Web Interfaces for Database Servers

1. CGI, MySQLdb, and Sockets
   - glueing the connections with Python
   - functions of the server: connect, count, and main
   - development of the code for the client

2. Displaying all Records in HTML Table
   - extending the web interface
   - retrieving and packing records
   - the client displays HTML table

3. Displaying Sorted Records in Order
   - radio buttons in HTML form
   - processing forms with CGI scripts
   - updated code for server and two clients

MCS 275 Lecture 29
Programming Tools and File Management
Jan Verschelde, 17 March 2017
Web Interfaces for Database Servers

1. CGI, MySQLdb, and Sockets
   - glueing the connections with Python
   - functions of the server: connect, count, and main
   - development of the code for the client

2. Displaying all Records in HTML Table
   - extending the web interface
   - retrieving and packing records
   - the client displays HTML table

3. Displaying Sorted Records in Order
   - radio buttons in HTML form
   - processing forms with CGI scripts
   - updated code for server and two clients
CGI, MySQLdb, and Sockets

Glued by Python scripts

Goal: build web interface to MySQL database.

Components:
1. server is Python script using `pymysql` (or `MySQLdb`)
2. client is a CGI script: web interface

Example database: `OurPyFiles with scripts table`.

Steps in incremental development:
1. script to count number of records
2. server listens to one connection sends to client number of records
3. run client script first on command line
4. second version of client script writes plain text on web page
counting the number of records

scripts_count.py prints number of records in the table scripts of the MySQL database OurPyFiles.

```python
# import MySQLdb
# db = MySQLdb.connect(db='OurPyFiles')

import pymysql

OURDB = pymysql.connect(db='OurPyFiles')
CRS = OURDB.cursor()
QRY = 'select count(*) from scripts'
CRS.execute(QRY)
RES = CRS.fetchone()
NBR = int(RES[0])
print('the number of scripts : %d' % NBR)
```
Web Interfaces for Database Servers

1. CGI, MySQLdb, and Sockets
   - glueing the connections with Python
   - functions of the server: connect, count, and main
   - development of the code for the client

2. Displaying all Records in HTML Table
   - extending the web interface
   - retrieving and packing records
   - the client displays HTML table

3. Displaying Sorted Records in Order
   - radio buttons in HTML form
   - processing forms with CGI scripts
   - updated code for server and two clients
a simple client/server interaction

In one terminal window, we launch the server script:

```
$ python scripts_server.py
server waits for connection
server accepted connection from ('127.0.0.1', 54814)
server connects to database
server sends #scripts to client
count sent, closing off
$
```

In another terminal, we execute the client script:

```
$ python scripts_client.py
client is connected
client received "96"
$
```
the functions of the server

import pymysql # import MySQLdb
from socket import socket as Socket

def connect():
    """
    Returns client and server socket to communicate with one client.
    """

def count():
    """
    Returns the number of scripts.
    """

def main():
    """
    Accepts connection and sends #scripts.
    """
defining network connections

HOSTNAME = ''  # use any address
PORTNUMBER = 11267  # number for the port
BUFFER = 80  # size of the buffer

def connect():
    """
    Returns client and server socket to communicate with one client.
    """

    from socket import AF_INET, SOCK_STREAM
    server_address = (HOSTNAME, PORTNUMBER)
    server = Socket(AF_INET, SOCK_STREAM)
    server.bind(server_address)
    server.listen(1)
    print('server waits for connection')
    client, client_address = server.accept()
    print('server accepted connection from ',
          client_address)
    return client, server
def count():
    """
    Returns the number of scripts.
    """
    ourdb = pymysql.connect(db='OurPyFiles')
    crs = ourdb.cursor()
    qry = 'select count(*) from scripts'
    crs.execute(qry)
    res = crs.fetchone()
    nbr = int(res[0])
    return nbr
def main():
    
    """
    Accepts connection and sends result.
    """
    client, server = connect()
    print('server connects to database')
    nbr = count()
    print('server sends #scripts to client')
    data = str(nbr)
    client.send(data.encode())
    print('count sent, closing off')
    server.close()
Web Interfaces for Database Servers

1. CGI, MySQLdb, and Sockets
   - gluing the connections with Python
   - functions of the server: connect, count, and main
   - development of the code for the client

2. Displaying all Records in HTML Table
   - extending the web interface
   - retrieving and packing records
   - the client displays HTML table

