DJANGO GENERIC EDITING VIEW

江琳

PHILOSOPHY

- **The edting views are used to provide a foundation for editing content.**
  - **django.views.generic.edit.CreateView**
  - **django.views.generic.edit.UpdateView**
  - **django.views.generic.edit.DeleteView**

REVIEW -- LISTVIEW

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**

REVIEW -- LISTVIEW

- **Let's using ListView to create the book list page.**

```
class BsList(ListView):
    '''
    backstage book list page
    '''

    template_name = 'backstage/bs_list.html'
    model = Book
```

URL.PY

Add url for book list page

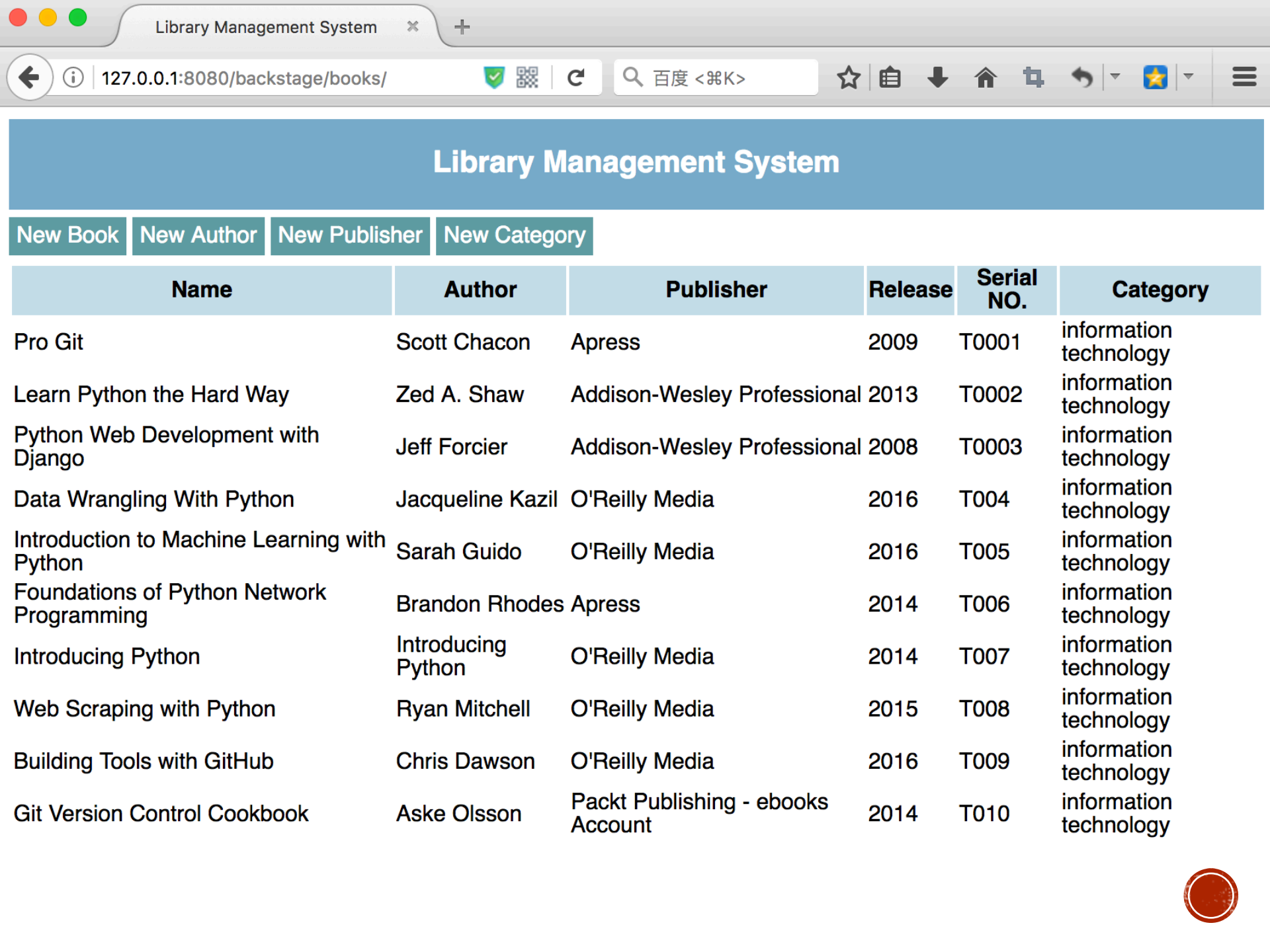
```
url(r'^backstage/books/$', BsList.as_view(),
name='bs_list'),
```

