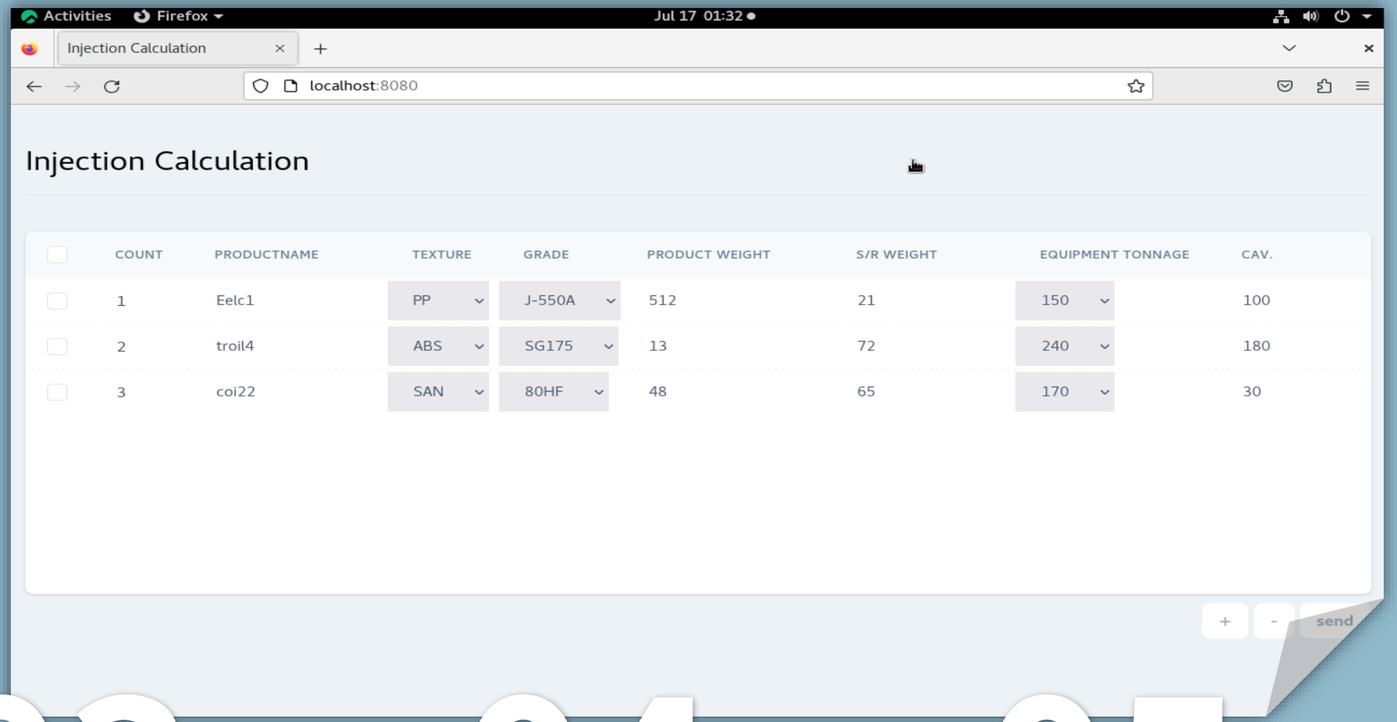
글로벌 직업 전문 학교

사출 계산 PROJECT

한석희, 이대환

목차



01

Project Outline

프로젝트 개요

개발 배경

개발 목표

개 요

1. 프로젝트 개요

사무자동화 프로그램을 활용한 사출 계산 자동화 및 서버 기반 데이터베이스 적재 시스템 개발

2. 개발 배경

최근 사무자동화 프로그램의 도입으로 업무 효율성을 높이려는 수요가 증가함에 따라, 정확한 데이터 관리 및 자동화된 보고서 생성이 중요해졌습니다. 이를 위해 리눅스 서버와 도커 컨테이너를 이용해 톰캣 서버와 오라클 DB, 백업용 오라클 DB를 구축하고, 웹 페이지에서 입력된 데이터를 기반으로 사출 계산을 수행한 후 결과값을 제공하며, 데이터를 DB에 적재하고, RPA를 통해 엑셀 파일로 자동 변환하는 프로그램을 개발하게 되었습니다.

