

엘리바이저 실습 보고서

2024.03.04~2024.03.29

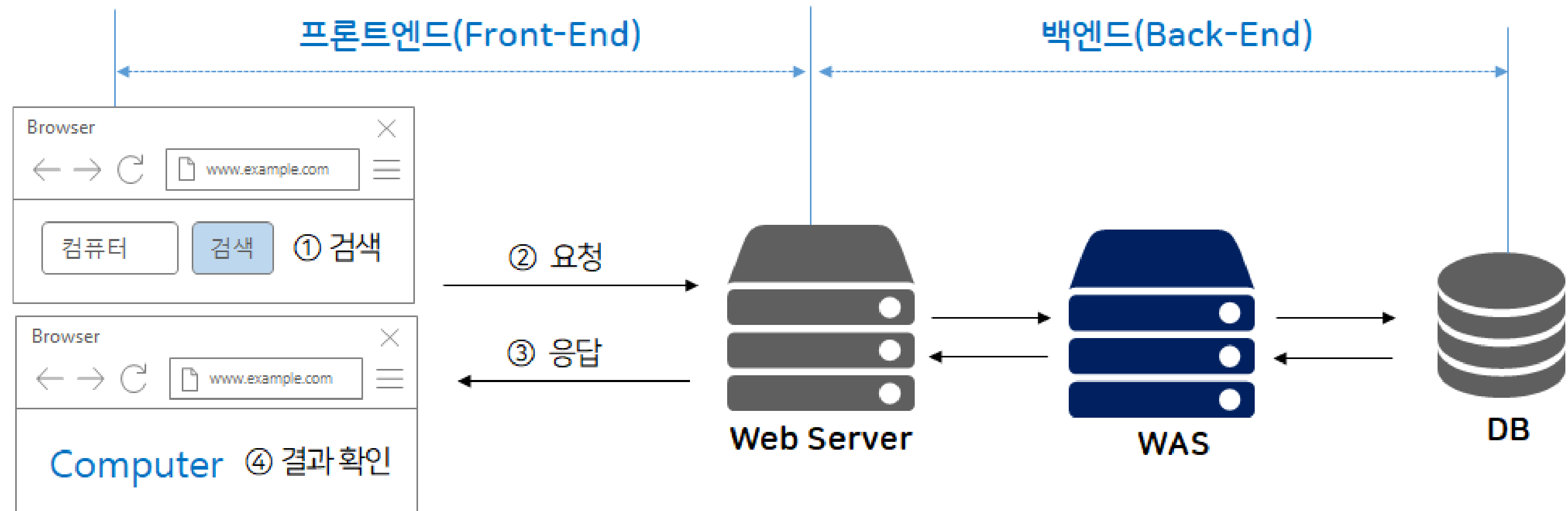
소속 : 실습생

성명 : 강지은

목차

- 1 웹 개발 환경
- 2 차트 제작 및 톰캣
- 3 리눅스 서버에 오라클 설치
- 4 DB연동(Servlet)
- 5 엘리바이저 적용

웹개발환경(WEB, WAS, DB)



- WEB: 웹브라우저(클라이언트)로부터 요청을 받아 정적인 콘텐츠(html,css)를 처리
- WAS: DB 조회나 다양한 로직 등 동적인 콘텐츠를 처리
- DB: 데이터의 집합 또는 저장소

