



常用技术及工具介绍 (1)

2025/09/18



常用技术

LLM工具辅助软件开发实践



Peking
University

口为什么使用 LLM 工具？

- 更快的开发速度、更高的开发质量
- GitHub Copilot 报告：提升 55% 的开发速度
- 贯穿软件工程的全周期：需求分析、设计、编码、调试、测试、维护
- LLM 不仅帮你“写代码”，而是帮你“更快更好地思考”

口如何对待 LLM 工具？

- LLM 可能出错、生成无法运行的代码、引入安全漏洞
- 理解，检查并验证 LLM 的生成内容
- “你不是被AI取代，而是被会用AI的同事取代”





□ 大模型工具可以辅助学生完成以下几个方面的软件开发实践活动，包括需求获取与分析、软件设计、代码编写、软件测试、运行维护等。

□ 需求获取与分析

- 目标: 确定软件项目，构思和明确软件需求，定义软件功能和非功能需求。
- 项目选题和需求构思
 - 辅助创意生成：可以利用大模型，针对特定应用领域或技术栈，生成软件项目的需求创意。
 - 例如，输入“假设你是一个富有创意的软件工程师，请你帮助我生成一个基于AI的高校校园应用创意”，deepseek给出如下结果

应用名称：EduPal - 你的AI校园共生伙伴

核心理念：EduPal 不再是一个功能堆砌的“超级App”，而是一个以AI为核心驱动的、具有预测性、主动性和个性化的智能体。它旨在与学生的学术和校园生活“共生”，提前预见需求，无缝解决问题，并促进学生全面成长。



Deepseek从四个方面对这个创意进行了详细介绍：

一、核心AI驱动功能亮点

➤ 1. 学术智能体 (Academic Agent)

- **个性化学习路径规划**: AI通过分析学生的专业培养方案、当前成绩、选课历史和学习行为（如图书馆借阅、视频观看停留点），动态生成并调整专属学习路径。例如：“根据你的线性代数期中成绩，建议你本周优先复习第二章，并完成我为你筛选的这三道典型习题。”
- **AI课程表+**: 不仅是静态课表。它能与校园系统联动，自动提示：“下节课的教室从A栋换到了B栋，已为你规划最优路径，预计步行8分钟，现在出发刚好。”
- **智能笔记与知识图谱生成**: 学生上传课堂录音或拍摄板书照片，AI自动进行语音转文字、提取关键概念、生成结构化笔记，并进一步构建本课程的**可视化知识图谱**，帮助学生看清知识点之间的联系。
- **“救命”模式 (考前冲刺)** : 输入考试日期和范围，AI自动生成一个冲刺时间表，整合了课程重点、往届试题和相关的学习资料（如MO视频链接），并提供自测题。



Peking
University



➤ 2. 生活智能体 (Life Agent)

- **上下文感知的校园导航**: 不仅告诉你怎么走，还能结合实时数据：“去第三食堂？根据当前人流数据，建议你去第二食堂，排队时间少5分钟。”
- **社交联结者**: AI基于兴趣社团、选课重叠度、图书馆借阅相似度等，**匿名且隐私保护地**推荐“可能志同道合”的同学。例如：“发现一位同学和你在同一时间都在图书馆研究‘元宇宙与建筑设计’，并有3个共同社团兴趣，是否愿意匿名打个招呼？”
- **心理健康伙伴**: 内置一个基于大型语言模型的、经过安全训练的聊天机器人。它可以进行日常情绪疏导，识别压力过大的语言模式，并主动推荐校园心理资源（预约咨询、正念冥想课程等）。

➤ 3. 行政与效率智能体 (Admin Agent)

- **智能文档助手**: 帮助学生自动填写各种重复性的行政表格（如奖学金申请、活动报备），只需核对信息即可。
- **一站式问答中心**: 用自然语言询问任何校园相关问题：“奖学金什么时候申请？”“体育馆今晚开门到几点？”“我的校园网为什么断了？”AI直接给出精准答案或操作步骤，无需在不同公众号或网站上寻找。
- **消费与补贴洞察**: 连接校园卡消费数据，AI分析你的消费习惯，并主动提示：“检测到你本月餐饮消费较低，符合‘阳光助餐’补贴的申请条件，是否立即办理？”



二、技术栈构想（工程师视角）

➤ 后端：

- **框架：** Python (FastAPI / Django) 或 Go (Gin)
- **AI核心：** LangChain/LlamaIndex 用于构建AI智能体框架，集成多种大语言模型（如GPT-4, LLaMA 3）和专用小模型（语音转文字、OCR）
- **数据存储：** PostgreSQL (主数据库), Redis (缓存), Pinecone/Chroma (向量数据库，用于存储知识图谱和语义搜索)
- **API集成：** 通过校园开放平台API对接教务、一卡通、图书馆等系统

➤ 前端：

- **移动端：** React Native (跨平台) 或 Swift/Kotlin (原生)
- **设计理念：** 极简主义，对话式UI，大量运用卡片和自然语言交互，减少层级菜单。

➤ 基础设施：

- **云服务：** AWS/Azure/GCP
- **容器化：** Docker + Kubernetes
- **MLOps：** 用于持续训练和部署AI模型（如消费模式识别模型）





三、商业模式与可持续性

- **To C (学生)**: 基础功能完全免费。高级功能（如更详细的生涯规划报告、个性化深度分析）可采用订阅制（EduPal Pro）。
- **To B (学校)**: 向高校销售SaaS平台和部署服务，帮助学校数字化升级，费用基于学生人数和功能模块。这是主要的盈利模式。
- **生态合作**: 与校园内外的书店、打印店、咖啡厅合作，提供基于位置的智能优惠券推送（“你常去的图书馆旁边的咖啡店，今天拿铁半价”），从中获得佣金。

四、为什么这个创意能成功？

- **真痛点，非痒点**: 解决了学生“信息孤岛”、“选择困难”和“效率低下”的核心痛点。
- **AI不是点缀，是核心**: AI驱动从“人找信息”变为“信息找人”，创造了前所未有的流畅体验。
- **平台化潜力**: 可以成为高校数字生态的统一入口，价值巨大。
- **高度可扩展**: 初始版本可以聚焦核心功能（课表、导航、问答），后续逐步迭代加入更复杂的智能体。

希望这个名为 EduPal 的创意能激发您的灵感。我愿意继续深入讨论任何一个功能模块的技术细节或实现方案。



➤ 项目选题和需求构思

- **市场调研辅助**: 模型可以帮助分析某个领域的市场需求或技术趋势，提供相关背景信息，帮助团队确定项目的可行性和创新性。(纯生成式大模型工具不合适直接用于市场调研，建议结合有智能搜索功能的大模型工具，例如纳米 AI 搜索、腾讯元宝、豆包等)

➤ 细化和分析软件需求，生成软件需求文档

- **用户故事生成**: 根据用户角色和场景，生成符合实际应用场景的用户故事。
 - 例如：输入“作为顾客，想要搜索商品，以便找到想要购买的商品，请你根据顾客的需求生成用户故事”，大模型工具可以生成详细的用户故事，包括前置条件、后置条件、正常流程、异常流程等。
- **需求澄清与角色扮演**: 让大模型扮演不同的客户和用户角色，通过模拟对话的方式，帮助学生澄清模糊的需求，完善需求细节。





- 例如：
 - 场景 1：同学提出“网站要美观”，大模型可以扮演设计师，询问具体的设计风格、配色方案、字体选择等，帮助学生明确“美观”的具体含义。
 - 场景 2：同学提出“商品搜索功能”，大模型可以扮演电商平台商家，询问搜索算法的精准度、排序规则、筛选条件等，帮助学生优化搜索功能的设计。
- 需求文档生成：利用大模型工具，自动生成初步的需求文档模板。
 - 例如：输入“假设你是一个资深的软件需求分析师，我正在开发一个在线购物网站，请你帮我生成包含功能列表、用户角色以及用况图等内容的需求文档框架，包括功能需求、非功能需求(如性能、安全性等)。”



Peking
University



如何应用LLMs工具辅助软件开发

➤ 在与大模型或其他专业人员交互时，如何组织问题、明确需求以及有效沟通是至关重要的。

➤ 1. 明确问题

- 先明确你要解决的核心问题是什么。避免模糊或过于宽泛的问题。
- 反例：如何开发好一个项目？
- 正例：在开发一个校园管理系统时，如何设计一个安全且高效的用户认证模块？

➤ 2. 交代背景和领域

- 角色预设：假设大模型是一个某某领域非常有经验的专家。示例：“假设你 是一位资深的后端开发专家，专注于高并发系统的设计与优化。”



Peking
University



- **项目背景:** 提供项目的背景信息, 包括目标、用户群体、技术栈等。
 - **示例:** “我们正在开发一个在线考试系统, 目标用户是大学生和教师, 技术栈是 React 前端和 Spring Boot 后端。”
- **领域知识:** 如果问题涉及特定领域(如机器学习、区块链等), 简要说明相关背景知识。
 - **示例:** “我们正在开发一个基于机器学习的推荐系统, 使用的是 Python 和 TensorFlow。”

➤ 3. 问题的组织与分解

- **问题分解:** 将复杂问题分解为多个子问题, step-by-step 逐步解决。
 - **示例:** 对于“如何设计一个在线考试系统?”可以分解为:“如何设计用户认证模块?如何实现考试创建和管理功能?如何确保系统的安全性和稳定性?”
- **优先级排序:** 根据问题的紧急程度或重要性排序, 优先解决关键问题。
 - **示例:** 在开发初期, 优先解决用户认证和考试管理功能, 后期再优化性能和安全性。





➤ 4. 向大模型提问(Prompt设计)

- **结构化提问：**使用清晰、结构化的语言提问，确保问题易于理解。虽然它是机器，但是可以将它想象成一个专业人员，考虑它是否能够清晰的理解。
 - **示例：**“在开发一个在线考试系统时，如何设计一个支持高并发的用户认证模块？我们目前使用的是 Spring Security，但担心性能问题。”
- **提供上下文：**在提问时提供足够的上下文信息，避免对方需要猜测你的需求。如果可能，提供具体的示例或代码片段，帮助对方更好地理解问题。
 - **示例：**“我们正在开发一个网购平台，使用微服务架构，目前遇到的问题是订单服务的响应时间较慢。以下是我们当前的代码，看起来是循环嵌套太多，如何优化？”

➤ 5. 汇总回答后进一步提问

- **总结回答：**在获得回答后，先复述总结对方提供的信息，确保你理解正确。示例：“你建议我们使用缓存来优化订单服务的性能，具体可以使用 Redis，对吗？”



Peking
University



- **追问细节：**如果回答不够详细或存在疑问，可以进一步追问。
 - **示例：**“关于使用 Redis 缓存，你能提供一些具体的实现示例吗？”
- **建议验证：**如果对方提供了解决方案，但其超出了我们的知识范围，可以让大模型工具提供方案验证其可行性或请求更多参考资料。
 - **示例：**“你上面提到的微服务架构优化方案，是否有相关的案例研究或文档可以参考？”

➤ 6. 总结梳理

- **多轮对话过程梳理总结：**根据和大模型的沟通交互，不断地明确共识和分歧，多次迭代后，最终达成共识，将最终的方案总结梳理出来。
 - **示例：**在与大模型的多轮对话后，“经过与你的多反讨论，我们确定了以下解决方案：(1)使用 Redis 缓存优化订单服务的性能。(2)采用分布式锁解决数据一致性问题。(3)使用消息队列异步处理订单，进一步提高系统吞吐量。你看看还有什么其他的建议吗？没有问题我们就按照这个方案实施了。”

□ GitHub Copilot

- GitHub Copilot 是由 GitHub (微软) 与 OpenAI 联合推出的 AI 编码助手，内嵌于 VS Code 等主流 IDE
- 免费订阅计划限制每月 2000 次代码补全，完成 GitHub 教育认证可以获得无限制代码补全
- 在 VSCode 中，安装 GitHub Copilot 与 GitHub Copilot Chat 扩展，登录与授权后即可开始使用





□ GitHub Copilot

- 自动代码生成功能，Tab 补全

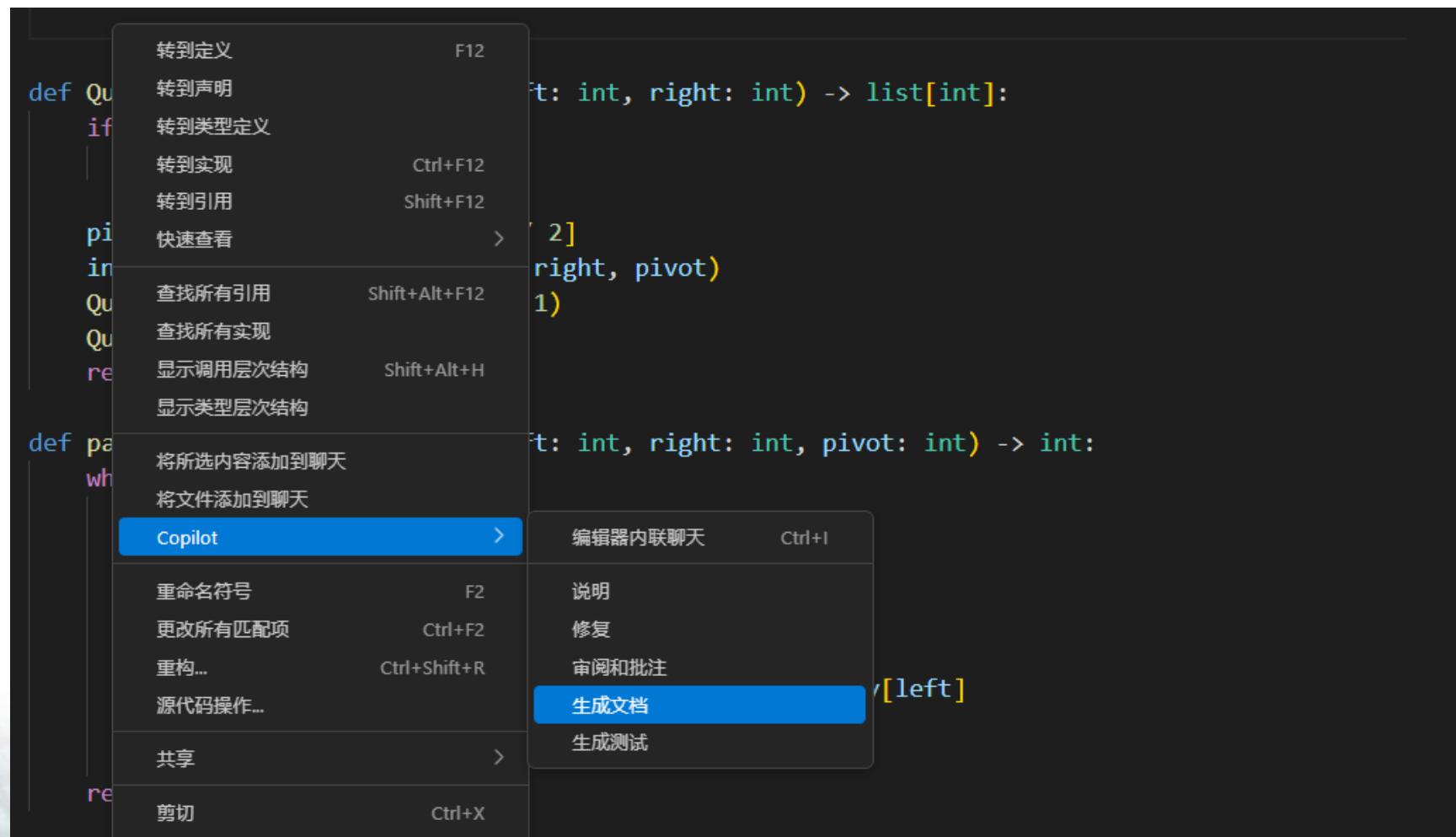
```
# 示例：开始编写一个排序函数
def QuickSort(array: list[int], left: int, right: int) -> list[int]:
    if left >= right:
        return array

    pivot = array[(left + right) // 2]
    index = partition(array, left, right, pivot)
    QuickSort(array, left, index - 1)
    QuickSort(array, index, right)
    return array
```



□ GitHub Copilot

➤ 自动生成文档





□ GitHub Copilot

➤ 自动生成文档

```
def QuickSort(array: list[int], left: int, right: int) -> list[int]:
    """
    This module implements the QuickSort algorithm for sorting a list of integers.
    Functions:
        QuickSort(array: list[int], left: int, right: int) -> list[int]:
            Recursively sorts the input list using the QuickSort algorithm.
        partition(array: list[int], left: int, right: int, pivot: int) -> int:
            Partitions the input list around a pivot value, ensuring that all elements
            less than the pivot are on the left and all elements greater than the pivot
            are on the right.
    Usage:
        - Call the `QuickSort` function with the list to be sorted and the indices of
          the range to sort (typically 0 and len(array) - 1).
        - The `partition` function is used internally by `QuickSort` to rearrange the
          elements around the pivot.
    """
    if left >= right:
        return array

    pivot = array[(left + right) // 2]
    index = partition(array, left, right, pivot)
    QuickSort(array, left, index - 1)
```

□ GitHub Copilot Chat

- Ask模式：用于一般性问题解答和代码解释
- Edit模式：专注于多文件代码编辑和重构
- Agent模式：可以自主完成复杂编码任务

The screenshot shows the GitHub Copilot Chat interface. On the left, there is a code editor window displaying a Python file named `sort.py`. The code implements the QuickSort algorithm. On the right, there is a panel titled "Explain the structure of this file." which provides an analysis of the code's structure and logic.

```

1
2
3
4
5
6 def Quicksort(array: list[int], left: int, right: int) -> list[int]:
7     """
8         This module implements the Quicksort algorithm for sorting a list of integers.
9     Functions:
10        Quicksort(array: list[int], left: int, right: int) -> list[int]:
11            Recursively sorts the input list using the Quicksort algorithm.
12        partition(array: list[int], left: int, right: int, pivot: int) -> int:
13            Partitions the input list around a pivot value, ensuring that all elements
14            less than the pivot are on the left and all elements greater than the pivot
15            are on the right.
16        Usage:
17            - Call the `Quicksort` function with the list to be sorted and the indices of
18            the range to sort (typically 0 and len(array) - 1).
19            - The `partition` function is used internally by `Quicksort` to rearrange the
20            elements around the pivot.
21        ...
22
23    if left >= right:
24        return array
25
26    pivot = array[(left + right) // 2]
27    index = partition(array, left, right, pivot)
28    Quicksort(array, left, index - 1)
29    Quicksort(array, index, right)
30    return array
31
32 def partition(array: list[int], left: int, right: int, pivot: int) -> int:
33     while left <= right:
34         while array[left] < pivot:
35             left += 1
36         while array[right] > pivot:
37             right -= 1
38         if left >= right:
39             break
40         array[left], array[right] = array[right], array[left]
41
42     return right

```

Explain the structure of this file.

sort.py

> 已使用 1 个引用

该文件实现了快速排序 (Quicksort) 算法，结构如下：

- **Quicksort 函数**：主排序函数，递归地对列表进行排序。参数包括待排序列表 `[array]`，排序范围的左、右索引 `[left]` 和 `[right]`。核心逻辑是选取中间元素为 pivot，通过 `[partition]` 函数分区后递归排序左右两部分。
- **partition 函数**：辅助分区函数。根据 pivot 值，将列表分为左侧小于 pivot、右侧大于 pivot 的两部分，并返回新的分区点索引。
- 文档字符串：在 `[Quicksort]` 函数顶部，详细说明了模块用途、函数功能和使用方法。

整体结构清晰，分为排序主逻辑和分区辅助逻辑，适合用于整数列表的快速排序。

Ask GPT-4.1

□ Vibe Coding 氛围编程

- 将代码编写的主导权完全移交给AI
- 开发者通过自然语言指令“指挥”AI，观察和验证程序的运行结果，并再向AI提供反馈
- 其核心工具生态包括 GitHub Copilot、Cursor Chat 等IDE集成与编辑器插件
- 最理想的应用场景是快速原型开发。它能够迅速地将一个想法变为可交互的原型，实现快速迭代
- 但在生产环境中直接使用Vibe Coding是很危险的，其中潜在的漏洞、性能瓶颈等问题仍需人工评审



口其他流行的 LLM 工具

- Cursor: 无缝衔接 VSCode 的 IDE，在自动编辑、生成代码之外，运行用户配置外部文档作为知识库，以及加入内置的 System Prompt
- Trae: 与 Cursor 类似的 AI 原生 IDE，其主要特性为以 AI 为主导的多轮迭代式协作。AI 不仅响应指令，而且主动提问，与用户共同决策
- ChatGPT, DeepSeek, 豆包、Qwen 等通用 LLM: 在调试复杂报错、设计软件系统架构、理解算法/数据结构原理、帮助学习新技术框架等方面具有优势。



大模型工具推荐

工具名称	特点	适用场景	优点
IDE 插件	Cursor <ul style="list-style-type: none">- 深度集成到 IDE 中，可用于代码生成、代码适配、代码优化和注释生成等开发任务。- 通过实时提示和建议，优化开发体验。	编码实现阶段：代码生成、适配、优化、注释生成等代码开发任务。 运维与维护阶段：对现有代码进行优化。	<ul style="list-style-type: none">- 集成 IDE，开发体验流畅，不需要从聊天界面来回复制粘贴
	aiXcoder <ul style="list-style-type: none">- 中文交互能力优秀- 支持多种编程语言	编码实现阶段：补全或生成代码片段、生成代码注释。 测试阶段：生成单元测并做 Bug 修复。 维护阶段：对当前代码库进行代码解释。	<ul style="list-style-type: none">- 国产开源免费使用- 集成 IDE，开发体验流畅
	Github Copilot/ Copilot <ul style="list-style-type: none">- 支持代码生成、代码优化、注释生成以及跨文件和项目级代码理解任务。- 基于 GitHub 仓库的强大支持，可提供项目上下文相关的高质量建议。	编码实现阶段：快速生成复杂代码逻辑。 维护阶段：对代码库进行理解和重构。	<ul style="list-style-type: none">- GitHub 网页版，跨文件、项目级程序理解能力强- 集成 IDE 版，开发体验良好





LLMs 工具	Claude	<ul style="list-style-type: none">- 理解能力强，可生成高质量文档内容。	<p>需求分析阶段：生成用户故事、需求文档模板及澄清模糊需求。</p> <p>测试阶段：生成测试用例和测试报告。</p>	<ul style="list-style-type: none">- 理解能力强，生成质量高，适合各种软件文档的生成，英文交互效果更好
	DeepSeek	<ul style="list-style-type: none">- 轻量级代码生成、文档生成功能，中文能力好。- 对文档处理和简单代码生成任务的支持。	<p>需求分析阶段：分析市场需求或生成项目创意。</p> <p>文档整理阶段：生成需求文档或初步设计文档。</p>	<ul style="list-style-type: none">- 中文支持好- 适合处理多文档和轻量级任务
	KiMi	<ul style="list-style-type: none">- 适用于文档总结处理、网页总结等需要联网搜索的任务。- 擅长提取和总结海量文本信息。	<p>需求分析阶段：生成项目背景调研报告或市场分析总结。</p> <p>文档处理阶段：快速总结用户反馈或需求说明。</p>	<ul style="list-style-type: none">- 对文档和网页总结效果好，适合综合信息处理任务。- 中文支持出色，适合本地化需求。





