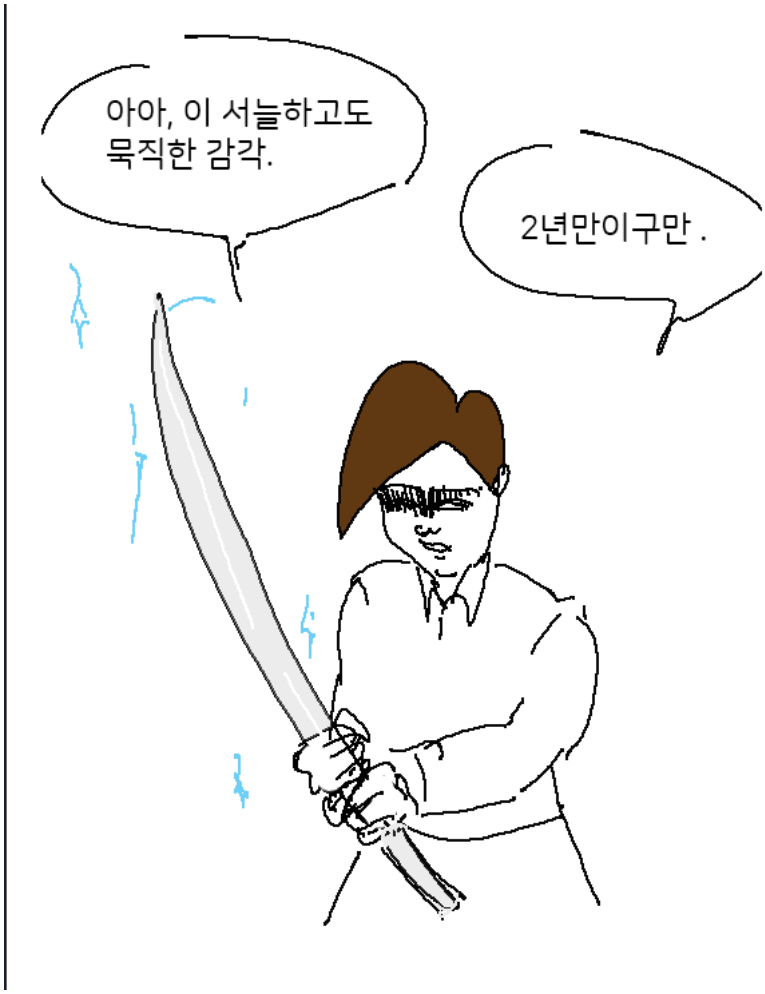
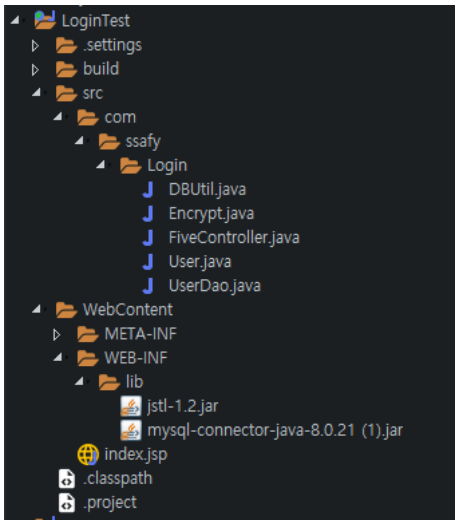
회원가입 암호화



(*알 고 시 러 웹 백 조 아*)



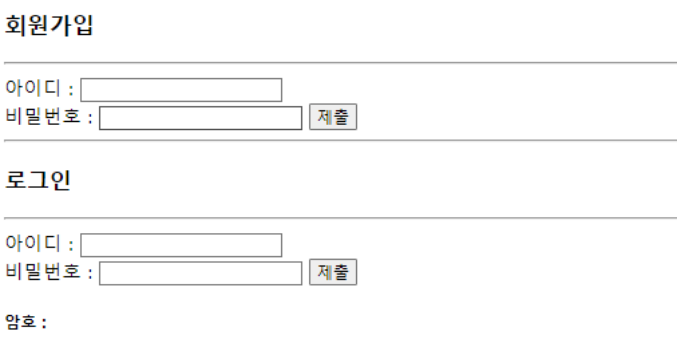
오늘은 간단한 암호화 작업과, 암호화 된 상태에서의 로그인 기능을 만들어봅시다. 크게 우리가 만들어 볼 파일들입니다.

