교과목 포트폴리오

20202263 조재호

목차

Part 1,

졸업작품 수행 중 학습 내용

졸업작품 수행 중 학습내용

네이버 API

NAVER OpenAPI

네이버 검색 API : 책 적용

APP 등록을 한 뒤, Client ID와 Client Secret 값을 받아 요청

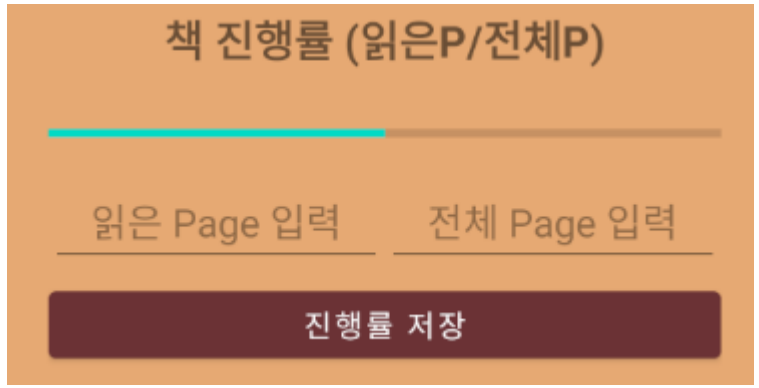
요청 변수 : query, display, start, sort

주요 출력 결과 : title(책 제목), image(썸네일), author(저자), price(가격)

졸업작품 수행 중 학습내용

ProgressBar

나만의 서재 My Books 기능에서 사용
책 진행률을 저장




```
edtin = findViewById(R.id.edt_in);     // 읽은 페이지 수 edittext
edtfull = findViewById(R.id.edt_full); // 전체 페이지 수 edittext
progressBar = findViewById(R.id.progressBar);
btnst = findViewById(R.id.btn_st); // 계산 버튼

btnst.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        full = Integer.parseInt(edtfull.getText().toString());
        progressBar.setMax(full);      // progressbar의 max값을 full(전체 페이지 수)로 설정
        in = Integer.parseInt(edtin.getText().toString());
        progressBar.setProgress(in); // progressbar의 min값을 in(읽은 페이지 수)로 설정
    }
});
```

Progressbar는 Max와 Progress로 진행률을 나타냄

EditText를 이용해 Max와 Progress를 입력한 값으로 설정해 진행 정도를 시각화

졸업작품 수행 중 학습내용

Firebase 회원가입

```
mBtnRegister.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // 회원가입 처리 시작
        String strEmail = mEtEmail.getText().toString();
        final String strPwd = mEtPwd.getText().toString();

        if(strEmail.length()>0 && strPwd.length()>0 ) {

                // firebase auth 진행
                mFirebaseAuth.createUserWithEmailAndPassword(strEmail, strPwd).addOnCompleteListener(RegisterActivity.this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            FirebaseUser firebaseUser = mFirebaseAuth.getCurrentUser();
                            UserAccount account = new UserAccount();
                            account.setIdToken(firebaseUser.getUid());
                            account.setEmailId(firebaseUser.getEmail());
                            account.setPassword(strPwd);

                            //setValue: 데이터베이스 삽입

                            mDatabadeRef.child("UserAccount").child(firebaseUser.getUid()).setValue(account);

                            Toast.makeText(RegisterActivity.this, "회원가입에 성공했습니다.", Toast.LENGTH_SHORT).show();
                        } else {
                            Toast.makeText(RegisterActivity.this, "회원가입에 실패했습니다.", Toast.LENGTH_SHORT).show();
                        }
                    }
                });

        }
```

