RecyclerView面试宝典：7大高频问题解析

鸿洋 2024-05-21 08:35 北京

以下文章来源于Android补给站 ，作者Rouse

Android补给站

Android&小程序&前端程序员，目标大前端，终身学习者。

在Android开发领域，RecyclerView是展示动态数据列表的强大工具，凭借其灵活性和高性能，成为了面试中的热门话题。本文旨在深入探讨与RecyclerView相关的高频面试问题，并提供详尽的解答技巧，帮助求职者在面试中脱颖而出。

1 功能理解

问题：RecyclerView与ListView有什么不同？

出发点：考察面试者对设计、功能和性能上的差异了解。

参考简答：

主要区别如下：

1. **布局管理器**：RecyclerView引入了LayoutManager的概念，支持更复杂的布局，如线性布局、网格布局和瀑布流布局等，而ListView默认只支持垂直线性布局。
2. **缓存机制**：ListView有两级缓存，但RecyclerView有四级缓存，缓存效率更高。同时ListView缓存的是View，而RecyclerView缓存的是ViewHolder。
3. **Item装饰和动画**：RecyclerView通过ItemDecoration和ItemAnimator提供了装饰和动画的支持，使得添加分隔线、实现列表动画变得更加简单。
4. **灵活的数据更新**：RecyclerView提供了局部更新方法，如`notifyItemInserted()`、`notifyItemRemoved()`和`notifyItemChanged()`等。
5. **性能优化**：RecyclerView在设计时就考虑到了更高效的性能，尤其是在处理大量数据或需要动态加载不同类型视图时。ListView在这些方面表现较为逊色。

2 工作原理

问题：了解RecyclerView的缓存吗？请详细描述一下它的机制。

出发点：考察面试者对四级缓存的作用以及它们之间的工作流程的理解。

参考简答：

RecyclerView通过一系列精细的缓存机制优化性能，包括：

1. AttachedScrap

- **作用**：存储暂时从RecyclerView中分离，但很快会重新绑定和重新使用的ViewHolders。这些ViewHolders没有被完全回收，仍然保持与RecyclerView的连接。
- **特点**：它们主要用于动画处理，如移动动画或者删除动画，因为RecyclerView可以直接访问这些ViewHolders，而无需通过Adapter重新创建。

2. CachedViews

- **作用**：存储已经离开屏幕但是仍然保留在内存中，可以被快速复用的ViewHolders。与AttachedScrap不同，这些ViewHolders已经从RecyclerView中彻底分离，但是它们的数量有限制，默认是2个。
- **特点**：CachedViews可以快速复用，减少布局的重新绘制和测量，提高滑动性能。

3. ViewCacheExtension

- **作用**：是一个可选的缓存层，允许开发者自定义缓存策略，存储更多的或者特定类型的ViewHolders。
- **特点**：通过实现ViewCacheExtension，开发者可以控制哪些ViewHolders应该被缓存，以及如何被复用，提供了更大的灵活性和控制力。

4. RecycledViewPool

- **作用**：存储大量的被回收的ViewHolders，供同一个RecyclerView或者不同的RecyclerView复用。
- **特点**：RecycledViewPool可以跨多个RecyclerView共享，特别适合于有多个相似列表页面的应用，能够显著减少内存占用和提升性能。

工作流程:

1. 当Item滑出屏幕时，它的ViewHolder首先尝试加入AttachedScrap，如果不适用，则加入CachedViews。
2. 如果CachedViews已满，ViewHolder则会被放入RecycledViewPool。
3. ViewCacheExtension作为一个扩展层，可以由开发者根据具体需求来实现和使用。
4. 当需要新的ViewHolder时，RecyclerView会按照以下顺序尝试复用：AttachedScrap → CachedViews → ViewCacheExtension → RecycledViewPool。

问题：请解释一下RecyclerView的局部刷新机制。

出发点：考察面试者对局部刷新的核心实现原理的理解。

参考简答：

涉及核心组件:

1. **Adapter**：负责提供ViewHolders和绑定数据到这些视图上。如`notifyItemChanged(int position)`方法。
2. **ViewHolder**：代表列表中的每个项的视图容器。通过ViewHolder，RecyclerView可以有效地重用视图，减少视图创建的开销。
3. **ItemAnimator**：负责处理项变更时的动画。当局部更新发生时，RecyclerView会利用ItemAnimator来添加、移除或更新项的动画效果，提升用户体验。
4. **LayoutManager**：负责Item的布局和回收策略。当数据发生变更时，LayoutManager决定哪些视图需要被重新布局，哪些可以保持不变。

局部刷新的实现流程:

1. **变更通知处理**：RecyclerView接收到Adapter的变更通知后，标记相应的视图位置需要更新。
2. **视图重用**：对于被标记需要更新的项，RecyclerView检查对应的ViewHolder是否可以重用。如果可以，RecyclerView会重新绑定新数据到这个ViewHolder上，而不是创建新的ViewHolder。
3. **视图更新**：ViewHolder绑定了新数据后，RecyclerView利用ItemAnimator来处理这些变更的动画效果，如淡入淡出或滑动效果，最终呈现给用户。
4. **清理和完成**：最后，RecyclerView完成更新流程，清理所有临时标记和缓存。

3 实战使用

问题：在RecyclerView中，如何只刷新列表项中的某个控件而不是整个item?

