R · I · T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences and Technologies

4002-XXX
Software Development on Linux Systems
Lab 10 – Apache and Django

Name: _____ **Section:** _____

In this lab you will be installing and configuring Apache for a web services. This will be a Linux Apache MySQL Python setup. For this lab we will be using the Django framework. Though we will be using Django, it is good to note that the same results could be achieved using other frameworks.

Note: This lab may be completed using Ubuntu or Fedora

Activity – Installing Apache

Apache is an extremely popular open source web server software. Apache is setup using many configuration files.

These are the main configuration files:
- apache2.conf which is the main Apache file containing all of the global settings.
- httpd.conf which is one of the main Apache configuration file and contains all of the global user specific settings for Apache.
- ports.conf which is a configuration file that controls which ports Apache is using/listening to

*The above was derived from https://help.ubuntu.com/11.04/serverguide/C/httpd.html

1) Using the installation method of your choice install Apache
These are the package names:

Ubuntu: libapache2-mod-python
apache2
python-django
python-mysqldb

Fedora: mod_python
httpd
Django
My**SQL**-python

2) Now that Apache is installed you need to start it
Ubuntu:
In terminal type “sudo /etc/init.d/apache2 restart”

Fedora:
In terminal type “su -”
“/etc/init.d/apache2 restart”

3) cd to /etc/apache2/sites-available

4) ls the files in /etc/apache2/sites-available
These are the configuration files for each individual site. Currently, the default file points to a default testing web directory for Apache. When this is enabled, that directory will be available in a browser.

5) Make a copy of the config file named “default” with the name “new-site” in the same folder

6) To enable the site type “a2ensite default”. You will need to use sudo on Ubuntu or su on Fedora. (a2ensite stands for apache 2 enable site)

7) After you enable the site, you will have to reload apache

As root (sudo or su -) run the following to reload apache

/etc/init.d/apache2 reload

8) Open a web browser on the local machine and go to http://localhost
It should say “It works!” on the page

9) Now we will load the site configuration you just made as “new-site”
As root type “a2dissite default”

(a2dissite stands for apache 2 disable site)

10) Reload apache with “/etc/init.d/apache2 reload”

11) Now in a web browser go to “http://localhost
It should say “Not Found”

12) Now enable your site “new-site”
As root type “a2ensite new-site”

13) Reload apache with “etc/init.d/apache2 reload”

14) Now in a web browser go to “http://localhost
It should say “It Works!”

Instructor/TA Sign-Off _____

Activity – Installing Apache with SSL

To use Apache with SSL you need a certificate. This is a multiple step process. You will need a private and public key pair, a certificate signing request (CSR) and a certificate. The CSR is the file that you would send to a web Certificate Authority (CA).

In this lab, we will be self-signing, in which case we will not send the CSR to a CA. However, this also means that browsers will indicate this is a self-signed certificate that the user will have to verify.

1) Openssl is a library for using transport layer security. This is what is used to generate a secure certificate for use with websites. Open SSL has a variety of parameters for encryption.

a) In your home folder in terminal type “openssl genrsa -des3 -out mySite.key 2048”

b) Type in a password **YOU MUST REMEMBER!**

[**Breakdown of the command**==========]
genrsa means to generate public/private RSA key pair.
Des3 is the flag for using Triple Des encryption.
Out is the flag for signifying the file name you want to use for the key.
mySite.key is the filename you want the key to be called
2048 is the number of bits to use for the key; the more the more secure.
[==========]

This will output an RSA key called mySite.key in your home folder

2) Now that you have a private key, you will need to do the CSR. This is the file that you would send to a certificate authority for a real site. We will not be sending this file to a CA.

a) In your home directory type "openssl req -new -key mySite.key -out mySite.csr"

[**Breakdown of the command**==========]
req means to request the CSR
new means a new request
key means you are providing the public key
out is the flag for signifying the file name you want for the CSR
[==========]

b) Enter the country code "US"

c) Enter the state name "New York"

d) Enter the locality "Rochester"

e) Enter the organization "MyCompany"

d) Enter the organizational unit name "**.**" [that's a period]

e) Enter the common name "localhost"

f) Enter your email address

g) Enter a **DIFFERENT PASSWORD YOU MUST REMEMBER**

h) Leave optional company name blank and hit enter

You should now have a file called mySite.csr

3) Now that you have the CSR, you will create the certificate

a) In your home directory type
"openssl x509 -req -days 365 -in mySite.csr -signkey mySite.key -out mySite.pem"

b) Enter the password for the key from step 1 of this activity

You should now have a certificate named mySite.pem in your home folder

[**Breakdown of the command**==========]

x509 is a flag for the certificate data type x509
req is the flag for requesting a new x509 certificate
days is the flag for the number of days the certificate will last
365 is the value for the days flag so that the certificate lasts 365 days
in is the flag for the csr file
mySite.csr is the csr file you created in the previous step
signkey is the flag to give your public key
mySite.key is the key generated in step 1 of this activity
out is the flag for the filename you want the certificate to be called
mySite.pem is the provided filename for the out flag to name the certificate

