**Lab 5 Part (4)**

**Add a new app to our Newspaper Project**

In this exercise we will continue working on our Newspaper project to create a new app called articles where users can view, create, update, and delete articles. Bootstrap will also be used to style the pages.

**Articles app**

Start by creating a new articles app.

**Command Line**

(env) djangoprojects\lab5> python manage.py startapp articles

**Settings**

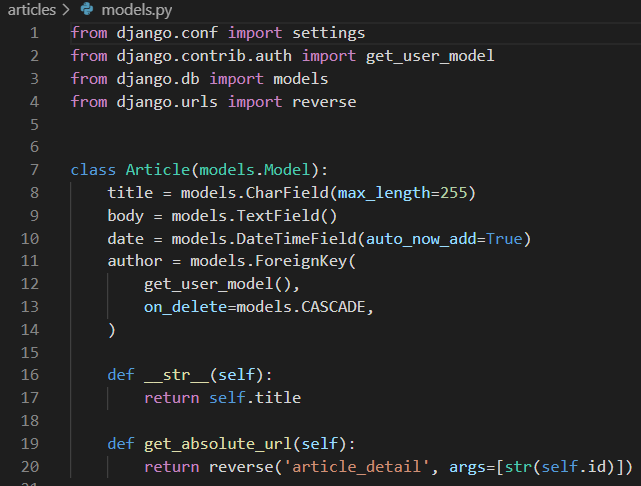
Open settings.py and register the articles app

**Database Model**

Next up we define our database model which contains four fields: title, body, date, and author. Note that we are letting Django automatically set the time and date based on the TIME\_ZONE setting in settings.py. For the author field we want to reference our custom user model 'users.CustomUser' which we set in the settings.py file as AUTH\_USER\_MODEL.

We can do this via get\_user\_model. And we also implement the best practices of defining a get\_absolute\_url from the beginning and a \_\_str\_\_ method for viewing the model in our admin interface.

**Code**

Next we need to make a new migration file and then apply it to the database.

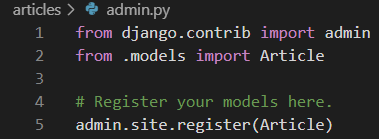
**Command Line**

(env) djangoprojects\lab5> python manage.py makemigrations articles

(env) djangoprojects\lab5> python manage.py migrate

**Admin**

We need to update admin.py so our new app is displayed.



Now we start the server.

**Command Line**

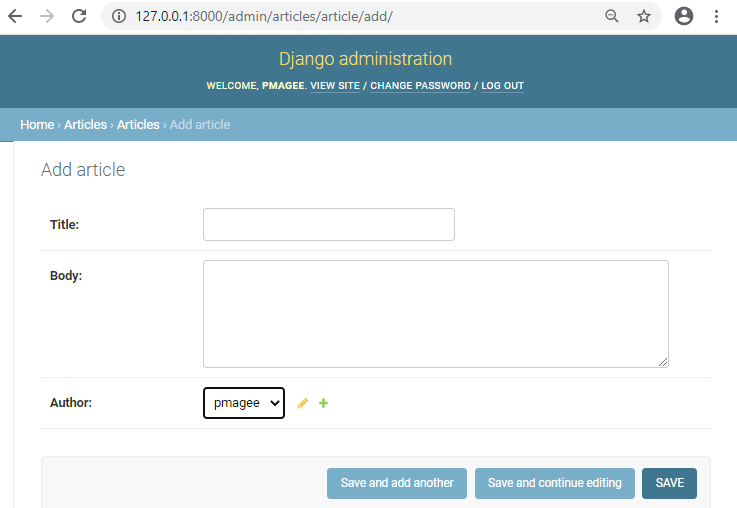
(env) djangoprojects\lab5> python manage.py runserver

Navigate to http://127.0.0.1:8000/admin/ and log in.

