DJANGO BASIC VIEW

江琳

PHILOSOPHY

- **The following three classes provide much of the functionality needed to create Django views.**
  - **View**
  - **TemplateView**
  - **RedirectView**

VIEW

VIEW

- *class* **django.views.generic.base.View**
- **The master class-based base view.**
- **All other class-based views inherit from this base class. It isn't strictly a generic view and thus can also be imported from django.views.**

LIBRARY SYSTEM

- **Part one: Library System**
  - **Homepage**
  - **Book list page**
  - **Book detail page**
- **Part two: Library Management System**
  - **Login page**
  - **Create book, author, publisher, category**
  - **Book list**
  - **Update book**

LOGIN PAGE WITH VIEW

```
from django.contrib.auth import authenticate, login
from django.contrib import messages
from django.core.urlresolvers import reverse
from django.shortcuts import render, redirect
from django.views.generic import View
class Index(View):

    def get(self, request):
        return render(request, 'backstage/index.html')

    def post(self, request):
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(username=username, password=password)
        if user and user.is_superuser:
            login(request, user)
            return redirect(reverse('bs_list'))
        messages.error(request, 'username or password is
incorrect!')
        return redirect(reverse('index'))
```

LOGIN PAGE WITH VIEW

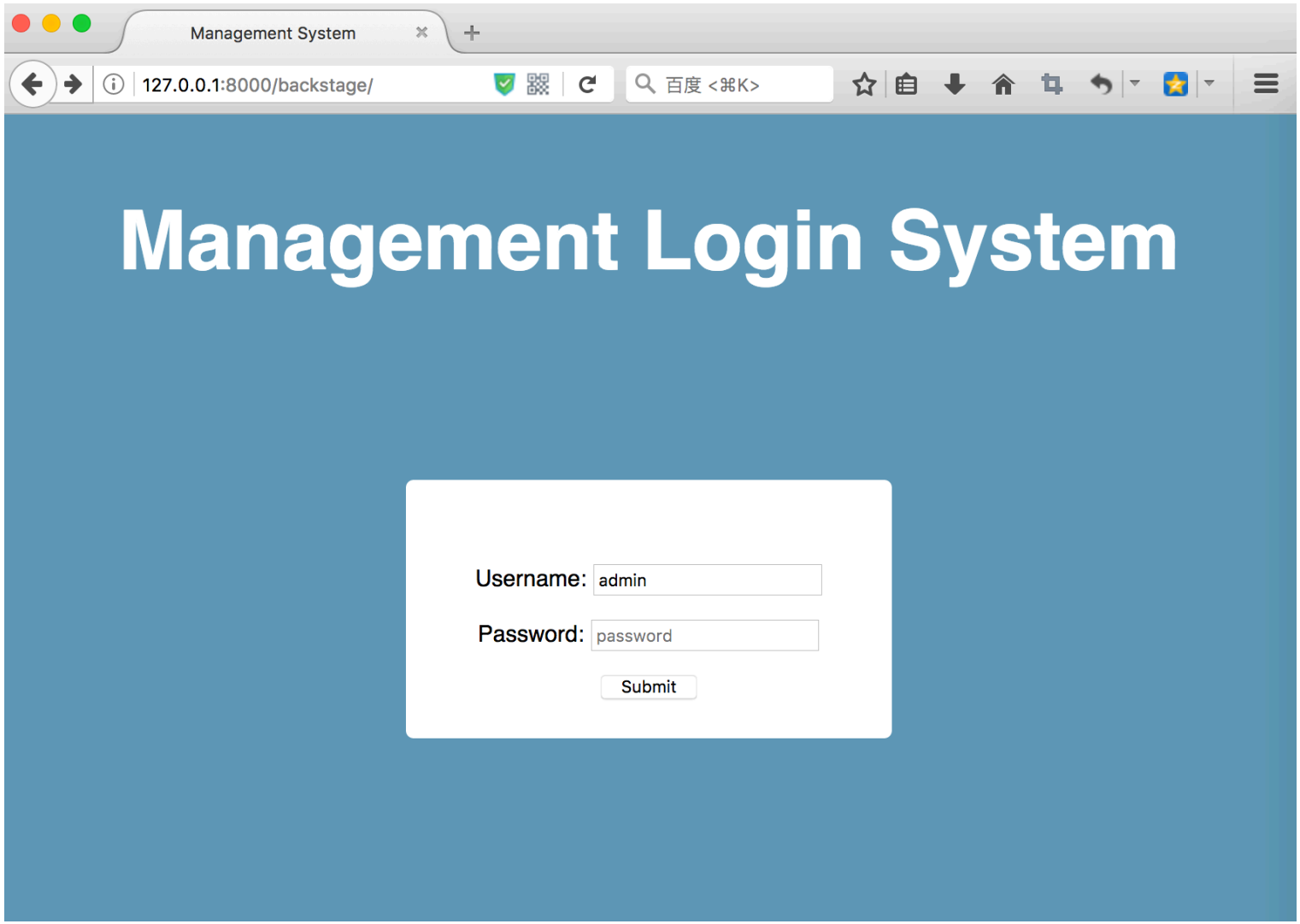
- **Add url**

```
url(r'^backstage/$', Index.as_view(),
name='index'),
```

```
<body>
<div class="title">Management Login System</div>
<div class="login">
  <form method="post">
    {% csrf_token %}
    <p style="height: 15px">
      {% for msg in messages %}
      <span class="{{ msg.tags }}">{{ msg }}</span>
      {% endfor %}
    </p>
    <p>
      <label>Username:</label>
      <input type="text" name="username">
    </p>
    <p>
      <label>Password:</label>
      <input type="password" name="password">
    </p>
    <input type="submit" value="Submit">
  </form>
</div>
</body>
```

```
<style>
    body {
        background-color: #609ab6;
        text-align: center;
    }
    .title {
        margin-top: 60px;
        color: #fff;
        font-size: 54px;
        font-weight: bold;
    }
    .login {
        text-align: center;
        margin: 15% auto;
        background-color: #fff;
        border-radius: 5px;
        width: 300px;
        height: 150px;
        padding: 10px;
    }
    .error {
        color: #ff0000;
    }
</style>
```