3. 개발 목표

누구에게나 편리한 사용 및 데이터에 대한 무결성 보존

- ✓ 업무에 대한 효율성 증대
- ✓ 사용자에 대한 편의성 향상
- ✓ 자동화된 보고서 생성
- ✓ 서버와 데이터 베이스간 정확한 데이터베이스 관리

02

Team Position

팀원 역할 분담

팀원 역할 분담

한석희(팀장)

- RPA 기본 구성 및 도커 내 Oracle DB 데이터 엑셀 삽입 작업
- Rocky Linux환경에 도커 설치 및 Tomcat 서버와 Oracle DB 이미지 설치
- Tomcat 서버에서의 JSP 구성 및 Servlet 제작
- Shell Script를 활용하여 데이터베이스 백업 생성
- Crontab을 이용하여 백업 자동화 환경 구성

- RPA를 이용한 자동 Excel 유무 판단 및 Tomcat server 연결
- Rocky Linux환경에 도커 설치 및 Tomcat 서버와 Oracle DB 이미지 설치
- Create&OpenExcel 액티비티로 사출 원가 계산 표 자동화 생성 및 데이터 삽입
- JSP에 JS를 이용한 정규표현식 지정
- Crontab을 이용하여 백업 자동화 환경 구성

이대환(팀원)

03

Technology

활용 기술 및 방법

Project Flow

화면 GUI 구성

프로세스 흐름도

활용 기술 및 방법

Brity RPA

Brity RPA (사무자동화 프로그램)

비즈니스 프로세스를 자동화하는 기술로, 주로 반복적이고 규칙 기반의 작업을 자동화하는데 사용

.

톰캣 서버 연결 및 DBconnect 액티비티를 이용한 자동화 엑셀 데이터 입력

Rocky Linux™

Linux 운영 체제 (Operating System)

오픈 소스 소프트웨어, 주로 서버나 클라우드 환경, 임베디드 시스템 등에 사용

.

Shell Script를 이용한 도커 컨테이너 내 오라클 DB, 백업 오라클 DB를 연결하여 crontab을 통한 자동화 백업 sh 생성

Apache Tomcat

톰캣 서버 (Web Server)

Java Servlet과 JSP(JavaServer Pages)를 실행하기 위한 오픈소스 컨테이너

.

