**Session | Database Session in Django 5**

Video56

code file 47

by default djago stores the data of sennsion in database by default

for sessions in settings.py:

INSTALLED\_APPS = [

    'django.contrib.sessions',

]

This is necessary in the settings.py file for sessions.  
We also need to add this middleware in settings.py:

MIDDLEWARE = [

    'django.contrib.sessions.middleware.SessionMiddleware',

]

If we want to use sessions, we must run the following commands:

Python manage.py makemigrations

Python manage.py migrate

Than:

def setsession(request):

    request.session['fname']= 'talha'

request.session.set\_expiry(10)# tset session for 10 second

    return render(request, 'student/setsession.html')

#here we can set session by default its default time 2 weeks

#and here we use get sessions

def getsession(request):

    first\_name = request.session['fname']

#  first\_name = request.session['fname']

#  first\_name = request.session.get('fname')

#  last\_name = request.session.get('lname')

#  first\_name = request.session.get('fname', 'Guest')

    return render(request, 'student/getsession.html',{'first\_name':first\_name})

In db.sqlite3 in Django sessions, the session\_data increases whenever we set or get a session.

Deletesession:

def delsession(request):

    if 'fname' in request.session:  # use  if condition beucause if session not exist it shows error

        del request.session['fname']

    return render(request, 'student/delsession.html')

After deleting through this, we noticed that in db.sqlite3 our data is still there and also in the browser.  
So, how can we flush it? Here we can see that:

Flush session

def flushsession(request):

    request.session.flush() #We use this when we want to ensure that the previous session data cannot be accessed by the user

    # use case : we use this when user logout or other like if user logout so we use this to flush all their session data

    return render(request, 'student/flushsession.html')

Through this, the session data is deleted from both the browser and the db.sqlite3 database. It flushes all the data from the browser and the database.

Now, let’s look at some functions of sessions

sessionmethodsinview

def sessionmethodsinview(request):

    key = request.session.keys() #We use this to access the key that we set.

    print(key)

Using this, we get in the terminal:  
dict\_keys(['fname', 'lname'])

This gives us the dictionary of our session items that we use in our views.  
From this, we only get the keys.  
If we want to get both the keys and the values, we use the items() method.

Using this:

def sessionmethodsinview(request):

    items = request.session.items() # using this we get both key and vlaues

    print(items)

We get dict\_items([('fname', 'talha'), ('lname', 'mudassar')]) in the terminal.

Now, we have another method called setdefault().  
If the key is present, it returns its value.  
If the key does not exist, then it sets the key with the given value

setdefault().

def sessionmethodsinview(request):

age = request.session.setdefault('age',31)

    print(age)  #If the key is present, it returns its value.

    #If the key does not exist, then it sets the key with the given value

Here are other methods that we use :

    # Returns the default age (in seconds) for session cookies from settings.py (SESSION\_COOKIE\_AGE).

    session\_cookies\_age = request.session.get\_session\_cookie\_age()

    print("session cookie age", session\_cookies\_age)

    # Useful for checking how long session cookies last (default = 1209600 seconds = 2 weeks).

    # Returns how many seconds until the session expires.

    expire\_age = request.session.get\_expiry\_age()

    print("Expire age", expire\_age)

    # Helpful if you want to know remaining time before session deletion.

    # Returns the exact date & time when the session will expire.

    expiry\_date = request.session.get\_expiry\_date()

    print("Expire date", expiry\_date)

    # Good for logging or reminding users when their session will end.

    # Returns True if the session will expire when the browser is closed.

    expire\_at\_browser\_close = request.session.get\_expire\_at\_browser\_close()

    print("Expire at browser", expire\_at\_browser\_close)

    # Useful when SESSION\_EXPIRE\_AT\_BROWSER\_CLOSE = True in settings.py.

Now,

We use this to set the expiry date of the session

def setsession(request):

    request.session['fname']= 'talha'

    request.session['lname']= 'mudassar'

    request.session.set\_expiry(10)# tset session for 10 second

    return render(request, 'student/setsession.html')

But by using this, the data cannot be deleted from the database — we need flush to delete it.  
However, using flush, the expired data is not removed.  
flush cannot clear the expired data.  
If we want to remove the expired data, we use clear\_expired."

clear\_expired:

def sessionclear(request):

    request.session.clear\_expired()

    return render(request, 'student/sessionclear.html')

We can also use flush and clear\_expired at a time .  
If we don’t, our database can become overloaded or grow too fast

In settins we also set cookie age:

Mainproject/settings.py:

