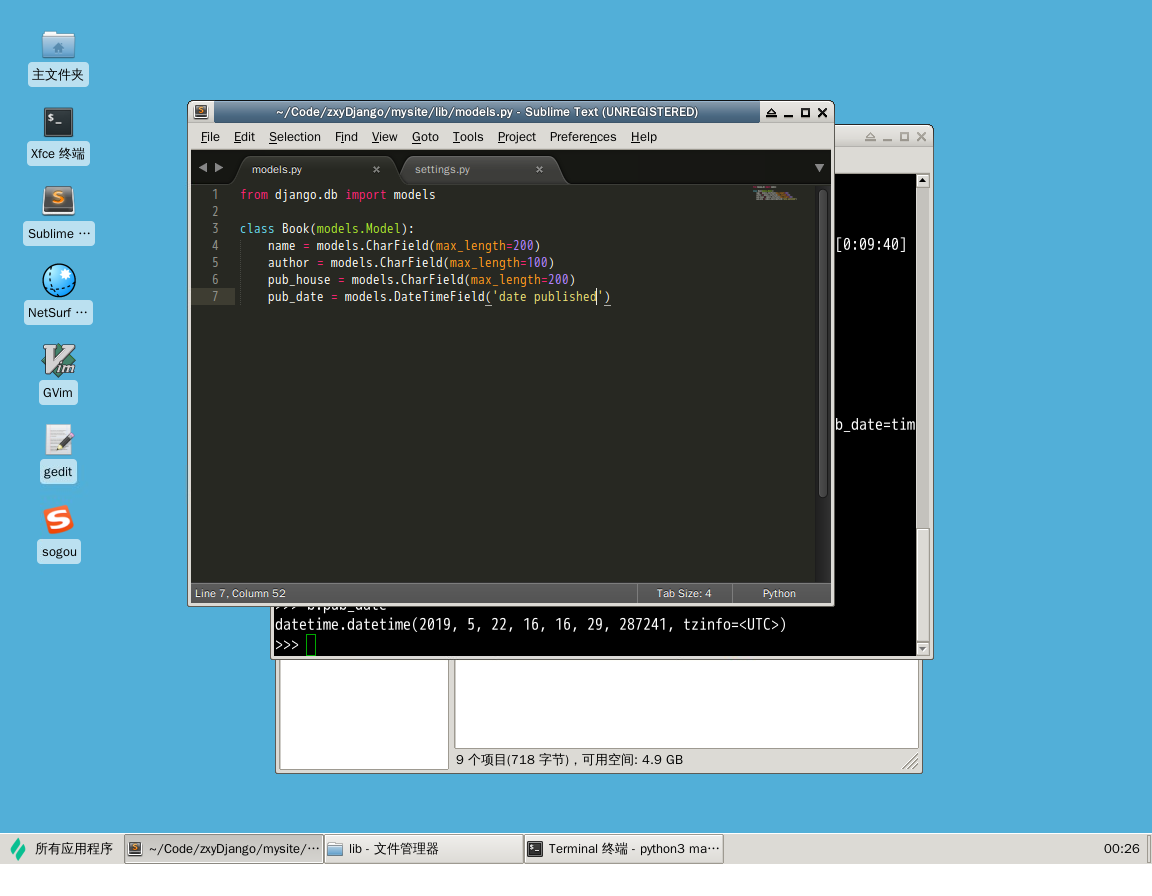
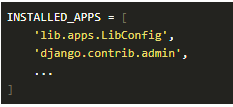
Django模型基础：

