스프링 개요

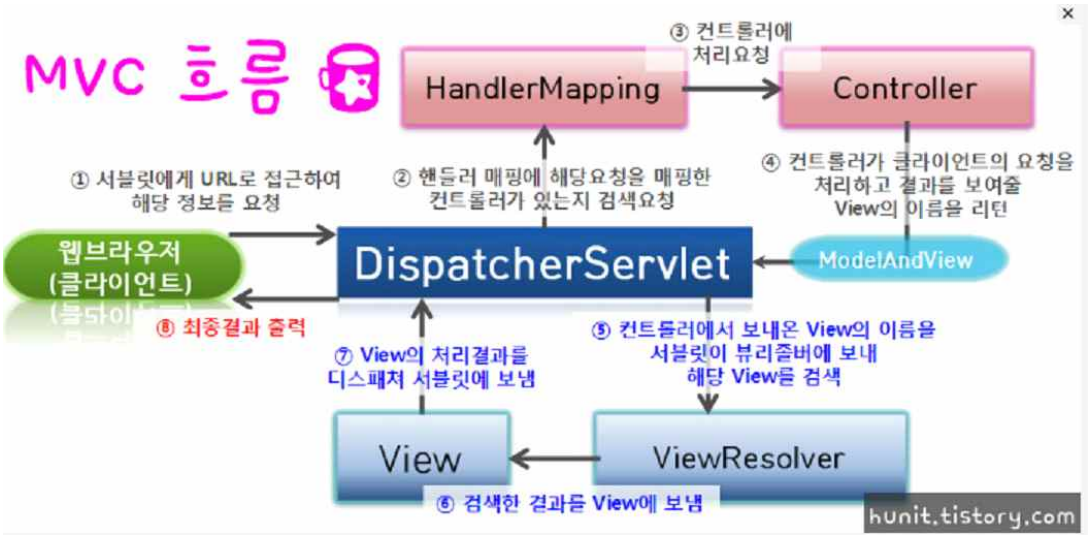
1. 웹사이트 개발을 **쉽고, 편리하게 개발할 수 있도록 지원해주는 오픈소스(공개된 의미) 프레임워크(외부 라이브러리에 의존)**로 경량급 애플리케이션 프레임워크

2. 기존 서블릿 MVC 자바언어 개발에 비해서 **개발 기간이 단축되고,코드 작성이 줄어들면서 유지 보수가 쉽다.**

3.스프링은 **로드 존슨이 2003년 오픈소스 프레임워크 프로젝트로 개발하기 시작함. 2004년 3월에 스프링 1.0이 발표되었다.**

4. 스프링은 철저히 MVC 패턴을 따르기 때문에 **개발자 코드영역과 UI 개발자 영역이 철저히 분리됨.** 그러므로 유지 보수가 뛰어나고, **개발자와 UI를 담당하는 분들과 출동 염려가 현격히 사라짐**.

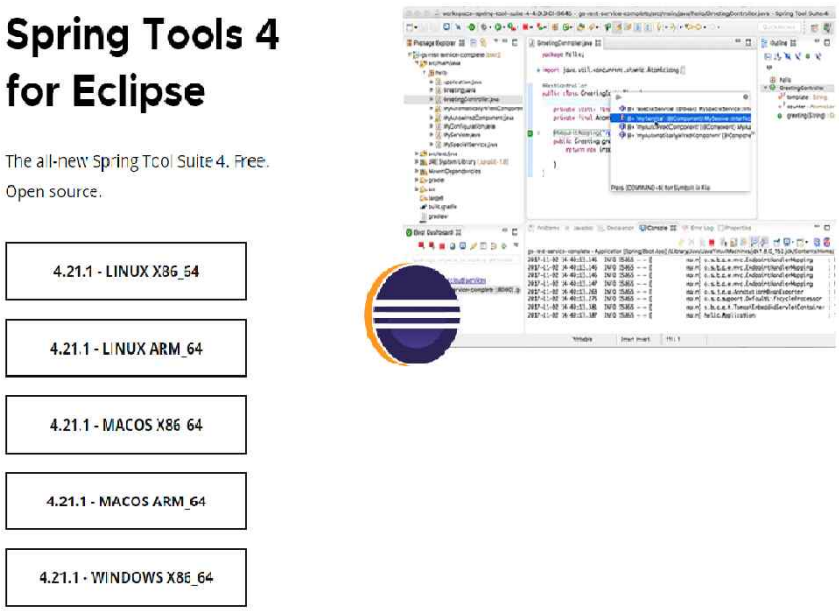
스프링 MVC 구조(여기서는 MVC에서 컨트롤과 뷰만 표시됨)



