ViewModel

발표자: 장범준

ViewModel 그게 뭔데?

- 안드로이드 앱을 개발하다 보면 **코드가 복잡**해지기 마련인데, 이때 **특정한 패턴**이나 **소프트웨어 모델**을 적용해 개발하면 **깔끔하게 구조화**할 수 있습니다.(MVVM, ACC 등)
- MVVM디자인 패턴으로부터 파생된 개념이 **ViewModel**입니다.
- ViewModel은 view와 model사이에서 **업무를 중계해주고 전달**합니다.(두개를 불리하려는 목적)
- 생명주기에 따라 **리소스 제거가 요구되는 이벤트**가 발생할 때 이것을 **방지**해주기도 함

AAC?MVVM?

- AAC의 핵심 구조는 MVVM 패턴입니다.

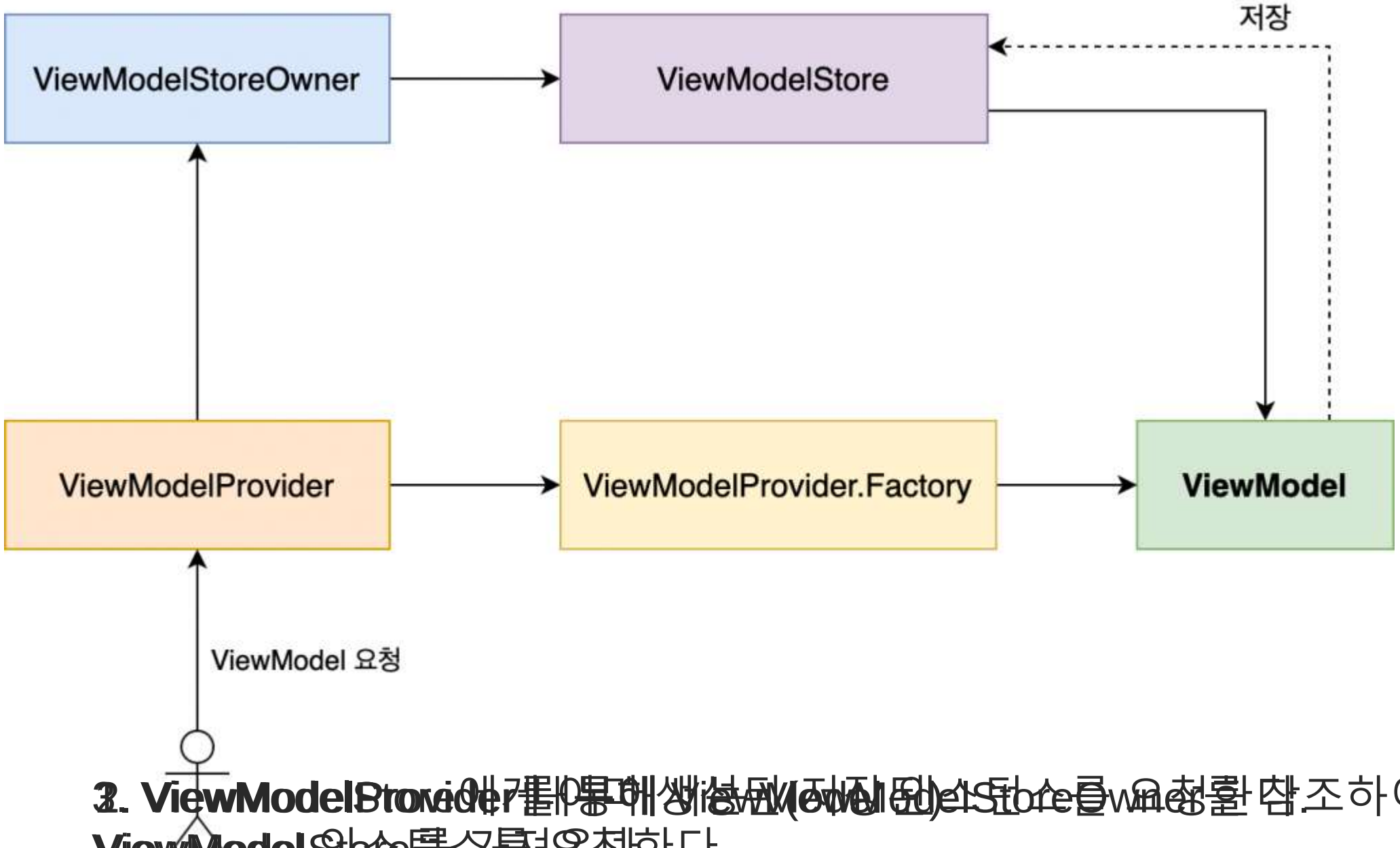
- 한마디로 AAC안에 MVVM 개념이 있는 것입니다.

