회원관리 만들기

- 어떤기능을 넣을까?

  [사용자]

  회원가입

  로그인

  로그아웃

  회원 정보 수정

  회원 탈퇴

  - 무엇을 저장할까?

idx	: primary key
user_id	: 사용자 아이디
password	: 비밀번호
name	: 이름
birth	: 생일
gender	: 성별
email	: 이메일
phone	: 전화번호
postCode	: 우편번호
addr1	: 주소
addr2	: 상세주소
use	: 인증여부(0은 미인증, 1은 인증, 2는 휴면계정)
level	: 권한(0은 준회원(미인증), 1는 회원, 2은 관리자......)

테이블을 만들자

```
CREATE SEQUENCE member_idx_seq;

CREATE TABLE member(
    idx NUMBER PRIMARY KEY,
    userid varchar2(100) NOT NULL UNIQUE,
    password varchar2(100) NOT NULL,
    name varchar2(50) NOT NULL,
    birth DATE NOT NULL,
    gender char(1) check(gender IN ('M','F')),
    email varchar2(100) NOT NULL,
    phone varchar2(20) NOT NULL,
    postcode varchar2(10) NOT NULL,
    addr1 varchar2(200) NOT NULL,
    addr2 varchar2(200) NOT NULL,
    use NUMBER check(use IN (0,1,2,3,4,5,6,7,8,9)),
    lev NUMBER check(lev IN (0,1,2,3,4,5,6,7,8,9))
);
INSERT INTO MEMBER VALUES
(member_idx_seq.nextval,'root','1234','최고관리자','1988-09-05','M'
,'ithuman202204@gmail.com','010-1234-5678',' ',' ',' ',1, 9);
INSERT INTO MEMBER VALUES
(member_idx_seq.nextval,'admin','1234','최고관리자','1988-09-05','M'
,'ithuman202204@gmail.com','010-1234-5678',' ',' ',' ',1, 9);
INSERT INTO MEMBER VALUES
(member_idx_seq.nextval,'master','1234','최고관리자','1988-09-05','M'
,'ithuman202204@gmail.com','010-1234-5678',' ',' ',' ',1, 9);
INSERT INTO MEMBER VALUES
(member_idx_seq.nextval,'webmaster','1234','최고관리자','1988-09-05','M'
,'ithuman202204@gmail.com','010-1234-5678',' ',' ',' ',1, 9);
INSERT INTO MEMBER VALUES
(member_idx_seq.nextval,'system','1234','최고관리자','1988-09-05','M'
,'ithuman202204@gmail.com','010-1234-5678',' ',' ',' ',1, 9);
INSERT INTO MEMBER VALUES
(member_idx_seq.nextval,'sys','1234','최고관리자','1988-09-05','M'
,'ithuman202204@gmail.com','010-1234-5678',' ',' ',' ',1, 9);

SELECT * FROM MEMBER;
```

패키지 3개를 만들자

- kr.human.member.vo
- kr.human.member.dao
- kr.human.member.service

MemberVO클래스를 만들자

```java
package kr.human.member.vo;

import java.util.Date;

import lombok.Data;

@Data
public class Member {
    private int idx;
    private String userid;
    private String password;
    private String name;
    private Date   birth;
    private String gender;
    private String email;
    private String phone;
    private String postCode;
    private String addr1;
    private String addr2;
    private int use;
    private int lev;
}
```

mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!-- 프로퍼티 파일을 사용 하겠습니다. -->
    <properties resource="db.properties" />
    <!-- 타입의 별칭을 지정한다. 줄여서 사용하겠다. VO를 만들때마다 추가한다. -->
    <typeAliases>
        <package name="kr.human.member.vo"/>
    </typeAliases>
    <!-- DB연결 정보 -->
    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC" />
            <dataSource type="POOLED">
                <property name="driver" value="${o.driver}" />
                <property name="url" value="${o.url}" />
                <property name="username" value="${o.username}" />
                <property name="password" value="${o.password}" />
            </dataSource>
        </environment>
    </environments>