1. 스프링 개발툴 STS(SPring Tool Suite)

가. 일반 이클립스에는 스프링 프로젝트를 만들 수 있는 메뉴가 없다. 그래서 일반 이클립스에 스프링 프로젝트 만들 수 있는 STS 플러그인이 설치된 개발툴이 바로 STS이다. 이 STS 4 최신 버전을 다운받기 위해서 구글에서 STS 다운로드로 검색하면 다운받을 수 있는 주소 https://spring.io/tools 가 검색된다.

나. STS 다운로드 주소 : https://spring.io/tools에서 각 운영체제에 맞는 STS 4 최신 버전을 다운로드 받는다.



다. 되도록 영문 숫자 조합으로 된 폴더에 압축을 푼다. 압축이 푼 폴더에 SpringToolSuite4.exe 실행파일이 있다. 이 파일을 실행해서 프로젝트 폴더가 저장될 워크스페이스 폴더를 생성한다.

라. Spring Starter Project를 생성하면 C:\Users\mun\.m2가 생성되고 그 하위 폴더에 repository 나 .lemminx-maven 폴더가 생성되면서 스프링 관련 라이브러리가 다운로드 된다. 이 때 인터넷 연결이 불안정 하든지 네트워크가 불안하면 제대로 다운로드가 안되어 지고 해당 프로젝트에 에러 표시가 나타난다. 이런 경우는 에러가 없어 질 때 까지 반복적으로 .m2 폴더와 프로젝트를 지우면서 다시 프로젝트 생성해서 라이브러리를 다운받아 설치해야 한다. 하지만 과거와는 다르게 요즘은 인터넷만 잘 연결되면 설치가 잘 이루어지는 편이다.

마. **반드시 주의해야 할 한글 사용자 이름**

가끔 제대로 스프링 라이브러리가 설치가 안되는 경우는 사용자 이름과 폴더가 한글로 된 경우이다. 이런 경우는 사용자와 폴더를 영문 숫자조합으로 바꾸고 다시 다운받아 설