View에 필요한 데이터를 관리하여 바인딩 해주고, 비즈니스 로직을 담당해 데이터를 처리하는 요소

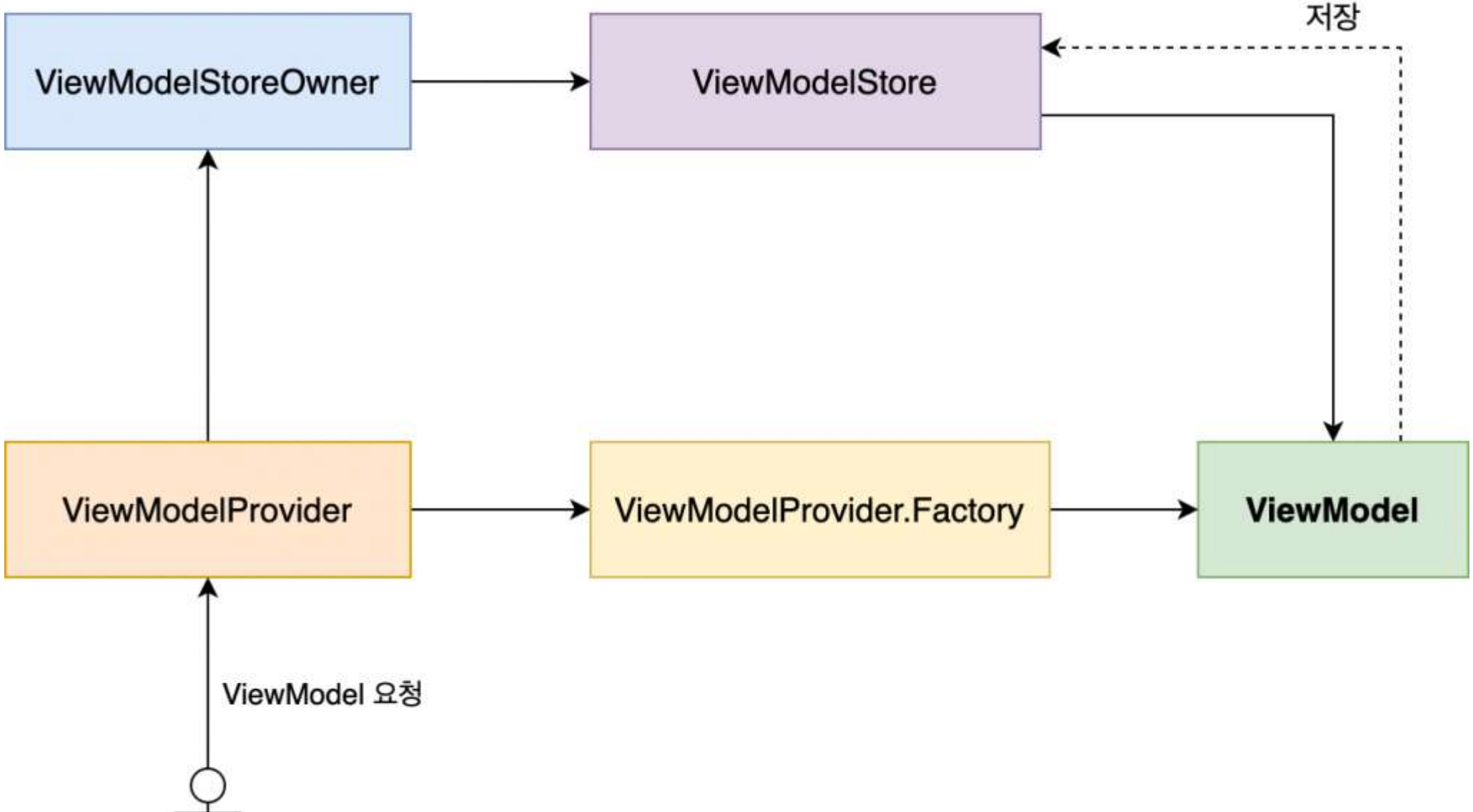
- MVVM 패턴의 목표는 비즈니스 로직과 프레젠테이션 분리하는 것입니다.
- 그렇게 되면 테스트, 유지 보수 측면에서 용이합니다.

