上海大学
SHANGHAI UNIVERSITY

数据库原理 (2)

课程项目报告

姓	名	江 俊
姓	名	王子潇
姓	名	严昕宇
日	期	2023 年 5 月 20 日

《数据库原理 (2)》课程项目报告

江俊 (20121412) 王子潇 (20121419) 严昕宇 (20121802)
上海大学 计算机工程与科学学院

1 项目概述

鼠洞是一个专为 SHU 学子量身打造的在线问答网站，它提供了一个友好的辩论和宣泄情绪的平台。该系统具有多种功能模块，包括用户注册、登录、权限管理、问题提问、回答、点赞、合法校验、不同场景下的排序处理、统计分析、详情展示和内容搜索等。此外，为了实现核心业务系统的高可用性和异步解耦，系统还剥离出了短信验证、日志处理和索引处理子系统。管理员可以通过后台管理模块对系统进行管理，以确保网站的正常运行。网站的整体架构如图 1 所示。

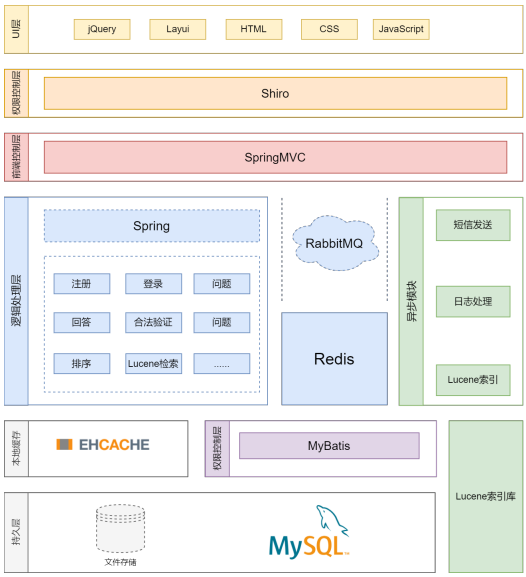

图 1: 项目整体架构

项目整体采用了 Spring 框架，一个开源的 Java 平台，它提供了一系列功能，包括依赖注入、面向切面编程和事务管理等。项目实现前后端分离、交互，主要功能及实现方法如下：

- 前端安全框架将 Apache Shiro 与 Spring 框架集成，提供了认证、授权、加密和会话管理等功能，并且支持多种数据源，包括 LDAP、JDBC 和 INI 文件等。同时它还提供了丰富的认证和授权策略，可以满足不同应用程序的安全需求。
- 前端控制层采用 Spring MVC，提供了一个强大的 Web 应用程序开发框架。它建立在 Spring 框架的核心功能之上，提供了一种简单、灵活且可扩展的方法来构建 Web 应用程序。通过将应用程序分为模型 (Model)、视图（View）和控制器（Controller）三个部分来实现松耦合和高内聚：模型负责封装应用程序的数据和业务逻辑；视图负责呈现数据；控制器负责处理用户请求并协调模型和视图。
- UI 部分主要使用 Mockplus 设计网站界面，并由 HTML、CSS、JavaScript 等提供支持。
- 后端 RabbitMQ 作为一个消息队列中间件，可以帮助应用程序在分布式环境中进行异步通信，并与 Spring 框架集成，以提供可靠的消息传递功能。

- Redis 是一个内存数据存储系统，它可以用作缓存、消息代理和数据库，与 Spring 框架集成以提供快速的数据访问功能，实现了本项目的很多核心功能。
- 存储方面 MySQL 是一个关系数据库管理系统，可以用来存储应用程序的数据，以提供数据持久化功能。
- 持久层框架本项目采用的是 MyBatis，避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集，可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs（Plain Old Java Objects，普通的 Java 对象）映射成数据库中的记录。

