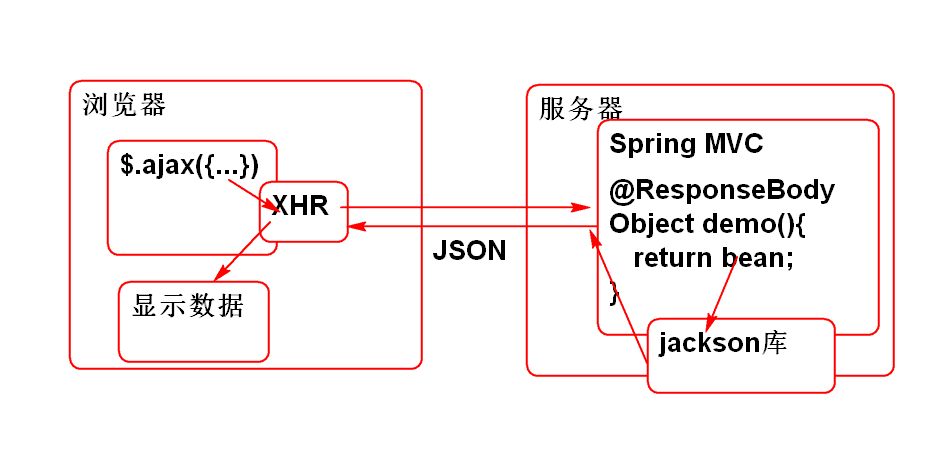
* **云笔记**

**Spring MVC AJAX**

Spring MVC 提供了 Ajax支持:



**1.必须导入 jackson API**

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-annotations</artifactId>

<version>2.2.3</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-core</artifactId>

<version>2.2.3</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.2.3</version>

</dependency>

**2.在控制器方法上使用 @ResponseBody 注解**, 就可以自动的将控制器的返回值(JavaBean)转换为JSON发送到客户端.

案例:

**作业**

1. 实现Spring MVC AJAX 功能
2. windows 用户安装MySQL
   * 建议安装 MySQL 5.7 版本, 默认就是UTF-8编码.
   * 注意: 数据库编码一定选择 UTF-8, 否则会中文乱码
   * 注意: 安装时候不要忘记密码!
   * 如果有问题, 可以找项目经理老师协助解决.

* **云笔记**

**1.JSON 是什么**

JSON 就是JS的一部分: JSON 是 JavaScript中的对象直接量语法, 用于声明JS对象.

如:

var i = 5;

var str = 'abc';

var obj = {age:5, name:'Tom'};

var ary = ['Tom','Jerry','Andy'];

var list = [{name:"Tom",age:8},{name:"Jerry",age:9}];

符合JSON语法的字符串称为JSON字符串:

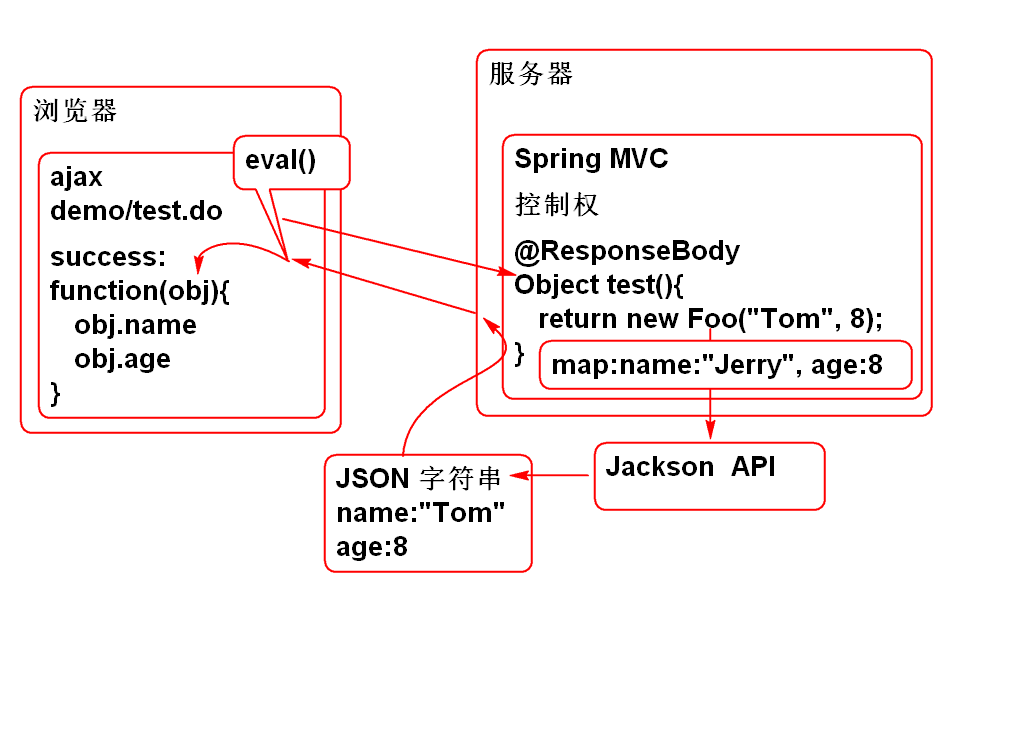
var str = '{"name":"Tom","age":6}';

