JavaWeb

Java web

1、基本概念

1.1、前言

web开发

web网页的意思，www.baidu.com

静态web

- html、css
- 提供给所有人看的数据始终不会发生变化

动态

- 淘宝、几乎是所有网站
- 提供给所有人看的数据始终不会发生变化，每个人在不同的时间，不同的地点看到的信息各不相同！
- 技术栈：Servlet、JSP，ASP，PHP

在JAVA中，动态web资源开发的技术统称为JavaWeb

1.2、web应用程序

web应用程序可以提供 浏览器访问的程序，

a.html，b.html....多个静态资源，这些资源能够被外界资源访问，对外界提供服务

你们能访问到的任何一个页面或者资源，都存在于这个世界的某个角落的计算机上

URL

这个统一的web资源会被放在统一个文件夹下，web应用程序--》Tomcat服务器

一个web应用由多个部分组成（静态web、动态web）

- html，css，js
- jsp，servlet
- java程序
- jar包
- 配置文件（Properties）

web应用程序编写完毕后，若想提供给外界访问，需要一个服务器管理；

1.3、静态web

*.html都是网页的后缀，如果服务器上一直存在这些东西我们就可以直接进行访问

静态web存在的缺点

- web页面无法动态更新，所有用户看到的都是同一个页面
    - 轮播图、点击特效：伪动态

    - javaScript实际开发中他用的最多
    - VBScript
- 他无法和数据库交互（数据无法持久化，用户无法交互）

1.4、动态web

页面会动态展示，“web的页面展示的效果因人而异”

缺点：

- 假如服务器的动态web资源出现了错误，我们需要重新编写我们的后台程序，重新发布；
    - 停机维护

优点：

- web页面可以动态更新，所有用户看到的都不是同一个页面
- 他可以和数据库交互（数据持久化：注册，商品信息，用户信息）

2、WEB服务器

2.1、技术服务器

ASP：

- 微软，国内最早流行的就是ASP；
- 在HTML中嵌入了VB的脚本，ASP+COM；
- 在ASP开发中，基本一个界面都有几千行的业务代码，页面极其混乱
- 维护成本高
- C#
- IIS

PHP

- PHP开发速度很快，功能很强大，跨平台，代码很简单（）
- 无法承载大访问量的情况（局限性）

JSP/Servlet

- sun公司主推的B/S（浏览器和服务器）架构
- C/S（客户端和服务端）
- 基于java语言的（所有的大公司，或者一些开源的组件，都是用java写的）
- 可以继承三高问题带来的影响（高并发，高性能，高可用）
- 语法像ASP，加强市场的强度

2.2、web服务器

服务器是一种被动的操作，用来处理一些客户的请求和给用户一些响应的信息

IIS

微软的：ASP。。。windows中自带的

Tomcat

面向百度编程，自行百度

下载tomcat：

1、安装or下载

2、了解配置文件及目录结构

3、这个东西的作用

3、Tomcat

3.1、安装

启动Tomcat

访问localhost:8080

可能遇到的问题

1、java环境变量没有配置

2、闪退问题：需要配置兼容性

3、乱码文件：配置文件中设置

3.3、配置

tomcat的默认端口号：8080

mysql：3306

http：80

https：443

高难度面试题

请你谈谈一个网站是如何进行访问的

1、输入一个域名

2、检查本机的hosts下有没有这个域名的映射；

- 1、有、返回对应的IP地址
- 2、没有、去DNS（全世界的域名都在这里管理）服务器找，找不到就返回找不到

3.4、发布一个web网站

将自己写的网站，放到服务器（Tomcat）中指定的web应用的文件夹（webapps）下，就可以访问了

网站应该有的结构

```
--webapps:Tomcat服务器的web目录
    --ROOT
    -kuangstudy：网站的目录名
    -WEB-INF
    -classes：java程序
    -lib：web应用所依赖的jar包
    -web.xml：网站配置文件
    -index.html：默认的首页
    -static
    -css
    -style.css
    -js
    -img
```

4、Http

4.1、什么是Http

超文本传输协议（Hyper Text Transfer Protocol，HTTP）是一个简单的请求-响应协议

- 文本：html，字符串，。。。
- 超文本：图片，音乐，视频，定位，地图
- 80

https：安全的

- 443

4.2、两个时代

http1.0

- http1.0：客户端可以与web服务器连接后，只能获得一个web资源，断开连接

http2.0

- http1.1：客户端可以与web服务器连接后，可以获得多个web资源。

4.3、Http请求

客户端发请求到服务器

服务器响应给客户端

- 请求 URL:

  https://baike.baidu.com/api/wikiui/guesslike?url=https%3A%2F%2Fbaike.baidu.com%2Fitem%2FHTTP&lemmaTitle=HTTP&eid=85312
- 请求方法: GET
- 状态代码: 200 OK
- 远程地址: 220.181.43.193:443
- 引用站点策略: unsafe-url

请求方式：Get、Post、Head、delete

get：能够携带的参数比较小，大小有限制，会在浏览器的URL地址栏显示数据内容，不安全但高效

post：能够携带的参数没有限制，大小没有限制，会在浏览器的URL地址栏显示数据内容，安全但不高效

响应状态码

200：请求响应成功

3XX：重定向

4XX：找不到资源

5XX：服务器代码错误 500 502：网关错误

常见面试题：

当你的浏览器中地址并回车 的一瞬间到页面能够展示回来，经历了什么

5、Maven

为什么要学习maven

1、在javaweb开发中，我们需要使用大量的jar包，我们手动去导入

2、如何能够让一个东西自动帮我们导入并配置这个jar包

由此maven诞生

5.1、maven项目架构管理工具

我们目前用来就是方便导入jar包

maven的核心思想：**约定大于配置**

有约束，不要去违反

Maven会规定好你该如何区编写我们的java代码，必须按照这个规范来

5.2、下载安装maven

下载完成后解压即可

5.3、配置环境变量

在系统环境变量中：

- M2_HOME maven目录下的bin目录
- MAVEN_HOME maven的目录
- 在系统的path位置：%MAVEN_HOME%\bin

测试maven是否安装成功

5.4、阿里云镜像

- 镜像mirrors进行镜像的配置
    - 作用：加速我们的下载
- 国内建议使用阿里云镜像下载

```
<mirrors>
     <mirror>
        <id>nexus-aliyun</id>
        <mirrorOf>*</mirrorOf>
        <name>Nexus aliyun</name>
        <url>http://maven.aliyun.com/nexus/content/groups/public</url>
     </mirror>
</mirrors>
```

