Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

Lập trình Android

Bài 34: *Mô hình Model – View – View Model*

Phòng LT & Mạng

http://csc.edu.vn/lap-trinh-va-csdl

2019

Nội dung

1. Mô hình Model – View – View Model (MVVM)
2. Áp dụng mô hình MVVM trong lập trình ứng dụng Android

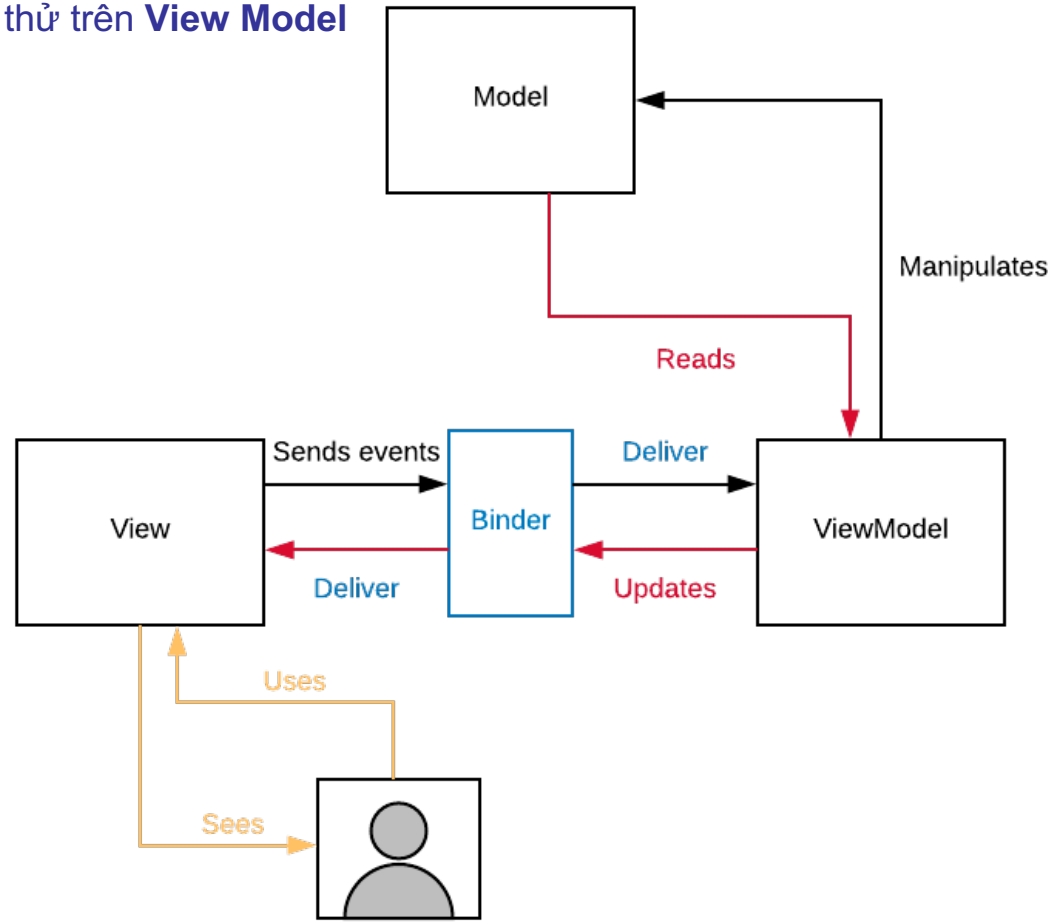
Mô hình Model – View – View Model (MVVM)

- So với mô hình **MVP**, mô hình **Model – View – View Model** (**MVVM**) giúp tách rời thành phần **View** và **View Model** bằng cách sử dụng một **Binder.**
- **View Model** không hề biết về **View**. **Binder** sẽ gửi các thay đổi từ **View Model** tới **View**. → Giúp kiểm thử trên **View** độc lập với việc kiểm thử trên **View Model**
- **3** thành phần chính của **MVVM** là:
  - **Model**
  - **View**
  - **View Model**
- Điểm khác với **MVC**, **MVP** là