(마.숙 0번 : 각각의 파일의 용도에 맞추어서 패키지 작업을 해봅시다)

index페이지

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
   <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>회원가입 / 로그인</title>
```

```
</head>
<body>

  <h3>회원가입</h3>
  <hr>
  <form method="post" action="main">
    <input type="hidden" name="act" value="regist">
    아이디 : <input type="text" name="id"> <br>
    비밀번호 : <input type="password" name="password">
    <input type="submit">
  </form>
  <hr>
  <h3>로그인</h3>
  <hr>
  <form method="post" action="main">
    <input type="hidden" name="act" value="login">
    아이디 : <input type="text" name="id"> <br>
    비밀번호 : <input type="password" name="password">
    <input type="submit">
  </form>

  <h5>암호 : ${password}</h5>

</body>
</html>
```



이런 화면이 나오는 간단한 회원가입 / 로그인 페이지입니다

(마.숙 1번 : 회원가입 / 로그인 창을 꾸며봅시다. 모달을 사용하거나, 메인 페이지에서 로그인 버튼을 누르면 로그인 페이지로 이동하거나, 메뉴바에 로그인 기능을 넣거나 등등 방식은 다양합니다)

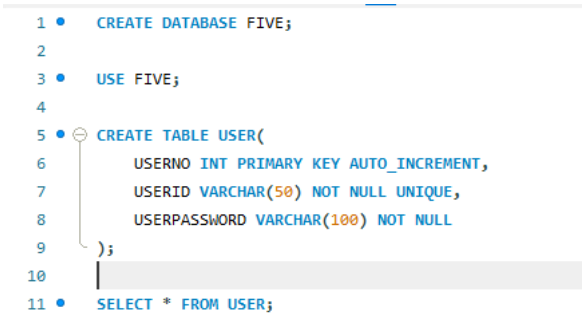
User 객체

```
public class User {

  private String id;
  private String password;
  public User() {
    super();
  }
  public User(String id, String password) {
    super();
    this.id = id;
    this.password = password;
  }
  public String getId() {
    return id;
  }
  public void setId(String id) {
    this.id = id;
  }
  public String getPassword() {
    return password;
  }
  public void setPassword(String password) {
    this.password = password;
  }
  @Override
  public String toString() {
```

```
    return "User [id=" + id + ", password=" + password + "]";
  }

}
```

User 객체도 아이디와 패스워드 두 개만 가지고 있습니다.

(마.숙 2번 : 우리가 평소에 회원가입을 하며 입력하던 정보들을 생각해보고(이름, 닉네임, 전화번호, 이메일 등등) 추가하여 User객체를 만들어봅시다! 그리고 추가한 정보에 맞추어 회원가입 페이지도 변경해봅시다)



```
CREATE DATABASE FIVE;

USE FIVE;

CREATE TABLE USER(
    USERNO INT PRIMARY KEY AUTO_INCREMENT,
    USERID VARCHAR(50) NOT NULL UNIQUE,
    USERPASSWORD VARCHAR(100) NOT NULL
);

SELECT * FROM USER;
```

FIVE라는 데이터베이스를 생성하고, AUTO_INCREMENT인 USERNO 컬럼과 아이디 / 패스워드 컬럼만으로 이루어진 USER 테이블을 생성합니다.

(마.숙 3번 : 추가하여 만든 USER 객체에 맞추어 테이블을 생성합시다. 데이터의 자료형을 어떻게 할 지 고민해보면서 만들어보세요)

DBUtil

```
package com.ssafy.Login;

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

/**
 * Mysql DB 연결 객체를 제공하주고, 사용했던 자원을 해제하는 기능을 제공하는 클래스입니다.
 */
public class DBUtil {
  /**
     * DB 접속에 필요한 url을 작성한다.
     * url은 jdbc:mysql://[host][:port]/[database][?propertyName1][=propertyValue1]형태로 작성한다.
     * serverTimezone=UTC 설정이 없으면 오류가 발생하므로 주의한다.
     */
  // DB와 연결하기위해 필요한 DB의 URL
  private final String url = "jdbc:mysql://localhost:3306/FIVE?serverTimezone=UTC";
  // DB의 USER 이름
  private final String username = "ssafy";
  // 위 USER의 PASSWORD
  private final String password = "ssafy";
  // Mysql 드라이버 클래스 이름
  private final String driverName = "com.mysql.cj.jdbc.Driver";

