**JAR** (**J**ava **Ar**chive) **WAR** (**W**eb **A**pplication A**r**chive) 모두 JAVA의 jar 툴을 이용하여 생성된 압축(아카이브) 파일이며 **어플리케이션을 쉽게 배포하고 동작시킬 수 있도록 있도록 관련 파일(리소스, 속성파일 등)들을 패키징**해주는 것이 주 역할입니다.

개발하시다 보면 두 파일 형식은 많이 보셨을 거라 생각됩니다. Maven 등을 통해 내려받는 라이브러리들은 CLASS 파일들이 묶인 jar파일로 구성되어 있는 것을 확인 하실 수 있습니다. 그리고 서비스 배포시에는 프로젝트를 WAR 포맷으로 묶어서 /webapps 등의 지정된 경로에 넣고 Tomcat 등의 웹 컨테이너(Web Contaioner)를 이용하여  deploy하는 식으로 서비스를 많이 올리곤 하셨을 거에요. 그럼 이처럼 우리에게 익숙한 JAR / WAR 파일의 특징과 차이점을 간단히 알아 보겠습니다.

**| JAR ( Java Archive )**

**.jar** 확장자 파일에는 Class와 같은 Java 리소스와 속성 파일,  라이브러리 및 액세서리 파일이 포함되어 있습니다.

쉽게 JAVA 어플리케이션이 동작할 수 있도록**자바 프로젝트를 압축한 파일**로 생각하시면 되겠네요. 실제로 **JAR 파일은 플랫폼에 귀속되는 점만 제외하면 WIN ZIP파일과 동일한 구조**입니다.

JAR 파일은 **원하는 구조로 구성이 가능**하며 JDK(Java Development Kit)에 포함하고 있는 **JRE(Java Runtime Environment)만 가지고도 실행이 가능**합니다.

**|** **WAR ( Web Application Archive )**

**.war** 확장자 파일은  servlet / jsp 컨테이너에 배치 할 수 있는 **웹 어플리케이션(Web Application) 압축 파일 포맷**입니다.  JSP,  SERVLET, JAR, CLASS, XML, HTML, JAVASCRIPT 등  Servlet Context 관련 파일들로 패키징 되어있습니다. WAR는 웹 응용 프로그램를 위한 포맷이기 때문에 **웹 관련 자원만 포함하고 있으며 이를 사용하면 웹 어플리케이션을 쉽게 배포**하고 테스트 할 수 있습니다.

원하는 구성을 할 수 있는 JAR 포맷과 달리 WAR은 WEB-INF 및 META-INF 디렉토리로 **사전 정의 된 구조를 사용**하며 **WAR파일을 실행하려면 Tomcat, Weblogic, Websphere 등의 웹 서버 (WEB)또는 웹 컨테이너(WAS)가 필요**합니다.

**WAR 파일도 JAVA의 JAR 옵션( java - jar)을 이용해 생성하는 JAR파일의 일종**으로 웹어플리케이션 전체를 패키징하기 위한 JAR파일로 생각하시면 될 것 같습니다.

개요

Spring Boot를 이용해서 어플리케이션을 만들다 보면 외부에서 특정 값들을 주입받아야 하는 경우가 있다. 예를 들면 AWS의 특정 컴포넌트를 사용하기 위한 secret key가 될 수도 있고 외부 API를 사용하기 위한 API key가 될 수도 있다. 이러한 값들을 소스 코드에 하드 코딩한다면 악의적인 의도를 가진 사람이 값을 탈취하여 사용하면서 큰일로 이어질 수 있다.

따라서 이렇게 중요한 값들을 application.properties 혹은 application.yml 과 같은 외부 설정값을 관리하는 파일에 적어두고 사용하기도 하고 .jar 파일을 실행하기 위한 커맨드에서 직접 값을 넘겨주기도 한다.

이번 글은 Spring Boot에서 외부 파일(ex.application.properties, application.yml)에 있는 값들을 소스 코드에 주입해서 사용하는 방법에 대해 살펴보고자 한다.

예제로 사용된 코드는 [Github](https://github.com/pci2676/post-for-blog/tree/master/javable/ymlpropertiesbinding)에서 확인 할 수 있다.

.properties vs .yml

글에서는 application.yml을 사용할 것이기에 application.properties와 application.yml의 차이를 가볍게 짚고 넘어가고자 한다.

Spring Boot 어플리케이션을 생성하면 기본적으로 resources 디렉토리에 application.properties 파일이 생성되어있는 것을 확인할 수 있다. 이 파일을 그대로 이용해도 문제는 없지만 .yml을 이용하는 것보다 불편한 점이 있다.

만약 .properties 로 설정을 다음과 같이 작성한다면

spring.datasource.hikari.driver-class-name=org.mariadb.jdbc.Driver

spring.datasource.hikari.jdbc-url=jdbc:mariadb://localhost:3306/testdb

spring.datasource.hikari.username=root

spring.datasource.hikari.password=root

spring.datasource.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect

spring.datasource.jpa.properties.hibernate.format\_sql=true

spring.datasource.jpa.show-sql=true

spring.datasource.jpa.generate-ddl=true

.yml은 다음과 같이 표현이 가능해진다.

spring:

datasource:

hikari:

driver-class-name: org.mariadb.jdbc.Driver

jdbc-url: jdbc:mariadb://localhost:3306/testdb

username: root

password: root

jpa:

properties:

hibernate:

dialect: org.hibernate.dialect.MySQL5InnoDBDialect

format\_sql: true

show-sql: true

generate-ddl: true

.yml을 이용함으로써 계층 구조로 설정값을 표현할 수 있고 prefix의 중복 제거가 가능해진다.

참고로 .yml 을 이용하기 위해서는 SnakeYAML 라이브러리가 classpath에 존재해야 한다. spring-boot-starter 의존성은 이 라이브러리를 기본적으로 제공해준다.