MIT AITI
Mobile, Python, Software Development

Django Blog Project
Part 5 – Forms

You will be adding forms to your blog app. These forms will allow you to add and edit comments. If you get stuck, take a look at these resources:

1. Lecture slides
2. Previous labs
3. Other group members
4. Django documentation
    a. Forms: https://docs.djangoproject.com/en/1.3/topics/forms/
    b. Forms from Models: https://docs.djangoproject.com/en/1.3/topics/forms/modelforms/
5. Google
6. Instructors

When is the last time you pushed to github?
Maybe you should do it now.

Part 1: Adding Comments

1. cd to your myblog django project.

2. Go to your templates/blog directory and open post_detail.html. Add this code after the template code that outputs all the comments

```
<h3>New comment:</h3>
<div>
    <form method='post' action="">
        {{ form.as_p }}
        <input type='submit' name='Submit'>
    </form>
</div>
```

3. At the top of your blog views.py, add the following lines:

```
from django.forms import ModelForm
from django.views.decorators.csrf import csrf_exempt
from django.http import HttpResponseRedirect
```

4. In views.py, right before your post_detail method, create a class called CommentForm. The model of this form should be Comment.
   Hint: use `class Meta:`

5. In views.py, update your post_detail definition to look like this:

```
@csrf_exempt
def post_detail(request, id, showComments=False):
```

   If we don't put in the @csrf_exempt, django will give us a security error. We may teach some information on CSRF later on, but for now we will just exempt our views.

6. Near the start of your `post_detail` method (below where you get the target post), insert the following code. Make sure you understand what is happening, and ask us for help if you find any of it confusing.

```
if request.method == 'POST':
    form = CommentForm(request.POST)
    if form.is_valid():
        form.save()
    return HttpResponseRedirect(request.path)
else:
    form = CommentForm()
```

7. There is one more change to make before your comment form will be displayed. In `post_detail`, update the context that is used to render the template to include the form by adding the key-value pair `{ 'form' : form }`

8. Start your server. Go to the website (http://localhost:8000/blog/posts/) and choose a post.

9. Add comments to your post and make sure they show up. Make sure you select a blog post from the list.

10. Right now, this works, but users shouldn't have to manually choose a blog post! We can fix that by excluding the post field from the form and automatically choosing a post for a new comment:
    - In the `Meta` class of your `CommentForm` class, add this line: `exclude=['post']`
    - Insert the bold lines into your `post_detail`

```
if request.method == 'POST':
    comment = Comment(post=target_post)
    form = CommentForm(request.POST, instance=comment)
    if form.is_valid():
```

11. You should now have a working submission form that doesn't force the user to choose a post! Add some more comments to test it out.

Part 2: Editing Comments

1. Edit your `post_detail.html` template file and add the following Template code **inside** of the loop that iterates over `comments`. This will add an "Edit Comment" link for each comment on the detail page.

```
<div>
  <a href="/blog/comments/{{ comment.id }}/edit">
      Edit Comment
  </a>
</div>
```

   Note: we call the loop variable `comment`*, but if you called it something else, please do not type* `comment` *anyway!*

2. Now you have to make this link do something. Here's what you need to do:
   a. Set up a template called `edit_comment.html`
      (Hint: use your other templates as to guide you!)

   b. Write a view that will display the form. Make sure that you populate the form fields with the information that is stored in the database.
      (Hint: You should be able use a lot of the code from other views)

   c. When the user has properly submitted the form, it should redirect them to the blog page for that comment.
      (Hint: use HttpResponseRedirect (and this will be easier if you defined a get_absolute_url method!!))

      Note: If you get a CSRF verification error, did you put `@csrf_exempt`?

      Note: You should **not** have to create a new form class. Keep in mind that one of the main features of python and django is code reuse.

   d. Edit your `urls.py` to include the route to your view

3. Test and make sure you can edit a comment!
   NICE JOB!!!!!

Now push to github and heroku