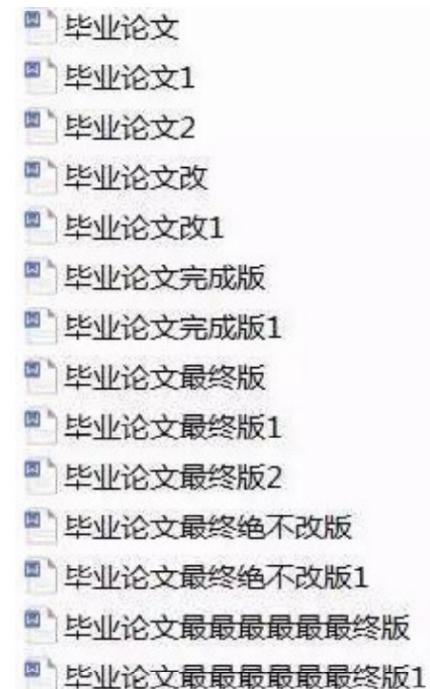
常用技术 GIT介绍



Peking
University

□ 版本控制系统（即VCS）会跟踪用户或团队在合作开发项目时的修改历史。随着项目的发展，团队可以随时运行测试，修复错误并贡献新代码，并且可以恢复到之前的任何版本。开发者可以查看项目历史记录以找出：

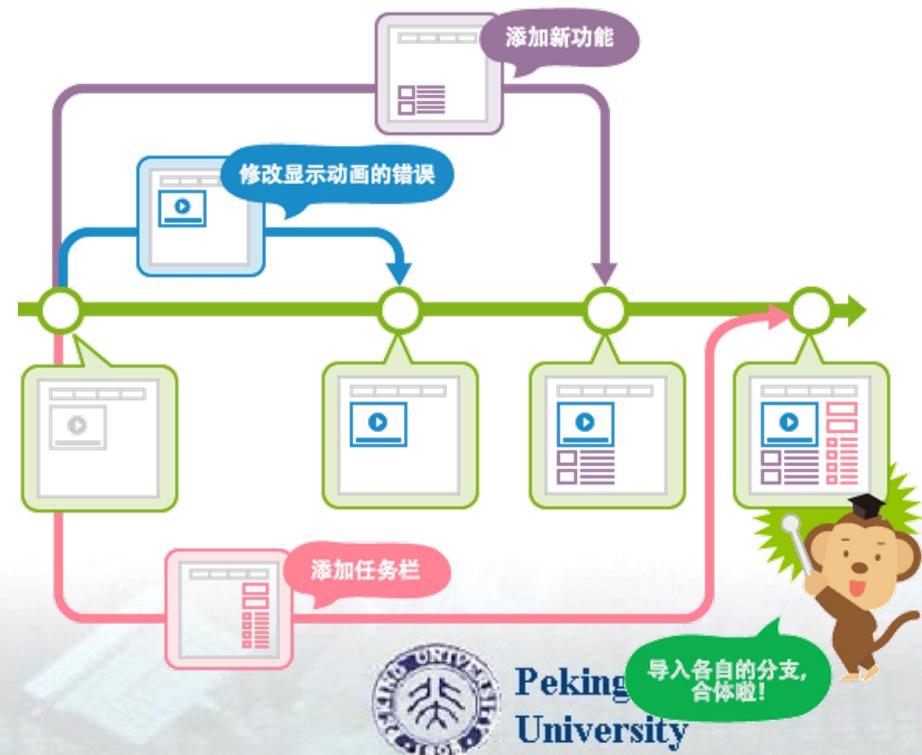
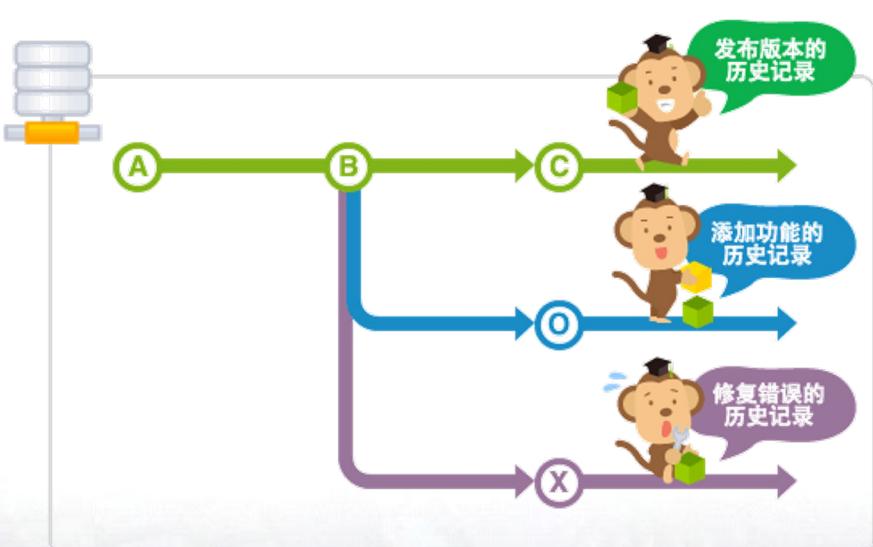
- 做了哪些更改？
- 谁做出了改变？
- 什么时候做出改变？
- 为什么需要更改？



如果没有版本控制，开发者将承担更多冗余的任务以及造成许多代码副本。为了减少不必要的工作，Git和其他VCS为每个贡献者提供了对项目的统一和一致的视图，从而显示出已经在进行中的工作。看到透明的变更历史以及他们如何为项目的开发做出贡献，有助于团队成员在独立工作时保持一致。

口分支是什么

- “分支”是版本控制系统中保存项目修改的普遍方式
- 分支上的每个节点都代表着对项目的一次修改（即形成一个版本）
- 分支可以分叉，分叉后的分支不受其他分支的影响，所以使用分支后、在同一个项目里可以同时进行多个修改。
- 分叉的分支可以合并，将不同的修改整合到一个版本上。



□ 使用命令行进行操作，不要使用图形界面

- 让你了解每个命令的作用与结果

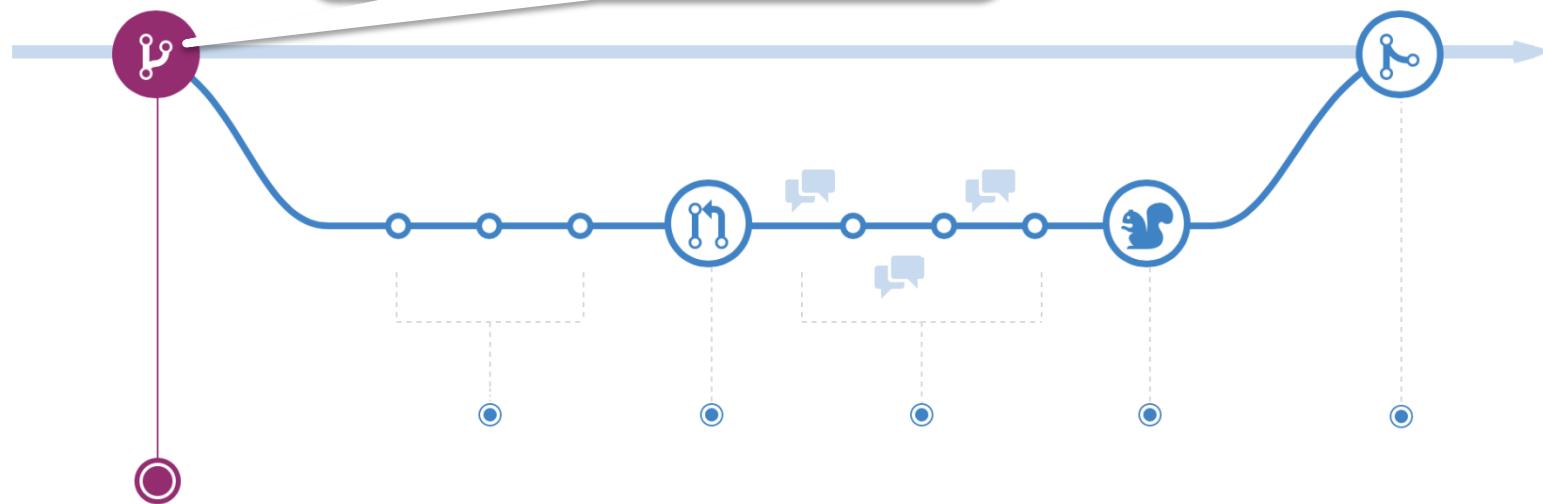
□ 多多实践

□ 猴子都能懂的Git入门

- https://backlog.com/git-tutorial/cn/intro/intro1_1.html



使用流程



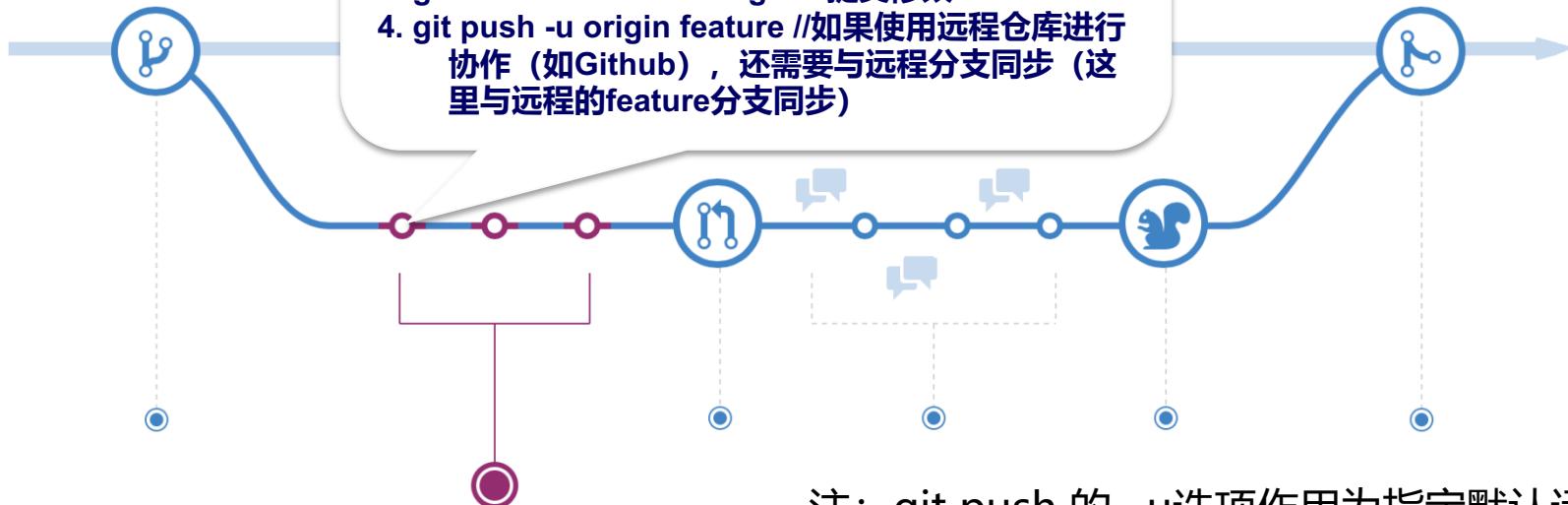
□ 创建分支

- 创建一个分支意味着创造一个可以尝试新想法的环境。在分支上所做的更改不会影响master分支，因此可以自由地尝试并提交更改。



Peking
University

使用流程



注: git push 的 -u 选项作用为指定默认远程主机，只需使用一次，以后就可以不加参数使用git push

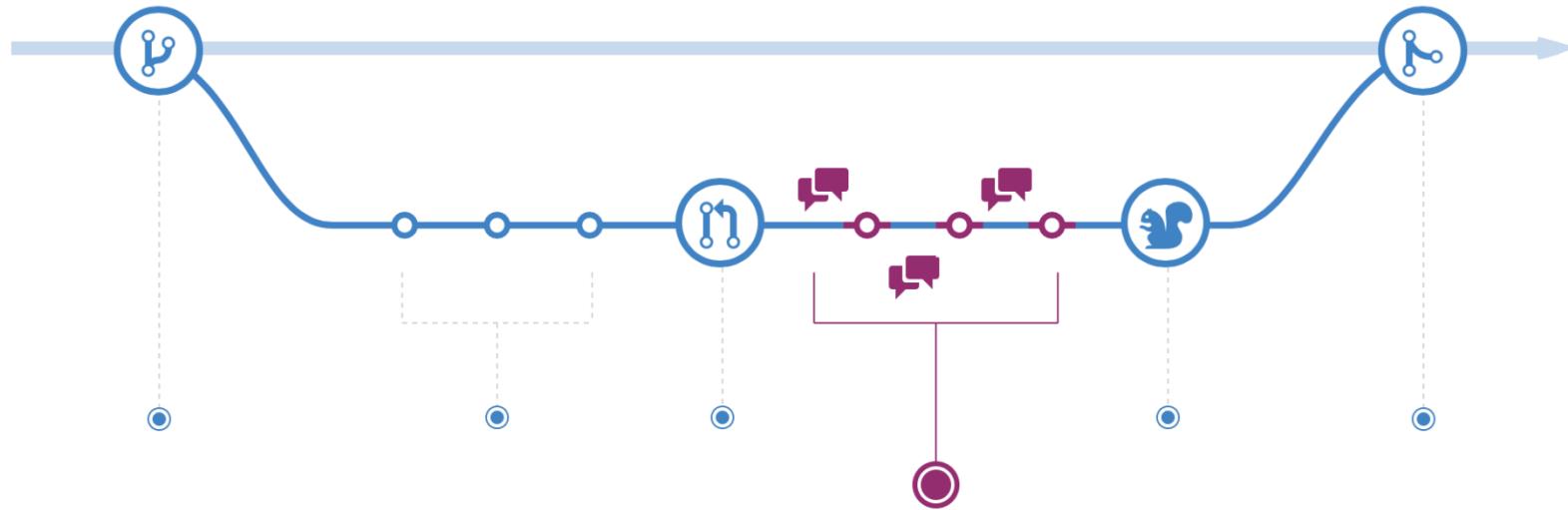
□ 添加提交

➤ 提交会创建一个透明的工作历史记录，其他人可以根据这些记录了解你所做的工作以及原因。每个提交都有一个关联的message，它是解释为什么进行特定更改的说明。而且，每个提交都被视为一个单独的变更单位。如果发现错误，你可以回滚这些提交。





使用流程

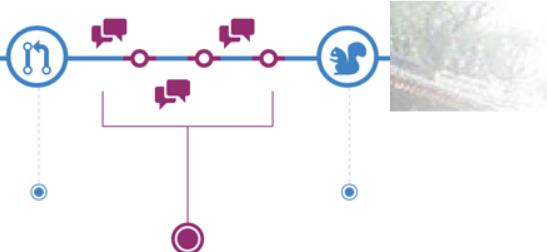


Code Review

- 当你希望将你的工作merge到其他分支时，你可以将其提出来（pull request），并与 other 参与者进行讨论，Review 你的代码并进行修改，直到所有人同意将其合并。



Peking
University



在Github上提出pull request

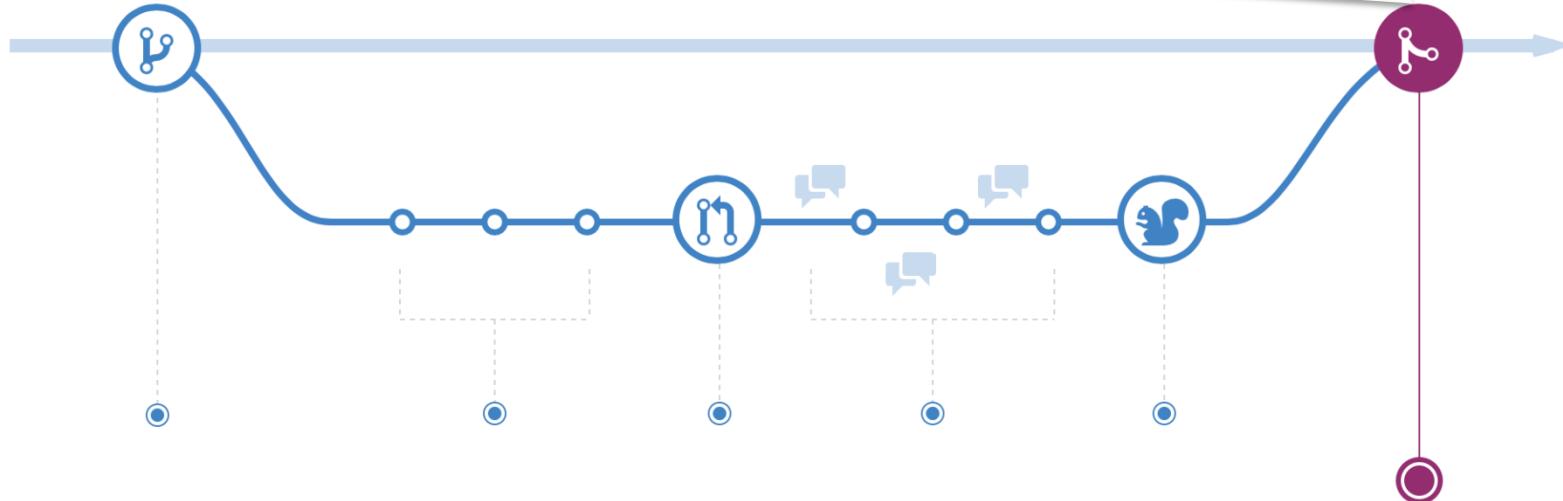
1. 进入小组github仓库，点击上方的Pull requests一栏，选择new pull request
2. 在进入的页面上方选择将feature1分支和dev分支进行比较，提示able to merge后，选择create pull request，填写信息后创建一个pull request
3. Pull request被创建，组员都可以在仓库的Pull requests页面看到，经过讨论可以后，点击merge pull request就可以将feature1与dev分支进行合并

The first screenshot shows the GitHub repository interface with the 'Pull requests' tab selected. A green arrow points down to the second screenshot, which displays a comparison between the 'base: dev' and 'compare: feature1' branches. It shows 1 commit, 1 file changed, and 0 commit comments, with 1 contributor. A green arrow points down to the third screenshot, which shows the details of the newly created pull request titled 'feature1 #1'. The pull request summary indicates that 'delavet wants to merge 1 commit into dev from feature1'. The conversation tab shows a comment from 'delavet' and a note about no description provided. The 'Merge pull request' button is highlighted at the bottom.



使用流程

```
git merge feature1 -m "message" //将feature1分支的修改合并到当前的分支
```



口合并

- 现在可以将代码合并到主分支中。合并后，合并请求将保留对代码的历史更改的记录。由于它们是可搜索的，它们让任何人都能及时了解每次提交所解决的问题与原因。



Peking
University



Git 基本命令

- git init 初始化一个全新的Git仓库，并开始跟踪一个现有的目录。它在现有目录中添加一个隐藏的子文件夹，该目录包含版本控制所需的内部数据结构。
- git clone 创建一个远程存在的项目的本地副本。克隆包括所有项目的文件，历史和分支。
- git add 增加一个改变。Git会跟踪开发者代码库的变化，但是有必要对这些变化进行阶段和快照，以将其包含在项目的历史记录中。任何变更都将成为下一个快照的一部分，也是项目历史的一部分。单独进行分段和提交可以使开发人员完全控制其项目历史，而不必改变他们的代码和工作方式。
- git commit 将快照保存到项目历史记录并完成更改跟踪过程。简而言之，提交功能就像拍照一样。使用git add都将成为git commit快照的一部分。

- git status显示更改的状态为untracked, modified或staged。
- git branch显示了本地正在运行的分支。
- git merge将分支合并在一起。此命令通常用于组合在两个不同分支上所做的更改。例如，当开发人员想要将功能分支的更改组合到主分支中进行部署时，他们会进行合并。
- git pull使用远程仓库来更新本地。如果队友向远程分支提交代码，使用此命令并可以将这些更改拉取到本地。
- git push将本地仓库更新到远程仓库。





Git 较高级命令

- git cherry-pick 将指定的提交（commit）应用于其他分支
- git reset 回滚本地的commit到指定HEAD，--soft保留更改，--hard丢弃更改
- git bisect 在指定的commit范围内进行二分查找，在寻找第一次出现错误的commit非常有用
- git rebase 使用-i 命令可以合并复数个commit为一个，直接使用rebase相当于merge
-



- 初学者可以先尝试使用master作为主分支，针对自己负责的功能单独开branch的方法来进行开发。
- 有一定基础的小组可以尝试使用更加科学的git flow。



□ 在提交时必须附上明确的message以说明该commit。
同样每个团队有自己的风格，例如：

- Feat: add map sdk

□ 可以换行进行详细描述

□ 严禁使用例如“最新版本”、“Fix bug”、“test”等message



口常见message类别有：

- feat: 新增feature
- fix: 修复bug
- docs: 仅仅修改了文档，比如README, CHANGELOG, CONTRIBUTUTE等等
- style: 仅仅修改了空格、格式缩进、逗号等等，不改变代码逻辑
- refactor: 代码重构，没有加新功能或者修复bug
- perf: 优化相关，比如提升性能、体验
- test: 测试用例，包括单元测试、集成测试等
- chore: 改变构建流程、或者增加依赖库、工具等
- revert: 回滚到上一个版本

- 在当前目录新建.gitignore文件可以让git忽略其中所列路径/文件
- 可以使用正则表达式进行匹配
- 一般会对中间文件、临时文件、过于庞大的文件、配置文件等进行忽略。



- 在进行git merge或者rebase时，当对同一个文件都做了不同改动时便会出现冲突，这个时候需要开发者手动进行冲突解决。
- 直接编辑冲突文件，修复下图的冲突信息，并创建一个新提交即可

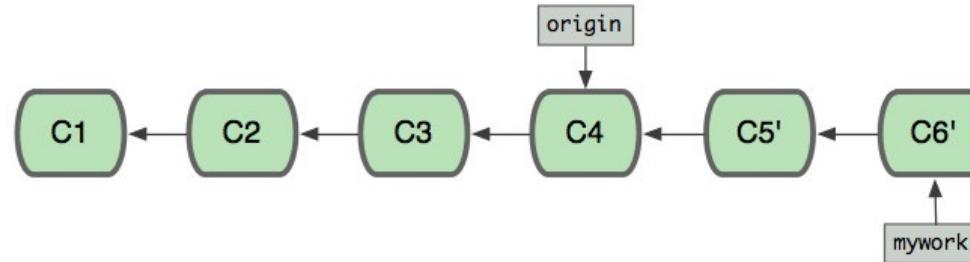
```
<<<<< HEAD
<div id="footer">contact : email.support@github.com</div>
=====
<div id="footer">
  please contact us at support@github.com
</div>
>>>>> iss53
```



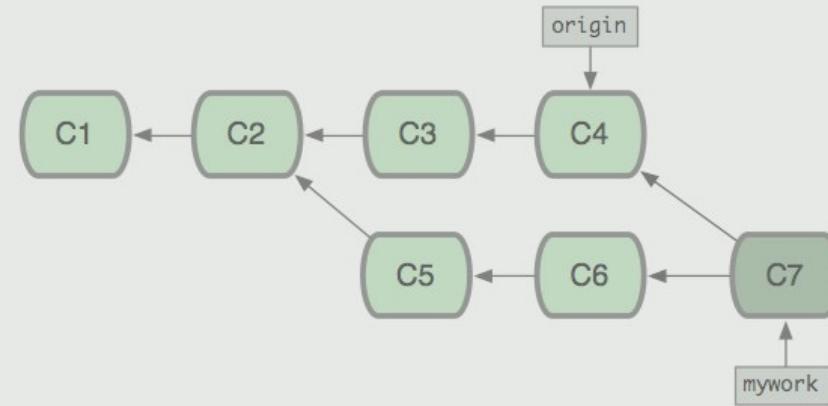


Merge vs rebase

git rebase



git merge



git rebase: 可看做一种隐性merge，不会有显式commit产生，也不会保留分支信息
为了对项目分支进行明确地管理和方便助教查看，建议大家还是使用merge

