每次不重样，带你收获最新测试技术！

微信扫一扫关注我们

投稿邮箱：editor@51testing.com

EasyOCR 在 UI 自动化测试中的应用

◆ 作者：Tynam

EasyOCR 是一个基于深度学习的开源 OCR（光学字符识别）库，由 Jaided AI 开发。EasyOCR 的整体架构基于 PyTorch 实现，它采用 CRAFT 算法进行文字检测，结合 CRNN 模型进行文字识别。EasyOCR 的核心特性和功能有支持超过 80 种语言、高精度识别、GPU 加速、易于安装和使用、灵活的输入格式、自定义输出、开源免费。EasyOCR 的这些特性使其成为一个功能强大且易于使用的 OCR 工具，适用于文档处理、自动化测试、图像分析和无障碍阅读等多种应用场景。

EasyOCR 在 UI 测试中的应用场景

在 UI 自动化测试中，EasyOCR 作为文本内容断言、元素属性断言等的补充，以提供更全面的测试覆盖。EasyOCR 在 UI 自动化测试中的应用主要体现在以下几个方面：

1.界面元素验证：在 UI 自动化测试中，EasyOCR 可以用来识别和验证界面上的文本元素是否符合预期。通过截图获取应用界面的图像，然后使用 EasyOCR 进行 OCR 识别，提取出图像中的文字，并与预期的文字进行比较，从而自动化地验证界面显示是否正确。

2.提高测试效率：传统的 UI 测试可能需要手动比对界面上的文本，这既耗时又容易出错。EasyOCR 通过自动化的 OCR 识别，可以快速准确地识别图像中的文字，大大提高了测试的效率和准确性。

3.处理复杂场景：EasyOCR 基于深度学习模型，即使在面对模糊、倾斜、遮挡等复杂场景时，也能保持较高的识别准确率，这使得它在自动化测试中非常实用。

4.文本检测功能：除了文本识别外，EasyOCR 还具备文本检测功能，能够定位图像中的文本框，并返回文本框的坐标信息，这对于后续的处理如文本提取、排版分析等非

常有用。

5.易用性：EasyOCR 提供了简洁易用的 API，开发者可以轻松地将其集成到自己的自动化测试项目中。同时，它也支持多种图像格式，如 JPEG、PNG 等，这使得在不同测试场景下都能灵活使用。

6.多语言支持：EasyOCR 支持超过 80 种语言的文字识别，这使得它在国际化的 UI 自动化测试中非常有用，可以处理多种语言的界面文本。

通过上述功能，EasyOCR 成为了 UI 自动化测试中一个强大的工具，帮助测试人员提高测试的自动化水平和效率。下面我们就来体验 EasyOCR 的魅力。

EasyOCR 安装

你可以通过 Python 的包管理器 pip 来安装 EasyOCR：

```
pip install easyocr
```

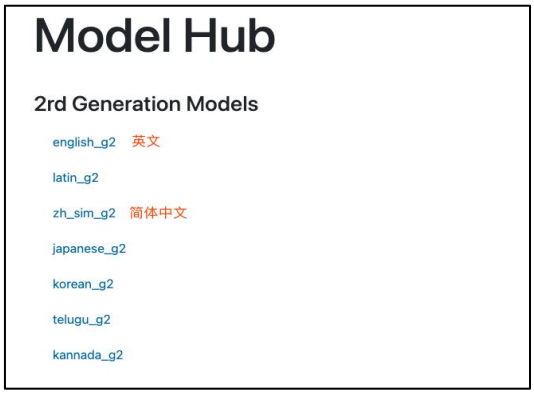
接下来是下载文本检测模型和语言模型。如果模型不存在，EasyOCR 会在第一次运行时自动下载文本检测模型和所需的语言模型。

不过由于某些原因，自动下载可能会失败，所以需要大家提前下载所需的语言模型。

手动下载的方式是进入语言模型下载中心：https://www.jaided.ai/easyocr/modelhub/

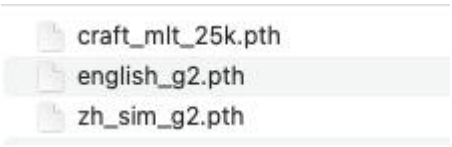
找到文本检测模型并进行下载：

然后是下载识别模型，不同的语言有对应的识别模型，如下图标注出了英文和简体中文的识别模型：



也可在百度网盘进行下载:
https://pan.baidu.com/s/1NTnpmzud2gNol1gnFPrLgw?pwd=redx 提取码: redx。

下载后会得到对应的 zip 包，然后解压。之后便会得到对应的 PTH 文件，如下图所示。craft_mlt_25k.pth 为文本检测模型文件；english_g2.pth 为英文的识别模型文件；zh_sim_g2.pth 为简体中文的识别模型文件。



将得到的 PTH 文件被放置在 EasyOCR 指定的模型目录下，通常是在用户的主目录下的.EasyOCR/model 文件夹中。例如，在 Windows 系统中，路径可能是 C:\Users\<当前电脑用户>\.EasyOCR\model。

EasyOCR 使用

下面看一个简单的示例。我们来识别一张百度首页的图片，如下图所示。

EasyOCR 使用起来很简单，先实例化一个 Reader 对象，然后使用 readtext()方法读取图片，最后将结果输出。

```
import easyocr

# 创建一个 EasyOCR 的 Reader 对象
reader = easyocr.Reader(['en', 'ch_sim'], gpu=False)

# 读取一张图片并识别文本
result = reader.readtext('baidu.png')

# 打印识别结果
for (bbox, text, prob) in result:
    print(f"Detected {text} with probability {prob}")
```

如果一切正常，EasyOCR 将输出图片中的文本及其置信度。

运行上面代码后，结果输出如下：

```
Using CPU. Note: This module is much faster with a GPU.
Detected Baidw 百度 with probability 0.061702461704590796
Detected 百度-下 with probability 0.9840897917747498
```

从输出结果中可以看到：识别出了两部分文字，符合预期。其中 LOGO 内容识别值为“Baidw 百度”， u 识别成了 w，有误差，其置信度是 0.06；按钮文字识别值为“百度-下”，其置信度是 0.98。可以看到，置信度越高，准确性也就越高。

了解了简单的使用，下面介绍创建 Reader()实例时常用的参数及 readtext()方法文字识别后返回的数据。

创建 Reader()实例时常用的参数有：

• lang_list (list)：EasyOCR 在分析图像时应该识别哪些语言的文字，例如，['en']代表英语，['ch_sim']代表简体中文。

• gpu (bool)：用于启用或禁用 GPU 支持，默认值是 True。

•model_storage_directory (string)：指定模型数据的存储路径。如果未指定，EasyOCR 会按照环境变量 EASYOCR_MODULE_PATH 指定的路径、环境变量 MODULE_PATH 指定的路径、默认路径~/.EasyOCR/（当前用户的主目录下的.EasyOCR 文件夹）顺序依次查找模型。

• user_network_directory (string)：指定自定义网络的存储路径。如果未指定，EasyOCR 会按照环境变量 EASYOCR_MODULE_PATH 指定的路径、环境变量 MODULE_PATH 指定的路径、默认路径~/.EasyOCR/（当前用户的主目录下的.EasyOCR 文件夹）顺序依次查找模型。

• download_enabled (bool)：用于启用或禁用通过 HTTP 下载模型数据的功能。默认值为 True，如果没有本地模型文件，EasyOCR 会自动尝试从网络上下载所需的模型数据。

• detector (bool) ：是否加载检测模型，默认值是 True。 recognizer (bool) ：是否加载识别模型，默认值是 True。

readtext()返回的数据：

1、默认输出是一个列表，其中每个元素代表图像中检测到的一段文本及其相关信息。依次是边界框、文本和置信度。边界框是一个包含四个整数的列表，表示检测到的文本区域在图像中的坐标位置。坐标通常是[x_min, y_min, x_max, y_max]的形式，其中(x_min, y_min)是文本区域左上角的坐标，(x_max, y_max)是文本区域右下角的坐标；文本是识别出的文本字符串；置信度是一个介于 0 和 1 之间的浮点数，表示模型对识别结果的置信程度。值越接近 1，表示置信度越高。

```
result = reader.readtext('baidu.png')
print(result)
# 输出结果
# [([[236, 64], [456, 64], [456, 112], [236, 112]], 'Baidw 百度', 0.061702461704590796), ([[582, 144], [656, 144], [656, 170], [582, 170]], '百度-下', 0.9840897917747498)]
```

2、添加 detail=0 参数，可只输出识别出来的文本列表。

```
result = reader.readtext('baidu.png', detail=0)
print(result)
# 输出结果
# ['Baidw 百度', '百度-下']
```

3、输出 json 或 字典格式数据。添加 output_format= 'dict' 或 output_format= 'json' 参数，可以以 json 或 字典格式将结果输出。

```
result = reader.readtext('baidu.png', output_format='json')
print(result)
# 输出结果
# ['{"boxes": [[236, 64], [456, 64], [456, 112], [236, 112]], "text": "Baidw 百度", "confident": 0.061702461704590796}', '{"boxes": [[582, 144], [656, 144], [656, 170], [582, 170]], "text": "百度-下", "confident": 0.9840897917747498}']
```

借助 EasyOCR 实现断言

掌握了 EasyOCR 的使用后，下面将其应用到 UI 自动化测试中，通过一个示例说明 EasyOCR 帮助我们实现图片文字提取并断言。

先实现两个 read_img_text()和 assert_img_contain_text()函数。read_img_text()用于从图像中提取文本内容并将结果返回。assert_img_contain_text()用于图片文本断言，如果给定的预期文本存在于图片中，且 EasyOCR 提取的文本置信度大于一个设定值，则返回 True，意味着断言成功，除此之外的任何情况则抛出 AssertionError，意味着断言失败，用例不通过。在此添加置信度判断，可以提高断言的准确性。

代码实现也很简单，如下所示：

```
import easyocr
def read_img_text(img) -> list:
    """从图像中提取文本"""
    reader = easyocr.Reader(['en', 'ch_sim'], gpu=False)
    result = reader.readtext(img, output_format='dict')
    return result
def assert_img_contain_text(img, text, confident=0.6):
    """断言图片中是否含有指定文本"""
    img_result = read_img_text(img)
    print(img_result)
    for element in img_result:
        if text in element.get("text"):
            if element.get("confident") >= confident:
```

```
                return True
            else:
                raise AssertionError(f"置信度为 {round(element.get('confident'))}，低于基准置信度 {confident}")
        raise AssertionError(f"元素中未识别出预期内容或预期内容在图片中不存在")
```

完成了断言图片是否含有指定文本的函数，接下来就是将其应用到自动化测试用例。

假如有一条用例：访问 51testing 主页（http://www.51testing.com/），检查轮播图的第一张图片展示正确，如下图所示。从图片中可以看到有文本“学软件测试”、“选 51Testing”、“二十年育才历程”、“全国十八大校区就近学”和“立即咨询”。

下面使用 playwright 实现。创建一条名为 test_first_slide_img 的测试用例函数，函数中先启动浏览器；然后访问 51testing 主页；接着定位轮播图并获取第一张图片元素；然后获取元素的 src 属性，即图片链接；有了图片链接实际上就是获取了图片，将图片和预期文本传入 assert_img_contain_text()函数进行断言；最后关闭浏览器。

此处断言使用了两次，分别断言了图片中含有“二十年育才历程”和“立即咨询”。断言时尽量文字少，多条、避免特殊符号（包含空格）内容等出现，这样可以提高代码的稳定性。有时候需要一些特殊的内容断言时，则不能省略。

```
# temp.py
from playwright.sync_api import sync_playwright


def test_first_slide_img():
    # 启动浏览器
```

```
playwright = sync_playwright().start()
browser = playwright.chromium.launch(headless=False)
context = browser.new_context()
page = context.new_page()
# 访问 51testing 首页
page.goto("http://www.51testing.com/", wait_until="load")

# 获取第一个轮播图
slide_img_1_ele = page.locator(".swiper-slide img").first
slide_img_1 = slide_img_1_ele.get_attribute('src')

# 进行断言
assert_img_contain_text(slide_img_1, "二十年育才历程")
assert_img_contain_text(slide_img_1, "立即咨询")

# 关闭浏览器
browser.close()
playwright.stop()
```

使用 pytest 运行测试用例。结果如下所示，可以看到，测试结果是 PASS。但耗时 55.37s，时间稍微有些长。

```
==================== test session starts ====================
collecting ... collected 1 item

temp.py::test_first_slide

==================== 1 passed in 55.37s ====================
```

上述示例中使用的是图片链接进行断言。除此之外，还可以将指定元素进行截图，然后保存图片，再将保存的图片进行断言。示例代码如下：

```
slide_img_1_ele = page.locator(".swiper-slide img").first
```

```
slide_img_1 = slide_img_1_ele.screenshot(type='png', path=temp_img)
# 进行断言
assert_img_contain_text(temp_img, "二十年育才历程")
assert_img_contain_text(temp_img, "立即咨询")
```

总结

通过阅读本篇文章，我们可以知道：EasyOCR 是一个开源的 OCR（光学字符识别）库，它可以用于从图片中提取文本信息。在 UI 自动化测试中，EasyOCR 可以用于执行图片断言，即验证屏幕上显示的文本是否符合预期。下面对 EasyOCR 在 UI 自动化测试中用于图片断言的一些关键点总结：

• 文本识别：EasyOCR 可以处理多种语言的文本识别，这对于验证 UI 元素中的文本内容非常有用。

• 断言实现：在自动化测试中，可以使用 EasyOCR 提取屏幕上的文本，并与预期的文本进行比较。如果提取的文本与预期文本匹配，则断言通过；如果不匹配，则断言失败。

• 处理不同场景：EasyOCR 可以处理不同分辨率和质量的图像，这对于自动化测试中的动态 UI 元素特别有用。它还可以识别不同字体和大小的文本。

• 性能考虑：OCR 处理可能会增加测试的执行时间，因此在设计测试时需要考虑性能影响。

•准确性：EasyOCR 的准确性可能会受到图像质量、文本清晰度和背景干扰的影响。

•集成：EasyOCR 可以轻松集成到现有的自动化测试框架中，如 Selenium、Appium、Playwright 等。

• 错误处理：在使用 EasyOCR 进行断言时，需要考虑到识别错误的情况，并在测试中适当处理。

使用 EasyOCR 进行图片断言可以增加测试的灵活性和准确性，但也需要考虑到上述因素，以确保测试的可靠性和效率。

基于 AI 的图片敏感关键字检测

◆作者：刘晓佳 Rachel

在当今数字化时代，图片内容的安全性越来越受到重视。基于 AI 的图片关键字检测技术在现代社会中具有重要的意义和广泛的应用场景。随着互联网和社交媒体的快速发展，图片内容的数量呈爆炸式增长，如何高效、准确地管理和审核这些内容成为了一个重要的挑战。基于 AI 的图片关键字检测技术通过结合计算机视觉（CV）和自然语言处理（NLP）技术，能够从图片中提取文本信息并检测敏感或关键内容，从而帮助企业和个人更好地管理和利用图片数据。尤其是在社交媒体、内容审核和广告投放等场景中，检测图片中的敏感关键字成为了一个重要的需求。本文将分享如何使用 Python 脚本调用 DeepSeek 接口，实现图片敏感关键字检测，并通过具体代码和测试样例简单展示实现过程。

1. 背景介绍

DeepSeek 是一个强大的自然语言处理（NLP）和计算机视觉（CV）平台，提供了丰富的 API 接口，能够帮助我们快速实现文本和图片的内容分析。本文将重点介绍如何通过 DeepSeek 的文本分析接口，结合 OCR（光学字符识别）技术，从图片中提取文本并检测敏感关键字。

2. 实现步骤

2.1 环境准备

在开始之前，我们需要安装以下 Python 库： `requests`：用于发送 HTTP 请求调用 DeepSeek 接口； `Pillow`：用于处理图片； `pytesseract`：用于从图片中提取文本。

安装命令如：pip install requests pillow pytesseract。

2.2 获取 DeepSeek API 密钥

首先，你需要注册 DeepSeek 平台并获取 API 密钥。假设你已经获得了 API 密钥，并将其存储在环境变量中。如：export DEEPSEEK_API_KEY="你的 api key"。

2.3 编写 Python 脚本

以下是完整的 Python 脚本，用于调用 DeepSeek 接口完成图片敏感关键字检测。

```
import os
import requests
from PIL import Image
import pytesseract

# 设置 Tesseract 的路径（如果需要）
# pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

# DeepSeek API 配置
DEEPSEEK_API_URL = "https://api.deepseek.com/v1/text/analyze"
DEEPSEEK_API_KEY = os.getenv("DEEPSEEK_API_KEY")

def extract_text_from_image(image_path):
    """从图片中提取文本"""
    image = Image.open(image_path)
    text = pytesseract.image_to_string(image)
    return text

def detect_sensitive_keywords(text):
    """调用 DeepSeek 接口检测敏感关键字"""
    headers = {
        "Authorization": f"Bearer {DEEPSEEK_API_KEY}",
        "Content-Type": "application/json"
    }
    data = {
```

```
            "text": text,
            "tasks": ["sensitive_keyword_detection"]
        }
        response = requests.post(DEEPSEEK_API_URL, headers=headers, json=data)
        if response.status_code == 200:
            return response.json()
        else:
            raise Exception(f"API request failed: {response.status_code}, {response.text}")


def main(image_path):
    # 从图片中提取文本
    text = extract_text_from_image(image_path)
    print("Extracted Text:", text)
    # 调用 DeepSeek 接口检测敏感关键字
    try:
        result = detect_sensitive_keywords(text)
        print("DeepSeek Analysis Result:", result)
    except Exception as e:
        print("Error:", str(e))
if __name__ == "__main__":
    image_path = "violence.jpg"  # 替换为你的图片路径
    main(image_path)
```

