20250217移动项目仿真测试方案

RCS任务接口测试

1. 测试目的

2. 测试背景

3. 测试工具

4. 测试范围

5. 测试数据

6. 测试流程

7. 测试脚本

8. 测试步骤

9. 预期结果

10. 测试总结

RCS任务接口测试

1. 测试目的

验证 **RCS任务添加接口** 的功能性和稳定性。特别是检查通过提供不同货位对（**fromShelfBoxId**, **toShelfBoxId**）来创建任务时，接口是否能够成功接收、处理并返回正确响应。

2. 测试背景

根据仿真地图，测试场景为遍历一千个货位，每个货位进行一次任务测试。每次任务请求将包含从源货位（**fromShelfBoxId**）到目标货位（**toShelfBoxId**）的任务数据。根据提供的脚本，结合 Excel 文件中存储的货位数据，通过 **pytest** 框架进行接口的自动化测试。

3. 测试工具

- **测试框架**：Pytest
- **接口请求工具**：Requests
- **Excel 数据读取**：Openpyxl
- **测试环境**：RCS任务接口服务（`http://10.168.1.154:8888/rcs/external/task/add?token=z247`）

4. 测试范围

- 测试的内容为 RCS 任务添加接口的功能，涵盖货位对（fromShelfBoxId, toShelfBoxId）是否正确。
- 每个货位对（从Excel中获取）会触发一个API请求，并且验证接口是否返回成功。

5. 测试数据

- 数据来源：**rcs_task.xlsx**
  - Excel中包含一列源货位（**fromShelfBoxId**）和目标货位（**toShelfBoxId**）数据。
  - 每行数据代表一个货位对。
  - 测试将遍历所有的货位对，执行任务请求。

6. 测试流程

1. 准备测试数据

- 打开 Excel 文件 `rcs_task.xlsx`，读取第一列（**fromShelfBoxId**）和第二列（**toShelfBoxId**）的数据。
- 将每一对货位作为测试输入传递给测试用例。

2. 接口请求

- 对于每一对 **fromShelfBoxId** 和 **toShelfBoxId**，构造POST请求的数据包，并发送到任务添加接口 `http://10.168.1.154:8888/rcs/external/task/add?token=z247`。
- 请求数据结构如下：

JSON

```
{
  "batchNumber": "随机生成的批次号",
  "fromShelfBoxId": "fromShelfBoxId",
  "toShelfBoxId": "toShelfBoxId",
  "notifyUrl": "http://127.0.0.1:18002/xxx"
}
```

3. 验证接口响应

- 验证返回的接口响应是否包含正确的状态码（例如 200）。
- 验证响应数据是否符合预期，检查是否有任务创建成功的标识（例如状态字段、任务ID等）。

4. 日志和结果输出

- 每次任务接口请求后，记录响应结果，输出到控制台或日志文件中，方便后续分析。

- 如果请求失败或返回异常，需要能够记录详细的错误信息，便于排查问题。

5. 测试结束

- 完成所有货位对的任务创建请求后，汇总结果，输出成功和失败的统计数据。

7. 测试脚本

```python
class Test_RcsTask:
    @pytest.mark.run(order=1)
    @pytest.mark.parametrize('fromShelfBoxId,toShelfBoxId',
                             yaml.safe_load(open("rcs_task.yaml", encoding='utf-8')))
    def test_rcs01(self,fromShelfBoxId,toShelfBoxId):
        url = "http://10.168.1.154:8888/rcs/external/task/add?token=z247"
        data = {
            "batchNumber": "{}".format(random.randint(1111, 9999)),
            "fromShelfBoxId": fromShelfBoxId,
            "toShelfBoxId": 1,
            "notifyUrl": "http://127.0.0.1:18002/xxx",
        }
        res = requests.post(url, json=data).json()
        print(res)
        while True:
            db = test_mysql.test_sql()
            # 使用 cursor() 方法创建一个游标对象 cursor
            cursor = db.cursor()
            sql = "SELECT * from link_wms_rcs;"
            cursor.execute(sql)
            rows = cursor.fetchall()
            # 判断所有的status是否都为3
            if all(row["status"] == 3 for row in rows):  # 替换<status_column_index>为正确的status列索引
                print("所有的status都是3，跳出循环")
                break  # 如果所有的status都是3，跳出循环
            else:
                print("存在status不为3，继续循环")
            db.commit()
            db.close()
```

8. 测试步骤

1. 部署环境

- 确保测试环境中的 RCS 服务已经启动，并且接口 `http://10.168.1.154:8888/rcs/ext`

`ernal/task/add?token=z247` 可用。

2. **执行测试**
   - 使用 `pytest` 执行测试脚本，遍历所有货位对数据并向接口发送请求。
   - 例如，运行命令：

```bash
pytest test_rcs_task.py
```

3. **结果分析**
   - 查看控制台输出的响应结果，分析每个请求的成功与失败情况。
   - 若有失败的请求，查看错误信息并排查问题。

9. 预期结果

- 所有任务请求应返回 **成功** 状态（依据接口实际定义返回的字段，可能是 Response: {'code': 0, 'msg': 'ok'}）。
- 对于不符合条件的数据（例如无效的货位ID），接口应返回错误信息，并能进行错误捕获和记录。

10. 测试总结

- 在所有的测试数据执行完成后，汇总成功与失败的任务请求数量，并进行后续分析，确保接口的稳定性和高效性。
- 如有测试失败，可以根据错误日志进行问题定位并修复。