

Topic: Login / Logout & User Restrictions Module Part 15

Speaker: Udemy Instructor Rathan Kumar | Notebook: Django: Automating Common Tasks



1. We completed the REGISTER module previously. To make your Login link work, we need to update our URLS.PY

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name='home'),
    # Links to our dataentry app's URLS.PY
    path('dataentry/', include('dataentry.urls')),
    path('celery-test/', views.celery_test),
    # registration and login
    path('register/', views.register, name='register'),
    path('login/', views.login, name='login'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

2. Create our LOGIN function in the VIEWS.PY

```
def login(request):
    return render(request, 'login.html')

def logout(request):
    return render(request, 'home.html')
```

3. Create our LOGIN.HTML

```
1 {% extends 'base.html' %}
2 {% block content %}
3
4
5 {% load crispy_forms_tags %}
6
7 <div class="container mt-5" style="max-width: 600px;">
8     <<h2 class="text-center">Login</h2>
9     <<form action="{% url 'login' %}" method="POST">
10         <<{{ csrf_token %}}
11
12         <<<div class="form-group">
13             <<{{ form | crispy }}
14         <<</div>
15
16         <<<div class="form-group">
17             <<<input type="submit" value="Login" class="btn btn-primary">
18         <<</div>
19     <</form>
20     <<{% include 'alerts.html' %}
21
22 </div>
23
24 {% endblock %}
```

4. Update our BASE.HTML

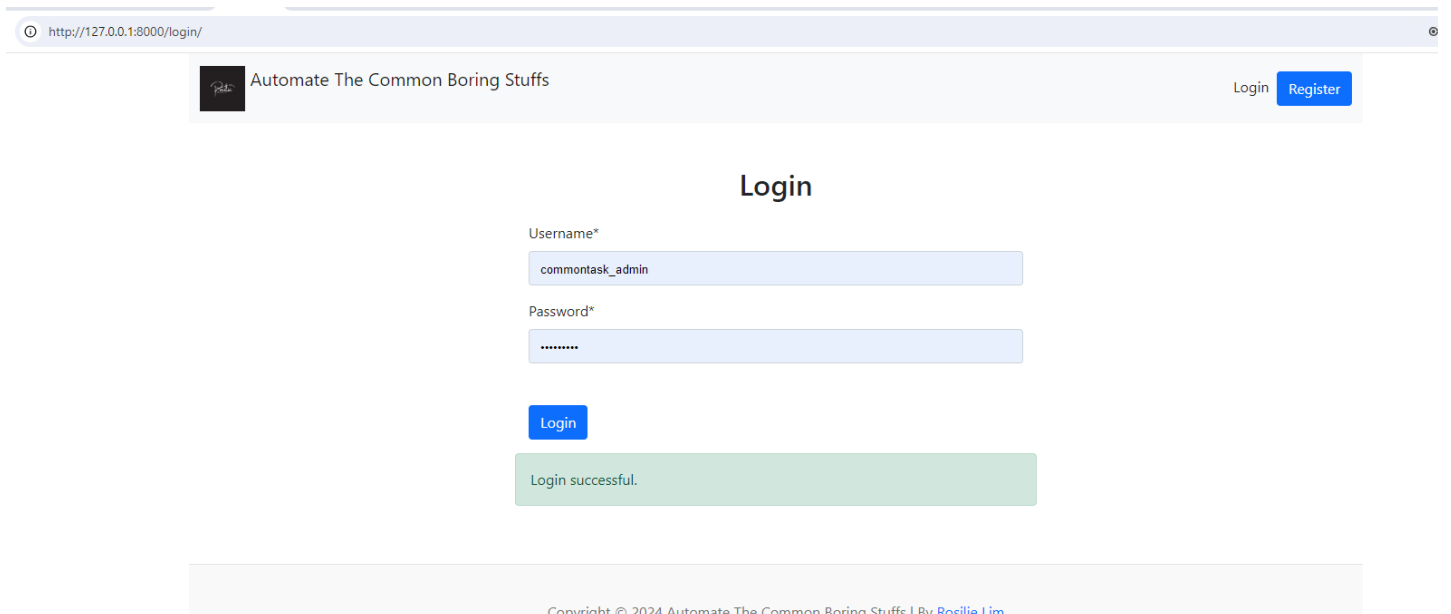
```
<div>
    <a href="{% url 'login' %}" class="float-right text-decoration-none text-dark">Login</a>
    &nbsp;<a href="{% url 'register' %}" class="float-right btn btn-primary">Register</a>
</div>
```

