전북대 소중해유(You)

환경 변수 관리:
보안을 위한 환경 변수 사용

전북대학교 JEONBUK NATIONAL UNIVERSITY
전북대학교 SOFTWARE중심대학사업단
SW중심대학
과학기술정보통신부
IITP 정보통신기획평가원

목차

환경 변수의 중요성

환경 변수(environment variable)란?

- 일반적으로 웹 서비스 개발에서 베이스 코드는 하나만 관리하고, 개발, 테스트, 운영 등 여러 환경에 애플리케이션을 배포하게 됨.
- 이때, 어느 환경에 배포하느냐에 따라서 다르게 설정되야하는 값들은 운영 체제 수준에서 환경 변수를 통해 관리하는 것이 기본!
- 예를 들어, 데이터베이스(DB) 설정이 그 대표적인 사례로, 운영 환경에서는 데이터 센터나 클라우드 인프라의 상용 DB를 사용하고, 개발 환경에서는 개발자의 PC에서 실행되는 로컬 DB를 사용하는 경우가 많음.
- 이때, 데이터베이스의 비밀번호나 서드파티(3rd-party) 서비스의 API 키와 같이 **민감한 인증 정보**는 GitHub와 같은 코드 저장소(repository)에 올리면 상당히 위험할 수 있기 때문에 환경 변수로 저장해놓고 사용하는 것이 권장됨.

환경 변수의 중요성

1. 보안 강화를 위한 사용

- 환경 변수는 비밀번호, API 키, 데이터베이스 연결 정보와 같은 민감한 데이터를 코드에 직접 포함하지 않고 안전하게 관리할 수 있게 도와 줌. 이를 통해, 소스 코드가 외부에 노출되거나 실수로 유출되더라도 민감한 정보는 보호될 수 있음. 즉, 환경 변수를 통해 중요한 **정보가 코드에 하드코딩되는 것을 방지**하고, 이를 **안전하게 분리하여 관리**할 수 있음.

2. 유연성 및 환경별 설정 관리

- 환경 변수를 사용하면 애플리케이션을 개발, 테스트, 프로덕션 환경 등 다양한 환경에서 동일한 코드로 실행할 수 있지만, 각 환경에 맞는 설정을 손쉽게 적용할 수 있음. 이를 통해 **개발자는 코드 수정 없이도 환경에 따라 다른 설정(예: 데이터베이스 URI, API 엔드포인트)을 적용**할 수 있으며, 배포와 관리가 용이해짐.

