Spring Framework

JAVA ENTERPRISE APPLICATION 개발에 사용되는 자바플랫폼을 위한 오픈소스 어플리케이션 프레임워크이다

특징

- 자바 객체를 담고 있는 경량 컨테이너이다
  객체의 생성, 소멸과 같은 라이프 사이클을 관리하며 스프링으로 부터 필요한 객체를 얻어올 수 있다

- POJO(Plain Old Java Object) 기반의 개발

- 제어 반전(IoC - Inversion of Control)을 지원
  컨트롤의 제어권이 사용자가 아니라 프레임워크에 있어서 필요에 따라 스프링에서 사용자 코드를 호출할 수 있다

- 의존성 주입 DI (Dependency Injection)을 지원
  설정파일을 통해서 객체간의 의존 관계를 설정할 수 있다

- AOP(Aspect Oriented Programming) 지원

- 트랜잭션 처리를 위한 방법을 제공

- 영속성과 관련되 다양한 서비스를 제공
  myBatis, hibernate등 데이터베이스 처리 라이브러리와 연결할 수 있는 인터페이스를 제공한다.

- 동적인 웹 사이트 개발하기 위한 여러가지 서비스를 제공
  우리나라에서는 공공기관의 웹 서비스 개발 시 사용을 권장하는 전자정부 표준프레임워크의 기반 기술로 사용되고 있다

- MVC Framework를 제공

- 인터페이스 베이스 설계와 스프링을 활용함으로써 소스 파일은 변경하지 않고 스프링 설정 파일만 변경해서 다양한 객체를 생성하는, 변경이 유연한 어플리케이션을 작성할 수 있게 되었다

- 스프링을 사용하는 이유는 "**필요한 인스턴스를 스프링에서 미리 생성해 준다**" 라는 장점을 얻을 수 있다.

- 스프링은 자주 변경이 되거나 컴포넌트의 재활용이 높은 유연한 어플리케이션을 작성할 수 있게 하며 테스트도 쉽다.

- 스프링은 프레임워크지만 스트럿츠처럼 웹 어플리케이션 전용의 프레임워크와는 다른 특징이 있다

- 스프링은 '어플리케이션 프레임워크'로 불린다
  콘솔 어플리케이션나 스윙과 같은 GUI 어플리케이션등 어떤 어플리케이션에도 적용 가능한 프레임워크이다.

- 스프링은 EJB와 같이 복잡한 순서를 거치지 않아도 간단하게 이용할 수 있기 때문에 "경량 컨테이너" 라고 부른다.

- 스프링은 **Dependency Injection(DI)와 Aspect Oriented Programming(AOP)을 가장 중점적인 기술**로 사용하지만 여러가지 기능도 제공하고 있다.

- 스프링은 크게 나눠 7개 모듈로 구성되어 있으며, 필요에 따라 이들을 서로 조합하여 사용할 수 있다
- Spring Core
- Spring AOP
- Spring ORM
- Spring DAO
- Spring Web
- Spring Context
- Spring Web MVC

1. 스프링 설치

https://spring.io

기존에는 공식사이트에서 zip 압축파일을 받아 jar를 구했는데 지금 사이트에서 Maven, Gradle로 라이브러리를 받게 끔 하고 있다

2. 스프링 편집기 - STS

http://dist.springsource.com/release/STS/index.html --- X
https://spring.io/tools
https://github.com/spring-projects/toolsuite-distribution/wiki/Spring-Tool-Suite-3

http://commons.apache.org/logging/download_logging.cgi

Spring Framework 은 JDK 11, Tomcat 9.0, Spring 5.x 사용해야 잘 처리가 된다.

Spring 6.x 에서는 Tomcat 9.0 과 연결이 제대로 되지 않는다.

=> The superclass "jakarta.servlet.http.HttpServlet" was not found on the Java Build Path 에러 발생한다.

그래서 STS.ini 파일에서 JDK 11로 설정하고, 프로젝트를 생성한 뒤 JDK 버전을 바꾸어서 사용하면 된다.

=> Spring Legacy Project가 아직 JAVA 17을 지원하지 않아서 발생하는 에러이다.

STS.ini

```
-product
org.springsource.sts.ide
--launcher.defaultAction
openFile
-vm
C:\Program Files\Java\jdk-11\bin\javaw.exe
-vmargs
-Dosgi.requiredJavaVersion=11
-Dosgi.dataAreaRequiresExplicitInit=true
-Xms256m
-Xmx2048m
--add-modules=ALL-SYSTEM
-Dosgi.module.lock.timeout=10
-javaagent:D:\Spring\sts-bundle\sts-3.9.18.RELEASE\lombok.jar
```

[실습]

Project : Chapter01_XML
Chapter01_ANNO

프로젝트 생성하는 방법

1. Spring Legacy Project

① New → Spring Legacy Project → 프로젝트명 → Simple Java → Next
→ JDK 버전 → Finish → Don't Create 클릭

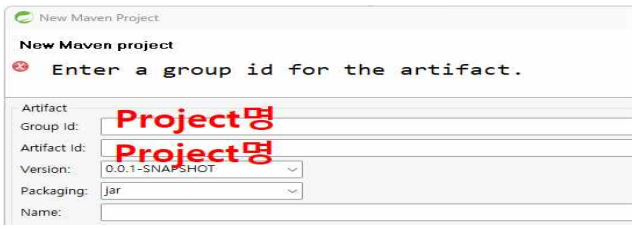
② Maven 변환

프로젝트 → 우클릭 → Configure → Convert to Maven Project → Finish

2. Maven Project

① New → Maven Project 생성 → Next → Finish

Create a simple project (skip archetype selection) 체크
Use default Workspace location 체크



Group Id : 프로젝트명 (아무거나 써도 된다.)
Artifact Id : 프로젝트명 (아무거나 써도 된다.)

② 프로젝트 → 우클릭 → Maven → **Update Project**
③ 프로젝트 → 우클릭 → Properties → Project Facets → JDK 버전 변경
프로젝트 → 우클릭 → Properties → Java Build Path → JDK 버전 변경

※ pom.xml에 추가해야 할 <build></build> 부분은 미리 복사해둔다.

POM.XML?
빌드 툴을 Maven으로 사용한다면 프로젝트 생성 시 pom.xml 파일이 생겨있을 것이다.
POM은 "Project Object Model"의 약자로, 프로젝트의 다양한 정보를 처리하기 위한 객체 모델이다.
pom.xml 파일에는 프로젝트 관리 및 빌드에 필요한 환경 설정, 의존성 관리 등의 정보들을 기술한다.

https://mvnrepository.com/

src/main/resources : **applicationContext.xml**

Project : Chapter01_XML
Package : sample01
Interface : MessageBean.java

Class : HelloSpring.java - public static void main(String[] args)

MessageBeanKo.java - MessageBean 를 Override

MessageBeanEn.java - MessageBean 를 Override

Package : sample02
Interface : Calc.java
추상Method : public void calculate(int x, int y);
Class : HelloSpring - public static void main(String[] args)
CalcAdd.java - Calc.java의 Override
CalcMul.java - Calc.java의 Override

[실행결과]
25 + 36 = xx
25 * 36 = xxx

[문제] 성적계산

이름, 국어, 영어, 수학점수를 입력하여 총점과 평균을 구하여 출력하시오
- Scanner는 <bean> 으로 생성하는 것이 아니라 new 하면 된다.