LOGIN PAGE WITH VIEW



LOGIN PAGE WITH VIEW

- **http method name**
- **The list of HTTP method names that view will accept:**
  - **['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace']**

LOGIN PAGE WITH VIEW

- **user = authenticate(username=username, password=password)**
- **Use authenticate() to verify a set of credentials. It takes credentials as keyword arguments, username and password for the default case, checks them against each authentication backend, and returns a User object if the credentials are valid for a backend. If the credentials aren't valid for any backend or if a backend raises PermissionDenied, it returns None.**

LOGIN PAGE WITH VIEW

- **login(request, user)**
- **login() is used to log a user in. It takes an HttpRequest object and a User object. login() saves the user's ID in the session, using Django's session framework.**

LOGIN PAGE WITH VIEW

- **return redirect(reverse('bs_list'))**
- **redirect() returns an HttpResponseRedirect to the appropriate URL for the arguments passed.**

LOGIN PAGE WITH VIEW

- **Messages tags**
  - **Message tags are a string representation of the message level plus any extra tags that were added directly in the view**

Level Constant	Tag
DEBUG	debug
INFO	info
SUCCESS	success
WARNING	warning
ERROR	error

{{ CSRF_TOKEN }}

- **The CSRF middleware provides easy-to-use protection against Cross Site Request Forgeries.**
- **This type of attack occurs when a malicious website contains a link, a form button or some JavaScript that is intended to perform some action on your website.**

TEMPLATEVIEW

TEMPLATEVIEW

- **Renders a given template, with the context containing parameters captured in the URL.**
- **Method flowchart**
  - **dispatch()**
  - **http_method_not_allowed()**
  - **get_context_data()**

LIBRARY SYSTEM

- **Part one: Library System**
  - **Homepage**
  - **Book list page**
  - **Book detail page**
- **Part two: Library Management System**
  - **Login page**
  - **Create book, author, publisher, category**
  - **Book list**
  - **Update book**