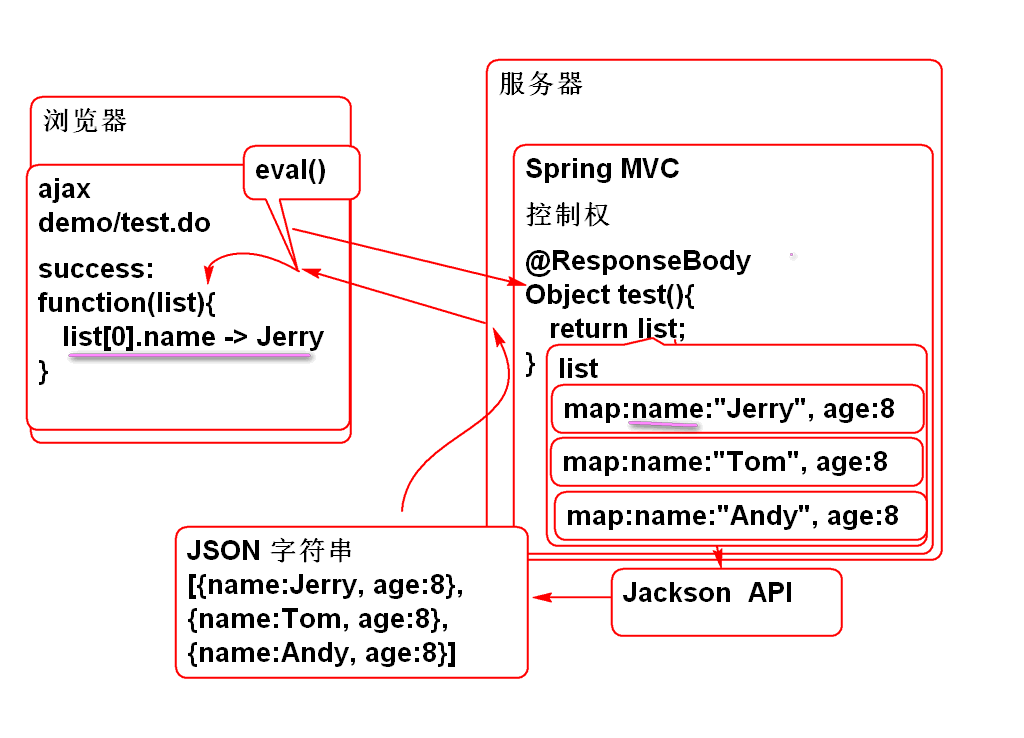
var str = '["name","Tom","age"]';

如何将JSON字符串转换为JSON对象(JS对象)

var obj = eval("("+str+")");

