数据库文件表存在baplatform/baweb/views/models.py中

要添加的数据库文件表结构：

*使用的是 Django 框架（基于 Django 的模型类定义）以及关系型数据库（SQLite）*

1. 课程资源表（course\_resources）

| 字段名 | 数据类型 | 描述 | 约束条件 |
| --- | --- | --- | --- |
| resource\_id | 整数类型（如 INT） | 资源的唯一标识符，主键，自动递增 | 无 |
| course\_id | 整数类型（如 INT） | 外键，关联到课程表（courses）中的课程 ID，标识资源所属课程 | 无 |
| file | 文件类型（如 Django 中的 FileField 对应的数据库类型，具体取决于数据库） | 存储课件文件本身，对应代码中的file字段 | 无 |
| file\_name | 字符类型（如 VARCHAR (64)） | 课件名，对应代码中的file\_name字段 | 默认值为file的文件名，可为空（blank=True） |
| resource\_intro | 文本类型（如 VARCHAR 或 TEXT，根据实际需求确定） | 资源简介，新增字段用于存储用户输入的资源简介内容 | 非空约束（确保每个资源都有简介），可根据实际情况设置字数限制约束 |
| category\_id | 整数类型（如 INT） | 资源所属类别，外键关联到资源类别表（resource\_categories）中的 category\_id 字段 | 无 |
| upload\_time | 日期时间类型（如 DATETIME） | 资源文件的上传时间，用于按时间排序资源，新增字段 | 上传资源时自动记录时间，用于排序操作 |
| lesson\_id | 整数类型（如 INT） | 外键，关联到课节表（lessons）中的课节 ID，标识资源所属的具体课节，新增字段 | 无 |

2. 资源类别表（resource\_categories）

| 字段名 | 数据类型 | 描述 | 无 |
| --- | --- | --- | --- |
| category\_id | 整数类型（如 INT） | 类别唯一标识符，主键，自动递增 | 无 |
| category\_name | 字符类型（如 VARCHAR (50)） | 类别名称 | 无 |
| category\_icon\_path | 字符类型（如 VARCHAR (255)） | 类别标识图标的存储路径（用于在界面展示类别标识图标） | 无 |

3.新增课节表（lessons）：

| 字段名 | 数据类型 | 描述 | 约束条件 |
| --- | --- | --- | --- |
| lesson\_id | 整数类型（如 INT） | 课节的唯一标识符，主键，自动递增 | 无 |
| course\_id | 整数类型（如 INT） | 外键，关联到课程表（courses）中的课程 ID，表明该课节所属的课程 | 无 |
| lesson\_name | 字符类型（如 VARCHAR (100)） | 课节名称，方便在界面展示以及区分不同课节 | 无 |
| lesson\_order | 整数类型（如 INT） | 课节在课程中的顺序编号，用于确定课节的先后顺 | 无 |

在上述设计中：

1. 课程资源表（course\_resources）：
   1. resource\_id作为主键确保每条资源记录的唯一性，方便后续的查询、更新和删除操作。
   2. course\_id通过外键关联到课程表，明确资源所属的课程。
   3. file和file\_name保留了原代码中关于课件文件及文件名的设置。
   4. **新增**的resource\_intro字段满足了自主添加资源简介的功能需求，其数据类型和约束条件按照数据需求进行了设置。
   5. category\_id字段通过外键与resource\_categories表关联，使得每个资源都能明确所属的类别，便于后续按类别筛选资源。
   6. **新增** upload\_time 字段：其数据类型设置为日期时间类型（如 DATETIME），用于记录资源文件上传的时间。当用户上传资源时，可以通过 Django 的相关逻辑自动获取并填充这个时间值（例如在视图函数中处理资源上传操作时，使用 Python 的 datetime 模块获取当前时间并赋值给这个字段）。有了这个字段后，就可以方便地在查询资源时按照上传时间进行排序了，比如在 Django 的视图函数中使用 order\_by('upload\_time') 方法对查询结果进行排序（升序排序，如果想降序可以使用 order\_by('-upload\_time')）。
   7. **新增** lesson\_id 字段：通过外键关联到新增的 lessons 表中的 lesson\_id 字段，这样每个资源就能明确所属的具体课节了。例如，在一个在线课程平台中，不同课节可能会有对应的课件资源，通过这个关联可以更精准地组织和展示资源，方便用户根据课节查找相应资源，也便于课程管理和维护。
2. **新增**资源类别表（resource\_categories）：主要用于管理和维护资源的类别信息。
   1. category\_id作为主键确保每个类别记录的唯一性。
   2. category\_name明确了每个类别的具体名称，方便在界面展示给用户以及在查询筛选时进行匹配操作。
   3. **新增**的category\_icon\_path字段用于存储类别标识图标的存储路径，以便在界面上能够直观地展示每个类别对应的标识图标，提高用户对类别选项的辨识度。
3. **新增**课节表（lessons）：主要用于管理和维护课程中的各个课节信息。（暂定20个week->20个lesson）
   1. lesson\_id 作为主键确保每个课节记录的唯一性。
   2. course\_id 外键关联到课程表，表明该课节属于哪门课程，这样就构建了课程与课节之间的层级关系。
   3. lesson\_name 字段用于清晰地标识每个课节的名称，方便在界面展示给用户查看具体是哪个课节的内容。
   4. lesson\_order 字段可以用于定义课节在课程中的顺序，比如按照课程的教学大纲，第一节的 lesson\_order 值可以设为 1，第二节设为 2 等等，便于在前端界面按照正确的顺序展示课节内容以及进行相应的导航操作。
4. models.py中，添加：

class ResourceCategories(models.Model):

category\_id = models.AutoField(primary\_key=True)

category\_name = models.CharField(max\_length=50)

category\_icon\_path = models.CharField(max\_length=255)

class Lessons(models.Model):

lesson\_id = models.AutoField(primary\_key=True)

course\_id = models.ForeignKey('Courses',on\_delete=models.CASCADE)

lesson\_name = models.CharField(max\_length=100)

lesson\_order = models.IntegerField(blank=True, null=True)

课程资源表新增：

class CourseFiles(models.Model):

resource\_intro = models.TextField()

upload\_time = models.DateTimeField(blank=True, null=True)

lesson\_id = models.ForeignKey(Lessons,on\_delete=models.CASCADE)

教师：管理-上传、编辑、备注

学生：使用-

其他：默认简介生成/提取

时间

查找（？）

考虑子目录

可能是ORM

SQLite最后的数据库表是完整的

Axure RP

SQLite 管理工具 下载安装

Sqlite\_master 自动生成的数据库表，记载表信息

viewData.py 一个程序，自助查询某个表是否存在。

web的session

Objects.filter Django的ORM方法

HTML/JavaScript提取信息到网页界面

Course\_list.html 按<table>展示功能-需要直接到页面上改。

权限 if is\_teacher

也可以用权限的方式，到数据库改

每个东西到程序的哪个地方去修改。

目的：搞清Django架构的逻辑；从前端到逻辑到数据库跑通。