5.5、本地仓库

远程仓库；

建立一个仓库：localRepository

5.6、在diea中使用maven

1、启动IDEA

2、创建一个maven项目

3、进行项目配置

4、观察maven仓库

5、idea中的maven配置

6、到这里maven在diea中的配置就完成了

5.7、创建一个普通的maven项目

集成tomcat

maven由于他的约定大于配置，我们之后可能遇到我们写的配置文件，无法被导出或者生效的问题，解决方案：

在build中配置resource，来防止我们资源导出失败的问题

5.8、解决遇到的问题

idea每次重复配置maven

在idea 中的全局默认配置中去配置

6、servlet

6.1、servlet简介

- servlet就是sun公司开发动态web的一门技术
- sun在这些API中体重一个接口叫做：Servlet，如果你想开发一个servlet程序，只需要完成两个小步骤：
    - 编写一个类，实现servlet接口
    - 把开发好的java类部署到web服务器中

把实现了servlet接口的Java程序叫做servlet

6.2、HelloServlet

Servlet接口在sun公司有两个默认的实现类：HttpServlet、GenericServlet

1、构建一个普通的maven项目，删掉里面的src目录，以后我们的学习就在这个项目里建立Moudel，这个空的工程就是maven主工程

2、关于maven父子工程的理解

- 父项目中会有
- 子项目中会有
- 父项目中的java子项目可以直接使用

3、maven环境的优化

- 1、修改web.xml文件为最新的
- 2、将maven的结构搭建完整

4、编写一个servlet

- 1、编写一个普通类
- 2、实现Servlet接口，这里我们直接继承HttpServlet

```
public class HelloServlet extends HttpServlet {

    //由于get或者post只是请求实现的不同方式，可以相互调用，业务逻辑都一样
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
//        ServletInputStream inputStream = req.getInputStream();

//        ServletOutputStream outputStream = resp.getOutputStream();
        PrintWriter writer = resp.getWriter();
        writer.print("hello servlet");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        super.doGet(req, resp);
    }
}
```

5、编写servlet的映射

为什么需要映射，我们写的是 java 程序，但是需要通过浏览器访问，而浏览器需要连接web服务器，所以我们需要在web服务中注册我们写的servlet，还需要给他一个浏览器能够访问的路径；

```
<servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>com.lwq.servlet.HelloServlet</servlet-class>
    </servlet>
    <!--Servlet的请求路径-->
    <servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
```

6、配置Tomcat

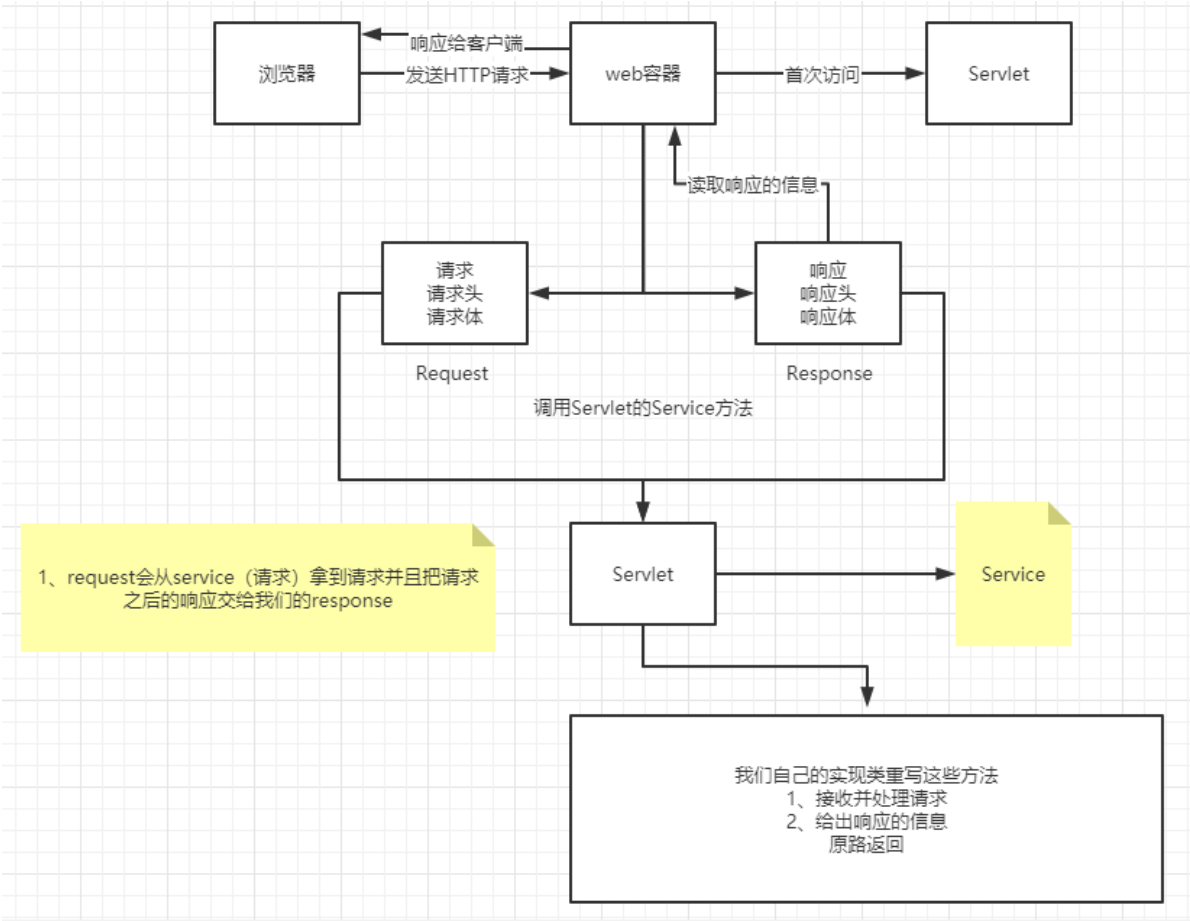
配置项目发布的路径

7、启动测试

启动问题：tomcat版本太高不符合maven依赖会报500错误，并不是代码问题，需要降低Tomcat版本

6.3、Servlet原理

Servlet是由web服务器调用，web服务器在收到浏览器的请求之后，会：



6.4、mapping问题

1、一个servlet可以指定一个映射路径

```
<!--注册Servlet-->
    <servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>com.lwq.servlet.HelloServlet</servlet-class>
    </servlet>
    <!--Servlet的请求路径-->
    <servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
```

2、一个servlet可以指定多个映射路径

```
<!--注册Servlet-->
    <servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>com.lwq.servlet.HelloServlet</servlet-class>
    </servlet>
    <!--Servlet的请求路径-->
    <servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
<servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello2</url-pattern>
```

```
    </servlet-mapping>
<servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello3</url-pattern>
    </servlet-mapping>
<servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello4</url-pattern>
    </servlet-mapping>
```

3、一个servlet可以指定通用映射路径

```
    <!--注册Servlet-->
    <servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>com.lwq.servlet.HelloServlet</servlet-class>
    </servlet>
    <!--Servlet的请求路径-->
    <servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello/*</url-pattern>
    </servlet-mapping>
```

