CS1116/CS5018

Web Development 2

Dr Derek Bridge

School of Computer Science & Information Technology

University College Cork

Spot the difference: two versions of `cursor.execute()`

- A version that takes a single parameter — a string of SQL:

```
cursor.execute("""SELECT gig_date FROM gigs
                  WHERE band = '%s'""" % (bandname))
```

- A version that takes two parameters — a string of SQL and a tuple:

```
cursor.execute("""SELECT gig_date FROM gigs
                  WHERE band = %s""", (bandname))
```

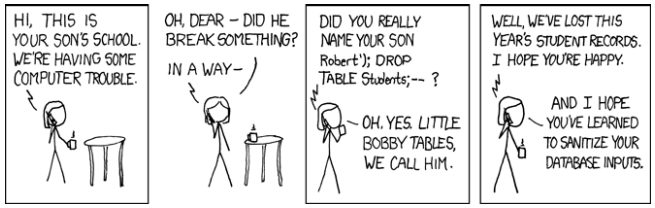
Always use the second version

- In the second version, use %s for all placeholders — it is smart about how it replaces placeholders by data
- Assuming `bandname` contains `'Belated Tonic'` and `bandnumber` contains 2, what happens here?

```
from datetime import date
today = date.today() # today's date

cursor.execute("""SELECT gig_date FROM gigs
                  WHERE band = %s""", (bandname))

cursor.execute("""SELECT band FROM gigs
                   WHERE num = %s""", (bandnumber))

cursor.execute("""INSERT INTO gigs (band, gig_date)
                  VALUES (%s, %s)""", (bandname, today))
```

- As well as being convenient, this avoids **SQL injection attacks**

xkcd's *Exploits of a Mom*



http://xkcd.com/327/

A successful attack

- Into the form, enter:
  `Belated Tonic'; DROP TABLE gigs; --`
  (with a space at the end)
- If you are using the first version of `cursor.execute()`, what do we get?

```
cursor.execute("""SELECT gig_date FROM gigs
                  WHERE band = '%s'""" % (bandname))
```

A thwarted attack

- The second version of `cursor.execute()` **sanitizes database input**:
  - It **escapes** characters in the user's data that have a special meaning in SQL
- What do we get?

```
cursor.execute("""SELECT gig_date FROM gigs
                  WHERE band = %s""", (bandname))
```

Another database program

```
#!/usr/local/bin/python3

from cgitb import enable
enable()

from cgi import FieldStorage
from html import escape
import pymysql as db

print('Content-Type: text/html')
print()

form_data = FieldStorage()
bandname = ''
when = ''
result = ''
if len(form_data) != 0:
    try:
        bandname = escape(form_data.getfirst('bandname'))
        when = escape(form_data.getfirst('when'))
        connection = db.connect('localhost', 'userid', 'password', 'database_name')
        cursor = connection.cursor(db.cursors.DictCursor)
        cursor.execute("""INSERT INTO gigs (band, gig_date)
                          VALUES (%s, %s)""", (bandname, when))
        connection.commit()
        result = '<p>Succesfully inserted!</p>'
        cursor.close()
        connection.close()
    except db.Error:
        result = '<p>Sorry! We are experiencing problems at the moment. Please call back later.</p>

print("""
    <!DOCTYPE html>
    <html lang="en">
        <head>
            <meta charset="utf-8" />
            <title>Insert gigs</title>
        </head>
        <body>
            <form action="new_gigs.py" method="post">
                <label for="bandname">Band: </label>
                <input type="text" name="bandname" value="%s" size="50" maxlength="50" id="bandname
                <label for="when">Date: </label>
                <input type="date" name="when" value="%s" id="when" />
                <input type="submit" value="Insert" />
            </form>
            %s
        </body>
    </html>""" % (bandname, when, result))
```

HTTP requests

- There are two methods the browser can use to pass form data to the server: get and post.
  - If `method="get"`, the data is added to the end of the URL after a question mark, e.g.:

```
GET response.py?first=Hugh&surname=Jeegoh
```

  - If `method="post"`, the data is included in the HTTP request body, not the header:

```
POST response.py
first=Hugh&surname=Jeegoh
```

The get method versus the post method

- When to use `method="get"`,
  - for a short form with few fields
- When to use `method="post"`,
  - for a longer, more complex form, where encoding the form data in the URL would result in a URL that is too long

The get method versus the post method

- When to use `method="get"`,
  - to allow the URL (with the form data) to be bookmarked, or used elsewhere
- When to use `method="post"`,
  - for security, because encryption and other encodings of the data is possible for `method="post"`

The get method versus the post method

- When to use `method="get"`,
  - where the outcome wouldn't differ if you issued the same request more than once
- When to use `method="post"`,
  - for requests that make a change in the server (e.g. update a database)