졸업작품 수행 중 학습내용

Firebase Login

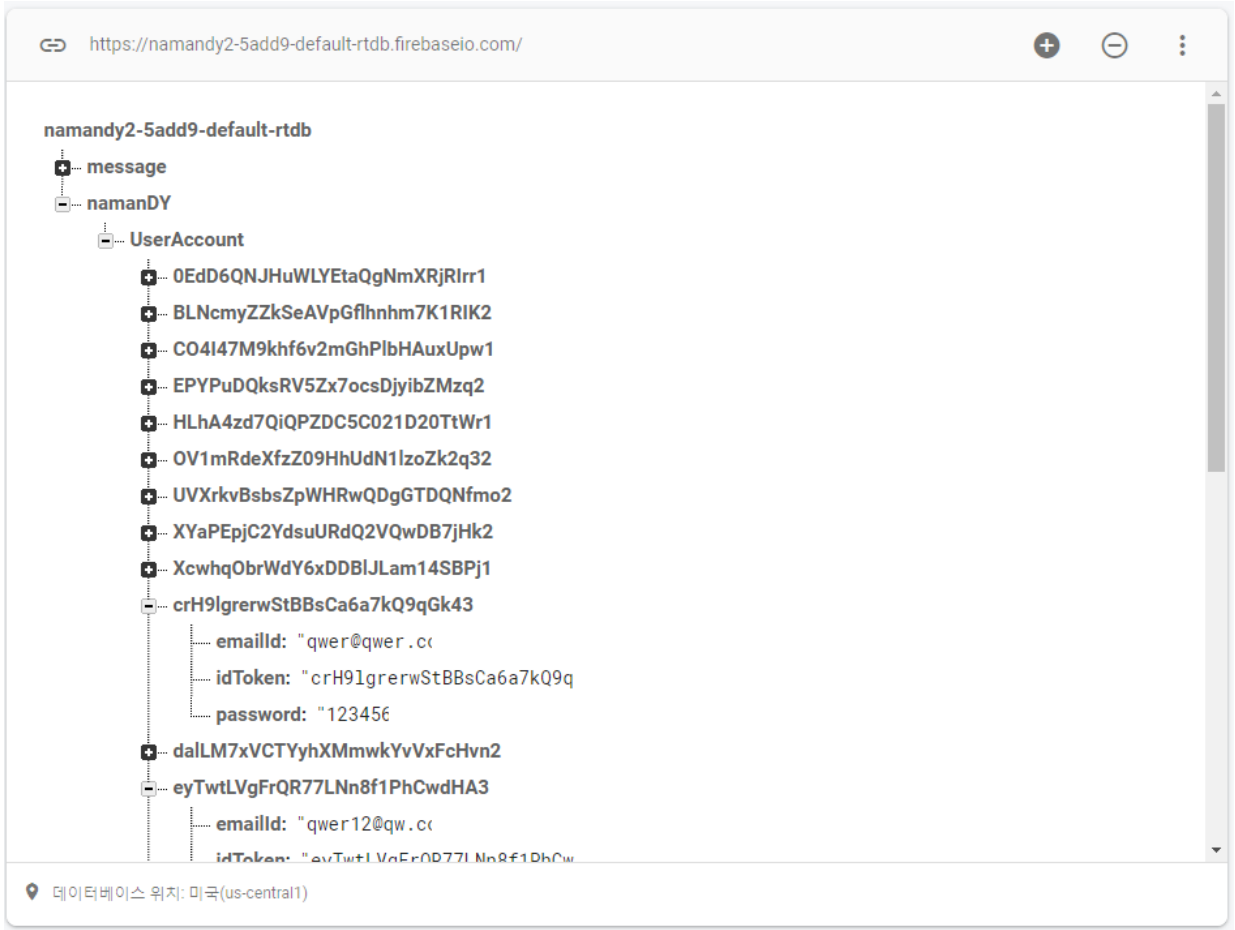
```
TextView text_login = findViewById(R.id.text_login);
text_login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //로그인 요청
        String strEmail = mEtEmail.getText().toString();
        final String strPwd = mEtPwd.getText().toString();

        mFirebaseAuth.signInWithEmailAndPassword(strEmail, strPwd).addOnCompleteListener(LoginActivity.this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // 로그인 성공
                    Intent intent = new Intent(LoginActivity.this, MoveActivity.class);
                    startActivity(intent);
                    finish();

                } else {
                    Toast.makeText(LoginActivity.this, "로그인 실패", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
});
```