4、指定一些后缀等。。

```
<!--注册Servlet-->
    <servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>com.lwq.servlet.HelloServlet</servlet-class>
    </servlet>
    <!--Servlet的请求路径-->
    <!--可以自定义后缀实现请求映射-->
    <!--*面前不能加项目的路径映射-->
    <servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>*.lwq</url-pattern>
    </servlet-mapping>
```

5、优先级问题

指定了固有的映射路径优先级最高，如果找不到就会走默认的处理请求

6.5、ServletContext

web容器在启动的时候，他会为每个web程序都创建一个对应的ServletContext对象，它代表了当前的web应用；

1、共享数据

我在这个Servlet中保存的数据可以在另一个servlet中拿到

```
 //放置数据的类
ServletContext context = this.getServletContext();
        String username = "刘";
        context.setAttribute("username",username);
//获得数据的类
        ServletContext context = this.getServletContext();
        String username = (String) context.getAttribute("username");
        resp.setContentType("text/html");
        resp.setCharacterEncoding("utf-8");
        resp.getWriter().print("名字"+username);
```

测试访问结果

2、获取初始化参数

```
<context-param>
        <param-name>url</param-name>
        <param-value>jdbc:mysql://localhost:3306</param-value>
    </context-param>
```

```
ServletContext context = this.getServletContext();

        String url = context.getInitParameter("url");

        resp.getWriter().print(url);
```

3、请求转发

```
ServletContext context = this.getServletContext();

        RequestDispatcher requestDispatcher =
context.getRequestDispatcher("/hello");
        requestDispatcher.forward(req,resp);
```

4、读取资源文件

properties

- 在java目录中新建properties
- 在resource目录下新建properties

发现都被打包到了同一个路径下：classes，我们称这个路径为classpath

思路：需要一个文件流；

```
InputStream is = this.getServletContext().getResourceAsStream("/WEB-
INF/classes/db.properties");
        Properties properties = new Properties();
        properties.load(is);
        String username = properties.getProperty("username");
        String pwd = properties.getProperty("password");
        resp.getWriter().print(username);
        resp.getWriter().print(pwd);
```

maven约定大于配置解决方案

```
<build>
        <resources>
            <resource>
                <directory>src/main/resources</directory>
                <includes>
                    <include>**/*.properties</include>
                    <include>**/*.xml</include>
                </includes>
            </resource>
            <resource>
                <directory>src/main/java</directory>
                <includes>
                    <include>**/*.properties</include>
                    <include>**/*.xml</include>
                </includes>
                <filtering>true</filtering>
            </resource>
        </resources>
    </build>
```

访问测试即可ok

6.6、HttpServletResponce

web服务器接收到客户端的http请求，针对这个请求，分别创建一个代表请求的HttpServletRequest对象，代表响应的一个HttpServletResponce；

- 如果要获取客户端请求过来的参数，找HttpServletRequest
- 如果要给客户端响应一些信息：找HttpServletResponce

1、简单分类

负责向浏览器发送数据的方法

```
ServletOutputStream getOutputStream() throws IOException;

PrintWriter getWriter() throws IOException;
```

负责向浏览器发送响应头的方法

```
void setCharacterEncoding(String var1);

void setContentLength(int var1);

void setContentLengthLong(long var1);

void setContentType(String var1);

void setBufferSize(int var1);

void setHeader(String var1, String var2);

void setIntHeader(String var1, int var2);
```

响应状态码

```
int SC_CONTINUE = 100;
int SC_SWITCHING_PROTOCOLS = 101;
int SC_OK = 200;
int SC_CREATED = 201;
int SC_ACCEPTED = 202;
int SC_NON_AUTHORITATIVE_INFORMATION = 203;
int SC_NO_CONTENT = 204;
int SC_RESET_CONTENT = 205;
int SC_PARTIAL_CONTENT = 206;
int SC_MULTIPLE_CHOICES = 300;
int SC_MOVED_PERMANENTLY = 301;
int SC_MOVED_TEMPORARILY = 302;
int SC_FOUND = 302;
int SC_SEE_OTHER = 303;
int SC_NOT_MODIFIED = 304;
int SC_USE_PROXY = 305;
int SC_TEMPORARY_REDIRECT = 307;
int SC_BAD_REQUEST = 400;
int SC_UNAUTHORIZED = 401;
int SC_PAYMENT_REQUIRED = 402;
int SC_FORBIDDEN = 403;
int SC_NOT_FOUND = 404;
int SC_METHOD_NOT_ALLOWED = 405;
int SC_NOT_ACCEPTABLE = 406;
int SC_PROXY_AUTHENTICATION_REQUIRED = 407;
int SC_REQUEST_TIMEOUT = 408;
int SC_CONFLICT = 409;
int SC_GONE = 410;
int SC_LENGTH_REQUIRED = 411;
int SC_PRECONDITION_FAILED = 412;
int SC_REQUEST_ENTITY_TOO_LARGE = 413;
int SC_REQUEST_URI_TOO_LONG = 414;
int SC_UNSUPPORTED_MEDIA_TYPE = 415;
int SC_REQUESTED_RANGE_NOT_SATISFIABLE = 416;
int SC_EXPECTATION_FAILED = 417;
int SC_INTERNAL_SERVER_ERROR = 500;
int SC_NOT_IMPLEMENTED = 501;
int SC_BAD_GATEWAY = 502;
int SC_SERVICE_UNAVAILABLE = 503;
int SC_GATEWAY_TIMEOUT = 504;
int SC_HTTP_VERSION_NOT_SUPPORTED = 505;
```

2、常见应用

1、向浏览器输出信息

2、下载文件

1. 要获取下载文件的路径
2. 下载的文件名是啥
3. 设置想办法让浏览器能够支持下载我们需要的东西
4. 获取下载文件的输入流
5. 创建缓冲区
6. 获取outputstream对象
7. 将fileoutputstream流写入到buffer缓冲区

8. 将缓冲区的数据输出到客户端

```
//        1. 要获取下载文件的路径
        String realPath =
"E:\\workspace\\javaweb\\servlet\\target\\classes\\1630458354(1).jpg";
//        2. 下载的文件名是啥
        String filename = realPath.substring(realPath.lastIndexOf("\\") + 1);
//        3. 设置想办法让浏览器能够支持下载我们需要的东西
        resp.setHeader("Content-Disposition","attachment;filename="+
URLEncoder.encode(filename,"utf-8"));
//        4. 获取下载文件的输入流
        FileInputStream in = new FileInputStream(realPath);
//        5. 创建缓冲区
        int len=0;
        byte[] buffer = new byte[1024];
//        6. 获取outputstream对象
        ServletOutputStream out = resp.getOutputStream();
//        7. 将fileoutputstream流写入到buffer缓冲区
        while ((len=in.read(buffer))!=-1){
            out.write(buffer,0,len);
        }
//        8. 将缓冲区的数据输出到客户端
        out.close();
        in.close();
```