- 在实际生产过程中会有Master、Dev、Feature、Hotfix等分支，每个团队会有自己的工作流。一般来说Master分支为线上版本，是实际被使用的版本，其要求分支上的每个节点都是可被部署的。而Dev分支主要用于开发时的主分支，对每个Feature的开发一般会在Dev上单独开一个分支。另外遇到紧急问题，也可能会在Master上紧急开Hotfix分支以做修复。视具体情况而定。
- 由于Git仓库的公开性，所有需求保密的信息均不可加入Git中，所有信息在被提交到仓库中的瞬间视为已公开，秘钥、配置信息等均需加入ignore中，仓库中可保留配置示例文件。

- 每个commit需要保持独立性，只针对一个问题进行commit，不要在一个commit中提交过多过杂的内容。
- 可以尝试使用git rebase -i对本地分支（非Dev、Master）的commit进行精简与合并
- 在使用git push -f 的时候一定要想清楚
- 任何对远程仓库提交历史的修改/删除，都需要有组长或涉及的所有人的同意

- GitHub中Git的功能远不止如此，欢迎大家积极使用，例如将敏捷开发方法结合kanban、工单、issue、pull request、code review、CI等。
- 如果对其他组的工作内容有疑问，欢迎向他们的项目提issue。

① 3,099 Open ✓ 19,913 Closed		Author ▾	Label ▾	Projects ▾	Milestones ▾	Assignee ▾	Sort ▾
①	Can't open the camera/webcam due to a tensorflow installation problem. stat:awaiting response subtype:windows type:build/install						1
#36725	opened 39 minutes ago by Zayr0						
①	Bug for TF2.x + TensorRT(7) for batch size 1 when minimum_segment_size=2 comp:gpu:tensorrt type:bug						1
#36724	opened 2 hours ago by mankeyboy						
①	Allow TrtGraphConverterV2 to accept Frozen Graph input as well as saved_model comp:apis type:feature						1
#36723	opened 5 hours ago by mankeyboy						
①	Bug: Cannot load model with tf.keras.layers.Lambda & eager functions enabled						1
#36722	opened 5 hours ago by phemmer						
①	Github issue creation for bugs missing						1
#36721	opened 5 hours ago by phemmer						
①	Getting SavedModel format from Checkpoint automatically	TF 2.1	comp:apis	type:feature			1
#36720	opened 6 hours ago by mankeyboy						
①	name: Bug Issue about: why the performance of nnapi is much lower than cpu . labels: 'type:bug_template'						1
#36719	opened 7 hours ago by songxuemei						
①	[TPU Colab] [TF2.1] Many issues only on TPU due to data types, batch sizes and memory (gist provided with explanations)	TF 2.1	comp:tpus	type:bug			1
#36718	opened 7 hours ago by MichaelKarpe						
①	MKL no longer works with tensorflow 1.15	TF 1.15	type:bug	type:build/install			1
#36717	opened 9 hours ago by pavank						



□ Git Handbook

- <https://guides.github.com/introduction/git-handbook/>

□ Understanding the GitHub Flow

- <https://guides.github.com/introduction/flow/>

□ 猴子都能懂的Git入门

- https://backlog.com/git-tutorial/cn/intro/intro1_1.html

□ Git message

- <https://github.com/thoughtbot/dotfiles/blob/master/gitmessaging>

□ Git 分支练习

- <https://learngitbranching.js.org/>





常用技术 框架模式

□ 框架

- 软件系统发展至今天已经十分复杂，框架帮助你完成编写软件系统的很多基础工作，使你能够专注于系统的业务逻辑设计

□ 模式

- 模式是在某一背景下对某个问题的一种解决方案

□ 框架模式

- 顾名思义，框架所使用的模式
- 一般关注的问题：如何统合系统的界面、逻辑和数据

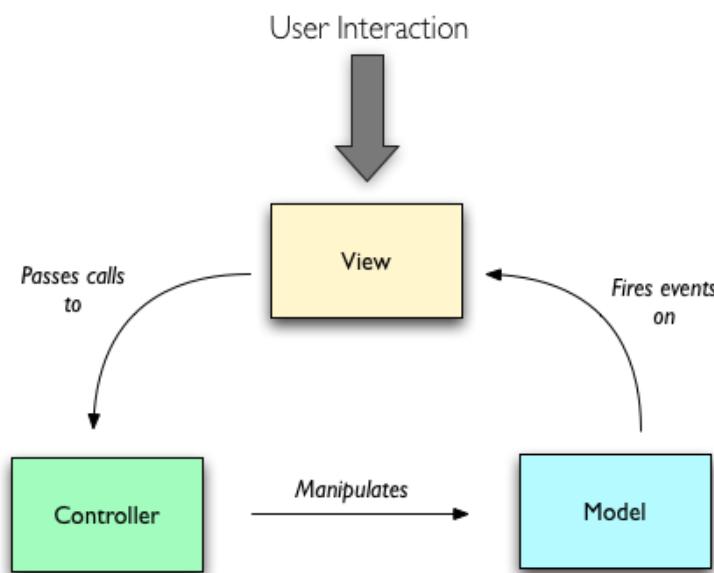


□ 三个部分

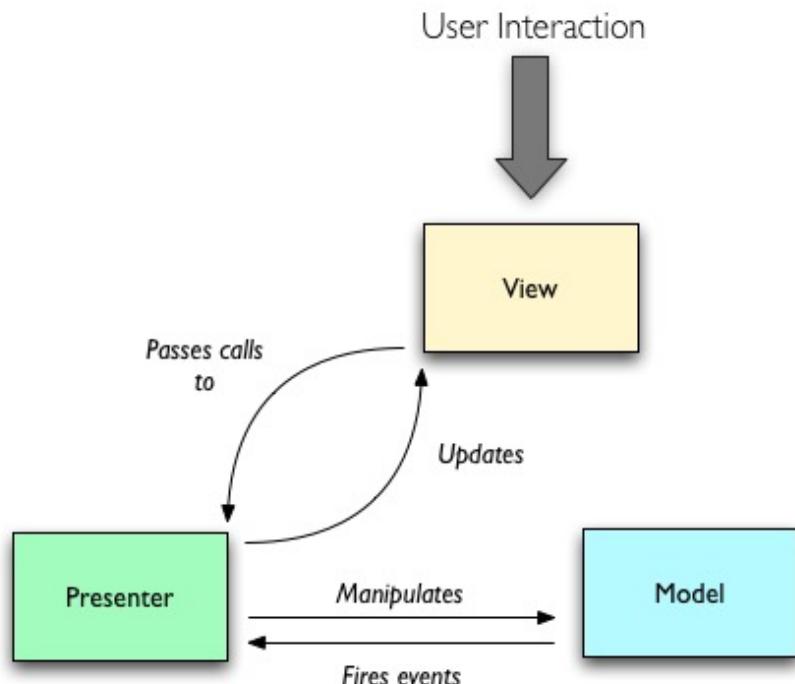
- 视图 (View) : 用户界面
- 控制器 (Controller) : 业务逻辑
- 模型 (Model) : 数据保存

□ 通信流程

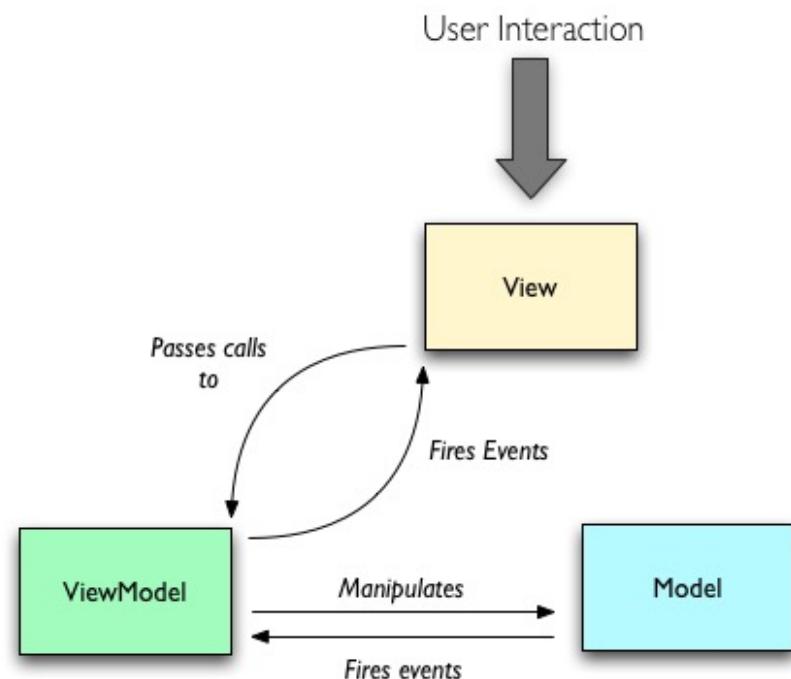
- View 传送指令到 Controller
- Controller 完成业务逻辑后，要求 Model 改变状态
- Model 将新的数据发送到 View
用户得到反馈



- 各部分之间的通信，都是双向的。
- View 与 Model 不发生联系，都通过 Presenter 传递。
- View 非常薄，不部署任何业务逻辑，称为“被动视图”（Passive View），即没有任何主动性，而 Presenter 非常厚，所有逻辑都部署在那里。



- MVVM 模式将 Presenter 改名为 ViewModel，基本上与 MVP 模式完全一致。
- 唯一的区别是，它采用双向绑定 (data-binding)：View 的变动自动反映在 ViewModel，反之亦然。





常用技术 前端技术



Peking
University

世界上第一个web页面

The screenshot shows a vintage web browser window titled "The World Wide Web project". The page content discusses the WorldWideWeb (W3) project, which aims for universal access to a large universe of documents. It links to various resources like executive summaries, mailing lists, and frequently asked questions. Below the main text, there's a section titled "What's out there?" followed by a list of pointers to online information, subjects, W3 servers, etc. Other sections include Help (on the browser), Software Products (a list of project components), Technical (details of protocols, formats, program internals), Bibliography (paper documentation on W3 and references), People (a list of people involved), History (a summary of the history of the project), How can I help? (information on supporting the web), and Getting code (instructions for getting the code via anonymous FTP). The browser interface includes a toolbar, address bar, and a tab bar with other open tabs.



Peking
University

现代web页面

哔哩哔哩 (゜-゜)つロ干杯~-bilibili

https://www.bilibili.com

北京大学网络服务 哔哩哔哩 - (゜-゜)つロ干杯~-bilibili 北京大学校内信息... 学习一个 126网易免费邮~你... dblp: computer sc... papers with code 知乎 GEXF - Network X... 正则表达式语言~... 虎牙直播-技术驱动... AcFun弹幕视频网... 其他收藏夹

主站 番剧 游戏中心 直播 会员购 漫画 赛事 下载APP

用起风了打开诸葛丞相的一生

会员 消息 动态 收藏 历史 创作中心 投稿

bilibili

动画 999+ 音乐 999+ 舞蹈 999+ 知识 999+ 生活 999+ 时尚 999+ 娱乐 999+ 放映厅 254+ 专栏 活动 小黑屋

首页 动态 热门 频道 番剧 263 国创 590 游戏 999+ 数码 999+ 鬼畜 146 资讯 999+ 影视 999+ 更多

直播 课堂 音乐PLUS

联系客服

换一换

直播 动画 番剧 国创 漫画 音乐 舞蹈 游戏 知识 课堂 数码 生活 美食 动物圈

推广 动物迷惑行为大赏 每天一个甩脂小技巧!

3月11日 江湖见

号外! 和马退赛另有隐情? 导演揭秘内幕!

攒多少人品才能拿到这样的神底牌

观看列表

https://www.bilibili.com/video/BV1Vz4y1174V7spm_id_from=333.851.b_7265636f6d6d656e64.4



Peking
University

□ HTML5

- 新一代的Web技术标准
- 实现富Web应用的本地化，摆脱Flash和Silverlight等浏览器插件的依赖
- 能够创建更简单的Web程序，书写更简洁的HTML代码

- HTML5结构元素
- 表单与文件
- 绘制图形
- 多媒体播放
- 本地存储
- ...

• CSS3

- Cascading Style Sheets
- 复杂效果简单化
 - 多背景
 - 圆角
 - 3D动画
 - 文字阴影
 - 颜色透明度



CSS



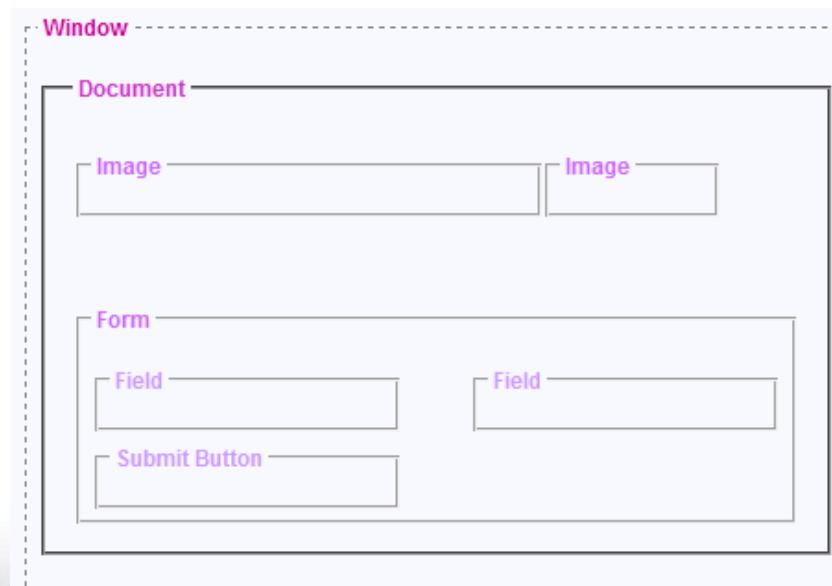
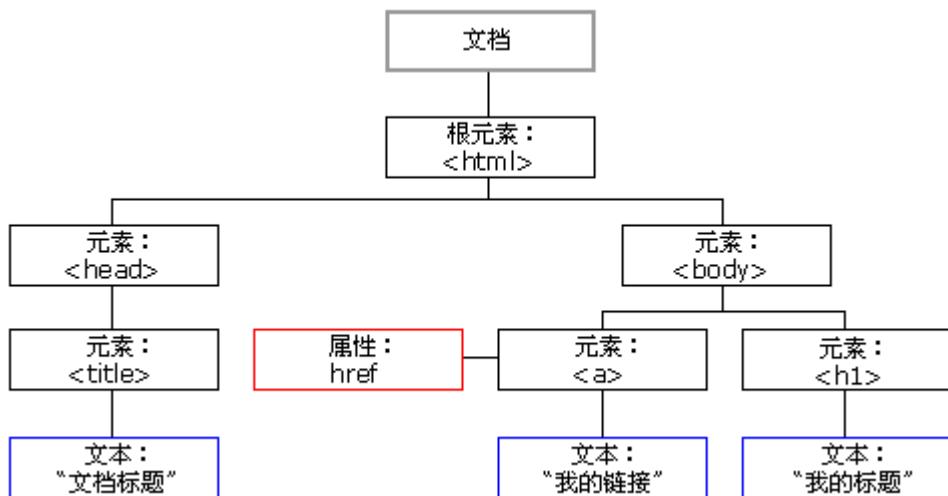
Peking
University

□ JavaScript

- JavaScript是一种脚本语言，其源代码在发往客户端运行之前不需经过编译，而是将文本格式的字符代码发送给浏览器由浏览器解释运行
- 目前JavaScript已经不限于在浏览器中使用，一些服务器开始支持JavaScript，如Node.js



- HTML DOM 是是 HTML 文档对象模型的英文缩写，Document Object Model for HTML。
- HTML DOM 定义了用于 HTML 的一系列标准的对象，以及访问和处理 HTML 文档的标准方法。



口事件是可以被 JavaScript 侦测到的行为

- JavaScript与HTML之间的交互是通过事件实现的
- 是文档或浏览器窗口中发生的一些特定的交互瞬间
- 可以使用监听器（处理程序）来预订事件，以便事件发生时执行相应的代码（观察员模式）
- 支持页面外观（HTML/CSS）与行为（JS）的松散耦合

口事件举例：

- 鼠标点击
- 页面或图像载入
- 鼠标悬浮于页面的某个热点之上
- 在表单中选取输入框
- 键盘按键





从静态页面到动态加载页面——AJAX

□ AJAX为“*Asynchronous JavaScript And XML*”

- 一种广泛应用于浏览器的网页开发技术
- 利用浏览器内置的XMLHttpRequest类，AJAX能够向服务器请求额外的数据而无须卸载页面
- 改变了自Web诞生以来一直沿用的“单击，等待”交互模式，增强了用户体验。

□ JSON

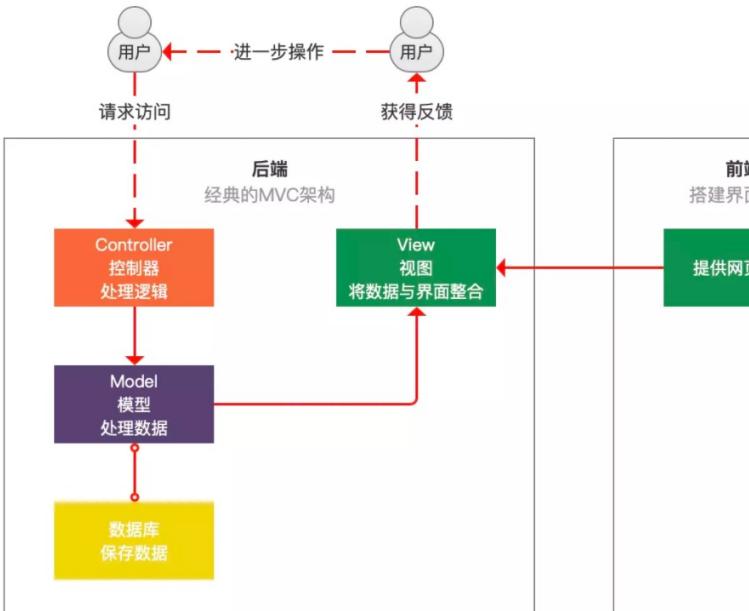
- JSON(*JavaScript Object Notation*) 是一种轻量级的数据交换格式。
- 现在AJAX多采用JSON格式传输数据。
- 对象：键值对；数组：值的有序列表。
- 可直接被JavaScript拿来使用。
- 如

```
{ 'obj' :{a:str,2:123," name" :" s" },' array' :[1 ,2,3,' carl' ]}
```

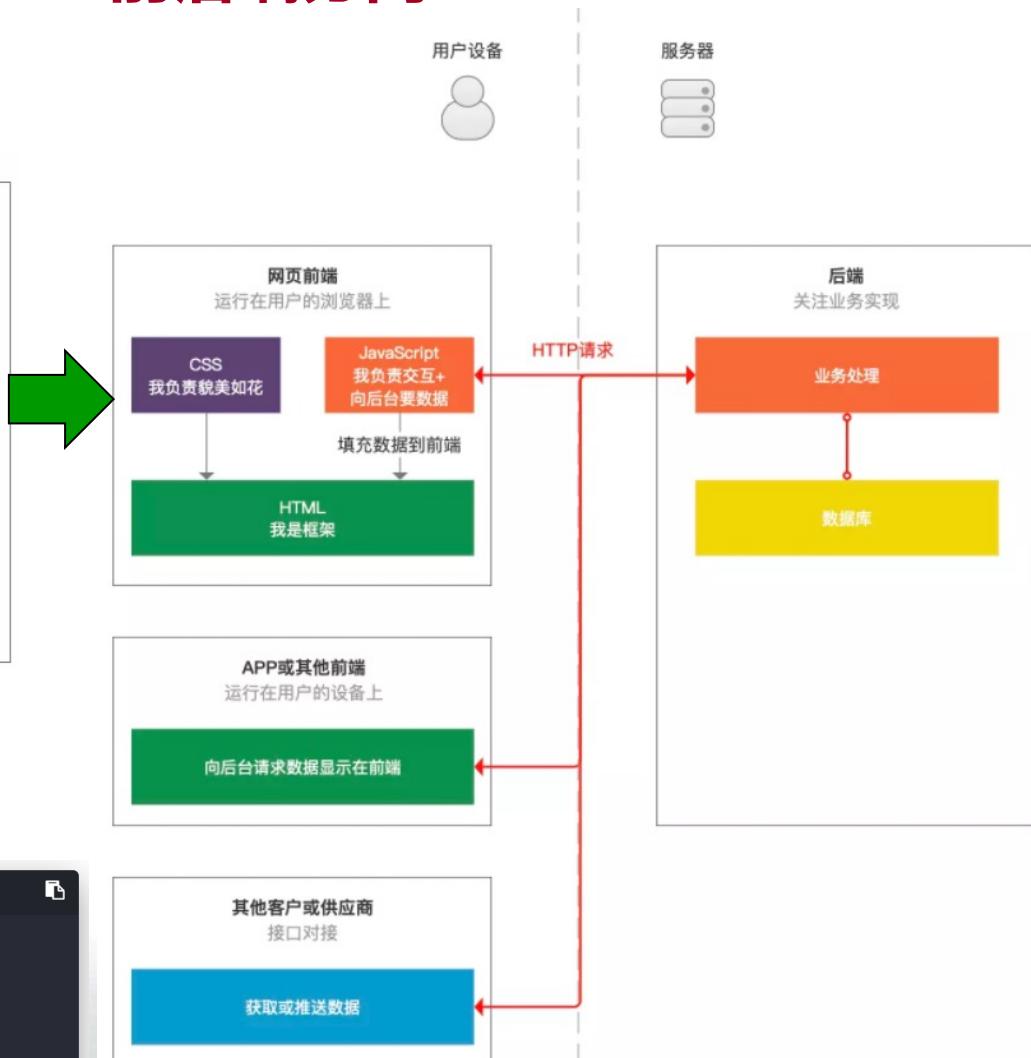


什么是前后端分离

前端后端不分离



前后端分离



分离后，前后端仅通过http请求交换数据。
常见的数据格式：JSON

```
1 {  
2   code: 200,  
3   data: {  
4     message: "success"  
5   }  
6 }
```