차트제작 및 톱캣 - 차트

1 주제

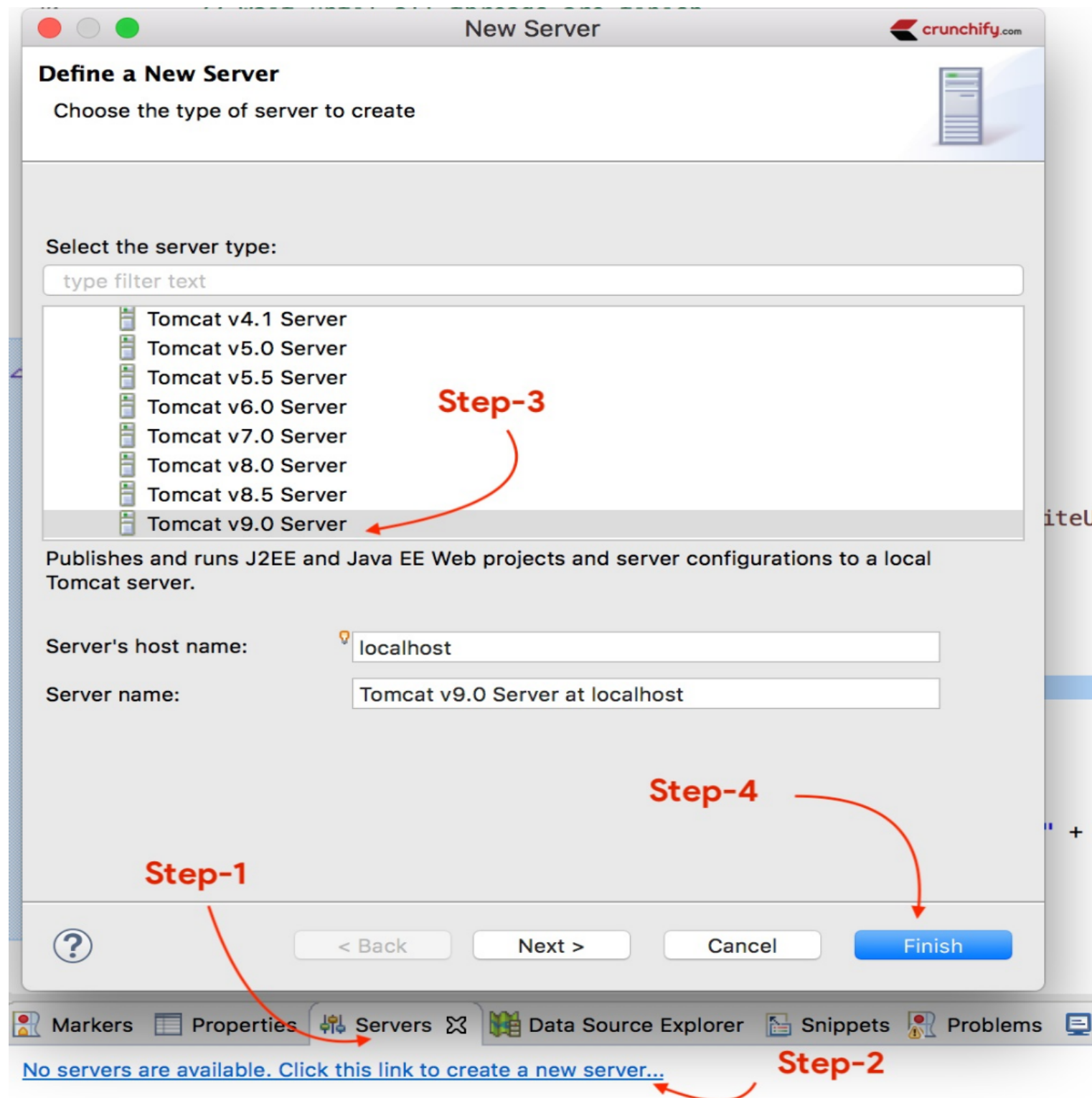
- 연도별 출생자 수 주제로 선정
- 출생자 수 데이터는 통계 기관에서 수집하고 유지하는 것이 일반적으로 데이터를 쉽게 얻을 수 있고, 시각화하기에 용이
- 연도별로 변하는 추세를 시각적으로 보여주기에 바 그래프에 사용하기 적합

