deti universidade de aveiro
departamento de electrónica,
telecomunicações e informática

Technologies and Web Programming

Django Framework

Django Framework

Django Authentication

Django Authentication

- Django comes with a user authentication system. It handles user accounts, groups, permissions and user sessions.
- It handles both authentication and authorization.
  - Authentication verifies if users are who they claim to be.
  - Authorization determines what authenticated users are allowed to do.
- It implements:
  - Users, Groups and Permissions
  - Forms and view tools for logging users, or restricting content.
  - A password hashing system.

Django Authentication

- To use Django authentication system, verify if the following modules are loaded at MIDDLEWARE_CLASSES and INSTALLED_APPS keys.

```
MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
```

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app',
```

- Also, the database must be initialized. If not, run the command: *python manage.py migrate*.

Creating Login and Logout

- In the file “urls.py”, import the needed modules and define the needed urls:

urls.py

```
16
17  from django.contrib import admin
18  from django.contrib.auth import views as auth_views
19  from django.urls import path, include
20
21  from app import views
22
23  urlpatterns = [
24      path('login/', auth_views.LoginView.as_view(template_name='login.html'), name='login'),
25      path('logout', auth_views.LogoutView.as_view(next_page='/'), name='logout'),
26
```

Creating Login and Logout

- Use the file “login.html” to create the login template, and make the following modifications:

login.html

```
<form action="." method="post" class="form-horizontal">
    {% csrf_token %}
    <h4>Use a local account to log in.</h4>
    <hr />
    <div class="form-group">
        {{ form.username.label_tag }}
        <div class="col-md-10">
            {{ form.username }}
        </div>
    </div>
    <div class="form-group">
        {{ form.password.label_tag }}
        <div class="col-md-10">
            {{ form.password }}
        </div>
    </div>
```

Creating Login and Logout

- Use the file “loginpartial.html” to create an area where to show the login status. For that, modify “layout.html” file as follows:

layout.html

```
27      <div class="collapse navbar-collapse" id="navbarDefault">
28        <ul class="navbar-nav mr-auto">
29          <li class="nav-item active">
30            <a class="nav-link" href="{% url 'home' %}">Home</a>
31          </li>
32          <li class="nav-item">
33            <a class="nav-link" href="{% url 'about' %}">About</a>
34          </li>
35          <li class="nav-item">
36            <a class="nav-link" href="{% url 'contact' %}">Contact</a>
37          </li>
38        </ul>
39        {% include 'loginpartial.html' %}
40      </div>
41    </nav>
```

Authorization

- Authorization can be automatically managed by Django.
  - Through object “request.user”, it’s possible to verify if a given user is authenticated and have authorization to do operations.

views.py

```
36  def authorins(request):
37      if not request.user.is_authenticated or request.user.username != 'admin':
38          return redirect('/login')
39      # if POST request, process form data
40      if request.method == 'POST':
41          # create form instance and pass data to it
42          form = AuthorInsForm(request.POST)
43          if form.is_valid(): # is it valid?
44              name = form.cleaned_data['name']
45              email = form.cleaned_data['email']
46              a = Author(name=name, email=email)
47              a.save()
48              return HttpResponse("<h1>Author Inserted!!!</h1>")
49      # if GET (or any other method), create blank form
50      else:
51          form = AuthorInsForm()
52      return render(request, 'authorins.html', {'form': form})
```

Django Framework

Django Sessions

State

- HTTP protocol is a stateless protocol, which means that it doesn’t have any mechanism to save the connection state and as so it doesn’t allow sessions creation.
- To do this, some exterior mechanisms were developed in web clients and servers, which allow to save state data over multiple HTTP connections, producing artificial sessions.
- Mechanisms, like:
  - Cookies;
  - High level tools, using databases, to manage users, authentications and sessions.

Cookies

- A cookie is a little piece of information sent by a web server to its client, a browser, to save it while they are in communication.
- It’s possible to save some kind of information in this cookie, like the user’s username, for example.
- This cookies mechanism is in wide use by almost web sites, but it has some disadvantages:
  - Saving cookies in the browser is not compulsory, which doesn’t allow to offer warranty of a good service;
  - They can’t be used to save important information – they aren’t secure;
  - The server can be inhibited, at some time, to access crucial information to continue the interaction with the client.

Django Sessions

- Django offers a high level mechanism for sessions establishment, which allows to save all kind of information in the server itself.
  - This information is saved in the database.
- To use this mechanism, verify the presence of the following lines in “settings.py” file.

```
MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
```

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app',
```

Django Sessions (ii)

- Django manage automatically the sessions with its clients in a simple and clean way, through “request.session” object, which is a dictionary.

views.py

```
def bookquery(request):
    # if POST request, process form data
    if request.method == 'POST':
        # create form instance and pass data to it
        form = BookQueryForm(request.POST)
        if form.is_valid(): # is it valid?
            query = form.cleaned_data['query']
            if 'searched' in request.session and request.session['searched'] == query:
                return HttpResponse('Query already made!!!')
            request.session['searched'] = query
            books = Book.objects.filter(title__icontains=query)
            return render(request, 'booklist.html', {'boks': books, 'query': query})
    # if GET (or any other method), create blank form
    else:
        form = BookQueryForm()
    return render(request, 'bookquery.html', {'form': form})
```