UNIT 5

Python MySQL Database Access

- MySQLdb is an interface for connecting to a MySQL database server from Python.

Creating Database Table:

Example:

```
import MySQLdb

db=MySQLdb.connect("localhost","testuser","test123","TESTDB")

cursor = db.cursor()

sql = "CREATE TABLE EMPLOYEE(FIRST_NAME  CHAR(20) NOT
NULL,LAST_NAME  CHAR(20),AGE INT,SEX CHAR(1),INCOME FLOAT )"

cursor.execute(sql)

db.close()
```

INSERT Operation:

- It is required when you want to create your records into a database table.

Example:

```
import MySQLdb

db=MySQLdb.connect("localhost","testuser","test123","TESTDB" )

cursor = db.cursor()

sql = "INSERT INTO EMPLOYEE(FIRST_NAME, LAST_NAME, AGE, SEX,
INCOME) VALUES ('Mac', 'Mohan', 20, 'M', 2000)"

try:

   cursor.execute(sql)
```

```
   db.commit()
except:
   db.rollback()
db.close()
```

READ Operation:

- READ Operation on any database means to fetch some useful information from the database.
- **fetchone():**
  - Method to fetch single record from a database table.
  - It fetches the next row of a query result set. A result set is an object that is returned when a cursor object is used to query a table.
- **fetchall():**
  - Method to fetch multiple values from a database table.
  - It fetches all the rows in a result set. If some rows have already been extracted from the result set, then it retrieves the remaining rows from the result set.
- **rowcount:** This is a read-only attribute and returns the number of rows that were affected by an execute() method.

Example:

- The following procedure queries all the records from EMPLOYEE table having salary more than 1000

```
import MySQLdb
db=MySQLdb.connect("localhost","testuser","test123","TESTDB" )
cursor = db.cursor()
sql = "SELECT * FROM EMPLOYEE WHERE INCOME > '%d'" %(1000)
try:
   cursor.execute(sql)
```

```
      results = cursor.fetchall()
      for row in results:
         fname = row[0]
         lname = row[1]
         age = row[2]
         sex = row[3]
         income = row[4]
         print "fname=%s,lname=%s,age=%d,sex=%s,income=%d"
   %(fname, lname, age, sex, income )
   except:
      print "Error: unable to fecth data"
   db.close()
```

DELETE Operation:

- DELETE operation is required when you want to delete some records from your database.

Example: Following is the procedure to delete all the records from EMPLOYEE where AGE is more than 20

```
import MySQLdb
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
cursor = db.cursor()
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
   cursor.execute(sql)
   db.commit()
except:
```

```
   db.rollback()
db.close()
```

Update Operation:

- UPDATE Operation on any database means to update one or more records, which are already available in the database.

Example:

```
import MySQLdb
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
cursor = db.cursor()
sql = "UPDATE EMPLOYEE SET AGE = AGE + 1 WHERE SEX = '%c'" %
('M')
try:
   cursor.execute(sql)
   db.commit()
except:
      db.rollback()
db.close()
```

COMMIT Operation:

- Commit is the operation, which gives a green signal to database to finalize the changes, and after this operation, no change can be reverted back.

Example: db.commit()

ROLLBACK Operation:

- If you are not satisfied with one or more of the changes and you want to revert back those changes completely, then use rollback() method.

Example: db.rollback()

Disconnecting Database

- To disconnect Database connection, use close() method.

Example: db.close()

Python CGI Programming

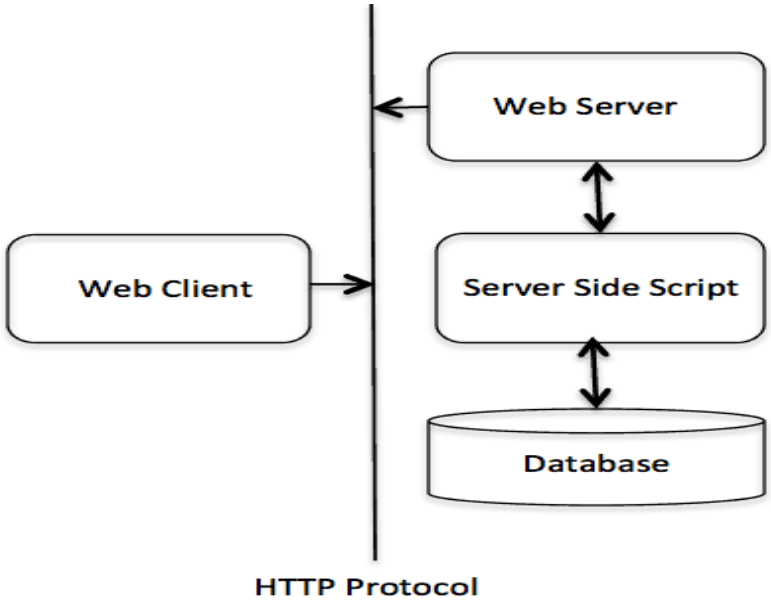
The Common Gateway Interface (CGI)

