**스프링 프로필 설정(Spring Profile)**

Spring 프레임워크를 쓰다 보면 로컬PC, 개발서버, 운영서버 등 여러 곳에서 같은 애플리케이션 코드를 사용하지만 몇가지 속성을 바꿔주고 싶을 때가 있습니다. 기존에는 property 파일이나 또는 별도의 설정 파일을 애플리케이션 외부에 두는 방식으로 쓰고 있었는데, Spring Profile이라는 기능에 대해 알아보고자 합니다.

1. 스프링 프로필(Spring Profile)

가장 대표적인 것은 DB 설정 변경 같은 것이 있겠지만 이번 포스팅에서는 간단하게 Root Path만 바꿔보는걸로 하겠습니다. 만약 로컬 머신이 Windows 환경이고 개발/운영 서버가 Linux 환경일 때 Root Path가 달라서 생기는 문제가 종종 있었어요. 이것을 해결하려면 외부 property 파일 등에 rootPath=C:\home 이라던가, rootPath=/home/... 등으로 써두곤 해당 프로퍼티 파일을 읽어 내부 로직에서 활용했습니다. 하지만 이 방법은 war 파일로 묶어서 배포 할 때 문제점이 외부 프로퍼티 파일 신경 쓰기가 참 애매하다는거죠.

스프링 프로필 기능을 활용하면 war 배포할때도 web.xml 파일 또는 jvm 옵션을 줘서 애플리케이션을 옵션에 맞게 돌릴 수 있어요. 우선 코드를 보는게 빠를 것 같네요.

Spring Profile 기능은 3.1 버전 이상부터 지원합니다.

1. 프로필 기능에서 사용할 클래스 만들기

우선 간단하게 프로필 기능에서 활용할 기본 클래스를 하나 만들겠습니다. 이름은 ProfileCls 에요.

|  |
| --- |
| public class ProfileCls {  private String rootPath;  public String getRootPath() {  return rootPath;  }  public void setRootPath(String rootPath) {  this.rootPath = rootPath;  }  } |

보시다시피 매우 간단한 클래스입니다. rootPath라는 변수 하나를 set/get 할 수 있는 클래스입니다. rootPath를 로컬/개발/운영 서버에 맞게끔 호출 할 수 있도록 해봅시다.

* 1. djkfjl

1. 프로필 설정
   1. XML 파일 설정

root-context.xml 파일에 해당 내용을 기술했지만 필요 여하에 따라 다른 XML 파일에도 기술 가능합니다. 다음처럼 작성 해보세요.

|  |
| --- |
| <beans profile="local">  <bean id="ProfileCls" class="your.package.name.ProfileCls">  <property name="rootPath" value="C:/home/profile"/>  </bean>  </beans>  <beans profile="dev">  <bean id="ProfileCls" class="your.package.name.ProfileCls">  <property name="rootPath" value="/home/dev"/>  </bean>  </beans>  <beans profile="production">  <bean id="ProfileCls" class="your.package.name.ProfileCls">  <property name="rootPath" value="/home/release"/>  </bean>  </beans> |

3개의 프로필을 설정했습니다. 로컬 머신에서는 windows 환경이라는 가정이었고, 개발(dev)/운영(production) 서버는 linux 환경이라는 가정으로 만든 XML 파일입니다.

1. 프로필 적용 방법
   1. web.xml 수정

프로필 설정이 끝났으면 이번엔 web.xml 파일을 수정해볼 차례입니다. 이 부분을 스위칭 해서 프로필을 변경 할 수 있습니다. 중복 기입도 가능합니다.

|  |
| --- |
| ...  <!-- use profile -->  <context-param>  <param-name>spring.profiles.active</param-name>  <param-value>local</param-value>  <!-- <param-value>local, dev</param-value> -->  </context-param>  ... |

이제 해당 어플리케이션은 "local" 프로필로 설정 되었습니다. 정말로 그렇게 돌아가는지 확인도 해볼 필요가 있겠죠.

* 1. JVM option 설정

web.xml 외에도 별도의 WAS를 사용하고 있을 때 (톰캣이라던가, 톰캣이라던가?) JVM 옵션을 통해 프로필 설정도 가능합니다.

|  |
| --- |
| -Dspring.profiles.active=local,dev |

여러개를 설정할 수도 있습니다. 톰캣은 /tomcatPath/bin/catalina.sh에 JAVA\_OPTS="$JAVA\_OPTS -Dspring.profiles.active=local,dev" 같은 형식으로 넣어주세요.

1. 현재 프로필 확인

현재 프로필이 어떤것인지 확인하기 위해선 다음처럼 사용하면 됩니다. 컴포넌트가 하나 있다는 가정으로 시작하겠습니다. 컨트롤러도 가능합니다.

|  |
| --- |
| @Component("your.package.name.Comp")  public class Comp {  private static final Logger logger = LoggerFactory.getLogger(Comp.class);  ...  @Autowired private ConfigurableWebApplicationContext subContext;  @Autowired private Environment env;  ...  // ConfigurableWebApplicationContext 이용  String[] profiles = subContext.getEnvironment().getActiveProfiles();  for(String profile : profiles){  logger.debug( profile );  }  // Environment 이용  profiles = env.getActiveProfiles();  for(String profile : profiles){  logger.debug( profile );  }  ...  } |

1. 이젠 정말로 사용하자

프로필 XML, 프로필에 사용할 클래스, web.xml 변경, 현재 프로필 설정까지 전부 확인했다면 이제 쓰는 것 밖에 안남았습니다. 실제로 사용하기 위해 다음처럼 해봅시다.

|  |
| --- |
| ...  @Autowired private ProfileCls profileCls;  ...  logger.debug(profileClas.getRootPath()); // C:/home/profile 라는 문자열이 나옵니다. |

이걸