**2.Spring MVC 支持AJAX原理**





**3.需求分析**

软件= 数据结构+算法

**4.设计数据的存储**

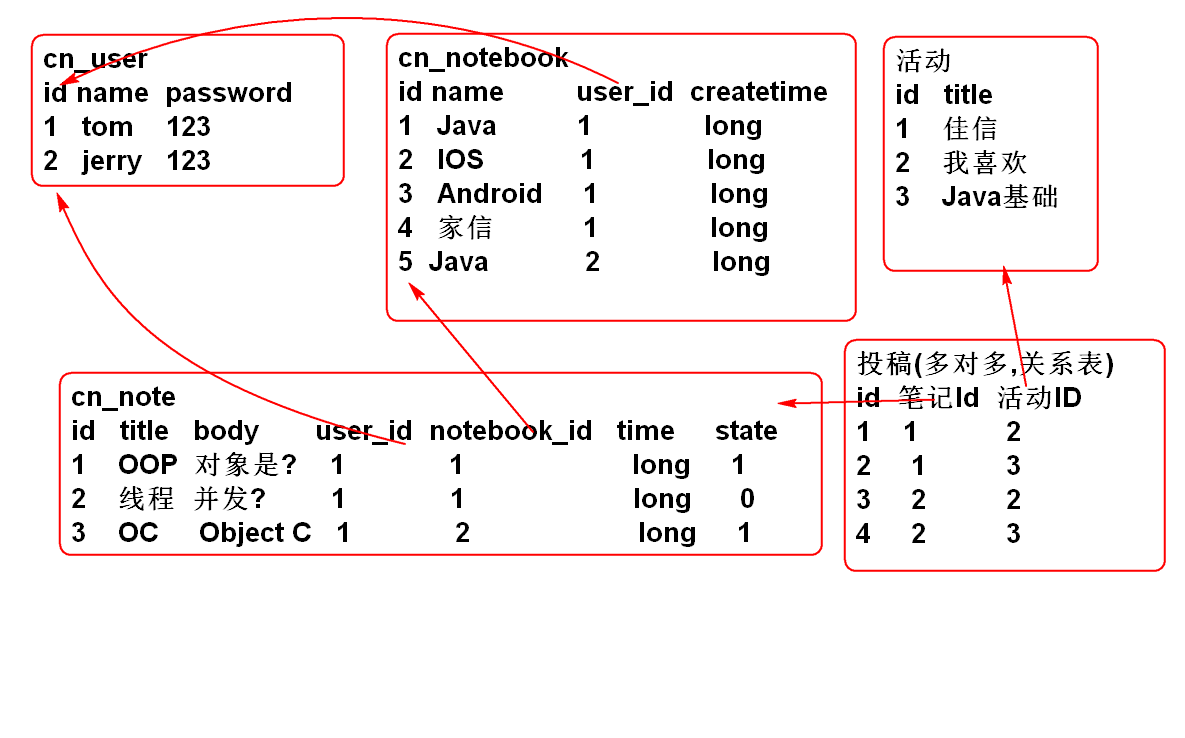
1.内存: 数据临时计算存储场所, 关闭电源就消失, 称为瞬态的运行内存分配的是对象

2.外存: (SSD固体硬盘, HDD机械硬盘, SD, 闪存, U盘), 关闭电源不消失, 称为持久状态存储容量

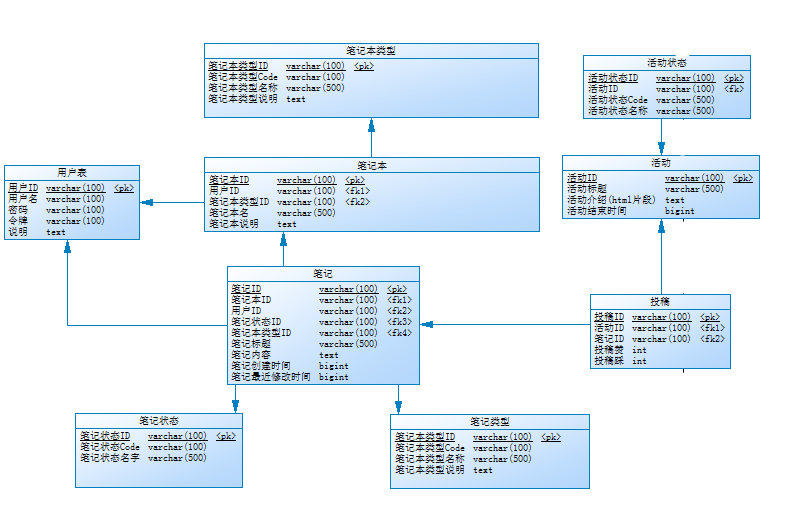
保存的是文件 单机(单线程)程序可以使用文件存储数据

数据库(文件) 数据库管理系统,提供了并发访问管理! 适合多线程访问云笔记采用数据库存储数据!