3、验证码功能

验证怎么来

- 前端实现
- 后端实现，需要用到 java 的图片类，生成一个图片

```
        //如何让浏览器自动刷新一次
        resp.setHeader("refresh","5");
        //在内存中创建一个图片
        BufferedImage image = new
BufferedImage(80,20,BufferedImage.TYPE_INT_RGB);
        //得到图片
        Graphics2D g = (Graphics2D) image.getGraphics();
        //设置图片的背景颜色
        g.setColor(Color.white);
        g.fillRect(0,0,80,20);
        //给图片写数据
        g.setColor(Color.cyan);
        g.setFont(new Font(null,Font.BOLD,20));
        g.drawString(makeNum(),0,20);

        //告诉浏览器这个请求用图片的方式打开
        resp.setContentType("image/jpeg");
        //网站存在缓存，不让浏览器缓存
        resp.setDateHeader("expires",-1);
        resp.setHeader("Cache-Control","no-cache");
        resp.setHeader("Pragma","no-cache");

        //把图片写给浏览器
        ImageIO.write(image, "jpg", resp.getOutputStream());
```

```
    }

    //创建随机数
    private String makeNum(){
        Random random = new Random();
        String s = random.nextInt(9999999)+" ";
        StringBuffer stringBuffer = new StringBuffer();
        for (int i = 0; i < 7-s.length(); i++) {
            stringBuffer.append("0");
        }
        String s1 = stringBuffer.toString() + s;

        return s1;
    }
```

4、实现重定向

一个web资源收到客户端请求，他会通知客户端去访问另外一个web资源，这个过程叫做重定向

常见场景：

用户登录

```
void sendRedirect(String var1) throws IOException;
```

测试

```
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
//        resp.setHeader("Location","/hello");
//        resp.setStatus(HttpServletResponse.SC_MOVED_TEMPORARILY);

        resp.sendRedirect("/servlet_war_exploded/hello");
    }
```

面试题：请你聊聊重定向和转发的区别？

相同点：

- 页面都会跳转

不同点：

- 请求转发的时候url不会发生变化
- 重定向的时候url会发生改变

6.7、HttpServletRequest

代表客户端的请求，用户通过HTTP访问服务器；http请求中的所有信息会被封装到httpservletrequest，通过这个httpservletrequest的方法，获得客户端的所有信息

1、获取前端传递的参数并且请求转发

```
req.setCharacterEncoding("utf-8");
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        String[] hobbies = req.getParameterValues("hobby");

        //输出中文乱码问题
        System.out.println(username);
        System.out.println(password);
        System.out.println(Arrays.toString(hobbies));


        req.getRequestDispatcher("/success.jsp").forward(req,resp);
```

小结：

如果使用的是请求转发则不用写contentPath，如果是重定向则需要使用

7、Cookie、Session

7.1、会话

会话：用户打开一个浏览器，点击了很多超链接，访问多个web资源，关闭浏览器，这个过程称之为会话

有状态会话：一个同学来过教室，下次再来教室，我们知道这个同学，曾经来过，称之为有状态会话

一个网站，怎么证明你来过?

客户端 服务端

1. 服务端给客户端一个信件，客户端下次访问服务端带上信件就可以了cookie
2. 服务器登记你来过了，下次你来的时候我来匹配你session

7.2、保存会话的两种技术

cookie

- 客户端技术（响应，请求）

session

- 服务器技术，利用这个技术，可以保存用户的会话信息，我们可以把信息或者数据放在Session中

常见实例：网站登陆之后，你下次进入同样的网站，下次会自动登陆

7.3、cookie

```
//保存用户上一次访问的时间
public class LoginServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        //服务器，告诉你来的时间，把这个时间封装成为一个新建，你下次带来，我就知道你来了

        //解决中文乱码问题
```

```
        req.setCharacterEncoding("utf-8");
        resp.setCharacterEncoding("utf-8");
        resp.setContentType("text/html;charset=UTF-8");

        PrintWriter out = resp.getWriter();

        //cookie服务器从客户端获取
        Cookie[] cookies = req.getCookies();//这里返回数组说明数组可能返回多个

        //判断cookie是否存在
        if (cookies!=null){
            //如果存在怎么办
            out.print("你上一次来的时间是: ");
            for (Cookie cookie : cookies) {
                //获取cookie 的名字
                if (cookie.getName().equals("lastTime")){
                    //获取cookie中的值
                    Long l = Long.parseLong(cookie.getValue());
                    Date date = new Date(l);
                    out.write(date.toLocaleString());
                }
            }
        }else {
            out.print("第一次来");
        }

        //服务器给客户端响应一个cookie
        Cookie cookie = new Cookie("lastTime", System.currentTimeMillis() + "");
        cookie.setMaxAge(24*60*60);
        resp.addCookie(cookie);
    }
```

一个网站cookie是否存在上限

- 一个cookie只能保存一个信息
- 一个web站点可以给浏览器发送多个cookie，最多存放20个cookie
- cookie大小限制4kb
- 300个cookie浏览器上限

删除cookie

- 不设置有效期，关闭浏览器，自动失效；
- 设置有效期时间为0；

7.4、session(重点)

什么是session：

- 服务器会给每一个用户（浏览器）创建一个session 对象
- 一个session独占一个浏览器，只要浏览器没有关闭这个session就存在
- 用户登陆之后整个网站他都可以访问--》保存用户的信息，保存购物车

session和cookie的区别

- cookie是把用户的数据写给用户的浏览器，浏览器保存（可以保存多个）
- session是把用户的数据写到用户独占的session中，服务器端保存（保存重要的信息）
- session对象由服务器创建

使用场景：保存一个用户的登陆信息，购物车信息，在整个网站中经常会使用的数据，我们将他保存在session中

```
req.setCharacterEncoding("utf-8");
        resp.setCharacterEncoding("utf-8");
        resp.setContentType("text/html;charset=UTF-8");
        //得到session
        HttpSession session = req.getSession();
        PrintWriter out = resp.getWriter();
        session.setAttribute("name","秦江");
        //获取session的id
        String id = session.getId();
        out.write(id);

        Object name1 = session.getValue("name");
        out.write(" "+name1+" ");
        boolean aNew = session.isNew();
        out.write(aNew+"");
        //session 创建的时候做了什么
//        Cookie cookie = new Cookie("...", id);
//        resp.addCookie(cookie);
    }
```

会话自动过期在web.xml中

```
<session-config>
        <session-timeout>15</session-timeout>
    </session-config>
```

