

直接回答

以下是对代码的详细分析和功能注释，基于提供的三个Python文件：common_util.py、crawl_util.py和dbutil.py。我会简要说明每个文件的主要功能，并列出其函数及其用途。

- **common_util.py** 提供通用工具函数，涵盖加密、生成唯一ID、显示对话框、时间处理和输入验证。
- **crawl_util.py** 用于从豆瓣Top 250书籍列表抓取数据，并将提取的信息存储到MySQL数据库。
- **dbutil.py** 包含一个数据库帮助类，管理用户、书籍、借阅、消息和公告的相关数据库操作。

common_util.py 的功能

这个文件包含一些通用的工具函数，主要用于辅助应用程序的各种操作：

- `get_md5(data)`：生成输入数据的MD5哈希，用于数据完整性检查。
- `get_uuid()`：基于主机ID、序列号和当前时间生成唯一标识符。
- `msg_box(widget, title, msg)`：显示一个带“确定”按钮的警告对话框。
- `accept_box(widget, title, msg)`：显示一个带“是”和“否”按钮的警告对话框，默认选择“否”。
- `get_current_time()`：返回当前日期和时间，格式为“YYYY-MM-DD HH:MM:SS”。
- `read_qss(qss_file)`：读取并返回指定的QSS（Qt样式表）文件内容。
- `get_return_day(day)`：计算从当前日期起指定天数后的日期。
- `set_le_reg(widget, le, pattern)`：为控件设置正则表达式验证，限制输入格式。

crawl_util.py 的功能

这个文件专注于从豆瓣网站抓取书籍信息，并将其存储到数据库：

- `get_html(url)`：通过HTTP GET请求获取网页HTML内容，使用随机User-Agent和Cookie避开反爬机制。
- `ana_by_bs4(html)`：使用BeautifulSoup解析HTML，提取豆瓣Top 250书籍的详细信息，如图片、标题、作者等，并打印和组织数据。
- `insert_data(data_list)`：将提取的书籍数据插入MySQL数据库的 `douban_book` 表。

dbutil.py 的功能

这个文件定义了一个 DBHelp 类，提供了丰富的数据库操作方法：

- 连接管理：初始化数据库连接，确保单例模式。
- 查询操作：支持查询所有记录、条件查询、消息查询等。
- 插入操作：添加用户、书籍、借阅信息、消息和公告。
- 更新操作：更新书籍信息、借阅状态、消息状态等。
- 删除操作：根据条件删除记录。

这些文件似乎是一个书籍管理系统的一部分，涵盖了通用工具、数据抓取和数据库管理功能。

调查报告

以下是对用户提供的三个Python文件（common_util.py、crawl_util.py和dbutil.py）的详细分析和功能注释，旨在全面了解其作用和用途。这份报告基于对文件内容的提取和总结，力求覆盖所有相关细节。

背景与概述

用户提供了三个Python文件，分别用于不同的功能模块：

- **common_util.py**：提供通用工具函数，涵盖加密、生成唯一ID、对话框显示、时间处理和输入验证。
- **crawl_util.py**：专注于从豆瓣Top 250书籍列表抓取数据，并将其存储到MySQL数据库。
- **dbutil.py**：定义了一个 DBHelp 类，提供了丰富的数据库操作方法，管理用户、书籍、借阅、消息和公告相关数据。