Project : Chapter01_XML
Package : sample05

Interface : SungJuk.java
추상메소드 : public void calc();
public void display();

Class : SungJukImpl.java - SungJuk를 Override하는 클래스
Field : name, kor, eng, math, tot, avg
Method : 기본 생성자 - Scanner 통해서 데이터 입력
public void calc() - 총점, 평균 계산
public void display() - 출력

Class : HelloSpring.java - public static void main(String[] args)

[실행결과]
이름 입력 :
국어 입력 :
영어 입력 :
수학 입력 :

이름	국어	영어	수학	총점	평균
홍길동	95	78	96	269	89.667

DI (Dependency Injection)

스프링의 핵심 개념

객체사이의 의존 관계를 자기 자신이 아닌 외부에 의해서 설정된다는 개념이다

스프링에서는 설정파일을 사용하여 손쉽게 객체간의 의존관계를 설정하기에 스프링을 DI컨테이너라고 부르기도 한다.

DI 컨테이너는 어떤 클래스가 필요로 하는 인스턴스를 자동으로 생성, 취득하여 연결시켜주는 역활을 한다.

DI 컨테이너가 인스턴스를 생성하도록 하려면 프로그램 소스 내부에서 new 로 직접 생성하지 않고 설정파일에서 필요로 하는 클래스의 정보를 설정해 주어야한다.

스프링은 각 클래스간의 의존관계를 관리하기 위한 방법

- Constructor Injection
- Setter Injection

1. Constructor Injection

: 생성자를 통해서 의존 관계를 연결시키는 것을 말한다.

: 생성자를 통해서 의존 관계를 연결하기 위해서는 XML 설정 파일에서 <bean>요소의 하위요소로 <constructor-arg>를 추가해야 한다.

① 전달인자가 2개 이상인 경우

기본데이터 타입일 경우에는 value 요소를 사용하여 의존관계를 연결시키기 위한 값을 지정

```
public class Foo {
    public Foo(int a, String b) { }
}
```

[applicationContext.xml]

```
<bean id="foo"    class="Foo">
    <constructor-arg>
        <value>25</value>
    </constructor-arg>
    <constructor-arg    value="Hello" />
</bean>
```

② index 속성을 이용하여 지정

[applicationContext.xml]

```
<bean id="foo"    class="Foo">
    <constructor-arg    index="1"    value="Hello" />
    <constructor-arg    index="0">
        <value>25</value>
    </constructor-arg>
</bean>
```

③ type 속성을 이용하여 지정

[applicationContext.xml]

```
<bean id="foo"    class="Foo">
    <constructor-arg    type="int"    value="25" />
    <constructor-arg    type="java.lang.String"    value="Hello" />
</bean>
```

④ 객체를 전달할 경우에는 ref 요소를 사용

```
public class Foo {
    private Bar bar;

    public Foo(Bar bar){
        this.bar = bar;
    }
}

public class Bar { }
```

[applicationContext.xml]

```
<bean id="foo" class="Foo">
    <constructor-arg>
        <ref bean="bar" />
    </constructor-arg>
</bean>

<bean id="bar"    class="Bar" />
```

2. Setter Injection

: setter메소드를 이용하여 의존 관계를 연결시키는 것을 말한다.

: <property>요소의 name 속성을 이용하여 값의 의존 관계를 연결시킬 대상이 되는 필드값을 지정한다

① 전달인자가 2개 이상인 경우

기본데이터 타입일 경우에는 value 요소를 사용하여 의존관계를 연결시키기 위한 값을 지정

```
public class Foo {
    private int a
    private String b;

    public void setA(int a) { }
    public void setB(String b) { }
}
```

[applicationContext.xml]

```
<bean id="foo"    class="Foo">
    <property name="a">
        <value>25</value>
    </property>
    <property name="b" value="Hello" />
</bean>
```

② 객체를 전달할 경우에는 ref 요소를 사용

```
public class Foo {
    private Bar bar;

    public void setBar(Bar bar){
        this.bar = bar;
    }
}

public class Bar { }
```

[applicationContext.xml]

```
<bean    id="foo"    class="Foo">
    <property    name="bar"    ref="bar"></property>
</bean>

<bean    id="bar"    class="Bar" />
```

[실습]

Project : Chapter02_ANNO

Project : Chapter02_XML
Package : sample01
Interface : MessageBean.java
추상메소드 : public void sayHello();
public void sayHello(String fruit, int cost);
public void sayHello(String fruit, int cost, int qty);

Class : MessageBeanImpl.java - MessageBean.java Override 클래스

Class : HelloSpring.java - public static void main(String[] args)

src/main/resources : **applicationContext.xml**

[문제1]

Project : chapter02_XML
Package : sample02
Interface : Calc.java
추상Method : public void calculate();
Class : HelloSpring - public static void main(String[] args)
CalcAdd.java - 필드 x, y 설정
CalcMul.java - 필드 x, y 설정

[실행결과]

25 + 36 = xx
→ xml에서 CalcAdd를 빈(add)으로 Constructor Injection

25 * 36 = xx
→ xml에서 CalcMul를 빈(mul)으로 Setter Injection

[문제2] 다음의 조건에 맞게 이름, 국어, 영어, 수학의 데이터를 입력하여 총점과 평균을 구하시오

*** 조건**

1. [Chapter02_XML] 에서는 XML를 이용한 빈 생성 한다
   [Chapter02_ANNO] 에서는 어노테이션을 이용한 빈 생성 한다
2. SungJukDTO는 Setter Injection을 사용 한다
   SungJukImpl 클래스는 Constructor Injection을 사용 한다
3. SungJukDTO의 필드는 name, kor, eng, math, tot, avg를 사용 한다

Project : Chapter02_XML
Package : sample03
Class : HelloSpring.java - public static void main(String[] args)

Interface : SungJuk.java
추상Method : public void calcTot(); //총점 계산
public void calcAvg(); //평균 계산
public void display(); //출력
public void modify(); //수정

Class : SungJukImpl.java - SungJuk.java를 Override한 클래스
- Constructor Injection

Class : SungJukDTO.java - Setter Injection

[실행결과]

이름	국어	영어	수학	총점	평균
홍길동	97	100	95	xxx	xx.xx

* 수정 - modify()에서 처리
이름 입력 : 코난
국어 입력 : 100
영어 입력 : 100
수학 입력 : 95

이름	국어	영어	수학	총점	평균
코난	100	100	95	xxx	xx.xx

[문제3] 성적 계산

*** 조건**

1. HelloSpring.java에 menu()를 작성 한다
   - 5번을 입력할 때까진 계속 무한루프를 돌린다.
2. [Chapter02_XML] 에서는 XML를 이용한 빈 생성 한다
   [Chapter02_ANNO] 에서는 어노테이션을 이용한 빈 생성 한다
3. Setter Injection을 사용 한다
4. SungJukDTO2의 필드는 name, kor, eng, math, tot, avg를 사용 한다
   - 같은 이름은 입력하지 않는다.
5. [Chapter02_XML] 에서는 List를 applicationContext.xml에서 빈으로 생성 한다
   => List<SungJukDTO2> list = new ArrayList<SungJukDTO2>(); 로 하지 않는다.