1编辑 models.py 文件，改变模型。



2将lib应用安装到项目中，因为LibConfig类写在文件lib/apps.py中，所以它的路径为lib.apps.LibConfig



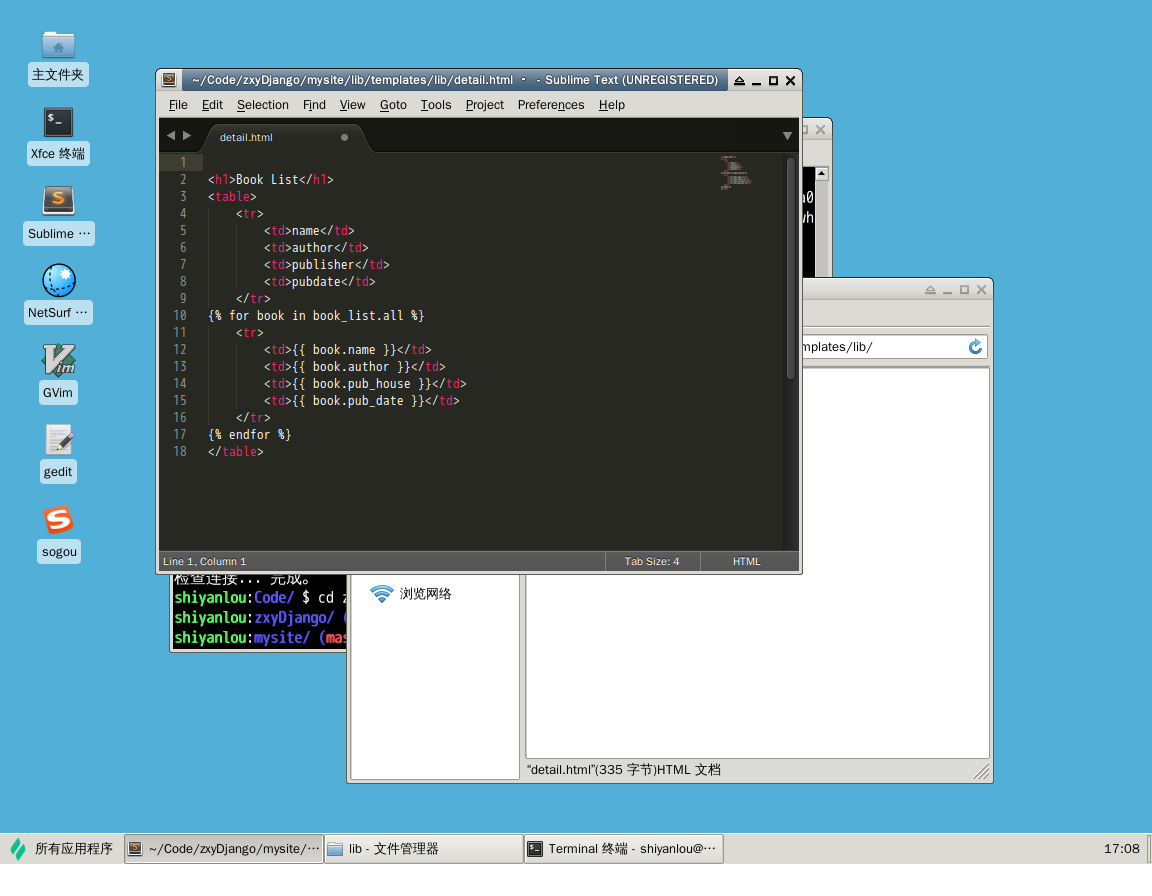
3. **运行 python manage.py makemigrations lib 为模型的改变生成迁移文件。**

**运行 python manage.py migrate 来应用数据库迁移**

Django模板

在 mysite/mysite/settings.py 文件中的 TEMPLATES 配置项描述了 Django 如何载入和渲染模板。默认的设置文件设置了 DjangoTemplates 后端，并设置 APP\_DIRS = True。这一选项将会让 DjangoTemplates 在每个 INSTALLED\_APPS 文件夹中寻找 templates 子目录。

