

Lab 4 Part (2)

In this exercise we will continue to work on our blog application from lab 4 Part (1) by adding forms so a user can create, edit, or delete any of their blog entries.

Forms

Forms are very common in web applications and can be difficult to implement correctly. When accepting data input from a user there are always security concerns in terms of XSS attacks and because of this, proper error handling is required. We also need to alert the user to problems with the form in terms of validation. Fortunately for us, Django's built-in Forms abstract away much of the difficulty and provide a rich set of tools to handle common challenges we face when working with forms.

View

Open the **views.py** file and create a new view using the code provided below. Here we are importing a new generic class called **CreateView** at the top and then subclassing it to create a new view called **BlogCreateView**.

Code

```
views.py
blog > views.py
1  # blog/views.py
2  from django.views.generic import ListView, DetailView
3  from django.views.generic.edit import CreateView # new
4
5  from .models import Post
6
7  class BlogListView(ListView):
8      model = Post
9      template_name = 'home.html'
10     context_object_name = 'all_posts_list'
11
12  class BlogDetailView(DetailView):
13      model = Post
14      template_name = 'post_detail.html'
15
16  class BlogCreateView(CreateView): # new
17      model = Post
18      template_name = 'post_new.html'
19      fields = ['title', 'author', 'body']
```

Within **BlogCreateView** we specify our database model **Post**, the name of our template **post_new.html**. For fields we explicitly set the database fields we want to use which are title, author, and body.

Template

Create a new template in VS Code called **post_new.html** and type in the following code:

Code

```
templates > <> post_new.html > ...
1  {% extends 'base.html' %}
2  {% block title %} New Post Page {% endblock title %}
3
4  {% block content %}
5      <h1>New Post</h1>
6      <form action="" method="POST">
7          {% csrf_token %}
8          {{ form.as_p }}
9          <input type="submit" value="Save"/>
10     </form>
11 {% endblock content %}
```

Let's break down what we have done:


- Line 1: we inherit from our base template
- Line 12 we use the HTML **<form>** tags with the POST method since we are sending data. We also add a **{% csrf_token %}** which Django provides to protect our form from cross site scripting attacks. This should be used for all your Django forms
- Line 14 we use **{{ form.as_p }}** which renders the data within paragraph **<p>** tags
- Line 15 we specify an input type of submit and assign it the value "Save"

URLConf

Let's add a new URLConf for **post_new**.

Open the file **blog/urls.py** and update it with the code shown below.

Code

```
blog >  urls.py
1  from django.urls import path
2  from .views import BlogListView, BlogDetailView, BlogCreateView
3
4  urlpatterns = [
5      path('', BlogListView.as_view(), name='home'),
6      path('post/<int:pk>/', BlogDetailView.as_view(), name='post_detail'),
7      path('post/new/', BlogCreateView.as_view(), name='post_new'),
8  ]
```

We just imported our view called **BlogCreateView** at line 2 and then we added the path to the new URL which will start with **post/new/** and is called **post_new**.

Update Template

We will need to update our base template to display a link to a page for entering new blog posts. It will take the form `` where **post_new** is the name for our URL.

Open the **base.html** file. Update the code inside the header to look like that shown below. Make sure to add the code to define the div classes **nav-left** ¹ and **nav-right** ² for both URLs in the header to enable CSS styling to be applied from **base.css**.

```
10  <header>
11      <div class='nav-left'>
12          <h1><a href="{% url 'home' %}">My Blog</a></h1>
13      </div>
14      <div class='nav-right'>
15          <h1><a href="{% url 'post_new' %}">New Blog Post</a></h1>
16      </div>
17  </header>
```

