName . " user=" . $userName . " password=" . $password)
```

2. Execute a query

```
$result = pg_query($db_handle, "SELECT * FROM " . $tableName)
```

3. Fetch the result

```
foreach(pg_fetch_row($result, $ri) as $value)
```

4. Close the connection

```
pg_close($db_handle)
```

DBMS-independent Database Access using PHP Data Objects

Another option to abstract libpq in PHP is to use PHP Data Objects (PDO). This is an higher level of abstraction and has the advantage of easy DBMS exchange (just change some values in the connection string). There is also a PHP-PDO-example-script provided, which illustrates the usage.

A short abstract to illustrate the four steps, to access a PostgreSQL database with PHP using PDO:

1. Open the connection

```
$db_handle = new PDO("pgsql:" . "host=" . $host . ";port=" . $port .
";dbname=" . $databaseName, $userName, $password)
```

2. Execute a query

```
$result = $db_handle->query("SELECT * FROM " . $tableName, PDO::FETCH_NUM)
```

3. Fetch the result (iterating over the result object fetches the content internally)

```
foreach($result as $row)
```

4. Close the connection

```
$result = null
$db_handle = null
```

Tasks
Hints and more description follow on the following pages

Preparation Task:

1. Download and compare the three provided example scripts
2. Get a PostgreSQL database
3. Import the provided pizzen.sql into your PostgreSQL database

PHP Task:

4. Modify both provided PHP scripts to use your database and created table
5. Run both PHP scripts and see the data from your table in the returned HTML pages

CGI Task:

6. Modify the provided C script to use your database and created table
7. Compile the C script as a CGI script
8. Run your CGI script and see the data from your table in the returned HTML page

Preparation Task 1: Download and compare the three provided example scripts

- Download the provided C example script **`postgresql_cgi.c`** or copy and paste the content to some text-file
- Do the same with the provided PHP example scripts **`postgresql_api.php`** and **`postgresql_pdo.php`**
- Try to understand the functioning of all three scripts and see the similarities and differences
  - the introduction slides to this exercise might help to get along

Preparation Task 2: Get a PostgreSQL database

You can

- install one locally on your own computer or set it up on a server, you have access to
- or simply use the service form the URZ and get one at **`IdM-Portal → Databases → PostgreSQL`**
  ➔https://idm.hrz.tu-chemnitz.de/user/service/database/postgresql/add/
  - enter name (important) and description (doesn't matter) of database
  - memorize (or save) read-write-username and password
  - please be aware, that the PostgreSQL-server from URZ is only reachable from inside the network of TU Chemnitz (use a VPN to access from outside)

URZ provides PostgreSQL version 9.6, but it should be possible to work with an older or more recent one, too.

Preparation Task 3: Import the provided pizzen.sql into your PostgreSQL database

- If you use URZ service, you can just use phpPgAdmin service from URZ
  ➔https://wfm.hrz.tu-chemnitz.de/phppgadmin/
  - you might like to set *Sprache* to **English** or something different (default language is German)
  - select server **PostgreSQL** at the left side
  - enter read-write-username with corresponding password
  - select your database and open SQL-console windows in the upper right corner
- Otherwise set up your own phpPgAdmin for your database or any other frontend you like or be familiar with (e.g. psql, pgAdmin3 or pgAdmin4)
- Execute the SQL queries from the provided **`pizzen.sql`** in your database
  - you can just copy and past the lines into some SQL-console window in your used GUI
  - this creates a new table named **`pizzen`** and adds eleven pizzas
    - will be required by next exercise and can be used in this exercise (otherwise create different table)

PHP Task 4: Modify both PHP scripts to use your database and created table

- Edit your saved PHP scripts with any editor your are familiar with (even Windows Notepad should be sufficient)
- There are several variables defined at the beginning (lines 5 to 10)
  - change them accordingly (and replace **`your_…`** with the correct values)
  - you might use the provided **`pizzen`** table or any other available table of your choice

PHP Task 5: Run both PHP scripts and see the data from your table in the returned HTML page

You can

- run PHP locally on your own computer as stand-alone or within some web-server
- or set it up on a server, you have access to
- or simply use the service form the URZ.

URZ provides version 7.2 but it should also work with an older or more recent one.

If you like to use the service from the URZ, you have to do several steps, as described at the next page.

PHP Task 5: Run both PHP scripts and see the data from your table in the returned HTML page by using the service from URZ

- Activate your personal homepage in IDM portal:
  ➔https://idm.hrz.tu-chemnitz.de/user/security/#server_security
  - changes take about 15 minutes
- Put your PHP scripts into your **`public_html`** directory inside your URZ home directory
  - you may use SSH (if activated in IDM portal) to upload the file, but there is also Web-File-Manager:
    ➔https://wfm.hrz.tu-chemnitz.de/wfm/
  - make sure you rename the PHP-files to extension **`.php`** (otherwise they won't work)
  - there is also a PHP-file-editor in Web-File-Manager, so you can just edit and test your files online
- Access your files by calling the URL in the browser
  - https://www-user.tu-chemnitz.de/~*`your_urz_id`*/*`your_php_script_filename.php`*
- If you don’t see any result, you certainly did some mistake
  - to see PHP-error-messages create a file named **`.htaccess`** in your **`public_html`** directory, edit this file and put **`php_flag display_errors on`** there

CGI Task 6: Modify the provided postgresql_cgi_test.c script to use your database and created table

- Download the provided **`postgresql_cgi_test.c`** script or copy and paste the content to some text-file
- Edit your file with any editor your are familiar with (even Windows Notepad should be sufficient)
- There are several variables defined at the beginning (lines 11 to 16)
  - change them accordingly (and replace **`your_…`** with the correct values)

CGI Task 7: Compile the C script as a CGI script

- This could be done at any computer with the required C compiler
  - PostgreSQL header-files and libpq are also required
    - try to use the same version as your database (otherwise your program might crash)
- You can also use some server from URZ
  - connecting there trough SSH
    - **`ssh your_urz_id@login.tu-chemnitz.de`**
    - no problem for Linux or MacOS users from the Console / Terminal / Shell
    - Windows users might use PuTTY
    - you have to activate this service in IDM portal (see PHP Task 5)
  - compile as CGI script
    - copy your C script to your working directory
    - **`cc -I/usr/include/ -lpq -o output_name.cgi postgresql_cgi_test.c`**
    - the file-extension **`.cgi`** is required by many servers

CGI Task 8: Run your CGI script and see the data from your table in the returned HTML page

You can

- run a CGI activated web-server locally on your own computer
- or use some web-server, you have access to
- or simply use the service form the URZ.

If you like to use the service from the URZ, you have to do several steps, as described at PHP Task 5 (but replace PHP with CGI).

If you use your own server on a Non-Windows-OS, you might have to give the script execution permissions (**`chmod +x output_name.cgi`**).

Please be aware, that you have to compile your script for your target OS. If you compile and run it on an URZ-Linux-server, you can't copy the CGI to your Windows-PC and expect, that this magically can be executed.