开发体会和小结

杨樾人 3160104875

此次我用Flask完成了一个二手书网站的项目，用户可以在这个网站发布要交易的书籍，也可以求书，并且还聚合了首页，提供分类检索和搜索的功能，也可以在网站上面下单，虽然功能点不多，但是前前后后改了两次代码的架构，也深有感触。

Python最火的Web框架有两个，Django和Flask。我之前用Django写过其他的网站，Django给人的感觉就是高大上，哪方面来看都很优秀，但是有些庞大，这次想尝试Flask。事实证明Flask虽然可以一行代码创建Web服务，但是要想完善一个Web项目一行代码是绝对搞不定的。一个完整的Flask项目，需要调用的扩展库不计其数。我以为学习Flask就只要看一个Flask文档就够了，但是每个扩展库都有自己独立的文档。有的甚至还没有中文，都要认真看。在完成这个作业的过程中，我也看完了一本几百页的介绍Flask的书。

Flask的定位是微框架，Flask框架本身只用到了两个第三方库，这两个库也是官方内置的第三方库。一个是Werkzeug用于提供Flask的底层Web框架支持，另一个就是Jinja2，Jinja2是一个模板渲染引擎，用于提供动态数据的渲染支持。除此之外，ORM，表单验证，登录权限管理，数据库支持等等一系列Web开发必要的功能，我借助第三方支持库来实现。

**Jinja2**

Jinja2是Flask框架内置为数不多的第三方扩展之一。做Web开发需要将服务器处理出来的动态信息渲染到静态页面中，直接用Python操作HTML之类往往是繁锁且不切实际的。Jinja2的存在就是解决这个问题的。Jinja2定位为模板渲染引擎，就是方便我们向静态页面渲染数据的。当然并不是这么简单的功能，Jinja2的沙箱功能和自动转义功能，对网站的安全性也起着至关重要的作用。

**Flask-SQLAlchemy**

Flask-SQLAlchemy是对SQLAlchemy的封装使其更好地支持Flask，可以在写业务逻辑的时候不用关心如何写SQL查询，只用关系具体的业务逻辑。不用关心如何建表，因为Flask-SQLAlchemy可以根据我们写的代码自动生成相应的数据表。ORM节省了很多开发流程，让数据库的使用变得不再那么繁锁。但是缺点也很明显，对待复杂查询会显得有些力不从心，不过我的数据量还没有到如此庞大的地步，所以就使用ORM了。

**WTForms**

Web服务建立之后总是需要处理来自不同请求的表单数据，并不是所有表单都符合我们的数据要求。对于验证不通过的数据我们自然不能提交到数据库。而WTForm就是提供后台表单验证的数据的扩展。它内置了很多常用的表单验证规则，通过简单的代码就可以验证form数据的正确性。

**Flask-Login**

Web开发总是需要有很多登录的场景，记录登录状态一般是通过session实现。而Flask-Login就是提供登录权限管理的扩展。只需要在注册的路由下面再带一个装饰器，这个路由就被重置为登录状态访问。细节代码我没有深入的了解，只看了Flask-login的简单配置，比之前自己进行用户验证要快很多。

通过这次网站的开发，促使自己学会了一项新的技能，也对于前端的一些东西进行了使用，总体来说是有了较大进步和提高的。之后我想，我需要开发前后端分离的项目，虽然Jinja2模板很好用，但是这只是对于我一个人开发而言，如果是在团队中开发项目前后端分离用ajax、js来进行数据的请求和展示效果会更好，代码更有健壮性。

最后，此次的作业让我提高了很多，对于B/S的理解也深入了很多，感谢老师一学期的教学。