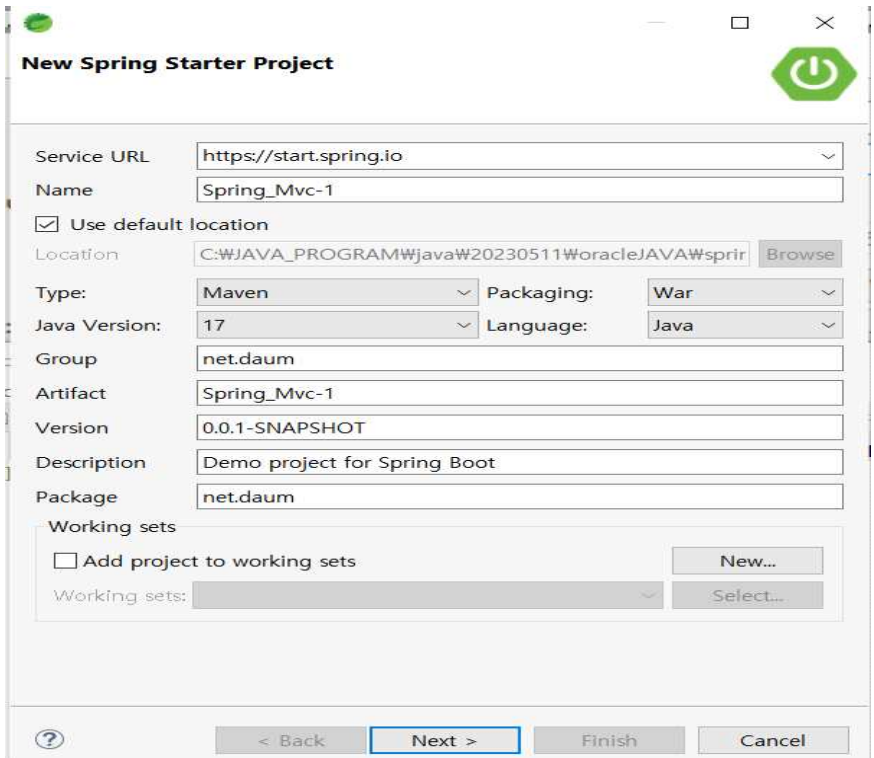
치해야 한다. 그래도 안되면 운영체제를 다시 포맷해서 사용자 이름과 컴퓨터 이름을 반드시 한글이 아닌 영문 숫자조합으로 하고 해당 프로젝트를 생성해야 한다. 그래야 제대로 라이브러리가 다운 받아 설치된다. 이 한글 사용자이름 경로는 오라클에서도 해당 디비 서버를 sql development 클라이언트 연결 툴에서도 연결을 못하는 문제를 발생시킨다.

2. 스프링 부트 프로젝트

가. 스프링 부트는 기존의 스프링 프레임워크를 이용해 왔던 사용자에게 좀 더 빠른 개발이 가능하고,손쉽게 프로젝트를 구성할 수 있는 다양한 방식을 제공한다. 스프링 부트를 사용하다 보면 기존에 스프링으로 개발하던 방식에서 어려웠던 설정이나, 개발환경 등이 하나의 선물 세트처럼 만들어 졌다는 느낌을 받는다. 스프링 개발에 필요했던 복잡한 설정이나 라이브러리 간의 충돌 등 여러문제가 해결되어 있기도 하고, 개발 부분에 있어 설정이 자동화된 부분이 많아져서 개발 자체에 집중할 수 있는 환경을 갖추고 있다.

나. 스프링 부트 1.5.4는 java 7, **스프링 부트 2.0이후는 java 8을 기본**으로 한다. **스프링 부트 3.0버전 이상은 JAVA 17 이상으로 지원하기 때문에 JDK 1.8을 사용하는 사용자 분들은 에러가 발생할 수 밖에 없다. 결국 Java 11이나 1.8 버전으로는 스프링 부트 3.0.X를 사용할 수 없다는 것이다.**

다. STS에서 File->New->Spring Starter Project에서 프로젝트 이름을 입력하고,Type을 Maven, Packaging을 War, Java Version 17, Language Java를 선택한다.



라. Group을 net.daum을 입력한다. 톰캣 WAS 서버가 내장되어 있어서 별도의 톰캣 서버를 설치할 필요가 없다.

마. Next를 누르고 다음과정에서 Core에서 Lombok을 선택한다. Lombok은 setter/getter등을 자동으로 생성해 준다.

바. 프로젝트를 생성할 때에는 필요한 라이브러리를 다운로드 받아야 하기 때문에 처음 설정할 때는 시간이 많이 걸린다.

사. 별도의 xml 설정이 필요 없고,프로퍼티 파일에서 설정내용을 기입한다.

아. Run-Spring Boot App을 선택하면 실행할 수 있다.

3. 스프링 프레임워크와 스프링 부트

가. 스프링 프레임워크는 1.0 버전이 2004년에 출시 되었고, 국내 도입은 조금 늦어졌고, java 웹개발자에겐 필수과목이 되었다.