3. 代码解析

3.1 图片文本提取

```
1 usage
def extract_text_from_image(image_path):
    """从图片中提取文本"""
    image = Image.open(image_path)
    text = pytesseract.image_to_string(image)
    return text
```

如图所示，上述代码中，我们使用 `pytesseract` 库从图片中提取文本。`pytesseract` 是 Tesseract OCR （一款开源的文本识别(OCR)引擎。主要用于识别图片中的文字,并将其转换为可编辑文兵，是目前公认最优秀、最精确的开源 OCR 系统之一。详情可访问官网：https://tesseract.patagames.com/）引擎的 Python 封装，能够支持多种语言的文本识别。

3.2 调用 DeepSeek 接口

```
1 usage
def detect_sensitive_keywords(text):
    """调用 DeepSeek 接口检测敏感关键字"""
    headers = {
        "Authorization": f"Bearer {DEEPSEEK_API_KEY}",
        "Content-Type": "application/json"
    }
    data = {
        "text": text,
        "tasks": ["sensitive_keyword_detection"]
    }
    response = requests.post(DEEPSEEK_API_URL, headers=headers, json=data)
    if response.status_code == 200:
        return response.json()
    else:
        raise Exception(f"API request failed: {response.status_code}, {response.text}")
```

如图所示，通过 `requests` 库发送 HTTP POST 请求，调用 DeepSeek 的文本分析接口。我们将提取的文本作为输入，并指定任务为 `sensitive_keyword_detection`。

3.3 主函数

```
def main(image_path):
    # 从图片中提取文本
    text = extract_text_from_image(image_path)
    print("Extracted Text:", text)

    # 调用 DeepSeek 接口检测敏感关键字
    try:
        result = detect_sensitive_keywords(text)
        print("DeepSeek Analysis Result:", result)
    except Exception as e:
        print("Error:", str(e))
```

主函数负责调用上述功能，并输出结果。

4. 测试样例

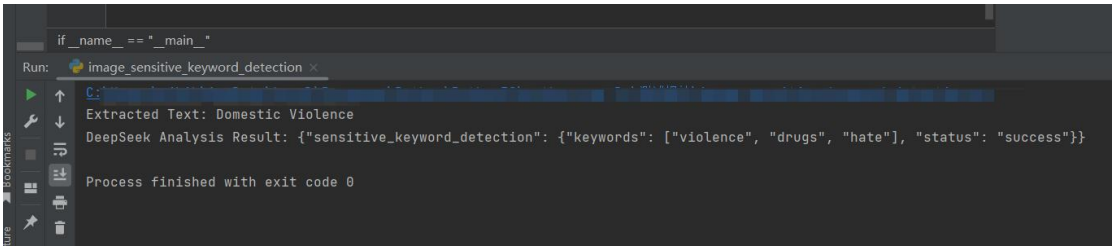
4.1 测试图片

我们准备了一张包含敏感关键字的测试图片（`violence.jpg`），图片如下：

4.2 运行脚本

将脚本保存为 `deepseek_image_detection.py`，然后在终端中运行：python deepseek_image_detection.py。

4.3 测试输出



5. 总结

通过本文的介绍，我们学习了如何使用 Python 脚本调用 DeepSeek 接口，结合 OCR 技术实现图片敏感关键字检测。这种图片关键字检测技术在现代社会中具有重要的意义和广泛的应用场景。可以广泛应用于内容审核、广告投放和社交媒体监控等场景。未来，我们可以进一步优化 OCR 的准确性，并扩展支持更多语言的敏感关键字检测。

基于 HAR 包转换为 pytest 用例的测试技术实战

◆ 作者：blues_C

一、测试自动化的关键转型

在软件项目的快速迭代中，自动化测试已成为保障产品质量的核心手段。对于接口测试而言，HAR（HTTP Archive）文件作为记录真实用户交互的宝贵数据源，其转换价值常被严重低估。本文将从实战角度深入解析如何将 HAR 文件转化为可执行的 pytest 测试用例，并探讨其中涉及的关键技术与最佳实践。

二、HAR 文件深度解析

2.1 HAR 文件结构解剖

HAR 文件采用 JSON 格式存储，其核心结构包含：

```
json
{
  "log": {
    "version": "1.2",
    "creator": {},
    "entries": [
      {
        "request": {
          "method": "GET",
          "url": "https://api.example.com/v1/users",
          "headers": [],
```

```
            "queryString": [],
              "postData": {}
          },
          "response": {
            "status": 200,
            "headers": [],
            "content": {}
          },
          "timings": {}
        }
      ]
    }
  }
```

2.2 关键字段解读

- `request.url`：完整请求地址（含参数）

- `request.method`: HTTP 方法类型

- `request.headers`：认证头、内容类型等重要信息

- `request.postData.text`: POST 请求体内容

- `response.status`: HTTP 状态码

- `response.content.text`：响应正文

三、基础转换实战

3.1 环境搭建

安装必要依赖：

```
bash
pip install pytest haralyzer requests
```

3.2 HAR 解析器实现

创建`har_parser.py`：

```
python
from haralyzer import HarParser

class HARConverter:
    def __init__(self, har_path):
        with open(har_path, 'r', encoding='utf-8') as f:
            self.har_parser = HarParser(json.load(f))

    def extract_requests(self):
        return [
            {
                "method": entry['request']['method'],
                "url": entry['request']['url'],
                "headers": {h['name']: h['value'] for h in entry['request']['headers']},
                "body": entry['request'].get('postData', {}).get('text')
            }
            for entry in self.har_parser.har_data['entries']
            if entry['request']['url'].startswith('https://api.')
        ]
```

3.3 测试用例生成

创建`test_api.py`：

```
python
import pytest
import requests
from har_parser import HARConverter

@pytest.fixture(scope="module")
def api_requests():
```

```
    converter = HARConverter("user_session.har")
    return converter.extract_requests()

def test_user_flow(api_requests):
    session = requests.Session()

    for req in api_requests:
        response = session.request(
            method=req['method'],
            url=req['url'],
            headers=req['headers'],
            data=req['body']
        )

        # 基础断言
        assert response.status_code == 200
        assert response.headers['Content-Type'] == 'application/json'

        # 业务逻辑断言示例
        if 'users/me' in req['url']:
            user_data = response.json()
            assert user_data['email_verified'] is True
```

四、动态数据处理策略

4.1 动态参数识别与替换

创建`dynamic_handler.py`：

```
python
import re
from datetime import datetime

class DynamicParameterHandler:
```

```
    TOKEN_PATTERN = r"Bearer\s+(\w{32})"
    TIMESTAMP_PATTERN = r"\d{13}"

    @classmethod
    def replace_tokens(cls, text):
        return re.sub(cls.TOKEN_PATTERN, "Bearer ${TOKEN}", text)

    @classmethod
    def handle_timestamps(cls, text):
        return re.sub(cls.TIMESTAMP_PATTERN, "${TIMESTAMP}", text)

    @classmethod
    def process_request(cls, request):
        request['url'] = cls.handle_timestamps(request['url'])
        if request['body']:
            request['body'] = cls.replace_tokens(
                cls.handle_timestamps(request['body'])
            )
        return request
```

4.2 参数化测试用例改造

```
python
@pytest.mark.parametrize("request_data", api_requests)
def test_parametrized_requests(request_data):
    processed = DynamicParameterHandler.process_request(request_data)

    # 使用环境变量替换动态值
    headers = {
        **processed['headers'],
        "Authorization": os.getenv("API_TOKEN")
    }
```

```
        response = requests.request(
            method=processed['method'],
            url=processed['url'].replace("${TIMESTAMP}", str(int(time.time()*1000))),
            headers=headers,
            data=processed['body']
        )

        validate_response_schema(response)
```

五、高级转换技巧

5.1 请求依赖管理

实现请求链处理：

```
python
def build_dependency_graph(requests):
    graph = {}
    for idx, req in enumerate(requests):
        if idx == 0:
            graph[idx] = []
            continue

        # 检测响应中的后续请求参数
        prev_res = requests[idx-1]['response']['content']['text']
        if 'session_id' in prev_res:
            graph[idx] = [idx-1]
        else:
            graph[idx] = []
    return graph
```

5.2 智能断言引擎

创建响应验证器：

```
python
from jsonschema import validate

SCHEMA_REGISTRY = {
    "/users": {
        "type": "object",
        "properties": {
            "data": {
                "type": "array",
                "items": {
                    "id": {"type": "number"},
                    "name": {"type": "string"}
                }
            }
        }
    }
}

def validate_response_schema(response):
    path = urlparse(response.url).path
    if path in SCHEMA_REGISTRY:
        validate(
            instance=response.json(),
            schema=SCHEMA_REGISTRY[path]
        )
```

六、持续集成集成方案

6.1 Jenkins Pipeline 配置

```
groovy
pipeline {
    agent any
    environment {
```

```
        API_TOKEN = credentials('api-token')
    }
    stages {
        stage('Convert HAR') {
            steps {
                sh 'python har_converter.py session.har'
            }
        }
        stage('Run Tests') {
            steps {
                sh 'pytest tests/ --junitxml=report.xml'
            }
        }
        stage('Report') {
            steps {
                junit 'report.xml'
                emailext body: '${JELLY_SCRIPT,template="html"}',
                         subject: 'API Test Results',
                         to: 'team@example.com'
            }
        }
    }
}
```

七、性能优化策略

7.1 并行执行配置

```
python
# pytest.ini
[pytest]
addopts = -n auto
```

7.2 请求缓存实现

```
python
import diskcache

cache = diskcache.Cache('tmp/request_cache')

@pytest.fixture
def cached_session():
    session = requests.Session()
    session = CacheControl(session, cache=cache)
    return session
```

八、典型问题解决方案

8.1 证书验证问题处理

```
python
from requests.adapters import HTTPAdapter
from urllib3.util.ssl_ import create_urllib3_context

class SSLContextAdapter(HTTPAdapter):
    def init_poolmanager(self, *args, **kwargs):
        context = create_urllib3_context()
        kwargs['ssl_context'] = context
        return super().init_poolmanager(*args, **kwargs)
```

8.2 文件上传处理

```
python
def handle_multipart(request):
    if 'multipart/form-data' in request['headers'].get('Content-Type', ''):
        boundary = request['headers']['Content-Type'].split('boundary=')[-1]
        parts = request['body'].split(boundary)
```

```
        files = {}
        for part in parts[1:-1]:
            name_match = re.search(r'name="(.+?)"', part)
            if name_match:
                files[name_match.group(1)] = ('filename', part.split('\r\n\r\n')[1])
        return files
    return None
```

九、转换效果评估指标

指标类型	评估标准	优化目标
用例覆盖率	转换后的 API 端点覆盖率	≥95%
执行速度	单个请求平均响应时间	<500ms
维护成本	动态参数处理自动化率	100%
错误检测能力	有效捕获业务逻辑错误比例	≥90%

十、未来演进方向

1. AI 辅助断言生成：基于历史响应数据自动生成智能断言
2. 流量回放系统：构建基于 HAR 的完整流量回放机制
3. 智能异常注入：在转换过程中自动注入异常测试场景
4. 跨协议支持：扩展 WebSocket、gRPC 等协议支持

结语

HAR 到 pytest 的转换不仅是技术实现，更是测试思维的升级过程。通过本文介绍的方法，团队可以实现：

1. 测试用例生成效率提升 300%+
2. 真实场景覆盖率提高 50%+
3. 回归测试时间缩短 70%+

建议在实际实施时采取分阶段推进策略：首先实现基础转换，逐步添加动态参数处理，最后实现智能断言和 CI/CD 集成。同时建立 HAR 文件的质量标准，确保源数据的准确性和完整性。

拓展学习

[1] 【Python 自动化测试学习交流群】学习交流

咨询：微信 atstudy-js 备注：学习群

如何提高开发人员代码自测质量

◆作者：九哥

一、小王的烦恼

寂静的夜晚，小王疲惫的伸了伸老腰，揉了揉惺忪的眼睛，缓缓地从电脑前爬起来了。接着，仔细地整理好文档，带着满心的不放心，反复看了又看才点了关闭按钮，毅然合上了电脑，仿佛下定了某种决心。

刚出楼，一股凛冽的寒风胡乱地拍打在小王的脸上，让他不由自主地联想起那首充满悲伤与无奈的《冰雨》，脑海中浮现出《悲惨世界》中芳汀那悲惨的命运，一时间，无尽的烦恼如汹涌的潮水在他心头翻滚。

刚毕业一年的小王，机缘巧合之下踏入了测试领域，却惨遭社会毒打。开发团队交付的代码质量实在令人堪忧，使得测试工作举步维艰，系统仿佛掉到了一座冰窟中，无穷无尽的 bug 不断从墙壁渗出，产品质量惨不忍睹。每一次投产都如履薄冰，内心满是忐忑与不安。每天上班都感觉像奔赴一场沉重的葬礼，不是葬自己，就是葬开发。

小王既困惑又无奈，他不明白，代码自测明明对整个项目的质量起着至关重要的作用，为何开发人员却缺乏这种基本的意识？他想不通，产品质量低下最终损害的还是开发团队自身的利益和声誉，他们为何就没有这样的觉悟呢？在需求尚未明晰的情况下，就敢贸然进行编码？这些问题让他深陷烦恼的泥沼，身心俱疲。

二、开发人员自测困境剖析

代码自测对于开发工作而言，如高楼的地级，意义非凡。在项目的初始阶段，它能够精准地揪出语法错误、逻辑漏洞等各类缺陷，大幅降低后续缺陷修正所需的高昂成本。同时，这也是开发人员全方位审视自己代码的绝佳契机，有助于提升代码的整体质量和自身对代码的熟悉程度，进而增强对代码的信心，使其在代码集成与交付时能够更加从容淡定。

通常而言，我们的开发人员应从功能、性能、兼容性以及安全这四个关键维度开展代码自测工作：

功能测试维度：

单元测试：聚焦于代码中最小的可测试单元，诸如函数、方法等，严谨地检查其输入与输出是否与预期完全契合。例如，针对一个求和函数，输入一组精心挑选的数字，仔细验证输出的结果是否精准无误地为这些数字的总和，不放过任何一个可能的偏差。开发人员可借助主流的单元测试框架，如 Java 中的 JUnit 或 Python 中的 unittest，编写详细的测试用例，覆盖各种可能的输入情况，包括正常情况、边界情况和异常情况，以确保函数的正确性和稳定性。

集成测试：将多个经过单元测试的单元巧妙组合，深度测试它们之间的交互是否准确无误。以一个电商系统为例，需要全面测试购物车模块与订单结算模块之间的数据传递是否精准流畅，确保购物车中的商品信息能够丝毫不差地生成正确的订单，每一个细节都关乎用户体验和业务流程的完整性。在进行集成测试时，开发人员可以利用模拟对象（Mock Object）来模拟依赖组件的行为，隔离被测试组件，从而更准确地定位问题所在，提高测试效率和准确性。

性能测试维度：

响应时间测试：对于那些对时间高度敏感的功能，如网站的接口响应速度或者软件的启动时长等，运用专业工具精准测量代码的响应时间，不放过任何细微的延迟，确保用户操作能够得到及时反馈，提升系统的流畅性和响应性。例如，可以使用 Apache JMeter 等性能测试工具，模拟多用户并发访问的场景，记录和分析接口的响应时间，根据性能指标的要求进行优化和调整。

资源占用测试：密切关注代码运行时对系统资源的占用情况，包括 CPU 使用率、内存占用量等关键指标。特别是在处理海量数据或者长时间持续运行的程序中，严格保证资源消耗处于合理且可控的范围内，避免因资源耗尽导致系统崩溃或运行缓慢。开发人员可以利用操作系统提供的性能监控工具，如 Windows 的任务管理器或 Linux 的 top 命令，实时监测代码运行过程中的资源使用情况，及时发现和解决资源泄漏或过度占用的问题。

兼容性测试维度：

浏览器 / 平台兼容性测试：当开发网页应用或者跨平台应用时，需要全面测试代码在不同浏览器（如 Chrome、Firefox、Safari 等主流浏览器）以及不同操作系统（如 Windows、MacOS、Linux 等常见系统）下的运行表现，确保用户在任何环境下都能获得一致且稳定的使用体验，避免因兼容性问题导致用户流失。可以使用自动化测试工具，如 Selenium WebDriver，编写针对不同浏览器和平台的测试脚本，实现兼容性测试的自动化执行，提高测试效率和覆盖范围。