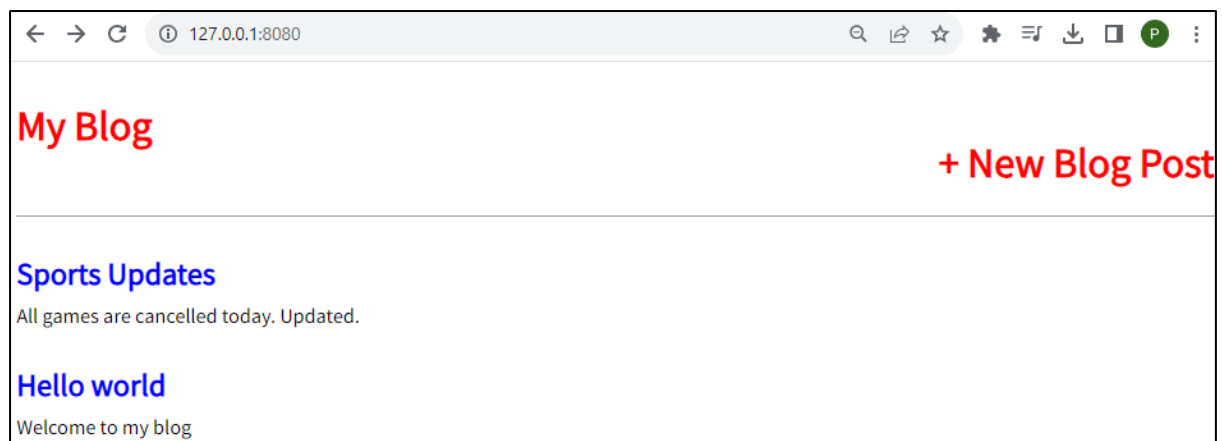
Code - Update CSS

Open **base.css** and add the following code to the style nav-left and nav-right classes:

```
.nav-left {  
    margin-right: auto;  
}  
  
.nav-right {  
    display: flex;  
    padding-top: 2rem;  
}
```

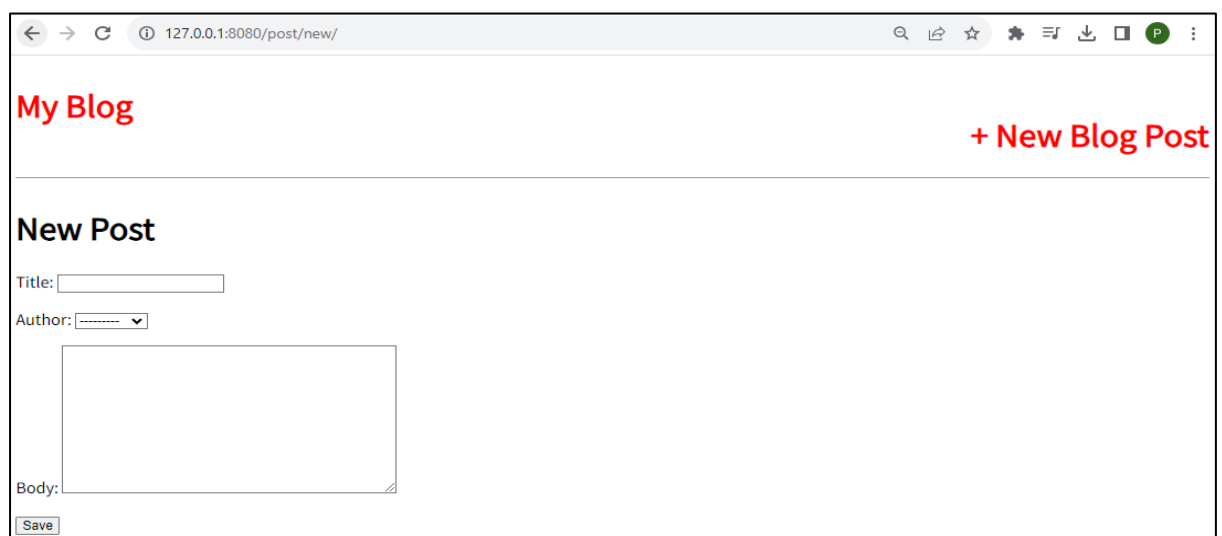
Run Server

Start the server with the command **python manage.py runserver** and go to the homepage at <http://127.0.0.1:8080/>.