**5.设计数据的存储**



ER 图:



**6.MySQL的简单使用**

1. 使用MySQL命令行工具
   * Windows 用户使用: MySQL Client, 输入密码
   * Linux:

mysql -u用户名 -p密码 mysql -uroot -p

1. 显示数据库命令

show databases;

1. 创建数据库命令

create database 数据库名;

1. 删除数据库命令

drop database 数据库名;

1. 切换当前数据库

use 数据库名 use mysql

1. 显示当前数据库中的全部表

show tables;

1. 建表语句, 在当前数据库中创建表

create table 表名 (列的声明...)

1. 设置当前命令行窗口的编码: 设置当前窗口的文本编码为UTF-8

set names utf8;

1. 执行sql脚本命令: 执行文本文件中的一批SQL命令.
   * 如果SQL文件是UTF-8编码的, 就必须先执行 set names utf8;

source 文本文件的路径名; source D:\Robin\Note\note*ziliao\cloud*note.sql

source /home/soft01/note*ziliao/cloud*note.sql

1. 请注意: 一定要清楚路径, 保障路径的正确性!!!

案例: 执行脚本建立数据表:

source /home/soft01/note\_ziliao/cloud\_note.sql show databases; use cloud\_note; show tables;

案例: 创建一张表, 并且插入数据.

create database demo; use demo create table MyTable(id int, name varchar(100));

insert into MyTable (id, name) values (1, 'Tom');

insert into MyTable (id, name) values (2, 'Jerry');

select id, name from MyTable; drop table MyTable;

drop database demo;

**7.环境搭建**

1. 创建项目
   * 创建Maven项目
   * 生成web.xml
   * 导入目标运行环境:Tomcat运行环境
2. 导入要的包: pom.xml
3. 配置Spring MVC控制器: web.xml

<servlet>

<description></description>

<display-name>DispatcherServlet</display-name>

<servlet-name>DispatcherServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<description></description>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:conf/spring-\*.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>DispatcherServlet</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

1. 添加Spring-MVC 配置文件: conf/spring-mvc.xml

<!-- 配置组件扫描 -->

<context:component-scan base-package="cn.tedu.note.controller"/>

<!-- 配置MVC注解扫描 --> <mvc:annotation-driven />

1. 添加数据库连接参数文件: conf/jdbc.properties:

driver=com.mysql.jdbc.Driver

url=jdbc:mysql://localhost:3306/cloud\_note

user=root

password=root

maxActive=20

1. 配置MyBatis: conf/spring-mybatis.xml

<!-- 配置 spring-mybatis.xml -->

<!-- 读取配置文件 -->

<util:properties id="jdbc" location="classpath:conf/jdbc.properties"/>

<!-- 配置数据库连接池 -->

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">

<property name="driverClassName" value="#{jdbc.driver}"/>

<property name="url" value="#{jdbc.url}"/> <property name="username" value="#{jdbc.user}"/>

<property name="password" value="#{jdbc.password}"/>

<property name="maxActive" value="#{jdbc.maxActive}"></property>

</bean>

<!-- 配置MyBatis的 SessionFactory --> <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"> <property name="dataSource" ref="dataSource"/>