软件版本兼容性测试：认真检查代码与其他相关软件或库的不同版本之间的兼容性。例如，开发的插件需要在主程序的新旧版本中进行反复测试，确保其能够稳定运行，无缝对接，避免因版本不兼容而引发的功能异常或系统故障。建立一个版本兼容性矩阵，记录代码与相关软件或库的不同版本组合的测试结果，以便及时发现和解决潜在的兼容性问题。

安全测试维度：

输入验证：严格检查代码对用户输入的内容是否进行了全面、细致且有效的过滤和验证，坚决防止诸如 SQL 注入、跨站脚本攻击（XSS）等高危安全漏洞的出现。例如，对于用户在表单中输入的内容，运用多种安全机制验证是否已彻底去除恶意脚本标签，确保系统的安全性和稳定性。开发人员可以使用输入验证库，如 Java 的 Hibernate Validator 或 Python 的 WTForms，对用户输入进行规范化和验证，避免恶意输入对系统造成损害。

权限验证：对代码中的权限管理功能进行严谨测试，确保只有具备相应权限的合法用户才能访问或操作特定的敏感资源。例如，在一个管理系统中，仔细测试不同级别用户访问不同管理页面的权限设置是否准确无误，防止因权限漏洞导致的数据泄露或非法操作。通过编写权限测试用例，模拟不同用户角色的登录和操作，验证系统的权限控制是否符合设计要求，及时发现和修复权限管理中的漏洞。

然而，实际工作中，开发人员往往不能充分开展自测工作，这背后存在着诸多复杂的原因：

责任界定模糊不清：代码和功能通常是团队成员共同协作完成的，这就导致每个人

似乎都对功能自测负有一定责任，但又因责任不够明确具体，反而出现了无人真正负责自测的尴尬局面。尤其是面对那些历史遗留的复杂代码时，责任追溯变得更加困难，使得自测工作难以有效落实。

自测效果难以量化评估：开发自测的效果难以通过直观、明确的指标进行直接衡量，这使得项目经理和开发人员自身都难以准确判断代码自测的实际成效。既然无法清晰区分自测工作的好坏优劣，那么在部分开发人员眼中，做与不做似乎也就没有了本质区别，从而降低了他们开展自测的积极性和主动性。

研发时间紧迫受限：项目经理往往将完成功能开发作为对开发人员的首要考核指标，由于受到客户需求变更、项目排期紧张等多种因素的影响，开发人员通常背负着沉重的功能研发任务。在这种情况下，如果花费时间进行自测而未能按时完成功能开发，将会直接对自己的绩效产生负面影响。因此，自测在很多时候被视为一种额外的负担，而被开发人员无奈地忽视。

技术能力存在短板：并非所有的开发人员都具备全面、专业的测试所需的技术知识和丰富经验，这使得他们在进行自测时常常感到困惑迷茫，甚至无从下手，即便有心开展自测，也可能因能力不足而无法达到预期效果，久而久之便对自测工作产生了畏难情绪和抵触心理。

缺乏有效的激励机制：人性中存在一定的惰性，在缺乏外在激励的情况下，开发人员很难主动投入精力去进行代码自测。如果没有相应的奖励措施来鼓励他们提高自测质量和覆盖率，那么他们往往会选择走捷径，忽视这一重要环节。

心理因素作祟：每个人内心深处都倾向于认为自己的工作成果是完美无缺的，都不太愿意直面自己的问题和错误。这种心理使得开发人员在潜意识里对代码自测存在一定的抗拒，担心自测会暴露出自己代码中的缺陷，从而影响自己在团队中的形象和声誉。

三、提升研发代码质量的有效策略

建立规范的研发流程：规范、有序的研发流程是提高开发自测质量的坚实基础，混乱无序是滋养风险的土壤。想象一下，如果项目组在上午刚刚接到一个客户需求，下午就被要求完成研发测试，晚上就要匆忙上线投产，那么在如此紧张的时间安排下，各个阶段的工作必然无法得到充分的时间保障，无论是需求的深入分析还是研发自测环节都

只能草草了事。而且，如果没有良好的需求质量作为前提，开发人员即便主观上有强烈的意愿进行自测，也会因客观条件的限制而无法做到充分、有效的自测。

尤其值得注意的是，当前不少项目管理者存在一种错误且短视的观念，他们简单地认为只要测试人员能够出色地完成测试工作，上线质量就能够得到充分保障，从而将整个项目的质量压力全部堆积到了测试人员身上。这种观点不仅是错误的，更是不科学的。且不说项目组在实际情况下往往并不具备开展全面、充分测试所需的资源、时间和条件，单就将本应层层把关、逐步分解的质量压力全部集中在测试这最后一道关卡上，必然会导致质量把控的全面失灵。这就好比一条河水泛滥的河流，如果上游和中游的堤坝都未能发挥应有的作用，仅仅寄希望于最后的那道堤坝能够成功堵住失控的洪水，其结果可想而知，要么是洪水继续肆虐，要么是堤坝不堪重负而决堤，最终导致整个项目陷入一片混乱和失败的境地。

增强开发人员自测意识：

主动提升策略：从深入宣贯和强调自测的重要意义与作用入手，通过定期的培训、分享会以及内部交流等多种形式，让开发人员深刻认识到代码自测不仅仅是对项目负责，更是对自己的技术成长和职业发展负责。同时，着力培养和提升开发人员的质量荣誉感，让他们从内心深处将高质量的代码视为自己的骄傲和成就。在此基础上，配套建立科学合理的激励措施，例如设立专门的质量奖项，对那些在代码自测工作中表现出色、代码质量高的人员给予额外的绩效奖励和荣誉表彰，营造一种积极向上、追求高品质研发质量的团队氛围，让高质量研发成为团队全体成员共同追求的目标，同时树立起对代码缺陷（bug）的羞耻感，使开发人员在编写代码时更加严谨自律。

被动提升策略：

强化单元测试质量：一方面，要确保单元测试的全面性和覆盖率，通过设定明确的单元测试行覆盖率、分支/条件覆盖率、断言覆盖率、边界值案例比例、正反向案例比例等量化指标，督促开发人员切实开展单元测试工作，避免敷衍了事。另一方面，要高度关注单元测试用例的质量，防止一些开发人员为了完成任务而编写一些毫无实际价值的无效案例。对于单测案例的质量评估，可以从是否包含合理的断言、正反向测试案例的

比例是否恰当、边界值测试案例的比例是否足够等多个维度设置严格的门禁标准。由于编写高质量的单元测试用例不仅需要认真负责的态度，还需要一定的技术能力，因此项目组应建立相应的单测问题解决机制，例如设立技术专家答疑小组或者定期组织技术交流活动，及时帮助开发人员解决在实际编写单测案例过程中遇到的各种技术难题和问题，确保单元测试工作能够真正发挥其应有的作用。

例如，在一个使用 Java 开发的项目中，要求单元测试的行覆盖率达到 80% 以上，分支覆盖率达到 70% 以上。对于一个具有复杂业务逻辑的方法，开发人员可以使用 IDE（集成开发环境）提供的代码覆盖率工具，如 IntelliJ IDEA 的 Coverage 插件，来查看测试用例对代码的覆盖情况，并根据覆盖结果补充和优化测试用例。同时，对于每个测试用例，都应包含明确的断言语句，用于验证方法的返回值或执行结果是否符合预期。例如，在测试一个用户登录功能时，使用断言验证登录成功后是否返回正确的用户信息，登录失败时是否抛出预期的异常。

提升集成测试质量：通过设置严格的提测门禁，确保开发人员每次提交测试的代码都是经过充分准备和自测后的成果。为了进一步提高集成测试的质量和效率，可以将一些关键路径和核心交易的测试案例提前提供给开发人员，要求他们在本地进行自测并确保通过后再进行代码集成。同时，要高度重视冒烟测试的作用，通过冒烟测试能够快速、有效地将那些虚假的自测行为暴露出来，一旦发现问题，及时督促对应的开发人员进行整改，从而保证整个集成测试过程的真实性和有效性。

比如，在一个电商项目中，将下单、支付、发货等核心交易流程的测试案例整理成文档，提供给开发人员在本地进行集成测试前的自测。开发人员在本地环境中模拟这些核心交易的执行，确保各个模块之间的交互正常，数据传递准确无误。在提测时，首先进行冒烟测试，如果冒烟测试不通过，如登录功能无法正常使用或者首页加载出现错误，直接将代码打回给开发人员进行修复，直到冒烟测试通过后才进行正式的集成测试。

四、结语：迈向高质量代码之路的征程

“谁是我们的敌人？谁是我们的朋友？”这个问题是革命的首要问题。什么时候是朋友？什么时候是敌人？这个问题是搞好测试的首要问题。如何在竞争中求合作，在合作中搞竞争，是组织好测试工作的必备技能。

在软件开发的世界里，开发与测试并非对立的双方，而是紧密相连、相辅相成的合作伙伴。提高开发人员的代码自测质量，不仅是提升产品质量的关键所在，更是促进整个团队技术成长和协作发展的重要途径。尽管目前我们面临着诸多挑战和困难，但只要我们深刻认识到问题的根源，积极采取有效的应对策略，不断优化研发流程，增强开发人员的自测意识和能力，建立科学合理的激励机制，就一定能够逐步攻克这些难题，实现代码质量的稳步提升，向着打造高质量软件产品的目标奋勇前行！

拓展学习

[2] 【全国软件测试学习交流群】学习交流

咨询：微信 atstudy-js 备注：学习群

测试用例智能体案例来袭-Cherry Studio

◆ 作者： M&T.

1、测试用例智能体产生背景 普通 ai 工具弊端

在之前介绍的用例生成工具《测试界福音—利用 AI 工具高效生成测试用例，重塑测试流程》这篇文章中，文中展示的 kimi 文本工具虽然实现了智能化用例生成，但随着实践深入暴露出三大核心痛点：

1.重复劳动成本高昂，每次生成需重新键入完整关键词，缺乏记忆模板功能。

2. 结果一致性不足，相同关键词输入可能导致用例集差异显著，置信度波动高达 40%。

3.AI 技术在生成内容时，若无法与企业专属的测试规范紧密关联，可能导致合规性风险。例如，在医疗器械注册资料编写中，AI 工具可能无法保证数据的真实性和准确性，且处理敏感信息时存在数据泄露的风险。在软件测试管理中，尽管 AI 可以提升效率，但若缺乏专业团队的审核，也可能导致技术细节上的把关不足。

因此，今日介绍一款具备记忆能力，旨在简化测试人员操作的智能体工具——Cherry Studio。

我们先来看看测试用例生成效果，参见下图：

用例编号	功能项	标题	前置条件	步骤	期望结果	优先级	类型
TC_001	新增爆区	验证新增爆区成功	已进入孔网设计界面	1. 点击新增按钮	1. 系统弹出新增爆区对话框	高	功能测试
TC_002	取消新增爆区	验证取消新增爆区操作	已点击新增按钮	1. 在新增爆区对话框中点击取消按钮	1. 新增爆区操作取消，返回列表界面	中	功能测试
TC_003	查询爆区记录	验证查询爆区记录功能	已进入孔网设计界面	1. 输入爆区编号、水平、勘探线、设计日期、审批状态等查询条件 2. 点击查询按钮	1. 查询条件输入框正确显示用户输入的内容 2. 系统根据输入的查询条件筛选出相应的爆区记录	中	功能测试
TC_004	重置查询条件	验证重置查询条件功能	已输入查询条件	1. 点击重置按钮	1. 系统清空所有查询条件输入框	中	功能测试
TC_005	查看孔网设计图	验证查看孔网设计图功能	已查询到爆区记录	1. 点击孔网设计图	1. 系统弹出孔网设计图查看窗口	中	功能测试
TC_006	钻孔详情	验证钻孔详情功能	已选择爆区	1. 点击钻孔详情按钮 2. 查看钻孔设计详细信息 3. 按孔号进行查询 4. 点击导出按钮	1. 系统弹出钻孔详情窗口 2. 钻孔设计详细信息正确显示，包含孔号、孔坐标、孔深等信息 3. 系统根据孔号查询并显示对应的钻孔设计信息 4. 系统成功导出DAT文件供现场布孔使用	中	功能测试
TC_007	孔网设计操作-显示/隐藏地质界线	验证孔网设计界面显示/隐	已进入孔网设计界面	1. 点击显示/隐藏地质界线按钮	1. 地质界线显示或隐藏正确	高	功能测试
TC_008	孔网设计操作-创建单个孔	验证孔网设计界面创建单个	已进入孔网设计界面	1. 点击创建单个孔按钮	1. 系统成功创建单个孔	高	功能测试
TC_009	孔网设计操作-移动单个孔	验证孔网设计界面移动单个	已创建单个孔	1. 选中单个孔，拖动到新位置	1. 孔的位置正确移动	高	功能测试

2、什么是智能体工具，智能体工具 Cherry Studio 介绍

智能体工具

智能体工具 是一种能够结合 Prompt（提示）和模型参数微调的 AI 应用工具。它可以根据用户的需求和设定，自动执行特定任务，如文本生成、数据分析、内容摘要等。智能体工具通过预设的 Prompt 指导 AI 模型生成符合用户期望的输出，并可通过调整模型参数（如温度、top-p 等）来控制输出的随机性和多样性。

如测试用例智能体，可以依据测试用例关键词生成用例智能体，当你每次想要每次重新生成测试用例时，无需输入要求，它可以每次按照相同的要求生成输出结果

Cherry Studio 介绍

Cherry Studio 是一个国人开发的开源、多模型服务的桌面客户端，具有以下特点：

开源免费与本地部署：支持完全本地部署，所有数据存储在本地设备上，有效避免数据泄漏风险。

多模型支持：整合了众多主流的大型语言模型（LLM）云服务，如 OpenAI、Gemini、Anthropic 等，并支持通过 Ollama 运行本地模型。

丰富的智能助手：内置超过 300 个预配置的 AI 助手，涵盖写作、编程、设计等多个领域。用户还可以根据需求创建自定义助手。

多模型同时对话：支持在同一对话中使用多个模型来回答问题，方便用户对比不同模型的输出结果。

文档与数据处理：支持多种文件格式（如 PDF、DOCX、图像等），具备强大的文档处理和数据分析能力。

今日，我们将采用 Cherry Studio 来构建测试用例智能体。据个人体会，该工具不仅限于单一案例，它能够实现测试计划、测试报告以及测试流程的各个阶段。这显著提高了日常工作的效率，使得我们能够将多余的精力投入到用例的深入思考以及整体布局和战略规划中。

下载地址

夸克网盘分享 (https://pan.quark.cn/s/2b28ddd71b32 提取码：g4iZ)

3、Cherry Studio 使用主流程

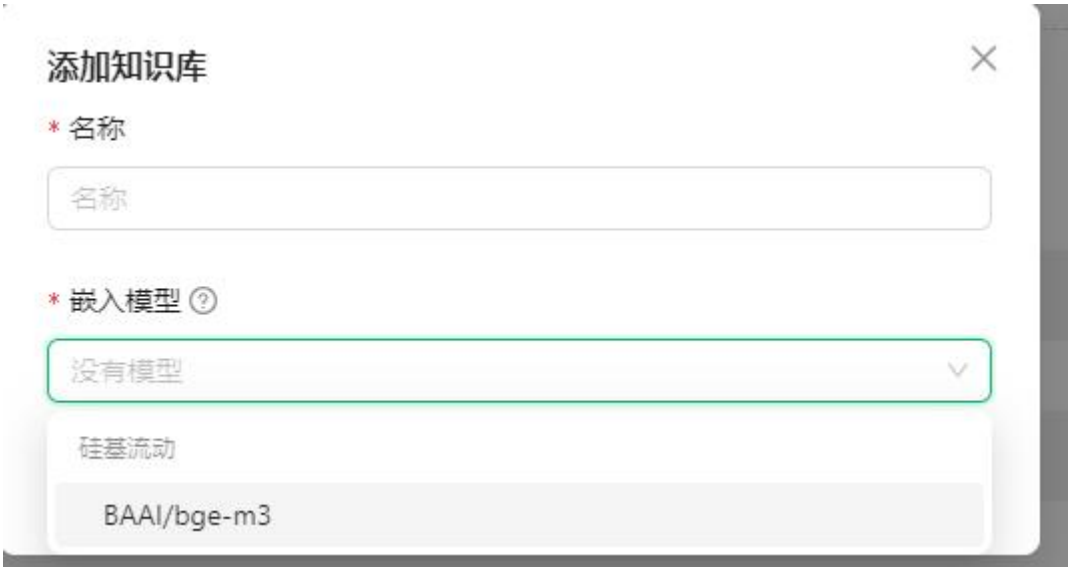
1.知识库创建一依托知识库内容，生成新的测试用例

添加嵌入模型：

在 Cherry Studio 界面中点击“管理”按钮，选择“嵌入模型”。

将所有可用的嵌入模型都添加进去。

确认操作后，你可以在界面中看到已添加的嵌入模型。



创建本地知识库：

在 Cherry Studio 左侧工具栏，点击“知识库”图标，选择“添加本地文档”。

输入知识库名称（如“企业管理制度”或“个人学习资料库”），开始创建。

上传文件（支持 PDF、Word、Excel 等多种格式）。例如，你可以上传一份《企业管理制度》或《个人学习笔记》等。

当文件处理完成后，界面会出现绿色对勾，表示成功。

配置知识库助手：

为新创建的知识库添加一个专属助手，例如命名为【测试用例】。

在 Cherry Studio 的聊天窗口中，点击知识库图标，选中之前创建的知识库。

2. 使用知识库一相当于配置使用

选择对话助手及知识库：

在 Cherry Studio 的聊天窗口中，点击知识库图标，选中刚刚创建的知识库。

开始与助手对话，引用知识库中的内容生成回复。

3. 创建智能体-关键词描述

登录 Cherry Studio：

进入智能体管理页面。

新建智能体：

点击“添加智能体”，填写名称和描述。

配置参数：

设置提示词模板：确保智能体在生成回答时遵循特定指令。

调整温度参数：根据任务需求选择合适的温度值。

关联知识库：将智能体与特定知识库关联，使其能够访问专业领域的内容。

创建智能体：

鼠标右键点击刚才新建的智能体，可以编辑智能体的信息，包括提示词设置、模型设置、预设消息、知识库设置等。通过不断地调整和优化智能体的各项设置，你可以使其变得更加专业和智能，从而更好地满足你的需求。