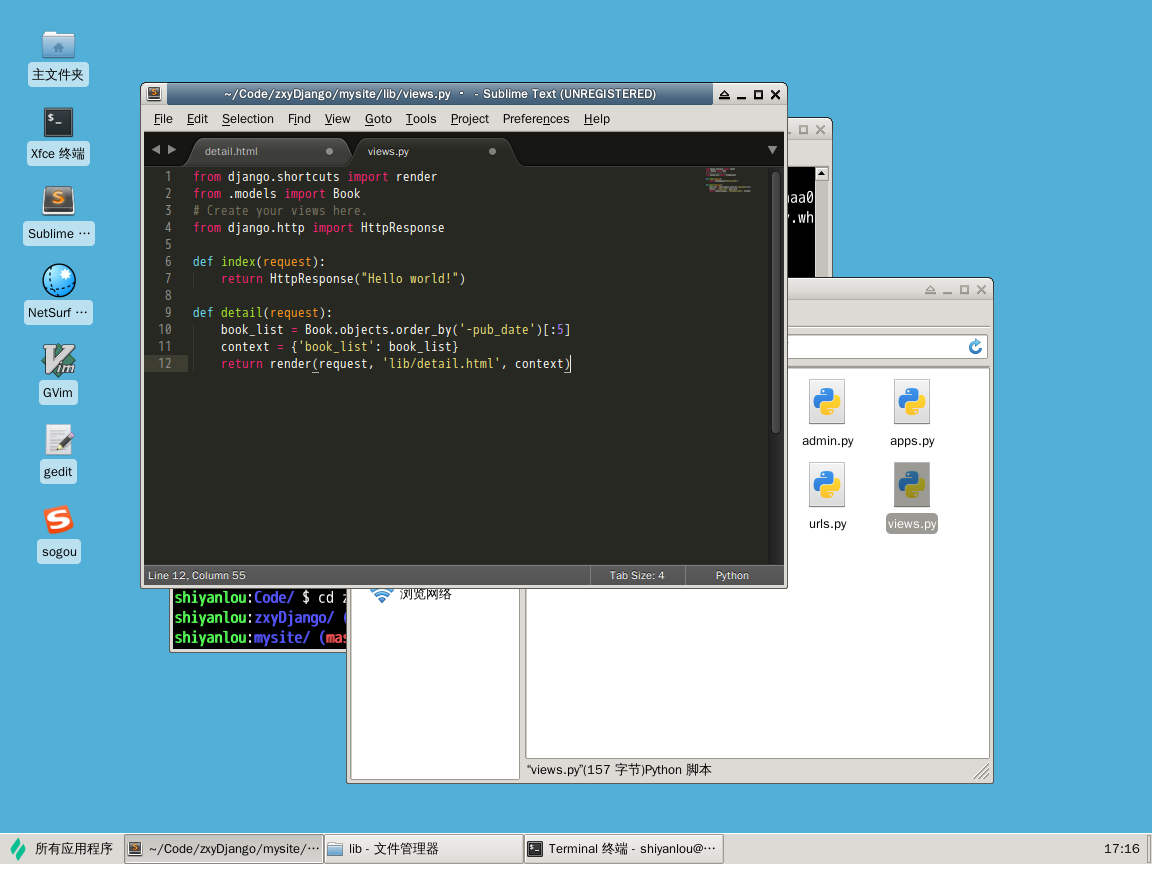
在lib/templates/lib 下新建模板



模板统一使用点符号.来访问变量的属性。在示例 {{ book.name }} 中，首先 Django 尝试对 book 对象使用字典查找（也就是使用 obj.get(str) 操作），如果失败了就尝试属性查找（也就是 obj.str 操作），结果是成功了。如果这一操作也失败的话，将会尝试列表查找（也就是 obj[int] 操作）。