   [Chapter02_ANNO] 에서는 List를 @Component / spring.conf 에서 생성 한다

Project : Chapter02_XML
Package : sample04

Class : HelloSpring.java

Interface : SungJuk.java
추상메소드 : public void execute();

SungJuk.java를 Override 한 클래스들
Class : SungJukInput.java
SungJukOutput.java
SungJukUpdate.java
SungJukDelete.java

Class : SungJukDTO2.java
name, kor, eng, math, tot, avg (같은 이름은 입력하지 마시오)

[실행결과]

menu() - HelloSpring.java

```
**************
   1. 입력
   2. 출력
   3. 수정
   4. 삭제
   5. 끝
**************
  번호 :
```

1번인 경우 - SungJukInput.java

```
이름 입력 :
국어점수 입력 :
영어점수 입력 :
수학점수 입력 :

xxx님의 데이터를 입력 하였습니다
```

2번인 경우 - SungJukOutput.java

이름	국어	영어	수학	총점	평균
홍길동	95	100	97	xxx	xx.xx
또치	90	85	75	xxx	xx.xx

3번인 경우 - SungJukUpdate.java

```
수정할 이름 입력 : 코난
찾고자하는 이름이 없습니다.

수정할 이름 입력 : 홍길동
```

이름	국어	영어	수학	총점	평균
홍길동	95	100	97	xxx	xx.xx

```
국어점수 입력 :
영어점수 입력 :
수학점수 입력 :

xxx님의 데이터를 수정 하였습니다
```

4번인 경우 - SungJukDelete.java

```
삭제할 이름 입력 : 코난
찾고자하는 이름이 없습니다.

xxx님의 데이터를 삭제 하였습니다
```

[실습] 내용을 파일로 출력

Project : Chapter02_XML
Package : sample05
Interface : MessageBean.java
Class : MessageBeanImpl.java
HelloSpring.java - public static void main(String[] args)

Interface : Outputter.java
Class : FileOutputter.java - 파일로 출력

스프링 AOP(Aspect Oriented Programming) - 관점지향프로그램

OOP(Object Oriented Programming) - 객체지향프로그램

Aspect란

어플리케이션의 핵심 기능은 아니지만, 어플리케이션을 구성하는 중요한 요소이고, 부가적인 기능을 담당하는 요소

Aspect = Advice + Pointcut

어드바이저는 아주 단순한 형태의 애스펙트라고 볼 수 있다

AOP란

어플리케이션의 핵심적인 기능에서 부가적인 기능을 분리해서 애스펙트라는 모듈로 만들어서 설계하고 개발하는 방법을 AOP(Aspect Oriented Programming)이라고 한다

AOP는 새로운 프로그래밍 패러다임이 아니라, OOP를 돕는 보조적인 기술이다

어플리케이션의 핵심기능을 따라 코딩하지 않고, 핵심기능 대신, 부가적인 기능을 바라보고 집중해서 설계하고 개발할 수 있다
어플리케이션을 부가기능 관점에서, 새롭게 바라볼 수 있게 해준다 라는 의미로 AOP를 관점 지향 프로그래밍이라고도 한다

스프링이 제공하는 AOP는 프록시를 이용한다
프록시를 통해 타겟 오브젝트의 메소드가 호출될 경우, 프록시가 제어를 가로채고, InvocationHandler와 같은 오브젝트를 통해 타겟 메소드의 실행 전 후로 부가적인 기능을 실행한다

AOP 기술의 원조인 AspectJ는 프록시를 사용하지 않는 대표적인 AOP 기술이다
타깃 오브젝트를 뜯어고쳐서 부가기능을 직접 넣어주는 방식을 사용한다
소스코드를 고치는 것이 아니라, 컴파일 된 타깃 클래스의 파일 자체를 수정하거나 클래스가 JVM에 로딩되는 시점을 가로채서 바이트코드를 조작 한다

프록시를 사용하지 않고 클래스 파일 조작과 같은 복잡한 방법을 사용하는 이유는
1. 스프링과 같은 DI 컨테이너의 도움이 필요 없다
스프링과 같은 컨테이너가 사용되지 않는 환경에서 AOP 적용이 가능하다

2. 프록시보다 훨씬 강력하고 유연하다
프록시 방식은 타깃 오브젝트가 생성되고 난 후부터 적용이 가능하다
하지만 AspectJ는 어떤 순간에든지 적용이 가능하다
클래스 바이트코드를 직접 조작하는 것이기 때문에 거의 제한이 없다
대부분의 부가기능은 프록시 방식을 사용해 메소드의 호출 시점에 부여하는 것으로도 충분하다
AspectJ와 같은 고급 AOP 기술은 바이트코드 조작을 위해 JVM의 실행옵션을 변경하고, 별도의 바이트코드 컴파일러를 사용하고, 특별한 클래스 로더를 사용하는 등 번거로운 작업이 필요하다

: 일반적으로 스프링의 AOP를 사용하고, 스프링의 AOP 수준을 넘어서는 기능이 필요하다면
AspectJ를 사용하는 것이다

: 핵심관심사항과 공통관심사항을 기준으로 프로그래밍 함으로써 공통 모듈을 여러 코드에 쉽게 적용

: 공통관심사항을 구현한 코드를 핵심 로직을 구현한 코드에 삽입하는 것

: 핵심로직을 구현할 때 트랜잭션 적용이나 보안 검사와 공통 기능을 삽입할 필요가 없다

AOP 용어

1. Target : 부가기능을 부여할 대상, 핵심기능이 담긴 클래스

2. Advice : 부가기능을 담은 모듈
언제 공통 관심 기능을 핵심 로직에 적용할 지를 정의
Before, After Returning, After Throwing, Around 등이 있다

3. Joinpoint : Advice를 적용 가능한 지점을 의미
메소드 호출, 필드 값 변경 등
스프링의 프록시 AOP에서 조인 포인트는 메소드의 실행 단계뿐이다
타깃 오브젝트가 구현한 인터페이스의 모든 메소드가 조인 포인트가 된다

4. Pointcut : 조인 포인트를 선별하는 기능을 정의한 모듈
가능한 조인 포인트들 중에 실제로 부가기능을 적용할 것들을 선별한다
클래스를 선정하고, 그 안의 메소드를 선정하는 과정을 거친다
실제로 Advice가 적용되는 Joinpoint를 나타낸다

5. Proxy : 클라이언트와 타깃 사이에 존재하면서 부가기능을 제공하는 오브젝트
클라이언트는 타깃을 요청하지만, 클라이언트에게는 DI를 통해 타깃 대신 프록시가 주입된다
클라이언트의 메소드 호출을 대신 받아서 타깃에게 위임하며, 그 과정에서 부가기능을 부여한다
스프링 AOP는 프록시를 이용한다