  /**
     * Singleton Design Pattern을 적용해준다.
     */
    private static DBUtil instance = new DBUtil();

    private DBUtil() {
        // JDBC 드라이버를 로딩한다. 드라이버 로딩은 객체 생성 시 한번만 진행하도록 하자.
        try {
            Class.forName(driverName);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    public static DBUtil getInstance() {
        return instance;
    }
```

```
    /**
     * DriverManager를 통해 Connection을 생성하고 반환한다.
     *
     * @return
     * @throws SQLException
     */
    public Connection getConnection() throws SQLException{
      return DriverManager.getConnection(url, username, password);
    }

  /**
     * 사용한 리소스들을 정리한다.
     * Connection, Statement, ResultSet 모두 AutoCloseable 타입이다.
     * ... 을 이용하므로 필요에 따라서
     * select 계열 호출 후는 ResultSet, Statement, Connection
     * dml 호출 후는 Statement, Connection 등 다양한 조합으로 사용할 수 있다.
     *
     * @param closeables
     */
    public void close(AutoCloseable... closeables) {
        for (AutoCloseable c : closeables) {
            if (c != null) {
                try {
                    c.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

우리가 수업 때 사용했던 Util을 재사용했습니다. 데이터베이스에 맞추어 url의 데이터베이스를 FIVE로 작성했습니다.

Encrypt

```
package com.ssafy.Login;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class Encrypt {

  // password : 우리가 입력할 비밀번호
  public String getEncrypt(String password) {

    String result = "";

    try {

      // SHA256 알고리즘 객체 생성
      // MessageDigest : 자바에 내장된 암호화를 도와주는 클래스입니다
      MessageDigest md = MessageDigest.getInstance("SHA-256");

      // 문자열(우리가 입력한 패스워드)에 SHA256 적용
      md.update(password.getBytes());
      byte[] pwd = md.digest();

      // byte to String(10진수의 문자열로 변경)
      StringBuffer sb = new StringBuffer();
      for(byte b : pwd) {
        sb.append(String.format("%02x", b));
      }

      result = sb.toString();

    } catch (NoSuchAlgorithmException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }

    // 반환
    return result;

  }

}
```

오늘 가장 중요한 기능! 암호화를 위한 클래스입니다. 오늘 사용할 암호화 알고리즘은 SHA-256 알고리즘입니다.(참고:http://wiki.hash.kr/index.php/SHA256) 알고리즘에 대해서도 추가적으로 검색해보시면 좋습니다! 물론 오늘의 목적은 알고리즘에 대

해서 자세히 아는게 아닙니다. 회원가입 / 로그인 시에 이런 알고리즘들을 사용하며 기능을 만들고 있다는 사실을 아는 것, 우리는 어떤 내용을 검색하고 참고하며 기능을 만들어볼까 고민하는게 목적입니다.

위의 코드를 보시면 아시겠지만 매우 간단합니다. 석박사들이 만든 알고리즘에 따라 암호화를 진행해줘! 하면 알아서 해줍니다. 우리는 그렇게 암호화 된 문자열을 DB에 넣어주기만 하면 됩니다. 물론 SHA256 알고리즘이 어떤 과정으로 암호화시키는 지 알면 더 좋겠지만, 오늘은 크게 몰라도 상관없습니다.

(마.숙 4번 : SHA 256 알고리즘의 특징에 대해 알아보세요. 그리고 다른 암호화 알고리즘에는 무엇이 있는지도 알아보세용)