8、JSP

8.1、什么是jsp

java server pages：java服务器端页面，也和servlet一样的，用于动态web技术

最大的特点

- 写jsp就像在写html

区别

- html只会给用户提供静态的数据
- jsp页面中可以嵌入java代码，为用户提供动态数据

8.2、JSP原理

思路：JSP到底怎么执行的

代码层面没有任何问题

- 服务器内部工作
- tomcat中有一个work目录
- idea中使用tomcat的会在idea的tomcat中生成一个work目录

浏览器向服务器发送请求，不管访问什么资源，其实都是在访问Servlet

jsp本质就是一个servlet

- 在JSP页面中只要是java代码就会原封不动的输出
- 如果是HTML代码，就会被转成out.write输出到前端

8.3、JSP基础语法

任何语言都有自己的语法，java中有jsp作为Java技术的一种应用，它拥有一些自己扩充的语法（了解，知道即可），java所有的语法都支持

JSP表达式

```
<%=new java.util.Date()%>
```

脚本片段的实现

```
<%
   int sum = 0;
   for (int i = 0; i <= 100; i++) {
     sum+=i;
   }
   out.println(sum);
 %>
```

JSP声明

```
<%!
    代码
    %>
```

JSP声明：会被编译到JSP生成JAVA的类中！其他的就会被生成到JavaServeice方法中

在JSP中，嵌入java代码即可

8.4、JSP指令

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
```

8.5、9大内置对象

- PageContext 存东西
- Request 存东西
- Response
- Session 存东西
- Application【ServletContext】存东西
- config【ServletConfig】
- out
- page
- exception

```
pageContext.setAttribute("name1","秦江1");
 application.setAttribute("name2","秦江2");
 session.setAttribute("name3","秦江3");
 request.setAttribute("name4","秦江4");
```

request：客户端向服务器发送请求，产生的数据，用户看完就没用了，比如：新闻，用户看完没用的

session：客户端向服务器发送请求，产生的数据，用户用完一会儿还有用，比如：购物车

application：客户端向服务器发送请求，产生的数据，一个用户用完了，其他的用户还能使用，比如：聊天数据

8.6、JSP 标签、JSTL标签、EL表达式

导包

```
<dependency>
            <groupId>javax.servlet.jsp.jstl</groupId>
            <artifactId>jstl-api</artifactId>
            <version>1.2-rev-1</version>
        </dependency>
        <dependency>
            <groupId>taglibs</groupId>
            <artifactId>standard</artifactId>
            <version>1.1.2</version>
```

EL表达式：

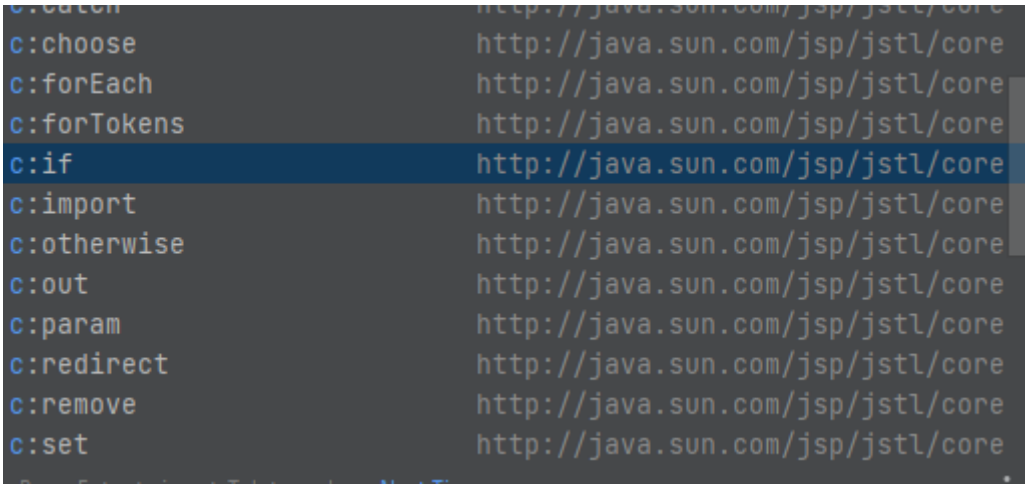
- **获取数据**
- **执行运算**
- **获取web开发的常用对象**

  JSP 标签

```
<jsp:forward page="success.jsp"></jsp:forward>
```

JSTL标签

核心标签（掌握部分）



记得导入<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

格式化标签

SQL标签

XML标签

```
<form action="coreif.jsp" method="get">
    <%--EL表达式获取表单中的数据${param.参数名}--%>
    <input type="text" name="username" value="${param.username}">
    <input type="submit" value="登陆">
  </form>
  <c:if test="${param.username=='lwq'}" var="isAdmin">
    <c:out value="管理员欢迎你"/>
  </c:if>
  <c:out value="${isAdmin}"/>
```

9、JavaBean

实体类

javabean有特定的写法：

必须要有一个无参构造

属性必须私有化

必须有对应的get、set方法

一般用来和数据库的字段做映射ORM（对象关系映射）

- 表--》类
- 字段--》属性
- 行记录--》对象

id	name	age	address
1	秦江1号	3	西安
2	秦江2号	18	西安
3	秦江3号	100	西安

```
class People{
    private int id;
    private String name;
    private int age;
    private String address;
}

class A{
    new People(1,"qinjiang",3,"xian")
}
```

10、MVC三层架构

什么是mvc

Model View Controller 模型、视图、控制器

Model

- 业务处理：业务逻辑（service）
- 数据持久层：CRUD（Dao）

View

- 展示数据
- 提供链接发起Servlet请求

Controller（Servlet）

- 接收用户的请求（req：请求参数、Session信息。。。）
- 交给业务层处理对应的代码
- 控制视图跳转

登陆--》接收用户的请求--》处理用户的请求（获取用户登陆的参数、username、password）--》交给业务层处理登陆业务（判断用户名密码是否正确，事务）--》Dao层查询用户名和密码是否数据--》数据库

11、过滤器Filter

用来过滤网站的数据

- 处理中文乱码
- 登录验证

Filter开发步骤

1、导包

2、编写过滤器

```
public class CharacterEncoding implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        System.out.println("characterEncoding已经初始化");
    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain) throws IOException, ServletException {
        servletRequest.setCharacterEncoding("utf-8");
        servletResponse.setCharacterEncoding("utf-8");
        servletResponse.setContentType("text/html;charset=UTF-8");
        System.out.println("Filter执行前");
        filterChain.doFilter(servletRequest,servletResponse);//让请求继续走
        System.out.println("Filter执行后");
    }

    @Override
    public void destroy() {
        System.out.println("characterEncoding已经销毁");
    }
}
```

3、在web.xml中配置过滤器

```
<filter>
        <filter-name>CharacterEncoding</filter-name>
        <filter-class>com.lwq.filter.CharacterEncoding</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>CharacterEncoding</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