6. Advisor : 어드바이스와 포인트컷을 하나로 묶어 취급한 것
AOP의 가장 기본이 되는 모듈이다
스프링은 자동 프록시 생성기가 어드바이저 단위로 검색해서 AOP를 적용한다

7. Aspect : 다수의 포인트컷과 어드바이스의 조합으로 만들어진다
보통 싱글톤 형태의 오브젝트로 존재한다
어드바이저는 아주 단순한 애스펙트라고 볼 수 있다

8. Weaving : Advice를 핵심로직코드에 적용하는 것을 Weaving라고 한다.

Advice를 Weaving하는 3가지 방식

- 컴파일 시에 Weaving
  AspectJ에서 사용하는 방식
  AOP가 적용된 클래스 파일이 생성된다

- 클래스 로딩 시에 Weaving
  AspectJ 5/6 버전이 컴파일 방식과 클래스 로딩방식을 함께 지원

- 런타임 시에 Weaving
  소스코드나 클래스 정보 자체를 변경하지 않는다
  프록시를 이용한다(핵심로직을 구현한 객체에 직접 접근하는 것이 아니라 중간에 프록시를 생성하여 프록시를 통해서 핵심로직을 구현한 객체에 접근한다.)
  프록시는 메소드가 호출될 때에만 적용할 수 있다(필드값 변경에 대해서는 적용 불가능)

: 스프링은 자체적으로 프록시 기반의 AOP를 지원한다
  메소드 호출 Joinpoint만 지원(필드값 변경과 같은 Joinpoint는 불가능)

: 스프링 AOP는 자바 기반이다.
  AspectJ는 Aspect를 위한 별도의 문법이 필요하다

: 대상 객체의 메소드를 실행하기 전/후에 원하는 기능을 삽입(Around Advice)하는데 캐시기능, 성능 모니터링 기능과 같은 Aspect를 구현

[실습]

Project : Chapter03_ANNO
Chapter03_XML

Project : Chapter03_XML

Package : sample01
Interface : MessageBean.java
Class : MessageBeanImpl.java
HelloSpring.java - main()
LoggingAdvice.java - 공통관심사항
src/main/resources : acQuickStart.xml

스프링과 JDBC

스프링은 JDBC를 비롯하여 ORM프레임워크(myBatis, hibernate, JPA(Java Persistence API))를 지원한다
스프링의 목표는 인터페이스에 의한 개발인데 DAO는 데이터베이스에서 데이터를 읽거나 쓰는 수단을 제공하기 위해 존재하며, 반드시 인터페이스를 통해 외부에 제공돼야 한다.
서비스 객체는 인터페이스를 통해서 DAO에 접근한다
서비스 객체를 특정 데이터 액세스 구현체에 결합시키지 않음으로써 테스트를 용이하게 한다
DAO인터페이스는 DAO구현과 서비스 객체 사이에서 느슨한 결합이 유지될 수 있게 한다

서비스 객체 → DAO 인터페이스
UserInsertService UserDAO
↑
DAO 구현 객체
UserDAOImpl

스프링은 데이터베이스 연동을 위한 템플릿 클래스를 제공함으로써
Connection, Statement(PreparedStatement), ResultSet등을 생성하고 처리한 다음 close(반환)하는 JDBC의 중복된 코드를 줄일 수 있다

JDBC는 무조건 SQLException의 예외만 발생하므로 정확히 Connection에서 발생했는지 아니면 Statement에서 발생했는지 따져봐야한다. JdbcTemplate클래스는 SQLException이 발생하면 스프링이 제공하는 예외 클래스중 알맞는 것으로 변환해서 발생한다.
스프링은 JDBC보다 다양한 예외 계층을 제공하고 어떤 퍼시스턴스 솔루션과도 연관성을 갖지 않는다
스프링을 사용하면 퍼시스턴스 기술과 관계없이 일관성있게 예외를 발생시킬 수 있다

스프링의 DataAccessException는 비검사형 예외(try~catch블럭을 사용하지 않아도 컴파일이 되는 예외)로서 반드시 잡아서 처리할 필요가 없다

스프링이 제공하는 DataSource를 설정하는 3가지 방법

1. 커넥션풀을 이용한 DataSource 설정
스프링이 직접적으로 커넥션풀을 제공하진 않지만 DBCP(Jakarta Commons Database Connection Pool) API와 같은 커넥션 풀 라이브러리를 이용
DBCP에는 풀링 기능을 제공하는 다양한 데이터 소스가 포함되 있지만 BasicDataSource가 가장 많이 사용된다