2 MySQL 介绍

2.1 MySQL 核心表设计

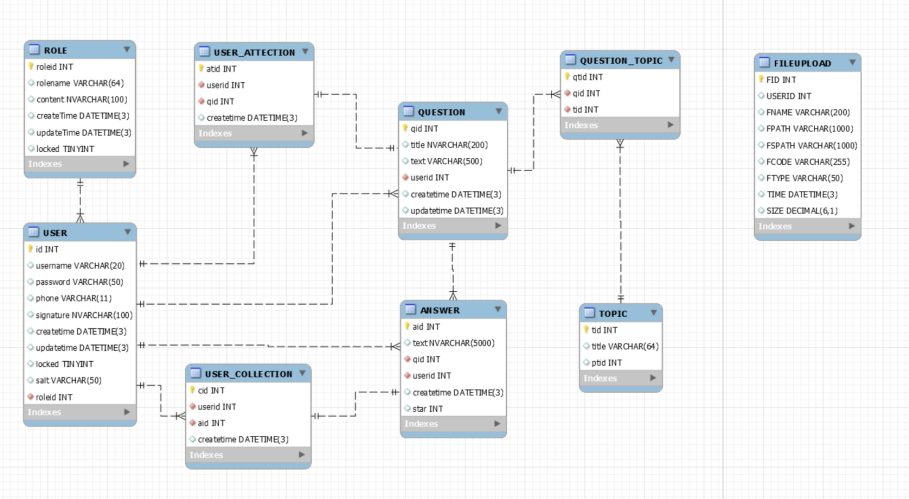

图 2: MySQL 表设计

数据库中主要的表和表之间的关系如图 2 所示，包括以下功能：

- 用户表：主要用于存储用户登陆时的账号、密码、手机号及账号相关的创建信息等。
- 角色表：主要用于存储用户名、ID、描述信息等。
- 问题表：主要用于存储问题 ID、标题、内容及提问的用户 ID 等信息。
- 回答表：主要用于存储回答 ID、问题 ID、回答的用户 ID、内容等信息。
- 关注表：主要用于存储用户关注问题 ID、用户 ID、问题 ID 等信息。将此表从角色表中分离，避免了角色表中其他元素的大量重复，产生冗余。其他表将其从角色表中分离出的作用类似。
- 用户收藏表：主要用于存储用户 ID 及其收藏的问题 ID 等信息。
- 话题表：主要用于存储话题 ID、父级话题 ID 等信息，层层递进。
- 问题、话题关联表：主要用于存储某一话题下的各类问题，因此包括话题 ID 和问题 ID 等信息。
- 附件表：主要用于存储用户的各类附件，如头像路径、下载的默认地址等等。

表中的各元组数据并不会实时更新，而是采取 Redis 作为缓存。Redis 可以缓存 MySQL 数据库中的热点数据如问题发布后可能会同时有多个回复，Redis 可基于内存访问处理速度快，而不是直接对数据库（外存）做修改，减少对 MySQL 数据库的访问次数。这样，当应用程序需要访问这些热点数据时，它可以直接从 Redis 中获取，而不需要访问 MySQL 数据库。这样可以大大减少 MySQL 数据库的负载，提高应用程序的响应速度。

2.2 MySQL 配置

配置过程与实验二同理，但在 IDEA 中 maven 项目的 pron.xml 配置文件中添加 MySQL 的依赖包，例如：

```
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.15</version>
    </dependency>
</dependencies>
```