□ 动态页面前端主要处理的问题

➤ 控制页面渲染

- 编写模板和样式
- 通过 DOM 操作渲染初始数据
- 处理 DOM 事件控制组件样式
- 数据变更后通过 DOM 操作更新局部页面内容

➤ 维护本地内存中的数据

- 发送请求与服务器同步数据
- 数据加工适配和校验
- 更新用户操作时相对应的数据



- 现代前端项目一般选用一种现代前端框架完成开发
- 现代前端框架的主要作用

- 现代前端框架将页面渲染的能力抽象和封装出来

- 开发者只需要按照文档配置好模板即可，现代框架会自行将初始数据渲染成页面。
 - 如果数据有变更，现代框架也提供能力渲染页面保持与数据的同步

- 现代前端框架提供了可选的数据管理能力

- 通过框架提供的可选数据管理功能，开发者可以对一个复杂系统中涉及的各种系统状态（数据）进行方便地管理



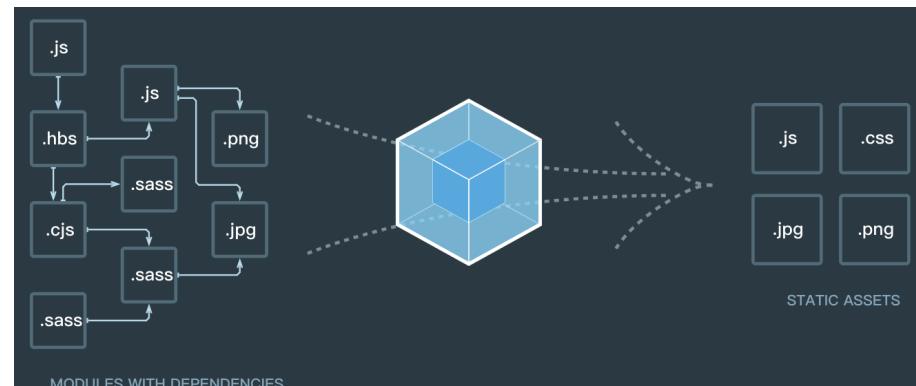
现代前端工程化与webpack

□ 前端的工程化困境

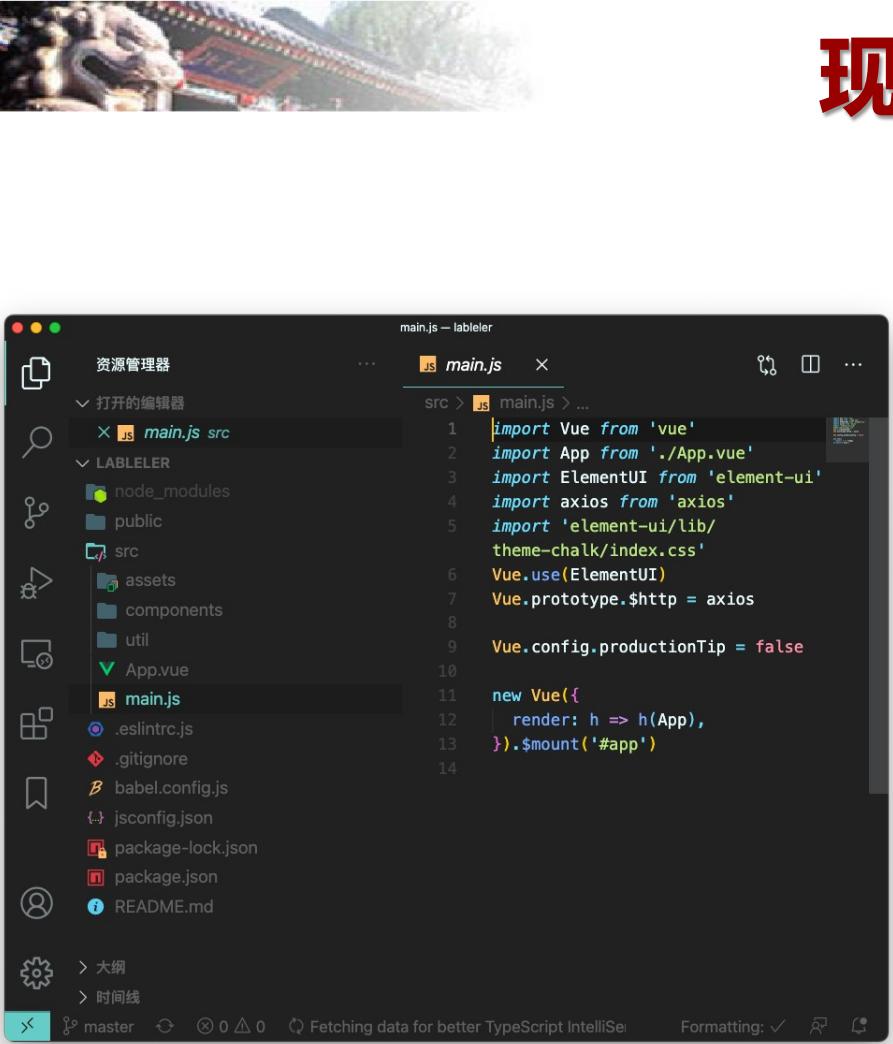
- 如何实现模块化?
 - 没有模块化的机制、所有相关代码只能写在一个文件里
- 如何进行包的复用和管理?
 - 运行在浏览器中的javascript无法直接使用第三方提供的包/库
- 如何实现组件化?
 - 所有的html组件，若想复用只能复制粘贴一遍代码
- 如何实现项目的自动化?
 - 自动化构建
 - 自动化部署
 - 自动化测试

□ Webpack

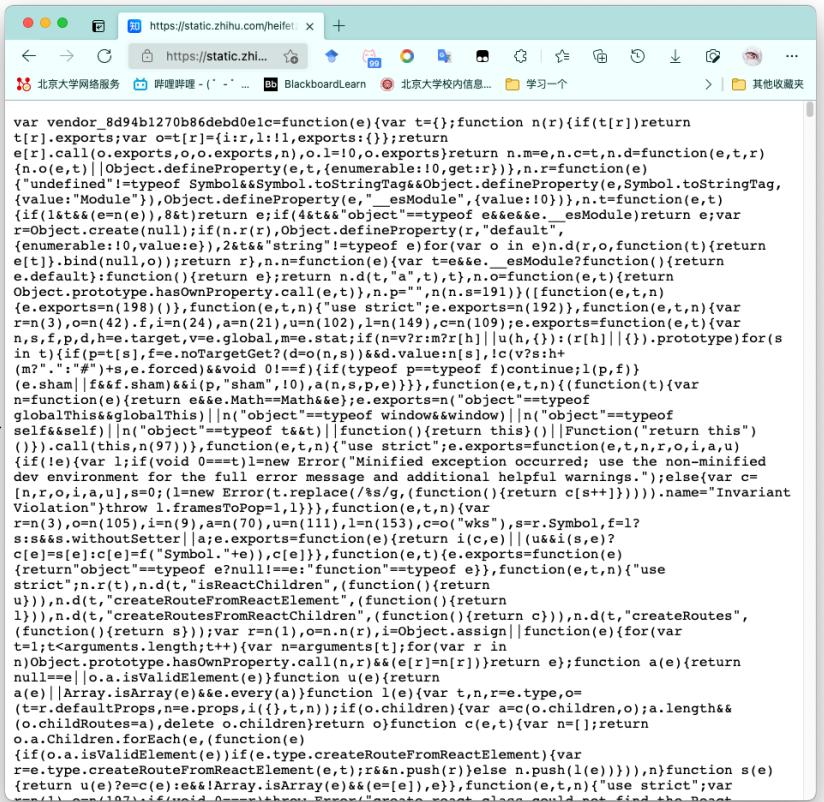
- 一个基于node.js的前端资源构建工具
 - 基于node，获得模块化、包管理、组件化等功能
- 将所有前端资源视为模块、并将所有模块最终打包成浏览器能执行的代码



现代前端工程化与webpack



Webpack打包前



Webpack打包后

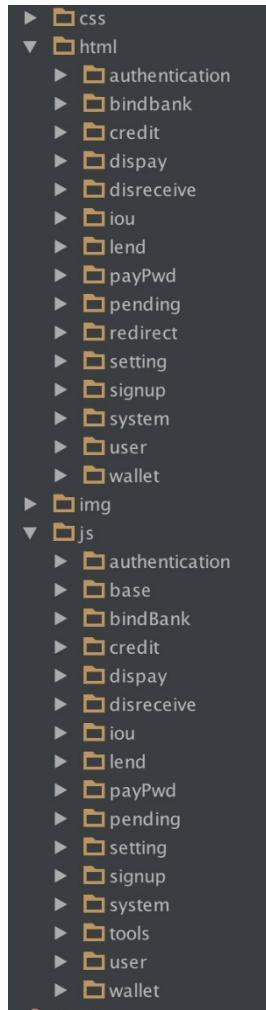


PEKING
UNIVERSITY

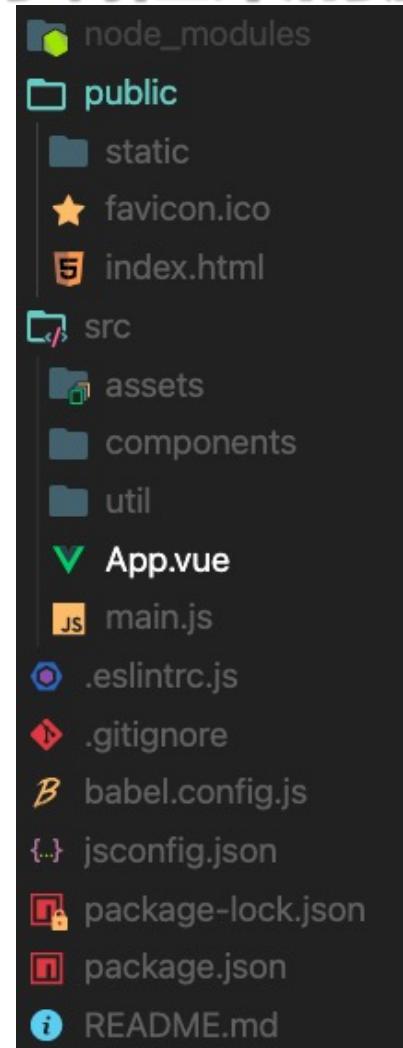
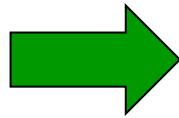
口现代前端框架的脚手架

- “脚手架”是现代前端框架为解决项目自动化、包管理等工程化问题而引入的命令行工具
- 脚手架一般基于Webpack构建，并提供默认的Webpack配置
- 通过使用脚手架，工程中的很多操作（搭建、测试、构建等）都可以通过命令自动化完成，使得前端的工程化开发更加快速与规范

现代前端框架的脚手架



不使用脚手架的前端项目



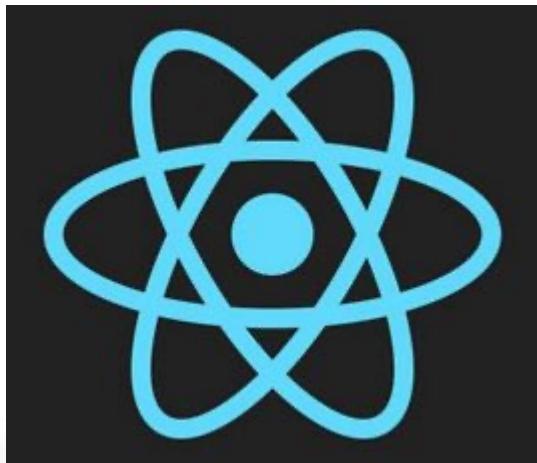
使用脚手架的前端项目



Peking
University

口目前web前端开发中最流行的三个框架

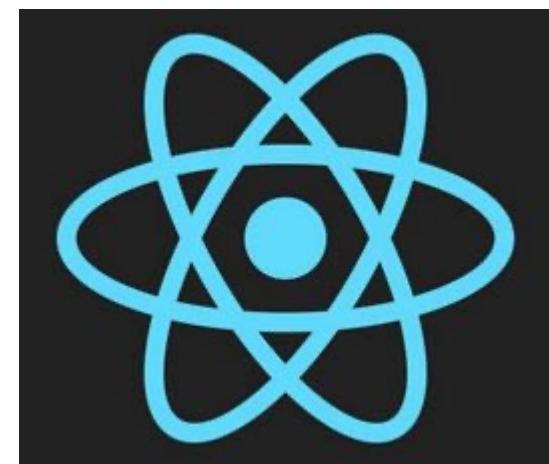
- React
- Vue.js
- Angular



Peking
University

□ React

- React是Facebook设计的一款前端的视图层框架
- React的核心思想
 - 虚拟DOM
 - 自定义组件
 - JSX



□ React

➤ 定义组件

```
class Contacts extends React.Component {  
  constructor(props) {  
    super(props);  
  }  
  handleClick() {  
    console.log(this); // null  
  }  
  render() {  
    return (  
      <div onClick={this.handleClick}></div>  
    );  
  }  
}
```

➤ 常用配套库

- Redux和Mobx，用于管理state
- Material UI, Ant design等UI库
- React-router前端路由

➤ 脚手架

- Create react app



□ Vue.js

- vuejs是一名华人设计师的项目，使用reactive的 databinding
- 设计理念上和React比较类似
 - 使用 Virtual DOM
 - 提供了响应式（Reactive）和组件化（Composable）的视图组件。
 - 将注意力集中保持在核心库，伴随于此，有配套的路由和负责处理全局状态管理的库。
- 渲染速度更快
- 模板机制
- 更加便捷的CSS



□ Vue.js

➤ 模板渲染

- 监听模板涉及data
- 编译模板为render函数
- 函数执行结果返回vnode
- Patch初次渲染或更新视图

```
<div id="app">  
  {{ message }}  
</div>
```

➤ 常用配套库

- Element, Vux等UI库
- vuex状态管理模式

```
var app = new Vue({  
  el: '#app',  
  data: {  
    message: 'Hello Vue!'  
  }  
})
```

➤ 脚手架

- Vue-cli



□ Angular

- Angular 是 google 的前 java 团队人员设计，设计模式使用较为深刻，有各种依赖注入、service，追求大而全。



如何选择

口建议

- 没有前端基础的选择 Vue
- 有基础可以选择任何一个框架开发



Peking
University

口下载node.js

WSH > vue-demo > 11:54

WSH > vue-demo > 11:55

Vue CLI v5.0.8

? Please pick a preset: (Use arrow keys)

> Default ([Vue 3] babel, eslint)

Default ([Vue 2] babel, eslint)

Manually select features

\$ cd vue-demo 运行vue项目

\$ npm run serve

DONE Compiled successfully in 6976ms

App running at:

- Local: http://localhost:8080/

- Network: http://10.7.187.22:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.

下载vue脚手架

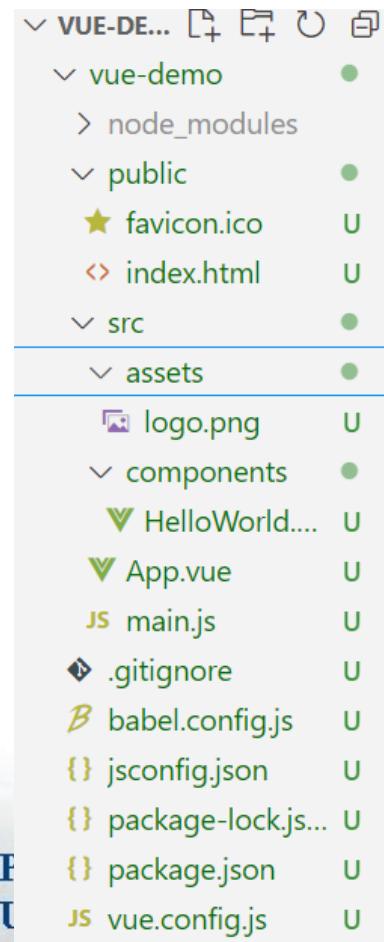
npm install -g @vue/cli

vue create vue-demo

创建vue项目

选择vue的版本

vue项目格式展示



一个vue的页面示例



Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the [vue-cli documentation](#).

Installed CLI Plugins

[babel](#) [eslint](#)

Essential Links

[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#) [News](#)

Ecosystem

[vue-router](#) [vuex](#) [vue-devtools](#) [vue-loader](#) [awesome-vue](#)

一个.vue代码文件包含三个部分，template部分是HTML代码，用来表示页面格式，script部分是JavaScript代码，主要是处理逻辑，style部分是css，表示页面风格

```
1 > <template>...
31   </template>
32
33 > <script>...
40   </script>
41
42   <!-- Add "scoped" attribute to limit CSS to this component only -->
43 > <style scoped>...
58   </style>
59   |
```



Vue开发案例

```
<template>
  <div class="hello">
    <button @click="decrement">-</button>
    <span>{{ count }}</span>
    <button @click="increment">+</button>
  </div>
</template>

<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      count: 0
    };
  },
  methods: {
    increment() {
      this.count++;
    },
    decrement() {
      this.count--;
    }
  }
}
</script>
```

一个简单的小组件的格式，
@click调用decrement和
increment，在methods中进行
逻辑处理，简单实现计数器功能



vue文件的组织格式

The screenshot shows a GitHub repository interface for a Vue project named "SE-Frontend". The left sidebar displays the file structure under the "dev" branch:

- components
- router
- stores
- views
 - DetailView.vue
 - ListView.vue
 - PostView.vue
- modules
 - DetailView.vue
 - ListView.vue
 - PostView.vue
- user
 - AboutView.vue
 - HomeView.vue
 - LoginView.vue
- App.vue
- main.js

The main area shows a list of commits in the "src" directory:

Name	Last commit message	Last commit date
..		
assets	feat:add some picture	2 years ago
components	chore: profile in NavBar	2 years ago
router	feat: localStorage for login status	2 years ago
stores	feat: localStorage for login status	2 years ago
views	Merge pull request #78 from SakuraWSH/wsh-new	2 years ago
App.vue	modify: no login status	2 years ago
main.js	chore: handle all error codes in login or signup	2 years ago



Vue开发案例

The screenshot shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure with folders like dev, public, src, assets, components, router, and user. The router folder contains index.js, stores, views, and modules. The modules folder contains DetailView.vue, ListView.vue, and PostView.vue. The code editor shows the content of router/index.js, which defines three routes: home, user, and chat, each pointing to a specific view component.

```
SE-Frontend / src / router / index.js
Code Blame 70 lines (68 loc) · 1.38 KB
18     },
19     {
20         path: '/home',
21         name: 'home',
22         meta: {
23             isLogin: true
24         },
25         component: () => import('../views/HomeView.vue')
26     },
27     {
28         path: '/user',
29         name: 'user',
30         meta: {
31             isLogin: true
32         },
33         component: () => import('../views/user/UserView.vue')
34     },
35     {
36         path: '/chat',
37         name: 'chat',
38         meta: {
39             isLogin: true
40         },
41         component: () => import('../views/user/ChatView.vue')
42     },
```

vue文件的路由和跳转

The screenshot shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure with folders like dev, router, stores, views, and user. The views folder contains modules, which in turn contains DetailView.vue, ListView.vue, and PostView.vue. The code editor shows the content of views/modules/DetailView.vue, which contains JavaScript and Vue template code for fetching post details and handling comments.

```
SE-Frontend / src / views / modules / DetailView.vue
Code Blame 305 lines (280 loc) · 6.19 KB
117     const _this = this;
118     axios({
119         method: "get",
120         url: "/api/post/detail",
121         params: {
122             id: _this.$route.query.pid,
123         },
124     }).then(data => {
125         console.log(data);
126         _this.detailItems = data.data;
127     });
128     this.fetchComments();
129 },
130 sendComment() {
131     if (this.comment.length == 0) {
132         this.$message({
133             message: '你好像什么都没输入哦？',
134             type: "warning"
135         })
136         return;
137     }
138     let _this = this;
139     axios({
140         method: "post",
```

与后端通信

口 测试框架

➤ Mocha

- 功能丰富的javascript测试框架，运行在node.js和浏览器中，方便进行异步测试。Mocha测试连续运行，允许灵活和准确的报告，同时将未捕获的异常映射到正确的测试用例。

➤ Chai

- 针对 Node.js 和浏览器的行为驱动开发和测试驱动开发的诊断库，可与任何 JavaScript 测试框架集成。



□ React

➤ <https://react.docschina.org/>

□ Vue

➤ <https://cn.vuejs.org/>

□ Angular

➤ <https://angular.cn/>

□ Mocha

➤ <https://github.com/mochajs/mocha>





常用技术 (微信) 小程序介绍

81

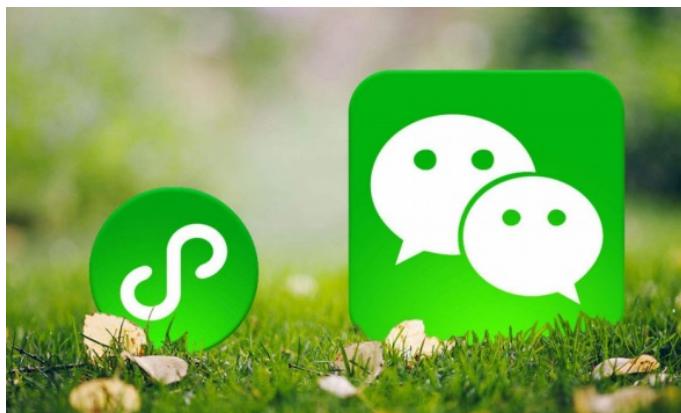
为什么选择小程序

□ 小程序现况

- 截至2024年10月，微信小程序用户规模约 9.49 亿人，接近或超过 90% 的微信活跃用户群体经常使用。微信小程序已覆盖全球约 92 个国家和地区，横跨 100+ 个细分行业。境外小程序月活、境外交易额等正在快速增长。

□ 小程序优势

- 小程序推广容易，冷启动难度降低
- 小程序开发适配成本低
- 小程序容易小规模试错，快速迭代
- 小程序跨平台
- 小程序具有丰富的商业化、生态支持



□ 设计理念

- 采用了MVVM设计模式，视图层与逻辑层相互分离
- 架构与语法类似于Vue.js

□ 文件类型

- JSON：静态的配置文件，如设置页面标题，网络超时时间等
- WXML：类似于HTML，用于描述页面的结构
- WXSS：类似于CSS，用于控制页面元素的样式
- JS：页面的处理逻辑



□ 框架组件

- App: 小程序，一个项目只能注册一个App，并且只能在app.js中声明
- Page: 页面，没错就是你们理解的那个页面
- Component: 组件，注册组件和引用组件需要在json文件中声明，父子组件之间可以通过properties传递参数
- Behavior: 行为，将组件间代码的共性抽取出来，当作一个“小组件”被引用，具有自己的生命周期、属性、方法等。

□ 微信提供的接口

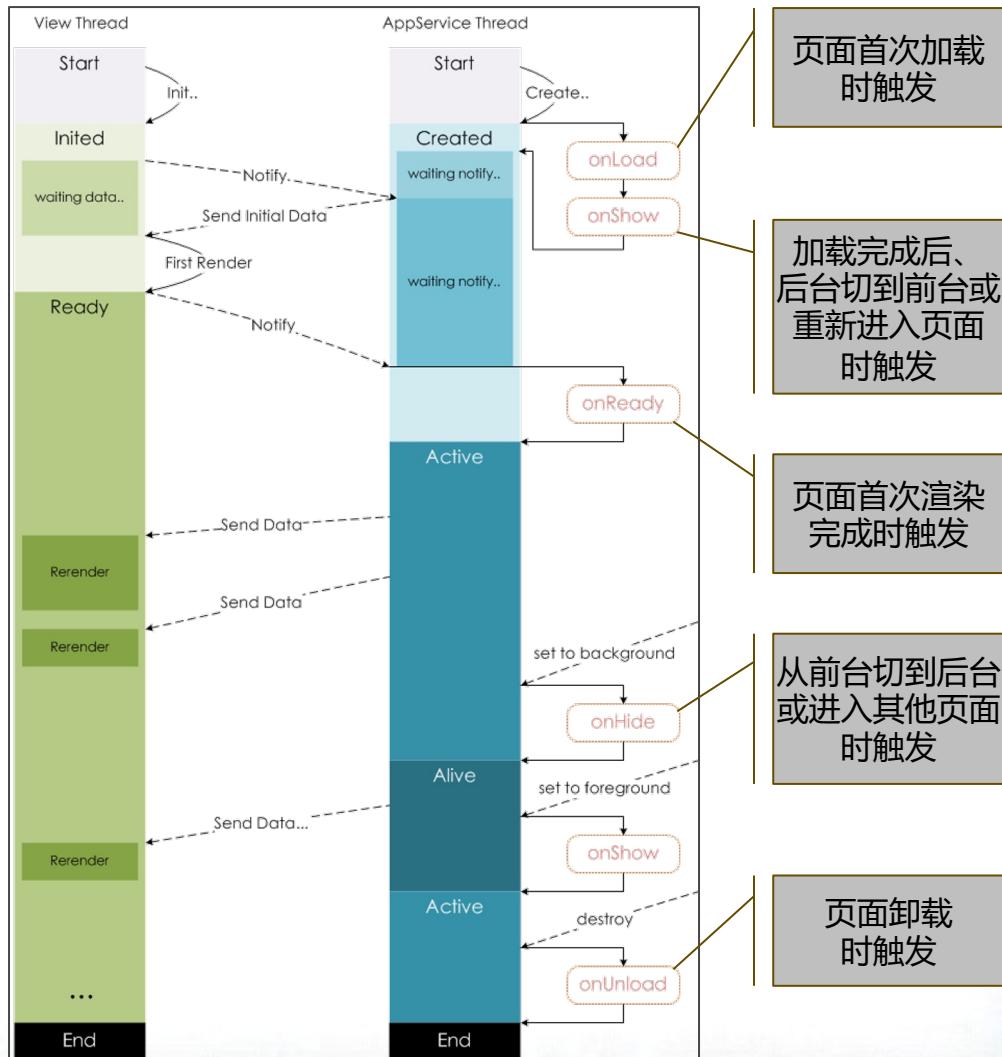
- 基础: wx.on...事件监听, wx.getSystemInfo获取系统信息等
- 界面: wx.showToast展示消息提示框等
- 路由: wx.switchTab, wx.navigateTo跳转页面等
- 网络: wx.request发起网络请求, wx.downloadFile下载文件等
- 更多请参考:
<https://developers.weixin.qq.com/miniprogram/dev/api/>