<property name="mapperLocations" value="classpath:mapper/\*.xml"/>

</bean>

<!-- Mapper接口组件扫描 -->

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="cn.tedu.note.dao"/>

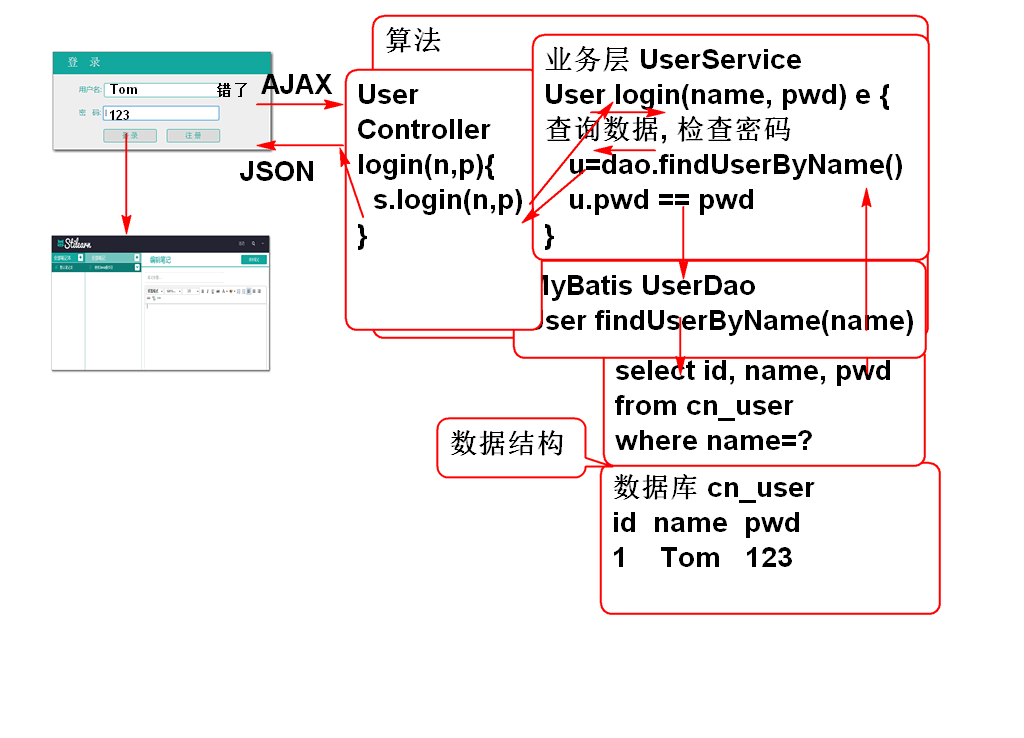
</bean>

* **登录功能**

程序 = 数据结构 + 算法

编程思路:

1. 搞清楚业务需求.
2. 考虑数据如何存储.
3. 如何操作数据才能达到业务目的
   * 设计SQL
4. 编程: 将业务功能和数据操作整合起来, 如何编程才能执行SQL操作数据



**1. 数据持久层开发**

1. 添加UserDao接口:

public interface UserDao { User findUserByName(String name); }

1. 添加实体类 User

public class User implements Serializable{

private static final long serialVersionUID = -288301177049569372L;

private String id;

private String name;

private String password;

private String token;

private String nick;

无参构造

有参构造

Get.Set方法

重写ToString方法

1. 添加SQL: mapper/UserMapper.xml:

<mapper namespace="cn.tedu.note.dao.UserDao">

<select id="findUserByName" parameterType="string" resultType="cn.tedu.note.entity.User">

select

cn\_user\_id as id,

cn\_user\_name as name,

cn\_user\_password as password,

cn\_user\_token as token,

cn\_user\_nick as nick from

cn\_user where cn\_user\_name = #{name}

</select> </mapper>

1. 测试 UserDaoTest:

public class UserDaoTest { ClassPathXmlApplicationContext ctx; @Before public void initCtx(){ ctx = new ClassPathXmlApplicationContext( "conf/spring-mvc.xml", "conf/spring-mybatis.xml"); } @After public void closeCtx(){ ctx.close(); } @Test public void testFindUserByName(){ String name = "demo"; UserDao dao = ctx.getBean( "userDao", UserDao.class); User user = dao.findUserByName(name); System.out.println(user); } }

1. Good Luck!

**2. 业务层**

1. 创建业务层接口

public interface UserService {

/\*\* \* 登录功能, 登录成功返回用户信息, 登录失败 \* 则抛出异常.

