Android 实用技巧之: 用好泛型, 少写代码

阅读 2482 收藏 180 2017-01-10
原文链接：www.jianshu.com

Android实用技巧之:用好泛型,少写代码

本篇为项目T-MVP解读篇

1、基类使用泛型限定ViewDataBinding，子类直接指定泛型，一劳永逸：

```
基类：
public abstract class DataBindingActivity<B extends ViewDataBinding> extends AppCompatAct
      public B mViewBinding;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        View rootView = getLayoutInflater().inflate(this.getLayoutId(), null, false);
        mViewBinding = DataBindingUtil.bind(rootView);
}

子类：
public class AboutActivity extends DataBindingActivity<ActivityAboutBinding>
public abstract class BaseActivity<P extends BasePresenter, B extends ViewDataBinding> e
    public P mPresenter;

    @Override
    protected void initPresenter() {
        if (this instanceof BaseView &&
                this.getClass().getGenericSuperclass() instanceof ParameterizedType &&
                ((ParameterizedType) (this.getClass().getGenericSuperclass())).getActual
            Class mPresenterClass = (Class) ((ParameterizedType) (this.getClass()
                    .getGenericSuperclass())).getActualTypeArguments()[0];
            mPresenter = InstanceUtil.getInstance(mPresenterClass);
            mPresenter.setView(this);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mPresenter != null) mPresenter.onDetached();
    }
}
```

3、BaseViewHolder使用ViewDataBinding泛型限定，CoreAdapter使用BaseBean泛型限定，从告别Adapter，ViewHolder，一劳永逸：

```
    private List<M> mItemList = new ArrayList<>();

    @Override
    public BaseViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        return new BaseViewHolder(DataBindingUtil.inflate(LayoutInflater.from(parent.get(
    }

    @Override
    public void onBindViewHolder(BaseViewHolder holder, int position) {
        holder.mViewDataBinding.setVariable(BR.item, getItem(position));
        holder.mViewDataBinding.executePendingBindings();
    }
```

4、TRecyclerView使用BaseBean泛型限定，从告别OnRefresh，OnLoadMore，一劳永逸：

```
public class TRecyclerView<M extends BaseBean> extends FrameLayout implements AdapterPres
    private SwipeRefreshLayout swipeRefresh;
    private RecyclerView recyclerview;
    private CoreAdapter<M> mCommAdapter;
    private AdapterPresenter mCoreAdapterPresenter;

    public void init(Context context, AttributeSet attrs) {
        swipeRefresh.setOnRefreshListener(()->mCoreAdapterPresenter.fetch(););
        recyclerview.addOnScrollListener(new RecyclerView.OnScrollListener() {
            int lastVisibleItem;
            if (recyclerview.getAdapter() != null
                    && newState == RecyclerView.SCROLL_STATE_IDLE
                    && lastVisibleItem + 1 == recyclerview.getAdapter()
                    .getItemCount() && mCommAdapter.isHasMore)
                mCoreAdapterPresenter.fetch();
        }

        @Override
        public void onScrolled(RecyclerView recyclerView, int arg0, int arg1) {
            super.onScrolled(recyclerView, arg0, arg1);
            lastVisibleItem = mLayoutManager.findLastVisibleItemPosition();
        }
    });
}
public TRecyclerView<M> setData(List<M> data) {
    mCommAdapter.setBeans(data);
}
```

5、TypeSelector使用泛型类型，viewType对应layoutId，轻松实现复杂列表多viewType的选择器，一劳永逸：

```
public interface TypeSelector<M> {
    int getType(M m);
}

TypeSelector<MessageInfo> mTypeSelector = (item) -> TextUtils.equals(item.creater.ob
public void initView() {
    mViewBinding.lvMsg.setFootData(C.getAdminMsg()).setTypeSelector(mTypeSelector);
    mViewBinding.lvMsg.getPresenter()
        .setRepository(ApiFactory::getMessageList)
        .setParam(C.INCLUDE, C.CREATER)
        .setParam(C.UID, SpUtil.getUser().objectId)
        .fetch();
}
```