환경 변수의 중요성

3. 유지보수 및 관리 용이성

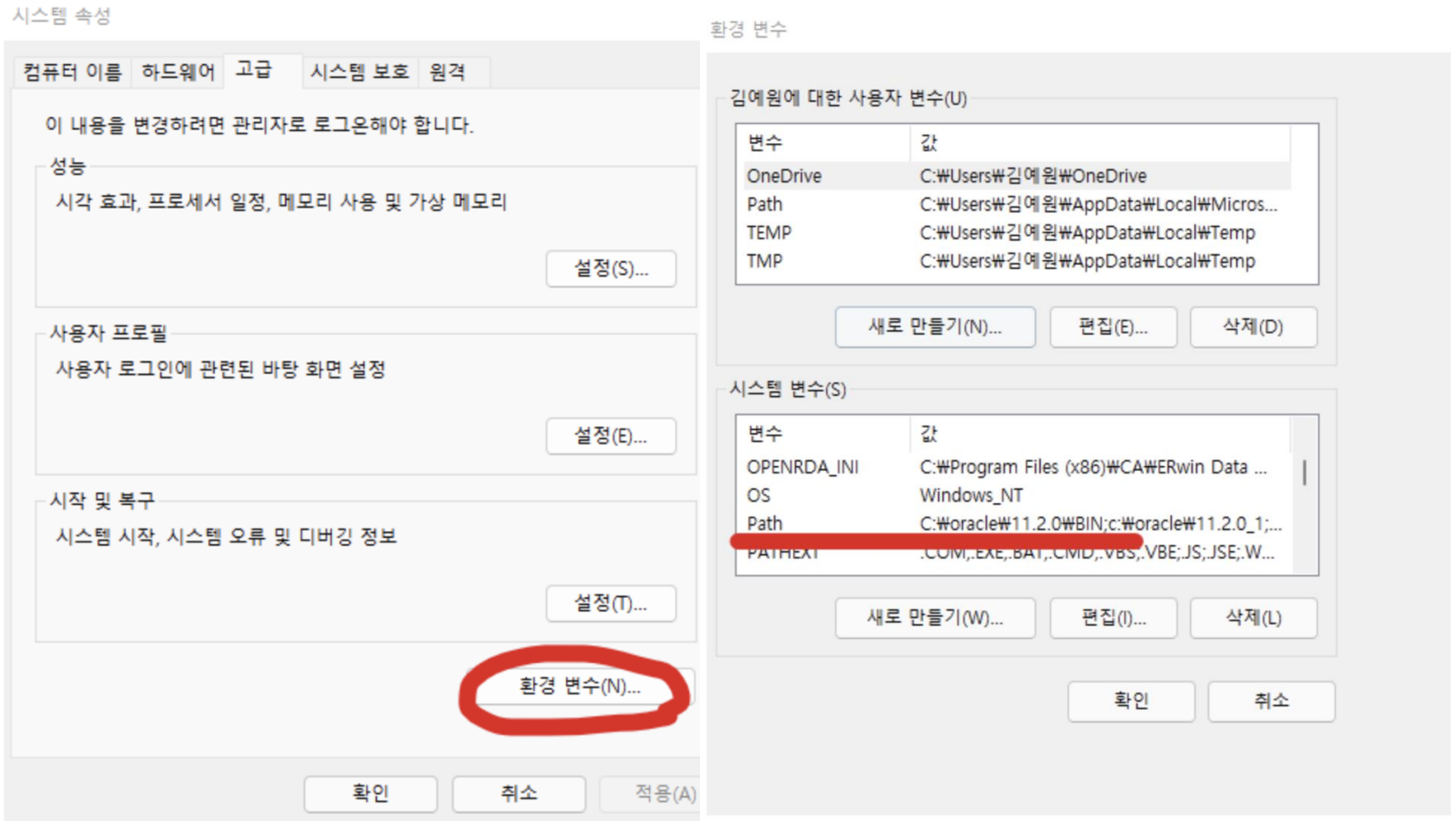
- 환경 변수는 애플리케이션 설정을 코드와 분리함으로써 **유지보수성**을 높이는데 도움을 줌. 설정 값이 변경되더라도 코드 자체를 수정할 필요 없이 환경 변수만 업데이트하면 되므로, 유지보수가 간편하고 빠르게 이루어질 수 있으며, 설정 변경 시 코드 변경이 필요 없기 때문에 배포 과정에서 발생할 수 있는 오류를 줄일 수 있음.

환경 변수 설정 방법 (운영체제 기준)

- Windows와 macOS/Linux에서 환경 변수를 설정하는 방법은 GUI를 통해 시스템 수준에서 영구적으로 환경 변수를 설정하거나, CLI 명령어를 통해 현재 세션에만 일시적으로 환경 변수를 설정할 수 있음!
- 각 방법은 상황에 따라 적절히 선택하여 사용하면 되며, 사용하는 개발 환경에 따라 선택적으로 적용하면 됨!

환경 변수 설정 방법 (운영체제 기준)

- **Windows**
  - 환경 변수 창에서 직접 설정
  - PATH에 필요한 폴더 경로를 입



환경 변수 설정 방법 (운영체제 기준)

- **MacOS / Linux**
  - CLI에서 홈 디렉토리의 ~/.bashrc 또는 ~/.zshrc에 직접 입력!
  - 파일의 마지막에 아래 명령어 삽입
  - **export MY_ENV_VAR=my_value**
  - 변경사항 적용 후, 아래 명령어 입력
  - **source ~/.bashrc # bash 사용 시**
  - **source ~/.zshrc # zsh 사용 시**

