目录

[一、开发工具： 2](#_Toc8652426)

[1、 Eclipse 2](#_Toc8652427)

[2、 相关语言：java、jsp、Servlet、HTML、js、jQuery、CSS等 2](#_Toc8652428)

[搭建开发环境：（Java 1.8、Tomcat8.5、Eclipse+Maven） 2](#_Toc8652429)

[二、部署工具：Tomcat。 8](#_Toc8652430)

[三 文件说明： 9](#_Toc8652431)

[四、相关问题及解决方法。 12](#_Toc8652432)

[如何在eclipse的maven环境中创建Servlet3.0并部署到Tomcat？？ 12](#_Toc8652433)

[2）编写后台 Servlet 14](#_Toc8652434)

[Eclipse与Tomcat+MySQL的简单应用 20](#_Toc8652435)

[编写第一个Servlet 26](#_Toc8652436)

[在Eclipse 中编写servlet时出现：The import javax.servlet cannot be resolved 错误，如下图 31](#_Toc8652437)

[**MySQL的下载与安装。** 33](#_Toc8652438)

[maven3.0+；eclipse EE；jdk 9；Tomcat8.5，MySQL---动态web之数据库的使用。 39](#_Toc8652439)

[问题：servlet重定向到jsp后，css样式和图片都没了，解决办法 47](#_Toc8652440)

[问题：使用jsp遍历数据库中表的某列数据。 48](#_Toc8652441)

[问题：访问数据库检查用户身份是否合法，不合法则。。。合法则。。。 49](#_Toc8652442)

[问题：如何把war包部署到Tomcat服务器？ 50](#_Toc8652443)

[错误代码： 52](#_Toc8652444)

[Could not initialize class dbc.JdbcUtil 52](#_Toc8652445)

[问题：MySQL workbench外键的使用 53](#_Toc8652446)

[问题：MySQL workbench的数据库表中使用外键，容易造成数据插入失败，其原因是因为表间的约束关系。 54](#_Toc8652447)

[问题：ajax的使用—封装创建ajax到js，method方法为POST 54](#_Toc8652448)

[问题：将中文资源文件中的中文字符转换为Unicode编码 58](#_Toc8652449)

[问题：异步加载Ajax与json数据实现登录 59](#_Toc8652450)

[问题：href="#"与href="javascript:void(0)"的区别 61](#_Toc8652451)

[javascript:void(0) 含义 61](#_Toc8652452)

[问题：import com.wm.utils.GetList;与import com.wm.utils.HandlePage;的使用方法 62](#_Toc8652453)

[问题：报表的简单使用Chart.js 62](#_Toc8652454)

[问题： ichart教程 62](#_Toc8652455)

[问题： CKeditor5的使用 63](#_Toc8652456)

[问题： 如何配置Tomcat虚拟路径（浏览器无法直接访问盘符上的目录及文件） 68](#_Toc8652457)

[问题： 使用GitHubAPI的相关使用 70](#_Toc8652458)

[问题： 使用Servlet发送get或Post请求 70](#_Toc8652459)

[问题： 将json格式的字符串数据转换为json形式，并读取其中数据 73](#_Toc8652460)

[问题： java编写自动录入数据的程序 79](#_Toc8652461)

[传入Java对象 自动创建动态表 并录入数据 79](#_Toc8652462)

[问题Eclipse开发遇到："The type org.apache.http.HttpResponse cannot be resolved."问题的解决办法 79](#_Toc8652463)

[问题：java语言需要保存用户输入的表单数据，且该数据要多次读取使用，应该如何设置数据的存储类型？ 79](#_Toc8652464)

[问题：将数据库备份成.sql格式 80](#_Toc8652465)

[部署问题：上传图片后立即显示在编辑器上？抓取数据存储在什么地方？多位老师同时抓取数据会出现什么问题？项目名称问题，需要与二级域名的一样？ 81](#_Toc8652466)

[JavaEmail 发送邮件 81](#_Toc8652467)

# 一、开发工具：

1. Eclipse，图标长这样 
2. 相关语言：java、jsp、Servlet、HTML、js、jQuery、CSS等

## 搭建开发环境：（Java 1.8、Tomcat8.5、Eclipse+Maven）

1. Eclipse中使用Maven创建Servlet3.0 Web 项目。

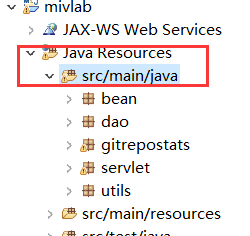
参加：<https://www.open-open.com/lib/view/open1389227144328.html>。

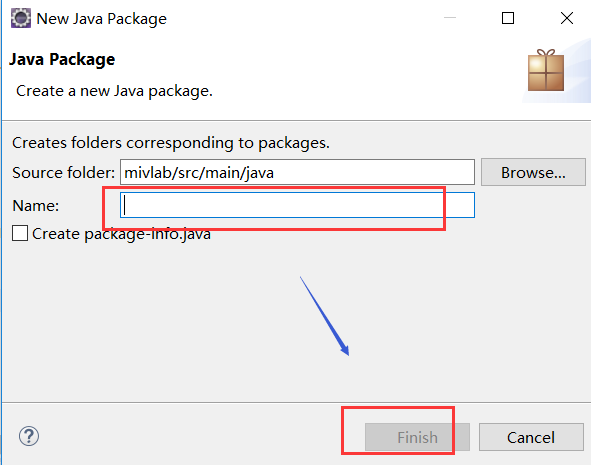
参考上述链接教程。需要注意的是把下图1.6改为1.8，也就是java的版本



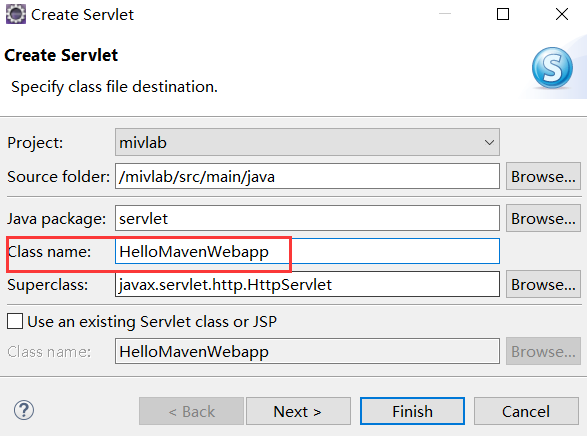
**创建Servlet测试：**

Servlet的编写：在创建好工程后，在工程中找到下图圈起来的地方，右击Java Resource 中的Src/main/java。选择new->Package，输入包的名称，如servlet，然后完成即可

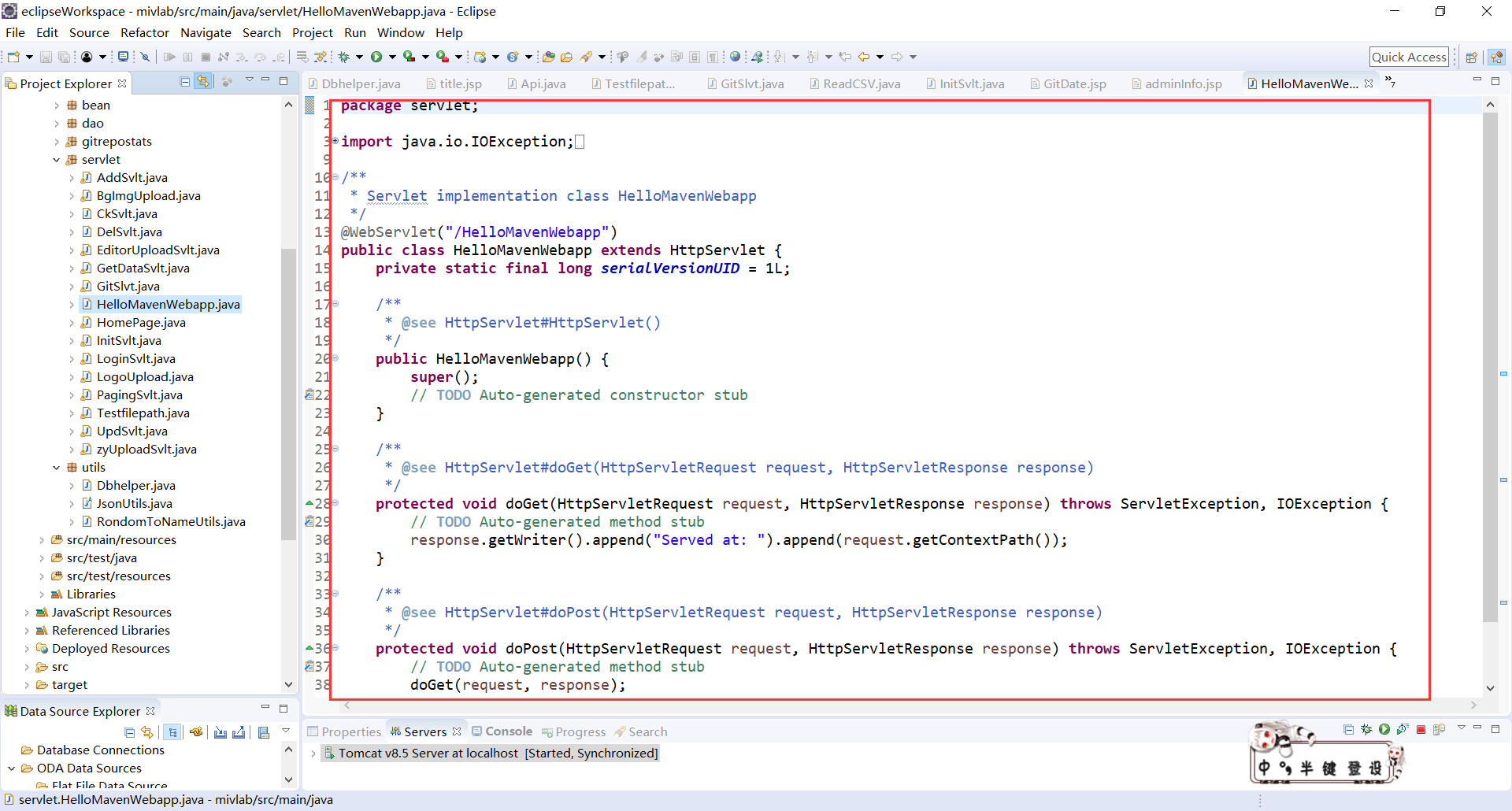




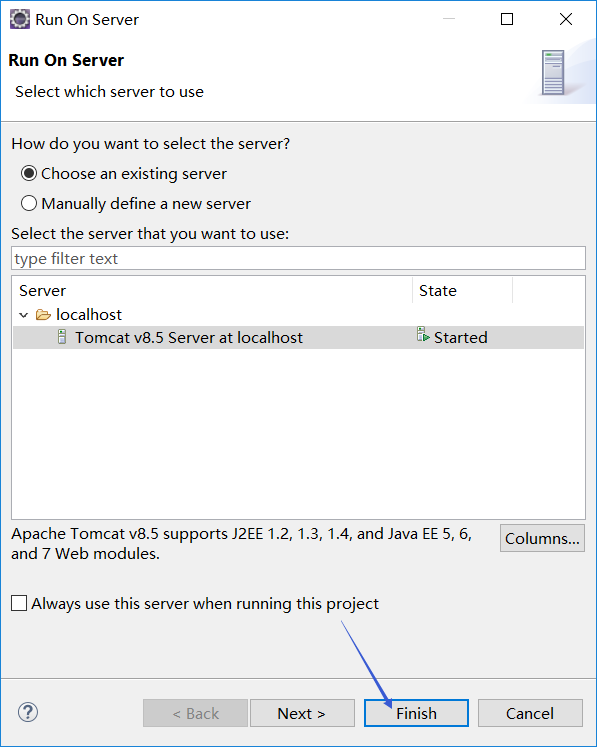
然后在Servlet中找到新创的package（也就是Servlet），右击它，然选择new->servlet->输入名字



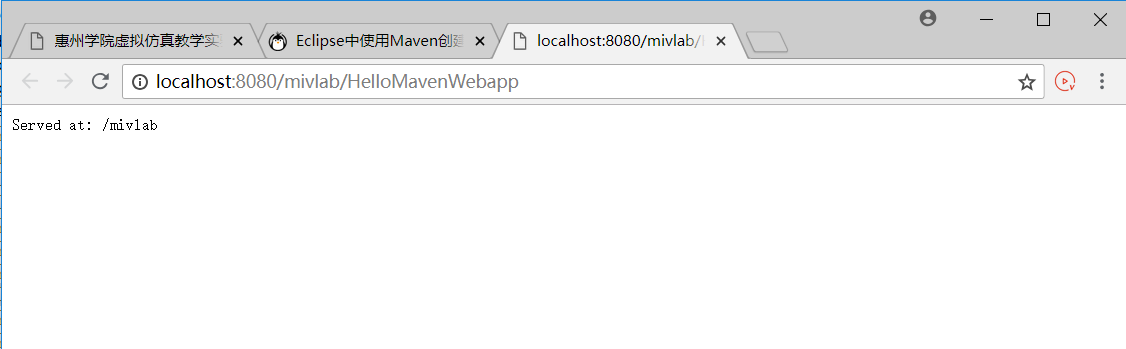
Eclipse会自动生成一个可以使用的Servlet，直接测试即可



**项目的部署**就是右击项目名，点击run as > run on server->选择本地的Tomcat。



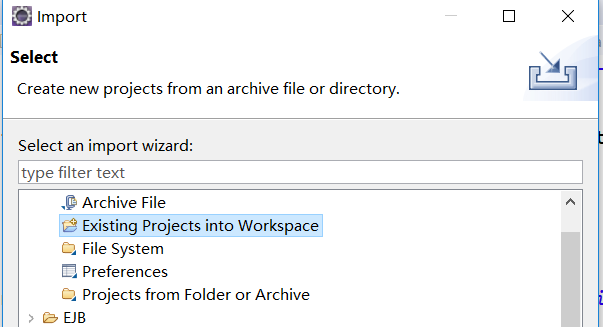
然后浏览器就会出现下图，先把下图实现，能完成这一步，再进行下一步

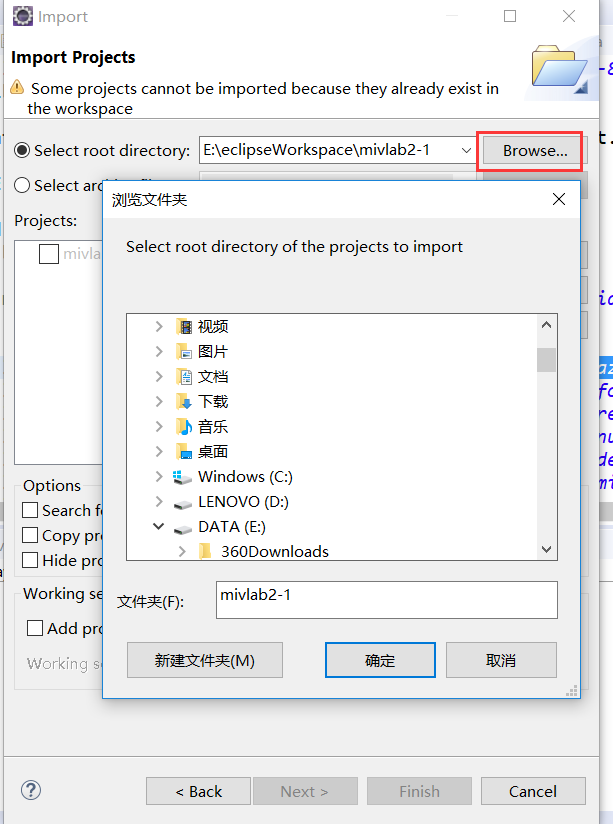


1. 环境搭建好后，将项目导入开发工具eclipse。

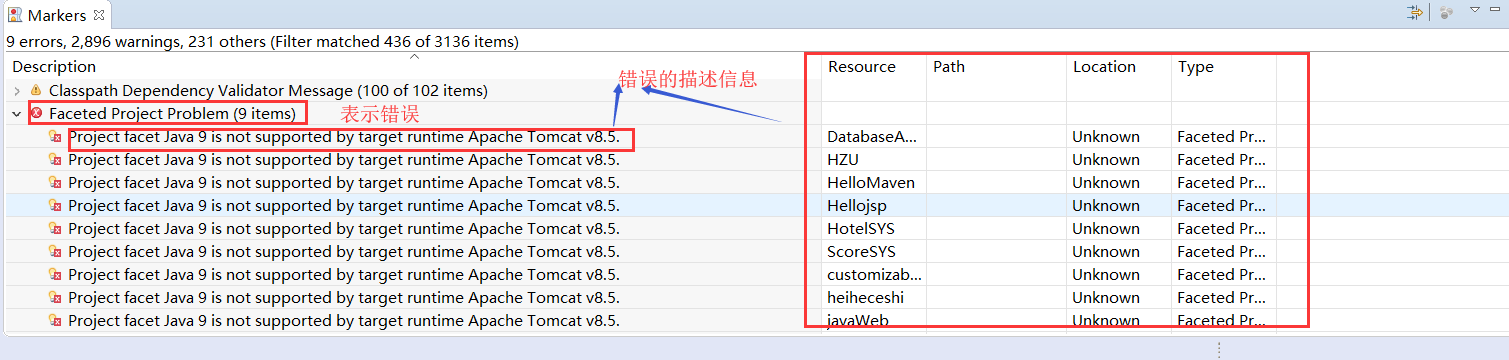
将GitHub上的mivlab源代码下载解压后放入eclipse工作的文件夹（就是eclipse中的workspace对应的文件夹）