\* @param name 用户名 \* @param password 密码 \* @return 如果登录成功就返回登录用户信息

\* @throws UserNotFoundException 用户不存在

\* @throws PasswordException 密码错误

\*/ User login(String name, String password) throws UserNotFoundException, PasswordException; }

1. 创建业务异常类

public class UserNotFoundException extends RuntimeException {

。。。。。。

}

1. 实现业务层方法

@Service("userService")

public class UserServiceImpl implements UserService {

@Resource

private UserDao userDao;

public User login(String name, String password) throws UserNotFoundException, PasswordException {

if(password==null || password.trim().isEmpty()){ throw new PasswordException("密码空"); }

if(name==null || name.trim().isEmpty()){ throw new UserNotFoundException("用户名空"); }

User user = userDao.findUserByName( name.trim());

if(user==null){ throw new UserNotFoundException("name错误"); }

if(password.trim().equals(user.getPassword())){ return user; } throw new PasswordException("密码错误"); } }

1. 配置Spring, 扫描业务层组件的注解 conf/spring-service.xml:

<context:component-scan base-package="cn.tedu.note.service"/>

1. 测试
   * 定义测试基类, 封装Spring容器初始化功能:

public abstract class BaseTest {

protected ClassPathXmlApplicationContext

ctx;

@Before

public void initCtx() {

ctx = new ClassPathXmlApplicationContext( "conf/spring-mvc.xml", "conf/spring-mybatis.xml", "conf/spring-service.xml"); }

@After

public void close() { ctx.close(); } }

* + 测试登录方法

public class UserServiceTest {

@Test

public void testLogin(){ String name = "demo"; String password = "123456"; UserService service = ctx.getBean("userService", UserService.class);

User user = service.login( name, password); System.out.println(user); } }

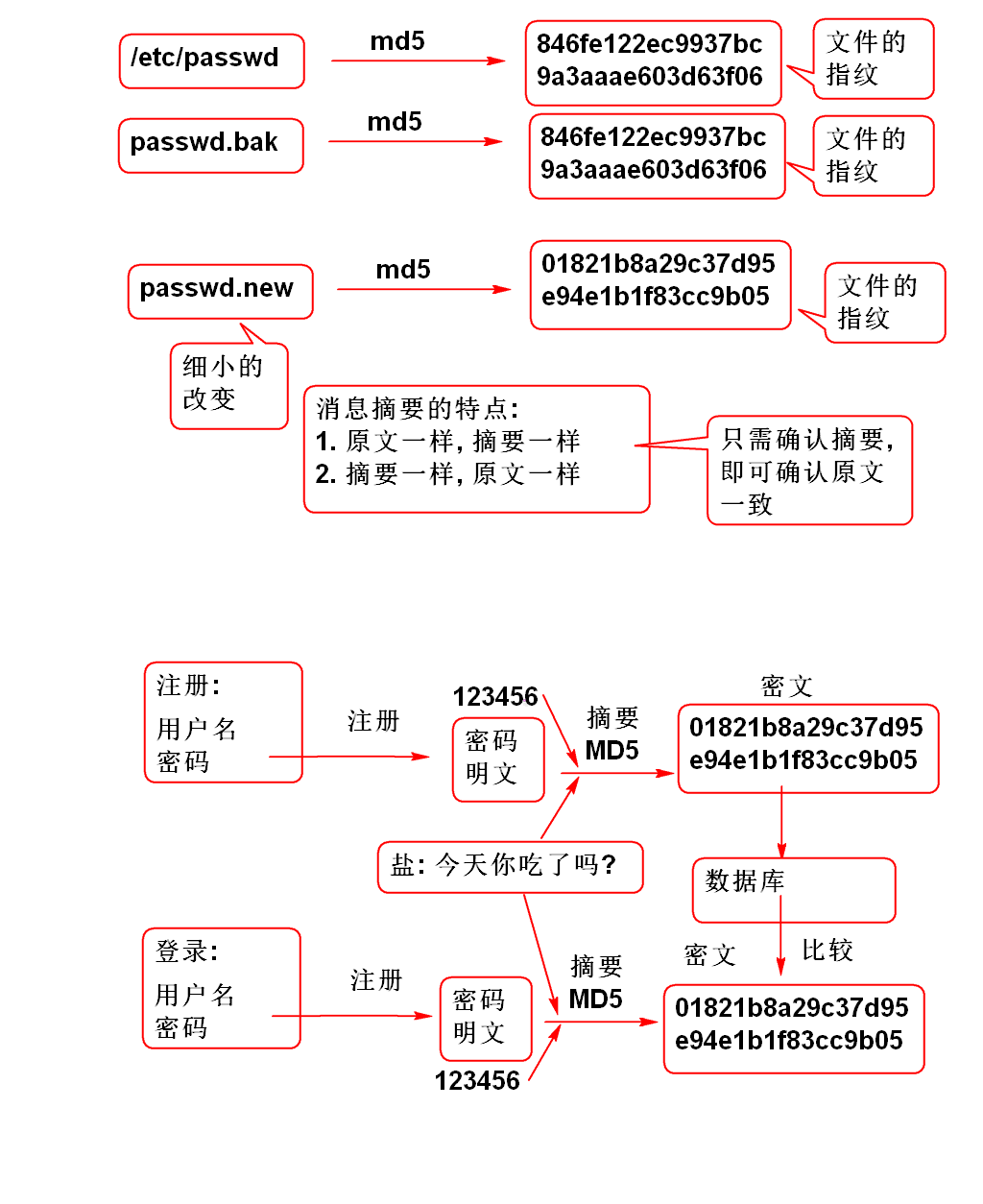
1. 为了测试, 需要更新数据库, 更新数据库的SQL
2. use cloud\_note;