12、监听器

实现一个监听器的接口；（有N种）

1、实现监听器的接口

```
public class Listener implements HttpSessionListener {
    @Override
    public void sessionCreated(HttpSessionEvent se) {
        ServletContext ctx = se.getSession().getServletContext();
        Integer onlineCount = (Integer)ctx.getAttribute("OnlineCount");

        if (onlineCount==null){
            onlineCount=new Integer(1);
        }else {
            int count = onlineCount.intValue();
            onlineCount=new Integer(count+1);
        }
        ctx.setAttribute("OnlineCount",onlineCount);
    }

    @Override
    public void sessionDestroyed(HttpSessionEvent se) {
        ServletContext ctx = se.getSession().getServletContext();
        Integer onlineCount = (Integer) ctx.getAttribute("OnlineCount");

        if (onlineCount==null){
            onlineCount=new Integer(0);
        }else {
            int count = onlineCount.intValue();
            onlineCount=new Integer(count-1);
        }
        ctx.setAttribute("OnlineCount",onlineCount);
    }
}
```

2、在web.xml中注册监听器

```
<listener>
        <listener-class>com.lwq.filter.Listener</listener-class>
    </listener>
```

3、实现监听器

```
<body>
  <h1>当前在线用户<%=config.getServletContext().getAttribute("OnlineCount")%></h1>
  </body>
```

13、过滤器、监听器常见应用

监听器：GUI编程中经常使用

用户登陆之后才能进入首页，用户注销之后就不能进入首页了

```
String username = req.getParameter("username");
        String password = req.getParameter("password");
        if ((username.equals("lwq"))&&(password.equals("123456"))){

 req.getSession().setAttribute("USER_SESSION",req.getSession().getId());
            resp.sendRedirect("/servlet_war_exploded/success.jsp");
        }else {
            resp.getWriter().write("用户名或密码错误");
            resp.sendRedirect("/servlet_war_exploded/error.jsp");
        }
    }
```

过滤器中实现其他页面拦截

```
public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain) throws IOException, ServletException {
        servletRequest.setCharacterEncoding("utf-8");
        servletResponse.setCharacterEncoding("utf-8");
        servletResponse.setContentType("text/html;charset=UTF-8");
        HttpServletResponse resp = (HttpServletResponse) servletResponse;
        HttpServletRequest req = (HttpServletRequest) servletRequest;


        if (req.getSession().getAttribute("USER_SESSION")==null){
            resp.sendRedirect("/servlet_war_exploded/error.jsp");
        }


        System.out.println("Filter执行前");
        filterChain.doFilter(servletRequest,servletResponse);//让请求继续走
        System.out.println("Filter执行后");
    }
```

14、JDBC

什么是jdbc：java连接数据库 （统一驱动）

mysql：mysql.Driver

oracle:oracal.Driver

需要jar包支持：

- java.sql
- javax.sql

- mysql-connecter-java...连接驱动，必须导入

1、创建数据库

2、导包

3、idea连接数据库

4、配置JDBC连接

- 创建JDBC源
- 加载驱动
- 链接数据库
- 编写SQL
- 执行SQL
- 关闭连接

```
        //配置信息
        String url = "jdbc:mysql://localhost:3306/people?
useUnicode=true&characterEncoding=utf-8";
        String username = "root";
        String password = "root";

        //加载驱动
        Class.forName("com.mysql.cj.jdbc.Driver");
        //连接数据库
        Connection connection = DriverManager.getConnection(url, username,
password);
        //向数据库发送SQL的对象statment：CRUD
        Statement statement = connection.createStatement();
        //编写SQL
        String sql = "select * from people";
        ResultSet rs = statement.executeQuery(sql);

        while (rs.next()){
            System.out.println("id"+rs.getObject("id"));
            System.out.println("name"+rs.getObject("name"));
            System.out.println("password"+rs.getObject("password"));
            System.out.println("address"+rs.getObject("address"));
        }

        //关闭连接以免浪费资源
        rs.close();
        statement.close();
        connection.close();
    }
```

事务

要么都成功要么都失败

ACID原则保证数据的安全

- 开启事务
- 关闭事务
- 事务提交
- 事务回滚

```
 String url="jdbc:mysql://localhost:3306/foods?
useUnicode=true&characterEncoding=UTF-8";
        String username = "root";
        String password = "root";
        Connection connection = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            connection = DriverManager.getConnection(url,username,password);
            connection.setAutoCommit(false);


            String sql="update people set money=money-100 where name
='qinjiang1'";
            connection.prepareStatement(sql).executeUpdate();

//            int i=1/0;
            String sql2="update people set money=money+100 where name
='qinjiang2'";
            connection.prepareStatement(sql2).executeUpdate();

            connection.commit();
            System.out.println("success");
        }catch (Exception e){
            connection.rollback();
            e.printStackTrace();
        }finally {
            connection.close();
        }
```

SMBMS

项目准备工作

1、创建一个新项目

2、配置Tomcat

3、测试能否正常运行

4、导包

5、创建项目包结构

6、编写实体类：ORM映射：表-类映射

7、编写基础公共类

1.
```
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/smbms?
useUnicode=true&characterEncoding=utf8&serverTimezone=GMT%2B8&useSSL=false
user=root
password=root
```

2. 编写数据库公共类

```
package com.smbms.dao;
```

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.sql.*;
import java.util.Properties;

public class BassDao {
    //读取配置文件
    public static String driver;
    public static String url;
    public static String username;
    public static String password;
    //静态代码块，类加载的时候就要去读
    static {
        //通过类加载器的方式去读
        InputStream is =
BassDao.class.getClassLoader().getResourceAsStream("db.properties");
        Properties properties = new Properties();
        try {
            properties.load(is);
        } catch (IOException e) {
            e.printStackTrace();
        }
        driver=properties.getProperty("driver");
        url=properties.getProperty("url");
        username=properties.getProperty("user");
        password=properties.getProperty("password");
    }
    //获取数据库连接
    public static Connection getConnection() throws Exception {
        Class.forName(driver);
        Connection connection =
DriverManager.getConnection(url,username,password);
        return connection;
    }
    //编写查询公共类
    public static ResultSet excute(Connection connection,String sql,Object[]
params,ResultSet rs) throws SQLException {
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        for (int i = 0; i < params.length; i++) {
            preparedStatement.setObject(i+1,params[i]);
        }
        rs= preparedStatement.executeQuery();
        return rs;
    }
    //编写增删改公共类
    public static int excute(Connection connection,String sql,Object[] params)
throws SQLException {
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        for (int i = 0; i < params.length; i++) {
            preparedStatement.setObject(i+1,params[i]);
        }
        int update = preparedStatement.executeUpdate();
        return update;
    }
    public static boolean ResourceClose(Connection connection,PreparedStatement
preparedStatement,ResultSet rs) throws Exception {
```

