MVP的实现

所谓的mvp，即是（model-处理业务逻辑（主要是数据读写，或者与后台通信（其实也是读写数据）），view-处理ui控件，presenter-主导器，操作model和view）

1. 创建一个bean

public class IUser {  
 private String ID;  
 private String userName;  
 private int age;  
public IUser(){}  
 public IUser(String ID, String userName, int age) {  
 this.ID = ID;  
 this.userName = userName;  
 this.age = age;  
 }  
 public String getID() {  
 return ID;  
 }  
 public void setID(String ID) {  
 this.ID = ID;  
 }  
 public String getUserName() {  
 return userName;  
 }  
 public void setUserName(String userName) {  
 this.userName = userName;  
 }  
 public int getAge() {  
 return age;  
 }  
 public void setAge(int age) {  
 this.age = age;  
 }  
 @Override  
 public String toString() {  
 return "IUser{" +  
 "ID='" + ID + '\'' +  
 ", userName='" + userName + '\'' +  
 ", age=" + age +  
 '}';  
 }  
}

1. 建立model（处理业务逻辑，这里指数据读写），先写接口，后写实现

public interface IUserModel {  
 public String getID() ;  
 public void setID(String ID);  
 public void setUserName(String userName);  
 public void setAge(int age);  
 public boolean saveUser(IUser user);  
 public IUser loadUser(String ID);  
}

1. 具体实现如下

public class UserModel implements IUserModel {  
 private String ID;  
 private String userName;  
 private int age;  
 private MyDB db;  
 public UserModel(Context context){  
 db = new MyDB(context);  
 }  
 @Override  
 public String getID() {  
 return this.ID!=null?this.ID:null;  
 }  
 @Override  
 public void setID(String ID) {  
 this.ID=ID;  
 }  
 @Override  
 public void setUserName(String userName) {  
 this.userName=userName;  
 }  
 @Override  
 public void setAge(int age) {  
 this.age=age;  
 }  
 @Override  
 public boolean saveUser(IUser user) {  
 return db.addUser(user);  
 }  
 @Override  
 public IUser loadUser(String ID) {  
 return db.findUserById(ID);  
 }  
}

1. 建立view（更新ui中的view状态），这里列出需要操作当前view的方法，也是接口

public interface IUserView {  
 public String getID();  
 public void setID(String ID);  
 public String getUserName();  
 public void setUserName(String userName);  
 public int getAge();  
 public void setAge(int age);  
}

5.建立presenter（主导器，通过iView和iModel接口操作model和view），activity可以把所有逻辑给presenter处理，这样java逻辑就从手机的activity中分离出来。

public class UserPresenter {  
 private IUserView iUserView;  
 private IUserModel iUserModel;  
 public UserPresenter(IUserView view){  
 this.iUserView=view;  
 iUserModel=new UserModel(((Context)view));  
 }  
 public boolean saveUser(String ID,String userName,int age){  
 return iUserModel.saveUser(new IUser(ID,userName,age));  
 }  
 public void loadUser( String id) {  
 IUser user = iUserModel.loadUser(id);  
 iUserView.setUserName(user.getUserName());  
 iUserView.setAge(user.getAge());  
 }  
}

1. activity的实现了IUserView接口，

public class MainActivity extends AppCompatActivity implements IUserView,View.OnClickListener{  
private EditText editId,editName,editAge;  
 private UserPresenter presenter;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 editId = (EditText)findViewById(R.id.userID);  
 editName=(EditText)findViewById(R.id.userName);  
 editAge = (EditText)findViewById(R.id.userAge);  
 ((Button)findViewById(R.id.save)).setOnClickListener(this);  
 ((Button)findViewById(R.id.load)).setOnClickListener(this);  
 presenter = new UserPresenter(this);  
 }  
 @Override  
 public String getID() {  
 return editId.getText().toString().trim();  
 }  
 @Override  
 public void setID(String ID) {  
 }  
 @Override  
 public String getUserName() {  
 return editName.getText().toString().trim();  
 }  
  
 @Override  
 public void setUserName(String userName) {  
 editName.setText(userName);  
 }  
  
 @Override  
 public int getAge() {  
 return Integer.parseInt(editAge.getText().toString().trim());  
 }  
  
 @Override  
 public void setAge(int age) {  
 editAge.setText(age+"");  
 }  
  
 @Override  
 public void onClick(View v) {  
 switch (v.getId()){  
 case R.id.save:  
 Boolean tag = presenter.saveUser(getID(),getUserName(),getAge());  
 if (tag){  
 Toast.makeText(this,"保存成功",Toast.LENGTH\_SHORT).show();  
 }else {  
 Toast.makeText(this,"保存失败",Toast.LENGTH\_SHORT).show();  
 }  
 break;  
 case R.id.load:  
 presenter.loadUser(getID());  
 break;  
 default:  
 break;  
 }  
 }  
}

* View 对应于Activity，负责View的绘制以及与用户交互
* Model 依然是业务逻辑和实体模型
* Presenter 负责完成View于Model间的交互