  - 또는 export 명령어 사용 가능
  - **export MY_ENV_VAR=my_value**

~/.zshrc

```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/opt/anaconda3/bin/conda' 'shell.zsh' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/opt/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/opt/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/opt/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<


[[ -d ~/.rbenv  ]] && \
  export PATH=${HOME}/.rbenv/bin:${PATH} && \
  eval "$(rbenv init -)"

export PATH="$PATH:/Users/m2air/development/flutter/bin"
```

환경 변수 설정 및 보안 관리

환경 변수 설정 방법 (Node.js에서의 설정)

- **기본적인 파일 관리를 통한 환경 변수 설정**
  - process.env 객체를 사용하여 환경 변수를 참조
  - 기본적인 사용 예제: 데이터베이스 URI, 포트 번
  - 모든 변수들은 .env 파일에 저장해서 관리해야

.env 파일

```
# 데이터베이스 설정
DB_HOST=localhost
DB_PORT=5432
DB_USER=myuser
DB_PASS=mypassword
DB_NAME=mydatabase

# 서버 설정
PORT=3000
NODE_ENV=development

# API 키 및 비밀 키
API_KEY=12345-abcde-67890-fghij
API_SECRET=abcdef1234567890

# 타사 서비스 설정
REDIS_HOST=localhost
REDIS_PORT=6379

# 이메일 설정
EMAIL_HOST=smtp.mailtrap.io
EMAIL_PORT=2525
EMAIL_USER=myemailuser
EMAIL_PASS=myemailpass

# 기타 설정
DEBUG=true
LOG_LEVEL=info
```

Node 실행

```
Welcome to Node.js v22.2.0.
Type ".help" for more information.
> require('dotenv').config();
{
  parsed: {
    DB_HOST: 'localhost',
    DB_PORT: '5432',
    DB_USER: 'myuser',
    DB_PASS: 'mypassword',
    DB_NAME: 'mydatabase',
    PORT: '3000',
    NODE_ENV: 'development',
    API_KEY: '12345-abcde-67890-fghij',
    API_SECRET: 'abcdef1234567890',
    REDIS_HOST: 'localhost',
    REDIS_PORT: '6379',
    EMAIL_HOST: 'smtp.mailtrap.io',
    EMAIL_PORT: '2525',
    EMAIL_USER: 'myemailuser',
    EMAIL_PASS: 'myemailpass',
    DEBUG: 'true',
    LOG_LEVEL: 'info'
  }
}
> process.env.DB_HOST
'localhost'
> process.env.API_KEY
'12345-abcde-67890-fghij'
> process.env.EMAIL_USER
'myemailuser'
```

환경 변수 설정 방법 (Node.js에서의 설정)

- Node.js에서 환경 변수 사용

```
// 터미널에서 환경 변수 설정
// Windows: set DB_HOST=localhost
// macOS/Linux: export DB_HOST=localhost

// Node.js 코드에서 사용
console.log("Database Host:", process.env.DB_HOST);
```

OS에서 설정한 환경 변수 사용

.env 파일에서 설정한 환경 변수 사용

```

require('dotenv').config();

const dbHost = process.env.DB_HOST;
const dbUser = process.env.DB_USER;
const dbPass = process.env.DB_PASS;

console.log(`Connecting to database on ${dbHost} with user ${dbUser}`);
```

환경 변수 보안 관리

- **비밀번호 및 API 키 관리**
  - 비밀번호, API 키, 토큰 등 민감한 정보를 환경 변수로 관리하여 코드에 **하드코딩**하지 않도록 주의!!
  - 환경 변수를 로깅하거나 디버깅 정보로 출력하지 않도록 주의!

- **환경 변수 파일 보호**
  - .env 파일은 로컬 환경에서만 사용하고, 서버에 배포할 때는 보안이 강화된 비밀 관리 도구 (예: AWS Secrets Manager, Google Cloud Secret Manager) 사용을 고려
  - Github 같은 공공 레포에 제한적으로 올리는 것을 고려! (.gitignore)
  - 프로덕션 환경에서는 환경 변수 파일을 암호화된 저장소에 보관하고, 액세스 권한을 제한

환경 변수 보안 관리

- **환경 변수 파일 암호화**
  - dotenv-safe, dotenv-encrypt 등을 사용하여 환경 변수를 안전하게 관리하는 방법도 고려할 수 있음!