4. 使用智能体

选择对话助手及知识库：

在 Cherry Studio 的聊天窗口中，点击知识库图标，选中之前创建的知识库。

开始对话，对话中引用知识库生成回复。

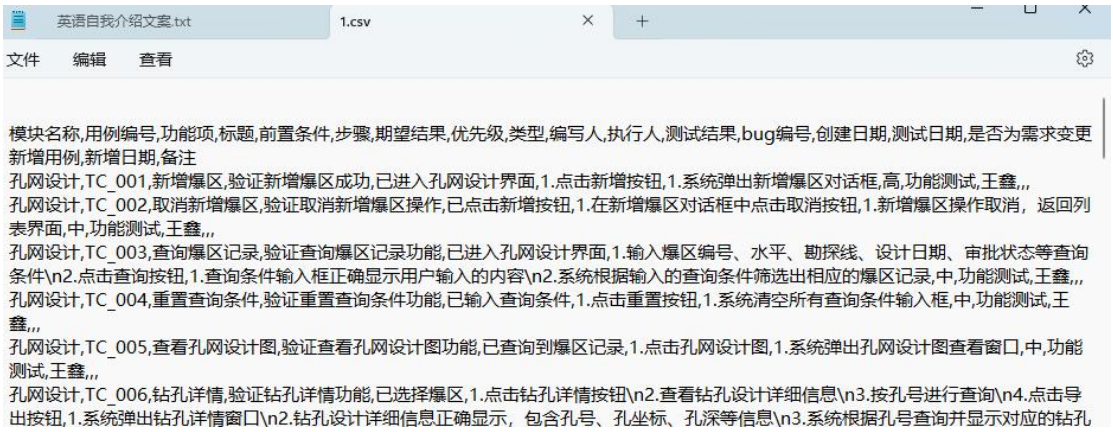
验证效果时，请确保对话中引用的知识与本地文件的内容完全一致，以确保信息的准确性和可靠性。

4、保存为 csv 格式

拷贝到文本编辑器：将 AI 工具生成的测试用例拷贝到文本编辑器（如 Notepad++ 或 VS Code）中。此时，测试用例通常是以纯文本格式呈现，每条用例包括测试步骤、输入数据和预期结果。

转换为 CSV 格式：为了方便后续导入到测试管理工具中，我们需要将文本格式的测试用例转换为 CSV 格式。另存为 csv

打开 WPS 表格，选择“数据”菜单中的“从文本/CSV”选项，导入刚刚转换好的 CSV 文件。WPS 会自动将 CSV 数据转换为表格形式。



5、总结

以上，通过 Cherry Studio 配置测试用例的智能体已经完成，期间所有操作配置一次即可完成，但关键词的优化需要通过实际测试和验证来不断调整，以达到最优效果。这一过程体现了智能体工具的灵活性和用户友好的特性，使得测试用例的生成更加高效和准确。

拓展学习

[3] 【AI 测试学习交流群】学习交流，获取 AI 智能体工具包

咨询：微信 atstudy-js 备注：AI 测试

软件测试中如何进行性能调优

◆作者： Eike

性能调优是软件测试中的重要环节，旨在提高系统的响应时间、吞吐量、并发能力、资源利用率，并降低系统崩溃或卡顿的风险。通常，性能调优涉及发现性能瓶颈、分析问题根因、优化代码和系统配置等步骤，调优之前需要先发现性能瓶颈。

一. 性能调优的流程

性能调优通常遵循以下流程，测试-调优-回归测试-线上监控：

1.性能测试：使用工具进行压力测试、负载测试，发现性能问题。

2.瓶颈分析：通过日志、监控、分析工具找出性能瓶颈，如 CPU 占用过高、数据库慢查询、线程锁等。

3.优化方案设计：结合业务需求选择合适的优化方案，如调整缓存策略、优化 SQL、代码优化等。

4.优化实施：修改代码、调整配置、升级硬件等。

5.回归测试：再次进行性能测试，确保优化方案有效，未引入新问题。

6.持续监控：使用 APM 工具持续监测性能，防止问题复发。

二. 常见性能瓶颈及调优方案

（1）CPU 相关瓶颈

常见问题

• 计算密集型任务（如加密、解密、大量数学运算）导致 CPU 飙高。

• 线程竞争、死锁导致 CPU 资源浪费。

• 不合理的线程池设置，导致线程上下文切换过多。

优化方案

• 减少 CPU 密集型计算：

o 使用缓存，避免重复计算，如 Redis 存储计算结果。

o 异步计算，将高计算量任务交给消息队列（如 Kafka、RabbitMQ）目前很多业务场景都在使用。

• 优化线程管理：

o 调整线程池参数（ corePoolSize、maxPoolSize）。

o 使用无锁编程（CAS 操作） 避免锁竞争， AtomicInteger。

• 代码优化：

o 避免不必要的对象创建，如使用对象池（Object Pool）。

o 使用高效的算法和数据结构（如 HashMap 替代 LinkedList）。

（2）内存相关瓶颈

常见问题

• 内存泄漏：对象未正确释放，导致 OOM（Out Of Memory）。

• GC（垃圾回收）频繁：GC 时间过长影响性能，如 Full GC 触发频繁。

• 大对象占用内存：如一次性加载大数据集合到内存。

优化方案

• 内存泄漏优化：

o 使用工具**Heap Dump（堆转储）**分析对象，如 VisualVM、MAT（Memory Analyzer Tool）。

o 确保不必要的对象可被 GC 回收，如关闭 Stream、Socket。

- 优化 GC 策略：
  - o 调整 JVM 参数，如 -Xms -Xmx -XX:+UseG1GC（G1 垃圾回收器）。
  - o 避免创建过多短生命周期对象，减少 GC 压力。
- 优化数据结构：
  - o Lazy Loading（懒加载）：仅在需要时加载对象，减少内存占用。
  - o 使用 LinkedList 代替 ArrayList 处理大数据集合，减少扩容开销。

(3）数据库相关瓶颈

常见问题

•慢查询：SQL 查询时间过长。 这个问题在很多业务场景中出现，当查询时间过长，返回数据较多，容易导致数据挂掉

- 数据库连接池耗尽：请求过多导致连接池被占满。
- 锁冲突：高并发写入导致数据库锁等待。

优化方案

- SQL 查询优化：
  - o 添加索引，提高查询效率（注意不要过多索引，影响写入性能）。
  - o 使用分页查询，避免一次性查询大量数据：

```
SELECT * FROM users LIMIT 100 OFFSET 200;
```

- 数据库架构优化：
  - o 读写分离：使用主从数据库，分担查询压力。
  - o 分库分表：拆分大表，提高查询速度。
- 数据库连接优化：
  - o 使用 连接池（HikariCP、Druid），减少数据库连接创建开销。
  - o SQL 预编译，减少解析和优化开销：

```
PreparedStatement ps = conn.prepareStatement("SELECT * FROM users WHERE id = ?");
```

（4）磁盘 I/O 相关瓶颈

常见问题

- 日志文件过大，写入过频繁，影响性能。
- 数据库磁盘 I/O 过载，如索引未优化导致大量磁盘扫描。
- 读取大文件时，未使用流式读取导致内存溢出。

优化方案

- 日志优化：
  - o 采用异步日志框架（如 Logback、Log4j2 AsyncAppender）。
  - o 设置日志级别，减少 DEBUG 级别日志的写入。
- 文件 I/O 优化：
  - o 使用流式读取：

```
try (BufferedReader br = new BufferedReader(new FileReader("largefile.txt"))) {
    String line;
    while ((line = br.readLine()) != null) {
        process(line);
    }
}
```

  - o 采用 内存映射文件（MappedByteBuffer） 读取大文件，提高读取性能。

（5）网络相关瓶颈

常见问题

- HTTP 请求过多，导致带宽耗尽。
- 高并发请求导致服务器连接数不足。
- 频繁查询远程 API，导致响应变慢。

优化方案

- 减少 HTTP 请求：
  - 启用 Keep-Alive，减少 TCP 连接建立次数。
  - 使用 CDN 缓存静态资源，减少服务器压力。
- 优化 API 调用：
  - 采用本地缓存，减少重复请求，如 Guava Cache：

```
Cache<String, String> cache = CacheBuilder.newBuilder().expireAfterWrite(10, TimeUnit.MINUTES).build();
```

  - 使用 批量请求 代替多次小请求：

```
SELECT * FROM users WHERE id IN (1, 2, 3, 4);
```

三. 持续性能优化

1.自动化性能测试：

使用 JMeter、Gatling 进行自动化性能测试。

2.性能监控：

- 结合 Prometheus + Grafana 监控系统资源。
- 采用 APM 工具（如 New Relic、SkyWalking）分析慢请求。

3.持续优化：

在 CI/CD 中加入性能测试，每次代码变更后进行回归测试。

四. 常用性能调优工具

1. 性能测试工具

（1）负载/压力测试

用于模拟大量用户请求，测试系统在高负载下的性能表现。

工具	介绍	适用场景
Apache JMeter	开源性能测试工具，可用于 HTTP、WebSocket、数据库等压力测试	Web 应用、API 测试
LoadRunner	企业级性能测试工具，支持多协议，提供详细分析	大型企业系统
Gatling	高性能开源负载测试工具，基于 Scala，支持实时报告	API、大型 Web 应用
k6	现代化负载测试工具，基于 JavaScript，轻量级，支持云端运行	DevOps 流水线中的性能测试
wrk	超轻量级 HTTP 性能测试工具，支持 Lua 脚本扩展	Web 服务高并发测试
Locust	Python 编写的分布式压力测试工具，易于扩展	API、大规模 Web 应用

2. 监控与分析工具

（1）服务器监控

这些工具用于监控 CPU、内存、磁盘 I/O、网络 等系统资源。

工具	介绍	适用场景
Prometheus + Grafana	监控指标采集 + 数据可视化	服务器、微服务监控
New Relic	云端 APM 监控，支持代码级性能分析	Web 应用、云原生系统
Zabbix	开源企业级监控工具，支持分布式架构	服务器集群监控
Nagios	监控 IT 基础设施，支持警报	数据中心、IT 运维
Glances	命令行实时监控工具，轻量级	服务器端实时监控

（2）应用性能监控（APM，Application Performance Monitoring）

用于监测应用程序的响应时间、调用链、数据库查询等。

工具	介绍	适用场景
SkyWalking	开源 APM 监控工具，支持 Java、Go、Python	微服务、云原生环境
Zipkin	分布式追踪工具，分析微服务调用链	微服务架构
Jaeger	CNCF 维护的分布式跟踪工具	复杂服务链路跟踪
AppDynamics	商业 APM 工具，可视化应用性能	大型企业应用

3. 数据库性能优化工具

（1）数据库查询分析

这些工具帮助分析 SQL 语句的执行情况，优化数据库查询。

工具	介绍	适用数据库
EXPLAIN / EXPLAIN ANALYZE	MySQL、PostgreSQL 自带的 SQL 执行计划分析工具	MySQL、PostgreSQL
Slow Query Log（慢查询日志）	记录 SQL 运行时间超长的查询	MySQL
pg_stat_statements	PostgreSQL 记录 SQL 执行统计	PostgreSQL
AWR（Automatic Workload Repository）	Oracle 数据库的性能分析报告	Oracle
SQL Profiler	SQL Server 自带的 SQL 监控工具	SQL Server
DBeaver	可视化数据库管理工具，支持 SQL 分析	多种数据库

（2）数据库连接池优化

数据库连接池用于管理数据库连接，减少连接创建开销，提高并发性能。

工具	介绍	适用数据库
HikariCP	高性能数据库连接池，速度快、资源占用少	MySQL、PostgreSQL
Druid	阿里巴巴开源数据库连接池，支持 SQL 监控	MySQL、Oracle
C3P0	传统数据库连接池，支持自动回收连接	MySQL、PostgreSQL

4. JVM（Java 应用）性能分析工具

用于分析 JVM 内存使用、GC（垃圾回收）、线程状态 等，优化 Java 应用的性能。

工具	介绍	适用场景
JConsole	Java 自带的监控工具，可查看线程、GC、内存	本地 Java 应用
VisualVM	可视化 JVM 监控，支持 Heap Dump 分析	Java 内存分析
GCViewer	可视化 GC 日志，分析垃圾回收情况	JVM GC 调优
MAT（Memory Analyzer Tool）	Eclipse 提供的内存分析工具，检测内存泄漏	分析 Heap Dump
Async Profiler	轻量级 Java Profiler，分析 CPU、内存、锁竞争	高性能 Java 应用
Arthas	阿里开源的 Java 诊断工具，可分析线程堆栈、JVM 变量	线上 Java 问题排查

5. 磁盘 I/O 优化工具

用于检测和优化文件读写、磁盘吞吐量等。

工具	介绍	适用场景
iostat	Linux 监控磁盘 I/O	服务器磁盘性能分析
fio	文件系统和磁盘 IO 压力测试工具	存储系统测试
iotop	监控进程的磁盘读写	Linux 服务器
blktrace	分析块设备 I/O	磁盘性能调优

6. 网络优化工具

用于检测网络性能、带宽占用、延迟等。

工具	介绍	适用场景
ping	测试网络连通性	基础网络排查
traceroute	跟踪数据包的路径	分析网络延迟
netstat	查看网络连接、端口占用	服务器网络排查
Wireshark	抓包工具，分析网络流量	网络协议分析
tcpdump	命令行抓包工具	网络流量分析

7. 前端性能调优工具

用于分析 Web 页面性能，包括 加载时间、渲染速度、资源优化。

工具	介绍	适用场景
Google Lighthouse	Chrome 开发者工具，分析网页性能	Web 应用优化
WebPageTest	在线页面加载速度测试	Web 性能监控
Chrome DevTools	浏览器开发者工具，支持 JS Profiling	前端调试
GTmetrix	网站性能分析，提供优化建议	网站优化

以下为常用的性能调优工具汇总：

- 压力测试：JMeter、LoadRunner、Gatling
- 服务器监控：Prometheus + Grafana、Zabbix
- APM 监控：SkyWalking、New Relic、Jaeger
- 数据库优化：EXPLAIN、Slow Query Log、HikariCP

• JVM 调优：VisualVM、MAT、Async Profiler

• 磁盘/网络调优：iostat、Wireshark、tcpdump

不同的性能调优工具适用于不同场景，结合自身的业务场景，选择合适的工具可以更高效地发现并解决性能问题。

拓展学习

[4] 【性能测试学习交流群】学习交流

咨询：微信 atstudy-js 备注：学习群

普通 APP 和鸿蒙 APP 测试分析

◆ 作者：启航

前言：

普通 APP 的测试与鸿蒙 APP 的测试有一些共同的特征，但是也有一些区别，其中共同特征是，它们都可以通过 cmd 的命令提示符工具来进行 app 的性能测试。

其中区别主要是，对于稳定性测试的命令的区别，性能指标获取方式的命令的区别，安装的命令，卸载的命令，等等。此外，鸿蒙 APP 还有一个专项测试。

这篇文章就将普通 APP 的测试方法以及鸿蒙 APP 的测试的方法进行一定程度的梳理。希望能从中发现一些共通点。

这篇文章作为鸿蒙 APP 以及安卓 APP 的第一篇总结性文章，将带着以下问题进行展开：

1：什么是 hdc？什么是 adb？

2：adb 的 shell 命令可以用在鸿蒙 hdc 的 shell 上面吗？

3：adb 的性能测试的日志文件与 hdc 的性能测试的日志一样吗？

对于五类性能测试的方法及其 log 的主要分析。这五类的方法分别是： hdc 稳定性测试，hdc 性能测试，hdc 专项测试，adb 稳定性测试，adb 性能测试。