3 NoSQL-Redis 介绍

3.1 Redis 概述

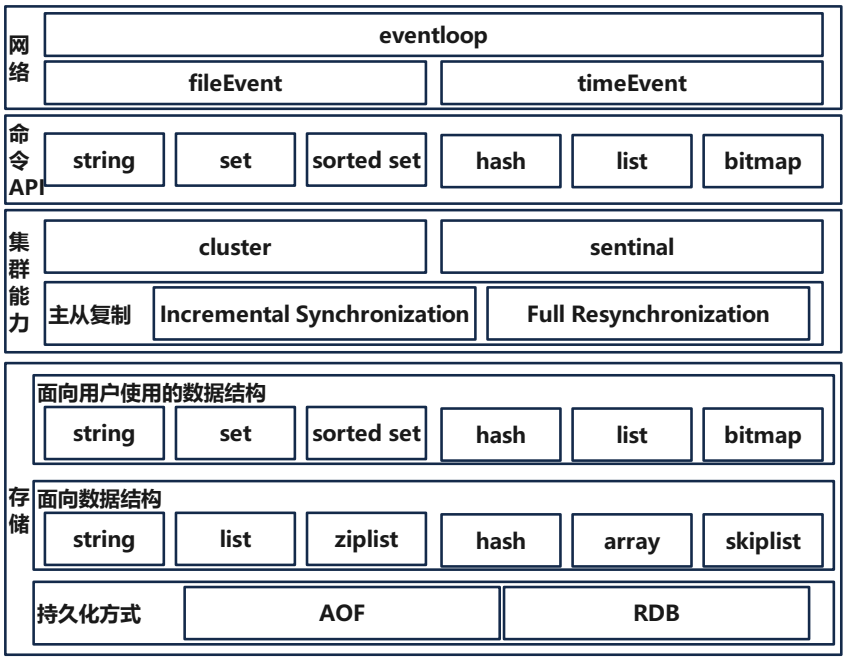

图 3: Redis 整体架构

Redis（Remote Dictionary Server）是一款快速、开源、内存数据结构存储系统，常用作缓存、消息队列、会话存储等用途。其支持多种数据结构，包括字符串、哈希表、列表、集合、有序集合等，且可以在内存中存储数据，并通过异步复制将数据持久化到磁盘上，以实现数据持久化。此外 Redis 的网络模型基于非阻塞 IO 和事件循环，可以高效地处理大量并发连接；而线程模型采用单线程的事件驱动模型，可以更好地利用系统资源，提高系统的性能和吞吐量。在集群方面，Redis 集群使用一种称为哈希槽（hash slot）的方式来进行数据分片，每个哈希槽对应着一个数据分片，集群中的每个节点负责管理一部分哈希槽，并存储对应的数据。当一个节点宕机或者加入集群时，集群会进行哈希槽的重新分配，保证数据仍然可以正常访问。另外，Redis 集群还支持节点的复制，每个主节点可以配置若干个从节点，从节点会自动复制主节点的数据，以实现数据的备份和容错。当一个主节点宕机时，它的从节点会自动成为新的主节点，保证数据的可

用性。除了数据分片和节点复制，Redis 集群还提供了一些其他的容错和恢复机制，如故障转移、自动故障恢复等，可以有效提高集群的可用性。

3.2 Redis 配置

- 更新软件包：sudo apt-get update
- 执行 sudo apt-get install redis-server，输入 y 确认安装
- 执行完成后，查看 redis 服务的状态，执行 ps -ef|grep redis 查看或者 service redis status 命令查看，状态为 running，说明安装完成后系统自动启动了服务。

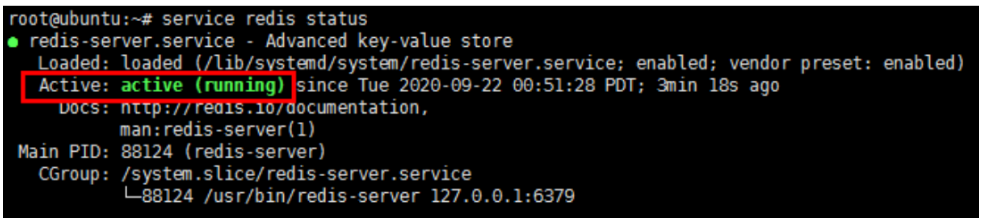

图 4: service redis status 运行结果

- 输入 sudo vim /etc/redis/redis.conf，修改 redis 的相关配置，默认端口号 port 是 6379（服务器需要开放此端口），注释 bind 127.0.0.1，使得所有的设备都可以远程连接服务器的 Redis，protected_mode 改成 no。
- 输入 service redis restart 重启 redis 服务。
- 输入 netstat -talnp 查看端口情况，若发现 6379 端口已打开且所有的 IP 均可访问即为成功：