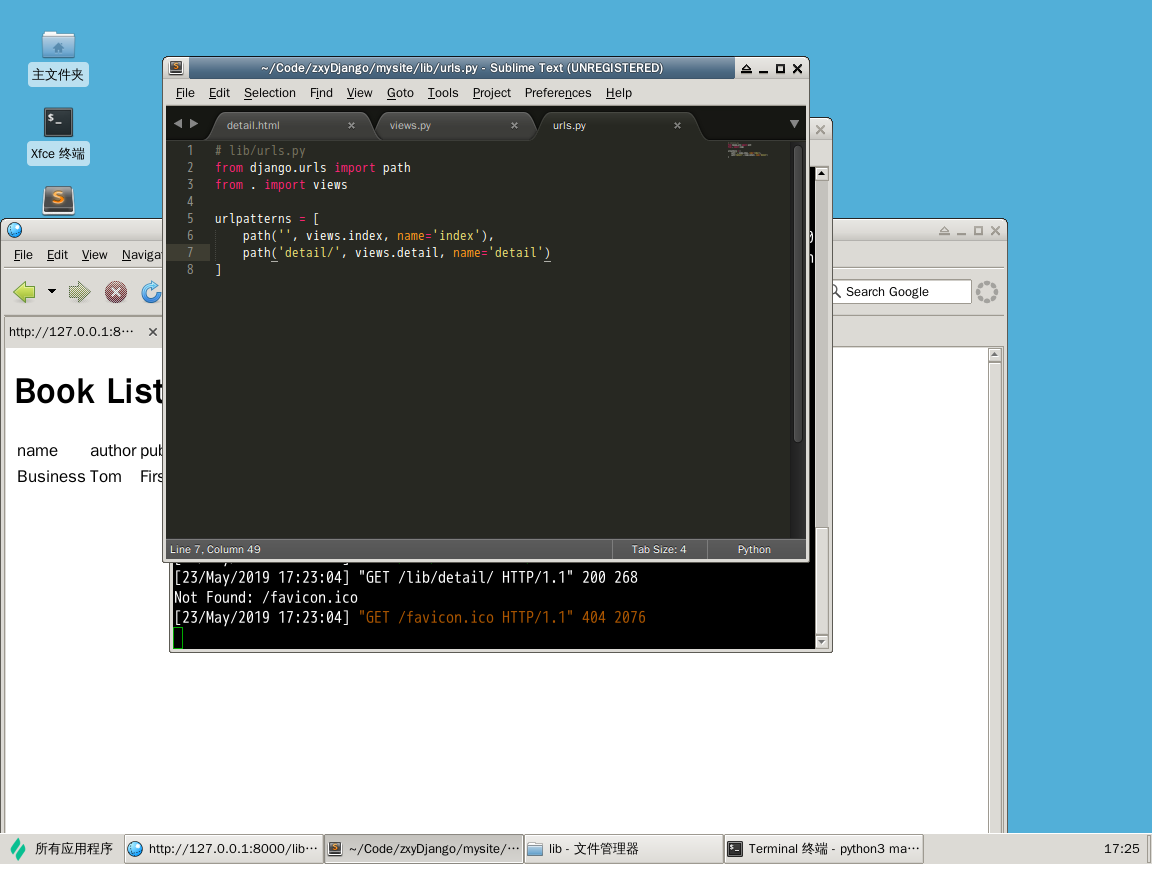
book\_list.all 被解释为 Python 代码 book\_list.objects.all() ，将会返回一个可迭代的 Book 对象