2 차트

- Canvas 태그를 사용하여 생성
- X축에는 연도 Y축에는 출생자 수를 표시
- 사용자가 선택한 연도에 따라 해당 연도의 출생자 수가 변경

차트제작 및 톰캣 - 톰캣

✓톰캣설치



파일 C:/Users/user/Desktop/작업폴더/240305/index.html

localhost:8080/test/index.html

톰캣을 설치 후 주소창의 내용이 localhost 형식으로 변경

리눅스 서버에 오라클 설치

✓도커를 이용하여 가상서버에 접근 후 오라클을 설치

```
root@v7: ~  
login as: elevisor  
elevisor@192.168.0.177's password:  
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-21-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/pro  
  
Applications를 위한 확장된 보안 유지보수 비활성화됨.  
  
34개의 업데이트가 즉시 적용 가능합니다.  
추가 업데이트를 확인하려면 apt list --upgradable 을 실행하세요.  
  
ESM Apps를 (를) 활성화해서 미래의 추가적인 업데이트를 받을 수 있습니다.  
https://ubuntu.com/esm 을 참조하거나 sudo pro status 를 실행하십시오  
  
*** System restart required ***  
Last login: Mon Mar 25 08:47:23 2024 from 192.168.0.119  
elevisor@v7:~$ sudo su -  
[sudo] elevisor@v7:~$  
root@v7:~# docker ps  
CONTAINER ID   IMAGE                                COMMAND                  CREATED  
STATUS        PORTS  
6552dd70d4fa   banglamon/oracle193db:19.3.0-ee    "/bin/sh -c 'exec $O..." 13 day  
s ago        Up 13 days (healthy)    0.0.0.0:1521->1521/tcp, :::1521->1521/tcp, 5500/t  
cp          oracle19db
```