- Is a set of standards that define how information is exchanged between the web server and a custom script.
- Is a standard for external gateway programs to interface with information servers such as HTTP servers.

Web Browsing

- Your browser contacts the HTTP web server and demands for the URL i.e., filename.
- Web Server will parse the URL and will look for the filename in if it finds that file then sends it back to the browser, otherwise sends an error message indicating that you have requested a wrong file.
- Web browser takes response from web server and displays either the received file or error message.

CGI Architecture Diagram



Example CGI Program:

```
print "Content-type:text/html\r\n\r\n"
print '<html>'
print '<head>'
print '<title>Hello Word - First CGI Program</title>'
print '</head>'
print '<body>'
print '<h2>Hello Word! This is my first CGI program</h2>'
print '</body>'
print '</html>'
```

HTTP Header

- All the HTTP header will be in the following form:

  HTTP Field Name: Field Content

For Example

```
Content-type: text/html\r\n\r\n
```

– There are few other important HTTP headers, which you will use frequently in your CGI Programming.

Header	Description
Content-type:	A MIME string defining the format of the file being returned. Example is Content-type:text/html
Expires: Date	It is used by the browser to decide when a page needs to be refreshed. A valid date string is in the format 01 Jan 1998 12:00:00 GMT.
Location: URL	The URL that is returned instead of the URL requested. You can use this field to redirect a request to any file.
Last-modified: Date	The date of last modification of the resource.
Content-length: N	The length, in bytes, of the data being returned. The browser uses this value to report the estimated download time for a file.
Set-Cookie: String	Set the cookie passed through the string

CGI Environment Variables:

– CGI program will have access to the following environment variables.

Variable Name	Description
CONTENT_TYPE	The data type of the content. Used when the client is sending attached content to the server. For example, file upload.
CONTENT_LENGTH	The length of the query information. It is available only for POST requests.
HTTP_COOKIE	Returns the set cookies in the form of key & value pair.

HTTP_USER_AGENT	The User-Agent request-header field contains information about the user agent originating the request. It is name of the web browser.
PATH_INFO	The path for the CGI script.
QUERY_STRING	The URL-encoded information that is sent with GET method request.
REMOTE_ADDR	The IP address of the remote host making the request. This is useful logging or for authentication.
REMOTE_HOST	The fully qualified name of the host making the request. If this information is not available, then REMOTE_ADDR can be used to get IR address.
REQUEST_METHOD	The method used to make the request. The most common methods are GET and POST.
SCRIPT_FILENAME	The full path to the CGI script.
SCRIPT_NAME	The name of the CGI script.
SERVER_NAME	The server's hostname or IP Address

Passing Information using GET Method:

- The GET method sends the encoded user information appended to the page request.
- The page and the encoded information are separated by the ? character as follows: http://www.test.com/cgi-bin/hello.py?key1=value1&key2=value2
- The GET method is the default method to pass information from browser to web server and it produces a long string that appears in your browser's Location:box.
- Never use GET method if you have password or other sensitive information to pass to the server.
- The GET method has size limitation: only 1024 characters can be sent in a request string.
- The GET method sends information using **QUERY_STRING** header

Example:

--------------- hello_get.py ------------------

```
import cgi, cgitb

form = cgi.FieldStorage()

first_name = form.getvalue('first_name')

last_name  = form.getvalue('last_name')

print "Content-type:text/html\r\n\r\n"

print "<html>"

print "<body>"

print "<h2>Hello %s %s</h2>" % (first_name, last_name)

print "</body>"

print "</html>"
```

- **hello_get.py** script to handle input given by web browser. We are going to use cgi module, which makes it very easy to access passed information
- Here is a simple example which passes two values using HTML FORM and submit button.

```
<form action="/cgi-bin/hello_get.py" method="get">

First Name: <input type="text" name="first_name">  <br />

Last Name: <input type="text" name="last_name" />

<input type="submit" value="Submit" />

</form>
```

