전북대
소중해유(You)

로그와 에러 관리:
로그 생성, 관리 및 에러 처리 기법

전북대학교 JEONBUK NATIONAL UNIVERSITY
전북대학교 SOFTWARE중심대학사업단
SW중심대학
과학기술정보통신부
IITP 정보통신기획평가원

목차

로그란?

- 컴퓨터 시스템이나 애플리케이션에서 발생하는 다양한 이벤트나 작업을 기록한 일련의 데이터를 의미
- 로그는 시스템이 어떻게 동작하고 있는지, 어떤 이벤트가 발생했는지, 특정 작업이 언제 실행되었는지

```
// 서버 시작 로그
console.log('Server is starting...');

// 정보 로그
console.info('Connecting to the database...');

// 경고 로그
console.warn('Warning: Database connection is slow.');

// 에러 로그
console.error('Error: Failed to connect to the database.');

// 서버 실행 완료 로그
console.log('Server is running on port 3000');
```

```
Server is starting...
Connecting to the database...
Warning: Database connection is slow.
Error: Failed to connect to the database.
Server is running on port 3000
```

로그의 역할

1. 문제 진단 및 해결

- 로그는 시스템이 정상적으로 동작하지 않을 때 문제의 원인을 파악하는 데 도움을 줄 수 있음. 예를 들어, 애플리케이션이 충돌하거나 오류가 발생할 경우 로그를 통해 문제의 위치와 원인을 추적할 수 있음.

2. 운영 모니터링

- 로그는 시스템의 성능, 상태, 사용 패턴 등을 모니터링하는 데 사용. 운영팀은 로그를 분석하여 시스템의 안정성을 평가하고, 성능 개선이 필요한 부분을 식별할 수 있음.

3. 보안 감사

- 로그는 보안과 관련된 활동을 추적하는 데 중요한 역할을 수행. 예를 들어, 사용자 로그인 시도, 권한 없는 데이터 접근 시도, 네트워크 활동 등을 기록하여 보안 사고를 분석하고 대응할

로그의 역할

4. 이벤트 기록

- 로그는 애플리케이션이나 시스템에서 발생한 주요 이벤트를 기록하는 역할. 예를 들어, 파일 생성, 데이터베이스 업데이트, API 호출 등의 이벤트가 로그에 기록 되어, 이를 통해 시스템이 어떻게 사용되고 있는지를 파악할 수 있음

5. 법적 준수 및 감사

- 일부 산업에서는 법적 요구사항에 따라 특정 활동을 기록하고 보관해야 함. 로그는 이러한 법적 요구사항을 충족하는 데 중요한 자료가 될 수 있음.

로그의 중요성

- **문제 해결:** 애플리케이션에서 발생하는 오류와 예외를 추적하여 문제의 원인을 빠르게 파악할 수 있음
- **운영 모니터링:** 시스템의 상태, 성능, 사용자 활동 등을 실시간으로 모니터링
- **보안:** 보안 이벤트(예: 로그인 시도, 데이터 접근)를 기록하여 이상 징후를 감지
- **감사 및 규정 준수:** 로그는 감사 및 규정 준수 요구사항을 충족하는 데 중요한 역할
- **성능 최적화:** 애플리케이션의 성능을 분석하고 최적화하는 데 도움을 줄 수 있음. 로그를 통해 요청 처리 시간, 리소스 사용량, 병목 현상 등을 파악할 수 있으며, 이를 기반으로 시스템 성능을 개선하는데 주요한 역할

로그란?