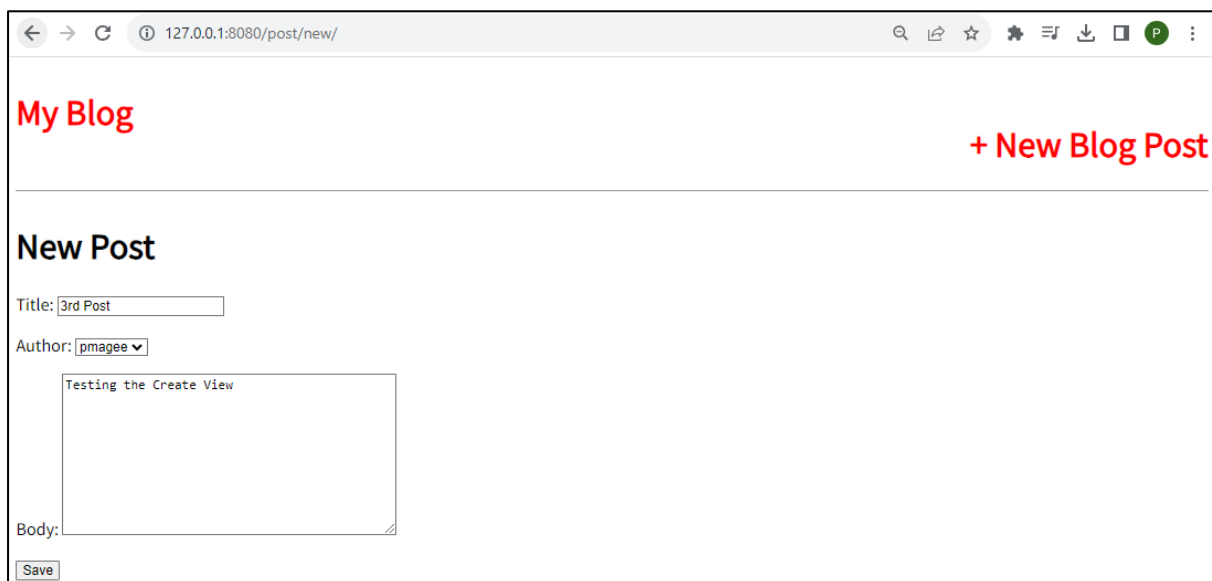


Click on our link for “+ New Blog Post” which will redirect you to:

<http://127.0.0.1:8080/post/new/>.



Go ahead and try to create a new blog post and submit it.



My Blog

+ New Blog Post

New Post

Title:

Author:

Body:

Something has gone wrong!



← → ↻ ⓘ 127.0.0.1:8080/post/new/

ImproperlyConfigured at /post/new/

No URL to redirect to. Either provide a url or define a get_absolute_url method on the Model.

Request Method: POST
Request URL: http://127.0.0.1:8080/post/new/
Django Version: 4.2.5
Exception Type: ImproperlyConfigured
Exception Value: No URL to redirect to. Either provide a url or define a get_absolute_url method on the Model.
Exception Location: C:\Users\pmagee\django\projects\env\Lib\site-packages\django\views\genericedit.py, line 127, in get_success_url
Raised during: blog.views BlogCreateView
Python Executable: C:\Users\pmagee\django\projects\env\Scripts\python.exe
Python Version: 3.11.5
Python Path: ['C:\\Users\\pmagee\\django\\projects\\lab-4-blog-application-pmagee',
'C:\\Users\\pmagee\\AppData\\Local\\Programs\\Python\\Python311\\python311.zip',
'C:\\Users\\pmagee\\AppData\\Local\\Programs\\Python\\Python311\\DLLs',
'C:\\Users\\pmagee\\AppData\\Local\\Programs\\Python\\Python311\\Lib',
'C:\\Users\\pmagee\\AppData\\Local\\Programs\\Python\\Python311',
'C:\\Users\\pmagee\\django\\projects\\env',
'C:\\Users\\pmagee\\django\\projects\\env\\Lib\\site-packages']
Server time: Wed, 27 Sep 2023 18:27:17 +0000

Django's error message is telling us that we did not specify where to send the user after successfully submitting the form. Let's send a user to the new post detail page so that they can see their new post.