小程序运行

组件生命周期

- 不同的组件生命周期并不相同
- 路由切换以及更多操作导致的生命周期状态的变化
请参考官方文档：
<https://developers.weixin.qq.com/miniprogram/dev/framework/app-service/route.html>等
- 右图展示的是Page实例的生命周期情况。



小程序项目目录

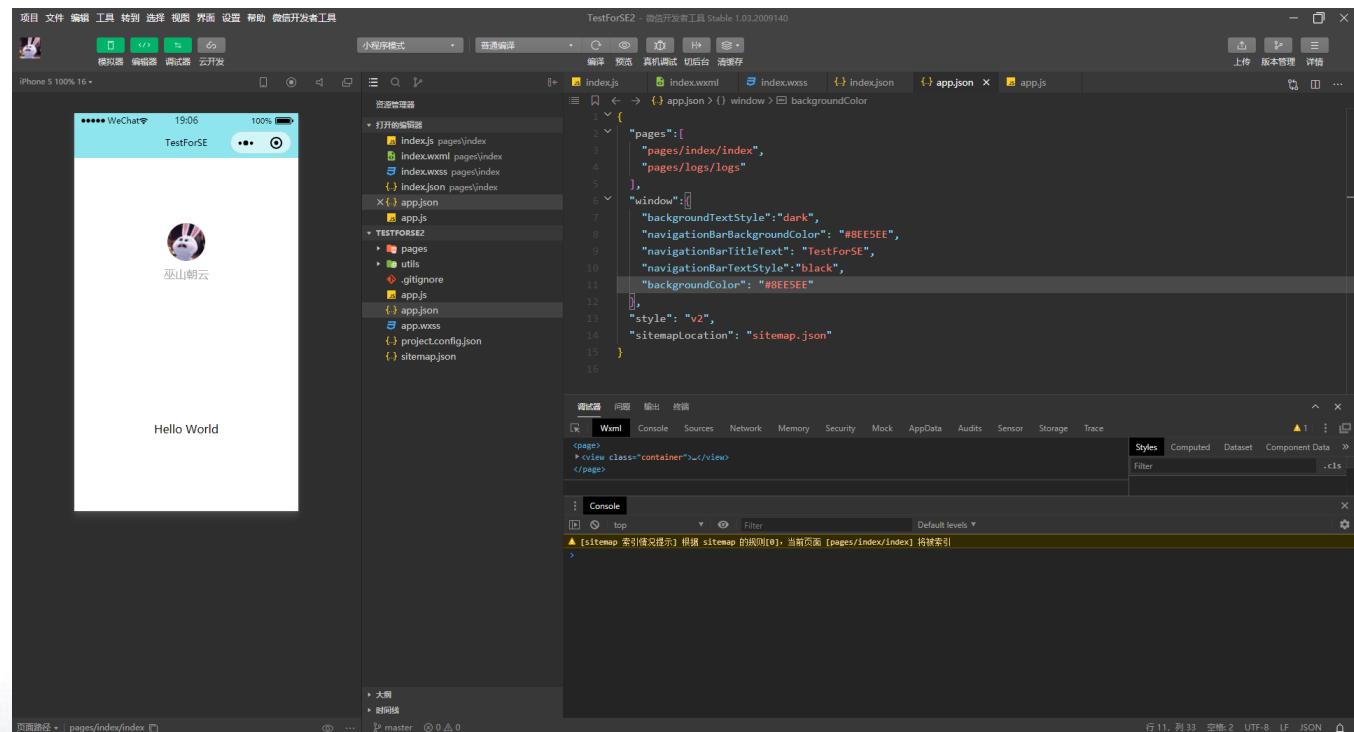
pages	—————	页面文件夹
index	—————	首页
index.js	—————	首页的逻辑文件
index.json	—————	首页的配置文件
index.wxml	—————	首页的配置文件
index.wxss	—————	首页的样式文件
logs	—————	日志 页面
logs.js		
logs.json		
logs.wxml		
logs.wxss		
utils	—————	第三方的工具 js (可以删除)
util.js		
app.js	—————	项目的全局 入口文件
app.json	—————	全局配置文件
app.wxss	—————	全局样式文件
project.config.json	—————	项目的配置文件 如appid
sitemap.json	—————	微信索引配置文件



小程序开发流程

□ 开发流程

- 注册账号，成为一名光荣的小程序开发者，获取AppId
- 下载并打开微信开发者工具
- 扫码登录，新建一个好项目
- 愉快地和小伙伴一起code
- 发布上线



□ 设计理念

- 提供完整的原生云端支持和微信服务支持
- 无需搭建服务器，弱化后端和运维概念
- 快速上线和迭代

□ 提供组件

- 数据库：可在小程序端和云端同时使用的json数据库
- 云存储：可在小程序端处理文件上传下载等任务
- 云函数：在云端运行的代码，微信私有协议天然鉴权
- 云调用：基于云函数免鉴权使用小程序开放接口的能力，包括服务端调用
- 云开发控制台：提供一个可视化的环境管理平台

□ 使用限制

- 新用户免费试用一个月后，统一使用“基础套餐+按量付费”模式

□ “有限”的框架

➤ 优点：

- 基于微信宿主环境，可以跨平台，可以利用微信已经封装好的一系列API
- 体量较小，启动快，易于开发、调试和迭代

➤ 缺点：

- WXML ≠ HTML
- WXSS ≠ CSS
- JS运行环境受限
- 包体积与性能限制

□ 更多信息

- <https://developers.weixin.qq.com/miniprogram/dev/framework/>
- <https://github.com/>
-



口微信小程序

➤ [官方文档](#)

- [awesome-wechat-weapp](#): 微信小程序开发资源：
组件、第三方库和示例软件
- [imageslr/weapp-library](#): 在线借书平台





移动开发 **ANDROID**



Peking
University



- 1. 开发工具 (IDE)
- 2. 开发语言
- 3. 项目结构
- 4. 其他



Peking
University



开发工具 Android Studio

- Android Studio是一个为Android平台开发程序的集成开发环境。2013年5月16日在Google I/O上发布，可供开发者免费使用。
- 基于JetBrains IntelliJ IDEA，为Android开发特殊定制，并在Windows、OS X和Linux平台上均可运行
- 当前最新版本为2025.1.3.7，已经较为稳定、成熟
- 下载和使用可以参考其官网：
<https://developer.android.google.cn/studio/>
- 官方学习资源：
<https://developer.android.google.cn/>
- 微信公众号：谷歌开发者





RoadCrack – LoginActivity.kt [RoadCrack.app.main]

File Edit View Navigate Code Refactor Build Run Tools Git Window Help

roadcrack-front-end > app > src > main > java > com > roadcrack > activity > LoginActivity > onCreate

Project

Android

Manifests

Java

com.roadcrack

activity

BaiduMapActivity

LoginActivity

NewRequestActivity

SelectionActivity

entity

exception.net

net

plain

repository

scaffold

ui

util

viewmodel

RoadCrackApplication

Live Edit of literals: ON

Code Split Design

Device Manager

Resource Manager

Commit

Profiler com.roadcrack (Xiaomi M2012K11AC)

SESSIONS + Dismiss Learn more

Timing data from debuggable processes will deviate significantly from real world performance. A profileable process may be more suitable.

上午9:30 roadcrack (Xiaomi M2012K11AC) 15 sec

CPU 100 %

MEMORY 128 MB

ENERGY Medium

activity.LoginActivity

N/A

96.7 MB

Light

Network Profiler has moved.

结构

Favorites

Build Variants

Git Run TODO Problems Terminal Build Logcat Profiler App Inspection

Launch succeeded (moments ago)

Event Log Layout Inspector

30:5 LF UTF-8 4 spaces develop

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    setContent {
        LoginScreen(
            onLoginButtonClick = onLoginButtonClick,
            onRegisterButtonClick = onRegisterButtonClick
        )
    }
}

private val onLoginButtonClick: (String, String) -> Unit = { username, password ->
    CoroutineScope(Dispatchers.IO).launch { this:CoroutineScope
        val result = LoginRepository.loginWithNetwork(username, password)
        if (!result) {
            lifecycleScope.launch { toast( msg: "登录失败, 请重试. " ) } // toast只能在UI线程上做
            return@launch
        }
        lifecycleScope.launch { toast( msg: "登录成功! " ) }
        SelectionActivity.navigateFrom( context: this@LoginActivity )
        finish()
    }
}

@OnDesign
private val onRegisterButtonClick: () -> Unit = { }
```



Peking
University

Why Kotlin

Modern,
concise and safe
programming language

Easy to pick up, so you can create powerful
applications immediately.

Get started →

Concise Safe Expressive Interoperable Multiplatform

```
data class Employee(  
    val name: String,  
    val email: String,  
    val company: String  
) // + automatically generated equals(), hashCode(), toString(), and copy()  
  
object MyCompany {  
    const val name: String = "MyCompany"  
}  
  
fun main() {  
    val employee = Employee("Alice",  
        "alice@mycompany.com", MyCompany.name)  
    println(employee)  
}
```

[Open in Playground →](#)

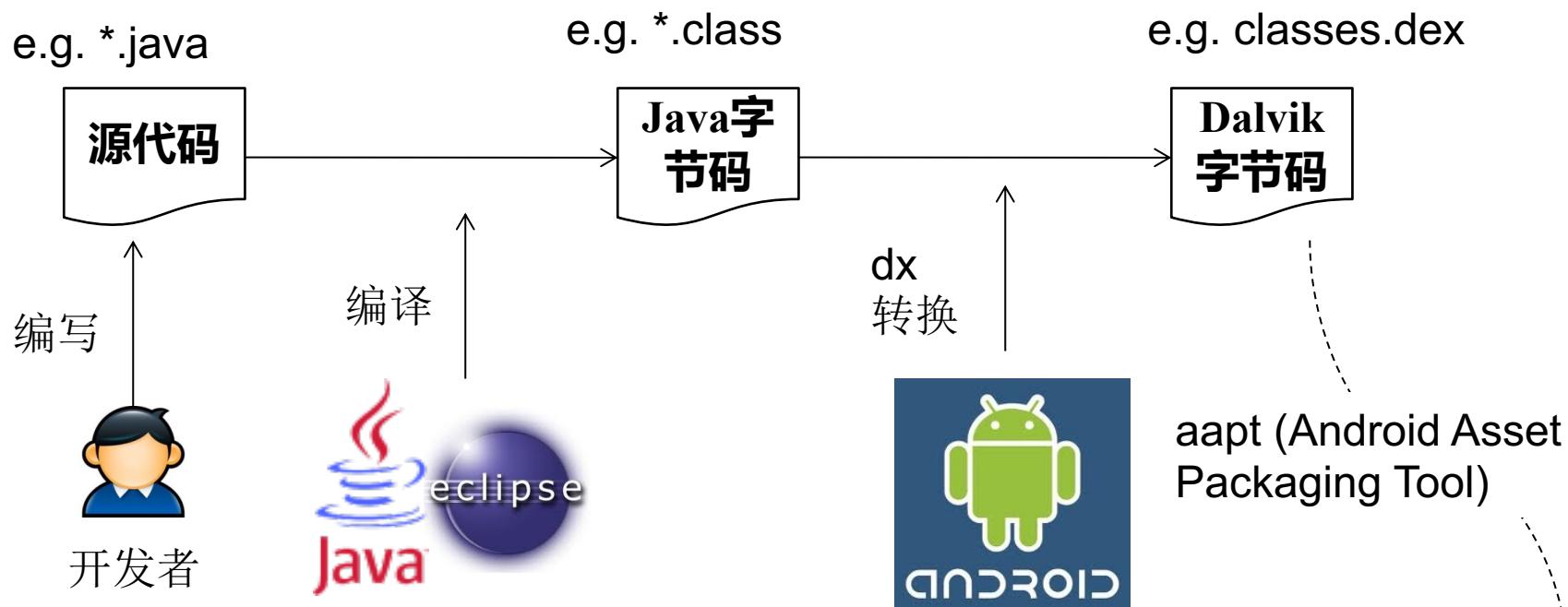
Target: JVM Running on v.1.7.10



Peking
University



开发语言-编译过程



名称	大小	压缩后大小	类型	修改时间
..			Folder	
assets			Folder	2011/6/14 4:02
lib			Folder	2011/6/14 4:08
META-INF			Folder	2011/6/14 4:08
res			Folder	2011/6/14 4:05
AndroidManifest.xml	4,180	1,371	XML Document	2011/6/14 4:08
classes.dex	475,104	206,465	文件 dex	2011/6/14 4:08
resources.arsc	51,108	11,180	文件 arsc	2011/6/14 4:08

Android应用程序组成——APK视图

Android应用(.apk)

AndroidManifest.xml

Activities

Views

Intents

Services

BroadcastReceivers

ContentProviders

代码组成(classes.dex)

主资源文件(resources.arsc)

Assets

lib

META-INF

res



名称	大小	压缩后大小	类型
..	4,180	1,371	Folder XML Docu
assets	475,104	206,465	Folder 文件 dex
lib	51,108	11,180	Folder 文件 arsc
META-INF			
res			
AndroidManifest.xml			
classes.dex			
resources.arsc			



Peking
University



Android应用程序组成——源码视图



案例：开发一个天气预报App: Sunny Weather

口功能需求：

➤ Sunny Weather应该具备的功能有：

- 可以搜索全球大多数国家的各个城市数据；
- 可以查看全球绝大多数城市的天气信息。



Peking
University



案例：开发一个天气预报App: Sunny Weather

□ 开发步骤：

- 注册彩云天气账号，获取天气API访问权限；
- 在Android Studio中创建项目，并在GitHub上托管；
- 设计并确定项目的架构；
- 添加相关依赖和图片等资源；
- 编写逻辑层代码；
- 实现UI层代码；
- 签名、生成APK文件。



Peking
University



案例：开发一个天气预报App: Sunny Weather

注册彩云天气账号，获取天气API访问权限

➤ 注册地址：<https://dashboard.caiyunapp.com/>

```
https://api.caiyunapp.com/v2/place?query=北京&token={token}&lang=zh_CN
```

```
https://api.caiyunapp.com/v2.5/{token}/116.4073963,39.9041999 realtime.json
```

```
{"status": "ok", "query": "北京",  
"places": [  
{"name": "北京市", "location": {"lat": 39.9041999, "lng": 116.4073963},  
"formatted_address": "中国北京市"},  
{"name": "北京西站", "location": {"lat": 39.89491, "lng": 116.322056},  
"formatted_address": "中国 北京市 丰台区 莲花池东路118号"},  
{"name": "北京南站", "location": {"lat": 39.865195, "lng": 116.378545},  
"formatted_address": "中国 北京市 丰台区 永外大街车站路12号"},  
{"name": "北京站(地铁站)", "location": {"lat": 39.904983, "lng": 116.427287},  
"formatted_address": "中国 北京市 东城区 2号线"}]
```

```
{  
    "status": "ok",  
    "result": {  
        "realtime": {  
            "temperature": 23.16,  
            "skycon": "WIND",  
            "air_quality": {  
                "aqi": { "chn": 17.0 }  
            }  
        }  
    }  
}
```

我的申请信息 3. 完成

彩云天气 API 彩云小译 API

SunnyWeather

如果您开发的应用中使用到开放平台的数据接口，且已在iOS/安卓应用市场上架，请提交应用所在商店地址（如果是web应用请提供网址，如果是pc端应用，请提供下载地址）

应用链接：

应用开发情况：

第一行代码中的实战项目开发

如果应用仍处于开发过程中，请在此栏位里说明，暂可不需要提供相关文件，待上线后再提交即可。

提交

 University

案例：开发一个天气预报App: Sunny Weather

在Android Studio中创建项目，并在GitHub上托管

Owner: guolindev / Repository name: SunnyWeather

Great repository names are short and memorable. Need inspiration? How about bookish-octo-pancake?

Description (optional):

Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer.

Add .gitignore: Android | Add a license: Apache License 2.0 | ⓘ

Create repository

创建github项目

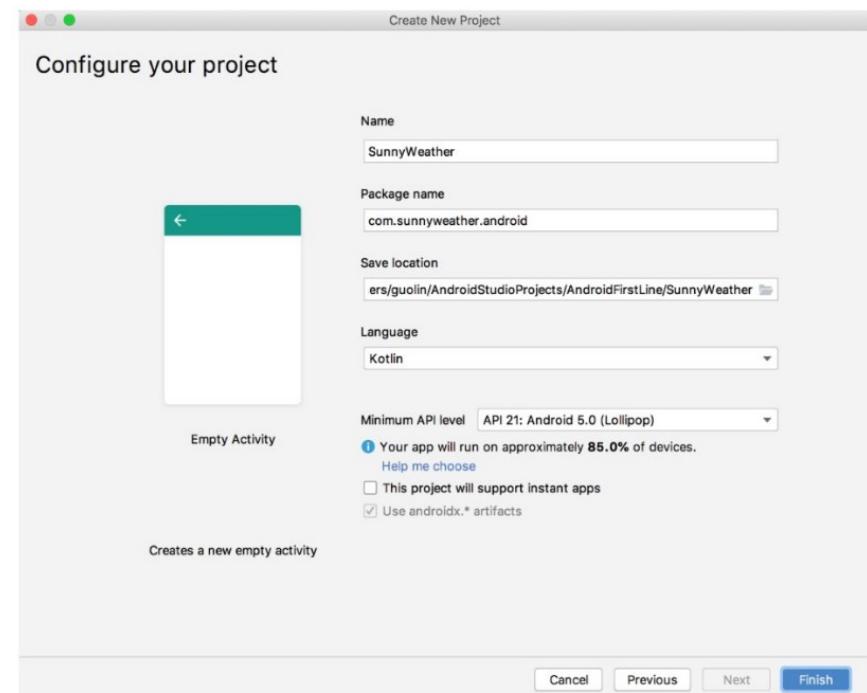
Create new file | Upload files | Find file | Clone or download

Clone with HTTPS ⓘ | Use SSH

Use Git or checkout with SVN using the web URL.
https://github.com/guolindev/SunnyWee

Open in Desktop | Download ZIP

复制项目链接



在Android Studio中创建项目

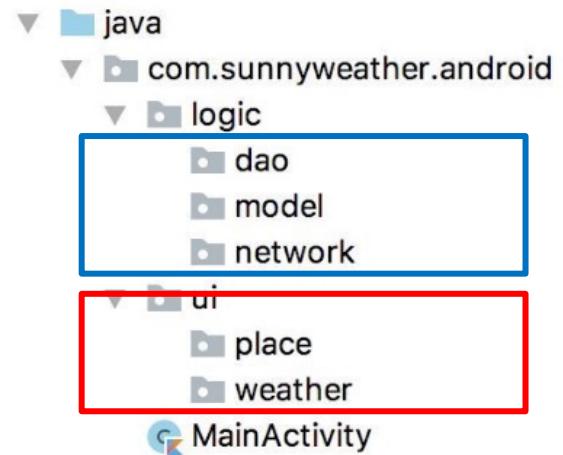
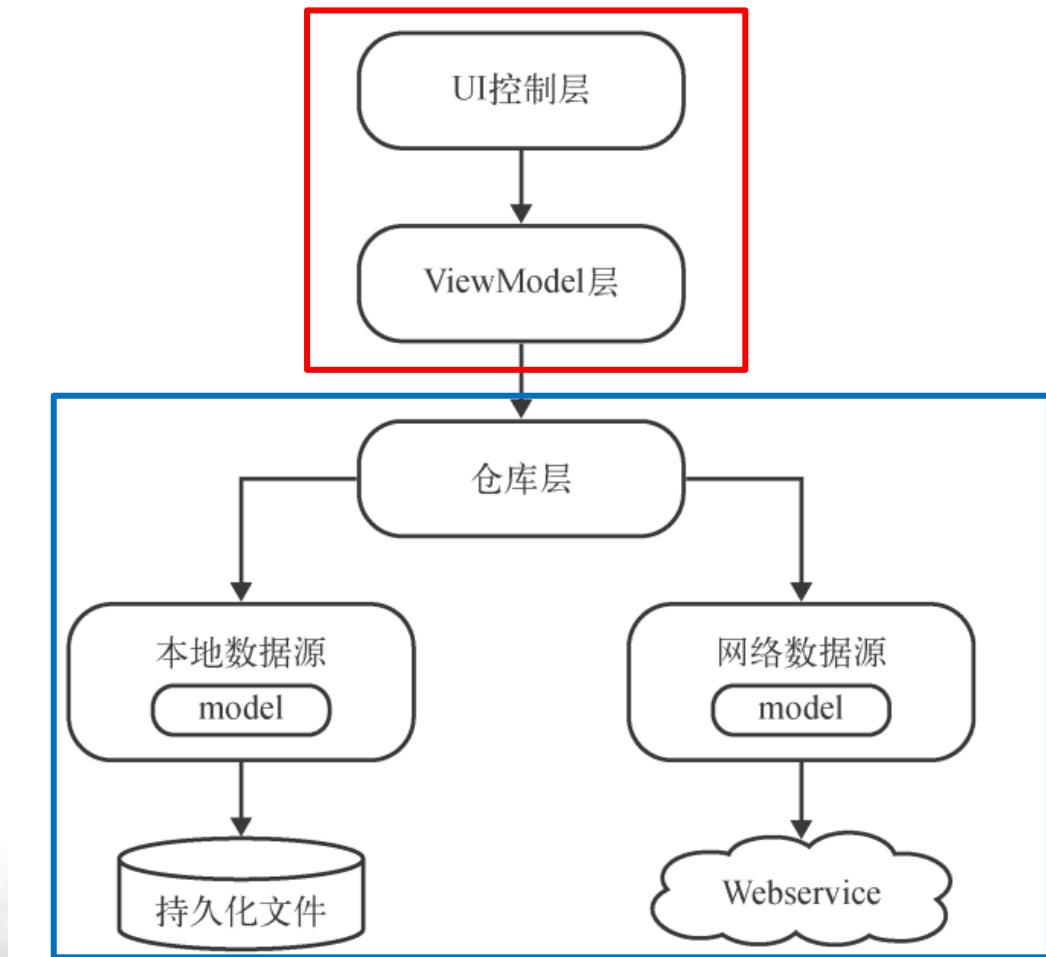
创建项目后执行：

