多布局

对于多布局，提供了：

`BaseMultiItemQuickAdapter`、`BaseDelegateMultiAdapter`、`BaseProviderMultiAdapter`

三种基础类型。

1、BaseMultiItemQuickAdapter

说明：适用于类型较少，业务不复杂的场景，便于快速使用。

所有的数据类型，都必须实现`MultiItemEntity`接口（注意，这里不是继承抽象类，而是实现接口，避免对业务的实体类带来影响）

```
public class MultipleItemQuickAdapter extends BaseMultiItemQuickAdapter<QuickMultipleEntity, BaseViewHolder> {

    public MultipleItemQuickAdapter(List<QuickMultipleEntity> data) {
        super(data);
        // 绑定 layout 对应的 type
        addItemType(QuickMultipleEntity.TEXT, R.layout.item_text_view);
        addItemType(QuickMultipleEntity.IMG, R.layout.item_image_view);
        addItemType(QuickMultipleEntity.IMG_TEXT, R.layout.item_img_text_view);
    }

    @Override
    protected void convert(@NonNull BaseViewHolder helper, QuickMultipleEntity item) {
        // 根据返回的 type 分别设置数据
        switch (helper.getItemViewType()) {
            case QuickMultipleEntity.TEXT:
                helper.setText(R.id.tv, item.getContent());
                break;
            case QuickMultipleEntity.IMG_TEXT:
                ...
                ...
                break;
            default:
                break;
        }
    }
}
```

数据实体类写法：

```
class 你的数据实体类 implements MultiItemEntity {

  // 你的数据内容
  ...
  ...

  /**
   * 实现此方法，返回类型
   */
  @Override
    public int getItemType() {
        return itemType;
    }
}
```

2、BaseDelegateMultiAdapter

说明：通过代理类的方式，返回布局 id 和 item 类型；

适用于: 1、实体类不方便扩展，此Adapter的数据类型可以是任意类型，只需要在`BaseMultiTypeDelegate.getItemType`中返回对应类型 2、item 类型较少 如果类型较多，为了方便隔离各类型的业务逻辑，推荐使用`BaseProviderMultiAdapter`

方式一：

```
public class DelegateMultiAdapter extends BaseDelegateMultiAdapter<DelegateMultiEntity, BaseViewHolder> {

    public DelegateMultiAdapter() {
        super();

        // 第一步，设置代理
        setMultiTypeDelegate(new BaseMultiTypeDelegate<DelegateMultiEntity>() {
```

```
            @Override
            public int getItemType(@NotNull List<? extends DelegateMultiEntity> data, int position) {
                // 根据数据，自己判断应该返回的类型
                switch (position % 3) {
                    case 0:
                        return DelegateMultiEntity.TEXT;
                    case 1:
                        return DelegateMultiEntity.IMG;
                    case 2:
                        return DelegateMultiEntity.IMG_TEXT;
                    default:
                        break;
                }
                return 0;
            }
        });
        // 第二部，绑定 item 类型
        getMultiTypeDelegate()
                .addItemType(DelegateMultiEntity.TEXT, R.layout.item_text_view)
                .addItemType(DelegateMultiEntity.IMG, R.layout.item_image_view)
                .addItemType(DelegateMultiEntity.IMG_TEXT, R.layout.item_img_text_view);
    }

    @Override
    protected void convert(@NotNull BaseViewHolder helper, @NotNull DelegateMultiEntity item) {
        switch (helper.getItemViewType()) {
            case QuickMultipleEntity.TEXT:
                helper.setText(R.id.tv, "CymChad " + helper.getAdapterPosition());
                break;
            case QuickMultipleEntity.IMG_TEXT:
                switch (helper.getLayoutPosition() % 2) {
                    case 0:
                        helper.setImageResource(R.id.iv, R.mipmap.animation_img1);
                        break;
                    case 1:
```

```
                        helper.setImageResource(R.id.iv, R.mipmap.animation_img2);
                        break;
                    default:
                        break;
                }
                helper.setText(R.id.tv, "ChayChan " + helper.getAdapterPosition());
                break;
            default:
                break;
        }
    }
}
```