Using reverse

Open the **models.py** file. Add an import on the second line for **reverse** and a new **get_absolute_url** method.

Code

```
models.py x
blog > models.py
1 from django.db import models
2 from django.urls import reverse # new
3
4 # Create your models here.
5 class Post(models.Model):
6     title = models.CharField(max_length=200)
7     author = models.ForeignKey(
8         'auth.User',
9         on_delete=models.CASCADE,
10    )
11
12    body = models.TextField()
13
14    def __str__(self):
15        return self.title
16
17    def get_absolute_url(self): # new
18        return reverse('post_detail', args=[str(self.id)])
```

reverse is a very handy utility function Django provides us to reference an object by its URL template name, in this case **post_detail**. If you recall our URL pattern is the following:

Code

```
path('post/<int:pk>/', BlogDetailView.as_view(), name='post_detail'),
```

For this route to work we must also pass in an argument with the pk or primary key of the object. Confusingly pk and id are interchangeable in Django but the Django docs recommend using self.id with **get_absolute_url**. So, we are telling Django that the ultimate location of a **Post** entry is it's **post_detail** view which is **posts/<int:pk>/** so the route for the first entry we have made will be at **posts/1**.

Try to create a new blog post again at <http://127.0.0.1:8080/post/new/>

← → ↻ ⓘ 127.0.0.1:8080/post/new/ 🔍 📄 ☆ ⚙️ ☰ ⬇️ 📱 P ⋮

My Blog

+ New Blog Post

New Post

Title:

Author:

Body:

Testing the Create View

Upon clicking the “Save” button you are now redirected to the detailed view page where the post appears.

← → ↻ ⓘ 127.0.0.1:8080/post/4/ 🔍 📄 ☆ ⚙️ ☰ ⬇️ 📱 P ⋮

My Blog

+ New Blog Post

3rd Post

Testing the Create View

Access the homepage at <http://127.0.0.1:8080/> and you will also notice that our earlier blog post is also there. It was successfully sent to the database, but Django didn't know how to redirect us after that.

← → ↻ ⓘ 127.0.0.1:8080 🔍 📄 ☆ ⚙️ ☰ ⬇️ 📱 P ⋮

My Blog

+ New Blog Post

[Sports Updates](#)
All games are cancelled today.

[Hello world](#)
Welcome to my blog

[3rd Post](#)
Testing the Create View

[3rd Post](#)
Testing the Create View

Update Form

In order to create an update form so users can edit blog posts we will use a built-in Django class-based generic view, **UpdateView**, and create the template, url, and view.

We need to import **UpdateView** on the second-from-the-top line and then subclass it in our new view **BlogUpdateView**. Add the code highlighted below to the **views.py** file.

Code

```
blog > views.py
1  # blog/views.py
2  from django.views.generic import ListView, DetailView
3  from django.views.generic.edit import CreateView, UpdateView # new
4
5  from .models import Post
6
7  class BlogListView(ListView):
8      model = Post
9      template_name = 'home.html'
10     context_object_name = 'all_posts_list'
11
12     class BlogDetailView(DetailView):
13         model = Post
14         template_name = 'post_detail.html'
15
16     class BlogCreateView(CreateView): # new
17         model = Post
18         template_name = 'post_new.html'
19         fields = ['title', 'author', 'body']
20
21     class BlogUpdateView(UpdateView): # new
22         model = Post
23         template_name = 'post_edit.html'
24         fields = ['title', 'body']
```

Notice that in **BlogUpdateView** we are explicitly listing the fields we want to use ['title', 'body']. This is because we assume that the author of the post is not changing; we only want the title and text to be editable.