나. 스프링 프레임워크는 기존의 EJB라는 무겁고 복잡한 개발에서 벗어나 경량화된 개발 추구하는 것에서 시작되었고, 로드 존슨이 자신의 책에서 작성한 코드에서부터 시작을 하여 빠른 속도로 발전되었다.

다. 스프링 부트는 이러한 간편한 개발 방식에 영감을 얻어서 만들어진 스프링 서브 프로젝트라고 할 수 있다. 스프링 부트는 기존의 스프링 개발 방식에서 불편했던 설정이나 버전 충돌 등의 불편했던 점들을 없애는 대신에 빠르고 쉬운 서버 환경과 테스트 환경 등을 한 번에 제공해서 훨씬 간편한 개발 환경을 마련하게 되었다.

라. 스프링 부트는 자동화된 환경설정,XML없는 환경구축,테스트환경, 내장 톰캣서버등을 들수 있다.

4. 스프링 MVC 프로젝트 템플릿 구조

가. src/main/java : 패키지 이하 원본 자바파일

나. src/main/resources : mybatis 관련 xml파일 경로, application.properties 파일에서 기존 스프링 프로젝트와는 다르게 톰켓포트 번호, 데이터 베이스 연결정보, mybatis 관련 정보, 뷰페이지 경로와 확장자인 뷰리졸브 경로나, JPA 관한 정보등을 설정한다. 또한 스프링 부트에서는 기본적으로 *.jsp 뷰페이지나 JSTL 문법을 제공하지 않고 근본적으로 탑재된 타임리프 뷰페이지 템플릿 엔진을 사용한다. 이 타임리프 뷰페이지 관한부분이 들어가는 경로이다.

다. src/test/java : 자바 테스트 경로

라. src/main/webapp: html,css,javascript,jQuery,이미지 등 리소스 자원 경로

마. src/main/webapp/WEB-INF/views : *.jsp 뷰페이지 파일 경로인 뷰리졸브

바. pom.xml : 메이븐 설정 파일

5. pom.xml 수정 내용

```
..중략

<!--  jsp 실행 가능하게 함. -->
            <dependency>
             <groupId>org.apache.tomcat.embed</groupId>
             <artifactId>tomcat-embed-jasper</artifactId>
             <scope>provided</scope>
            </dependency>

            <!-- JSTL과 서블릿 가능하게 함. 스프링 부트 2.0대 버전은 javax.servlet 패키지를 사용한다.
             <dependency>
             <groupId>javax.servlet</groupId>
             <artifactId>jstl</artifactId>
             <version>1.2</version>
            </dependency> -->

            <!-- JSTL 가능하게 함. 스프링부트 3.0이상이라면 아래 코드를 사용하여 의존성을 주입해줘야한다.
            스프링부트 3.0에서는 javax.servlet      =====>     jakarta.servlet으로 서블릿 패키지 경로가 변경됨-->
            <dependency> <!--서블릿 실행 -->
              <groupId>jakarta.servlet</groupId>
              <artifactId>jakarta.servlet-api</artifactId>
              <version>6.0.0</version>
              <scope>provided</scope>
        </dependency>

       <!--스프리 부트 3.0에서는 아래 JSTL 의존성 주입을 사용한다. -->
       <!-- copy begin -->
       <dependency>
            <groupId>jakarta.servlet.jsp.jstl</groupId>
            <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
            <version>3.0.0</version>
       </dependency>

      <dependency>
```

```
            <groupId>org.glassfish.web</groupId>
            <artifactId>jakarta.servlet.jsp.jstl</artifactId>
            <version>3.0.1</version>
        </dependency>
        <!--copy end -->

..중략..

        <!-- lombok 라이브러리 추가 : 다운로드 주소 ~ projectlombok.org    ,
cmd java -jar lombok.jar -->
            <!-- lombok 라이브러리 추가 -->
                <dependency>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                        <optional>true</optional>
                </dependency>

                <!-- 스프링 부트 2.0 버전에서는 톰캣 내장 서버 부분 주석문 처리
안해도 되지만 스프링 부트 3.0에서는 톰캣 10버전과 서블릿 , JSTL을 사용하기 위해서
는 이 부분을 주석문 처리해야 한다.
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-tomcat</artifactId>
                        <scope>provided</scope>
                </dependency>
                -->
..중략...
```