这些文件似乎是一个书籍管理系统的组成部分，集成了通用工具、网页抓取和数据库操作功能。

以下是每个文件的详细分析。

common_util.py 的详细分析

这个文件包含8个主要函数，旨在提供通用的辅助功能，适用于各种应用程序场景。以下是每个函数的详细描述：

- **get_md5(data)**

功能：生成输入数据的MD5哈希。

用途：常用于数据完整性检查或密码哈希，基于MD5算法加密明文数据。

示例场景：验证文件或数据的完整性，确保传输过程中未被篡改。

- **get_uuid()**

功能：基于主机ID、序列号和当前时间生成唯一标识符。

用途：确保生成的ID在全局范围内唯一，适用于记录或对象的唯一标识。

示例场景：为数据库记录或会话生成唯一标识。

- **msg_box(widget, title, msg)**

功能：显示一个带“确定”按钮的警告对话框。

用途：用于向用户显示警告信息，等待用户确认。

示例场景：提示用户操作完成或错误信息。

- **accept_box(widget, title, msg)**

功能：显示一个带“是”和“否”按钮的警告对话框，默认选择“否”。

用途：用于需要用户确认的操作，提供“是/否”选择。

示例场景：询问用户是否删除某条记录。

- **get_current_time()**

功能：返回当前日期和时间，格式为“YYYY-MM-DD HH:MM:SS”。

用途：获取系统当前时间，用于日志记录或时间戳生成。

示例场景：记录操作时间或生成时间戳。

- **read_qss(qss_file)**

功能：读取并返回指定的QSS（Qt样式表）文件内容。

用途：加载自定义样式表，用于界面自定义。

示例场景：为GUI应用程序加载自定义样式。

- **get_return_day(day)**

功能：计算从当前日期起指定天数后的日期，返回格式为“YYYY-MM-DD HH:MM:SS”。

用途：用于日期计算，例如计算还书日期或任务截止日期。

示例场景：为借阅书籍计算归还日期。

- **set_le_reg(widget, le, pattern)**

功能：为控件设置正则表达式验证，限制输入格式。

用途：确保用户输入符合特定格式，例如邮箱或电话号码。

示例场景：验证用户输入的表单数据。

crawl_util.py 的详细分析

这个文件包含3个函数，专注于从豆瓣Top 250书籍列表抓取数据，并将其存储到数据库。以下是每个函数的详细描述：

- **get_html(url)**

功能：通过HTTP GET请求获取网页HTML内容，使用随机User-Agent和Cookie避开豆瓣的反爬机制。

用途：获取豆瓣Top 250书籍列表的网页内容，确保抓取成功。

示例场景：抓取豆瓣书籍页面，提取相关信息。

- **ana_by_bs4(html)**

功能：使用BeautifulSoup解析HTML，提取豆瓣Top 250书籍的详细信息，包括图片、标题、作者、出版社、出版年份、价格、评分、评论数和引言等，并打印和组织数据。

用途：从抓取的HTML中提取结构化书籍信息，便于后续存储或展示。

示例场景：解析抓取的网页，生成书籍列表。

- **insert_data(data_list)**

功能：将提取的书籍数据插入MySQL数据库的douban_book表。

用途：持久化抓取的书籍信息，便于后续查询或分析。

示例场景：将解析后的书籍数据存储到数据库。

从功能上看，`crawl_util.py` 是一个专门的爬虫模块，集成了反爬策略（如随机User-Agent和Cookie）和数据解析功能，目标是获取豆瓣书籍数据并存储到数据库。

dbutil.py 的详细分析

这个文件定义了一个 `DBHelp` 类，提供了23个方法，涵盖数据库连接管理、查询、插入、更新和删除操作，涉及用户、书籍、借阅、消息和公告等模块。以下是方法的详细分类和描述：

数据库连接与管理

- `__init__(self, host='127.0.0.1', port=3306, user='root', pwd='111111', db='dbchai1', charset='utf8')`