在file->import->Existing Projects into Workspace->找到存放代码的文件夹，导入即可





检查有没有报错。

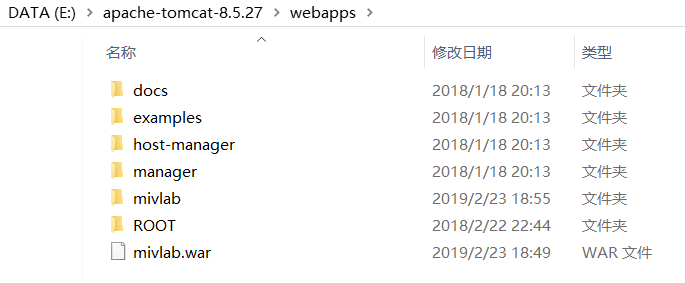


有错的先改正错误，没有错的就可以直接部署运行。（错误一般是因为缺少相关.jar包）

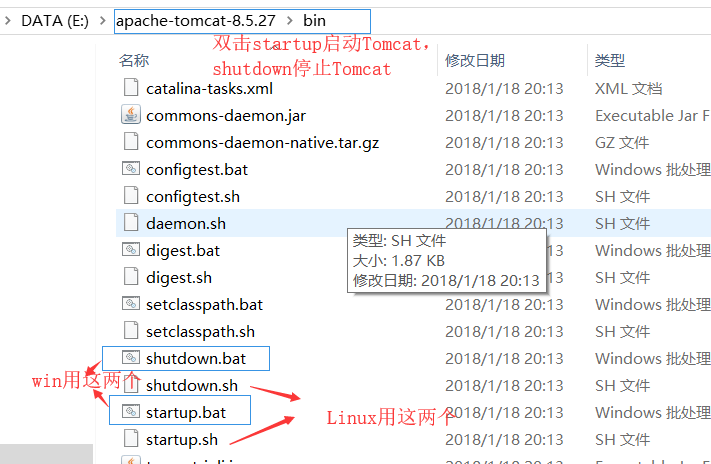
1. 安装配置java和Tomcat。百度教程
2. 数据库：MySQL5.2

# 二、部署工具：Tomcat。

1. 先在开发工具上测试，无误后再在本地的Tomcat上测试，无误后在部署到服务器，也需要进行测试。
2. 在eclipse中将文件导出，格式为.war文件。
3. 本地测试时，将文件放到E:\apache-tomcat-8.5.27\webapps目录下。

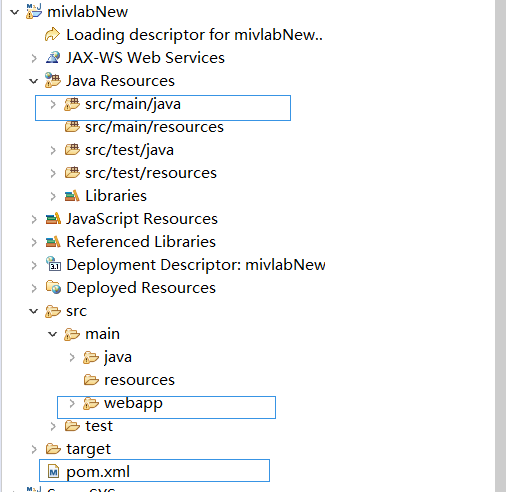


然后，启动Tomcat，



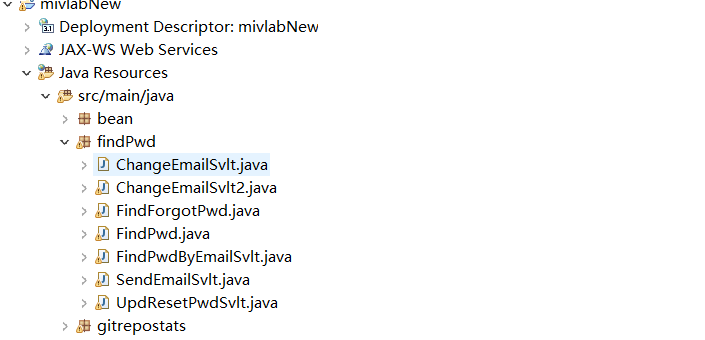
4、 部署到服务器时，需要将.War文件命名为“ROOT.war” 这样访问域名就直接显示设置的默认网页了。 .

# 三 文件说明：



Pox.xml: 用于管理：源代码、配置文件、开发者的信息和角色、问题追踪系统、组织信息、项目授权、项目的url、项目的依赖关系等等



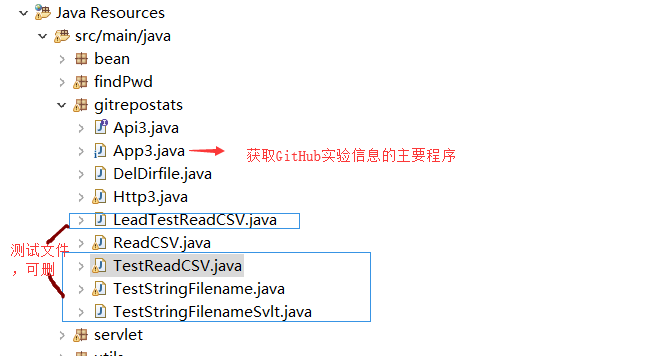


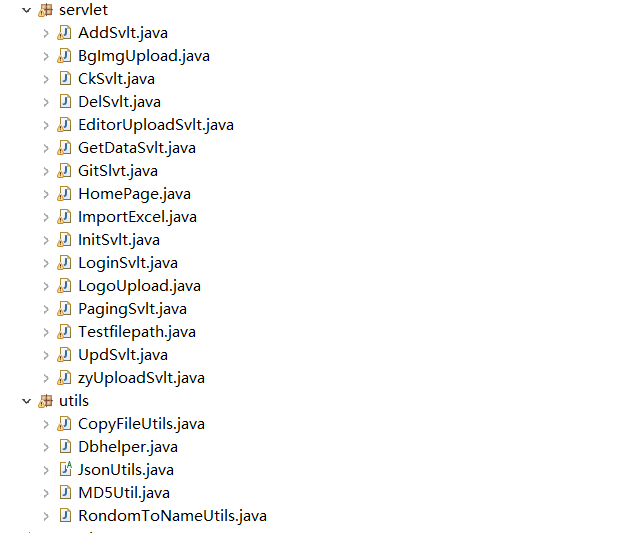
忘记密码：login.jsp---》FindPwd.jsp---》FindPwdByEmailSvlt.java（发送邮件）---》FindForgotPwd.java（验证）---》ResetPwd.jsp-🡪 UpdResetPwdSvlt.java

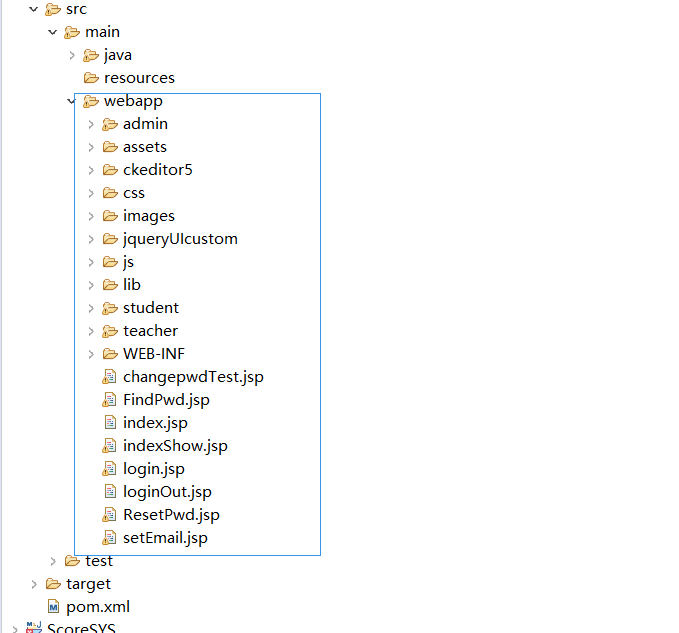
首次登录绑定邮箱：login.jsp---》LoginSvlt.java---》setEmail.jsp---> SendEmailSvlt.java（发送邮件）-🡪 FindPwd.java（验证）-🡪 changepwdTest.jsp-🡪 UpdSvlt.java

重新绑定邮箱：adminInfo.jsp或Students.jsp或TeacherInfo.jsp-🡪 ChangeEmailSvlt.java（发送邮件）-🡪 ChangeEmailSvlt2.java（验证）

注意：运行时将发送邮件的那几java文件中的发件人和密钥改成你们自己的邮箱，不然邮件的功能用不了。







# 四、相关问题及解决方法。

## 如何在eclipse的maven环境中创建Servlet3.0并部署到Tomcat？？

工具：maven3.0+；eclipse EE；jdk 9；Tomcat8.5

环境的搭建：

<http://www.open-open.com/lib/view/open1389227144328.html>

删除web.xml后会出现pom.xml文件报错。

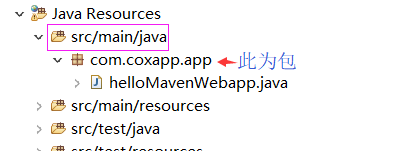


解决如右：

创建Servlet：

如上述操作创建好maven工程后

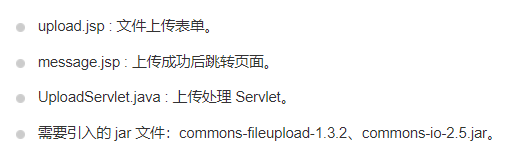
方法1：

* 在src/main/java下创建包，右击鼠标new—>package.-->填写包名。
* 在包下创建类class.—>填写类名。然后再编写Servlet。

方法2：

* 右击项目名称后new—>servlet🡪填写包名类名。

栗子：Servlet文件上传。



1）编写upload.jsp文件。（右击项目new🡪jsp file）

代码如下：

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

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>文件上传实例 - 菜鸟教程</title>

</head>

<body>

<h1>文件上传实例 - 菜鸟教程</h1>

<form method="post" action=" /HelloMaven/helloMavenWebapp " enctype="multipart/form-data">

    选择一个文件:

    <input type="file" name="uploadFile" />

    <br/><br/>

    <input type="submit" value="上传" />

</form>

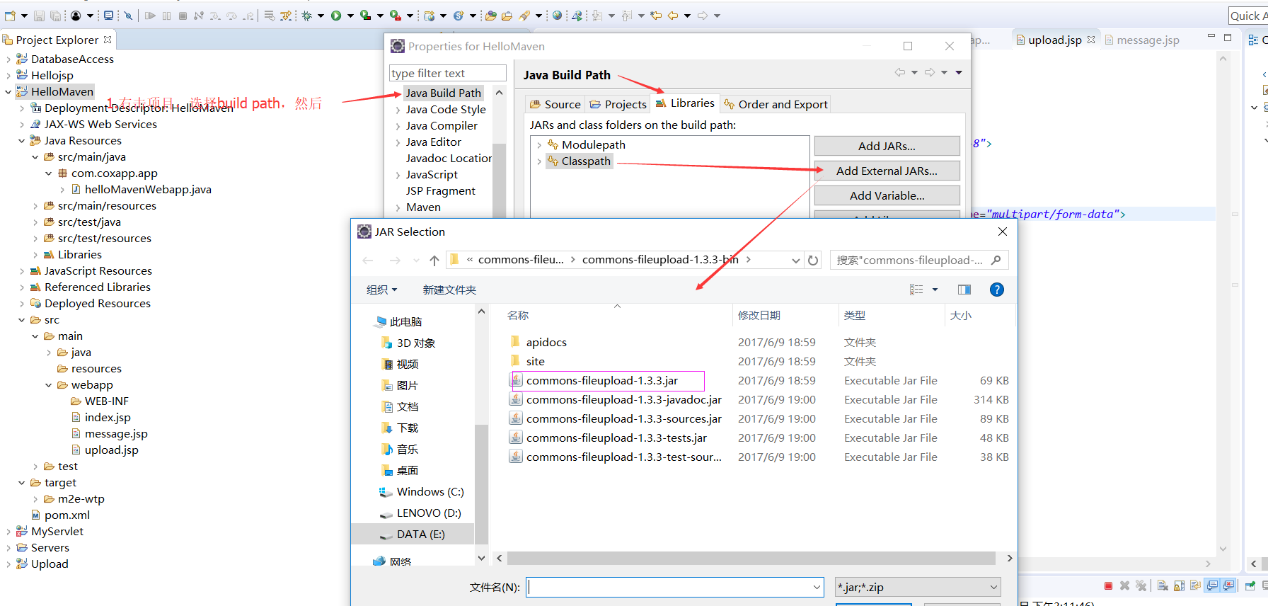
</body>

</html>

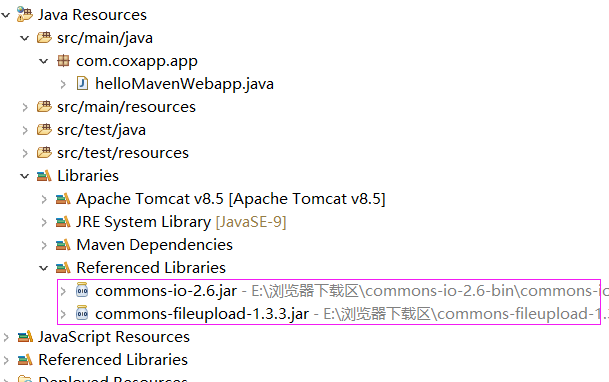
## 2）编写后台 Servlet

以下是 UploadServlet 的源代码，同于处理文件上传，在这之前我们先确保依赖包已经引入到项目的 WEB-INF/lib 目录下：

* 下面的实例依赖于 FileUpload，所以一定要确保在您的 classpath 中有最新版本的 **commons-fileupload.x.x.jar** 文件。可以从 <http://commons.apache.org/proper/commons-fileupload/> 下载。
* FileUpload 依赖于 Commons IO，所以一定要确保在您的 classpath 中有最新版本的 **commons-io-x.x.jar** 文件。可以从 <http://commons.apache.org/proper/commons-io/> 下载。
* 添加方法如下图。**commons-io-x.x.jar文件也是如此。**



然后可以看到



之后，需要把这两个文件放到Tomcat安装目录下的lib目录中。不然会出现启动Tomcat失败。

接着编写

helloMavenWebapp源代码 如下所示：

package com.coxapp.app;

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

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 org.apache.commons.fileupload.FileItem;

import org.apache.commons.fileupload.disk.DiskFileItemFactory;

import org.apache.commons.fileupload.servlet.ServletFileUpload;

/\*\*

\* Servlet implementation class UploadServlet

\*/

@WebServlet("/helloMavenWebapp ")

public class helloMavenWebapp extends HttpServlet {

private static final long serialVersionUID = 1L;

// 上传文件存储目录

private static final String UPLOAD\_DIRECTORY = "upload";

// 上传配置

private static final int MEMORY\_THRESHOLD = 1024 \* 1024 \* 3; // 3MB

private static final int MAX\_FILE\_SIZE = 1024 \* 1024 \* 40; // 40MB

private static final int MAX\_REQUEST\_SIZE = 1024 \* 1024 \* 50; // 50MB

/\*\*

\* 上传数据及保存文件

\*/

protected void doPost(HttpServletRequest request,

        HttpServletResponse response) throws ServletException, IOException {

        // 检测是否为多媒体上传

        if (!ServletFileUpload.isMultipartContent(request)) {

         // 如果不是则停止

         PrintWriter writer = response.getWriter();

         writer.println("Error: 表单必须包含 enctype=multipart/form-data");

         writer.flush();

         return;

        }

// 配置上传参数

DiskFileItemFactory factory = new DiskFileItemFactory();

// 设置内存临界值 - 超过后将产生临时文件并存储于临时目录中

factory.setSizeThreshold(MEMORY\_THRESHOLD);

// 设置临时存储目录

factory.setRepository(new File(System.getProperty("java.io.tmpdir")));

ServletFileUpload upload = new ServletFileUpload(factory);

// 设置最大文件上传值

upload.setFileSizeMax(MAX\_FILE\_SIZE);

// 设置最大请求值 (包含文件和表单数据)

upload.setSizeMax(MAX\_REQUEST\_SIZE);

// 中文处理

upload.setHeaderEncoding("UTF-8");

// 构造临时路径来存储上传的文件

// 这个路径相对当前应用的目录

String uploadPath = request.getServletContext().getRealPath("./") + File.separator + UPLOAD\_DIRECTORY;

// 如果目录不存在则创建

File uploadDir = new File(uploadPath);

if (!uploadDir.exists()) {

uploadDir.mkdir();

}

try {

// 解析请求的内容提取文件数据

//@SuppressWarnings("unchecked")

List<FileItem> formItems = upload.parseRequest(request);

if (formItems != null && formItems.size() > 0) {

// 迭代表单数据

for (FileItem item : formItems) {

// 处理不在表单中的字段

if (!item.isFormField()) {

String fileName = new File(item.getName()).getName();

String filePath = uploadPath + File.separator + fileName;

File storeFile = new File(filePath);

// 在控制台输出文件的上传路径

System.out.println(filePath);

// 保存文件到硬盘

item.write(storeFile);

request.setAttribute("message",

"文件上传成功!");

}

}

}

} catch (Exception ex) {

request.setAttribute("message",

"错误信息: " + ex.getMessage());

}

// 跳转到 message.jsp

request.getServletContext().getRequestDispatcher("/message.jsp").forward(

request, response);

}

}

message.jsp 文件代码如下：

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

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>文件上传结果</title>

</head>

<body>

<center>

<h2>${message}</h2>

</center>

</body>

</html>

**Pom.xml文件如下**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.coxapp.app</groupId>

<artifactId>HelloMaven</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>HelloMaven Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<failOnMissingWebXml>false</failOnMissingWebXml>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<finalName>HelloMaven</finalName>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.0</version>

<configuration>

<source>9</source>

<target>9</target>

</configuration>

</plugin>

</plugins>

</build>

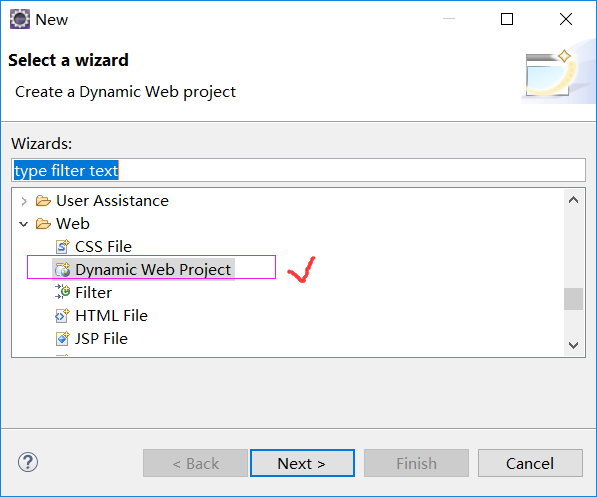
</project>

**。。**当您在浏览器中访问：http://localhost:8080/HelloMaven/upload.jsp 后选择文件上传将跳转到<http://localhost:8080/HelloMaven/helloMavenWebapp>