6. 스프링에 대한 간단한 소개

가. 프레임워크란 말 그대로 '뼈대나 근간'을 이루는 코드들의 묶음이라고 볼 수 있다. 프레임워크의 최대 장점은 개발에 필요한 구조를 이미 코드로 만들어 놓았기 때문에 실력이 부족한 개발자라 하더라도 반쯤 완성한 상태에서 필요한 부분을 조립하는 형태의 개발이 가능하다는 점이다.

나. 의존성이란 어떤 객체가 혼자 일을 처리할 수 없다는 것을 의미한다.

다. 의존성 주입의 종류
 1) 생성자를 통한 의존성 주입 2)setter() 메서드를 통한 의존성 주입

라. AOP는 Aspect Oriented Programming의 약어로서 반복적인 코드를 줄이고,핵심 비

즈니스 로직(개발코드)에만 집중할수 있게 해준다.
이 aop를 통한 데이터 베이스 트랙잭션 처리를 하게 한다. 스프링은 이런 트랜잭션의 관리를 애노테이션이나 xml로 설정할 수 있다.

7. JDBC 연결 자바 테스트 코드 만들기
jUnit을 이용한 JDBC의 연결 코드 작성은 src/test/java밑에 net.daum 패키지 하위에 OracleConTest.java파일을 작성한다.

```
package net.daum;

import java.sql.Connection;
import java.sql.DriverManager;

import org.junit.Test;

public class OracleConTest {

	private static final String DRIVER=
				"oracle.jdbc.OracleDriver";
	private static final String URL=
				"jdbc:oracle:thin:@127.0.0.1:1521:xe";
	//오라클 접속 주소, 1521은 포트번호, xe는 데이터베이스
	//명
	private static final String USER="week";
	//오라클 접속 사용자
	private static final String PW="week";//접속 비번

	@Test
	public void testCon() throws Exception{
		Class.forName(DRIVER);//오라클 jdbc 드라이버 클래
		//스 로드
		try(Connection con=
	DriverManager.getConnection(URL,USER,PW)){
				System.out.println(con);
		}catch(Exception e) {e.printStackTrace();}
/* jdk 1.7이후에 추가된 try-with구문이다. 이것은 명시적인
 * close()를 하지 않아도 알아서 close() 해준다.
 * AutoCloseable 인터페이스를 구현한 타입이어야 한다.
 */
	}
}
```

8. 스프링+mybatis+oracle 설정

가. mybatis는 ibatis프레임웍이 단종되고 나온 것이다. 이 둘은 부모와 자식 관계이다. 지금도 단종된 ibatis가 현업에서 조금 사용된다. 스프링을 이용한 개발중에서 국내에서 가장 많이 쓰이는 형태는 mybatis와의 연동작업을 통해서 SQL문 처리에 대한 개발 생산성을 높이는 형태로 사용되는 것이다.

나. 스프링 MVC프로젝트 실행 순서
컨트롤러-> 서비스 -> 모델 DAOImpl -> mybatis를 통한 쿼리문 실행이고 컨트롤러를 통해서 뷰페이지 즉 jsp 파일로 이동한다.

다. mybatis 프레임웍의 장점
간결한 코드로 sql문을 처리한다. sql문을 따로 분리운영할 수 있다. 스프링과의 연동으로 자동화된 처리가 가능하다.

9. mybatis 연결

가. MyBatis는 SQL 매핑 프레임워크를 별도의 설정 파일을 통해 관리 할 수 있다.

나. 'src/main/resources' 경로에 mybatis-config.xml 파일 하나를 생성해 둔다. 이 파일은 자바 저장빈 클래스 별칭 객체를 관리한다.