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
    <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
    <property name="username" value="" />
    <property name="password" value="" />
    <property name="initialSize" value="5" />
  → 해당 풀이 시작될 때 생성할 커넥션 수, 0은 무제한
    <property name="maxActive" value="10" />
  → 해당 풀에서 동시에 제공할 수 있는 최대 커넥션 수, 0은 무제한
</bean>
```

2. JNDI을 이용한 DataSource 설정
WebLogic이나 JBoss와 같은 JEE 어플리케이션 서버를 사용할 경우
톰캣이나 Resin등의 웹 컨테이너를 사용할 경우

<jee:jndi-lookup id="dataSource" jndi-name="oracle" resource-ref="true" />
<jee:jndi-lookup>태그를 이용 - 스프링2.0부터 지원
jndi-name 프로퍼티는 JDNI에 있는 리소스의 이름을 지정한다
jndi-name프로퍼티만 지정된 경우에는 데이터소스는 지정된 jndi-name을 그대로 이용해서 검색한다
자바 애플리케이션 서버에서 가동되는 경우에는 resource-ref=true이면 jndi-name값의 앞에 "java:comp/env"가 붙은 이름을 사용한다.
<jee:jndi-lookup>태그를 사용하지 않고 JndiObjectFactoryBean클래스를 이용할 수 있다

3. DriverManager을 이용한 DataSource 설정
: 스프링에 설정할 수 있는 가장 단순한 데이터 소스는 JDBC드라이버를 통해 정의된 것이다

- DriverManagerDataSource
애플리케이션이 커넥션을 요청할 때마다 새로운 커넥션을 반환한다
DBCP의 BasicDataSource와는 달리 커넥션은 풀링되지 않는다
멀티스레드에서도 동작은 하지만 커넥션이 필요할 때마다 새로 커넥션을 생성하므로 심각한 성능 저하를 유발한다.

- SingleConnectionDataSource
커넥션을 요청하면 항상 동일한 커넥션을 반환한다
커넥션 풀링 기능은 없지만 오직 하나의 커넥션만을 풀링하는 데이터소스
사용할 수 있는 커넥션이 오직 하나뿐이라서 멀티스레드 애플리케이션에서는 제대로 동작하지 않을 것이다
: 스프링의 JDBC 프레임워크는 자원관리와 예외 처리를 대신 해주므로 JDBC코드가 훨씬 간결해진다
스프링은 단순 반복적인 데이터 액세스 코드를 템플릿 클래스 뒤로 추상화해 숨긴다

1. JdbcTemplate
스프링의 가장 기본적인 JDBC 템플릿
색인된 파라미터(indexed parameter)기반의 쿼리를 통해 데이터베이스를 쉽게 액세스하는 기능을 제공

2. NamedParameterJdbcTemplate
SQL과 값들을 색인된 파라미터 대신 명명된 파라미터(named parameter)로 바인딩하여 쿼리를 수행할 수 있게 해준다

*** 스프링이 제공하는 기반클래스**
JdbcTemplate - JdbcDaoSupport
- JdbcDaoSupport 상속 받으면 JdbcTemplate 빈으로 생성할 필요가 없다
- **JdbcDaoSupport 에서는 getJdbcTemplate() 메소드를 제공하므로 JdbcTemplate을 편리하게 사용할 수 있다.**

NamedParameterJdbcTemplate - NamedParameterJdbcDaoSupport

[실습]

Project : Chapter04_ANNO
Chapter04_XML

Project : Chapter04_XML
Package : user.main
Class : HelloSpring.java - main 메소드

Package : user.service
Interface : UserService.java
public void execute(); - 추상메소드
Class : UserInsertService.java
UserSelectService.java
UserUpdateService.java
UserDeleteService.java

Package : user.dao
Interface : UserDAO.java
Class : UserDAOImpl.java

Package : user.bean
Class : UserDTO.java

src/main/resources
Folder : spring
File : applicationContext.xml
db.properties

스프링과 myBatis

ORM(Object Relational Mapping) 프레임워크는 데이터베이스와 객체와의 관계를 맵핑시켜 퍼시스턴스 로직 처리를 도와주는 프레임워크이다.
대표적으로 MyBatis와 hibernate, JPA가 있다

myBatis는 SQL쿼리문, 예외처리, 트랙잭션 관리들을 XML형식으로 관리한다.
POJO(Plain Old Java Object)객체와 테이블의 컬럼들을 편리하고 빠르고 정확하게 매칭할 수 있다.

myBatis의 목표와 특징은 쉽고, 간단하고, 의존성이 적다는 것이다
SQL문과 자바코드를 분리함으로 인해 자바 개발자는 쿼리문을 신경 쓰지 않아도 된다.

myBatis는 자바오브젝트와 SQL문 사이의 자동 매핑 기능을 지원하는 ORM 프레임워크이다
자바코드와 SQL를 분리하므로서 SQL문의 변경이 있을 때 마다 자바를 수정하지 않아도 되고 컴파일을 하지 않아도 된다

[실습]
Project : Chapter05_ANNO
Chapter05_XML

Packagge : user.dao
Interface : UserDAO.java
Class : **UserDAOMybatis.java**

src/main/resources
Folder : spring
File : applicationContext.xml
mybatis-config.xml
db.properties
