CH 12

데이터베이스와 제이쿼리 활용

1. [실습] 도서관리 웹앱 개발 요약

>> 학습목표 <<

❖웹 SQL 데이터베이스 API 사용 방법을 익힌다.

❖제이쿼리 모바일과 DB를 활용한 예제를 만들어본다.

1.1 로컬 DB 기반의 도서관리 앱 : bookApp

- HTML5의 기본 로컬 저장소
  - 웹 스토리지(Web Storage) : 키-값 저장 형식
    - 적은 양의 간단한 데이터를 저장하기에 적합
  - 웹 SQL 데이터베이스(Web SQL Database) : 테이블 저장 형식
    - HTML5 API 지원 표준에서는 제외되었지만 아직도 유효하고 효과적인 저장소
    - 구조적이고 체계화된 많은 양의 데이터를 저장하기에 적합
    - 관계형 데이터 모델을 통해 데이터를 안정적이고 유연하게 다룰 수 있음
  - 인덱스 데이터베이스(Indexed Database) : 객체 저장 형식
    - 새로운 표준으로 제시
    - SQL 언어와는 무관하며 단순한 저장 구조를 사용
    - 간단한 자바스크립트 API 만으로도 데이터베이스 조작이 가능하고 표준화가 용이함

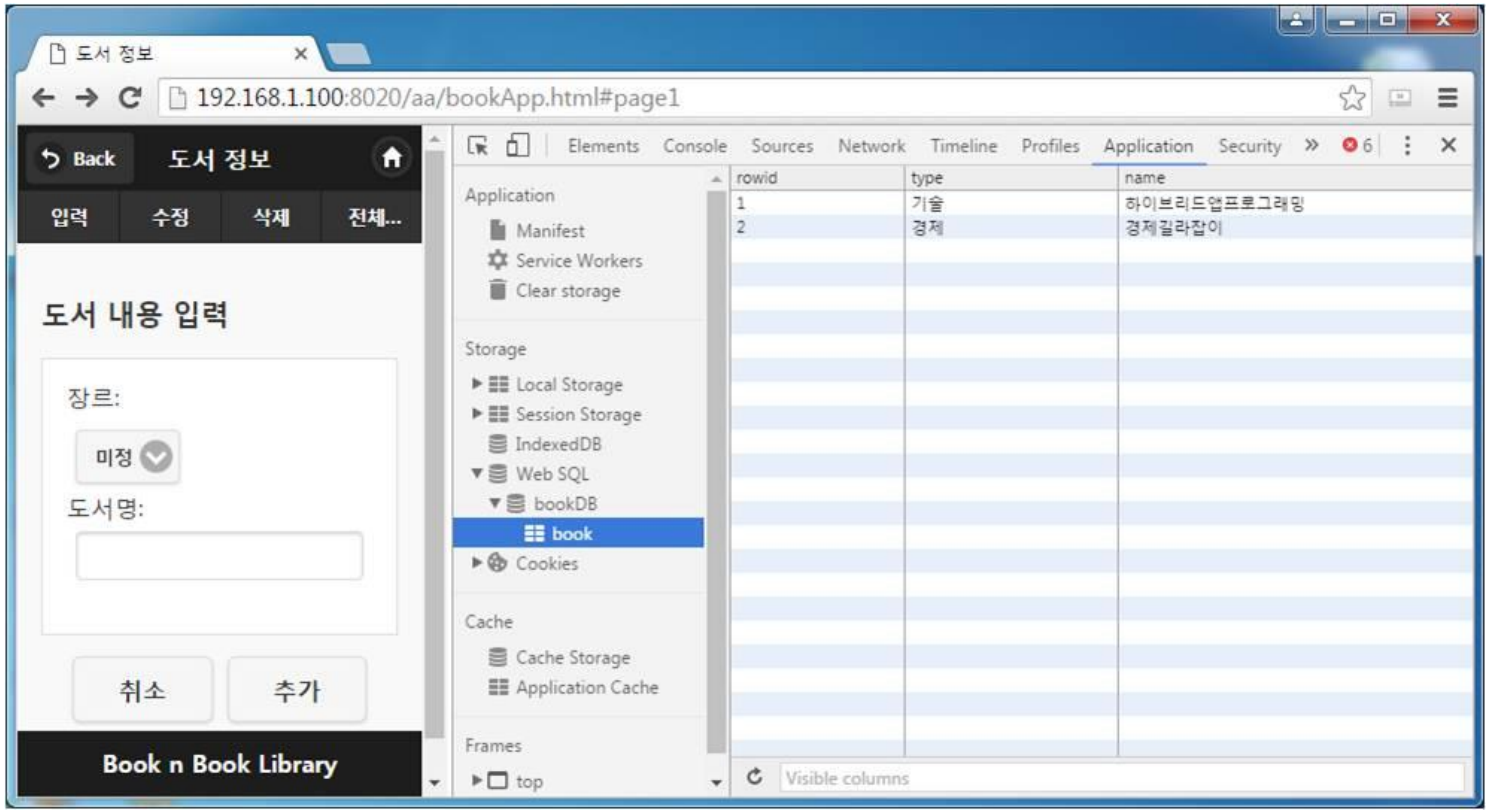
- SQLite DB
  - 최근 웹 SQL 데이터베이스를 대부분의 모바일 브라우저가 DB 엔진으로 내장
  - 서버 DB 기반의 앱으로 자연스럽게 확장 가능