TEMPLATEVIEW

- **Let's modify Homepage using TemplateView**

```
class Home(TemplateView):

    template_name = 'homepage.html'

    def get_context_data(self, **kwargs):
        context = super(Home,
self).get_context_data(**kwargs)
        context['lastest_books'] = Book.objects.all().\
            order_by('-id')[:5]
        return context
```

HOMEPAGE.HTML

```
<body style="text-align: center">
<h1>Library System</h1>
<form method="get" action="{% url 'book_list' %}">
    <input type="text" style="width: 60%;height: 30px"
placeholder="searching" name="name">
    <input type="submit" value="Search">
</form>
<div class="new-books">
    <h2>New Books</h2>
    {% for book in lastest_books %}
    <p><a href="{% url 'book_detail' pk=book.id %}">
        <span style="font-size: 20px;font-weight:
bold">{{ book.name }}</span>
        <span>--{{ book.author.name }}</span></a></p>
    {% endfor %}
</div>
```

HOMEPAGE.HTML

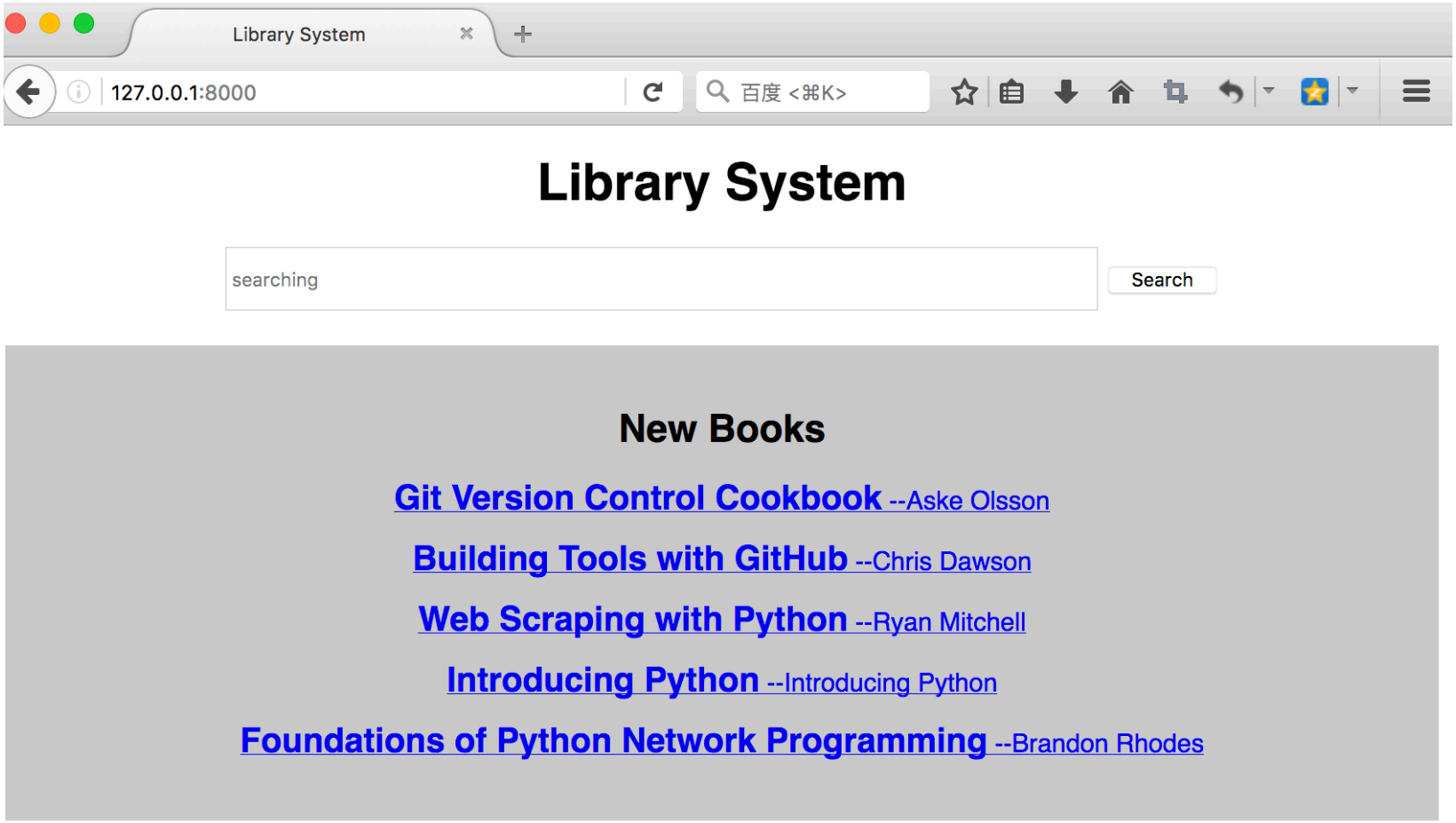
```
<style>
    .new-books{
        background-color: #cccccc;
        padding: 20px 0;
        margin-top: 20px;
    }
</style>
```