5. Update our VIEWS.PY

```
EXPLOSER
AUTOMATINGCOMMONTASKS
  autocommontasks_main
    __pycache__
    static
    __init__.py
    asgi.py
    celery.py
    forms.py
    settings.py
    uris.py
    views.py
    wsgi.py
  dataentry
  env
  media
  Resources
  static
  templates
    dataentry
      exportdata.html
      importdata.html
    alerts.html
    base.html
    home.html
    login.html
    register.html
  uploads
  .env
  .gitignore
  db.sqlite3
  manage.py

44
45
46 def login(request):
47
48     if request.method == 'POST':
49         form = AuthenticationForm(request, request.POST)
50         if form.is_valid():
51             # gets the value from the Django Form
52             username = form.cleaned_data['username']
53             password = form.cleaned_data['password']
54
55             user = auth.authenticate(username=username, password=password)
56             if user is not None:
57                 # authenticated user is login
58                 auth.login(request, user)
59                 return redirect('home')
60             else:
61                 messages.error(request, 'Invalid credentials.')
62                 return redirect('login')
63
64         else:
65             form = AuthenticationForm()
66             context = {
67                 'form': form,
68             }
69
70     return render(request, 'login.html', context)
71
```

Run the code:



6. We should not show REGISTER button if we are logged in, so we update our BASE.HTML AS:

```

<nav class="navbar navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="{% url 'home' %}">
      
      Automate The Common Boring Stuffs
    </a>
    {% if user.is_authenticated %}
    <!-- show Logout button -->
    <div>
      Logged in as: {{user}}
      <a href="{% url 'logout' %}" class="float-right btn btn-outline-danger">Logout</a>
    </div>
    {% else %}
    <div>
      <a href="{% url 'login' %}" class="float-right text-decoration-none text-dark">Login</a>
      &nbsp;<a href="{% url 'register' %}" class="float-right btn btn-primary">Register</a>
    </div>
    {% endif %}
  </div>
</nav>

```

7. To make our logout button work, we update our VIEWS.PY AS:

```

def logout(request):
    auth.logout(request)
    return redirect('home')

```

8. Test our code:



Automate The Boring Stuff With Django

Import Data

Imports data from CSV file to any table.

Export Data

Exports data from a specific table to a CSV file.

Bulk Emails

Sends bulk emails to users

Email Tracking

Tracks emails and shows open and click rate

Compress Images

Compresses Image files and reduces the size

Web Scraping

Scrapes websites for useful information

9. To make see only the tools when they log in, we must update our HOME.HTML. We add the IF ELSE statement so that the tools would only appear if we log in otherwise we see the homepage with some information instead.

```

AutomatingCommonTasks
EXPLORER
AUTOMATINGCOMMONTASKS
  autocommontasks_main
  _pycache_
  static
    css
      custom.css
    images
    js
  _init_.py
  asgi.py
  celery.py
  forms.py
  settings.py
  urls.py
  views.py
  wsgi.py
  dataentry
  env
  media
  Resources
  static
  templates
    dataentry
      exportdata.html
      importdata.html
      alerts.html
      base.html
      home.html
      login.html
      register.html
    uploads
  .env
  .gitignore
  db.sqlite3
  manage.py
templates > home.html > div.container
1 {% extends 'base.html' %}
2 {% block content %}
3 <div class="container" style="padding: 50px;margin-top: 5%;">
4
5     <!-- Only logged-in user can see the tools -->
6
7     {% if user.is_authenticated %}
8     <h2 class="text-center">Automate The Common Boring Stuffs with Django</h2>
9     <div class="row pt-5">
10
11         <!-- Import data -->
12         <div class="col-md-4">
13             <a href="{% url 'import_data' %}" class="text-decoration-none text-dark">
14                 <div class="card alert-primary shadow">
15                     <div class="card-header">
16                         <h4>Import Data</h4>
17                     </div>
18                     <div class="card-body">
19                         <p class="card-text">Imports data from CSV file to any table.</p>
20                     </div>
21                 </div>
22             </a>
23         </div>
24         <!-- Export Data -->
25         <div class="col-md-4">
26             <a href="{% url 'export_data' %}" class="text-decoration-none text-dark">
27                 <div class="card alert-success shadow">
28                     <div class="card-header">