Folder : mapper
File : **userMapper.xml**

스프링 MVC

스프링 MVC도 컨트롤러를 사용하여 클라이언트의 요청을 처리한다.
스프링에서 DispatcherServlet 이 MVC에서 C(Control) 부분을 처리한다.
개발자가 처리할 부분은 클라이언트의 요청을 처리할 컨트롤러와 응답화면을 전송할 JSP나 Velocity 템플릿 등 뷰 코드이다
DispatcherServlet, HandlerMapping, ViewResolver등은 스프링이 기본적으로 제공하는 구현 클래스를 사용한다.

스프링 MVC의 구성 요소

1. DispatcherServlet
클라이언트의 요청을 전달 받는다
컨트롤러에게 클라이언트의 요청을 전달하고 Controller가 리턴한 결과값을 View에 전달하여 응답을 생성하도록 한다.

2. HandlerMapping
클라이언트의 요청 URL을 어떤 Controller가 처리할지를 결정한다.

3. Controller
클라이언트의 요청을 처리한 뒤 결과를 DispatcherServlet에 알려준다

4. ModelAndView
컨트롤러가 처리한 결과 정보 및 뷰 선택에 필요한 정보를 담는다.

5. ViewResolver
컨트롤러의 처리 결과를 생성할 뷰를 결정한다.

6. View
컨트롤러의 처리 결과 화면을 생성한다.
JSP나 Velocity 템플릿 파일등을 뷰로 사용한다.

[실습]

Project : Chapter06 (Dynamic Web Project) - pom.xml

project명에서 우클릭 - Spring - Add Spring Project Nature
project명에서 우클릭 - Configure - Convert to Maven Project
project명에서 우클릭 - Maven - Update Project...

Project : SpringProject (Spring MVC Project)

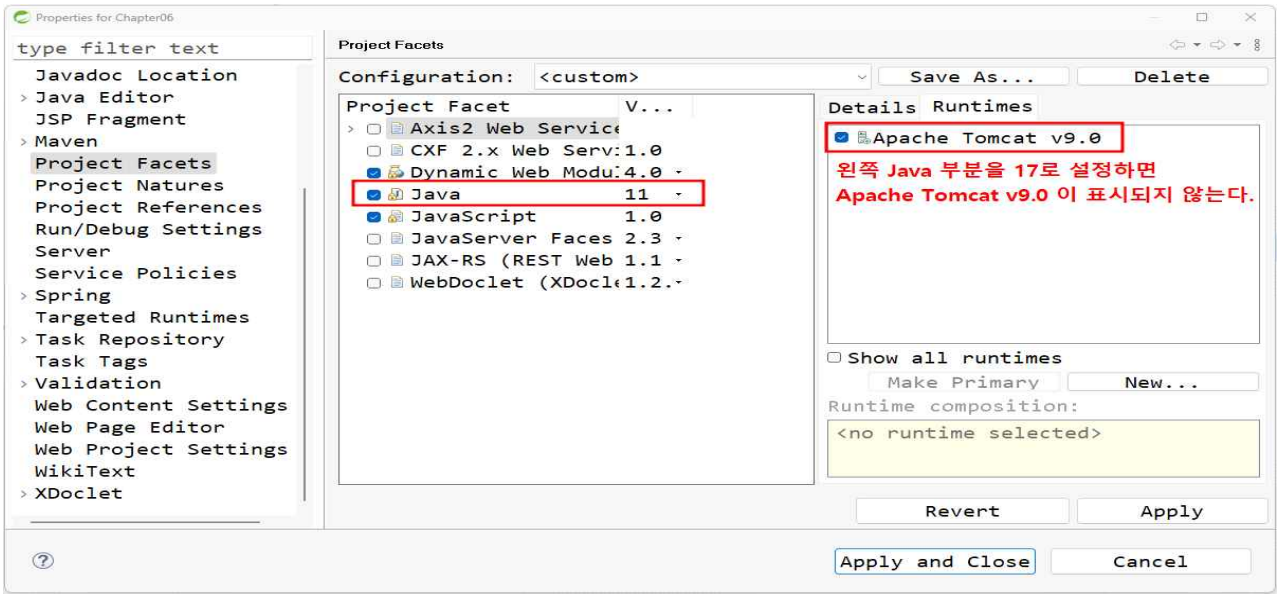
- Package를 3단계로 작성해야한다.
- Package를 3단계로 설정할 때 패키지의 마지막을 Project명(SpringProject)과 똑같이 잡아야한다.
  그래야 URL의 경로를 /SpringProject로 인식한다.
- 만약 Project명과 Package명을 다르게 할 경우에는 실행 시 URL를 패키지명으로 하면 된다.

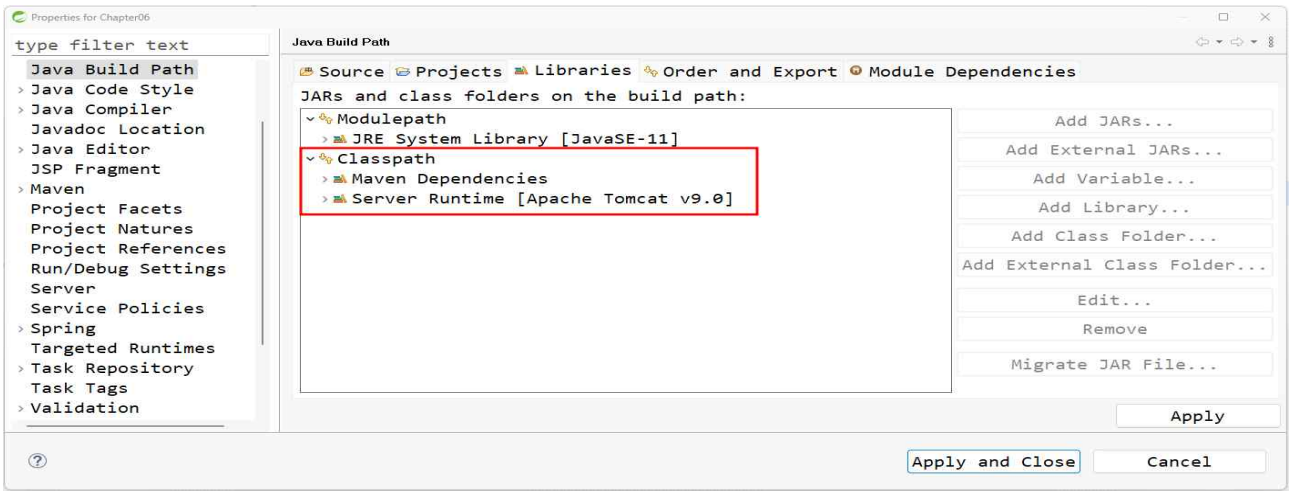
※ The superclass "jakarta.servlet.http.HttpServlet" was not found on the Java Build Path 에러

=> 스프링 버전 6.x 지원이 안 된다.

=> 스프링 버전 5.x 다운시킨다.

=> sts.ini (JDK 11로 설정), Tomcat 9.0, Spring 5.3.23





※ STS3에서 Spring Legacy Project 생성 시 Spring MVC Project 메뉴가 안 나온다.

STS3를 2024년 02월 이후에 설치하면 3.0xml이 다운로드가 안되어 Spring MVC Project 메뉴를 찾을 수 없다.

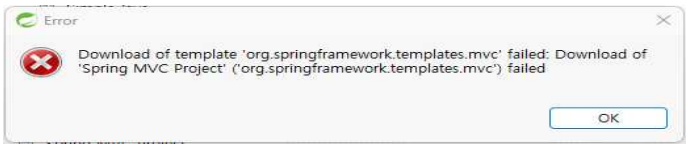
해결 방법

1. 이클립스 workspace
\.metadata\.plugins\org.springsource.ide.eclipse.commons.content.core

2. **https-cotent.xml**을 붙여 넣는다.

[참조] https://blog.naver.com/outrogallery0322/223367935948

※ Spring Legacy Project > Spring MVC Project > Next >
누르면 템플릿이 다운로드가 되지 않을 경우 아래의 에러가 뜬다.

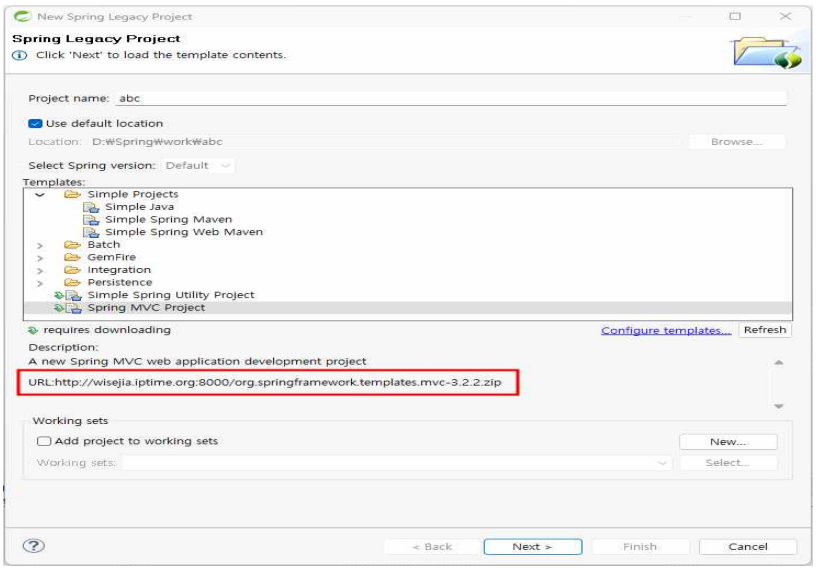


=> 기존에 http://wisejia.iptime.org:8000/org.springframework.templates.mvc-3.2.2.zip 있는데 이곳에서는 템플릿이 다운로드가 안 되기 때문이다.

=> 만약에 Invalid thread access 라고 에러가 뜨면 .metadata를 지우고 다시 처음부터 설정한다.

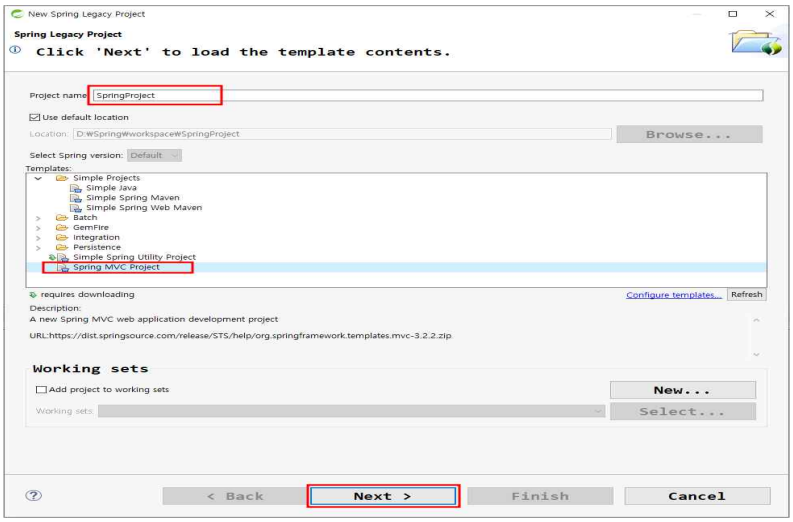
① 위의 https-content.xml 파일을 연다.
② 파일 안에서 기존의 템플릿 주소(기존에 있는 URL) 찾는다.
③ 기존의 템플릿 주소를 아래의 주소로 대체한다.
http://timespace12.dothome.co.kr/org.springframework.templates.mvc-3.2.2.zip

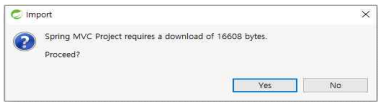
http://wisejia.iptime.org:8000/org.springframework.templates.mvc-3.2.2.zip (기존에 있는 URL)



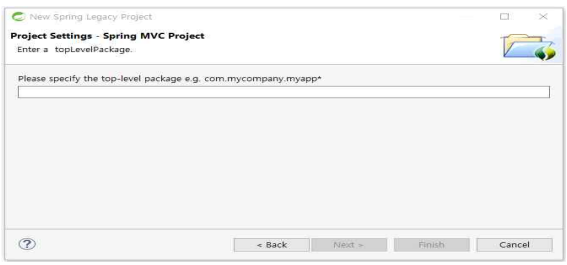
Project ： SpringProject

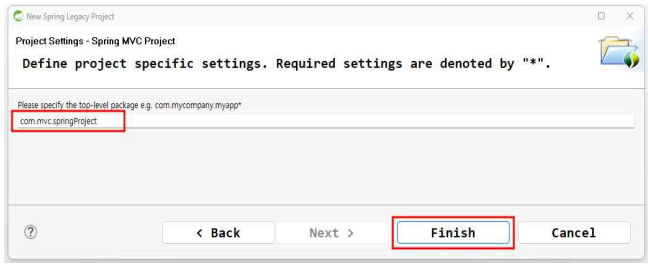
Package ： com.mvc.SpringProject





3단계 Package로 설정 - **com.mvc.SpringProject**





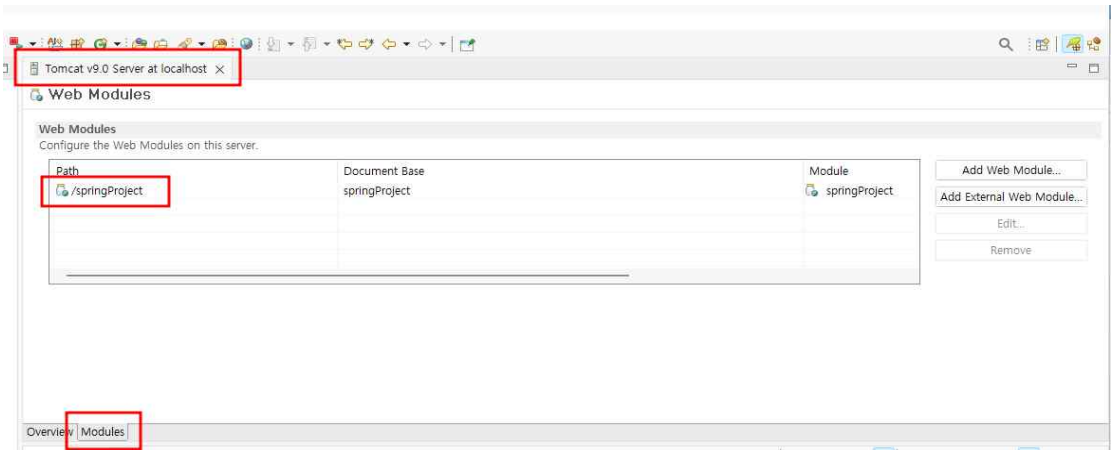
pom.xml

- JDK와 Spring 버전을 변경 한다
- default로 JDK는 1.6 으로 잡혀있다.
  => <java-version>17</java-version>로 변경
- default로 스프링은 3.1.1 으로 잡혀있다.
  => <org.springframework-version>5.3.23</org.springframework-version> 변경

Build Path → Configure Build Path...
→ Libraries(탭) → JRE System Library → **JDK 17로 변경**