로그의 종류

- **정보 로그 (Info Logs):** 시스템의 정상적인 동작에 대한 정보를 기록. 예를 들어, 서버 시작 시간, 사용자 로그인 성공 등이 포함
- **디버그 로그 (Debug Logs):** 개발 중에 발생하는 문제를 찾기 위해 상세한 정보를 기록. 예를 들어, 함수 호출, 변수 값 확인 등이 포함
- **경고 로그 (Warning Logs):** 잠재적인 문제가 될 수 있는 상황을 기록. 예를 들어, 메모리 사용량이 임계치에 근접했을 때의 로그 등이 포함
- **에러 로그 (Error Logs):** 시스템이 예기치 않은 상황에 직면했을 때의 에러를 기록. 예를 들어, 데이터베이스 연결 실패, 파일 접근 오류 등이 포함
- **치명적 오류 로그 (Fatal Logs):** 애플리케이션이 지속적으로 동작할 수 없을 때 발생하는 로그. 예를 들어, 치명적인 시스템 오류로 인한 서버 종료 등이 포함

Node.js에서의 로그 활용

- **기본 함수 사용**
  - **console.log 사용:**
    - console.log는 Node.js에서 가장 기본적인 로그 출력 방법이며, 이를 통해 애플리케이션의 상태나 중요한 정보를 터미널에 출력할 수 있음.
    - 특정 이벤트가 발생했을 때 이를 기록하고자 할 때 console.log를 사용함.
    - 이 방법은 개발 중에 간단하게 로그를 확인하고자 할 때 유용함.
  - **다양한 로그 레벨 사용:**
    - Node.js에서는 로그의 심각도에 따라 다양한 로그 레벨을 사용할 수 있음.
    - **console.info**는 정보성 메시지를 기록할 때 사용하며, 시스템의 정상적인 동작을 알리기 위한 용도로 사용됨. **console.warn**은 경고 메시지를 기록할 때 사용되며, 잠재적인 문제가 발생했음을 알림. **console.error**는 오류나 예외가 발생했을 때 사용되며, 시스템이 예상대로 동작하지 않을 때 기록함.
      이러한 로그 레벨을 통해 로그의 중요도와 심각도를 구분할 수 있음.

Node.js에서의 로그 활용

- **기본 함수 사용**
  - **console.log 사용:**
    - console.log는 Node.js에서 가장 기본적인 로그 출력
      에 출력할 수 있음.
    - 특정 이벤트가 발생했을 때 이를 기록하고자 할 때 co
    - 이 방법은 개발 중에 간단하게 로그를 확인하고자 할
  - **다양한 로그 레벨 사용:**
    - Node.js에서는 로그의 심각도에 따라 다양한 로그 레
    - **console.info**는 정보성 메시지를 기록할 때 사용하며,
      console.warn은 경고 메시지를 기록할 때 사용되며, 심각한 문제가 발생했음을 알림. **console.error**는 오류나 예외가
      발생했을 때 사용되며, 시스템이 예상대로 동작하지 않을 때 기록함.
      이러한 로그 레벨을 통해 로그의 중요도와 심각도를 구분할 수 있음.

```
// app.js

// 서버 시작 로그
console.log('Server is starting...');

// 정보 로그
console.info('Connecting to the database...');

// 경고 로그
console.warn('Warning: Database connection is slow.');

// 에러 로그
console.error('Error: Failed to connect to the database.');

// 서버 실행 완료 로그
console.log('Server is running on port 3000');

```

Node.js에서의 로그 활용

- **winston 라이브러리를 사용한 고급 로그 관리**
  - winston 라이브러리는 Node.js에서 로그 관리를 체계적으로 구현하기 위한 강력한 도구로, 해당 라이브러리를 사용하면 다양한 로그 레벨, 로그 포맷, 로그 전송 대상을 설정할 수 있음.
  - **winston 설치 및 사용:**
    - npm install winston 명령어를 통해 설치
    - 설치 후, 로그 레벨을 정의하고 포맷을 설정할 수 있어 편리하게 사용 가능함.
    - 포맷은 메시지의 형식을 지정하며, timestamp, log-level, message등을 포함할 수 있으며, 출력되는 대상을 선택할 수 있어 편리한 사용이 가능함.
    - 예를 들어, 에러 로그는 파일에 저장하고, 정보 로그는 콘솔에 출력하는 식으로 설정할 수 있음!!