**Admin page**

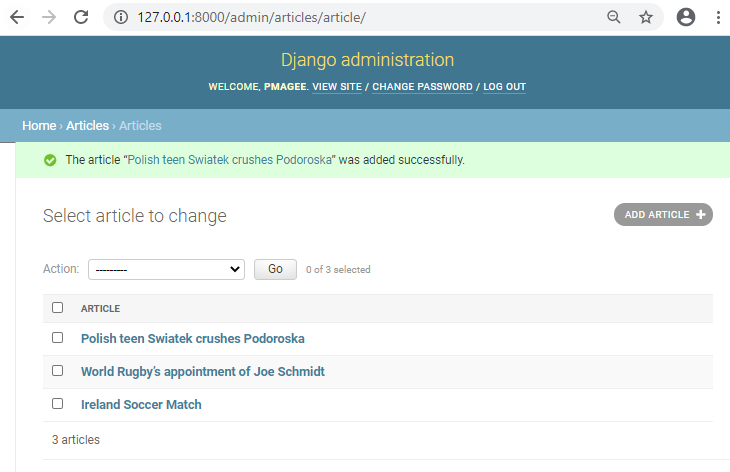
Click on “+ Add” next to “Articles” at the top of the page to create 3 articles and enter in some sample data. You will likely have three users available at this point: your superuser, and the two other accounts you created in an earlier exercise. Use your superuser account as the author of all three articles.

**Admin articles add page**



I have added three new articles as you can see on the updated Articles page.

**Admin three articles**



If you click on an individual article you will see that the title, body, and author are displayed but not the date. That is because the date was automatically added by Django for us and therefore cannot be changed in the admin.

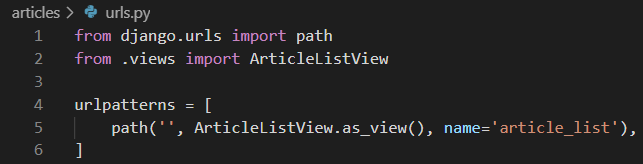
Even though date is not displayed here we will still be able to access it in our templates so it can be displayed on web pages.

**URLs and Views**

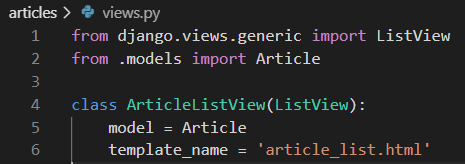
The next step is to configure our URLs and views. We will configure out urls so that our articles appear at articles/. Add a URL pattern for articles at line 10 in our newspaper\_project/urls.py file as shown below:



Next create an articles/urls.py file.



Now create the view using the built-in generic ListView from Django.



The only two fields we need to specify are the model Article and our template name

which will be article\_list.html.

**Template**

Create a template inside the templates folder called article\_list.html.

Bootstrap has a built-in component called Cards that we can customize for our individual articles. Recall that ListView returns an object called object\_list which we can iterate over using a for loop.

Within each article we display the title, body, author, and date. We will also provide links to “edit” and “delete” functionality that we will create later on.

The html code is available here for you to copy and paste.

**Code**

{% extends 'base.html' %}

{% block title %}Articles{% endblock title %}

{% block content %}

{% for article in object\_list %}

<div class="card">

<div class="card-header">

<span class="font-weight-bold">{{ article.title }}</span> &middot;