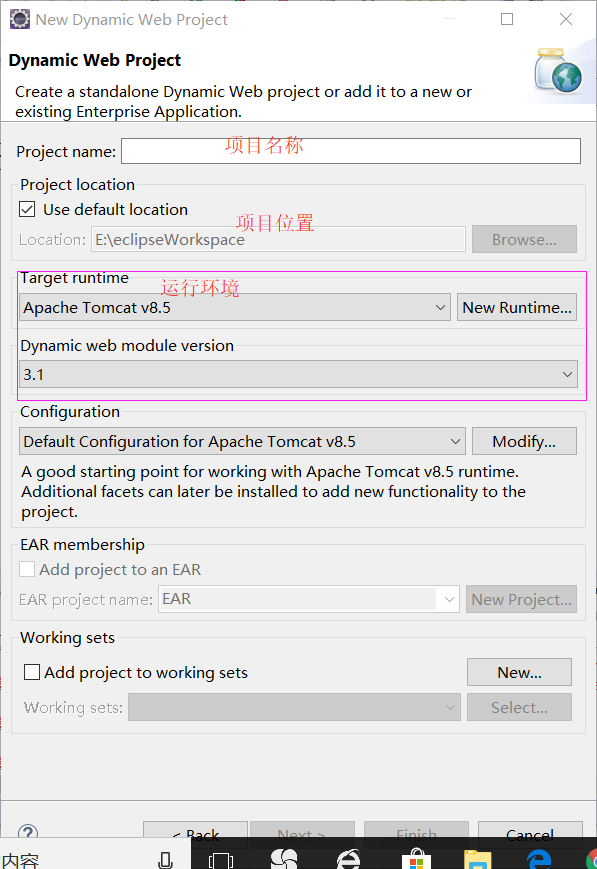
## Eclipse与Tomcat+MySQL的简单应用

1. 在eclipse中创建Servlet项目并在Tomcat中显示。

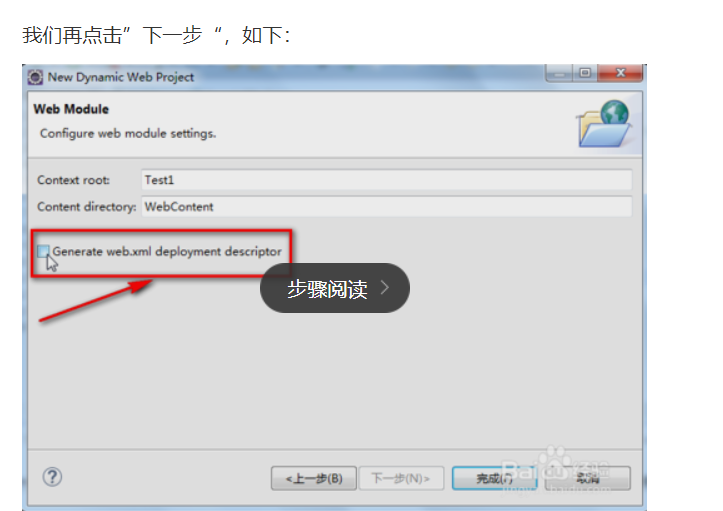
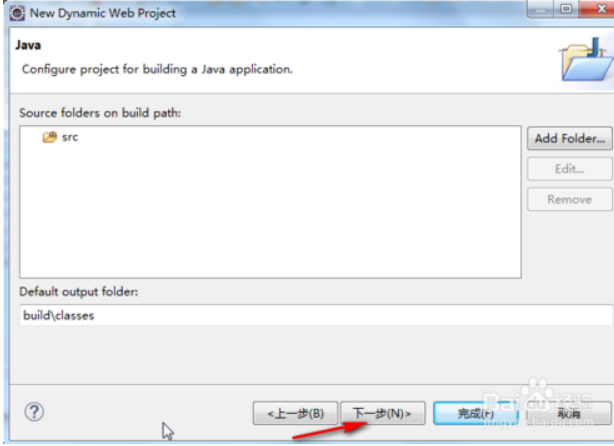
File->new->other->



R然后填写项目名称，选择运行环境



点击下一步

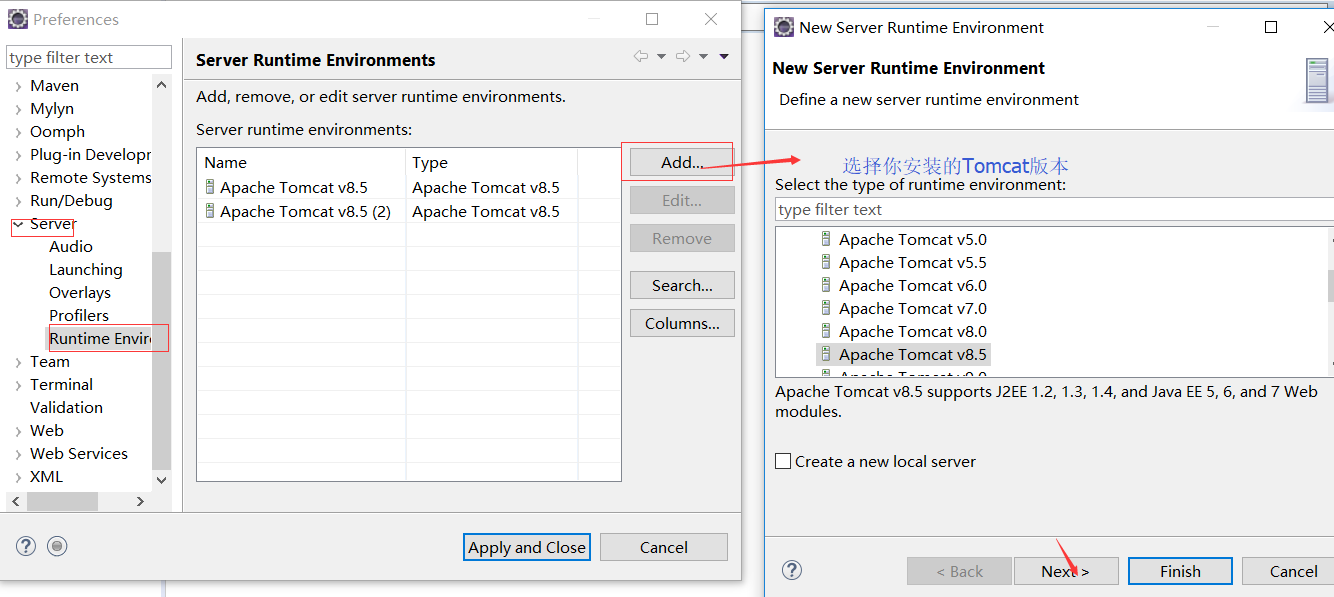


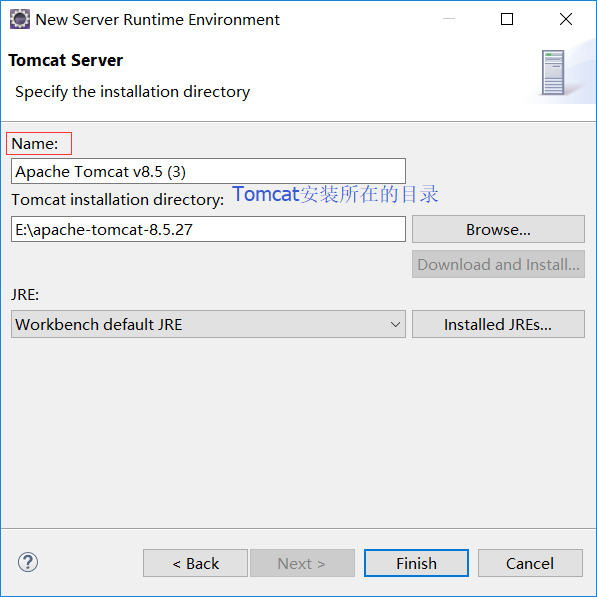
然后完成。



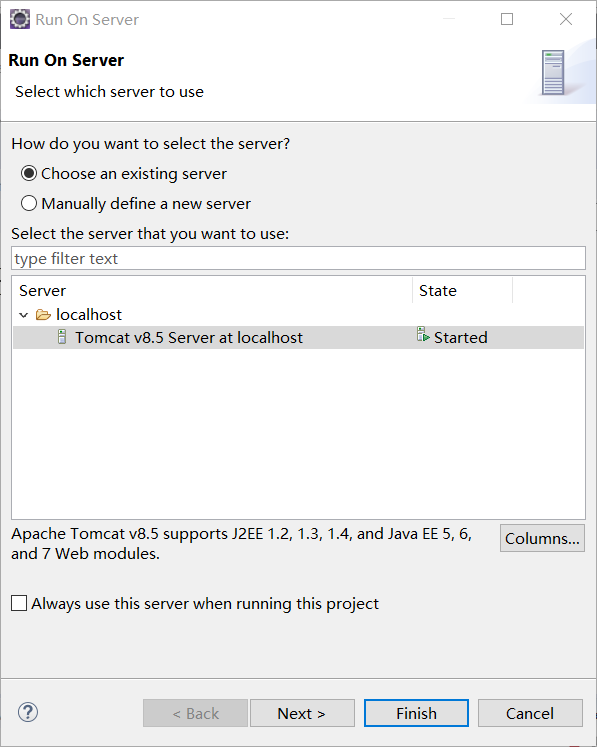
若需创建一个静态的HTML。点击左侧栏WebContent->new->HTML file.

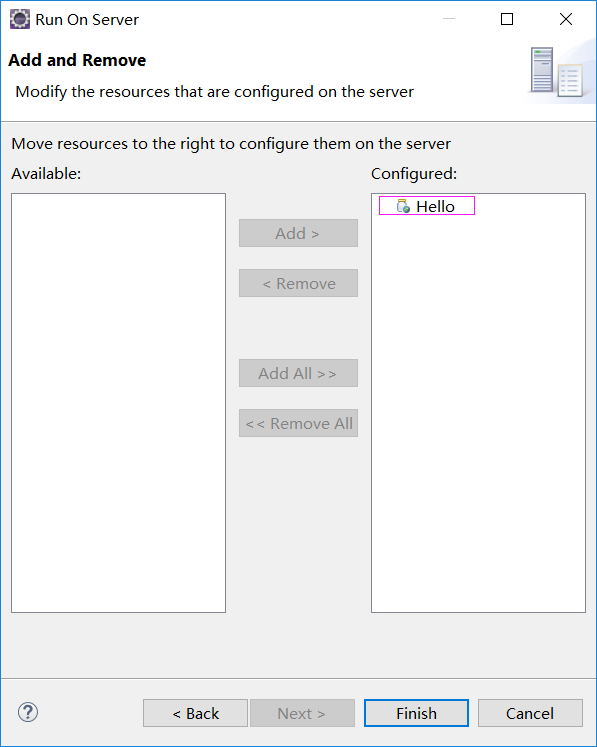
创建成功，填写需要的内容，然后在Window->Preferences





点击项目总名称，run on Servlet，就可以了，步骤如下。

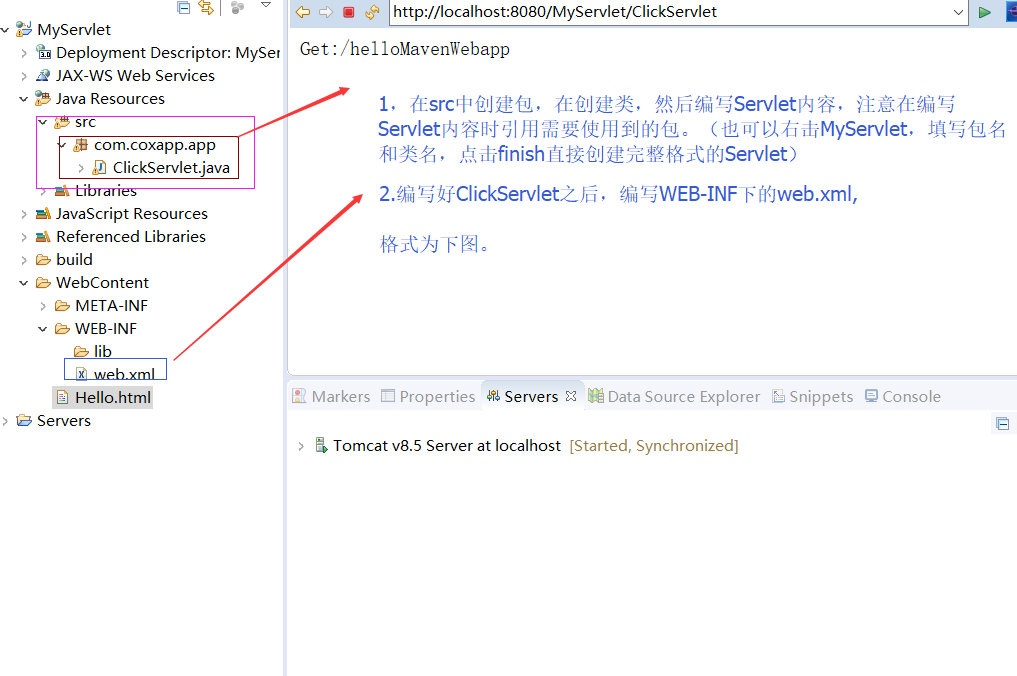


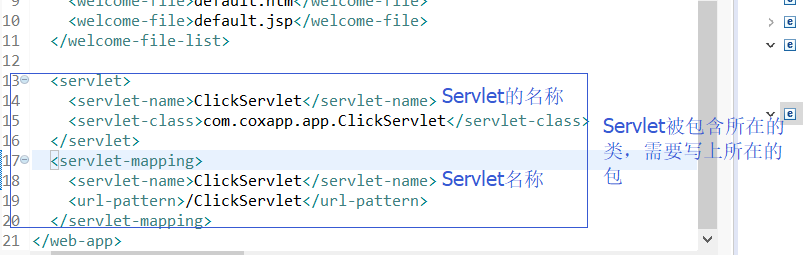


注意：若在这之前打开了Tomcat，则会报错（因为在eclipse中运行时会再次打开Tomcat）。此时应该先关闭Tomcat。

## 编写第一个Servlet

首先创建一个动态网页项目。步骤如上，在上面的侧栏中的





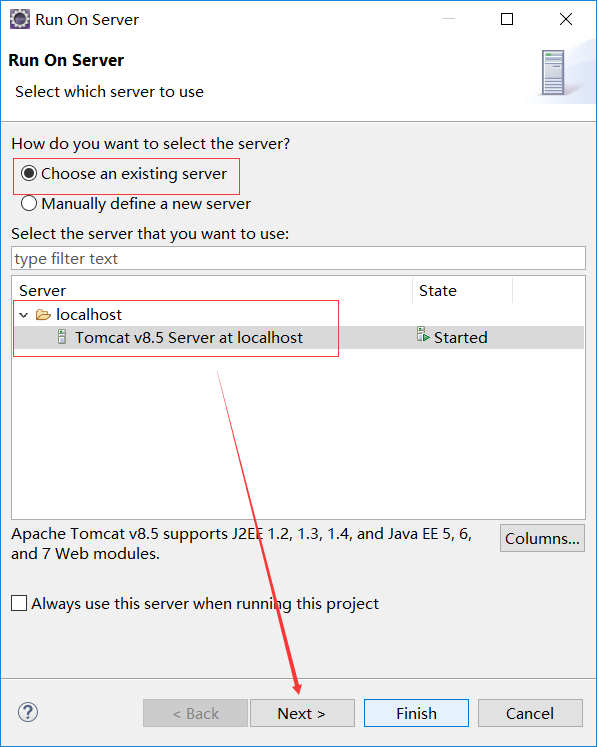
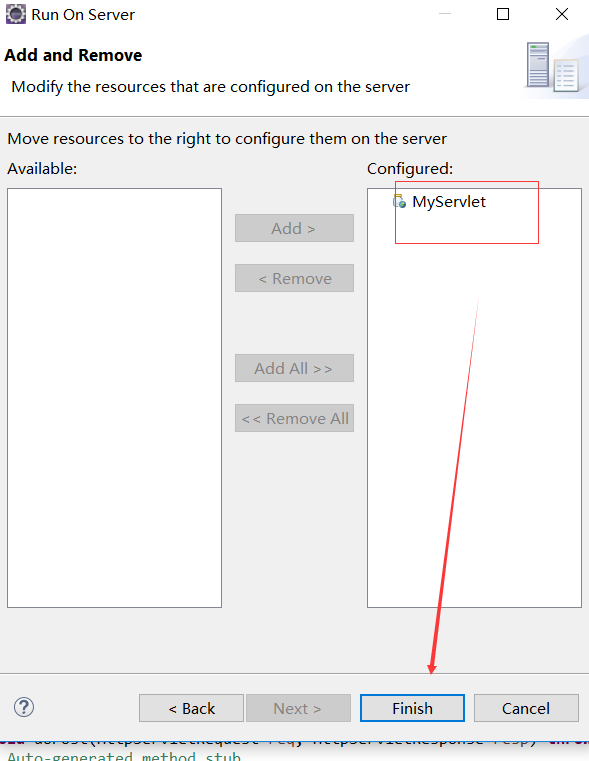
上图中<url-pattern>/ClickServlet</url-pattern>（表映射，当某Servlet被触发时，根据此找到与此对应的Servlet，也就是<servlet>中的内容,将此内容渲染在浏览器）；

其中的/ClickServlet与

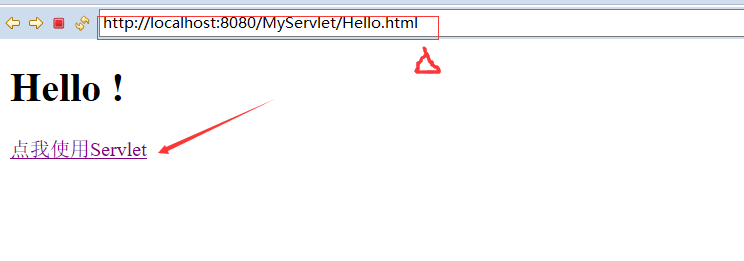


中的href=”ClickServlet”相对应。（当点击这个链接时，会执行对应的Servlet）

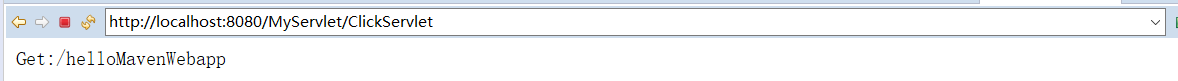
保存然后点击MyServlet->Run as->Run on server

然后输入要访问的页面，单击后会跳转



跳转后显示



也有可以会出现启动Tomcat失败的错误提示，原因及解决方法如下



附录：

Servlet代码如下，展开查看

package com.coxapp.app;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class ClickServlet

\*/

//@WebServlet("/ClickServlet")

public class ClickServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* Default constructor.