```

```

AutomatingCommonTasks
EXPLORER
AUTOMATINGCOMMONTASKS
  autocommontasks_main
  _pycache_
  static
    css
      custom.css
    images
    js
  _init_.py
  asgi.py
  celery.py
  forms.py
  settings.py
  urls.py
  views.py
  wsgi.py
  dataentry
  env
  media
  Resources
  static
  templates
    dataentry
      exportdata.html
      importdata.html
      alerts.html
      base.html
      home.html
      login.html
      register.html
    uploads
  .env
  .gitignore
  db.sqlite3
  manage.py
templates > home.html > div.container > div.row-pt-5 > div.col-md-4 > a.text-decoration-none.text-dark > div.card.alert-primary.shadow
3 <div class="container" style="padding: 50px;margin-top: 5%;">
52 <div class="row pt-5">
81
82 <!-- Web Scraping -->
83 <div class="col-md-4">
84 <a href="#" class="text-decoration-none text-dark">
85 <div class="card alert-warning shadow">
86 <div class="card-header">
87 <h4>Web Scraping</h4>
88 </div>
89 <div class="card-body">
90 <p class="card-text">Scrapes websites for useful information</p>
91 </div>
92 </a>
93 </div>
94 </div>
95
96 {% else %}
97 <!-- Home Page content if user is not logged in -->
98 <div class="home-info">
99 <h2 class="text-center">Automate The Common Boring Stuffs with Django</h2>
100 <p class="lead text-center text-muted">
101     Automate the Boring Stuff with Django is a project-based course to build various tools using the Django
102     framework to automate routine tasks, ranging from data processing to scheduling, and more.
103     By leveraging Django's capabilities, users can streamline their workflows, saving time and enhancing
104     productivity.
105 </p>
106 <div style="display: flex; justify-content: center;">
107 <a href="{% url 'login' %}" class="btn btn-success">LOGIN</a>
108 &nbsp;<a href="{% url 'register' %}" class="btn btn-primary">REGISTER</a>
109 </div>
110 </div>
111 </div>
112 </div>
113 {% endif %}
114 </div>
115
116 {% endblock %}

```

9. Load the page:



Automate The Common Boring Stuff with Django

"Automate the Boring Stuff with Django" is a project-based course to build various tools using the Django framework to automate routine tasks, ranging from data processing to scheduling, and more. By leveraging Django's capabilities, users can streamline their workflows, saving time and enhancing productivity.

LOGIN REGISTER

From here, you can login or register to see the Django tools.

We have made some custom CSS for our footer and home-info: We run COLLECTSTATIC to make sure that this external CSS is added to our root folder for deployment.

The image shows a code editor interface with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'static' and 'css', and files like 'urls.py', 'views.py', 'login.html', 'home.html', and 'register.html'. The 'css' folder is expanded, and 'custom.css' is selected. The code editor shows the following CSS code:

```
1 /*
2  * Footer
3  */
4 .blog-footer {
5     padding: 2.5rem 0;
6     color: #727272;
7     text-align: center;
8     background-color: #f9f9f9;
9     border-top: .05rem solid #e5e5e5;
10 }
11
12 .blog-footer p:last-child {
13     margin-bottom: 0;
14 }
15
16 /*
17  * Form Group
18  */
19 .form-group {
20     padding: 15px !important;
21 }
22
23
24 /*
25  * Home Info
26  */
27 .home-info {
28     /* position: absolute;
29     top: 50%;
30     left: 50%;
31     transform: translate(-50%, -50%); */
32     padding: 20px;
33 }
```