转载请注明链接: https://blog.csdn.net/feather_wch/article/details/88744255

> Android paging library是一种分页库。
>
> 1. 按需加载数据进行展示，避免网络流量和系统资源的损耗

Paging分页库的基本使用

版本号:2019-03-24(1:30)

简介

1、分页加载的前世今生

> 1. 分页加载共有两种模式
> 2. 一种是传统的上拉加载更多的分页效果
> 3. 一种是无限滚动的分页效果

2、无限滚动的这种无感知的分页效果无疑是最好的

> Paging library就是这种分页库

1、Paging library 的核心组件是 `PagedList`

> 1. 能分页加载app需要的数据(先加载一部分)

> 2. 如果有任何加载的数据变化，一个新的 `Pagedlist`对象 会更新到 `LiveData`或者`RxJava2`依赖的对象 中

依赖添加

1、Paging的依赖添加(build.gradle)

```
/*=========================================
* Paging的依赖
*=============================================*/
implementation "android.arch.paging:runtime:1.0.1"
implementation "android.arch.paging:rxjava2:1.0.1"     // Paging对RxJava2的原生支持
```

数据库中加载数据

1、Activity中使用RecyclerView并且设置对数据的监听

```
public class DailyPlanActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // 1. RecyclerView + Adapter，正常借助DataBinding进行数据绑定
        final GoalListAdapter goalListAdapter = new GoalListAdapter(this);
        RecyclerView recyclerView = findViewById(R.id.goal_recyclerview);
        // 【LayoutManager!!!!!】
        recyclerView.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManager.VERTICA
        // adapter
        recyclerView.setAdapter(goalListAdapter);
        // 2. ViewModel存放LiveData<PagedList>，数据改变后调用PagedListAdapter的submitList
        GoalViewModel goalViewModel = ViewModelProviders.of(this, new GoalViewModel.GoalViewMod
                .get(GoalViewModel.class);
        goalViewModel.getGoalList().observe(this, new Observer<PagedList<Goal>>() {
            @Override
            public void onChanged(@Nullable PagedList<Goal> goals) {
                // submitList户必须不过数据刷新和比对
                goalListAdapter.submitList(goals);
            }
        });
    }
}
```

> Activity的布局

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res,
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DailyPlanActivity">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/goal_recyclerview"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

</android.support.constraint.ConstraintLayout>
```

2、RecyclerView的Adapter，需要继承自PagedListAdapter, 需要做四部分的工作

> 1. onCreateViewHolder()-创建ViewHolder
> 2. `public static class GoalViewHolder extends RecyclerView.ViewHolder`
> 3. onBindViewHolder()-绑定数据和UI
> 4. `private static DiffUtil.ItemCallback<Goal> DIFF_CALLBACK` 对新旧数据进行差异对比

```java
public class GoalListAdapter extends PagedListAdapter<Goal, GoalListAdapter.GoalViewHolder>{
    Context mContext;

    public GoalListAdapter(Context context) {
        super(DIFF_CALLBACK);
        mContext = context;
    }

    @NonNull
    @Override
    public GoalViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        LayoutInflater inflater = LayoutInflater.from(mContext);
        GoalItemBinding binding = DataBindingUtil.inflate(inflater, R.layout.recyclerview_goal_
        return new GoalViewHolder(binding.getRoot());
    }

    @Override
    public void onBindViewHolder(@NonNull GoalViewHolder holder, int position) {
        /**=====================================
         * 1、【重新绑定数据】
         *===============================================*/
        GoalItemBinding binding = DataBindingUtil.getBinding(holder.itemView);
        // 1. 绑定User
        Goal goal = getItem(position);
        binding.setGoal(goal);
        // 2. 立即执行绑定
        binding.executePendingBindings();
    }

    public static class GoalViewHolder extends RecyclerView.ViewHolder{
        public GoalViewHolder(View itemView) {
            super(itemView);
        }
    }