제목을 입력하세요

Firebase DB

```
public class UserAccount {

    private String idToken;    // firebase Uid (고유 토큰정보)
    private String emailId;    // 이메일 아이디
    private String password;   // 비밀번호

    public UserAccount() { }

    public String getIdToken() { return idToken; }

    public void setIdToken(String idToken) { this.idToken = idToken; }

    public String getEmailId() { return emailId; }

    public void setEmailId(String emailId) { this.emailId = emailId; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }

}
```

Class UserAccount



Firebase Realtime Database

Part 2,
깃과 깃 허브 학습 내용

Part 2,

깃과 깃허브

git : 분산 버전 관리 시스템

GitHub : 깃의 저장소 및 관리 서비스

깃과 깃허브

Create a Repository

From scratch -- Create a new local repository
```
$ git init [project name]
```

Download from an existing repository
```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed
```
$ git status
```

Show the changes to files not yet staged
```
$ git diff
```

Show the changes to staged files
```
$ git diff --cached
```

Show all staged and unstaged file changes
```
$ git diff HEAD
```

Show the changes between two commit ids
```
$ git diff commit1 commit2
```

List the change dates and authors for a file
```
$ git blame [file]
```

Show the file changes for a commit id and/or file
```
$ git show [commit]:[file]
```

Show full change history
```
$ git log
```

Show change history for file/directory including diffs
```
$ git log -p [file/directory]
```

Working with Branches

List all local branches
```
$ git branch
```

List all branches, local and remote
```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory
```
$ git checkout my_branch
```

Create a new branch called new_branch
```
$ git branch new_branch
```

Delete the branch called my_branch
```
$ git branch -d my_branch
```

Merge branch_a into branch_b
```
$ git checkout branch_b
$ git merge branch_a
```

Tag the current commit
```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit
```
$ git add [file]
```

Stage all changed files, ready for commit
```
$ git add .
```

Commit all staged files to versioned history
```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history
```
$git commit -am "commit message"
```

Unstages file, keeping the file changes
```
$ git reset [file]
```

Revert everything to the last commit
```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)
```
$ git fetch
```

Fetch the latest changes from origin and merge
```
$ git pull
```

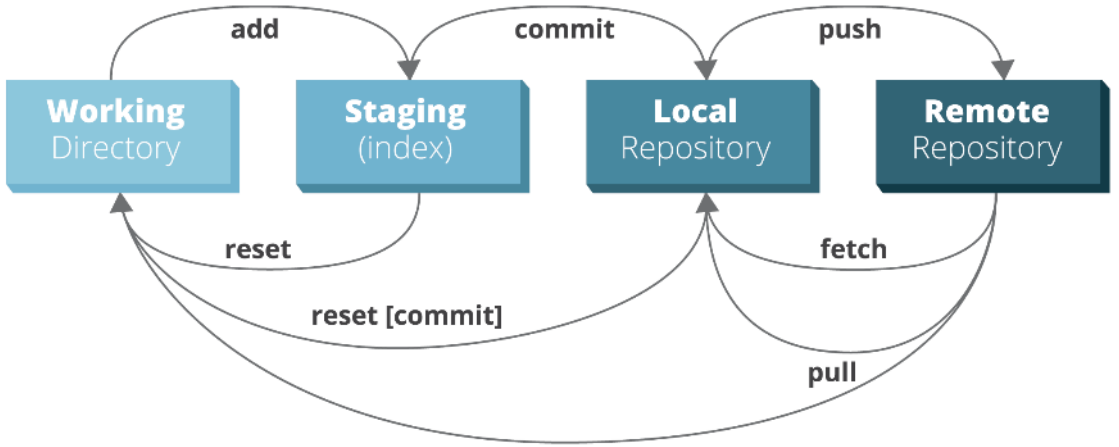
Fetch the latest changes from origin and rebase
```
$ git pull --rebase
```

Push local changes to the origin
```
$ git push
```

Finally!

When in doubt, use git help
```
$ git command --help
```

Or visit https://training.github.com/ for official GitHub training.




Thank You