1.2 UI 화면 구성

- 도서관리 앱(bookApp)의 화면 구성

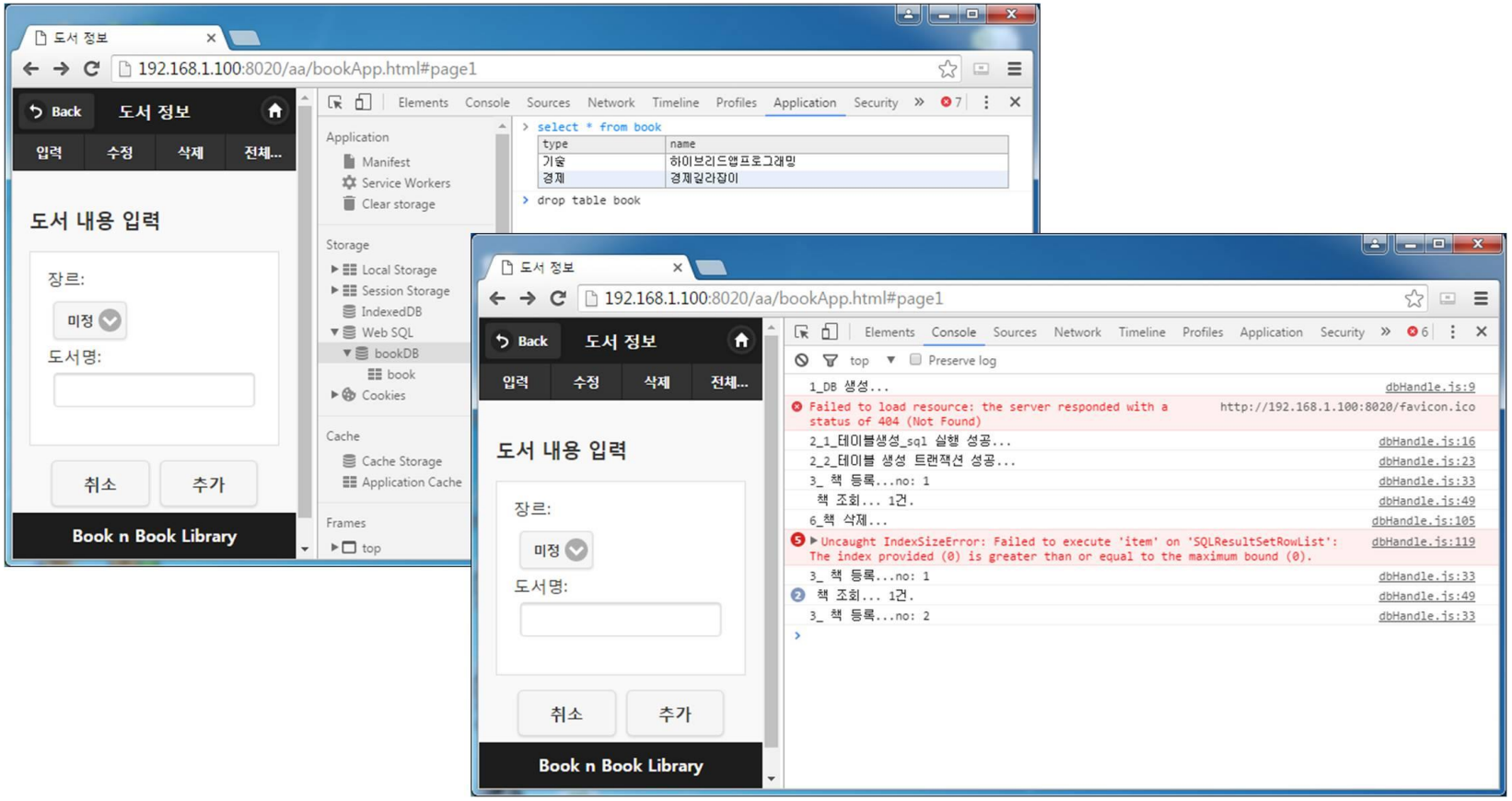
SQLite 활용

- 크롬 브라우저 안에 SQLite 엔진을 내장
  - 브라우저 개발자 도구의 'Application' 패널을 이용, 데이터베이스 내용을 확인하고 관리
  - 예제 'bookDB' 데이터베이스 안의 'book' 테이블 정보를 확인



