1. 스프링 프레임워크 개요

- 특징
  - 엔터프라이즈 애플리케이션 구축을 위한 솔루션이다.
  - 모듈화 되어 있어 필요한 부분만 사용 가능 하다.
  - 선언적 트랜젝션 관리가 가능하다.
  - 완전한 기능을 갖춘 MVC 프레임워크를 제공
  - AOP기능을 사용할 수 있다.
- 장점
  - 생산성
  - 품질보증
  - 유지보수

2. 스프링 프레임워크의 특징

이름	특징
spring-beans	스프링 컨테이너를 이용해서 객체를 생성하는 기본 기능을 제공한다.
spring-context	객체 생성, 라이프 사이클 관리, 스키마 확장
spring-aop	프락시 기반 AOP기능을 제공
spring-web	REST 클라이언트, 데이터 변환 처리, 서블릿 필터, 파일 업로드 지원 등 개발에 필요한 기능을 제공
spring-webmvc	스프링 기반의 웹 MVC 프레임워크. 웹 애플리케이션을 개발하는데 필요한 컨트롤러 구현을 제공
spring-websocket	스프링 MVC에서 웹소켓을 사용하기 위한 기능을 지원
spring-oxm	XML과 자바 객체간의 매핑을 처리하기 위한 API를 제공
spring-tx	트랜젝션 처리를 위한 추상 레이어를 제공
spring-jdbc	JDBC 프로그래밍을 보다 쉽게 할 수 있는 템플릿을 제공. 이를 이용하면 JDBC프로그램에서 반복적으로 입력해야 하는 코드를 줄일 수 있다.
spring-orm	하이버네이트, JPA, MyBatis 등과의 연동을 지원한다.
spring-jms	JMS서버와 메시지를 쉽게 주고받을 수 있도록 템플릿과 어노테이션 등을 제공
spring-context-support	스케줄링, 메일발송, 캐시연동, 벨로시티 등 부가 기능을 제공

POJO(Plain Old Java Object) 기반의 프레임워크

> 자바 객체의 라이프사이클을 스프링 컨테이너가 직접 관리하며, 스프링 컨테이너로부터 필요한 객체를 얻어올 수 있다.

DI(Dependency Injection)을 지원

> 각 계층이나 서비스들 사이 또는 객체들 사이에 의존성이 존재할 경우 스프링 프레임워크가 서로를 연결시켜준다. 이는 클래스들 사이에 약한 결합을 가능케 한다.

AOP(Aspect Oriented Programming)를 지원

> 트랜젝션, 로깅, 보안 등 여러 모듈에서공통적으로 지원하는 기능을 분리하여 사용할 수 있다.

확장성이 높다.

> 스프링 프레임워크의 소스는 모두 라이브러리로 분리시켜 놓음으로써 필요한 라이브러리만 가져다 쓸수 있다. 그리고 많은 외부 라이브러리들도 이미 스프링 프레임워크와 연동되고 있다.

Model2 방식의 MVC Framework를 지원

3. 스프링 프로젝트 생성

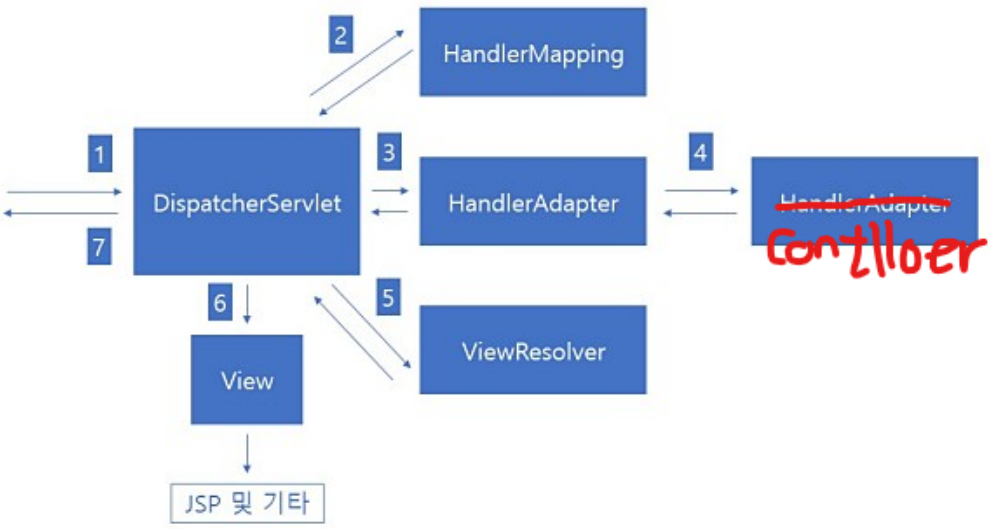
File -> new -> Spring Legacy Project

프로젝트명 우클릭 -> Properties -> Project Factes -> Java옆 화살표 눌러서 1.8로 변경

pom.xml 들어가서

```
pom.xml

<properties>
    <java-version>1.6</java-version>
    <org.springframework-version>4.3.9.RELEASE</org.springframework-version> # 버전 수정
    <org.aspectj-version>1.6.10</org.aspectj-version>
    <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```



웹 기본주소 (http://localhost:8080/test/) 로 들어오면 appServlet를 찾아서 실행시켜준다.

```
/src/main/webapp/WEB-INF/web.xml

<servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

아래를 실행하게 됨

```
/src/main/webapp/WEB-INF/web.xml
    <servlet>
        <servlet-name>appServlet</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-
value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
```

> /WEB-INF/spring/appServlet/servlet-context.xml 의 contextConfigLocation를 던져줌

Handler Mapping은 Request의 처리를 담당하는 컨트롤러를 찾기 위해서 존재한다. HandlerMapping 인터페이스를 구현한 여러 객체들 중 RequestMappingHandlerMappling 같은 경우는 개발자가 @RequestMapping 어노테이션이 적용된 것을 기준으로 판단하게 된다. 적절한 컨트롤러가 찾아졌다면 HandlerAdapter를 이용해서 해당 컨트롤러를 동작시킨다.

Controller는 개발자가 작성하는 클래스로 실제 Request를 처리하는 로직을 작성하게 된다. 이때 View에 전달해야 하는 데이터는 주로 Model이라는 객체에 담아서 전달한다. Controller는 다양한 타입의 결과를 반환하는데 이에 대한 처리는 ViewResolver를 이용하게 된다.