10. application.properties

```
#tomcat port number
server.port=8066

#oracle connect
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=week
spring.datasource.password=week

#view page path
spring.mvc.view.prefix=/WEB-INF/views/
#view page extension
spring.mvc.view.suffix=.jsp

#MyBatis
mybatis.config-location=classpath:mybatis-config.xml
mybatis.mapper-locations=classpath:org/zerock/mappers/**/*.xml
```

11. 기초적인 스프링 컨트롤러 생성 실습

org.zerock.controller 패키지내에 SampleController를 아래와 같이 작성한다.

```
package org.zerock.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller //@Controller 애너테이션을 사용하면 해당 컨트
//롤 클래스는 스프링에서 인식한다.
public class SampleController {

	@RequestMapping("doA") //get or post로 접근하는 doA
	//매핑주소를 웹주소창에서 실행하게 한다.
	public void doA7() {//리턴타입이 없는 void형이면 매핑
		//주소 doA가 jsp 파일명(doA.jsp) 즉 뷰페이지가 됨
		System.out.println("doA매핑주소가 실행됨");
	}

	@RequestMapping("doB")
	public void doB() {
		System.out.println("doB매핑주소가 실행됨.");
	}
}
```

12. 컨트롤러에서 메서드 리턴타입이 문자열인 경우는 '문자열+.jsp' 파일을 찾아서 실행하게 된다. SampleController2.java를 작성한다.

```
package org.zerock.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class SampleController2 {

	@RequestMapping("doC") //doC 매핑주소 등록
	public String doC(@ModelAttribute("msg") String msg) {
		//@ModelAttribute("msg")는 msg피라미터 이름에 인자
		//값을 문자열로 전달한다.웹주소 실행 매핑주소값으
		//doC?msg=문자열값 형태의 웹주소창에 노출되는 get
		//방식으로 전달한다.보안성이 안좋다.
		return "result";
		// WEB-INF/views/result.jsp 뷰페이지가 실행
```

```
        }
}
```

/WEB-INF/views/result.jsp는 아래와 같이 작성할 수 있다.

```
<%@ page contentType="text/html; charset=UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> </title>
</head>
<body>
  전달된 피라미터 값:${msg}<%--${출력할 값} 형태의 표현
  언어로 사용 --%>
</body>
</html>
```

13. org.zerock.controller 패키지에 SampleController3.java를 작성한다.

```
package org.zerock.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.zerock.domain.ProductVO;

@Controller
public class SampleController3 {

        @RequestMapping("/nameprice")
        public String nameprice(Model m) {
                ProductVO p=new ProductVO("신발",120000);//오버로
                //딩된 생성자를 호출하면서 멤버변수값 초기화
                m.addAttribute("p",p);//p키이름에 p객체 저장
                return "shop/pro_name";
                // WEB-INF/views/shop/pro_name.jsp 뷰페이지 파일
                //이 실행
        }
}
```

데이터 저장빈 클래스를 활용한 컨트롤러와 뷰페이지 작성 실습

먼저 org.zerock.domain패키지에 데이터 저장빈 클래스 ProductVO.java를 작성한다.

```
package org.zerock.domain;

public class ProductVO {

	private String name;//상품명
	private double price;//가격

	public ProductVO(String name,int price) {
		this.name=name;
		this.price=price;//생성자의 주된기능인 멤버변수
		//초기화
	}//생성자를 오버로딩(매개변수가 없는 기본생성자를 묵시
	//적 제공하지 않는다.)

	public String getName() {
		return name;
	}

	public double getPrice() {
		return price;
	}

	@Override
	public String toString() {
		return "ProductVO [name="+name+",price="+
	price+"]";
	}
}
```

WEB-INF/views/shop/pro_name.jsp 뷰페이지 파일을 작성한다.

```
<%@ page contentType="text/html; charset=UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> </title>
</head>
<body>
```

```
 상품명=${p.name}<br/>
 상품가격=${p.price}<hr/>
</body>
</html>
```

14. 서버단 스프링 컨트롤러에서 리다이렉트를 해야 하는 경우
가. 가끔은 특정한 컨트롤러 로직을 처리할 때 다른 경로를 호출해야 하는 경우가 있다. 이 경우에는 스프링 MVC의 특별한 문자열인 'redirect:'을 이용하는대 ':'을 이용하는 것에 주의할 필요가 있다.