BS_LIST.HTML

```
<body>
<div class="head">Library Management System</div>
<div>
    <div style="padding: 10px 0">
        <a href="" class="btn">New Book</a>
        <a href="" class="btn">New Author</a>
        <a href="" class="btn">New Publisher</a>
        <a href="" class="btn">New Category</a>
    </div>
```

```
    <table style="width: 100%">
        <tr style="background-color: #c4dce8">
            <th>Name</th>
            <th>Author</th>
            <th>Publisher</th>
            <th>Release</th>
            <th>Serial NO.</th>
            <th>Category</th>
        </tr>
        {% for book in object_list %}
            <tr>
            <td>{{ book.name }}</td>
            <td>{{ book.author.name }}</td>
            <td>{{ book.publisher.name }}</td>
            <td>{{ book.pub_year }}</td>
            <td>{{ book.serial_num }}</td>
            <td>{{ book.category.name }}</td>
            </tr>
        {% endfor %}
    </table>
</div>
</body>
```

```
<head>
    <meta charset="UTF-8">
    <title>Library Management System</title>
    <style>
    .head{
        padding: 20px 0;
        color: #ffffff;
        font-size: 20px;
        font-weight: bold;
        text-align: center;
        background-color: #79aec8;
    }
    .btn{
        background-color: cadetblue;
        color: #ffffff;
        padding: 5px;
    }
    .btn:link{
        text-decoration: none;
    }
</style>
</head>
```



Library Management System

New Book | New Author | New Publisher | New Category

Name	Author	Publisher	Release	Serial NO.	Category
Pro Git	Scott Chacon	Apress	2009	T0001	information technology
Learn Python the Hard Way	Zed A. Shaw	Addison-Wesley Professional	2013	T0002	information technology
Python Web Development with Django	Jeff Forcier	Addison-Wesley Professional	2008	T0003	information technology
Data Wrangling With Python	Jacqueline Kazil	O'Reilly Media	2016	T004	information technology
Introduction to Machine Learning with Python	Sarah Guido	O'Reilly Media	2016	T005	information technology
Foundations of Python Network Programming	Brandon Rhodes	Apress	2014	T006	information technology
Introducing Python	Introducing Python	O'Reilly Media	2014	T007	information technology
Web Scraping with Python	Ryan Mitchell	O'Reilly Media	2015	T008	information technology
Building Tools with GitHub	Chris Dawson	O'Reilly Media	2016	T009	information technology
Git Version Control Cookbook	Aske Olsson	Packt Publishing - ebooks Account	2014	T010	information technology

CREATEVIEW

CREATEVIEW

- **A view that displays a form for creating an object, redisplaying the form with validation errors (if there are any) and saving the object.**
- **django.views.generic.edit.CreateView**

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**

CREATEVIEW

- Let's write a page used to create author using CreateView

```
class CreateAuthor(CreateView):
    model = Author
    template_name = 'backstage/create_author.html'
    fields = ['name', 'intro']
    def get_success_url(self):
        messages.success(self.request, 'Create Success!')
        return resolve_url('create_author')
```

URL.PY

```
url(r'^create/author/', CreateAuthor.as_view(),
name='create_author')
```