1

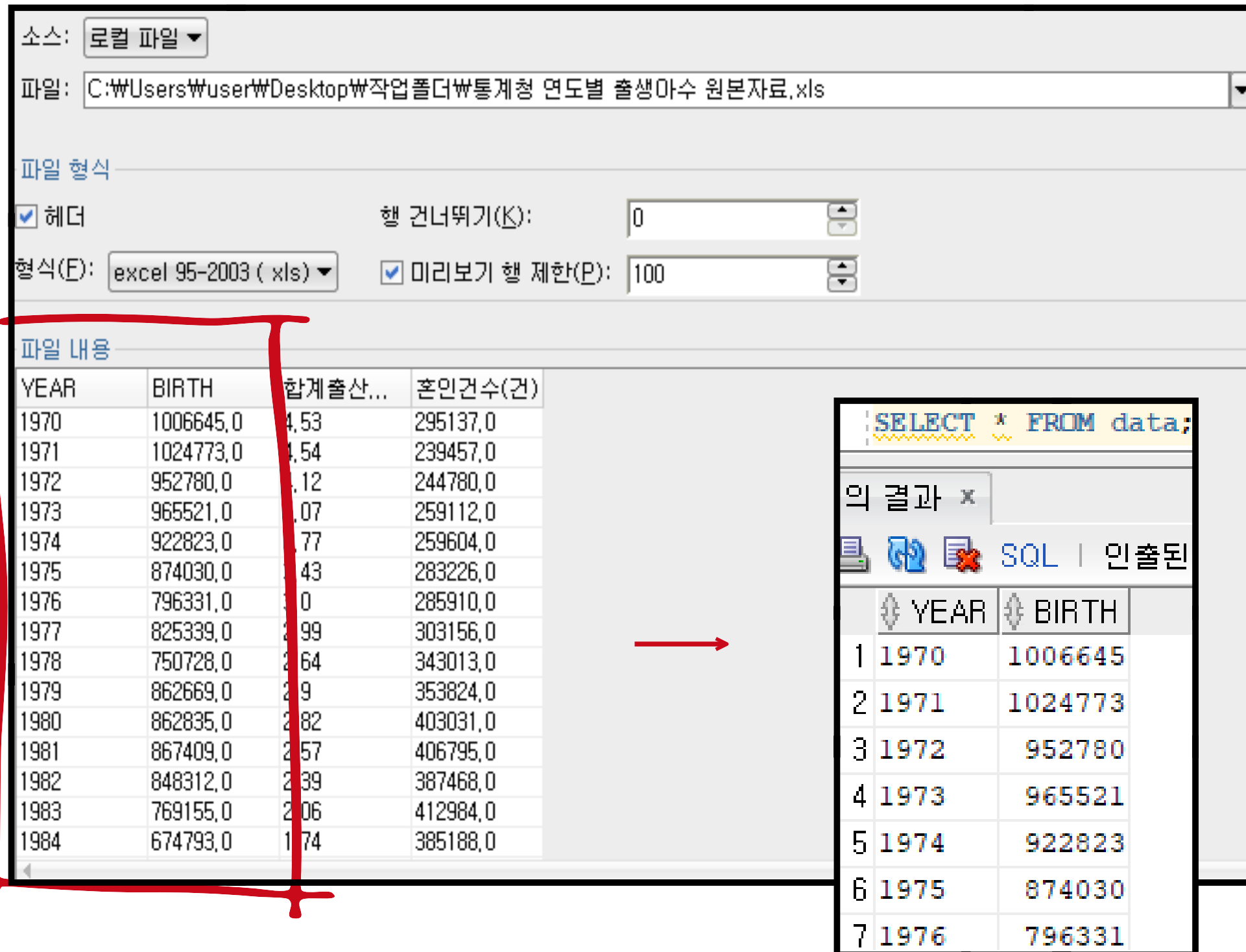
가상서버에 도커를 사용하여 오라클을 설치

2

docker ps 명령어로 오라클 컨테이너 확인 및 정상적으로 실행되고 있는지 확인.

리눅스 서버에 오라클 설치 - 데이터

✓엑셀 데이터 импорт



소스: 로컬 파일

파일: C:\Users\User\Desktop\작업폴더\통계청 연도별 출생아수 원본자료.xls

파일 형식

☒ 헤더 행 건너뛰기(K): 0

형식(F): excel 95-2003 (.xls) ☒ 미리보기 행 제한(P): 100

파일 내용

YEAR	BIRTH	합계출산...	혼인건수(건)
1970	1006645,0	4,53	295137,0
1971	1024773,0	4,54	239457,0
1972	952780,0	4,12	244780,0
1973	965521,0	4,07	259112,0
1974	922823,0	3,77	259604,0
1975	874030,0	3,43	283226,0
1976	796331,0	3,0	285910,0
1977	825339,0	2,99	303156,0
1978	750728,0	2,64	343013,0
1979	862669,0	2,9	353824,0
1980	862835,0	2,82	403031,0
1981	867409,0	2,57	406795,0
1982	848312,0	2,39	387468,0
1983	769155,0	2,06	412984,0
1984	674793,0	1,74	385188,0