    <!-- SQL명령이 들어있는 매퍼파일을 지정한다. 맵퍼파일을 여러개 지정이 가능하다.--
>
    <mappers>
        <mapper resource="testMapper.xml" />
        <mapper resource="memberMapper.xml" />
    </mappers>
</configuration>
```

testMapper.xml

```
<?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="test">

    <select id="selectToday" resultType="string">
        select sysdate from dual
    </select>

</mapper>
```

memberMapper.xml

```xml
<?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">

    <insert id="insert" parameterType="MemberVO">
    INSERT INTO MEMBER VALUES
    (member_idx_seq.nextval,#{userid},#{password},#{name},#{birth},#{gender}
    ,#{email},#{phone},#{postCode},#{addr1},#{addr2}, 0, 0);
    </insert>

    <update id="update" parameterType="MemberVO">
        update member set
            email = #{email},
            postCode = #{postCode},
            addr1 = #{addr1},
            addr2 = #{addr2}
        where
            idx=#{idx}
    </update>

    <delete id="delete" parameterType="int">
        delete from member where idx=#{idx}
    </delete>

    <select id="selectByIdx" parameterType="int" resultType="MemberVO">
        select * from member where idx=#{idx}
    </select>

    <select id="selectByUserid" parameterType="string"
resultType="MemberVO">
        select * from member where userid=#{userid}
    </select>

    <select id="selectList" resultType="MemberVO">
        select * from member order by idx desc
    </select>

    <select id="selectUseridCount" parameterType="string" resultType="int">
        select count(*) from member where userid=#{userid}
    </select>

    <update id="updateUse" parameterType="hashmap">
        update member set use=#{use} where idx=#{idx}
    </update>

    <update id="updateLevel" parameterType="hashmap">
        update member set lev=#{lev} where idx=#{idx}
    </update>
</mapper>
```

MemberDao 인터페이스를 만들자

```java
package kr.human.member.dao;

import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;

import org.apache.ibatis.session.SqlSession;

import kr.human.member.vo.MemberVO;

public interface MemberDAO {
    // 저장
    void insert(SqlSession sqlSession, MemberVO memberVO) throws
SQLException;
    // 수정
    void update(SqlSession sqlSession, MemberVO memberVO) throws
SQLException;
    // 삭제
    void delete(SqlSession sqlSession, int idx) throws SQLException;
    // 1개얻기(idx로 얻기)
    MemberVO selectByIdx(SqlSession sqlSession, int idx) throws
SQLException;
    // 1개얻기(userid로 얻기)
    MemberVO selectByUserid(SqlSession sqlSession, String userid) throws
SQLException;
    // 모두얻기(관리자)
    List<MemberVO> selectList(SqlSession sqlSession) throws SQLException;
    // 동일한 아이디 개수 얻기(중복확인)
    int selectUseridCount(SqlSession sqlSession, String userid) throws
SQLException;

    // 인증정보 변경
    void updateUse(SqlSession sqlSession, HashMap<String, Integer> map)
throws SQLException;
    // 레벨 변경
    void updateLevel(SqlSession sqlSession, HashMap<String, Integer> map)
throws SQLException;
}
```

MemberDaoImpl 클래스를 만들자

```java
package kr.human.member.dao;

import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;

import org.apache.ibatis.session.SqlSession;

import kr.human.member.vo.MemberVO;

