```
# Author: JT Hamrick
# design based off of http://chrisdianamedia.com/simplestore/
import os.path
import tornado.ioloop
import tornado.web
from tornado.options import define, options
import tornado.template
import MySQLdb
import uuid
import urllib
import re
import magic

# define values for mysql connection
define("port", default=8895, help="run on the given port", type=int)
define("mysql_host", default="127.0.0.1", help="database host")
define("mysql_port", default=3306, help="database port", type=int)
define("mysql_database", default="group8", help="database name")
define("mysql_user", default="group8", help="database user")
define("mysql_password", default="yUIAYjfdQlgoMf2eb5gsqY745ZFKpZZE", help="database password")

__UPLOADS__ = "static/uploads/"


class Application(tornado.web.Application):
    def __init__(self):
        handlers = [
            (r"/", HomeHandler),
            (r"/details/([^/]+)", DetailsHandler),
            (r"/cart", CartHandler),
            (r"/product/add", AddToCartHandler),
            (r"/product/remove/([^/]+)", RemoveFromCartHandler),
            (r"/cart/empty", EmptyCartHandler),
            (r"/upload", UploadHandler),
            (r"/userform", UserformHandler),
            (r"/welcome/([^/]+)", WelcomeHandler),
            (r"/directory/([^/]+)", DirectoryTraversalHandler)
        ]
        settings = dict(
            template_path=os.path.join(os.path.dirname(__file__), "templates"),
            static_path=os.path.join(os.path.dirname(__file__), "static"),
            ui_modules={"Small": SmallModule},
            xsrf_cookies=False,
            debug=True,
            cookie_secret="2Xs2dc.y2wqZVB,qRrnyoZuWbUTnjRBG4&uxaMYtM&r%KnpL7e"
        )
        super(Application, self).__init__(handlers, **settings)
        # Have one global connection to the store DB across all handlers
        self.myDB = MySQLdb.connect(host=options.mysql_host,
                                    port=options.mysql_port,
                                    db=options.mysql_database,
                                    user=options.mysql_user,
                                    passwd=options.mysql_password)


class BaseHandler(tornado.web.RequestHandler):
    @property
    def db(self):
        return self.application.myDB

    # if there is no cookie for the current user generate one
    def get_current_user(self):
        if not self.get_cookie("webstore_cookie"):
            self.set_cookie("webstore_cookie", str(uuid.uuid4()))


class HomeHandler(BaseHandler):
    def get(self):
        # get all products in the database for the store's main page
        temp = []
        c = self.db.cursor()
        c.execute("SELECT * FROM products")
        products = c.fetchall()
        # add urlencoded string to tuple for product image link
        for k, v in enumerate(products):
            temp.append(products[k] + (urllib.parse.quote_plus(products[k][2]),))

        authorized = self.get_cookie("loggedin")
        self.render("home.html", products=tuple(temp), auth=authorized)


class DetailsHandler(BaseHandler):
    def get(self, slug):
        # get the selected product from the database
        temp = []
        # remove non numerical characters from slug
```

```
        item_number = re.findall(r'\d+', slug)
        c = self.db.cursor()
        c.execute("SELECT * \
                   FROM products p \
                   LEFT JOIN (SELECT `option`, \
                                     GROUP_CONCAT(`value`) AS `value`, \
                                     product_id \
                              FROM `product_options` \
                              WHERE `product_id` = " + item_number[0] + " \
                              GROUP BY `option`) AS o ON o.product_id = p.id \
                   WHERE p.id = " + item_number[0])
        product = c.fetchall()
        # add urlencoded string to tuple for product image link
        quoted_url = urllib.parse.quote_plus(urllib.parse.quote_plus(product[0][2]))
        temp.append(product[0] + (quoted_url,))

        authorized = self.get_cookie("loggedin")
        self.render("details.html",
                    product=tuple(temp),
                    sku=item_number[0],
                    auth=authorized)


class CartHandler(BaseHandler):
    def get(self):
        # get the current user's cookie
        cookie = self.get_cookie("webstore_cookie")
        # get the current user's cart based on their cookie