4：鸿蒙的包名的格式和安卓的包名的格式一样吗？

带着以上的疑问我将写一篇相关的文章，同时，此次测试的鸿蒙官方文档的出处以及测试的心得也将列出来。

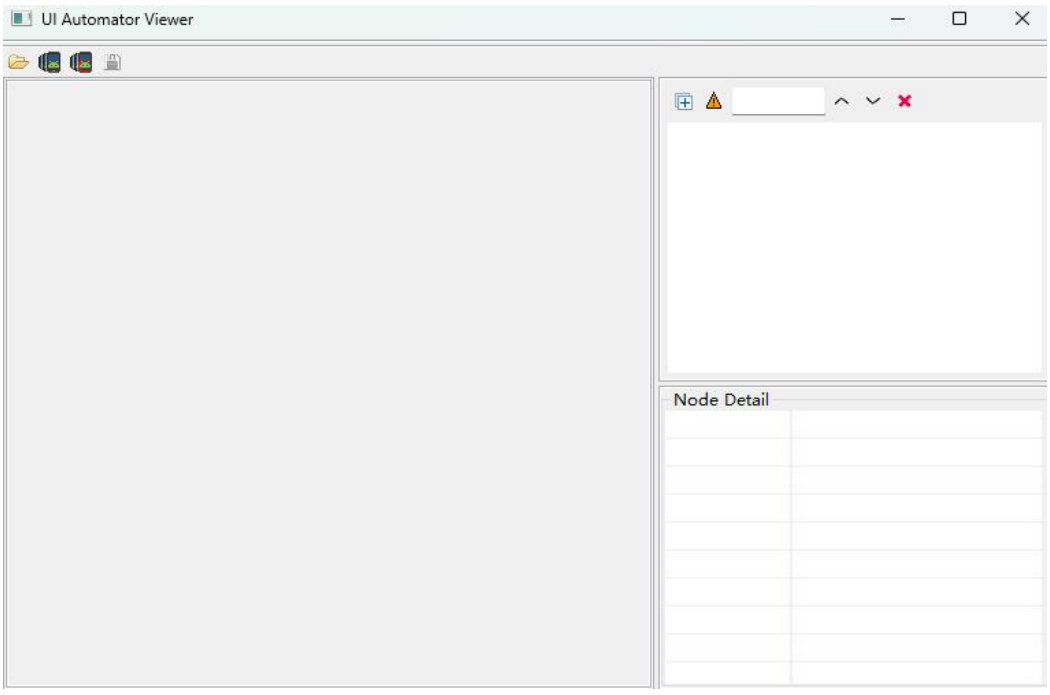
本文很多的描述都是来自 <文心快码>等AI搜索生成。

第一部分 普通 APP 的测试详细步骤

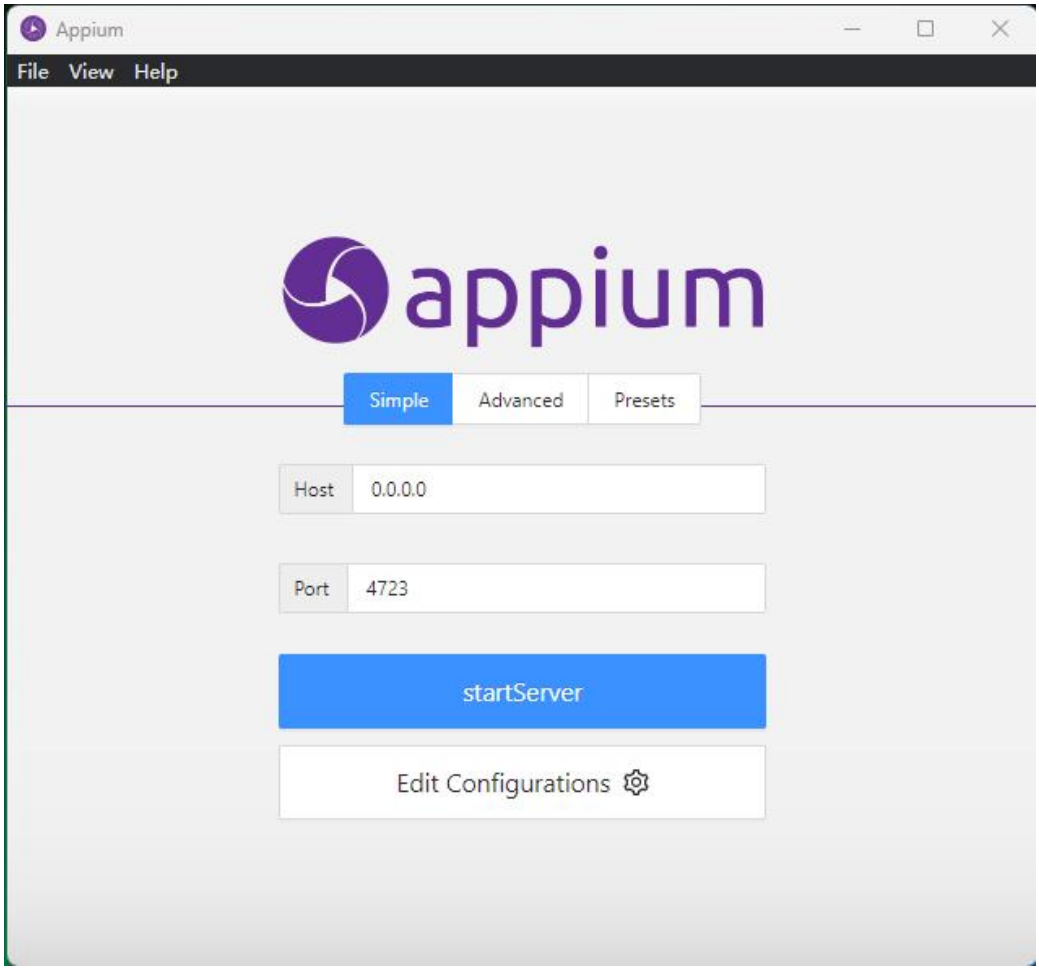
第一步 打开 uiautomatorviewer.bat 作为元素定位的工具：

查看快捷方式的指向

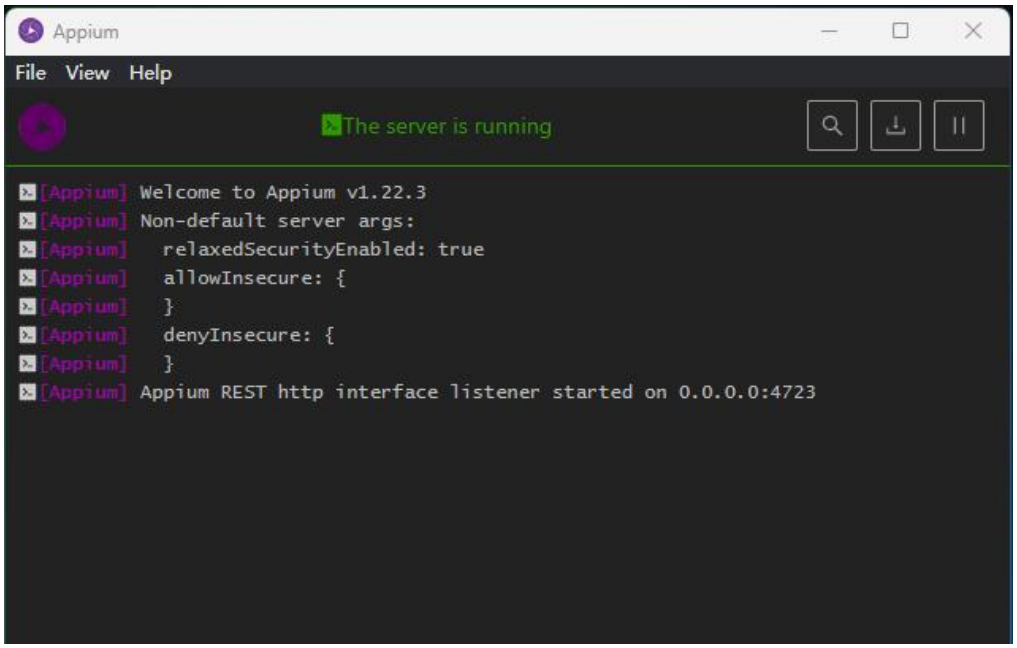
C:\adt-bundle-windows-x86_64-20140702\adt-bundle-windows-x86_64-20140702\sdk\tools\uiautomatorviewer.bat



第二步 打开 APPIUM



第三步 点击 **“Start Server”**



第四步 打开模拟器

第五步 打开 CMD 运行 adb devices 查看连接设备的情况，此时显示没有匹配的客户端

```
Microsoft Windows [版本 10.0.26100.2894]
(c) Microsoft Corporation。保留所有权利。

[redacted]>adb devices
adb server version (31) doesn't match this client (41); killing...
* daemon started successfully
List of devices attached
```

第六步 运行 adb connect 127.0.0.1:7555

连接到模拟器：在命令行中输入 adb connect 127.0.0.1:7555。

127.0.0.1 代表本地计算机，而 7555 是模拟器的端口号。

```
C:\Users\[redacted]>adb connect 127.0.0.1:7555
connected to 127.0.0.1:7555
```

第七步 再次运行 adb devices

```
C:\Users\[redacted]>adb devices
List of devices attached
127.0.0.1:7555  device
```

第八步 使用 pycharm 工具运行最基本的 python 命令查看是否可以正常启动 APPIUM

```
from appium.webdriver import Remote
import time



caps = {
    "platformName":"Android",
    "udid":"127.0.0.1:7555",
    "appPackage":" name of package  ",
    "appActivity":"  name of activity          "
}
driver=Remote(desired_capabilities=caps,
              command_executor='http://127.0.0.1:4723/wd/hub')
```

第九步 常用的 adb 命令的演示

1）包名查询

获取当前焦点窗口的信息： adb shell dumpsys window | findstr mCurrentFocus

```
C:\Users\      >adb shell dumpsys window | findstr mCurrentFocus
  mCurrentFocus=Window{7719d0d u0 com.mumu.launcher/com.mumu.launcher.Launcher}
```

列出所有已安装的应用程序包： adb shell pm list packages

adb shell pm list packages 命令还可以使用一些选项。例如：

-f 选项：列出应用程序包的完整路径。

-3 选项：仅列出第三方（非系统）应用程序包。

```
C:\Users\      >adb shell pm list packages
package:com.android.internal.display.cutout.emulation.corner
package:com.android.internal.display.cutout.emulation.double
package:com.android.providers.telephony
package:io.appium.settings
package:com.android.internal.systemui.navbar.gestural_wide_back
package:com.android.documentsui
package:com.android.externalstorage
package:com.android.providers.downloads
package:com.android.internal.systemui.onehanded.gestural
package:com.android.browser
package:io.appium.uiautomator2.server
package:com.android.networkstack
package:com.android.connectivity.resources
package:com.android.internal.display.cutout.emulation.hole
package:com.android.internal.display.cutout.emulation.tall
package:com.android.modulemetadata
package:com.android.certinstaller
package:com.android.internal.systemui.navbar.threebutton
package:android
package:com.android.camera2
package:com.android.internal.systemui.navbar.gestural_extra_wide_back
```

2）日志收集之： adb logcat

adb logcat 命令的主要作用是捕获和显示 Android 设备的日志信息。这些日志信息包括系统日志、应用日志、错误日志等 。

app 中任何操作的日志情况都会在 adb logcat 的日志中显示。

比如打开 “设置”组件等操作，都会实时地更新 adb logcat 的日志信息。

```
03-01 14:28:14.896  1287  1316 I ActivityManager: Start proc 3857:com.android.settings/1000 for pre-top-activity {com.an
droid.settings/com.android.settings.Settings}
```

```
03-01 14:30:16.271  1287  1316 I ActivityManager: Start proc 3982:com.android.settings/1000 for pre-top-activity {com.an
droid.settings/com.android.settings.Settings}
```

3）adb 的文件传输命令

• adb push

将本地的文件传送到模拟器中

adb push C:\app\test.apk /sdcard/

```
C:\Users\      >adb push C:\app\test.apk /sdcard/
C:\app\test.apk: 1 file pushed, 0 skipped.
```

• adb pull

将模拟器中的文件传送到本地

adb pull /sdcard/test.apk C:\app\

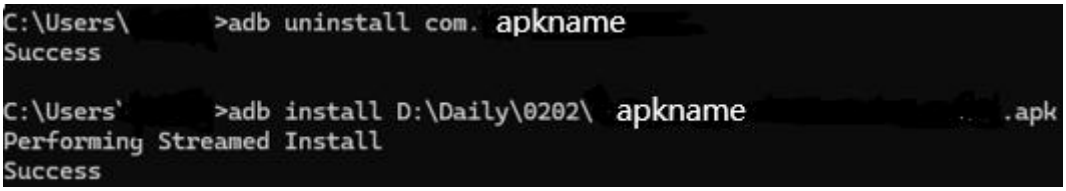
4）adb 的安装 卸载命令

卸载软件

adb uninstall <name of apk with com>

安装软件

adb install apkname.apk



5）启动 app，停止app

adb shell am start <name of apk with com/Activity>

adb shell am force-stop <name of apk with com>

6）稳定性测试- 点击 600 次 并保存到某个 log 中

adb shell monkey -p <name of apk with com> -v -v 600 命令是 Android 开发中常用的一条命令行指令，它主要用于对指定的 Android 应用进行压力测试。

解释如下:

• adb shell:用于在连接的 Android 设备上执行 shell 命令。

• monkey:用于向系统发送伪随机的用户输入事件流（如点击、触摸、手势等）

• -p <name of apk with com>：这个选项指定了要测试的应用的包名。

• -v -v：设置日志的详细程度。 -v -v 则表示启用比-v 更详细的日志输出。

• 600：这个数字表示 monkey 工具将发送的事件的数量。在这个例子中，monkey 工具将向<name of apk with com>应用发送 600 个伪随机的用户输入事件。

检查日志中是否有异常的关键词，并提取相关日志给开发。

常见的关键字：

1.ANR 问题（应用无响应）?：

关键词：ANR、anr in

2.闪退问题：

关键词：crash

3.异常：

关键词：exception

4.强制退出：

关键词：force closed

其他常见异常类型：

• 算术异常：ArithmeticException

• 空指针异常：NullPointerException

• 类型强制转换异常：ClassCastException

• 数组负下标异常：NegativeArrayException

• 数组下标越界异常：ArrayIndexOutOfBoundsException

• 违背安全原则异常：SecurityException

• 文件已结束异常：EOFException

• 文件未找到异常：FileNotFoundException

• 字符串转换为数字异常：NumberFormatException

- 操作数据库异常：SQLException
- 输入输出异常：IOException
- 违法访问错误：IllegalAccessError
- 内存不足错误：OutOfMemoryError
- 堆栈溢出错误：StackOverflowError

7）稳定性测试 让 -v 的次数变动查看 log 的变化

--pct-touch <percent>指定了触摸事件在所有生成的事件中所占的百分比。例如，如果你设置--pct-touch 87，那么在执行测试时，大约 87%的事件将是触摸事件。

以下命令是 pct-touch 的概率是百分之百，发送 600 个伪随机的用户输入事件，间隔时间为 300 毫秒。

adb shell monkey -p <name of apk with com> --pct-touch 100 --throttle 300 -v 600 > C:\Users\test\Desktop\0228\v\-v\v.log

adb shell monkey -p <name of apk with com> --pct-touch 100 --throttle 300 -v -v 600 > C:\Users\test\Desktop\0228\v\-v-v\vv.log

比对 -v 的 log 与 -v -v 的 log 之间的区别：

目前发现 -v-v 的日志中含有：

Sleeping for 300 milliseconds

但是 -v 的日志中是没有 Sleeping for 300 milliseconds 的

```
Sleeping for 300 milliseconds
:Sending Touch (ACTION_DOWN): 0:(90.0,583.0)
:Sending Touch (ACTION_UP): 0:(94.72916,604.22437)
```

除了--pct-touch 之外，还有，--pct-motion<percent>(滑动事件生成百分比)等参数。

8）性能测试之内存查询

adb shell dumpsys meminfo 命令用于获取 Android 设备上指定进程的内存使用情况。

当你附加一个应用包名（如 name of apk with com）作为参数时，它会返回该应用的内存使用信息。

```
C:\Users\      >adb shell dumpsys meminfo com.android.settings
Applications Memory Usage (in Kilobytes):
Uptime: 6563870 Realtime: 6563870

** MEMINFO in pid 3982 [com.android.settings] **
                   Pss  Private  Private     Swap      Rss     Heap     Heap     Heap
                 Total    Dirty    Clean    Dirty    Total     Size    Alloc     Free
                ------   ------   ------   ------   ------   ------   ------   ------
  Native Heap    13893    13848        0        0    14740    22916    10295     7911
  Dalvik Heap    16018    15924        0        0    17412    19233     9617     9616
 Dalvik Other     2035     1912        0        0     3744
        Stack      548      548        0        0      556
       Ashmem       24        0        0        0      332
    Other dev       32        0       32        0      308
     .so mmap     8130      812     3916        0    36456
    .jar mmap     6399        0     1952        0    29828
    .apk mmap    14030        0    13596        0    15328
    .ttf mmap       76        0       20        0      180
    .dex mmap      232        4      204        0      608
    .oat mmap     2719        0     1080        0    14628
    .art mmap     2902     1856        0        0    19828
   Other mmap     2405        8      812        0     5628
      Unknown      637      632        0        0      920
        TOTAL    70080    35544    21612        0   160496    42149    19912    17527
```

