1. Query set
2. Session management
3. Middleware
4. Custom filter

Validate through server

If user logout and try to go back page so it can use the previous page after logout. So we validate session through Server instead of browser. For this we use decorator

from django.views.decorators.cache import cache\_control #validate thourgh server instead of browser

# Create your views here.

#apply decorator to validate through server instead of browser because browser store the previous session data

@cache\_control(no\_cache=True ,must\_revalidate=True,no\_store=True)

def studenthome(request):

    try:

        if request.session['rollno']!=None:

            rollno=request.session['rollno']

            stu = Student.objects.get(rollno=rollno)

            return render(request,'studenthome.html',locals())

    except KeyError:

        return render('nouapp:login')

locals()

if there are more then one variable in the view so instead of using dictionary to send the variable in template we use the locals functions.

def studenthome(request):

    try:

        if request.session['rollno']!=None:

            rollno=request.session['rollno']

            stu = Student.objects.get(rollno=rollno)

            return render(request,'studenthome.html',locals())

access the variable(access through object of class)

{{stu.name}}

Another way of logout

def studentlogout(request):

    try:

        del request.session['rollno']

    except KeyError:

        return redirect('nouapp:login')

    return redirect('nouapp:login')

Difference between redirect and render

Render will tell Rails what view it should use (with the same parameters you may have already sent) but redirect\_to sends a new request to the browser.

Redirect - we call one view using another view.

The admin cannot register itself. The registration is done by superuser and then id password is given to admin