31장. 스프링 Tiles Log AOP 기능

1. 뷰리졸버 설정 없이 기능별로 해당 폴더에 쉽게 접근하기

MemberControllerImpl 클래스 getViewName() 메서드 수정

http://localhost:8080/shi/member/listMembers.do

contextPath : /shi
uri : /shi/member/listMembers.do

begin : 4
end : 26

viewName : /member/listMembers.do
최종 viewName : /member/listMembers

2. log4j란?

❖println() 메서드 사용 시 유지 보수 시 불편함

2. log4j란?

log4j

- 로그 기능을 제공하는 오픈 소스 라이브러리
- 애플리케이션에서 웹 사이트에 접속한 사용자 정보나 각 클래스의 메서드 호출 시각 등 여러 가지 정보를 로그로 출력해서 관리
- 메이븐에선 프로젝트 생성 시 자동으로 log4j 라이브러리가 설치됨

2. log4j란?

log4j.xml을 이루는 태그

태그	설명
〈Appender〉	로그의 출력 위치를 결정(파일, 콘솔, DB 등)함. log4J API 문서의 XXXAppender로 끝나는 클래스들의 이름을 보면 출력 위치를 알 수 있음.
〈Layout〉	Appender가 어디에 출력할 것인지 결정했다면 어떤 형식으로 출력할지 출력 레이아웃을 결정함.
〈Logger〉	로깅 메시지를 Appender에 전달함. 개발자가 로그 레벨을 이용해 로그 출력 여부를 조정할 수 있음. logger는 로그 레벨을 가지고 있으며, 로그의 출력 여부는 로그문의 레벨과 로거의 레벨을 가지고 결정함.

2. log4j란?

여러 가지 Appender 클래스

Appender 클래스	설명
ConsoleAppender	org.apache.log4j.ConsoleAppender 클래스로, 콘솔에 로그 메시지를 출력함.
FileAppender	org.apache.log4j.FileAppender 클래스로, 파일에 로그 메시지를 출력함.
RollingFileAppender	org.apache.log4j.rolling.RollingFileAppedner 클래스로, 파일 크기가 일정 기준을 넘으면 기존 파일을 백업 파일로 바꾸고 처음부터 다시 기록함.
DailyRollingAppender	org.apache.log4j.DailyRollingFileAppender 클래스로, 설정한 기간 단위로 새 파일을 만들어 로그 메시지를 기록함.

2. log4j란?

PatternLayout 클래스에서 사용되는 여러 가지 출력 속성들

속성	설명
%p	debug, info, error, fatal 등 로그 레벨 이름 출력
%m	로그 메시지 출력
%d	로깅 이벤트 발생 시각 출력
%F	로깅이 발생한 프로그램 파일 이름 출력
%l	로깅이 발생한 caller의 정보 출력
%L	로깅이 발생한 caller의 라인 수 출력
%M	로깅이 발생한 method 이름 출력
%c	로깅 메시지 앞에 전체 패키지 이름이나 전체 파일 이름 출력
…	…

2. log4j란?

log4j의 여러 가지 로그 레벨들

속성	설명
FATAL	시스템 차원에서 심각한 문제가 발생해 애플리케이션 작동이 불가능할 경우에 해당하는 레벨임. 일반적으로 애플리케이션에서는 사용할 일이 없음.
ERROR	실행 중 문제가 발생한 상태를 나타냄.
WARN	향후 시스템 오류의 원인이 될 수 있는 경고 메시지를 나타냄.
INFO	로그인, 상태 변경과 같은 실제 애플리케이션 운영과 관련된 정보 메시지를 나타냄.
DEBUG	개발 시 디버깅 용도로 사용한 메시지를 나타냄.
TRACE	DEBUG 레벨보다 상세한 로깅 정보를 출력하기 위해 도입된 레벨임.

2. log4j란?

log4j.xml에서 로그 레벨을 info로 설정했기 때문에 debug() 메서드로 설정한 메시지는 레벨이 낮아 출력되지 않음.

log4j.xml의 로그 레벨을 debug로 변경한 후 브라우저에서 요청하면 이번에는 debug 레벨 메시지와 상위의 info 레벨 메시지가 모두 출력됨.

2. log4j란?

- C:\\workspace-spring\\logs폴더에는 날짜별로 output.log 파일이 생성됨

2. log4j란?