\*/

public ClickServlet() {

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest req, HttpServletResponse resp)

\*/

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

// TODO Auto-generated method stub

resp.getWriter().println("Get:/helloMavenWebapp");

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest req, HttpServletResponse resp)

\*/

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

// TODO Auto-generated method stub

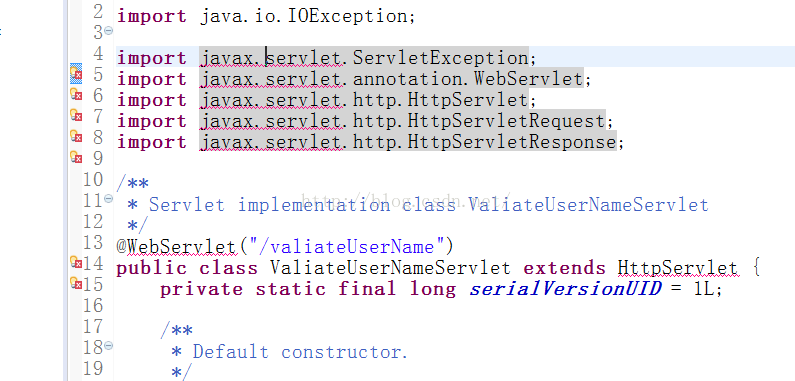
resp.getWriter().println("Post:/helloMavenWebapp");

}

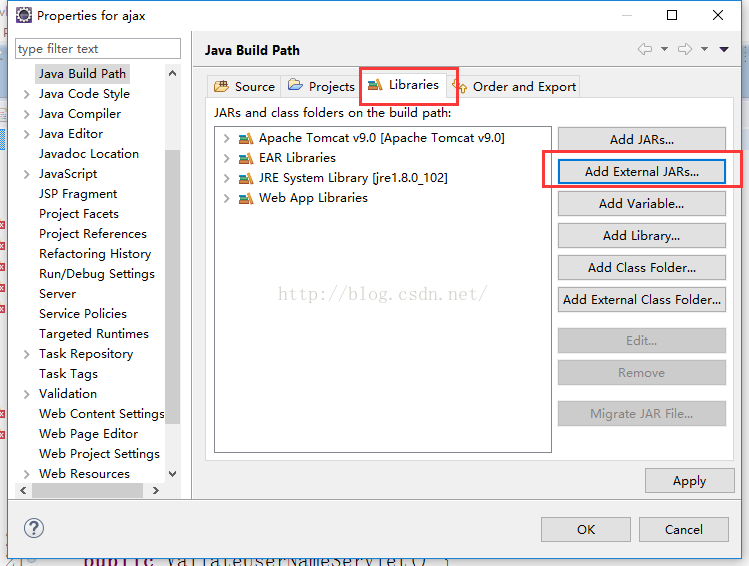
}

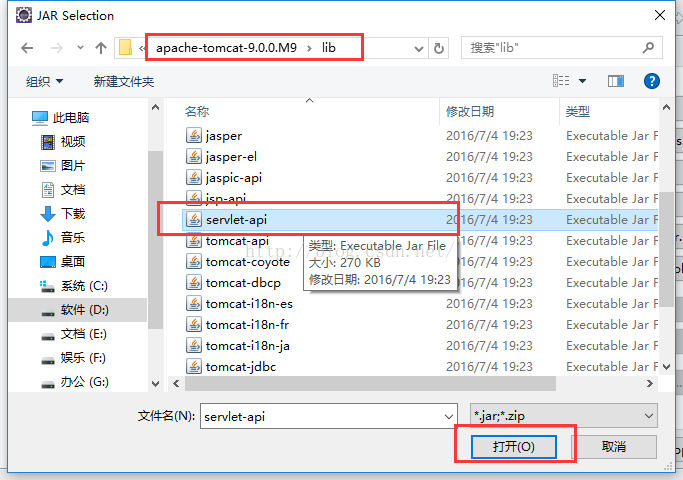
若出现出现----The import javax.servlet cannot be resolved 的解决方法

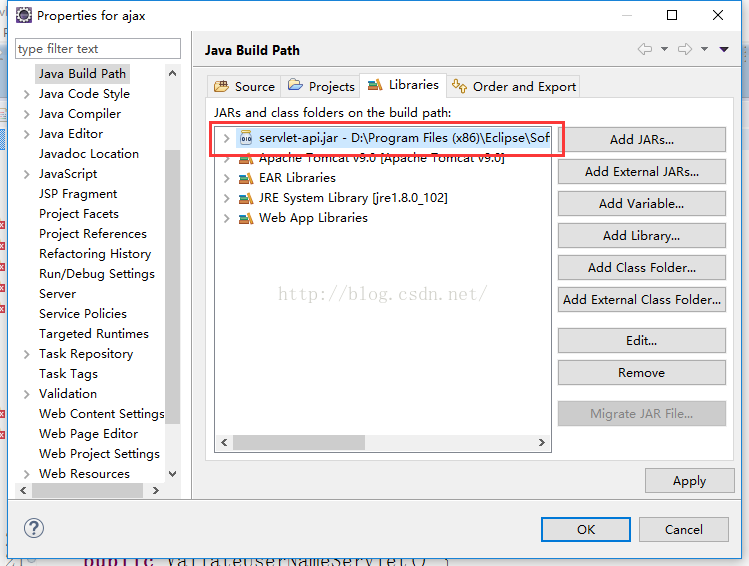
## 在Eclipse 中编写servlet时出现：The import javax.servlet cannot be resolved 错误，如下图

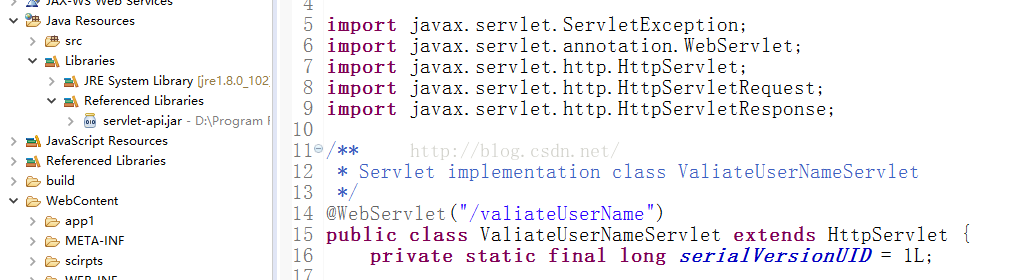


解决方法就是：在Eclipse中，右击项目，选择Build Path->configure build path->Libraries->Add External JARs，找到你计算机中tomcat的解压路径，在lib文件夹下选中"servlet-api.jar"，添加点击“确定”  ，如下图：









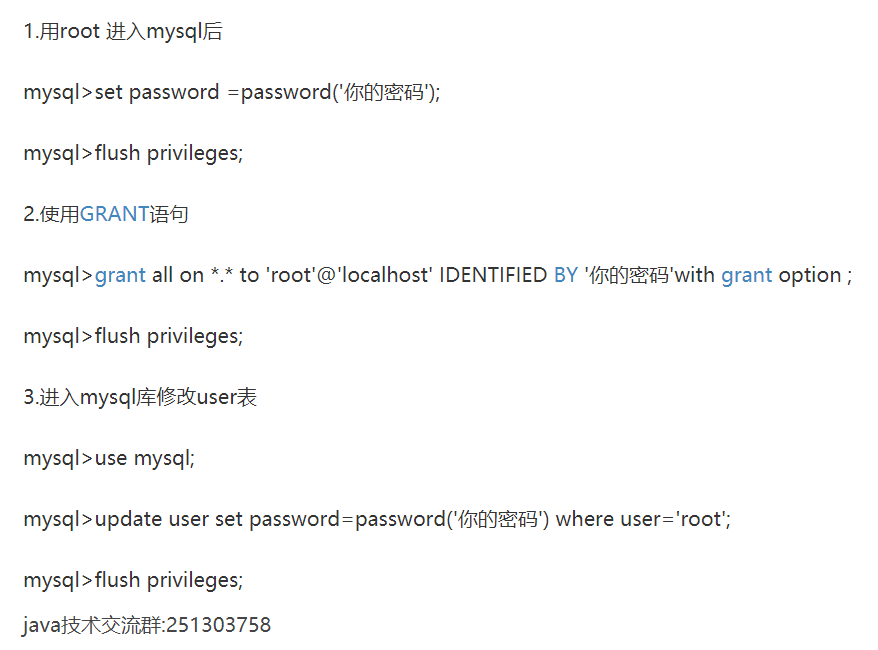
心得：在学习使用一个新的编译器时，应该先确定使用的版本号，查找对应的视频教程，除了认真听讲的同时，注意自己编译器出现的错误，查看控制台信息，百度或者进行翻译查看错误的原因及解决方法。

**MySQL的下载与安装。**

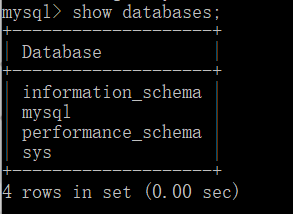
1. 基本安装—

mysql-5.7.16-winx64解压版安装超详细图文教程<http://blog.csdn.net/li_magic/article/details/70186609>

1. 设置密码 <http://blog.csdn.net/li_magic/article/details/70187609>



1. 使用mysql -hlocalhost -uroot -p或者是mysql -u root -p；命令进入MySQL
2. 数据库的创建；
3. 进入MySQL后，输入show databases 命令，展示数据库列表。



使用时，首先创建数据库，



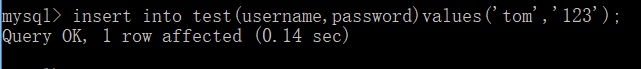
切换到需要使用的数据库

创建表；

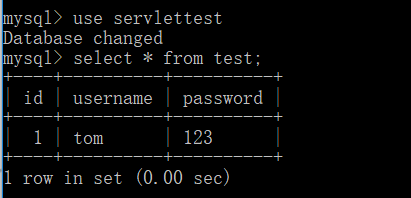


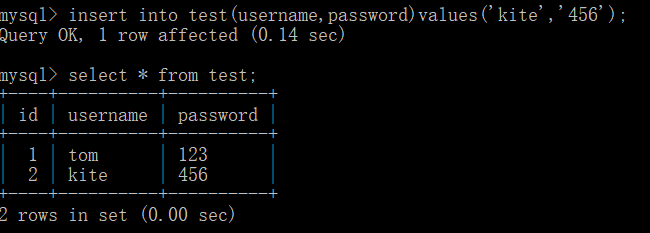
进行对表的更改

如：插入



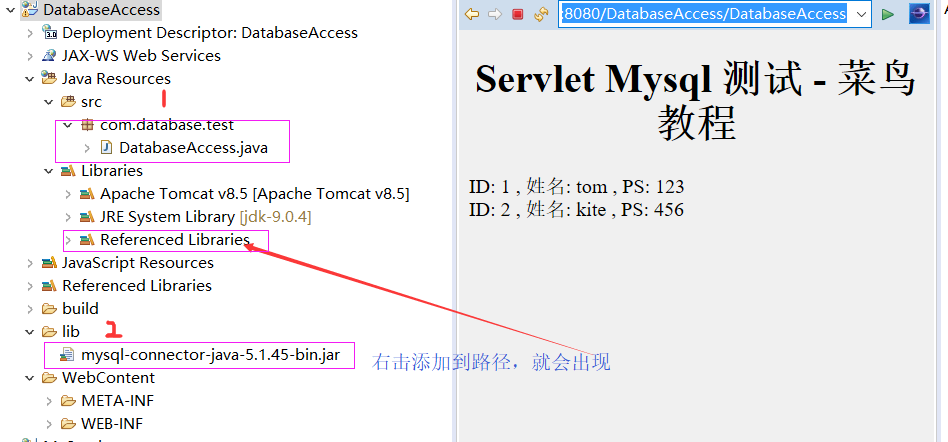
如需查看某数据库的某个表。





（退出MySQL命令：exit; ）

1. 数据库创建好后，进行Servlet与数据库的连接。



DatabaseAccess.java代码

package com.runoob.test;

//改成自己的包

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.\*;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class DatabaseAccess

\*/

@WebServlet("/DatabaseAccess")

public class DatabaseAccess extends HttpServlet {

    private static final long serialVersionUID = 1L;

    // JDBC 驱动名及数据库 URL

    static final String JDBC\_DRIVER = "com.mysql.jdbc.Driver";

    static final String DB\_URL = "jdbc:mysql://localhost:3306/RUNOOB";

    // 数据库的用户名与密码，需要根据自己的设置

    static final String USER = "root";

    static final String PASS = "123456";

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public DatabaseAccess() {

super();

// TODO Auto-generated constructor stub

}

    /\*\*

     \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

     \*/

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        Connection conn = null;

        Statement stmt = null;

        // 设置响应内容类型

        response.setContentType("text/html;charset=UTF-8");

        PrintWriter out = response.getWriter();

        String title = "Servlet Mysql 测试 - 菜鸟教程";

        String docType = "<!DOCTYPE html>\n";

        out.println(docType +

        "<html>\n" +

        "<head><title>" + title + "</title></head>\n" +

        "<body bgcolor=\"#f0f0f0\">\n" +

        "<h1 align=\"center\">" + title + "</h1>\n");

        try{

            // 注册 JDBC 驱动器

            Class.forName("com.mysql.jdbc.Driver");

            // 打开一个连接

            conn = DriverManager.getConnection(DB\_URL,USER,PASS);

            // 执行 SQL 查询

            stmt = conn.createStatement();

            String sql;

            sql = "SELECT id, name, url FROM websites";

            ResultSet rs = stmt.executeQuery(sql);

            // 展开结果集数据库

            while(rs.next()){

                // 通过字段检索

                int id = rs.getInt("id");

                String name = rs.getString("name");

                String url = rs.getString("url");

                // 输出数据

                out.println("ID: " + id);

                out.println(", 站点名称: " + name);

                out.println(", 站点 URL: " + url);

                out.println("<br />");

            }

            out.println("</body></html>");

            // 完成后关闭

            rs.close();

            stmt.close();

            conn.close();

        } catch(SQLException se) {

            // 处理 JDBC 错误

            se.printStackTrace();

        } catch(Exception e) {

            // 处理 Class.forName 错误

            e.printStackTrace();

        }finally{

            // 最后是用于关闭资源的块

            try{

                if(stmt!=null)

                stmt.close();

            }catch(SQLException se2){

            }

            try{

                if(conn!=null)

                conn.close();

            }catch(SQLException se){

                se.printStackTrace();

            }

        }

    }

    /\*\*

     \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

     \*/

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        // TODO Auto-generated method stub

        doGet(request, response);

    }

}

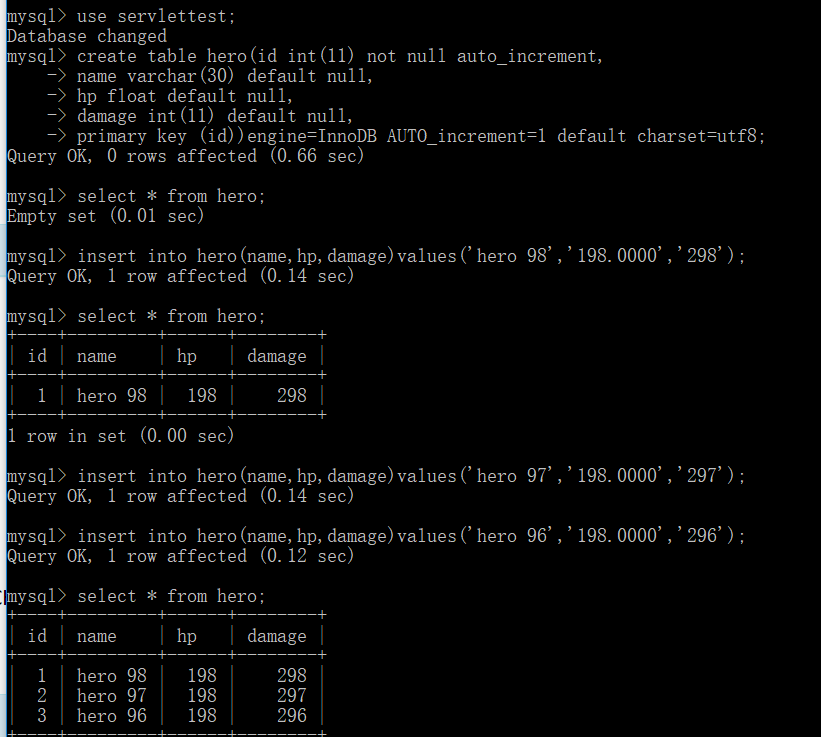
若出现无法连接到数据库的错误提示，检查是否将第二步的jar包引入路径，再者就是运行时重新启动Tomcat