创建视图来返回图书列表



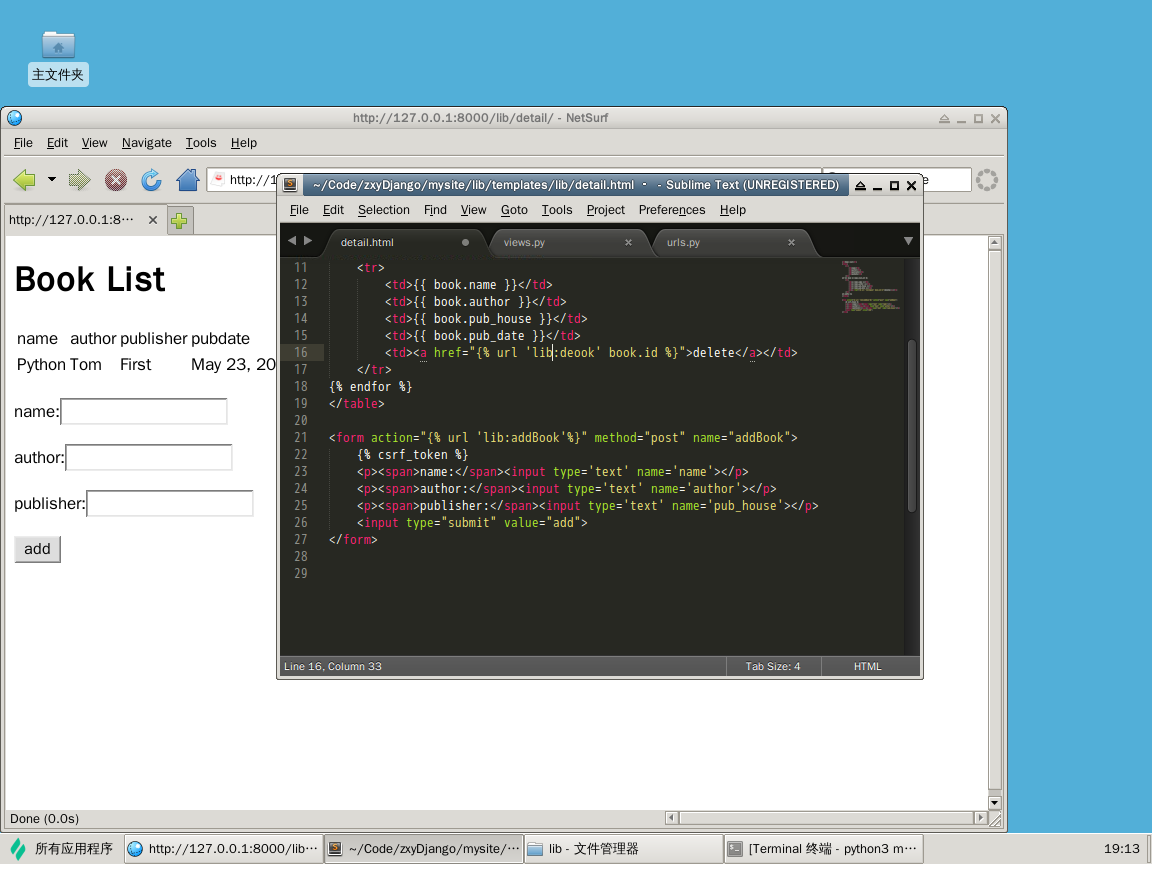
render()函数把request对象作为它的第一个参数，模板作为第二个参数，字典作为它的可选的第三个参数。它返回给定模板呈现的给定文本的一个HttpResponse对象。

