

A singleton class is handy when you always only want one instance of an object returned. This can be used for example in a web request where keep calling the db.connection without worrying about new connections been created. Lets look at an example:

Here is a inheritable Singleton class:

```
class Singleton(object):

    _instance = None

    def __new__(cls, *args, **kwargs):

        if not cls._instance:

            cls._instance = super(Singleton, cls).__new__(cls, *args, **kwargs)

        return cls._instance
```

We can now make a crude Db class:

```
import MySQLdb

from Contrado.Core.Singleton import Singleton
from Contrado.Core.Config import *

class Db(Singleton):

    """Singleton implementation for accessing database connections"""

    def __init__(self):

        self.__connection = None

    def __connect(self):

        self.__connection = MySQLdb.connect(user=DB_USR, passwd=DB_PASS, db=DB_NAME)

    def getConnection(self):

        if not self.__connection: self.__connect()

        return self.__connection

    connection = property(getConnection)
```

Now we can test it using the python interpreter notice the same object id that is generated:

```
IDLE 1.2
>>> from Contrado.Core.Db import Db
>>> x = Db()
>>> y = Db()
>>> z = Db()
>>> id(x)
13987088
>>> id(y)
13987088
>>> id(z)
13987088
>>>
```