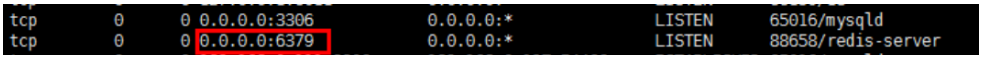

图 5: netstat -talnp 运行结果

- 本地 IDEA 访问 redis 需要下载 Redis 插件，填入相关配置后连接，若连接成功即可使用 Java 的 API 操纵远程服务器的 Redis

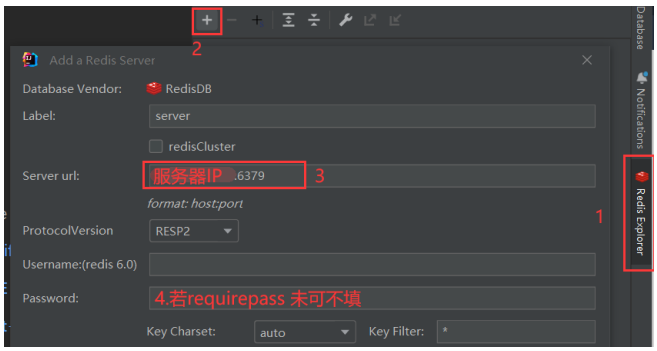

图 6: 本地 IDEA 上配置远程 Redis

- 在 IDEA 的 maven 项目的依赖文件 porn.xml 上添加以下内容：

```
<dependencies>
    <dependency>
        <groupId>redis.clients</groupId>
```

```
        <artifactId>jedis</artifactId>
        <version>3.0.0</version>
    </dependency>
</dependencies>
```

至此，可在 IDEA 中用 Java 的 API 操纵远程的 Redis 数据库。

3.3 在项目中应用 Redis

- **Redis 作为 MySQL 查询结果的缓存：**当应用程序需要查询 MySQL 数据库时，可以先查询 Redis 缓存，如果缓存中有相应的数据，则直接返回结果，否则再去查询 MySQL 数据库，并将查询结果存储在 Redis 缓存中。在下次查询时，如果 Redis 缓存中有相应的数据且未过期，则直接从缓存中获取结果，避免了对 MySQL 数据库的访问，提高了查询效率。Redis 作为 MySQL 查询结果的缓存，本项目也考虑缓存的过期时间、缓存的更新机制等问题，以保证缓存的一致性和正确性。此外本项目对于需要进行 MySQL 更新操作的数据，也同时会定期更新 Redis 中的缓存，以避免数据不一致的问题。

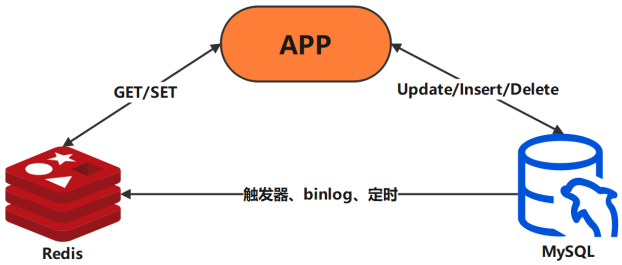

图 7: Redis 作为 MySQL 查询结果的缓存

表 1: Redis 表设计

表名	类型	作用	键/值/权重
REDIS_HASH_PHONE_TIME	Hash	已申请验证码手机号	手机号 | 时间戳
REDIS_HASH_PHONEVERCODES	Hash	手机号以及发送的验证码	手机号 | 验证码
REDIS_HASH_USER_HEAD_IMGS	Hash	用户头像路径信息	用户 ID | 头像路径
REDIS_SET_HASREGISTERPHONE	Set	已注册用户手机号	用户手机号
REDIS_SET_QUESTIONS_PID	Set	用户已提问问题	问题 ID
REDIS_SET_STAR_ANSWERS_PID	Set	用户已点赞回答	回答 ID
REDIS_ZSET_QUESTIONS_TIME	Sorted Set	新增问题信息	问题 ID | 时间戳
REDIS_ZSET_QUESTIONS_HOT	Sorted Set	热门问题	问题 ID | 热门指数
REDIS_ZSET_QUESTIONS_HOT_UID	Sorted Set	热门问题临时查询缓存	问题 ID | 热门指数
REDIS_ZSET_ANSWERS_QID	Sorted Set	问题回答信息	问题 ID | 点赞数