## maven3.0+；eclipse EE；jdk 9；Tomcat8.5，MySQL---动态web之数据库的使用。

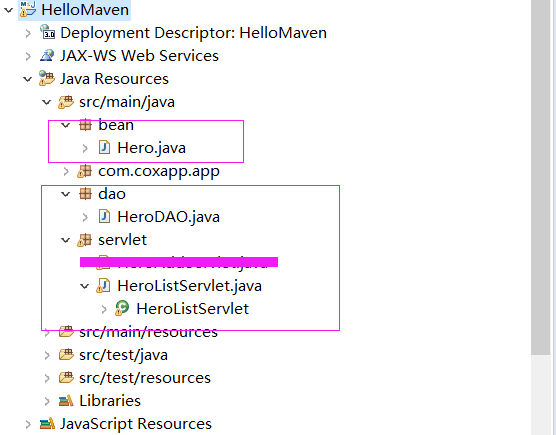
**学习网址：**<http://how2j.cn/k/servlet/servlet-query/563.html>

数据库CRUD（C - Creation 增加 ；R - Retrieve 查询 ；U - Update 修改 ；D - DELETE 删除 ）的综合学习使用。

1. 查询数据库的信息并显示在网页上。
2. 先建立数据库，创建表hero，并添加一些数据，方法同上。（连接数据库）



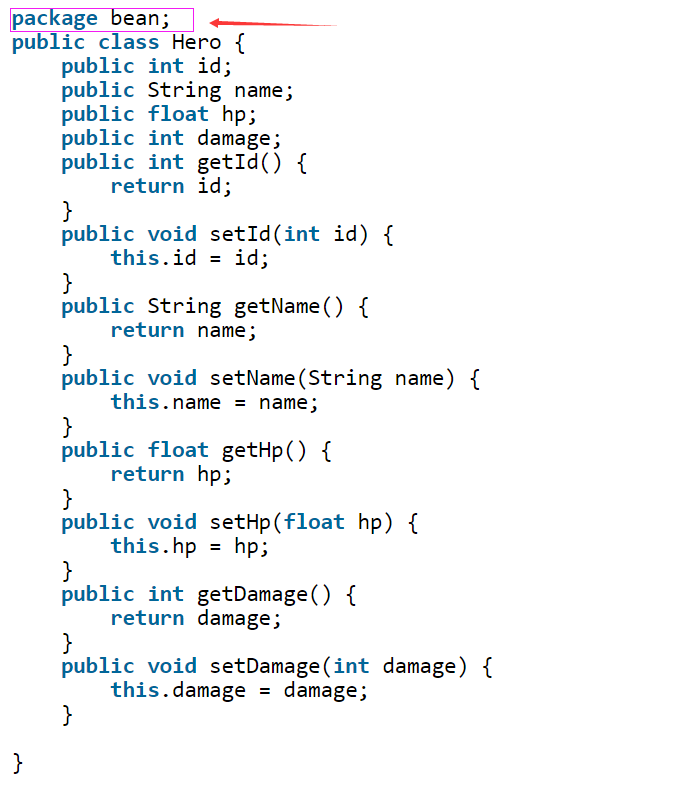
1. 准备实体类Hero
2. 准备DAO类和HeroDAO
3. 编写HeroListServlet
4. Run as Servlet。访问<http://localhost:8080/HelloMaven/HeroListServlet>
5. 在工程中的位置



1. 代码如下

Hero类有id,name,hp,damage等属性。

并且为每一个属性提供public的getter和setter。



准备一个HeroDAO，提供增加，删除，修改，查询等常规数据库操作方法（注意引进的包）

package dao;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import bean.Hero;

public class HeroDAO {

    public HeroDAO() {

        try {

            Class.forName("com.mysql.jdbc.Driver");

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

    }

    public Connection getConnection() throws SQLException {

        return DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8","root",

                "admin");

    }

    public int getTotal() {

        int total = 0;

        try (Connection c = getConnection(); Statement s = c.createStatement();) {

            String sql = "select count(\*) from hero";

            ResultSet rs = s.executeQuery(sql);

            while (rs.next()) {

                total = rs.getInt(1);

            }

            System.out.println("total:" + total);

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return total;

    }

    public void add(Hero hero) {

        String sql = "insert into hero values(null,?,?,?)";

        try (Connection c = getConnection(); PreparedStatement ps = c.prepareStatement(sql);) {

            ps.setString(1, hero.name);

            ps.setFloat(2, hero.hp);

            ps.setInt(3, hero.damage);

            ps.execute();

            ResultSet rs = ps.getGeneratedKeys();

            if (rs.next()) {

                int id = rs.getInt(1);

                hero.id = id;

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    public void update(Hero hero) {

        String sql = "update hero set name= ?, hp = ? , damage = ? where id = ?";

        try (Connection c = getConnection(); PreparedStatement ps = c.prepareStatement(sql);) {

            ps.setString(1, hero.name);

            ps.setFloat(2, hero.hp);

            ps.setInt(3, hero.damage);

            ps.setInt(4, hero.id);

            ps.execute();

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    public void delete(int id) {

        try (Connection c = getConnection(); Statement s = c.createStatement();) {

            String sql = "delete from hero where id = " + id;

            s.execute(sql);

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    public Hero get(int id) {

        Hero hero = null;

        try (Connection c = getConnection(); Statement s = c.createStatement();) {

            String sql = "select \* from hero where id = " + id;

            ResultSet rs = s.executeQuery(sql);

            if (rs.next()) {

                hero = new Hero();

                String name = rs.getString(2);

                float hp = rs.getFloat("hp");

                int damage = rs.getInt(4);

                hero.name = name;

                hero.hp = hp;

                hero.damage = damage;

                hero.id = id;

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return hero;

    }

    public List<Hero> list() {

        return list(0, Short.MAX\_VALUE);

    }

    public List<Hero> list(int start, int count) {

        List<Hero> heros = new ArrayList<Hero>();

        String sql = "select \* from hero order by id desc limit ?,? ";

        try (Connection c = getConnection(); PreparedStatement ps = c.prepareStatement(sql);) {

            ps.setInt(1, start);

            ps.setInt(2, count);

            ResultSet rs = ps.executeQuery();

            while (rs.next()) {

                Hero hero = new Hero();

                int id = rs.getInt(1);

                String name = rs.getString(2);

                float hp = rs.getFloat("hp");

                int damage = rs.getInt(4);

                hero.id = id;

                hero.name = name;

                hero.hp = hp;

                hero.damage = damage;

                heros.add(hero);

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return heros;

    }

}

一般会这样命名,以查询为例 HeroListServlet ： [表][行为]Servlet 这样一种命名规则。  
  
所以对于Hero而言就会如此命名：  
增加 HeroAddServlet  
删除 HeroDeleteServlet  
编辑 HeroEditServlet  
修改 HeroUpdateServlet  
查询 HeroListServlet  
  
  
在HeroListServlet中，会使用HeroDAO把数据查询出来，然后拼接成一个table用于显示其内容

package servlet;

import java.io.IOException;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import bean.Hero;

import dao.HeroDAO;

public class HeroListServlet extends HttpServlet {

protected void service(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html; charset=UTF-8");

List<Hero> heros = new HeroDAO().list();

StringBuffer sb = new StringBuffer();

sb.append("<table align='center' border='1' cellspacing='0'>\r\n");

sb.append("<tr><td>id</td><td>name</td><td>hp</td><td>damage</td></tr>\r\n");

String trFormat = "<tr><td>%d</td><td>%s</td><td>%f</td><td>%d</td></tr>\r\n";

for (Hero hero : heros) {

String tr = String.format(trFormat, hero.getId(), hero.getName(), hero.getHp(), hero.getDamage());

sb.append(tr);

}

sb.append("</table>");

response.getWriter().write(sb.toString());

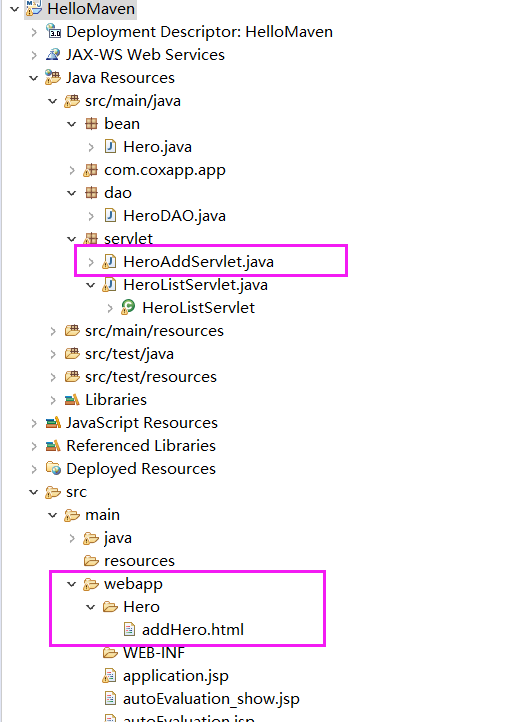
}

}

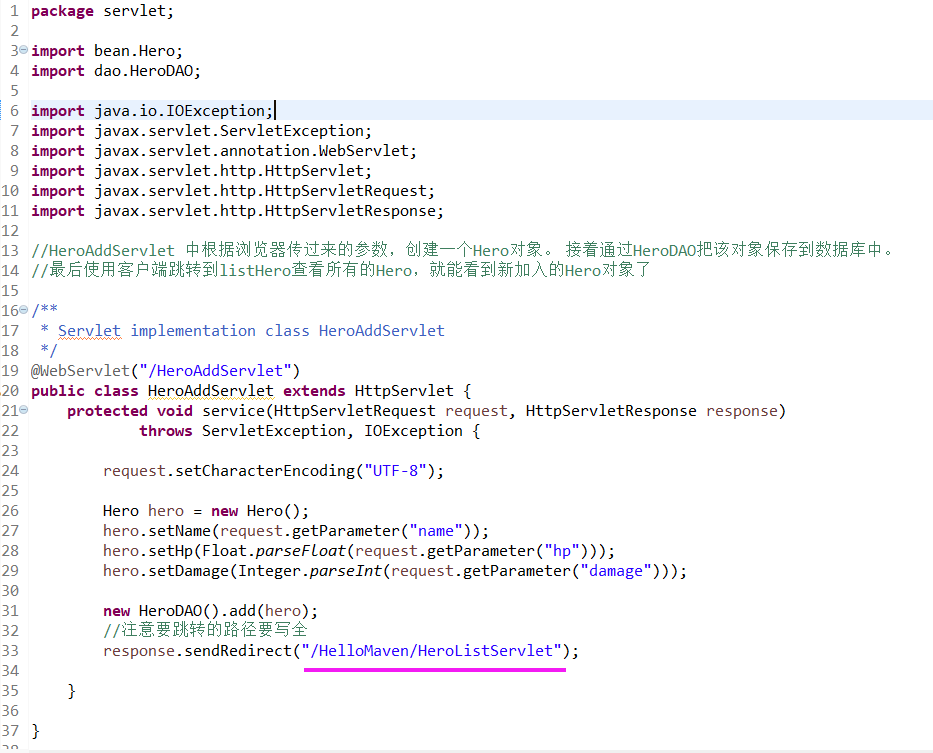
1. 通过在网页上添加数据，将数据存储到数据库中

在“一”的基础上，进行“二”的操作。

1. 准备增加的页面 addHero.html
2. 编写 HeroAddServlet（HeroAddServlet 中根据浏览器传过来的参数，创建一个Hero对象。 接着通过HeroDAO把该对象保存到数据库中。  
   最后使用客户端跳转到listHero查看所有的Hero，就能看到新加入的Hero对象了）
3. 在工程中的位置



1. 代码

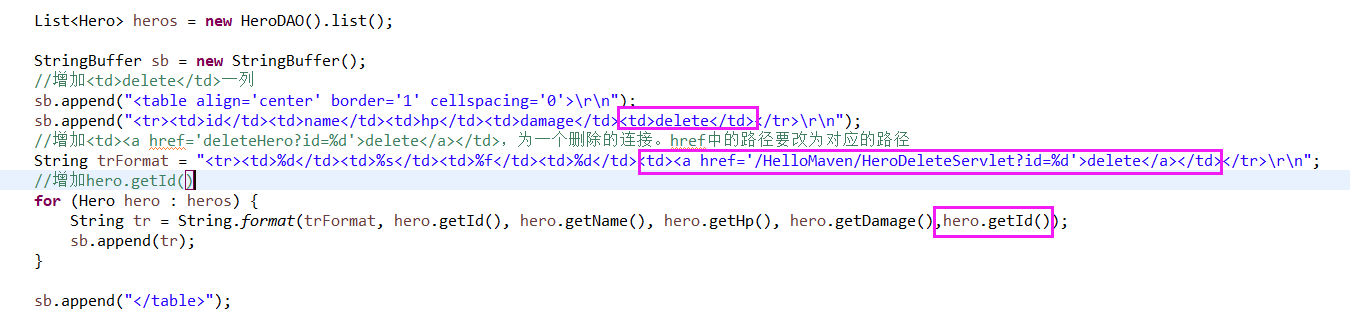
访问网址：<http://localhost:8080/HelloMaven/Hero/addHero.html>

添加后，也可在数据库中查询，可以看到新添加的数据。

1. 删除

在“二”的基础上操作

1. 更改HeroListServlet.java,使表格多一列delete。



1. 新增HeroDeleteServlet.java.

**package** servlet;

**import** dao.HeroDAO;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class HeroDeleteServlet

\*/

@WebServlet("/HeroDeleteServlet")

**public** **class** HeroDeleteServlet **extends** HttpServlet {

**protected** **void** service(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

**int** id = Integer.*parseInt*(request.getParameter("id"));

**new** HeroDAO().delete(id);

response.sendRedirect("/HelloMaven/HeroListServlet");

}

}

1. 编辑

在“三”的基础上操作。（编辑后需要第“五”步的HeroUpdataServlet.java的程序将其更新）

1. 更改HeroListServlet.java,使表格多一列edit。



1. 新增HeroEditServlet.java。（注意表单中action的URL）

package servlet;

import bean.Hero;

import dao.HeroDAO;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class HeroEditServlet

\*/

@WebServlet("/HeroEditServlet")

public class HeroEditServlet extends HttpServlet {

protected void service(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

int id = Integer.parseInt(request.getParameter("id"));

Hero hero = new HeroDAO().get(id);

StringBuffer format = new StringBuffer();

response.setContentType("text/html; charset=UTF-8");

format.append("<!DOCTYPE html>");

format.append("<form action='/HelloMaven/HeroUpdataServlet' method='post'>");

format.append("名字 ： <input type='text' name='name' value='%s' > <br>");

format.append("血量 ： <input type='text' name='hp' value='%f' > <br>");

format.append("伤害： <input type='text' name='damage' value='%d' > <br>");

format.append("<input type='hidden' name='id' value='%d'>");

format.append("<input type='submit' value='更新'>");

format.append("</form>");

String html = String.format(format.toString(), hero.getName(), hero.getHp(), hero.getDamage(), hero.getId());

response.getWriter().write(html);

}

}

1. 更新。（有缺陷，就是如果之前有删除，那么新添加的信息的序号会出现断号问题，即序号不衔接！！）

帮助“四”完成更新。只需添加一个HeroUpdataServlet.java程序。（！！注意response重定位的URL）

package servlet;

import bean.Hero;

import dao.HeroDAO;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class HeroUpdataServlet

\*/

@WebServlet("/HeroUpdataServlet")

public class HeroUpdataServlet extends HttpServlet {

protected void service(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

Hero hero = new Hero();

hero.setId(Integer.parseInt(request.getParameter("id")));

hero.setName(request.getParameter("name"));

hero.setHp(Float.parseFloat(request.getParameter("hp")));

hero.setDamage(Integer.parseInt(request.getParameter("damage")));

new HeroDAO().update(hero);

response.sendRedirect("/HelloMaven/HeroListServlet");

}

}

使用Servlet进行CRUD开发的弊端：

在Servlet编写html代码很痛苦，效率不高，可读性差，难以维护。  
  
最好可以在html文件里面写html代码，同时又能在里面调用java的变量，那么这样就需要学习[JSP](http://how2j.cn/k/jsp/jsp-tutorials/530.html)了。（jsp，可以在HTML中写java代码）

程序代码中：try……catch的用法。

首先要清楚，如果没有try的话，出现异常会导致程序崩溃。

而try则可以保证程序的正常运行下去，比如说：

try{

int i = 1/0;

}catch(Exception e){

........