- 마이바티스 SQL문을 로그로 출력하기

3. AOP

- **AOP** 용어 **(review)**

3. AOP

- **Advice** 종류

Around 어드바이스	▪ 타겟의 메서드가 호출되기 이전(before) 시점과 이후(after) 시점에 모두 처리해야 할 필요가 있는 부가기능을 정의한다. ➔ Joinpoint 앞과 뒤에서 실행되는 Advice
Before 어드바이스	▪ 타겟의 메서드가 실행되기 이전(before) 시점에 처리해야 할 필요가 있는 부가기능을 정의한다. ➔ Joinpoint 앞에서 실행되는 Advice
After Returning 어드바이스	▪ 타겟의 메서드가 정상적으로 실행된 이후(after) 시점에 처리해야 할 필요가 있는 부가기능을 정의한다. ➔ Jointpoint 메서드 호출이 정상적으로 종료된 뒤에 실행되는 Advice
After Throwing 어드바이스	▪ 타겟의 메서드가 예외를 발생된 이후(after) 시점에 처리해야 할 필요가 있는 부가기능을 정의한다. ➔ 예외가 던져질 때 실행되는 Advice

3. AOP

- **AOP** 용어

포인트 컷(Pointcut)

- 어드바이스를 적용할 타겟의 메서드를 선별하는 정규표현식이다.
- 포인트컷 표현식은 execution으로 시작하고, 메서드의 Signature를 비교하는 방법을 주로 이용한다.

애스펙트(Aspect)

- 애스펙트는 AOP의 기본 모듈이다.
- 애스펙트 = 어드바이스 + 포인트컷
- 애스펙트는 싱글톤 형태의 객체로 존재한다.

3. AOP

- **AOP** 용어

지시자 종류

지시자	기능
execution	–가장 정교한 포인트컷을 만들 수 있음 –리턴타입 , 패키지, 클래스명, 메서드명,메서드 매개변수에 포인트컷 지정
within	–타임패턴 내에 해당하는 모든 것들을 포인트컷 지정
bean	–bean이름으로 포인터컷 지정

PointCut 표현식 문법

- AspectJ 포인트컷 표현식은 포인트컷 지시자를 이용하여 작성한다.
- 포인트컷 지시자 중에서 가장 대표적으로 사용되는 것은 execution()이다.
- execution() 지시자를 사용한 포인트컷 표현식의 문법구조는 다음과 같다.

public, private과 같은 접근제한자, 생략가능

패키지와 클래스 이름에 대한 패턴, 생략가능 사용할 때 '.'을 사용해 연결함

메서드 이름 타입패턴

execution([접근제한자 패턴] 타입패턴 [타입패턴.] 이름패턴 (타입패턴 | " .. ", ...)

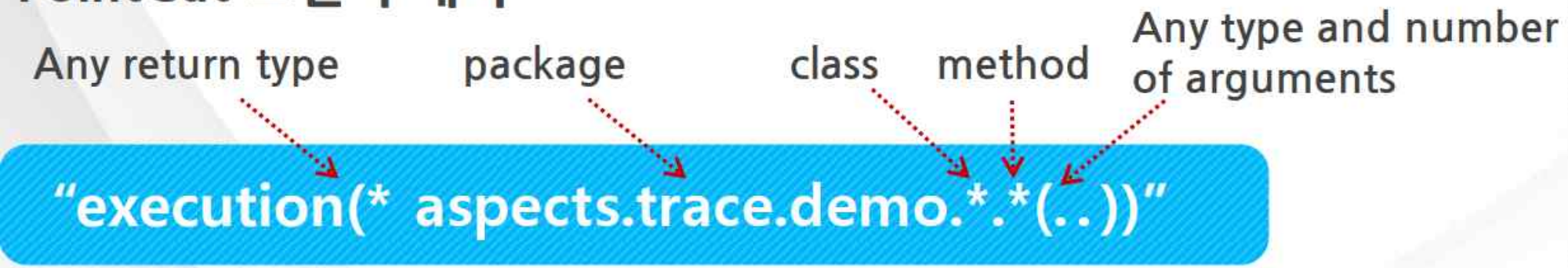
[throws 예외패턴])

예외이름패턴

리턴값의 타입 패턴

파라미터의 타입 패턴을 순서대로 넣을 수 있다.
와일드카드를 이용해 파라미터 개수에 상관없는 패턴을 만들 수 있다.