[==========]

4) Now we need to install the new certificate and encryption key

a) Copy the mySite.pem file to folder /etc/ssl/certs/

b) Copy the mySite.key file folder /etc/ssl/private/

5) Now enable SSL security on Apache with the command

“a2enmod ssl”

6) You now need to create a file for the ssl site.

a) cd to /etc/apache2/sites-available/

b) Make a copy of the file “default-ssl” called “mySite-ssl” in the same folder

c) Open the file mySite-ssl in an editor (suggested vi [vim] or gedit)

d) Find the lines

"SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem"
"SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key"

and replace them with

"SSLCertificateFile /etc/ssl/certs/mySite.pem"
"SSLCertificateKeyFile /etc/ssl/private/mySite.key"

e) Save the file
(Hint: If it doesn't save or you are unable to edit it, you did not open the file as root")

7) Using a2ensite from the previous activity, enable "mySite-ssl"

8) Now you must **restart apache NOT reload**

a) With root permissions run "/etc/init.d/apache2 restart"

b) Type in the password for your **key** in step 1 of this activity

9) Testing the https connection

a) Open a web browser and go to https://localhost

It should indicate that the connection is untrusted and that it could not verity the identity of the server. This is because none of the certificate authorities have your CSR so they can't verify it.

b) Choose to add the exception / proceed

c) Now when you go to https://localhost you should see "It Works!"

Instructor/TA Sign-off_____

R · I · T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences and Technologies

Activity – Apache & Django

In this activity you will make a MySQL user for Django and Apache

1) Download the messageboard files zip from mycourses

2) Open terminal and login to mysql

3) Create a MySQL user called “django” with the password “dblab”
You may change the password if you would like

4) Now create a database called django

5) Grant the django MySQL user access to the django database

“ GRANT SELECT, ALTER, INSERT, UPDATE, INDEX DELETE, CREATE, DROP
ON django**.***
TO django@localhost
IDENTIFIED BY 'dblab' “

Note: Replace the password 'dblab' if you changed the django user password

6) Close the mysql terminal

7) Create a directory called myServer in your home directory.

8) Now cd to that directory

9) We will create a new django project

“ django-admin startproject myProject “

10) Now cd to the myProject directory. There should be several files there now. These are django files.

11) Open the new file “settings.py”

-Find the lines for the database (usually around line 12)
-Modify the line

'ENGINE' : 'django.db.backends.'

to

'ENGINE' : 'django.db.backends.mysql'

-In the single quotes fill in the database name, user and password. If you chose a different password, then use that one instead

'NAME' : 'django'
'USER' : 'django'
'PASSWORD' : 'dblab'

12) Sync your project with the database with

“ ./manage.py syncdb “

-When it asks for a superuser type 'yes'
-For the username make it ' django_super '
-Type in your email
-Give the superuser the password ' dblab ' or one of your choice

If this fails, there is a mistake in the settings.py configuration or the mysql django user permissions. You may want to login to mysql as the django user and ensure that they can see the django database. If they can, ensure that the settings.py file has mysql in the engine line, has properly spelled fields and has the needed single quotes.

13) Login to mysql and use the django database. Look at the tables created for you. These are the tables django will use for authorization.

14) Now exit the mysql command line

15) Start the server with " ./manage.py runserver "

When the server starts, it will mention the address it is running on.
Go to that address in a web browser. Most likely this will be 127.0.0.1:8000
It should say "It worked!"

Instructor/TA Sign-off_____

16) Stop the manage.py process

17) Unzip the messageboards zip file in your downloads folder

18) Now we need to make a new app with

" ./manage.py startapp messageboards "

19) Copy the media, messageboards and templates folders to the myProject folder. This will overwrite the messageboards folder.

The "media" folder is where all of the css files and other static files are located

The "templates" folder is where all of the html page templates are located

The "messageboards" folder is where all of the configuration files and views files for the site are located.

20) Copy the urls.py file into the myProject folder overwriting the current one. The urls file is a file that contains regular expressions for requested urls and forwards them to the appropriate application.

21) Open the settings.py file again and find the INSTALLED_APPS section. We need to add the new messageboards folder to the apps. Add 'messageboards' to the list.