사용자가 입력한 데이터를 이용한 사출계산 작업 및 Servlet을 통한 데이터 도커 내 오라클 DB 적재

docker

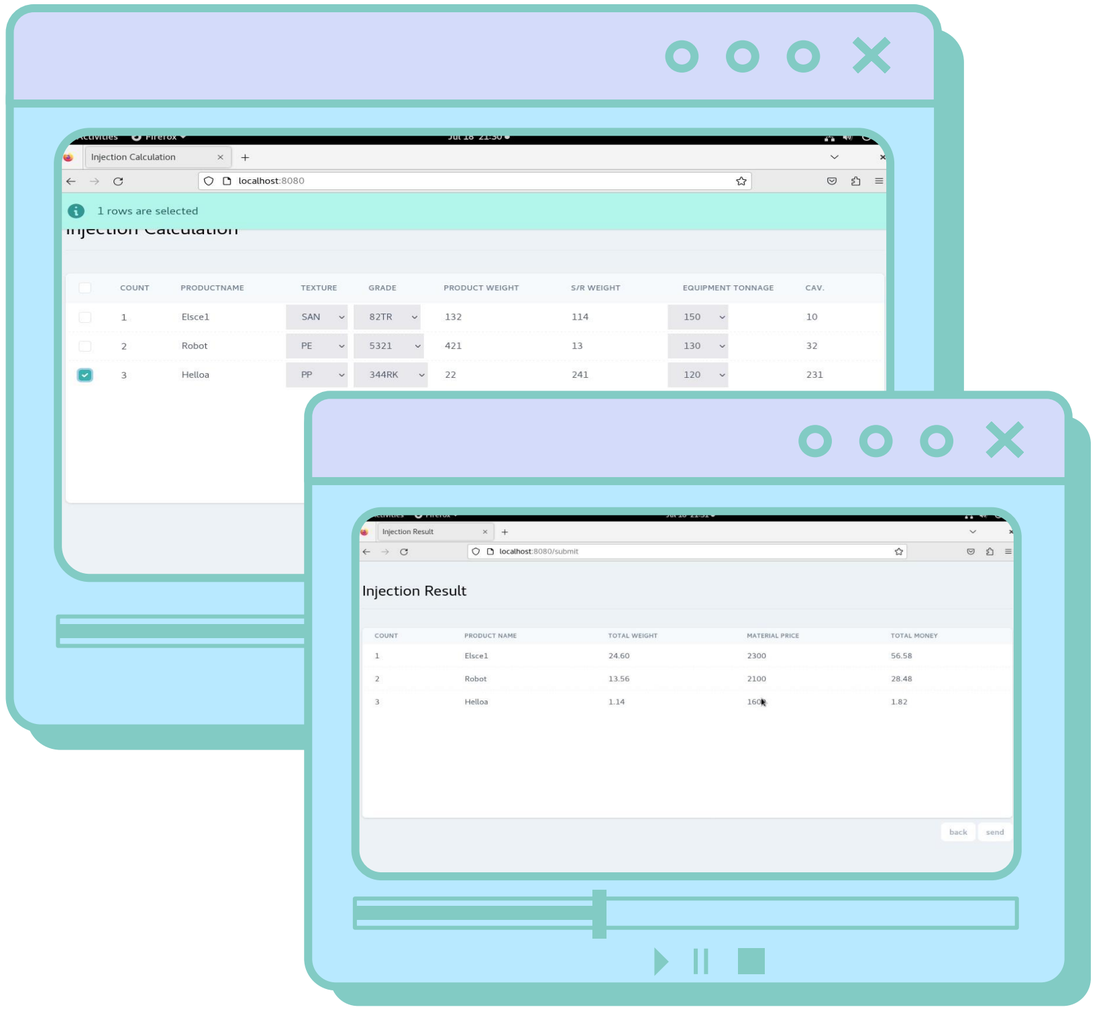
Docker (Containerization Platform)

애플리케이션을 컨테이너라는 가상 환경에서 실행할 수 있게 해주는 오픈 소스 플랫폼

.

Tomcat server, Oracle database 컨테이너를 적재하여 Linux 운영체제를 서버로서의 역할을 수행 하도록 하며, window와 RPA가 Linux 접근을 통해 데이터 적재 및 추출 작업 수행

GUI 화면 구현



사출 원가 계산

순서	품명	재질	그레이드	제품중량(g)	S/R중량(g)	개당총중량(g)	원자재단가(Kg)	개당원재료비	설비톤수	Cav.
Ex	Example	Texture	Grade	ProductWeight	S/R_Weight	TotalWeight	MaterialPrice	TotalMoney	EquipmentTonnag	Cav
1	a	ABS	HF380	1	1	0.02	2600	0.05	100	100
2	asd	ABS	HF380	123	123	2.46	2600	6.4	100	100
3	b	ABS	HF380	2	2	0.04	2600	0.1	100	100
4	c	ABS	HF380	3	3	0.06	2600	0.16	100	100
5	d	ABS	HF380	4	43	0.47	2600	1.22	100	100
6	e	ABS	HF380	5	5	0.1	2600	0.26	100	100

