PBL 2 - 파이어베이스 수행 결과 보고서

목차

파에어베이스 연동

eclass에 올라온 파이어베이스 영상 참고

파이어베이스 데이터 저장

테이블 생성

- 데이터베이스에 데이터를 쓰려면 `DatabaseReference` 가 필요하다.

mDatabase = FirebaseDatabase.getInstance().getReference();

- Product 클래스에서 상품번호,이름,카테고리,가격,상품 정보의 변수를 넣습니다.

```
public class Product {
    public String category;
    public String name;
    public int price;
    public String product_info;
    public int product_num;
    public Product() {    }
    public Product(int product_num, String name, String category,int price ,
                   String product_info) {
          this.product_num = product_num;
          this.name = name;
          this.category = category;
          this.price = price;
          this.product_info = product_info;
      }
    }
```

- `setValue()` 로 상품을 추가할 수 있다.

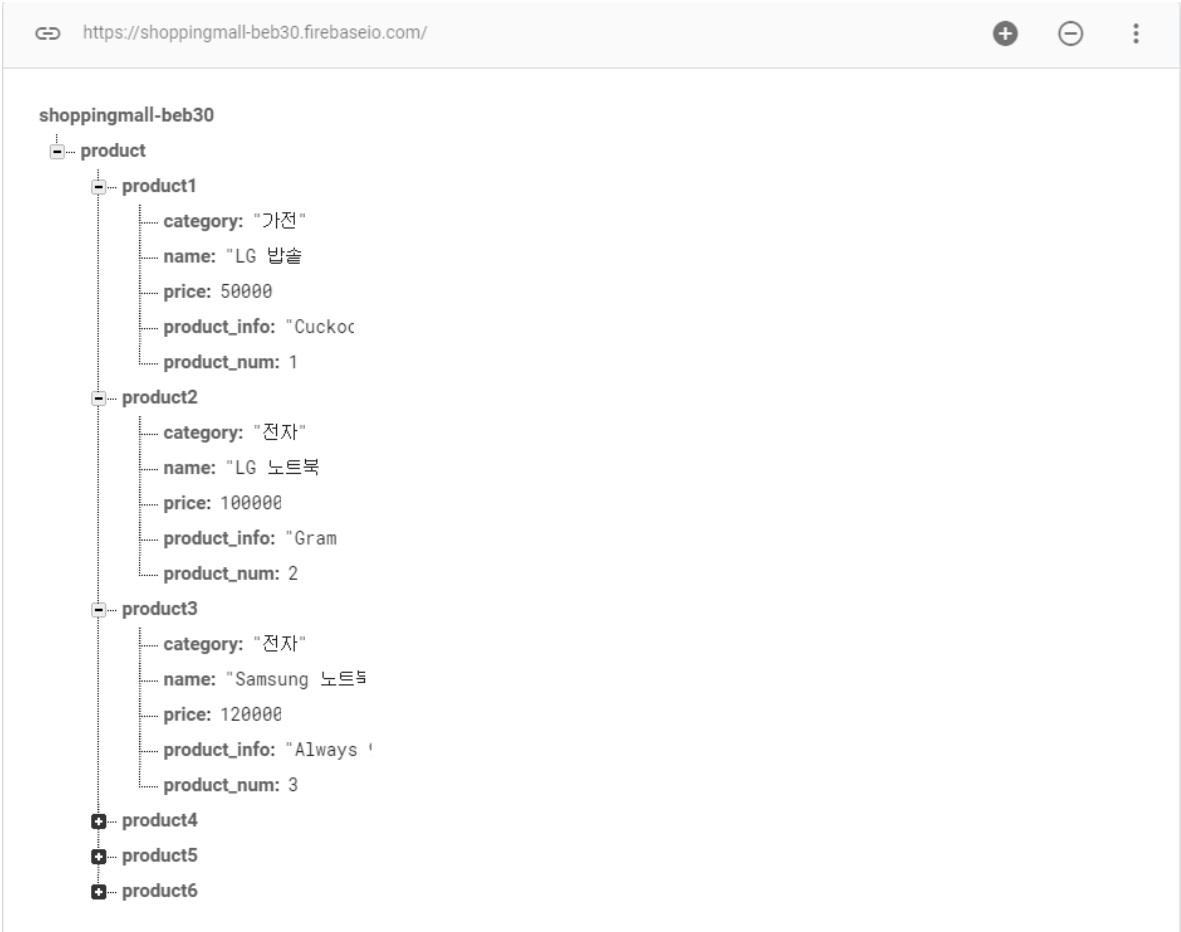
```
private void writeProduct(String productId,int product_num, String name,
                          String category,int price ,String product_info){
   Product product = new Product(product_num,name,category, price
,product_info);

   mDatabase.child("product").child(productId).setValue(product);
 }
```