URL.PY

```
from books.views import Home

urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    url(r'^$', Home.as_view(), name='home'),
]
```

TEMPLATEVIEW



TEMPLATEVIEW

- **Something we don't pass context to template, we can write the TemplateView to url.py directly**

```
from django.views.generic import TemplateView


urlpatterns = [
    url(r'^$',
TemplateView.as_view(template_name='homepage.html
'), name='home'),
]
```

REDIRECTVIEW

REDIRECTVIEW

- **Redirects to a given URL.If the given URL is None, Django will return an HttpResponseGone (410).**
- **Method Flowchart:**
  - **dispatch()**
  - **http_method_not_allowed()**
  - **get_redirect_url()**

REDIRECTVIEW

- **Accume we change the url from**

```
url(r'^book/(?P<pk>\d+)/', BookDetail.as_view(),
name='book_detail'),
```

to

```
url(r'^detail/(?P<pk>\d+)/', BookDetail.as_view(),
name='book_detail'),
```

The old url will never usable, but some people don't know the change and may use it again.

Now we need redirect the old url to the new one.

VIEW.PY

```
from django.views.generic import RedirectView
class BookDetailRedirect(RedirectView):
    permanent = False
    query_string = True
    pattern_name = 'book_detail'

    def get_redirect_url(self, *args, **kwargs):
        return super(BookDetailRedirect,
self).get_redirect_url(*args, **kwargs)
```

URL.PY

```
urlpatterns = [
    url(r'^detail/(?P<pk>\d+)/', BookDetail.as_view(),
name='book_detail'),
    url(r'^book/(?P<pk>\d+)/',
BookDetailRedirect.as_view()),
```

REDIRECTVIEW

- **Now if you open the page http://127.0.0.1:8000/book/1, it will automatic redirect to http://127.0.0.1:8000/detail/1.**

REDIRECTVIEW -- ATTRIBUTES

- **url**
  - **The URL to redirect to, as a string. Or None to raise a 410 (Gone) HTTP error.**
- **pattern_name**
  - **The name of the URL pattern to redirect to.**

REDIRECTVIEW -- ATTRIBUTES

- **permanent**
  - **Whether the redirect should be permanent. The only difference here is the HTTP status code returned. If True, then the redirect will use status code 301. If False, then the redirect will use status code 302. By default, permanent is False.**
- **query_string**
  - **Whether to pass along the GET query string to the new location. If True, then the query string is appended to the URL. If False, then the query string is discarded. By default, query_string is False.**

REDIRECTVIEW

- **We can write the RediectView into url.py as well.**

```
from django.views.generic import RedirectView


urlpatterns = [
    url(r'^detail/(?P<pk>\d+)/', BookDetail.as_view(),
name='book_detail'),
    url(r'^book/(?P<pk>\d+)/',
RedirectView.as_view(pattern_name='book_detail')),
]
```

Questions?