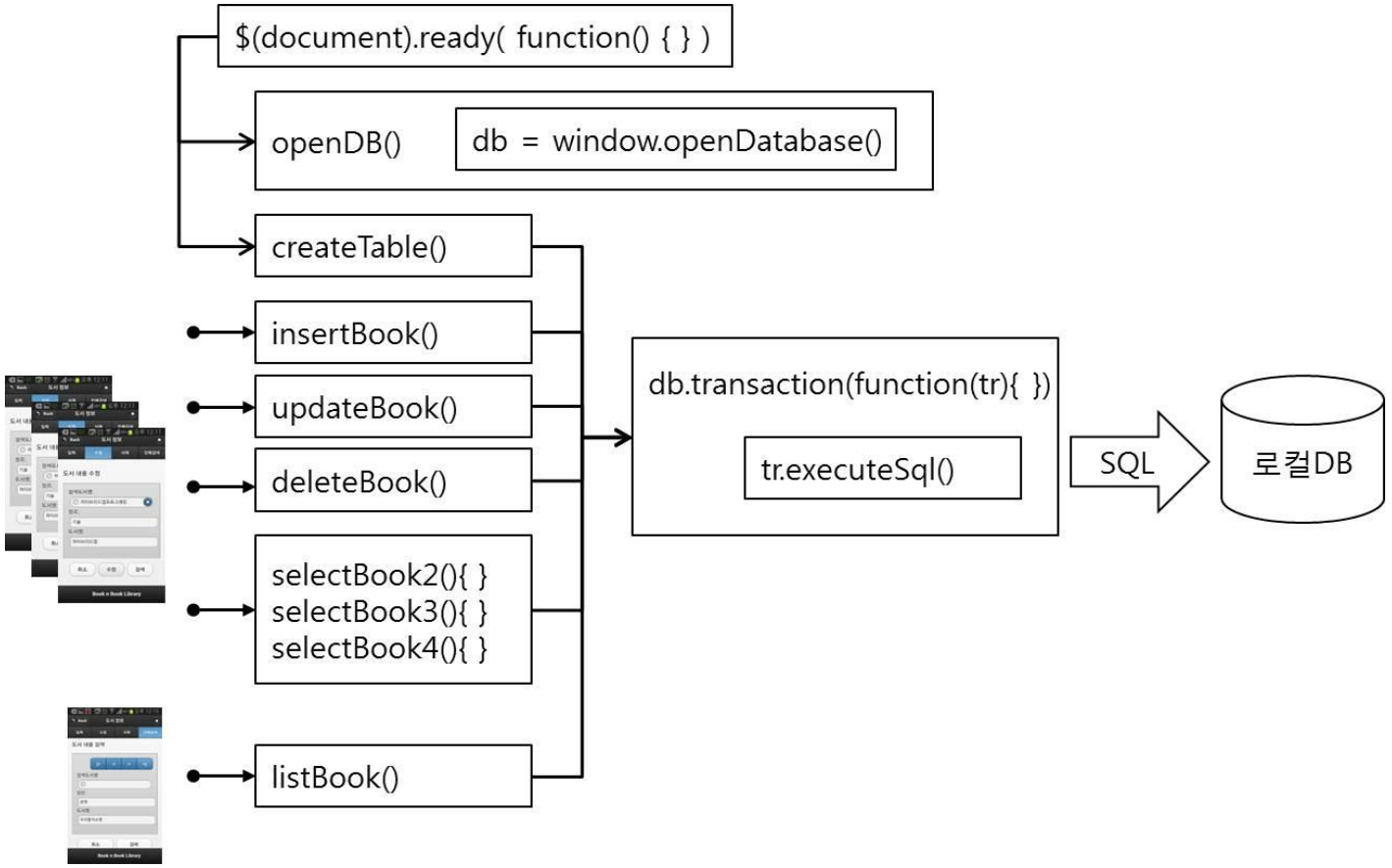
SQLite 활용

- 패널에서 직접 SQL 문장을 작성하여 실행하고 그 결과 확인 가능
  - 잘못 생성된 테이블을 삭제하는 등의 기본적인 데이터베이스 관리 가능
  - 'console' 창에서는 bookApp 앱 실행에 따른 처리 과정을 메시지를 통해 확인
  - 필요한 메시지를 코드 안에 추가하여 앱 프로그램 디버깅할 때 활용



1.3 내부 스크립트 함수 구성

- bookApp 앱의 스크립트 함수 구성



HTML5 웹 SQL 데이터베이스 관련 API(1)

- DB 생성 : openDatabase( ) 메소드

```
window.openDatabase( "DB이름", "버전", "DB표시이름", DB크기 [, 성공콜백함수] ) ;
```

[예제12-3] 데이터베이스 생성 및 열기 **dbHandle.js의 일부**

```
function openDB(){
    db = window.openDatabase("bookDB", "1.0", "북DB", 1024*1024);
                . . . 생략 . . .
 }
```

- DB 트랜잭션 실행 : transaction( ) 메소드

```
db.transaction( function(tr) { 트랜잭션실행 콜백함수 내용 }
                        [, function(err) { 트랜잭션실패 콜백함수 내용 } ]
                        [, function() { 트랜잭션성공 콜백함수 내용 } ]
);
```

HTML5 웹 SQL 데이터베이스 관련 API(2)

- DB 명령어(SQL) 실행 : executeSql( ) 메소드

