Practice #1

연습 전 프로젝트세팅

1. https://github.com/yimsb01/team-outback 의 저장소를 SourceTree에서 clone한다.
2. 자신의 이름으로 된 branch를 만들어서 checkout한다.
3. 명령 프롬프트를 열고 [git\team-outback\assignments\자신의 이름] 경로로 가서 gradle init 실행
4. 만들어진 프로젝트의 폴더에 들어가서 build.gradle - plugins 에 아래와 같이 추가한다.
   (프로젝트를 웹 어플리케이션 프로젝트로 빌드하기 위함임)
   - id 'war'
   - id 'eclipse-wtp'
5. build.gradle의 가장 밑에 있는 application {mainClassName = '....'} 지우기
   (mainClassName에 경로가 com.eomcs.lms.App.class 등으로 되어 있으면 git에 clone한 루트폴더에 bin/com/eomcs/lms 폴더를 만들어 App.class를 생성해버려서 프로젝트 폴더가 꼬여버린다.)
6. 자신의 이름 폴더에서 gradle eclipse를 실행하여 빌드하고 이클립스에 import
7. src/main/java와 src/test/java에 있는 App.java와 AppTest.java 지우기

1. Spring IoC 컨테이너 사용 연습

- Spring IoC 컨테이너에 bean(객체)을 생성해 저장하고 꺼내 쓰는 연습

필요한 세팅 - *spring-context*

build.gradle의 dependencies에 Spring IoC 컨테이너를 추가하고 빌드
(mvnrepository.com에서 spring context 검색)

프로세스

- domain 패키지를 만들고 그 안에 Student 클래스를 만든다.
  - 클래스 멤버 : int no / String name / String name / Date registeredDate
    getter() / setter() / toString()
- AppConfig 클래스를 만든다.
- main()을 포함하는 Practice01 클래스를 만든다.
- Student 객체를 만들 때 new Student()로 만드는 것이 아닌 Spring IoC 컨테이너를 사용하여 자동으로 만들게 하고 setter()를 호출하여 값을 세팅하고 toString()을 사용하여 해당 객체를 화면에 출력

  출력 예

  : Student [no=1, name=홍길동, email=hong@test.com, registeredDate=2019-09-16]

Keyword

ApplicationContext, AnnotationConfigApplicationContext, @Component

Tip

- bitcamp-java-basic의 ch29를 참고

- getBean에서 첫번째 파라미터에 bean의 이름을 주고 두 번째 파라미터에 클래스 타입을 주면 (~~~.class) 리턴타입이 Object가 아닌 해당 클래스의 객체가 된다.

2. Spring IoC 컨테이너 사용 연습 2 (메소드 호출)

- **계산기 프로그램** : spring에 담겨져 있는 객체에서 메소드를 꺼내와서 그 메소드를 저장하는 클래스를 만들고, 입력을 받았을 때 해당 메소드를 호출하여 계산을 한 결과를 출력하는 프로그램을 작성

필요한 세팅 - *spring-webmvc*

build.gradle의 dependencies에 spring-webmvc를 추가하고 빌드
(mvnrepository.com에서 spring web mvc 검색)

프로세스

- Calculator 클래스를 만든다.
- AppConfig 클래스를 만든다. (1번에서 사용하던 것을 써도 무방함)
- 메소드를 보관할 클래스인 RequestMappingHandlerMapping를 만든다.
- main()을 포함한 Practice02 클래스를 만든다.
- 아래와 같이 입출력을 받는다.

  계산 커맨드? add

  값1? 3

  값2? 4

  > 7

Notes

- bitcamp-java-application4-server의 App에서 처럼 프록시 객체를 걸러내는 과정을 넣지 않고 모든 객체의 메소드를 체크한다.
- 너무 어렵게 생각하지 말것; main()에 쓰여진 실제로 돌아가는 코드만 보면 30줄 내외이다.