- **验证码机制：**Redis 使用键值对的方式存储数据，因此可以使用手机号作为键，验证码作为值来存储数据。当 Redis 接收到 SET 命令时，它会在内部创建一个哈希表（hash table），将手机号作为键，验证码作为值，然后将哈希表存储到内存中。如果 Redis 的持久化机制已启用，那么这个哈希表还会被写入磁盘中，以便于在 Redis 重启后可以恢复数据。在验证用户输入的验证码时，我们可以使用 Redis 的 GET 命令获取手机号对应的验证码，并将其与用户输入的验证码进行比较。如果两者相等，则说明验证码正确，否则说明验证码错误。时间戳是给予验证码一个生命周期，当超过时限时验证码失效，同时也限制短时间（如 1min）内一个手机号只能申请一次验证码，避免重复请求让后台压力过大。使用的部分 Redis Hash 相关 API 如下所示：

```
redisTemplate.opsForHash().put(hashKey, key, value); // 添加Hash键值
redisTemplate.opsForHash().get(hashKey, key);        // 获取Hash指定键值
redisTemplate.opsForHash().hasKey(hashKey, key);     // 判断Hash是否存在指定键
redisTemplate.delete(key);                           // 删除Hash指定键
```

- **手机号存储：**在 Redis 中使用 SET 命令存储手机号时，Redis 会将手机号作为键，一个空字符串作为值存储在内存中，采用了哈希算法来快速定位键值对。同时这个空字符串并不占用太多的空间，因为在 Redis 内部，空字符串实际上是一个指针，指向一个空的字符串对象，这个对象是 Redis 内部维护的。因此，对于每个手机号，Redis 只需要存储一个指向空字符串对象的指针，而不需要存储一个实际的空字符串对象。此外，为了保证 Redis 的高性能和可靠性，它通常将数据存储在内存中，并定期将数据持久化到磁盘上。这样可以确保数据即使在服务器断电或宕机时也能够得到保护。Redis 提供了多种持久化方式，包括 RDB 快照和 AOF 日志等。用户已提问问题和点赞回答同理，键是用户 ID 而值是问题 ID 或回答 ID。使用的部分 Redis Set 相关 API 如下所示:

```
redisTemplate.opsForSet().add(setKey, itemValue);     // 往Set中添加值
redisTemplate.opsForSet().isMember(setKey, value);    // 判断某个值是否存在于指定的Set中
redisTemplate.opsForSet().remove(setKey, itemKey);    // 删除Set中的某个值
redisTemplate.opsForSet().size(setKey);               // 获取Set中的元素数量
```