Field Storage: The FieldStorage instance can be indexed like a Python dictionary, and also supports the standard dictionary methods has_key() and keys().

cgitb(Traceback manager for CGI scripts) : The cgitb module provides a special exception handler for Python scripts

---

Passing Information using POST Method:

- This packages the information in exactly the same way as GET methods, but instead of sending it as a text string after a ? in the URL it sends it as a separate message.
- Parameters cant be seen in url, so recommended to use POST method dealing with sensitive information
- Parameters are not saved in the browser history
- When the post method is used, the browser sends a URL request first and then the data is sent separately.

Example:

---------- hello_get.py ----------

```
import cgi, cgitb

form = cgi.FieldStorage()

first_name = form.getvalue('first_name')

last_name  = form.getvalue('last_name')

print "Content-type:text/html\r\n\r\n"

print "<html>"

print "<body>"

print "<h2>Hello %s %s</h2>" % (first_name, last_name)

print "</body>"

print "</html>"
```

- Here is a simple example which passes two values using HTML FORM and submit button.

```
<form action="/cgi-bin/hello_get.py" method="post">

First Name: <input type="text" name="first_name"><br />

Last Name: <input type="text" name="last_name" />

<input type="submit" value="Submit" />
```

```
</form>
```

Passing Checkbox Data to CGI Program:

- Checkboxes are used when more than one option is required to be selected.
- Here is example HTML code for a form with two checkboxes

```
<form action="/cgi-bin/checkbox.cgi" method="POST"
target="_blank">

<input type="checkbox" name="maths" value="on" /> Maths

<input type="checkbox" name="physics" value="on" /> Physics

<input type="submit" value="Select Subject" />

</form>
```

- Below is **checkbox.cgi** script to handle input given by web browser for checkbox button.

```
------------------- checkbox.cgi ----------------------

import cgi, cgitb

form = cgi.FieldStorage()

if form.getvalue('maths'):

   math_flag = "ON"

else:

   math_flag = "OFF"

if form.getvalue('physics'):

   physics_flag = "ON"

else:

   physics_flag = "OFF"

print "Content-type:text/html\r\n\r\n"
```

```
print "<html>"

print "<body>"

print "<h2> CheckBox Maths is : %s</h2>" % math_flag

print "<h2> CheckBox Physics is : %s</h2>" % physics_flag

print "</body>"

print "</html>"
```

Passing Radio Button Data to CGI Program:

- Passing Radio Button Data to CGI Program
- Here is example HTML code for a form with two radio buttons

```
<form        action="/cgi-bin/radiobutton.py"        method="post"
target="_blank">

<input type="radio" name="subject" value="maths"/> Maths

<input type="radio" name="subject" value="physics"/> Physics

<input type="submit" value="Select Subject" />

</form>
```

- Below is **radiobutton.py** script to handle input given by web browser for radio button

---------- **radiobutton.py** ----------

```
import cgi, cgitb

form = cgi.FieldStorage()

if form.getvalue('subject'):

   subject = form.getvalue('subject')

else:

   subject = "Not set"


print "Content-type:text/html\r\n\r\n"
```

```
print "<html>"

print "<head>"

print "<title>Radio - Fourth CGI Program</title>"

print "</head>"

print "<body>"

print "<h2> Selected Subject is %s</h2>" % subject

print "</body>"

print "</html>"
```

Passing Text Area Data to CGI Program:

- TEXTAREA element is used when multiline text has to be passed to the CGI Program.
- Here is example HTML code for a form with a TEXTAREA box

```
<form action="/cgi-bin/textarea.py" method="post"
target="_blank">

<textarea name="textcontent" cols="40" rows="4">

Type your text here...

</textarea>

<input type="submit" value="Submit" />

</form>
```

- Below is **textarea.cgi** script to handle input given by web browser

```
---------------------- textarea.cgi --------------------------

import cgi, cgitb

form = cgi.FieldStorage()

if form.getvalue('textcontent'):

   text_content = form.getvalue('textcontent')
```

```
else:
   text_content = "Not entered"
print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>";
print "<title>Text Area - Fifth CGI Program</title>"
print "</head>"
print "<body>"
print "<h2> Entered Text Content is %s</h2>" % text_content
print "</body>"
```