나. 리다이렉트를 하는 경우 RedirectAttributes라는 클래스를 피라미터로 같이 사용하게 되면 리다이렉트 시점에 원하는 데이터를 임시로 추가해서 넘기는 작업이 가능하다.

먼저 org.zerock.controller 패키지에 다음과 같은 컨트롤러 클래스를 만든다.

```
package org.zerock.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

@Controller
public class SampleController4 {

	@RequestMapping("/doE")
	public String doE(RedirectAttributes rttr) {
		rttr.addFlashAttribute("msg","this is seven");
		//다른 매핑주소로 msg키이름에 값을 담아서 전달할때
		//사용한다. 서버상에서 실행되기 때문에 웹주소창에
		//전달되는 값이 보이지 않는다. 보안이 좋다.
		return "redirect:/doF";//doE매핑주소가 실행되면
		//다른 매핑주소인 doF로 이동
	}

	@RequestMapping("/doF")
	public void doF(@ModelAttribute("msg") String name) {
		System.out.println("전달된 값:"+name);
	}
}
```

15. JSON 데이터를 생성하는 경우

스프링 MVC의 장점 중 하나는 최근 프로그래밍에서 많이 사용되는 JSON(Javascript Object Notation) 데이터에 대한 처리를 너무나 간단하게 처리할 수 있다는 것이다.

스프링 컨트롤러에서 JSON데이터를 생성하기 위한 작업을 해야 한다. JSON객체를 반환하기 위해서 @ResponseBody 애노테이션을 추가해 주는 작업만 하면 된다. JSON데이터는 키,값 쌍구조로 되어 있다.

org.zerock.controller 패키지에 다음과 같은 컨트롤로 클래스 작업을 한다.

```
package org.zerock.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.zerock.domain.ProductVO;

@Controller
public class SampleController5 {

	@RequestMapping(value="/doJSON",produces="application/json")
	public @ResponseBody ProductVO doJSON() {
//@ResponseBody 애노테이션을 사용하면 jsp파일을 만들지 않
//고도 웹브라우저에 키,값 쌍 의 json데이터를 쉽게 출력할
//수 있다.
		ProductVO p=new ProductVO("수박",15000);
		return p;//json데이터 키이름에 ProductVO빈클래스
		//변수명이 된다.
	}
}
```

16. 스프링 + mybatis

mybatis는 JDBC에서 개발자가 직접 처리하는 PreparedStatement의 ?에 대한 설정이나 ResultSet을 이용한 처리가 이루어지기 때문에 기존 방식에 비해 개발의 생산성이 좋아진다. 국내의 대부분 프로젝트는 XML만을 이용해서 SQL문을 작성하고, 별도의 DAOImpl 만드는 방식을 선호한다. 이 방식의 최대장점은 sql문을 완전히 분리해서 처리하기 때문에 향후에 sql문의 변경이 일어날 때, 대처가 수월하다는 장점이 있다.

가장 먼저 개발에 필요한 테이블을 생성한다.

```
--tbl_member 테이블 만들기
create table tbl_member(
 userid varchar2(50) primary key --회원 아이디
 ,userpw varchar2(50) not null --회원비번
 ,username varchar2(50) not null --회원이름
 ,email varchar2(100) --전자우편
 ,regdate date --가입날짜
 ,updatedate date --수정날짜
);
```