关键指标

•PSS (Proportional Set Size): 表示进程独占的内存页的数量，考虑到共享库的内存占用。

• RSS (Resident Set Size): 表示进程实际占用的物理内存大小。

•OutOfMemoryError: 如果应用显示内存不足错误，需要进一步分析是哪个部分的内存使用过高。

如果要查询系统的内存信息则是

adb shell cat /proc/meminfo

9）性能测试之 cpu 查询

adb shell dumpsys cpuinfo

可以获取当前 CPU 的使用情况，帮助分析 CPU 的性能瓶颈?，找出占用资源较多的进程进行优化。

对于以下的命令输出的详细解释如下：

1.7% 1590/m.mumu.launcher: 1.3% user + 0.3% kernel / faults: 5056 minor 28 major

这段信息是从 adb shell dumpsys cpuinfo 命令的输出中提取的，它提供了关于特定进程 CPU 使用情况的详细信息。下面是对这段信息的详细解释：

1.7%：这是进程 m.mumu.launcher 的总体 CPU 使用率。它表示该进程在当前采样时间范围内占用了 CPU 总时间的 1.7%。这个数值是用户态使用率和内核态使用率之和的一个总体反映。

1590/m.mumu.launcher：这里的 1590 是进程的 ID（PID），而 m.mumu.launcher 是进程名。这个信息告诉你哪个进程正在使用 CPU 资源。

1.3% user：这是进程在用户态下的 CPU 使用率。用户态是指进程执行用户级代码时的状态，如应用程序的逻辑处理部分。1.3%的用户态使用率表示进程在执行用户级代码时占用了 CPU 总时间的 1.3%。

0.3% kernel：这是进程在内核态下的 CPU 使用率。内核态是指进程执行内核级代码时的状态，如系统调用、中断处理等。0.3%的内核态使用率表示进程在执行内核级代码时占用了 CPU 总时间的 0.3%。

faults: 5056 minor 28 major：这里的 faults 表示进程在执行过程中遇到的页错误（page faults）数量。页错误是指进程试图访问其虚拟内存空间中尚未映射到物理内存的地址时发生的情况。5056 minor 表示发生了 5056 次轻微页错误，这通常是由于进程的工作集（working set）超出了物理内存的大小，但操作系统能够通过将部分内存页换出到磁盘来解决这些问题，而不会导致进程被阻塞。28 major 表示发生了 28 次主要页错误，这通常是由于进程试图访问已被换出到磁盘的内存页，导致操作系统必须等待磁盘 I/O 操作完成才能继续执行进程，这可能会导致性能下降。

对于性能分析来说，这些信息非常有用：

•总体 CPU 使用率可以帮助你了解进程是否占用了过多的 CPU 资源，从而可能影响到系统的整体性能。

• 用户态和内核态的使用率可以帮助你判断进程的性能瓶颈是在用户级代码还是内核级代码上。

• 页错误数量，特别是主要页错误数量，可以帮助你了解进程的内存使用情况，以及是否存在内存不足或内存访问模式不佳导致的性能问题。

综上所述，这段信息提供了关于进程 CPU 使用情况的详细信息，对于性能分析和优化非常有帮助。

<以上解释参考自文心快码>

如果要查询系统的 cpu 信息用的命令是

adb shell cat /proc/cpuinfo

10）性能测试之 top 命令

top： adb shell top | findstr <key word of process name>

此命令可以实时刷新 top 的数值内容：

```
C:\Users\[redacted]>adb shell top | findstr com.android.set
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
 4293 system      10 -10  12G 145M  92M S  0.0   3.6   0:00.36 com.android.set+
```

以下是 top 命令的详细分析

对于以下的命令输出的详细解释如下：

2374 u0_a41 10 -10 28G 203M 130M S 3.8 5.1 0:24.27 com.test +

这段文本提供了关于 Android 系统中一个进程的详细信息。下面是对各个字段的具体解释：

• 2374：这是进程的 ID（PID），它是系统为每个进程分配的唯一标识符。通过这个 ID，你可以对进程进行各种操作，如查看详细信息、终止进程等。

• u0_a41：这是进程的用户 ID。在 Android 系统中，每个应用和用户都有对应的用户 ID。这里的 u0 通常表示第一个用户（即主用户），a41 是该用户下的应用 ID。这个 ID 用于系统权限控制和资源隔离。

• 10 -10：这两个数字通常表示进程的优先级和调度策略。不过，在这个上下文中，它们的具体含义可能因 Android 版本和厂商定制而有所不同。一般来说，较高的数字可能表示较高的优先级。但这里的-10 可能是一个特殊的调度策略或优先级调整值。

• 28G 203M 130M：这些数字表示进程的内存使用情况。28G 可能是指系统总的可用内存（尽管这个值在这里可能不太准确，因为通常进程信息中不会显示系统总内存），203M 表示进程当前使用的内存大小，130M 可能表示进程使用的某种特定类型的内存（如 Dalvik 堆内存、Native 堆内存等）的大小。不过，具体的内存类型需要根据 Android 版本和厂商定制来确定。

注意：这里的 28G 很可能是一个误解或错误，因为单个进程的内存使用情况通常不会以整个系统的内存总量来表示。更可能的是，这里显示的是某种内存统计信息的格式错误或混淆。

• S：这是进程的状态码。在 Android 系统中，进程状态码通常表示进程的当前状态，如 R（运行中）、S（睡眠中）、D（不可中断的睡眠状态）、T（跟踪/停止状态）等。S 状态表示进程正在睡眠中，等待某个事件或资源。

• 3.8 5.1：第一个数字（3.8%）可能是指进程在过去某个时间窗口内的用户态 CPU 使用率，第二个数字（5.1%）是某个进程的内存利用率

• 0:24.27：这表示进程自启动以来已经占用的 CPU 时间，格式为小时:分钟.秒。在这个例子中，进程已经占用了大约 0 小时 24 分钟 27 秒的 CPU 时间。

com.test：这是进程的名称，通常与应用的包名相对应。在这个例子中，进程名称是 com.test，

综上所述，这段文本提供了关于 Android 系统中一个进程的详细信息，包括进程 ID、用户 ID、内存使用、CPU 使用率、进程状态等。这些信息对于理解进程的当前状态、进行性能分析或故障排查非常有帮助。不过，请注意，某些字段的具体含义可能因 Android 版本和厂商定制而有所不同。

<以上解释参考自文心快码>

11）性能测试之流量查询

以下是运用 adb 进行流量测试的详细步骤：

一、获取应用的进程号

首先，你需要获取目标应用的进程 ID（pid）。这可以通过以下命令实现：

adb shell ps | findstr <key word of process name>

```
C:\Users\      >adb shell ps | findstr set
root           860     2       0      0 0                   0 I [nvme-reset-wq]
u0_a38        2656  1120 12529220 101236 0                   0 S io.appium.settings
system        4293  1120 12556644 149684 0                   0 S com.android.settings
```

获得 com.android.settings 的进程号为 4293。

二、查看应用流量情况

使用上一步的进程号来查看该应用的流量使用情况：

adb shell cat /proc/<pid number>/net/dev

执行该命令后，你将看到与该应用相关的网络流量数据。这些数据包括接收（Recv）和发送（Transmit）的字节数、包数、错误数和丢弃数等。

```
C:\Users\      >adb shell cat /proc/4293/net/dev
Inter-|   Receive                                                |  Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs drop fifo colls carrier compressed
ip6tnl0:       0       0    0    0    0     0          0         0       0       0    0    0    0     0       0
0
 wlan0: 4705313   21237    0    0    0     0          0         0 11484560   11960    0    0    0     0       0          0
  sit0:       0       0    0    0    0     0          0         0        0       0    0    0    0     0       0          0
    lo:   16740      84    0    0    0     0          0         0    16740      84    0    0    0     0       0          0
```

Interface：网络接口的名称。

Receive（接收）：

- bytes：接收的总字节数。
- packets：接收的总数据包数。
- errs：接收过程中发生的错误数。
- drop：接收过程中丢弃的数据包数。

Transmit（发送）：

- bytes：发送的总字节数。
- packets：发送的总数据包数。
- errs：发送过程中发生的错误数。

• drop：发送过程中丢弃的数据包数。

ip6tnl0 是 Linux 系统中用于 IPv6 隧道的网络接口

wlan0 代表设备上的第一个无线网络接口，它主要负责 Wi-Fi 通信

sit0 是一个在 Linux 系统中出现的特殊网络接口，它属于 sit（Simple Internet Transition）设备的一种。

lo 代表本地回环接口（Loopback Interface）

以下命令可以用来获取更详细的网络流量统计信息：

adb shell dumpsys netstats detail

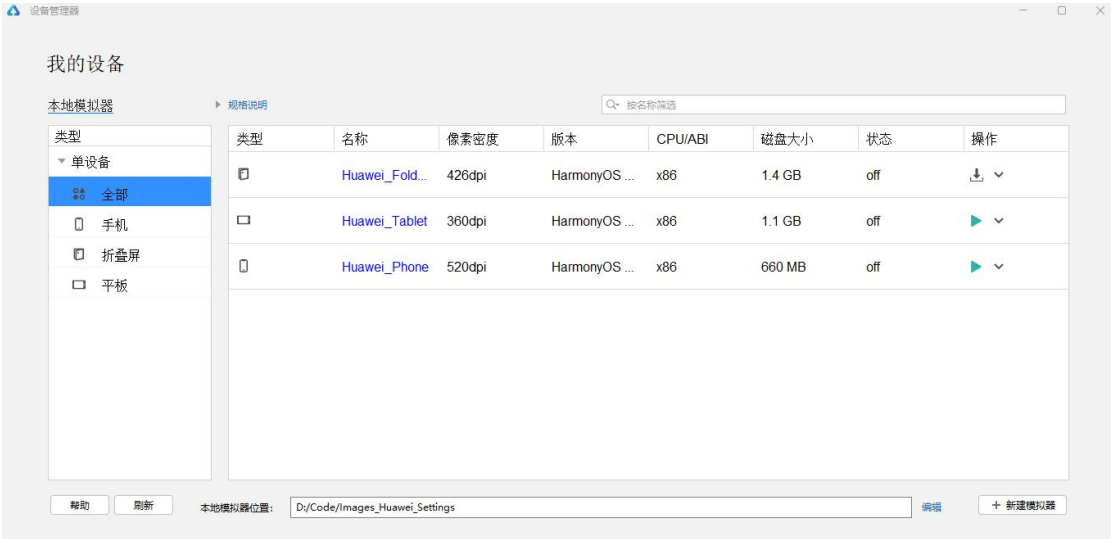
三、计算使用流量

比较两次连续操作性能测试流量数值的区别，从而得到某一段时间内的流量使用情况。经过研究，鸿蒙是不支持 dumpsys 的。

第二部分 鸿蒙 APP 的测试详细步骤

第一步 开启 DevEco Studio

第二步 选择 “设备管理器”



第三步 在模拟器中启动该设备并打开模拟器

第四步 打开命令提示符 输入 hdc list targets

如果显示的是 127.0.0.1:5555 则说明是查询到了已经连接的所有目标设备为本地的 127.0.0.1:5555

```
C:\Users\      >hdc list targets
127.0.0.1:5555
```

添加-v 参数，则会打印设备详细信息

```
C:\Users\      >hdc list targets -v
127.0.0.1:5555          TCP     Connected       localhost       hdc
```

<以下是参考出处 ：

https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V13/hdc-V13>

list targets	查询已连接的所有目标设备。

第五步 使用 pycharm 运行自动化测试脚本

验证是否可以使用 pycharm 工具进行自动化测试的运行。

第六步 使用 hdc 命令进行稳定性随机测试及其日志分析

稳定性随机测试就是设置参数，并且让程序随机的启动程序，随机的点击，并且设置执行次数的方法，同时也设置了启动程序的时间间隔。

在执行完成之后，可以根据 log 中保留的日志情况，查看日志中的异常数据，如果含有异常数据，则提交异常数据给开发进行修复。

这种稳定性测试是随机测试，任何应用都可能拉起的，因此它并没有指定 APP 的名称。而后面介绍的专项测试则指定了应用的名称。

wukong exec 命令含义

-s 参数设置随机种子

-i 参数设置应用拉起间隔

-a 参数设置应用随机拉起测试比例

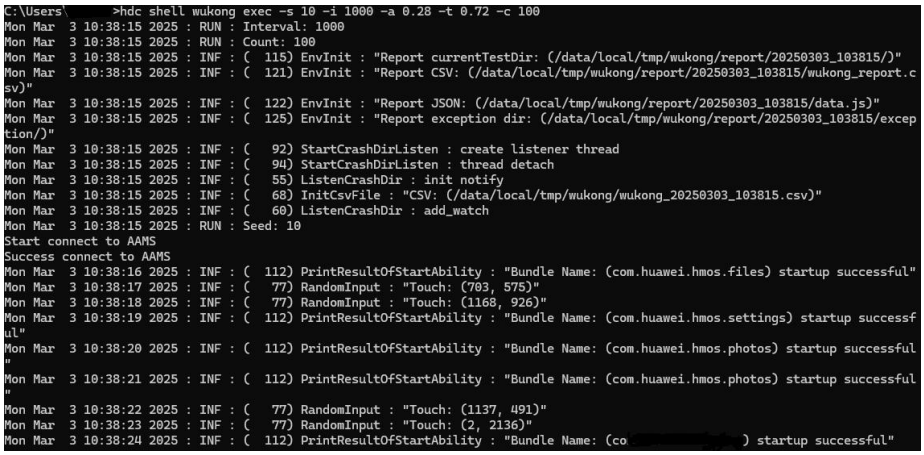
-t 参数设置屏幕随机 touch 测试比例

-c 参数设置执行的次数

比如命令为：

```
hdc shell wukong exec -s 10 -i 1000 -a 0.28 -t 0.72 -c 100
```

<LOG 存放路径自动显示在日志中>



```
C:\Users\      >hdc shell wukong exec -s 10 -i 1000 -a 0.28 -t 0.72 -c 100
Mon Mar  3 10:38:15 2025 : RUN : Interval: 1000
Mon Mar  3 10:38:15 2025 : RUN : Count: 100
Mon Mar  3 10:38:15 2025 : INF : (  115) EnvInit : "Report currentTestDir: (/data/local/tmp/wukong/report/20250303_103815/)"
Mon Mar  3 10:38:15 2025 : INF : (  121) EnvInit : "Report CSV: (/data/local/tmp/wukong/report/20250303_103815/wukong_report.c
sv)"
Mon Mar  3 10:38:15 2025 : INF : (  122) EnvInit : "Report JSON: (/data/local/tmp/wukong/report/20250303_103815/data.js)"
Mon Mar  3 10:38:15 2025 : INF : (  125) EnvInit : "Report exception dir: (/data/local/tmp/wukong/report/20250303_103815/excep
tion/)"
Mon Mar  3 10:38:15 2025 : INF : (   92) StartCrashDirListen : create listener thread
Mon Mar  3 10:38:15 2025 : INF : (   94) StartCrashDirListen : thread detach
Mon Mar  3 10:38:15 2025 : INF : (   55) ListenCrashDir : init notify
Mon Mar  3 10:38:15 2025 : INF : (   68) InitCsvFile : "CSV: (/data/local/tmp/wukong/wukong_20250303_103815.csv)"
Mon Mar  3 10:38:15 2025 : INF : (   60) ListenCrashDir : add_watch
Mon Mar  3 10:38:15 2025 : RUN : Seed: 10
Start connect to AAMS
Success connect to AAMS
Mon Mar  3 10:38:16 2025 : INF : (  112) PrintResultOfStartAbility : "Bundle Name: (com.huawei.hmos.files) startup successful"
Mon Mar  3 10:38:17 2025 : INF : (   77) RandomInput : "Touch: (703, 575)"
Mon Mar  3 10:38:18 2025 : INF : (   77) RandomInput : "Touch: (1168, 926)"
Mon Mar  3 10:38:19 2025 : INF : (  112) PrintResultOfStartAbility : "Bundle Name: (com.huawei.hmos.settings) startup successf
ul"
Mon Mar  3 10:38:20 2025 : INF : (  112) PrintResultOfStartAbility : "Bundle Name: (com.huawei.hmos.photos) startup successful
"
Mon Mar  3 10:38:21 2025 : INF : (  112) PrintResultOfStartAbility : "Bundle Name: (com.huawei.hmos.photos) startup successful
"
Mon Mar  3 10:38:22 2025 : INF : (   77) RandomInput : "Touch: (1137, 491)"
Mon Mar  3 10:38:23 2025 : INF : (   77) RandomInput : "Touch: (2, 2136)"
Mon Mar  3 10:38:24 2025 : INF : (  112) PrintResultOfStartAbility : "Bundle Name: (co              ) startup successful"
```

<以下是参考出处 ：

https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V13/wukong-guidelines-V13 >

- 设置100次事件注入

```
# wukong exec -s 10 -i 1000 -a 0.28 -t 0.72 -c 100
```

命令中各参数含义：

命令	参数值	说明
wukong exec	-	主命令。
-s	10	参数设置随机种子，10为种子值。
-i	1000	参数设置应用拉起间隔为1000ms。
-a	0.28	参数设置应用随机拉起测试比例28%。
-t	0.72	参数设置屏幕随机touch测试比例为72%。
-c	100	参数设置执行次数为100次。