```
        boolean flag = true;
        if (rs!=null){
            rs.close();
            rs=null;
        }else
            flag=false;

        if (preparedStatement!=null){
            preparedStatement.close();
            preparedStatement=null;
        }else
            flag=false;

        if (connection!=null){
            connection.close();
            connection=null;
        }else
            flag=false;

        return flag;
    }
}
```

3. 编写过滤器

```
 public void doFilter(ServletRequest req, ServletResponse resp, FilterChain
filterChain) throws IOException, ServletException {
        req.setCharacterEncoding("utf-8");
        resp.setCharacterEncoding("utf-8");
        resp.setContentType("text/html;charset=UTF-8");

        filterChain.doFilter(req,resp);
    }
```

8、导入静态资源

登陆功能实现

1、编写前端页面

2、设置首页

```
<welcome-file-list>
        <welcome-file>login.jsp</welcome-file>
    </welcome-file-list>
```

3、编写dao层用户登陆的接口

```
public interface UserDao {
    //得到要登陆的用户
    public User getLoginUser(Connection connection,String userCode) throws
SQLException;

}
```

4、编写dao接口的实现类

```
public class UserDaoImpl implements UserDao{
    @Override
    public User getLoginUser(Connection connection, String userCode) throws
SQLException {
        PreparedStatement preparedStatement=null;
        User user=null;
        ResultSet resultSet = null;

        if (connection!=null){
            String sql="select * from smbms_user where userCode=?";
            Object[] params={userCode};
            resultSet = BaseDao.excute(connection, preparedStatement, params,
sql, resultSet);
            if (resultSet.next()){
                user=new User();
                user.setId(resultSet.getInt("id"));
                user.setUserCode(resultSet.getString("userCode"));
                user.setUserName(resultSet.getString("userName"));
                user.setUserPassword(resultSet.getString("userPassword"));
                user.setGender(resultSet.getInt("gender"));
                user.setBirthday(resultSet.getDate("birthday"));
                user.setPhone(resultSet.getString("phone"));
                user.setAddress(resultSet.getString("address"));
                user.setUserRole(resultSet.getInt("userRole"));
                user.setCreatedBy(resultSet.getInt("createdBy"));
                user.setCreationDate(resultSet.getTimestamp("creationDate"));
                user.setModifyBy(resultSet.getInt("modifyBy"));
                user.setModifyDate(resultSet.getTimestamp("modifyDate"));
            }
        }
        BaseDao.closeResource(null,preparedStatement,resultSet);
        return user;
    }

}
```

5、业务层接口

```
public interface UserService {
    public User login(String userCode, String password) throws SQLException;
}
```

6、业务层的实现类

```
public class UserServiceImpl implements UserService{

    //业务层都会调dao层
    private UserDao userDao;
    public UserServiceImpl() {
        userDao=new UserDaoImpl();
    }
    @Override
    public User login(String userCode, String password) throws SQLException {
```

```
        Connection connection = null;
        User user=null;
        try {
            user = userDao.getLoginUser(connection, userCode);
        }catch (Exception e){
            e.printStackTrace();
        }finally {
            BaseDao.closeResource(connection,null,null);
        }
        return  user;
    }
}
```

7、编写Servlet

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //获取用户名和密码
        String userCode = req.getParameter("userCode");
        String password = req.getParameter("password");

        //和数据库的用户名密码进行对比,调业务层
        UserService userService = new UserServiceImpl();
        User user = userService.login(userCode, password);

        if (user!=null){
            //将用户的信息放在session中
            req.getSession().setAttribute(Constants.USER_SESSION,user);
            resp.sendRedirect("jsp/frame.jsp");
        }else {
            req.setAttribute("error","用户名或者密码错误");
            req.getRequestDispatcher("/login.jsp").forward(req,resp);
        }

    }
```

8、注册Servlet

```
    <servlet>
        <servlet-name>login</servlet-name>
        <servlet-class>com.smbms.servlet.UserServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>login</servlet-name>
        <url-pattern>/login.do</url-pattern>
    </servlet-mapping>
```

9、测试访问

登录功能优化

注销功能：

思路：移除session返回登陆界面

编写注销servlet

```
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.getSession().removeAttribute(Constants.USER_SESSION);
        resp.sendRedirect("/SMBMS_war_exploded/login.jsp");
    }
```

注册

```
<servlet>
    <servlet-name>logout</servlet-name>
    <servlet-class>com.smbms.servlet.LogoutServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>logout</servlet-name>
    <url-pattern>/jsp/logout.do</url-pattern>
</servlet-mapping>
```

过滤器优化

```
public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain) throws IOException, ServletException {
    HttpServletRequest req = (HttpServletRequest)servletRequest;
    HttpServletResponse resp = (HttpServletResponse)servletResponse;

    if (req.getSession().getAttribute(Constants.USER_SESSION)==null){
        resp.sendRedirect("/SMBMS_war_exploded/error.jsp");
    }

    filterChain.doFilter(servletRequest,servletResponse);
}
```

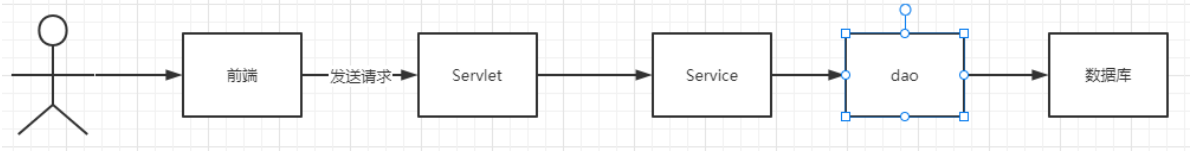
注册

```
<filter>
    <filter-name>LoginFilter</filter-name>
    <filter-class>com.smbms.filter.LoginFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>LoginFilter</filter-name>
    <url-pattern>/jsp/*</url-pattern>
</filter-mapping>
```

密码修改

1、导入前端素材

2、写项目建议从底层向上写



3、UserDao

```
public int updatePwd(Connection connection,int id,String password) throws
SQLException;
```

4、UserDaoImpl

```
public class UserDaoImpl implements UserDao{
    @Override
    public User getLoginUser(Connection connection, String userCode) throws
SQLException {
        PreparedStatement preparedStatement=null;
        User user=null;
        ResultSet resultSet = null;

        if (connection!=null){
            String sql="select * from smbms_user where userCode=?";
            Object[] params={userCode};
            resultSet = BaseDao.excute(connection, preparedStatement, params,
sql, resultSet);
            if (resultSet.next()){
                user=new User();
                user.setId(resultSet.getInt("id"));
                user.setUserCode(resultSet.getString("userCode"));
                user.setUserName(resultSet.getString("userName"));
                user.setUserPassword(resultSet.getString("userPassword"));
                user.setGender(resultSet.getInt("gender"));
                user.setBirthday(resultSet.getDate("birthday"));
                user.setPhone(resultSet.getString("phone"));
                user.setAddress(resultSet.getString("address"));
                user.setUserRole(resultSet.getInt("userRole"));
                user.setCreatedBy(resultSet.getInt("createdBy"));
                user.setCreationDate(resultSet.getTimestamp("creationDate"));
                user.setModifyBy(resultSet.getInt("modifyBy"));
                user.setModifyDate(resultSet.getTimestamp("modifyDate"));
            }
        }
        BaseDao.closeResource(null,preparedStatement,resultSet);
        return user;
    }

    @Override
    public int updatePwd(Connection connection, int id, String password) throws
SQLException {
        String sql="update smbms_user set userPassword = ? where id = ?";
        PreparedStatement pstm = null;
        Object[] params= {id,password};
        int excute = 0;
        if (connection!=null){
            excute = BaseDao.excute(connection, pstm, params, sql);
            BaseDao.closeResource(null,pstm,null);
        }
        return excute;
    }

}
```