public class MemberDAOImpl implements MemberDAO {
    private static MemberDAO instance = new MemberDAOImpl();
    private MemberDAOImpl() {}
    public static MemberDAO getInstance(){
        return instance;
    }
    //-----------------------------------------------------------------------------
---------
    @Override
    public void insert(SqlSession sqlSession, MemberVO memberVO) throws
SQLException {
        sqlSession.insert("member.insert", memberVO);
    }
    @Override
    public void update(SqlSession sqlSession, MemberVO memberVO) throws
SQLException {
        sqlSession.update("member.update", memberVO);
    }
    @Override
    public void delete(SqlSession sqlSession, int idx) throws SQLException {
        sqlSession.delete("member.delete", idx);
    }
    @Override
    public MemberVO selectByIdx(SqlSession sqlSession, int idx) throws
SQLException {
        return sqlSession.selectOne("member.selectByIdx", idx);
    }
    @Override
    public MemberVO selectByUserid(SqlSession sqlSession, String userid)
throws SQLException {
        return sqlSession.selectOne("member.selectByUserid", userid);
    }
    @Override
    public List<MemberVO> selectList(SqlSession sqlSession) throws
SQLException {
        return sqlSession.selectList("member.selectList");
    }
    @Override
    public int selectUseridCount(SqlSession sqlSession, String userid)
throws SQLException {
        return sqlSession.selectOne("member.selectUseridCount", userid);
    }
    @Override
    public void updateUse(SqlSession sqlSession, HashMap<String, Integer>
map) throws SQLException {
        sqlSession.update("member.updateUse", map);
    }
    @Override
    public void updateLevel(SqlSession sqlSession, HashMap<String, Integer>
map) throws SQLException {
        sqlSession.update("member.updateLevel", map);
    }

}
```

메일보내는 서비스 클래스를 만들자

```
package kr.human.member.service;

import java.util.Properties;

import javax.mail.Address;
import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import lombok.extern.slf4j.Slf4j;

@Slf4j
public class EmailService {
    private static Session mailSession;

    static {
        Properties p = new Properties(); // 정보를 담을 객체
        p.put("mail.smtp.host","smtp.naver.com"); // 네이버 SMTP 또는 gmail
SMTP
        p.put("mail.smtp.port", "465");
        p.put("mail.smtp.starttls.enable", "true");
        p.put("mail.smtp.auth", "true");
        p.put("mail.smtp.debug", "true");
        p.put("mail.smtp.socketFactory.port", "465");
        p.put("mail.smtp.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
        p.put("mail.smtp.socketFactory.fallback", "false");

        mailSession = Session.getInstance(p, new Authenticator(){
            protected PasswordAuthentication getPasswordAuthentication(){
                return new PasswordAuthentication("자기이메일", "자기비번");
            }
        });
    }

    public static void sendMail(String to, String subject, String content) {
        try {
            MimeMessage message = new MimeMessage(mailSession); // 메일의 내용
을 담을 객체
            Address fromAddress = new InternetAddress("자기이메일");
            message.setFrom(fromAddress);
            Address toAddress = new InternetAddress(to);      // 받는 사람
            message.addRecipient(Message.RecipientType.TO, toAddress);
            message.setSubject(subject);    // 제목
            message.setContent(content, "text/html;charset=UTF-8"); // 내용
            // 전송
            Transport.send(message);
            log.info("{}에게 메일전송 성공!!!", to);
        } catch (AddressException e) {
            log.info("에러발생 : {}", e.getMessage());
            e.printStackTrace();
        } catch (MessagingException e) {
            log.info("에러발생 : {}", e.getMessage());
            e.printStackTrace();
        }
    }
}
```

메일을 보내보자

```
package kr.human.member.service;

public class MailServiceTest {
    public static void main(String[] args) {
        EmailService.sendMail("ithuman202204@gmail.com", "제목", "<h1>내용입니
다.</h1>");
    }
}
```

임시 비번을 만들어주는 서비스 클래스를 만들어 보자

```
package kr.human.member.service;

import java.util.Random;

public class PasswordService {
    public static String makeNewPassword() {
        Random random  = new Random();
        String newPassword="";

        for(int i=0;i<10;i++) {
            switch (random.nextInt(4)) {
            case 0:
                newPassword += random.nextInt(10);
                break;
            case 1:
                newPassword += (char)('A'+ random.nextInt(26));
                break;
            case 2:
                newPassword += (char)('a'+ random.nextInt(26));
                break;
            case 3:
                newPassword += "~!@#$%^&*+-".charAt(random.nextInt(11));
            }
        }
        return newPassword;
    }

    public static void main(String[] args) {
        for(int i=0;i<10;i++) {
            System.out.println(makeNewPassword());
        }
    }
}
```