Node.js에서의 로그 활용

- **winston 라이브러리를 사용한 로**
  - winston 라이브러리는 Node.js에서
    양한 로그 레벨, 로그 포맷, 로그 전
  - **winston 설치 및 사용:**
    - npm install winston 명령어를
    - 설치 후, 로그 레벨을 정의하고
    - 포맷은 메시지의 형식을 지정하
      어 편리한 사용이 가능함.
    - 예를 들어, 에러 로그는 파일에

```
// logger.js
const winston = require('winston');

// winston 로거 설정
const logger = winston.createLogger({
    level: 'info',
    format: winston.format.combine(
        winston.format.timestamp({
            format: 'YYYY-MM-DD HH:mm:ss'
        }),
        winston.format.printf(({ timestamp, level, message }) => {
            return `${timestamp} [${level.toUpperCase()}]: ${message}`;
        })
    ),
    transports: [
        new winston.transports.Console(),
        new winston.transports.File({ filename: 'combined.log' }),
        new winston.transports.File({ filename: 'error.log', level: 'error' })
    ]
});

module.exports = logger;

```

Node.js에서의 로그 활용

- **winston 라이브러리를 사용한 고급 로그 관리**
  - **Winston 로거 생성 및 사용:**
    - winston에서는 로거 인스턴스를 생성하여 애플리케이션 전반에 걸쳐 일관된 로그를 기록할 수 있어 일관적인 포맷팅이 가능함
    - 로거는 설정된 대로 로그를 기록하며, 이를 통해 로그를 체계적으로 관리할 수 있다는 장점!! << 꼭 사용하는 것을 권장!
    - winston의 로거는 다양한 로그 레벨을 지원하며, 필요에 따라 다른 로그 레벨을 사용할 수 있음.
    - 예를 들어, logger.info('정보 메시지'), logger.error('에러 메시지')와 같이 로그를 기록함. 이를 통해 애플리케이션의 상태, 성능, 오류 등을 일관되게 기록하고 관리할 수 있음.

Node.js에서의 로그 활용

- **winston 라이브러리를 사용한 고급 로그 관리**
  - **Winston 로거 생성 및 사용:**
    - winston에서는 로거 인스턴스를 생성하여 애플리케이션
      가능함
    - 로거는 설정된 대로 로그를 기록하며, 이를 통해 로그를
    - winston의 로거는 다양한 로그 레벨을 지원하며, 필요
    - 예를 들어, logger.info('정보 메시지'), logger.error('에러
      성능, 오류 등을 일관되게 기록하고 관리할 수 있음.

```
// app.js
const logger = require('./logger');

// 서버 시작 로그
logger.info('Server is starting...');

// 정보 로그
logger.info('Connecting to the database...');

// 경고 로그
logger.warn('Warning: Database connection is slow.');

// 에러 로그
logger.error('Error: Failed to connect to the database.');

// 서버 실행 완료 로그
logger.info('Server is running on port 3000');

```

로그 사용 예제

Winston을 사용한 로그 예제

- 기본 로그 설정

```
1  const winston = require('winston');
2
3  const logger = winston.createLogger({
4      level: 'info',
5      format: winston.format.json(),
6      transports: [
7          new winston.transports.File({ filename: 'error.log', level: 'error' }),
8          new winston.transports.File({ filename: 'combined.log' })
9      ]
10 });
11
12 if (process.env.NODE_ENV !== 'production') {
13     logger.add(new winston.transports.Console({
14         format: winston.format.simple()
15     }));
16 }
17
18 logger.info('Hello, this is an info message');
19 logger.error('Hello, this is an error message');
20
```

Winston을 사용한 로그 예제

- **로그 파일 회전**
  - 로그 파일의 크기 관리: 로그 파일이 너무 커지지 않도록 로그 파일을 회전(rotation)하는 방법.
  - 로그 보관 기간 설정: 로그 파일을 일정 기간 동안만 보관하고, 오래된 로그는 삭제하는 방법.

```
1  const winston = require('winston');
2  require('winston-daily-rotate-file');
3
4  const transport = new winston.transports.DailyRotateFile({
5      filename: 'application-%DATE%.log',
6      datePattern: 'YYYY-MM-DD',
7      zippedArchive: true,
8      maxSize: '20m',
9      maxFiles: '14d'
10 });
11
12 const logger = winston.createLogger({
13     level: 'info',
14     format: winston.format.json(),
15     transports: [
16         transport,
17         new winston.transports.Console()
18     ]
19 });
20
21 logger.info('This log will be rotated daily and archived.');
22
```