}

try { name=dis.readUTF(); age=dis.readInt(); maths=dis.readFloat(); english=dis.readFloat(); physical=dis.readFloat(); chinese=dis.readFloat(); } catch(Exception e){  
//直接输出异常，帮助分析  
e.printStackTrace();  
//如果上面是语句发生异常，则可以自己处理  
name="" //default value  
age =0 ;  
....  
//或往外抛出，自己处理异常  
throw e;  
}

try catch 是捕捉try部分的异常，当你没有trycatch的时候，如果出现异常则程序报错，加上trycatch，出现异常程序正常运行，只是把错误信息存储到Exception里，所以catch是用来提取异常信息的，你可以在Catch部分加上一句System.out.println(e.ToString());，如果出现异常可以把异常打印出来。

## 问题：servlet重定向到jsp后，css样式和图片都没了，解决办法

servlet重定向到jsp后，css样式和图片都没了，原因是在servlet中转发时css和图片的路径变成相对于这个servlet的相对路径而非相对于web项目的路径了。

解决办法：导入css样式和图片时把css写成动态绝对路径，

<http://blog.csdn.net/sun9528/article/details/72935178>



## 问题：使用jsp遍历数据库中表的某列数据。

方法：<https://zhidao.baidu.com/question/378330059.html>



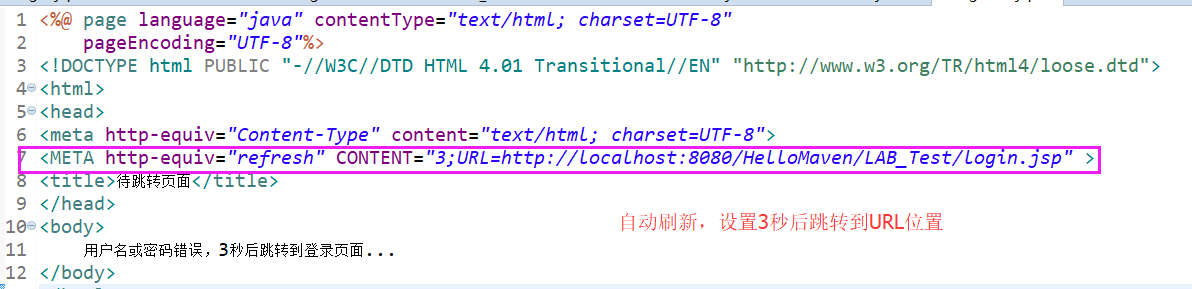


## 问题：访问数据库检查用户身份是否合法，不合法则。。。合法则。。。

可以使用一个跳转页面，在此页面输出错误信息并跳转到登录界面。



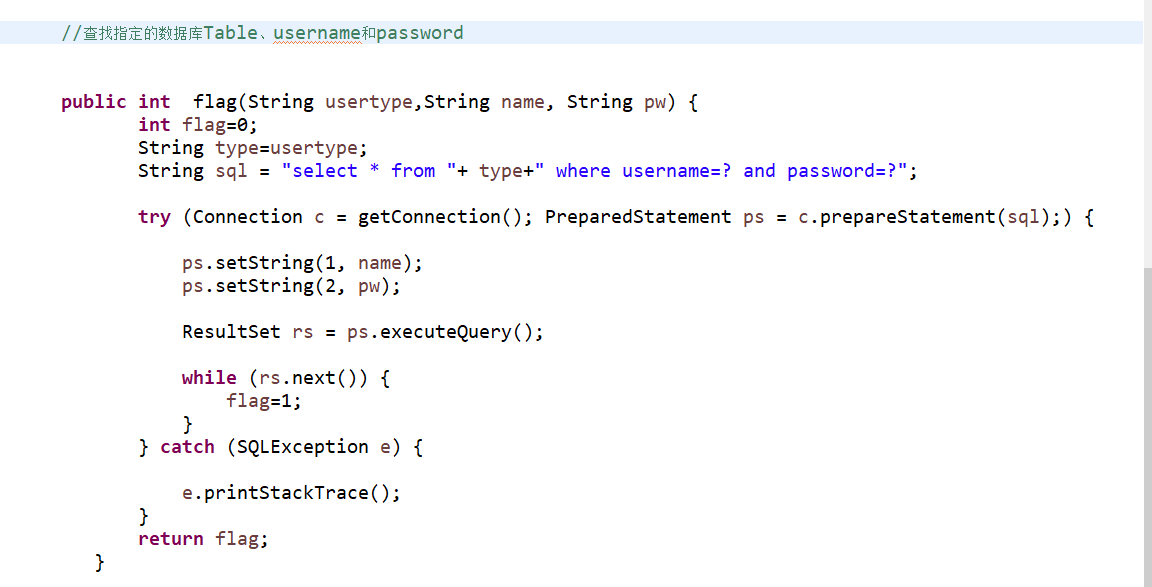
跳转页面如下：



通过读取用户类型，将其值传送到DAO中，在Servlet中访问不同的数据库表

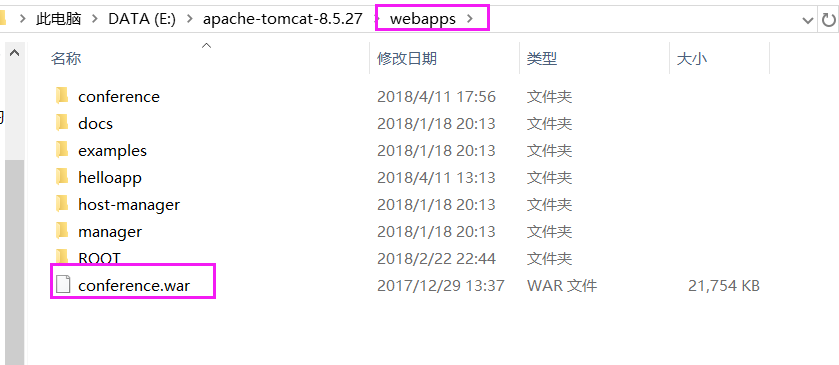


**DAO中的内容，如下图**

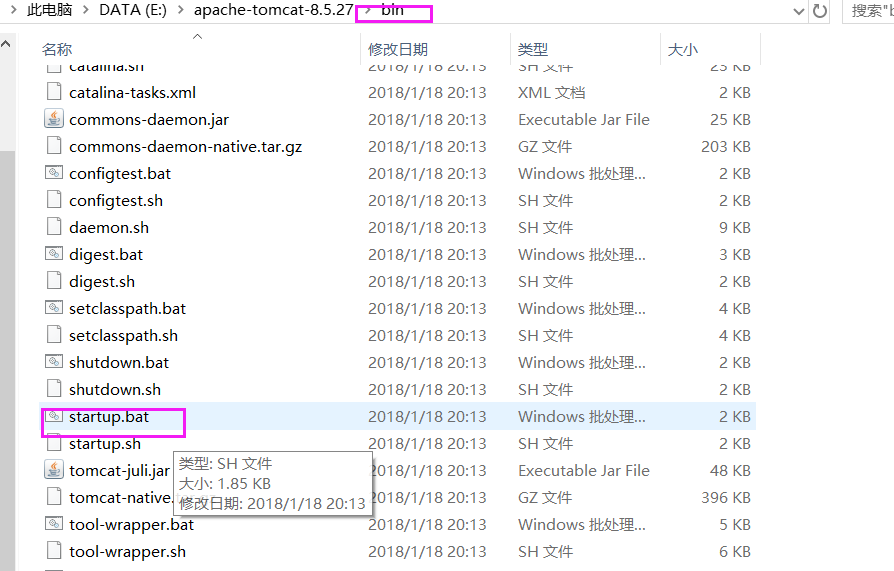


## 问题：如何把war包部署到Tomcat服务器？

1. 导出war包
2. 找到Tomcat安装目录下的webapps，将war包放在此目录



1. 然后在Tomcat的bin目录下找到startup.bat.启动Tomcat服务器，



1. 在浏览器中输入<http://localhost:8080/conference>即可



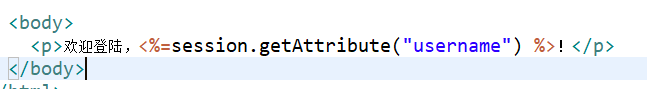
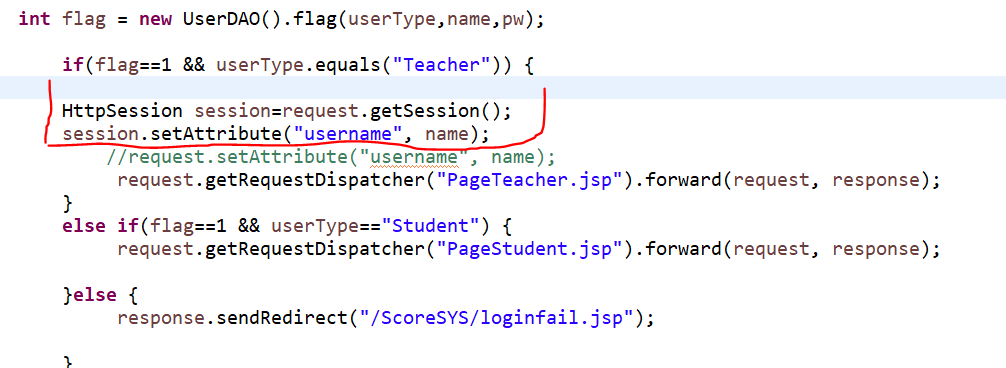
## 错误代码：

## Could not initialize class dbc.JdbcUtil

**原因很大可能是数据库的****没添加进去，或者是连接数据库的代码写错了**

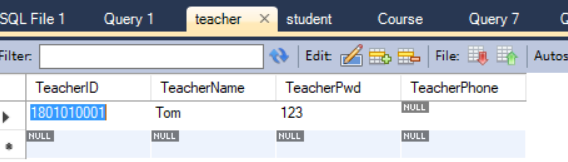
**注意：jsp页面传值不能超过两个页面，如果需要在多个页面中使用某个值，有两种方法。**

1. **在一个纯java代码的jsp页面做中转处理**
2. **通过使用session，session的使用方法**

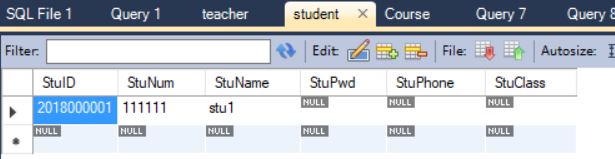


## 问题：MySQL workbench外键的使用

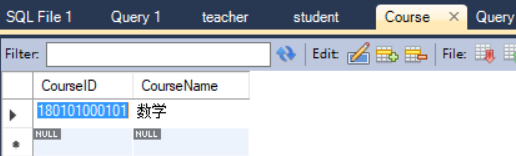
教师表



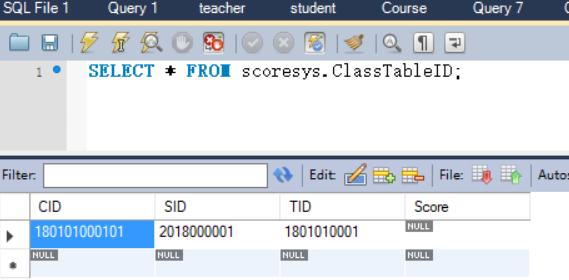
学生表



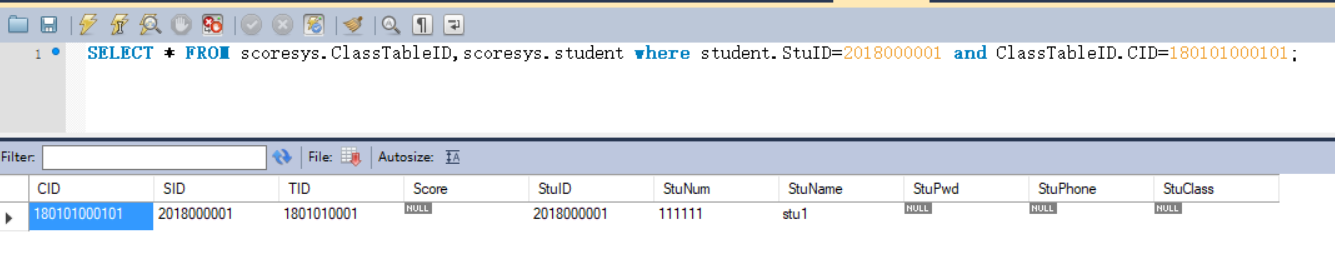
课程表



选课表



通过外键查询信息

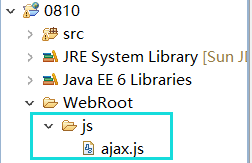


## 问题：MySQL workbench的数据库表中使用外键，容易造成数据插入失败，其原因是因为表间的约束关系。

如约束表中不存在的约束键数据，将导致数据插入失败。解决方法为，在外部使用SQL语句前，现在数据库中对此SQL语句进行测试。还有一个需要注意的点是，主键不能重复。

## 问题：ajax的使用—封装创建ajax到js，method方法为POST

**Ajax.js封装目录：**



**Ajax.js代码（ajax创建对象函数和发动请求函数）**

//声明XMLHttpRequest对象

**var** xmlHttpRequest=**null**;

//创建XMLHttpRequest对象实例的方法

**function** createXHR(){

**if** (window.XMLHttpRequest)

{

// IE7+, Firefox, Chrome, Opera, Safari 浏览器执行代码

xmlHttpRequest=**new** XMLHttpRequest();

}

**else**

{

// IE6, IE5 浏览器执行代码

xmlHttpRequest=**new** ActiveXObject("Microsoft.XMLHTTP");

}

**if**(xmlHttpRequest == **null**) {

alert("不能创建Ajax对象！");

}

}

//发送客户端的请求，该方法有4个参数，其中method取值为Post或get

**function** sendRequest(url,params,method,handler){

createXHR();

**if**(!xmlHttpRequest) **return** **false**;

xmlHttpRequest.onreadystatechange = handler;

**if**(method == "GET"){

xmlHttpRequest.open(method,url+ '?' + params,**true**);

xmlHttpRequest.send(**null**);

}

**if**(method == "POST"){

xmlHttpRequest.open(method,url,**true**);

xmlHttpRequest.setRequestHeader("Content-type","application/x-www-form-urlencoded");

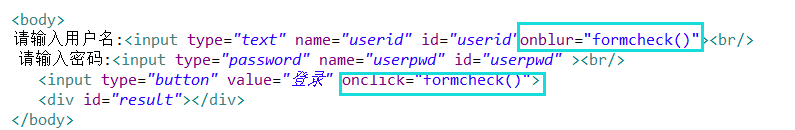
xmlHttpRequest.send(params);

}

}

在页面中如何使用Ajax。

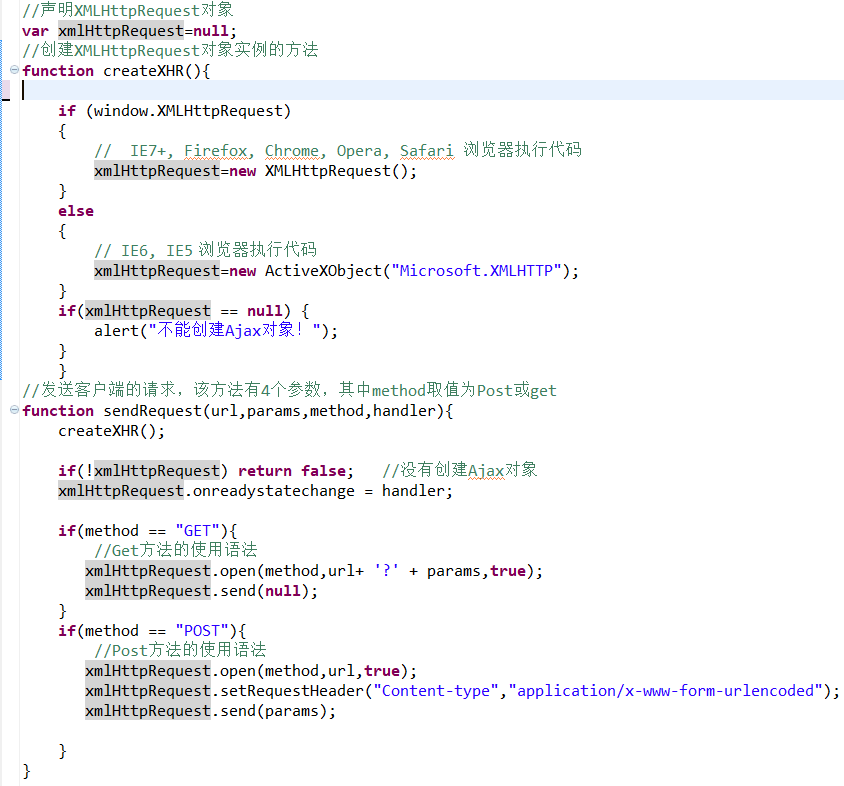
1. 页面内容



1. 使用Ajax



对比HTML页面与封装的ajax间是如何传递参数与调用的。

**MySQL基本语句的使用**

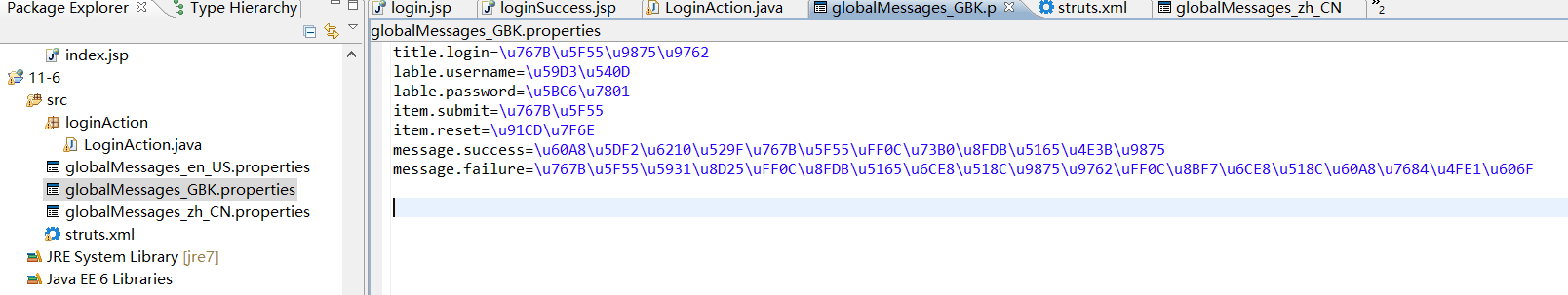
**插入：insert into classtable (TeacherID,StuID,SourceID) values(1514001,151002,151400101);**

**查询多表数据：select s.StuID,s.StuClass,s.StuName ,ct.Score from student s,classtable ct where s.StuID=ct.StuID and ct.TeacherID='1514001' and ct.SourceID='151400103' order by s.StuID asc ;**

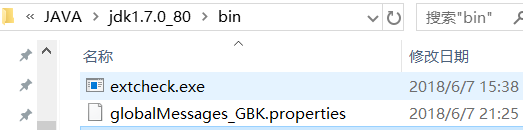
**更新数据：update classtable set Score= 88 where TeacherID=1514001 and StuID=151001 and SourceID=151400101;**