설비톤수	엄률
100	160000
120	180000
130	200000
150	250000
160	250000
170	250000
180	270000
200	300000
220	300000
240	320000
280	350000
380	400000
400	450000

재질	그레이드	단가
ABS	HF380	2600
ABS	HI121	2400
ABS	SG175	2600
PP	344RK	1600
PP	4017	1600
PP	J-550A	1800
PP	J-560S	1600
PE	M850	1700
PE	5321	2100
PE	XJ700	2100
SAN	80HF	2300
SAN	82TR	2300

ABS	PP	PE	SAN
HF380	344RK	M850	80HF
HI121	4017	5321	82TR
SG175	J-550A	XJ700	
	J-560S		

프로세스 흐름도

01 RPA

- 사용자 Excel.exe 유무 확인
- Calc.xlsx 유무 확인
- DBconnect 액티비티를 통한 리눅스 도커 내 오라클 접근
- 사출 계산 데이터 DB 적재를 위한 웹 페이지 연결작업
- 엑셀 파일에 대한 DB 데이터 입력 자동화

02 Rocky Linux

- SSH를 통한 Win ↔ Linux 연결
- Docker Container 생성
- Shell Script를 이용한 DB 데이터 백업 DB로 imp, exp sh 파일 생성
- sh 파일 crontab 등록하여 자동 실행 등록

03 Docker

- 도커 웹 페이지 참조하여 도커 환경 설정
- Tomcat 이미지를 활용하여 컨테이너 적재
- OracleDB 이미지 활용한 컨테이너 적재
- bash를 이용하여 interface 활용

04 Tomcat

- JSP를 활용한 웹 페이지 생성
- Java Servlet을 통해 도커 내 OracleDB 데이터 적재
- 웹 페이지간 Post방식 데이터 전송 작업 및 JS를 이용하여 정규식 적용

05 OracleDB

- Java Servlet 데이터 적재
- Dmp파일을 통한 시간별 backupDB Table 생성
- 원본 데이터와 백업 데이터를 동시에 생성하여 데이터 관리의 효율성 증가

04

Project Code

Brity RPA

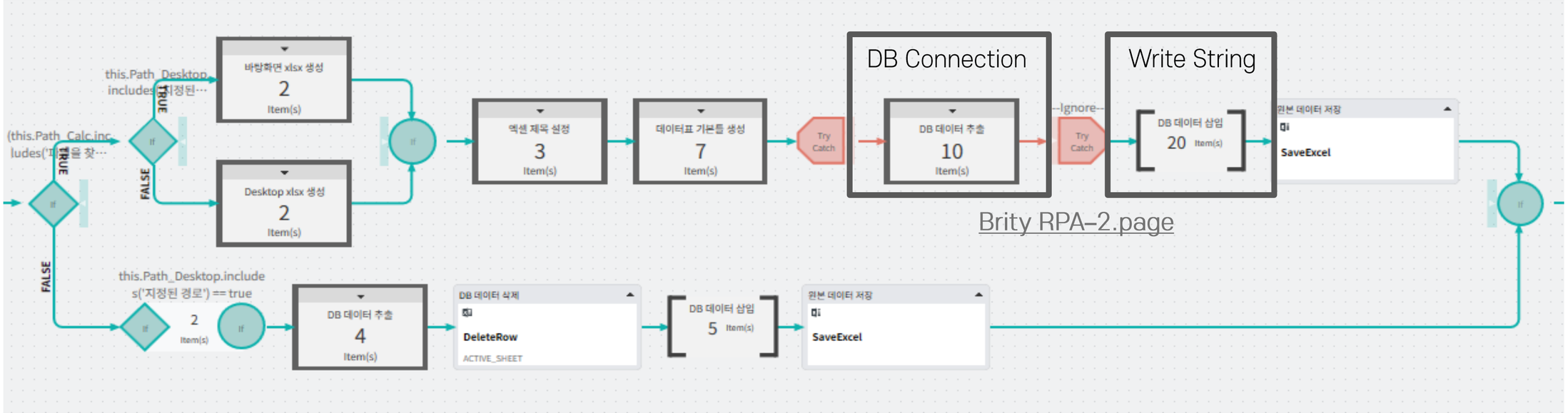
JSP(JavaScript Page)