```
git clone https://github.com/guolindev/SunnyWeather.git  
git commit -m "first commit"  
git push origin master (输入GitHub账号和Token)
```



案例：开发一个天气预报App: Sunny Weather

设计并确定项目的架构





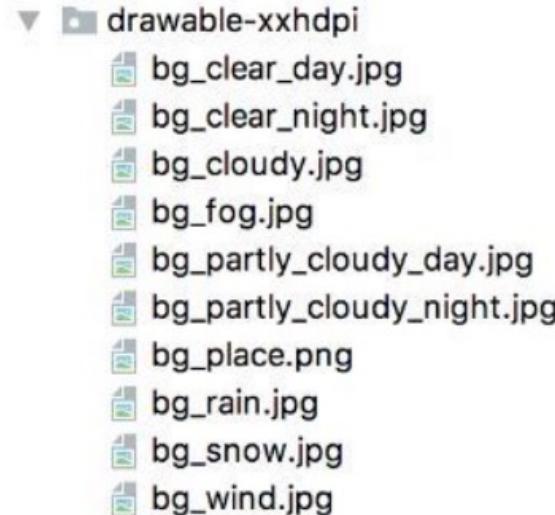
案例：开发一个天气预报App: Sunny Weather

添加相关依赖和图片等资源

```
dependencies {  
    ...  
    implementation 'androidx.recyclerview:recyclerview:1.0.0'  
    implementation "androidx.lifecycle:lifecycle-extensions:2.2.0"  
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.2.0"  
    implementation 'com.google.android.material:material:1.1.0'  
    implementation? "androidx.swiperefreshlayout:swiperefreshlayout:1.0.0"  
    implementation 'com.squareup.retrofit2:retrofit:2.6.1'  
    implementation 'com.squareup.retrofit2:converter-gson:2.6.1'  
    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.0"  
    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.1.1"  
}
```

↑ 在 **build.gradle** 中添加依赖项

在**res**中添加图片资源 →



案例：开发一个天气预报App: Sunny Weather

编写逻辑层代码

➤ 创建Application

- 在com.sunnyweather.android包下创建名为SunnyWeatherApplication的类（继承 android.app.Application），并在 AndroidManifest.xml 中指定

```
1 package com.sunnyweather.android
2
3 import android.annotation.SuppressLint
4 import android.app.Application
5 import android.content.Context
6
7 class SunnyWeatherApplication : Application() {
8
9     companion object {
10
11         const val TOKEN = "" // 填入你申请到的令牌值
12
13         @SuppressLint("StaticFieldLeak")
14         lateinit var context: Context
15     }
16
17     override fun onCreate() {
18         super.onCreate()
19         context = applicationContext
20     }
21 }
22 }
```

在 SunnyApplication 中添加 context 和 TOKEN 字段，以便APP整个生命周期均可引用

在 AndroidManifest.xml 中指定 Application
(第7行↓)

```
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3   <uses-permission android:name="android.permission.INTERNET" />
4
5   <application
6       android:name=".SunnyWeatherApplication"
7       android:allowBackup="true"
8       android:icon="@mipmap/ic_launcher"
9       android:label="@string/app_name"
10      android:roundIcon="@mipmap/ic_launcher_round"
11      android:supportsRtl="true"
12      android:theme="@style/AppTheme">
13
14     <activity android:name=".ui.weather.WeatherActivity"></activity>
15     <activity android:name=".MainActivity">
16         <intent-filter>
17             <action android:name="android.intent.action.MAIN" />
18
19             <category android:name="android.intent.category.LAUNCHER" />
20         </intent-filter>
21     </activity>
22
23 </application>
24
25 </manifest>
```

案例：开发一个天气预报App: Sunny Weather

编写逻辑层代码

➤ 定义API访问接口：使用Retrofit包

- 定义一个访问API的接口 (`logic.network.PlaceService`)
- 定义若干单例对象，用于创建服务和访问接口

```
1 package com.sunnyweather.android.logic.network
2
3     import com.sunnyweather.android.SunnyWeatherApplication
4     import com.sunnyweather.android.logic.model.PlaceResponse
5     import retrofit2.Call
6     import retrofit2.http.GET
7     import retrofit2.http.Query
8
9     interface PlaceService {
10
11         @GET("v2/place?token=${SunnyWeatherApplication.TOKEN}&lang=zh_CN")
12         fun searchPlaces(@Query("query") query: String): Call<PlaceResponse>
13
14     }
15
16     data class PlaceResponse(val status: String, val places: List<Place>)
```

使用Retrofit与API交互
整个接口定义了访问网页的服务；
圈出的函数定义了交互的方法：
调用方法时，访问注解中的API

```
object ServiceCreator {
    private const val BASE_URL = "https://api.caiyunapp.com/"

    private val retrofit = Retrofit.Builder()
        .baseUrl(BASE_URL)
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    fun <T> create(serviceClass: Class<T>): T = retrofit.create(serviceClass)

    inline fun <reified T> create(): T = create(T::class.java)
}
```

创建 retrofit-service 的工厂

```
10     object SunnyWeatherNetwork {
11
12         private val weatherService = ServiceCreator.create(WeatherService::class.java)
13
14         suspend fun getDailyWeather(lng: String, lat: String) = weatherService.getDailyWeather(lng, lat).await()
15
16         suspend fun getRealtimeWeather(lng: String, lat: String) = weatherService.getRealtimeWeather(lng, lat).await()
17
18         private val placeService = ServiceCreator.create(PlaceService::class.java)
19
20         suspend fun searchPlaces(query: String) = placeService.searchPlaces(query).await()
21
22     }
23
24     private suspend fun <T> Call<T>.await(): T { ... }
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

使用工厂创建service，包装成函数供上层调用

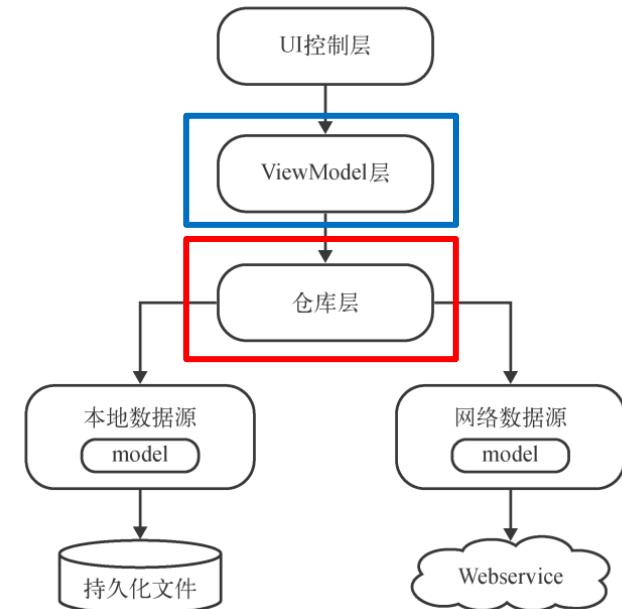
案例：开发一个天气预报App: Sunny Weather

编写逻辑层代码

- 定义Repository：用于封装统一入口
- 在 ViewModel 层使用接口 (ui.place.PlaceViewModel)

```
13  <object Repository {  
14  
15      fun searchPlaces(query: String) = fire(Dispatchers.IO) {  
16          val placeResponse = SunnyWeatherNetwork.searchPlaces(query)  
17          if (placeResponse.status == "ok") {  
18              val places = placeResponse.places  
19              Result.success(places)  
20          } else {  
21              Result.failure(RuntimeException("response status is ${placeResponse.status}"))  
22          }  
23      }  
24  }  
25  ...  
26 }  
27 }
```

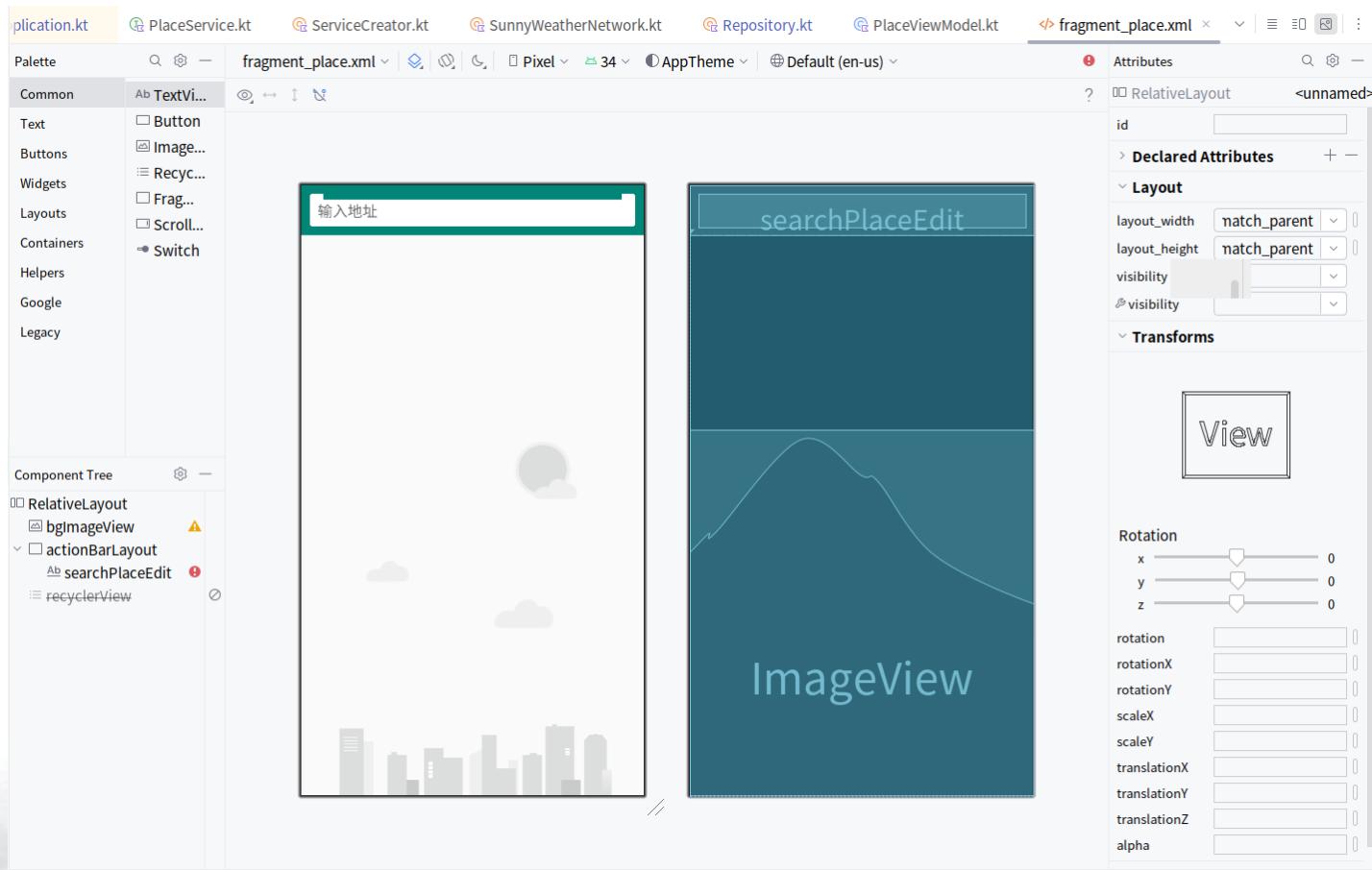
```
7  class PlaceViewModel : ViewModel() {  
8  
9      private val searchLiveData = MutableLiveData<String>()  
10  
11     val placeList = ArrayList<Place>()  
12  
13     val placeLiveData = Transformations.switchMap(searchLiveData) { query →  
14         Repository.searchPlaces(query)  
15     }  
16  
17     fun searchPlaces(query: String) {  
18         searchLiveData.value = query  
19     }  
20  
21     fun savePlace(place: Place) = Repository.savePlace(place)  
22  
23     fun getSavedPlace() = Repository.getSavedPlace()  
24  
25     fun isPlaceSaved() = Repository.isPlaceSaved()  
26 }  
27 }
```



案例：开发一个天气预报App: Sunny Weather

实现UI层代码

- 定义Fragment，并添加到Activity中
- 细节工作：和ViewModel的绑定、Fragment的Java/Kotlin实现等





案例：开发一个天气预报App: Sunny Weather

实现UI层代码

- 定义Fragment，并添加到Activity中
- 细节工作：和ViewModel的绑定、Fragment的Java/Kotlin实现等

```
1 package com.sunnyweather.android.ui.place
2
3 > import ...
18
19 </> class PlaceFragment : Fragment() {
20
21     val viewModel by lazy { ViewModelProviders.of(fragment: this).get(PlaceViewModel::class.java) }
22
23     private lateinit var adapter: PlaceAdapter
24
25 @+< override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {
26         return inflater.inflate(R.layout.fragment_place, container, attachToRoot: false)
27     }
28
29 @+> override fun onActivityCreated(savedInstanceState: Bundle?) [...]
30
31 }
32 }
```

```
<activity_weather.xml >
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.drawerlayout.widget.DrawerLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/drawerLayout"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.swiperefreshlayout.widget.SwipeRefreshLayout ... >
9
10     <FrameLayout
11         android:layout_width="match_parent"
12         android:layout_height="match_parent"
13         android:layout_gravity="start"
14         android:clickable="true"
15         android:focuseable="true"
16         android:background="@color/colorPrimary">
17
18         <fragment
19             android:id="@+id/placeFragment"
20             android:name="com.sunnyweather.android.ui.place.PlaceFragment"
21             android:layout_width="match_parent"
22             android:layout_height="match_parent"
23             android:layout_marginTop="25dp"/>
24
25     </FrameLayout>
26
27 </androidx.drawerlayout.widget.DrawerLayout>
```

↑ 和界面关联的 Fragment
(Fragment 是比 Activity 更细粒度管理界面的类)

← 在 activity 中添加 Fragment

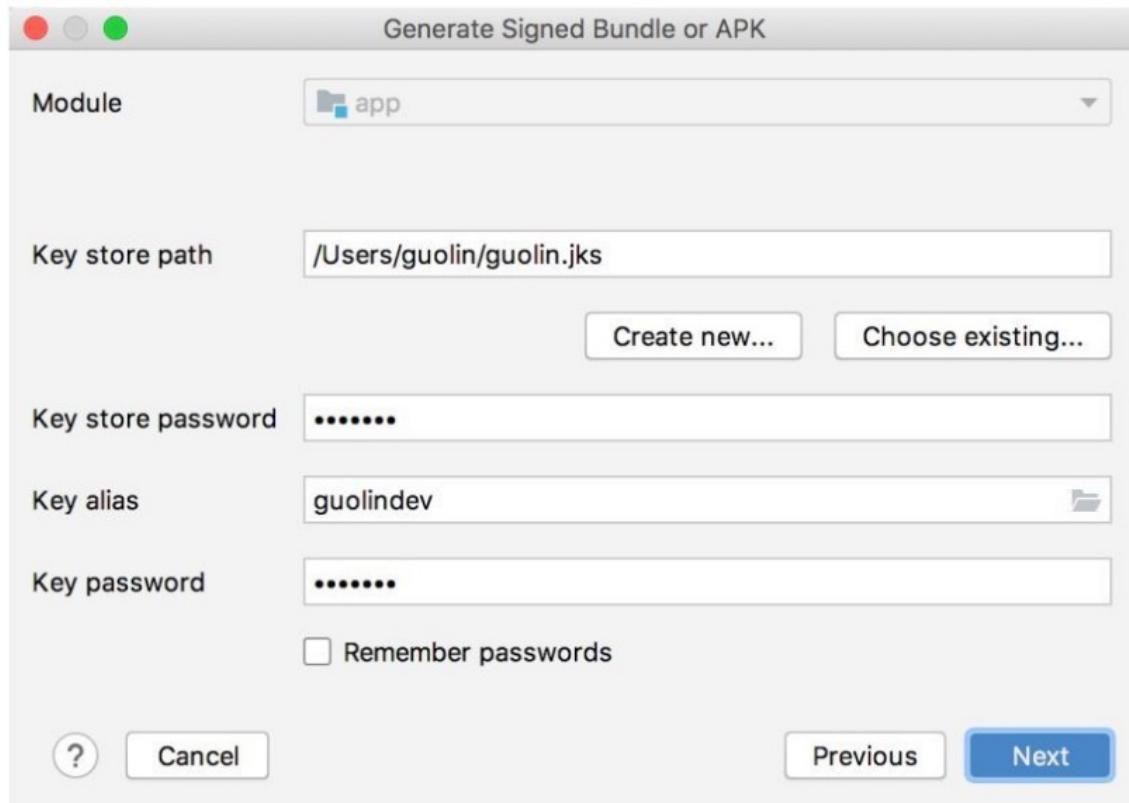


Peking
University

案例：开发一个天气预报App: Sunny Weather

签名、生成APK文件

➤ 使用Android Studio生成：Build→Generate Signed Bundle/APK



← 生成 keystore

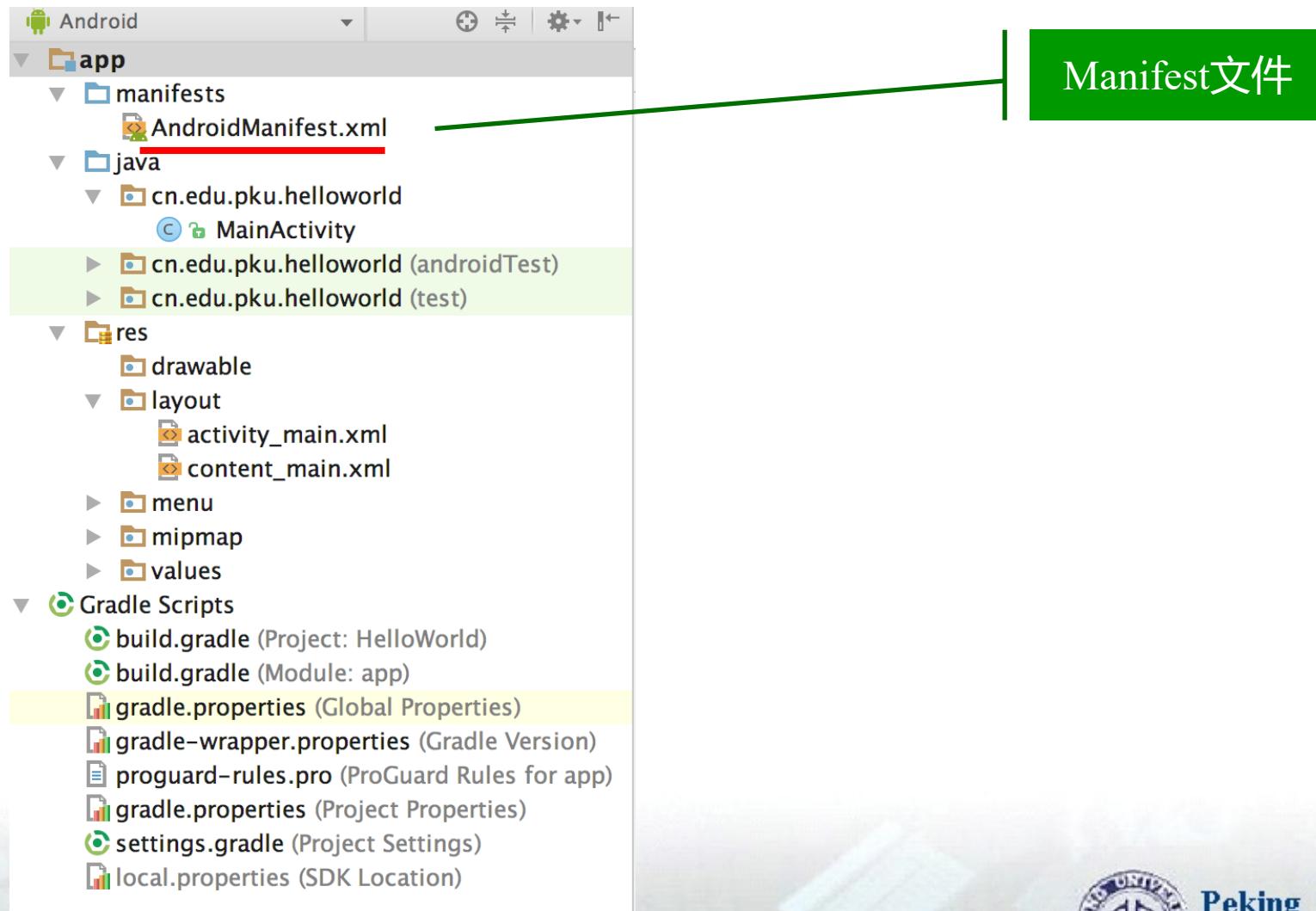
↓ 生成 apk

ⓘ Generate Signed APK
APK(s) generated successfully for module 'app'
with 1 build variant:
Build variant 'release': [locate or analyze](#) the APK.