5、Repository使用泛型结果和HashMap包装多个参数，使用apt自动生成的ApiFactory返回不带泛型的Observable，从此列表类型的网络请求交给AdapterPresenter，一劳永逸：

```
Repository:

public interface Repository {
    Observable<DataArr> getData(HashMap<String, Object> param);

      public class DataArr<T> {
            public ArrayList<T> results;
      }
}

ApiFactory：

    /**
```

```
        return Api.getInstance().service.getCommentList(
                ApiUtil.getInclude(param),
                ApiUtil.getWhere(param),
                ApiUtil.getSkip(param),
                C.PAGE_COUNT)
                .compose(RxSchedulers.io_main());
    }
```

AdapterPresenter：

```
public class AdapterPresenter {
    private Repository mRepository;//仓库
    private HashMap<String, Object> param = new HashMap<>();//设置仓库钥匙
    private int begin = 0;
    private final IAdapterView view;

    public interface IAdapterView {
        void setEmpty();

        void setData(DataArr response, int begin);

        void reSetEmpty();
    }

    public AdapterPresenter(IAdapterView mIAdapterViewImpl) {
        this.view = mIAdapterViewImpl;
    }
    }

    public AdapterPresenter setParam(String key, String value) {
        this.param.put(key, value);
        return this;
    }

    public void setBegin(int begin) {
        this.begin = begin;
    }

    public void fetch() {
        begin++;
        view.reSetEmpty();
        if (mRepository == null) {
            Log.e("mRepository", "null");
            return;
        }
        param.put(C.PAGE, begin);
        mRepository
                .getData(param)
                .subscribe(
                        res -> view.setData(res, begin),
                        e -> view.setEmpty());
    }
}
```

使用：

```
    @Override
    public void initView() {
        mViewBinding.lvUser.getPresenter().setRepository(ApiFactory::getAllUser).fetch()
    }
}
```

用户列表的itemType也就是其layoutId，通过attr在xml中设置：

```
  <com.base.adapter.TRecyclerView
            android:id="@+id/lv_user"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:isRefreshable="false"
            app:itemType="@layout/list_item_user"
            app:layout_behavior="@string/appbar_scrolling_view_behavior" />
```

更多泛型的实际应用，请参考项目T-MVP

或者加群来搞基：

QQ群：AndroidMVP 555343041

更新日志：

2017/1 / 8： 使用Apt封装Retrofit生成ApiFactory替换掉所有的Repository，狂删代码

2016/12 / 30：使用Apt生成全局路由TRouter，更优雅的页面跳转，支持传递参数和共享view转场动画

2016/12 / 29：去掉BaseMultiVH新增VHClassSelector支持更完美的多ViewHolder

2016/12 / 28：使用Apt生成全局的ApiFactory替代所有的Model

2016/12 / 27：增加了BaseMultiVH扩展支持多类型的ViewHolder

2016/12 / 26：抽离CoreAdapterPresenter优化TRecyclerView

Android实用技巧之:用好泛型,少写代码

安卓AOP实战:APT打造极简路由

> 全局路由TRouter，更优雅的页面跳转

安卓AOP实战:Javassist强撸EventBus

> 加入OkBus，实现注解传递事件

安卓AOP三剑客:APT,AspectJ,Javassist

> 1、去掉所有反射>2、新增apt初始化工厂，替换掉了dagger2。>3、新增aop切片，处理缓存和日志

Android

掘金 首页

登录 · 注册

加入掘金

Android 交流微信群,

随时随地阅读干货, 交流见解。

相关热门文章

Fruit - 让你像解析Json一样解析Html
CoderGhui 15

中国国内可用**API**合集
LeviDing 718 13

Android内存分配**/**回收的一个问题**-**为什么内存使用很少...
看书的小蜗牛 10

使用自定义动画实现 **ImageView** 的神奇移动效果
Cyandev

移动大脑**-SpringMVc**搭建**RestFul**后台服务（四）**-**添加**T..**
ywl5320 22