update cn\_user set cn\_user\_password='123456' where cn\_user\_name='demo';

select cn\_user\_password from cn\_user where cn\_user\_name='demo';

**3. 密码的加密处理**

利用消息摘要技术加密用户密码, 避免保存密码明文, 可以有效的提供系统的安全性. 其原理如下:



加密算法测试:

public class Md5Test {

@Test

public void testMd5(){

String str = "123456";

String md5 = DigestUtils.md5Hex(str);

System.out.println(md5);

//e10adc3949ba59abbe56e057f20f883e

//加盐摘要

String salt = "今天你吃了吗?";

md5 = DigestUtils.md5Hex(salt+str);

System.out.println(md5);

//8bbf37d0962bf915a780f410ec1a4741

// update cn\_user

// set cn\_user\_password='8bbf37d0962bf915a780f410ec1a4741'

// where cn\_user\_name='demo'; } }

实现密码加密验证功能:

1. 导入消息摘要工具类:

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

<version>1.10</version>

</dependency>

1. 重构登录逻辑 验证摘要密码: UserServiceImpl

public User login(String name, String password) throws UserNotFoundException, PasswordException {

if(password==null || password.trim().isEmpty()){ throw new PasswordException("密码空"); }

if(name==null || name.trim().isEmpty()){ throw new UserNotFoundException("用户名空"); }

User user = userDao.findUserByName( name.trim());

if(user==null){ throw new UserNotFoundException("name错误"); }

String salt="今天你吃了吗?";

String pwd = DigestUtils.md5Hex( salt+password.trim());

if(pwd.equals(user.getPassword())){ return user; } throw new PasswordException("密码错误"); }

1. 更新数据库, 将测试用户的密码更新为加密的 123456

update cn\_user set cn\_user\_password='8bbf37d0962bf915a780f410ec1a4741' where cn\_user\_name='demo';

1. 测试:

@Test public void testLogin(){

String name = "demo";

String password = "123456";

UserService service = ctx.getBean("userService", UserService.class);

User user = service.login( name, password);