- 상품에 대한 정보를 적는다.

```
writeProduct("product1",1,"LG 밥솥","가전",500000,"Cuckoo");
writeProduct("product2",2,"LG 노트북","전자",1000000,"Gram");
writeProduct("product3",3,"Samsung 노트북","전자",1200000,"Always 9");
writeProduct("product4",1,"Samsung 세탁기","가전",950000,"Bubble");
writeProduct("product5",1,"Apple 노트북","전자",2000000,"Mac");
writeProduct("product6",1,"Apple 핸드폰","전자",1100000,"11 pro");
```

- 애뮬레이터가 실행되면 파이어베이스에서 데이터가 써진다.



파이어베이스 데이터 저장 코드 수정

```
private void writeProduct(String productId, int product_num, String name, String
category, int price, String product_info) {
            Map<String, Object> childUpdates = new HashMap<>();
            Map<String, Object> postValues = null;
            Product product = new Product(product_num, name, category, price,
product_info);
            postValues = product.toMap();
            childUpdates.put("/product/" + productId, postValues);
            mDatabase.updateChildren(childUpdates);
        }
```

- 각 상품 아이템 데이터를 writeProduct함수에 매개변수로 넣어주어 호출
- Product 객체 생성, writeProduct로 받은 변수를 Product 객체에 넣음
- product.toMap() 으로 미리 생성해둔 Map 형식 postValues 에 넣는다
- childUpdates 객체(Map) 에 key값으로 파이어베이스 데이터 경로명, value값으로 postValues 객체를 준다
- firebaseDatabase Reference 값을 가진 mDatabase에 childUpdates를 파이어베이스에 `updateChildren()` 으로 업데이트 한다.

```
sortbyProduct =
FirebaseDatabase.getInstance().getReference().child("product").orderByChild("nam
e");
sortbyProduct.addListenerForSingleValueEvent(productListener);
```

- Firebasebase의 `child("product")` 로 product 까지 내려간 후 각 데이터 별 name 필드를 기준으로 정렬한다.
- `addListenerForSingleValueEvent(productListener)` 로 데이터가 변경될때마다 한번씩 productListener 함수를 호출한다.

파이어베이스 데이터 읽어온 후 정렬하기

```
//dataSanpshot
ValueEventListener productListener = new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    for (DataSnapshot postSnapshot : dataSnapshot.getChildren())
{
                        //String key=postSnapshot.getKey();
                        Product get = postSnapshot.getValue(Product.class);

                        if (get.category.equals("가전"))
                            addAppItem(getDrawable(R.mipmap.ic_launcher),
                                    get.name, String.valueOf(get.price));
                        else
                            addPhoneItem(getDrawable(R.mipmap.ic_launcher),
                                    get.name, String.valueOf(get.price));
                    }

                }
```

- google-services.json 으로 파이어베이스에서 미리 저장되어있는 데이터를 받아온 후 DataSnapshot이 `getChildren()` 을 사용해 모든 노드를 캐치한다

- for문으로 데이터베이스에 order된 순서대로 하나씩 데이터를 처리한다
    - Product 객체를 생성 후 postSnapshot 값을 Product.class 형식으로 저장
    - 받아온 Product의 category 값이 "가전" 일 경우 addAppItem을 호출
    - 아니면 addPhoneItem을 호출한다

```
public void addAppItem(Drawable icon, String title, String desc) {
            ProductFromDatabase item = new ProductFromDatabase();

            item.setIcon(icon);
            item.setTitle(title);
            item.setDesc(desc);

            mAppList.add(item);
            mAllList.add(item);
        }
```

- addAppItem 함수는 카테고리 중 "가전" 인 아이템을 ProductFromDatabase 형식으로 캐스팅하여
- mAppList(가전 카테고리 리스트) 나 mAllList(전체 항목 보기) 에 넣는다

```
public void addPhoneItem(Drawable icon, String title, String desc) {
            ProductFromDatabase item = new ProductFromDatabase();

            item.setIcon(icon);
            item.setTitle(title);
            item.setDesc(desc);

            mPhoneList.add(item);
            mAllList.add(item);
        }
```

- addPhoneItem 함수는 카테고리 중 "전자" 인 아이템을 ProductFromDatabase 형식으로 캐스팅하여
- mPhoneList(전자 카테고리 리스트) 나 mAllList(전체 항목 보기) 에 넣는다
    - mPhoneList, mAppList, mAllList 는 각각의 항목(가전, 전자, 전체) 을 보여줄 때 좀 더 편리하게 보여주기 위한 것이다.