- **热门问题实时更新：**热门问题列表采用 Redis 的 Sorted Set 存储，它是一种有序集合数据结构，使用了跳跃表和哈希表的组合来实现，具有快速的插入、删除和查询元素的能力。Sorted Set 中的每个元素都是一个字符串值，同时还关联着一个分数值，表示该元素的权重。Redis 会根据分数值对元素进行排序，并保证每个元素的唯一性。每次有新的问题被提出或已有问题的权重发生变化时，只需要更新该问题在 Sorted Set 中的分数值即可。Sorted Set 会根据新的分数值重新排序，保证列表中的问题始终是最热门的。此外，由于 Redis 支持分布式部署，使用 Sorted Set 存储热门问题时，可以将数据分散到多个 Redis 实例中，以提高数据的可扩展性和可用性。
- **热门问题列表和查询缓存分开存储：**热门问题列表一般是由系统维护的一个静态列表，包含了一些常见的、最近比较热门的问题。这些问题可能会被多个用户频繁查询，因此可以将它们存储在 Redis 的 Sorted Set 中，以提高数据查询的效率。但是对于某些比较冷门或者新提出的问题，可能还没有被加入到热门问题列表中，此时如果也使用 Sorted Set 来存储查询缓存的话，可能会导致 Sorted Set 的长度过长，查询速度变慢。此时，可以考虑将这部分查询缓存单独存储在一个 Hash 表或者 List 中，以减少 Sorted Set 的长度，提高查询效率。同时，对于热门问题列表和查询缓存的更新操作也可能会导致锁竞争，降低系统的性能。因此也可以减少锁竞争，提高系统的并发性和性能。本项目的 Sorted Set 通过 Zset 实现，使用的部分相关 API 如下所示:

```
// 判断Zset是否存在
redisTemplate.opsForZSet().zCard(zsetKey);
// 往 Zset 中添加值
redisTemplate.opsForZSet().add(zsetKey, itemKey, value);
// 指定区间内查询zset键值信息
redisTemplate.opsForZSet().reverseRangeByScore(zsetKey, minValue, maxValue);
// 指定区间内查询zset键值信息，同时返回键、值信息
redisTemplate.opsForZSet().reverseRangeByScoreWithScores(zsetKey, minValue, maxValue);
// 指定区间内查询zset键值信息，同时返回键、值信息，由小到大排序
redisTemplate.opsForZSet().rangeByScoreWithScores(zsetKey, minValue, maxValue);
// 查询指定区间内最大几条记录对应的键
```

```
redisTemplate.opsForZSet().reverseRange(zsetKey, start, end);
// 返回指定区间内最大几条记录对应的键值对信息
redisTemplate.opsForZSet().reverseRangeWithScores(zsetKey, start, end);
// 指定区间内分页查询zset键值信息
redisTemplate.opsForZSet().reverseRangeByScore(zsetKey, minValue,
                                                maxValue, index, count);
// 指定区间内分页查询zset键值信息，同时返回键、值信息
redisTemplate.opsForZSet().reverseRangeByScoreWithScores(zsetKey, minValue,
                                                          maxValue, index, count);
```

4 其他技术介绍

4.1 缓冲队列：RabbitMQ

RabbitMQ 是一款使用 Erlang 语言开发的，实现 AMQP (高级消息队列协议) 的开源消息中间件。它轻巧且易于在本地和云端部署，支持多种消息协议。RabbitMQ 可以部署在分布式和联邦配置中，以满足高规模、高可用性的要求。应用场景包括异步处理，应用解耦和流量削峰。此外 RabbitMQ 具有多种特性，包括可靠性 (Reliability)，灵活的路由 (Flexible Routing)，消息集群 (Clustering)，高可用 (Highly Available Queues)，多种协议 (Multi-protocol)，多语言客户端 (Many Clients)，管理界面 (Management UI)，跟踪机制 (Tracing) 和插件机制 (Plugin System)。

安装 RabbitMQ 需要先安装 Erlang 语言环境。然后再从官网下载 RabbitMQ 的安装包并进行安装。在安装过程中还需要安装 socat 依赖插件。RabbitMQ 的配置可以通过主配置文件进行，通常名为 rabbitmq.conf，包括核心服务器和插件的配置。 在 Spring 中，可以在 springrabbit.xml 文件中加入 RabbitMQ 的配置信息，例如：

```
<!-- 配置Rabbit连接工厂 -->
<rabbit:connection-factory
        id="connectionFactory"
        username="${rabbit_username}"
        password="${rabbit_password}"
        host="${rabbit_host}"
        port="${rabbit_port}"/>
```