PointCut 표현식 예시

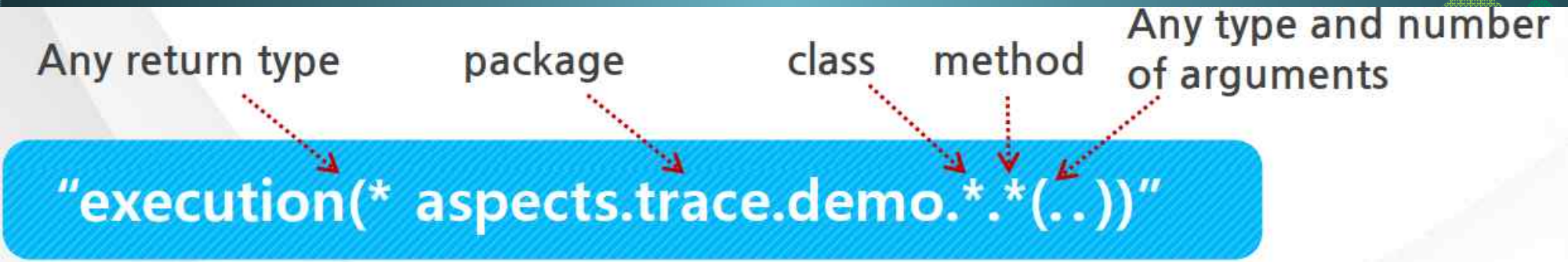


01 execution(* hello(..))

- hello라는 이름을 가진 메서드를 선정하는 것이다.
  파라미터는 모든 종류를 다 허용한다.

02 execution(* hello())

- 파라미터 패턴이 ()로 되어 있으니
  hello 메서드 중에서 파라미터가 없는 것만 선택한다.

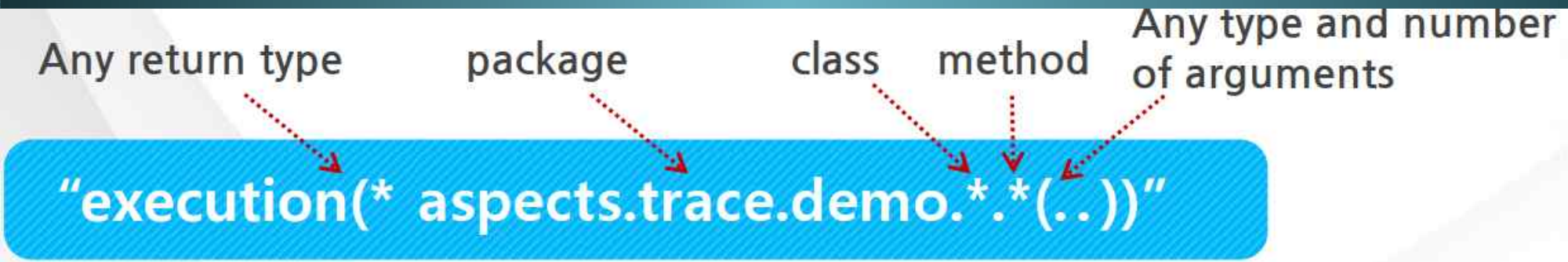


03 execution(* myspring.user.service.UserServiceImpl.*(..))

- myspring.user.service.UserServiceImpl 클래스를 직접 지정하여 이 클래스가 가진 모든 메서드를 선택한다.

04 execution(* myspring.user.service.*.*(..))

- myspring.user.service 패키지의 모든 클래스에 적용된다. 하지만 서브패키지의 클래스는 포함되지 않는다.



05 execution(* myspring.user.service..*.*(..))

- myspring.user.service 패키지의 모든 클래스에 적용된다.
  그리고 '..' 를 사용해서 서브패키지의 모든 클래스까지 포함한다.

06 execution(* *..Target.*(..))

- 패키지에 상관없이 Target이라는 이름의 모든 클래스에 적용된다.
  다른 패키지에 같은 이름의 클래스가 있어도 적용이 된다는 점에
  유의해야 함

3. AOP

여러가지 포인트컷 표현식

Pointcut	JointPoints
execution(public *.*(..))	public 메서드 실행
execution(* set*(..))	이름이 set으로 시작하는 모든 메서드명 실행
execution(* get*(..))	이름이 get으로 시작하는 모든 메서드명 실행
execution(* com.xyz.service.AccountService.*(..))	AccountService 인터페이스의 모든 메서드 실행
execution(* com.xyz.service.*.*(..))	service 패키지의 모든 메서드 실행
execution(* com.xyz.service..*.*(..)	service 패키지와 하위 패키지의 모든 메서드 실행
within(com.xyz.service.*)	service 패키지 내의 모든 결합점(클래스 포함)
bean(*Repository)	이름이 "Repository"로 끝나는 모든 빈
bean(*)	모든 빈
bean(board*)	이름이 "board"로 시작하는 모든빈
bean("*service) || bean(*Service)	이름이 "service"나 "Service"로 끝나는 모든 빈

감사합니다.