0. 목차

1. 프로젝트 주제

도서관리 서비스 프로그램

2. 프로젝트 목표

(1) 서블릿(Servlet) 사용하여 웹 페이지와 상호작용하고, JDBC를 사용하여 데이터베이스에 데이터를 저장하고 조회하는 코드를 작성한다.
DBC를 사용하여 MySQL 데이터베이스에 도서 및 회원 관련 데이터를 생성 및 저장하는 웹 애플리케이션을 구현하여 기본적인 도서관리 기능 및 고객 기반의 도서 추천 및 리뷰 서비스등을 제공한다.

(2) 서비스는 도서판매와 도서대여를 모두 제공하는 것으로 한다.
도서 판매 서비스 측면에서 재고, 입고, 출고 수량 관리와 가격을 데이터베이스에 저장하여 서비스 관를 관리한다.
도서 대여 서비스 측면에서 재고 여부, 반납 여부, 대여여부, 대여일자 관리, 반납일자관리 등의 데이터를 저장하여 서비스를 관리한다.

(3) 주요사항으로 사용자에게 최적화된 도서 서비스를 제공하기 위해 도서별 별점 리뷰, 리뷰 코멘트 등을 DB에 저장할 수 있도록 하고, 이를 통해 양질의 서비스 제공을 목표로 한다.

(4) 또한 도서 검색시 검색 데이터를 일정 주기의 임시파일로 저장하여 도서별 검색 된 키워드를 제공하여 해당 도서를 찾는 유저의 관심사를 공유할수 있도록 하고, 검색 키워드별 관련 도서를 제공하여 관련도서의 판매 대여 서비스를 촉진할 수 있다. 이는 트렌드를 반영하는 독서문화를 고려하여 사용자에게 적절한 서비스를 제공하기 위함이다.

3. 프로젝트 수행 과정

- 기본 기능과 사용자 편의성을 갖는 새로운 기능을 추가하기 위한 Brainstorming 단계
- 자바의 콘솔창, mysql DB를 이용한 소스코드 설계 단계
- Servlet 을 사용하여 웹페이지와 상호작용 단계

4. BrainStorming

- CRUD 기능을 중심으로 세부 기능 고안

- CRUD 기능에서 확장된 응용 기능 고안

(1) CRUD 기능 설계안

Create(관리자 측면의 도서관리 DB생성, 사용자 측면의 DB 생성, 도서별 DB생성)

- DB t_book 생성
- Column, row 추가기능
- 도서 별 연체율, 대여여부, 빈도, 가격 등의 기타 정보 insert

Read (고객측면 제목, 출판사, 저자, 써머리, 카테고리, 관리자 검색, 도서별 검색)

- 기본 검색 엔진 : 검색 항목을 설정하고 세부 검색
- 결과 내 재검색 기능
- 연관검색 : 사용자의 검색 키워드를 통해 연관된 다른 도서 검색
- 키워드 검색 : 도서의 summary 등을 참고하여 키워드로 도서검색
- 추천검색1 : 사용자의 검색 데이터를 기반하여 관련된 도서 추천
- 추천검색2: 사용자의 검색 데이터에 기반하여 도서별 키워드 추천
- 검색 빈도수, 대여수량 , 연체율등에 따른 입고주문

Update (DB에서 1개 테이블과 일자별 메모장을 활용 실시간 업데이트)

- 일자별 메모장의 Temp.data를 DB로 업데이트 by 크론탭
- 크론탭으로 데이터베이스에 연체료 실시간 갱신
- 예약일자 만료시 예약 초기화
- 카테고리별, 책별, 기타 topic 에 대한 통계치 update

Delete

- 절판 등 Row 삭제
- 회원 탈퇴 , 반납 등의 상황에서 DB에서 데이터 삭제

(2) 창의성, 편의성을 위한 기능 설계안

- 책이름의 일부나 관심사를 키워드로 검색
- 크론탭으로 데이터베이스에 연체료 실시간 갱신
- 카테고리 스위칭 및 뎁스 늘리기
- 다음 에디션 입고 시 전 에디션 할인
- 검색 빈도, 대출 빈도 저장
- 빈도 수 기반 추천서비스, 자동입고 서비스
- 책 표지, 페이지 수 컬럼 추가
- 유저 정보 기반 추천 서비스
- 책 리뷰 서비스
- 예약일자 지나면 예약일자 초기화

- MBTI 기반 추천 서비스
- 도서 많이 구입하면 할인
- 대출 이력 기반 소개팅 서비스
- 통계

5. 기능 설명

- 회원 가입
  - html 폼으로 사용자이름을 받도록 함
  - request에 받은 정보를 저장하기위해 userRepository를 싱글톤으로 구현

- 회원 목록
  - UserRepository에 저장된 모든 사용자가 담긴 리스트를 반환하는 getUser 메소드 구현
  - /lists와 매핑된 ListServlet에 getUser 메소드를 호출하여 request에 정보를 담는다.
  - ListServlet과 매핑된 list.jsp에서 request에서 정보를 파싱하여 렌더링

- 도서 등록
  -insert 쿼리를 사용자로 부터 html 폼으로 리퀘스트를 받아서 insert쿼리에 정보를 담아서 보낸다

- 도서 검색
  - html에서 request를 받아서 파라미터로 설정하고, mysql로 전달할 쿼리문에 대입한다..

- 도서 삭제
  관리자용 서비스로 html에서 Request를 받아 도서 삭제, 반납 등의 상황에 DB에서 데이터를 삭제한다.