然后可以定义 Spring Rabbit Template，相当于 Rabbit 操作对象，声明队列用于发送注册短信、检索处理、日志处理等等。

4.2 持久层 ORM：MyBatis

MyBatis 是一个持久层框架，它通过 JDBC API 与 MySQL 数据库进行交互。MyBatis 通过配置文件或注解来定义 SQL 语句，并将这些 SQL 语句映射到 Java 方法上。当在 Java 代码中调用这些方法时，MyBatis 会自动执行相应的 SQL 语句并将结果映射回 Java 对象。例如可以在 MyBatis 的 XML 配置文件中定义一个查询语句，然后将这个查询语句映射到一个 Java 方法上。当在 Java 代码中调用这个方法时，MyBatis 会自动执行这个查询语句并将结果映射回 Java 对象。

MyBatis 的 XML 配置文件通常命名为 mybatis-config.xml。在这个文件中可以配置 MyBatis 的核心设置，包括获取数据库连接实例的数据源 (DataSource) 以及决定事务作用域和控制方式的事务管理器

(TransactionManager)。也可以指定 MyBatis 的映射文件的位置，这些映射文件包含了 SQL 语句和映射定义信息。例如：

```
<configuration>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"/>
      <dataSource type="POOLED">
        <property name="driver" value="${driver}"/>
        <property name="url" value="${url}"/>
        <property name="username" value="${username}"/>
        <property name="password" value="${password}"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="org/mybatis/example/BlogMapper.xml"/>
  </mappers>
</configuration>
```

4.3 前端框架：SpringMVC

SpringMVC 是 Spring 框架提供的一个 Web 组件，全称是 Spring Web MVC。它是目前主流的实现 MVC 设计模式的框架之一，提供了前端路由映射、视图解析等功能。SpringMVC 可以更快速地完成 Web 应用程序的开发，提高开发效率。包括用于企业级应用、电商网站、金融系统等多种场景。

SpringMVC 的核心思想是将一个 Web 应用程序分为三个部分：模型（Model）、控制器（Controller）和视图（View）。控制器接收客户端的请求并调用模型层处理请求，然后将结果返回给视图层进行展示。这种分层设计使得 Web 应用程序更加模块化，易于维护和扩展。在使用 SpringMVC 进行开发时，需要配置前端控制器 DispatcherServlet。这是由框架提供的组件，在 web.xml 中配置，负责接收请求并响应结果，相当于转发器和中央处理器。开发后端控制器 Handler/Controller 即程序员开发的部分，负责接收用户请求信息并调用业务方法处理请求。开发视图 View，也是需要程序员开发的部分。View 是一个接口，实现类支持不同的 View 技术（如 jsp、freemarker、pdf 等），负责将数据展现给用户。

5 项目总结

本项目采用了 Spring 框架，实现了前后端分离和交互，并成功将后端部署到了服务器上。前端安全框架采用了 Apache Shiro 与 Spring 框架集成，提供了认证、授权、加密和会话管理等功能。前端控制层采用了 Spring MVC，提供了一个强大的 Web 应用程序开发框架。UI 部分使用 Mockplus 设计网站界面，并由 HTML、CSS、JavaScript 等提供支持。

后端方面，RabbitMQ 作为一个消息队列中间件，帮助应用程序在分布式环境中进行异步通信。Redis 作为一个内存数据存储系统，用作缓存、消息代理和数据库，提供快速的数据访问功能。MySQL 作为一个关系数据库管理系统，用来存储应用程序的数据，以提供数据持久化功能。持久层框架采用了 MyBatis，避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。

总的来说，在本项目的实现过程中，我们小组学习到了如何使用 Spring 框架、Apache Shiro、Spring MVC、RabbitMQ、Redis、MySQL 和 MyBatis 等技术来构建一个完整的 Web 应用程序。也深刻地体会

到了如何将一个复杂的 Web 应用程序分解为多个模块，并使用不同的技术来实现各个模块的功能。学习到了如何使用前端安全框架来保护 Web 应用程序，如何使用消息队列中间件来实现异步通信，以及如何使用内存数据存储系统和关系数据库管理系统来存储和访问数据。让我们对 Web 应用程序开发、数据库管理，NoSQL 的应用等方面有了更深入的理解。