Build Path → Configure Build Path...
→ Project Facets(왼쪽) → **Java 1.7로 변경**
→ Runtimes(오른쪽 탭) → **Apache Tomcat v9.0 체크**
→ Apply

빨강 박스 부분을 /spring 또는 /mini 로 바꾸면
URL은 http://localhost:8080/spring/ 또는 http://localhost:8080/mini/ 로 하면 된다.



[실행]

http://localhost:8080/Chapter06/hello.do

Project : Chapter06
Package : com.controller
Class : HelloController.java

Folder : WEB-INF
web.xml
dispatcher-servlet.xml

Folder : view
hello.jsp

* **web.xml**

1. web.xml에 DispathcerServlet 설정을 통해 스프링 설정 파일을 지정합니다.
2. *.do로 들어오는 클라이언트의 요청을 DispatcherServlet이 처리하도록 했다
3. DispatcherServlet은 **/WEB-INF/서블릿이름-servlet.xml** 파일로부터 스프링 설정 정보를 읽어온다.

* **HelloController.java**

1. 컨트롤러를 구현하려면 @Controller 어노테이션을 클래스에 적용한다.
2. @RequestMapping 어노테이션을 이용해서 클라이언트의 요청을 처리할 메소드를 지정한다.
3. ModelAndView.setViewName()메소드를 이용해서 컨트롤러의 처리 결과를 보여줄 뷰 이름을 지정한다.

* **dispatcher-servlet.xml**

1. DispatcherServlet은 스프링 컨테이너에서 컨트롤러 객체를 검색하기 때문에 스프링 설정파일에 컨트롤러를 빈으로 등록해주어야 한다
2. DispatcherServlet은 뷰이름과 매칭되는 뷰 구현체를 찾기 위해 ViewResolver를 사용한다.
3. 스프링 MVC는 JSP, Velocity, FreeMarker등의 뷰 구현 기술과의 연동을 지원하는데, JSP를 뷰 기술로 사용할 경우 InternalResourceViewResolver 구현체를 빈으로 등록해주면 된다.
4. ViewResolver가 /view/뷰이름.jsp를 뷰JSP로 사용한다는 의미이다

[문제]

Project : Chapter06_1
Package : com.controller
Class : SumController.java

Package : com.bean
Class : SumDTO.java

Foler : /WEB-INF/sum
File : input.jsp
result.jsp

스프링 설정 파일 (WAC) → 변경하시오
/WEB-INF/**dispatcher**-servlet.xml → /WEB-INF/spring/appServlet/servlet-context.xml

[실행결과]

http://localhost:8080/Chapter06_1/**input.do**

input.jsp —get→ result.jsp

X : □
Y : □
계산
submit

25 + 36 = xx

http://localhost:8080/Chapter06_1/**result.do**

[문제]

Project : Chapter06_1
Package : com.controller
Class : SungJukController.java

Package : com.bean
Class : SungJukDTO.java
Field : name, kor, eng, math, tot, avg

Foler : /WEB-INF/sungJuk
File : input.jsp
result.jsp

스프링 설정 파일
/WEB-INF/dispatcher-servlet.xml → /WEB-INF/spring/appServlet/mvc-context.xml

[실행결과]

namespace
http://localhost:8080/Chapter06_1/sungJuk/input.do

input.jsp post result.jsp
이름 : ㅁ *** xxx 성적 ***
국어 : ㅁ 총점 : xxx
영어 : ㅁ 평균 : xx.xxx
수학 : ㅁ
계산

namespace
http://localhost:8080/Chapter06_1/sungJuk/result.do

WAC (Web Application Context) 등록

1. ContextLoaderListener가 생성하는 Root WAC

- 웹 환경과 독립적인 빈 등록
- 디폴트 설정 파일 /WEB-INF/applicationContext.xml 으로 설정 된다
- 서비스계층과 데이터 액세스 계층을 포함해서 웹 환경과 직접 관련이 없는 모든 빈들을 여기에 등록 한다

→ 만약에 사용할 이름이 다르거나 설정파일이 여러 개인 경우
contextConfigLocation 파라미터를 추가해서 설정해주면 된다

2. DispatcherServlet이 생성하는 WAC

- DispatcherServlet이 직접 사용하는 컨트롤러를 포함한 웹 관련 빈을 등록
- 디폴트 설정 파일 /WEB-INF/서블릿이름-servlet.xml으로 설정된다

→ 만약에 사용할 이름이 다르거나 설정파일이 여러 개인 경우
contextConfigLocation 파라미터를 추가해서 설정해주면 된다

<mvc:annotation-driven />

1. **Spring Web MVC를 하기 위해 설정해야 하는 값들을 자동으로 추가해준다.**
2. Spring MVC가 @Controller에 요청을 보내기 위해 필요한 HandlerMapping과 HandlerAdapter를 bean으로 등록한다.
- HandlerMapping : HTTP 요청정보를 이용해서 컨트롤러를 찾아주는 기능
- HandlerAdapter : HandlerMapping을 통해 찾은 컨트롤러를 직접 실행하는 기능을 수행
