---------- DJANGO TUTORIAL 6 ----------

In this video I'll show you how to allow users to add new and update articles on our website. I'll also style the site and add a toolbar.

# 1. In articles/views.py we'll use a CreateView which will display a form that users can use to add articles

```
from django.views.generic import ListView, DetailView
from django.views.generic.edit import CreateView
from .models import Article

class HomePageView(ListView):
    model = Article
    template_name = 'home.html'
    context_object_name = 'all_articles_list'

class ArticleDetailView(DetailView):
    model = Article
    template_name = 'article_detail.html'

class ArticleCreateView(CreateView):
    # Define DB model to use
    model = Article
    # Define template
    template_name = 'add_article.html'
    fields = ['title', 'author', 'text', 'photo']
```

# 2. Add our add_article URL to articles/urls.py

```
from django.urls import path
from .views import HomePageView, ArticleDetailView, ArticleCreateView

urlpatterns = [
path('article/new/', ArticleCreateView.as_view(), name='add_article'),
path('article/<int:pk>/', ArticleDetailView.as_view(), name='article_detail'),
path('', HomePageView.as_view(), name='home'),
]
```

# 3. Update base.html to include a toolbar

```
<body>
    <ul>
      <li><a href="{% url 'home' %}">Home</a></li>
      <li><a href="{% url 'add_article' %}">Add Article</a></li>
    </ul>
    <div class="wrapper">
```

# 4. Update base.css to style the toolbar as well as the h3 tags

```
h3{
  margin-bottom: 1em;
  font-family: Arial, Helvetica, sans-serif;
}

a:link {
  text-decoration: none;
}

/* Toolbar Styling */
/* Tool bar is green */
ul {
  list-style-type: none; /* Get rid of bullets */
  margin: 0;
  padding: 0;
  overflow: hidden; /* Hide content that overflows */
  background-color: #007F66;
}

li {
  float: left;
}

li a {
  display: block; /* Display as a block element */
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 1.8em;
  font-family: Arial, Helvetica, sans-serif;
}

/* Change the link color to bright green on hover */
li a:hover {
  background-color: #00AB66;
}
```

# 5. Create the add_article.html file

```
{% extends 'base.html' %}
{% block content %}
<h3>New Article</h3>
<!-- csrf_token protects against cross-site scripting attacks
form.as_p outputs our form data in p tags -->
<form action="" method="post">{% csrf_token %} {{ form.as_p }}
  <input type="submit" value="Save" />
</form>
{% endblock content %}
```

# 6. Update articles/models so we can define where to send the user after they upload an article

```
from django.urls import reverse

class Article(models.Model):
    title = models.CharField(max_length=200)
    text = models.TextField()
author = models.ForeignKey(
    'auth.User',
    on_delete = models.CASCADE,
    )

    photo = models.ImageField(upload_to="gallery", default = 'drop-bear.png')

    # Returns the article title when the object
    # is printed
    def __str__(self):
        return self.title

    # You should add a __str__ and get_absolute_url to every
    # model that you make
    # This sends a user to the article_detail page after
    # data is submitted to the DB
    def get_absolute_url(self):
        # reverse allows us to refer to an object using its
        # URL template name
        return reverse('article_detail', args=[str(self.id)])
```

# 7. Clean up form styling in base.html add

```
<head>
  <title>Django Blog</title>
  <link href="{% static 'css/base.css' %}" rel="stylesheet">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
  </head>
```

And at the bottom

```
</div> <!-- End of wrapper -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/
ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/
JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
  </body>
```

# 8. Fix the images in base.css

```
.art-img {
  float: left;
  margin: 55px 15px 15px 15px;
  height: 100px;
  width: 100px;
```

```
}
# 9. Create templates/article_edit.html so users can edit articles

{% extends 'base.html' %}

{% block content %}
<h3>Edit Article</h3>
<form action="" method="post">{% csrf_token %} {{ form.as_p }}
  <input type="submit" value="Update" />
</form>
{% endblock content %}

# 10. Add a button to article_detail.html so that users can go to the article_edit.html page

{% extends 'base.html' %}

{% block content %}
<main>
  <section>
    <h3>{{ article.title }}</h3>
    <article>
      <img src="{{ article.photo.url}}" alt="{{ article.title }}" class="art-img">
      <p>{{ article.text|linebreaks }}</p>
    </article>
    <a href="{% url 'article_edit' article.pk %}" class="btn btn-primary btn-sm">Edit Article</a>
  </section>
</main>
{% endblock content %}

# 11. Add the UpdateView to articles/views.py

from django.views.generic import ListView, DetailView, UpdateView

class ArticleUpdateView(UpdateView):
    model = Article
    template_name = 'article_edit.html'
    # define we only want to change the title or article text
    fields = ['title', 'text']

# 12. Add the new url pattern to articles/urls.py

from .views import HomePageView, ArticleDetailView, ArticleCreateView, ArticleUpdateView

urlpatterns = [
path('article/<int:pk>/edit/', ArticleUpdateView.as_view(), name='article_edit'),
```