Capítulo 2

Python:
Módulo CGI, PyMySQL

Módulo CGI

- Provee el soporte para desarrollar programas CGI en Python
  - Código fuente: https://github.com/python/cpython/blob/3.8/Lib/cgi.py
- Se debe importar como módulo:

```
import cgi
```

- En tiempo de desarrollo, se recomienda incluir:

```
import cgitb
cgitb.enable(display=0, logdir="/path/to/logdir")
```

Módulo CGI

- Para acceder a los datos que nos envían al CGI podemos usar la clase `FieldStorage`
  - Los datos son manejados como un diccionario de Python

```
form = cgi.FieldStorage()
if "name" not in form or "addr" not in form:
    print("<H1>Error</H1>")
    print("Please fill in the name and addr fields.")
    return
print("<p>name:", form["name"].value)
print("<p>addr:", form["addr"].value)
...further form processing here...
```

Módulo CGI

- Si los datos enviados contienen más de un valor para el mismo nombre:
  - Obtendremos una lista al llamar:

    `form[“nombre”]`

    `form.getValue(“nombre”)`
  - Si esperamos una lista de valores para un nombre se sugiere usar:

```
value = form.getlist("username")
usernames = ",".join(value)
```

Módulo CGI

- Si estamos esperando un archivo enviado por el usuario:
  - Se recomienda hacer un test con el atributo "`file`" del dato recibido en el formulario:

```
fileitem = form["userfile"]
if fileitem.file:
    # It's an uploaded file; count lines
    linecount = 0
    while True:
        line = fileitem.file.readline()
        if not line: break
        linecount = linecount + 1
```

Módulo CGI

- Para el caso de archivos **no se recomienda** llamar getValue()
  - cargará en memoria el contenido completo como un arreglo de bytes
- Pueden ocurrir errores en la carga de archivo:
  - Usuario podría presionar botón “back” de su navegador
  - Se recomienda revisar el atributo “`done`”, en caso de error tendrá valor -1
- También se pueden recibir múltiples archivos con el mismo nombre
  - Se debe iterar sobre los elementos

Módulo CGI

- Imaginemos que tenemos estos input de un formulario:

```
<input type="checkbox" name="item" value="1" />
<input type="checkbox" name="item" value="2" />
```

- Podríamos procesar estos datos como:

```
item = form.getvalue("item")
if isinstance(item, list):
    # The user is requesting more than one item.
else:
    # The user is requesting only one item.
```

Módulo CGI

- Como desarrolladores debemos:
  - Nunca esperar que el cliente informará datos válidos
  - Un usuario podría agregar muchos pares `nombre=valor` con nombres repetidos
  - Este código generará una excepción `AttributeError`

```
user = form.getvalue("user").upper()
```

  - `getvalue()` retornará una lista y el método "`upper()`" no es válido para una lista

Módulo CGI

- Mejor forma: usar los métodos `getFirst()` y `getList()`

```
import cgi
form = cgi.FieldStorage()
user = form.getfirst("user", "").upper()    # This way it's safe.
for item in form.getlist("item"):
    do_something(item)
```

Módulo CGI

- Consideración de seguridad:
  - Si ejecutará un programa externo (`os.system()`), nunca entregue como argumento la información recibida del cliente
  - Los datos que recibimos podrían no venir desde nuestros formularios
  - Si necesitamos pasar argumentos a comandos de sistema desde los parámetros recibidos
    - Debemos asegurar que sean solo caracteres alfanuméricos, /, _, .

Módulo CGI

- Evitar inyección de código
  - En caso de generar mensajes al usuario a partir de información que el usuario envía
  - Usar módulo `html` con función `escape`

`html.`**escape**(*s*, *quote=True*)

Convert the characters `&`, `<` and `>` in string *s* to HTML-safe sequences. Use this if you need to display text that might contain such characters in HTML. If the optional flag *quote* is true, the characters (`"`) and (`'`) are also translated; this helps for inclusion in an HTML attribute value delimited by quotes, as in `<a href="...">`.

New in version 3.2.

https://docs.python.org/3/library/html.html

PyMySQL

- https://pypi.org/project/PyMySQL/
- Biblioteca para comunicar Python con MySQL
  - Implementa PEP 249

    https://www.python.org/dev/peps/pep-0249/
  - Documentación

    https://pymysql.readthedocs.io/en/latest/
- Permite las funciones básicas
  - Conexión a base de datos
  - Ejecución de sentencias SQL
    - Obtener, insertar, actualizar, eliminar
  - Manejo de errores

PyMySQL

- Se consideran dos objetos principales:
- `Connection`: representación de un *socket* de comunicación con el servidor MySQL
  - Establece la conexión con la base de datos
  - Acepta varios argumentos:
    - Host
    - User
    - Password
    - Database
    - Port
    - Charset
    - init_command
    - autocommit

https://pymysql.readthedocs.io/en/latest/modules/connections.html

PyMySQL

- `Cursor`: se utiliza para interactuar con la base de datos
  - Se debe obtener desde el objeto `Connection`
- Provee funciones para ejecutar sentencias SQL
  - `execute(query, args=None)`
    - Query: sentencia SQL
    - Args: opcional, parámetros que se usarán en la sentencia SQL
    - Retorna un número entero que indica la cantidad de filas que fueron afectadas por la consulta

PyMySQL

- Conexión a base de datos

```
import pymysql

# conexion a base de datos
conn = pymysql.connect(
    db='ejemplo',
    user='cc5002',
    passwd='cc5002',
    host='localhost',
    charset='utf8')
c = conn.cursor()
```

PyMySQL

- Insertar información

```
datos = cgi.FieldStorage()
if len(datos) > 0:
    # insertar
    sql = "INSERT INTO persona (nombre, apellido) VALUES (%s, %s)"
    try:
        resultado = c.execute(sql, (datos['nombre'].value, datos['apellido'].value))
        conn.commit()
        if resultado == 1:
            mensaje = "insertada nueva persona: " +\
                      datos['nombre'].value + " " +\
                      datos['apellido'].value
        else:
            mensaje = "error, no se insertó persona en la base de datos"
    except pymysql.Error as e:
        mensaje = "Error con base de datos: {0} {1} ".format(e.args[0], e.args[1])
```

PyMySQL

- Obtener información

```
# obtenemos las personas desde la base de datos
sql = "SELECT id, nombre, apellido FROM persona";
c.execute(sql)
conn.commit()
personas = c.fetchall()

if len(personas) > 0:
    print("<table border=1>")
    print("<tr><td><b>ID</b></td>\
               <td><b>Nombre</b></td>\
               <td><b>Apellido</b></td></tr>")
    for p in personas:
        print("<tr><td>", p[0], "</td><td>", p[1], "</td><td>", p[2], "</td></tr>")

    print("</table>")
```