```
tr.executeSql( SQL문 [, 치환값]
                        [, function(tr, rs) { SQL실행성공 콜백함수 내용 } ]
                        [, function(err) { SQL실행실패 콜백함수 내용 } ]
);
```

[예제12-4] 테이블 생성 트랜잭션 실행하기 **dbHandle.js의 일부**

```
function createTable() {
    db.transaction(function(tr){
                    var createSQL = "create table if not exists book(type text, name text)";
                    tr.executeSql(createSQL, [], function(){
                . . . 생략 . . .
}
```

SQL

- SQL(Structured Query Language)
  - 관계형 데이터베이스(RDB)를 위한 표준 데이터베이스 언어
  - SQLite도 표준 SQL을 지원

• 테이블 생성

```
create table ( 컬럼명1 컬럼유형1, 컬럼명2 컬럼유형2, . . . , 컬럼명n 컬럼유형n )
```

• 테이블 행 입력

```
insert into 테이블명[ ( 컬럼명1, 컬럼명2, . . . , 컬럼명n ) ]
values ( 입력값1, 입력값2, . . . , 입력값n )
```

• 테이블 행 수정

```
update 테이블명
set 컬럼명1 = 수정값1, 컬럼명2 = 수정값2, . . . , 컬럼명n = 수정값n
[ where 컬럼명 비교연산자 조건값 ]
```