3. bean을 생성하기 위해 xml 파일에 context:component-scan을 명시하면 이 태그를 포함하지 않아도 MVC 애플리케이션은 작동한다.

<context:component-scan />

특정 패키지 내의 클래스를 스캔하고 Annotation(@Component @Controller @Service @Repository)을 확인한 후 Bean 인스턴스로 생성한다.
이를 이용하면 @Autowired와 @Qualifier Annotation을 인식할 수 있다.
<context:component-scan /> 을 선언했다면 <context:annotation-config /> 를 선언할 필요가 없다.

<context:annotation-config />

ApplicationContext 안에 이미 등록된 Bean들의 Annotation을 활성화하기 위해 사용된다.
Component-scan과의 차이점은 이 설정은 Bean을 등록하는 작업을 수행하지 않는다는 것이다.

스프링은 MessageConverter를 가지고 있다. 기본값은 JSON이다.
요청 시 데이터를 JSON으로 바꾸는데 JSON으로 바꿔주는 역할을 MessageConverter가 해준다.
MessageConverter는 기본적으로 **Jackson** (JSON 데이터로 변경해주는 라이브러리)를 이용한다.

MessageConverter

우리가 HTTP 요청을 모델에 바인딩하고 클라이언트에 보낼 HTTP 응답을 만들기 위해 뷰를 사용했던 방식과는 달리, HTTP 요청 본문과 HTTP 응답 본문을 통째로 메세지로 다루는 방식이다.

HTTP 프로토콜에서 메시지는 문자열을 통해 전송된다.

MessageConvert는 HTTP 통신에서 일반 문자열이 아닌 XML이나 JSON으로 통신하기 위해 사용된다.
주로 XML이나 JSON을 이용한 AJAX 기능이나 웹 서비스를 개발할 때 사용된다.

스프링의 @RequestBody와 @ResponseBody를 통해 구현할 수 있다.

애노테이션을 명시해두게 되면 스프링은 메세지 컨버터라는 것을 사용하여 HTTP 요청이나 응답을 메세지로 변환하게 된다.

즉 파라미터 부분에 @RequestBody를 입력할 경우, 파라미터 타입에 맞는 메세지 컨버터를 선택한 뒤 HTTP 요청 본문을 통째로 메세지로 변환하여 파라미터에 바인딩하는 것이다.
메서드의 상단에 @ResponseBody를 입력할 경우 또한 마찬가지로 리턴타입에 맞는 MessageConverter를 선택한 뒤 리턴 값을 통째로 메세지로 변환한 뒤 리턴해주는 것이다.

참고로 GET 방식의 요청일 경우 HTTP 요청 본문이 없으므로 @RequestBody를 사용할 수 없다.
@RequestParam이나 @ModelAttribute를 사용해야 한다.

DispatchServlet은 handlerMethod()를 호출하여 응답으로 String, ModelAndView, Object 중 하나를 반환한다.

String이 반환되면 뷰 이름을 찾아서 JSP를 렌더링 하고, 뷰가 없으면 404를 반환한다
그런데 메서드 위에 @ResponseBody 어노테이션이 있다면 뷰를 찾지 않고, String을 그대로 반환한다.
DispatcherServlet이 ModelAndView를 반환하면 ViewResolver가 실행된다.
그런데 @ResponseBody가 있으면 Object는 MessageConverter가 실행되어 반환된다.

InternalResourceViewResolver

- Resource를 대상으로 View를 찾는데, 정적 자원 즉, webapp 아래 자원들을 반환값으로 찾게 된다.
  컨트롤러가 지정한 View 이름으로부터 실제로 사용될 View를 선택한다.
- 컨트롤러가 지정한 뷰 이름 앞뒤로 prefix 프로퍼티와 suffix 프로퍼티를 붙인 값이 실제로 사용될 자원의 경로가 된다.

파일 업로드

1. 등록 폼에 파일 업로드 필드 추가

- 컨트롤러를 MultipartFile 객체를 이용해서 업로드 가능한 컨트롤로 구현해야 한다.
- 업로드된 파일은 대부분 바이너리 파일이다
- 파일이 첨부된 폼을 전송하는 경우 콘텐츠형식은 multipart/form-data를 해야 한다
- 폼이 전송되면 이미지 파일의 바이너리 데이터를 포함하는 한 부분이 멀티파트 형식으로 전달된다.

2. Controller의 업로드된 파일을 받도록 수정

3. 스프링에 멀티파트 파일 리졸버(multipart file resolver) 설정

- 실제로 파일 업로드 기능이 동작하기 위해서는 반드시 사용자가 업로드한 파일 정보가 MultipartFile 객체에 설정되어야 하며, 이를 위해서 멀티파트 리졸버 객체가 반드시 필요하다.

4. 다중 파일 업로드 시에는 <input type="file" name="">이 2개 이상 있을 때는 name속성에 같은 이름을 지정해야한다

*** 파일 업로드를 위한 스프링 설정**

DispatcherServlet은 컨트롤러에 해당 데이터를 제공하기 위하여 POST 요청에서 멀티파트 데이터를 추출하는 멀티파트 리졸버가 필요하다

스프링이 CommonsMultipartResolver만 제공한다.

파일 업로드가 정상적으로 동작하기 위해서는 내가 만든 컨트롤러뿐만 아니라 이를 위해 멀티파트 리졸버 객체를 메모리에 올리는 두 개의 객체 생성 과정이 필요한 것이다.

```
<bean id="multipartResolver"
        class="org.springframework.web.multipart.commons.CommonsMultipartResolver"
        p:maxUploadSize="5000000" />
```

*** pom.xml**

```
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.13.0</version>
</dependency>

<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.5</version>
</dependency>
```