5、UserService

```
//根据用户id修改密码
public boolean updatePwd(int id ,String password) throws Exception;
```

6、UserServiceImpl

```
@Override
public boolean updatePwd(int id, String password) throws Exception {
    Connection connection = null;
    connection = BaseDao.getConnection();
    boolean flag = false;
    if (userDao.updatePwd(connection,id,password)>0){
        flag=true;
    }
    BaseDao.closeResource(connection,null,null);
    return flag;
}
```

7、Servlet复用

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
    String method = req.getParameter("method");
    if (method!=null && method.equals("savepwd")){
        this.updatePwd(req,resp);
    }
}

@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
    doGet(req, resp);
}

public void updatePwd(HttpServletRequest req, HttpServletResponse resp) throws
Exception {
    Object o = req.getSession().getAttribute(Constants.USER_SESSION);
    String newpassword =req.getParameter("newpassword");
    boolean flag = false;
    if (o!=null && newpassword!=null){
        UserService userService = new UserServiceImpl();
        flag = userService.updatePwd(((User) o).getId(), newpassword);
        if (flag){
            req.setAttribute("message","修改密码成功，请退出用新密码登陆");
            //修改密码成功移除session
            req.getSession().removeAttribute(Constants.USER_SESSION);
        }else {
            req.setAttribute("message","密码修改失败");
        }
    }else {
        req.setAttribute("message","密码有问题");
    }
    req.getRequestDispatcher("pwdmodify.jsp").forward(req,resp);
}
```

8、测试

优化密码修改使用Ajax

1、阿里巴巴fastjson

2、

```
//验证旧密码
public void pwdModify(HttpServletRequest req, HttpServletResponse resp){
    Object o = req.getSession().getAttribute(Constants.USER_SESSION);
    String oldpassword = req.getParameter("oldpassword");
    Map<String, String> resultMap = new HashMap<String,String>();
    if (o==null){
        resultMap.put("result","sessionerror");
    }else if (StringUtils.isNullOrEmpty(oldpassword)){
        resultMap.put("result","error");
    }else{
        String userPassword = ((User) o).getUserPassword();
        if (oldpassword.equals(userPassword)){
            resultMap.put("result","true");
        }else {
            resultMap.put("result","false");
        }
    }

    try {
        resp.setContentType("application/json");
        PrintWriter writer = resp.getWriter();
        writer.write(JSON.toJSONString(resultMap));
        writer.flush();
        writer.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
```

用户管理实现

1、导入分页的工具类

2、用户列表页面导入

1、获取用户数量

1、UserDao

```
public int getUserCount(Connection connection,String username,int userRole)
throws SQLException;
```

2、UserDaoImpl

```
@Override
public int getUserCount(Connection connection, String username, int userRole)
throws SQLException {
    PreparedStatement pstm=null;
    ResultSet rs = null;
    int count = 0;
    ArrayList list = new ArrayList();
```

```
    if (connection!=null){
        StringBuffer sql = new StringBuffer();
        sql.append("select count(1) as count from smbms_user u,smbms_role r
where u.userRole=r.id");
        if (username!=null){
            sql.append(" and u.username like ?");
            list.add("%"+username+"%");
        }
        if (userRole>0){
            sql.append(" and u.userRole = ?");
            list.add(userRole);
        }
        Object[] params = list.toArray();
        rs = BaseDao.excute(connection, pstm, params, sql.toString(), rs);
        if (rs.next()){
            count=rs.getInt("count");
        }
        BaseDao.closeResource(null,pstm,rs);
    }
    return count;
}
```

3、UserService

```
public int getUserCount(String username,int userRole) throws Exception;
```

4、UserServiceImpl

```
public int getUserCount(String username, int userRole) throws Exception {
    Connection connection =null;
    int count = 0;
    connection = BaseDao.getConnection();
    count = userDao.getUserCount(connection, username, userRole);

    BaseDao.closeResource(connection,null,null);

    return count;
}
```

2、获取用户列表

1、UserDao

```
public List<User> getUserList(Connection connection, String userName, int
userRole, int currentPageNo, int pageSize)throws Exception;
```

2、UserDaoImpl

```
public List<User> getUserList(Connection connection, String userName, int
userRole, int currentPageNo, int pageSize) throws Exception {
    PreparedStatement pstm = null;
    ResultSet rs=null;
    User user=null;

    List<User> userList = new ArrayList<User>();
```

```
    StringBuffer sql = new StringBuffer();
    sql.append("select u.*,r.roleName as userRoleName from smbms_user
u,smbms_role r where u.userRole = r.id");
    ArrayList<Object> list = new ArrayList<>();
    if (userName!=null){
        sql.append(" and u.username like ?");
        list.add("%"+userName+"%");
    }if(userRole > 0){
        sql.append(" and u.userRole = ?");
        list.add(userRole);
    }sql.append(" order by creationDate DESC limit ?,?");
    currentPageNo = (currentPageNo-1)*pageSize;
    list.add(currentPageNo);
    list.add(pageSize);

    Object[] params = list.toArray();
    rs=BaseDao.excute(connection,pstm,params,sql.toString(),rs);
    if (rs.next()){
        user=new User();
        user.setId(rs.getInt("id"));
        user.setUserCode(rs.getString("userCode"));
        user.setUserName(rs.getString("userName"));
        user.setGender(rs.getInt("gender"));
        user.setBirthday(rs.getDate("birthday"));
        user.setPhone(rs.getString("phone"));
        user.setUserRole(rs.getInt("userRole"));
        user.setUserRoleName(rs.getString("userRoleName"));
        userList.add(user);
    }
    BaseDao.closeResource(null,pstm,rs);
    return userList;
}
```

3、UserService

```
public List<User> getUserList(String queryUserName, int queryUserRole, int
currentPageNo, int pageSize) throws Exception;
```

4、UserServiceImpl

```
public List<User> getUserList(String queryUserName, int queryUserRole, int
currentPageNo, int pageSize) throws Exception {
    Connection connection=null;
    List<User> userList = null;
    connection = BaseDao.getConnection();
    userList = userDao.getUserList(connection, queryUserName, queryUserRole,
currentPageNo, pageSize);
    BaseDao.closeResource(connection,null,null);
    return userList;
}
```