JAVA(Servlet)

Oracle Database

Shell Script

Brity RPA-1



Brity RPA-2.page

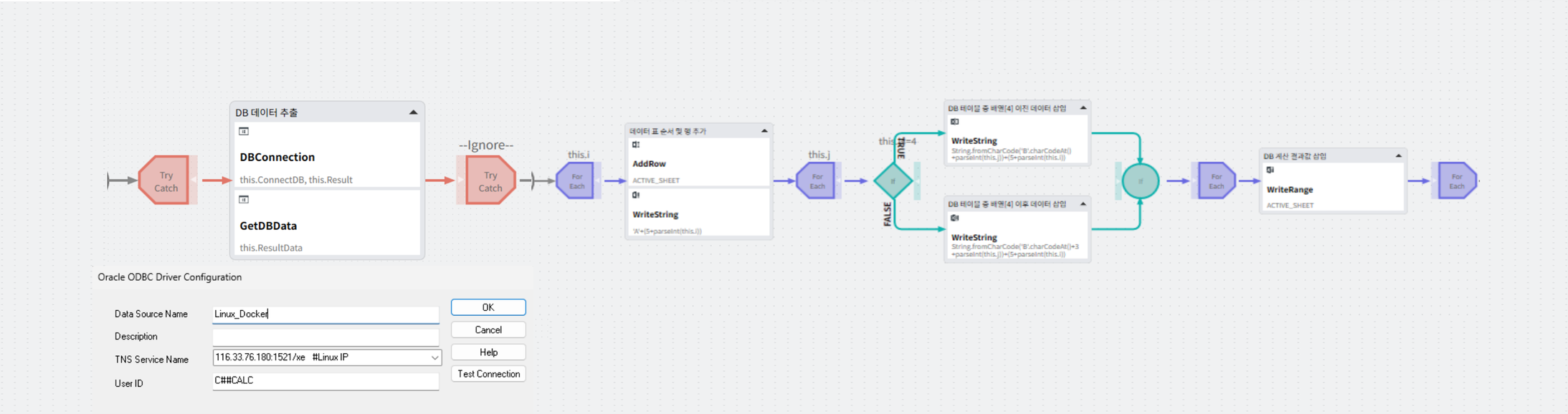
Create xlsx

- 조건을 통한 “Desktop” or “바탕 화면” xlsx 파일 생성
- 서버 내 컨테이너의 OracleDB 접근하여 적재 데이터 및 기준 데이터 추출
- RPA로 미리 작성한 데이터 테이블에 추출된 데이터 삽입
- 파일 저장 및 백업 폴더/파일 생성

Open xlsx

- 조건을 통한 ““Desktop” or “바탕 화면” xlsx 파일 실행
- 서버 내 컨테이너의 OralceDB 접근하여 신규 및 기존 데이터 적재
- 기존 작성되어 있는 데이터 행 전체 삭제 및 데이터 삽입
- 원본 파일 저장 및 백업 폴더/파일 생성

Brity RPA-2



DB Connection

- 도커로 실행한 OracleDB를 DBConnection 액티비티를 통해 ODBC 기반으로 DSN을 접속
- 도커 내 컨테이너의 OracleDB에 접근하여 테이블 기반으로 데이터 추출

Write String Excel

- Excel 내 데이터 삽입 공간을 반복문 안에 있는 AddRow 액티비티를 이용하여 DB의 데이터 개수만큼 반복하며 행을 추가
- OracleDB에 접근하여 추출한 데이터인 사용자 입력 데이터와 사출 계산된 데이터를 삽입