CREATE_AUTHOR.HTML

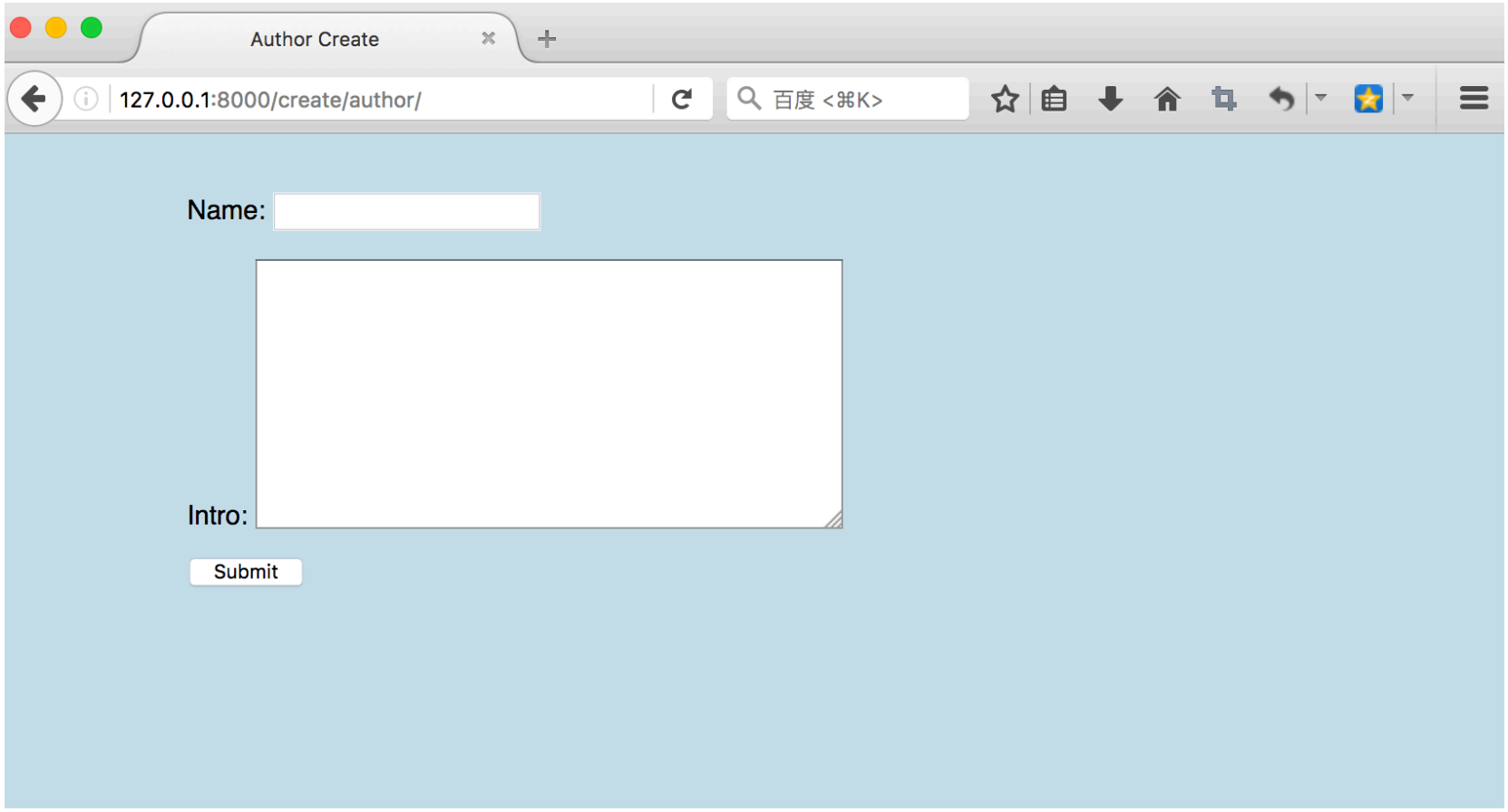
```
 <body>
{% for msg in messages %}
    <div class="{{ msg.tags }}">{{ msg }}</div>
{% endfor %}

<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Submit">
</form>
</body>
```

CREATE_AUTHOR.HTML

```
<head>
    <meta charset="UTF-8">
    <title>Author Create</title>
    <style>
        body{
            padding: 10px 100px;
            background-color: #c4dce8;
        }
        .success{
            background-color: #79aec8;
            padding: 10px;
            color: #ffffff;
        }
    </style>
</head>
```

CREATEVIEW


Author Create
127.0.0.1:8000/create/author/
百度 <⌘K>
Name:
Intro:
Submit

{{ FROM.AS_P }}

- {{ form }} will render

  <label> and <input> elements appropriately.