出发点：考察面试者是否理解RecyclerView的细粒度更新机制。

参考简答：

实现更细粒度的更新，可以通过调用Adapter的`notifyItemChanged(int position, Object payload)`方法实现，其中payload参数用于指定具体需要更新的控件或数据。在Adapter的`onBindViewHolder`方法中，通过检查payloads参数来区分是进行整个项的全量更新还是仅更新特定控件。

问题：如何处理RecyclerView中的并发修改异常（`ConcurrentModificationException`）？

出发点：考察面试者对并发数据操作中常见问题的理解及其解决方案，特别是在动态数据集合操作时如何保持数据一致性和应用稳定性。

参考简答：

`ConcurrentModificationException`通常发生在尝试迭代一个集合的同时，另一个线程或迭代过程中的方法修改了这个集合。以下是处理这种异常的几种策略：

1. **使用同步集合**：考虑使用线程安全的集合，如`Collections.synchronizedList()`包装器或CopyOnWriteArrayList。这些集合实现了同步访问控制，可以减少并发修改的风险。CopyOnWriteArrayList在迭代期间通过创建集合的副本来避免并发修改，非常适合读多写少的场景。

2. **避免在迭代期间修改集合**：如果可能，避免在遍历集合的循环中直接修改集合。如果需要修改，可以先标记需要添加或删除的项，在迭代完成后统一处理。

3. **使用迭代器的remove()方法**：如果需要在迭代过程中删除元素，使用Iterator的`remove()`方法而不是直接调用集合的删除方法。这样可以安全地在遍历时修改集合。

4. **主线程中更新数据**：确保所有对RecyclerView数据集的修改都在主线程中进行。这样可以避免多个线程同时修改数据集。

5. **使用锁或同步块**：在修改数据集之前手动同步代码块。这需要在代码中显式管理锁，可以使用synchronized关键字或显式的锁机制（如ReentrantLock），但必须小心管理以避免死锁。

6. **正确使用局部更新方法**：在数据集更改后，确保调用适当的`notifyItemChanged()`等方法来通知Adapter数据已更改。这有助于RecyclerView正确处理数据更新，避免在使用不一致的数据时引发异常。

问题：`Adapter的setHasStableIds`方法有用过吗？解释一下它的作用。

出发点：考察面试者对该方法的理解，是否有做个相关的优化。

参考简答：

`setHasStableIds(boolean hasStableIds)`方法用于告知RecyclerView每个列表项的ID是否固定不变。当Adapter的这个设置被激活时（即传入true），意味着您保证`getItemId(int position)`方法返回的每个ID在列表中是唯一的并且不会改变。

这个方法的作用主要体现在两个方面：

1. **性能优化**：启用稳定ID可以显著提高RecyclerView的性能。当`setHasStableIds(true)`被调用时，RecyclerView可以使用这些稳定的ID来避免重复的布局计算和视图重绘，因为它知道即使数据发生变化，每个列表项的ID仍然保持不变。这允许RecyclerView在处理数据集更改时做出更智能的决策，如局部刷新而非全量刷新。

2. **改善动画效果**：在数据集发生变化时（如添加、移除、移动等），如果开启了稳定ID，RecyclerView可以更准确地识别和定位变化的项，从而产生更平滑的动画效果。RecyclerView

能够利用稳定ID追踪哪些项是新的、哪些项被移除，以及哪些项的位置发生了变化，从而为这些变化提供更流畅的视觉反馈。

为了正确使用稳定ID，需要重写Adapter的`getItemId(int position)`方法，返回每个项的唯一ID。

4 性能优化

问题：做过RecyclerView性能优化吗？说下你是如何做的？

出发点：考察面试者在实践中应用RecyclerView性能优化的经验。

参考简答：

1. **局部更新数据**：通过`notifyItemChanged(int position)`等方法进行局部数据更新，而不是使用`notifyDataSetChanged()`刷新整个列表。这减少了RecyclerView的重新布局次数，优化了性能。

2. **利用DiffUtil计算数据差异**：使用DiffUtil类来计算新旧数据集的最小差异，并根据这些差异来更新RecyclerView。这样可以减少不必要的视图更新，仅对变化的部分进行重绘，进一步提升了更新效率。

3. **优化列表滑动**：通过自定义ItemDecoration、ItemAnimator以及合理使用LayoutManager的特性来优化列表的滑动和动画效果，减少卡顿现象。

4. **图片加载优化**：对列表中加载的图片进行大小调整和缓存处理，来减少内存占用和避免内存泄漏。同时对滑动中列表停止加载图片，进步提升滑动性能。

5. **预加载数据**：当用户滑动接近列表底部时，提前加载更多的数据，以避免到达列表末尾时出现明显的加载等待时间。

6. **减少过度绘制**：通过分析布局的过度绘制情况，优化Item的布局，减少不必要的背景和透明度使用，降低渲染压力。

7. **减少测量**：对于固定高度的item，启用`setHasFixedSize(true)`，避免requestLayout导致的资源浪费。

8. **内存优化**：针对Adapter一样的两个列表，共享一个RecyclerViewPool以提高性能。

5 总结

本文通过对RecyclerView相关面试题的分析，从面试的角度，带大家加深对RecyclerView的理解，同时也希望能够帮助大家在面试中脱颖而出。

最后推荐一下我做的网站，玩Android: *wanandroid.com* ，包含详尽的知识体系、好用的工具，还有本公众号文章合集，欢迎体验和收藏！

推荐阅读：

2024 Google I/O Android 相关内容

Apk安装之谜

Android 描边动画实现母亲节祝福效果

扫一扫 关注我的公众号

如果你想要跟大家分享你的文章，欢迎投稿~

┏(^ 0 ^)┛明天见！

喜欢此内容的人还喜欢

Flutter鸿蒙终端一体化-天下一统

鸿洋

微信如果放弃鸿蒙适配，你会选择谁？

鸿洋

请大家拿下软考，现在！立刻！马上！！

鸿洋