方式二：

```
public class DelegateMultiAdapter extends BaseDelegateMultiAdapter<DelegateMultiEntity, BaseViewHolder> {

    public DelegateMultiAdapter() {
        super();

        // 实现自己的代理类：
        setMultiTypeDelegate(new MyMultiTypeDelegate());
    }

    @Override
    protected void convert(@NotNull BaseViewHolder helper, @NotNull DelegateMultiEntity item) {
        switch (helper.getItemViewType()) {
            case QuickMultipleEntity.TEXT:
                helper.setText(R.id.tv, "CymChad " + helper.getAdapterPosition());
                break;
            case QuickMultipleEntity.IMG_TEXT:
                switch (helper.getLayoutPosition() % 2) {
                    case 0:
```

```
                    helper.setImageResource(R.id.iv, R.mipmap.animation_img1);
                    break;
                case 1:
                    helper.setImageResource(R.id.iv, R.mipmap.animation_img2);
                    break;
                default:
                    break;
            }
            helper.setText(R.id.tv, "ChayChan " + helper.getAdapterPosition());
            break;
        default:
            break;
    }
}

// 方式二：实现自己的代理类
final static class MyMultiTypeDelegate extends BaseMultiTypeDelegate<DelegateMultiEntity> {

    public MyMultiTypeDelegate() {
        // 绑定 item 类型
        addItemType(DelegateMultiEntity.TEXT, R.layout.item_text_view);
        addItemType(DelegateMultiEntity.IMG, R.layout.item_image_view);
        addItemType(DelegateMultiEntity.IMG_TEXT, R.layout.item_img_text_view);
    }

    @Override
    public int getItemType(@NotNull List<? extends DelegateMultiEntity> data, int position) {
        switch (position % 3) {
            case 0:
                return DelegateMultiEntity.TEXT;
            case 1:
                return DelegateMultiEntity.IMG;
            case 2:
                return DelegateMultiEntity.IMG_TEXT;
            default:
```

```
                    break;
                }
                return 0;
            }
        }
    }
```

3、BaseProviderMultiAdapter

说明：当有多种条目的时候，避免在`convert()`中做太多的业务逻辑，把逻辑放在对应的 `ItemProvider` 中。以及最大化自定义 `VH` 类型。

1、此`Adapter`的数据类型可以是任意类型，只需要在`getItemType`中返回对应类型 2、`Adapter`不限定`ViewHolder`类型。`ViewHolder` 由 `BaseItemProvider` 实现，并且每个`BaseItemProvider`可以拥有自己类型的`ViewHolder`类型。

第一步，`Adapter` 代码如下：

```
public class ProviderMultiAdapter extends BaseProviderMultiAdapter<ProviderMultiEntity> {

    public ProviderMultiAdapter() {
        super();
        // 注册 Provider
        addItemProvider(new ImgItemProvider());
        addItemProvider(new TextImgItemProvider());
        addItemProvider(new TextItemProvider());
    }

    /**
     * 自行根据数据、位置等信息，返回 item 类型
     */
    @Override
```

```
    protected int getItemType(@NotNull List<? extends ProviderMultiEntity> data, int position) {
        switch (position % 3) {
            case 0:
                return ProviderMultiEntity.IMG;
            case 1:
                return ProviderMultiEntity.TEXT;
            case 2:
                return ProviderMultiEntity.IMG_TEXT;
            default:
                break;
        }
        return 0;
    }
}
```

第二步，`Provider`代码如下：

```
public class ImgItemProvider extends BaseItemProvider<ProviderMultiEntity> {

    // item 类型
    @Override
    public int getItemViewType() {
        return ProviderMultiEntity.IMG;
    }

    // 返回 item 布局 layout
    @Override
    public int getLayoutId() {
        return R.layout.item_image_view;
    }

    /*
     * （可选）
```

```
     * 重写返回自己的 ViewHolder。
     * 默认返回 BaseViewHolder()
     */
    @NotNull
    @Override
    public BaseViewHolder onCreateViewHolder(@NotNull ViewGroup parent) {
        return super.onCreateViewHolder(parent);
    }

    @Override
    public void convert(@NotNull BaseViewHolder helper, @NotNull ProviderMultiEntity data) {
        // 设置 item 数据
        if (helper.getAdapterPosition() % 2 == 0) {
            helper.setImageResource(R.id.iv, R.mipmap.animation_img1);
        } else {
            helper.setImageResource(R.id.iv, R.mipmap.animation_img2);
        }
    }

    // 点击 item 事件
    @Override
    public void onClick(@NotNull BaseViewHolder helper, @NotNull View view, ProviderMultiEntity data, int positi
        Tips.show("Click: " + position);
    }

    @Override
    public boolean onLongClick(@NotNull BaseViewHolder helper, @NotNull View view, ProviderMultiEntity data, int
        Tips.show("Long Click: " + position);
        return true;
    }
}
```