로그 사용 예제

Node.js에서 에러 핸들링

- 파일 I/O 핸들링

```
const fs = require('fs');


// 동기 코드
try {
    const data = fs.readFileSync('tmp.txt');
} catch (err) {
    console.error('Error reading file:', err);
}

// 비동기 코드 (Promise)
fs.promises.readFile('tmp.txt')
    .then(data => console.log(data))
    .catch(err => console.error('Error reading file:', err));

// 비동기 코드 (async/await)
async function readFile() {
    try {
        const data = await fs.promises.readFile('tmp.txt');
    } catch (err) {
        console.error('Error reading file:', err);
    }
}

```

로그 사용 예제

Node.js에서 에러 핸들링

- express에서 handle

```
1  const express = require('express');
2  const app = express();
3
4  // 일반적인 라우트 핸들러
5  app.get('/', (req, res) => {
6      throw new Error('Something went wrong');
7  });
8
9  // 에러 핸들링 미들웨어
10 app.use((err, req, res, next) => {
11     console.error(err.stack);
12     res.status(500).send('Something broke!');
13 });
14
15 app.listen(3000, () => {
16     console.log('Server is running on port 3000');
17 });
18
```

로그 사용 예제

Node.js에서 에러 핸들링

- express에서 handle

```
const express = require('express');
const winston = require('winston');
const morgan = require('morgan');
const path = require('path');

// winston 로거 설정
const logger = winston.createLogger({
    level: 'info',
    format: winston.format.combine(
        winston.format.timestamp({
            format: 'YYYY-MM-DD HH:mm:ss'
        }),
        winston.format.printf(({ timestamp, level, message }) => {
            return `${timestamp} [${level.toUpperCase()}]: ${message}`;
        })
    ),
    transports: [
        new winston.transports.Console(),
        new winston.transports.File({ filename: 'combined.log' }),
        new winston.transports.File({ filename: 'error.log', level: 'error' })
    ]
});

// Express 애플리케이션 생성
const app = express();

// morgan 미들웨어를 사용해 HTTP 요청 로그 기록 (winston과 연동)
app.use(morgan('combined', {
    stream: {
        write: (message) => logger.info(message.trim())
    }
}));

// 기본 라우트
app.get('/', (req, res) => {
    res.send('Hello, World!');
    logger.info('Responded to GET /');
});

// 오류 처리 미들웨어
app.use((err, req, res, next) => {
    logger.error(`Error occurred: ${err.message}`);
    res.status(500).send('Something went wrong!');
});

// 서버 실행
const PORT = 3000;
app.listen(PORT, () => {
    logger.info(`Server is running on port ${PORT}`);
});

```

로그 best practice

- **로그 레벨 설정:** 프로덕션에서는 info나 warn 레벨의 로그만 기록하고, 디버깅 로그는 제외하는 것을 권장.
- **로그의 보관 및 보안:** 로그 파일이 민감한 정보를 포함할 수 있으므로, 보관 기간을 정하고 접근 권한을 제한
- **에러 처리의 일관성:** 모든 에러를 일관된 방식으로 처리하고, 애플리케이션의 모든 부분에서 공통된 에러 핸들링 전략
- **로그와 에러 관리 도구의 선택:** 애플리케이션의 규모와 요구사항에 맞는 로그 및 에러 관리 도구를 선택

감사합니다.

- 본 온라인 콘텐츠는 2024년도 과학기술 정보통신부 및 정보통신기획평가원의 'SW중심대학사업' 지원을 받아 제작되었습니다.
- 본 결과물의 내용을 전재할 수 없으며, 인용(재사용)할 때에는 반드시 과학기술정보통신부와 정보통신기획평가원이 지원한 'SW중심대학'의 결과물이라는 출처를 밝혀야 합니다.

전북대학교 JEONBUK NATIONAL UNIVERSITY
전북대학교 SOFTWARE중심대학사업단
SW중심대학
과학기술정보통신부
IITP 정보통신기획평가원