功能：初始化数据库连接，使用提供的MySQL参数创建连接和游标对象。

用途：建立与数据库的连接，为后续操作准备。

示例场景：应用程序启动时初始化数据库连接。

- `get_instance(cls)`

功能：实现单例模式，确保 `DBHelp` 类只有一个实例。

用途：避免重复创建数据库连接，提高资源利用率。

示例场景：确保整个应用程序共享一个数据库连接。

- `db_commit(self)`

功能：提交当前数据库事务。

用途：确保数据更改被持久化到数据库。

示例场景：完成插入或更新操作后提交。

- `db_rollback(self)`

功能：在错误发生时回滚当前事务。

用途：恢复到事务开始前的状态，防止数据不一致。

示例场景：操作失败时回滚数据库更改。

以下方法支持各种查询需求：

- **query_all(self, table_name)**

功能：查询指定表的所有列和记录，返回记录数和结果集。

用途：获取表中的所有数据。

示例场景：列出所有书籍信息。

- **query_super(self, table_name, column_name, condition)**

功能：根据条件查询指定表的特定列，返回记录数和结果集。

用途：灵活查询表中的部分数据。

示例场景：查询特定用户的借阅记录。

- **query_message_super(self, column_name, condition)**

功能：根据条件查询消息表，按发送时间降序排序，返回记录数和结果集。

用途：获取满足条件的消息列表。

示例场景：查询未回复的消息。

- **query_user_message(self, username)**

功能：查询指定用户的发送和接收消息，按发送时间降序排序，返回记录数和结果集。

用途：获取用户的消息历史。

示例场景：显示用户的所有消息。

- **query_announce(self)**

功能：查询所有公告，按公告时间降序排序，返回记录数和结果集。

用途：获取所有公告信息。

示例场景：显示系统公告列表。

- **query_douban(self)**

功能：随机选择 douban_book 表中的15条记录，返回记录数和结果集。

用途：随机获取豆瓣书籍数据，可能用于推荐。

示例场景：为用户推荐书籍。

插入操作

以下方法用于添加新记录：

- **add_user(self, data)**

功能：将用户数据插入 user 表，包括ID、用户名、密码、角色、创建时间等。

用途：注册新用户。

示例场景：用户注册时添加记录。

- **add_book(self, data)**

功能：将书籍数据插入 book 表，包括ID、书名、作者、出版社、库存数量等。

用途：添加新书籍。

示例场景：管理员添加新书。

- **insert_borrow_info(self, data)**

功能：将借阅信息插入 borrow_info 表，包括书籍ID、书名、借阅用户等。

用途：记录借阅操作。

示例场景：用户借书时记录。

- **insert_message_info(self, data)**

功能：将消息数据插入 message 表，包括发送者、接收者、内容和时间等。

用途：发送新消息。

示例场景：用户发送消息给管理员。

- **insert_announce_info(self, data)**

功能：将公告数据插入 announcement 表，包括标题、内容和时间等。

用途：发布新公告。

示例场景：管理员发布系统通知。

更新操作

以下方法用于更新现有记录：

- `update_super(self, table_name, column_name, condition, data)`

功能：根据条件更新指定表的记录，包括书名、作者等字段。

用途：灵活更新表中的数据。

示例场景：更新书籍信息。

- `update_borrow(self, book_id)`

功能：更新书籍的库存和借阅数量，减少库存并增加借阅次数。

用途：记录书籍被借出。

示例场景：用户借书后更新状态。

- `update_borrow_return(self, book_id)`

功能：更新书籍的库存和借阅数量，增加库存并减少借阅次数。

用途：记录书籍归还。

示例场景：用户还书后更新状态。

- `update_borrow_statue(self, book_id)`

功能：更新 `borrow_info` 表中的 `return_flag` 字段，标记书籍已归还。

用途：更新借阅状态。

示例场景：确认书籍归还。

- `update_renew(self, data)`

功能：更新借阅信息表中的借阅天数和归还时间。

用途：处理书籍续借。

示例场景：用户续借书籍。

- `update_message_info(self, data)`

功能：更新消息表，标记消息已回复，并添加回复内容和时间。

用途：处理消息回复。

示例场景：管理员回复用户消息。

删除操作

以下方法用于删除记录：

- `delete(self, table_name, column_name, condition)`

功能：根据条件从指定表中删除记录。

用途：删除不需要的数据。

示例场景：删除过期公告。

- `del_message_info(self, id)`

功能：根据ID从 message 表中删除消息。

用途：删除特定消息。

示例场景：用户删除已读消息。

综合分析与应用场景

从以上分析可以看出，这三个文件共同构成了一个书籍管理系统的功能模块：

- `common_util.py` 提供了通用的辅助工具，如时间处理、加密和对话框显示，适用于整个系统的各种场景。
- `crawl_util.py` 负责从豆瓣抓取书籍数据，体现了系统的扩展性，可能用于初始化数据库或更新书籍信息。
- `dbutil.py` 是数据库操作的核心，涵盖了用户管理、书籍管理、借阅管理、消息管理和公告管理，体现了系统的完整性。

这些文件之间的关系可以推测为：`crawl_util.py` 抓取的书籍数据可能通过 `dbutil.py` 的方法存储到

数据库，而 common_util.py 的函数则可能在界面交互或数据处理中被调用。

总结

通过对三个文件的详细分析，我们了解了它们的具体功能和用途。common_util.py 提供了通用工具，crawl_util.py 实现了数据抓取，dbutil.py 管理了数据库操作。这些文件共同支持了一个功能齐全的书籍管理系统，涵盖了从数据获取到存储和管理的各个方面。

引用来源：

- 附件0：common_util.py
- 附件1：crawl_util.py
- 附件2：dbutil.py