<span class="text-muted">by {{ article.author }} | {{ article.date }}</span>

</div>

<div class="card-body">

{{ article.body }}

</div>

<div class="card-footer text-center text-muted">

<a href="">Edit</a> |

<a href="">Delete</a>

</div>

</div>

<br />

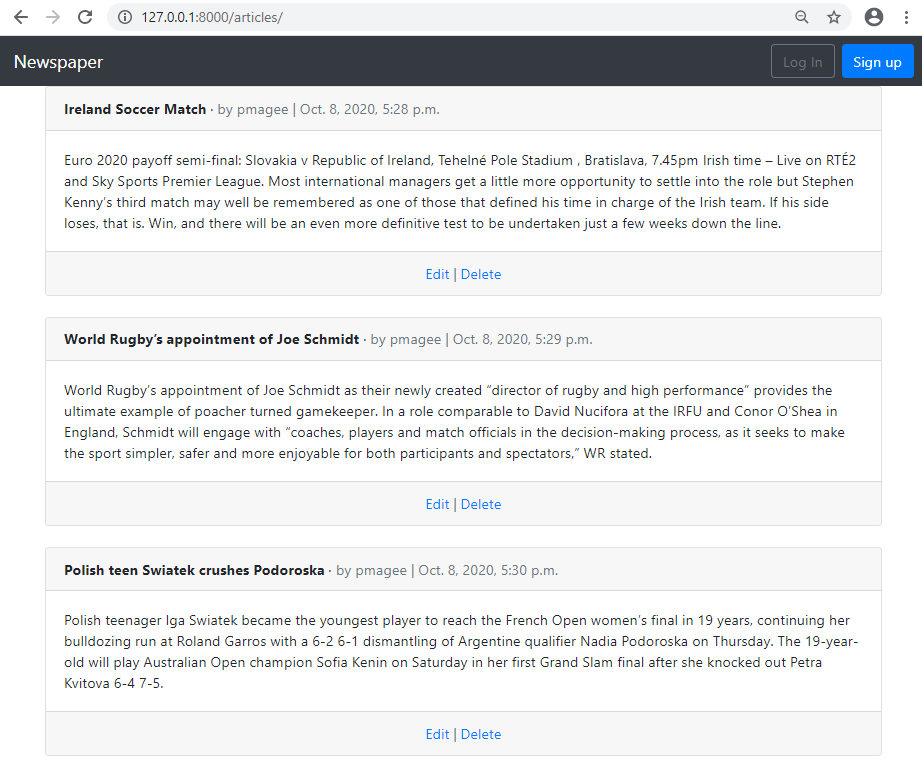
{% endfor %}

{% endblock content %}

Start the server again with python manage.py runserver and check out the new page at http://127.0.0.1:8000/articles/.

**Articles page**

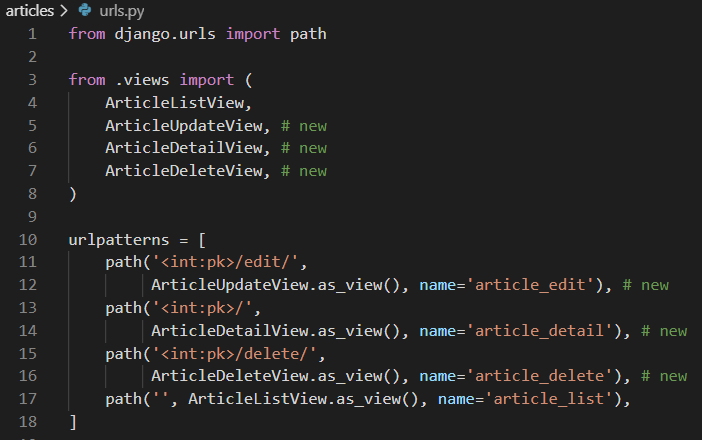
The Bootstrap Card feature has certainly made the page look good. Notice also that the date is included.