System.out.println(user); }

**4. 实现控制器**

1. 编写 JsonResult 封装JSON返回值:

public class JsonResult implements Serializable{

public static final int SUCCESS = 0;

public static final int ERROR = 1;

private int state; /\*\* 错误消息 \*/

private String message; /\*\* 返回正确时候的数据 \*/

private Object data; public JsonResult() { } public JsonResult(Object data){ state = SUCCESS; this.data = data; } public JsonResult(Throwable e) { state = ERROR; message = e.getMessage(); }

public int getState() { return state; }

public void setState(int state) { this.state = state; }

public String getMessage() { return message; } public void setMessage(String message) { this.message = message; } public Object getData() { return data; }

public void setData(Object data) { this.data = data; }

@Override

public String toString() { return "JsonResult [state=" + state + ", message=" + message + ", data=" + data + "]"; } }

1. 实现控制器: UserController

@Controller

@RequestMapping("/user")

public class UserController { @Resource private UserService userService;

@RequestMapping("/login.do")

@ResponseBody

public Object login( String name, String password){ User user = userService.login( name, password); return new JsonResult(user); }

/\*\* \* 在其他控制器方法执行出现异常时候, 执行 \* 异常处理方法

handleException \*/

@ExceptionHandler(Exception.class)

@ResponseBody public Object handleException( Exception e){ e.printStackTrace(); return new JsonResult(e); } }

其中 @ExceptionHandler 注解用于异常处理

1. 测试:

<http://localhost:8080/note/user/login.do?name=demo&password=123456>

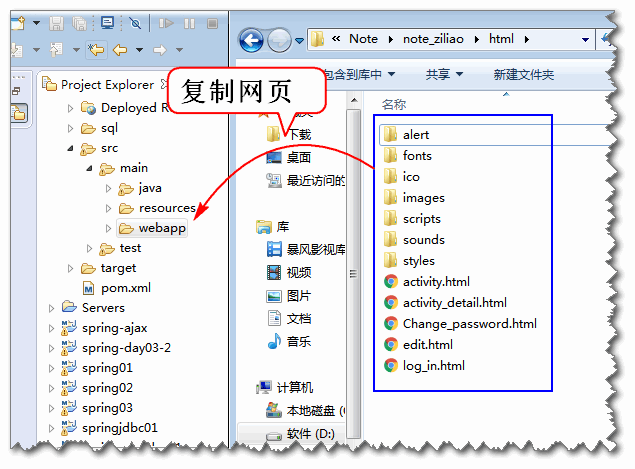
<http://localhost:8080/note/user/login.do>

http://localhost:8080/note/user/login.do?name=demo&password=123

全面测试正常和异常情况.

**5. 表现层脚本**

1. 将html页面复制到 webapp 文件夹:



1. 更新log\_in.html 添加脚本:

<meta charset="utf-8">

<link rel="stylesheet" href="styles/login.css"/>

<script type="text/javascript" src="scripts/jquery.min.js"></script>

<script type="text/javascript" src="scripts/login.js"></script>

1. 添加scripts/login.js文件
2. 在login.js文件添加事绑定脚本:

$(function(){ //console.log('Hello World!');

$('#login').click(loginAction); $('#count').blur(checkName);

$('#password').blur(checkPassword); });

1. 添加用户名和密码检查方法:

function checkName(){

var name = $('#count').val();

var rule = /^\w{4,10}$/;

if(! rule.test(name)){ $('#count').next().html('4~10个字符'); return false; }

$('#count').next().empty(); return true; }

function checkPassword(){ var password = $('#password').val(); var rule = /^\w{4,10}$/;

if(! rule.test(password)){ $('#password').next().html('4~10个字符'); return false; }

$('#password').next().empty(); return true; }

1. 添加登录按钮事件处理方法:

function loginAction(){ //console.log("loginAction");

//获取用户输入的用户名和密码

var name = $('#count').val();

var password = $('#password').val();

//检查 用户名和密码输入框 var n=checkName()+checkPassword();

if(n!=2){ return; } //data 对象中的属性名要与服务器控制器的参数 // 名一致!

login(name, password) var data = {"name":name, "password":password};

$.ajax({ url:'user/login.do', data:data, type:'post', dataType:'json', success: function(result){ console.log(result); if(result.state==0){ //登录成功! var user = result.data; console.log(user);

//跳转到 edit.html location.href='edit.html';

}else{ var msg = result.message; $('#count').next().html(msg); } },

error: function(e){ alert("通信失败!"); } }); }

1. 测试, Good Luck!