데이터 검색

- equalTo() : 키워드가 아닌 검색명이 상품 네임과 정확히 일치해야 검색이 되는 메서드였다.

```
String search=searchText.getText().toString();
Query sortbyProduct;
sortbyProduct =
FirebaseDatabase.getInstance().getReference().child("product").orderByChild(
"name").startAt(search).endAt(search+"\uf8ff");
```

- 그래서 위와 같이 수행했다.
- equalTo() 를 사용하지 못하고 startAt()과 endAt() 을 사용했다.
    - String search: 사용자가 입력한 문자열
    - startAt(): search를 받아온 후 해당 문자로 시작하는 name 노드를 검출하여 정렬

- endAt(): search로 받아온 문자열+"\uf8ff"(대부분의 문자열을 포함하는 코드)
- **최종적으로 해당 키워드가 포함된 항목을 화면에 정렬한다.**

위 코드의 문제점

- 검색한 결과 데이터를 데이터베이스에서 불러오는 코드 구현은 성공하였으나
- 받아온 데이터를 화면에 출력할때 해당 화면을 clear 하지 않았기 때문에 기존 리스트에 새로운 항목으로 추가되어 출력되는 문제가 있다.

문제점 솔루션

- 결과를 보여주기 전 기존 리스트 clear함수를 사용할 계획이다.

```
arrayData.clear();
arrayIndex.clear();
arrayAdapter.clear();
arrayAdapter.addAll(arrayData);
arrayAdapter.notifyDataSetChanged();
```

이미지 추가하기

```
Intent intent = new Intent();
intent.setType("image/*");
intent.setAction(Intent.ACTION_GET_CONTENT);
startActivityForResult(Intent.createChooser(intent, "Select Picture"),
PICK_IMAGE);
```

```
public static final int PICK_IMAGE = 1;

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (requestCode == PICK_IMAGE) {
        //TODO: action
    }
}
```

- 해당 코드 예시처럼 이미지 선택

```
// Create a storage reference from our app
StorageReference storageRef = storage.getReference();

// Create a reference to "mountains.jpg"
StorageReference mountainsRef = storageRef.child("mountains.jpg");

// Create a reference to 'images/mountains.jpg'
StorageReference mountainImagesRef = storageRef.child("images/mountains.jpg");

// While the file names are the same, the references point to different files
mountainsRef.getName().equals(mountainImagesRef.getName());    // true
mountainsRef.getPath().equals(mountainImagesRef.getPath());    // false
```

- 선택한 이미지를 업로드한 후 다운로드 URL을 받아옴

```
Map<String, Object> city = new HashMap<>();
city.put("name", "Los Angeles");
city.put("state", "CA");
city.put("country", "USA");

db.collection("cities").document("LA")
        .set(city)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                Log.d(TAG, "DocumentSnapshot successfully written!");
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Log.w(TAG, "Error writing document", e);
            }
        });
```

- DB에 저장한 뒤 필요할 경우 위의 예시처럼 다운로드 URL을 호출

```
// For a simple view:
@Override public void onCreate(Bundle savedInstanceState) {
  ...
  ImageView imageView = (ImageView) findViewById(R.id.my_image_view);

  Glide.with(this).load("http://goo.gl/gEgYUd").into(imageView);
}

// For a simple image list:
@Override public View getView(int position, View recycled, ViewGroup container)
{
  final ImageView myImageView;
  if (recycled == null) {
    myImageView = (ImageView) inflater.inflate(R.layout.my_image_view,
container, false);
  } else {
    myImageView = (ImageView) recycled;
  }

  String url = myUrls.get(position);

  Glide
    .with(myFragment)
    .load(url)
    .centerCrop()
    .placeholder(R.drawable.loading_spinner)
    .into(myImageView);

  return myImageView;
```

```
}
```

- Glide 라이브러리를 이용해 띄우는 방식으로 구현 시도

구현 실패 사유

- 파이어베이스 데이터 저장 및 검색 기능에 초점을 두어 구현을 시도하였기 때문에 이미지 업로드 기능은 구현할 시간 부족.
- 그동안 주로 사용했던 sql문과 firebase 사용법이 판이하게 달라 메서드를 이용해 데이터를 관리하는 방식에 적응하고 해당 메서드들을 파악하는 것에 시간을 많이 소요하였다.
- firebase에서는 복잡한 query문을 firebase database 형식으로 구현하는 것이 어려웠다.

구현 첨부 파일

- PBL2_ShoppingMall_DBList: 데이터 생성~읽은 후 정렬 기능까지 완료된 프로젝트
- ShoppingMall_Search: 위 프로젝트에서 검색기능 추가 구현 시도한 프로젝트