MIT AITI
Django Lab 6: Forms

In this lab we will add a form to let site visitors post comments on your blog. Read the whole lab and reference section before starting.

Django resources:
Forms: docs.djangoproject.com/en/1.5/topics/forms/
Forms from Models: docs.djangoproject.com/en/1.5/topics/forms/modelforms/

1. Change directory to your myblog project.

2. When someone goes to a page for an individual blog post, we want them to be able to submit a comment by writing their comment in a form. First we will add HTML to display the form. Go to your templates/blog directory and open detail.html. Add this code after the {% endfor %}

```
<br />
<div>New Comment</div>
<form method='post' action="">
    {{ form }}
    <input type='submit' name='Submit'>
</form>
```

This code creates a div that displays "New Comment", creates an HTML form, and prints out the form from the form variable that we will create in the view.

3. Now we will write the code to create the form in views.py. At the top of your views.py, add the following lines:

```
from django.views.decorators.csrf import csrf_exempt
from django.http import HttpResponse, HttpResponseRedirect
from django import forms
import datetime
```

4. In views.py, right before your `blog_detail` method, create a class called `CommentForm` that inherits from `forms.Form`

This should be a form that lets a user type in the body and the author of the comments. The form should have 2 fields: `body` and `author`. Think about what types of fields they should be. See the example and reference at the end of the lab.

5. In views.py, put this line:
   `@csrf_exempt`

right before the line
`def blog_detail(request,id,showComments=False):`

This tells Django to ignore security errors for now.

6. In your `blog_detail` method, insert the following code after `comments = Comment.objects.filter(post=blog)`. Pay attention to indentation! See below for an explanation of what the code is doing.

```
#start of form code
if request.method=="POST":
     form = CommentForm(request.POST)
     if form.is_valid():
          <your code here!>
          return HttpResponseRedirect(request.path)
else:
     form = CommentForm()
#end of form code
```

This code first checks if the form has been submitted, by checking if the form is trying to send data through a "POST" request. If the form has been submitted, the code gets the data from the form and checks if the form submission is valid (all fields filled in correctly). Then it says <your code here>, which you will replace with code saying what to do with the form data!
The else statement is called if the form has not been submitted, i.e. when someone just navigated to that page. In that case, it just displays a blank form.

Make sure you understand what is happening, and ask us for help if you find any of it confusing.

7. Now replace <your code here> with code that takes the form data and creates a Comment object from it. To create a Comment object, you must set all the fields that a Comment has. Check what those fields are in models.py.

Your code should:

- Get the body variable from the form
- Get the author variable from the form
- Set a current_time variable to be now (if you used date in your models: `datetime.date.today(),` if you used datetime: `datetime.datetime.now()`)
- Create a new Comment object called comment, with all of the fields that a Comment has. Think carefully about what each field should be. What should the value of `created` be? How about `post`?
- Save the comment object with `comment.save()`

8. There is one more change to make before your comment form will be displayed. In blog_detail, update the context to include the form:
```
c = Context({'blog':blog, 'comments':comments,
'form':form.as_p()})
```

9. Start your server. Go to the website (http://localhost:8000/blog/list) and choose a post. Add comments to your post and make sure they show up.

Forms reference

If your Django model uses a field, your Django form should use the same exact field type, except in the cases below:

If your Model field is…	Your form field should be…
`TextField`	`CharField(widget=forms.Textarea)`
`ForeignKey`	`ModelChoiceField`
`ManyToManyField`	`ModelMultipleChoiceField`

Example form:

```
class AuthorForm(forms.Form):
     name = forms.CharField(max_length=100)
     bio = forms.CharField(widget=forms.Textarea)
     birth_date = forms.DateField()
```

Creating a new Model instance from a form (this code goes in the view):

```
# Get the data from the form
name = form.cleaned_data[‘name’]
bio = form.cleaned_data[‘bio]
birth_date = form.cleaned_data[‘birthdate’]

# Create a new Author object
author = Author(name=name, bio=bio, birth_date=birth_date)

# Save the object
author.save()
```