- {{ form.as_p }} will render them wrapped

  in <p> tags

- {{ form.as_ul }} will render them wrapped

  in <li> tags

CREATEVIEW

- **Let's create a author using this page**

UPDATEVIEW

UpdateView

- **A view that displays a form for editing an existing object, redisplaying the form with validation errors (if there are any) and saving changes to the object. This uses a form automatically generated from the object's model class.**
- ***class*** **django.views.generic.edit.UpdateView**

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**

UpdateView

```
from django.views.generic import UpdateView

class UpdateBook(UpdateView):
    model = Book
    template_name = 'backstage/create_book.html'
    fields = ['name', 'pub_year', 'cover', 'serial_num',
              'available', 'author', 'publisher',
              'category', 'intro']

    def get_success_url(self):
        messages.success(self.request, 'Create Success!')
        return resolve_url('update_book',
pk=self.kwargs['pk'])
```

URL.PY

```
from books.views import UpdateBook

urlpatterns = [
    url(r'^update/book/(?P<pk>\d+)/',
UpdateBook.as_view(), name='update_book'),
]
```

CREATE_BOOK.HTML

```
<body>
{% for msg in messages %}
    <div class="{{ msg.tags }}">{{ msg }}</div>
{% endfor %}

<form method="post" enctype="multipart/form-
data">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Submit">
</form>
</body>
```

CREATE_BOOK.HTML

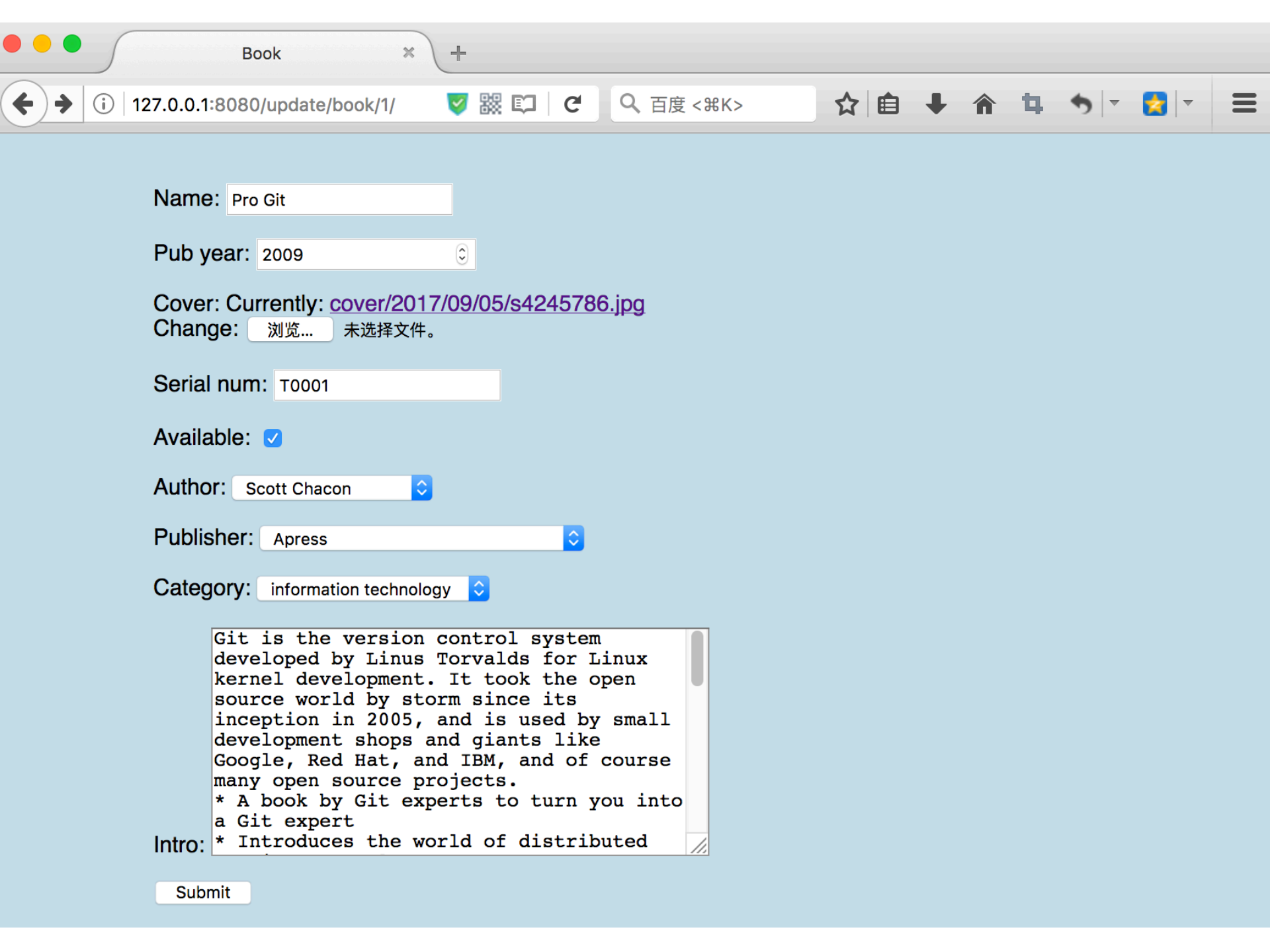
```
<head>
    <meta charset="UTF-8">
    <title>Book</title>
    <style>
        body{
            padding: 10px 100px;
            background-color: #c4dce8;
        }
        .success{
            background-color: #79aec8;
            padding: 10px;
            color: #ffffff;
        }
    </style>
</head>
```