第七步 使用 hdc 命令进行性能测试 SP_Daemon 及其日志分析

性能测试通过使用不同的性能测试的指标，来查看系统的性能情况，

包括 FPS、CPU、GPU、RAM、Temp 等等性能指标

常用命令：

hdc shell SP_daemon -N 2 -t

解释 ：

-N 设置采集次数，一秒采集一次

-t 采集 GPU 温度，系统芯片温度

<以下是参考出处 ：

https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V13/smartperf-guidelines-V13 >

命令参数	必选	说明
-N	是	设置采集次数，一秒采集一次
-PKG	否	设置包名
-c	否	采集cpu的频点和使用率。 设置应用包名时，采集整机和应用CPU信息 不设置应用包名时，采集整机CPU信息
-g	否	采集gpu的频点和负载信息
-f	否	采集指定应用的fps以及屏幕刷新率，必须设置应用包名
-t	否	采集GPU温度、系统芯片温度
-r	否	采集内存。 设置应用包名时，采集整机和应用内存信息 不设置应用包名时，采集整机内存信息
-snapshot	否	屏幕截图
-net	否	采集网络速率
-VIEW	否	设置图层，需要先获取应用图层名
-d	否	采集DDR
-sections	否	设置分段采集
-nav	否	采集页面导航信息，必须设置应用包名

指标说明

- CPU：每秒读取一次设备节点下CPU大中小核的频点和各核使用率，衡量应用占用CPU资源的情况，占用过多的CPU资源会导致芯片发烫。
- GPU：每秒读取一次设备节点下GPU的频点和负载信息，衡量应用占用GPU资源的情况，当GPU占用过多时，会导致性能下降，应用程序的运行速度变慢。
- FPS：应用界面每秒刷新次数，衡量应用画面的流畅度，FPS越高通常表示图像流畅度越好，用户体验也越好。
- TEMP：每秒读取一次设备节点下GPU温度、系统芯片温度信息。
- RAM：每秒读取一次应用进程的实际物理内存，衡量应用的内存占比情况。
- snapshot：每2秒截取一张应用界面截图。

以上是不指定包名的，如果要指定包名，则增加 -PKG 参数 如：

SP_daemon -N 2 -PKG ohos.samples.ecg -c

<LOG 存放路径没有自动显示在日志中>

采集结果默认输出路径：/data/local/tmp/data.csv

第八步 使用 hdc 命令测试稳定性专项测试及其日志分析

专项测试是指定了某一个应用的名称的测试，运行后可以自动生成测试报告”汇总”。

常用命令 ： hdc shell wukong special -C [bundlename] -p

<以下是参考出处 ：

https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V13/wukong-guidelines-V13>

专项测试

命令参数

命令	功能	必选	说明
-h, --help	获取当前专项测试的帮助信息。	否	-
-k, --spec_insomnia	休眠唤醒专项测试。	否	-
-c, --count	设置执行次数。	否	单位次数，默认10次。
-i, --interval	设置执行间隔。	否	单位ms，默认1500ms。
-S, --swap	滑动测试。	否	-
-s, --start[x,y]	设置滑动测试起点坐标。	否	坐标均为正值。
-e, --end[x,y]	设置滑动测试终点坐标。	否	坐标均为正值。
-b, --bilateral	设置往返滑动。	否	默认不往返滑动。
-t, --touch[x,y]	点击测试。	否	-
-T, --time	设置测试总时间。	否	单位分钟，默认10分钟。
-C, --component	控件顺序遍历测试。	否	需要设置测试应用名称。
-r, --record	录制。	否	需要指定录制文件。
-R, --replay	回放。	否	需要指定回放文件。
-p, --screenshot	控件测试截图。	否	-

<LOG 存放路径自动显示在日志中>

```
C:\Users\      >hdc shell wukong special -C                -p
Start connect to AAMS
Success connect to AAMS
Mon Mar  3 11:18:28 2025 : INF : (  115) EnvInit : "Report currentTestDir: (/data/local/tmp/wukong/report/20250303_111826/)"
Mon Mar  3 11:18:28 2025 : INF : (  121) EnvInit : "Report CSV: (/data/local/tmp/wukong/report/20250303_111826/wukong_report.c
sv)"
Mon Mar  3 11:18:28 2025 : INF : (  122) EnvInit : "Report JSON: (/data/local/tmp/wukong/report/20250303_111826/data.js)"
Mon Mar  3 11:18:28 2025 : INF : (  125) EnvInit : "Report exception dir: (/data/local/tmp/wukong/report/20250303_111826/excep
tion/)"
Mon Mar  3 11:18:28 2025 : INF : (   92) StartCrashDirListen : create listener thread
Mon Mar  3 11:18:28 2025 : INF : (   94) StartCrashDirListen : thread detach
Mon Mar  3 11:18:28 2025 : INF : (   68) InitCsvFile : "CSV: (/data/local/tmp/wukong/wukong_20250303_111826.csv)"
Mon Mar  3 11:18:28 2025 : INF : (   55) ListenCrashDir : init notify
Mon Mar  3 11:18:28 2025 : INF : (   60) ListenCrashDir : add_watch
Mon Mar  3 11:18:28 2025 : INF : (  112) PrintResultOfStartAbility : "Bundle Name: (com                ) startup successful"
Mon Mar  3 11:18:29 2025 : RUN : .
Mon Mar  3 11:18:32 2025 : INF : (  237) ComponentTouchInput : "component Content: Touch Position: (81, 207)"
Mon Mar  3 11:18:35 2025 : INF : (  237) ComponentTouchInput : "component Content: Touch Position: (81, 207)"
Mon Mar  3 11:18:37 2025 : INF : (  237) ComponentTouchInput : "component Content: Touch Position: (630, 207)"
Mon Mar  3 11:18:40 2025 : INF : (  237) ComponentTouchInput : "component Content: Touch Position: (1179, 207)"
Mon Mar  3 11:18:42 2025 : INF : (  237) ComponentTouchInput : "component Content: Touch Position: (630, 641)"
```

第九步 研究使用 adb shell 中的命令是否可以在 hdc 中运行

dumpsys 是 Android 操作系统中的一个命令行工具，并不适用于鸿蒙系统。

```
C:\Users\      >hdc shell dumpsys window
/bin/sh: dumpsys: inaccessible or not found
```

第十步 使用 hdc 进行包名查询

<以下是参考出处 ：
https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V13/bm-tool-V13>

显示所有已安装的 Bundle 名称： hdc shell bm dump -a

```
# 显示所有已安装的Bundle名称
bm dump -a
```

```
C:\Users\      >hdc shell bm dump -a
ID: 100:
        com.huawei.hmos.arkwebcore
        com.huawei.hmos.calendardata
        com.huawei.hmos.filemanager
        com.huawei.hmos.files
        com.huawei.hmos.hipreview
        com.huawei.hmos.hiviewx
        com.huawei.hmos.huaweicast
        com.huawei.hmos.inputmethod
        com.huawei.hmos.instantshare
        com.huawei.hmos.location
        com.huawei.hmos.mediacontroller
        com.huawei.hmos.ouc
        com.huawei.hmos.photos
        com.huawei.hmos.projectmenu
        com.huawei.hmos.screenrecorder
        com.huawei.hmos.screenshot
        com.huawei.hmos.security.privacycenter
        com.huawei.hmos.settings
        com.huawei.hmos.superhub
        com.huawei.hmos.themedataservice
        com.huawei.hmos.useriam.idmwidget
        com.huawei.hms.pushservice
```

获取 app 的 ability 名字 hdc shell aa dump -a

<以下是参考出处 ：

https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V13/aa-tool-V13 >

打印命令（dump(deprecated)）

用于打印应用组件的相关信息。

```
aa dump -a
```

说明

aa dump命令从API version 7开始支持，从API version 9废弃，替换命令为hidumper -s AbilityManagerService。

打印命令参数列表

参数	二级参数	参数说明
-h/--help	-	帮助信息。
-a/--all	-	打印所有mission内的应用组件信息。

```
C:\Users\      >hdc shell aa dump -a
User ID #100
  current mission lists:{
    Mission ID #33  mission name #[#com.huawei.hmos.files:entry:EntryAbility]  lockedState #0  mission affinity #[]
      AbilityRecord ID #14
        app name [com.huawei.hmos.files]
        main name [EntryAbility]
        bundle name [com.huawei.hmos.files]
        ability type [PAGE]
        state #BACKGROUND  start time [86767]
        app state #BACKGROUND
        ready #1  window attached #0  launcher #0
        callee connections:
        isKeepAlive: false
  }
    Mission ID #34  mission name #[#com.huawei.hmos.settings:phone_settings:com.huawei.hmos.settings.MainAbility]  lockedState
  #0  mission affinity #[]
      AbilityRecord ID #15
        app name [com.huawei.hmos.settings]
        main name [com.huawei.hmos.settings.MainAbility]
        bundle name [com.huawei.hmos.settings]
        ability type [PAGE]
        state #BACKGROUND  start time [89806]
        app state #BACKGROUND
        ready #1  window attached #0  launcher #0
        callee connections:
        isKeepAlive: false
  }
```

第十一步 hdc 如何抓取终端的日志

<以下是参考出处：

https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V13/hdc-V13>

hilog	打印设备端的日志信息。

hdc hilog> a.log

```
C:\Users\      >hdc hilog> a.log
```

第十二步 hdc 的文件传输命令

<以下是参考出处：

https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V13/hdc-V13>

hdc file send localpath remotepath　　比如 /data/local/tmp

```
C:\Users\      >hdc file send   C:\app\test.apk /data/local/tmp/
FileTransfer finish, Size:0, File count = 1, time:11ms rate:0.00kB/s
```

hdc file recv remotepath localpath

```
C:\Users\      >hdc file recv  /data/local/tmp/data.csv C:\app
FileTransfer finish, Size:38, File count = 1, time:9ms rate:4.22kB/s
```

第十三步 hdc 的安装 卸载命令

<以下是参考出处：

https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/hdc-V5 >

安装

hdc install [-r|-s] src

hdc install E:\example.hap

参数名	说明
src	应用安装包的文件名
-r	替换已存在应用（.hap）
-s	安装一个共享包（.hsp）

卸载

hdc uninstall [-k|-s] packageName

hdc uninstall com.example.hello

参数名	说明
packageName	应用安装包。
-k	保留/data和/cache目录。
-s	卸载共享包。

第十四步 启动 APP 和停止 APP

<以下是参考出处：

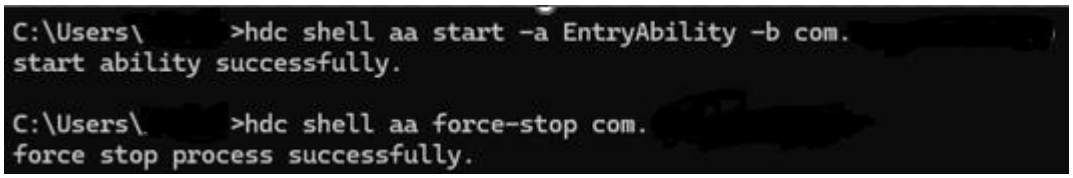
https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/aa-tool-V5 >

以下是启动 app 的命令：

aa start [-d <deviceId>] [-a <abilityName> -b <bundleName>]

以下是停止 app 的命令：

aa force-stop <bundleName>



第三部分 普通 APP 和鸿蒙 APP 之间的区别简要分析

第一点区别

鸿蒙不支持 dumpsys，安卓支持 dumpsys

第二点区别

包名的区别：

鸿蒙是 app name + ability name

安卓是 name of apk + activity 的名字

鸿蒙系统

在鸿蒙系统中，一个应用通常包含一个或多个 Ability（能力）。每个 Ability 都代表了应用的一个功能模块，可以独立运行和提供服务。因此，鸿蒙系统的应用名称往往由 app name 和 ability name 两部分组成，以清晰地反映应用的核心功能和提供的服务。

• app name：用于标识整个应用，通常与应用的品牌或核心功能相关。

• ability name：用于标识应用中的具体功能模块或能力，帮助用户快速找到所需的功能。

安卓系统

在安卓系统中，一个应用由一个或多个 Activity（活动）组成。Activity 是应用与用户交互的界面组件，每个 Activity 都代表了应用的一个屏幕或功能区域。因此，安卓系统

的应用名称通常由 apk 的名字（即包名）和 Activity 的名字组成。

• apk 名字（包名）：用于唯一标识一个安卓应用程序，通常采用反向域名的格式，以确保在不同开发者之间的应用程序命名不会冲突。

• Activity 名字：用于标识应用中的具体界面或功能区域，帮助用户了解应用的结构和功能布局。

第三点区别

• 鸿蒙应用的后缀名是 hap

• 安卓应用的后缀名是 apk

第四点区别

鸿蒙有 SP_daemon 等性能测试工具，但是安卓没有，安卓可以通过 shell 命令来进行性能测试的，?dumpsys?是 Android 操作系统中的一个命令行工具。

第五点区别

查询当前运行包名的命令区别

以下是安卓：

adb shell dumpsys window | findstr mCurrentFocus

以下是鸿蒙

hdc shell aa dump -a 然后查找运行的状态。

第六点区别

查询所有包名的区别

安卓

adb shell pm list packages

鸿蒙

hdc shell bm dump -a

第七点区别

稳定性测试测试的区别，安卓是用的 monkey 命令，鸿蒙用的是 wukong 命令。

第八点区别

安装和卸载的区别

安卓

adb install myapp.apk

adb uninstall com.example.myapp

鸿蒙

hdc install [-r|-s] src

hdc install E:\example.hap

hdc uninstall [-k|-s] packageName

hdc uninstall com.example.hello

第九点区别

上传文件和下载文件的区别

安卓

adb push localfile remotepath

adb push C:\app\test.apk /sdcard/

adb pull remotefile localpath

adb pull /sdcard/test.apk C:\app\

鸿蒙

hdc file send [-a|-sync|-z|-m] localpath remotepath

hdc file send E:\example.txt /data/local/tmp/example.txt

hdc file recv [-a|-sync|-z|-m] remotepath localpath

hdc file recv /data/local/tmp/data.csv C:\app

第十点区别

鸿蒙有专项测试 special， 但是安卓没有专项测试。

hdc 专项测试命令：

hdc shell wukong special -C [bundlename] -p

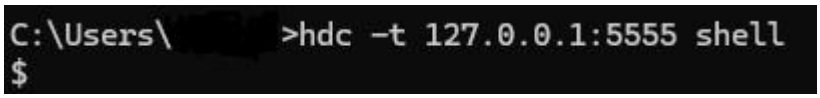
第十一点区别

指定 ip 地址从而进入 shell 编辑环境的命令不同

hdc = > -t ; adb => -s

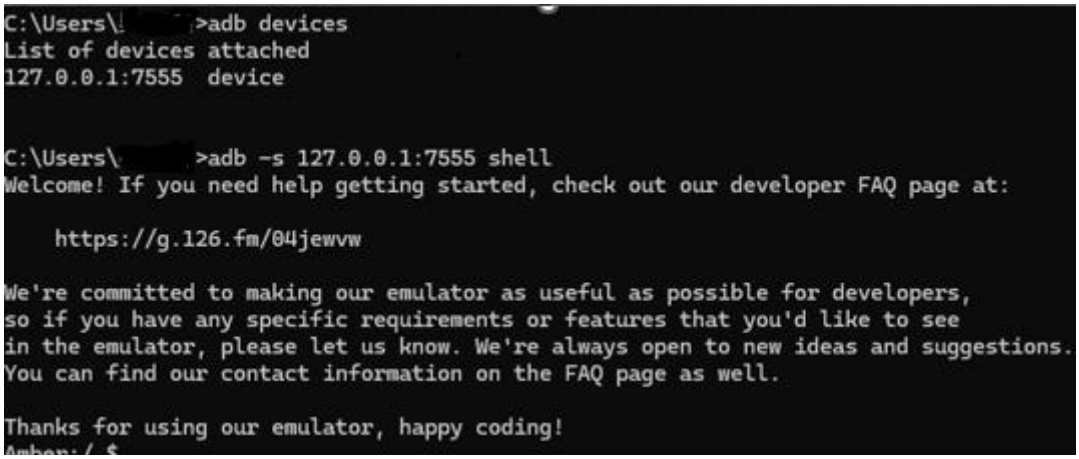
鸿蒙

hdc -t 127.0.0.1:5555 shell



安卓

adb -s 127.0.0.1:7555 shell



第十二点区别

hdc 绑定 ip 地址的命令

hdc tconn 127.0.0.1:5555

hdc tconn 127.0.0.1:5555 -remove

```
C:\Users\      >hdc list targets
127.0.0.1:5555

C:\Users\      >hdc tconn 127.0.0.1:5555 -remove

C:\Users\      >hdc list targets
[Empty]

C:\Users\      >hdc tconn 127.0.0.1:5555
[Info]Target is connected, repeat operation

C:\Users\      >hdc list targets
127.0.0.1:5555
```

tconn	指定连接设备：通过“IP地址：端口号”来指定连接的设备。

adb 绑定 ip 地址的命令

adb connect 127.0.0.1:7555

adb disconnect 127.0.0.1:7555

```
C:\Users\      >adb devices
List of devices attached
127.0.0.1:7555  device


C:\Users\      >adb disconnect 127.0.0.1:7555
disconnected 127.0.0.1:7555

C:\Users\      >adb devices
List of devices attached


C:\Users\      >adb connect 127.0.0.1:7555
connected to 127.0.0.1:7555

C:\Users\      >adb devices
List of devices attached
127.0.0.1:7555  device
```