Note: Be sure to include the comma at the end of the last uncommented line before if there is not one.

```
INSTALLED_APPS(
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.sites',
 'django.contrib.messages',
 'django.contrib.staticfiles' ,
 #'django.contrib.admin',
 #'django.contrib.admindocs',
 'messageboards'
)
```

22) Now we need to setup the media files path so we can use static files (css, javascript, images, etc)

In settings.py find the line **MEDIA_ROOT = ' '**

and replace it with the full path to the media folder you added into the myProject folder

MEDIA_ROOT = '/home/*Lastname*/myServer/myProject/media/'

23) Now make the media url. This is the path that will be displayed to browsers hiding the real path to static files (css, javascript, images, etc)

In settings.py find the line **MEDIA_URL = ' '**

and replace it with the path you would like to be seen in browser strings instead.

MEDIA_ROOT = '/uploaded/'

24) Now we need to add the html templates.

    In settings.py find the line **TEMPLATE_DIRS = (**

    Add **'templates'** to the list

    ```
    TEMPLATE_DIRS = (
            'templates',
    )
    ```

25) Now save the settings file and sync the database

    ./manage.py syncdb

26) Start the server

    ./manage.py runserver

    Note: Optionally, you may start the server with an IP address and port pair.
    Example: ./manage.py runserver 192.168.1.100:8000

27) Open the address in a browser now http://127.0.0.1:8000 and you should see a login page. This was loaded from the templates folder.

    Note: If you did not, there was likely a typo in the settings file or the database was not synced.

28) Click the register button on the login page and make an account. This will be stored in the MySQL database.

29) Now login with your new account. You will see you can create thread and post messages, but you are unable to use the logout button. This is because the logout button has not been coded yet.

30) Stop the server process

Instructor/TA Sign-off_____

31) Open in the messageboards folder within the myProject folder, open the urls.py file. This is NOT the urls.py file that is located in the myProject folder.

    When Django receives a request, it is sent to the urls.py file. This file checks requested url against a list of regular expressions. When it finds one, it sends it to the location specified. In this case, the urls file sends the requests to the messageboard urls file in the messageboards folder.

    The messageboards url file also checks the requested url against its own regular expressions, then calls the intended method in the views.py file.

    In this file you will see several regular expressions and the methods they call.

    You will see the line
    (r'^create/$', 'create'),

    This means that any urls that end with “create/” will call the method 'create' in the views.py file.

    We need to make one for logout. **Above** the line (r'^create/$', 'create') , create a regular expression that ends with “logout/” and calls a method called logout.

    MAKE SURE YOU END THIS LINE WITH A COMMA. REGULAR EXPRESSIONS ARE CHECKED IN ORDER. IF YOUR EXPRESSION COMES AFTER THE LINE WITH .* IT WILL NEVER RUN BECAUSE .* MATCHES ANY URL AND WILL COME FIRST.

32) Save the urls.py file. Now any requests for a logout url will call the logout method, which does not exist yet. Open the views.py file.

33) In views.py, under the login method, create a method called logout that accepts a request. Remember that python is space sensitive so use tabs.

34) In this method check if a user is logged in with

    request.user.is_authenticated()

    Else, send redirect them to the home page with

    return HttpResponseRedirect('/boards/')

35) Inside of the authentication if statement, log the user out with

    auth.logout(request)

    then do an Http Response Redirect to the home page

36) Save the views.py file and start the server again

37) Now when you login, you should also be able to logout.

    If this does not work, Django should give you an error page describing the error and potentially giving a line number.

38) Finally, we will look at the Django templates. In the templates folder, open the file list.html

    In Django variables in pages are surrounded by {{ while functions are surrounded by {%. For example, if a variable were passed to the page called "pageNum", the variable would be put in the template as

    {{ pageNum }}

    Find the line in the div with ID "header" that says

    Welcome

    A variable has been passed to the list page called ***name***, which is the username of the current logged in user.

    After the word welcome, add the variable for name. There are other example of this in the other html templates.

39) Save the list.html page and reload the home page in a browser. This is the first page after you login. It should now say

    Welcome *username*

Instructor/TA Sign-off_____

R · I · T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences and Technologies

Activity – Apache & Django II

In this activity, you will have two choices. You may add a feature to your Django web page or you may deploy Django to Apache with SSL.

Option 1. Add a new feature or page that requires its own function call in the messageboards urls.py file and a function in views.py.

This does not need to be complex. You may just add a new button, form or page.

- You may copy any of the template files to use as a base.
- You also may modify the existing template files.
- You may also use javascript or css
- You may add new database fields to the models.py file, **but** we have not covered this

NOTE: If you make new pages or make any changes to the models.py file, you must sync the database again. You may want to sync the database before starting the server anyway.

./manage syncdb

Option 2. Deploy your Django project to Apache and set it up to be used with your SSL certificate for Https. This is the harder of the two options.

https://docs.djangoproject.com/en/dev/howto/deployment/

Instructor/TA Sign-off_____