Forms and Authentication

Reindert-Jan Ekker

@rjekker http://nl.linkedin.com/in/rjekker

Summary

Authentication

- LoginView and LogoutView
- Login form
- Restricting access to views
- Redirect, named URLs

Invitations

- ModelForm: generate HTML form
- Show form and handle submit
- Validation
- Styling
- Views with an argument

Demo

Restricting access to views

- is_authenticated
- login_required

Redirection and named URLs

Demo

Login and logout

- Using built-in view classes
- Template for login form
- Configuration

Named Views

```
urlpatterns = [
    url(r'home$', home, name="player_home"),
    # ...
]


# example use in a template with {% url %}
<a href="{% url 'player_home' %}"> Player home <a>
# Don't forget the quotes around the name!
```

```
from django.contrib.auth.decorators import login_required


@login_required
def my_view(request):
    # etc
```

Reacting to User Login Status (1)

Decorate views with `@login_required`

Non-logged in users will be redirected to `LOGIN_URL`

Set `LOGIN_URL` in settings.py

```
url(r'logout$',
    LogoutView.as_view(),
    name="player_logout")


# in settings.py
LOGOUT_REDIRECT_URL=
    "tictactoe_welcome"
```

- Using the LogoutView in `urls.py`
- It's a class; call `as_view()` on it
- Don't forget to name it

- Redirect after logout
- Takes URL or view name

```
url(r'login$',
    LoginView.as_view(
        template_name="..."
    ),
    name="player_login")
```

- LoginView works similarly
- `as_view()` can take parameters
- LoginView needs a template

```
# in settings.py
LOGIN_REDIRECT_URL=
    "player_home"
```

- Redirect after login
- Takes URL or view name

```
def my_view(request):
    if request.user.is_authenticated:
        # etc
```

```
{% if user.is_authenticated %} You're logged in {% endif %}
```

Reacting to User Login Status (2)

User object has attribute `is_authenticated`

Use this in view logic or template

Template Tag: if

```
{% if user.is_authenticated %}
    <a href="{% url 'player_logout' %}"> Logout </a>
{% else %}
    <a href="{% url 'player_login' %}"> Login </a>
{% endif %}
```

Redirect

```
from django.shortcuts import redirect


def welcome(request):
    if request.user.is_authenticated:
        return redirect('player_home’)


# redirect can also take a URL as its argument
```

Demo

Forms: sending invitations

- New Model class: Invitation
- Run Migration
- Add a ModelForm class
- Add a View function
- Render the form with a Template

Forms: General Flow

Show the Form

HTML with form elements

Generated by ModelForm

HTTP GET

Submit Form Data

User clicks submit

HTTP POST

Validation

Redirect

If form validates correctly: show result page

Otherwise: show form again

Demo

Handling form submit

- View function: GET vs POST
- Validating user input
- Showing validation errors
- Redirect on success
- CSRF protection

```
from django.forms import ModelForm
from .models import Invitation


class InvitationForm(ModelForm):
    class Meta:
        model = Invitation

        exclude = ('from_user',
                   'timestamp')
```

- **ModelForm:**
  a form based on a Django Model

- **Inherit from** `ModelForm`
- **Inner class:** `Meta`
- **The model that we want to generate a form for**
- **Model fields to leave out of form**
- **Docs:** https://goo.gl/pgYwMq

Showing the form

```
def new_invitation(request):
    form = InvitationForm()
    return render(request,
                  "player/new_invitation_form.html",
                  {'form': form}
                  )
```

Form Template

```
<form method="post"
      action="{% url 'player_new_invitation' %}">
      {{ form }}
      {% csrf_token %}

      <button type="submit">Send the invitation</button>
</form>
```

Handling Form Submit

```
# in the view
if request.method == "POST":
    # This is a form submit
    # Does input validate? Then process, redirect
    # If not: show form with errors
else:
    # This is a GET: show the form
```

Form Validation

```
# in view, when method is POST
form = InvitationForm(request.POST)
if form.is_valid():
    form.save()
    return redirect(‘player_home’)
return render(request, “form_tmplt.html", {'form': form})
```

The Complete View

```
if request.method == "POST":
    invitation = Invitation(from_user=request.user)
    form = InvitationForm(instance=invitation,
                          data=request.POST)
    if form.is_valid():
        form.save()
        return redirect('player_home')
else:
    form = InvitationForm()
return render(request, “….html", {'form': form})
```

Demo

Accepting Invitations

- Showing pending invitations
- Form View without ModelForm
- View takes an argument

Get or 404

```
from django.shortcuts import get_object_or_404

from .models import Invitation

def accept_invitation(request, id):
    invitation = get_object_or_404(Invitation, pk=id)
```

```
url(r'accept_invitation/(?P<id>\d+)/$',
        accept_invitation, name="player_accept_invitation")

# In HTML: using url tag with argument
<a href="{% url 'player_accept_invitation' id=inv.id %}">
```

Named Groups in URL Mappings

Use named groups in your expression to capture parts of URL

Syntax: `(?P<name>expr)`**, see** http://goo.gl/5uJsfy

Captured values are passed to view as keyword arguments

Demo

Tuning a form

- Helpful messages
- Styling with crispy-forms

Summary

Authentication

- LoginView and LogoutView
- Restricting access to views
- Named URLs, `{% url %}`, `redirect()`

Invitations

- ModelForm: form based on model
- Show form and handle submit
- Validation
- Named groups in URLs
- `get_or_404()`
- Styling