在lib/urls中添加链接



数据的添加与删除

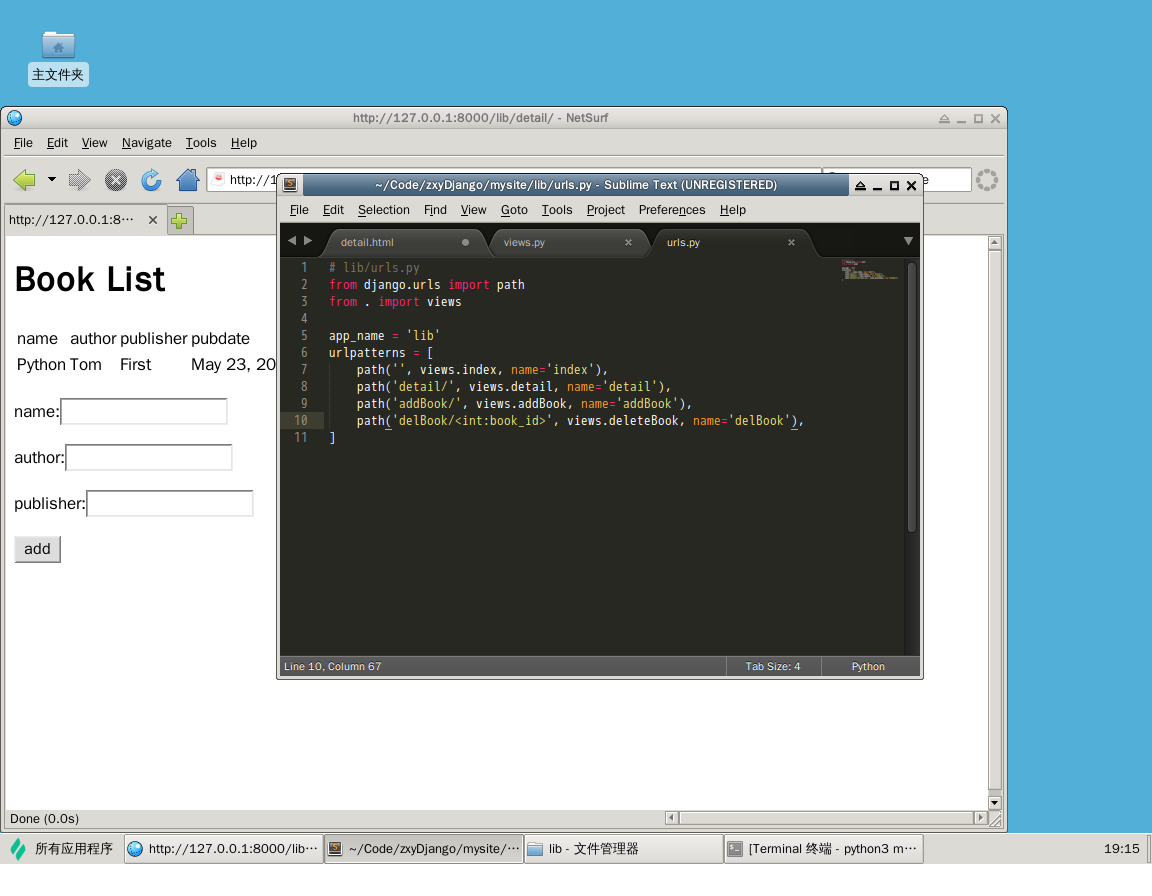
添加一个表单



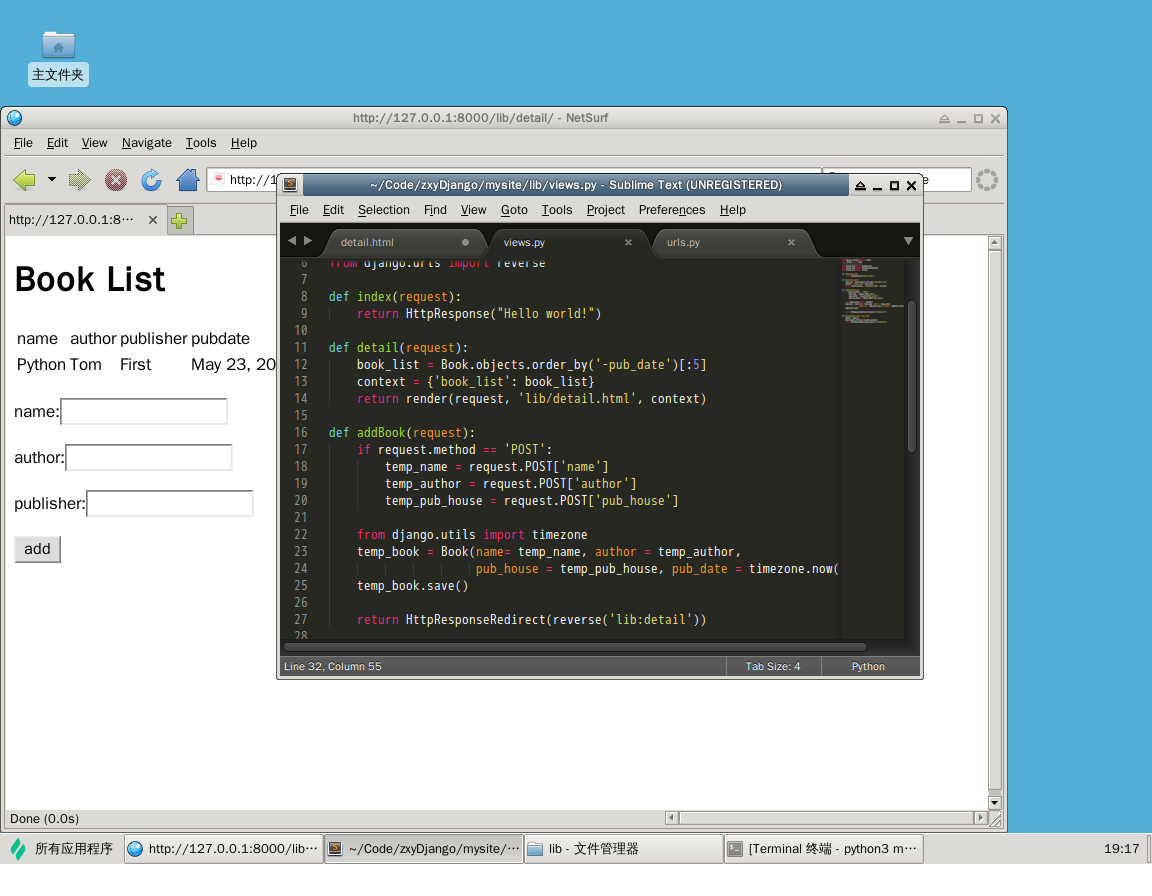
{% csrf\_token %} 模板标签用来防止跨域请求攻击

lib/urls.py 文件中稍作修改，加上 app\_name 设置命名空间：app\_name = 'lib'

添加地址映射path('addBook/', views.addBook, name='addBook'),



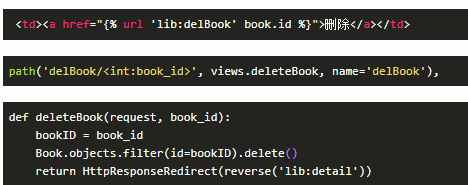
在views中添加对应函数



request.POST 是一个类字典对象，可以通过关键字的名字获取提交的数据。 这个例子中，request.POST['name'] 以字符串形式返回name的值。 request.POST 的值永远是字符串。