  - # .env 파일을 .env.enc로 암호화
  - openssl enc -aes-256-cbc -in .env -out .env.enc

  - # .env.enc 파일을 복호화
  - openssl enc -aes-256-cbc -d -in .env.enc -out .env

환경 변수 보안 관리

- **dotenv-safe를 사용한 필수 환경 변수 검증**
  - dotenv-safe, dotenv-encrypt 등을 사용하여 환경 변수를 안전하게 관리하는 방법도 고려할 수 있음!
  - npm install dotenv-safe 설치 후 .env.example 파일을 생성하여 안전하게 관리할 수 있음.

  - require('dotenv-safe').config({
  - example: '.env.example'
  - });
  - console.log("All required environment variables are set.");

환경 변수에 기본 값 설정

- process.env를 사용할 때, 값이 설정되지 않은 경우를 대비하여 기본값을 지정하는 방법을 고려할 수 있음

```
1 const port = process.env.PORT || 3000;
2 const dbHost = process.env.DB_HOST || 'localhost';
3 const dbUser = process.env.DB_USER || 'admin';
4 const dbPass = process.env.DB_PASS || 'password';
5
6 console.log(`Server running on port ${port}`);
7 console.log(`Connecting to database on ${dbHost} with user ${dbUser}`);
8
```

환경 변수 파일의 관리 및 배포

- .env 파일의 버전 관리 제외: .env 파일을 소스 코드 관리(Git 등)에서 제외하기 위해 .gitignore 파일에 추가하여 관리 해야하는 것을 권장
- CI/CD 환경에서의 환경 변수 관리: Jenkins, GitHub Actions 등 CI/CD 도구에서 환경 변수를 관리하고 사용하는 방법을 필수적으로 고려하는 것이 권장 됨.

```
# .gitignore 파일에 추가
.env
```

환경 변수 파일의 관리 도구 및 툴 소개

- **AWS Secrets Manager:**
  - AWS에서 제공하는 비밀 관리 서비스로, 환경 변수를 안전하게 저장하고 관리할 수 있음.
- **Google Cloud Secret Manager:**
  - Google Cloud에서 제공하는 비밀 관리 서비스로, 환경 변수를 안전하게 저장하고 관리 가능.
- **Vault by HashiCorp:**
  - 비밀 관리를 위한 오픈소스 도구로, 다양한 환경에서 중요한 정보들을 안전하게 관리할 수 있게 도와줌.
- **Docker Compose와 환경 변수:**
  - Docker Compose를 사용할 때 환경 변수를 편리하게 관리할 수 있음.

환경변수 w/ MongoDB

```
const { MongoClient } = require('mongodb');
require('dotenv').config();

const uri = process.env.MONGO_URI;
const client = new MongoClient(uri);

async function connectToDatabase() {
    try {
        await client.connect();
        console.log("Connected to MongoDB!");
    } catch (err) {
        console.error(err);
    } finally {
        await client.close();
    }
}

connectToDatabase();
```

개발 환경에 따른 환경 변수 분리

- .env.development, .env.test, .env.production 파일 사용.
- NODE_ENV 변수를 사용하여 현재 환경을 구분하고, 그에 따라 다른 환경 변수를 로드하는 방법으로 구현해야 함!

```
1 const dotenv = require('dotenv');
2 const env = process.env.NODE_ENV || 'development';
3 dotenv.config({ path: `.env.${env}` });
4
5 console.log(`Running in ${env} mode`);
6 console.log("Database Host:", process.env.DB_HOST);
7
```

환경 변수 파일의 관리 모범 사례

- .env 파일을 소스 코드 관리(Git)에서 제외하기 위해 .gitignore에 추가.
- 민감한 정보는 반드시 환경 변수로 관리.
- 환경 변수를 사용할 때 기본 값을 지정하여 안전성을 높임 (process.env.VAR_NAME || 'default_value').
- 프로덕션에서는 환경 변수를 안전한 저장소나 관리 도구(예: AWS Secrets Manager, Azure Key Vault)에서 관리하는 방법 고려.

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