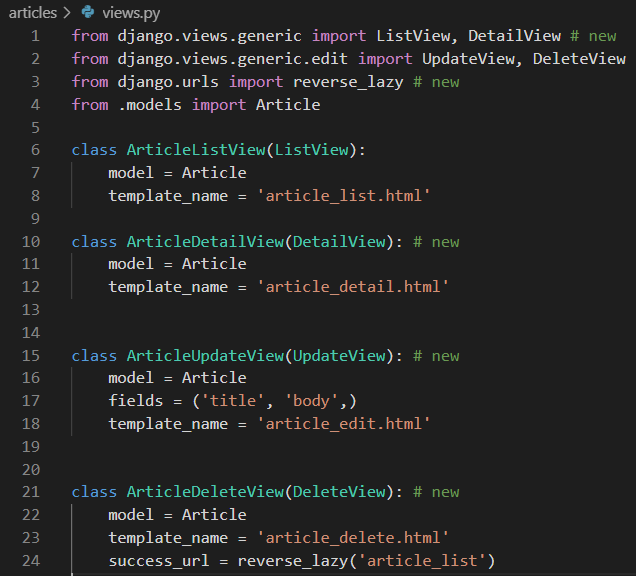
**Edit/Delete**

How do we add edit and delete options? We need new urls, views, and templates. Let’s start with the urls. We can take advantage of the fact that Django automatically adds a primary key to each database. Therefore our first article with a primary key of 1 will be at articles/1/edit/ and the delete route will be at articles/1/delete/.

Add the following code to articles/urls.py. Note the addition of the brackets around the imports for the different views:



Next create the views which will use Django’s generic class-based views for DetailView, UpdateView and DeleteView. We specify which fields can be updated–title and body–and where to redirect the user after deleting an article: article\_list.



Finally we need to add our new templates. Create the files article\_edit.html, article\_detail.html, and article\_delete.html inside the templates folder. The code for these templates is provided here for you to copy and paste.

article\_detail.html

{% extends 'base.html' %}

{% block content %}

<div class="article-entry">

<h2>{{ object.title }}</h2>

<p>by {{ object.author }} | {{ object.date }}</p>

<p>{{ object.body }}</p>

</div>

<p><a href="{% url 'article\_edit' article.pk %}">Edit</a> |

<a href="{% url 'article\_delete' article.pk %}">Delete</a></p>

<p>Back to <a href="{% url 'article\_list' %}">All Articles</a>.</p>

{% endblock content %}

article\_edit.html

{% extends 'base.html' %}

{% block content %}

<h1>Edit</h1>

<form action="" method="post">{% csrf\_token %}

{{ form.as\_p }}

<button class="btn btn-info ml-2" type="submit">Update</button>

</form>

{% endblock content %}

article\_delete.html

{% extends 'base.html' %}

{% block content %}

<h1>Delete</h1>

<form action="" method="post">{% csrf\_token %}

<p>Are you sure you want to delete "{{ article.title }}"?</p>

<button class="btn btn-danger ml-2" type="submit">Confirm</button>

</form>

{% endblock content %}

Next we need to add the edit and delete links to the article\_list.html page as shown below:

article\_list.html

……………

<div class="card-footer text-center text-muted">

<a href="{% url 'article\_edit' article.pk %}">Edit</a> |

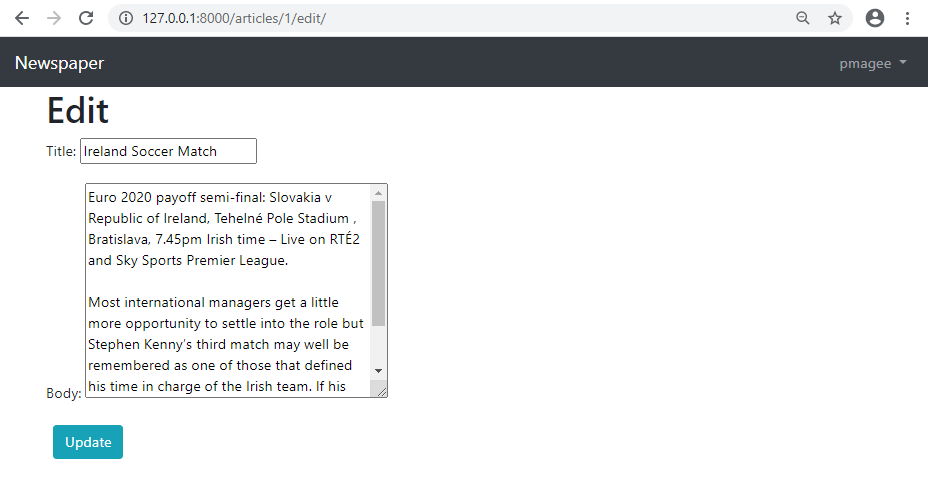
<a href="{% url 'article\_delete' article.pk %}">Delete</a>