## 问题：将中文资源文件中的中文字符转换为Unicode编码

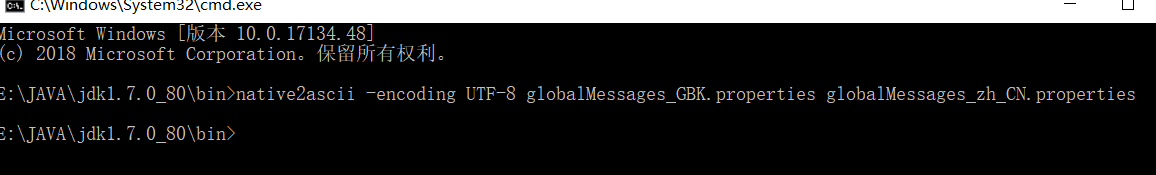
1. 编写源文件

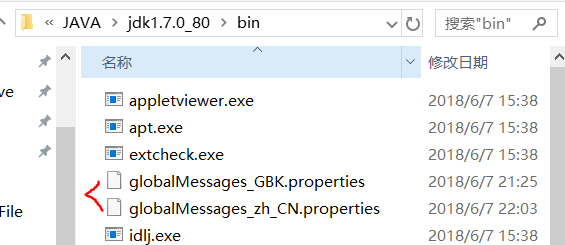


1. 将源文件复制到项目环境所以jdk的bin目录下



1. 在bin目录下进入cmd
2. 编译

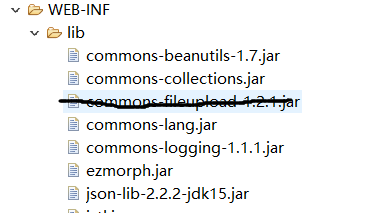




## 问题：异步加载Ajax与json数据实现登录

1. 在工程中通过build path加载使用json—>

所需要的.jar文件，并且将这些.jar文件添加到工程的lib目录下。所需文件（6个）如下：



登录表单：

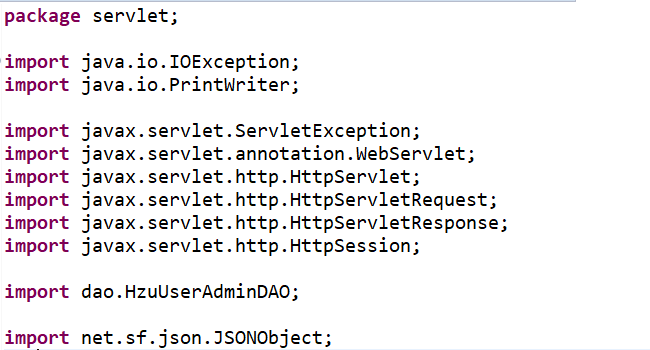


jQuery：



登录检查Servlet：

1. 引入包



1. 正文



## 问题：href="#"与href="javascript:void(0)"的区别

**#** 包含了一个位置信息，默认的锚是**#top** 也就是网页的上端。

而javascript:void(0), 仅仅表示一个死链接。

在页面很长的时候会使用 **#** 来定位页面的具体位置，格式为：**# + id**。

如果你要定义一个死链接请使用 javascript:void(0) 。

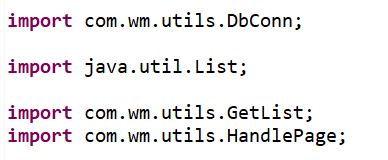
使用方法：

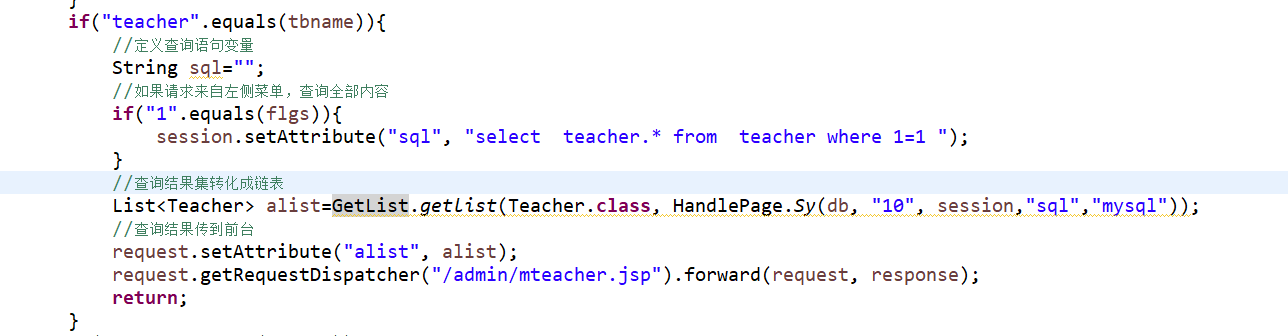
# javascript:void(0) 含义

我们经常会使用到 javascript:void(0) 这样的代码，那么在 JavaScript 中 javascript:void(0) 代表的是什么意思呢？

javascript:void(0) 中最关键的是 void 关键字， void 是 JavaScript 中非常重要的关键字，该操作符指定要计算一个表达式但是不返回值。

# 问题：import com.wm.utils.GetList;与 import com.wm.utils.HandlePage;的使用方法。





**List<Teacher> alist=GetList.*getlist*(Teacher.class, HandlePage.*Sy*(db, "10", session,"sql","mysql"));**

**分析？？Teacher.class为类；HandlePage.*Sy（参数1，参数2，参数3，参数4，参数5）***

***参数1,db 连接数据库***

***参数2，10 执行数据库语句为10类型？？***

***参数3，session 参数4为session类型***

***参数4，sql 要执行的具体SQL语句内容***

***参数5，mysql 数据库类型？？***

## 问题：报表的简单使用Chart.js

**下载js**

<https://cdnjs.com/libraries/Chart.js>

**Chart.js教程**

<http://www.chartjs.org/docs/latest/getting-started/>

**相关连接**

<https://www.cnblogs.com/xiyangbaixue/p/4034437.html>

**参考教程**

<https://blog.csdn.net/kiss_mark/article/details/47005651>

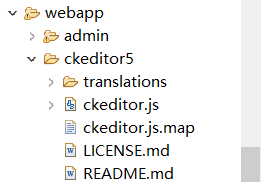
## 问题： ichart教程

<http://www.bubuko.com/infodetail-759634.html>

## 问题： CKeditor5的使用

1. **在eclipse中使用**

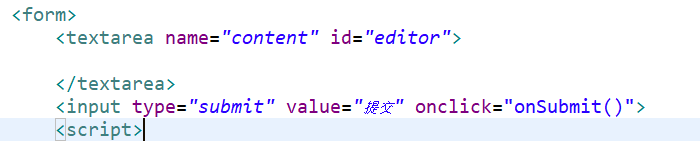
将压缩包解压后复制到根目录



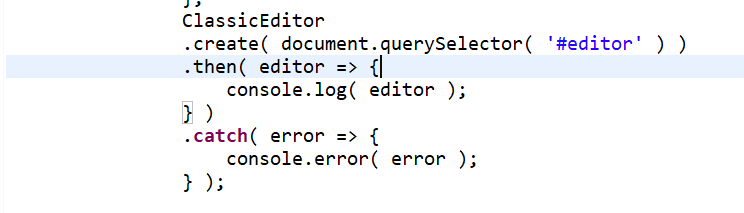
在页面中引用



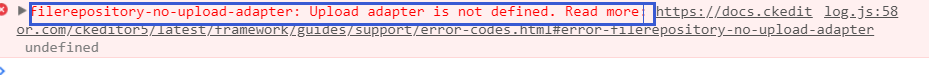
在页面中创建表单和textarea



用编辑器替换掉textarea



此时可以在页面显示编辑器，但上传图片及预览功能未实现，浏览器报错

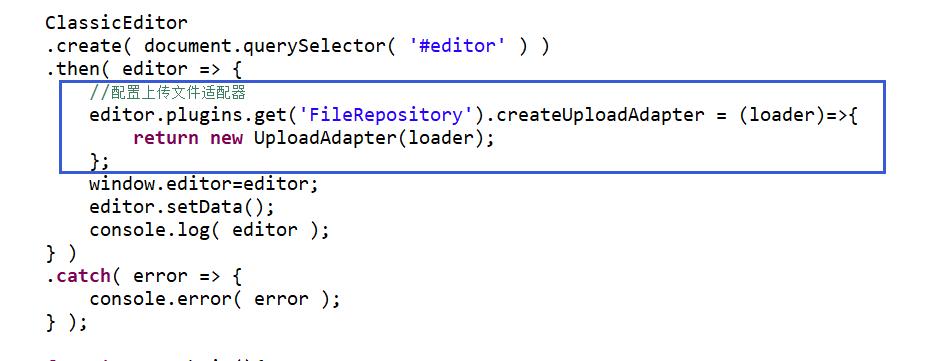


点击链接<https://docs.ckedit>可查看文档，寻找解决方法

1. **上传图片**
2. **加载适配器**



1. **为页面中的编辑器配置适配器**



1. **获取编辑器中的值**

参考资料<http://www.leftso.com/blog/457.html>



**获取上传图片的路径并将图片存入硬盘指定目录（参考Servlet中的文件上传，原理一样）**

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

// 上传文件存储目录

**private** **static** **final** String ***UPLOAD\_DIRECTORY*** = "uploadCommunityImage";

// 上传配置

**private** **static** **final** **int** ***MEMORY\_THRESHOLD*** = 1024 \* 1024 \* 3; // 3MB

**private** **static** **final** **int** ***MAX\_FILE\_SIZE*** = 1024 \* 1024 \* 40; // 40MB

**private** **static** **final** **int** ***MAX\_REQUEST\_SIZE*** = 1024 \* 1024 \* 50; // 50MB

/\*\*

\* 上传数据及保存文件

\*/

**protected** **void** doPost(HttpServletRequest request,

HttpServletResponse response) **throws** ServletException, IOException {

response.setCharacterEncoding("utf-8");

response.setContentType("text/html");

//创建json对象

JSONObject json=**new** JSONObject();

HttpSession session=request.getSession();

// 检测是否为多媒体上传

**if** (!ServletFileUpload.*isMultipartContent*(request)) {

// 如果不是则停止

PrintWriter writer = response.getWriter();

writer.println("Error: 表单必须包含 enctype=multipart/form-data");

writer.flush();

**return**;

}

// 配置上传参数

DiskFileItemFactory factory = **new** DiskFileItemFactory();

// 设置内存临界值 - 超过后将产生临时文件并存储于临时目录中

factory.setSizeThreshold(***MEMORY\_THRESHOLD***);

// 设置临时存储目录

factory.setRepository(**new** File(System.*getProperty*("java.io.tmpdir")));

ServletFileUpload upload = **new** ServletFileUpload(factory);

// 设置最大文件上传值

upload.setFileSizeMax(***MAX\_FILE\_SIZE***);

// 设置最大请求值 (包含文件和表单数据)

upload.setSizeMax(***MAX\_REQUEST\_SIZE***);

// 中文处理

upload.setHeaderEncoding("UTF-8");

// 构造临时路径来存储上传的文件

// 这个路径相对当前应用的目录

String uploadPath = request.getServletContext().getRealPath("./") + File.***separator*** + ***UPLOAD\_DIRECTORY***;

System.***out***.print("uploadPath="+uploadPath);

// 如果目录不存在则创建

File uploadDir = **new** File(uploadPath);

**if** (!uploadDir.exists()) {

uploadDir.mkdir();

}

**try** {

// 解析请求的内容提取文件数据

//@SuppressWarnings("unchecked")

List<FileItem> formItems = upload.parseRequest(request);

**if** (formItems != **null** && formItems.size() > 0) {

// 迭代表单数据

**for** (FileItem item : formItems) {

// 处理不在表单中的字段

**if** (!item.isFormField()) {

String fileName = **new** File(item.getName()).getName();

String filePath = uploadPath + File.***separator*** + fileName;

File storeFile = **new** File(filePath);

// 在控制台输出文件的上传路径

System.***out***.println("这是CKeditor的图片上传，filename="+fileName);

System.***out***.println(filePath);

String showpath="/images/"+fileName;

// 保存文件到硬盘

item.write(storeFile);

request.setAttribute("message",

"文件上传成功!");

session.setAttribute("filepath", fileName);

json.put("res", "1");

json.put("url", showpath);

}

}

}

} **catch** (Exception ex) {

request.setAttribute("message",

"错误信息: " + ex.getMessage());

}

String text=request.getParameter("htmlStr");

System.***out***.print("text="+text);

PrintWriter out=response.getWriter();

out.print(json.toString());

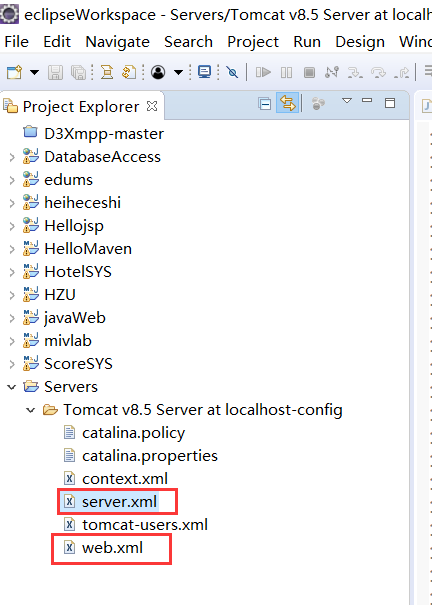
out.flush();

out.close();

## 问题： 如何配置Tomcat虚拟路径（浏览器无法直接访问盘符上的目录及文件）

1. **修改eclipse中的Tomcat中的配置文件web.xml与server.xml**

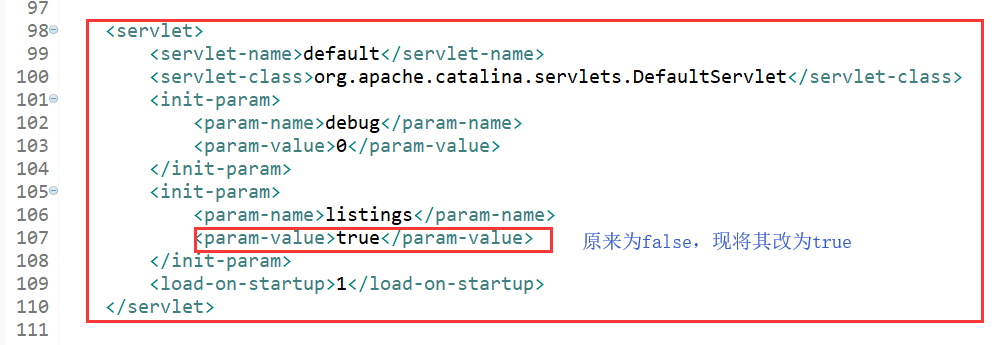
**结构目录**



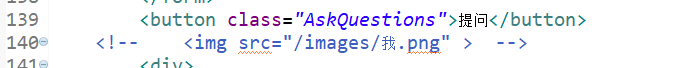
**（1）在server.xml文件中找到<Host></Host>在此标签内添加如下语句，其中path为虚拟路径的名称，docBase为盘符目录的路径**



**（2）在web.xml中修改如下**



**（3）在HTML文件中，可以这样使用虚拟目录中的文件（注释部分）**



## 问题： 使用GitHubAPI的相关使用

1. 网站接入github开放平台openid

<https://m.baidu.com/from=1000953c/bd_page_type=1/ssid=0/uid=0/pu=usm%401%2Csz%40320_1002%2Cta%40iphone_2_8.0_2_7.3/baiduid=65A9DE89DC317B912F250110C1823A83/w=0_10_/t=iphone/l=3/tc?ref=www_iphone&lid=8859427263115196377&order=1&fm=alop&tj=www_normal_1_0_10_title&vit=osres&m=8&srd=1&cltj=cloud_title&asres=1&title=%E7%BD%91%E7%AB%99%E6%8E%A5%E5%85%A5github%E5%BC%80%E6%94%BE%E5%B9%B3%E5%8F%B0openid&dict=32&wd=&eqid=7af3022813647800100000005b4f5df5&w_qd=IlPT2AEptyoA_yk5sAga6hGvBDxVjGUnw63WLDpQBzktNumWixm&tcplug=1&sec=31335&di=970eeba5819972f2&bdenc=1&tch=124.3142.54.179.2.3230&nsrc=IlPT2AEptyoA_yixCFOxXnANedT62v3IEQGG_yVE0zeniE4te4viZQRAWyOqAp7UGknwtyPQpxsDxn_c0mAl9RN0t_RotX9hbSybs_WbrsCLHqILeg2nOw3SH8Rxo0ib8K9vduJ5Kw--KDS&clk_info=%7B%22srcid%22%3A1599%2C%22tplname%22%3A%22www_normal%22%2C%22t%22%3A1531928101635%2C%22xpath%22%3A%22div-a-h3-em%22%7D&sfOpen=1>

## 问题： 使用Servlet发送get或Post请求

**public** **static** String sendPost(String url, String param) {

PrintWriter out = **null**;

BufferedReader in = **null**;

// String result = "";

**try** {

URL realUrl = **new** URL(url);

// 打开和URL之间的连接

URLConnection conn = realUrl.openConnection();

// 设置通用的请求属性

conn.setRequestProperty("accept", "\*/\*");

conn.setRequestProperty("connection", "Keep-Alive");

conn.setRequestProperty("user-agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;SV1)");

conn.setRequestProperty("Accept"," application/json");

//System.out.print(realUrl);

// 发送POST请求必须设置如下两行

conn.setDoOutput(**true**);

conn.setDoInput(**true**);

// 获取URLConnection对象对应的输出流

out = **new** PrintWriter(conn.getOutputStream());

// 发送请求参数

out.print(param);

// flush输出流的缓冲

out.flush();

// 定义BufferedReader输入流来读取URL的响应

in = **new** BufferedReader(**new** InputStreamReader(conn.getInputStream()));

String line;

**while** ((line = in.readLine()) != **null**) {

*result* += line;

}

//System.out.print("//nresult=="+result);

} **catch** (Exception e) {

System.***out***.println("[POST请求]向地址：" + url + " 发送数据：" + param + " 发生错误!");

} **finally** {// 使用finally块来关闭输出流、输入流

**try** {

**if** (out != **null**) {

out.close();

}

**if** (in != **null**) {

in.close();

}

} **catch** (IOException ex) {

System.***out***.println("关闭流异常");

}

}

**return** *result*;

}

//发送GET请求

**public** **static** String sendGet(String url) {

String result = "";

BufferedReader in = **null**;

**try** {

String urlName = url;

URL realUrl = **new** URL(urlName);

URLConnection conn = realUrl.openConnection();// 打开和URL之间的连接

conn.setRequestProperty("accept", "\*/\*");// 设置通用的请求属性

conn.setRequestProperty("connection", "Keep-Alive");

conn.setRequestProperty("user-agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)");

conn.setRequestProperty("Accept"," application/json");

conn.setConnectTimeout(4000);

conn.connect();// 建立实际的连接

in = **new** BufferedReader(**new** InputStreamReader(conn.getInputStream(), "UTF-8"));// 定义BufferedReader输入流来读取URL的响应