3. Displaying Sorted Records in Order
   - radio buttons in HTML form
   - processing forms with CGI scripts
   - updated code for server and two clients
first version in `scripts_client.py`

```python
def main():
    from socket import socket as Socket
    from socket import AF_INET, SOCK_STREAM

    HOSTNAME = 'localhost'  # on same host
    PORTNUMBER = 11267  # same port number
    BUFFER = 80  # size of the buffer

    SERVER_ADDRESS = (HOSTNAME, PORTNUMBER)
    CLIENT = Socket(AF_INET, SOCK_STREAM)
    CLIENT.connect(SERVER_ADDRESS)

    print('client is connected')
    DATA = CLIENT.recv(BUFFER).decode()
    print('client received "' + DATA + '"')

    CLIENT.close()
```

Programming Tools (MCS 275)
the second client is a web interface

http://localhost...0/scripts_web.py

localhost:8000/scripts_web.py

client is connected
Number of scripts : 96
making it work ...

For this web interface to work, the following must be

1. The mysql daemon is up and running.
   
   $ sudo mysqld_safe

2. The database server script runs.
   
   $ python scripts_server.py
   server waits for connection

3. The CGI server script runs.
   
   $ python myserver.py
   welcome to our cgi server
   press ctrl c to shut down the server

Then we point the browser to localhost:8000/scripts_web.py
second client in scripts_web.py

```python
print("Content-Type: text/plain\n")

from socket import socket as Socket
from socket import AF_INET, SOCK_STREAM

HOSTNAME = 'localhost'  # on same host
PORTNUMBER = 11267      # same port number
BUFFER = 80            # size of the buffer

SERVER_ADDRESS = (HOSTNAME, PORTNUMBER)
CLIENT = Socket(AF_INET, SOCK_STREAM)
CLIENT.connect(SERVER_ADDRESS)

print('client is connected')
DATA = CLIENT.recv(BUFFER).decode()
print('Number of scripts: ' + DATA)
```
# Web Interfaces for Database Servers

1. **CGI, MySQLdb, and Sockets**
   - gluing the connections with Python
   - functions of the server: connect, count, and main
   - development of the code for the client

2. **Displaying all Records in HTML Table**
   - extending the web interface
   - retrieving and packing records
   - the client displays HTML table

3. **Displaying Sorted Records in Order**
   - radio buttons in HTML form
   - processing forms with CGI scripts
   - updated code for server and two clients
To see all records on a web page:

1. server sends number of records to client
2. client receives number of records
3. server sends all records to client
4. client receives all records and makes HTML table to display

Synchronization is very important: *for every send of the server, there must be a matching recv by the client!*
listing of the scripts in a browser

Number of scripts: 96

0  L-25 2016-03-09 filldb.py
1  L-25 2016-03-09 grabpyhead.py
2  L-25 2016-03-09 viewdbdata.py
3  Q-1  2016-01-12 bell.py
4  Q-5  2016-02-09 bellmem.py
5  L-5  2016-01-22 billiards_basic.py
6  L-15 2016-02-15 binarysearch.py
functions of the server in scripts_servdb.py

def connect_client():
    """
    Returns client and server socket.
    """

def count_records(crs):
    """
    Given cursor crs, returns the number of scripts.
    """

def retrieve_records(crs):
    """
    Given cursor crs, returns all records.
    """

def pack_tuple(tup):
    """
    Packs the data tuple as a string.
    """
The main function in `scripts_servdb.py`

```
def main():
    """
    Accepts connection and sends records.
    """
    ourdb = pymysql.connect(db='OurPyFiles')
    crs = ourdb.cursor()
    nbr = count_records(crs)
    client, server = connect_client()
    client.send(str(nbr).encode())
    records = retrieve_records(crs)
    for record in records:
        client.send(pack_tuple(record))
    server.close()
```
Web Interfaces for Database Servers

1. CGI, MySQLdb, and Sockets
   - glueing the connections with Python
   - functions of the server: connect, count, and main
   - development of the code for the client

2. Displaying all Records in HTML Table
   - extending the web interface
   - retrieving and packing records
   - the client displays HTML table

3. Displaying Sorted Records in Order
   - radio buttons in HTML form
   - processing forms with CGI scripts
   - updated code for server and two clients
counting and retrieving records

def count_records(crs):
    """
    Returns the number of scripts, given the cursor crs.
    """
    qry = 'select count(*) from scripts'
crs.execute(qry)
result = crs.fetchone()
nbr = int(result[0])
return nbr

def retrieve_records(crs):
    """
    Given cursor crs, returns all records.
    """
    qry = 'select * from scripts'
crs.execute(qry)
return crs.fetchall()
packing tuple records into a string