第十三点区别

日志的形式不同

鸿蒙

1）稳定性专项测试 wukong

LOG 存放路径自动显示在日志中

2）稳定性随机测试 wukong

LOG 存放路径自动显示在日志中

3）性能测试 SP_Daemon

LOG 存放路径没有自动显示在日志中

安卓

1）稳定性测试 monkey

LOG 存放路径没有自动显示在日志中

2）性能测试 dumpsys

LOG 存放路径没有自动显示在日志中

第十四点区别

启动停止应用的区别

安卓应用：

adb shell am start <name of apk with com/Activity>

adb shell am force-stop <name of apk with com>

以下是鸿蒙相关命令

hdc shell aa start [-d <deviceId>] [-a <abilityName> -b <bundleName>]

hdc shell aa force-stop <bundleName>

MySQL 事务隔离：为什么你改了我看不见

◆作者：王伟超&陈昊

一、背景与意义

在数据库管理系统中，事务（Transaction）是确保数据一致性和完整性的基本单位，即保证一组数据库操作，要么全部成功，要么全部失败。事务的 ACID 特性——原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）和持久性（Durability）——为数据的可靠操作提供了保障。其中，事务的隔离性尤为重要，它确保了并发执行的事务之间互不干扰，从而避免了数据不一致的问题。然而，在实际应用中，开发者常常会遇到“你改了我看不见”的现象，这实际上正是事务隔离机制在起作用。本文将深入探究 MySQL 事务隔离的底层实现原理，帮助读者理解这一现象的本质。

二、关键技术

（一）锁机制

锁是实现事务隔离的重要手段之一。MySQL 使用了多种类型的锁，包括共享锁（Shared Lock）和排他锁（Exclusive Lock）。共享锁允许多个事务同时读取同一数据资源，而排他锁则在事务修改数据时使用，阻止其他事务对该数据的读写操作。例如，在一个事务对某行数据执行 SELECT...FOR UPDATE 语句时，会获取该行数据的排他锁，确保在该事务完成修改并提交之前，其他事务不能对该行数据进行修改或读取不一致的数据。尽管依赖锁机制保证了事务的隔离性，然而单纯的锁机制会降低数据库并发度，为此 MySQL 通常采用锁+MVCC 的策略来保证隔离性与并发度。

（二）多版本并发控制（MVCC）

MVCC 是 MySQL 实现事务隔离的核心技术之一。它通过为每个数据行维护多个版

本，使得不同事务在不同时间点看到不同版本的数据。在 MVCC 机制下，事务在读取数据时，根据自身的事务 ID 和数据行的创建版本号、删除版本号等来确定能够看到的数据版本。这样，即使有其他事务正在修改数据，读取事务也能够读取到在其开始时刻之前已提交的数据版本，从而实现了一定程度的并发读取而不被阻塞，提高了数据库系统的并发性能。

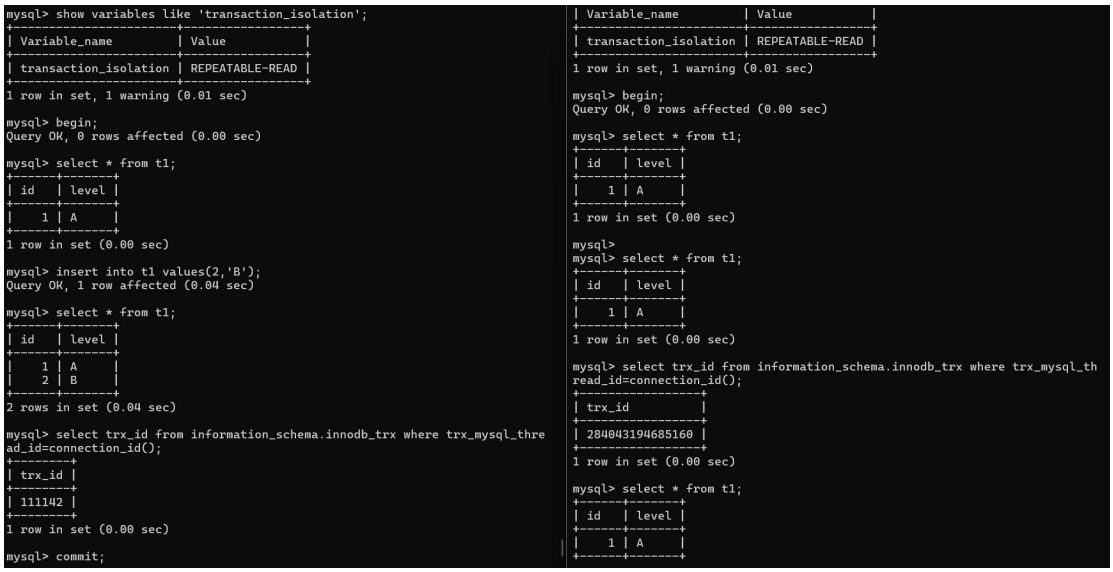
三、原理介绍与解析

（一）事务隔离级别及现象

1.读未提交（Read Uncommitted）：这是最低的事务隔离级别，事务可以读取到其他未提交事务修改的数据。这种级别下，并发性能最高，但数据的一致性和准确性难以保证，可能会出现脏读（Dirty Read）现象（下图所示，事务 2 在 T5 时刻读取到了事务 1 在 T4 时刻未提交的修改），即读取到了其他事务未提交的修改数据，而这些数据可能随后被回滚，导致读取到的数据是无效的。

2.读已提交（Read Committed）：事务只能读取到已提交事务修改的数据。相比读未提交级别，它避免了脏读，但可能出现不可重复读（Non - Repeatable Read）问题。在一个事务内多次读取同一数据行，可能由于其他事务在这期间提交了对该行数据的修改，导致每次读取到的数据不一致（下图中事务 2 在 T2、T4 时刻读取数据一致，在事务 1 提交后的 T6 时刻读取的数据和 T2、T4 时刻不一致）

3.可重复读（Repeatable Read）：这是 MySQL 默认的事务隔离级别。在该级别下，事务在执行期间多次读取同一数据行时，会得到相同的数据，即使其他事务对该行数据进行了修改并提交。这是通过 MVCC 机制实现的，事务开始时确定了其能够看到的数据版本范围，在事务执行期间不会受到其他事务提交的修改的影响。然而，在可重复读级别下可能会出现幻读（Phantom Read）现象，即事务在执行过程中，按照某个查询条件多次查询数据，发现符合条件的数据行数不一致，这是由于其他事务插入或删除了符合条件的数据行。



4.可串行化（Serializable）：最高的事务隔离级别，事务串行执行，完全避免了并发事务之间的相互影响，保证了数据的强一致性，但并发性能最差。（下图中事务 2 查看数据时，事务 1 无法对数据进行修改；当事务 2 提交后，事务才能对数据进行修改）

（二）MVCC 原理深入解析

在 MVCC 中，每个事务在开始时被分配一个唯一的事务 ID。对于数据库中的每一行数据，除了存储实际的数据值外，还维护了创建版本号（creation_version）和删除版本号（deletion_version）。当一个事务插入一行数据时，该行数据的创建版本号被设置为该事务的 ID。当一个事务更新一行数据时，实际上是插入了一个新的版本，新数据行的创建版本号为当前事务的 ID，而原数据行的删除版本号被设置为当前事务的 ID，表示原数据行在该事务中被删除（逻辑删除）。当一个事务删除一行数据时，该行数据的删除版本号被设置为当前事务的 ID。

在事务读取数据时，根据事务 ID 和数据行的创建版本号、删除版本号来判断是否可见。如果数据行的创建版本号小于等于当前事务的 ID，并且删除版本号大于当前事务的 ID 或者未定义（表示数据行未被删除），则该数据行对当前事务可见。通过这种方式，不同事务在不同时间点能够看到不同版本的数据，实现了事务隔离中的数据读取一致性。

四、底层实现的分析与探究

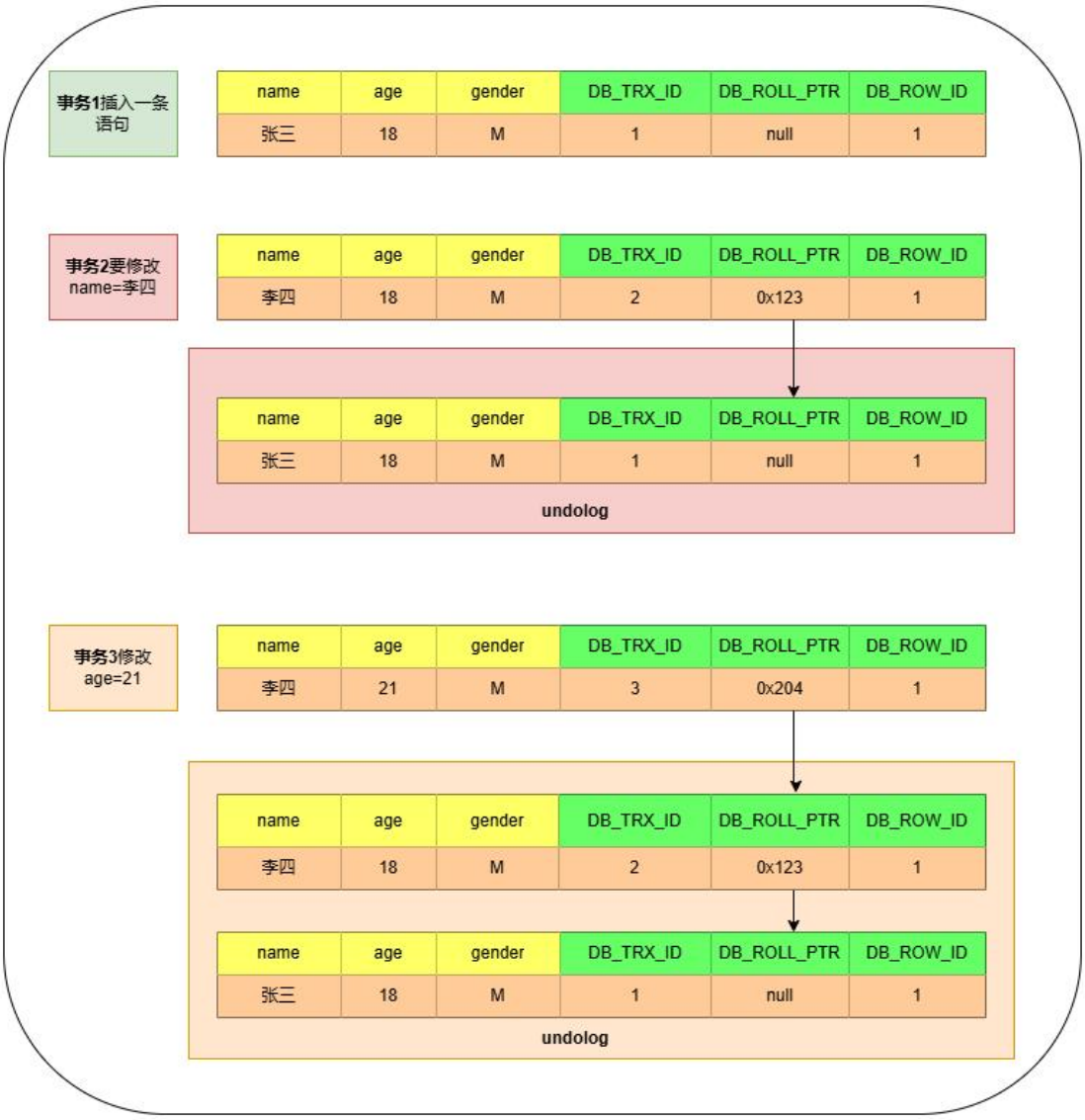
（一）InnoDB 存储引擎中的事务隔离实现

InnoDB 是 MySQL 的默认存储引擎，其在事务隔离实现方面有着深入的设计。在 InnoDB 中，锁和 MVCC 机制紧密结合。例如，在可重复读事务隔离级别下，事务在首次读取数据时，会根据 MVCC 机制确定能够看到的数据版本，并获取相应的共享锁（如果只是读取操作）。如果事务需要对数据进行修改，则会将共享锁升级为排他锁。在事务执行过程中，InnoDB 通过维护一个事务 ID 列表和数据行的版本信息，来判断数据行对不同事务的可见性。

对于锁的管理，InnoDB 采用了多种锁算法，如记录锁（Record Lock）、间隙锁（Gap Lock）和临键锁（Next - Key Lock）等。记录锁锁定单个数据行，间隙锁锁定数据行之间的间隙，临键锁则是记录锁和间隙锁的组合，用于防止幻读现象。在可重复读级别下，当事务执行范围查询（如 SELECT...WHERE...）时，InnoDB 会自动使用临键锁，锁定查询范围内的数据行以及相邻的间隙，确保在事务执行期间，其他事务插入符合查询条件的数据行时会被阻塞，从而避免幻读。

（二）事务日志与事务隔离

事务日志在 MySQL 事务隔离实现中也起着重要作用。InnoDB 使用重做日志（Redo Log）和 undo 日志。重做日志用于在数据库发生故障时恢复已提交事务对数据的修改，确保数据的持久性。undo 日志则用于事务回滚操作，记录了事务修改数据前的原始值。在事务隔离的底层实现中，通过 undo 日志可以获取数据行的历史版本信息，以便在事务读取数据时提供正确的数据版本。具体逻辑为：unlog 通过事务 ID 标识 DB_TRX_ID、数据回滚指针 DB_ROLL_PTR、数据行标识 DB_ROW_ID 三个隐藏字段来实现历史版本信息的记录。如下图所示，首先事务 1 在 t 时刻插入了一条数据，该修改行会维护相关的插入数据信息及隐藏字段信息；之后事务 2 在 t2 时刻（t2>t1)修改更新了同一行数据，此时在 undolog 日志中会保留这行数据的历史版本，最新版本的 DB_ROLL_PTR 字段指向这行数据的历史版本；之后事务 3 在 t3 时刻（t3>t2)对同一行数据又做了更新修改，同理将事务 2 修改的版本存入 undolog，同时最新版本的数据行中隐藏字段 DB_ROLL_PTR 指向事务 2 修改后的版本。这样事务修改得到所有的版本通过隐藏字段 DB_ROLL_PTR 连接形成一个版本链，历史版本记录在 undolog 中，头节点始终为最新的版本信息。通过控制不同事务读取不同的版本即可实现部分隔离级别。



（三）MVCC 原理深入解析

ReadView（读视图）是快照读 SQL 执行时 MVCC 提取数据的依据，记录并维护系统当前活跃的事务（未提交的）ID，MVCC 在 undolog 中记录的数据的历史版本信息，通过控制事务读取不同的读视图即能实现隔离级别的控制。在 MVCC 中，每个事务在开始时被分配一个唯一的事务 ID。对于数据库中的每一行数据，除了存储实际的数据值外，还维护了创建版本号（creation_version）和删除版本号（deletion_version）。当一个事务插入一行数据时，该行数据的创建版本号被设置为该事务的 ID。当一个事务更新一行数据时，实际上是插入了一个新的版本，新数据行的创建版本号为当前事务的 ID，而原数据行的删除版本号被设置为当前事务的 ID，表示原数据行在该事务中被删除（逻辑删

除）。当一个事务删除一行数据时，该行数据的删除版本号被设置为当前事务的 ID。

在事务读取数据时，根据事务 ID 和数据行的创建版本号、删除版本号来判断是否可见。如果数据行的创建版本号小于等于当前事务的 ID，并且删除版本号大于当前事务的 ID 或者未定义（表示数据行未被删除），则该数据行对当前事务可见。通过这种方式，不同事务在不同时间点能够看到不同版本的数据，实现了事务隔离中的数据读取一致性。事务的不同隔离级别也正是通过 MVCC 机制与锁赖共同实现的，具体而言不同隔离级别的实现逻辑如下：

读未提交：读不加锁，写加排他锁。

读已提交：读不加锁，写加排他锁，在事务中每一次执行快照读（不加锁）时生成 ReadView。

可重复读：读不加锁，写加临键锁，在事务中仅第一次执行快照读时生成 ReadView，后续复用该 ReadView。

串行化： 读加共享锁，写加排他锁。

五、总结

MySQL 事务隔离是一个复杂而重要的数据库特性，其底层实现涉及锁机制、MVCC、事务日志等多个关键技术。通过不同的事务隔离级别，MySQL 能够在数据一致性和并发性能之间进行权衡，满足不同应用场景的需求。深入理解事务隔离的底层实现原理，对于数据库管理员和开发人员来说至关重要。合理设置事务隔离级别、优化查询语句和事务设计，可以有效提高数据库应用的性能和数据的可靠性。在未来的数据库技术发展中，随着对高并发和大数据量处理需求的不断增长，MySQL 事务隔离机制可能会进一步优化和演进，例如在更高效的锁管理、更智能的 MVCC 实现以及与分布式数据库事务处理的融合等方面，以适应不断变化的数据库应用环境。

-------- END ---------