thành phần thứ 4 **Binder** của **MVVM**

- **Binder** chịu trách nhiệm liên kết

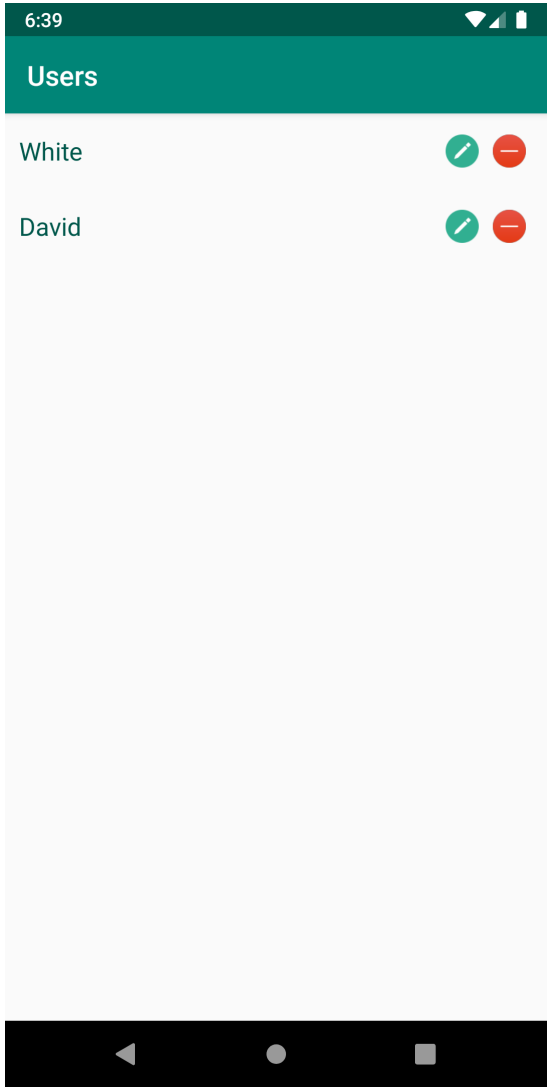
View và **View Model**



Minh hoạ mô hình MVVM

- Ứng dụng minh hoạ hiển thị một danh sách các người dùng (**User**)
  - Dữ liệu user được tải từ **Firebase Cloud Firestore**
  - Có thể thực hiện các thao tác edit, delete một User
- Ứng dụng sử dụng đến các class **ViewModel** và **LiveData** trong gói **lifecycle** của Android SDK. Để sử dụng thư viện này cần cấu hình dependencies trong file **build.gradle** của module:

```
implementation 'androidx.lifecycle:lifecycle-extensions:2.1.0'
```



Xây dựng Model

- **Model** trong trường hợp này là class **User**:

```
public class User {
    public String id;
    public String name;

    public User() {
        super();
    }

    public User(String id, String name) {
        super();
        this.id = id;
        this.name = name;
    }
}
```

- Tạo interface **UserRepository** với các phương thức thao tác với dữ liệu **User**:

```
public interface UserRepository {

    MutableLiveData<List<User>> getUsers();

    void deleteUser(User user);

    void updateUser(User user);

}
```

Xây dựng Model (2)

- Trong minh hoạ này, **Firebase Cloud Firestore** được dùng làm nơi chứa dữ liệu, ứng dụng có thể thực hiện các thao tác trên kho chứa dữ liệu này bằng cách tạo class **FirebaseUserRepository** kế thừa từ interface **UserRepository** và implements các phương thức hình mẫu từ interface đó:

```
public class FirebaseUserRepository implements UserRepository {
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    @Override
    public MutableLiveData<List<User>> getUsers() {
        final MutableLiveData<List<User>> users = new MutableLiveData<>();
        db.collection("users").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    List<User> lst = new ArrayList<>();
                    for (QueryDocumentSnapshot doc : task.getResult()) {
                        User u = doc.toObject(User.class);
                        lst.add(u);
                    }
                    users.setValue(lst); // cập nhật dữ liệu của MutableLiveData
                }
            }
        });
        return users;
    }

    @Override
    public void deleteUser(User user) {
    }

    @Override
    public void updateUser(User user) {
    }
}
```

Xây dựng View Model

- Tạo class **UserViewModel** tương ứng với Model **User**, kế thừa từ **ViewModel** của **Android SDK**:

```
public class UserViewModel extends ViewModel {
    private MutableLiveData<List<User>> users; // 1
    private UserRepository userRepository; // 2

    public LiveData<List<User>> getUsers() { // 3
        if (users == null)
            users = userRepository.getUsers();
        return users;
    }

    public void deleteUser(User user) { // 4
        userRepository.deleteUser(user);
        users.getValue().remove(user);
        users.setValue(users.getValue());
    }

    public void updateUser(int index, User user) { // 5

    }

    public void setUserRepository(UserRepository userRepository) { // 6
        this.userRepository = userRepository;
    }
}
```

Xây dựng View Model (2)