Linux(Docker & OracleDB)

Docker install

· Docker 홈페이지 접속하여 운영체제에 맞게 설치

· 설치 완료시 Docker를 항시 실행되도록 enable 설정

Tomcat image

· Docker 허브에 있는 톰캣 이미지 run으로 설치 및 실행

· Port 8080:8080 으로 설정

· cpu-quota를 사용하여 CPU를 5% 제한

Docker network

· Network create를 이용하여 Oracle 네트워크 구조 생성

Oracle image

· docker login을 통해 오라클 공식 컨테이너에 로그인

· network 옵션을 통해 Oracle 네트워크 설정하여 브릿지를 통해 외부와 통신

· Oracle DB 이미지를 run으로 설치 및 실행하고, 각 각의 Port를 1521, 1522로 설정

· cpu-quota를 사용하여 CPU를 각 각 10% 5% 제한

```
sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin return go(f, seed, [])

systemctl start docker & systemctl enable docker

docker run -d --name tomcat -p 8080:8080 --cpu-quota=5000 tomcat:9.0

docker login https://container-registry.oracle.com/
    * Input UserId, UserPassword

docker network create --driver bridge Oracle

docker run -d --name Oracle1 --network Oracle -p 1521:1521 --cpu-quota=10000
                  container-registry.oracle.com/database/express:21.3.0-xe
docker run -d --name Oracle2 --network Oracle -p 1522:1522 --cpu-quota=5000
                  container-registry.oracle.com/database/express:21.3.0-xe
```

JSP & Java(Servlet)

submitForm()

- 사용자가 입력한 데이터 Post 형식으로 전송
- 행 추가 형식의 데이터 입력 방식으로 저장 데이터 배열 형식으로 선언

hasEmptyField

- 배열 형식의 데이터를 전송하기 전 for문을 통한 공백 데이터 검사
- 공백 데이터 발생시 break 후 alert을 통해 사용자에게 전송

@WebServlet("/submit")

- JSP에서 전송한 form 데이터 호출
- 해당 데이터를 삼항 연산자를 통한 null값 확인 후 success.jsp로 반환

```jsp
<script>
    function submitForm() {
        let NameInputs = document.getElementsByName('productName[]');
        let PweightInputs = document.getElementsByName('productWeight[]');
        let SweightInputs = document.getElementsByName('srWeight[]');
        let hasEmptyField = false;

        for (let i = 0; i < NameInputs.length; i++) {
            let NameValue = NameInputs[i].value.trim();
            let PweightValue = PweightInputs[i].value.trim();
            let SweightValue = SweightInputs[i].value.trim();

            if (NameValue === "" || PweightValue === "" || SweightValue === "") {
                hasEmptyField = true; break;}}
        if (hasEmptyField) { alert("There is a null value.");}
        else {return document.getElementById("dataForm").submit();}
    }
</script>
```

```java
@WebServlet("/submit")
public class DataServlet extends HttpServlet {
    private final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String[] productNames = request.getParameterValues("productName[]");
        String[] textures = request.getParameterValues("texture[]");
        String[] grades = request.getParameterValues("grade[]");

        request.setAttribute("productNames", productNames != null ? productNames : new String[0]);
        request.setAttribute("textures", textures != null ? textures : new String[0]);
        request.setAttribute("grades", grades != null ? grades : new String[0]);

        RequestDispatcher dispatcher = request.getRequestDispatcher("/success.jsp");
        dispatcher.forward(request, response);
    }
}
```

JSP(JavaServer Pages)

RPA

LINUX

INDEX

RESULT

LINUX

Tag_tbody

- Servlet에서 받은 데이터를 변수로 저장
- 사출계산 작업을 수행 후 사용자에게 table 형식으로 반환
- 사용자가 입력한 데이터 및 사출 계산된 데이터 input tag의 hidden 형식으로 form을 통한 Servlet 전송

submitForm()