17. Mybatis에서 sql문을 저장하는 존재를 Mapper라는 용어로 표현한다. 매퍼태그를 작성해서 쿼리문을 작성한다. org.zercok.mappers.member패키지에 매퍼 태그 xml파일 member.xml을 작성한다. mybatis는 기본적으로 PreparedStatement를 이용해서 처리된다. 개발자가 PreparedStatement에 들어가는 피라미터를 사용할 때는 #{}기호를 이용해서 처리한다.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper
 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

 <mapper namespace="Member">
 <!-- mybatis쿼리문을 다루는 태그를 매퍼라 한다. -->

  <!-- 회원가입 -->
  <insert id="m_in" parameterType="m">
   <!-- parameterType속성은 전달인자 타입 -->
   insert into tbl_member values(#{userid},#{userpw},
   #{username},#{email},sysdate,sysdate)
   <!-- #{userid}는 자바코드로 m.getUserid()로 표현 -->
  </insert>
 </mapper>
```

18. 테이블 컬럼명과 일치하는 변수명을 가진 데이터 저장빈 클래스를 만든다.

```
package org.zerock.domain;



public class MemberVO {
/* mybatis을 사용하기 위해서 테이블 컬럼명과 일치하는 빈
 * 클래스 변수명을 정의한다.
 */
	private String userid;//회원아이디
	private String userpw;//비번
	private String username;//회원명
	private String email;//전자우편
	private String regdate;//가입날짜
	private String updatedate;//수정날짜

	public String getUserid() {
		return userid;
	}
	public void setUserid(String userid) {
		this.userid = userid;
	}
	public String getUserpw() {
		return userpw;
	}
	public void setUserpw(String userpw) {
		this.userpw = userpw;
	}
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getRegdate() {
		return regdate;
```

```
	}
	public void setRegdate(String regdate) {
		this.regdate = regdate;
	}
	public String getUpdatedate() {
		return updatedate;
	}
	public void setUpdatedate(String updatedate) {
		this.updatedate = updatedate;
	}
}
```

19. src/test/java 자바 테스트 코드 작성하기

```
package net.daum;

import org.junit.jupiter.api.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

import net.daum.dao.MemberDAO;
import net.daum.vo.MemberVO;

@RunWith(SpringRunner.class)
@SpringBootTest
public class MemberDAOTest {

	@Autowired
	private MemberDAO memberDao;

	@Test
    public void testInsertMember() throws Exception{
		MemberVO m=new MemberVO();
		m.setUserid("ccccc9");
		m.setUserpw("99999");
		m.setUsername("홍길동");
		m.setEmail("hong@zercok.com");

		this.memberDao.insertM(m);//회원저장
	}
}
```

20. MemberDAO 인터페이스를 구현한 MemberDAOImpl.java 클래스를 작성한다.

```
package org.zerock.persistence;

import javax.inject.Inject;

import org.apache.ibatis.session.SqlSession;
import org.springframework.stereotype.Repository;
import org.zerock.domain.MemberVO;

@Repository //@Repository 애노테이션은 DAO를 스프링에서
//인식하게 한다.
public class MemberDAOImpl implements MemberDAO{

	@Inject //자동의존성 주입(DI설정)
	private SqlSession sqlSession;//mybatis쿼리문 실행객체
	//sqlSession생성

	@Override
	public void insertM(MemberVO m) {
	    this.sqlSession.insert("m_in",m);
	    /* 1.m_in은 member.xml에서 설정한 insert매퍼태그의
	     * 아이디명이다.동일한 아이디명이 있어면 안된다.
	     * 2.mybatis쿼리문 실행메서드
	     *  가.insert():레코드 저장
	     *  나.update():레코드 수정
	     *  다.delete():레코드 삭제
	     *  라.selectOne():단 한개의 레코드만 검색
	     *  마.selectList():하나이상의 레코드를 검색해서
	     *  컬렉션 List로 반환
	     */
	}
}
```

org.zerock.persistence 패키지에 MemberDAO 인터페이스를 만든다.

```
package org.zerock.persistence;

import org.zerock.domain.MemberVO;

public interface MemberDAO {

        void insertM(MemberVO m);
}
```