• 테이블 행 삭제

```
delete from 테이블명
[ where 컬럼명 비교연산자 조건값 ]
```

• 테이블 행 검색

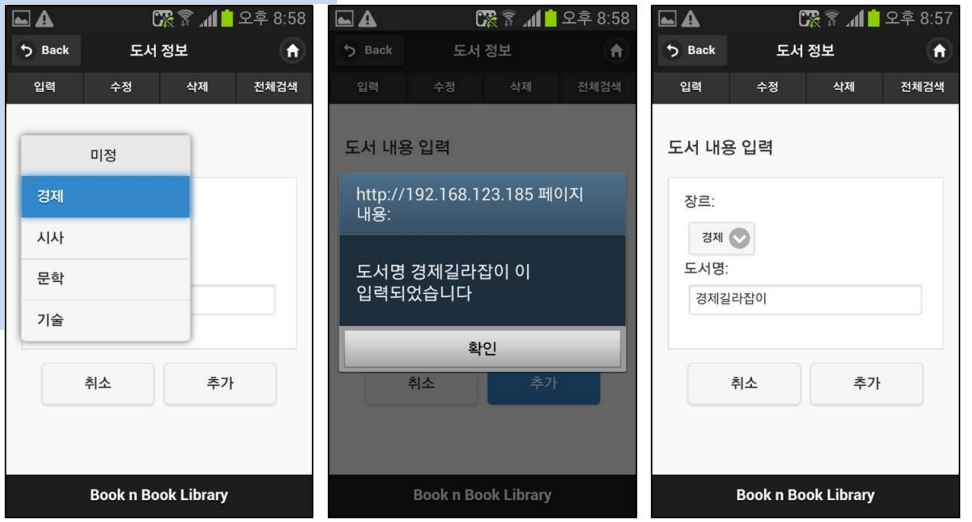
```
select 컬럼명1, 컬럼명2, . . . , 컬럼명n
from 테이블명
[ where 컬럼명 비교연산자 조건값 [ 논리연산자 컬럼명 비교연산자 조건값 ] . . . ]
```

1.4 데이터베이스 관련 함수 정의

- 입력 트랜잭션 처리 함수 예
  - [그림 12-5] 스마트폰 실행 결과_bookApp 입력 화면(예제12-5)

[예제12-5] 데이터 입력 트랜잭션 실행하기 **dbHandle.js의 일부**

```
function insertBook(){
   db.transaction(function(tr){
      var type = $('#bookType1').val();
      var name = $('#bookName1').val();
      var insertSQL = 'insert into book(type, name) values(?, ?)';
      tr.executeSql(insertSQL, [type, name],
            function(tr, rs){
                  console.log('3_ 책 등록...no: ' + rs.insertId);
                  alert('도서명 ' + $('#bookName1').val() + ' 이 입력되었습니다');
                  $('#bookName1').val('');
                  $('#bookType1').val('미정').attr('selected', 'selected');
                  $('#bookType1').selectmenu('refresh');
            }, function(tr, err){
                  alert('DB오류 ' + err.message + err.code);
            }
      );
   });
}
```



1.4 데이터베이스 관련 함수 정의

- 검색 트랜잭션 처리 함수 예
  - [그림 12-6] 스마트폰 실행 결과_bookApp 전체 검색 화면(예제12-6)