</div>

………………

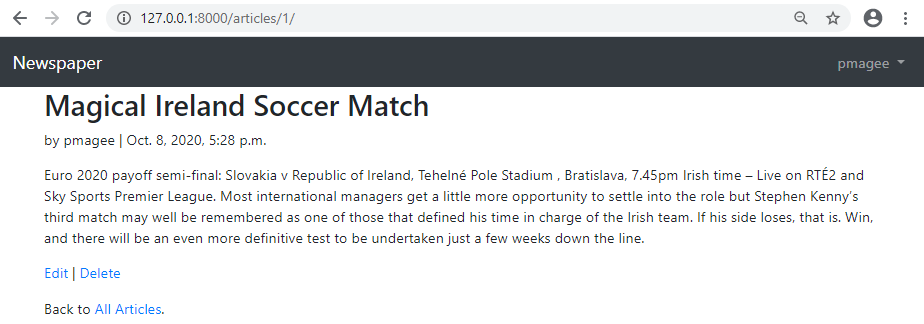
Start up the server with python manage.py runserver and navigate to articles page at http://127.0.0.1:8000/articles/. Click on the link for “edit” on the first article and you’ll be redirected to: <http://127.0.0.1:8000/articles/1/edit/> as shown here:

**Edit page**



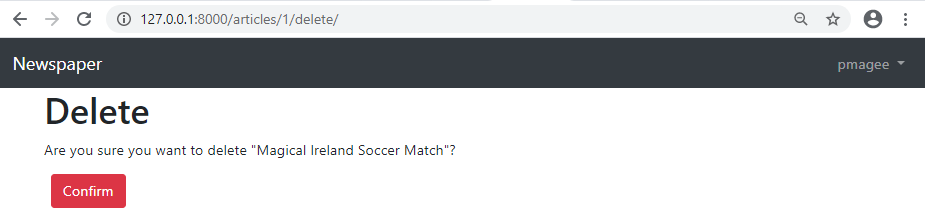
If you update the “title” field and click update you’ll be redirected to the detail page which shows the new change.

Detail page



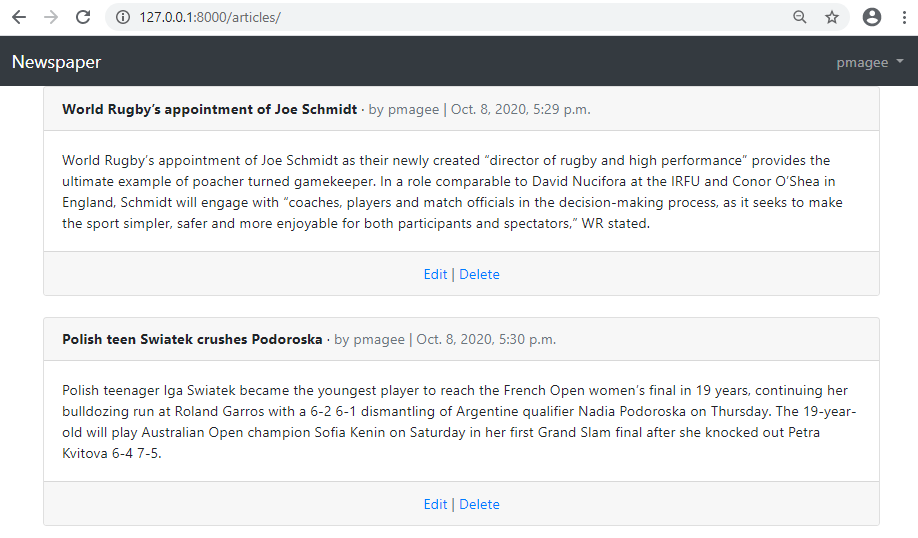
If you click on the “Delete” link you will be redirected to the delete page.

