환경 변수 문서화 가이드

1 목표

코드 변경 없이 다양한 환경에서 서비스를 실행할 수 있도록 구성한다.

2 독자

다양한 환경에서 애플리케이션을 구성하는 개발자

3 필수 조건

- 2. 개발 환경 설정 가이드

4 단계

4.1 프론트엔드 애플리케이션

4.1.1 Dev Server 프록시 설정

Dev 서버에 대한 사용자 정의 프록시 규칙을 구성한다

```
export default defineConfig({
        ...
  server: {
    proxy: {
      // : http://localhost:5173/foo -> http://localhost:4567/foo
      '/foo': 'http://localhost:4567',
      //  : http://localhost:5173/api/bar-> http://jsonplaceholder.typicode.com/bar
      '/api': {
        target: 'http://jsonplaceholder.typicode.com',
        changeOrigin: true,
        rewrite: (path) => path.replace(/^\/api/, '')
      },
      // (RegEx) : http://localhost:5173/fallback/ -> http://jsonplaceholder.typicode.com/
      '^/fallback/.*': {
        target: 'http://jsonplaceholder.typicode.com',
        changeOrigin: true,
        rewrite: (path) => path.replace(/^\/fallback/, '')
      },
      //
      '/api': {
        target: 'http://jsonplaceholder.typicode.com',
        changeOrigin: true,
        configure: (proxy, options) => {
          // proxy  'http-proxy'
        }
      },
      //   socket.io : ws://localhost:5173/socket.io -> ws://localhost:5174/socket.io
      '/socket.io': {
        target: 'ws://localhost:5174',
        ws: true
      }
    }
  }
})
```

4.2 백엔드 설정

4.2.1 Spring Boot Property 개발 가이드

Externalized Configuration

- Spring Boot의 type-safe configuration properties를 소개한다.

1. PropertiesConfig를 추가한다.

```
@Configuration
@ConditionalOnMissingBean(ZedGatewayProperties.class)
@EnableConfigurationProperties(ZedGatewayProperties.class)
class ZedGatewayPropertiesConfig {
}
```

2. Configuration Properties를 추가한다.

```
@ConfigurationProperties("zed.gateway")
record ZedGatewayProperties(
        // property    default value   .
        @DefaultValue("localhost:8093") String origin,
        @DefaultValue("/api/v1/sessions") String sessionEntrypoint,
        @DefaultValue("http://localhost:8081") String supportAuthUri,
        @DefaultValue("/api/**") String apiEntrypoint,
        @DefaultValue("http://localhost:8091") String apiUpstreamEntrypoint
) {

    ZedGatewayProperties {
        Objects.requireNonNull(origin);
        Objects.requireNonNull(sessionEntrypoint);
        Objects.requireNonNull(supportAuthUri);
        Objects.requireNonNull(apiEntrypoint);
        Objects.requireNonNull(apiUpstreamEntrypoint);
    }
}
```

3. 환경 변수를 추가한다.
    a. .properties, .yml 파일들은 Kebab case를 사용한다
        - `acme.my-project.person.first-name`
    b. 환경 변수에는 Upper case를 사용한다
        - `ACME_MYPROJECT_PERSON_FIRSTNAME`
    c. 환경 변수와 bean property name 변환에 도움이 되는 사이트
        - Environment Variable Generator

- 추가 자료
    - Externalized Configuration을 사용해야하는 이유
        - The Twelve Factors - 3. 환경(environment)에 저장된 설정
            - 설정을 코드에서 엄격하게 분리하여 환경 변수에 저장한다.
        - Microscrvices - Externalize configuration
            - 코드 변경 없이 다양한 환경에서 서비스를 실행할 수 있도록 한다.
    - Relaxed Binding
        - https://docs.spring.io/spring-boot/docs/current/reference/html/features.html#features.external-config.typesafe-configuration-properties.relaxed-binding
        - Spring Boot는 @ConfigurationProperties 빈으로 등록되는 환경 변수들에 대해 relaxed binding rule을 적용하기 때문에 bean property name과 환경 변수의 이름이 정확히 일치할 필요가 없다
            - 일반적인 예로 대시로 구분된 환경 변수(예: context-path → contextPath) 및 대문자 환경 변수(예: PORT → port)가 있다

4.2.2 README.md 작성

- 아래와 같이 프로젝트의 README.md에 환경 별 필요한 환경 변수 설정 및 실행 script를 구성할 수 있다.