[예제12-6] 전체 데이터 검색 트랜잭션 실행하기 **dbHandle.js의 일부**

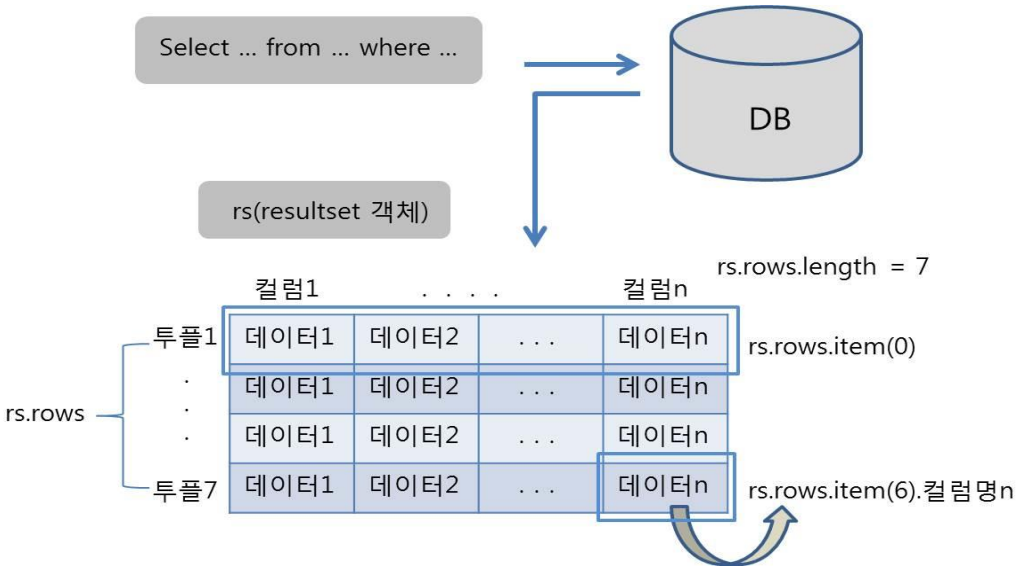
```
function listBook(){
  db.transaction(function(tr){
    var selectSQL = 'select * from book';
    tr.executeSql(selectSQL, [], function(tr, rs){
      console.log(' 책 조회... ' + rs.rows.length + '건.');
      if (position == 'first') {
        if (index == 0)
          alert('더 이상의 도서가 없습니다');
        else
                index = 0;
      }
      else if (position == 'prev') {
        if (index == 0)
                alert('더 이상의 도서가 없습니다');
        else
                index = --index;
      }
      else if (position == 'next') {
        if (index == rs.rows.length-1)
                alert('더 이상의 도서가 없습니다');
        else
                index = ++index;
      }
      else if (position == 'last') {
      {
        if (index == rs.rows.length-1)
                alert('더 이상의 도서가 없습니다');
        else
                index = rs.rows.length-1;
      }
      $('#bookType4').val(rs.rows.item(index).type);
      $('#bookName4').val(rs.rows.item(index).name);
    });
  });
}
```

SQL 결과 집합 : rs 객체

- SQL 결과 집합(ResultSet)
  - SQL 문장을 수행한 결과로 받게 되는 DB 반환 객체

rs 객체의 속성	기능
rs.insertId	insert문 실행 결과로 자동 생성된 ID
rs.rowAffected	update, delete문 실행 결과로 처리된 투플(행)의 수
rs.rows	select문 실행 결과로 생성된 결과 집합 안의 투플(행) 리스트(객체)

rows 객체의 속성	기능
rows.length	결과 집합 안의 투플(행)들의 전체 수
rows.item(첨자)	결과 집합 안의 첨자에 해당하는 투플(행)
rows.item(첨자).컬럼명	결과 집합 안의 첨자에 해당하는 투플(행)의 특정 컬럼값



이벤트 핸들러 함수 정의하기

- 이벤트 핸들러 함수 예
  - 검색, 수정, 삭제, 취소 등의 버튼 엘리먼트에 대해 클릭 이벤트가 발생할 때 필요한 정보를 얻거나 설정하는 콜백 함수를 이벤트 핸들러로 선언
  - 전체 검색 화면의 처음, 이전, 다음, 마지막 버튼 클릭했을 때 이벤트 콜백 함수도 선언

[예제12-7] 이벤트 핸들러 함수 정의하기 **bookApp.html의 일부**

```
$(document).ready( function() {                    // 문서 준비 이벤트 핸들러
        openDB();
        createTable();

        $('#submit1').click( function(){           // 입력 요청 이벤트 핸들러
                insertBook();
        });
        $('#cancel1').click( function(){           // 취소 요청 이벤트 핸들러
                $('#bookType1').val('미정').attr('selected', 'selected');
                $('#bookType1').selectmenu('refresh');
        });
        $('#search2').click( function(){           // 검색 요청 이벤트 핸들러
                var name = $('#sBookName2').val();
                selectBook2(name);
        });
        . . . 생략 . . .
        $('#last').click( function(){              // 이동 요청 이벤트 핸들러
                position='last';
                listBook();
        });
});
```