컨트롤러

```
package com.ssafy.Login;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class ssafitController
 */
@WebServlet("/main")
public class FiveController extends HttpServlet {
  private static final long serialVersionUID = 1L;

  private UserDao uDao = UserDao.getInstance();
  private Encrypt ecp = new Encrypt();

  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html; charset=UTF-8");

    try {
      doProcess(request, response);
    } catch (NumberFormatException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } catch (SQLException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }

  }

  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    doGet(request, response);
  }


  private void doProcess(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException, NumberFormatE

    String act = request.getParameter("act");

    switch(act) {
    case "regist" : doRegistUser(request, response); break;
    case "login" : doLoginUser(request, response); break;
    }

  }

  // 회원가입
  private void doRegistUser(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    // input에 입력한 id와 password
    String id = request.getParameter("id");
    String password = request.getParameter("password");

    // 암호화!!
    String ecpPassword = ecp.getEncrypt(password);

    // 아이디와 암호화한 비밀번호로 새로운 user 객체 생성
    User u = new User(id, ecpPassword);
```

```
    // 그렇게 생성한 user를 회원가입
    uDao.registUser(u);
    request.getRequestDispatcher("/index.jsp").forward(request, response);

  }

  // 로그인
  private void doLoginUser(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    // input에 입력한 id와 password
    String id = request.getParameter("id");
    String password = request.getParameter("password");

    // 로그인 시에도 암호화를 진행해야한다.
    // 우리가 회원가입할 때 비밀번호를 암호화해서 DB에 저장했기 때문에
    // 비교하기 위해서 암호화 한 비밀번호를 들고가야한다
    String ecpPassword = ecp.getEncrypt(password);

    User u = uDao.loginUser(id, ecpPassword);

    // 암호화 한 비밀번호를 REQUEST에 담아줌
    request.setAttribute("password", u.getPassword());
    request.getRequestDispatcher("/index.jsp").forward(request, response);

  }

}
```

UserDao

```
package com.ssafy.Login;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import com.ssafy.Login.DBUtil;

public class UserDao {

  private final DBUtil util = DBUtil.getInstance();

  private static UserDao instance = new UserDao();

  private UserDao() {

  }

  public static UserDao getInstance() {
    return instance;
  }

  public void registUser(User u) {

    String sql = "INSERT INTO USER(USERID, USERPASSWORD) VALUES(?, ?);";

    Connection conn = null;
    PreparedStatement pstmt = null;

    int count = 0;

    try {
      conn = util.getConnection();
      pstmt = conn.prepareStatement(sql);

      pstmt.setString(1, u.getId());
      pstmt.setString(2, u.getPassword());

      count = pstmt.executeUpdate();

    } catch (SQLException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } finally {
      util.close(pstmt, conn);
    }


  }
```

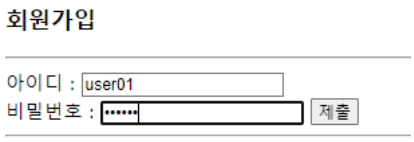
```
  public User loginUser(String id, String password) {

    String sql = "SELECT * FROM USER WHERE USERID = ? AND USERPASSWORD = ?";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rset = null;
    User u = new User();

    int count = 0;

    try {
      conn = util.getConnection();
      pstmt = conn.prepareStatement(sql);

      pstmt.setString(1, id);
      pstmt.setString(2, password);

      rset = pstmt.executeQuery();
      if(rset.next()) {
        u.setId(rset.getString("USERID"));
        u.setPassword(rset.getString("USERPASSWORD"));
      }

    } catch (SQLException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } finally {
      util.close(pstmt, conn);
    }

    return u;

  }

}
```

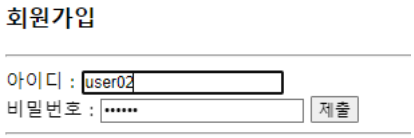
편의성을 위해서 인터페이스도 만들지 않고 진행했습니다. 컨트롤러에는 약간의 주석을 남겨놨지만, DAO에는 일부러 주석을 생략했습니다. 간단한 SQL문과 간단한 기능이니 Connection 객체가 무슨 역할인지, stmt와 pstmt가 무슨 차이인지 등을 생각하면서 코드를 해석해보세요!

(마.숙 5번 : USER 객체에 다른 정보들을 추가하여 생성했다면, 그로 인한 SQL문의 변경도 필요합니다! 츄라이츄라이

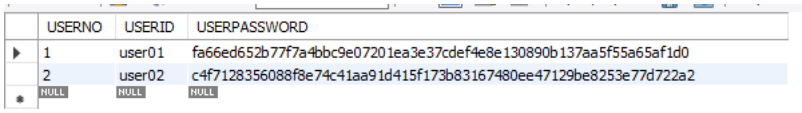
마.숙 6번 : MVC패턴 (USER 객체, FiveController, UserDao)의 흐름에 대해서 생각해보세요!

이렇게 만든 회원가입 기능을 통해 user01 / pass01, user01 / pass02라는 아이디 / 비밀번호로 회원가입을 해보겠습니다.