SESSION\_COOKIE\_AGE = 400

SESSION\_COOKIE\_NAME = 'sessionname'

SESSION\_COOKIE\_PATH = '/home'

Complete code:

from django.shortcuts import render

from datetime import datetime, timedelta, timezone

def setsession(request):

    # Store values in the session (just like key-value pairs in a dictionary)

    request.session['fname'] = 'talha'

    request.session['lname'] = 'mudassar'

    # Now session contains something like: {'fname': 'talha', 'lname': 'mudassar'}

    # Control how long the session should last:

    # Option 1 → Uncomment this to make the session expire after 10 seconds

    # request.session.set\_expiry(10)

    # Option 2 → If expiry = 0, the session will expire when the browser is closed

    request.session.set\_expiry(0)

    # Render response and save session data

    return render(request, 'student/setsession.html')

def getsession(request):

    # first\_name = request.session['fname'] # method 1 to get the session

    first\_name = request.session.get('fname') # second method to get the session its also work

    last\_name = request.session.get('lname')

    #  first\_name = request.session.get('fname', 'Guest') # here we use for default value if fname are empty it shows default value

    return render(request, 'student/getsession.html',{'first\_name':first\_name,'last\_name':last\_name})

def delsession(request):

    if 'fname' in request.session:  # use  if condition beucause if session not exist it shows error

        del request.session['fname']

        del request.session['lname']

    return render(request, 'student/delsession.html')

def flushsession(request):

    request.session.flush() #We use this when we want to ensure that the previous session data cannot be accessed by the user

    # use case : we use this when user logout or other like if user logout so we use this to flush all their session data

    return render(request, 'student/flushsession.html')

def sessionmethodsinview(request):

    # Access all the keys stored in the current session

    key = request.session.keys()

    print(key)

    # Example Output: dict\_keys(['fname', 'lname'])

    # We use this when we only want the keys (like dictionary keys).

    # Access both keys and values stored in the session

    items = request.session.items()

    print(items)

    # Example Output: dict\_items([('fname', 'talha'), ('lname', 'mudassar')])

    # We use this when we need both keys and values together.

    # setdefault() → If 'age' exists, return its value. If not, set it to 31.

    age = request.session.setdefault('age', 31)

    print(age)

    # Example: If session has {'age': 25}, it will print 25.

    # If 'age' is not in session, it will create {'age': 31} and print 31.

    # Returns the default age (in seconds) for session cookies from settings.py (SESSION\_COOKIE\_AGE).

    session\_cookies\_age = request.session.get\_session\_cookie\_age()

    print("session cookie age", session\_cookies\_age)

    # Useful for checking how long session cookies last (default = 1209600 seconds = 2 weeks).

    # Returns how many seconds until the session expires.

    expire\_age = request.session.get\_expiry\_age()

    print("Expire age", expire\_age)

    # Helpful if you want to know remaining time before session deletion.

    # Returns the exact date & time when the session will expire.

    expiry\_date = request.session.get\_expiry\_date()

    print("Expire date", expiry\_date)

    # Good for logging or reminding users when their session will end.

    # Returns True if the session will expire when the browser is closed.

    expire\_at\_browser\_close = request.session.get\_expire\_at\_browser\_close()

    print("Expire at browser", expire\_at\_browser\_close)

    # Useful when SESSION\_EXPIRE\_AT\_BROWSER\_CLOSE = True in settings.py.

    # Sending data to template for display

    return render(request, 'student/sessionmethodsinview.html', {

        'key': key,

        'items': items,

        'age': age,

        'session\_cookies\_age': session\_cookies\_age,

        'expire\_age': expire\_age,

        'expiry\_date': expiry\_date,

        'expire\_at\_browser\_close': expire\_at\_browser\_close,

    })

def sessionclear(request):

    request.session.clear\_expired() # If we want to remove the expired data, we use clear\_expired."

    return render(request, 'student/sessionclear.html')

def sessionmethodsintemplate(request):

    keys= request.session.keys()

    items = request.session.items()

    return render(request, 'student/sessionmethodsintemplate.html',{'keys':keys,'items':items})

# here some method to check either user browser can support  set or get cookies or not

def settestcookie(request):

    request.session.set\_test\_cookie()

    return render(request, 'student/settestcookie.html')

def checktestcookie(request):

    print(request.session.test\_cookie\_worked())

    return render(request, 'student/checktestcookie.html')

def deltestcookie(request):

    request.session.delete\_test\_cookie()

    return render(request, 'student/deltestcookie.html')