EDIT BS_LIST.HTML

Library Management System

New Book | New Author | New Publisher | New Category

Name	Author	Publisher	Release	Serial NO.	Category	
Pro Git	Scott Chacon	Apress	2009	T0001	information technology	Edit
Learn Python the Hard Way	Zed A. Shaw	Addison-Wesley Professional	2013	T0002	information technology	Edit
Python Web Development with Django	Jeff Forcier	Addison-Wesley Professional	2008	T0003	information technology	Edit
Data Wrangling With Python	Jacqueline Kazil	O'Reilly Media	2016	T004	information technology	Edit
Introduction to Machine Learning with Python	Sarah Guido	O'Reilly Media	2016	T005	information technology	Edit
Foundations of Python Network Programming	Brandon Rhodes	Apress	2014	T006	information technology	Edit
Introducing Python	Introducing Python	O'Reilly Media	2014	T007	information technology	Edit
Web Scraping with Python	Ryan Mitchell	O'Reilly Media	2015	T008	information technology	Edit
Building Tools with GitHub	Chris Dawson	O'Reilly Media	2016	T009	information technology	Edit
Git Version Control Cookbook	Aske Olsson	Packt Publishing - ebooks Account	2014	T010	information technology	Edit


Book
127.0.0.1:8080/update/book/1/
百度 <⌘K>
Name: Pro Git
Pub year: 2009
Cover: Currently: cover/2017/09/05/s4245786.jpg
Change: 浏览... 未选择文件。
Serial num: T0001
Available:
Author: Scott Chacon
Publisher: Apress
Category: information technology
Intro: Git is the version control system developed by Linus Torvalds for Linux kernel development. It took the open source world by storm since its inception in 2005, and is used by small development shops and giants like Google, Red Hat, and IBM, and of course many open source projects.
* A book by Git experts to turn you into a Git expert
* Introduces the world of distributed
Submit