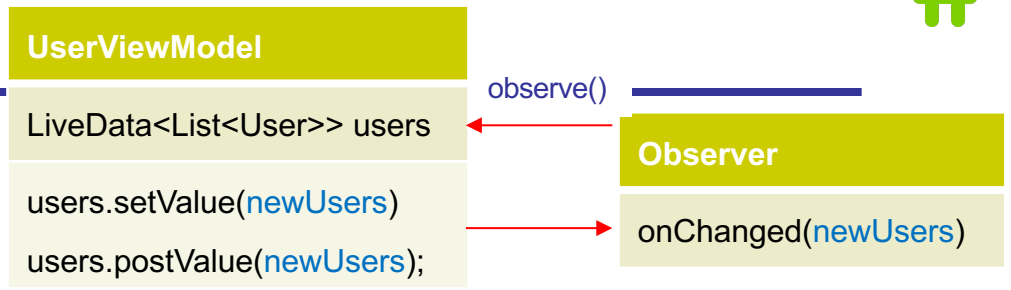
- // 1 – Khai báo danh sách các users (model) để **View Model** quản lý. Ở đây sử dụng đến kiểu dữ liệu **LiveData** (và **MutableLiveData**) để bao bọc danh sách users và cho phép có thể quan sát được sự thay đổi của danh sách đó. Việc quan sát này sẽ được thực hiện tại **View**.
- // 2 – Khai báo tham chiếu đến **UserRepository**, đối tượng này giúp **View Model** có thể tương tác với các kho chứa dữ liệu
- // 3 – Lấy danh sách user, nếu có sẵn có thể trả về ngay, ngược lại nhờ userRepository thực hiện lấy giúp
- // 4 - Xoá một user khỏi danh sách
- // 5 – Cập nhật thông tin một user
- // 6 – Phương thức nhằm thiết lập userRepository

Xây dựng View

- Giả sử **MainActivity** sử dụng **RecyclerView** để hiển thị danh sách users:

```
public class MainActivity extends AppCompatActivity implements UserAdapter.OnUserClickListener {
    RecyclerView rvVideo;
    UserAdapter adapter;
    UserViewModel viewModel; // 1
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        rvVideo = findViewById(R.id.rvVideo);
        adapter = new UserAdapter(new ArrayList<User>(), this);
        rvVideo.setAdapter(adapter);

        viewModel = ViewModelProviders.of(this).get(UserViewModel.class); // 2
        viewModel.setUserRepository(new FirebaseUserRepository()); // 3
        viewModel.getUsers().observe(this, new Observer<List<User>>() { // 4
            @Override
            public void onChanged(List<User> users) {
                adapter.setUsers(users); // 5
            }
        });
    }

    @Override
    public void onDeleteUserClick(User user) {
        viewModel.deleteUser(user); // 6
    }

    @Override
    public void onEditUserClick(int index, User user) {

    }
}
```

Xây dựng View (2)



- // 1 – Khai báo đối tượng **UserViewModel**
- // 2 – Khởi tạo đối tượng **UserViewModel**
- // 3 – Thiết lập **UserRepository** cho **UserViewModel** là đối tượng **FirebaseUserRepository**
- // 4 – **UserViewModel** thực hiện gọi phương thức getUsers() và quan sát (observe) trên danh sách users.
  - Khi thực hiện getUsers(), có thể thấy **FirebaseUserRepository** phải thực hiện một tác vụ bất đồng bộ để lấy danh sách users về ứng dụng. Khi tác vụ thành công, **MutableLiveData** thực hiện phương thức setValue() để cập nhật và thông báo sự thay đổi về dữ liệu do nó đang quản lý, khi đó callback onChanged() của đối tượng **Observer** sẽ được gọi, và thực hiện cập nhật lại **RecyclerView**
- // 5 – Khi có sự thay đổi về danh sách users, thực hiện cập nhật lại dữ liệu của **UserAdapter** và hiển thị lại danh sách đó trên **RecyclerView**
- // 6 – Khi nhấn button "Xoá" ⛔ một user (trên một dòng của **RecyclerView**), **UserViewModel** thực hiện phương thức xoá user đó khỏi danh sách mà nó đang quản lý, đồng thời cập nhật lại dữ liệu tại **Firebase Cloud Firestore** thông qua **UserRepository**
  - Khi thực hiện xoá một user, **MutableLiveData** cũng thực hiện setValue()/postValue() để cập nhật và thông báo sự thay đổi về dữ liệu nó đang quản lý → callback onChanged() của **Observer** tiếp tục được gọi → cập nhật **RecyclerView**

Q&A