今回のテーマは「LoginViewで作成したログイン画面をカスタマイズする」です。ここまではDjangoに組み込まれたビューを使用して認証機能を実装してきましたが、ここからはカスタマイズを加えていきます。LoginViewをベースとしたカスタマイズをすることでフォームに手を加えたり、テンプレート名を変更するなど自由なカスタマイズが出来るようになります。

※本ページは「PasswordResetViewを使用してパスワードリセット画面を作成する」まで読まれた方を対象としています。そのためサンプルソースコードが省略されている場合があります。

URLのカスタマイズ

ここまではdjango.contrib.auth.urlsのurlpatternsをincludeしただけでした。これではカスタマイズは難しいのでaccounts/urls.pyにURLを設定し直しましょう。

mysite/urls.py(一部抜粋)

```
  urlpatterns = [
      path('admin/', admin.site.urls),
-     path('accounts/', include('django.contrib.auth.urls')),
      path('accounts/', include('accounts.urls')),
      path('', include('base.urls')),
      path('thread/', include('thread.urls')),
      path('api/', include('api.urls')),
      path('search/', include('search.urls')),
      path('sitemap.xml', sitemap, {'sitemaps': sitemaps}),
  ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

accounts/urls.py

```
from django.urls import path, include
from django.contrib.auth import views as av
from . import views

app_name = 'accounts'

urlpatterns = [
    # copy from django.contrib.auth.urls.py
    path('login/', av.LoginView.as_view(), name='login'),
    path('logout/', av.LogoutView.as_view(), name='logout'),

    path('password_change/', av.PasswordChangeView.as_view(), name='password_change'),
    path('password_change/done/', av.PasswordChangeDoneView.as_view(), name='password_change_done'),

    path('password_reset/', av.PasswordResetView.as_view(), name='password_reset'),
    path('password_reset/done/', av.PasswordResetDoneView.as_view(), name='password_reset_done'),
    path('reset///', av.PasswordResetConfirmView.as_view(), name='password_reset_confirm'),
    path('reset/done/', av.PasswordResetCompleteView.as_view(), name='password_reset_complete'),

    path('create/', views.UserCreateView.as_view(), name="create"),
    path('profile/', views.UserProfileView.as_view(), name="profile"),
    path('change/', views.UserChangeView.as_view(), name="change"),
]
```

avはauth_viewsの頭文字をとった省略形です。この時点では前回まで実装した内容とほとんど変わりありません。これでログイン、ログアウト、パスワード変更画面等、別々に設定可能な準備が整いました。

URLショートコードの修正

以降、URLのショートコードには'accounts'が付くことに注意して下さい。例えばログイン画面へのリダイレクトは redirect(reverse_lazy('accounts:login'))となります。テンプレートのurlの表記も全て変わりますので注意してくだい。

templates/base/base.html(一部抜粋)

```
  <div class="right menu">
      {% if user.is_authenticated %}
      <a class="item" href="{% url 'accounts:profile' %}">ユーザー情報</a>
-     <a class="item" href="{% url 'logout' %}">ログアウト</a>
+     <a class="item" href="{% url 'accounts:logout' %}">ログアウト</a>
      {% else %}
-     <a class="item" href="{% url 'login' %}">ログイン</a>
+     <a class="item" href="{% url 'accounts:login' %}">ログイン</a>
      <a class="item" href="{% url 'accounts:create' %}">ユーザー登録</a>
      {% endif %}
  </div>
```

templates/registration/login.html(一部抜粋)

```
  <a class="ui item" href="{% url 'accounts:create' %}">ユーザー登録</a>／
- <a class="ui item" href="{% url 'password_reset' %}">パスワードを忘れた場合</a>
+ <a class="ui item" href="{% url 'accounts:password_reset' %}">パスワードを忘れた場合</a>
```

templates/registration/profile.html

```
  <a class="ui button" href="{% url 'accounts:change' %}">登録情報変更</a>
- <a class="ui button" href="{% url 'password_change' %}">パスワード変更</a>
+ <a class="ui button" href="{% url 'accounts:password_change' %}">パスワード変更</a>
```

カスタムフォームの作成

LoginViewはデフォルトでAuthenticationFormを使っていますが、これでは少々融通が利かないので自分でカスタマイズ可能なフォームクラスを作成しておきましょう。

accounts/forms.py(一部抜粋)

```
+ from django.contrib.auth.forms import AuthenticationForm
+
+ class CustomAuthenticationForm(AuthenticationForm):
+     def __init__(self, *args, **kwargs):
+         kwargs.setdefault('label_suffix', '')
+         super().__init__(*args, **kwargs)
```

もしフォームにclassを付与したいなどの場合はここで追加します。

ケース１：LoginViewを使用してas_view()でプロパティを渡す

このケースではビューを自作する必要はありません。as_view関数でLoginViewのクラス変数を書き換える方法です。

accounts/urls.py

```
+ from .forms import CustomAuthenticationForm

- path('login/', av.LoginView.as_view(), name='login')
+ path('login/', av.LoginView.as_view(form_class=CustomAuthenticationForm
                                      ), name='login'),
  path('logout/', av.LogoutView.as_view(), name='logout'),
```

今回はform_classのみ書き換えましたがtemplate_name等も書き換え可能です。LoginViewの詳細は「」をご覧ください。

ケース２：LoginViewを継承するビュークラスを作る

次にLoginViewを継承して新しいビュークラスを作る場合を考えましょう。as_view関数でプロパティを書き換える方法よりも応用の効く方法です。
accounts/views.pyに書き加えていきます。

accounts/views.py(一部抜粋)

```
#importは行頭に追加
+ from .forms import UserChangeForm, CustomAuthenticationForm

+ class CustomLoginView(LoginView):
+     form_class = CustomAuthenticationForm
```

accounts/urlsも書き換えます。
accounts/urls.py(一部抜粋)

```
- path('login/', av.LoginView.as_view(form_class=CustomAuthenticationForm
-                                     ), name='login'),
+ path('login/', views.CustomLoginView.as_view(), name='login'),
```

ログイン後の遷移先を変更する

ログイン後の遷移については以前触れましたが、改めて説明しておきます。ログイン後の遷移先URLは'/accounts/profile/'が指定されています。これを変更するにはmysite/settings.pyでLOGIN_REDIRECT_URLを指定することで設定出来ます。例えば、ログイン後にトップページに遷移したい場合は

mysite/settings.py(一部抜粋)

```
+ LOGIN_REDIRECT_URL = '/'
```

のように追加すればOKです。

最後に

カスタマイズと言ってもさほど大変なことはありません。Djangoの予めある機能を活かしてカスタマイズをしていきましょう。次回はログアウト画面をカスタマイズしていきます。