        c = self.db.cursor()
        c.execute("SELECT c.item, \
                          p.price, \
                          p.name, \
                          COUNT(*) AS quantity, \
                          SUM(p.price) AS subtotal, \
                          `options`, \
                          GROUP_CONCAT(c.id) AS `id` \
                   FROM cart c \
                   INNER JOIN products p on p.id = c.item \
                   WHERE c.user_cookie = '" + cookie + "' \
                   GROUP BY c.item, c.options")
        products = c.fetchall()
        # calculate total and tax values for cart
        total = float(sum([x[4] for x in products]))
        count = sum([x[3] for x in products])
        tax = float("{0:.2f}".format(total * 0.08517))
        shipping = 5.27

        if not total:
            shipping = 0.00

        authorized = self.get_cookie("loggedin")
        self.render("cart.html",
                    products=products,
                    total=total,
                    count=count,
                    shipping=shipping,
                    tax=tax,
                    auth=authorized)


class AddToCartHandler(BaseHandler):
    def post(self):
        # get the product information from the details page
        id = self.get_argument("product", None)
        cookie = self.get_cookie("webstore_cookie")
        product_options = ",".join(self.get_arguments("option"))
        # add the product to the user's cart
        c = self.db.cursor()
        c.execute("INSERT INTO cart (id, user_cookie, item, options) \
                   VALUES (0, '"+cookie+"', "+id+", '"+product_options+"')")
        self.application.myDB.commit()
        self.redirect("/cart")


class RemoveFromCartHandler(BaseHandler):
    def get(self, slug):
        # get the current user's cookie
        cookie = self.get_cookie("webstore_cookie")
        # use that cookie to remove selected item from the user's cart
        c = self.db.cursor()
        c.execute("DELETE FROM cart \
                   WHERE user_cookie = '" + cookie + "'  \
                       AND id IN(" + slug + ")")
        self.application.myDB.commit()
        self.redirect("/cart")
```

```
class EmptyCartHandler(BaseHandler):
    def get(self):
        # get the current user's cookie
        cookie = self.get_cookie("webstore_cookie")
        # use that cookie to remove all items from user's cart
        c = self.db.cursor()
        c.execute("DELETE FROM cart WHERE user_cookie = {0}".format(cookie))
        self.application.myDB.commit()
        self.redirect("/cart")


class WelcomeHandler(BaseHandler):
    def get(self, name):
        TEMPLATE = open("templates/welcome.html").read()
               #template_data = TEMPLATE.replace("FOO" , name)
        t = tornado.template.Template(TEMPLATE)
        self.write(t.generate(name=name))

class UserformHandler(tornado.web.RequestHandler):
    def get(self):
        self.render("fileuploadform.html")


class UploadHandler(tornado.web.RequestHandler):
    def post(self):
        fileinfo = self.request.files['filearg'][0]
        fname = fileinfo['filename']
        # extn = os.path.splitext(fname)[1]
        # cname = str(uuid.uuid4()) + extn
        fh = open(__UPLOADS__ + fname, 'w')
        fh.write(fileinfo['body'])
        self.finish(fname + " is uploaded!! Check %s folder" % __UPLOADS__)
        # self.write(fileinfo)


class DirectoryTraversalHandler(BaseHandler):
    def get(self, slug):
        mime = magic.Magic(mime=True)
        filename = urllib.parse.unquote(urllib.parse.unquote(slug))
       ##############################################
        expDir = '/directory/'
        path = os.path.abspath(os.path.join(expDir, filename))
        if  path.startswith(expDir):
              raise tornado.web.HTTPError(400)
       ##############################################
        mime_type = mime.from_file(filename)
        self.set_header('Content-Type', mime_type)
        with open(filename, 'rb') as f:
            self.write(f.read())


class SmallModule(tornado.web.UIModule):
    def render(self, item):
        return self.render_string("modules/small.html", item=item)


def main():
    http_server = tornado.httpserver.HTTPServer(Application())
    http_server.listen(options.port)
    print(f"Web server started. In your browser, go to 10.10.0.13:{options.port}")
    tornado.ioloop.IOLoop.current().start()


if __name__ == "__main__":
    main()
```