A record is returned as a data tuple

('L', '26', '2017-03-10', 'guidb1.py')

To send to the client, we pack as 'L-26:2017-03-10:guidb1.py'.

def pack_tuple(tup):
    """
    Packs the tuple as string with items separated by colons. Notice padding!
    """
    result = tup[0] + '-' + str(int(tup[1])) + ':' + str(tup[2]) + ':' + tup[3] + ':'
    result += (BUFFER - len(result))*''
    return result
Web Interfaces for Database Servers

1. CGI, MySQLdb, and Sockets
- glueing the connections with Python
- functions of the server: connect, count, and main
- development of the code for the client

2. Displaying all Records in HTML Table
- extending the web interface
- retrieving and packing records
- the client displays HTML table

3. Displaying Sorted Records in Order
- radio buttons in HTML form
- processing forms with CGI scripts
- updated code for server and two clients
code for the client in scripts_showall.py

#!/usr/bin/python

from socket import socket as Socket
from socket import AF_INET, SOCK_STREAM

HOSTNAME = 'localhost'  # on same host
PORTNUMBER = 11267  # same port number
BUFFER = 80  # size of the buffer

def print_header(title):
    """
    writes title and header of page
    """
    print("""Content-type: text/html

<html>
<head>
<title>%s</title>
</head>
<body>"" % title)
the main function in the client

def main():
    
    """
    Connects and prints data of server.
    """

    print_header('showing all scripts')
    server_address = (HOSTNAME, PORTNUMBER)
    client = Socket(AF_INET, SOCK_STREAM)
    client.connect(server_address)
    data = client.recv(BUFFER).decode()
    nbr = int(data)
    print("<B>Number of scripts : %d</B>" % nbr)
    retrieve_table(client, nbr)
    client.close()
def retrieve_table(sock, nbr):
    """
    Retrieves table of nbr records, using socket sock to communicate.
    """
    print("<table>")
    for i in range(nbr):
        data = sock.recv(BUFFER).decode()
        record = data.split(':')
        print("<tr>")
        print("<td>%d</td>" % i)
        print("<td>%s</td>" % record[0])
        print("<td>%s</td>" % record[1])
        print("<td>%s</td>" % record[2])
        print("</tr>")
    print("</table>")
Web Interfaces for Database Servers

1. CGI, MySQLdb, and Sockets
   - glueing the connections with Python
   - functions of the server: connect, count, and main
   - development of the code for the client

2. Displaying all Records in HTML Table
   - extending the web interface
   - retrieving and packing records
   - the client displays HTML table

3. Displaying Sorted Records in Order
   - radio buttons in HTML form
   - processing forms with CGI scripts
   - updated code for server and two clients
Radio Buttons for Sort Order

This HTML form is stored in

- users public_html directory on Unix
- users Sites directory on Mac OS X
CGI script to confirm the choice

This CGI script is stored in

- `/var/www/cgi-bin` on Unix
- `/Library/WebServer/CGI-Executables` on Mac OS X
HTML code in the body of `sort_order.html`

```html
<h1> determine sort order </h1>
<form action="http://localhost/cgi-bin/sort_order.py">
<p>
sort by
<input type="radio" name="sort" value = 0 checked> type
<input type="radio" name="sort" value = 1> date
<input type="radio" name="sort" value = 2> name
<br>
order is
<input type="radio" name="order" value = True checked> ascending
<input type="radio" name="order" value = False> descending
</p>
<p> <input type="submit"> </p>
```
the script `sort_order.py`

```python
#!/usr/bin/python
import cgi
import cgitb
cgitb.enable()
FORM = cgi.FieldStorage()
SORTBY = FORM['sort'].value
ORDERIS = FORM['order'].value
if SORTBY == '0':
    PRT = 'sort by type and number'
elif SORTBY == '1':
    PRT = 'sort by date'
else:
    PRT = 'sort by name'
if ORDERIS == 'True':
    PRT = PRT + ' in ascending order'
else:
    PRT = PRT + ' in descending order'
print("Content-Type: text/plain
"
)
print(PRT)
```
Web Interfaces for Database Servers

1. CGI, MySQLdb, and Sockets
   - glueing the connections with Python
   - functions of the server: connect, count, and main
   - development of the code for the client

2. Displaying all Records in HTML Table
   - extending the web interface
   - retrieving and packing records
   - the client displays HTML table

3. Displaying Sorted Records in Order
   - radio buttons in HTML form
   - processing forms with CGI scripts
   - updated code for server and two clients
Processing Forms with CGI Scripts

how to do it

Good for testing:

1. *f.html* has form, action refers to *f.py*
2. *f.py* defines CGI script, invoked by *submit*

Integrated approach: Python scripts printing HTML.