Delete page



Press the red button for “Delete” and you will be redirected to the articles page which now only has two entries.

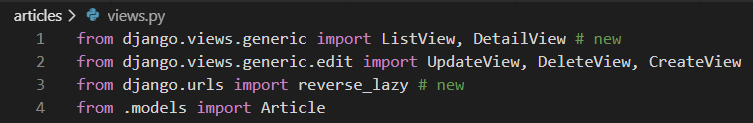
Articles page two entries

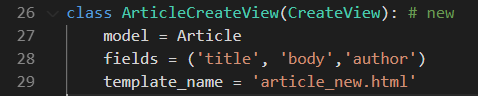


**Create page**

The final step is a create page for new articles which we can do with Django’s CreateView. Our three steps are to create a view, url, and template. These steps should feel familiar by now.

Add the following code to articles/views.py:





Update articles/urls.py with the following code:



Create a new template called article\_new.html and copy and paste the code here into it:

{% extends 'base.html' %}

{% block content %}

<h1>New article</h1>

<form action="" method="post">{% csrf\_token %}

{{ form.as\_p }}

<button class="btn btn-success ml-2" type="submit">Save</button>

</form>

{% endblock content %}

Finally add a link to creating new articles in the navbar so it is accessible everywhere on the site to logged-in users.

Open the file base.html and copy the block of code provided below and paste it just after line 14 as shown in the screen shot:



……………

{% if user.is\_authenticated %}

<ul class="navbar-nav mr-auto">

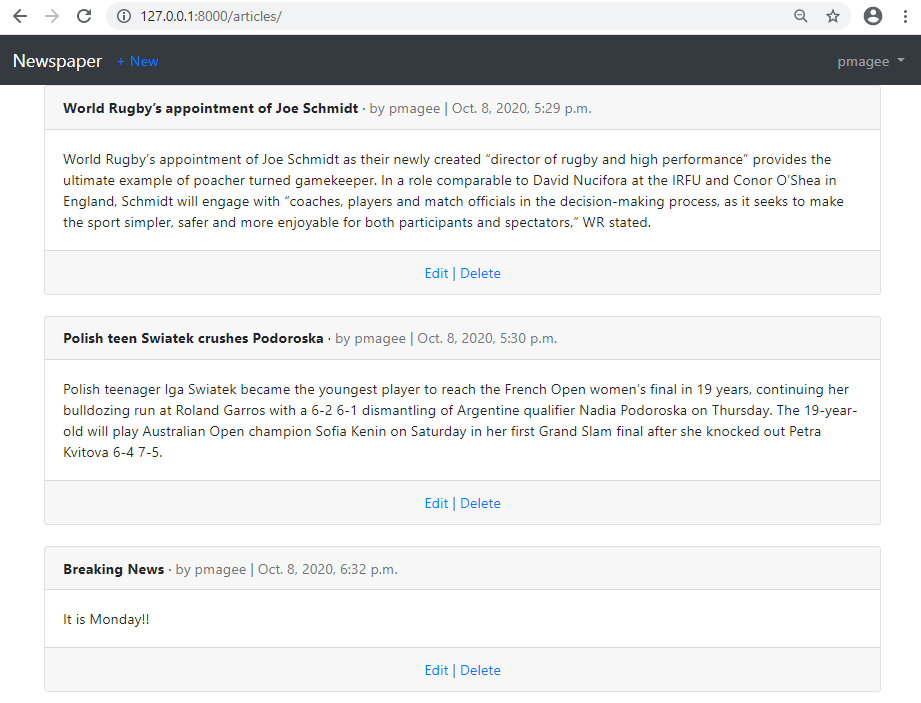
<li class="nav-item"><a href="{% url 'article\_new' %}">+ New</a></li>

</ul>

{% endif %}

………….