Template

Create the template in VS Code for the edit page called **post_edit.html** & add in the following code.

Code

```
templates > <> post_edit.html > ...
1  {% extends 'base.html' %}
2  <!DOCTYPE html>
3  <html lang="en">
4      <head>
5          {% block title %}
6              Post Update Page
7          {% endblock title %}
8      </head>
9      {% block content %}
10         <h1>Edit Post</h1>
11         <form action="" method="POST">
12             {% csrf_token %}
13             {{ form.as_p }}
14             <input type="submit" value="Update"/>
15         </form>
16     {% endblock content %}
17 </html>
```

We again use HTML **<form></form>** tags, Django's **csrf_token** for security, **form.as_p** to display our form fields with paragraph tags, and finally give it the value **"Update"** on the submit button.

URLConf

Update the **urls.py** file as follows. Add the **BlogUpdateView** at the end of line 2 and then add the new route at line 8.

Code

```
blog > urls.py
1  from django.urls import path
2  from .views import BlogListView, BlogDetailView, BlogCreateView, BlogUpdateView
3
4  urlpatterns = [
5      path('', BlogListView.as_view(), name='home'),
6      path('post/<int:pk>/', BlogDetailView.as_view(), name='post_detail'),
7      path('post/new/', BlogCreateView.as_view(), name='post_new'),
8      path('post/<int:pk>/edit/', BlogUpdateView.as_view(), name='post_edit'),
9  ]
```

At the top we added our view **BlogUpdateView** to the list of imported views, then created a new url pattern for /post/pk/edit and gave it the name **post_edit**.

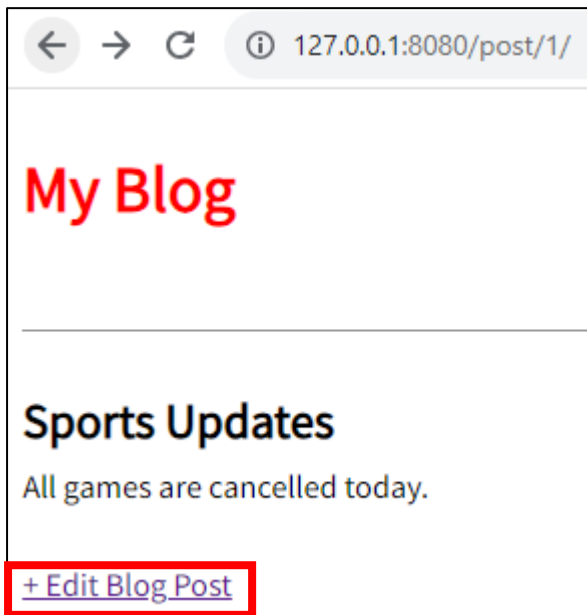
To start, add a new link shown at line 16 below to **post_detail.html** so that the option to edit a blog post appears on an individual blog page