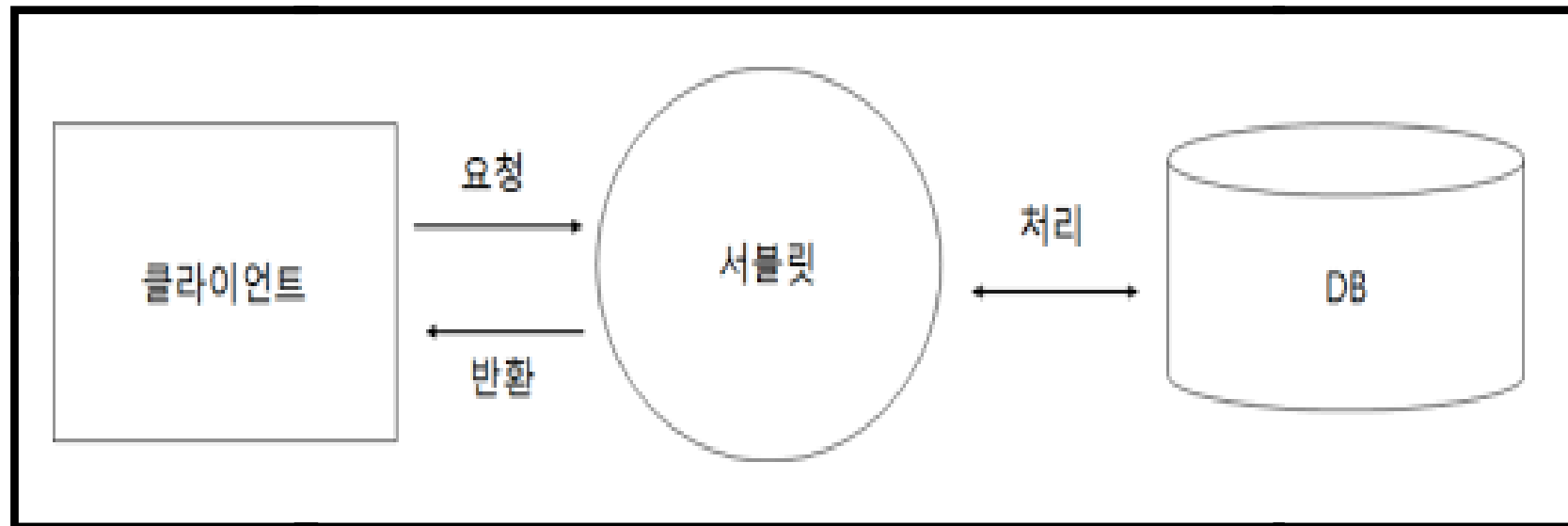
→

SQL | 인출된

	YEAR	BIRTH
1	1970	1006645
2	1971	1024773
3	1972	952780
4	1973	965521
5	1974	922823
6	1975	874030
7	1976	796331

- sql developer 에서 엑셀 데이터를 импорт
- 테이블 조회하면 원하는 테이블만 생성되어있음을 확인.

DB연동 - Servlet



- 클라이언트로부터 요청을 받으면 서블릿은 SQL 문을 이용해 DB에 접근하여 동작한다.

```
// 데이터베이스 연결 정보 설정
String url = "jdbc:oracle:thin:@192.168.0.177:1521/MONGO";
String user = "C##je";
String password = "je123";

// 데이터베이스 연결 객체 선언
Connection connection = null; //데이터베이스 연결 객체 선언
Statement statement = null; //쿼리실행 객체 선언
ResultSet resultSet = null; //쿼리 실행 결과로 반환되는 데이터를 담는 객체 선언

try {
    // JDBC 드라이버 로드
    Class.forName("oracle.jdbc.driver.OracleDriver");

    // 데이터베이스에 연결
    connection = DriverManager.getConnection(url, user, password);

    // SQL 쿼리 작성
    String sql = "SELECT * FROM data";

    // Statement 쿼리 보내고 실행
    statement = connection.createStatement();

    // 쿼리 실행
    resultSet = statement.executeQuery(sql);
}
```


DB연동 - Servlet

✓JSON형식 데이터 생성

```
// JSON 형식의 데이터 생성
StringBuilder jsonBuilder = new StringBuilder(); //생성자
jsonBuilder.append("[");
while (resultSet.next()) {
    if (jsonBuilder.length() > 1) {
        jsonBuilder.append(",");
    }
    // 각 컬럼의 값을 가져와서 JSON 형식으로 추가
    jsonBuilder.append("{");
    jsonBuilder.append("\"year\":").append(resultSet.getString("year")).append(",");
    jsonBuilder.append("\"birth\":").append(resultSet.getInt("birth"));
    jsonBuilder.append("}");
}
jsonBuilder.append("]");

// 생성된 JSON 데이터를 출력
out.println(jsonBuilder.toString());
```