名称	修改日期	大小	种类
app-release.apk	下午1:12	4.9 MB	文稿
output.json	下午1:12	234 字节	纯文本文稿



Android应用程序组成——源码视图



Peking
University

□ Android应用的部署描述符文件

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.edu.pku.helloworld">
    <uses-permission android:name="android.permission.INTERNET" />

    <application>
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
            <activity>
                <uandroid:name=".MainActivity"
                    android:label="HelloWorld"
                    android:theme="@style/AppTheme.NoActionBar">
                    <intent-filter>
                        <action android:name="android.intent.action.MAIN" />
                        <category android:name="android.intent.category.LAUNCHER" />
                    </intent-filter>
                </activity>
            </application>
    </manifest>
```

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-permission android:name="android.permission.SET_WALLPAPER"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.BATTERY_STATS"></uses-permission>
```

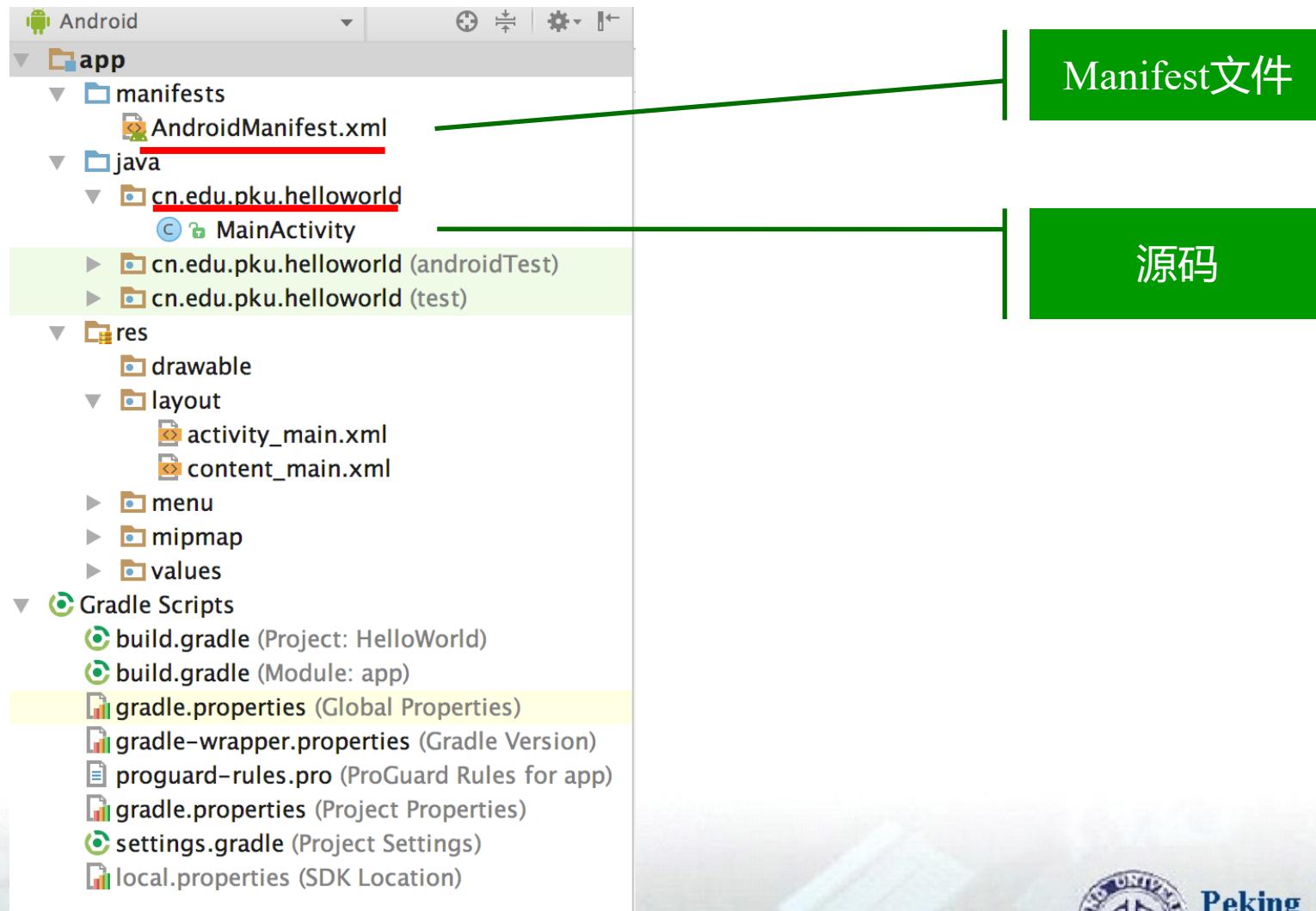
□权限举例 (参考`android.manifest.permission`)

权限名称	权限描述
接收短信	android.permission.RECEIVE_SMS
拨打电话	android.permission.CALL_PHONE
系统启动完毕通知	android.permission.RECEIVE_BOOT_COMPLETED
读取联系人信息	android.permission.READ_CONTACTS
修改联系人信息	android.permission.WRITE_CONTACTS

注意：Android 6.0后权限管理的变化(`targetSdkVersion >= 23`)



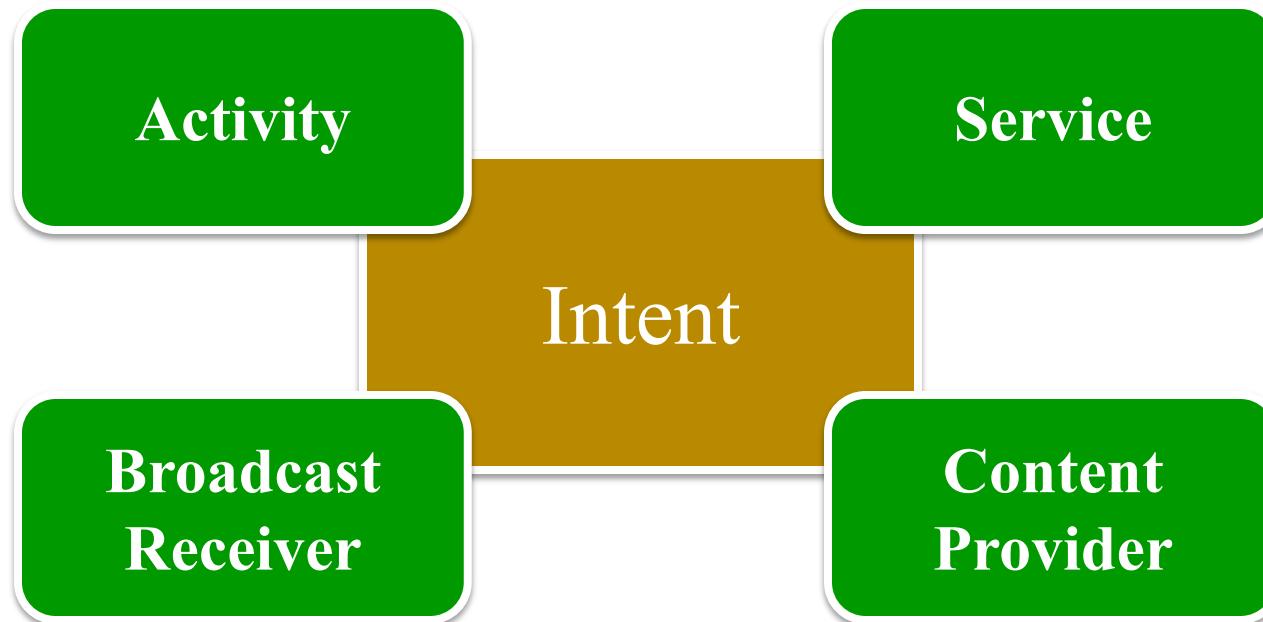
Android应用程序组成——源码视图



Peking
University

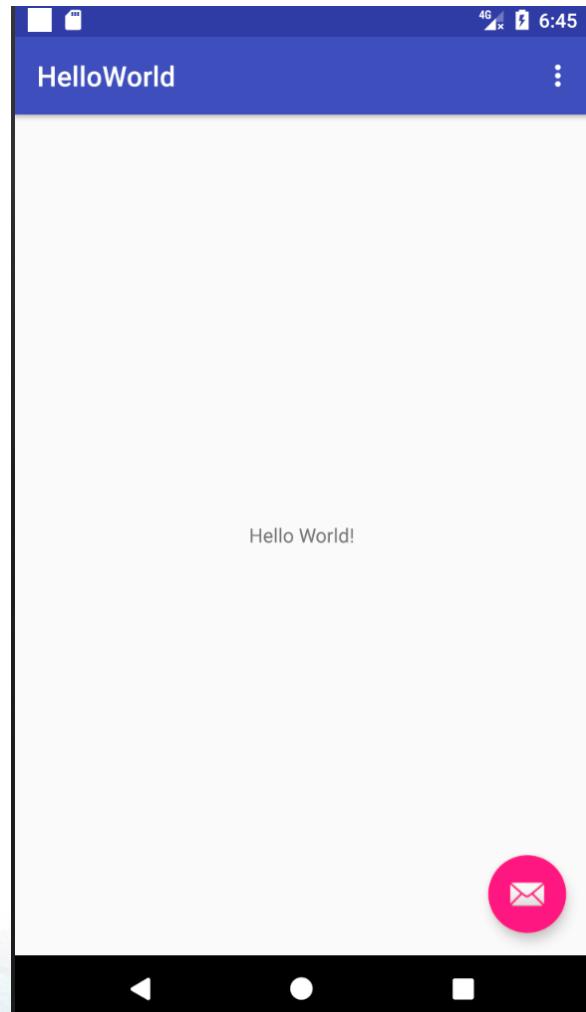


组成Android应用程序的基本构件



Activity(活动)

- 最基本的应用程序构件
- 应用程序中，一个Activity通常可理解为一个单独的屏幕
- Activity 主要作用是对用户在屏幕上的UI界面的输入进行处理并做出响应（更新UI、启动其他Activity等）
- 继承android.app.Activity 类，并覆盖相应的方法
 - onCreate
 - ...



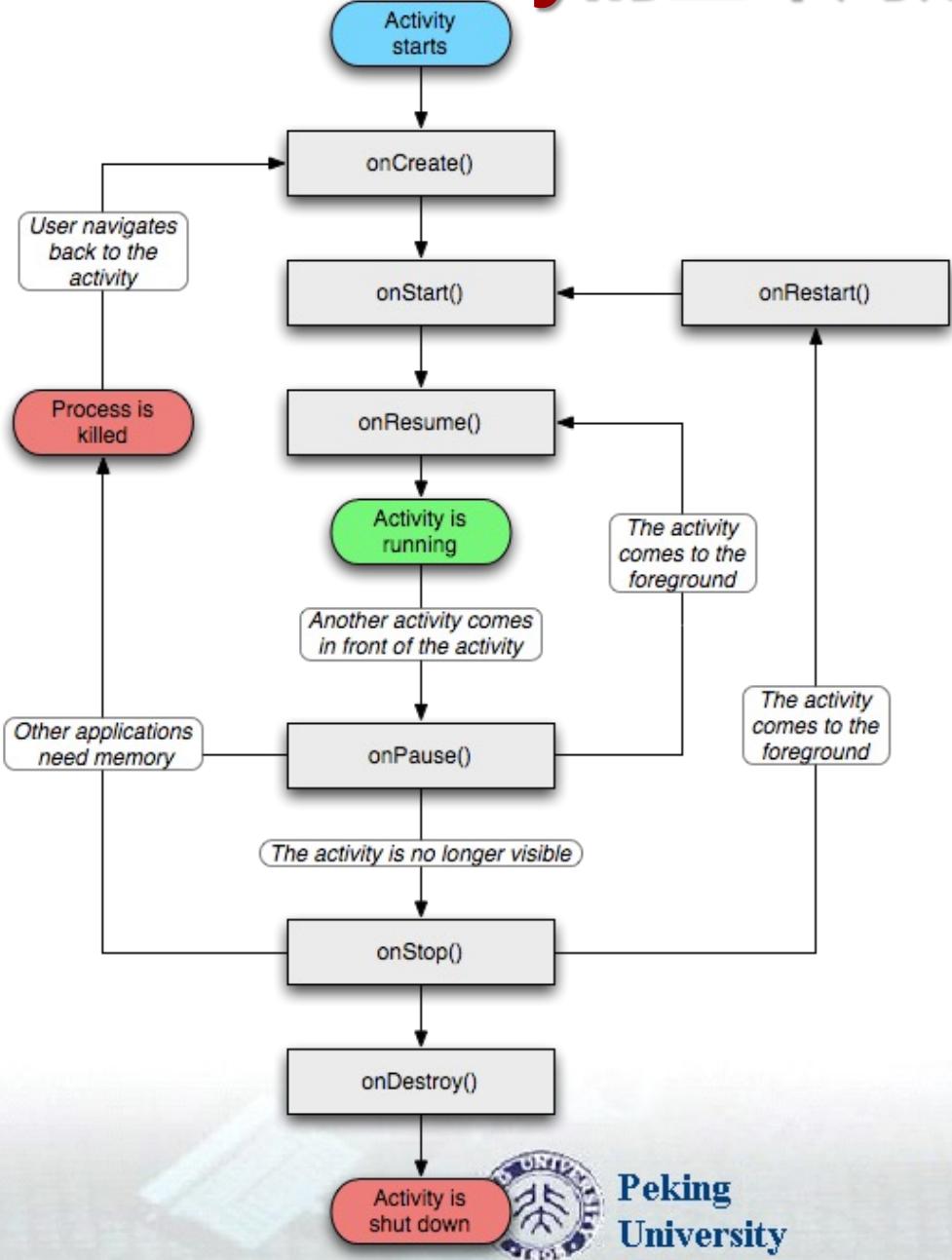


Activity的代码结构与配置文档

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  
        setSupportActionBar(toolbar);  
  
        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);  
        fab.setOnClickListener(view -> {  
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)  
                .setAction("Action", null).show();  
        });  
    }  
  
<activity  
    android:name=".MainActivity"  
    android:label="HelloWorld"  
    android:theme="@style/AppTheme.NoActionBar">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN" />  
        <category android:name="android.intent.category.LAUNCHER" />  
    </intent-filter>  
</activity>
```

Activity的生命周期

- onCreate**
- onStart**
- onResume**
- onPause**
- onStop**
- onDestroy**



□ TextView extends View

- `setText(String text)`
- `getText()`

□ EditText extends View

- `getText()`

□ ImageView extends View

□ Button extends View

□ View

- `setOnClickListener(View.OnClickListener listener)`
- `setVisibility(int visibility)`



Service (服务)

□什么样的逻辑程序需要实现为Service?

- 执行长时间的计算任务
- 且不与用户直接进行界面交互

□Service是做“地下工作的”



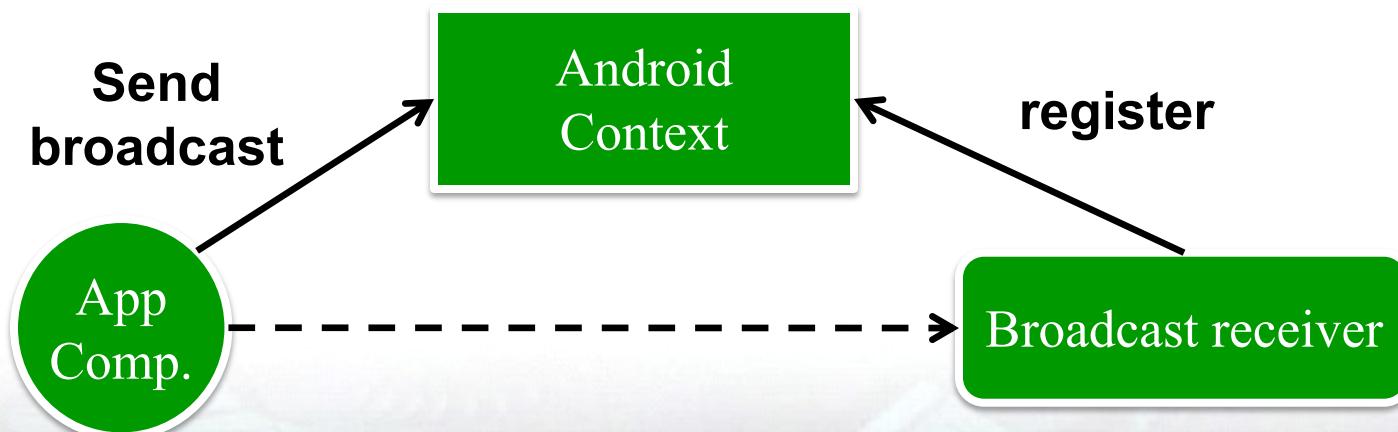


Broadcast Receiver (广播接收器)

□ 接收和处理Android的广播消息

□ Android的广播机制

- Receiver (Observer)向系统注册
- 构件广播Intent
 - Activity/Service/Broadcast Receiver/ContentProvider
- 系统触发相应的Receiver来使其接收Intent

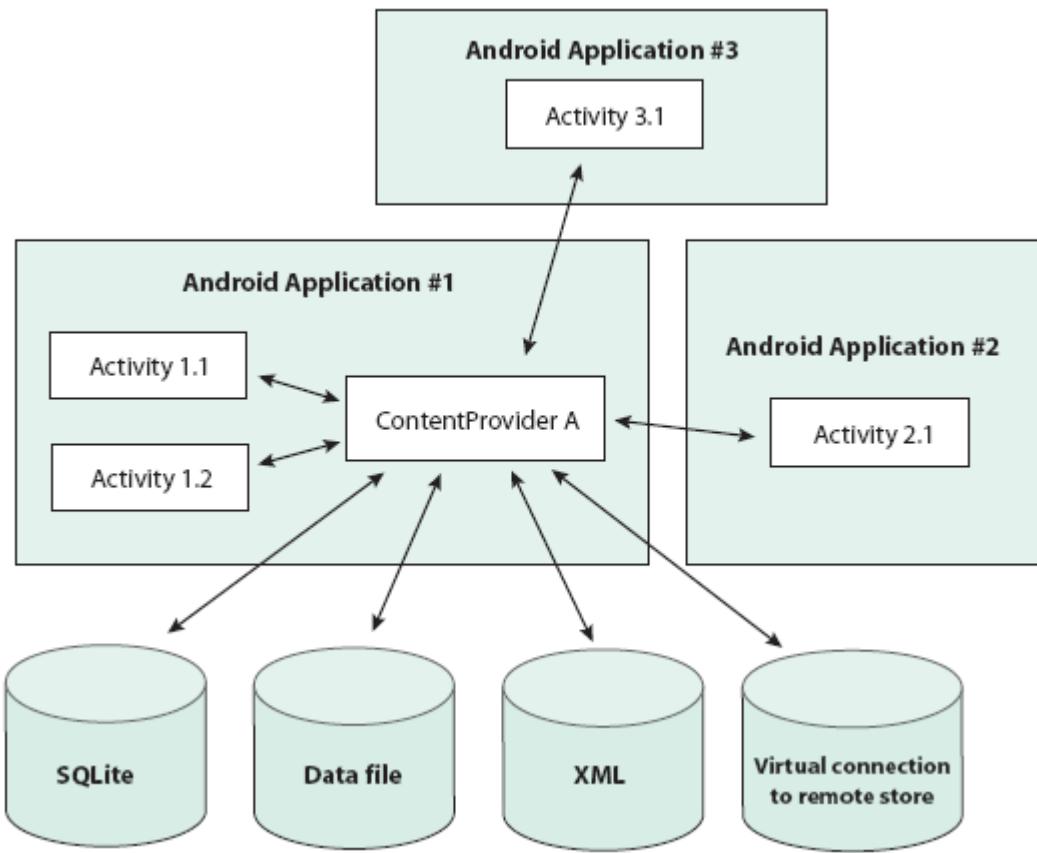




Content Provider (内容供应器)

- Android中的跨应用的数据访问机制
- 为何需要content provider?

- Android中每一个app的资源是私有的
- app通过content provider实现和其他apps的私有数据共享
 - Preference
 - Contact
 - ...

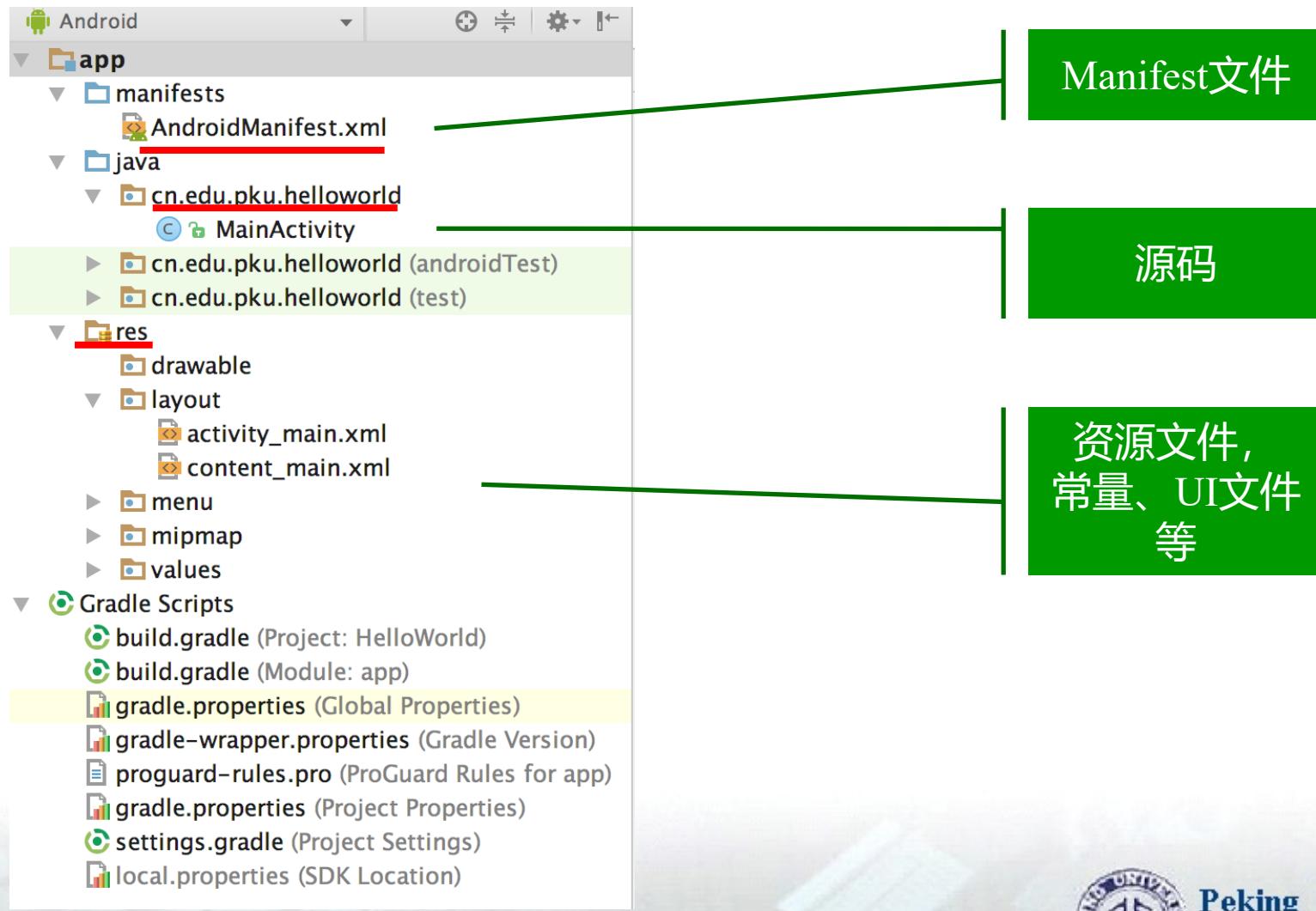


- 可理解为带请求协助意味的消息或事件
- 关联Activity、Service、BroadcastReceiver、ContentProvider的桥梁

```
public class LoginActivity extends AppCompatActivity {  
  
    private int mUserId;  
  
    @Override  
    protected void onCreate(@Nullable Bundle savedInstanceState) {...}  
  
    private void gotoMain() {  
        Intent intent = new Intent(LoginActivity.this, MainActivity.class);  
        intent.putExtra("id", mUserId);  
        startActivity(intent);  
    }  
}
```



Android应用程序组成——源码视图



Peking
University

□ 资源文件

□ 一般有 anim、assets、drawable、layout、menu、mipmap、values (包括 colors、dimens、strings、styles)

□ 通过自动生成的 R 类访问

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  
    setSupportActionBar(toolbar);  
  
    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);  
    fab.setOnClickListener((view) -> {  
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)  
            .setAction("Action", null).show();  
    });  
}
```

□ 或者在别的 xml 文件中直接引用

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="@color/blue_1">  
  
    <LinearLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:orientation="vertical"  
        android:padding="16dp">
```

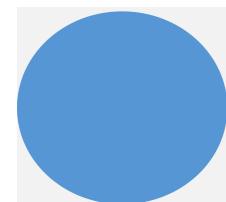
```
<color name="blue_6">#2C82C9</color>  
<color name="blue_5">#7BBBD6</color>  
<color name="blue_4">#A4D4E8</color>  
<color name="blue_3">#C3EAF1</color>  
<color name="blue_2">#E1F2F9</color>  
<color name="blue_1">#F8F9FB</color>
```

colors.xml

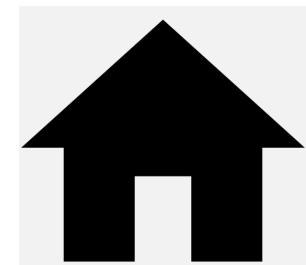


□ 使用 xml 的方式绘图

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="@color/google_blue" />
</shape>
```



```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <path
        android:fillColor="#FF000000"
        android:pathData="M10,20v-6h4v6h5v-8h3L12,3 2,12
    </vector>
```



Layout

The screenshot shows the Android Studio interface with the project 'HelloWorld' open. The left sidebar displays the project structure, including the app module with its manifest, Java files (MainActivity), and various resource folders like res/layout and res/values. The main workspace shows the XML code for 'activity_main.xml' and its corresponding preview.

activity_main.xml (Text View)

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="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="cn.edu.pku.helloworld.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

        </android.support.design.widget.AppBarLayout>

        <include layout="@layout/content_main" />

        <android.support.design.widget.FloatingActionButton
            android:id="@+id/fab"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom|end"
            android:layout_margin="16dp"
            app:srcCompat="@android:drawable/ic_dialog_email" />