Code

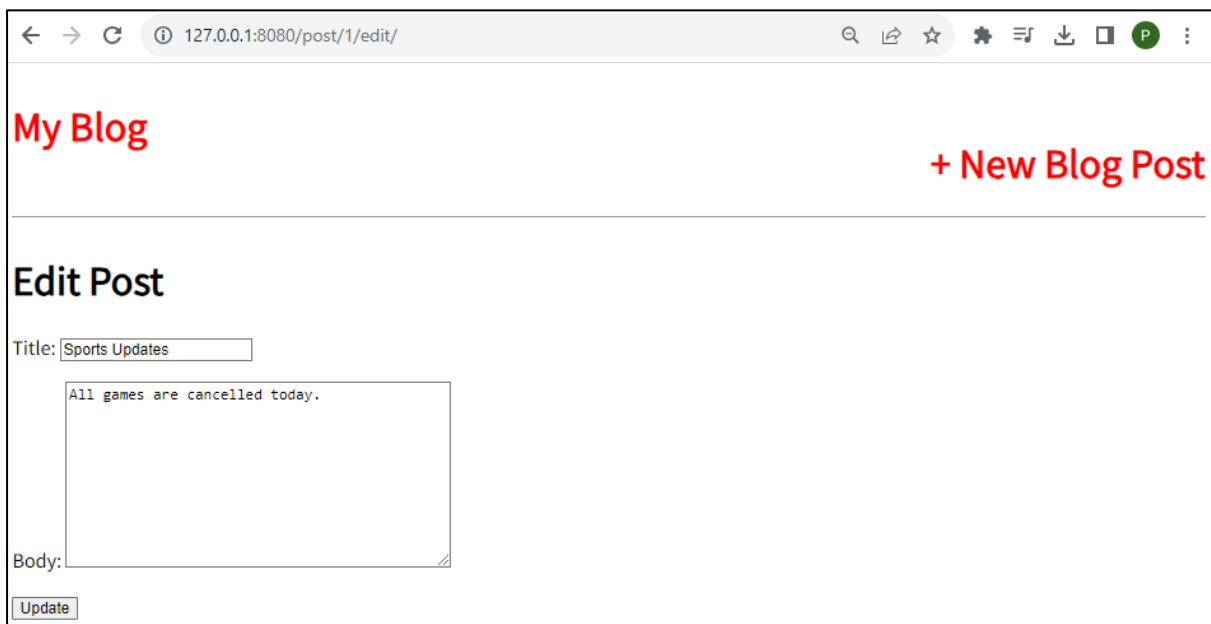
```
templates > <> post_detail.html > ...
1  {% extends 'base.html' %}
2  <!DOCTYPE html>
3  <html lang="en">
4      <head>
5          {% block title %}
6              Post Detail Page
7          {% endblock title %}
8      </head>
9      <body>
10         {% block content %}
11             <div class="post-entry">
12                 <h2>{{post.title}}</h2>
13                 <p>{{post.body}}</p>
14             </div>
15
16             <a href="{% url 'post_edit' post.pk %}">+ Edit Blog Post</a>
17         {% endblock content %}
18     </body>
19 </html>
```

We have just added a link using **<a href>...** and the Django template engine's **{% url ... %}** tag. Within it we have specified the target name of our url, which will be called **post_edit** and we also passed the parameter needed, which is the primary key of the post **post.pk**.

1. If you click on a blog entry you will see the new Edit button.



2. Click on “+ Edit Blog Post” and you will be redirected to the following URL <http://127.0.0.1:8080/post/1/edit/> if you selected your first blog post.



Note that the form is pre-filled with our database's existing data for the post.

3. Change the text in the blog post and click **Update**

And after clicking the “Update” button we are redirected to the detail view of the post where you can see the change. This is because of our `get_absolute_url` method in `models.py`.



Delete Post

The process for creating a form to delete blog posts is very similar to that for updating a post. We will use yet another generic class-based view, **DeleteView**, to help and we will then create a view, url, and template for the functionality.

View

Update the **views.py** file using the code provided below, by importing **DeleteView** and **reverse_lazy** at the top, then create a new view that subclasses **DeleteView**.

Code

Import **DeleteView**

```
3 from django.views.generic.edit import CreateView, UpdateView, DeleteView # new
```

Add this new import **reverse_lazy**

```
5 from django.urls import reverse_lazy # new
```

Add this new class

```
28 class BlogDeleteView>DeleteView): # new
29     model = Post
30     template_name = 'post_delete.html'
31     success_url = reverse_lazy('home')
```

We use **reverse_lazy** as opposed to just **reverse** so that it won't execute the URL redirect until our view has finished deleting the blog post.

Template

1. Add a new link to **post_detail.html** to delete blog posts. This time we have surrounded the link by `<p>` tags to create a bit of space.
2. Add `<p>` tags to the Edit Blog Post link on line 16 as well.