- form을 통해 Servlet으로 데이터가 정상적으로 전송 되었을 시, alert을 통한 사용자에게 알림 설정

```
<tbody>
    <%
        String[] productNames = (String[]) request.getAttribute("productNames");
        String[] textures = (String[]) request.getAttribute("textures");
        String[] grades = (String[]) request.getAttribute("grades");
        String[] productWeights = (String[]) request.getAttribute("productWeights");
        String[] srWeights = (String[]) request.getAttribute("srWeights");
        String[] equipmentTonnages = (String[]) request.getAttribute("equipmentTonnages");
        String[] equipmentNumbers = (String[]) request.getAttribute("equipmentNumbers");
    %>
      <input type="hidden" name="productNames" value="<%= productNames[i] %>">
      <input type="hidden" name="totalWeights" value="<%= formattedWeight %>">
      <input type="hidden" name="materialPrices" value="<%= materialPrice %>">
      <input type="hidden" name="totalMoneys" value="<%= formattedMoney %>">
      <input type="hidden" name="textures" value="<%= textures[i] %>">
      <input type="hidden" name="grades" value="<%= grades[i] %>">
      <input type="hidden" name="productWeights" value="<%= productWeights[i] %>">
      <input type="hidden" name="srWeights" value="<%= srWeights[i] %>">
      <input type="hidden" name="equipmentTonnages" value="<%= equipmentTonnages[i] %>">
      <input type="hidden" name="equipmentNumbers" value="<%= equipmentNumbers[i] %>">

</tbody>

<script>
    function datatables() {
        return {
        };
    }

    function submitForm() {
        document.getElementById('dbForm').submit();
        alert("Success send to Oralce database.");
    }
</script>
```

Java(Servlet)

RPA

LINUX

INDEX

RESULT

LINUX

@WebServlet(“/result”)

- 도커 컨테이너 내 Oralce DB에 대한 사용자 접근 URL, USER, PASSWORD 작성
- JSP에서 전송한 form 데이터 호출
- 해당 데이터를 DB의 2개 테이블에 나누어 저장
- 데이터가 정상적으로 DB 저장시 index.jsp로 이동

```java
@WebServlet("/result")
public class DBServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private static final String DB_URL = "jdbc:oracle:thin:@Oracle1:1521:xe";
    private static final String DB_USER = "C##calc";
    private static final String DB_PASSWORD = "1234";

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        String[] productNames = request.getParameterValues("productNames");
        String[] totalWeights = request.getParameterValues("totalWeights");
        String[] materialPrices = request.getParameterValues("materialPrices");
        String[] totalMoneys = request.getParameterValues("totalMoneys");
        String[] textures = request.getParameterValues("textures");
        String[] grades = request.getParameterValues("grades");
        String[] productWeights = request.getParameterValues("productWeights");
        String[] srWeights = request.getParameterValues("srWeights");
        String[] equipmentTonnages = request.getParameterValues("equipmentTonnages");
        String[] equipmentNumbers = request.getParameterValues("equipmentNumbers");

        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD)) {
                String calc_sql = "INSERT INTO calcdata
                  (proname, texture, grade, proweight, srweight, tonnage, cav) VALUES (?, ?, ?, ?, ?, ?, ?)";

                String result_sql = "INSERT INTO Result
                  (PRODUCT_NAME, TOTAL_WEIGHT, MATERIAL_PRICE, TOTAL_MONEY) VALUES (?, ?, ?, ?)";
            }

            response.sendRedirect(request.getContextPath() + "/index.jsp");
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            throw new ServletException("error", e);
        }
    }
}
```