MEDIA FILE

MEDIA FILE

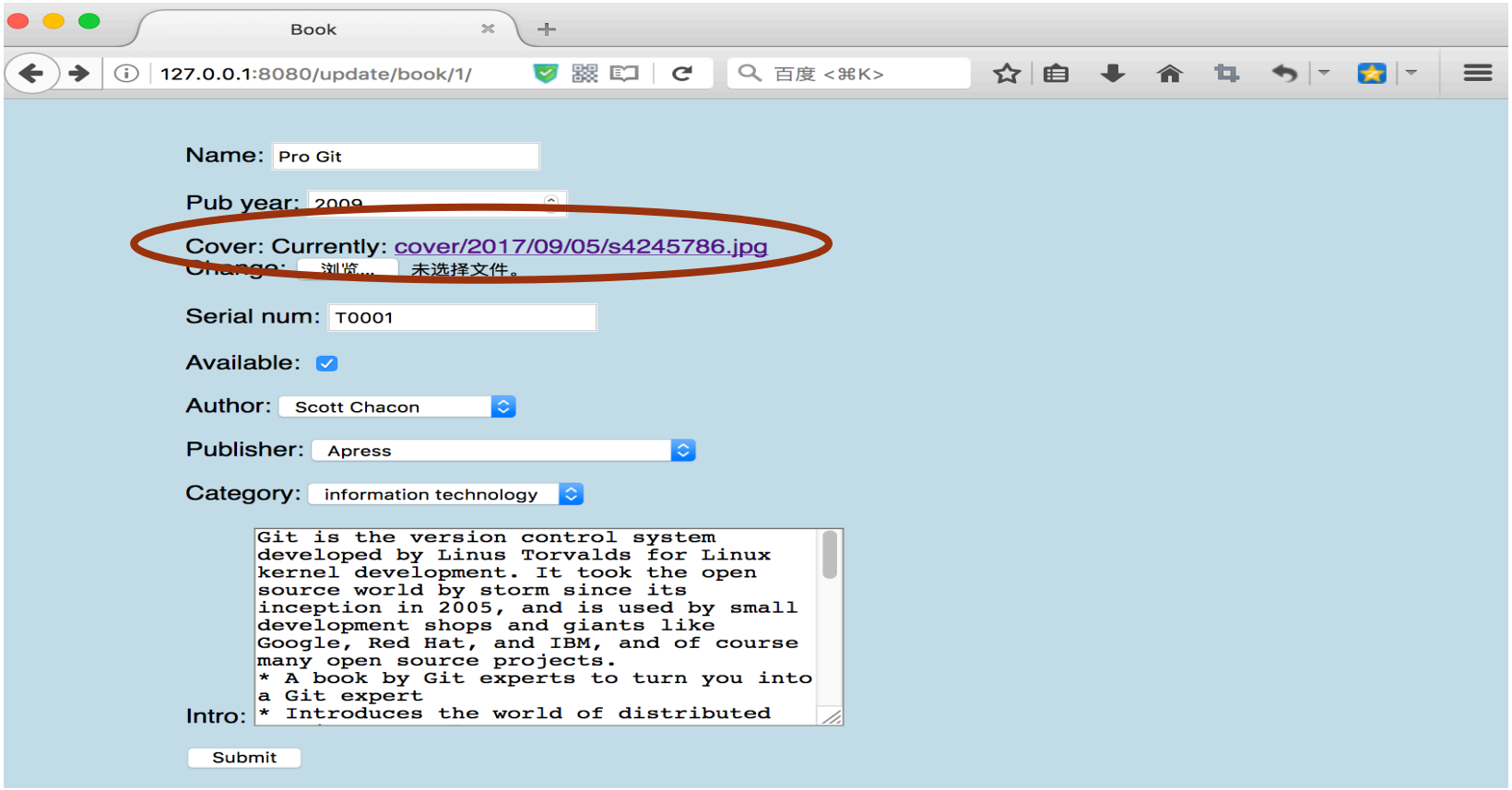
- **The file which user is uploaded will store in media file such as FildField and ImageField.**
- **Before we use it, we should set it up.**
- **Let's open the settings.py and configure it.**