Android의 수명 주기를 고려하여 UI 관련 데이터를 저장하고 관리하는 요소

- ViewModel의 생명 주기는 위의 그림 성되고 파괴되기 전까지 ViewModel 됩니다.
- 이 때문에 화면 회전과 같은 View가 정에서 데이터를 보존할 수 있습니다



3. **ViewModelProvider**에게 **ViewModel**을 요청한다.



4. 생성한 ViewModel 인스턴스를 ViewModelStore에 저장하고, 이후에 재사용할 수 있도록 관리한다.

5. 생성한 ViewModel 인스턴스를 ViewModelProvider에게 반환한다.

6. 생성한 ViewModelStore는 적절한 ViewModel 인스턴스를 가지고 있지 않으면, ViewModelProvider.Factory를 통해 새로운 ViewModel 인스턴스를 생성한다.

7. 생성한 ViewModelStore는 적절한 ViewModel 인스턴스를 가지고 있으면, 해당 인스턴스를 ViewModelProvider에게 반환한다.

앞에 것들로 알 수 있는것

- ViewModel은 ViewModelStore라는 객체에서 관리합니다. (내부적으로 HashMap<String, ViewModel>를 두어 관리)

- ViewModelStore 객체는 ViewModelStoreOwner가 관리합니다. (ViewModelStoreOwner는 interface이고 ComponentActivity와 Fragment클래스가 이를 구현)

- 즉, ViewModel을 생성하기 위해서는 Fragment나 Activity가 필요하다는 것을 알 수 있습니다.

```
class SimpleViewModel : ViewModel() {

    override fun onCleared() {
        super.onCleared()
        //여기서 ViewModel 종료 시 메모리 해제할 친구들을 해제해주자
    }

}
```

```
class MainActivity : AppCompatActivity() {

    private lateinit var viewModel:SimpleViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //ViewModel 인스턴스 생성
        viewModel = ViewModelProvider(this,
ViewModelProvider.NewInstanceFactory()).get(SimpleViewModel::class.java
)
    }

}
```

```java
public ViewModelProvider(@NonNull ViewModelStoreOwner owner, @NonNull Factory factory) {
    this(owner.getViewModelStore(), factory);
}
```

```java
simpleViewModel = ViewModelProvider(this, ViewModelProvider.NewInstanceFactory())
    .get(SimpleViewModel::class.java)    You, 2023-03-28 오후 9:25 • Uncommitted
```

```java
public class ViewModelStore {
    // HashMap 선언(Key는 String, Value는 ViewModel)
    private final HashMap<String, ViewModel> mMap = new HashMap<>();

    final void put(String key, ViewModel viewModel) {
        ViewModel oldViewModel = mMap.put(key, viewModel);
        if (oldViewModel != null) {
            oldViewModel.onCleared();
        }
    }

    final ViewModel get(String key) {
        return mMap.get(key);
    }

    Set<String> keys() {
        return new HashSet<>(mMap.keySet());
    }

    /**
     *  Clears internal storage and notifies ViewModels that they are no longer used
     */
    public final void clear() {
        for (ViewModel vm : mMap.values()) {
            vm.clear();
        }
        mMap.clear();
    }
}
```

감사합니다!