    /**=====================================
     * DiffUtil对比数据的新旧程度，合理更新数据
     *=====================================*/
    private static DiffUtil.ItemCallback<Goal> DIFF_CALLBACK =
            new DiffUtil.ItemCallback<Goal>() {
                @Override
                public boolean areItemsTheSame(Goal oldGoal, Goal newGoal) {
                    return oldGoal.getId() == newGoal.getId();
                }

                @Override
                public boolean areContentsTheSame(Goal oldGoal,
                                                  Goal newGoal) {
                    return oldGoal.equals(newGoal);
                }
            };
}
```

布局

```xml
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data class="com.hao.featherdailyplan.GoalItemBinding">
        <variable
            name="goal"
            type="com.hao.architecture.Goal"/>
    </data>
    <android.support.constraint.ConstraintLayout
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        tools:context="com.hao.architecture.DailyPlanActivity">

        <ImageView
            android:id="@+id/goal_icon_img"
            android:layout_width="80dp"
            android:layout_height="80dp"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            tools:src="@drawable/ic_launcher_background"
            android:layout_marginLeft="10dp"/>

        <ImageView
            android:id="@+id/goal_start_img"
            android:layout_width="80dp"
            android:layout_height="80dp"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            tools:src="@drawable/ic_launcher_background"
            android:layout_marginRight="10dp">

        </ImageView>

        <TextView
            android:id="@+id/goal_total_hours_txt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toTopOf="@+id/goal_today_hours_txt"
            app:layout_constraintEnd_toStartOf="@+id/goal_start_img"
            tools:text="72.7h"
            android:text='@{String.valueOf(goal.goaltime) + "h"}'
            app:layout_constraintVertical_chainStyle="packed"
            android:layout_marginEnd="10dp"/>

        <TextView
            android:id="@+id/goal_today_hours_txt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/goal_total_hours_txt"
            tools:text="3.7h"
            app:layout_constraintEnd_toStartOf="@+id/goal_today_hours_indivitual_txt"/>

        <TextView
            android:id="@+id/goal_today_hours_indivitual_txt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintTop_toTopOf="@+id/goal_today_hours_txt"
            app:layout_constraintBottom_toBottomOf="@+id/goal_today_hours_txt"
            app:layout_constraintEnd_toStartOf="@+id/goal_today_expected_hours_txt"
            tools:text="/"/>

        <TextView
            android:id="@+id/goal_today_expected_hours_txt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintTop_toTopOf="@+id/goal_today_hours_txt"
            app:layout_constraintBottom_toBottomOf="@+id/goal_today_hours_txt"
            tools:text="7.7h"
            app:layout_constraintEnd_toEndOf="@+id/goal_total_hours_txt"/>

        <TextView
            android:id="@+id/goal_timer_txt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            tools:text="00:10:49"
            app:layout_constraintStart_toEndOf="@id/goal_icon_img"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toTopOf="@+id/goal_title_txt"
            android:layout_marginStart="10dp"/>

        <TextView
            android:id="@+id/goal_title_txt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            tools:text="决战2019"
            app:layout_constraintStart_toStartOf="@+id/goal_timer_txt"
            app:layout_constraintTop_toBottomOf="@+id/goal_timer_txt"
            app:layout_constraintBottom_toBottomOf="parent"
            android:text="@{goal.title}"/>
    </android.support.constraint.ConstraintLayout>

</layout>
```

3、GoalViewModel: 内部构造 `LiveData<PagedList<Goal>> mGoalList`

```java
public class GoalViewModel extends ViewModel{
    /**===================================
     * 1、数据相关，构造LiveData<PagedList<Goal>>
     *====================================*/
    private GoalDao mGoalDao;
    private LiveData<PagedList<Goal>> mGoalList;
    private static final int PAGE_SIZE = 10;

    public LiveData<PagedList<Goal>> getGoalList() {
        if(mGoalList == null){
            mGoalList = new LivePagedListBuilder(mGoalDao.getGoalListFactory(), new PagedList.(
                    .setPageSize(PAGE_SIZE)                             //配置分页加载的数量
                    .setInitialLoadSizeHint(PAGE_SIZE)                  //初始化加载的数量
                    .setPrefetchDistance(PAGE_SIZE)
                    .setEnablePlaceholders(true)
                    .build()).build();
        }
        return mGoalList;
    }