Linux

backupScript.sh

- 백업을 위한 백업 Oracle과 경로 등 변수화
- 메인 Oracle에서 exp를 통한 Linux 경로에 dmp파일 생성
- dmp파일을 백업 Oracle복사한 뒤, imp를 통한 테이블 적재
- ALTER ~ RENAME ~을 이용하여 각 각의 테이블 이름 현재 시간에 맞추어 변경

```
Oracle1_container="Oracle1"
Oracle2_container="Oracle2"
Oracle_expfile="/home/oracle/backupexp.dmp"
Linux_backupdir="/home/sukhee/backupDMP/"
Timestamp=$(date +%Y%m%d%H%M)

docker exec -i $Oracle1_container exp userid=c##calc/1234 file=$Oracle_expfile tables=result, calcdata

docker cp $Oracle1_container:$Oracle_expfile $Linux_backupdir

docker cp $Linux_backupdir/$(basename $Oracle_expfile) $Oracle2_container:/home/oracle/

docker exec -i $Oracle2_container imp userid=c##calc/1234 file=/home/oracle/$(basename $Oracle_expfile)
fromuser=c##calc touser=c##calc tables=result, calcdata

docker exec -i $Oracle2_container sqlplus -S c##calc/1234 <<EOF
ALTER TABLE result RENAME TO result_${Timestamp};
ALTER TABLE calcdata RENAME TO calcdata_${Timestamp};
EXIT;
EOF
```

Crontab

- backupScript.sh를 자동화 시키기 위해 시간 설정하여 30분 마다 Oracle1 컨테이너의 테이블을 백업 Oracle에 테이블 생성

```
//crontab -e를 이용한 background process 실행
*/30 * * * * /bin/bash /home/roo/backupDMP/backupScript.sh
```

05

Evaluation

개인 평가 의견

개인 평가 의견

한석희(팀장)

목표 :

· 팀원과의 협업 및 서버 구축에 대한 개념 이해
· RPA 프로그램을 통해 사용자 편의성 및 GUI 구성
· 데이터베이스 데이터 관리 및 백업 환경 구성
· 포트 포워딩을 통한 Windows ↔ Linux(Tomcat, Oracle)간 통신 환경 구성

결과 :

· 팀원과의 업무 분배 및 의견 조율 능력 향상
· Linux 서버 환경 구성 및 도커 활용 능력 향상
· 백업 환경을 통해 데이터 손실 및 손상 방지
· 누구나 사용할 수 있는 RPA 환경 구성

원인 :

· 데이터베이스 관리 시 시퀀스 설정 미흡으로 인한 데이터 정돈 부족
· RPA 사용 경험 부족으로 인한 최적화 미흡

피드백 :

프로젝트를 진행하면서 팀원과 의사소통의 중요성을 깨달았습니다. 목표로 했던 많은 부분은 달성 했으나, 데이터베이스 관리와 RPA 부분에서는 부족함을 느꼈습니다. 이번 경험을 통해 앞으로의 발전에 중요한 교훈을 얻게 되었습니다.

이대환(팀원)

목표 :

· RPA의 구조와 Tomcat server의 이해도 향상
· Docker을 이용한 Tomcat serve와 OracleDB의 연동
· Backup 용 Oracle Database server 생성
· 협업 프로젝트를 통해 팀원과의 커뮤니케이션 스킬 향상

결과 :

· Docker을 이용한 Tomcat server와 OracleDB 의 통신 스킬 학습
· 팀원 간의 커뮤니케이션, 역할 분담 능력 향상
· RPA의 사출 원가 계산표 자동화 생성 및 적재

원인 :

· 사전 개념부족으로 인한 Docker로 Tomcat server 구현 지연
· Dead Line으로 인한 팀원 간의 역할 분담 필요

피드백 :

Docker를 사용한 Tomcat server의 관련한 지식의 부족성을 느끼게 되었습니다. 원활한 시간 활용을 위해 팀원과의 적절한 역할 분담의 중요성을 깨닫게 되었으며, 이번 프로젝트로 부족한 부분과 문제 해결을 통해 많은 교훈을 얻게 되었습니다

글로벌 직업 전문 학교

THANK YOU

한석희, 이대환