Keyword

@RequestMapping

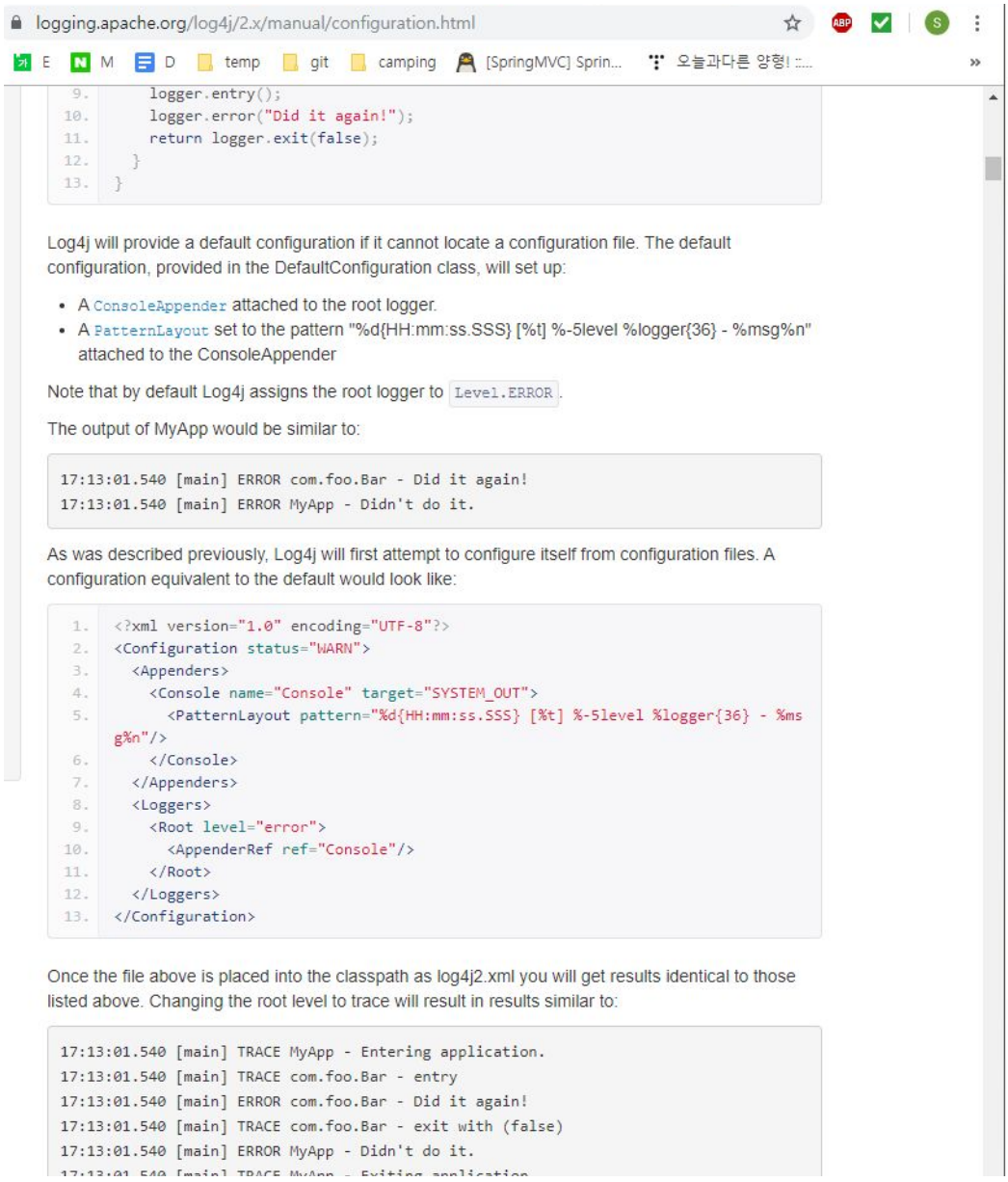
Tip

bitcamp-java-application-server의 App.java를 참고

3. Log4J2를 사용하여 출력

필요한 세팅 - *log4j-core 및 xml 설정*

- build.gradle의 dependencies에 log4j-core를 추가하고 빌드
  (mvnrepository.com에서 Apache Log4j Core 검색)
- src/main/resources에 log4j2.xml이라는 파일을 만든다.
- https://logging.apache.org/log4j/2.x/manual/configuration.html 에서 스크롤을 내리다 보면 xml 관련 세팅이 템플릿이 있다. log4j2.xml에 코드를 복붙한다.



프로세스

- log4j2.xml에서 Loggers 프로퍼티 안의 Root level를 INFO 레벨로 바꾼다
- Practice01에서 System.out.println()을 사용한 부분을 logger.info()로 출력한다.

Keyword

log4j Core

Notes

[bitcamp-docs]-[log4j] 경로의 log4j.md의 내용도 참고

Github에 업로드

- 자신의 브랜치를 checkout하고 commit하여 로컬 브랜치에 저장한다.
- master 브랜치를 checkout하고 pull을 실행하여 컴퓨터 로컬저장소의 master 브랜치를 최신 상태로 만든다.
- 자신의 브랜치에 master 브랜치의 변경사항을 merge해 와서 업데이트 하고 commit-push한다.
    - conflict가 났다면 해결하고 commit한 뒤 push한다.
- master 브랜치를 checkout하고 자신의 브랜치를 master 브랜치에 merge하고 commit-push한다.
    - conflict가 났다면 해결하고 commit한 뒤 push한다.

(물론 서로 다른 폴더를 수정했기 때문에 conflict가 날 가능성은 없다.)