✓결과

```
[{"year":1970,"birth":1006645}, {"year":1971,"birth":1024773},
{"year":1972,"birth":952780}, {"year":1973,"birth":965521},
{"year":1974,"birth":922823}, {"year":1975,"birth":874030},
{"year":1976,"birth":796331}, {"year":1977,"birth":825339},
{"year":1978,"birth":750728}, {"year":1979,"birth":862669},
{"year":1980,"birth":862835}, {"year":1981,"birth":867409},
{"year":1982,"birth":848312}, {"year":1983,"birth":769155},
{"year":1984,"birth":674793}, {"year":1985,"birth":655489},
{"year":1986,"birth":636019}, {"year":1987,"birth":623831},
{"year":1988,"birth":633092}, {"year":1989,"birth":639431},
{"year":1990,"birth":649738}, {"year":1991,"birth":709275},
{"year":1992,"birth":730678}, {"year":1993,"birth":715826},
{"year":1994,"birth":721185}, {"year":1995,"birth":715020},
{"year":1996,"birth":691226}, {"year":1997,"birth":675394},
{"year":1998,"birth":641594}, {"year":1999,"birth":620668},
{"year":2000,"birth":640089}, {"year":2001,"birth":559934},
{"year":2002,"birth":496911}, {"year":2003,"birth":495036},
{"year":2004,"birth":476958}, {"year":2005,"birth":438707},
{"year":2006,"birth":451759}, {"year":2007,"birth":496822},
{"year":2008,"birth":465892}, {"year":2009,"birth":444849},
{"year":2010,"birth":470171}, {"year":2011,"birth":471265},
{"year":2012,"birth":484550}, {"year":2013,"birth":436455},
{"year":2014,"birth":435435}, {"year":2015,"birth":438420},
{"year":2016,"birth":406243}, {"year":2017,"birth":357771},
{"year":2018,"birth":326822}, {"year":2019,"birth":302676},
{"year":2020,"birth":272337}, {"year":2021,"birth":260500}]
```

데이터베이스에서 가져온 결과를 JSON 형식으로 변환하여 출력하는데,
반복적으로 각 행의 데이터를 JSON 객체로 추가하고, 그 결과를 출력합니다.

DB연동 - 차트

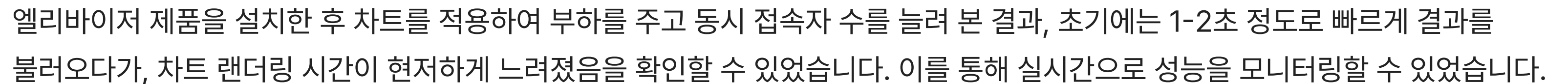
✓AJAX 통신

```
// AJAX를 사용하여 데이터를 가져와서 차트를 그리는 함수
function fetchDataGraph(startYear, endYear) {
  const xhr = new XMLHttpRequest();
  xhr.open("GET", "DataServlet", true);
  xhr.onreadystatechange = function() {
    if (xhr.readyState === XMLHttpRequest.DONE) {
      if (xhr.status === 200) {
        const data = JSON.parse(xhr.responseText);
        const filterData = data.filter(item => item.year >= startYear && item.year <= endYear);
        drawGraph(filterData);
        console.log(filterData);
      }
    }
  };
  xhr.send();
}
```

데이터를 서블릿으로부터 가져와 차트를 그립니다.

XMLHttpRequest 객체를 사용하여 서블릿으로 데이터 요청을 보내고, 응답으로 받은 데이터를 JSON 형식으로 파싱하여 필요한 연도 범위의 데이터만 추출하여 그래프를 그리게 됩니다.

✓엘리바이저 적용



실습 회고

- 다양한 기술 및 도구 학습: 이번 실습을 통해 Canvas를 이용한 차트 생성부터 웹, WAS, DB의 개념을 이해하고 톰캣과 이클립스를 연동하여 환경을 구축하는 등 다양한 기술과 도구를 학습할 수 있었습니다.
- 실제 환경에서의 적용 경험: 실습을 통해 배운 내용을 실제 환경에 적용하고, 이를 통해 이론적인 지식뿐만 아니라 실무적인 경험을 할 수 있었습니다.
- 다양한 영역에서의 경험: 웹, 데이터베이스 관리, 서버 환경 설정 등 다양한 영역에서의 경험을 통해 전반적인 IT 업무 프로세스에 대한 이해도를 높일 수 있었습니다.
- 성장과 발전: 이번 프로젝트를 통해 문제 해결 능력과 기술적 역량을 향상시키는 데에 큰 도움이 되었습니다. 또한, 새로운 기술 및 도구에 대한 이해도도 높일 수 있었습니다.

감사합니다.