Database server listens to two clients:

1. First client displays number of records, prints the form for the sort order, and activates the second client
2. Second client processes the form, sends sort order to server, and retrieves and displays sorted records

Both clients after connection receive the number of records in the table.
Web Interfaces for Database Servers

1 CGI, MySQLdb, and Sockets
   • glueing the connections with Python
   • functions of the server: connect, count, and main
   • development of the code for the client

2 Displaying all Records in HTML Table
   • extending the web interface
   • retrieving and packing records
   • the client displays HTML table

3 Displaying Sorted Records in Order
   • radio buttons in HTML form
   • processing forms with CGI scripts
   • updated code for server and two clients
The server listens to two clients.

def main():
    """
    Accepts connection and sends records.
    """
    ourdb = pymysql.connect(db='OurPyFiles')
    crs = ourdb.cursor()
    nbr = count_records(crs)
    sortclient, server = connect_client()
    sortclient.send(str(nb).encode())
    print('wait for submit client')
submitclient, adr = server.accept()
print('submit client is connected')
submitclient.send(str(nbr).encode())
sortorder = submitclient.recv(BUFFER).decode()
print('received sort order "' + sortorder + '"')
records = retrieve_records(crs, sortorder)
print('sending records ...')
for record in records:
    submitclient.send(pack_tuple(record).encode())
print('closing connection')
server.close()
def main():
    """
    Connects and prints data of server.
    """
    print_header('sorting all scripts')
    server_address = (HOSTNAME, PORTNUMBER)
    client = Socket(AF_INET, SOCK_STREAM)
    client.connect(server_address)
    data = client.recv(BUFFER).decode()
    nbr = int(data)
    print("<B>Number of scripts : %d</B>" % nbr)
    prompt_sort_order()
    client.close()
def main():
    
    """
    Connects and prints data of server.
    """
    print_header('showing all scripts')
    server_address = (HOSTNAME, PORTNUMBER)
    client = Socket(AF_INET, SOCK_STREAM)
    client.connect(server_address)
    data = client.recv(BUFFER).decode()
    nbr = int(data)
    print("<b>Number of scripts : %d</b>" % nbr)
    send_sort_order(client)
    retrieve_table(client, nbr)
    client.close()
first client in scripts_sort.py prompts sort order

def prompt_sort_order():
    """
    Displays a form to ask the user for the field to sort on and the order.
    """
    print(""
    <form action="http://localhost/cgi-bin/scripts_sortall.py">
    <p>
        ... rest of html code ...
    """

    Note: if myserver.py is used, the action is defined as

    <form action="http://localhost:8000/scripts_sortall.py">
def send_sort_order(sock):
    
    Sends sort order to server
    using the client socket sock.
    
    form = cgi.FieldStorage()
    sortby = form[‘sort’].value
    if bool(form[‘order’].value):
        sortby = sortby + ‘+’
    else:
        sortby = sortby + ‘-’
    sock.send(sortby.encode())
def retrieve_records(crs, sortorder):
    """
    Given cursor crs, returns all records, taking sortorder into account.
    """
    qry = 'select * from scripts'
    if sortorder[0] == '0':
        qry = qry + ' order by t,n'
    elif sortorder[0] == '1':
        qry = qry + ' order by d'
    else:
        qry = qry + ' order by f'
    if sortorder[1] == '+':
        qry = qry + ' asc'
    else:
        qry = qry + ' desc'
    crs.execute(qry)
    return crs.fetchall()
Summary + Exercises

We introduced web interfaces to database servers. Python glues CGI, MySQLdb, and Sockets.

Exercises:

1. Provide a web interface to enter data in the table scripts. Use an HTML page to enter all data where the submit will activate a client of the database server. The client sends the user data to the server, the server adds it and sends feedback to the client.

2. Use tables typedate and filedata of Lecture 23 to make a web interface to retrieve records based on keys. Start at an HTML page with an input element to enter a key. The action in the form launches a client of the database server. The server retrieves the record and sends the data to the client for display.