    /**=============================
     * 2、构造“目标”的ViewModel需要的内容
     *=============================*/
    @SuppressLint("StaticFieldLeak")
    private Context mContext;
    public GoalViewModel(Context context){
        this.mContext = context;
        this.mGoalDao = GoalDatabase.getInstance(mContext).getGoalDao();
    }
    public static class GoalViewModelFactory extends ViewModelProvider.NewInstanceFactory{
        private Context mContext;
        public GoalViewModelFactory(Context context){
            mContext = context;
        }
        @NonNull
        @Override
        public <T extends ViewModel> T create(@NonNull Class<T> modelClass) {
            return (T) new GoalViewModel(mContext);
        }
    }

}
```

4、Room数据库相关:Goal、GoalDao、GoalDatabase

> GoalDao： 提供数据的查询---核心在于提供 `DataSource.Factory`

```java
@Dao
public interface GoalDao {

    @Query("select * from goal")
    Flowable<Goal> queryGoalList();

    @Query("select * from goal")
    DataSource.Factory<Integer, Goal> getGoalListFactory();

    @Insert
    void insertGoal(Goal... goals);

    @Delete
    void deleteGoal(Goal goal);

    @Update
    void updateGoal(Goal goal);
}
```

> Goal：数据实体

```
@Entity
public class Goal {
    @PrimaryKey
    private int id;
    private int goaltype; // 0: 主目标 1: 子目标
    private int analyticsGoalType; // 分析学类型，按照各个领域的目标进行分类 学习、生活、兴趣、未来
    private String title; // 标题
    private String description; // 描述
    private long goaltime; // 目标多少小时
    private String goalImgUrl; // 目标图片的url
    private int goalImgColor; // 图片的色值

    @Embedded(prefix = "begin")
    private Date beginDate;
    @Embedded(prefix = "bend")
    private Date endDate;

    public Goal(int id, int goaltype, int analyticsGoalType, String title, String description,
        this.id = id;
        this.goaltype = goaltype;
        this.analyticsGoalType = analyticsGoalType;
        this.title = title;
        this.description = description;
        this.goaltime = goaltime;
        this.goalImgUrl = goalImgUrl;
        this.goalImgColor = goalImgColor;
        this.beginDate = beginDate;
        this.endDate = endDate;
    }

    // xxx
}
```

> GoalDatabase：数据库

```
@Database(entities = {Goal.class}, version = 1, exportSchema = false)
// 表的变更，需要提高version
public abstract class GoalDatabase extends RoomDatabase
{
    private static GoalDatabase INSTANCE;
    private static final Object sLock = new Object();

    public abstract GoalDao getGoalDao();

    public static GoalDatabase getInstance(Context context) {
        synchronized (sLock) {
            if (INSTANCE == null) {
                INSTANCE = Room.databaseBuilder(context.getApplicationContext(), GoalDatabase.c
                        .build();
            }
            return INSTANCE;
        }
    }
}
```

RxJava形式

1、将LiveData替换为Observable或者Flowable

> RxPagedListBuilder调用buildObservable进行构造

```
public class GoalViewModel extends ViewModel{
    private Observable<PagedList<Goal>> mGoalList;

    public Observable<PagedList<Goal>> getGoalList() {
        if(mGoalList == null){
          // RxPagedListBuilder是重点
            mGoalList = new RxPagedListBuilder(xxx).buildObservable();
        }
        return mGoalList;
    }
}
```

> 使用

```
goalViewModel.getGoalList()
             .subscribe(flowableList -> goalListAdapter.submitList(flowableList));
```

placeholders

paging configuration

invalidate

data source type

自定义

Consider how content updates work

Provide data mapping

参考资料

1. Android官方架构组件Paging：分页库的设计美学
2. Paging官方文档
3. Android Jetpack架构组件之 Paging（使用、源码篇）
4.