그리고 DB에서 저장된 비밀번호를 확인해보면…

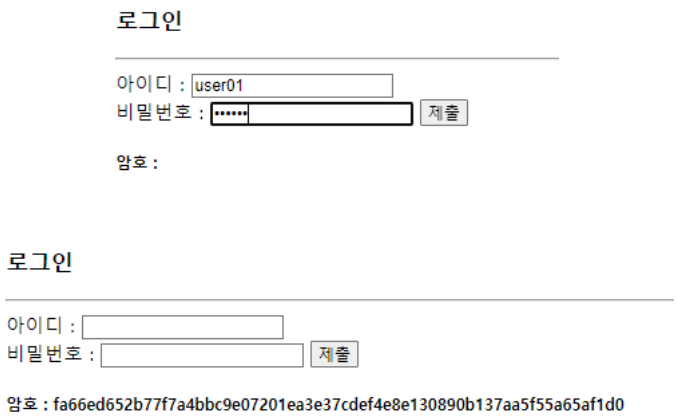


USERNO	USERID	USERPASSWORD
1	user01	fa66ed652b77f7a4bbc9e07201ea3e37cdef4e8e130890b137aa5f55a65af1d0
2	user02	c4f7128356088f8e74c41aa91d415f173b83167480ee47129be8253e77d722a2
NULL	NULL	NULL

이게 암호구나 싶은 문자열이 저장되어 있습니다!

pass01과 pass02는 숫자 하나의 차이지만 암호화된 문자열은 굉장히 다른 모습을 가지고 있는 것도 알 수 있네요.

그리고 로그인을 진행해보면



정상적으로 진행되어 암호화 된 비밀번호를 볼 수 있습니다!

우리가 했던 회원가입/로그인과는 약간 다른 암호화 회원가입/로그인을 진행했습니다. 제가 생각나는 추가사항들을 몇몇 적어봤지만 이외의 다른 추가사항들이 무수히 많을겁니다. 여기서 이렇게 하면 어떨까, 저렇게 하면 어떨까 하는 생각이 떠오르신다면 한번 추가해보세용 ㅎㅎ 이렇게 만든 코드들은 우리의 싸피 프로젝트와는 별개이기 때문에 깃허브에 레포지터리를 만들어 커밋해도됩니다. 아직 백준허브로만 커밋을 진행하신 분들이 많으실텐데, 이렇게 하나하나 기능을 만들어서 커밋하는 것도 해봅시다!!!

그리고 아직 이클립스 설정 / 라이브러리 추가 / 톰캣 서버 설정 등에 익숙치 않으신 분들이 많으실텐데, 우리는 매일매일 같은 컴퓨터 / 프로젝트를 사용하다보니 이러한 어려움을 잘 겪지 않고 있습니다. 하지만 이번 기회에 새롭게 워크스페이스를 파서 이것저것 설정하면서 진행해보세요. 진행하며 생기는 오류들은 어지간하면 구글에 있습니다! 개발자에게 키보드는 없어도 코딩할 수 있지만 구글이 없으면 코딩할 수 없다는 말도 있습니다.(없을지도?) 여하튼 한번 해보세용

물론 가장 권장하는 방식은 스스로 생각하며 기능을 구현하는 겁니다. 하지만 너무 막막하고 어렵다면 제 코드를 따라쓰기라도 하면서 이 코드가 무슨 역할인지 생각해보며 진행해보세요! 하지만 컨트롤씨 컨트롤브이는 절대 안됩니다. 진.짜.안.댐(궁서체)

그리고 이번 미니 실습은 오류 / 모르는게 생겨도 우선순위는 구글입니다. 혼자서 어떻게 어디서 검색해서 무엇을 참조할 지 고민하면서 진행해보세요. 뭘 검색해야 하는지 너무너무 모르겠다 하시면 저에게 " ~~ 한 부분에서 막혔는데 모르겠어요" 하시면 뭘 검색하라고 추천해드리겠습니다.(저한테 물어보지말라했다고 다른 옆사람한테 물어보지마세요) 그렇게 고민해보며 진행하시고, 그래도 모르겠다면 다시 찾아오세요! 어떻게 해결할지 같이 고민해봅시다.

완성하시면 요렇게저렇게 했다고 저에게 말씀해주시면 보람찰 것 같습니다 ㅎㅎㅎㅎ

추가사항들은 생각날때마다 추가하고 뭐뭐 추가했다고 써놓을게용