    </android.support.design.widget.CoordinatorLayout>
```

Preview (Design View)

The preview window shows a smartphone screen with a blue header bar labeled '7:00'. The main content area displays the text 'Hello World!' in white. At the bottom, there is a black navigation bar with three white icons: a triangle (left), a circle (center), and a square (right). A small red circular icon with a white envelope symbol is positioned in the bottom right corner of the content area.

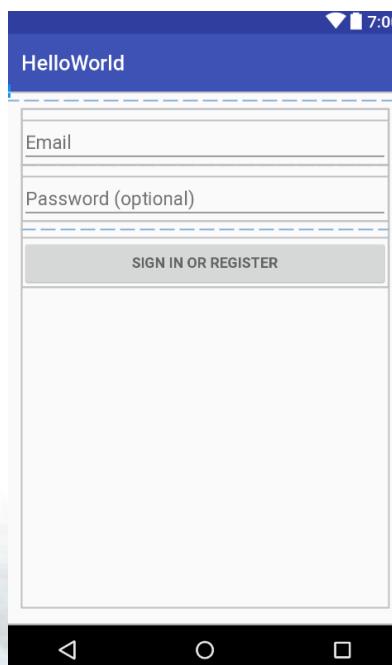
Bottom Navigation Bar

At the very bottom of the screen, there is a navigation bar with several icons: Messages (blue), Terminal (grey), Android Monitor (green), Problems (red), Run (yellow), and TODO (orange).



□ 设置元素属性

- Application
- Activity
- Views

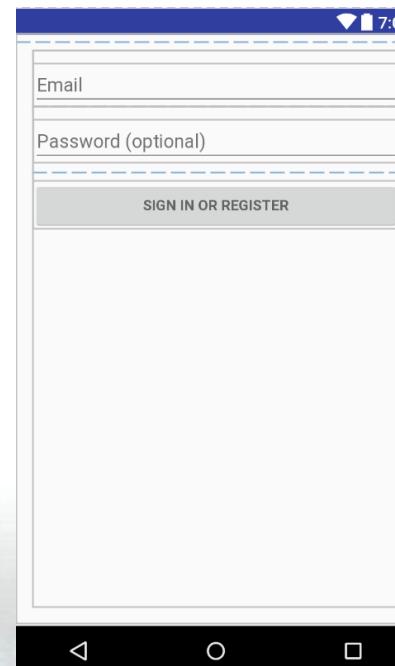


```
<activity  
    android:name=".activity.TagInfoActivity"  
    android:theme="@style/AppTheme.NoActionBar" />
```

Manifest.xml

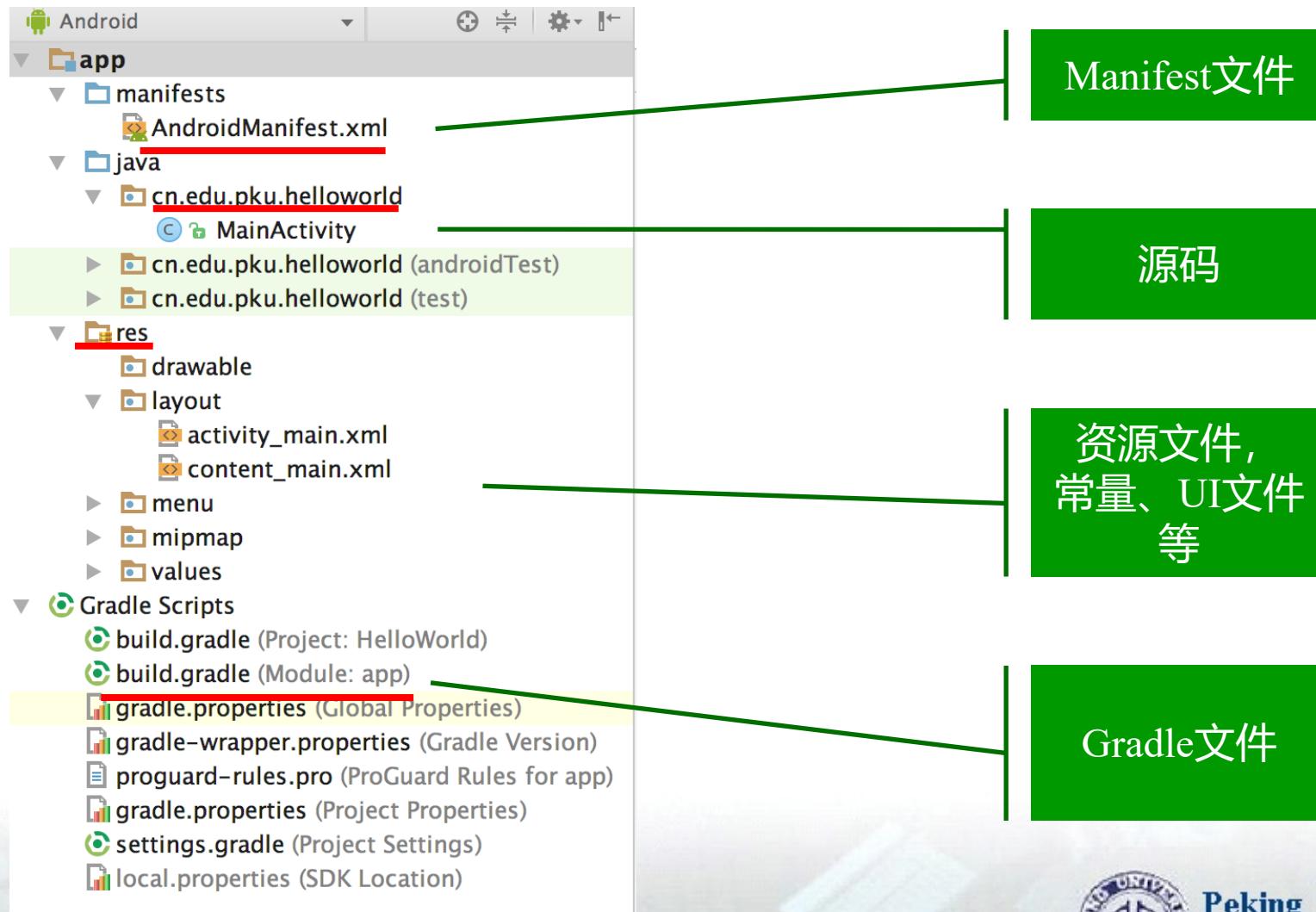
```
<style name="AppTheme.NoActionBar">  
    <item name="windowActionBar">false</item>  
    <item name="windowNoTitle">true</item>  
</style>
```

styles.xml





Android应用程序组成——源码视图



- 对你的项目进行编译、运行、签名、打包、依赖管理等一系列功能的合集
- 如设定 编译 sdk 版本、设置运行时 sdk 最低/最高的版本号、设置应用版本号、设置签名文件、管理依赖关系等
- 一般首选 Gradle 导入第三方依赖库，也支持 Maven

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {  
        exclude group: 'com.android.support', module: 'support-annotations'  
    })  
    compile 'com.android.support:appcompat-v7:25.3.1'  
    compile 'com.android.support.constraint:constraint-layout:1.0.1'  
    compile 'com.android.support:design:25.3.1'  
    testCompile 'junit:junit:4.12'  
}
```

- 网络请求: retrofit、volley、okhttp
- 序列化: Gson、Moshi、Jackson
- 图片载入: picasso、fresco、Glide
- 依赖注入: butterknife、dagger、hilt
- 事件总线: EventBus
- 响应式编程: LiveData, RxJava, RxAndroid,
Koroutine
- Socket: SocketIO
- <https://github.com/wasabeef/awesome-android-libraries>

□ 推送

- 小米、极光、友盟、腾讯信鸽、LeanCloud、GCM等

□ IM

- 环信、极光、融云、LeanCloud、网易等

□ 地图

- 百度、高德、腾讯等

□ 异常上报、应用统计

- 腾讯Bugly、友盟、LeanCloud等

□ 对象存储（图片等）

- 阿里云OSS、七牛云等

□



□ Google开发的Java序列化与反序列化工具。即将Json字符串转换为Java对象，或将Java对象转换为Json字符串。

String To Java Object

Java Object To String

```
class User {  
    private String id;  
    private String name;  
  
    public String getId() {  
        return id;  
    }  
  
    public void setId(String id) {  
        this.id = id;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}  
  
String json = "{ \"id\": \"user-id\", \"name\": \"root\" }";  
User user = new Gson().fromJson(json, User.class);  
  
User user = new User();  
String json = new Gson().toJson(user);
```



Peking
University

□ 配置

```
public static Retrofit getRetrofit() {  
    if (retrofit == null)  
        retrofit = new Retrofit.Builder()  
            // 设置域名  
            .baseUrl(mBaseUrl)  
            // 配合json使用  
            .addConverterFactory(GsonConverterFactory.create())  
            .build();  
    return retrofit;  
}
```

□ 定义接口

```
public interface GitHubService {  
    @GET("users/{user}/repos")  
    Call<List<Repo>> listRepos(@Path("user") String user);  
}
```

□ 直接调用

```
GitHubService service = getRetrofit().create(GitHubService.class);  
  
Call<List<Repo>> call = service.listRepos("username", 0);  
call.enqueue(new Callback<List<Repo>>() {  
    @Override  
    public void onResponse(Call<List<Repo>> call, Response<List<Repo>> response) {  
        List<Repo> repoList = response.body();  
        // do something  
    }  
    @Override  
    public void onFailure(Call<List<Repo>> call, Throwable t) {  
        t.printStackTrace(); // 错误处理  
    }  
});
```



Retrofit2+Koroutine

```
1 package com.roadcrack.net.meta
2
3 import ...
4
5 object MyRetrofit { ... }
6
7
8     package com.roadcrack.net
9
10    import ...
11
12    interface RoadService {
13
14        @POST("/road/id")
15        suspend fun fetchRoadId(
16            @Query("districtCode") districtCode: Long,
17            @Query("roadName") roadName: String,
18            @Body /*("districtInfo")*/ districtInfo: RequestBody
19        ): WebResult<Long>
20
21
22        companion object {
23            operator fun invoke() = MyRetrofit<RoadService>()
24        }
25    }
26
27
28    suspend fun retrieveRoadId(loc: BDLocation, roadName: String): Long {
29        val districtCode: Long = loc.adCode.toLong()
30        val formBody = FormBody.create(
31            okhttp3.MediaType.parse("application/json; charset=utf-8"),
32            // 目前只需要这3个信息
33            mapOf( ... ).toJson()
34        )
35
36        return roadSrv.fetchRoadId(districtCode, roadName, formBody).validIt()
37            .also { BriefInfo.roadId.value = it }
38    }
39
40 }
```



□ A powerful image downloading and caching library for Android

```
Picasso.with(getApplicationContext())
    .load(url)
    .placeholder(R.drawable.error_image)
    .error(R.drawable.error_image)
    .into(imageView);
```

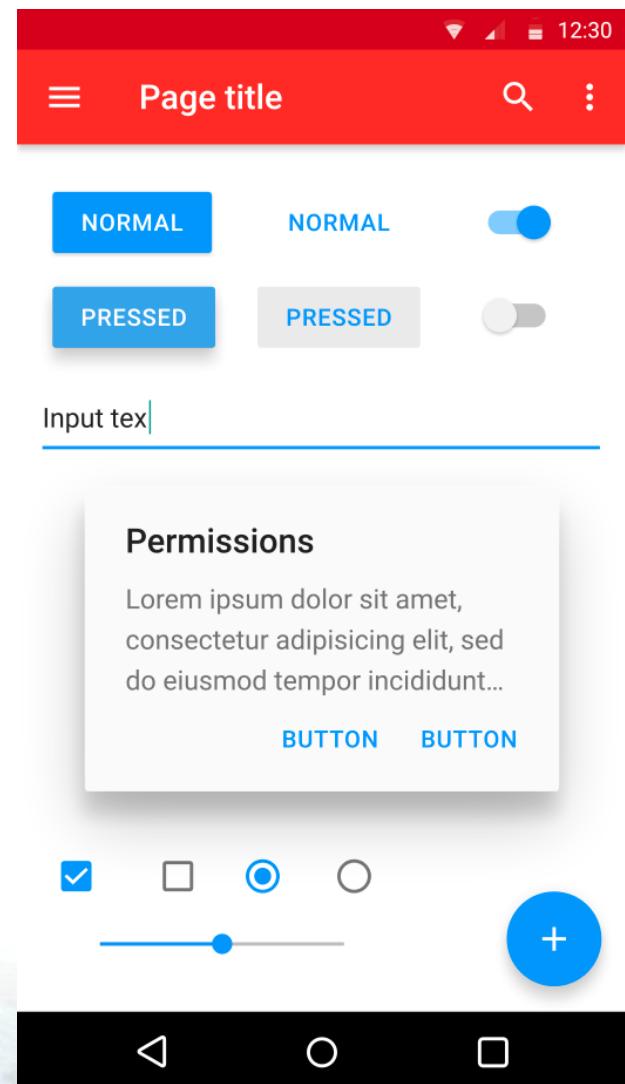


界面设计：Material Design

□ Material Design 是由 Google 开发的设计语言。扩展于 Google 即时的“卡片”设计，其基于网格的布局、响应动画与过渡、填充、深度效果（如光线和阴影）。

□ 可参考官网的介绍：

□ <https://developer.android.google.cn/design/material/>



□ JUnit

- 单元测试

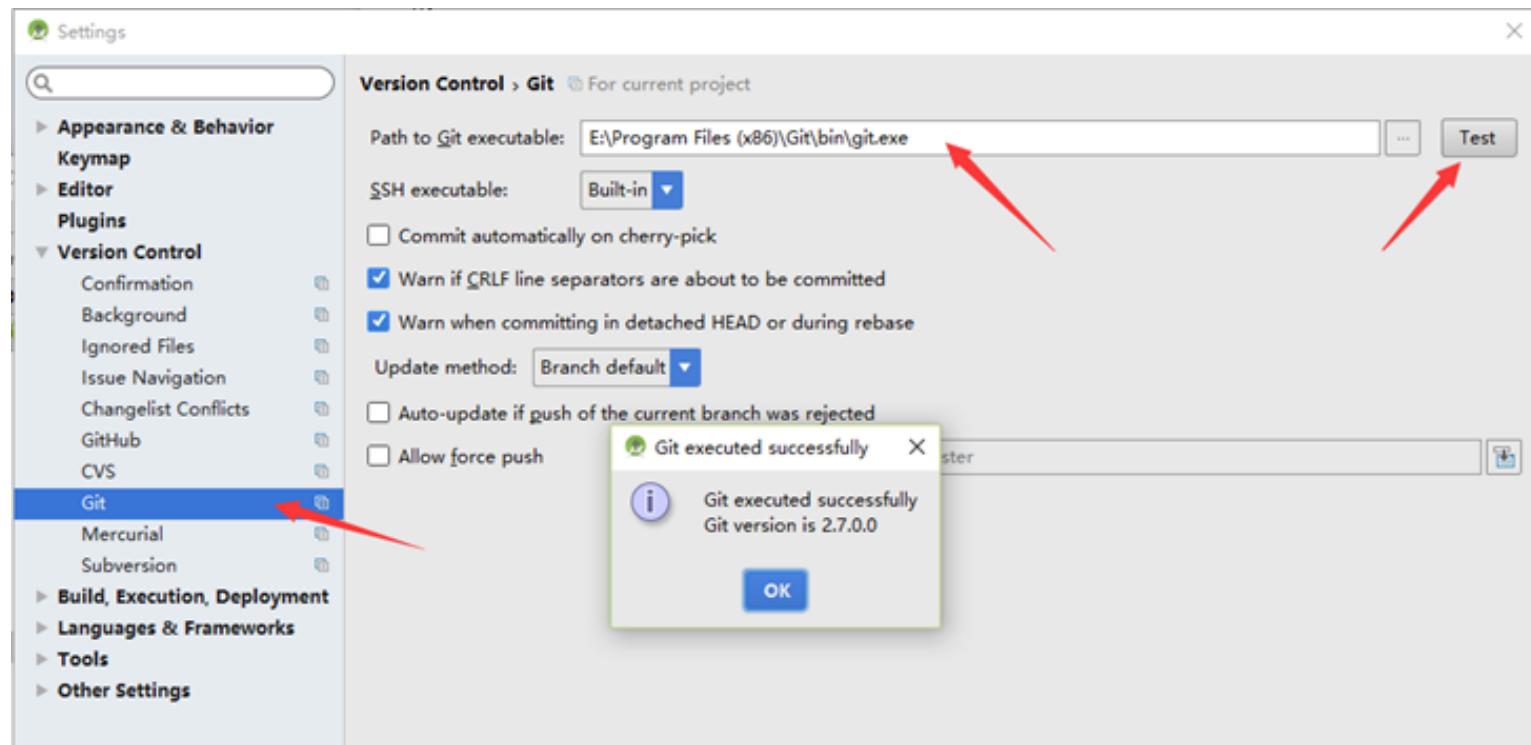
□ Espresso

- UI测试
- AS新建的项目中自动集成了Espresso
- <https://developer.android.com/training/testing/espresso/index.html>

□ Monkey Test

- 随机UI测试
- 脚本录制回放
- 鲁棒性测试





Android Studio中可以方便地进行Git操作

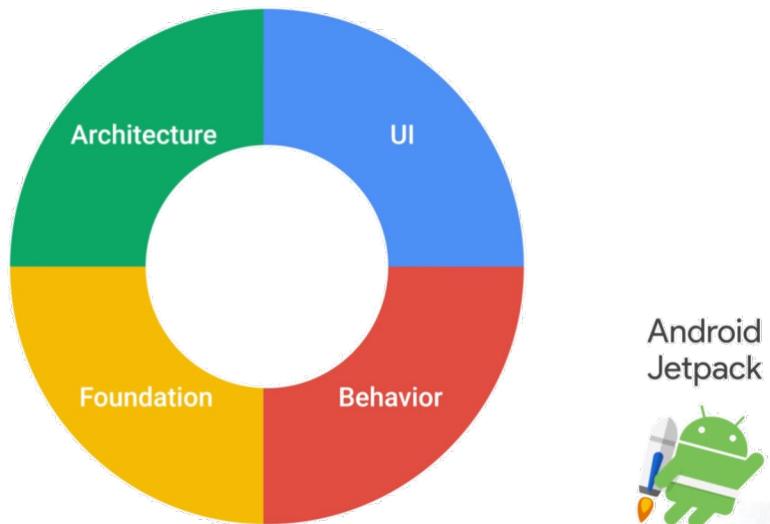


Android 新技术

库	说明
activity *	访问基于 Activity 构建的可组合 API。
appcompat *	允许在平台的旧版 API 上访问新 API（很多使用 Material Design）。
camera *	构建移动相机应用。
compose *	使用描述界面形状和数据依赖项的可组合函数，以编程方式定义界面。
databinding *	使用声明性格式将布局中的界面组件绑定到应用中的数据源。
fragment *	将您的应用细分为在一个 Activity 中托管的多个独立屏幕。
hilt *	扩展了 Dagger Hilt 的功能，以实现 androidx 库中某些类的依赖项注入。
lifecycle *	构建生命周期感知型组件，这些组件可以根据 Activity 或 Fragment 的当前生命周期状态调整行为。
Material Design 组件*	适用于 Android 的模块化、可自定义 Material Design 界面组件。
navigation *	构建和组织应用内界面，处理深层链接以及在屏幕之间导航。
paging *	在页面中加载数据，并在 RecyclerView 中呈现。
room *	创建、存储和管理由 SQLite 数据库支持的持久性数据。

□ Jetpack

□ **Jetpack是一个由多个库组成的套件，可帮助开发者遵循最佳做法、减少样板代码并编写可在各种 Android 版本和设备中一致运行的代码，让开发者可将精力集中于真正重要的编码工作。**



<https://developer.android.google.cn/jetpack?hl=zh-cn>



Android-基于Web开发

```
<WebView  
    android:id="@+id/wv_webview"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

```
public class WebViewActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_web_view);  
        //获得控件  
        WebView webView = (WebView) findViewById(R.id.wv_webview);  
        //访问网页  
        webView.loadUrl("http://www.baidu.com");  
        //系统默认会通过手机浏览器打开网页，为了能够直接通过WebView显示网页，则必须设置  
        webView.setWebViewClient(new WebViewClient(){  
            @Override  
            public boolean shouldOverrideUrlLoading(WebView view, String url) {  
                //使用WebView加载显示url  
                view.loadUrl(url);  
                //返回true  
                return true;  
            }  
        });  
    }  
}
```

WebViewDemo



百度一下

文库 图片 知道 新闻 百科
应用 地图 贴吧 hao123 更多

小说 游戏 下载

下载： 百度客户端 百度应用 地图

Baidu 京ICP证030173号



Peking
University

□ Android

➤ kotlin

- guolindev/SunnyWeather: 这个例子的最初来源是《第一行代码-第三版》，作者郭霖。这是一个天气预报的例子，非常小巧但涉及到的知识点很多，比如如何写界面（传统方法）、如何做网络交互、如何管理多个界面（Activity）和应用的状态。作者本人没有将代码放到github上，所以有一些人读完书后将自己实践的效果放了上来，同时用了一些google的新技术，作者在写书的时候这些技术还没出来。代码质量优秀。编译结果可运行
- android/Sunflower: 这个例子直接来自google。上一个例子中写界面用的是传统的XML，但安卓团队最近建议大家使用更加贴近底层的 jetpack-compose，直接用Kotlin的代码写界面；此外，还展示了如何使用 material design。这是一个展示各种向日葵种类的APP



□ Android

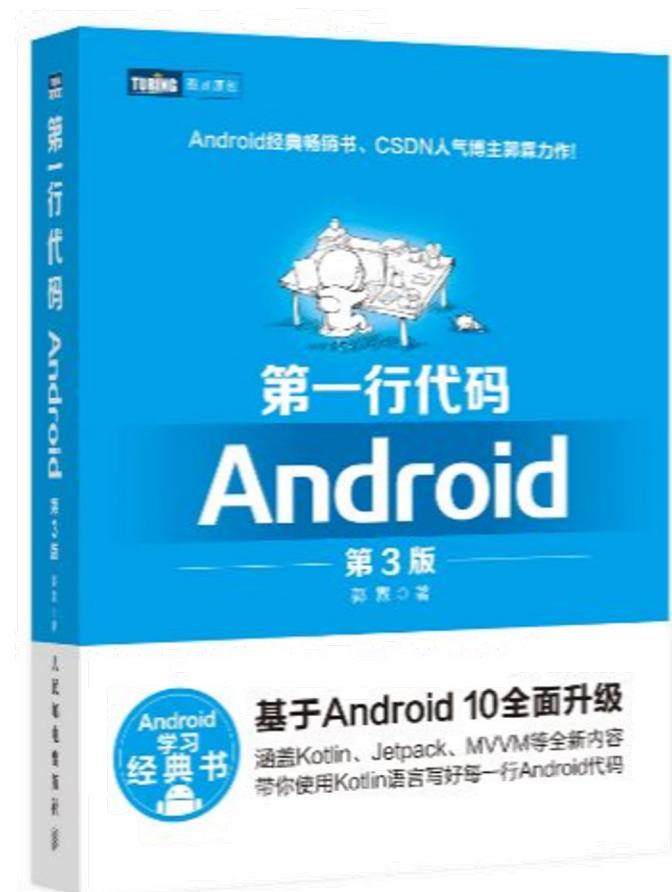
➤ java

- cachecats/LikeMeiTuan: 这是一个仿美团的开源项目，展示了如何使用Java做原生开发。除了网络请求的封装，该有的功能都有了。作者还给出了详细的开发流程。现在开发Android不建议使用Java，Android Studio对纯Java的项目支持也不是很好了（主要和Gradle版本有关）
- Yalantis/uCrop: 这个例子展示了如何使用 Java 编写一个简单的裁剪图片的APP，其中包含了一些Java使用原生共享库的例子。这个例子可能不适合用来给学生作为编写一个独立应用的全面的例子，但可以作为一个很好的补充
- 其他例子: 可以找到其他Android官方的例子——这些都不是完整的项目，是为开发中的概念编写的小例子（包括Kotlin）

推荐书目

口第一行代码（第三版）作者：郭霖

基于Kotlin语言，针对Android10，讲解了近些年开发的Android的主要技术和最新变化，以及Android项目的组织架构的最佳实践。



Peking
University