HttpResponseRedirect 重定向 reverse() 函数。 **这个函数避免了我们在视图函数中硬编码 URL** 。它需要我们给出想要跳转的视图的名字和该视图所对应的 URL 模式中需要给该视图提供的参数。 reverse()调用后将返回这样一个字符串：/lib/detail/.

删除数据



Django的MTV设计模式

MTV Django的MTV设计模式是借鉴和遵循MVC的。

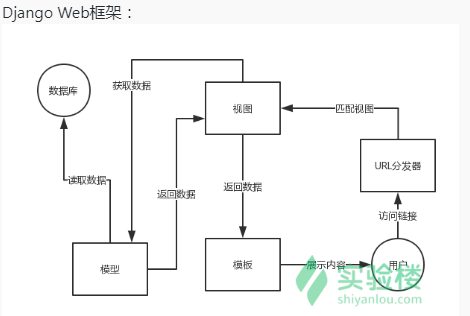
MTV具体定义如下：

M:模型（Model）,负责业务对象和数据库的关系映射。

T:模板（Template）,负责如何把页面展示给用户。

V:视图（View）,负责业务逻辑，并在适当时候调用模型和模板。

URL分发器 URL分发器的作用是将页面请求分发给不同的视图(View)处理，视图再调用相应的模型(Model)和模板(Template)。



在MTV模式中，MVC中的View分成了视图View（展现哪些数据）和模板Template（如何展现）2个部分，而控制器(Controller)这个要素由框架自己来实现了，我们需要做的就是把URL对应到视图V就可以了，通过这样的URL配置，系统将一个请求发送到一个合适的视图。