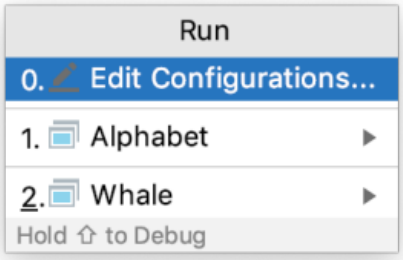
````
## Usage

### bypass

```shell
$ ZED_GATEWAY_BYPASS_ENABLED=true \
ZED_GATEWAY_APIENTRYPOINT=/api/** \
ZED_GATEWAY_APIUPSTREAMENTRYPOINT=http://localhost:8091 \
./gradlew :hr:infrastructure:gateway:zed-server:bootRun
```

### production ready

```shell
$ ZED_GATEWAY_ORIGIN=localhost:8093 \
ZED_GATEWAY_CONTENTPROVIDERID=ZED \
ZED_GATEWAY_SESSIONENTRYPOINT=/api/v1/sessions \
ZED_GATEWAY_SUPPORTAUTHURI=http://localhost:8081 \
ZED_GATEWAY_APIENTRYPOINT=/api/** \
ZED_GATEWAY_APIUPSTREAMENTRYPOINT=http://localhost:8091 \
./gradlew :hr:infrastructure:gateway:zed-server:bootRun
```
````

- IntelliJ를 통해 script를 실행시켜 애플리케이션을 실행시킬 수 있다.

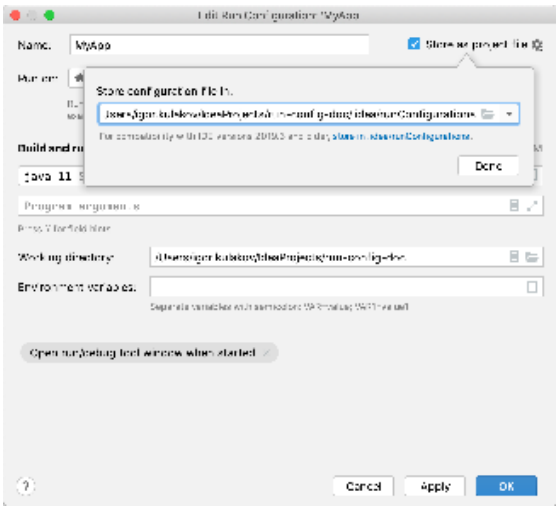
4.2.3 IntelliJ : Share run/debug configuration

run/debug configuration을 project file로 만들어 VCS를 통해 동일한 configuration을 프로젝트 멤버들과 공유할 수 있다.

1. 메인 메뉴에서 **Run | Edit Configurations** 를 선택한다. 단축키 '^ R', 이후 0.



2. 공유하고자하는 run/debug configuration를 클릭, **Store as project file** 옵션을 활성화하고 configuration을 **am-backend/.run** 하위에 위치시킨다.
    a. configuration에 Environment variables를 추가, 저장해 필요한 환경 변수를 저장할 수 있다.



4.2.4 IntelliJ : HTTP Client

IntelliJ IDEA code editor를 통해 HTTP Request를 생성, 수정, 실행할 수 있다.

1. 단축키 ' N', **HTTP Request** 선택한다.
2. HTTP Request를 구성한다.

```
### GET auth user with staffID
GET http://localhost:8091/api/v1/auth/users/0151266

### Error handling - NotFound
GET http://localhost:8091/api/v1/auth/users/1234567
```

3. HTTP 파일을 저장하여 로컬 환경에서의 테스트를 구성할 수 있다.

5 다음 단계

- 로컬 프로젝트 실행 가이드

6 추가 자료

- https://ko.vitejs.dev/config/server-options.html#server-proxy
- https://docs.spring.io/spring-boot/docs/current/reference/html/features.html#features.external-config
- https://www.jetbrains.com/help/idea/run-debug-configuration.html#share-configurations
- https://www.jetbrains.com/help/idea/http-client-in-product-code-editor.html