Refresh the articles page and the change is evident in the top navbar:



One last improvement to make is with the home page using Bootstrap.

Open the file home.html and replace the code with the code provided below:

{% extends 'base.html' %}

{% block title %}Home{% endblock title %}

{% block content %}

<br/>

<div class="jumbotron">

<h1 class="display-4">Newspaper app</h1>

<p class="lead">A Newspaper website built with Django.</p>

<p class="lead">

<a class="btn btn-primary btn-lg" href="{% url 'article\_list' %}"

role="button">View All Articles</a>

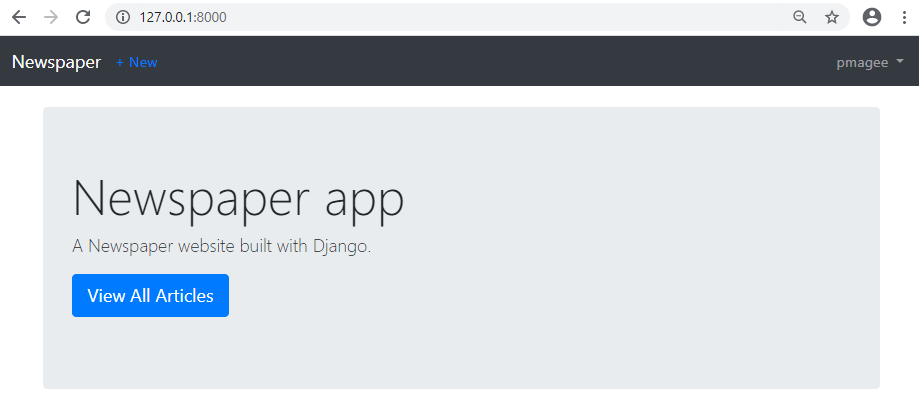
</p>

</div>

{% endblock content %}

Start up the server again python manage.py runserver and navigate to our homepage at: http://127.0.0.1:8000/.

Homepage with new link in nav



Click on the link for “+ New” in the top navbar and you will be redirected to the create

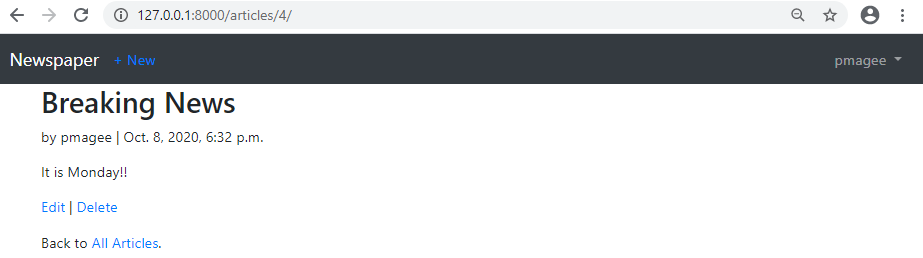
page.

**Create page**



Go ahead and create a new article. Then click on the “Save” button. You will be redirected to the detail page. Why? Because in our models.py file we set the get\_- absolute\_url method to article\_detail. This is a good approach because if we later change the url pattern for the detail page to, say, articles/details/4/, the redirect will still work. Whatever route is associated with article\_detail will be used; there is no hardcoding of the route itself.

**Detail page**



Note also that the primary key here is 4 in the URL. Even though we’re only displaying three articles right now, Django doesn’t reorder the primary keys just because we deleted one. In practice, most real-world sites don’t actually delete anything; instead they “hide” deleted fields since this makes it easier to maintain the integrity of a database and gives the option to “undelete” later on if needed. With our current approach once something is deleted it’s gone for good!

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

(env) djangoprojects\lab5>git add -A

(env) djangoprojects\lab5>git commit -m “lab 5 part 4 commit”

(env) djangoprojects\lab5>git push -u origin master