- 도서 수정
  update 쿼리 사용자로부터 html 폼으로 전달받는다. 이때 기존의 데이터를 표시하여 데이터 수정이 용이하도록 한다. 이를 통해 update 쿼리에 정보를 담아서 보낸다.

- 도서 평점 등록
  도서별로 등록된 리뷰를 조회하여, 평균값을 사용자에게 표시해준다.이를 위해 distinct를 이용하여 도서 테이블에 있는 도서 제목을 가져온다. 이를 통해 중복되지 않고 도서 제목별 평점의 합계를 구하고 평점을 계산한다.

- 도서 대출
  싱글톤으로 인스턴스를 생성하여 대출 이력을 관리한다. 등록된 사용자인지 체크하고 도서 수량을 확인한 후 대출이 완료되면 재고 수량을 마이너스 해준다.

- 도서 반납

대출에서 생성한 인스턴스를 바탕으로, 조회된 도서를 제거한다. 도서제거 후 대출한 도서의 수량을 플러스 해준다.

- 연체료 계산
  대여일자와 반납일자를 기준으로 레포지토리에 싱글톤으로 사용자를 설정하여 연체료를 반영한다.

- 도서 검색 키워드 제공 & 키워드 별 도서 목록 조회
  도서 검색시 검색 키워드와 검색된 도서의 pk를 맵핑, 저장하여 temporary 파일의 형식으로 저장한다. 이를 통해 도서별 사용자 기반의 키워드 데이터를 제공할수 있고, 키워드별 연관 도서를 제공할 수 있다

- 사용자 서버와 관리자 서버 각각 해당 서버에서 데이터베이스에 접근하는 경우 웹에서 접근가능한 항목을 구별하여 제공한다.

6. CRUD Code

```
public LinkedList<Book> getBook(BookListParamVo pv) throws SQLException {
    JdbcComm jdbc = new JdbcComm();
    String query = "SELECT book_id, category_1, category_2, qty, book_name, summary, author\n" +
            "FROM t_book\n" +
            "WHERE book_name LIKE ?" +
            "   OR author LIKE ?" +
            "   OR summary LIKE ?" +
            " ORDER BY book_id DESC " +
            "  LIMIT ?, ?;";
```

```
public int getBookTotal(BookListParamVo pv) throws SQLException {
    JdbcComm jdbc = new JdbcComm();
    int count = 0;
    String query = "SELECT COUNT(*) AS 'count' " +
            "FROM t_book\n" +
            "WHERE book_name LIKE ?" +
            "   OR author LIKE ?" +
            "   OR summary LIKE ?;";
```

```
public Book getBookQtyByBookName(String name) throws SQLException {
    JdbcComm jdbc = new JdbcComm();
    Book book = new Book();
    // 쿼리 실행
    String query = "SELECT qty, book_id, purchase_price, selling_price FROM t_book WHERE book_name = ?";
```

```
public Book getBookInfo(int book_id) throws SQLException {
    JdbcComm jdbc = new JdbcComm();
    String query = "SELECT category_1, category_2, book_name, summary, author," +
            " publisher, purchase_price, selling_price, qty, page " +
            "FROM t_book WHERE book_id = " + book_id;
```

```
public static HashMap<String, Integer> getRate() throws SQLException {
    JdbcComm jdbc = new JdbcComm();
    Statement statement = jdbc.getConnection().createStatement();
    ResultSet resultSet = null;
    HashMap<String, Integer> map = new HashMap<>();
    String query = "SELECT DISTINCT book_name FROM t_book; ";
    LinkedList<String> liBook = new LinkedList<>();
```

```
for (String bookName : liBook) {
    query = "SELECT AVG(rate) AS avg FROM book_review WHERE book_name = '" + bookName + "'";
    resultSet = statement.executeQuery(query);
    if (resultSet.next()) {
        map.put(bookName, resultSet.getInt( columnLabel: "avg"));
    }
```

```java
public int insert(Book book) throws SQLException {
    JdbcComm jdbc = new JdbcComm();
    Connection connection = jdbc.getConnection();

    String insertQuery = "INSERT INTO t_book (category_1, category_2, book_name, summary, author, publisher, purchase_price, selling_price, qty, page) " +
            "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
```

```java
public static int insert(BookReview bookReview) throws SQLException {
    JdbcComm jdbc = new JdbcComm();
    Connection connection = jdbc.getConnection();

    String insertQuery = "INSERT INTO book_review (user_name, book_name, book_id, rate) VALUES (?, ?, ?, ?)";
```

```java
public void updateBook(Book book, int bookId) throws SQLException {
    JdbcComm jdbc = new JdbcComm();
    String query = "UPDATE t_book SET " +
            "category_1 = ?, " +
            "category_2 = ?, " +
            "book_name = ?, " +
            "summary = ?, " +
            "author = ?, " +
            "publisher = ?, " +
            "purchase_price = ?, " +
            "selling_price = ?, " +
            "qty = ?, " +
            "page = ? " +
            "WHERE book_id = " + bookId;
```

```java
public int updateQty(Book book) throws SQLException {
    JdbcComm jdbc = new JdbcComm();
    Connection connection = jdbc.getConnection();
    // Update the row with the new values
    String updateQuery = "UPDATE t_book SET qty = ? WHERE book_id = ?";
```

```java
public void deleted(int bookId) throws SQLException {
    JdbcComm jdbc = new JdbcComm();
    Statement statement = jdbc.getConnection().createStatement();

    String query = "DELETE FROM t_book WHERE book_id = " + bookId;
    statement.execute(query);
}
```