Code

```
templates > <> post_detail.html > ...
1  {% extends 'base.html' %}
2  <!DOCTYPE html>
3  <html lang="en">
4      <head>
5          {% block title %}
6              <title>Home Page</title>
7          {% endblock title %}
8      </head>
9      <body>
10         {% block content %}
11             <div class="post-entry">
12                 <h2>{{post.title}}</h2>
13                 <p>{{post.body}}</p>
14             </div>
15
16             <p><a href="{% url 'post_edit' post.pk %}">+ Edit Blog Post</a></p>
17             <p><a href="{% url 'post_delete' post.pk %}">+ Delete Blog Post</a></p>
18         {% endblock content %}
19     </body>
20 </html>
```

Template

In VS Code, create a new template file called **post_delete.html** and enter the following code:

Code

```
templates > <> post_delete.html > ...
1  {% extends 'base.html' %}
2  {% block title %} Delete Post Page {% endblock title %}
3
4  {% block content %}
5      <h1>Delete Post</h1>
6      <form action="" method="POST">
7          {% csrf_token %}
8          <p>Are you sure you want to delete "{{ post.title }}"?</p>
9          <input type="submit" value="Confirm" />
10     </form>
11 {% endblock content %}
```

URLs

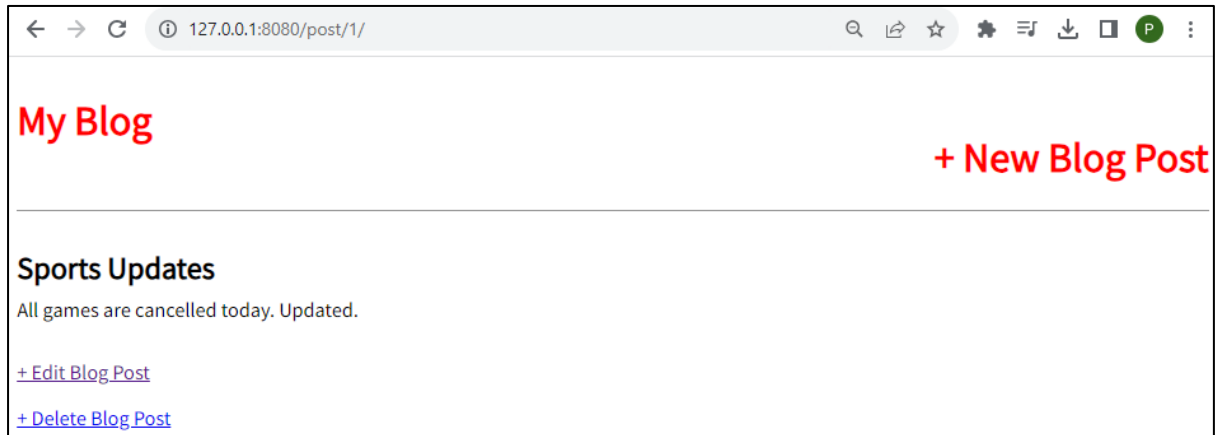
Open **blog/urls.py** and add the following code to create a URL