SETTINGS.PY

```
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

- **This means the uploaded file will automaticlly store to a folder named media.**

SETTINGS.PY


Book
127.0.0.1:8080/update/book/1/
百度 <⌘K>
Name: Pro Git
Pub year: 2009
Cover: Currently: cover/2017/09/05/s4245786.jpg
Change: 浏览... 未选择文件。
Serial num: T0001
Available:
Author: Scott Chacon
Publisher: Apress
Category: information technology
Intro: Git is the version control system developed by Linus Torvalds for Linux kernel development. It took the open source world by storm since its inception in 2005, and is used by small development shops and giants like Google, Red Hat, and IBM, and of course many open source projects.
* A book by Git experts to turn you into a Git expert
* Introduces the world of distributed
Submit

SETTINGS.PY

- **Now the image store to media folder correctly.**
- **But if we click the link, we still can not open the image.**
- **So we need set the url for media up as well.**

URL.PY

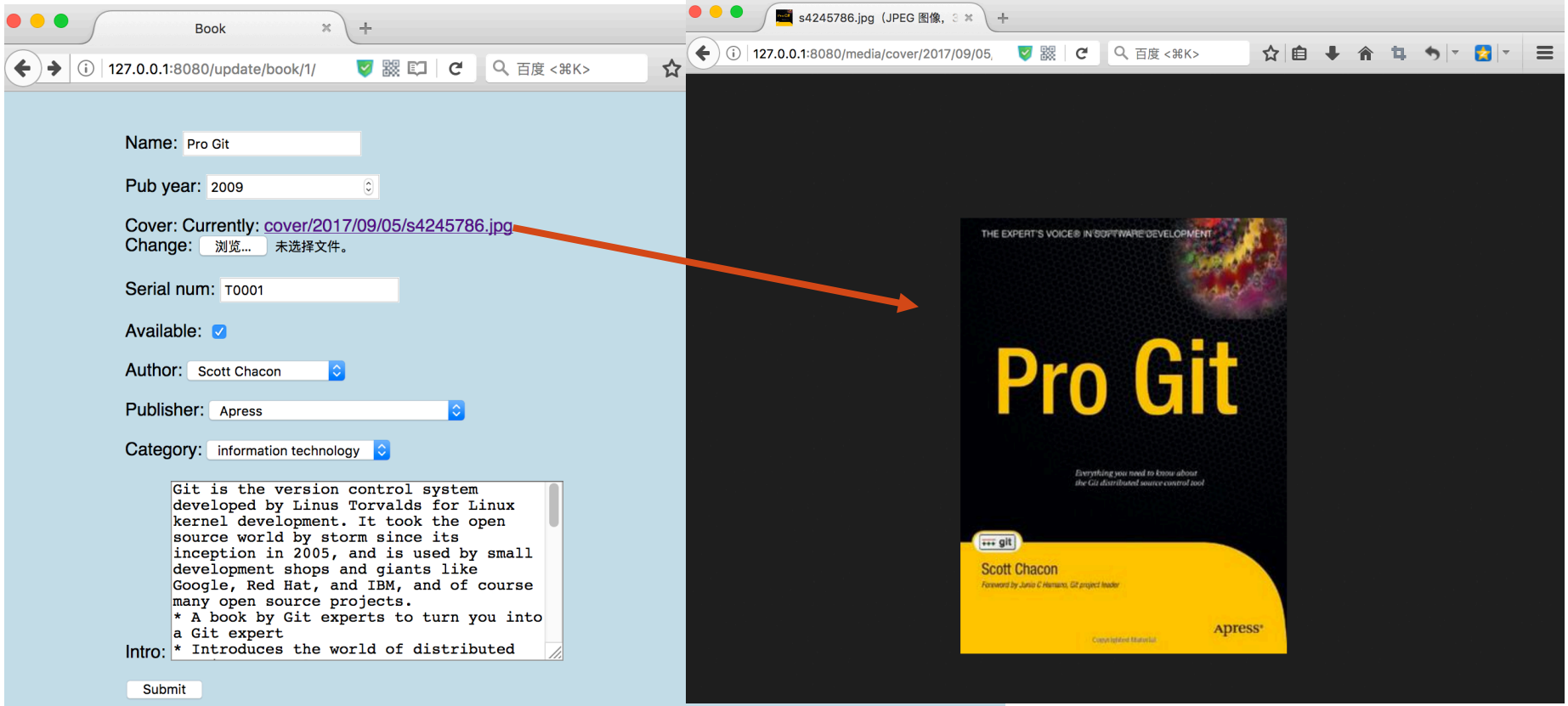
```
from django.conf import settings
from django.conf.urls.static import static

from books.views import UpdateBook

urlpatterns = [
    url(r'^update/book/(?P<pk>\d+)/',
UpdateBook.as_view(), name='update_book'),
] + static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

DONE

- **Now we can open the link to see the cover of the book.**



DELETEVIEW

DELETEVIEW

- **A view that displays a confirmation page and deletes an existing object. The given object will only be deleted if the request method is POST. If this view is fetched via GET, it will display a confirmation page that should contain a form that POSTs to the same URL.**
- ***class* django.views.generic.edit.DeleteView**

DELETEVIEW

```
from django.views.generic import DeleteView

class DeleteBook(DeleteView):
    model = Book
    template_name = 'backstage/delete_book.html'
    success_url = reverse_lazy('bs_list')
```

URL.PY

```
from books.views import UpdateBook, DeleteBook
urlpatterns = [
    url(r'^update/book/(?P<pk>\d+)/', UpdateBook.as_view(),
name='update_book'),
    url(r'^delete/book/(?P<pk>\d+)/', DeleteBook.as_view(),
name='delete_book'),
] + static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

DELETE_BOOK.PY

```
<body>
<form action="" method="post">{% csrf_token %}
    <p>Are you sure you want to delete
"{{ object.name }}"?</p>
    <input type="submit" value="Confirm" />
</form>
</body>
```

Questions?