## SENG 462 Tutorial #9

Chris Pearson pearson@csc.uvic.ca



# Need Help?

Schedule time with me for tomorrow



### **B203 Servers**

- Why are people using all of them?
- Do we need to assign certain machines to certain groups?
- Do we a scheduling mechanism for who is using the servers when?

## Phone & Server Communication

- Why is everybody using HTTP?
- Sockets are simple...

```
import socket
# Create the socket
s = socket.socket(socket.AF INET, socket.SOCK STREAM)
s.bind(('', 80)) # Bind the socket to a port
                        # Allow 1 waiting connection
s.listen(1)
conn, addr = s.accept() # Wait for a connection
# Could spawn a thread for the connection here,
# then return to listening for new connections
while 1:
                        # Echo the data back to the sender
    data = conn.recv(1024)
    if not data: break
    conn.send(data)
                        # Close the connection
conn.close()
```



## Send A Command

- Use Python objects to send a command
  - 1. Set up a socket listen thread on a server
  - 2. The phone creates an object that encapsulates a command
  - 3. The phone uses Pickle to serialize the object
    - a string = pickle.dumps(my object)
  - 4. The phone sends the string to the server
    - a\_socket.write(a\_string)
  - 5. The server unpickles the object
    - my\_object = pickle.load(a\_string)
  - 6. The server does its processing magic



## Send Back The Results

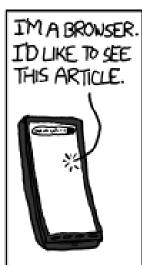
- Use Python objects to send back results
  - 1. Server creates an object the encapsulates the results of the command
  - 2. The server uses Pickle to serialize the results object
    - a\_string = pickle.dumps(results\_object)
  - 3. The server sends the string to the phone
    - a socket.write(a string)
  - 4. The phone unpickles the object
    - results\_object = pickle.load(a\_string)
  - 5. The phone uses the results directly as with any other Python object



# Database Connection Pooling

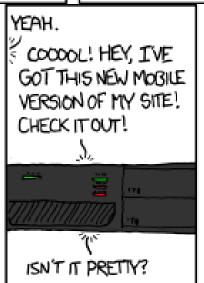
- Problem:
  - Need to keep database connections open
  - Web servers are stateless
- What do you do?



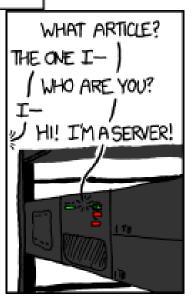




Source: xkcd.com/869







# **Database Connection Pooling**

- The idea:
  - Run an application as your transaction server.
  - Make several connections to the database
  - Hand those connections to needing threads
  - Keep track of whether connections are in use or not
- This can mean a 10x performance increase

# Simple Data Storage

- ▶ e32dbm
- Stores data in (key, value) pairs, similar to Python dictionaries
- All data is stored as Unicode strings
- ▶ See Section 5.4, *S60 Module Reference 2.0*



## e32dbm - Setup

```
import e32dbm
```

```
DBPATHNAME = u"c:\\Data\\Others\\AccelGrapher"
DBFILENAME = DBPATHNAME + u"\\agraph2.db"

# create the directory for the database, if it doesn't already exist
if not os.path.exists(DBPATHNAME):
    os.makedirs(DBPATHNAME)
```



### e32dbm - Load Values

```
# try to load values from the database
try:
 db = e32dbm.open(DBFILENAME, "r")
 currentTab = int(db[u"currentTab"])
 thicknessLines = int(db[u"thicknessLines"])
 thicknessDots = int(db[u"thicknessDots"])
 db.close()
# use default values if the database or one of its values
 isn't available
except:
 currentTab = 0
 thickness = 5
 thicknessDots = 5
```



### e32dbm - Save Values

```
Write the state variables out to the database
# c - opens the database for reading and writing (and
  creates a new database if the file does not exist)
# f - the database is not updated on disk until you close
  it or force it to be written to disk by using a special
  function.
try:
  db = e32dbm.open(DBFILENAME, "nf")
  db[u"currentTab"] = str(currentTab)
  db[u"thicknessLines"] = str(thicknessLines)
  db[u"thicknessDots"] = str(thicknessDots)
 db.close()
except:
 pass
```