Code

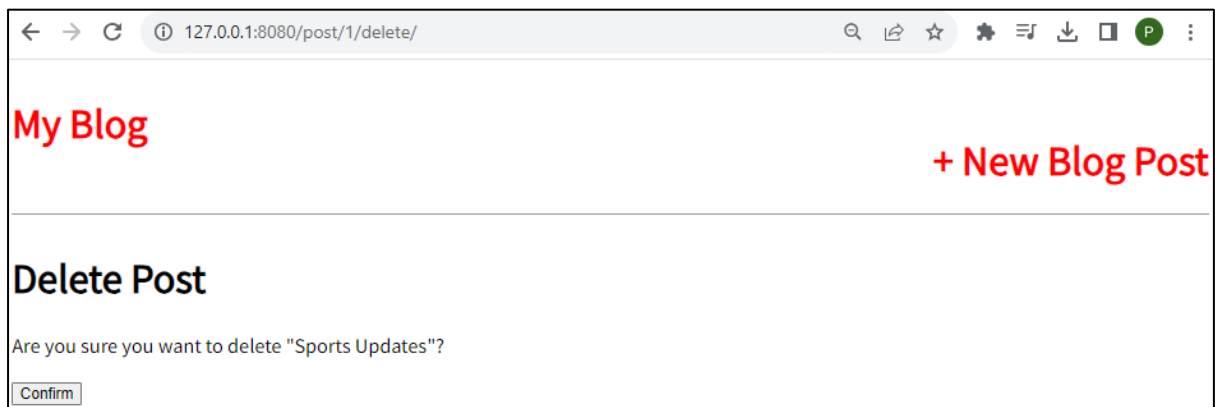
```
blog > urls.py
1  from django.urls import path
2  from .views import BlogListView, BlogDetailView, BlogCreateView, BlogUpdateView, BlogDeleteView
3
4  urlpatterns = [
5      path('', BlogListView.as_view(), name='home'),
6      path('post/<int:pk>/', BlogDetailView.as_view(), name='post_detail'),
7      path('post/new/', BlogCreateView.as_view(), name='post_new'),
8      path('post/<int:pk>/edit/', BlogUpdateView.as_view(), name='post edit'),
9      path('post/<int:pk>/delete/', BlogDeleteView.as_view(), name='post_delete'),
10 ]
```

Run server

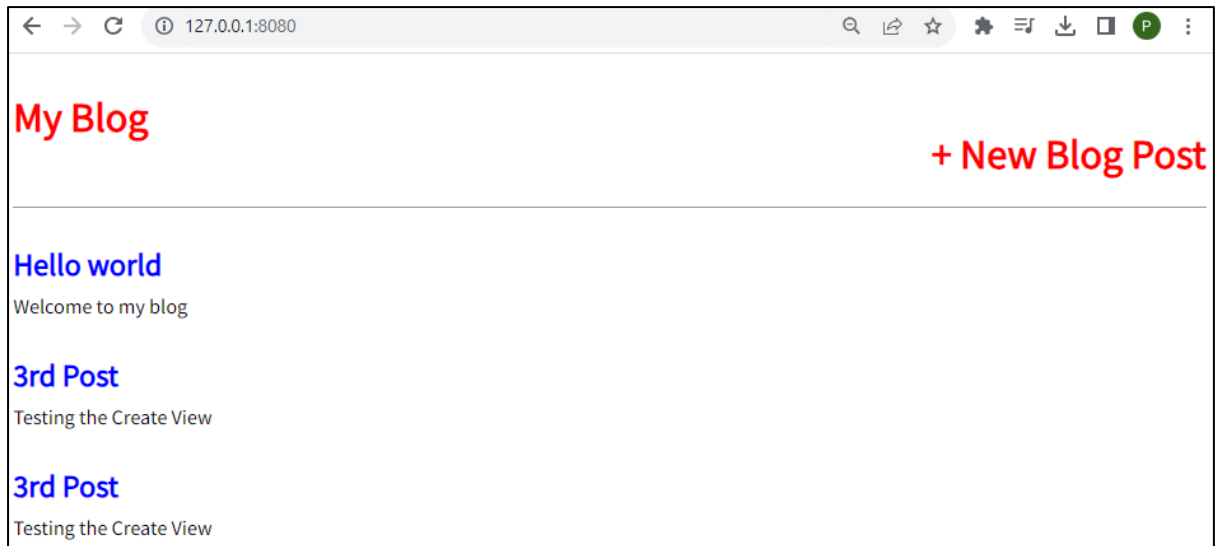
1. Start the server again using `python manage.py runserver` and refresh the individual post page you will see our “Delete Blog Post” link.



2. Click the link and it takes you to the delete page for the blog post, which displays the name of the blog post.



3. Click on the “**Confirm**” button and you are redirected you to the homepage where the blog post has been deleted!



Run the following git commands to update the local and remote repositories:

git add -A

git commit -m "lab 4 part 2 commit"

git push -u origin main