String line;

**while** ((line = in.readLine()) != **null**) {

result += line;

}

} **catch** (Exception e) {

System.***out***.println("发送GET请求出现异常！" + e);

} **finally** {// 使用finally块来关闭输入流

**try** {

**if** (in != **null**) {

in.close();

}

} **catch** (IOException ex) {

System.***out***.println("关闭流异常");

}

}

**return** result;

}

## 问题： 将json格式的字符串数据转换为json形式，并读取其中数据

**已知有json格式字符串数据s=null{“eee”:”23”,”eee2”:”45”}**

**操作过程如下：**

String s= *sendPost*("https://github.com/login/oauth/access\_token?","client\_id=3973251206d9e003b6c7&client\_secret=06b38dd03833945f7dc6fb14a115b2a46ac0f089&code="+code+"&scope=user%20public\_repo");

// System.out.println("//n s="+s+"##########");

String jieguo = s.substring(s.indexOf("{"),s.indexOf("}")+1); //获取json格式的字符串

// System.out.println("//n jieguo="+jieguo+"##########");

String[] str=**new** String[]{jieguo};

GithubAccess\_token ck = **new** GithubAccess\_token();

List<GithubAccess\_token> checkLists = **new** ArrayList<>();

**for**(String s2 :str){ //循环

JSONObject jsonObject= JSON.*parseObject*(jieguo);

ck.setAccess\_token((String) jsonObject.get("access\_token"));

ck.setScope((String) jsonObject.get("scope"));

ck.setToken\_type((String) jsonObject.get("token\_type"));

checkLists.add(ck);

}

// System.out.print("\naccess\_token="+checkLists.get(0).getAccess\_token());

String access\_token=checkLists.get(0).getAccess\_token();

String scope=checkLists.get(0).getScope();

String token\_type=checkLists.get(0).getToken\_type();

**此问题与上一问题的完整代码：（GitHub授权与获取用户信息）**

**package github;**

**import java.io.BufferedReader;**

**import java.io.IOException;**

**import java.io.InputStreamReader;**

**import java.io.PrintWriter;**

**import java.net.URL;**

**import java.net.URLConnection;**

**import java.util.ArrayList;**

**import java.util.HashMap;**

**import java.util.List;**

**import java.util.Map;**

**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;**

**import bean.GithubAccess\_token;**

**import bean.GithubUserInfo;**

**import bean.Teacher;**

**import bean.navList;**

**import utils.Dbhelper;**

**import com.alibaba.fastjson.JSON;**

**import com.alibaba.fastjson.JSONObject;**

**import com.wm.utils.DbConn;**

**import com.wm.utils.GetList;**

**/\*\***

**\* Servlet implementation class callback**

**\*/**

**@WebServlet("/callback")**

**public class callback extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**String code,state;**

**static String result;**

**/\*\***

**\* @see HttpServlet#HttpServlet()**

**\*/**

**public callback() {**

**super();**

**// TODO Auto-generated constructor stub**

**}**

**/\*\***

**\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)**

**\*/**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**//获取session对象**

**HttpSession session=request.getSession();**

**code=request.getParameter("code");**

**state=request.getParameter("state");**

**//System.out.print(code+"\*\*\*\*\*\*"+state);**

**String s= sendPost("https://github.com/login/oauth/access\_token?","client\_id=3973251206d9e003b6c7&client\_secret=06b38dd03833945f7dc6fb14a115b2a46ac0f089&code="+code+"&scope=user%20public\_repo");**

**// System.out.println("//n s="+s+"##########");**

**String jieguo = s.substring(s.indexOf("{"),s.indexOf("}")+1); //获取json格式的字符串**

**// System.out.println("//n jieguo="+jieguo+"##########");**

**String[] str=new String[]{jieguo};**

**GithubAccess\_token ck = new GithubAccess\_token();**

**List<GithubAccess\_token> checkLists = new ArrayList<>();**

**for(String s2 :str){ //循环**

**JSONObject jsonObject= JSON.parseObject(jieguo);**

**ck.setAccess\_token((String) jsonObject.get("access\_token"));**

**ck.setScope((String) jsonObject.get("scope"));**

**ck.setToken\_type((String) jsonObject.get("token\_type"));**

**checkLists.add(ck);**

**}**

**// System.out.print("\naccess\_token="+checkLists.get(0).getAccess\_token());**

**String access\_token=checkLists.get(0).getAccess\_token();**

**String scope=checkLists.get(0).getScope();**

**String token\_type=checkLists.get(0).getToken\_type();**

**// System.out.print("access\_token="+access\_token+"\*\*\*\*\*\*\*\*\*");**

**String getResult=sendGet("https://api.github.com/user?access\_token="+access\_token);**

**//System.out.print("getResult="+getResult);**

**String[] str2=new String[]{getResult};**

**GithubUserInfo ck2 = new GithubUserInfo();**

**List<GithubUserInfo> checkLists2 = new ArrayList<>();**

**for(String s1 :str){ //循环**

**JSONObject jsonObject2= JSON.parseObject(getResult);**

**ck2.setEmail((String) jsonObject2.get("email"));**

**//ck2.setGit\_id((String) jsonObject2.get("id"));**

**ck2.setLogin((String) jsonObject2.get("login"));**

**checkLists2.add(ck2);**

**}**

**//System.out.print("login=="+checkLists2.get(0).getLogin());**

**DbConn db=Dbhelper.getDb();**

**int bls=db.executeUpdate("update teacher set login='"+checkLists2.get(0).getLogin()+"' where id="+request.getSession().getAttribute("userid"));**

**if(bls>0){**

**System.out.print("操作成功");**

**//查询数据库获取想要的信息**

**List<Teacher> alist=GetList.getlist(Teacher.class, db.executeQuery("select \* from teacher where id="+request.getSession().getAttribute("userid")));**

**//创建对象**

**Teacher a=new Teacher();**

**//取查询结果给对象**

**if(alist.size()>0){**

**a=alist.get(0);**

**}**

**session.setAttribute("userinfo", a);**

**String resultrepo=sendPost("https://api.github.com/user/repos?","name=test&description=null");**

**System.out.println("//nNewRopeResult=="+resultrepo+"\*\*\*\*");**

**request.getRequestDispatcher("/teacher/test.jsp").forward(request, response);**

**}else {**

**System.out.print("操作失败");**

**}**

**}**

**/\*\***

**\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)**

**\*/**

**protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**doGet(request, response);**

**}**

**public static String sendPost(String url, String param) {**

**PrintWriter out = null;**

**BufferedReader in = null;**

**// String result = "";**

**try {**

**URL realUrl = new URL(url);**

**// 打开和URL之间的连接**

**URLConnection conn = realUrl.openConnection();**

**// 设置通用的请求属性**

**conn.setRequestProperty("accept", "\*/\*");**

**conn.setRequestProperty("connection", "Keep-Alive");**

**conn.setRequestProperty("user-agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;SV1)");**

**conn.setRequestProperty("Accept"," application/json");**

**//System.out.print(realUrl);**

**// 发送POST请求必须设置如下两行**

**conn.setDoOutput(true);**

**conn.setDoInput(true);**

**// 获取URLConnection对象对应的输出流**

**out = new PrintWriter(conn.getOutputStream());**

**// 发送请求参数**

**out.print(param);**

**// flush输出流的缓冲**

**out.flush();**

**// 定义BufferedReader输入流来读取URL的响应**

**in = new BufferedReader(new InputStreamReader(conn.getInputStream()));**

**String line;**

**while ((line = in.readLine()) != null) {**

**result += line;**

**}**

**//System.out.print("//nresult=="+result);**

**} catch (Exception e) {**

**System.out.println("[POST请求]向地址：" + url + " 发送数据：" + param + " 发生错误!");**

**} finally {// 使用finally块来关闭输出流、输入流**

**try {**

**if (out != null) {**

**out.close();**

**}**

**if (in != null) {**

**in.close();**

**}**

**} catch (IOException ex) {**

**System.out.println("关闭流异常");**

**}**

**}**

**return result;**

**}**

**//发送GET请求**

**public static String sendGet(String url) {**

**String result = "";**

**BufferedReader in = null;**

**try {**

**String urlName = url;**

**URL realUrl = new URL(urlName);**

**URLConnection conn = realUrl.openConnection();// 打开和URL之间的连接**

**conn.setRequestProperty("accept", "\*/\*");// 设置通用的请求属性**

**conn.setRequestProperty("connection", "Keep-Alive");**

**conn.setRequestProperty("user-agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)");**

**conn.setRequestProperty("Accept"," application/json");**

**conn.setConnectTimeout(4000);**

**conn.connect();// 建立实际的连接**

**in = new BufferedReader(new InputStreamReader(conn.getInputStream(), "UTF-8"));// 定义BufferedReader输入流来读取URL的响应**

**String line;**

**while ((line = in.readLine()) != null) {**

**result += line;**

**}**

**} catch (Exception e) {**

**System.out.println("发送GET请求出现异常！" + e);**

**} finally {// 使用finally块来关闭输入流**

**try {**

**if (in != null) {**

**in.close();**

**}**

**} catch (IOException ex) {**

**System.out.println("关闭流异常");**

**}**

**}**

**return result;**

**}**

**}**

## 问题： java编写自动录入数据的程序

## 传入Java对象 自动创建动态表 并录入数据

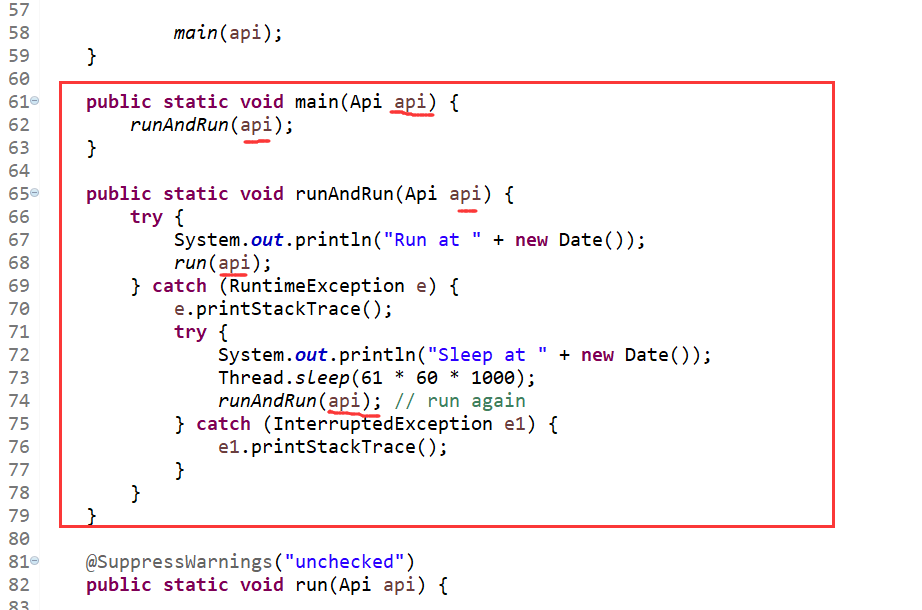
<https://blog.csdn.net/q1512451239/article/details/53144177>

## 问题Eclipse开发遇到："The type org.apache.http.HttpResponse cannot be resolved."问题的解决办法

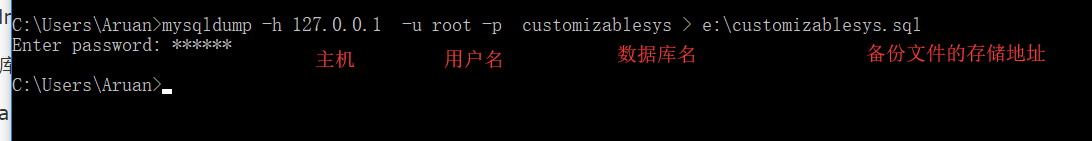
<https://blog.csdn.net/monkey_d_feilong/article/details/51532438>

## 问题：java语言需要保存用户输入的表单数据，且该数据要多次读取使用，应该如何设置数据的存储类型？

实例化对象。通过对象传递参数。

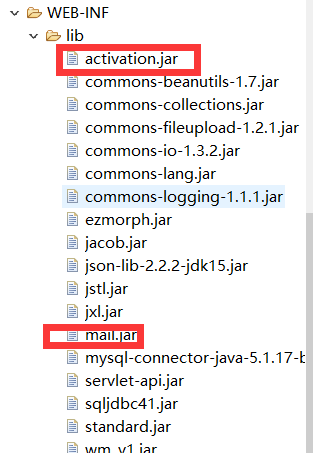
# 问题：将数据库备份成.sql格式



## 部署问题：上传图片后立即显示在编辑器上？抓取数据存储在什么地方？多位老师同时抓取数据会出现什么问题？项目名称问题，需要与二级域名的一样？

## JavaEmail 发送邮件

需要两个.jar包



举例代码：（需注意邮箱的配置）

**package** findPwd;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.Date;

**import** java.util.Properties;

**import** java.util.UUID;

**import** javax.mail.Authenticator;

**import** javax.mail.Message;

**import** javax.mail.PasswordAuthentication;

**import** javax.mail.Session;

**import** javax.mail.Transport;

**import** javax.mail.internet.InternetAddress;

**import** javax.mail.internet.MimeMessage;

**import** javax.mail.internet.MimeMessage.RecipientType;

**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** com.wm.utils.DbConn;

**import** net.sf.json.JSONObject;

**import** utils.Dbhelper;

**import** utils.MD5Util;

/\*\*

\* Servlet implementation class ChangeEmailSvlt

\*/

@WebServlet("/ChangeEmailSvlt")

**public** **class** ChangeEmailSvlt **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

/\*\*

\* **@see** HttpServlet#HttpServlet()

\*/

**public** ChangeEmailSvlt() {

**super**();

// **TODO** Auto-generated constructor stub

}

/\*\*

\* **@see** HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

//设置传输数据编码方式

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

response.setContentType("text/html");

//创建json对象

JSONObject json=**new** JSONObject();

**try** {

String email=request.getParameter("email"); //收件人的email

String utype=request.getParameter("tbname");//获取用户类型对应的表名

String id=request.getParameter("id"); //id

String name=request.getParameter("name"); //Num

System.***out***.println("name="+name);

System.***out***.println("id="+id);

//邮箱链接

String path=request.getContextPath();

String basePath=request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path;

//配置邮箱

Properties prop = **new** Properties();

prop.setProperty("mail.transport.protocol", "smtp");

prop.put("mail.smtp.port","465");

prop.setProperty("mail.smtp.host", "smtp.qq.com");

prop.setProperty("mail.smtp.auth", "true");

prop.setProperty("mail.smtp.ssl.enable","true");

//本地测试用的服务器与端口号

//prop.setProperty("mail.smtp.host", "smtp.qq.com");

//prop.put("mail.smtp.port","25");

//prop.setProperty("mail.debug", "true");

Session session = Session.*getInstance*(prop,

**new** Authenticator(){

@Override

**protected** PasswordAuthentication getPasswordAuthentication() {

//返回发件人账号与密码信息

**return** **new** PasswordAuthentication("1370763849@qq.com","此处填密码");//用户名和口令

}

}

);

//填写信封写信

Message msg = **new** MimeMessage(session);

//设置发邮件的原地址

msg.setFrom(**new** InternetAddress("1370763849@qq.com"));

//设置接收人

msg.setRecipient(RecipientType.***TO***, **new** InternetAddress(email));

//主题

msg.setSubject("找回密码!");

//内容

String content="error,请重新操作";

//判断是什么类型的用户

//1.admin

**if**("admin".equals(utype)) {

String tbname="adminEmail";

content=createLink( id,name,utype, tbname,email, basePath);

}

**if**("students".equals(utype)) {

String tbname="stuEmail";

content=createLink( id,name,utype, tbname,email, basePath);

}

**if**("teacher".equals(utype)) {

String tbname="teaEmail";

content=createLink( id,name,utype, tbname,email, basePath);

}

msg.setContent("<font color=\"red\">"+content+"</font>", "text/html;charset=gbk");

System.***out***.println("邮件开始发送！");

Transport.*send*(msg);

System.***out***.println("邮件发送成功！");

json.put("msg", 1);

}**catch**(Exception e){

e.printStackTrace();

json.put("msg", "发送失败");

}

PrintWriter out=response.getWriter();

out.print(json.toString());

out.flush();

out.close();

}

/\*\*

\* **@see** HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

doGet(request, response);

}

**public** String createLink(String id,String name,String utype,String tbname,String email, String basePath){

//生成密钥

String secretKey=UUID.*randomUUID*().toString();

//设置过期时间

Date outDate = **new** Date(System.*currentTimeMillis*() + 30 \* 60 \* 1000);// 30分钟后过期

System.***out***.println("System.currentTimeMillis()="+System.*currentTimeMillis*());

**long** date = outDate.getTime() / 1000 \* 1000;// 忽略毫秒数 mySql 取出时间是忽略毫秒数的

System.***out***.println("date="+date);

String date2=String.*valueOf*(date);

System.***out***.println("date2="+date2);

//此处应该更新StuinfoTest表中的过期时间、密钥信息

//创建数据库操作对象

DbConn db=Dbhelper.*getDb*();

**int** bls=db.executeUpdate("update "+utype+" set outDate='"+date2+"',validataCode='"+secretKey+"' where id='"+id+"'");

System.***out***.print(bls);

**if**(bls==1){

System.***out***.println("用户信息修改成功");

//将用户名、过期时间、密钥生成链接密钥

String key =name + "$" + date2 + "$" + secretKey;

System.***out***.println("getUsernumber="+name);

System.***out***.println("date2="+date2);

System.***out***.println("secretKey="+secretKey);

String digitalSignature = MD5Util.*MD5*(key);// 数字签名

String resetPassHref = basePath + "/ChangeEmailSvlt2?sid="+ digitalSignature+"&tb="+tbname+"&em="+email +"&id="+id;

String emailContent = "请勿回复本邮件.点击下面的链接,重设邮箱,本邮件超过30分钟,链接将会失效，需要重新申请设置邮箱." + resetPassHref;

**return** emailContent;

}**else**{

**return** "error";

}

}

}